

FORCE.COM

FUNDAMENTALS

An Introduction to Custom Application Development in the Cloud

Force.com Platform Fundamentals
An Introduction to Custom Application Development in the Cloud

Version 6
August 2011

By Phil Choi
Chris McGuire
Caroline Roth

With contributions by
Dave Carroll
Nick Tran
Andrea Leszek
Ari Langer

Force.com Platform Fundamentals

© Copyright 2000-2011 salesforce.com, inc. All rights reserved. Salesforce.com is a registered trademark of salesforce.com, inc., as are other names and marks. Other marks appearing herein may be trademarks of their respective owners.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

ISBN: 978-0-9789639-3-4

With special thanks to Grant Anderson, Steve Anderson, Dean Atchison, Sheri Bernard, Mysti Berry, Eric Bezar, James Bisso, Gina Blednyh, Michelle Chapman-Thurber, Leah Cutter, Steve Fisher, Carol Franger, Katia Hage, Ron Hess, Vincent Hurtel, Paul Kopacki, Sarah Marovich, Jon Mountjoy, Lauren Pederson, Vahn Phan, Kimberley Rathbun, Elizabeth Rice, Mary Scotton, Andrew Smith, Tom Tobin, Garen Torikian, Adam Torman, Andrew Waite, Joanne Ward, Sarah Whitlock, Clive Wong, and the Training & Certification team.

Table of Contents

Preface	1
Welcome to the Cloud!.....	1
Platforms for Cloud Computing.....	2
About This Book.....	2
Intended Audience.....	3
Chapter Contents.....	3
Choosing Your Development Environment.....	5
Sending Feedback.....	5
About Developer Force.....	5
Salesforce.com Training & Certification.....	6
Chapter 2: Introducing the Force.com Platform	7
The Basics of an App's User Interface.....	8
Tabs.....	8
Forms.....	8
Links.....	8
The Benefits of a Force.com Platform App.....	9
Data-Centric Apps.....	9
Collaborative Apps.....	10
The Technologies Behind a Force.com Platform App.....	10
A Multitenant Architecture.....	11
A Metadata-Driven Development Model.....	12
The Web Services API.....	13
Apex.....	14
Visualforce.....	14
Salesforce Mobile.....	14
Sites.....	15
The AppExchange Directory.....	16
Chapter 3: About the Sample Recruiting App	17
About Universal Containers.....	18
Considerations for the Recruiting App.....	18

Building the App: Our Design.....	19
Custom Objects.....	19
Security and Sharing Rules.....	20
Workflow and Approval Processes.....	20
Custom Reports and Dashboards.....	22
Visualforce.....	22
Chapter 4: Reviewing Database Concepts.....	23
What's a Database?.....	24
What's in a Database?.....	25
What's a Relational Database?.....	26
Summary of Database Concepts.....	28
Chapter 5: Building a Simple App.....	29
Becoming Familiar with the Setup Area.....	30
Introducing Apps.....	31
Try It Out: Defining an App.....	32
Look at What We've Done.....	34
Introducing Objects.....	35
The Position Custom Object.....	35
Try It Out: Defining the Position Custom Object.....	35
Introducing Tabs.....	39
Try It Out: Defining the Positions Tab.....	39
Look at What We've Done.....	41
Becoming Familiar with Setup Detail Pages and Related Lists.....	43
Introducing Fields.....	45
Try It Out: Adding Text Fields.....	46
Try It Out: Adding Currency Fields.....	49
Try It Out: Adding Checkbox Fields.....	49
Try It Out: Adding Date Fields.....	50
Look at What We've Done.....	50
Chapter 6: Enhancing the Simple App with Advanced Fields, Data Validation, and Page Layouts.....	53
Adding Advanced Fields.....	54
Introducing Picklists.....	54
Try It Out: Adding Picklists.....	54

Introducing Field Dependencies.....	57
Try It Out: Creating a Dependent Picklist.....	58
Look at What We've Done.....	61
Introducing Custom Formula Fields.....	62
Calculating How Long a Position Has Been Open.....	63
Try It Out: Defining a “Days Open” Custom Formula Field.....	64
Try It Out: Giving Fields Dynamic Default Values.....	68
Look at What We've Done.....	69
Introducing Validation Rules.....	69
Try It Out: Defining a Validation Rule for Min and Max Pay.....	70
Try It Out: Defining a Validation Rule for Close Date.....	72
Look at What We've Done.....	74
Introducing Page Layouts.....	76
Becoming Familiar with the Page Layout Editor.....	76
Try It Out: Grouping Fields into a New Section.....	78
Try It Out: Adding Spaces Between Fields.....	80
Try It Out: Editing Field Properties.....	81
Look at What We've Done.....	82
Chapter 7: Expanding the Simple App Using Relationships.....	85
Introducing Relationships.....	86
Introducing Relationship Custom Fields.....	87
Try It Out: Relating Hiring Managers to Positions.....	88
Look at What We've Done.....	88
Adding Candidates to the Mix.....	90
Try It Out: Creating the Candidate Object.....	90
Try It Out: Adding Fields to the Candidate Object.....	91
Try It Out: Modifying the Candidate Page Layout Properties.....	93
Look at What We've Done.....	94
Bringing Candidates and Positions Together with Job Applications.....	95
Try It Out: Creating the Job Application Object.....	96
Try It Out: Adding Fields to the Job Application Object.....	97
Look at What We've Done.....	98
Introducing Search Layouts.....	100
Try It Out: Adding Fields to the Candidate Lookup Dialog.....	102
Try It Out: Updating Additional Search Layouts.....	103
Managing Review Assessments.....	104

Try It Out: Creating the Review Object.....	105
Try It Out: Adding Fields to the Review Object.....	106
Introducing Roll-Up Summary Fields.....	112
Try It Out: Creating Roll-Up Summary Fields.....	112
Try It Out: Customizing the Review Object's Page and Search Layouts.....	114
Look at What We've Done.....	116
Creating a Many-to-Many Relationship.....	117
Try It Out: Creating the Employment Website Object.....	118
Try It Out: Adding the URL Field to the Employment Website Object.....	119
Try It Out: Creating the Job Posting Object.....	120
Try It Out: Adding Fields to the Job Posting Object.....	121
Customizing Related Lists in a Many-to-Many Relationship.....	123
Try It Out: Customizing the Positions and Employment Websites Related Lists.....	123
Look at What We've Done.....	124
Putting it All Together.....	125
Try It Out: Downloading Sample Data.....	127
Try It Out: Using the Import Wizard.....	128
Chapter 8: Securing and Sharing Data.....	131
Controlling Access to Data in Our App.....	132
Required Permissions for the Recruiter.....	132
Required Permissions for the Hiring Manager.....	133
Required Permissions for the Interviewer.....	135
Required Permissions for the Standard Employee.....	136
So Where Are We Now?.....	137
Data Access Concepts.....	138
Controlling Access to Objects.....	140
Introducing Profiles.....	140
Standard Profiles.....	140
Custom Profiles in Our Recruiting App.....	141
Try It Out: Creating the Recruiter Profile.....	142
Try It Out: Creating More Profiles.....	146
Controlling Access to Fields.....	149
Introducing Field-Level Security.....	149

Field-Level Security in Our Recruiting App.....	150
Accessing Field-Level Security Settings.....	151
Try It Out: Restricting Access to a Position's Minimum and Maximum Salary Fields.....	151
Try It Out: Restricting Access to a Candidate's Social Security Number.....	154
Try It Out: Setting Lookup Fields to Read-Only on a Job Application for Hiring Managers.....	157
Controlling Access to Records.....	157
Introducing Organization-Wide Defaults.....	157
Org-Wide Defaults in Our Recruiting App.....	158
Try It Out: Setting Org-Wide Defaults.....	161
Introducing Role Hierarchies.....	163
Comparing Roles and Profiles.....	164
Role Hierarchies in Our Recruiting App.....	164
Try It Out: Defining a Role Hierarchy.....	165
What's Left to be Shared?.....	168
Introducing Sharing Rules.....	168
Sharing Rules in Our Recruiting App.....	169
Try It Out: Defining a Public Group for Reviewers.....	170
Try It Out: Defining a Sharing Rule for Job Application and Review Records.....	171
Introducing Manual Sharing.....	174
Try It Out: Defining a Manual Sharing Rule.....	174
Displaying Field Values and Page Layouts According to Profiles.....	175
Try It Out: Creating Record Types.....	176
Putting It All Together.....	180
Try It Out: Creating Users for Our Recruiting App.....	180
Try It Out: Verifying that Everything Works.....	183
Delegating Data Administration.....	184
Overriding Sharing with Object-Level Permissions.....	185
Delegated Administration Groups.....	186
Try It Out: Defining the Recruiting Manager Administration Group.....	187
Try It Out: Verifying that Delegated Administration Works.....	188
Summing Up.....	190

Chapter 9: Collaborating with Chatter.....	191
Introducing Chatter.....	192
Try It Out: Enabling Feed Tracking on Positions and Job Applications.....	192
Look at What We've Done.....	194
Tracking Fields on Your Custom Objects.....	195
Following Records.....	195
Try It Out: Following a Job Application Record.....	195
Summing Up.....	197
Chapter 10: Using Custom Workflow and Approval Processes.....	199
Introducing Workflow.....	200
Introducing Workflow Rules.....	201
Introducing Workflow Actions: Tasks, Field Updates, and Alerts.....	201
Workflow in Our Recruiting App.....	202
Creating Workflow Rules That Assign Tasks.....	203
Try It Out: Creating the “Send Rejection Letter” Workflow Rule.....	203
Try It Out: Creating the “Send Rejection Letter” Workflow Task.....	205
Try It Out: Creating the “Extend an Offer” Workflow Rule and Task.....	208
Look at What We've Done.....	209
Creating a Workflow Rule That Updates Fields.....	210
Introducing Queues.....	211
Try It Out: Creating a Queue for Positions.....	211
Try It Out: Creating a Workflow Rule That Updates Fields.....	214
Introducing Time-Dependent Workflow Actions.....	215
Try It Out: Creating the “Notify Recruiting Manager” Time-Dependent Workflow Task.....	215
Look At What We've Done.....	217
Creating a Workflow Rule That Sends Email Alerts.....	218
Introducing Email Templates.....	219
Try It Out: Building an Email Template.....	219
Try It Out: Creating the New Position Workflow Rule and Alert.....	222
Introducing Approvals.....	224
Planning for Approval Processes.....	226

Try It Out: Creating an Email Template for Approvals.....	226
Try It Out: Creating an Approval Process.....	227
Try It Out: Creating Approval Steps.....	231
Try It Out: Creating Approval Actions.....	235
Try It Out: Activating Our Approval Process.....	238
Look At What We've Done.....	238
Summing Up.....	242
Chapter 11: Analyzing Data with Reports and Dashboards.....	243
Introducing Reports.....	244
Report Formats.....	246
Setting Up the Recruiting App for Reports.....	248
Try It Out: Adding the Reports Tab.....	249
Try It Out: Creating a Recruiting Reports Folder.....	249
Creating a Summary Report.....	251
Try It Out: Creating a Summary Report.....	251
Try It Out: Adding Columns and Filters.....	254
Try It Out: Adding a Pie Chart.....	256
Creating a Matrix Report with Summary Fields, Time-Based Filters, and Conditional Highlighting.....	257
Try It Out: Creating a Matrix Report.....	258
Try It Out: Adding Summary Fields.....	259
Try It Out: Adding Columns and Filters.....	261
Try It Out: Adding a Chart and Conditional Highlighting.....	262
Introducing Dashboards.....	264
Try It Out: Creating Additional Reports.....	264
Try It Out: Creating a Dashboard.....	267
Adding Dashboard Components.....	269
Try It Out: Adding a Chart Component.....	270
Try It Out: Adding a Gauge Component.....	270
Try It Out: Adding a Table Component.....	271
Try It Out: Adding a Metric Component.....	272
Refreshing Dashboards.....	273
Introducing Custom Report Types.....	274
Creating Report Types.....	274
Look At What We've Done.....	279

Chapter 12: Moving Beyond Point-and-Click App**Development.....281**

Introducing Mash-Ups and Web Services.....	283
Introducing Visualforce.....	284
Introducing Visualforce Development Mode.....	286
Try It Out: Enabling Visualforce Development Mode.....	286
Implementing the Candidate Map	287
Try It Out: Creating a Visualforce Page.....	287
Try It Out: Writing Visualforce Markup.....	288
Try It Out: Adding the Map to Our Visualforce Page.....	291
Try It Out: Adding the Candidate Map to Position Page Layouts.....	292
Try It Out: Testing the Candidate Map.....	294
Implementing the Mass Update Status Button.....	295
Planning the Mass Update Status Feature.....	295
Try It Out: Creating the Mass Update Status Page.....	297
Understanding the MassUpdateStatus Visualforce Markup.....	298
Try It Out: Creating a Custom List Button.....	302
Try It Out: Adding a Custom List Button to a Page Layout.....	304
Try It Out: Testing the Mass Update Status Feature.....	305
Taking Your App Public with Sites.....	305
Introducing Sites.....	306
Introducing Force.com Domain Names.....	307
Try It Out: Registering Your Force.com Domain Name.....	307
Try It Out: Creating a Force.com Site.....	309
Try It Out: Creating the Public Jobs Page.....	313
Setting Your Active Home Page.....	316
Granting Public Access Settings.....	316
Testing Your Site.....	317

Chapter 13: Learning More.....319

Developer Force.....	320
Help and Training Options.....	321
Multimedia.....	322
AppExchange Partner Program.....	322
What Do You Think?.....	323

Glossary.....325
Index.....349

Preface

As users of the Internet, we're all familiar with the fascinating, innovative, creative, and sometimes silly ways in which it has changed how we work and play. From social networking sites to wikis to blogs, and more, it's exciting to watch the innovations taking place that are changing the ways we communicate and collaborate.

While these changes have certainly impacted how we work with content, a similar set of Internet-driven ideas and technologies is changing how we build and work with business applications. While yesterday's business applications required thousands, if not millions, of dollars and sometimes years of professional services help to set up and customize, the technologies offered by the Internet today make it much easier to create, configure, and use business applications of all kinds. Indeed, the power of the Internet has given us the ability to solve new kinds of business problems that, because of complexity or cost, had previously remained out of reach.

Just as the changes that moved publishing technology from paper to bits made it possible for us to have information about anything in the whole world right at our fingertips, the changes in application technology make it similarly possible to imagine a robust, enterprise-class application for almost any business need. Sound pretty good? Then you're probably wondering: "What's the magic that makes this possible?"

Welcome to the Cloud!

These new ways of building and running applications are enabled by the world of *cloud computing*, where you access applications, or *apps*, over the Internet as utilities, rather than as pieces of software running on your desktop or in the server room. This model is already quite common for consumer apps like email and photo sharing, and for certain business applications, like customer relationship management (CRM).

Because almost all apps these days are delivered via a Web browser, it's increasingly hard to tell which applications are "traditional software," and which are run in the cloud. As with the

Internet, applications that run in the cloud have grown so ubiquitous that almost every business user interacts with at least one, whether it's an email service, a Web conferencing application, or a sales system.

Platforms for Cloud Computing

A new twist, the *platform in the cloud*, is making the delivery of application functionality even more interesting. Increasingly, applications that run in the cloud are starting to look less like websites and more like platforms, meaning they are starting to sprout Application Programming Interfaces (APIs), code libraries, and even programming models. Collectively, these new kinds of development technologies can be thought of as platforms to run apps in the cloud.

Similar to traditional platforms, cloud computing platforms provide tools that allow developers to leverage existing functionality to create something new; however, because these platform tools are accessed freely over the Internet rather than through an operating system or package that was installed on a local machine, developers don't need to worry about the logistics of putting together an executable that will be installed on a user's machine. Anyone with a Web browser can access it!

The possibilities presented by this new type of platform have emerged quickly, spurred on by the popularity of *mash-ups*—a website or application that combines tools from multiple cloud computing platforms to create new functionality. Some of the cloud computing platform tools used in today's mash-ups include innovations like Google's search API, which allows developers to use the power of that search engine in their applications, eBay's APIs for auctions and listings, or Amazon.com's system for creating entirely new storefronts. For example, almost any real estate website or application these days uses Google or Yahoo! maps under the hood, illustrating how these new APIs are now commonly running alongside the more traditional database, app server, or operating system platforms.

About This Book

This book introduces you to the Force.com platform, salesforce.com's platform for building and running business applications in the cloud.

To illustrate the technologies available on the Force.com platform, and to show you just how easy it is to create your own business application with the platform, this book walks you through the process of creating a new recruiting application that runs in the cloud. To follow along you won't need to learn any programming languages or hack your way through cryptic configuration documents—instead, you'll just need to point-and-click your way through a Web interface, following the easy step-by-step instructions in the book.



Note: Want an online version of this book? Go to www.developerforce.com/events/regular/registration.php.

Intended Audience

This book can be easily understood by anyone from a business user to a professional developer. However, to get the most out of the book, it helps to be familiar with basic Internet and database concepts, such as tables and fields.

While the book focuses primarily on using the declarative, point-and-click functionality of the Force.com platform, [Moving Beyond Point-and-Click App Development](#) on page 281 introduces you to the platform's user interface programming tools and how you can use them to build customer-facing websites. To fully understand that chapter, you should be familiar with HTML and JavaScript. However, all the code you need is provided, so even if you're not an experienced developer, you can still follow along to gain a deeper understanding of what can be done with the Force.com platform.

Chapter Contents

If you're already familiar with the Force.com platform, you can skip around to the chapters in which you're most interested:

Chapter	Description
Introducing the Force.com Platform	Learn about the technologies behind the Force.com platform, including the AppExchange directory.
About the Sample Recruiting App	Learn about the recruiting application that we'll be building in this book and the fictitious company for whom we'll be building it.
Reviewing Database Concepts	Review database concepts such as tables, records, fields, keys, and relationships.
Building a Simple App	Create the first custom object in our recruiting app, and add several basic fields.
Enhancing the Simple App with Advanced Fields, Data Validation, and Page Layouts	Add picklists, dependent picklists, validation rules, and formula fields to the custom object,

Chapter	Description
	and then edit the layout of the object's detail page.
Expanding the Simple App Using Relationships	Add five more custom objects to our recruiting app, and associate them with one another using relationships.
Securing and Sharing Data	Set up rules for who can read, create, edit, and delete records in the app.
Collaborating with Chatter	Enable Chatter for your organization so users can keep up with the information they care about.
Using Custom Workflow and Approval Processes	Define workflow rules and approval processes that assign tasks, update fields, and send emails when certain criteria are met.
Analyzing Data with Reports and Dashboards	Create custom reports, charts, and dashboards that give users a bird's-eye view of recruiting data.
Moving Beyond Point-and-Click App Development	Learn how to use Visualforce to extend the functionality of the platform by creating a mash-up with Yahoo! maps, adding a tool for mass updating records, and publishing a subset of your data to anyone on the Web.
Learning More	Find out where you can get more information about developing on the platform.
Glossary	Look up the definition of any term you find unfamiliar.



Note: This book contains lots of screenshots. Because the Force.com platform is a rapidly developing platform, the screenshots might vary slightly from what you see on the screen, but don't worry! These differences should be minor and won't affect your understanding of the system.

Choosing Your Development Environment

To follow along with the exercises in this book, you'll need a Salesforce account. If you're already a Salesforce customer, you can use a Force.com *sandbox*. A sandbox is a copy of your organization that you can use for testing configurations and training users without compromising the data in your production organization. Salesforce Enterprise and Unlimited Editions come with one free sandbox; users of other editions can use Developer Edition to do the exercises.

If you're new to Salesforce or if you don't want to use a sandbox, go to developer.force.com and sign up for a free Developer Edition account. Developer Edition is a fully-functional version of Salesforce that you can use to develop Salesforce apps. Since it's free, there are limits on the amount of users, bandwidth, and storage you're allowed, but it includes all of the features in Salesforce. When you sign up, you'll also automatically become part of the growing community of Force.com platform developers around the world.

Sending Feedback

Questions or comments about anything you see in this book? Suggestions for topics that you'd like to see covered in future versions? Go to the Developer Force discussion boards at community.salesforce.com/sforce?category.id=developers and let us know what you think! Or email us directly at developerforce@salesforce.com.

About Developer Force

Developer Force is a community of developers who customize and build applications that run in the cloud and are built with the Force.com platform. Developer Force members have access to a full range of resources, including sample code, toolkits, an online developer community, and the test environments necessary for building apps. The Developer Force website includes an online version of this book and has information about the Dreamforce event that we hold every year for Force.com platform developers. If you need more info, have a question to ask, are seeking a toolkit or sample, or just want to dig a little deeper into Force.com platform development, Developer Force is where it all comes together.

To find out more about the resources available on the Developer Force website, see developer.force.com, and review the [Learning More](#) on page 319 chapter.

Salesforce.com Training & Certification

A number of examples in this book have been provided by salesforce.com Training & Certification and are drawn from the expert-led training courses available around the world. Salesforce.com Training & Certification courses provide an opportunity to get hands-on experience with the Force.com platform and Salesforce applications, and prepare you to become Salesforce certified. Register for courses at www.salesforce.com/training.

Chapter 2

Introducing the Force.com Platform

In this chapter ...

- [The Basics of an App's User Interface](#)
- [The Benefits of a Force.com Platform App](#)
- [The Technologies Behind a Force.com Platform App](#)

The Force.com platform is the world's first Platform as a Service (PaaS), enabling developers to create and deliver any kind of business application in the cloud, entirely on-demand and without software. It's a breakthrough new concept that is making companies radically more successful by letting them translate their ideas into applications in record time. Building, sharing, and running business applications has never been so easy.

To best understand the kinds of business apps you can build with the platform, we'll first briefly examine the basic components of the most popular application for the platform, salesforce.com's sales automation app, Salesforce SFA. We'll then highlight some of the key technologies that differentiate the Force.com platform from other platforms you may have already used.

The Basics of an App's User Interface

If you haven't used Salesforce before, you'll find it worthwhile to log in and spend a bit of time clicking around. Most Salesforce editions (including Developer Edition) have a basic Salesforce SFA app, so we'll start by looking at that. The interface for these tasks has a lot in common with the interface of whatever app you're planning to build.



Note: Haven't signed up for a Developer Edition account yet? Go to www.developerforce.com/events/regular/registration.php.

Tabs

As you can see when you start clicking around, there are a few key elements that form the foundation of the Sales Automation app and of most applications created with the platform. First, across the top of the app is a set of *tabs* that segment the app into different parts. Each tab corresponds to a type of object, such as an account or contact, and within a tab you can perform actions on particular records of that tab's type. For example, when you click on the Accounts tab, you can create a new record for the “Acme” account. You can also edit existing accounts, or use a *list view* to filter lists of accounts by certain criteria. Most app development work revolves around creating tabs and defining the data and behaviors that support them.

Forms

A second key element is the *form* that is displayed as part of a tab. As in any business app, forms are the primary means of entering and viewing information in the system. Forms allow you to view and edit the data associated with a particular record on a tab, like the contact “Jerome Garcia” on the Contacts tab. When developing a new app you can define what information appears in each form, and how it is organized. For example, the form for a contact record includes fields such as Last Name, Home Phone, Mailing City, Title, Birthdate, Reports To, and Account. In a Force.com platform app, the form used to enter information is referred to as an *edit page* and the read-only view of that information is referred to as a *detail page*.

Links

Finally, because Force.com platform apps are delivered via a Web browser, they use *links* to provide navigation to related data. For example, on an account detail page, there are links to

related records, such as the contacts who belong to the account and the sales user who manages the account. Other links take you to recently visited records and to areas of the app where users can set personal preferences. These links provide navigation both within an app and out into the Web.

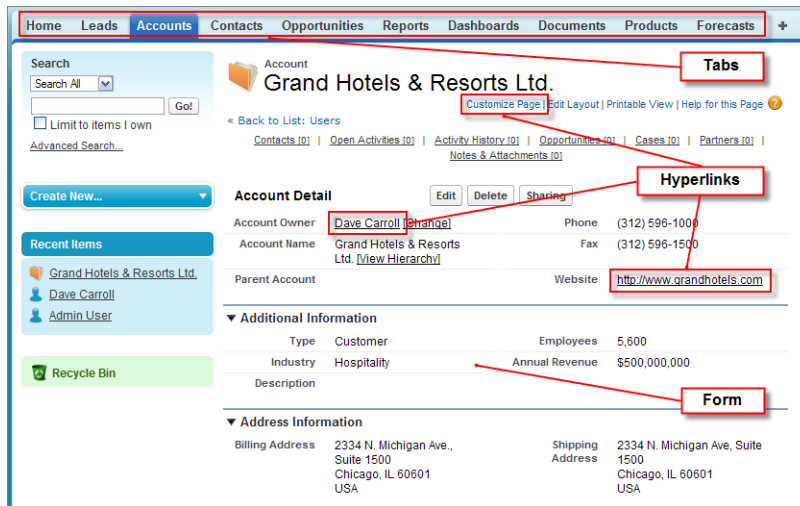


Figure 1: Force.com Platform Apps Include Tabs, Detail Pages, and Links

The Benefits of a Force.com Platform App

To better understand what the platform is best suited for, let's look beyond the core elements of tabs, forms, and links, and into the types of applications they enable. Two huge benefits start to come into focus when you look across Force.com platform apps in general: they're both data-centric and collaborative.

Data-Centric Apps

Because the platform is centered around a database, it allows you to write apps that are *data-centric*. A data-centric app is an application that is based on structured, consistent information such as you might find in a database or an XML file. We can find these data-centric apps everywhere, in small desktop databases like Microsoft Access or FileMaker, all the way to the huge systems running on database management systems like Oracle or MySQL. Unlike applications that are built around unstructured data, like plain text documents or HTML files, data-centric apps make it easy to control, access, and manage data.

For example, consider an exercise such as trying to determine the total sales for a month from a set of Microsoft Word-based contracts versus a set of contracts in a simple database. Whereas it takes a lot of effort to open each Word document, find the contract total, and then add them all together, if this data is stored in the database of a data-centric app, we can more efficiently get the same result by issuing a single query.

While most people don't need a data-centric application to keep track of anything other than contacts, photos, or possibly music, companies of all sizes constantly need to query and aggregate their large amounts of data to make fast business decisions. As a result, the data-centric nature of the Force.com platform makes it the perfect platform to build and host business applications.

Collaborative Apps

Because the platform can be accessed by multiple users at the same time, it also allows you to write apps that are *collaborative*. A collaborative app is an application with data and services that are shared by multiple users in different locations. Unlike more traditional forms of software that are installed on a single machine and are hard to access from a distance, collaborative apps on the platform can be accessed from anywhere in the world with only a Web browser. This makes it easy for teams to work together on activities like selling a product, managing a project, or hiring an employee.

In addition to easy access over a Web browser, a number of built-in platform features also facilitate productive group collaboration:

- The platform's security and sharing model allows you to finely control a user's access to different data
- Workflow rules allow you to automatically assign tasks, update data, or send email alerts when certain business events occur, such as the creation of a new record or a change in the value of a record field
- Approval processes allow you to set up a sequence of steps necessary for a record to be approved, including who must approve it at each step

Collectively, these features provide a framework for sharing apps across groups, divisions, and entire corporations without relinquishing administrative control over sensitive data.

The Technologies Behind a Force.com Platform App

Now that we've talked about the kinds of apps the platform can build, let's review some of the technologies behind the platform itself. These technologies have a big impact on what the platform supports, and what it's like to develop on it.

Table 1: Key Technologies Behind the Platform

Technology	Description
Multitenant architecture	An application model in which all users and apps share a single, common infrastructure and code base.
Metadata-driven development model	An app development model that allows apps to be defined as declarative “blueprints,” with no code required. Data models, objects, forms, workflows, and more are defined by metadata.
API Access	Several application programming interfaces (APIs) provide direct access to all data stored in Force.com from virtually any programming language and platform: <ul style="list-style-type: none"> • The Web services API • The REST API, an excellent technology for use with mobile applications and Web 2.0 projects • The RESTful Bulk API (also available using Data Loader) for operations on large numbers of records
Apex	The world’s first on-demand programming language, which runs in the cloud on the Force.com platform servers.
Visualforce	A framework for creating feature-rich user interfaces for apps in the cloud.
Salesforce Mobile	An application for mobile devices that provides access to your Force.com data.
Sites	Public websites and applications that are directly integrated with your Salesforce organization—without requiring users to log in with a username and password.
AppExchange directory	A Web directory where hundreds of Force.com apps are available to Salesforce customers to review, demo, comment upon, and/or install. Developers can submit their apps for listing on the AppExchange directory if they want to share them with the community.

A Multitenant Architecture

The platform's *multitenant architecture* means that all users share the same physical instance and version of any application that runs on it. In contrast to their single-tenant counterparts, such as client-server enterprise applications or email servers, multitenant applications are

designed so that any upgrades to the platform or the apps it supports happen automatically for all users at once. Consequently, no one has to worry about buying and maintaining their own physical stack of hardware and software, or making sure that their applications always have the latest patch installed.

Besides the Force.com platform, several popular, consumer-based applications also use a multitenant architecture, including eBay, My Yahoo!, and Google Mail. Multitenant architecture allows these applications to be low cost, quick to deploy, and open to rapid innovation—exactly the qualities for which salesforce.com has also become known.

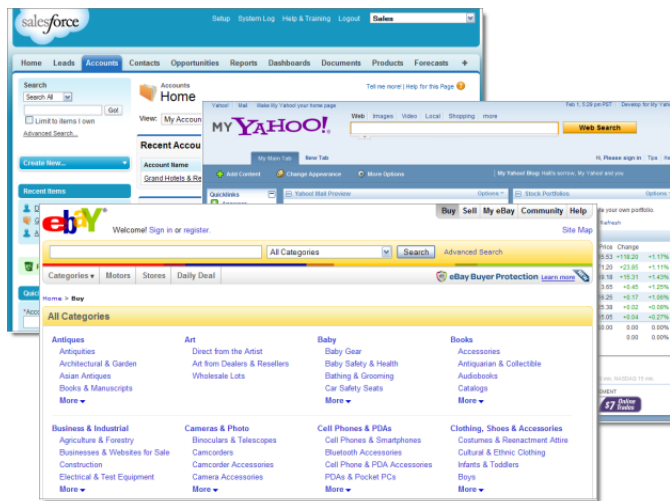


Figure 2: On-Demand, Multitenant Applications that Run in the Cloud

The platform's multitenant architecture also impacts how developers use the platform to create new applications. Specifically, it defines a clear boundary between the platform and the applications that run on it. A boundary is important because it allows applications to define their own components without jeopardizing the functionality of the core platform or the data stored by other users.

A Metadata-Driven Development Model

The Force.com platform also uses a *metadata-driven development model* to help app developers become more productive in putting together apps. It means that the basic functionality of an app—that is, the tabs, forms, and links—are defined as metadata in a database rather than being hard-coded in a programming language. When a user accesses an app through the Force.com platform, it renders the app's metadata into the interface the user experiences.

As a result of metadata-driven development, the Force.com platform app developers work at a much higher level of abstraction than if they developed applications using Java or C#, and are shielded from having to worry about low-level system details that the platform handles automatically. At the same time, Force.com platform developers can also leverage advanced features that the platform provides by default.

Customizing your app's metadata might sound intimidating, but as you'll see in this book, the platform's user interface makes it easy. Anyone who is familiar with using a Web browser can quickly get up to speed, even if he or she doesn't know any programming languages.



Tip: Developers can use the *Force.com Metadata API* to programmatically manage their app's setup. The Force.com Metadata API provides an alternative to the platform's user interface by allowing developers to directly modify the XML files that control their organization's metadata. Developers can also use the Metadata API to migrate configuration changes between organizations, and create their own tools for managing organization and application metadata. For more information, see www.salesforce.com/us/developer/docs/api_meta/index.htm.

Although at first glance metadata-driven development may seem somewhat esoteric, it's exactly the same model for how Web browsers work. Instead of hard coding the definition of a Web page in a free-form programming language, a Web page author first defines the page as HTML, which is itself a kind of metadata. When a user requests a page, the Web browser renders the page using the metadata provided in the HTML tags. Even though the HTML/browser combination does not allow authors as much formatting power as they might get in a regular publishing tool, it simplifies the work of publishing content to a wide audience and increases the Web page author's overall productivity.

Likewise, the Force.com platform vastly simplifies the work of building an app and increases a developer's overall productivity. And, like Web pages that use JavaScript or Flash to add functionality to HTML pages, the Force.com platform also provides ways for more advanced developers to add custom functionality to the apps you build.

The Web Services API

The platform's metadata-driven development model allows app developers to quickly build a lot of functionality with tools provided by the platform; however, sometimes app developers want to modify the actual data in an app, and use third-party services to create more customized app behaviors. To do this, they can use the Web services API.

The API provides a straightforward, powerful, and open way to programmatically access the data and capabilities of any app running on the platform. It allows programmers to access and

manipulate apps from any server location, using any programming language that supports Web services, like Java, PHP, C#, or .NET. Because Web services are, not surprisingly, based on Web standards, they're well suited to traverse firewalls and leverage the rest of the Internet infrastructure already in place. For more information, see www.salesforce.com/us/developer/docs/api/index.htm.

Apex

As you might expect from the company that delivered the world's first cloud computing platform, salesforce.com also introduced the world's first cloud computing programming language, Apex. Apex, whose syntax is similar to Java, the most popular programming language for Web apps, runs on the Force.com platform servers. Apex is specifically designed for building business applications to manage data and processes within the larger context of the Force.com platform. The language provides a uniquely powerful and productive approach to creating functionality and logic, allowing developers to focus just on the elements specific to their application, while leaving the rest of the “plumbing” to the Force.com platform.

The majority of this book is intended for readers who don't necessarily code, so Apex is beyond the scope of what we'll discuss here; however, you can learn everything there is to know at wiki.developerforce.com/index.php/Apex.

Visualforce

At the front of any great business application is a great user interface that's easy to use, powerful, and suited exactly for the tasks, users, and devices the application serves. Visualforce is a complete framework for creating such user interfaces, enabling any kind of interface design and interaction to be built and delivered entirely in the cloud. The user interfaces you build with Visualforce can extend the standard Force.com platform look and feel, or replace it with a completely unique style and set of sophisticated interactions. Because Visualforce markup is ultimately rendered into HTML, designers can use Visualforce tags alongside standard HTML, JavaScript, Flash, or any other code that can execute within an HTML page on the platform. And that's only the beginning: you can also use Visualforce pages to combine data from multiple Force.com platform objects, or blend data from Web services into your applications, as we discuss in [Moving Beyond Point-and-Click App Development](#) on page 281.

Salesforce Mobile

All around the globe, members of the business community are leaving their office chairs and going mobile. Companies are issuing the latest smartphones to their employees to keep them

connected and productive. Tablets are sprouting up in cafes and conference rooms, enabling efficiency everywhere. People are finding it less and less acceptable to be more than a click (or tap) away from their data at all times.

As the primary points of Internet access shift from desktops and laptops to iPhones and iPads, apps that don't provide mobile access to critical data will quickly become obsolete. Don't let your app get trampled by the mobile stampede! Instead, use Salesforce Mobile to deliver your Force.com app to mobile users.

Salesforce Mobile is an app itself. It installs on mobile devices and utilizes the native functionality of the device. When users log in to Salesforce Mobile, they can access and update their Force.com data via an intuitive interface specially designed for the smaller form factor of mobile device screens. The app stores a subset of data on the device to provide users with offline access when a wireless connection is unavailable, and synchronizes that data with Force.com when connectivity is reestablished.

The Force.com platform gives administrators full control over what users can and can't do with Salesforce Mobile. For example, administrators can specify which data is available for mobile access, limit the amount of data that users can transfer, and even remotely disable Salesforce Mobile if a device is lost or stolen.

Salesforce Mobile is supported on the Apple iPad, iPhone, and iPod, as well as various BlackBerry devices. When you're ready to take your app mobile, see the [Salesforce Mobile Implementation Guide](#).

Sites

The apps you build on the Force.com platform might contain data and functionality that you want to make accessible to people who are not Salesforce users. While it is possible to use the Web services API to integrate an external Web page or application with Salesforce, the Force.com platform provides an easier, more efficient way of sharing data and functionality with people outside of your organization: Sites.

Sites enables you to create public websites and applications that are directly integrated with your Salesforce organization—without requiring users to log in with a username and password. You can publicly expose any information stored in your organization through pages that match the look and feel of your company's brand. Because these sites are built and hosted on the Force.com platform servers, there are no data integration issues. And because sites are built with Visualforce pages on the platform, data validation on collected information is performed automatically. For more information, see developer.force.com/sites.

The AppExchange Directory

The final piece of technology that differentiates the Force.com platform from other platforms is the *AppExchange*. The AppExchange is a Web directory where apps built on the Force.com platform are available to salesforce.com customers to browse, demo, review, and install.

Developers can submit their apps for listing on the AppExchange directory if they want to share them with the community.

To fully appreciate the benefits of the AppExchange, take a quick tour at <http://sites.force.com/appexchange>. There you'll see the hundreds of innovative and exciting apps that exist today, including everything from payroll management to telephony integration, service and support surveys, adoption dashboards, and beyond. Some of these apps have been created inhouse at salesforce.com, but most are built by partners and individual developers who have chosen to take advantage of the Force.com platform.

Chapter 3

About the Sample Recruiting App

In this chapter ...

- [About Universal Containers](#)
- [Considerations for the Recruiting App](#)
- [Building the App: Our Design](#)

The goal of this book is to show you how easy it is to create powerful, multifaceted applications that solve common business problems. To do so, let's walk through the steps of creating a simple application for a make-believe company called Universal Containers.

Like many companies that have grown rapidly, Universal Containers has been experiencing a few growing pains, especially in its Human Resources department. In this book, we're going to build a Recruiting app for the company that allows it to move away from the Microsoft Word documents and Microsoft Excel spreadsheets that it has traditionally used to an application that's available on demand.

By the time we finish building the Recruiting app in this book, you should feel confident enough to build a custom application in the cloud that suits your own company's needs. So let's get started!

About Universal Containers

First, let's learn a little more about our fictional company, Universal Containers.

Universal Containers is a rapidly growing international supplier of container products. The company produces every kind of container from simple overnight letter mailers to custom equipment packaging to large cargo shipping containers. In addition, Universal Containers develops and maintains its own proprietary software to facilitate the design of its various types of containers. As such, Universal Containers has a very diverse group of employees, including facilities and operations professionals, software and design engineers, financial accountants, and legal and human resources personnel.

Historically, the Human Resources department has used Microsoft Word documents and Microsoft Excel spreadsheets to manage the recruiting and hiring process for new employees. However, over the last two quarters it's become evident that unless this process is replaced by one that is more collaborative, reliable, and scalable, the department won't be able to meet its hiring goals for this fiscal year. Universal Containers needs a centralized application that can bring all of its recruiting and hiring processes together, and the company has hired us to solve this problem. Our approach will be to leverage their Salesforce account and build a recruiting application on the Force.com platform. We're going to introduce Universal Containers to the world of cloud computing!

Considerations for the Recruiting App

After meeting with Megan Smith, Universal Containers' vice president of Human Resources, we've drawn up a few requirements for the new Recruiting app. The app needs to:

- Track positions in all stages of the process, from those that are open to those that have been filled or canceled
- Track all of the candidates who apply for a particular position, including the status of their application (whether they've had a phone screen, are scheduled for interviews, have been rejected or hired, or have passed on an offer that was presented)
- Track the posting of jobs on external employment websites, such as Monster.com
- Allow employees to post reviews for candidates whom they've interviewed
- Provide security for the recruiting data so that it's not mistakenly viewed, edited, or deleted by employees who shouldn't have access
- Automatically inform the relevant recruiter about the next steps that should be taken when a decision has been made about an applicant
- Automatically inform all employees of new positions that have been posted

- Make sure that a new job opening has executive approval before it becomes active
- Include reports that give users an overview of recruiting status
- Allow recruiters to map the locations of all candidates who are applying for a position, to better understand relocation expenses
- Make it easy to perform several similar tasks at once, like rejecting multiple job applications
- Automatically post open positions on Universal Containers' public website

An app that meets these requirements is going to greatly increase the efficiency of Universal Containers' recruiting and hiring processes.

Building the App: Our Design

Let's take a look at the different parts of the Force.com platform that we'll use to implement Universal Containers' Recruiting app. We're going to learn about all of these things in a lot more detail in later chapters, but for now, this quick preview of will give you an idea about what's in store.

Custom Objects

Custom objects are the native components that model the data we need to store in our Recruiting app. Similar to a database table, a custom object is composed of several fields that store information such as a job applicant's name, or the maximum salary for a particular position. However, unlike traditional database tables, we don't need to write any `SQL` in order to create custom objects. We can simply point and click in the platform to create as many objects as we need.

For our Recruiting app, we'll be creating six custom objects to track recruiting-related data:

- Position
- Candidate
- Job Application
- Review
- Job Posting
- Employment Website

Most of these objects will be displayed as tabs in our application. When a user clicks one of the tabs, he or she will have access to individual instances of that particular object, as shown in the following screenshot.

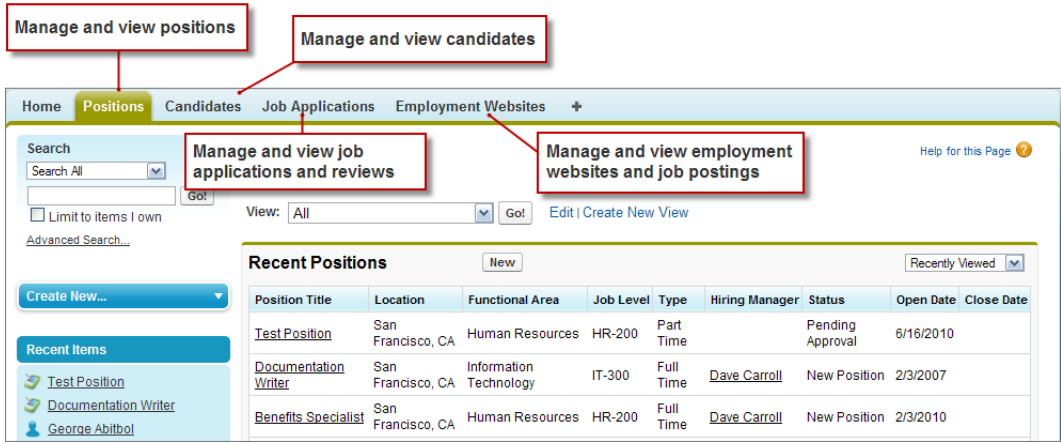


Figure 3: Recruiting App Tabs

One of the powerful features of a custom object is the fact that it can have relationships with other objects in the system. For example, for every review written by an interviewer and entered into the system, we'll want to associate it with the job application of the candidate who was being interviewed. Again, we won't need to write any SQL to make this happen—thanks to the platform, defining a relationship will be as simple as a few clicks of the mouse.

Security and Sharing Rules

Another important function that we'll need to build into our app is the ability to restrict access to data that particular users shouldn't see, without preventing other users from performing their jobs effectively. We're going to implement this requirement with a group of components that we've grouped under a single term: *security and sharing rules*.

With security and sharing rules, we'll first specify which custom objects a particular user should be allowed to create, view, or edit (for example, Candidate and Position), and then which instances of those objects should be accessible (for example, the records for candidate John Smith or the Senior Sales Manager position). Controlling our data either with the wide brush of object-level security or with the more detailed brush of record-level security will give us a lot of power and flexibility in controlling what users can and can't see.

Workflow and Approval Processes

Three of our requirements involve automating business processes, such as triggering an alert email to a recruiter whenever a job application's status has changed, and submitting new job

openings for executive approval. Once again, the Force.com platform makes these requirements easy for us to implement natively with the built-in *workflow* and *approval process* components.

Workflow and approval processes allow us to create business logic based on rules:

- Workflow rules can assign tasks to users, update fields, or send email alerts
- Approval processes allow users to submit sensitive records like new contracts or purchase orders to other users for approval

For example, in our Recruiting app, we can create a workflow rule that triggers an event whenever the status of a job application has changed to Reject or Extend an Offer, as illustrated below.

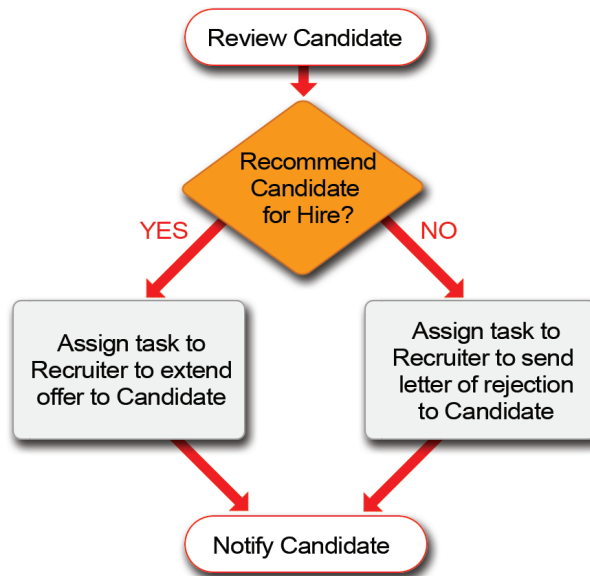


Figure 4: Workflow When a Job Application's Status Has Changed

When a hiring manager makes a decision to either extend an offer to or reject the candidate, changing the status of the application triggers the appropriate task to be assigned to the recruiter for that position. Based upon the hiring manager's decision, the recruiter performs the appropriate follow-up task.

Similarly, we can define an automatic approval process that sends all new positions to the appropriate management for approval. If the position is approved, its status automatically changes to Open - Approved and recruiters can start the hiring process. If the position is rejected, its status automatically changes to Closed - Not Approved and the position won't be filled.

Custom Reports and Dashboards

Finally, we need to give users a way to inspect the status of all positions and job applicants in the Universal Containers recruiting program. Managers need to delve into the intricate details of how each recruiter is performing, while executives just want a high-level overview of how departments are doing with staffing goals.

We can meet these requirements using reports and dashboards. Using report builder, we can create detailed reports with filters, conditional highlighting, subtotals, and charts. With dashboard builder, we can quickly create a dashboard of up to 20 different components on a single page.

Visualforce

We'll be able to use point-and-click tools to satisfy nearly all of our Recruiting app use cases; however, there are a few use cases, such as the mapping of candidate locations and the posting of positions on Universal Containers' public jobs site, that will require us to use Visualforce, the Force.com platform's tag-based markup language that allows you to build sophisticated, custom user interfaces for apps. We won't address these use cases until the very last chapter, and when we do, this book will provide all the code you need.

Although we haven't yet gone into detail about how any of this stuff is going to work, you can probably see now just how flexible and powerful the Force.com platform can be when you're creating a custom app.

In the next chapter, we'll start out by building our first custom object. We'll swiftly get a feel for how the platform interface works, and it won't be any time at all before you're building app components easily and quickly. When it's this easy, you can't help but become an expert!

Chapter 4

Reviewing Database Concepts

In this chapter ...

- [What's a Database?](#)
- [What's in a Database?](#)
- [What's a Relational Database?](#)
- [Summary of Database Concepts](#)

Now that we've introduced the power of the Force.com platform and learned about the requirements of the Recruiting app that we're going to be building, let's take a moment to talk about databases and why a simple understanding of database concepts can help you realize the full potential of the platform and make your app development a whole lot easier.

As you know, the underlying architecture of the platform includes a database where your data is stored. This means that all of the information you enter is stored in that database and then retrieved from the database whenever you view it within your app.

Historically, companies were required to buy, build, and maintain their own databases and IT infrastructures in order to distribute and run their applications. Cloud computing on the Force.com platform provides an alternative and makes it easy for you, as a company or as a sole developer, to build and deliver your app. Part of the simplicity of the cloud computing model is that the technical responsibilities of maintaining and running all of the database hardware and software is handled by the hosting company (in this case, salesforce.com), so you can focus on developing your app.

It's worth pointing out that although your data is stored in a database and a simple understanding of database concepts is helpful, you don't need to be a database developer to build an app on the platform. We won't be doing any traditional database programming in the course of developing our app.

What's a Database?

In simple terms, a *database* is an organized collection of information. Common examples include a phone book, a library catalog, an employee directory, a catalog of the MP3s you own, or in the case of our Recruiting app, information about the open positions at a company, the people who are applying for those positions, and the managers at our company who are in charge of hiring each position.

Typically, you use a database to collect information about people, things, or concepts that are important to you and whatever project you're working on. In standard database language, the category of a person, thing, or concept you want to store information about is referred to as an *entity*, although in standard Force.com platform terminology, we refer to this as an *object*.

In a database, each entity is represented by a *table*. A database table is simply a list of information, presented with rows and columns, about the category of person, thing, or concept you want to track. So in a phone book, you might have a table to store information about residences and another table to store information about businesses; or in a library catalog, you might have one table to store information about books and another to store information about authors.

In our Recruiting app, we'll have one table to store information about open positions, another table to store information about the candidates applying for the positions, and a table to store information about hiring managers. (Our Recruiting app will have more than just this, but we'll get to that later.)

In very simplistic terms, a Force.com platform object is similar to a database table in that you'll have a separate object for each person, thing, or concept about which you want to collect information. In reality, a Force.com platform object is much more than this because the full functionality of the platform is behind each object. Each object automatically has built-in features like a user interface, a security and sharing model, workflow processes, and much more that you'll learn about in the rest of this book.



Note: As we introduce database concepts, “object” and “table” will be used interchangeably because they are similar. Just remember that a Force.com platform object is much more than just a database table.

It's important to understand that a single database table, or Force.com platform object, should contain only one type of information. You don't want to lump all of your information into one table, so you wouldn't store positions, candidates, and hiring managers all in the same place. Not only is this not good database design, but it doesn't allow you to relate objects to one another. For example, if all of our data were in one table, how would we ever know which

candidates were applying for which positions, or which managers were in charge of hiring for which positions?

As we define our app, it's important for us to keep this in mind and ask ourselves questions like, “What kind of information do we want to store? Can we separate our information into distinct categories so that each object holds only one type of person, thing, or concept?” The answers to these questions will guide us as we design the structure of our application.

What's in a Database?

As we mentioned, a database table presents your information in rows and columns. Let's take a look at how a table of positions might look:

Position Title	Education Requirements	Functional Area	Max Pay	Min Pay
Executive Assistant	Associate degree	Human Resources	60,000	40,000
Recruiter	Bachelor's degree	Human Resources	110,000	85,000
SW Engineer	Bachelor's degree	Engineering	140,000	110,000
SQA Engineer	Bachelor's degree	Engineering	140,000	110,000

Figure 5: Position Information in a Table

Each row in the table represents the information about a specific instance of the object, for example, the Recruiter position or the SW Engineer position. In standard Force.com platform terminology, we call this a *record*. For every object you want to track in your app, you'll have multiple records to represent each individual item about which you're storing information. It's common for users who are new to the platform to confuse the meanings of object and record. It'll make your development a lot easier if you remember that an object is a category of information, such as a position or candidate, and the record is a single instance of an object, such as a SW Engineer.



Note: As a side note here, we'll mention that the platform includes a set of built-in objects when you first start using it; we call these *standard objects*. One example of a standard object is the User object, which stores information about each person who is a user of the app, like our hiring managers. You can also build your own objects to store information that's unique to your app; we call these *custom objects*. Both standard

objects and custom objects are not really all that different—one kind is prebuilt for you, and the other you build yourself. We'll talk more about these later as you start to build your app.

Now let's look at the columns in the table. Each column lists a particular piece of information such as the Position Title or Max Pay. We refer to these as *fields*. Every object has a set of fields that you use to enter the information about a particular record. For each field in the table, a single item of data that you enter, such as Human Resources in the Functional Area, is referred to as a *data value*.

Just like objects, fields come in two varieties: standard and custom. The standard fields are the ones that are built into the platform and automatically added for you. The custom fields are the ones you define to store specific pieces of information that are unique to your app. Fundamentally, there is no difference between standard and custom fields. Both are simply columns in the database table. We'll talk more about standard and custom fields later when you begin building your app.

What's a Relational Database?

Now you have some information stored in your database, but so what? You could easily make a list of positions using Microsoft Excel or some other spreadsheet software. For each position, you could even list the hiring manager in a field called Hiring Manager, like this:



Position Title	Education Requirements	Functional Area	Max Pay	Min Pay	Hiring Manager
Executive Assistant	Associate degree	Human Resources	60,000	40,000	Phil Katz
Recruiter	Bachelor's degree	Human Resources	110,000	85,000	Megan Smith
SW Engineer	Bachelor's degree	Engineering	140,000	110,000	Andrew Goldberg
SQA Engineer	Bachelor's degree	Engineering	140,000	110,000	Ben Stuart

Figure 6: Position Information with Hiring Manager Field

But what if a hiring manager is responsible for hiring more than one position? You would need to have duplicate records for the same hiring manager so you could capture every position for which that hiring manager is responsible, like this:

Hiring Managers duplicated for each of their positions

Position Title	Education Requirements	Functional Area	Max Pay	Min Pay	Hiring Manager
Executive Assistant	Associate degree	Human Resources	60,000	40,000	Phil Katz
Recruiter	Bachelor's degree	Human Resources	110,000	85,000	Megan Smith
Senior Recruiter	Bachelor's degree	Human Resources	115,000	95,000	Megan Smith
SW Engineer	Bachelor's degree	Engineering	140,000	110,000	Andrew Goldberg
Senior SW Engineer	Bachelor's degree	Engineering	140,000	110,000	Andrew Goldberg
SQA Engineer	Bachelor's degree	Engineering	140,000	110,000	Ben Stuart

Figure 7: Position Information with Duplicate Hiring Managers

This is not a good database design! Using this approach, data is repeated unnecessarily. In addition, there is really no way to capture additional information about our hiring managers, like their email addresses or phone numbers. And if we try to add information about which candidates are applying for each position, you can imagine that our simple table will quickly become extremely complex and unmanageable.

As we mentioned before, you want to create separate database tables, or objects, for each person, thing, or concept you want to track. A better way to model our scenario here would be to create one object for positions, one object for candidates, and one object for hiring managers. (Luckily, the platform has a standard object that we'll be able to use to represent our hiring managers—the User object.)

Once we have our data separated into discrete objects, we can easily relate objects to each other. This is what a relational database is all about! A *relationship* is an association between two or more tables. For example, we can relate positions to hiring managers so we know which positions each hiring manager is responsible for:

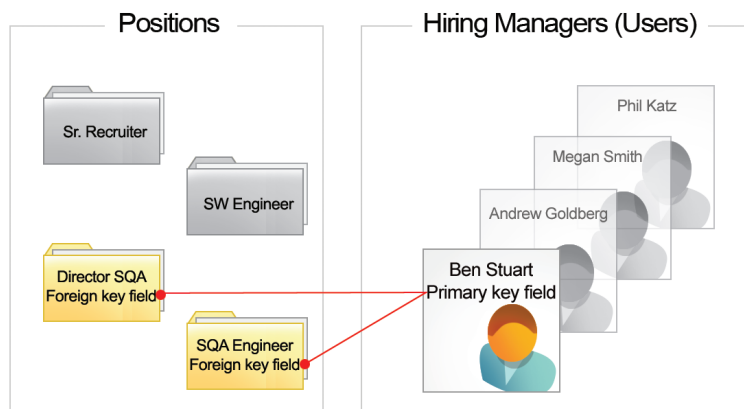


Figure 8: Positions Related to Hiring Managers

From a technical standpoint, each table in a relational database has a field in which the data value uniquely identifies the record. This field is called the *primary key*. The primary key is one part of what defines the relationship; the other part is the *foreign key*. A foreign key is a field whose value is the same as the primary key of another table. You can think of a foreign key as a copy of a primary key from another table. The relationship is made between two tables by matching the values of the foreign key in one table with the values of the primary key in another.

Primary and foreign keys are fundamental to the concept of relationships because they enable tables to be related to each other. As you begin building your app, you won't really need to think too much about primary keys and foreign keys. The important concept to understand here is that in a relational database, objects are related to each other through the use of common fields that define those relationships.

Summary of Database Concepts

At this point, we're ready to dive into the building of our Recruiting app. But first let's recap what we've learned about databases. Whether this was your first introduction to databases or whether you're already an experienced database developer who's new to the Force.com platform, the important things to remember are:

- A *database* is an organized collection of information.
- A database table stores information about a single type of person, thing, or concept—such as a job position. In the Force.com platform, we use the term *object* here (even though an object is much more than this, as you'll see).
- A database row, or *record* in Force.com platform terms, represents a single instance of an object—such as the SW Engineer position.
- A *field* stores a particular piece of information on a record.
- *Relationships* define the connection between two objects, and objects are related to each other through the use of common fields.

Now that we've got that all covered, let's get started building our first object!

Chapter 5

Building a Simple App

In this chapter ...

- [Becoming Familiar with the Setup Area](#)
- [Introducing Apps](#)
- [Introducing Objects](#)
- [Introducing Tabs](#)
- [Becoming Familiar with Setup Detail Pages and Related Lists](#)
- [Introducing Fields](#)
- [Look at What We've Done](#)

Just as traditional programming books first teach you how to write a simple “Hello World” program before getting into more complicated things, in this chapter, we're going to create a very simple version of the Recruiting app to show you just how easy it is to get started with the Force.com platform. Along the way we'll orient ourselves to the platform's user interface (where we'll be doing most of our work), and we'll learn how to create and configure our first custom object. Although easy and straightforward, the tasks we complete here will be the first step in developing a full-featured Recruiting app. So let's dive right in!

Becoming Familiar with the Setup Area

Since we're going to spend most of our time working in the Setup area of the platform, let's first become familiar with what it is and how to navigate to it.

The Setup area is a user preferences area, an application building and customization environment, and an administration tool, all in one. We perform almost every task we need to create our app in the Setup area, so most of the “Try It Out” sections of the book are going to start with an instruction like, “Click **Your Name** ► **Setup** ► **Create** ► **Apps**.” This is a short way of saying:

1. Click the **Your Name** ► **Setup** link in the top-right corner of the page (shown in the following screenshot).
2. Go to the App Setup area on the left side of the page.
3. Click the + icon to expand the **Create** menu, or just click the **Create** link.
4. Click the **Apps** link.

The final link that you click (in this example, **Apps**) will change depending on the task you're trying to perform, but you get the general idea.

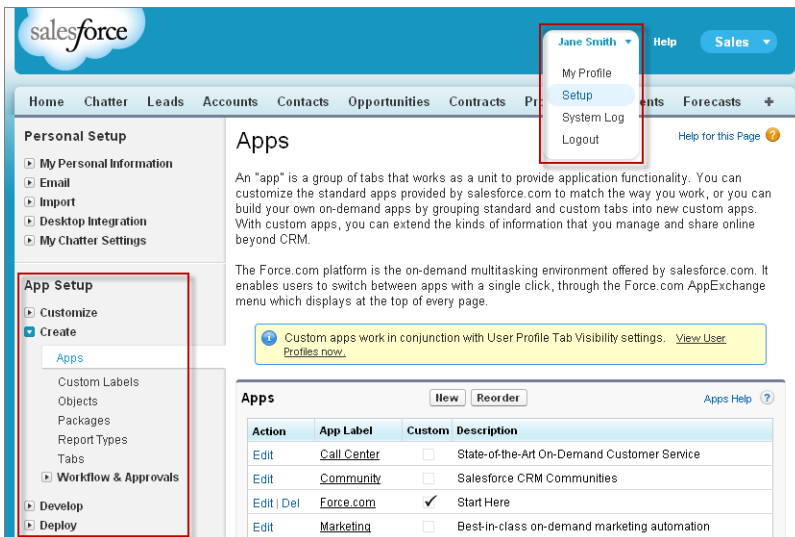


Figure 9: The Setup Area

Similar to the other parts of the application, the Setup area consists of a tab bar, a navigational sidebar, and a main window:

- The tab bar is made up of the same tabs that appear in the regular application. Just click on any one of the tabs to exit the Setup area and go to that tab in the main application.
- The navigational sidebar includes expandable lists of all the tools that are available in the Setup area:

Personal Setup

These tools control individual preferences and are available to all users.

App Setup

These tools configure the standard objects, custom objects, and custom apps that are deployed and are typically used only by administrators.

Administration Setup

These tools configure the platform as a whole and are typically used only by administrators.

Force.com Checkout

These tools let you purchase licenses and products from salesforce.com, change your billing information, and so forth.

- The main window is where the navigational links or a selected setup tool are actually displayed.

Now that we know what we're looking at, let's start creating our simple app.

Introducing Apps

What should we do first? If we were writing a software application, the first thing we'd need to do is build a project where we could store all the code that we were going to write. With the Force.com platform, the first thing we need to do is create a new app.

Like a programming project, an *app* is little more than a container for all of the objects, tabs, and other functionality that we're going to build as part of our Recruiting application. It consists simply of a name, a logo, and an ordered set of tabs. The simplest app contains only one tab—the Home tab—and a default logo. As we define more tabs in the remainder of this book, we can add them to the app later.

Let's start clicking through the process of actually creating a simple app now. Log in to your Salesforce account so you can follow along!



Note: Because the platform is continually evolving, you might find that the screenshots you see in this book vary slightly from what you see on your screen. These changes should be minor and shouldn't affect your understanding.

Try It Out: Defining an App

To create an app:

1. Open a browser and go to `www.salesforce.com`.
2. Click **Customer Login**.
3. Enter your username and password.
4. Click **Your Name** ► **Setup** ► **Create** ► **Apps**.
5. If you see an introductory splash page, simply click **Continue**.



Note: You'll find that many parts of the application have these splash pages to help you understand what you can do with the platform. If you never want to see a particular page again, just click **Don't show me this page again**.

Welcome to the Apps list page! Like many of the setup tools, the starting page for the Apps tool consists of a list of all the apps that are currently enabled for your organization. Depending on what edition you're using or what you've already installed from the AppExchange, you'll probably already have some standard apps listed here.

6. Click **New**. The New Custom App wizard appears.
7. In the App Label field, enter `Recruiting`.

The app label is the name that will represent our new app in the Force.com app menu that appears at the top right of all pages. Users can use this menu to switch back and forth between apps.

Notice that a vertical red bar appears just to the left of this `Label` field. This red bar indicates that you must provide a value for this field in order to save your work. If you don't enter a value here and try to proceed, an error message is displayed, as shown in the following screenshot.

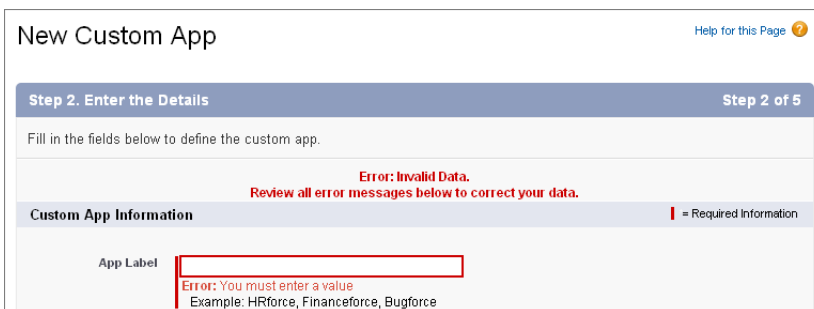


Figure 10: Required Fields Highlighted in Red

8. Click your mouse inside the `App Name` field.

The app name is what the developers use to identify an app when writing code for the Force.com platform. We won't be doing anything in this book that uses the app name, but the field is required, so it needs a value. Fortunately, when you enter a value in the `App Label` field, the same value should automatically appear in the `App Name` field. If it doesn't, enter `Recruiting` in the `App Name` field now.

9. In the `Description` field, enter `Manage positions, candidates, and job applications, and track job postings on employment websites.`

10. Click **Next**.

The next screen in the New Custom App wizard allows you to specify the image file that should be used for this app's logo. Whenever the app is selected in the Force.com app menu, this is the logo that appears in the upper-left corner of all pages. Since we're just creating a simple app, let's accept the default logo that's already provided. We can always change it later.

11. Click **Next**.

As we said before, an app is a container for an ordered collection of tabs, and this step of the New Custom App wizard allows us to specify which tabs we want to include in our new app. The `Available Tabs` list shows us the standard and custom tabs that are available for us to choose, and the `Selected Tabs` list shows us which tabs are already included, listed in the order that they should be displayed. You'll notice that one tab, the `Home` tab, is already included in our app by default. This is because the `Home` tab is required in every app, and must always be in the first position; however, you can use the `Default Landing Tab` drop-down menu to select which tab is first displayed when the app opens.

Again, since we're just creating a simple app, let's accept the defaults and move on. We'll add more tabs later.

12. Click **Next**.

Now that we've defined some of the basic features of our app, you might be wondering what remains to be done in the New Custom App wizard—shouldn't we already be done? It turns out that one crucial step remains: we need to define the users who will be allowed to access our app.

In this step of the New Custom App wizard, we can choose which user profiles should have access to the app. We'll learn more about profiles in [Securing and Sharing Data](#) on page 131. For now, just understand that every user is assigned to a profile, and profiles control which apps the users assigned to that profile can view.

13. Select the `Visible` checkbox next to the Standard User and System Administrator profiles.
14. Click **Save**.

That's it!

Look at What We've Done

Now that we've made it back to the Apps list page, let's see what we've just done. First of all, we've got a new entry in the Apps list—our Recruiting app! It shows up in the list in the same order it's going to appear in our Force.com app menu. In fact, let's go look at the Force.com app menu now.

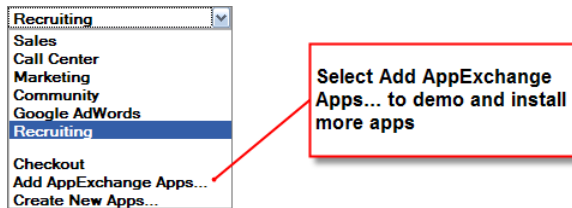


Figure 11: Force.com App Menu



Tip: If you want to change the position of our app in this menu, do so from the Apps list page by clicking **Reorder** and rearranging the available apps as you see fit.

Now select the Recruiting app from the menu and see what happens—our app is launched with a single Home tab! We've created the Recruiting app's Home tab, and we've added it to the Force.com app menu. That's how easy it is to get started.

You'll notice that the approach we're taking here is iterative: we'll build part of the app, look at what we've accomplished, and then add to it. This sequence not only reflects the fact that we're leading you through the steps of building an app in this book, but you'll also find that in building Force.com platform apps in general, this iterative process is common.

During the course of this book, you'll also notice that unlike with traditional coding projects, your app is always functional. There's no build or compile phase, and as a result, you'll almost never be chasing down syntax bugs or other typos. In fact, with this simple one-tab app, you can already utilize all of the built-in functionality that comes with the platform, including search, calendar events and tasks, user preferences, and a familiar user interface.

Introducing Objects

Now that our app is functional (but rather boring), let's make it a little more interesting by introducing our first object.

As you might remember from the last chapter, an *object* is very similar to a database table in the Force.com platform. The platform comes with a number of standard objects, like contacts, accounts, and cases, which support default apps like Salesforce Sales and Salesforce Call Center. We can also define custom objects that allow us to store information specific to our Recruiting app.

Whether they're standard or custom, Force.com platform objects not only provide a structure for storing data but they also power the interface elements that allow users to interact with the data, such as tabs, the layout of fields on a page, and lists of related records. Because any object can correspond to a tab, and an ordered collection of tabs makes up an app, objects make up the heart of any app that we create with the platform.

With custom objects being so important—they have lots to do with how our app will look, behave, and feel—what we do with custom objects and how we use them quickly becomes essential to creating a successful app. The design of the data model behind an app is typically the biggest factor in its success or failure.

That's enough talk about objects for now. Let's go define one!

The Position Custom Object

The first custom object that we'll create for our Recruiting app reflects a typical recruiting task: describing a position. Recruiters at Universal Containers need to keep track of all the positions they're hiring for, such as a Senior Developer, Sales Engineer, or Benefits Specialist. They'll need easy access to all positions in the system through a tab, and they'll need to include certain information for each position, such as its minimum and maximum salary range, the position's location, and its hiring manager. In Force.com platform terms, we'll create a custom object, create a custom tab for that object, and then define some custom fields.

Try It Out: Defining the Position Custom Object

To create the Position object, we're going to go back to the Setup area.

1. Click *Your Name* ► Setup ► Create ► Objects.
2. On the Custom Objects page, click **New Custom Object**.

Unlike defining a custom app, which we did through the New Custom App wizard, defining a custom object is confined to just one page. You'll find that the platform uses wizards or single pages depending on the amount of information that needs to be specified.

Figure 12: Custom Object Definition Page

3. In the `Label` field, enter `Position`.
4. In the `Plural Label` field, enter `Positions`.
5. The `Object Name` field is defaulted to `Position`. Let's leave it as is.

The `Label` and `Plural Label` of a custom object are what users see in all of the object's related user interface elements, such as the object's tab or in search results headings. Object labels work best as nouns, and the plural label is always used to label a custom object's tab (if you create a tab for your object).

The value of a custom object's `Object Name` represents the unique name for the object when it's referenced in other areas of the platform, such as formulas and Visualforce pages. This value is helpfully autogenerated based on the value that you enter for the `Label`, except that all spaces and punctuation are replaced with underscore characters. We'll talk more about formulas and Visualforce later in this book. For now, just keep in mind that the `Object Name` value must be unique across all objects defined in your organization.



Note: Within the platform, `Object Name` is actually stored with `__c` appended to the end as a suffix (for example, `Position__c`). This identifies it as a custom object.

6. In the `Description` field, enter `This object stores information about the open job positions at our company.`
7. For the `Context-Sensitive Help Setting`, accept the default. If you later want to provide customized help documentation for your users about this object, you can come back and choose the `Open a window using a custom Visualforce option.`
8. In the `Record Name` field, enter `Position Title.`

The `Record Name` is the label for the field that identifies individual position records in the system. A custom object cannot be saved without this identifying field.

9. In the `Data Type` drop-down list, select `Text.`

The `Data Type` drop-down list allows you to select the type of value that should be used for this identifying field: either `Text` or `Auto-Number`. Some objects, like `Positions` or `Accounts`, can be identified with a text field because there will always be a name for a position or account available. Other objects, like a `Case` (used in the standard `Call Center` app) are harder to identify with a single text field, so we would assign them auto-numbers instead.



Tip: Whenever possible, it's best to use text as the data type for an identifying field so that users can more easily identify a particular record when several of them appear together in a single list.

To illustrate how custom object and record name labels work together in the app, let's fast forward a bit to see where each label will appear once we've defined our `Position` custom object, its tab, and a single `Sr. Developer` position record.

The screenshot shows the Salesforce user interface for a custom object named 'Positions'. At the top, there is a navigation bar with 'Home' and 'Positions' tabs. Below the navigation bar, there is a search section with a search box and a 'Go!' button. To the right of the search section, there is a 'Positions Home' section with a 'View: All' dropdown and a 'Go!' button. Below this, there is a 'Recent Positions' section with a 'New' button and a 'Recently Viewed' dropdown. The 'Recent Positions' list shows a single record with the label 'Sr. Developer'. Two red boxes with arrows point to the 'Positions' tab and the 'Sr. Developer' label, with labels 'Custom Object Plural Label' and 'Record Name Label' respectively.

Figure 13: Custom Object and Record Name Labels

Let's move on.

10. In the Optional Features area, select the `Allow Reports`, `Allow Activities`, and `Track Field History` checkboxes.

These three checkboxes actually enable some really robust functionality:

Allow Reports

Selecting this option makes the data in the position records available for reporting purposes. The platform comes with a large number of standard reports, and users can also create custom reports by using a simple yet powerful report builder. (To find out more about reports, see [Analyzing Data with Reports and Dashboards](#) on page 243.)

Allow Activities

Selecting this option allows users to associate tasks and scheduled calendar events with a particular position. For example, a user could create a task, such as “Update salary range for Sr. Developer position,” and specify attributes such as priority, due date, and status. The user could then handle the task herself or assign it to someone else. (To find out more about tasks, see “Activity Overview” in the online help.)

Track Field History

Selecting this option allows the platform to automatically track edits to position records, such as who changed the value of a field, when it was changed, and what the value of the field was before and after the edit. History data is available for reporting, so users can easily create audit trail reports when this feature is enabled. (To find out how to select which data is tracked, see “Tracking Field History” in the online help.)

In general, you should select these options if there's any chance that they might be useful for whatever custom object you're defining.

11. In the Deployment Status area, select `Deployed`.



Note: This step assumes that you're working in a development environment. If you're not, and if you don't want users to see the Position object after you click **Save**, select `In Development`. Setting the status to `In Development` hides position records from all users except those with the “Customize Application” user permission (that is, just about anyone who isn't a System Administrator).

12. In the Object Creation Options area, select the `Add Notes & Attachments` related list to default page layout and `Launch New Custom Tab Wizard` after saving this custom object checkboxes.

These two options are available only when you're creating a new custom object. If you later decide to go back and edit some of the details about your custom object, you won't see them. But what do they do?

- Enabling notes and attachments for an object means that you can attach external documents to any position record, in much the same way that you can add a PDF or photo as an attachment to an email. It's handy functionality, so you should generally select it.
- Launching the New Custom Tab wizard does exactly what it says—it's a shortcut to launching the tab wizard after we've saved our Position object and will save us a few clicks if we know that we need a tab.

All set? Let's go ahead and save our Position custom object now.

13. Click **Save**.

That's all there is to it! As promised, the New Position Tab wizard is displayed instead of the list of custom objects that we'd normally see. Let's take a moment to talk about why we should even be defining a tab for our Position object in the first place. What's so great about tabs, anyway?

Introducing Tabs

If you're familiar with the Force.com platform, you know that clicking tabs is how you navigate around an app. Every tab serves as the starting point for viewing, editing, and entering information for a particular object. When you click a tab at the top of the page, the corresponding home page for that object appears. For example, if you click the Accounts tab, the Accounts tab home page appears, giving you access to all of the account records that are defined in your organization. Click the name of a particular account record and you'll view all of the record's information in its associated detail page.

What's really powerful about building an app with the platform is that you can create custom tabs that look and behave just like the tabs for standard objects that are already provided. From the perspective of your end users, any customizations that you make appear perfectly seamless, and as a developer, you don't have to do anything special to make it work that way! Let's see how quickly we can create a tab for our Position object.

Try It Out: Defining the Positions Tab

To create a custom tab for our Position object, we're going to use the New Custom Object Tab wizard that was so helpfully launched for us when we clicked **Save** after defining the object. However, in case you forgot to select the `Launch New Custom Tab Wizard` after

saving this custom object option or if you're coming back to work that you previously saved, have no fear! There's another way to launch the wizard.

1. Click **Your Name** > **Setup** > **Create** > **Tabs**.
2. In the Custom Object tabs area, click **New**.

Easy. Now that we're all on the same page, let's get started working through the wizard.

3. In the Object drop-down list, select Position.

If you launched the wizard directly after defining the custom object, the Position object is automatically selected for you.

4. Click the Tab Style lookup icon to launch the Tab Style Selector as shown in the following screenshot.

Every object that appears as a tab must have a unique color scheme and icon. This color scheme is what identifies the object, not only on its tab but also in different places in the user interface, such as in related lists and search results.

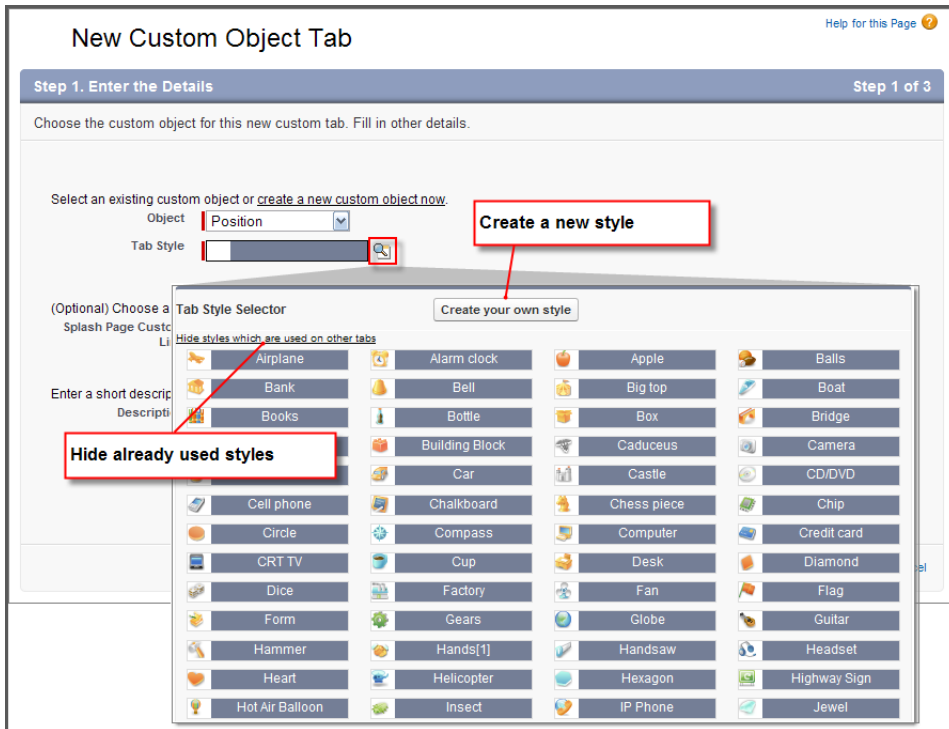


Figure 14: Custom Object Tab Setup Page and Tab Style Selector

In the Tab Style Selector, you can choose a predefined color and icon or you can create your own. To keep things simple, we're going to select an existing style.

5. Click the **Hide values which are used on other tabs** link to make sure you choose a unique style.
6. Click any colored box to choose a color scheme and icon.

Leave the `Splash Page Custom Link` drop-down list set to `--None--`. We'll learn more about custom links in [Moving Beyond Point-and-Click App Development](#) on page 281.

7. In the `Description` field, enter `A tab and color scheme for the Position custom object.`
8. Click **Next**.
9. Click **Next** again to accept the default user profile visibility.

Just as we controlled access to our Recruiting app by selecting user profiles in the New Custom App wizard, we can also control access to our Positions tab by selecting user profiles here. We'll learn more about user profiles and what they do in [Securing and Sharing Data](#) on page 131. For now, just know that accepting the defaults will make the tab visible to all users.

10. Deselect all of the `Include Tab` checkboxes except the one for our Recruiting app.

In performing this step, we're providing access to the Positions tab only when someone has access to our Recruiting app. Unless an employee is interested in recruiting, he or she probably doesn't need to see this tab.

11. Select the `Append tab to users' existing personal customizations` checkbox.

If you don't select this option, any users who have personalized their tab display will not immediately see the Positions tab. Also, if you've already created a new tab and didn't turn this option on, you have to first delete the existing tab and then recreate it with this option turned on to automatically push the tab to existing users. What a pain! Do yourself a favor and just always keep this option selected.

12. Click **Save**.

You'll notice when the page refreshes that the Positions tab has automatically been added next to the Home tab at the top of the page.

Look at What We've Done

To truly appreciate what we've just built with a few clicks, let's take a look at what we've done.

1. First, click the Positions tab to display the Positions tab home page, as shown in the following screenshot. Although the list is empty because we haven't yet created any records, you can see how this page will become the gateway to viewing, creating, editing, and deleting all of the positions that we create in our Recruiting app. It looks just like the tab home page of any other standard object.

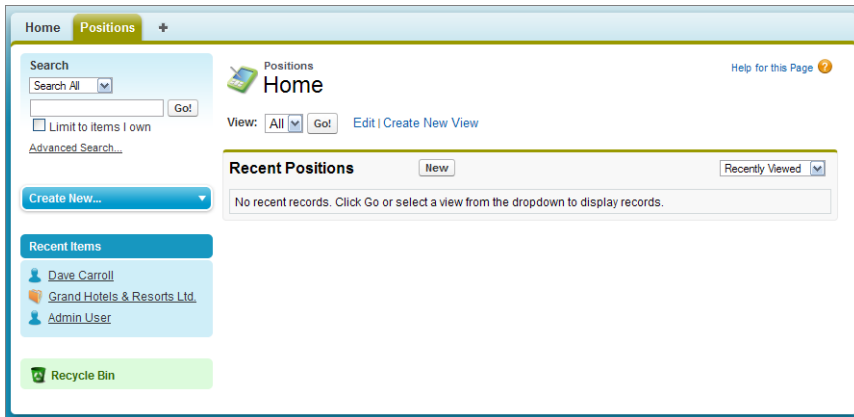


Figure 15: The Positions Tab Home Page

2. Now, check out the contents of the **Create New...** drop-down list in the left sidebar. As promised, our custom object has been seamlessly incorporated into the platform with the other standard objects like Event and Task. An end user need never know that the Positions tab was created with a custom object, because it shows up alongside the standard objects as well.
3. Select Position from the **Create New...** drop-down list, or click **New** in the Positions tab home page. Voilà—it's the Position edit page! Sadly, though, our position still doesn't have much room for data. At this point, all we have is a field for `Position Title` (the record identifier) and `Owner`, a default field that appears on every object to identify the user who created the object.
4. Click **Cancel**. It doesn't do much to create a position record with hardly any interesting data. We need more fields! And sure enough, that's what we'll get to next. First, though, let's revisit our Position custom object and orient ourselves to what else is available through a custom object detail page in the Setup area.

Becoming Familiar with Setup Detail Pages and Related Lists

You may recall when we first introduced the concept of objects that we learned: “Whether they're standard or custom, Force.com platform objects not only provide a structure for storing data but they also power the interface elements that allow users to interact with the data, such as tabs, the layout of fields on a page, and lists of related records.” If you've been following along closely, you might have been wondering why we didn't get to define any fields (other than the identifier field of Position Title) or user interface elements (other than the Positions tab) when we created our Position object. If fields and user interface elements are a part of the definition of what a custom object is all about, where do we get to define them?

It turns out that the Force.com platform differentiates between the initial creation of certain components and details related to those components. In other words, the information that we see when we define or edit a custom object is different from the information that we see when we view a custom object that's already defined. Let's go back to our custom object list page to see how this difference is reflected in the platform interface:

1. Click *Your Name* ► Setup ► Create ► Objects.

Here we are back in the custom object list page. You'll notice in the row for Position there are three links that we can click:

Edit

This link takes us back to the Custom Object edit page where we originally defined our Position object.

Del

This link deletes the custom object, including any records, tabs, reports, or other components associated with that object.

Position

This link takes us to the custom object detail page for our Position object.

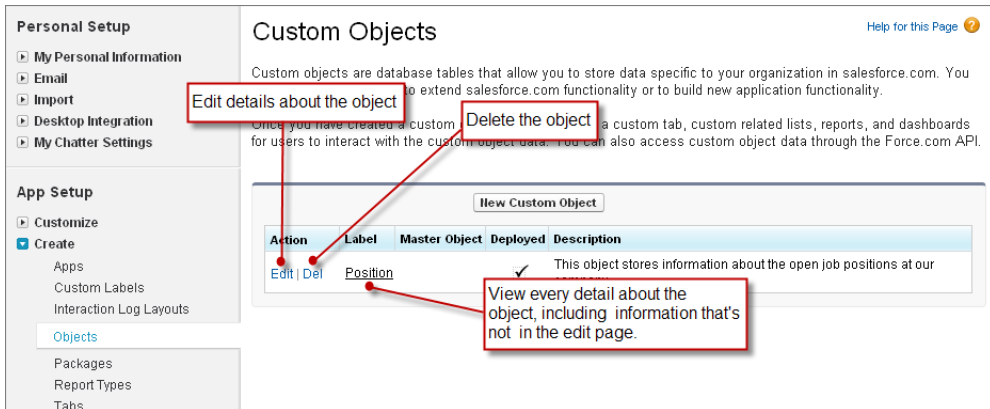


Figure 16: Custom Object List Page: Edit, Delete, and Detail Links

We're already familiar with the edit page from when we defined our Position object, and we certainly don't want to delete our object. Let's go ahead and open up the detail page to see what we can do there.

2. Click Position.

As you can see, the Custom Object edit page that we filled out when we defined our Position object was just the tip of the iceberg. The top two areas of the Position detail page (see Position Custom Object Detail Page) include all of the information that we originally specified, plus a few standard fields that the platform includes with every object. Below those areas are several additional groupings of data that allow us to do more with our Position object.

In Force.com platform terms, those groupings of data are called *related lists*, and they're a big part of what makes the platform so powerful. A related list is a list of records or other components that are associated with whatever we're viewing. Related lists appear in both the main application and in the Setup areas and represent a relationship between the items that appear in the related list and the object or record that we're viewing in the detail area. We'll learn a lot more about relationships in [Expanding the Simple App Using Relationships](#) on page 85, but for now, just understand that anything that appears in an object's related list is directly related to that object.

Now that we've found out where we can continue customizing our Position custom object, let's use the Custom Fields & Relationships related lists to create some more fields in our Position object.

Custom Object **Position** [Help for this Page](#)

[Standard Fields \[4\]](#) | [Custom Fields & Relationships \[0\]](#) | [Validation Rules \[0\]](#) | [Standard Buttons and Links \[8\]](#) | [Custom Buttons and Links \[0\]](#) | [Page Layouts \[1\]](#) | [Search Layouts \[8\]](#) | [Record Types \[0\]](#) | [Apex Sharing Reasons \[0\]](#) | [Apex Sharing Recalculation \[0\]](#)

Custom Object Definition Detail [Edit](#) [Delete](#) **Definition Details**

Singular Label	Position	Description	This object stores information about the open job positions at our company.
Plural Label	Positions	Enable Reports	<input checked="" type="checkbox"/>
Object Name	Position	Track Activities	<input checked="" type="checkbox"/>
API Name	Position__c	Track Field History	<input checked="" type="checkbox"/>
		Deployment Status	Deployed
		Help Settings	Standard salesforce.com Help Window
Created By	Admin User, 2/2/2010 2:07 PM	Modified By	Admin User, 2/2/2010 2:07 PM

Standard Fields [Standard Fields Help](#)

Action	Field Label	Data Type	Track History
	Created By	Lookup(User)	<input type="checkbox"/>
	Last Modified By	Lookup(User)	<input type="checkbox"/>
Edit	Owner	Lookup(User,Queue)	<input type="checkbox"/>
Edit	Position Title	Text(80)	<input type="checkbox"/>

Custom Fields & Relationships [New](#) [Field Dependencies](#) [Set History Tracking](#) [Custom Fields & Relationships Help](#)

No custom fields defined

Figure 17: Position Custom Object Detail Page

Introducing Fields

We're ready to add more fields to our Position custom object, but first, let's talk briefly about what a field is and how it fits into the world of the Force.com platform.

As you might remember from the last chapter, a *field* is like a database column. The primary characteristic of a field is its data type—some fields hold text values, while others hold currency values, percentages, phone numbers, email addresses, or dates. Some fields look like checkboxes, while still others are drop-down lists or record lookups from which a user makes a selection.

The data type of a field controls the way the field is ultimately displayed in the user interface and how data entered into the field is stored in the platform. To get a better feel for how the fields will look, let's take a sneak peak at what the Position object is ultimately going to look like and the types of custom fields we're going to create for it:

The screenshot shows the 'New Position' form in Salesforce. The form is divided into several sections, each with a light green header. The 'Information' section contains fields for Position Title, Status, Type, Functional Area, Job Level, Travel Required, Owner, Admin User, Location, Open Date, Hire By, and Close Date. The 'Compensation' section contains Min Pay and Max Pay. The 'Description' section contains Job Description, Responsibilities, Skills Required, and Educational Requirements. The 'Required Languages' section contains checkboxes for Java, JavaScript, C#, and Apex. Red boxes and arrows highlight specific fields with labels: 'Text field' (Position Title), 'Picklist field' (Status), 'Dependent picklist field' (Functional Area), 'Standard owner field' (Owner), 'Date field' (Open Date), 'Checkbox field' (Travel Required), 'Currency field' (Min Pay), and 'Long text area field' (Job Description).

Figure 18: Position Custom Object Fields

There are a lot of fields here that we need to define, some more complicated than others. To keep things simple, let's go through and create the simple text, currency, checkbox, and date fields. We can tackle the more complicated picklist and custom formula fields in [Enhancing the Simple App with Advanced Fields, Data Validation, and Page Layouts](#) on page 53.

Try It Out: Adding Text Fields

First let's define a few text fields. We already created a basic text field for `Position Title` when we defined our `Position` custom object. Looking at our screenshot, the only text fields that remain are the text fields under the `Description` heading. We'll start by defining the `Job Description` field.

1. Click *Your Name* ► Setup ► Create ► Objects.

2. Click **Position**.
3. In the Custom Fields & Relationships related list, click **New**.

Every time you create a custom field, you'll first choose a data type from the field type selection page.

The platform allows us to choose between different types of text fields:

- Basic text fields allow users to enter any combination of letters and numbers on a single line, up to as many as 255 characters.
- Text area fields also have a 255 character limit but also allow carriage returns so the text can be formatted on separate lines.
- Long text fields allow as many as 32,000 characters, on separate lines.

Since job descriptions can be lengthy, let's choose a long text area.

4. Choose the `Text Area (Long)` data type and click **Next**.



Tip: Carefully consider the data type you choose for each of your custom fields, because once you set it, it isn't always the best idea to change it later. See [Notes on Changing Custom Field Types](#) in the online help for details.

The second page of the custom field wizard allows us to enter details about our long text area field. The fields that appear in this step change depending on the data type that we selected in the previous page.

5. In the `Field Label` field, enter `Job Description`.

Like the other labels that we've seen in the platform so far, the `Field Label` specifies the text that should be displayed to users when the field is rendered in the user interface. Notice that when we enter a value for `Field Label`, `Field Name` is automatically populated with the same text but with all spaces and punctuation replaced by underscores. The value for `Field Name` is a unique name that is used to refer to the field when writing a custom formula or using the API.



Note: Within the platform, `Field Name` is actually stored with `__c` appended to the end as a suffix (for example, `Job_Description__c`). This identifies it as a custom field.

6. In the `Length` field, enter `32,000`.

The `Length` field allows us to restrict the maximum number of characters that are allowed. Since we don't get any benefit from this kind of restriction, leave this value set to 32,000.

7. In the `# Visible Lines` field, enter `3`.

This field allows us to specify how large our text box will appear on the page.

8. In the `Description` and `Help Text` fields, enter High-level description of the job and its duties.

This help text is displayed on record detail and edit pages when users hover over the field's label. Its purpose is to assist users in filling out the field correctly. It's optional to add help text for a field, but it's a good idea if you have fields that you think users might be confused by.

There's no obvious default value for a text field, so just leave `Default Value` blank.

9. Click **Next**.

The third page of the Custom Field wizard allows us to restrict access to this field from certain user profiles. We'll learn more about profiles and field-level security in [Securing and Sharing Data](#) on page 131, so for now, just accept the defaults.

10. Click **Next**.

The last page of the wizard allows us to automatically place our field on the Position page layout. Again, we'll learn about page layouts in the next chapter, so for now, just accept the defaults.

11. Click **Save & New**.

Instead of clicking **Save** and returning to the Position object detail page, clicking **Save & New** saves a few clicks and allows us to finish up the other text area fields that we need. Here's what you need to know to define them:

Table 2: Position Object Long Text Area Fields

Field Type	Field Label	Length	# Visible Lines	Default Value
Text Area (Long)	Responsibilities	32,000	3	Leave unspecified
Text Area (Long)	Skills Required	32,000	3	Leave unspecified
Text Area (Long)	Educational Requirements	32,000	3	Leave unspecified

Now that we've wet our feet with text fields, let's quickly create a few more fields of other types. You'll find that with few exceptions, they're all very similar to one another.

Try It Out: Adding Currency Fields

To keep track of a position's salary range, we need to add two currency fields: `Min Pay` and `Max Pay`. Note that unlike some fields, once we define these as currency fields, we won't be able to change them to any other type.

Defining a currency field is almost identical to defining a text field, with a few slight differences:

- The `Length` of a currency field actually corresponds to the number of digits to the left of the decimal point. An additional `Decimal Places` field handles the number of digits that should be displayed to the right.
- In the `Details` page of the wizard, a new checkbox called `Required` is displayed. We can select this option if we want to force our users to enter a value for this field when creating a new position.

Everything else should be familiar to you, so go ahead and use the custom field wizard to define the following fields:

Table 3: Position Object Currency Fields

Field Type	Field Label	Length	Decimal Places	Required	Default Value
Currency	Min Pay	7	2	Leave unchecked	Leave unspecified
Currency	Max Pay	7	2	Leave unchecked	Leave unspecified

Try It Out: Adding Checkbox Fields

Here are some easy ones. The `Position` object requires a few checkbox fields: one to indicate if travel is required for the position, and four others to indicate which programming languages are required. By default, these values should be unchecked. (Note that similar to currency fields, once you define a field as a checkbox, you can't change it to any other type.)

Use the custom field wizard to define these fields:

Table 4: Position Object Checkbox Fields

Field Type	Field Label	Default Value
Checkbox	Travel Required	Unchecked
Checkbox	Java	Unchecked
Checkbox	JavaScript	Unchecked
Checkbox	C#	Unchecked
Checkbox	Apex	Unchecked

Try It Out: Adding Date Fields

Finally, before we close out this chapter, let's add three date fields to our Recruiting app to track the date a position opens, the date it closes, and the date by which it should be filled. Date fields are great because they automatically include a popup calendar interface from which users can select a day without any typing—yet another built-in feature that we can leverage in our app without any extra work!

Once again we'll use the custom field wizard to define these three fields:

Table 5: Position Object Date Fields

Field Type	Field Label	Required	Default Value
Date	Open Date	Unchecked	Leave unspecified
Date	Hire By	Unchecked	Leave unspecified
Date	Close Date	Unchecked	Leave unspecified

Look at What We've Done

We've defined text, currency, checkbox, and date fields for our Position object. Let's take a look by going to the Positions tab and clicking **New**.

Figure 19: Position Object Fields

Check out all the fields we've just made! It's not the most elegant layout for all of our fields (each field got added to the page in the order that we created it), but it's definitely functional, and it looks just like any other page. Wasn't that easy?

Once again, welcome to the power of the Force.com platform. First we created a new recruiting app with a single Home tab, then we created a Position object and tab, and now we've just added a few fields, all in less than 15 minutes of clicking around. From start to finish we always had a fully functional app, and we never had to spend any time compiling or debugging our “code!”

In the next chapter, we'll enhance our simple Recruiting app even further by adding some additional fields that are more complex, defining validation rules to help our data stay clean, and then moving the fields around in a page layout so users can more easily find and enter the information they need. Let's keep going!

Chapter 6

Enhancing the Simple App with Advanced Fields, Data Validation, and Page Layouts

In this chapter ...

- [Adding Advanced Fields](#)
- [Introducing Validation Rules](#)
- [Introducing Page Layouts](#)

In the last chapter, we got our Recruiting app off to a quick start by defining the Position custom object, tab, and several simple fields. This simple version of our app had the same look and feel as any other page on the Force.com platform, and we were able to whip it together in a matter of minutes.

In this chapter, we're going to enhance the Positions tab: first by defining a few more advanced fields, then by defining a validation rule to make sure our data stays clean, and finally by moving our fields around within a page layout. These additions will help change the detail page of our Positions tab from a somewhat flat and inelegant user interface to something that users find powerful and intuitive to use. Let's get started!

Adding Advanced Fields

In this section, let's revisit the custom field wizard to help us create fields with more sophisticated functionality: picklists, dependent picklists, and fields that leverage custom formulas. We'll see how the platform's user interface helps guide us through the setup of these more complicated fields.

Introducing Picklists

When viewing the preview of what we wanted our Positions page to ultimately look like, there were several fields that were specified with drop-down lists. In Force.com platform terms, these fields are called *picklists*, and they consist of several predefined options from which a user can select.

Picklists come in two flavors: a standard picklist, in which a user can select only one option, and a multi-select picklist, in which a user can select multiple options at a time. For the purposes of our Position object, we need to define standard picklists for a position's location, status, type of job, functional area, and job level.

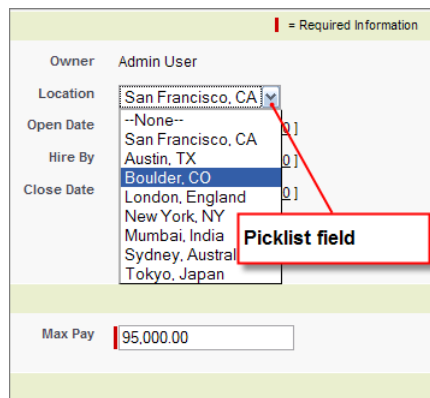


Figure 20: Location Picklist Field

Try It Out: Adding Picklists

Let's walk through the creation of the `Location` picklist field. Then, as in the previous chapter, we'll give you the information that you need to create the others on your own.

1. Click *Your Name* ► Setup ► Create ► Objects.

2. Click **Position**.
3. In the Custom Fields & Relationships related list, click **New**.
4. Select the `Picklist` data type and click **Next**.
5. In the `Field Label` text box, enter `Location`.
6. In the large text area box just below, enter the following picklist values, each on its own line:
 - San Francisco, CA
 - Austin, TX
 - Boulder, CO
 - London, England
 - New York, NY
 - Mumbai, India
 - Sydney, Australia
 - Tokyo, Japan
7. Select the `Use first value as default value` checkbox.

This option allows us to populate the field with a default value. If you leave it deselected, the field defaults to `None` on all new position records. Otherwise the field defaults to the first value that you specify in the list of possible picklist values. Because most positions at Universal Containers are based at its headquarters in San Francisco, CA, this should be the default.

8. Accept all other default settings for field-level security and page layouts.
9. Click **Save & New**.

Easy! Now specify the remaining picklists according to the table below:

Table 6: Status, Type, Functional Area, and Job Level Picklist Values

Data Type	Field Label	Picklist Values	Sort Alphabetically?	Use First Value as Default?
Picklist	Status	New Position Pending Approval Open - Approved Closed - Filled Closed - Not Approved	No	Yes

Data Type	Field Label	Picklist Values	Sort Alphabetically?	Use First Value as Default?
		Closed - Canceled		
Picklist	Type	Full Time Part Time Internship Contractor	No	No
Picklist	Functional Area	Finance Human Resources Information Technology Retail Operations Warehousing Miscellaneous	Yes	No
Picklist	Job Level	FN-100 FN-200 FN-300 FN-400 HR-100 HR-200 HR-300 HR-400 IT-100 IT-200	Yes	No

Data Type	Field Label	Picklist Values	Sort Alphabetically?	Use First Value as Default?
		IT-300		
		IT-400		
		RO-100		
		RO-200		
		RO-300		
		RO-400		
		WH-100		
		WH-200		
		WH-300		
		WH-400		
		MC-100		
		MC-200		
		MC-300		
		MC-400		

Introducing Field Dependencies

Now that we've made all those picklists, answer this question: How many times have you clicked on a drop-down list and found far too many values to choose from? For example, maybe you were selecting Uruguay from a list of countries, and every country in the world was on the list. That meant that you had to scroll all the way down to the countries that started with the letter U. What a pain!

Fortunately, the folks who built the Force.com platform have encountered that situation a few times themselves, and as a result, they've given us a tool to help us avoid this problem with our own picklist fields: *field dependencies*.

Field dependencies are filters that allow us to change the contents of a picklist based on the value of another field. For example, rather than displaying every value for Country in a single picklist, we can limit the values that are displayed based on a value for another field, like Continent. That way our users can find the appropriate country more quickly and easily.

Picklist fields can be either *controlling* or *dependent* fields. A controlling field controls the available values in one or more corresponding dependent fields. A dependent field displays values based on the value selected in its corresponding controlling field. In the previous example, the Continent picklist is the controlling field, while the Country picklist is the dependent field.

Try It Out: Creating a Dependent Picklist

Looking at the picklists that we've created, it's quickly obvious that our users might get frustrated with the length of our Job Level picklist. Let's make our users happy by turning Job Level into a dependent field of the Functional Area picklist. Doing this will allow users to see only the four relevant job level values when a department is selected in the Functional Area picklist:

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In the Custom Fields & Relationships related list, click **Field Dependencies**.
4. Click **New**.
5. For the Controlling Field drop-down list, choose Functional Area.
6. For the Dependent Field drop-down list, choose Job Level.
7. Click **Continue**.

A field dependency matrix displays with all the values in the controlling field across the top header row and the dependent field values listed in the columns below. For each possible value of the controlling field, we need to include the values that should be displayed in the dependent picklist when that controlling value is selected. In the field dependency matrix, yellow highlighting shows which dependent field values are included in the picklist for a particular controlling field value.

Click button to include or exclude selected values from the dependent picklist:

Showing Columns: 1 - 5 (of 6) < Previous **Next** > View All

Functional Area:	Finance	Human Resources	Information Technology	Miscellaneous	Retail Operations
Job Level:	FN-100	FN-100	FN-100	FN-100	FN-100
	FN-200	FN-200	FN-200	FN-200	FN-200
	FN-300	FN-300	FN-300	FN-300	FN-300
	FN-400	FN-400	FN-400	FN-400	FN-400
	HR-100	HR-100	HR-100	HR-100	HR-100
	HR-200	HR-200	HR-200	HR-200	HR-200
	HR-300	HR-300	HR-300	HR-300	HR-300
	HR-400	HR-400	HR-400	HR-400	HR-400
	IT-100	IT-100	IT-100	IT-100	IT-100
	IT-200	IT-200	IT-200	IT-200	IT-200
	IT-300	IT-300	IT-300	IT-300	IT-300
	IT-400	IT-400	IT-400	IT-400	IT-400
	MC-100	MC-100	MC-100	MC-100	MC-100
	MC-200	MC-200	MC-200	MC-200	MC-200
	MC-300	MC-300	MC-300	MC-300	MC-300
	MC-400	MC-400	MC-400	MC-400	MC-400
	RO-100	RO-100	RO-100	RO-100	RO-100
	RO-200	RO-200	RO-200	RO-200	RO-200
	RO-300	RO-300	RO-300	RO-300	RO-300
	RO-400	RO-400	RO-400	RO-400	RO-400
Functional Area:	Finance	Human Resources	Information Technology	Miscellaneous	Retail Operations
Job Level:	WH-100	WH-100	WH-100	WH-100	WH-100
	WH-200	WH-200	WH-200	WH-200	WH-200

Click Next to see more columns of controlling field values.

Blue highlighting indicates that you've selected these dependent field values for this controlling field value. Click the Include Values button to add them.

Column headings are repeated so they're visible when you scroll down the matrix

Figure 21: Field Dependency Matrix

To include a dependent field value, you simply double-click it. To exclude a dependent value from the list, double-click it again.

For example, let's try it out by including the values that should be displayed in the Job Level picklist whenever Finance is selected in the Functional Area picklist:

8. In the column labeled Finance, double-click FN-100, FN-200, FN-300, and FN-400.

Those four fields should now be shaded yellow in the Finance column.

Instead of double-clicking every Job Level value, we can also use SHIFT+click to select a range of values or CTRL+click to select multiple values at once. Once those values are highlighted in blue, we can click **Include Values** to include them, or **Exclude Values** to remove them. Let's try it out.

9. In the column labeled Human Resources, single-click HR-100 and then press and hold the SHIFT key while clicking HR-400.
10. Click **Include Values**.

Now we have values selected for both the Finance and Human Resources columns!

11. Continue highlighting the appropriate values for all of the remaining columns, as described in the following table.



Tip: To get to all of the values that you need to modify for this step, you'll need to click **Previous** or **Next** to see additional columns.

Table 7: Functional Area and Job Level Field Dependency Matrix

Functional Area (Controlling picklist field)	Job Level (Dependent picklist field)
Finance	FN-100
	FN-200
	FN-300
	FN-400
Human Resources	HR-100
	HR-200
	HR-300
	HR-400
Information Technology	IT-100
	IT-200
	IT-300
	IT-400
Retail Operations	RO-100
	RO-200
	RO-300
	RO-400

Functional Area (Controlling picklist field)	Job Level (Dependent picklist field)
Warehousing	WH-100
	WH-200
	WH-300
	WH-400
Miscellaneous	MC-100
	MC-200
	MC-300
	MC-400

12. Click **Preview** to test the results in a small popup window.
13. Click **Save**.

Look at What We've Done

Now that we've created all those picklists, let's revisit the Positions tab to see what we have so far.

1. Go to the Positions tab.
2. Click **New**.
3. In the Functional Area picklist, select Finance.
4. Open the Job Level picklist.

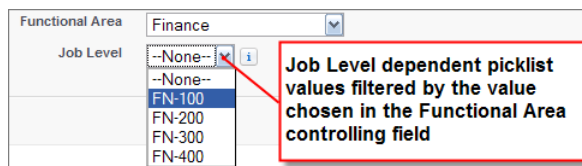


Figure 22: Dependent Picklist Fields

Our Recruiting app users are going to be very happy that they no longer have to deal with a long, onerous picklist. Now let's go add a field that's even more powerful and complex than a dependent picklist: a custom formula field.

Introducing Custom Formula Fields

Up to this point, the fields that we've defined have all had one thing in common—they each require a user to give them a value. Fields like that are very helpful for storing and retrieving data, but wouldn't it be great if we could somehow define a “smart” field? That is, what if we could define a field that looked at information that was already entered into the system and then told us something new about it?

Fortunately, *custom formula fields* give us the ability to do just that. Just as you can use a spreadsheet program like Microsoft Excel to define calculations and metrics specific to your business, we can use custom formula fields to define calculations and metrics that are specific to our Recruiting app.

For example, on our Position object, we've already created fields for minimum pay and maximum pay. If Universal Containers gives out yearly bonuses based on salary, we could create a custom formula field that automatically calculated the average bonus that someone hired to that position might receive.

How would we perform this calculation if we were using a spreadsheet? The columns in our spreadsheet would represent the fields that we defined on our Position object, and each row of the spreadsheet would represent a different position record. To create a calculation, we'd enter a formula in a new column that averages the values of `Min Pay` and `Max Pay` in a single row and then multiplies it by a standard bonus percentage. We could then determine the average bonus for every position record row in our spreadsheet.

Custom formulas work in a very similar way. Think of a custom formula like a spreadsheet formula that can reference other values in the same data record, perform calculations on them, and return a result. However, instead of using cell references, you use *merge fields*, which serve as placeholders for data that will be replaced with information from your records, user information, or company information. And, instead of typing fields, operators, and functions, you can click to select them.

The net result is that anyone can quickly and easily learn to create formula fields. And, as with all platform tools, the cloud computing delivery model makes it easy to experiment. You can create a formula, view the results, and change the formula again and again, as many times as you want! Your underlying data is never affected.



Tip: When defining your own custom formula fields, leverage the work of others. You can find more than a hundred sample formulas on the Salesforce.com Community website at <http://www.salesforce.com/us/developer/docs/usefulFormulaFields/>.

Calculating How Long a Position Has Been Open

Let's now think about another custom formula field that we could create for our Position object—a custom formula field that calculates how many days a position has been open. To do this, let's first think about the logic that we should use to define the field, and then we can go through the process of creating it in our Recruiting app.

Let's think about the data that we need to make this calculation: we need to know the current date and the date that the position was created. If we could somehow subtract these two, we'd have the number of days that the position has been open. Fortunately, it's easy to get both of these values:

- For the current date, we can simply use the platform's built-in `TODAY ()` function. `TODAY ()` returns today's date.
- For the date that the position was opened, we can use the `Open Date` field that we defined in the last chapter.

When using fields in formulas, you can't just refer to a field by its name. Instead, you need to refer to it by its merge field name, also called its API name. The format of the API name is typically the name of the field but with spaces instead of underscores. For custom fields, the API name is suffixed with two underscores and the letter “c,” like this: `Open_Date__c`. This naming convention in the platform helps distinguish between standard and custom fields.



Tip: You don't need memorize the API names of fields you want to use in formulas. Simply use the field picker in the formula editor to insert fields and the platform automatically inserts the API name for you. If you ever want to know the API name of a specific field and you aren't using the formula editor, you can view the field's detail page.

Now that we have our two dates, we want to subtract them: `TODAY () - Open_Date__c`. Even if the two dates span different months or years, the platform is sophisticated enough to know how to handle all the intricacies of such a calculation behind the scenes. We just have to provide the dates, and the platform can do all the rest.

So far so good, but one problem still remains—what if the position has already closed? Our formula only works if we assume the position is still open. Once it closes, however, the value of our current formula will keep incrementing every day as `TODAY ()` gets farther and farther

away from the original `Open Date`. If we can, we want to use the `Close Date` field in the formula instead of `TODAY()` after a position closes. How can we do this?

Once again, all we need to do is dip into the extensive library of platform functions. The `IF()` function allows us to perform a test and then return different values depending on whether the result of the test is true or false. The `IF()` function's syntax looks like this:

```
IF(logical_test,
    value_if_true,
    value_if_false)
```

For the *logical_test* portion, we'll test whether the `Close Date` field has a value—if it does, the position obviously must be closed. We'll test for this with a third built-in function: `ISBLANK()`. `ISBLANK()` takes a single field and returns true if it does not contain a value and false if it does. So now our formula looks like this:

```
IF( ISBLANK( Close_Date__c ) ,
    value_if_true,
    value_if_false)
```

By replacing *value_if_true* and *value_if_false* with the other formulas we talked about, we've now figured out our whole formula:

```
IF( ISBLANK( Close_Date__c ) ,
    TODAY() - Open_Date__c ,
    Close_Date__c - Open_Date__c )
```

Great! Our formula calculates the number of days a position has been open, regardless of whether it's currently open or closed. Now, let's go define a field for it on our `Position` object.

Try It Out: Defining a “Days Open” Custom Formula Field

We'll begin building the formula field the same way we created our other custom fields.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In the Custom Fields & Relationships related list, click **New**.
4. Select the **Formula** data type, and click **Next**.

Step 2 of the New Custom Field wizard appears.

Position Help for this Page

New Custom Field

Step 2. Choose output type Step 2 of 5

[Previous](#) [Next](#) [Cancel](#)

Field Label Field Name

Formula Return Type

Select one of the data types below.

- None Selected
- Currency

Calculate a dollar or other currency amount and automatically format the field as a currency amount.
Example: `Gross Margin = Amount - Cost_c`
- Date

Calculate a date, for example, by adding or subtracting days to other dates.
Example: `Reminder Date = CloseDate - 7`
- DateTime

Calculate a dateTime, for example, by adding a number of hours or days to another dateTime.
Example: `Next = NOW() + 1`
- Number

Calculate a numeric value.
Example: `Fahrenheit = 1.8 * Celsius_c + 32`
- Percent

Calculate a percent and automatically add the percent sign to the number.
Example: `Discount = (Amount - Discounted_Amount_c) / Amount`
- Text

Create a text string, for example, by concatenating other text fields.
Example: `Full Name = LastName & ", " & FirstName`

Options **Decimal Places** **Example:** 999

[Previous](#) [Next](#) [Cancel](#)

Figure 23: Custom Formula Field Wizard Step 2

5. In the `Field Label` field, enter `Days Open`.
6. Select the `Number` formula return type.

In this case, even though we're subtracting `Date` fields, we want to end up with just a regular numeric value.

7. Change the `Decimal Places` value to 0, and click **Next**.

Now it's time to enter the details of our formula.

8. Click the `Advanced Formula` tab, as shown in the following screenshot.

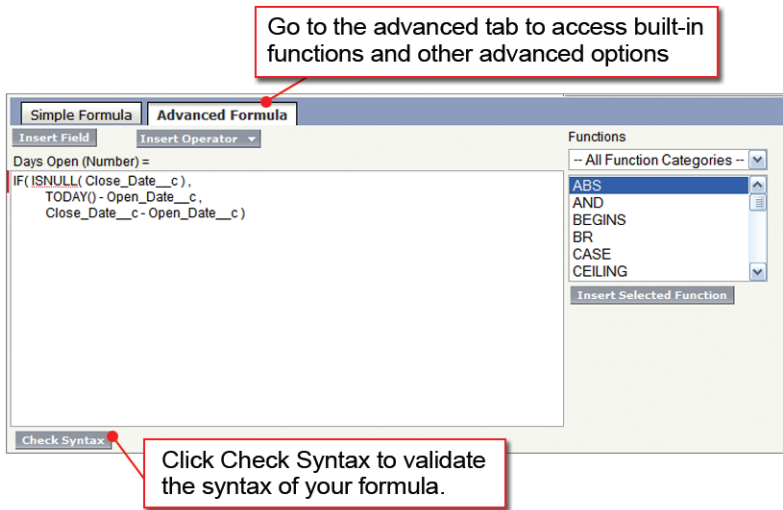


Figure 24: Custom Formula Field Editor

We want to use the Advanced Formula tab so we can access the platform's built-in functions through the Functions list on the right side.

9. From the `Functions` list, double-click `IF`.

Our formula now looks like this:

```
IF(logical_test, value_if_true, value_if_false)
```

Let's go ahead and define the logical test:

10. Delete `logical_test` from the formula, but leave your cursor there.
11. From the `Functions` list, double-click `ISBLANK`.
12. Delete `expression` from the `ISBLANK` function you just inserted, but leave your cursor there.
13. Click the **Insert Field** button. Two columns appear in an overlay.
14. In the left column, select `Position`.
15. In the right column, select `Close Date`.
16. Click **Insert**.

Did you notice that you didn't have to remember to use the API name of the `Close Date` field? The platform remembered for you when it inserted the value. Our formula now looks like this:

```
IF( ISBLANK( Close_Date__c ) , value_if_true, value_if_false)
```

Now, let's specify the value if our logical test evaluates to true:

17. Delete `value_if_true` from the formula, but leave your cursor there.
18. Press Enter on your keyboard, and space over 10 spaces.

Adding the carriage return and spaces makes our formula more legible for others.

19. From the `Functions` list, double-click `TODAY`.
20. Click the **Insert Operator** button and choose Subtract.
21. Click the **Insert Field** button.
22. In the left column, select `Position`.
23. In the right column, select `Open Date`.
24. Click **Insert**.

We're getting closer—our formula now looks like this:

```
IF( ISBLANK( Close_Date__c ) ,
    TODAY() - Open_Date__c , value_if_false)
```

Finally, let's specify the value if our logical test evaluates to false:

25. Delete `value_if_false` from the formula, but leave your cursor there.
26. Press Enter on your keyboard, and space over 10 spaces.
27. Click the **Insert Field** button.
28. In the left column, select `Position`.
29. In the right column, select `Close Date` and click **Insert**.
30. Click **Insert Operator** and choose Subtract.
31. Click the **Insert Field** button.
32. In the left column, select `Position`.
33. In the right column, select `Open Date` and click **Insert**.

Our formula now matches our original:

```
IF( ISBLANK( Close_Date__c ) ,
    TODAY() - Open_Date__c ,
    Close_Date__c - Open_Date__c )
```

Now that we've gone through those steps of the procedure, note that we could have just typed in the formula that we figured out in the last section. However, using the formula editor is a lot easier because you don't have to remember function syntax or API names of fields and objects. Let's keep going and finish up this field:

34. Click **Check Syntax** to check your formula for errors.

35. In the `Description` text box, enter `The number of days a position has been (or was) open.`
36. Add an optional `Help Text` description if you wish.
37. Select `Treat blank fields as blanks`, and click **Next**.
38. Accept all remaining field-level security and page layout defaults.
39. Click **Save**.

Try It Out: Giving Fields Dynamic Default Values

We can also use custom formulas to give our fields dynamic default values. While some fields like the `Travel Required` checkbox or the `Job Location` picklist have default values that apply in every situation, there are other fields with defaults that can't be so easily defined. For example, at Universal Containers, recruiters are generally expected to fill a position within 90 days of it being opened. While we can't choose a single date that will always be 90 days after a position is opened, we *can* define a custom formula that takes the date the position is created and adds 90 days. The platform allows us to specify this formula as the `Hire By` field's default value:

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In the Custom Fields & Relationships related list, click **Edit** next to the `Hire By` field.
4. Next to the `Default Value` text box, click **Show Formula Editor**.

Look familiar? This is similar to the editor that we used to define our `Days Open` custom formula field.

5. From the `Functions` list, double-click `TODAY`.
6. Click the **Insert Operator** button, and choose `Add`.
7. Type `90`.

Your default value formula should be:

```
TODAY () + 90
```

8. Click **Save**.

It's that easy! Now to wrap up the fields on our `Positions` tab, let's set the default value of the `Open Date` field to the day that the record is created. To do this, follow these steps again, but use `TODAY ()` as the `Default Value`.

Look at What We've Done

Let's revisit the Positions tab to take a look at what we've just done.

1. Click the Positions tab.
2. Click **New**.

Our `Days Open` formula field doesn't show up on our Position edit page—that's because it's a formula field and doesn't require any user input to display. However, we can see that our `Open Date` and `Hire By` fields already have default values: `Open Date` should be today's date, and `Hire By` is 90 days later. We can change these values if we want, or we can just leave them as they are.

In order to see our `Days Open` field, we'll have to define our first position record. Let's do that now.

3. Enter any values you want to define for a new position. At the very least, you must enter a value for the required `Position Title` field.
4. Click **Save**.

The new position is now displayed in its own record detail page. At the bottom of the page, notice our `Days Open` formula field, just above the `Created By` field. It should show 0, since we just created the position. If you want to see the value change, edit the record and set the `Open Date` to a week earlier. Isn't that neat?

Introducing Validation Rules

Now that we've defined all the fields that we want on our Position object, let's see if we can articulate a couple of rules about the data that should be entered into those fields. Even though the recruiters and hiring managers at Universal Containers are bright people, everyone sometimes makes mistakes when filling out a form, and a good app should catch the obvious errors.

For example, does it ever make sense for the value of the `Min Pay` field to be more than the value of the `Max Pay` field? Or should `Close Date` ever be unspecified if the `Status` field is set to `Closed - Filled` or `Closed - Not Approved`? Clearly not. We can catch these sorts of errors in our app with yet another built-in feature of the platform: *validation rules*.

Validation rules verify that the data a user enters in your app meets the standards that you specify. If it doesn't, the validation rule prevents the record from being saved, and the user sees

an error message that you define either next to the problematic field or at the top of the edit page. Let's build a couple of validation rules now for our Recruiting app.



Note: You can find dozens of sample validation rules on the Salesforce.com Community website at www.salesforce.com/community.

Try It Out: Defining a Validation Rule for Min and Max Pay

For our first validation rule, let's start off simple: `Min Pay` should never be greater than `Max Pay`:

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In the Validation Rules related list, click **New**.

Validation Rule Edit [Save] [Save & New] [Cancel]

Rule Name:

Active:

Description: Min Pay should never exceed Max Pay.

Quick Tips

- [Getting Started](#)
- [Resources on successforce.com](#)
- [Operators & Functions](#)

Error Condition Formula

Example: [More Examples...](#)

Display an error if Discount is more than 30%

If this formula expression is true, display the text defined in the Error Message area.

Insert Field: Insert Operator:

Check Syntax

ABS

AND
BEGINS
BR
CASE
CEILING

Insert Selected Function

ABS(number)
Returns the absolute value of a number, a number without its sign

[Help on this function](#)

Error Message

Example:

This message will appear when Error Condition formula is true

Error Message:

This error message can either appear at the top of the page or below a specific field on the page

Error Location: Top of Page Field: [i](#)

[Save] [Save & New] [Cancel]

Figure 25: Validation Rule Edit Page

4. In the Rule Name text box, enter `Min_Pay_Rule`.

The name of a validation rule can't include any spaces, but if you forget, the platform helpfully changes them to underscores (`_`) for you.

5. Select the `Active` checkbox.

This checkbox specifies whether the validation rule should start working as soon as it's saved. Because this rule is pretty straightforward (and because we want to test it later!), it makes sense to turn it on right away.

6. In the `Description` text box, enter `Min Pay should never exceed Max Pay`.

Now it's time to define the meat of our validation rule: the *error condition*. If you have a sense of déjà vu when looking at the Error Condition Formula area of the page, don't be alarmed! Just like formula fields and default field values, a validation rule can leverage a number of built-in operators and functions to define a true-or-false error condition that determines whether data is valid. When this condition evaluates to true, an error message displays and the record can't be saved.

We want our error condition to be true whenever `Min Pay` is greater than `Max Pay`, so let's use our formula editor to specify that now:

7. Click the **Insert Field** button. Just like in the formula field editor, two columns appear in an overlay.
8. In the left column, select `Position`.
9. In the right column, select `Min Pay`.
10. Click **Insert**.
11. Click the **Insert Operator** button, and choose `Greater Than`.
12. Click the **Insert Field** button once again.
13. In the left column, select `Position`.
14. In the right column, select `Max Pay`.
15. Click **Insert**.

You should now have an error condition formula that looks like this:

```
Min_Pay__c > Max_Pay__c
```

Now the only thing that remains is to specify the error message when our error condition evaluates to true.

16. In the `Error Message` text box, enter `Min Pay cannot exceed Max Pay`.
17. Next to the `Error Location` field, select the `Top of Page` radio button.



Tip: If a rule requires a user to review the values of multiple fields, it's more appropriate to place the error message at the top of the page because you don't know which field the user needs to change.

18. Click **Save**.

Easy! Now that we've familiarized ourselves with a simple validation rule, let's define one that's a little trickier.

Try It Out: Defining a Validation Rule for Close Date

For our next validation rule, let's ensure that `Close Date` has a value whenever the `Status` field is set to `Closed - Filled` or `Closed - Not Approved`.

The hardest part of this validation rule is defining the error condition formula. When defining a condition like this, it's sometimes easiest to think about it in logical terms first, and then translate that logic to the functions and operators that are provided in the formula editor. In this case, our error condition is true whenever:

```
Close Date is Not Specified
AND
(Status is "Closed - Filled" OR
 "Closed - Not Approved")
```

Let's start with the first piece: “Close Date is Not Specified.” To translate this into terms the formula editor understands, we'll need to use the `ISBLANK()` function again. As you might remember from defining the `Days Open` custom formula field, `ISBLANK()` takes a single field or expression and returns true if it doesn't contain a value. So, remembering that we have to use the internal field name of the `Close Date` field in our formula, `Close Date is Not Specified` translates to:

```
ISBLANK( Close_Date__c )
```

Next, let's figure out how to translate “Status is 'Closed - Filled'.” To test for picklist values, we'll need to use another function: `ISPICKVAL()`. `ISPICKVAL()` takes a picklist field name and value, and returns true whenever that value is selected. So “Status is 'Closed - Filled'” translates to:

```
ISPICKVAL( Status__c , "Closed - Filled")
```



Tip: When working with picklists in formulas, convert them to text using either the `ISPICKVAL()` function or the `TEXT()` function. For example, to check the value of a picklist using the `TEXT()` function, use `TEXT(Status__c) = "Closed - Filled"`.

Now we just have to combine these translations, which we can do using a mix of the `&&` and `||` functions. Both functions evaluate an unlimited number of expressions, but `&&` returns true if **all** of the expressions are true while `||` returns true if **any** of the expressions are true. For example:

```
exp1 && exp2 && exp3
```

returns true when *exp1*, *exp2*, and *exp3* are all true. Likewise,

```
exp1 || exp2 || exp3
```

returns true when any one of *exp1*, *exp2*, or *exp3* are true.

Put these functions all together with our other expression translations, and we get our completed error condition formula:

```
ISBLANK(Close_Date__c) &&
  (ISPICKVAL(Status__c , "Closed - Filled") ||
   ISPICKVAL(Status__c , "Closed - Not Approved"))
```

Phew! Now we can quickly define our second validation rule using this formula:

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In the Validation Rules related list, click **New**.
4. In the Rule Name text box, enter `Close_Date_Rule`.
5. Select the **Active** checkbox.
6. In the Description text box, enter `Close Date must be specified when Status is set to 'Closed - Filled' or 'Closed - Not Approved.'`
7. In the Error Condition Formula area, enter the following formula:

```
ISBLANK(Close_Date__c) &&
  (ISPICKVAL(Status__c , "Closed - Filled") ||
   ISPICKVAL(Status__c , "Closed - Not Approved"))
```

8. Click **Check Syntax** to make sure the format of the formula is correct.
9. In the Error Message text box, enter `Close Date must be specified when Status is set to 'Closed.'`

10. Next to the `Error Location` field, select the `Field` radio button, and then choose `Close Date` from the drop-down list.
11. Click **Save**.

Look at What We've Done

Let's revisit the `Positions` tab to test the validation rules that we've just made.

1. Click the `Positions` tab.
2. Click **New**.

First let's try defining a new position with a value for `Min Pay` that's larger than `Max Pay`.

3. Specify any value for the required `Position Title` field.
4. In the `Min Pay` field, enter 80,000.
5. In the `Max Pay` field, enter 40,000.
6. Click **Save**.

Did you see what happened? Your custom error message popped exactly like any other error message in the app!

The screenshot shows a 'Position Edit' form titled 'New Position'. At the top, there are buttons for 'Save', 'Save & New', and 'Cancel'. Below the title bar, a red error message is displayed: 'Error: Invalid Data. Review all error messages below to correct your data. Min Pay cannot exceed Max Pay.' A red arrow points from this message to a red-bordered box containing the text: 'Validation rule errors look and behave just like other errors in the app'. The form fields include: Position Title (Sales Representative), Job Description, Responsibilities, Skills Required, Educational Requirements, Min Pay (80,000.00), Max Pay (40,000.00), Travel Required, Java, JavaScript, C#, Apex, Open Date (2/2/2010), Hire By (5/3/2010), Close Date, Location (San Francisco, CA), Status (New Position), Type (--None--), Functional Area (--None--), and Job Level (--None--). There are also buttons for 'Save', 'Save & New', and 'Cancel' at the bottom.

Figure 26: Error Message from a Validation Rule

Now let's test our other validation rule:

7. In the `Min Pay` field, enter 40,000.
8. In the `Max Pay` field, enter 80,000.
9. From the `Status` drop-down list, choose `Closed - Not Approved`.
10. Click **Save**.

Our second validation rule is triggered, this time because we didn't specify a value for `Close Date`. Once we do, the record saves normally.

The `Positions` tab is now fully functional, with a couple of validation rules to ensure that users don't make certain mistakes. But are the fields where we want them? Are the fields that must

have values marked as required? In the next section, we'll fine-tune our Position custom object by modifying its page layout.

Introducing Page Layouts

After defining all those fields and validation rules, we now have a fully functional Position custom object. However, it doesn't look all that nice—all of the long text areas appear at the top, and it's hard to scan. Let's move some things around to make this page easier for our users. We can do that by customizing the Position object's page layout.

A *page layout* controls the position and organization of the fields and related lists that are visible to users when viewing a record. Page layouts also help us control the visibility and editability of the fields on a record. We can set fields as read-only or hidden, and we can also control which fields require users to enter a value and which don't.

Page layouts are powerful tools for creating a good experience for our users, but it's crucial that we remember one important rule: *page layouts should never be used to restrict access to sensitive data that a user shouldn't view or edit*. Although we can hide a field from a page layout, users can still access that field through other parts of the app, such as in reports or via the API. (We'll learn more about security that covers all parts of the app in [Securing and Sharing Data](#) on page 131.)

Now let's see if we can organize the fields on our Position object in a way that's more user friendly.

Becoming Familiar with the Page Layout Editor

The Force.com platform has two drag-and-drop tools for editing page layouts: the original page layout editor and the enhanced page layout editor.

The enhanced page layout editor provides all of the same functionality as the original editor, but with additional enhancements, including an intuitive WYSIWYG interface and the ability to customize the space between fields in your layouts. Since the enhanced page layout editor is enabled by default, it's the one we'll use to edit page layouts in this book.



Note: The enhanced page layout editor doesn't work with some older browsers. If you don't want to upgrade your browser, you can always switch to the original page layout editor by clicking **Your Name** ► **Setup** ► **Customize** ► **User Interface** and deselecting the `Enable Enhanced Page Layout Editor`; however, if you use the original page layout editor, what you see on your screen will not match the

procedures in this book, and you won't be able to customize the space between fields in your layouts.

Let's try using the page layout editor to edit the page layout of the Position object:

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In the Page Layouts related list, click **Edit** next to Position Layout.

Welcome to the page layout editor! As you can see, this editor is different from the ones that we've already used in other areas of the platform. That's because we're designing a user interface and need to see how our page will look as we're working. Before going any further, let's give ourselves a quick orientation to how this page is set up.

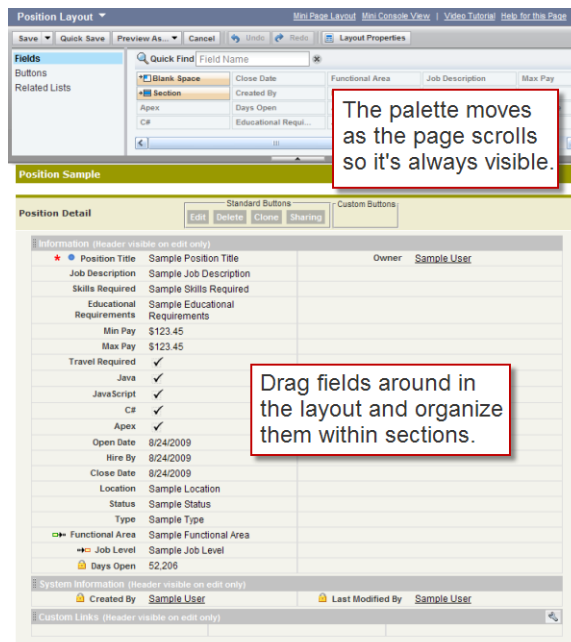


Figure 27: Page Layout Editor

The page layout editor consists of a palette at the top of the screen and the page layout below it. The palette contains the user interface elements that are available for you to add to the page layout, including fields, buttons, links, and related lists. To add one of these user interface elements to the page layout, simply select the category to which the element belongs on the left column of the palette and drag the element from the palette to the page layout. To remove a user interface element from the page layout, drag the element from the page layout to the

right side of the palette, or click the X (✖) icon that appears when you hover over the field. A toolbar above the palette provides various functions, such as saving and previewing your changes.

Now that we know what we're looking at, let's rearrange the fields in the way a user might want to see them.

Try It Out: Grouping Fields into a New Section

Let's start modifying our page layout by first defining a new section for salary information. On a page layout, a section is simply an area where we can group similar fields under an appropriate heading. This makes it easy for our users to quickly identify and enter the information for a record, especially if our object has a large number of fields:

1. In the palette, select the Fields category.
2. Drag the Section user interface element from the palette to just above the System Information section on the page layout.

When you drag a new section on to the page layout, the Section Properties popup window appears.

3. In the Section Name text box, enter Compensation.

The Section Name field controls the text that's displayed as the heading for the section.

4. In the Display Section Header On area, select both the Detail Page and Edit Page checkboxes.

Displaying the header on both the detail and edit pages helps users understand the context of the information, regardless of whether they are editing the position or just viewing it.

5. In the Layout drop-down list, choose 2-Column.

This option allows us to choose whether we want the fields in our section to be arranged in two columns or one. The default is two columns and is the most commonly-used choice. However, if our section is going to contain text area fields, the one-column layout gives them more space on the page for display.

6. In the Tab-key Order drop-down list, choose Left-Right.

This setting controls the direction that a user's cursor will move when using the Tab key to navigate from field to field.

7. Click OK.

We have a new section for Compensation just above the System Information section! Let's add the `Min Pay` and `Max Pay` fields:

- While pressing the CTRL key, click both the `Min Pay` and `Max Pay` fields in the System Information section, and drag them to the new Compensation section as shown in the following screenshot.

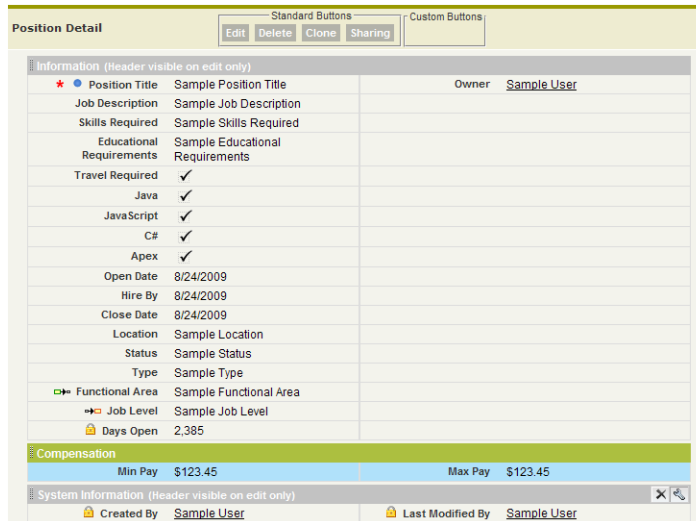


Figure 28: Adding Fields to the Compensation Section

Pressing the CTRL key allows you to select multiple individual user interface elements with your mouse. Alternatively, pressing the SHIFT key allows you to select a group of elements.

Now that we've gone through the process for building one section, let's build two more. As we do this, you might have to scroll up and down to view the entire layout, depending on the size of your screen. Rest assured that if you have to scroll, the palette will move with you, making it easy to add user interface elements even at the very bottom of the page layout.

- Create a new one-column Description section below the Compensation section, and drag `Job Description`, `Responsibilities`, `Skills Required`, and `Educational Requirements` into it.
- Create a new two-column Required Languages section below the Description section, and drag `Apex`, `C#`, `Java`, and `JavaScript` into it.

As you work, you may notice that the fields you add to the page layout are grayed-out in the palette, but the Section user interface element is never grayed-out. This is because fields can only appear one time on each page layout, but the Section user interface element can be reused to create as many sections as you want.



Tip: If you make a mistake while editing the page layout, you can use CTRL+Z and CTRL+Y to undo and redo your recent moves, respectively. The toolbar contains **Redo** and **Undo** buttons as well.

While we're shuffling fields around, let's reorganize the Information section so it's more readable.

11. Arrange the first column of the Information section as follows:

- Position Title
- Status
- Type
- Functional Area
- Job Level
- Travel Required
- Created By

12. Arrange the second column of the Information section as follows:

- Owner
- Location
- Open Date
- Hire By
- Close Date
- Days Open
- Last Modified By

That's much better—our fields are organized, and it's easy to locate all of the information we need.

The Information section still looks a little dense, though. Fortunately, the page layout editor provides a way to add blank spaces that separate the fields within the section to improve its readability even further.

Try It Out: Adding Spaces Between Fields

When designing a page layout, you'll often want to distinguish or “pop” certain fields within a section. An easy and effective way to do this is by inserting blank spaces between fields. For example, in the Information section, inserting blank spaces below the first row in each column draws a user's eye to the most important information in the section, the position title and

owner. We can also use blank spaces above the bottom row in each column to separate the `Created By` and `Last Modified By` fields from the critical position information. Let's give it a try!

1. Select any category in the palette except for Related Lists and Buttons.
2. Drag the Blank Space user interface element from the palette to the page layout right below the `Position Title` field.

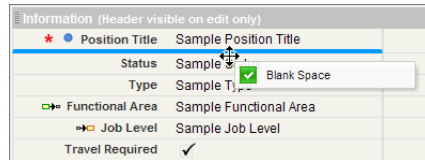


Figure 29: Dragging and Dropping Blank Spaces in a Page Layout

As with the Section user interface element, the Blank Space user interface element is never grayed-out in the palette when you drag it to the page layout because the element can be reused as many times as you want.

3. On the palette, select the Blank Space user interface element again and drag it below the `Owner` field.
4. Add two more blank spaces to the page layout, one above the `Created By` field and the other above the `Last Modified By` field.

Before we move on, you should be aware that if you accidentally navigate away from the page layout before saving your work, your changes will be lost. Rather than tempting fate, let's do a quick save of the work that we've done so far.

5. Click **Quick Save** in the toolbar above the palette, or press CTRL+S if you prefer using keyboard shortcuts.

The quick save feature allows you to save your changes and continue editing the page layout. Clicking **Save** in the toolbar also saves your work, but it takes you back to the page from which you accessed the page layout editor. We have a few more minor tweaks to make to our page layout, so we don't want to leave the page layout editor just yet.

Try It Out: Editing Field Properties

We just used the page layout editor to organize our Position page layout in a logical, easy-to-read fashion, but the page layout editor lets us do more than design the presentation of data—we can also use it to help determine which fields are required, and which fields are read-only:

- If it's required, a user won't be able to create a position record without specifying a value.

- If it's read-only, a user who views a position record edit page won't be able to change its value.

Required fields are denoted with a red asterisk (*), while read-only fields are denoted with a lock icon (🔒).



Caution: Although we can make these changes on the page layout, don't forget our earlier warning! Page layouts should never be used as the sole means to restrict access to sensitive data that a user shouldn't view or edit. That's because page layouts control only a record's edit and detail pages; they don't control access to those fields in any other part of the platform.

At this point, we don't have any fields that need to be read-only, but we do want to make sure that the salary range is always defined for each position, so let's make the `Min Pay` and `Max Pay` fields required. Once we do that, we'll be all done with our `Position` object!

To make the `Min Pay` and `Max Pay` fields required, we need to edit the properties of each field:

1. In the page layout editor, double-click the `Min Pay` field, or select the wrench icon (🔧) next to the field name.

This popup window allows us to edit the `Min Pay` field's properties. We can set the field to read-only and/or required. If we didn't want a user to see the `Min Pay` field at all, we could simply drag it off the layout and onto the palette. We want to make sure it stays visible for right now, so let's leave it in its current spot.

2. Select the `Required` checkbox, and click **OK**.
3. Repeat these steps for the `Max Pay` field.
4. Click **Save** to finish customizing the page layout.

Hooray! We're all done with our `Position` object's page layout.

Look at What We've Done

Congratulations. We've just built ourselves a simple Recruiting app that tracks details about an organization's open positions. Let's check out what we've done by revisiting the `Positions` tab and clicking **New**. Because of the changes that we've made in our page layout, our `Position` edit page should now look like this:

Position Edit Help for this Page ?

New Position

Position Edit

Information ! = Required Information

Position Title: Owner: Admin User

Status: Location:

Type: Open Date:

Functional Area: Hire By:

Job Level: Close Date:

Travel Required:

Compensation

Min Pay: Max Pay:

Description

Job Description:

Responsibilities:

Skills Required:

Educational Requirements:

Required Languages

Java: C#:

JavaScript: Apex:

Figure 30: Final Version of the Position Layout

We have an object with a tab, we've added custom fields, and we've arranged them in a page layout. We've finished our simple app, and now we're well on our way to creating the more complex Recruiting app that we described earlier.

Things are going to get even more interesting in the next chapter. We'll add a few more custom objects to track things like candidates, job applications, and reviews, and then we'll enhance our Recruiting app even further by defining how our objects relate to one another. Before you know it, we're going to have an incredibly powerful Web application, all implemented with a few clicks in the platform.

Chapter 7

Expanding the Simple App Using Relationships

In this chapter ...

- [Introducing Relationships](#)
- [Introducing Relationship Custom Fields](#)
- [Adding Candidates to the Mix](#)
- [Bringing Candidates and Positions Together with Job Applications](#)
- [Introducing Search Layouts](#)
- [Managing Review Assessments](#)
- [Creating a Many-to-Many Relationship](#)
- [Putting it All Together](#)

So far we've accomplished a fair amount—we've created the Recruiting app and built out a fully functional Position custom object with a tab and several types of fields. It's a good start, but there's more to do.

Having just one object in our Recruiting app is like having a party with just one guest—not all that interesting! We need to invite more “people” to the party by building custom objects to represent candidates, job applications, and reviews, and, even more importantly, we need to create *relationships* between them. Just like a party isn't all that fun if you don't know any of the other guests, an app isn't all that powerful unless its objects have links to other objects in the app. That's going to be the focus of this chapter, so let's get started!

Introducing Relationships

So what is a relationship, and why are they important for our app? Just as a personal relationship is a two-way association between two people, in terms of relational data, a relationship is a two-way association between two objects. Without relationships, we could build out as many custom objects as we could think of, but they'd have no way of linking to one another.

For example, after building a Position object and a Job Application object, we could have lots of information about a particular position and lots of information about a particular candidate who's submitted an application for it, but there would be no way of seeing information about the job application when looking at the position record, and no way of seeing information about the position when looking at the job application record. That's just not right!

With relationships, we can make that connection and display data about other related object records on a particular record's detail page. For example, once we define a relationship between the Position and Job Application objects we just talked about, our position record can have a related list of all the job applications for candidates who have applied for the position, while a job application record can have a link to the positions for which that candidate is applying. Suddenly the “people” at our Recruiting app “party” know some of the other guests, and the app just got a lot more interesting.

The screenshot shows a Salesforce record for a 'Technical Writer' position. The 'Position Detail' section includes fields for Position Title, Status (Closed - Canceled), Type, Functional Area, Job Level, Travel Required, Created By (Jane Smith), and Hiring Manager. Below this is a 'Candidate Map' section. At the bottom, there is a 'Job Applications' related list table. A red box highlights the table with the text: 'There is a relationship between this Technical Writer position and the Job Applications on the related list.'

Action	Job Application Number	Candidate	Status	Created Date	Owner First Name	Owner Last Name
Edit Del	JA-00001	C-00001	Rejected	7/22/2010	Jane	Smith
Edit Del	JA-00002	C-00001	New	7/29/2010	Jane	Smith

Figure 31: Relationships Allow Information about Other Object Records to be Displayed on a Record Detail Page

Introducing Relationship Custom Fields

As we learned in [Reviewing Database Concepts](#) on page 23, we can define a relationship between two objects through the use of common fields. On the platform, we can define relationships between objects by creating a *relationship* custom field that associates one object with another. A relationship field is a custom field on an object record that contains a link to another record. When we place a relationship custom field on an object, we're effectively creating a many-to-one relationship between the object on which the relationship field is placed and the other object.

There are different types of relationship fields, each with different implications. The simplest and most flexible type is a *lookup relationship* field, which creates a simple relationship between two objects. For example, if we place a lookup relationship field on a Job Application object that references position records, many job application records can be related to a single position record. This will be reflected both with a new `Position` field on the job application record and with a new Job Applications related list on the position record. You can also put multiple lookup relationship fields on a single object, which means that our Job Application object can also point to a Candidate object.


A second type of relationship field, *master-detail relationship*, is a bit more complex, but more powerful. Master-detail relationships create a special parent-child relationship between objects: the object on which you create the master-detail relationship field is the child or “detail,” and the object referenced in the field is the parent or “master.” In a master-detail relationship, the ownership and sharing of detail records are determined by the master record, and when you delete the master record, all of its detail records are automatically deleted along with it. Master-detail relationship fields are always required on detail records, and once you set a master-detail relationship field's value, you can't change it.

When do you use a master-detail relationship? If you have an object that derives its significance from another object. For example, say you have a Review custom object that contains an interviewer's feedback on a job application. If you delete a job application record, you will probably want all of its review records deleted as well, being that reviews of something that no longer exists aren't very useful. In this case, you want to create a master-detail relationship on the Review custom object with the Job Application object as the master object.

That's the sort of thing that we're going to do in this chapter. First, let's start with the really quick and easy example of putting a `Hiring Manager` field on our Position object—we'll create a many-to-one relationship between the Position object and the standard User object that comes with every organization, reflecting the fact that a hiring manager can be responsible for several positions at a time. Then we'll build out a few more objects and implement a more complex relationship involving positions, job applications, candidates, and reviews.

Try It Out: Relating Hiring Managers to Positions

For our first relationship, let's associate a hiring manager with a position by putting a lookup relationship field on the Position object. The lookup field will allow users to select the hiring manager for the position by selecting from all the users of the Recruiting app.

For example, if Ben Stuart, our recruiter, wants to assign Anastasia O'Toole as the hiring manager for the Benefits Specialist position, he'll be able to do so by clicking the lookup icon () next to the lookup relationship field that we are going to create. Her name will then appear on the Position detail page.


To create the lookup relationship field that accomplishes this, we'll need to go back to the now familiar Position object detail page.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In the Custom Fields & Relationships related list, click **New**.
4. Select **Lookup Relationship**, and click **Next**.
5. In the **Related To** drop-down list, choose **User**, and click **Next**.

As we've mentioned, **User** is a standard object that comes with all organizations on the platform. It contains information about everyone who uses the app in your organization.

6. In the **Field Label** text box, enter **Hiring Manager**. Once you move your cursor, the **Field Name** text box should be automatically populated with **Hiring_Manager**.
7. Click **Next**.
8. Accept the defaults in the remaining two steps of the wizard.
9. Click **Save**.

Look at What We've Done

Now return to the Positions tab, and click **New**. The Position edit page includes a new **Hiring Manager** lookup field! If you click the lookup icon next to this field () , you can search through all of the users of the Recruiting app and select one as the hiring manager. That user's name now appears on the position record:

The screenshot shows the 'Position Edit' form for a 'New Position'. The form is divided into sections: Information and Compensation. The Information section contains fields for Position Title (Apex Developer), Status (New Position), Type (--None--), Functional Area (--None--), Job Level (--None--), Travel Required (checkbox), and Hiring Manager (lookup field). The Compensation section contains Min Pay (90,000.00) and Max Pay. A red box highlights the Hiring Manager lookup field with the text 'Hiring Manager Lookup Relationship Field'.

Figure 32: Hiring Manager Lookup Relationship

As you can see, it was easy to set up this simple relationship between positions and users. And as a general rule, you'll find that relationships are pretty easy to set up.

What gets a little tricky is when we start wanting to create relationships that don't represent a simple many-to-one relationship. We'll see an example of one of those in a little bit. Right now, let's build a custom object for candidates so we'll be able to create some more relationships in our Recruiting app.

Beyond the Basics

Did you know you can make the valid values of a lookup field depend on the value of another field on the record?

Say you want to ensure that hiring managers only create positions for locations with open headcount. You could create a dependent lookup by creating a `Locations with Open Headcount` lookup field and relating it to the `All Locations` field. Then, you would relate the `Locations` field to the `Locations with Open Headcount` field. So when the `Locations with Open Headcount` field's values are "San Francisco, CA" and "Sydney, Australia," the user can only select those values in the `Locations` lookup field.

To find out more, see "About Dependent Lookups" in the Salesforce online help.

Beyond the Basics

Did you know you can restrict the valid values and lookup dialog results for lookup, master-detail, and hierarchical relationship fields?

Say you want to ensure that hiring managers can only create or edit hiring opportunities outside of the United States. You could create a lookup filter by restricting the `AccountName` field on opportunities to allow only hiring managers with the International Hiring Managers profile to create or edit opportunities for accounts outside the United States.

To find out more, see “About Lookup Filters” in the Salesforce online help.

Adding Candidates to the Mix

Let's add a Candidate custom object to our app so we can manage the information about our candidates. We'll also add fields to the object, modify the page layout properties, and create a candidate record. The process for creating the Candidate custom object is almost identical to the one we followed to create the Position custom object, so we'll zip through this quickly.

Try It Out: Creating the Candidate Object

To create our Candidate custom object, navigate back to *Your Name* ► **Setup** ► **Create** ► **Objects**, click **New Custom Object**, and fill out the page according to the following table.

Table 8: Values for Defining the Candidate Object

Field	Value
Label	Candidate
Plural Label	Candidates
Object Name	Candidate
Description	Represents an applicant who might apply for one or more positions
Context-Sensitive Help Setting	Open the standard Salesforce Help & Training window
Record Name	Candidate Number
Data Type	Auto Number

Field	Value
Display Format	C-{00000}
Starting Number	00001
Allow Reports	Yes
Allow Activities	Yes
Track Field History	Yes
Deployment Status	Deployed
Add Notes & Attachments related list to default page layout	Yes
Launch New Custom Tab Wizard after saving this custom object	Yes

To create the Candidates tab, select a `Tab Style` in the first step of the wizard, and then accept all the defaults until you get to the `Add to Custom Apps` page. On this page, select only the `Recruiting App` and click **Save**.

The Recruiting app now has three tabs: Home, Positions, and Candidates. Now let's add some custom fields to the Candidate object.

Try It Out: Adding Fields to the Candidate Object

To create custom fields on the Candidate object, click **Your Name** ► **Setup** ► **Create** ► **Objects**, and click **Candidate** to view its detail page. In the Custom Fields & Relationships related list, use the **New** button to create custom fields according to the following table. Where necessary, we've indicated some additional values you'll need to fill in. Otherwise, you can simply accept all defaults.

One difference you'll see in the Candidate object fields is that three of them—`First Name`, `Last Name`, and `Email`—have the `External ID` option selected. This option allows the values in these fields to be indexed for search from the sidebar of the application. If we didn't select these values as external IDs, we'd only be able to search for records based on the `Candidate Number` field. Setting the `Email` field as an external ID is also going to help us with importing data a little later in this chapter.

Table 9: Candidate Object Custom Fields

Data Type	Field Label	Other Values
Text	First Name	Length: 50 External ID: Selected
Text	Last Name	Length: 50 External ID: Selected
Phone	Phone	
Email	Email	External ID: Selected
Text	Street	Length: 50
Text	City	Length: 50
Text	State/Province	Length: 50
Text	Zip/Postal Code	Length: 15
Text	Country	Length: 50
Text	Current Employer	Length: 50
Number	Years of Experience	Length: 2 Decimal Places: 0
Text	SSN	Length: 9
Picklist	Education	Picklist values: <ul style="list-style-type: none"> • HS Diploma • BA/BS • MA/MS/MBA • Ph.D. • Post Doc
Checkbox	Currently Employed	Default: Checked
Checkbox	US Citizen	Default: Checked
Checkbox	Visa Required	Default: Unchecked
Phone	Mobile	

Data Type	Field Label	Other Values
Phone	Fax	

Try It Out: Modifying the Candidate Page Layout Properties

To finish up with this object, let's organize all of our fields on the page layout and mark some fields as required. To do so, let's go to the Candidate page layout page.

1. Click *Your Name* ► **Setup** ► **Create** ► **Objects**.
2. Click **Candidate**.
3. In the Page Layouts related list, click **Edit** next to the Candidate Layout.
4. Create three new double-column sections below the Information section: Address, Employment, and Additional Details. Drag the appropriate fields into them, as shown in the Candidate Object Page Layout image, and don't forget to click **Quick Save** so you can save your work as you go.
5. Set the `First Name`, `Last Name`, and `Email` fields to required as follows:
 - a. Use CTRL+click to select all three required fields.
 - b. Double-click your selection.
 - c. Select the `Required` checkbox in the Select All row, and click **OK**.
6. Click **Save**.

Your Candidate page layout should now look similar to the following screenshot.

The screenshot shows the 'Candidate Edit' page for candidate C-00001. The page has a header with the candidate number and a 'Help for this Page' link. Below the header are three buttons: 'Save', 'Save & New', and 'Cancel'. The main content is organized into four sections:

- Information:** Fields for Candidate Number (C-00001), First Name (George), Last Name (Schnell), SSN (987654321), Owner (Admin User), Phone ((619) 555-5555), Mobile ((510) 555-5555), Fax ((510) 555-5555), and Email (george@schnell.com).
- Address:** Fields for Street (111 Main St), City (Florida), State/Province (FL), Zip/Postal Code (92111), and Country (USA).
- Employment:** Checkboxes for 'Currently Employed' (checked) and 'Visa Required' (checked), a text field for 'Current Employer' (Seaworld), and a text field for 'Years of Experience' (3).
- Additional Details:** Checkboxes for 'US Citizen' (unchecked) and 'Visa Required' (checked), and a dropdown menu for 'Education' (Ph.D.).

At the bottom of the form are three buttons: 'Save', 'Save & New', and 'Cancel'.

Figure 33: Candidate Object Page Layout

Look at What We've Done

Here's a quick way to verify that you did everything correctly.

1. Click the Candidates tab.
2. Click **New**.
3. Create a new record for a candidate named Ethan Tran.
4. Enter a value for each of the required fields. Salesforce will not verify the email address you enter in the `Email` field right now, so feel free to enter a fictitious one.
5. Click **Save**.

How does the page layout look? Are the fields where you want them?

If your page layout doesn't look quite right and you need to make a few adjustments, click the **Edit Layout** link in the upper right corner.

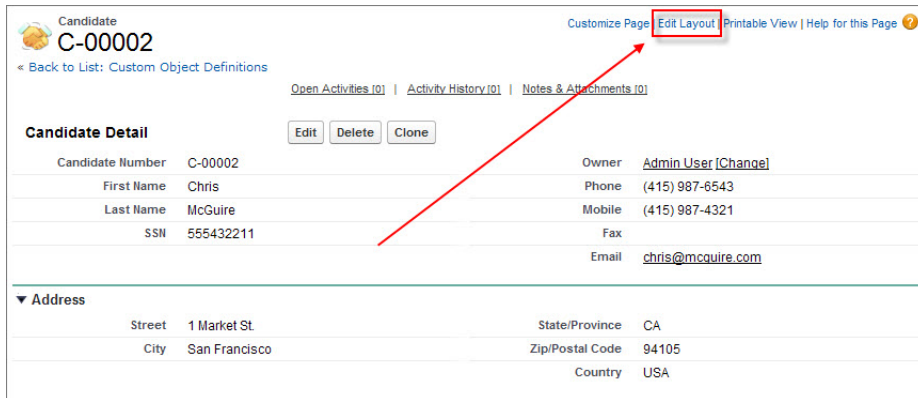


Figure 34: Edit Layout Link

The **Edit Layout** link takes you directly to the page layout editor and lets you modify the page you are currently viewing.

If you are able to successfully create a new candidate record and everything looks okay, let's move on to the Job Application object!

Bringing Candidates and Positions Together with Job Applications

Our app can track candidates and open positions, but there's a crucial element that's missing: how do we know which candidates are interested in which positions? We can create lookup relationship fields on the Candidate object that let recruiters specify the positions in which the candidate is interested, but what if we want to track additional information, such as whether the candidate is currently scheduled to interview for one of those positions? And wouldn't it be helpful if the recruiter has a way of storing the cover letters that candidates tailor for each specific job to which they are applying?

We can satisfy these requirements with a Job Application custom object that stores data about an individual candidate's application to a single position. Each time a candidate wants to apply for a position, the recruiter can create a job application record that contains the candidate's name and the position to which he or she is applying, as well as any cover letter that the candidate may have submitted specifically for that position. Recruiters will also be able to indicate the status of the candidate's application, such as whether he or she is scheduled for an interview or if the application has been rejected. After we create the Job Application object and its fields, we'll make a few small modifications to the Position, Candidate, and Job Application objects so that each position record displays the names of the candidates who have

applied to it, and each candidate record displays the name of the positions to which the candidate has applied.

Try It Out: Creating the Job Application Object

You should be a pro at this by now! To create our Job Application custom object, navigate back to **Your Name** ► **Setup** ► **Create** ► **Objects**, click **New Custom Object**, and fill out the page according to the following table.

Table 10: Values for Defining the Job Application Object

Field	Value
Label	Job Application
Plural Label	Job Applications
Object Name	Job_Application
Description	Represents a candidate's application to a position
Context-Sensitive Help Setting	Open the standard Salesforce Help & Training window
Record Name	Job Application Number
Data Type	Auto Number
Display Format	JA-{00000}
Starting Number	00001
Allow Reports	Yes
Allow Activities	Yes
Track Field History	Yes
Deployment Status	Deployed
Add Notes & Attachments related list to default page layout	Yes
Launch New Custom Tab Wizard after saving this custom object	Yes

To create the Job Applications tab, select a `Tab Style` in the first step of the wizard, and then accept all the defaults until you get to the Add to Custom Apps page. On this page, select only the Recruiting app, and then click **Save**.

We're now just a few custom fields away from linking the Job Application object with the Position and Candidate objects.

Try It Out: Adding Fields to the Job Application Object

Here's another procedure that we've done several times before, but this time we only need to define four custom fields instead of the nearly twenty that we built for the Candidate object. We'll need to add a text field for the candidate's cover letter, a picklist field so that we can track the application's status, and two lookup relationship fields that will create relationships between the Job Application object and the Position and Candidate objects.

Although these fields are almost identical to the ones we created earlier, you'll notice when you're defining the lookup relationship fields that there's a new step in the custom field wizard *Step 6: Add Custom Related Lists*. This step of the wizard is where we can specify a heading for the Job Applications related list that will show up on both the Candidate and Position detail pages.

Why didn't we see this step earlier when we created our `Hiring Manager` lookup field? It turns out that `User` is a unique standard object: it doesn't have a tab, and you cannot add related lists to it. The platform knows this, so it leaves out the related list step whenever someone adds a lookup relationship field that references the `User` object.

Now that we're all squared away with that small difference, let's finish up these Job Application fields. Click **Your Name** ► **Setup** ► **Create** ► **Objects**, and then click **Job Application** to view its detail page. In the Custom Fields & Relationships related list, use the **New** button to create custom fields according to the following table. Where necessary, we've indicated some additional values you'll need to fill in. Otherwise you can simply accept all defaults.

Table 11: Add Custom Fields to the Job Application Object

Data Type	Field Label	Other Values
Lookup Relationship	Candidate	Related To: Candidate Related List Label: Job Applications
Lookup Relationship	Position	Related To: Position Related List Label: Job Applications

Data Type	Field Label	Other Values
Text Area (Long)	Cover Letter	Length: 32,000 # of Visible Lines: 6
Picklist	Status	Picklist values: <ul style="list-style-type: none"> • New • Review Resume • Phone Screen • Schedule Interviews • Extend an Offer • Hired • Rejected Use first value as default value: Selected

Before we move on, there is one more important step: we need to enter a name for the child relationship that the **Position** field created between the Job Application and Position objects. This step is only necessary for the `Position` relationship field because of how we will use this relationship when we create our candidate map feature in [Moving Beyond Point-and-Click App Development](#) on page 281.

To name the child relationship:

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**, and click **Job Application**.
2. In the Custom Fields & Relationships related list, click `Position`, then click **Edit**.
3. In the Child Relationship Name field, enter `Job Applications` and click **Save**. The Force.com platform saves the child relationship name as `Job_Applications`.

Look at What We've Done

Tada! If you click on the new Job Applications tab and click **New**, you'll see the Candidate lookup field, a `Position` lookup field, the candidate's cover letter, and a `Status` picklist field.

The screenshot shows a web form titled "Job Application Edit" for application ID "JA-00001". The form includes fields for Job Application Number, Candidate, Position, Cover Letter, and Status. The Status field is currently set to "Phone Screen" and a dropdown menu is open, showing options: --None--, New, Review Resume, Phone Screen (highlighted), Schedule Interviews, Extend an Offer, Hired, and Rejected. Buttons for "Save", "Save & New", and "Cancel" are visible at the top and bottom of the form.

Figure 35: Custom Fields on the Job Application Edit Page

But there's more! Because we've built a couple of lookup relationships, our candidate and position record detail pages now each have a new Job Applications related list. And the Job Application detail page includes links to the candidate and position records that it references. All three objects are now related and linked to one another!

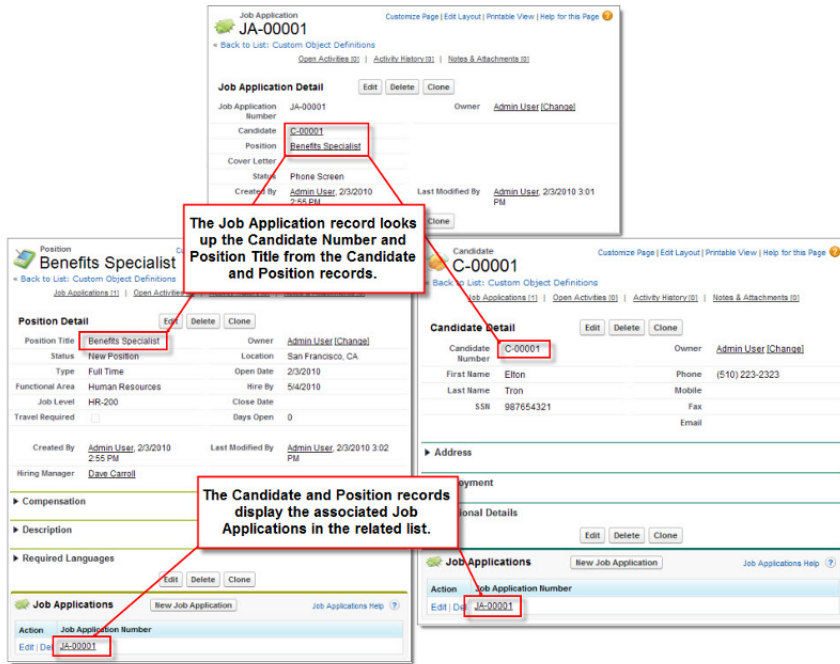


Figure 36: Job Application Links to Position and Candidate Data

Before we move on, let's see if we can clean up the usability of our app a bit more so our users don't have to identify candidates and job applications by number when they click the lookup button in the Job Application edit page, or when they look at the Job Applications related list on the Candidate or Position detail pages.

Introducing Search Layouts

By default, all lookup dialogs and related lists that result from new relationships, such as the ones we've defined in this chapter, only display the record name or number. For example, if you go ahead and create a job application, you might find the Candidate lookup dialog a little cryptic because the only listed field is Candidate Number, as shown in the following screenshot.

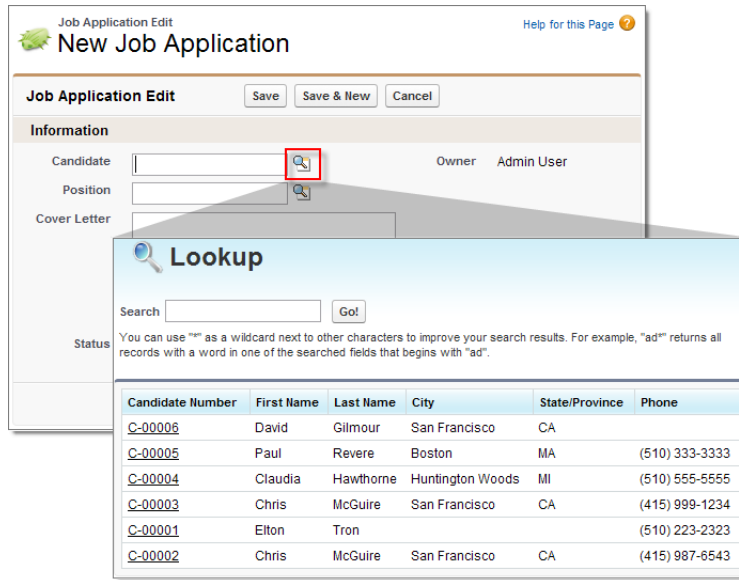



Figure 37: Default Candidate Lookup on the Job Application Object

Likewise, the Job Applications related lists on the Position and Candidate detail pages only display a job application number. It is much more useful if these related lists also include the associated candidate's name or position.

To fix these issues, we can add fields to the *search layouts* for the objects that we've defined. Search layouts are ordered groups of fields that are displayed when a record is presented in a particular context, such as in search results, a lookup dialog, or a related list. By adding fields, we can give users more information and help them locate records more quickly.

The Search Layouts related list on the custom object detail page is the place to modify these sets of fields. Go to **Your Name > Setup > Create > Objects** and select the Candidate object. You'll see that the available search layouts include the following:

Table 12: Available Search Layouts

Layout Name	Description
Search Results	The search results that originate from searching in the left Sidebar Search of the application or in Advanced Search.
Lookup Dialogs	The lookup dialog results that originate from clicking the  button next to a lookup field on an edit page.

Layout Name	Description
Tab	The list of recent records that appears on the home page of a tab, and in related lists on other object detail pages.
Search Filter Fields	The filters that can be applied to search results.



Note: The List View layout also appears in the Search Layouts related list, but it's not for specifying fields. Instead, it allows you to specify the buttons that appear on the list view page for an object.

Try It Out: Adding Fields to the Candidate Lookup Dialog

Let's add fields to our Candidate lookup dialog:

1. Click *Your Name* ► **Setup** ► **Create** ► **Objects**.
2. Click **Candidate**.
3. In the Search Layouts related list, click **Edit** next to the Lookup Dialogs layout.

The Edit Search Layout page includes a list of available fields from the Candidate object. You can choose up to ten fields to include in the lookup dialog, and order them in any way you choose, except that the object's unique name or number field (such as `Candidate Number`) must be listed first.

4. Move the following fields into the Selected Fields box under `Candidate Number`:
 - First Name
 - Last Name
 - City
 - State/Province
 - Phone
5. Click **Save**.

That's it! To try it out, return to the Job Applications tab, and click **New**. When you click the lookup icon next to the `Candidate` field, the dialog is now much more useful.

Lookup

Search

You can use "*" as a wildcard next to other characters to improve your search results. For example, "ad*" returns all records with a word in one of the searched fields that begins with "ad".

Candidate Number	First Name	Last Name	City	State/Province	Phone
C-00006	David	Gilmour	San Francisco	CA	
C-00005	Paul	Revere	Boston	MA	(510) 333-3333
C-00004	Claudia	Hawthorne	Huntington Woods	MI	(510) 555-5555
C-00003	Chris	McGuire	San Francisco	CA	(415) 999-1234
C-00001	Elton	Tron			(510) 223-2323
C-00002	Chris	McGuire	San Francisco	CA	(415) 987-6543

Figure 38: Modified Candidate Lookup on the Job Application Object

Try It Out: Updating Additional Search Layouts

Now that we've updated one search layout for lookups, the rest should be easy. Use the Search Layouts related list on the custom object detail page to modify the other search layouts as described in the following table.

Table 13: Additional Search Layouts

Object	Search Layout	Add These Fields
Candidate	<ul style="list-style-type: none"> Search Results Candidates Tab 	<ul style="list-style-type: none"> Candidate Number First Name Last Name City State/Province Phone
Candidate	<ul style="list-style-type: none"> Search Filter Fields 	<ul style="list-style-type: none"> Candidate Number First Name Last Name Education Years of Experience City State/Province Country Currently Employed

Object	Search Layout	Add These Fields
Position	<ul style="list-style-type: none"> • Search Results • Lookup Dialogs • Positions Tab • Search Filter Fields 	<ul style="list-style-type: none"> • Position Title • Location • Functional Area • Job Level • Type • Hiring Manager • Status • Open Date • Close Date
Job Application	<ul style="list-style-type: none"> • Search Results • Lookup Dialogs • Job Applications Tab • Search Filter Fields 	<ul style="list-style-type: none"> • Job Application Number • Candidate • Position • Status • Created Date • Owner First Name • Owner Last Name

Now let's create another custom object to provide our hiring managers and interviewers with a place to enter their comments about job applications.

Managing Review Assessments

Interviewers, recruiters, and hiring managers need to be able to create reviews so that they can record their comments about each candidate's job application, and rate the candidate's suitability for the position. They also need to see the reviews posted by other people. To allow our users to perform these tasks, we'll need to create a custom Review object and relate it to the Job Application object.

The Review object has a many-to-one relationship with the Job Application object because one job application can have one or more reviews associated with it. A related list on the job application record will show the associated reviews, representing the “many” side of the relationship.

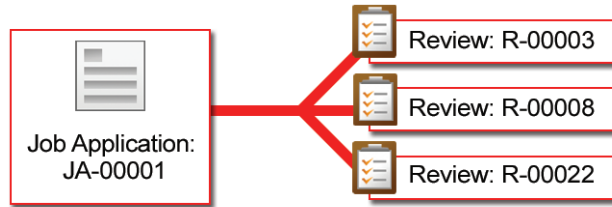


Figure 39: Review Has a Many-to-One Relationship with Job Application

However, instead of creating this relationship with a lookup relationship field, this time we'll use a master-detail relationship field. A master-detail relationship field makes sense in this case because reviews lose their meaning when taken out of the context of a job application, so we'll want to automatically delete reviews when we delete the job application to which they're related.

Try It Out: Creating the Review Object

To create the Review object, navigate back to **Your Name** ► **Setup** ► **Create** ► **Objects**, click **New Custom Object**, and fill out the page according to the following table.

Table 14: Values for Defining the Review Object

Field	Value
Label	Review
Plural Label	Reviews
Object Name	Review
Description	Represents an interviewer's assessment of a particular candidate
Context-Sensitive Help Setting	Open the standard Salesforce Help & Training window
Record Name	Review Number
Data Type	Auto Number
Display Format	R-{000000}
Starting Number	000001
Allow Reports	Yes

Field	Value
Allow Activities	Yes
Track Field History	Yes
Deployment Status	Deployed
Add Notes & Attachments related list to default page layout	Yes
Launch New Custom Tab Wizard after saving this custom object	No

Notice that we didn't launch the tab wizard this time. Reviews don't need a tab of their own because they can be accessed via a related list on the Job Application detail page. When you create an object with a tab, the platform provides access to that object's records in various places other than just the tab, such as in search results and the Recent Items list in the sidebar area of every page. Because most Recruiting app users won't need to see reviews unless it's in the context of a job application, we don't need to create a separate tab for them.

Now let's finish up the custom fields on the Review object.

Try It Out: Adding Fields to the Review Object

Let's start out by adding the master-detail relationship field, which will relate our Review object with the Job Application object. To create the master-detail relationship field, access the Review object detail page.

1. Click *Your Name* ► Setup ► Create ► Objects.
2. Click **Review**.
3. In the Custom Fields & Relationships related list, click **New**.
4. Select `Master-Detail Relationship`, and click **Next**.
5. In the `Related To` drop-down list, choose `Job Application`, and click **Next**.
6. In the `Field Label` text box, enter `Job Application`.

Notice that the `Required` checkbox is automatically selected and cannot be changed. As mentioned earlier, master-detail relationship fields are always required on detail records.

7. Select the `Read/Write` radio button.

This sharing setting prevents people from creating, editing, or deleting a review unless they can also create, edit, or delete the associated job application. We'll learn all about sharing and security in the next chapter.

8. Click **Next**.
9. Accept the defaults in the remaining three steps of the wizard.
10. Click **Save**.

Your master-detail relationship is complete!

Beyond the Basics

Did you know you can create master-detail relationships with multiple levels? With multilevel master-detail relationships you can create reports that roll up data from all levels of the data model, and trigger cascading deletes when a master record is deleted.

Say you want a candidate's applications and review records to be deleted when a hiring manager deletes a candidate. You can create a new master-detail relationship field that relates the Job Application object with the Candidate object. Because the Review and Job Application objects are already related to each other in a master-detail relationship, you've built a multilevel relationship where Candidate is the master, Job Application is the detail, and Review is the subdetail.

To find out more, see "Multilevel Master-Detail Relationships" in the Salesforce online help.

Now that your master-detail relationship is in place, let's think about the other types of fields that would be useful to people looking at a review record.

Most likely, you are going to want to see the name of the candidate and the position for which they are being reviewed. We could create a lookup relationship to the Position and Candidate objects, and then require reviewers to enter those fields when creating a review record, but what if they select the wrong value? Besides, wouldn't it be better if these fields were somehow automatically populated?

To solve this, we'll tap into the synergy of formulas and relationships to create *cross-object formulas*. Cross-object formulas are formulas that span two or more objects by referencing merge fields from related records. This means that formulas on our Review object can access fields on the Job Application object, and formulas on the Job Application object can access fields on both the Position and Candidate objects. We're going to take it even one step further by creating formula fields on our Review object that *span* the Job Application object to reference

fields on the Candidate and Position objects. You'll quickly discover that using related data is much easier than it sounds!

Let's begin by building a formula field on the Review object that references the title of the position on the review's parent job application record.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Review**.
3. In the Custom Fields & Relationships related list, click **New**.
4. Select the **Formula** data type, and click **Next**.
5. In the **Field Label** field, enter **Position**. Once you move your cursor, the **Field Name** text box should be automatically populated with **Position**.
6. Select the **Text** formula return type and click **Next**.
7. Click the **Insert Field** button.

The Insert Field overlay appears, as shown below.

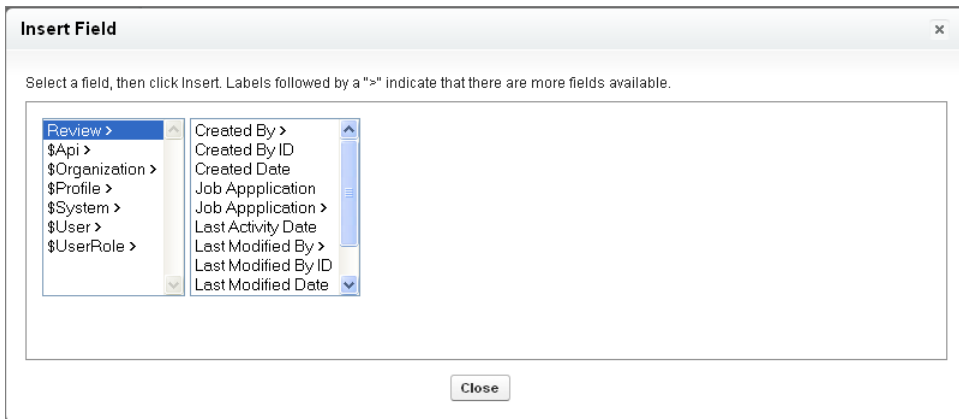


Figure 40: Insert Field Overlay

8. Select **Review** in the first column.

When you choose **Review**, the second column displays all of the **Review** object's fields as well as its related objects, which are denoted by a greater-than sign (>). Notice that the **Created By** and **Last Modified By** fields also have greater-than signs. This is because these are lookup fields to the **User** object.

9. Select **Job Application >** in the second column. The third column displays the fields of the **Job Application** object.
10. Select **Position >** in the third column. The fourth column displays the fields of the **Position** object.

Be sure that you select `Position >` (with the greater than sign) and not `Position`. The one with the greater-than sign is the `Position` object, while the one without the greater than sign is the `Position` lookup field on the Job Application object. In most cases, formulas that access lookup fields return a cryptic record ID. Instead, we want our formula to return the position's title.

11. Choose `Position Title` in the fourth column.
12. Click **Insert**.

Your formula now looks like this:

```
Job_Application__r.Position__r.Name
```

The formula spans to the review's related job application (`Job_Application__r`), then to the job application's related position (`Position__r`), and finally references the position's title (`Name`). Notice that each part of the formula is separated by a period, and that the relationship names consist of the related object followed by `__r`.

13. Click **Next**.
14. Accept all remaining field-level security and page layout defaults.
15. Click **Save**.

That wraps up our first cross-object formula field. Let's try another. This time, we'll add a cross-object formula field on our Review object that displays the first and last names of the candidate being reviewed. We'll also up the ante by using the `HYPERLINK` function so that users can access the candidate's record by clicking the field.

1. Click **Your Name > Setup > Create > Objects**.
2. Click **Review**.
3. In the Custom Fields & Relationships related list, click **New**.
4. Select the `Formula` data type, and click **Next**.
5. In the `Field Label` field, enter `Candidate`. Once you move your cursor, the `Field Name` text box should be automatically populated with `Candidate`.
6. Select the `Text` formula return type and click **Next**.
7. From the `Functions` list, double-click `HYPERLINK`.

The `HYPERLINK` function lets you create a hyperlink to any URL or record in Salesforce. The text of the hyperlink can differ from the URL itself, which is useful here because we want our hyperlink to display the first and last names of the candidate while the URL points to the candidate record itself.

8. Delete `url` from the `HYPERLINK` function you just inserted, but leave your cursor there.

9. Click the **Insert Field** button, and select `Review >, Job Application >, Candidate >, Record ID`, and click **Insert**.

Salesforce generates a unique ID for every record. By inserting the record ID of the candidate in our `HYPERLINK` function, we're enabling our formula field to locate and link to the candidate's record.

10. Delete `friendly_name` from the `HYPERLINK` function, but leave your cursor there.
11. Click the **Insert Field** button, and select `Review >, Job Application >, Candidate >, First Name`, then click **Insert**.
12. Enter a space, then click the **Insert Operator** button and choose `Concatenate`.

The `Concatenate` operator inserts an ampersand (&) in your formula, and joins the values on either side of the ampersand. Here we're going to use the `Concatenate` operator to join the first and last names of the candidate in a single field, even though they are stored in separate fields on the `Candidate` object. The `Concatenate` operator also lets us insert a space between the two names, as you'll see in the next step.

13. Enter another space, then type a blank space enclosed in quotes, like this:

```
" "
```

This appends a blank space after the first name of the candidate.

14. Enter a space, then click the **Insert Operator** button and choose `Concatenate` once more to add a second ampersand in your formula.
15. Click the **Insert Field** button, and select `Review >, Job Application >, Candidate >, Last Name`, then click **Insert**.
16. Delete `[target]` from the `HYPERLINK` function. This is an optional parameter that isn't necessary for our formula field.
17. Click **Check Syntax** to check your formula for errors. Your finished formula should look like this:

```
HYPERLINK
( Job_Application__r.Candidate__r.Id ,
  Job_Application__r.Candidate__r.First_Name__c
  &
  " "
  &
  Job_Application__r.Candidate__r.Last_Name__c )
```

18. Click **Next**.
19. Accept all remaining field-level security and page layout defaults.
20. Click **Save**.

Whew! That one required a little more thought, but using a bit of brainpower here has tremendously improved the usability of our app, which you'll see in a moment when we test our changes to the Review object. Before we start testing, though, let's quickly add two more easy fields to finish our Review object. We need a text area field for the reviewer's assessment, and a number field in which the reviewer can give the candidate a numeric score.

Go to **Your Name > Setup > Create > Objects** and select the Review object. Use the **New** button in the Custom Fields & Relationships related list to create the remaining custom fields for the Review object according to the following table. Where necessary, we've indicated some additional values you'll need to fill in. Otherwise, accept all defaults.

Table 15: Add Custom Fields to the Review Object

Data Type	Field Label	Other Values
Text Area (Long)	Assessment	Length: 32,000 # of Visible Lines: 6
Number	Rating	Length: 1 Always require a value in this field in order to save a record Help text: Enter a 1-5 rating of the candidate.

When you're done, add a quick validation rule to ensure that the Ratings field only accepts the numbers 1 through 5. This will keep our review rating system consistent throughout our organization.

1. Click **Your Name > Setup > Create > Objects**.
2. Click **Review**.
3. In the Validation Rules related list, click **New**.
4. In the Rule Name text box, enter `Rating_Scale_Rule`.
5. Select the **Active** checkbox.
6. In the Description text box, enter `Rating must be from 1 to 5`.
7. Enter the following error condition formula:

```
(Rating__c < 1 || Rating__c > 5)
```

This formula prevents the record from being saved if the value of the `Rating` field is less than one or greater than five.

8. In the `Error Message` text box, enter `Invalid rating. Rating must be from 1 to 5.`
9. Next to the `Error Location` field, select the `Field` radio button, and then choose `Rating` from the drop-down list.
10. Click **Save**.

Our `Review` object is complete! We've added several features that will help users access the data they need in order to assess each job application. There's just one more easy improvement we need to streamline our job application review process. It involves returning to our `Job Application` object and taking advantage of one of the benefits we gain by using a master-detail relationship.

Introducing Roll-Up Summary Fields

The rating system we created on the `Review` object lets users quickly see each reviewer's opinion of the candidate's suitability for the position. While each individual opinion is important, it would be even better to see these ratings compiled in a way that summarizes how the candidate did overall. For example, wouldn't it be great if we could have a `Total Rating` field on each `Job Application` record that shows the sum of all the job application's review ratings?

The good news is that we can! A simple roll-up summary field on the `Job Application` object can summarize data from a set of related detail records and automatically display the output on a master record. Use roll-up summary fields to display the sum, minimum, or maximum value of a field in a related list, or the record count of all records listed in a related list.

Try It Out: Creating Roll-Up Summary Fields

Begin creating your roll-up summary just as you create any other custom field:

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Job Application**.
3. In the `Custom Fields & Relationships` related list, click **New**.
4. Select the `Roll-Up Summary` data type, and click **Next**.

When creating a field on an object that is not the master in a master-detail relationship, the `Roll-Up Summary` data type is not available. This is because roll-up summary fields are only available on the master object in a master-detail relationship.

5. In the `Field Label` field, enter `Total Rating`. Once you move your cursor, the `Field Name` text box should be automatically populated with `Total_Rating`.
6. Click **Next**.
7. In the `Summarized Object` drop-down list, choose `Reviews`.
8. Under `Select Roll-Up Type`, select `SUM`.
9. In the `Field to Aggregate` drop-down list, select `Rating`.
10. Leave `All records should be included in the calculation selected`, and click **Next**.
11. Accept all remaining field-level security and page layout defaults.
12. Click **Save**.

Now our job application records aggregate the ratings of their related reviews. This data could be a little deceptive, though, since some job applications might get reviewed more than others. It would be more helpful if we could see the average rating.

Roll-up summary fields themselves don't allow you to average values together, but you can use them in formulas that do. Let's create a second roll-up summary field on the `Job Application` object, and then build a simple formula field that uses both roll-up summary fields to find the average rating.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Job Application**.
3. In the `Custom Fields & Relationships` related list, click **New**.
4. Select the `Roll-Up Summary` data type, and click **Next**.
5. In the `Field Label` field, enter `Number of Reviews`. Once you move your cursor, the `Field Name` text box should be automatically populated with `Number_of_Reviews`.
6. Click **Next**.
7. In the `Summarized Object` drop-down list, choose `Reviews`.
8. Under `Select Roll-Up Type`, select `COUNT`.

We don't need to specify a `Field to Aggregate` this time since we're just counting the number of related detail records and are not interested in any specific field.

9. Leave `All records should be included in the calculation selected`, and click **Next**.
10. Accept all remaining field-level security and page layout defaults.
11. Click **Save**.

Both roll-up summary fields are in place now. Let's build a formula field called `Average Rating` that divides the value of the first roll-up summary field by the value of the second.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Job Application**.
3. In the Custom Fields & Relationships related list, click **New**.
4. Select the **Formula** data type, and click **Next**.
5. In the **Field Label** field, enter **Average Rating**. Once you move your cursor, the **Field Name** text box should be automatically populated with **Average_Rating**.
6. Select the **Number** formula return type and click **Next**.
7. Click the **Insert Field** button.
8. Select **Job Application** >, then **Total Rating**, and click **Insert**.
9. Click the **Insert Operator** button and choose **Divide**.
10. Click the **Insert Field** button again.
11. Choose **Job Application** >, then **Number of Reviews**, and click **Insert**. Your formula should look like this:

```
Total_Rating__c / Number_of_Reviews__c
```

12. Click **Next**.
13. Accept the defaults in the remaining three steps of the wizard.
14. Click **Save**.

That wraps up all the fields and relationships we need to manage our reviews. Let's quickly organize the presentation of our fields and then test everything we've created.

Try It Out: Customizing the Review Object's Page and Search Layouts

First, let's update the page layout of the Review object so that the **Assessment** text field is in a single column section of the same name.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Review**.
3. In the Page Layouts related list, click **Edit** next to Review Layout.
4. Drag a new section just below the System Information section. The Section Properties dialog box opens.
5. Name the section **Assessment**, and configure it to contain one column.
6. Drag the **Assessment** and **Rating** fields into the **Assessment** section.
7. Click **Save**.

Now, let's configure our Review search layouts so that reviews are always displayed with the associated job application, position, and candidate.

1. In the Search Layouts related list on the Review object detail page, click **Edit** next to Lookup Dialogs and add the following fields:
 - Review Number
 - Rating
 - Job Application
 - Candidate
 - Position
 - Created Date
2. Repeat for the Search Filter Fields layout.

To update the Reviews related list that appears on the Job Application detail page, we'll have to edit the related list directly on the Job Application page layout. This is different from how we added fields to the Job Application related list on the position and candidate detail pages because the Review object doesn't have an associated tab, and therefore, doesn't have a tab search layout. Remember—the tab search layout is responsible for both the fields that appear in the list on the tab home page and the default fields that appear in related lists on other object detail pages.



Note: The tab search layout is responsible for the fields in the related list layout only if the related list properties have not been modified on other objects' page layouts. For example, if you modify the properties of the Job Application related list on the Position page layout, those changes will always override the field specifications of the Job Application tab search layout.

Because the Review object doesn't have a tab search layout, we have to set those fields another way.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Job Application**.
3. In the Page Layouts related list, click **Edit** next to Job Application Layout.
4. On the Job Application page layout, locate the Reviews related list and click the wrench icon (🔧) to edit its properties.
5. On the Related List Properties dialog box, add the following fields to the Selected Fields box:
 - Review Number
 - Rating
 - Candidate
 - Position
 - Created Date

6. From the `Sort By` drop-down list, choose `Review Number`.
7. Click **OK**.
8. Click **Save** on the page layout edit page.

Look at What We've Done

Terrific! Let's go see what we've made:

1. Click the `Job Applications` tab and select a record, or create one if you haven't already.



Tip: When you use the `Candidate` and `Position` lookup dialogs as you're creating a job application record, note that, by default, they only display the most recently viewed records. You can locate additional records by using the search box, which returns records based on the `Candidate Number` or `Position Title` fields, respectively.

Use the `*` wildcard with other characters to improve your search results. For example, searching on `C*` returns every candidate record. Likewise, searching on `*e` returns all position records that include the letter 'e' in the title.

After the job application is created, notice that the `Reviews` related list now appears on the `Job Application` detail page. That's because we related the `Review` object to the `Job Application` object with a master-detail relationship.

2. In the `Reviews` related list, click **New Review** to create a review.

Do you see how the platform automatically filled in the job application number in the review's edit page? That's one of the small, but important, benefits of using the platform to build an application like this—not only is it easy to create links and relationships between objects, but the platform anticipates what we're doing and helps us accomplish our task with as few clicks as possible.

3. Complete the fields on the review, and click **Save**.

Notice that the name of the candidate and the title of the position appear on the review detail page. If you click the candidate's name, his or her record displays.

Go ahead and click around the rest of the app, creating a few more positions, job applications, candidates, and reviews. Pretty neat, huh? Our data is all interconnected, and our edits to the search layouts allow us to view details of several related objects all at once.

Creating a Many-to-Many Relationship

Our Recruiting app now has quite a few many-to-one relationships, but what if we needed to create a many-to-many relationship? For example, what if we have an object that stored information about various employment websites, and we wanted to track which open positions we posted to those sites? This would require a many-to-many relationship because:

- One position could be posted on many employment websites.
- One employment website could list many positions.

Here's where we get a little creative. Instead of creating a relationship field on the Position object that directly links to the Employment Website object, we can link them using a *junction object*. A junction object is a custom object with two master-detail relationships, and is the key to making a many-to-many relationship.

For our app, we're going to create a junction object called Job Posting. A job posting fits into the space between positions and employment websites—one position can be posted many times, and one employment website can have many job postings, but a job posting always represents a posting about a single position on a single employment website. In essence, the Job Posting object has a many-to-one relationship with both the Position and the Employment Website objects, and through those many-to-one relationships, we'll have a many-to-many relationship between the Position and Employment Website objects.



Tip: In many apps, the sole purpose of a junction object is to simply relate two objects, so it often makes sense to give the junction object a name that indicates the association or relationship it creates. For example, if you wanted to use a junction object to create a many-to-many relationship between bugs and cases, you could name the junction object `BugCaseAssociation`.

Let's look at a typical scenario at Universal Containers. There are open positions for a Project Manager and a Sr. Developer. The Project Manager position is only posted on Monster.com, but the Sr. Developer position is more difficult to fill, so it's posted on both Monster.com and Dice. Every time a position is posted, a job posting record tracks the post. As you can see in the following diagram, one position can be posted many times, and both positions can be posted to the same employment website.

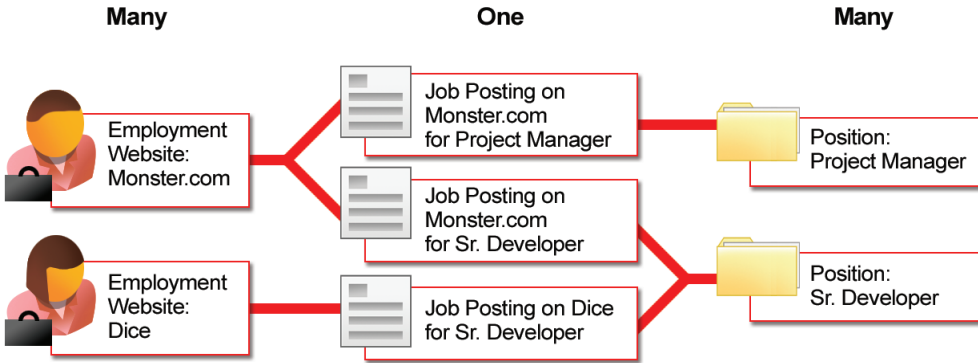


Figure 41: Using a Job Posting Object to Create a Many-to-Many Relationship Between Positions and Employment Websites

In relational database terms, each job posting record is a row in the Job Posting table consisting of a foreign key to a position record and a foreign key to an employment website record. The following entity relationship diagram shows this relationship.

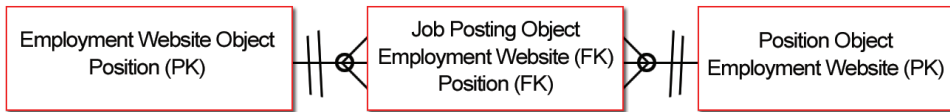


Figure 42: Entity Relationship Diagram for the Position, Job Posting, and Employment Website Objects

Consequently, in order to define a many-to-many relationship between the Position and Employment Website objects, we'll need to create a Job Posting object with the following fields:

- A Position master-detail relationship
- An Employment Website master-detail relationship

Let's get started.

Try It Out: Creating the Employment Website Object

To create our Employment Website custom object, navigate back to **Your Name** ► **Setup** ► **Create** ► **Objects**, click **New Custom Object**, and fill out the page according to the following table.

Table 16: Values for Defining the Employment Website Object

Field	Value
Label	Employment Website
Plural Label	Employment Websites
Object Name	Employment_Website
Description	Information about a particular employment website
Context-Sensitive Help Setting	Open the standard Salesforce Help & Training window
Record Name	Employment Website Name
Data Type	Text
Allow Reports	Yes
Allow Activities	Yes
Track Field History	Yes
Deployment Status	Deployed
Add Notes & Attachments related list to default page layout	Yes
Launch New Custom Tab Wizard after saving this custom object	Yes

To create the Employment Website tab, select a `Tab Style` in the first step of the wizard, and then accept all the defaults until you get to the `Add to Custom Apps` page. On this page, select only the `Recruiting App`, and then click **Save**.

Let's wrap up the Employment Website object by adding a few custom fields.

Try It Out: Adding the URL Field to the Employment Website Object

Obviously, the Employment Website object needs to store the Web address of the employment website. We'll use the URL data type for this field. That way, when users click the field, the URL will open in a separate browser window. In addition to the URL, since most employment websites charge per posting, we'll want to keep track of how much it costs to post there, as well as our maximum budget for posting on the site.

Click **Your Name** ► **Setup** ► **Create** ► **Objects**, and then click **Employment Website** to view its detail page. In the Custom Fields & Relationships related list, use the **New** button to create custom fields according to the following table. Where necessary, we've indicated some additional values you'll need to fill in. Otherwise you can simply accept all defaults.

Table 17: Add Custom Fields to the Job Application Object

Data Type	Field Label	Other Values
URL	Web Address	Required
Currency	Price Per Post	Length: 5 Decimal Places: 2 Required
Currency	Maximum Budget	Length: 6 Decimal Places: 2 Required

Try It Out: Creating the Job Posting Object

Now it's time to create our Job Posting junction object! Navigate back to **Your Name** ► **Setup** ► **Create** ► **Objects**, click **New Custom Object**, and fill out the page according to the following table.

Table 18: Values for Defining the Job Application Object

Field	Value
Label	Job Posting
Plural Label	Job Postings
Object Name	Job_Posting
Description	Represents the junction object between a position and an employment website
Context-Sensitive Help Setting	Open the standard Salesforce Help & Training window

Field	Value
Record Name	Job Posting Number
Data Type	Auto Number
Display Format	JP-{00000}
Starting Number	00001
Allow Reports	Yes
Allow Activities	Yes
Track Field History	Yes
Deployment Status	Deployed
Add Notes & Attachments related list to default page layout	Yes
Launch New Custom Tab Wizard after saving this custom object	No

That was simple enough, but we're not quite done. We need to create the master-detail relationship fields that relate the Job Posting object with the Position and Employment Website objects.

Try It Out: Adding Fields to the Job Posting Object

To turn the Job Posting object into the junction object that relates the Position and Employment Website objects, we'll need to add two master-detail relationship fields. The first master-detail relationship will be the *primary relationship*. The detail and edit pages of our junction object (Job Posting) will use the color and any associated icon of the primary master object (Position). In addition, the junction object records will inherit the value of the Owner field and sharing settings from their associated primary master record.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Job Posting**.
3. In the Custom Fields & Relationships related list, click **New**.
4. Select **Master-Detail Relationship**, and click **Next**.
5. In the **Related To** drop-down list, choose **Position**, and click **Next**.
6. In the **Field Label** text box, enter **Position**. When you move your cursor, the **Field Name** text box should be automatically populated with **Position** as well.

7. Accept the remaining defaults, and click **Next** until you reach the final step of the wizard.

Here, we are given the chance to add the Job Postings related list to the Position object page layout. Instead of displaying information about related job postings, we want this list to show all the employment websites where this position is posted. So let's add the Job Posting related list, but rename it `Employment Websites`.

8. In the `Related List Label` text box, enter `Employment Websites`.
9. Accept the other defaults and click **Save & New**.

We're halfway through the creation of our many-to-many relationship. The next step is to create a second master-detail relationship on the Job Posting object to link it with the Employment Website object.

The second master-detail relationship creates a *secondary relationship*. Unlike the primary relationship, the secondary relationship has no affect on the look and feel of the junction object. However, just as in the primary relationship, the sharing settings of the master record in the secondary relationship also affect who can access the junction record, and deleting a record of the secondary master object will automatically delete its associated junction object records. So in our app, if you delete an employment website record, all of its associated job posting records are deleted as well, even if the position is open.

10. Click **Your Name > Setup > Create > Objects**.
11. Click **Job Posting**.
12. In the Custom Fields & Relationships related list, click **New**.
13. Select `Master-Detail Relationship`, and click **Next**.
14. In the `Related To` drop-down list, choose `Employment Website`, and click **Next**.
15. In the `Field Label` text box, enter `Employment Website`. When you move your cursor, the `Field Name` text box should be automatically populated with `Employment_Website` as well.
16. Click **Next**. Because we are creating a master-detail relationship, these settings cannot be changed.
17. Click **Next**. These settings cannot be changed as well.
18. Click **Next** to view the final step of the wizard.

This time we are given the chance to add the Job Postings related list to Employment Website object page layout. We'll eventually configure this related list to show all the positions that are posted on this website, so let's add the Job Postings related list but rename it `Positions`.

19. In the `Related List Label` text box, enter `Positions`.
20. Accept the other defaults and click **Save**.

Now our many-to-many relationship is complete! Or is it?

While we have an Employment Websites related list on the Position object and a Positions related list on the Employment Websites object, both related lists still display job posting records. This won't do.

In order to achieve our goal of listing multiple positions on an employment website record and multiple employment websites on a position record, we need to customize the fields in these related lists.


Customizing Related Lists in a Many-to-Many Relationship

The capability to customize related lists in a many-to-many relationship is more robust than the capability to customize related lists in a lookup relationship. When you have a lookup relationship between two objects (like the one we created between the Job Application and Candidate objects), the related list on one object can only display fields from the object to which it is directly related; it cannot span to other objects the way formulas can. For example, the Job Applications related list on a candidate record can display any job application field, but it can't display any fields from the Position object, even though the Job Application object has lookup relationships with both the Candidate and Position objects.

Fortunately for us, many-to-many relationships allow for greater flexibility. When working with a many-to-many relationship, the junction object's related list on one master object can display the other master object's fields. We're going to take advantage of this by configuring the Positions related list on each employment website record to display fields from the Position object and vice versa, thus allowing these two objects to span to each other. It's all coming together now!

Try It Out: Customizing the Positions and Employment Websites Related Lists

Let's start by modifying the Employment Websites related list on the Position object.


1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In the Page Layouts related list, click **Edit** next to the Position Layout.
4. Locate the Employment Websites related list and click its wrench () icon.

In the popup window that appears, you'll notice the Available Fields column lists fields from both the Job Posting object and the Employment Website object. If there wasn't a

master-detail relationship between job postings and employment websites, the Available Fields column list would only list job posting fields.

5. Move the Employment Website: Employment Website Name and Employment Website: Web Address fields to the Selected Fields column, and use the up and down arrows to arrange the fields in the following order:
 - Employment Website: Employment Website Name
 - Employment Website: Web Address
 - Job Posting: Job Number
6. Click **OK**.
7. Click **Save** on the page layout.

Now do the same for the Positions related list on the Employment Website object as follows:

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Employment Website**.
3. In the Page Layouts related list, click **Edit** next to the Employment Website Layout.
4. Locate the Positions related list and click its wrench icon (.
5. Move the following fields to the Selected Fields column, and use the up and down arrows to arrange them in the following order:
 - Position: Position Title
 - Job Posting: Job Posting Number
 - Position: Functional Area
 - Position: Location
 - Position: Open Date
6. Click **OK**.
7. Click **Save** on the page layout.

Look at What We've Done

Our many-to-many relationship is complete! Let's see it in action.

1. Create a few sample position and employment website records.
2. Scroll down to the Employment Websites related list at the bottom of any position record, and click **New Job Posting**. The Job Posting edit page appears.
3. Use the lookup icon to select the employment website where you want to post the position, and click **Save**.

The Employment Websites related list on that position now shows the name and Web address of the website to which you just posted, as well as the job posting number. Click the name of the employment website in the related list and scroll down to view the Positions related list, which shows all the positions posted to that website.

Now you know how easy it is to make related information just a click away!

Putting it All Together

We just created several objects and a lot of relationships. The following simple diagram shows us what we've accomplished so far.

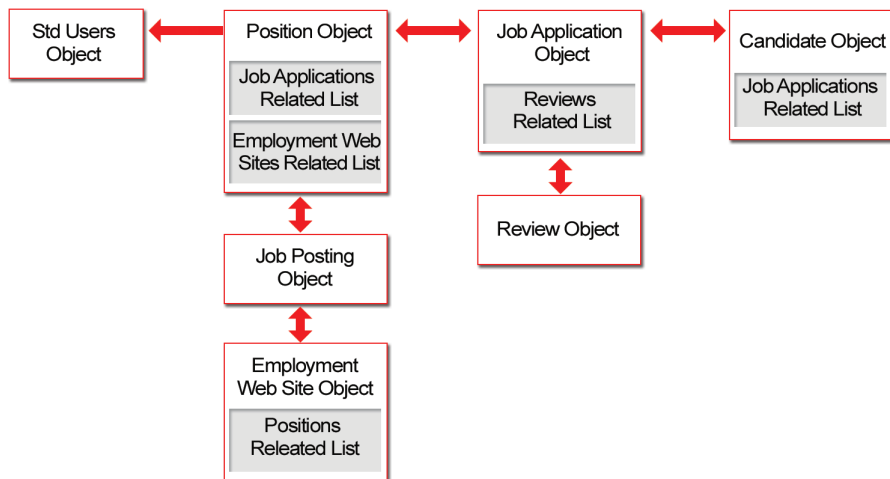


Figure 43: Recruiting App Relationships

All of these relationships, objects, and fields are shown below in an entity relationship diagram. An entity relationship diagram (ERD) is a conceptual representation of structured data, and is especially useful for planning and understanding an app.

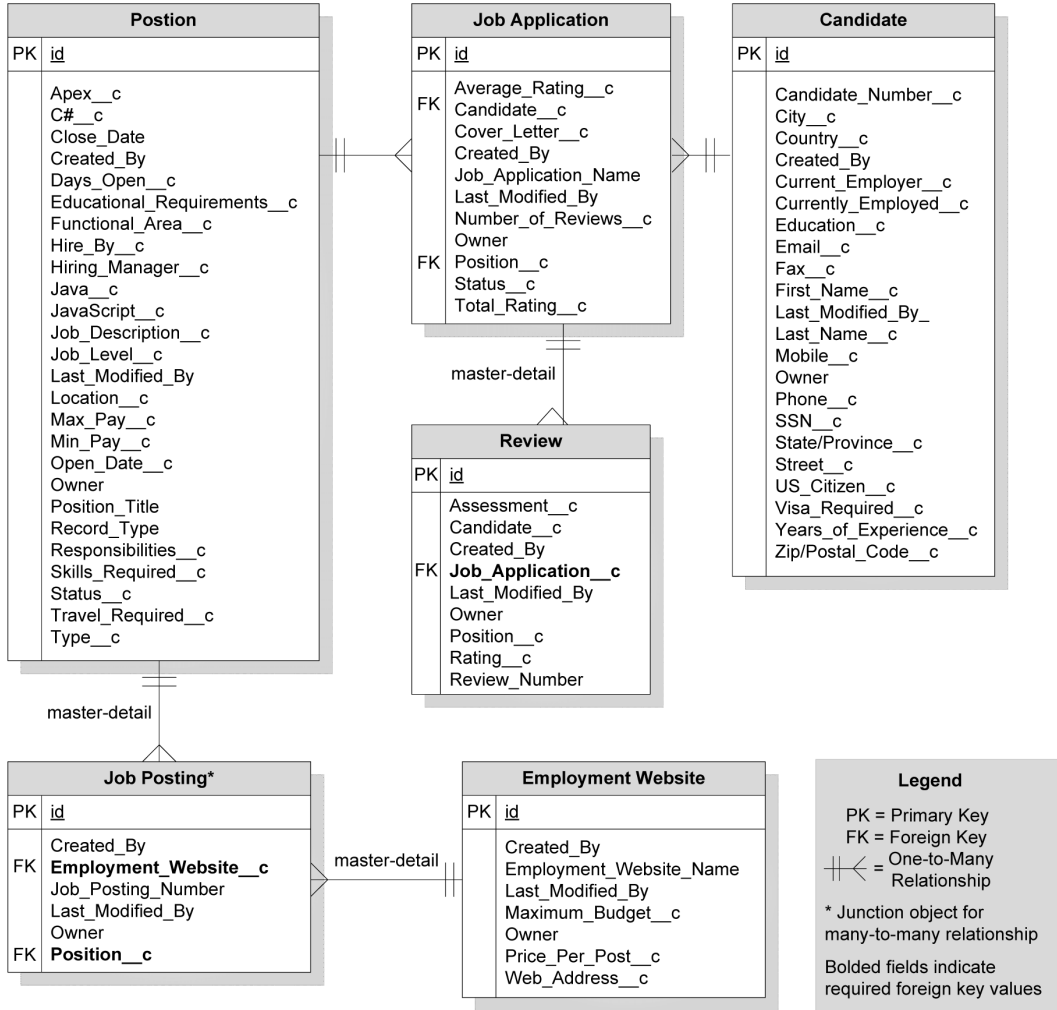


Figure 44: Recruiting App Entity Relationship Diagram

We've now built all of our Recruiting app objects and tabs, and we've defined lots of custom fields—everything from text fields and picklists to more complex formula fields and lookup relationship fields. We've created a robust user interface so that our recruiters and hiring managers can enter and retrieve data about positions and related candidates, job applications, and reviews, and we did all of this without writing a single line of code!

Remember when we assigned Clark Kentman as the hiring manager for the Benefits Specialist position? Let's look at what Clark can do now: He can create and update his positions, and track which websites he's posted them on. He can look at details about any candidates who have applied for the Benefits Specialist job, and he can review their related job applications.

He can also check the status of the job applications. He no longer has to go to Human Resources to search through Microsoft Word documents and spreadsheets to manage his tasks in the hiring process. The Recruiting app is well on its way to becoming a fully-functional and useful application!

However, before we leave this chapter behind, let's get ourselves prepared for the rest of this book by creating and importing some real data. It'll help us when we get to our next chapter on security and sharing if we have some records that we can work with.

Try It Out: Downloading Sample Data

In addition to entering data via our tabbed pages, we can also use the handy Import Wizard to import multiple records at a time. The ability to easily import data into your custom objects is one of the Force.com platform's key benefits. Let's download some sample data so we can add more records to our custom objects without tons of typing.

1. Download the `RecruitingApp-5_0.zip` file containing the sample CSV (comma-separated values) import files from developer.force.com/books/fundamentals.
2. Extract the zip file to `C:\recruiting` (or any directory on your computer).
3. Go to `C:\recruiting`. This directory contains three CSV files: `Positions.csv`, `Candidates.csv`, and `JobApplications.csv`. (The directory also other contains files that you'll use later in [Moving Beyond Point-and-Click App Development](#) on page 281.)

Before we import anything, we need to make a modification to the import file for positions. The sample `Positions.csv` you downloaded contains fictional users in the Hiring Manager column. The names of these users most likely won't match any user in your organization, and if you import the file "as is," the Import Wizard won't be able to find any matching users, and the Hiring Manager field on each position record will be left blank. So let's go ahead and make that change.

4. Go to `C:\recruiting`, and open `Positions.csv` in Excel, a text editor, or any other program that can read CSV files.
5. In the Hiring Manager column, replace the fictional users with the first and last name of a user in your organization.
6. Save the file, making sure to maintain the CSV format.



Note: If your locale isn't English (United States), the date and field values in `Positions.csv` are also invalid. You'll need to change them before you import.

Try It Out: Using the Import Wizard

Now, let's walk through the process of importing position records using the Import Wizard and the `Positions.csv` file you downloaded.

1. Click **Your Name** ► **Setup** ► **Data Management** ► **Import Custom Objects**.
2. Click **Start the Import Wizard!**. The Import Wizard appears.
3. Select `Position` for the type of record you're importing, and click **Next**.
4. Choose `Yes` to prevent duplicate position records from being created as a result of this import. Accept the other defaults for matching, and click **Next**.
5. Select `None` for the record owner field. We didn't include a `User` field in the CSV file to designate record owners. The Import Wizard assigns you as the owner of all new records.
6. Choose the `Hiring Manager` lookup relationship field so you can link position records with existing `User` records in the Recruiting app, and click **Next**.
7. Select `Name` as the field you want to match against as the Import Wizard compares `Hiring Manager` names in your import file with `User` names in the system, and click **Next**.
8. Click **Browse** and find `C:\recruiting\Positions.csv`. Click **Next**.
9. Use the drop-down lists to specify the Salesforce fields that correspond to the columns in your import file. For your convenience, identically matching labels are automatically selected. Click **Next**.
10. Click **Import Now!**

Use the following table to repeat the import process for candidate records. You'll notice the wizard skips the two steps about lookup relationship field matching—because the `Candidate` object doesn't have any lookup relationship fields, the Import Wizard automatically leaves those steps out.

Table 19: Importing the `Candidates.csv` File

For this wizard step...	Select these options...
Choose Record	Candidate
Prevent Duplicates	No—insert all records in my import file
Specify Relationships	None
File Upload	Browse to <code>C:\recruiting\Candidates.csv</code>
Field Mapping	Accept all defaults
Verify Import Settings	Click Import Now!

Finally let's do it one more time for job application records. In this iteration, we're going to make use of the `Email` field, an external ID on the `Candidate` object, to match up job applications with the correct candidate records.

Table 20: Importing the `Job_Applications.csv` File

For this wizard step...	Select these options...
Choose Record	Job Application
Prevent Duplicates	No—insert all records in my import file
Specify Relationships	Which user field...? None Which lookup fields...? Candidate, Position
Define Lookup Matching	Which field on Candidate...? Email (External ID) Which field on Position...? Position Title
File Upload	Browse to <code>C:\recruiting\Job_Applications.csv</code>
Field Mapping	Email (col 0): Candidate Position Title (col 1): Position
Verify Import Settings	Click Import Now!

Great! While the files are importing, you can go to **Your Name** ► **Setup** ► **Monitoring** ► **Imports** to check on their status.

Once the import operations have completed, return to the Positions, Candidates, or Job Applications tab and click **Go!** next to the `View` drop-down list. You'll see a list of all the new records you just imported.

We've just added a bunch of data to our app without a lot of work. In the next chapter, we'll take a look at all the ways we can control access to this data using the built-in tools of the platform. We'll get into the nitty-gritty about security, sharing rules, permissions, roles, and profiles.

Chapter 8

Securing and Sharing Data

In this chapter ...

- [Controlling Access to Data in Our App](#)
- [Data Access Concepts](#)
- [Controlling Access to Objects](#)
- [Controlling Access to Fields](#)
- [Controlling Access to Records](#)
- [Displaying Field Values and Page Layouts According to Profiles](#)
- [Putting It All Together](#)
- [Delegating Data Administration](#)
- [Summing Up](#)

In the last chapter, we expanded the Recruiting app to include advanced fields and complex object relationships. The new Candidate object tracks information about prospective employees, recruiters can relate candidates to positions through the new Job Application object, and interviewers can add assessments and ratings of the candidates on the new Review object. That's a pretty robust app! The enhanced data model also lays the groundwork for adding powerful functionality like workflow and approvals and reporting, which we'll cover in later chapters.

Now that we've got all of our object relationships in place, it's time to start thinking about who's actually going to be using the app and how much access they should have to its data. As with many apps, our Recruiting app exposes sensitive information, like social security numbers, salary amounts, and applicant reviews that could really come back to haunt us if the wrong people saw them. We need to provide security without making it harder for our recruiters, hiring managers, and interviewers to do their jobs.

Here we're going to see another one of the huge benefits that the Force.com platform has to offer. You get simple-to-configure security controls that easily allow us to restrict access to data that users shouldn't see, without a lot of headaches. Similar to Access Control Lists or Windows folder permissions, the Force.com platform allows us to specify who can view, create, edit, or delete any record or field in the app. In this chapter, we'll see how we can use the Force.com platform to implement those rules.

Controlling Access to Data in Our App

As we've already seen, there are three types of users who will need to access the data in our Recruiting app: recruiters, hiring managers, and interviewers. To these three, let's add a fourth type of user—a standard employee who doesn't perform any interviews and who never needs to hire anyone. (This employee will help us determine the default permissions that should apply to all of the new recruiting objects in our app.)

One by one, let's take a look at the kinds of access that each one of these users needs and, more importantly, the kinds of access they *don't* need to do their jobs. Once we've compiled a set of required permissions, we'll figure out how to implement them in the rest of the chapter.

Required Permissions for the Recruiter

For our first set of required permissions, let's take a look at Mario Ruiz, a recruiter at Universal Containers. To do his job, Mario needs to be able to create, view, and modify any position, candidate, job application, or review that's in the system, and have full control over job postings on employment websites. Likewise, Mario needs to view and modify the recruiting records that all of the other recruiters own, since all of the recruiters at Universal Containers work together to fill every position, regardless of who created it.

Although Mario has the most powerful role in our Recruiting app, we still can't give him complete free reign. While it's OK for job posting and employment website data to be permanently deleted at any time, state and federal public records laws require that all other recruitment-related records be saved for a number of years so that if a hiring decision is questioned, it can be defended in court. Consequently, we need to make sure that Mario will never accidentally delete a record that needs to be saved to fulfill the law.

But how will he keep the number of positions, candidates, job applications, and reviews in check if he can't delete them? Won't the app become swamped with old data? Not if we're smart about it—instead of having Mario delete old records, we can use the `Status` field on a record as an indication of whether it's current. We'll filter out all of the old records by using a simple list view.

Here's a summary of the required permissions that we need to implement for a recruiter:

Table 21: Summary of Required Permissions: Recruiter

	Position	Candidate	Job Application	Review	Job Posting	Employment Website
Recruiter	• Read	• Read	• Read	• Read	• Read	• Read
	• Create	• Create	• Create	• Create	• Create	• Create
	• Edit	• Edit	• Edit	• Edit	• Edit	• Edit
					• Delete	• Delete

Required Permissions for the Hiring Manager

Our next set of required permissions is more challenging. Ben Stuart, our hiring manager, needs to be able to access the recruiting records related to his open positions, but he shouldn't be mucking around with other recruiting records (unless they're owned by other hiring managers who report to him). Also, there are certain sensitive fields that he has no need to see, like the social security number field. Let's go object by object to really drill down on what Ben does and doesn't need to access in order to perform his job:

Position

First of all, Ben likes to post his own positions so that he can publicize them as fast as possible, but in our app, Mario the recruiter ultimately needs to take ownership of the record to make sure the position gets filled. As a result, Ben needs the ability to create positions, but then we'll need to find a mechanism to make sure that they ultimately get transferred to Mario for ownership. (Hint: as you'll see in [Using Custom Workflow and Approval Processes](#) on page 199, we'll tackle that problem with a workflow rule that transfers position ownership to a recruiter when a new position is created by a hiring manager. For now just assume that this already works.)

Ben should also be able to update and view all fields for positions for which he's the hiring manager, but he should only be able to view other managers' positions.

Candidate

Ben sometimes wants to poach a prime candidate who's applying for a position under another manager, but this is a practice that Universal Containers frowns upon. As a result, Ben should only be able to view those candidates who have applied for a position on which he's the hiring

manager. Also, since Ben has no reason to see a candidate's social security number, this field should be restricted from his view.

Job Application

As the hiring manager, Ben needs to be able to update the status of those job applications to specify which candidates should be selected or rejected. However, he should not be able to change the candidate listed on the job application, nor the position to which the candidate is applying, so we'll have to find a way of preventing Ben from updating the lookup fields on job applications.

Review

To make a decision about the candidates who are applying, Ben needs to see the reviews posted by the interviewers, as well as make comments on them if he thinks the interviewer was being too biased in his or her review. Likewise, Ben needs to be able to create reviews so that he can remember his own impressions of the candidates he interviews.

Job Posting

Ben wants to make sure his harder-to-fill positions are visible to the most talented people in the field. The most efficient way to do this is by posting open positions on various employment websites. Given that employment websites have different types of users with varying skill sets, we should give Ben the power to unilaterally create job postings on employment websites, since Ben is the best person to ascertain which skill sets are necessary for his open positions.

Employment Website

If Ben had his way, he would usurp all of the company's budget for posting his open positions on employment websites; therefore, Ben shouldn't be allowed to modify employment website records, as that would let him redefine the company's budget for posting jobs and could lead to an accounting fiasco. Still, we need to make sure that Ben can view employment website records to get an idea of the employment websites with which Universal Containers has accounts, and how much of the budget for that employment website is available.

Here's a summary of the required permissions we need to implement for a hiring manager:

Table 22: Summary of Required Permissions: Hiring Manager

	Position	Candidate	Job Application	Review	Job Posting	Employment Website
Hiring Manager	<ul style="list-style-type: none"> • Read • Create • Edit* 	<ul style="list-style-type: none"> • Read* (No SSN) 	<ul style="list-style-type: none"> • Read • Edit (No lookup fields) 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read* • Create* • Edit* 	<ul style="list-style-type: none"> • Read

* Only for those records that are associated with a position to which the hiring manager has been assigned

Required Permissions for the Interviewer

For our third set of required permissions, let's take a look at Melissa Lee's role as an interviewer. Ben, her manager, likes Melissa to interview candidates for highly technical positions, but doesn't want her speaking with folks who are applying for roles on the user interface team. As a result, Melissa should be able to view only the candidates and job applications to which she's assigned as an interviewer. No such restriction needs to exist on the positions that are out there, but she shouldn't be able to view the minimum and maximum salary values for any of them. Likewise, she shouldn't see the social security number of any candidate, since it's sensitive information that has nothing to do with her job.

Melissa must be able to create and edit her reviews so that she can record her comments about each candidate, but she shouldn't be able to see the reviews of other interviewers—reading them might sway her opinion one way or the other. As with hiring managers and recruiters, Melissa also shouldn't be allowed to delete any records to ensure that public records laws are fulfilled.

Finally, the posting of jobs to employment websites has no bearing on Melissa's responsibilities, so both the employment website and job posting records should be completely off-limits to her.

Here's a summary of the required permissions we need to implement for an interviewer:

Table 23: Summary of Required Permissions: Interviewer

	Position	Candidate	Job Application	Review	Job Posting	Employment Website
Interviewer	• Read (No min/max pay)	• Read* (No SSN)	• Read*	• Read** • Create • Edit**	-	-

* Only for those records that are associated with a position to which the interviewer has been assigned

** Only for those records that the interviewer owns

Required Permissions for the Standard Employee

Employees, such as Manny Damon on the Western Sales Team, are often the best resources for recruiting new hires, even if they are not active hiring managers or interviewers. For this reason, we need to make sure that employees like Manny can view open positions, but that they can't see the values for the positions' minimum and maximum salary fields—otherwise they might tip off friends to negotiate for a position's maximum salary value! Manny also shouldn't be able to view any other records in our Recruiting app.

Here's a summary of the required permissions we need to implement for a standard employee:

Table 24: Summary of Required Permissions: Standard Employee

	Position	Candidate	Job Application	Review	Job Posting	Employment Website
Standard Employee	• Read (No min/max pay)	-	-	-	-	-

So Where Are We Now?

Now that we've gone through the required permissions for each of our four users, let's organize our thoughts by summarizing them in the following table. In the rest of this chapter, we'll figure out how we can use the platform to implement these rules in our Recruiting app.

Table 25: Summary of Required Permissions

	Position	Candidate	Job Application	Review	Job Posting	Employment Website
Recruiter	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit • Delete 	<ul style="list-style-type: none"> • Read • Create • Edit • Delete
Hiring Manager	<ul style="list-style-type: none"> • Read • Create • Edit* 	<ul style="list-style-type: none"> • Read* (No SSN) 	<ul style="list-style-type: none"> • Read • Edit (No lookup fields) 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read* • Create* • Edit* 	<ul style="list-style-type: none"> • Read
Interviewer	<ul style="list-style-type: none"> • Read (No min/max pay) 	<ul style="list-style-type: none"> • Read* (No SSN) 	<ul style="list-style-type: none"> • Read* 	<ul style="list-style-type: none"> • Read** • Create • Edit** 	-	-
Standard Employee	<ul style="list-style-type: none"> • Read (No min/max pay) 	-	-	-	-	-

* Only for those records that are associated with a position to which the hiring manager/interviewer has been assigned

** Only for those records that the interviewer owns



Tip: When implementing the security and sharing rules for your own organization, it's often useful to create a required permissions table like this to organize your thoughts and make sure you don't forget to restrict or grant access to a particular user. You'll

see that we're going to refer back to this table again and again as we go through this chapter.

Data Access Concepts

Before we get started implementing our security and sharing rules, let's quickly take a look at all the ways that we can control data on the platform:

Object-Level Security

The bluntest way that we can control data is by preventing a user from seeing, creating, editing, or deleting any instance of a particular type of object, like a position or review. Object-level access allows us to hide whole tabs and objects from particular users, so that they don't even know that type of data exists.

On the platform, we set object-level access rules with object permissions on user profiles. We'll learn more about profiles in a little bit.

Field-Level Security

A variation on object-level access is field-level access, in which a user can be prevented from seeing, editing, and/or deleting the value for a particular field on an object. Field-level access allows us to hide sensitive information like the maximum salary for a position or a candidate's social security number without having to hide the whole object.

On the platform, we set field-level access rules with the field-level security. We'll also learn more about that shortly.

Record-Level Security

To control data with a little more finesse, we can allow particular users to view an object, but then restrict the individual object records that they're allowed to see. For example, record-level access allows an interviewer like Melissa Lee to see and edit her own reviews, without exposing the reviews of everyone else on her team.

On the platform, we actually have four ways of setting record-level access rules:

- *Organization-wide defaults* allow us to specify the baseline level of access that a user has in your organization. For example, we can make it so that any user can see any record of a particular object to which their user profile gives them access, but so that they'll need extra permissions to actually edit one.
- *Role hierarchies* allow us to make sure that a manager will always have access to the same records as his or her subordinates.

- *Sharing rules* allow us to make automatic exceptions to organization-wide defaults for particular groups of users.
- *Manual sharing* allows record owners to give read and edit permissions to folks who might not have access to the record any other way.

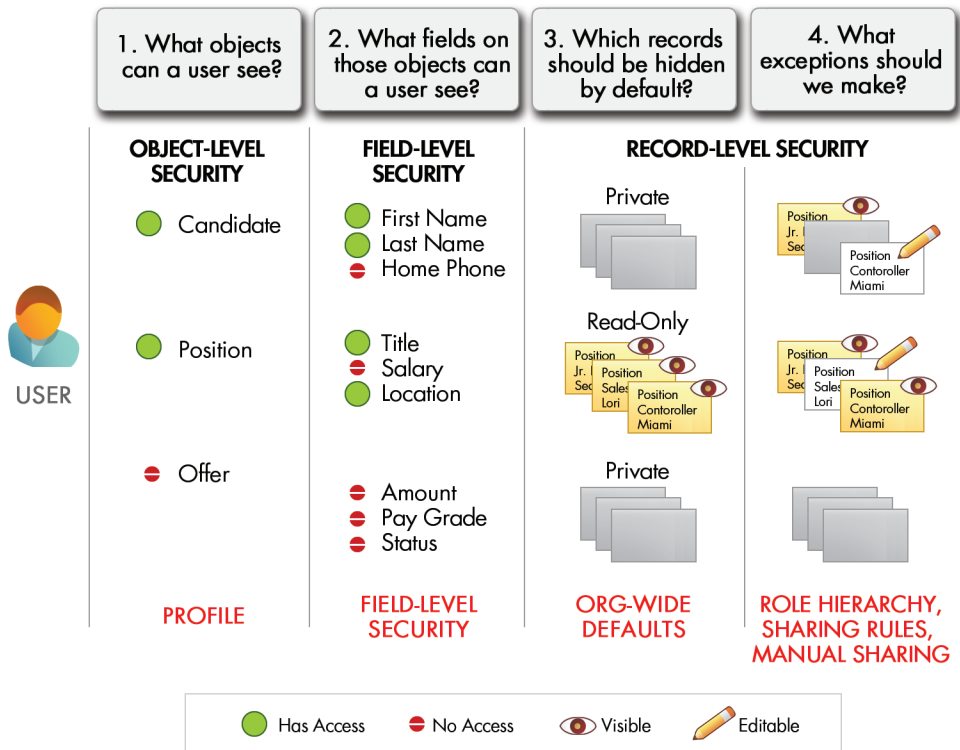


Figure 45: Controlling Data with the Force.com Platform

The combination of all of these sharing and security settings in the platform means that we can easily specify user permissions for an organization of thousands of users without having to manually configure the permissions for each individual. Pretty neat! Now let's get started learning more about each of these methods for controlling data, and actually implementing the security and sharing rules for our app.

Controlling Access to Objects

First let's configure access to our Recruiting app custom objects. As we mentioned previously, we can control whether a user knows that a particular object exists in the app by modifying his or her profile. But exactly what is a profile, and what does it control?

Introducing Profiles

A *profile* is a collection of settings and permissions that determine what a user can do in the platform, kind of like a group in a Windows network, where all of the members of the group have the same folder permissions and access to the same software. Profiles control:

- The objects the user can view, create, edit, and delete
- The object fields the user can view and edit (more on that later!)
- The tabs the user can view in the app
- The standard and custom apps the user can access
- The page layouts a user sees
- The record types available to the user
- The hours during which the user can log in to the app
- The IP addresses from which the user can log in to the app

Profiles are typically defined by a user's job function (for example, system administrator or sales representative), but you can have profiles for anything that makes sense for your organization. A profile can be assigned to many users, but a user can be assigned to only one profile at a time.

Standard Profiles

The platform provides the following set of standard profiles in every organization:

- Read Only
- Standard User
- Marketing User
- Contract Manager
- Solution Manager
- System Administrator

Each of these standard profiles includes a default set of permissions for all of the standard objects available on the platform. For example, users assigned to the Standard User profile can never create, edit, or delete a campaign.

When a custom object is created, most profiles (except those with “Modify All Data”) don't get access to the object. You can find more detailed descriptions of all the standard profiles in the online help, but the important thing to know is that you can never actually edit the permissions on a standard profile. Instead, if you have access to the Enterprise, Unlimited, or Developer Editions of the platform, you can make a copy of a standard profile and then customize that copy to better fit the needs of your organization. That's what we're going to end up doing for our Recruiting app (and as a result, Enterprise, Unlimited, and Developer Editions will be the only editions that the Recruiting app will support).

Custom Profiles in Our Recruiting App

For our app, we've talked about four types of users: recruiters, hiring managers, interviewers, and standard employees. We might just jump the gun and say that this equals four different user profiles, but let's take a closer look.

Recruiters are pretty straightforward—they definitely represent a particular job function, and they need access to different types of data than other users. They need their own profile.

A hiring manager, however, is not exactly a single type of position. For most organizations, a hiring manager in the Sales department will almost certainly need access to a different type of data than a hiring manager in Engineering. However, for the purposes of our app, sales managers and software managers still need the same types of access to recruiting data—reviews, candidates, positions, job applications, job postings, and employment websites. Let's keep this as a single profile for now, but if incorporating our app into an organization with other CRM functionality, we'll need to suggest as a best practice that the hiring manager permissions for recruiting-related data need to be replicated for any profile to which hiring managers belong.

Finally, let's look at interviewers and standard employees. Neither one of these user types reflects a particular job function, and when you think about it, just about anyone in an organization might be called upon to perform an interview. Let's define a single profile for a standard employee and find a way to grant interviewers access to the records that they need through some other mechanism. (Hint: we can use a combination of organization-wide defaults and sharing rules to make this work.)

Try It Out: Creating the Recruiter Profile

All right—we're finally ready to dig into the app and create our first profile! Let's start with the Recruiter profile.

1. Click *Your Name* ► **Setup** ► **Manage Users** ► **Profiles**.

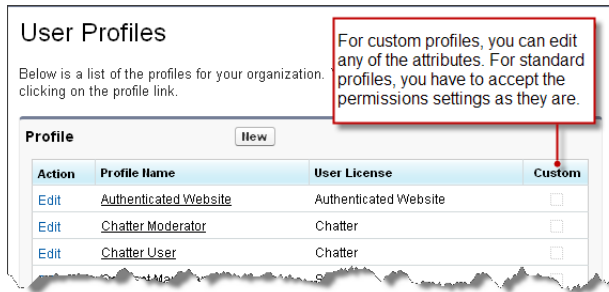


Figure 46: Standard Profiles

Here you should see the list of standard profiles that we talked about earlier. After we create our custom profiles, they'll also show up in this list.

First, we can quickly tell which profiles we can play with by looking at the `Custom` column—if it's checked, that means it's a custom profile and we can edit anything about it. If that column is not checked, we can still click the **Edit** link; we just can't modify any of the permission settings. (What does that leave for us to edit on a standard profile? Well, we can choose which tabs should appear at the top of a user's page, and we can also select the apps that are available in the Force.com app menu in the top-right corner of the page.)

2. Create a new profile named Recruiter based on the Standard User profile.

There are actually two ways of doing this—we can either click **New**, select an existing profile to clone, name it, and click **Save**, or we can simply click **Clone** in the detail page of the profile that we want to copy, name it, and click **Save**. Ultimately, it's the same number of clicks, so choose the method you like best. Standard User is the profile that most closely resembles what we want our new Recruiter profile to look like, so it's a good starting point.

3. In the new Recruiter profile's detail page, click **Edit**.

The Recruiter edit page should look and function exactly like the Standard User profile edit page except with one important difference: you have the ability to modify any of the permission settings.

- In the Custom App Settings area, make the Recruiting app visible to users assigned to the Recruiter profile, as shown in the following screenshot.

The screenshot shows a 'Custom App Settings' window with a header bar containing a red exclamation mark icon and the text '= Required Information'. Below the header, there are two columns of settings, each with a 'Visible' checkbox and a 'Default' radio button. The 'Visible' checkboxes are checked for Sales, Marketing, Service & Support, and Recruiting. The 'Default' radio buttons are unselected for Sales, Marketing, and Service & Support, but selected for Recruiting.

App Name	Visible	Default
Sales	<input checked="" type="checkbox"/>	<input type="radio"/>
Marketing	<input checked="" type="checkbox"/>	<input type="radio"/>
Service & Support	<input checked="" type="checkbox"/>	<input type="radio"/>
Recruiting	<input checked="" type="checkbox"/>	<input checked="" type="radio"/>

Figure 47: Profile Custom App Settings Area



Tip: You can also give this profile access to any of the other available apps as well. Every profile needs to have at least one visible app.

When an app is visible, a user can select it from the Force.com app menu at the top-right corner of the page. Be aware, however, that even if an app is visible, the app's tabs won't show up unless a profile has permissions to view the tabs and permission to view the associated object. (We'll set both of those permissions lower down in the Profile edit page.)

- Select **Default** next to the Recruiting app.

Making this selection means that the Recruiting app will be displayed when a user logs in. You'll notice that when you select an app as the default, its `visible` checkbox is automatically selected, because it doesn't make sense for an app to be the default if it's not visible to the user.

- In the Tab Settings area, select **Default On** for the Positions, Job Applications, Candidates, and Employment Websites tabs.



Tip: You can choose whether you want other tabs to be displayed based on the additional apps that you made visible in the last step.

For the purposes of our Recruiting app, all of our custom recruiting tabs are on by default. For any other tabs that you select, you can choose which should be displayed on top of the user's page (Default On), hidden from the user's page but available when he or she clicks the All Tabs tab on the far right (Default Off), or completely hidden from the user (Tab Hidden).

Realize that even if you completely hide a tab, users can still see the records that would have appeared in that tab in search results and in related lists. (To prevent a user from accessing data, we have to set the proper restrictions in the Standard and Custom Object Permissions areas lower down in the Profile edit page—we'll get there shortly!)

The `Overwrite users' personal tab customizations` setting appears if you have an organization that's currently in use and you want to make sure your existing users are viewing the tabs that you've selected. You don't need to select this for our app because we're defining

a brand-new profile and no one has personalized his or her tab visibility settings yet. However, if you do want to select this option at some point in the future, just make sure you're not going to annoy your users by deleting all of their customizations!

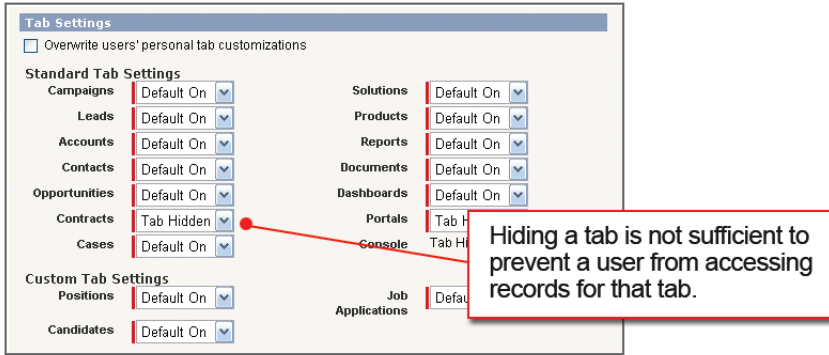


Figure 48: Profile Tab Settings Area

Just below the Tab Settings area, the Administrative and General User Permissions areas of the profile allow you to grant special access to features and functionality that don't map directly to particular objects. None of these permissions affects our Recruiting app, but you can learn more about them in the online help.

It's time to move on to the object-level permissions.

7. In the Custom Object Permissions area, specify the object-level permissions for our Recruiter profile according to the following table.

Table 26: Summary of Required Permissions: Recruiter

	Positions	Candidates	Job Applications	Reviews	Job Posting	Employment Website
Recruiter	<ul style="list-style-type: none"> • Read • Create • Edit • View • All 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit • Delete • View All • Modify All 	<ul style="list-style-type: none"> • Read • Create • Edit • Delete • View All • Modify All



Tip: Depending on the apps that you made visible previously, you can also set additional object permissions on standard or other custom objects.

Since there are no instances when a recruiter should be allowed to delete positions, candidates, job applications, and reviews, we should make sure that the object-level permissions for deletion are turned off for these objects. Also, make sure the “View All” and “Modify All” permissions are only selected for job postings and employment websites. These are special kinds of object permissions that we’ll discuss later in this chapter.

	Basic Access						Data Administration			Basic Access						Data Administration	
	Read	Create	Edit	Delete	View All	Modify All	View All	Modify All		Read	Create	Edit	Delete	View All	Modify All	View All	Modify All
	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Candidates	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Positions	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Employment Websites	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Reviews	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Google Campaigns	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Search Phrases	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Job Applications	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	SFGA Version	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Job Postings	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Text Ads	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Keywords	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>									

Figure 49: Recruiter Profile Permissions

By restricting the power to delete recruiting-related objects here, recruiters will *never* be able to delete these objects. However, the fact that we're granting recruiters permission to create, read, or edit our recruiting objects does not necessarily mean that recruiters will be allowed to read or edit *every* recruiting object record. Why?

Here we see the result of two really important concepts in the platform:

- The permissions on a record are always evaluated according to a combination of object-, field-, and record-level permissions.
- When object- versus record-level permissions conflict, the most restrictive settings win.

What this means is that even though we are granting this profile create, read, and edit permissions on the recruiting objects, if the record-level permissions for an individual recruiting record prove to be more restrictive, those will be the rules that will define what a recruiter can access.

For example, our new profile gives a recruiter permission to create, edit, and view reviews. However, if we set organization-wide defaults for reviews to Private (a record-level permission), our recruiter will be allowed to edit and view only his own reviews, and not the reviews owned by other users. We'll learn more about record-level permissions later and go through more examples of how they work with the object-level ones, but for now, just understand that object-level permissions are only one piece of the puzzle.

8. Click **Save** to create your profile and return to the profile detail page.

Congratulations! We're done with our first profile. As you can see, it really wasn't that hard, because we'd already analyzed our required permissions and knew what objects recruiters will need access to. In the next section, let's quickly finish up our other two profiles and then move on to field-level security.

Beyond the Basics

Did you know that you can use an improved user interface to manage profiles?

Say you manage a lot of profiles and you'd like a more streamlined experience. With the enhanced profile user interface, you can easily navigate, search, and modify profile settings.

To find out more, see “Enhanced Profile User Interface Overview” in the Salesforce online help.

Try It Out: Creating More Profiles

Now that we've created our Recruiter profile, let's finish up with profiles for hiring managers and standard employees. As we mentioned earlier, these profiles are probably too generic to be used in a company that's using the platform for other, non-recruiting functionality, but for our purposes, they'll work just fine for now.

To make each profile, go ahead and follow the steps that we outlined in the previous section. The important things to remember for these profiles are:

- The profiles should be named Hiring Manager and Standard Employee and should be based on the standard profile that best fits your needs. We used Standard User for our Recruiter profile, and that's probably a good one for Standard Employee as well. However, for your organization you might find that Hiring Manager more closely resembles the Contract or Solution Manager profile instead. It's up to you.
- The Recruiting app for both profiles must be set to `Visible`.
- For the Hiring Manager profile, set the tabs for Positions, Candidates, Job Applications, and Employment Websites to `Default On`.
- For the Standard Employee profile, set the tabs for Positions, Candidates, and Job Applications to `Default On`, but set the tab for Employment Website to `Tab Hidden`.
- Standard and custom object permissions should reflect the required permissions that we worked on before.

To refresh our memories, let's take a look at our required permissions summary table:

Table 27: Summary of Required Permissions: Hiring Manager, Interviewer, and Standard Employee

	Positions	Candidates	Job Applications	Reviews	Job Postings	Employment Websites
Hiring Manager	<ul style="list-style-type: none"> • Read • Create • Edit* 	<ul style="list-style-type: none"> • Read* (No SSN) 	<ul style="list-style-type: none"> • Read • Edit (No lookup fields) 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read* • Create* • Edit* 	<ul style="list-style-type: none"> • Read
Interviewer	<ul style="list-style-type: none"> • Read (No min/max pay) 	<ul style="list-style-type: none"> • Read* (No SSN) 	<ul style="list-style-type: none"> • Read* 	<ul style="list-style-type: none"> • Read** • Create • Edit** 	-	-
Standard Employee	<ul style="list-style-type: none"> • Read (No min/max pay) 	-	-	-	-	-

* Only for those records that are associated with a position to which the hiring manager/interviewer has been assigned

** Only for those records that the interviewer owns

This table is a little confusing right now, because Interviewer and Standard Employee are still separated into two different users. Let's go ahead and combine them into a single Standard Employee user so that it's a little easier to see what object-level permissions we need to grant. It's easy to do, because Interviewers and Standard Employees have the same permissions on the Position object, and we already have asterisks on Candidates, Job Applications, and Reviews that ensures these users won't look at anything to which they're not assigned as an interviewer:

Table 28: Summary of Required Permissions: Combining Interviewers and Standard Employees

	Positions	Candidates	Job Applications	Reviews	Job Postings	Employment Websites
Hiring Manager	<ul style="list-style-type: none"> • Read • Create • Edit* 	<ul style="list-style-type: none"> • Read* (No SSN) 	<ul style="list-style-type: none"> • Read* (No lookup fields) • Edit (No lookup fields) 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read* • Create* • Edit* 	<ul style="list-style-type: none"> • Read
Standard Employee (Interviewer)	<ul style="list-style-type: none"> • Read (No min/max pay) 	<ul style="list-style-type: none"> • Read* (No SSN) 	<ul style="list-style-type: none"> • Read* 	<ul style="list-style-type: none"> • Read** • Create • Edit** 	-	-

* Only for those records that are associated with a position to which the hiring manager/standard employee has been assigned

** Only for those records that the standard employee owns

Great, but what about all of those asterisks and restrictions on visible fields? Do we need to take those into account when we set our object-level permissions?

Not at all. Those asterisks and field restrictions represent record- and field-level security settings that we're going to have to specify elsewhere in our app. The only things we need to care about here are the permissions that these users will need to have access to at least *some* of the time—that's the whole point of object-level permissions:

- For hiring managers, that's create, read, and edit on Positions and Reviews, read on Candidates and Employment Websites, and read and edit on Job Applications
- For standard employees, that's read on Positions, Candidates, and Job Applications, and create, read, and edit on Reviews

Custom Object Permissions													
	Basic Access				Data Administration			Basic Access				Data Administration	
	Read	Create	Edit	Delete	View All	Modify All		Read	Create	Edit	Delete	View All	Modify All
Candidates	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Positions	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Employment Websites	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Reviews	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Google Campaigns	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Search Phrases	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Job Applications	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	SFGA Version	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Job Postings	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Text Ads	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Keywords	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>							

Figure 50: Hiring Manager Profile Permissions

Custom Object Permissions													
	Basic Access				Data Administration			Basic Access				Data Administration	
	Read	Create	Edit	Delete	View All	Modify All		Read	Create	Edit	Delete	View All	Modify All
Candidates	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Positions	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Employment Websites	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Reviews	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Google Campaigns	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Search Phrases	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Job Applications	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	SFGA Version	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Job Postings	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Text Ads	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Keywords	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>							

Figure 51: Standard Employee Profile Permissions

Fantastic! We've just finished defining profiles and object-level permissions for all the users in our Recruiting app. However, that's still just one piece of the security and sharing puzzle—we still need to make sure that sensitive data on these objects is protected from users who don't need access, and then we need to drill down on the actual records that each user should be allowed to view and edit.

Controlling Access to Fields

Now that we've restricted access to objects as a whole, it's time to use a finer-toothed comb to manage the security of individual object fields. These are the settings that allow us to protect sensitive fields such as a candidate's social security number without having to hide the fact that the candidate object even exists.

Introducing Field-Level Security

In the platform, we control access to individual fields with *field-level security*. Field-level security controls whether a user can see, edit, and delete the value for a particular field on an object.

Unlike page layouts, which only control the visibility of fields on detail and edit pages, field-level security controls the visibility of fields in any part of the app, including related lists, list views, reports, and search results. Indeed, in order to be absolutely sure that a user can't access a particular field, it's important to use the field-level security page for a given object to restrict access to the field. There are simply no other shortcuts that will provide the same level of protection for a particular field.

Field-Level Security in Our Recruiting App

Just to refresh our memories about what field-level security settings we need for our Recruiting app, let's take another look at our required permissions in the following table. We'll keep them organized by recruiter, hiring manager, and standard employee, because it turns out that (surprise!) field-level security settings are closely related to profiles:

Table 29: Revised Summary of Required Permissions

	Position	Candidate	Job Application	Review	Job Posting	Employment Website
Recruiter	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit • Delete 	<ul style="list-style-type: none"> • Read • Create • Edit • Delete
Hiring Manager	<ul style="list-style-type: none"> • Read • Create • Edit* 	<ul style="list-style-type: none"> • Read* (No SSN) 	<ul style="list-style-type: none"> • Read • Edit (No lookup fields) 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read* • Create* • Edit* 	<ul style="list-style-type: none"> • Read
Standard Employee	<ul style="list-style-type: none"> • Read (No min/max pay) 	<ul style="list-style-type: none"> • Read* (No SSN) 	<ul style="list-style-type: none"> • Read* 	<ul style="list-style-type: none"> • Read** • Create • Edit** 	-	-

* Only for those records that are associated with a position to which the hiring manager/interviewer has been assigned

** Only for those records that the interviewer owns

For field-level security settings, we'll first zero in on those rules that include field restrictions in parentheses, specifically:

- On the Position object, hide minimum and maximum pay from standard employees
- On the Candidate object, hide social security numbers from hiring managers and standard employees
- On the Job Application object, make the `Position` and `Candidate` lookup fields read-only for hiring managers.

So let's get down to it!

Accessing Field-Level Security Settings

So where do you access field-level security settings? It turns out that there are actually two ways to get to those settings—either through a profile's detail page, or by choosing **Your Name** ► **Setup** ► **Security Controls** ► **Field Accessibility**.

The path that you choose depends on whether you're focused solely on field-level security, or whether you want to fiddle with page layout settings at the same time. The first way is the simplest and requires the least number of clicks, but the second is sometimes more convenient when you're building a new app, because you can perform two related tasks in the same page.

Because we have two different field rules that we need to implement for our Recruiting app, we'll define one rule with the first method, and the other with the second.

Try It Out: Restricting Access to a Position's Minimum and Maximum Salary Fields

Let's get started actually implementing the first of our field-level security rules: *On Positions, hide minimum and maximum pay from standard employees*. We'll define this rule by accessing field-level security settings through the Standard Employee profile's detail page.

1. Click **Your Name** ► **Setup** ► **Manage Users** ► **Profiles**, and then select the Standard Employee profile.

Profile [Help for this Page](#)

Standard Employee

Users with this profile have the permissions and page layouts listed below. Administrators can change a user's profile by editing that user's personal information.

If your organization uses Record Types, use the Edit links in the Record Type Settings section below to make one or more record types available to users with this profile.

[Login IP Ranges \[0\]](#) | [Enabled Apex Class Access \[0\]](#) | [Enabled Visualforce Page Access \[0\]](#)

Profile Detail [Edit](#) [Clone](#) [Delete](#) [View Users](#)

Name	Standard Employee		
User License	Salesforce	Custom Profile	<input checked="" type="checkbox"/>
Description			
Created By	Jane Smith , 7/30/2010 10:49 AM	Modified By	Jane Smith , 7/30/2010 11:03 AM

Console Settings

Console Layout	[Edit]
----------------	------------------------

Page Layouts

Standard Object Layouts

Home Page Layout	DE Default [View Assignment]	Event	Event Layout [View Assignment]
------------------	---	-------	---

Figure 52: Standard Employee Profile Detail Page

The first thing you'll notice about the Standard Employee profile's detail page is that it includes several more areas than the edit page that we originally used to define the profile. These additional areas include Page Layouts (which we learned about earlier), Field-Level Security, Record Type Settings, Login Hours, and Login IP Ranges. Although we won't go into detail in this book about how to use any areas other than Field-Level Security (and record types later on), they're part of what makes a profile so powerful in our application. You can learn more about them in the online help.

2. In the Field-Level Security area, click **View** next to the Position object.
3. Click **Edit**.

Position Field-Level Security for profile
Standard Employee Help for this Page ?

Save Cancel

Field Name	Field Type	Visible	Read-Only
Apex	Checkbox	<input checked="" type="checkbox"/>	<input type="checkbox"/>
C#	Checkbox	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Close Date		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Created By		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Days Open	Number	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Educational Requirements	Long Text Area	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Functional Area		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Hire By		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Hiring Manager	Lookup	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Java	Checkbox	<input checked="" type="checkbox"/>	<input type="checkbox"/>
JavaScript	Checkbox	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Job Description	Long Text Area	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Job Level	Picklist	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Last Modified By	Lookup	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Location		<input checked="" type="checkbox"/>	<input type="checkbox"/>
Max Pay		<input type="checkbox"/>	<input type="checkbox"/>
Min Pay		<input type="checkbox"/>	<input type="checkbox"/>
Open Date	Date	<input checked="" type="checkbox"/>	<input type="checkbox"/>

If both boxes are selected, the field is Read-Only.

If only the Visible box is selected, the field is editable.

If neither box is selected, the field is hidden from the user.

Figure 53: Field-Level Security Edit Page

Here we see security settings for all of the fields on the Position object, including `Min Pay` and `Max Pay`, the two fields that we want to restrict. You'll notice that some field-level security settings on some fields cannot be modified—this is because either they are system-generated fields or they act as lookup relationship fields (foreign keys) to other records.

Since the security settings checkboxes can be a little bit confusing, let's do a quick exercise to map their values (`Visible` and `Read-Only`) to the three logical permission settings for a field: “Hidden,” “Read Only,” and “Editable”:

Table 30: Field-Level Permission Mappings

Permission	Visible	Read-Only
Hidden		
Read Only	X	X
Editable	X	

After doing this exercise, it's easy to see that most fields are editable, because their `Visible` checkbox is the only one selected. To restrict a field from ever being viewed by a user, all we have to do is deselect both checkboxes.

4. Next to the `Max Pay` field, deselect `Visible`.
5. Next to the `Min Pay` field, deselect `Visible`.
6. Click **Save**.

We're done!

Try It Out: Restricting Access to a Candidate's Social Security Number

For our second field-level security setting (*On Candidates, hide social security numbers from hiring managers and standard employees*), let's implement it using the Field Accessibility page. You'll see that this method is very similar to the last, but that the Field Accessibility page gives us more power and control over how we define our settings.

1. Click **Your Name** ► **Setup** ► **Security Controls** ► **Field Accessibility**, and then select the Candidate object.

Notice that with the Field Accessibility tool, we start off by choosing the relevant object, rather than a user profile. Then, we can choose to set field-accessibility either by selecting a single field and then seeing its security settings for every profile (**View by Fields**), or by selecting a profile and then seeing security settings for every field (**View by Profile**).

2. Click **View by Fields**, and then select SSN from the `Field` drop-down list.

Choose your view

[View by Fields](#) *Current View*

Choose a field to view a table of field accessibility for different profiles and record types.

Field:

[View by Profiles](#)

Use this option to choose a profile and view a table of field accessibility for different record types.

Field accessibility for Field: SSN

Click on a cell in the table below to change the field's accessibility.

Profiles	Field Access
Contract Manager	Editable
Custom: Marketing Profile	Editable
Custom: Sales Profile	Editable
Custom: Support Profile	Editable
Hiring Manager	Editable
Marketing User	Editable
Read Only	Editable
Recruiter	Editable
Solution Manager	Editable
Standard Employee	Editable
Standard User	Editable
System Administrator	Editable <small>Field is editable because of Page Layout</small>

Figure 54: Field Accessibility Page

The table that appears shows us the accessibility settings for social security number for each profile. If you move your mouse over the value for `Field Access`, hover text indicates whether the security setting is the result of a page layout setting or a field-level security setting.

Let's first set our field-level security for hiring managers.

3. Next to the Hiring Manager profile, click **Editable**.

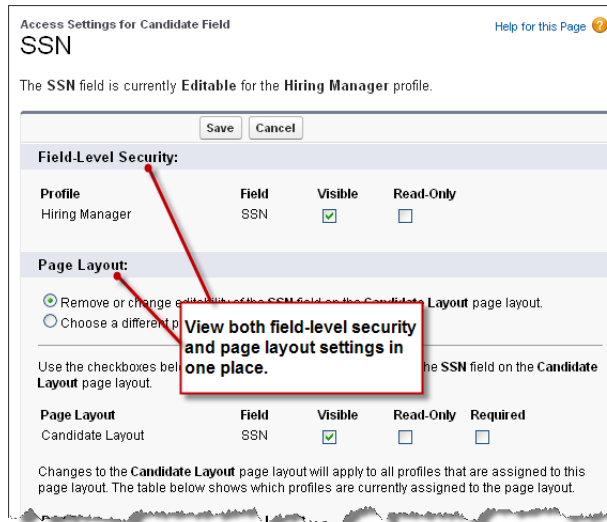


Figure 55: Access Settings Page

By clicking the value in a profile's **Field Access** column, we finally get to see the real reason why the Field Accessibility page is so convenient: we can view and edit a field's field-level security and page layout settings in the same page.

The Field-Level Security area includes the same checkboxes that we discussed previously—to hide the social security number from hiring managers, we can simply deselect the **Visible** checkbox as we did with the **Min Pay** and **Max Pay** fields on positions.

Before we do that, however, take a look at the Page Layout area. Here you can either change the visibility and editability of the field on the page layout, or you can choose a different page layout for the Hiring Manager profile altogether. Since we are focused on the security of the SSN field in this exercise we don't need to touch any of the Page Layout area settings. However, you should remember that this tool is here for the future, especially if you are ever trying to figure out why a field is or is not visible to a particular user.

4. In the Field-Level Security area, deselect **Visible**.
5. Click **Save**.

To finish up, all we need to do is repeat the process for the Standard Employee profile.

6. Next to the Standard Employee profile, click **Editable**.
7. In the Field-Level Security area, deselect **Visible**.
8. Click **Save**.

We have just one more field-level security setting to make! In this next one, we'll leave the field values visible, but prevent them from being edited by hiring managers.

Try It Out: Setting Lookup Fields to Read-Only on a Job Application for Hiring Managers

Our final field-level security rule will let hiring managers see which candidate and position are associated with each job application, but prevent hiring managers from changing those fields.

1. Click **Your Name** ► **Setup** ► **Manage Users** ► **Profiles**, and then select the Hiring Manager profile.
2. In the Field-Level Security area, click **View** next to the Job Application object.
3. Click **Edit**.
4. Next to the `Candidate` field, select `Read-Only`.
5. Next to the `Position` field, select `Read-Only`.
6. Click **Save**.

All done! We've just finished the second piece of our security and sharing puzzle by defining field-level security for the sensitive fields in our Recruiting app. Now, for the final (and most complicated) piece of the puzzle, we need to specify the individual records to which each user needs access. We need to protect our data without compromising any employee's ability to perform his or her job.

Controlling Access to Records

By setting object and field-level access rules for each of our three user profiles, we have effectively defined all of the objects and fields that any one of our Recruiting app users can access. In this section, we'll focus on setting permissions for the actual records themselves. Should our users have open access to every record, or just a subset? If it's a subset, what rules should determine whether the user can access them? We'll use a variety of platform security and sharing tools to address these questions and make sure we get it right.

Introducing Organization-Wide Defaults

When dealing with record-level access settings, the first thing we need to do is to determine the organization-wide defaults (commonly called “org-wide defaults”) for each object in our Recruiting app. Also called a sharing model, org-wide defaults specify the baseline level of access that the most restricted user should have. We'll use org-wide defaults to lock down our

data to this most restrictive level, and then we'll use our other record-level security and sharing tools (role hierarchies, sharing rules, and manual sharing) to open up the data to other users who need to access it.

Org-Wide Defaults in Our Recruiting App

To determine the org-wide defaults that we'll need in our Recruiting app, we need to answer the following questions for each object:

1. Who is the most restricted user of this object?
2. Is there ever going to be an instance of this object that this user shouldn't be allowed to see?
3. Is there ever going to be an instance of this object that this user shouldn't be allowed to edit?

Based on our answers to these questions, we can determine the sharing model that we need for that object as illustrated in the following diagram.

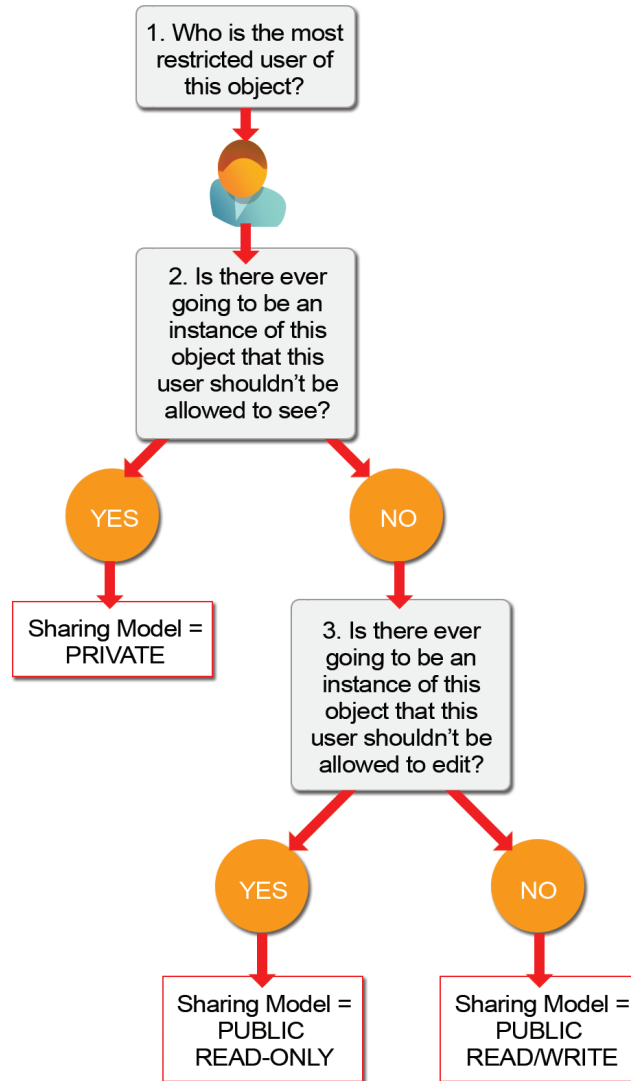


Figure 56: Determining the Sharing Model for an Object

For example, let's consider the Position object in our recruiting app. To refresh our memories, here's our table of required permissions:

Table 31: Revised Summary of Required Permissions

	Position	Candidate	Job Application	Review	Job Posting	Employment Website
Recruiter	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read • Create • Edit • Delete 	<ul style="list-style-type: none"> • Read • Create • Edit • Delete
Hiring Manager	<ul style="list-style-type: none"> • Read • Create • Edit* 	<ul style="list-style-type: none"> • Read* (No SSN) 	<ul style="list-style-type: none"> • Read • Edit (No lookup fields) 	<ul style="list-style-type: none"> • Read • Create • Edit 	<ul style="list-style-type: none"> • Read* • Create* • Edit* 	<ul style="list-style-type: none"> • Read
Standard Employee	<ul style="list-style-type: none"> • Read (No min/max pay) 	<ul style="list-style-type: none"> • Read* (No SSN) 	<ul style="list-style-type: none"> • Read* 	<ul style="list-style-type: none"> • Read** • Create • Edit** 	-	-

* Only for those records that are associated with a position to which the hiring manager/interviewer has been assigned

** Only for those records that the interviewer owns

Now let's go through and answer our list of questions for the Position object:

Table 32: Determining Org-Wide Defaults for the Standard Employee Profile

Question	Answer
1. Who is the most restricted user of this object?	A member of the Standard Employee profile. All that they're allowed to do is view a position.
2. Is there ever going to be an instance of this object that this user shouldn't be allowed to see?	No. Although the values for the minimum and maximum pay are hidden from standard employees, they're still allowed to view all position records.

Question	Answer
3. Is there ever going to be an instance of this object that this user shouldn't be allowed to edit?	Yes. Standard employees aren't allowed to edit any position record.

According to our flowchart, answering “Yes” to question #3 means that the sharing model for the Position object should be set to Public Read-Only.

The same is true for the Employment Website and Job Posting objects, except hiring managers are the most restricted users instead of standard employees. We want to allow hiring managers to view all employment website and job posting records without being able to edit them, so the answer to the second question is “No” while the answer to the third question is “Yes;” therefore, the sharing model for the Employment Website and Job Posting objects should be Public Read-Only.

Going through the rest of our recruiting objects required permissions, we can easily figure out their sharing models, too. The Standard Employee profile is the most restricted user for each object, and there are going to be candidate, job application, and review records that particular employees won't be able to view. Consequently, the sharing model for the Candidate, Job Application, and Review objects should all be set to Private.

Try It Out: Setting Org-Wide Defaults

Now that we've figured out the org-wide defaults for each of our recruiting objects, let's go ahead and implement them in our Recruiting app.

1. Click **Your Name** ► **Setup** ► **Security Controls** ► **Sharing Settings**. If you see an introductory splash page, click **Set Up Sharing** at the bottom of the page to skip to the actual tool.

The Sharing Settings page is where we control both org-wide defaults and sharing rules. We'll talk more about this page when we talk about sharing rules a little further down. For now, let's just edit our org-wide default settings.

2. In the Organization Wide Defaults area, click **Edit**.

Organization-Wide Sharing Defaults Edit [Help for this Page](#)

Edit your organization-wide sharing defaults below. Changing these defaults will cause all sharing rules to be recalculated. This could require significant system resources and time depending on the amount of data in your organization.

Object	Default Access	Grant Access Using Hierarchies
Lead	Public Read/Write/Transfer	<input checked="" type="checkbox"/>
Account, Contract and Asset	Public Read/Write	<input checked="" type="checkbox"/>
Contact	Controlled by Parent	<input checked="" type="checkbox"/>
Opportunity	Public Read/Write	<input checked="" type="checkbox"/>
Case	Public Read/Write/Transfer	<input checked="" type="checkbox"/>
Campaign	Public Full Access	<input checked="" type="checkbox"/>
Activity	Private	<input checked="" type="checkbox"/>
Calendar	Hide Details and Add Events	<input checked="" type="checkbox"/>
Price Book	Use	<input checked="" type="checkbox"/>
Candidate	Private	<input checked="" type="checkbox"/>
Employment Website	Public Read Only	<input checked="" type="checkbox"/>
Job Application	Public Read Only	<input checked="" type="checkbox"/>
Position	Public Read/Write	<input checked="" type="checkbox"/>

Some standard objects use different org-wide default options.

Custom object org-wide default options include Private, Public Read Only, or Public Read/Write.

Figure 57: Org-Wide Defaults Edit Page

This page controls the org-wide defaults for every object in our organization. You'll notice that some standard objects (like leads and calendars) use a different set of org-wide default values than we have available for our custom recruiting objects. You can learn more about them in the online help. For now, let's just set our recruiting objects to the org-wide defaults that we decided on in the last section.

3. Next to `Position` and `Employment Website`, select `Public Read Only`.
4. Next to `Candidate` and `Job Application`, select `Private`.

Right about now, you're probably wondering why you can't set the org-wide defaults for the `Review` and `Job Posting` objects. The reason is that those objects are on the detail side of master-detail relationships, and, as mentioned in the last chapter, a detail record automatically inherits the sharing setting of its parent. So in our app, the `Review` object is automatically set to `Private`, and the `Job Posting` object is automatically set to `Public Read Only`.

You also might be wondering about the `Grant Access Using Hierarchies` column of checkboxes. Leave these selected for now. We'll discuss hierarchies in the next section.

5. Click **Save**.

Easy! Now that we've locked down our data with org-wide defaults, users are currently allowed to work on only candidate, job application, and review records that they own, and are allowed

to view position, employment website, and job posting records that anyone owns. Because those settings are way too restrictive for any user to get any benefit out of our app, now we need to use role hierarchies, sharing rules, and manual sharing to open up candidate, job application, and review record access to those employees who'll need it.

Introducing Role Hierarchies

The first way that we can share access to records is by defining a role hierarchy. Similar to an org chart, a role hierarchy represents a level of data access that a user or group of users needs. Users assigned to roles near the top of the hierarchy (normally the CEO, executives, and other management) get to access the data of all the users who fall directly below them in the hierarchy. The role hierarchy ensures that a manager will always have access to the same data as his or her employees, regardless of the org-wide default settings. Role hierarchies also helpfully define groups of users who tend to need access to the same types of records—we'll use these groups later when we talk about sharing rules.

To illustrate, let's take a look at a portion of the role hierarchy for Universal Containers:

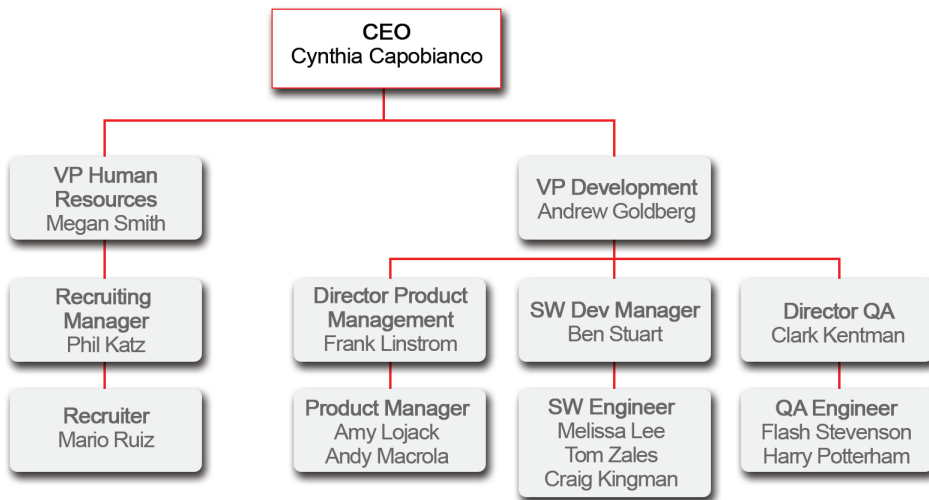


Figure 58: Universal Containers Role Hierarchy

Role hierarchies don't necessarily need to match your org chart exactly. Instead, each role in the hierarchy should just represent a level of data access that a user or group of users needs. For example, suppose your organization employs a corporate lawyer who needs to access all of the records in the app. One easy way to accomplish this is by assigning the lawyer to the CEO role in your organization's role hierarchy. Since the CEO role is placed at the top of the hierarchy, anyone assigned to that role automatically gets full access to any record in the

organization. It doesn't matter that technically the lawyer appears below the CEO in the regular org chart.

Also, role hierarchies don't necessarily need to apply to all of your custom objects. You can use the `Grant Access Using Hierarchies` checkbox to enable and disable record access through hierarchies on an object-by-object basis when you set your org-wide defaults. For our Recruiting app, however, we want our role hierarchy to apply to all of our custom objects, so leave all of the `Grant Access Using Hierarchies` checkboxes selected.

Comparing Roles and Profiles

Although it's easy to confuse profiles with roles, they actually control two very different things.

As we learned earlier in this chapter, profiles control a user's object- and field-level access permissions. Indeed, a user can't be defined without being assigned to a particular profile, since the profile specifies the apps and tabs that appear when he or she logs in, among a number of other useful things.

Roles, on the other hand, primarily control a user's record-level access permissions through role hierarchy and sharing rules. Although a role assignment isn't exactly required when we define a user, it would be foolish of us *not* to assign a role since it makes it so much easier to define our record-level permissions. Indeed, trying to define record-level permissions without assigning a role to a user would be a lot like trying to travel from New York to San Francisco by car when there's an airplane available—there's just a much more efficient way of doing it!

Because profiles control object- and field-level access whereas roles influence record-level access, a user is typically assigned to one of each. To help you remember which controls what, remember: **Roles control Records**.

Role Hierarchies in Our Recruiting App

Given the Universal Containers role hierarchy that's pictured in the Universal Containers Role Hierarchy image, let's think about how implementing this hierarchy will open up certain kinds of record-level permissions to various users of our Recruiting app. Remember, since defining our org-wide defaults, our hiring managers are currently allowed to only view all position, job posting, and employment website records, and to view and update other recruiting records that they own. That doesn't make our app all that useful. However, once we implement our role hierarchy, we'll automatically grant several kinds of record-level permissions to various users. For example:

- The CEO, Cynthia Capobianco, will be able to view and update every record that anyone else in the organization can view and update.
- The VP of Development, Andrew Goldberg, will be able to view and update any record that his managers or his managers' employees can view or update.
- The VP of Human Resources, Megan Smith, will be able to view and update any record that Phil Katz, her recruiting manager, or Mario Ruiz, Phil's recruiter, can view and update.
- The Recruiting Manager, Phil Katz, will be able to view and update any record that is owned by Mario Ruiz, his recruiter.
- The Software Development manager, Ben Stuart, will be able to view and update any record that is owned by Melissa Lee, Tom Zales, or Craig Kingman, his software engineers.
- The director of QA, Clark Kentman, will be able to view and update any record that is owned by Flash Stevenson or Harry Potterham, his QA Engineers.
- The director of Product Management, Frank Linstrom, will be able to view and update any record that is owned by Amy Lojack or Andy Macrola, his product managers.

As we can see, the role hierarchy is very powerful in opening up data for people high up in the role hierarchy tree! However, let's look at some of the gaps that we still have in our record-level permissions:

- Megan Smith (and her whole recruiting team) won't be able to view any reviews that are owned by members of Andrew Goldberg's Development team because she doesn't have a direct line down to any Development roles in the role hierarchy.
- Ben Stuart, the software development manager, also won't be able to see any reviews that were written by members of the QA or Product Management groups, even if QA engineers or product managers interviewed candidates for a software engineering position in his group.
- Melissa Lee, a software engineer, won't be able to see the records for candidates that she's supposed to interview.

Clearly we'll need to use other record-level sharing methods to open up data between peers in the same group, and also between groups that appear in different branches of the role hierarchy (we'll get to those later in this chapter). However, the role hierarchy does give us a good start toward opening up record access, so let's take a look now at how to define it.

Try It Out: Defining a Role Hierarchy

Implementing a role hierarchy in the platform is easy once you have an idea of what the hierarchy should look like. It's best to just start with your company's org chart and then consolidate different job titles into single roles wherever possible. For example, if Ben Stuart's software development group has a staff software engineer and a junior software engineer, these positions can be consolidated into a single Software Engineer role in the hierarchy.

Once that's all squared away, we can get started actually defining the role hierarchy itself. For our exercise, we'll go ahead and use the role hierarchy that we talked about in the previous sections.

1. Click **Your Name** ► **Setup** ► **Manage Users** ► **Roles**. If you see an introductory splash page called Understanding Roles, click **Set Up Roles** at the bottom of the page to skip to the actual tool.




Figure 59: Empty Role Hierarchy Page in Tree View Mode

The default view for this page is the tree view, as indicated in the drop-down list on the far right side of the Role Hierarchy title bar. When creating a role hierarchy, it's probably easiest to stick with this or the list view, because they both make it easy to see how the roles all fit together in the hierarchy. The sorted list view is best if you know the name of a role that you want to find but aren't sure where it fits in the hierarchy, or if you don't want to click open all the tree nodes. For our purposes, we'll stick with the tree view for now.

When you first start defining a role hierarchy, the tree view displays a single placeholder node with the name of your organization. From this point, we need to add the name of the role that is highest up in the hierarchy—in our case, the CEO.



Note: If you're building your Recruiting app with a free Developer Edition organization, you may have a role hierarchy predefined as a sample. That's alright. You can still follow along and create some more roles.

2. Just under Universal Containers, click **Add Role**.
3. In the `Role Name` text box, enter CEO.
4. In the `This role reports to` text box, click the lookup icon  and click **Select** next to the name of your organization.

By choosing the name of the organization in the `This role reports to` text box, we're indicating that the CEO role is a top-level position in our role hierarchy and doesn't report to anyone.

5. In the `Role Name` as displayed on reports text box, enter `CEO`. This text is used in reports to indicate the name of a role. Since you may not want a long role name, like `Vice President of Product Development`, taking up too much space in your report columns, it's advisable to use a shortened, yet easily identifiable, abbreviation.
6. Leave any other options, such as `Opportunity Access`, set to their defaults. These access options don't have anything to do with our Recruiting app, and only appear if you have the org-wide defaults for a standard object set to a level more restrictive than `Public Read/Write`.
7. Click **Save**.

Role
CEO [Help for this Page](#)

Below is the list of users assigned to this role. Click **Edit** to modify the role name. Click **Assign Users to Role** to assign existing users to this role. Click **New User** to create a user for this role.

Hierarchy: Universal Containers » CEO
Siblings:

[Users in CEO Role \(0\)](#) | [Category Group Visibility Settings \(0\)](#)

Role Detail [Edit](#) [Delete](#)

Role Name	CEO	Role Name as displayed on reports	CEO
This role reports to	None	Sharing Groups	Role, Role and Subordinates
Modified By	Jane Smith, 8/16/2010 1:58 PM		
Customer Portal Role	<input type="checkbox"/>		

Every role can have one or more assigned users.

[Users in CEO Role](#) [Assign Users to Role](#) [New User](#) [Users in CEO Role Help](#)

No records to display

Figure 60: CEO Role Detail Page

Now that we've created our first role, we can assign the appropriate user to it.

8. In the CEO role detail page, click **Assign Users to Role**.
9. In the `Available Users` drop-down list, select `All Unassigned`.
10. Choose a user from the list (in our case, Cynthia Capobianco), and click **Add** to move her to the `Selected Users for CEO` list.
11. Click **Save**.

If we return to the main Roles page by clicking *Your Name* ► **Setup** ► **Manage Users** ► **Roles**, we can now see our new CEO role in the hierarchy. Defining the rest of the roles is just an exercise that you can do on your own according to the Universal Containers Role Hierarchy diagram (If you don't define the role hierarchy, some of the tests that we talk about later won't work as described.)



Note: There's no need to assign users to every role at this point—we'll do that later when we test out our app.



Tip: To speed up the process of adding a new role, click **Add Role** directly under the name of the role to which the new role should report. When you do this, the `This role reports to` text box is automatically filled in with the name of the appropriate role.

Not too hard, right? With org-wide defaults and a role hierarchy in place, we're actually pretty close to finishing up our record-level access permissions. All we have left to do is share recruiting-related records between groups that appear in separate branches of the role hierarchy, and between peers in a single group. Fortunately, we can accomplish both of those tasks with a combination of sharing rules and manual sharing. We just need to figure out what's left that needs to be shared, and with whom.

What's Left to be Shared?

So what *is* left to be shared? After reviewing our table of required permissions, it turns out it's just a few more things (remember, since users always have access to the records that they own, we need to worry only about the read and update permissions for our record-level access settings):

- Recruiters need read and update access on every position, candidate, job application, and review record that exists in the app.
- Hiring managers need:
 - Read and update access on position and job posting records on which they're the hiring manager
 - Read access on candidate records for which they're the hiring manager
 - Read and update access on every job application and review record
- Interviewers need read access on the candidate and job application records for people they're interviewing, and the ability to update their reviews.

That shouldn't be too hard! Let's go do it.

Introducing Sharing Rules

First let's see what we can do with sharing rules. Sharing rules let us make automatic exceptions to org-wide defaults for particular groups of users. We've already defined several specific groups

with the roles that we created in the previous section, but we can also make up other groups as needed.

The thing to remember with sharing rules is that, like role hierarchies, we can use them only to open up record access to more users. Sharing rules and role hierarchies can never be stricter than our org-wide default settings.

Sharing Rules in Our Recruiting App

Sharing rules work best when they're defined for a particular group of users that we can determine or predict in advance, rather than a set of users that is frequently changing. For example, in our Recruiting app, we need to share every position, candidate, job application, and review with every recruiter. Since recruiters all belong to either the Recruiting Manager or Recruiter roles in the role hierarchy, we can easily use a sharing rule to share those objects with the Recruiting Manager role and its subordinates.

Alternatively, consider another use case from our Recruiting app: interviewers need read access on the candidates and job applications for people they're interviewing. In this case, the set of interviewers is a lot harder to predict in advance—hiring managers might use different sets of interviewers depending on the position for which they're hiring, and the interviewers might come from different groups in the role hierarchy. As a result, this use case probably shouldn't be handled with sharing rules—the team of interviewers for any given manager is just too hard to predict.

Let's go through the set of required permissions we still need to implement and pick out the ones that would work best with sharing rules:

Use Case	Should we use a sharing rule?
Recruiters need read and update access on every position, candidate, job application, and review record that exists in the app.	Yes. As we discussed previously, it's easy to pick out the group of recruiters in our role hierarchy.
Hiring managers need read and update access on position and job posting records on which they're the hiring manager.	No. It's too hard to predict which positions will be assigned to which hiring manager. We'll need to handle this use case some other way.
Hiring managers need read access on candidate records on which they're the hiring manager.	No. Again, it's too hard to predict which positions will be assigned to which hiring manager.

Use Case	Should we use a sharing rule?
Hiring managers need read and update access on every job application and review record.	Yes. Since we're not restricting which job applications and reviews a hiring manager gets to read and update, we can easily pick out all of the hiring managers from our role hierarchy and define a sharing rule for them.
Interviewers need read access on the candidate and job application records for people they're interviewing.	No. As we discussed previously, it's hard to predict who will be a member of an interview team for a particular position.

Great! Now that we know the required permissions we want to implement with sharing rules, let's go ahead and define them.

Try It Out: Defining a Public Group for Reviewers

Before we dive head first into creating our sharing rules, we need to make sure that we have the appropriate public groups set up. A *public group* is a collection of individual users, other groups, individual roles, and/or roles with their subordinates that all have a function in common. For example, users with the Recruiter profile as well as users in the SW Dev Manager role both review job applications. Using a public group when defining a sharing rule makes the rule easier to create and, more important, easier to understand later, especially if it's one of many sharing rules that you're trying to maintain in a large organization. You'll need to create a public group if you ever want to define a sharing rule that encompasses more than one or two groups or roles, or any individual.

Looking at the required permissions that we want to implement, there are just two objects that need a public group for their sharing rules: Job Application and Review. The good news is that we can cover these objects in a single group because the Review object is on the detail side of a master-detail relationship, so it inherits the sharing settings we apply to the Job Application object. Since both recruiters and hiring managers need read and update access to job applications and reviews, let's go ahead and make a public group called Reviewers that encompasses recruiters and hiring managers.

1. Click **Your Name** ► **Setup** ► **Manage Users** ► **Public Groups**.
2. Click **New**.

Figure 61: New Public Group Page

The New Public Group page allows you to choose other public groups, individual roles, individual roles including the roles' subordinates, or individual users.

3. In the `Group Name` text box, enter `Reviewers`.
4. In the `Search` drop-down list, choose `Roles`.
5. In the `Available Members` list, select `SW Development Manager`, `Director Product Management`, and `Director QA`, then click **Add**.
6. Go back up to the `Search` drop-down list, and this time choose `Role and Subordinates`.
7. In the `Available Members` list, select `Recruiting Manager`, and click **Add**.
8. Click **Save**.

Easy! Now we're ready to define our sharing rules.

Try It Out: Defining a Sharing Rule for Job Application and Review Records

Since we just defined our `Reviewers` public group, let's go use it to define our sharing rule for review records.

1. Click *Your Name* ► **Setup** ► **Security Controls** ► **Sharing Settings**.

Remember this page? We were last here when we defined our org-wide defaults.

2. In the `Manage sharing settings` for drop-down list, choose `Job Application`.

Choosing an object in this drop-down list allows us to focus in on the org-wide defaults and sharing rules for a single object at a time rather than looking at all of them in a long page—a really useful thing if you've got a large organization with several custom objects.

If you had chosen `Review` instead of `Job Application`, you would not have the option of creating sharing rules, since you cannot create sharing rules for a detail record in a master-detail relationship. However, since you chose `Job Application`, a `Sharing Rules` related list appears. We'll use that to create the sharing rules that will apply to both the `Job Application` and the `Review` objects.

3. In the `Sharing Rules` area, click **New**.
4. For the rule type, make sure `Based on record owner` is selected.
5. In the `Job Application: owned by members of` drop-down list, select `Public Groups`.
6. Next to that drop-down list, choose `Entire Organization`.

Just as we talked about already, you can define a sharing rule only for a single public group, role, or role with all of its subordinates. By default, the platform includes a default public group that encompasses every user in your organization.

7. In the `Share with` drop-down list, select `Public Groups`.
8. Next to that drop-down list choose `Reviewers`.
9. In the `Access Level` drop-down list, select `Read/Write`.
10. Click **Save**.
11. Click **OK** in the dialog box that says this operation could take significant time.

And that's it! We've just created a rule that shares reviews written and owned by any member of the organization with all recruiters and hiring managers. Since reviewers and hiring managers all need the power to read and update reviews, we handled everyone with a single sharing rule and a public group.

To finish up here, go ahead and create two more owner-based sharing rules according to the following table:

Table 33: Additional Sharing Rules

These records...	Owned by...	Should be shared with...	Access Level
Candidate	The role and subordinates of the Recruiting Manager	The role and subordinates of the Recruiting Manager	Read/Write
Employment Website	The role and subordinates of the Recruiting Manager	Reviewers	Read/Write

The sharing rule for the Employment Website object is necessary to let hiring managers post jobs, even though they will never be updating employment website records directly (the org-wide defaults will prevent that). Without the rule, hiring managers can see employment website records but cannot create job postings. This is because the Job Posting object is a junction object (as you may recall from the last chapter), and the Employment Website object is one of the Job Posting object's two master-detail relationships. Sharing access to a junction object record is determined by a user's sharing access to both associated master records (in this case, the associated position and employment website records) and the Sharing Setting option on the relationship field. For example, if the sharing setting on both parents is Read/Write, then the user must have Read/Write access to both parents in order to have Read/Write access to the junction object.

In the sharing rule for the Employment Website object, we opted to use the existing Reviewers public group. Doing this saved us a few clicks without granting access to any users who shouldn't be looking at employment website records.

Beyond the Basics

Did you know that you can use criteria-based sharing rules to open up record access to users?

Say you want to share records based on field values in records instead of record owners. You can set up your sharing rules based on field value criteria and apply filter logic to open up specific record access to users.

To find out more, see “Criteria-Based Sharing Rules Overview” in the Salesforce online help.

Introducing Manual Sharing

Now let's talk about what we have left to do to finish defining our sharing model. After implementing our sharing rules, the following required permissions remain:

- *Hiring managers need read and update access on position records on which they're the hiring manager.*
- *Hiring managers need read access on candidate records on which they're the hiring manager.*
- *Interviewers need read access on the candidate and job application records for people they're interviewing.*

We didn't implement those required permissions with sharing rules because it was too hard for us to come up with a consistent group of users who would need access to a particular set of records. Really, this is where the job of the recruiter comes into play. A recruiter like Mario Ruiz owns the position, candidate, and job application records for jobs that he's trying to fill, and he also knows the hiring manager and interviewers who should be assigned to them.

Fortunately, we have one final type of record-access setting that allows Mario to share specific records with other specific users: manual sharing. With manual sharing, Mario can grant read or read/write access on records that he owns to any other user, role, or public group. Although it isn't automated like org-wide defaults, role hierarchies, or sharing rules, manual sharing gives Mario the flexibility to share particular records with the ever-changing groups of interviewers and hiring managers with whom he has to deal every day.

Try It Out: Defining a Manual Sharing Rule

Let's pretend that we're a recruiter like Mario and we need to share a particular candidate record that we own with another role, group, or user:

1. On the detail page for the candidate, click **Sharing**.



Sharing Detail Help for this Page ?

C-00002
C-00002

This page lists the users, groups, roles, and territories that have sharing access to C-00002. Click **Expand List** to view all users who have access to it.

View:

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | **All**

User and Group Sharing			
Action	Type	Name	Access Level
	User	Jane Smith	Full Access

[User and Group Sharing Help ?](#)

Figure 62: Sharing Detail Page

Since we own this candidate record, we get to see details about who else can see the record and why. If we didn't own this record, there would be a message about not having sufficient privileges.



Tip: If we wanted to view the names of each user who has access to the record rather than just the names of the roles and public groups, we could click **Expand List** in this page. Although the operation can take some time depending on the number of users in our organization, it's helpful to determine whether we need to define a manual sharing rule for a particular user or if he or she already has access.

2. Click **Add**.
3. In the `Search` drop-down list, choose whether we want to manually share the record with a user, public group, role, or role and subordinates.
4. In the `Currently Not Shared` list, select the user, public group, or role that should have access to the record, and click **Add**.
5. In the `Access Level` drop-down list, specify whether the user, public group, or role should have read or read/write access to the record.
6. Click **Save**.

Not too hard! When we roll out our Recruiting app to users, we'll have to train our recruiters to take these steps for the position, candidate, and job application records that their hiring managers and interviewers need to access. Once this training is complete, we will have implemented all of the required sharing and security settings that we discussed at the beginning of the chapter—well done!

Displaying Field Values and Page Layouts According to Profiles

Before we give our security and sharing model a thorough testing, let's leverage the work we've done to further enhance our app's usability and, at the same time, give our data integrity a little boost.

Both usability and data integrity are adversely affected by irrelevant data. The less irrelevant data we display to users, the better off everyone will be. Not only can irrelevant data be confusing and impede a user's efficiency, but it also makes he or she more prone to entering incorrect values. By taking away unnecessary choices, we reduce the risk of making avoidable mistakes.

Although we've already put plenty of thought into our app's usability, there's always room for improvement. We won't go into a detailed usability analysis here, but let's make two minor

modifications to get a feel for some of the ways you can improve the usability and data integrity of your apps in the future.

The modifications we're going to make involve position records. Currently, each position record displays the same data to hiring managers and recruiters, even though there are a few items on the position record that are of no use to recruiters who create positions for departments other than Development. For example, a recruiter who is creating a position for a Sr. Financial Analyst would have no use for the Required Programming Languages section.

By the same token, there are some options on position records that have nothing to do with technical positions, such as the Human Resources and Warehousing values in the `Functional Area` picklist. Wouldn't it be nice if we could create two types of position records: one with IT-related data for IT managers, and another for non-IT personnel?

Fortunately, we can with *record types*! Record types allow you to show different picklist values and page layouts to different users based on their profiles.

To address the issues discussed above, we'll create two position record types. The first position record type will be for IT positions and will include the Required Programming Languages section of the page layout. Additionally, it will exclude all of the options in the `Functional Area` picklist except for Information Technology and Miscellaneous. The second position record type will be for all non-IT positions and will include all of the `Functional Area` picklist values except for Information Technology, but will omit the Required Programming Languages section.

We'll give Recruiters the option of choosing either record type when creating a position, since recruiters at Universal Containers work with every department, not just Development. However, since our hiring managers all work in the Development department, we'll restrict them to only creating positions that use the record type for IT positions.

Down the road, if you want to include hiring managers from other departments in your recruiting app, you can reconfigure the record types on the Position object to let all hiring managers choose which record type to use, or create more profiles. For now, though, let's use our app the way it's currently set up, and focus our attention on learning record types.

Try It Out: Creating Record Types

Let's start by creating the position record type for standard, non-IT positions.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In the Record Types related list, click **New**.

4. In the `Record Type Label` field, enter `Standard Position`. When you move your cursor, the value of the `Record Type Name` field changes to `Standard_Position`.
5. In the `Description` field, enter `Record type for all non-IT positions`.
6. Select the `Active` checkbox.

The bottom of the screen lists all your profiles. Here is where we can determine which profiles have access to this record type. All of them are selected by default, but we don't want hiring managers to use this record type.

7. Deselect the **Enable for Profile** checkbox next to `Hiring Manager`.
8. Click **Next**.
9. Leave the `Apply one layout to all profiles` radio button selected, and select `Position Layout` in the adjacent drop-down list.
10. Click **Save**.

The `Standard Position` record type detail page appears. The page lists the picklist fields found on the record type's associated page layout, the `Position Layout`.

11. Click **Edit** next to the `Functional Area` field.

Since this is the record type for all non-IT positions, let's remove `Information Technology` from the `Functional Area` picklist.

12. In the `Selected Values` box, select `Information Technology` and use the arrows to move it to the `Available Values` box.
13. Leave the `Default` drop-down list set to `None`, and click **Save**.

You're done creating your first record type, but it's not quite configured the way we want it. While it omits the `Information Technology` value in the `Functional Area` picklist, it still displays the `Required Programming Languages` section. We'll fix this later when we modify the page layouts for our record types, but first we have to create one more record type.

1. Click ***Your Name* > Setup > Create > Objects**.
2. Click **Position**.
3. In the `Record Types` related list, click **New**.
4. In the `Record Type Label` field, enter `IT Position`. When you move your cursor, the value of the `Record Type Name` field changes to `IT_Position`.
5. In the `Description` field, enter `Record type for all IT positions`.
6. Select the `Active` checkbox.

Once again, the bottom of the screen lists your org's profiles, although this time they are deselected by default. Let's make this the default record type for the Hiring Manager profile. We'll also enable the record type for recruiters, but we won't make it the default for them.

7. Next to Hiring Manager, select the `Enable for Profile` and `Make Default` checkboxes.
8. Next to Recruiter, select the `Enable for Profile` checkbox.
9. Click **Next**.

We are again given the option to apply different layouts to different profiles. We still need to create the page layout for this record type, though, so we'll have to apply the page layouts later.

10. Leave the `Apply one layout to all profiles` radio button selected, and select `Position Layout` in the adjacent drop-down list.
11. Click **Save**.

The IT Position record type detail page appears.

12. Click **Edit** next to the `Functional Area` field.

Since this is the record type for all IT positions, let's remove all the options from the `Functional Area` picklist except for `Information Technology` and `Miscellaneous`.

13. Use the arrows to move the values until the `Available Values` box only contains `Information Technology` and `Miscellaneous`.
14. In the `Default` drop-down list, select `Information Technology`.
15. Click **Save**.

Both record types are now in place, and both are omitting the picklist values they're supposed to omit. It's time to configure the page layouts for these record types.

We'll need a separate page layout for each record type. Lucky for us, we already have one page layout for the `Position` object (`Position Layout`), so we just need to create one more.

1. Click **Your Name > Setup > Create > Objects**.
2. Click **Position**.
3. In the `Page Layouts` related list, click **New**.
4. In the `Existing Page Layout` drop-down list, select `Position Layout`.

Selecting the existing page layout creates a copy on which we can base our new page layout. This saves us from having to create the layout from scratch.

5. In the `Page Layout Name` field, enter `IT Position Layout`, and click **Save**.

We're done creating our new IT position page layout. Now, let's edit both our new and original page layouts so they display relevant data. Since we're on the IT Position page layout, we'll start with that one.

This page layout already includes the Required Programming Languages section, so we don't need to add that; however, we do want to add the `Record Type` field to the page layout so users will instantly be able to tell what type of position record they're editing.

6. Select the Fields category in the palette, then drag the `Record Type` field to just below the `Last Modified By` field.
7. Click **Save**.

Now, let's edit the Position Layout page layout. This is the layout we'll use for our Standard Position record type, so we'll want to remove the Required Programming Languages section.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In the Page Layouts related list, click **Edit** next to Position Layout.
4. Click the X (✕) in the upper right corner of the Required Programming Languages section.

We'll want to add the `Record Type` field to this page layout as well.

5. Select the Fields category in the palette, then drag the `Record Type` field and drag it to just below the `Last Modified By` field.
6. Click **Save**.

We're on the verge of finishing! There's just one more easy task to complete: assigning our Position page layouts to our new record types.

Assigning page layouts is easy because you can make all of the assignments for an object on a single page.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In either the Page Layouts related list or Record Types related list, click **Page Layout Assignment**.
4. Click **Edit Assignment**.

A table shows the Position page layout assignments for all of the different profile and position record type combinations. In the table, you can select the profile and position record type combinations you want to change. Use SHIFT+click to select a range of cells or CTRL+click to select multiple cells at once. Use the drop-down list above the table to indicate the page layout to which you want to reassign your selections.

5. Click the IT Position column heading. This selects all of the values in the IT Position record type column.
6. Select IT Position Layout in the Page Layout To Use drop-down list.
7. Click **Save**.

Your record types are good to go!

Putting It All Together

Congratulations! We've just implemented all of our required security and sharing settings, first by defining object-level access with profiles, then by securing field-level access with field-level security, and finally by defining record-level access using org-wide defaults, role hierarchies, sharing rules, and manual sharing.

We learned about the difference between object-, field-, and record-level security, and how profiles and roles work together to first determine the objects and tabs that a user can possibly use, and then the specific records that the user can actually access and edit. We also learned ways to set up other profile-based features like record types to improve both our data integrity and our app's usability.

Let's now try it out for ourselves. To do so, we'll first have to define a number of users, and then we can play around with creating records and seeing who has access to what.

Try It Out: Creating Users for Our Recruiting App

To really put our Recruiting app through its paces, we'll first need to define the following users and assign a couple of them to some of the recruiting records that we imported earlier.



Note: If you're implementing the Recruiting app in a Developer Edition organization, you'll have only a few additional users to play with besides the System Administrator user. You can still try out all of the use cases that we describe here, but you'll have to update the user's profile and role for whatever use case you're working on.

Table 34: Recruiting App Users

User's First and Last Name	Title	Profile	Role	Owner of
Phil Katz	Recruiting Manager	Recruiter	Recruiting Manager	• DBA position

User's First and Last Name	Title	Profile	Role	Owner of
				<ul style="list-style-type: none"> • Mary Jane, candidate for DBA • Job Application linking the DBA position with Mary Jane
Mario Ruiz	Senior Recruiter	Recruiter	Recruiter	<ul style="list-style-type: none"> • Documentation Writer position • George Schnell, candidate for Documentation Writer • Job Application linking the Documentation Writer position with George Schnell
Ben Stuart	Software Development Manager	Hiring Manager	SW Dev Manager	
Melissa Lee	Staff Software Engineer	Standard Employee	SW Engineer	
Amy Lojack	Senior Product Manager	Standard Employee	Product Manager	<ul style="list-style-type: none"> • Review for the job application associated with the DBA position*

* You'll need to create this review record because it isn't part of the sample import data.

Let's walk through the creation of Phil Katz. Then you can finish the other four users on your own:

1. Click ***Your Name*** ► **Setup** ► **Manage Users** ► **Users**.
2. Click **New User**.
3. Fill out the fields in the User edit page according to Recruiting App Users.

Because we're defining this user to test the security and sharing settings of our app, enter a real email address that you have access to in the `Email` field and then a "fake" email address in the `Username` field (for example, `phil.katz@recruiting.com`). We'll use the "fake" value in the `Username` field to log in to the app as Phil Katz, but we'll get Phil's automatically generated password at the real email account that you specified. Without that password, we'd never be

able to log in! (For a real user, both `Email` and `Username` should generally have the same value.)



Figure 63: New User Edit Page



Tip: When creating a new user, you are also required to create the user's community nickname. The community nickname is used to identify the user in the Salesforce CRM Ideas app, which is a community of users who post, vote for, and comment on ideas. Consider it an online suggestion box that includes discussions and popularity rankings for any subject. The community nickname can contain up to 40 alphanumeric characters. For more information, see the Salesforce online help.

4. Click **Save**.

Now that we've created the Phil Katz user, let's give him ownership of the DBA position and its associated job application and candidate records.

5. Click the **Positions** tab.

6. From the `View` drop-down list, select **All** and click **Go**.



Tip: If you want to see more than just the `Position Title` field in this view, click **Edit** next to the `View` drop-down list and add additional fields in the `Select Fields to Display` section.

7. Click **DBA**.

8. Next to the `Owner` field, click **Change**.

9. Click the lookup icon  and choose Phil Katz.

10. Click **Save**.

11. In the Job Applications related list, click the name of the listed job application and repeat Steps 8-10.
12. Click the ID of the associated candidate on the Job Application detail page and repeat Steps 8-10.

All done! Now create the other four users in Recruiting App Users and assign them ownership of the indicated records. To fully complete this, note that you'll have to create Amy Lojack's review.

Try It Out: Verifying that Everything Works

Now that we've got data assigned to actual users, let's go through our Recruiting app and see how the security and sharing permissions that we defined in this chapter play out:

1. First log in as Mario Ruiz—verify that he can see and edit both positions, all three candidates, all three job applications, and Amy Lojack's review. Verify that the **New** buttons are there for all recruiting objects. Verify that he can create positions using either position record type.
2. Log in as Melissa Lee—verify that she can view positions but that there's no **New** button. Verify that she can't see any candidates, reviews, job applications, or employment websites.
3. Log in as Ben Stuart—verify that he can view positions and that there's a **New** button, but no record type selection. Verify that he can view but not edit employment websites. Verify that he can't see any candidates. Verify that he can view job applications, but not edit their lookup fields. Verify that he can view reviews and that there's a **New** button. (What do reviews look like? Can he see the names of the candidates and job applications on them?)
4. Log in again as Mario Ruiz—have him manually share read/write access on the SW Engineer position with Ben. Have him manually share read access on the candidate with Melissa and Ben. Have him manually share read access on the job application with Melissa and read/write access on the job position with Ben.
5. Log in again as Melissa Lee—verify that she can now see the candidate and job application that Mario just shared with her but that she can't see the candidate's social security number. Have her create a review for that candidate.
6. Log in again as Ben Stuart—verify that he can edit the Jr. Software Engineer position. Verify that he can read and update Melissa's review. Verify that he can update the job application to suggest that they hire the candidate.

How did we do? If all of these use cases worked correctly, you've just successfully set up security and sharing for our Recruiting app! However, there is one critical security-related issue that

we have yet to address: who will be responsible for overseeing the operation of the Recruiting app and its related data when the app goes live?

Delegating Data Administration

As with nearly all Force.com apps, our Recruiting app does not require tedious ongoing administration or a watchful eye monitoring its daily operation. Once the app is deployed, it just works! But from time to time, a decision or issue arises that requires human intervention, and some basic manual administration is required, like:

- A hiring manager is retiring and has forty open positions that need to be transferred to another manager
- A current Recruiting app user needs immediate access to private data owned by another user who happens to be on vacation
- Duplicate records have piled up in the Recruiting app and need to be removed
- A new employee just got hired and needs access to the Recruiting app

To handle these situations, someone might need to override the security and sharing configurations we just created. Who should have such powers within our app, and how can these powers be granted?

Obviously, your company's primary Salesforce administrator can handle just about any issue that users may encounter in Salesforce. Primary administrators are assigned to the System Administrator profile, which automatically grants several global administrative permissions, including:

- “View All Data”—View all data owned by other users in your organization
- “Modify All Data”—Modify all data owned by other users in your organization, mass update and mass delete records, and undelete records that other users deleted
- “Customize Application”—Customize just about anything in Salesforce, from page layouts to the data model
- “Manage Users”—Add and remove users, reset passwords, assign profiles, and more

For smaller companies, it makes sense to have a single administrator be the “go-to” person for all Salesforce issues. But for medium to large companies, assigning all Salesforce responsibilities to one person is not practical, especially when you consider that a company can run its entire business in the cloud using a different Force.com app to suit each of its business needs. This could add up to dozens of apps and hundreds or thousands of users! Your primary Salesforce administrator will likely go insane unless other folks can help with the administration. At the same time, every administrative privilege you grant increases the risk of exposing your company's sensitive data, so you need precise control over the amount of access you enable.

To preserve both your administrator's sanity and your company's security, the Force.com platform provides two ways to quickly delegate restricted data administration access: object-level permissions and delegated administration groups.

Overriding Sharing with Object-Level Permissions

Believe it or not, we already delegated a few administrative responsibilities a few pages ago when we created the Recruiter profile. If you go back to that section, you'll see that we selected the “View All” and “Modify All” object-level permissions for job postings and employment websites. As you might expect, “View All” lets users view all of the records of that object, while “Modify All” lets users read, edit, and delete all of the records of that object. So how does selecting these permissions differ from just selecting the “Create,” “Read,” “Edit,” and “Delete” profile permissions individually? And how do “View All” and “Modify All” help you delegate administration?

The keyword here is “all.” When you select “View All” or “Modify All” next to an object on a profile edit page, you grant those users access to **all** records of that object regardless of the sharing and security settings. In essence, the “View All” and “Modify All” permissions ignore the sharing model, role hierarchy, and sharing rules that the “Create,” “Read,” “Edit,” and “Delete” permissions respect. Furthermore, “Modify All” also gives a user the ability to mass transfer, mass update, and mass delete records of that specific object, and approve such records even if the user is not a designated approver. These tasks are typically reserved for administrators, but because “View All” and “Modify All” let us selectively override the system, responsibilities that are usually reserved for the administrator can be delegated to other users in a highly controlled fashion.



Note: We'll learn more about approving records in the next chapter. Other administrative actions, such as mass updates, are covered in the Salesforce online help.

You may wonder if the “View All” and “Modify All” object-level permissions are similar to the “View All Data” and “Modify All Data” global administration permissions discussed above. It is true that they all ignore the sharing model, hierarchy, and sharing rules, but bear in mind that object-level permissions only apply to records of a specific object, whereas global administration permissions apply to records of every object in your organization. As a rule of thumb, when global administration permissions are too permissive for a particular profile, use the object-level permissions instead to control data access on an object-by-object basis.



Tip: To quickly see a list of the profiles with permissions that override the sharing model, click **Your Name** ► **Setup** ► **Security Controls** ► **Sharing Settings**, then select an object in the **Manage sharing settings for** drop down list. The page displays

a Sharing Overrides related list that shows the profiles that can access the object, as well as the global and object-level permissions that are enabling the access.

Because we already applied object-level permissions when we created the Recruiter profile, there's no need to walk through the process again here; however, you can probably imagine other ways in which the “View All” and “Modify All” permissions might be useful for this app. For example, as we discussed before, certain laws may require your company to keep all position, candidate, and job application records for a specific amount of time. This law is why we opted not to give recruiters the ability to delete those records. When that legal time limit expires, though, your company may want to hire a contractor who specializes in data cleansing to remove old recruiting data from the system. By creating a profile with the “Modify All” object-level permission on positions, candidates, and job applications, you can quickly give the contractor the permissions needed to get the job done without exposing the rest of your company's data.

Delegated Administration Groups

We just saw how the “View All” and “Modify All” object-level permissions can be used to take a considerable load off of the primary administrator's shoulders; however, there are still a few other administrative responsibilities that you might want to delegate but can't with object-level permissions. For example, you might not want to burden the primary administrator with the tasks of manually adding every new employee to Salesforce or resetting a user's password every time it's forgotten. Also, as time passes, your company may need a new field or two added to review records, or a new record type for positions. Sometimes, it's more efficient to delegate basic administrative tasks like these to members of a group so the primary administrator can focus on other things.

A delegated administration group is a group of non-administrator users with limited administrative privileges. These privileges can include:

- Creating and editing users and resetting passwords for users in specified roles and all subordinate roles
- Assigning users to specified profiles
- Logging in as a user who has granted login access to an administrator
- Managing custom objects created by the primary administrator

Let's define a delegated administration group in our Recruiting app that enables its members to manage our Recruiting app's users and make adjustments to the app's custom objects without having access to all of the other data in Salesforce.

Try It Out: Defining the Recruiting Manager Administration Group

We'll start by creating the delegated administration group:

1. Click *Your Name* ► **Setup** ► **Security Controls** ► **Delegated Administration**.
2. Click **New**.
3. In the `Delegated Group Name` text box, enter `Recruiting App Admins`.
4. Select the `Enable Group for Login Access` checkbox.

The `Enable Group for Login Access` option allows the delegated administrators in this group to log in as a user who has explicitly granted login access to administrators for a specific period of time. This can be useful for troubleshooting problems that might arise for the user.

5. Click **Save**.

The `Delegated Group` detail page appears. We can use the `Delegated Administrators` related list to specify the users to include in this group.

6. Click **Add** in the `Delegated Administrators` related list.
7. In the first empty text box, enter `Phil Katz`, the Recruiting Manager at Universal Containers.



Note: Members of delegated administrator groups must be assigned to a profile that has the “View Setup and Configuration” permission.

Each delegated administration group can have up to five members, but we're only going to include our Recruiting Manager in this group, due to the focus of our app.

8. Click **Save**.

We're back on the `Delegated Group` detail page. The `User Administration` related list lets us specify the kinds of users this group can manage.

9. Click **Add** in the `User Administration` related list.

Instead of selecting individual users to manage, we'll select them by their roles. For each role we select, its subordinate roles are automatically selected too. (Flip back to the [Universal Containers role hierarchy](#) on page 163 if you can't remember which roles are subordinate.) We don't want to let members of this group administer the Salesforce accounts of executives at Universal Containers, but the group should be able to administer the Recruiting app for recruiters, directors, managers, and their subordinates.

10. In the first empty text box, enter `Recruiter`.
11. In the second text box, enter `Director Product Management`.
12. In the third text box, enter `Director QA`.
13. In the fourth text box, enter `SW Dev Manager`.
14. Click **Save**.

The Assignable Profiles related list lets us specify the profiles this group can assign to the users they manage. Note that delegated administrators cannot modify these profiles; they can only assign users to them.

15. Click **Add** in the Assignable Profiles related list.
16. In the first empty text box, enter `Recruiter`.
17. In the second text box, enter `Hiring Manager`.
18. In the third text box, enter `Standard Employee`.
19. Click **Save**.

The Custom Object Administration related list lets us specify the custom objects that delegated administrators of this group can administer. The delegated administrators can manage just about every aspect of the custom object, such as modifying page layouts, adding fields, and even deleting the object; however, they cannot set permissions for the custom object on profiles, manage workflow and sharing for the object, and modify relationships to other objects.

20. Click **Add** in the Custom Object Administration related list.

Let's give the members of this delegated administrator group (in this case, Phil) the ability to modify the Position object. That way, if position records need another field or hiring managers request another record type in the future, Phil can handle it. (Of course, we'll also want to warn Phil that he should be careful not to delete the Position object altogether—custom object administrative rights give him that power as well!)

21. In the first empty text box, enter `Position`.
22. Click **Save**.

That wraps up the configuration of our delegated administration group! Our Recruiting app now has a delegated administrator who can provide our primary Salesforce administrator with some relief.

Try It Out: Verifying that Delegated Administration Works

To see what a delegated Recruiting App administrator can and can't do, log in as Phil Katz.

1. Click *Your Name* ► **Setup** ► **Manage Users** ► **Users**.

Notice that the **Edit** link appears next to any user assigned to a role or subordinate role for which our delegate administrator group can manage users. The **Edit** link does not appear to users assigned to other roles, such as Cynthia Capobianco, the CEO.

You'll also see the **New User**, **Reset Password(s)**, and **Add Multiple User(s)** buttons are now available to Phil. Let's see what happens if Phil adds a new user.

2. Click **New User**.

The User Edit page appears. At first glance, this page looks just like the User Edit page we saw when we added users while logged in as the system administrator. If you look closely at the options available in the `Role` and `Profile` drop-down lists, though, you'll notice a difference—the options are limited to what we specified when we created our delegated administration group. For example, Phil can't create a user with an executive role, nor can he assign a user to any profile other than Recruiter, Hiring Manager, or Standard Employee.

We don't need to create any more users right now, so just click **Cancel**.

The last thing to verify is Phil's ability to modify the Position object.

3. Click *Your Name* ► **Setup** ► **Create** ► **Objects**.

We see that the **Edit** and **Del** links are available next to the Position object. This means that Phil now has the ability to both edit and delete the Position object. Our testing is complete!

Beyond the Basics

Did you know that you can set up single sign-on between your organization and other authorized Web resources?

Say you want your employees to only have to remember one password when they sign on to the recruiting app, as well as your company website. You can work with an identity provider, exchange information with them, then configure Salesforce for single sign-on.

To find out more, see “About Single Sign-on” in the Salesforce online help.

Summing Up

This chapter has covered quite a bit of ground! We discussed the differences between object-, field-, and record-level security. We learned how profiles and roles work together to first determine the objects and tabs that a user can possibly use, and then the specific records that the user can actually access and edit. We also discovered ways to set up other profile-based features like record types to improve both our data integrity and our app's usability. Finally, we tested everything, and delegated the administration of our Recruiting app in an efficient and secure way. Now that we've got security squared away, let's go incorporate some business logic into our app with workflow rules.

Chapter 9

Collaborating with Chatter

In this chapter ...

- [Introducing Chatter](#)
- [Tracking Fields on Your Custom Objects](#)
- [Following Records](#)
- [Summing Up](#)

In the last chapter, we established a security and sharing model to determine who gets to see which data in the expanded Recruiting app. We've seen how easy it is to restrict access to fields and records using the Force.com platform. Now that our application is secure, let's make it easier for recruiters, interviewers, and hiring managers to collaborate on the hiring process. By using Chatter, we provide them a fun, fast, and effective way to get and share information.

In this chapter, we'll introduce some of Chatter's basic features—like feed tracking and following records—and show how Chatter makes collaboration a piece of cake.

Introducing Chatter

Chatter makes your app collaborative like never before. Once you enable Chatter for your organization, you can turn on feeds for most standard objects and all custom objects, allowing you to see real-time updates whenever anything changes. Users can post to feeds, comment on other posts, share files and links, work together on documents, and more. With feed tracking, when a value changes for certain fields, Chatter posts an update to the feed, instantly alerting everyone following that record. With @mentions, users can mention a person's name to make sure that person sees their update. And with hashtags, users can add topics to posts and comments so other people can find their updates more easily.

Chatter users also have their own feeds, where they can post status updates and see updates from the people and records they're following. Users can also get together to form public or private groups for more focused collaboration. Chatter makes it easy to connect with the people and information you care about most.

For more information, see the “Chatter Overview” topic in the Salesforce online help.

Try It Out: Enabling Feed Tracking on Positions and Job Applications

The recruiters at Universal Containers may want to know when a position's status or location changes. Hiring managers may want to know when a job application is submitted. Let's track these fields for them in Chatter.

To enable Chatter for your organization:

1. Click **Your Name** ► **Setup** ► **Customize** ► **Chatter** ► **Settings**.
2. Click **Edit**.
3. Select the **Enable** checkbox.
4. Click **Save**.

Learn More...'. There is a checkbox labeled 'Enable' which is checked and highlighted with a red box. The second section is 'Email Notifications' with the instruction 'Allow users to receive personal Chatter email notifications.' and a checked checkbox labeled 'Allow Emails'."/>

Chatter Settings [Help for this Page](#) ?

Chatter is a corporate network that lets your users work together, talk to each other, and share information, all in real time.

Chatter Settings ! = Required Information

Turn on Chatter and Global Search features. We have given you a head start - your users may auto-follow a few people or records by default and your search box is in the header. [Learn More...](#)

Enable

Email Notifications

Allow users to receive personal Chatter email notifications.

Allow Emails

Figure 64: Chatter Settings Dialog

Now, let's track changes to the `Status` and `Location` fields for positions:

1. Click ***Your Name*** ► **Setup** ► **Customize** ► **Chatter** ► **Feed Tracking**.
2. Select the Position object.
3. Select the `Enable Feed Tracking` checkbox.
4. Select the `Status` and `Location` fields. If either of these fields changes for a position, that position's feed will be updated, alerting the recruiters following it.
5. Click **Save**.

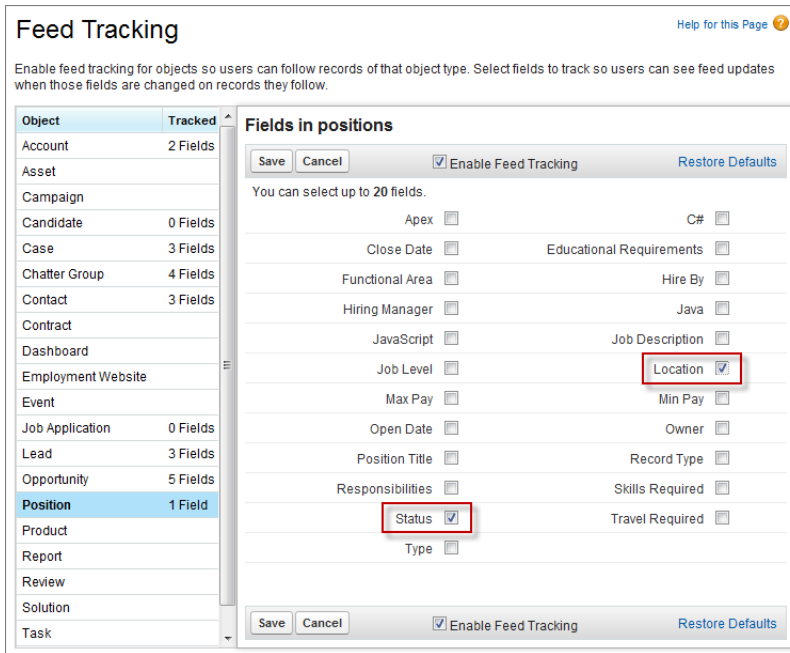


Figure 65: Chatter Feed Tracking for Positions

Similarly, the hiring manager wants to be notified when a Job Application's `Status` changes. Let's track that field:

1. Click *Your Name* ► **Setup** ► **Customize** ► **Chatter** ► **Feed Tracking**.
2. Select the Job Application object.
3. Select the `Status` field. When the value changes—from New to Review Resume, say—the Job Application feed is updated, and hiring managers following that application are notified.
4. Click **Save**.

Look at What We've Done

We chose to track updates to the `Status` and `Location` fields on the Position object. If people want to see field updates for a position, they can view the feed on that record's detail page. For example, when the recruiter helping to staff the Technical Publications department looks at the Documentation Writer record, she'll see that the location has changed to San Francisco, CA.

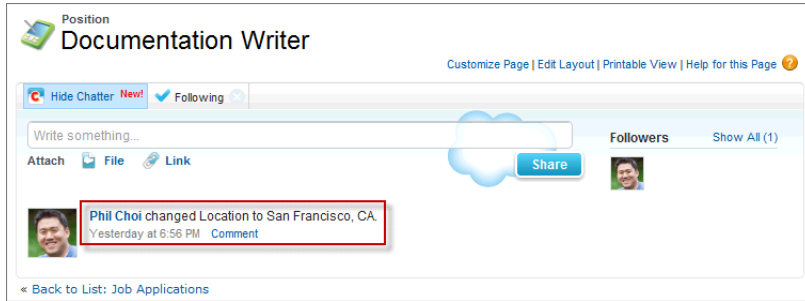


Figure 66: Chatter Location Field Update in a Position Feed

Feed tracking is great, but who wants to keep going to the record detail page to see updates? Next, we'll show how easy it is to follow records, which brings the updates to you.

Tracking Fields on Your Custom Objects

Chatter can track field updates for most objects, including custom objects, and automatically push that information to the record's feed. Simply enable feed tracking on an object, then choose the fields to track.

Following Records

Chatter lets you follow just the users, groups, documents, and records that you're interested in. Updates on the people and information you care about are posted to your personal feed. You can also enable email alerts for actions, like someone commenting on your post, as well as get daily or weekly digests for your own, as well as group feeds.

Try It Out: Following a Job Application Record

For our Recruiting app, we know that hiring managers want to keep track of job applications for their open positions. All they have to do is follow those records.

To try it yourself, just open the detail page for a job application and click **Follow**. It's that simple!

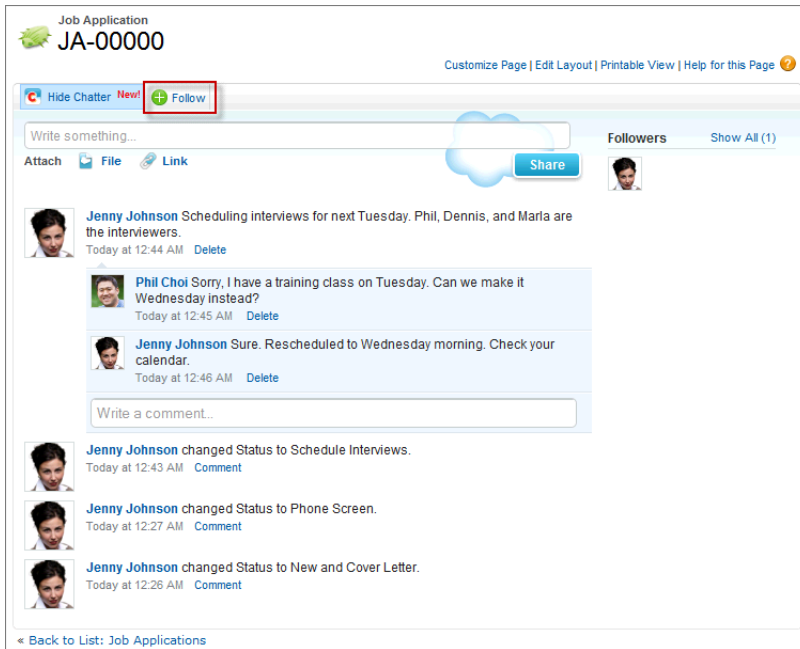


Figure 67: Following a Job Application Record

All tracked field updates, as well as other users' posts and comments on the records you're following, are posted to your personal feed. Use Chatter to connect the right people to the right records, and start collaborating in real time on the hiring process.

Beyond the Basics

Did you know you can use Chatter to collaborate outside of your browser?

Say you want to keep an eye on Chatter updates, but you can't log in through a browser. For example, maybe you're at the airport with your cell phone but don't have a laptop, or perhaps you're creating a slide deck on your laptop and there's not enough screen room for your browser. In these cases, it helps to have a Chatter client app installed.

Chatter client apps let you access Chatter data through mediums other than a Web browser. For example, the Chatter mobile app lets you access Chatter from your iPhone, iPad, iTouch, Blackberry, or Android device. Download it for free from the Apple App Store, BlackBerry App World, or Android Market.

Professionals who use Chatter extensively should also download Chatter Desktop—a Chatter client app that integrates your computer with Chatter. Chatter Desktop makes

collaboration easier than ever. Want alerts when people you're following post something new? Just install Chatter Desktop and watch it display pop-up notifications for each update. Need to share a file with your coworkers? Just drag it to the Chatter Desktop interface.

To install Chatter Desktop, simply log in to Salesforce and click **Your Name** ► **Setup** ► **Desktop Integration** ► **Chatter Desktop**.

Summing Up

We've added Chatter right into our Recruiting app, but we've barely scratched the surface. With Chatter, you can give your users the power to stay on top of everything that's happening in your company. And because it's part of the Force.com platform, you can incorporate real-time feed updates, user profiles, groups, and more into all of your custom applications.

If you dig a little further, you can learn how to programatically add posts and comments, add feeds to custom pages, and even expose Chatter actions for integration with external systems.

These advanced subjects are beyond the scope of this book, but you can find out more on the Chatter Developer Resource page at developer.force.com/chatter.

Chapter 10

Using Custom Workflow and Approval Processes

In this chapter ...

- [Introducing Workflow](#)
- [Workflow in Our Recruiting App](#)
- [Creating Workflow Rules That Assign Tasks](#)
- [Creating a Workflow Rule That Updates Fields](#)
- [Creating a Workflow Rule That Sends Email Alerts](#)
- [Introducing Approvals](#)
- [Summing Up](#)

Robust application? Check. Secure data access? Check. Fast, fun, and collaborative? Check! Our hiring team at Universal Containers has never been more productive. The last chapter demonstrated how easy it is to collaborate in real time with Chatter. Now it's time to find ways to make our app more powerful.

Up to this point, we've created little more than a glorified database—it holds the information we need and lets us search for records based on various criteria, but it doesn't help our recruiters and hiring managers do their jobs more effectively. There's no automation that lets a recruiter know when the status of a candidate has changed or when a new position has been entered into the system. Chatter will let them know about the records they're following, but we want to build automation into our app to keep the processes moving forward.

Once again, the Force.com platform helps us out with more built-in functionality. In this case, the tools that we'll use to solve our process automation problem are called *workflow* and *approval processes*. We'll take a look at the various ways that we can use workflow first, and then we'll tackle an approval process at the end of the chapter.

Introducing Workflow

Workflow is a Force.com platform business-logic engine that allows us to automatically send email alerts, assign tasks, or update field values based on rules that we define. Any time that changes to a record meet the conditions in a workflow rule, the platform automatically performs any actions associated with the rule.

For example, let's say Ben Stuart, a software development manager, has decided to extend an offer to Ethan Tran, a bright young candidate who's interested in the Jr. Software Engineer position. At Universal Containers, it's a recruiter's job to extend offers, but in this case, Mario Ruiz, the recruiter responsible for the Jr. Software Engineer position, doesn't know whether or not Ben has made a decision unless Ben emails or calls him directly.

Instead of relying on Ben to remember to tell Mario, we can set up a simple workflow that triggers the assignment of the appropriate task whenever the status on a job application record is set to Extend an Offer or Rejected. As soon as Ben changes the status of the candidate's job application, workflow creates the appropriate task and sends Mario a notification email, as shown in the following diagram.

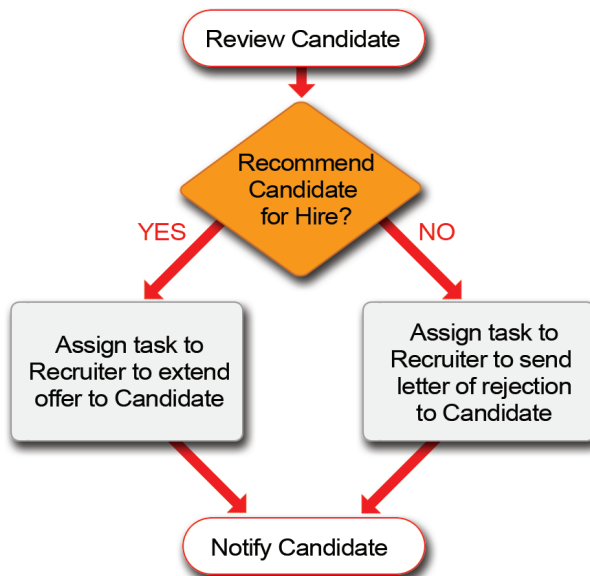


Figure 68: Automatically Assigning a Task to a Recruiter Using Workflow

Pretty powerful, isn't it? In general, if we can come up with a standard rule that specifies when a particular event should happen, we can make it happen automatically with workflow. Workflow

is one of the secret ingredients that's going to transform our Recruiting app from a glorified database into a fully functional tool that everyone finds useful.

Now that we've got a general idea of what workflow's all about, let's take a closer look at the individual components that make up workflow: rules, tasks, field updates, and alerts.

Introducing Workflow Rules

In general, a *workflow rule* is the main container for a set of workflow instructions. It includes the criteria for when the workflow should be activated, as well as the particular tasks, alerts, and field updates that should take place when the criteria for that rule are met.

Every workflow rule must be based on a single object you choose when you define the rule. This object influences the fields that are available for setting workflow activation criteria.

For example, if we define a workflow rule for the Job Application object, we'll be able to set workflow activation criteria based on the values of fields like `Job Application Number` and `Status`. We can also set workflow activation criteria based on standard fields, like `Record Owner` or `Created Date`, as well as fields based on the currently active user when a rule is evaluated, such as their `Role` or `Time Zone`.

We'll look at all the ways that we can set workflow activation criteria when we get to building our own workflow rules a little later. For now, just understand that the platform makes it very easy to create detailed workflow rules that target specific situations.

Introducing Workflow Actions: Tasks, Field Updates, and Alerts

When a workflow rule is triggered, the following types of actions can occur:

Workflow Tasks

A *workflow task* assigns a task to a user according to a particular template. Just as in Microsoft Outlook, tasks include information about something that needs to be done by a certain time, such as making a telephone call or returning a library book. Assigned tasks appear in a user's `My Tasks` related list on their `Home` tab and generate reminder messages that pop up when a user logs in.

When we define a workflow task, we provide default values for the `Assignee`, `Subject`, `Status`, `Priority`, and `Due Date` fields for tasks that are generated by an associated workflow rule. We can also make sure that a notification email is sent to the assignee when a task is automatically generated.

Workflow Field Updates

A *workflow field update* changes the value of a particular field on the record that initially triggered the workflow rule.

Workflow Email Alerts

A *workflow email alert* sends an email according to a specified email template. Unlike workflow tasks, which can only be assigned to users of the app, workflow alerts can be sent to any user or contact, as long as they have a valid email address.



Note: A fourth type of action, a *workflow outbound message*, sends data to an external Web service, such as another application in the cloud. Outbound messages are used primarily with application programming interfaces, so we won't be using them in this chapter.

A workflow rule can include any combination of these actions when the rule is triggered. For example, one rule might send out an alert and update two fields on a particular record. The action that one workflow rule takes can also trigger the execution of another workflow rule.

Workflow in Our Recruiting App

Now that we've oriented ourselves to the different components involved with workflow, let's take a look at our Recruiting app and see how we can use workflow to help us build out the requirements that we talked about in [About the Sample Recruiting App](#) on page 17. Then we'll spend time implementing the rules that we come up with here.

We've already talked about one instance where workflow will give us a big advantage: automatically assigning a task to a recruiter when the status of a job application record changes to Reject or Extend an Offer. This will be a great start, but what else can we do?

If we look back at our last chapter on security and sharing, recall that we wanted to grant both recruiters and hiring managers permission to create new positions, but that ultimately we always wanted a recruiter to own those records because filling them is the recruiter's job responsibility. We hinted in the security and sharing chapter that we could accomplish this with workflow, and indeed we can! We'll simply need to use a workflow field update to change the record owner of a position record to a recruiter if it was originally created by a hiring manager. To prevent a single recruiter from getting overloaded with all these additional positions, we'll also use another platform feature, *queues*, to divvy up the orphaned position records fairly. We'll place the record in a queue of other position records without owners, and then let individual recruiters claim the positions they want.

For a third way of using workflow, let's think about how position availability is advertised throughout Universal Containers. Like many organizations, Universal Containers prefers to fill positions with employee referrals, but employees often aren't aware of which positions are currently open. We can use a workflow alert to automatically send email to every employee whenever a new position opens up. This way employees learn about positions as they become available and can immediately think of friends and family members who might be interested.

Sound good? While there are probably many more ways to use workflow to build a killer on-demand Recruiting app in the cloud, let's stick with these three for now since they'll give us a good example of each of the three types of workflow actions that are available. To summarize what we'll be building:

1. A workflow task that assigns a task to a recruiter when the status of a job application changes to Rejected or Extend an Offer
2. A workflow field update that reassigns ownership of a position that's created by a hiring manager to a queue of position records without owners, so that individual recruiters can claim ownership of the positions they want
3. A workflow alert that sends an email to every employee when a new position is created

Now let's get started!

Creating Workflow Rules That Assign Tasks

For our first use of workflow, we want to create two rules: one that assigns a “Send Rejection Letter” task to the relevant recruiter whenever the `Status` field of a job application record is set to Rejected, and one that assigns an “Extend Offer” task to the relevant recruiter whenever the `Status` field of a job application record is set to “Extend an Offer.” Let's start with the “Send Rejection Letter” workflow rule.

Try It Out: Creating the “Send Rejection Letter” Workflow Rule

To make this rule entirely functional, we'll need to define both a workflow rule, which specifies the criteria for when the rule should be executed, and a workflow task that includes the “Send Rejection Letter” task template. Although we can define workflow components in any order, let's start with defining the workflow rule itself. It'll save us a couple of clicks a little later when we define the rule's associated workflow task.

1. Click ***Your Name*** ► **Setup** ► **Create** ► **Workflow & Approvals** ► **Workflow Rules**.

2. If you see an introductory splash page, simply click **Continue**.
3. Click **New Rule**.

The first thing we need to do is select the object that will be associated with our workflow rule. As we talked about earlier, every workflow rule must be associated with a single object to determine the fields that we can use to set criteria. Since we need to trigger this workflow rule when the `Status` field of a job application record is set to “Reject,” we need to select Job Application here:

4. In the Select object drop-down list, choose Job Application, and click **Next**.

Now it's time to set our workflow rule details:

5. In the Rule Name text box, enter `Send Rejection Letter`.
6. In the Description text box, enter `Send a rejection letter when a hiring manager changes the status of a job application to Rejected`.

Now use the Evaluation Criteria area to specify when this rule should be evaluated. We can choose `When a record is created`, or `when a record is edited and did not previously meet the rule criteria (which repeatedly triggers the rule every time a record is saved only until the rule's criteria are met)`, `Only when a record is created (which ignores updates to existing records)`, or `Every time a record is created or edited (which repeatedly triggers the rule every time the record is saved)`. Since we don't want to assign duplicate tasks to a recruiter every time the record is saved, we'll choose the first option.

7. Under “Evaluate rule,” select `When a record is created, or when a record is edited and did not previously meet the rule criteria`.

To finish up defining the rule, we need to specify the conditions that will trigger execution of the rule's associated actions. We can do this by defining a set of criteria that trigger the workflow rule when met, or we can create a formula that triggers the workflow rule if it evaluates to `True`. While creating a formula provides more flexibility, setting the criteria is easier. For this rule, we can achieve our objective by setting the criteria, so let's do that.

Every workflow rule requires at least one row of filter criteria, but we can set as many filters as we want using additional rows.

8. In the first row of Rule Criteria filters:
 - Set the `Field` column to `Status`.
 - Set the `Operator` column to `equals`.
 - Set the `Value` column to `Rejected`.

9. Click **Save & Next**.

Edit Rule Rejection Letter [Help for this Page](#)

Enter the name, description, and criteria to trigger your workflow rule. In the next step, associate workflow actions with this workflow rule.

Edit Rule ! Required Information

Object: Job Application

Rule Name:

Description:

Evaluation Criteria

Evaluate rule: How do I choose?

When a record is created, or when a record is edited and did not previously meet the rule criteria

Only when a record is created

Every time a record is created or edited

Rule Criteria

Run this rule if the following criteria are met:

Field	Operator	Value
Status	equals	Rejected
-None-	-None-	
-None-	-None-	
-None-	-None-	

AND

AND

AND

AND

Advanced Options...

Save **Cancel**

The rule criteria define what triggers the workflow rule to fire.

Figure 69: Creating a New Workflow Rule

At this point, we've just defined our “Send Rejection Letter” workflow rule. If we canceled out of the workflow wizard and clicked **Your Name** ► **Setup** ► **Create** ► **Workflow & Approvals** ► **Workflow Rules**, we'd see it in the list view. However, because workflow rules aren't all that useful without an associated action, the workflow wizard takes us directly to a screen where we can define the “Send Rejection Letter” workflow task. Let's work through that now.

Try It Out: Creating the “Send Rejection Letter” Workflow Task

The Specify Workflow Actions step of the workflow wizard allows you to specify the workflow tasks, field updates, and alerts that should occur when the condition of our workflow rule is met.

The screenshot shows the 'Edit Rule Rejection Letters' page at Step 3 of 3. The page title is 'Edit Rule Rejection Letters' with a 'Help for this Page' link. The step indicator shows 'Step 3: Specify Workflow Actions' and 'Step 3 of 3' with a 'Done' button. Below the step indicator, there is a text prompt: 'Specify the workflow actions that will be triggered when the rule criteria are met. See an example'. The rule criteria are 'Job Application: Status EQUALS Rejected' and the evaluation criteria are 'When a record is created, or when a record is edited and did not previously meet the rule criteria'. There are two main sections for defining actions: 'Immediate Workflow Actions' and 'Time-Dependent Workflow Actions'. The 'Immediate Workflow Actions' section contains a text field with 'No workflow actions have been added.' and an 'Add Workflow Action' button. A red box highlights this section with the text: 'You can define actions that occur immediately after the condition is met, or before or after a certain amount of time has elapsed.' The 'Time-Dependent Workflow Actions' section contains a text field with 'No workflow actions have been added. Before adding a workflow action, you must have at least one time trigger defined.' and an 'Add Time Trigger' button.

Figure 70: Specifying Workflow Actions


You can either define actions that occur immediately after the condition is met, or you can define actions that occur before or after a certain amount of time has elapsed (for example, seven days before the value of the `Hire By` field, or three days after the workflow rule is triggered). We'll talk more about these *time-dependent workflow actions* a little later. For now, we just need to define a single workflow task that executes as soon as our rule criteria is met:

1. In the Immediate Workflow Actions area, click **Add Workflow Action** and select **New Task**.

Did you notice that the `Object` field is already filled in with `Job Application`? Here's where we saved ourselves a couple of clicks—if we'd started building our workflow task before our workflow rule, we would have needed to specify the object with which the task would be associated. That's because, like workflow rules, workflow actions need to be associated with a single object.

In this case, we got to skip a page because the object with which a workflow action is associated must match the object of a workflow rule that uses it. For example, if we had a workflow rule associated with the `Candidate` object, any task, field update, or alert that's triggered by our candidate workflow rule must also be associated with the `Candidate` object. Because we started building our “Send Rejection Letter” workflow task in the wizard right after defining our “Send Rejection Letter” workflow rule, the platform knew that the object associated with our task had to match the rule that we'd already built. That's why our new workflow task is already associated with the `Job Application` object.

The rest of the fields on this edit page make up the template for any “Send Rejection Letter” tasks that our workflow rule will generate.

2. Next to the `Assigned To` field, click the lookup icon ()

Here we can choose the assignee of the task either by specifying a particular user, role, or the owner of the job application record that triggered the workflow rule in the first place. Since recruiters always own the job application records of the positions that they're responsible for, and recruiters are responsible for sending rejection letters at Universal Containers, let's select the record owner:

3. In the `Type` drop-down list, choose `Owner`.
4. Click **Job Application Owner**.



Caution: If you thought that choosing the `Recruiter` role for the `Assigned To` field might have been another valid option, be careful. If the assignee of a workflow task is a role that contains more than one assigned user, the person who triggered the rule will become the task assignee instead. For this reason, you never want to assign workflow tasks to roles unless you're sure that only one user will ever be assigned to them at a time.

The rest of the workflow task fields are easy:

5. In the `Subject` text box, enter `Send Rejection Letter`.
6. In the `Due Date` drop-down lists, choose `Rule Trigger Date plus 2 days`.

This `Due Date` setting will give our recruiters two days to notify the candidate after they're first assigned the task.

7. In the `Status` drop-down list, choose `Not Started`.
8. In the `Priority` drop-down list, choose `High`.

The `Notify Assignee` checkbox allows us to send an email to the assignee as soon as the task is created by the workflow rule. This ensures that the assignee knows about the task without having to log in to the application on a regular basis, so let's select it.

9. Select the `Notify Assignee` checkbox.
10. Click **Save**.
11. Click **Done**.

Figure 71: Creating the “Send Rejection Letter” Workflow Task

At this point, we finally get to see the detail page for the workflow rule that we just created. It includes the workflow rule criteria and a list of the associated actions. The only thing that remains to be done is to activate the rule.

12. Click **Activate**.

All done! We've just created our first workflow rule and task. You'll find that the remaining workflow actions all operate in a very similar way, so we'll speed through the rest of them, focusing just on the fields and options that are unique to each. First, let's finish up our second workflow rule that assigns a task by quickly creating the “Extend an Offer” workflow rule.

Try It Out: Creating the “Extend an Offer” Workflow Rule and Task

To wrap up our first use case, we need to create another workflow rule for when the status of a job application is set to Extend an Offer. This rule and task are almost identical to the “Send Rejection Letter” workflow rule and task, so we'll just list the values that you'll need in the following two tables.

Table 35: Values for Creating the “Extend an Offer” Workflow Rule

Field	Value
Object	Job Application
Name	Extend an Offer
Description	Make an offer when a hiring manager changes the status of a job application to Extend Offer.
Evaluation Criteria	When a record is created, or when a record is edited and did not previously meet the rule criteria
Filter Criteria	Status equals Extend an Offer

Table 36: Values for Creating the “Extend an Offer” Workflow Task

Field	Value
Assigned To	Job Application Owner
Subject	Extend an Offer
Due Date	Rule Trigger Date plus 1 days
Status	Not Started
Priority	High
Notify Assignee?	Yes

All done! Make sure the “Extend an Offer” rule is also activated, and let's go try out one of our new workflow rules.

Look at What We've Done

Let's try out our new “Send Rejection Letter” workflow rule and see what happens:

1. Click the Job Applications tab, and select a job application record.
2. Click **Edit**, and change *Status* to *Rejected*.
3. Click **Save**.

The Send Rejection Letter task automatically appears in the Open Activities related list on the Job Application detail page, as shown in the following screenshot.

Job Application
JA-00002

Customize Page | Edit Layout | Printable View | Help for this Page

< Back to List: Pages

Open Activities (1) | Activity History (0) | Notes & Attachments (0) | Reviews (3)

Job Application Detail [Edit] [Delete] [Clone]

Job Application Number JA-00002 Owner Cynthia Capobianca [Change]

Candidate C-00008

Position Documentation Writer

Cover Letter

Status Rejected

Total Rating 9

Number of Reviews 3

Average Rating 3.00

Created By Cynthia Capobianca, 2/4/2010 2:58 PM Last Modified By Dave Carroll, 6/10/2010 12:45 PM

[Edit] [Delete] [Clone]

Open Activities [New Task] [New Event] Open Activities Help

Action	Subject	Name	Task	Due Date	Status	Priority	Assigned To
[Edit] [Cls]	Send Rejection Letter		✓	2/6/2010	Not Started	High	Cynthia Capobianca

New task created by the workflow rule

Figure 72: Open Activities Related List on Job Application Detail Page

Pretty neat, don't you think? Not only that, but if you check in the recruiter's email inbox, he should have received an automatically-generated email notification that looks like the following:

```
Caroline Roth has assigned a new task to you: Send Rejection Letter
Job Application: JA-00007 Due Date: 2/11/2007
For more details on this task, click on the link below:
https://na1.salesforce.com/00Tx04123s5k1
```

The link in the email message takes the recruiter directly to the task detail page, where he or she can locate the contact information of the candidate and update the task status after the task is completed. The task also shows up in the My Tasks area on the recruiter's Home tab, and a reminder message will pop up in another day if the recruiter hasn't yet changed the task's status to "Completed." Suddenly, our Recruiting app has become a lot more interactive and powerful!

Creating a Workflow Rule That Updates Fields

For our next use case, we want to create a workflow rule that ensures the owner of a new position record is always a recruiter. To do this, we're going to define a workflow field update that reassigns ownership of a position that's created by a hiring manager to a *queue* of position records without owners. Once a position record is in that queue, individual recruiters can claim

ownership of the positions they want. But before we jump ahead of ourselves, let's stop a moment. Just what, exactly, is a queue?

Introducing Queues

Much like a collection of items in a lost and found drawer, a queue is a collection of records that don't have an owner. Users who have access to the queue can examine every record that's in it and claim ownership of the ones they want.

Queues are traditionally used in sales and support organizations to distribute new leads and support cases to the employees who have the most availability. Because the platform natively supports queues for Leads, Cases, and any custom object, we can create a queue for the Recruiting app's Position object.

Try It Out: Creating a Queue for Positions

To define a queue, we simply need to know the types of records that can be placed in the queue (in our case, Positions), and the users who are allowed to pull records out of it (in our case, Recruiters). Once we know those two things, defining the queue itself is just a matter of a few simple clicks:

1. Click **Your Name** ► **Setup** ► **Manage Users** ► **Queues**.
2. Click **New**.
3. In the **Queue Name** text box, enter `Unclaimed Positions Queue`.
4. In the **Queue Email** text box, enter an email address for an individual or a distribution list, such as `recruiters@universalcontainers.com`.
5. Select **Send Email to Members**.

Notice here that if `recruiters@universalcontainers.com` was a real email distribution list that went to all recruiters, we wouldn't need to select **Send Email to Members**. We do it here only because `recruiters@universalcontainers.com` is a fake email address and can't be used for testing later. The following table outlines the options you have for notifying queue members when new records are added to the queue:

Table 37: Options to Specify How Queue Members are Notified of New Records

	Queue Email Not Specified	Queue Email Specified
Send Email to Members Checkbox Not Selected	Because a queue email is not specified, individual queue members are always notified,	Only the specified queue email address is notified.

	Queue Email Not Specified	Queue Email Specified
	regardless of the <code>Send Email to Members</code> checkbox.	
Send Email to Members Checkbox Selected	Because a queue email is not specified, individual queue members are always notified, regardless of the <code>Send Email to Members</code> checkbox.	The specified queue email address <i>and</i> individual queue members are notified. Note that if an individual queue member also receives emails sent to the specified queue email address, they'll receive duplicate notifications.

6. In the Supported Objects section, move Position into the Selected Objects list.

New Queue

[Help for this Page](#)

Queue Edit Save Cancel

Queue Name and Email Address ! = Required Information

Enter the name of the queue and the email address to use when sending notifications (for example, when a case has been put in the queue). The email address can be for an individual or a distribution list. When an object is assigned to a queue, only the queue members will be notified.

Queue Name:

Queue Email:

Send Email to Members:

If a queue email is not specified, individual queue members are always notified, regardless of the checkbox.

Supported Objects

Select the objects you want to assign to this queue. Individual records for those objects can then be owned by this queue.

Available Objects

- Candidate
- Case
- Employment Website
- Job Application
- Lead

Selected Objects

- Position

Add Remove

Queue Members

To add members to this queue, select a type of member, then choose the group, role, or user from the "Available Members" and move them to the "Selected Members." If the sharing model for all objects in the Queue is Public Read/Write/Transfer, you do not need to assign users to the queue, as all users already have access to the records for those objects.

Search: for: Find

Available Members

- Role and Subordinates: CEO
- Role and Subordinates: Channel Sales Team
- Role and Subordinates: Customer Support, International
- Role and Subordinates: Customer Support, North America
- Role and Subordinates: Dir Product Management
- Role and Subordinates: Director QA
- Role and Subordinates: Director, Channel Sales
- Role and Subordinates: Director, Direct Sales
- Role and Subordinates: Eastern Sales Team
- Role and Subordinates: Installation & Repair Services
- Role and Subordinates: Marketing Team
- Role and Subordinates: Recruiter
- Role and Subordinates: SVP, Human Resources
- Role and Subordinates: SVP, Sales & Marketing

Selected Members

- Role and Subordinates: Recruiting Manager

Add Remove

Save Cancel

Figure 73: Defining a Queue

As you can see, a single queue can handle multiple objects—the platform allows you to do this so that you don't have to define multiple queues for the same group of users.

7. In the Queue Members section, select **Roles and Subordinates** from the Search drop-down list.
8. Move **Role and Subordinates: Recruiting Manager** to the Selected Members list.
9. Click **Save**.

Perfect! We've just defined a new queue that can act as a temporary owner for all of the position records that are created by hiring managers. Whenever a position is placed in the queue, all recruiters are notified, and the appropriate person can claim ownership. All we need to do now is define the workflow rule that places those position records into the queue.

Try It Out: Creating a Workflow Rule That Updates Fields

Now that we've got our queue ready to go, we can go ahead and define our workflow rule:

1. Click *Your Name* ► **Setup** ► **Create** ► **Workflow & Approvals** ► **Workflow Rules**.
2. Click **New Rule**.
3. In the Select object drop-down list, choose `Position`, and click **Next**.
4. In the Rule Name text box, enter `Assign Position to Recruiter`.
5. In the Description text box, enter `Reassign position records to a recruiter if they were created by another type of employee`.

While we know that recruiters should almost always own position records, we don't want to impede the organization if there's a special case in which a non-recruiter should own the record instead. Let's choose to evaluate this rule only when a record is created so that if an exception needs to be made, the workflow won't supersede any changes that were made by a recruiter.

6. In the Evaluation Criteria area, select `Only when a record is created`.

Finally, we need to make sure that this rule is initiated whenever a position record is created by someone who isn't a recruiter or a recruiting manager. We can specify this filter criteria in a single row by using a comma in the Value column as follows:

7. In the Rule Criteria section:
 - Set the Field column to `Current User: Role`.
 - Set the Operator column to `not equal to`.
 - Set the Value column to `Recruiter, Recruiting Manager`.
8. Click **Save & Next**.


Now let's create the field update action for this workflow rule:

9. In the Immediate Workflow Actions area, click **Add Workflow Action**, and select **New Field Update**.
10. In the Name text box, enter `Reassign Position to Queue`.
11. In the Description text box, enter `Assign the Position to the Unclaimed Positions Queue`.

12. In the `Field to Update` drop-down list, choose `Owner`.

Once you make a selection in this drop-down list, new options appear just below depending on the selection that you made.

13. In the `Owner` drop-down list, choose `Queue`.

14. Click the lookup icon () , and select **Unclaimed Positions Queue**.

15. Select `Notify Assignee`.

16. Click **Save**.

Not too hard! Before we leave this workflow rule behind, though, let's give it a second workflow action—one that ensures that no positions will languish in the queue without being claimed by a recruiter. This time, we'll use a time-dependent workflow action.

Introducing Time-Dependent Workflow Actions

As we mentioned earlier, time-dependent workflow actions are actions that occur before or after a certain amount of time has elapsed (for example, seven days before the value of the `Hire By` field, or three days after the workflow rule is triggered). We can use time-dependent workflow actions to fire tasks, field updates, and email alerts while the condition of a workflow rule remains true.

For example, the goal of reassigning position records to the Unclaimed Positions queue is to have the appropriate recruiter take ownership. However, there might be situations in which a position is placed in the queue and no recruiter claims it. Rather than leaving the position to languish unclaimed in the queue, we can define a time-dependent workflow action that alerts the recruiting manager if no recruiter claims a position record within a certain number of days. Because this action only takes place while the workflow condition remains true (that is, while the position is owned by a non-recruiter), the manager will only be alerted when necessary.

Neat, huh? Let's see how time-dependent actions are defined.

Try It Out: Creating the “Notify Recruiting Manager” Time-Dependent Workflow Task

At this point, we should still be on the `Edit Workflow Rule` page for our `Assign Position to Recruiter Queue` workflow rule. If not, you can return there by clicking **Your Name** ► **Setup** ► **Create** ► **Workflow & Approvals** ► **Workflow Rules**, clicking `Assign Position to Recruiter`, and then clicking **Edit** in the `Workflow Actions` area.

Before we can define a time-dependent workflow task, we first must specify a *time trigger*. Time triggers define when the time-dependent workflow actions should fire.

1. Click **Add Time Trigger**.

In this case, we want our recruiting manager to be notified three days after a position has been assigned to the Unclaimed Positions queue.

2. Use the text box and drop-down lists to specify 3 Days After Rule Trigger Date.
3. Click **Save**.

The screenshot shows a dialog box titled "Add Time Trigger" with a "Help for this Page" link. Below the title is the word "Position". The main area is titled "Workflow Time Trigger Edit". It contains a "Workflow Rule" field with the value "Assign Position to Recruiter". Below this, there are three input fields: a text box containing "3", a dropdown menu set to "Days", and another dropdown menu set to "After". To the right of these is a dropdown menu labeled "Rule Trigger Date". At the bottom of the dialog are "Save" and "Cancel" buttons.

Figure 74: The Time Trigger Edit Page

Our time trigger is now listed in the Time-Dependent Workflow Actions area. The **Add Workflow Action** drop-down button is now active, and we can define our workflow task as usual.

4. In the Time-Dependent Workflow Actions area, click **Add Workflow Action** and select **New Task**.
5. In the `Assigned To` field, select the Recruiting Manager role.



Note: Remember, workflow tasks should only be assigned to a role if you're confident that only one user will ever be assigned to that role at a time. If the role contains multiple users, the owner of the workflow rule is assigned the task.

6. In the `Subject` field, enter `Assign Unclaimed Position Record to Recruiter`. When you move your mouse from the `Subject` field, the `Unique Name` field should be `Assign_Unclaimed_Position_Record_to_Recruiter`.
7. Set the `Due Date` field to `Rule Trigger Date plus 4 days`.

Since this workflow action won't fire until three days after the original `Rule Trigger Date`, making the `Due Date` four days after the `Rule Trigger Date` gives the recruiting manager one additional day to assign the position to a recruiter.

8. Set the `Status` to `Not Started`.

9. Set the `Priority` to `High`.
10. Select `Notify Assignee`.
11. Click **Save**.
12. Click **Done**.

Almost done! At this point, all we need to do is activate our workflow rule. However, if you click the **Activate** button now, an error message appears saying that the `Default Workflow User` must be set before activating the rule. What's that?

The default workflow user is the user who should be assigned as the owner of a workflow action if the user who originally triggered the rule is no longer active. If your organization uses time-dependent actions in workflow rules, you must designate a default workflow user. Salesforce displays this username in the `Created By` field for tasks, the `Sending User` field for email, and the `Last Modified By` field for field updates. Salesforce does not display this username for outbound messages. If a problem occurs with a pending action, the default workflow user receives an email notification. To set the default workflow user:

13. Click *Your Name* ► **Setup** ► **Create** ► **Workflow & Approvals** ► **Settings**.



Note: If you click **Activate** without previously setting the default workflow user and then click **OK** in the error dialog, you're sent directly to the **Workflow & Approvals Settings** page.

14. Set the `Default Workflow User` field to any user in your organization. In most cases, it's best to choose a system administrator.
15. Click **Save**.

Now we can activate the workflow rule:



Note: If you reached the **Workflow & Approvals Settings** page by clicking **Activate** and then **OK** in the error dialog, clicking **Save** also automatically activates your workflow rule.

16. Click *Your Name* ► **Setup** ► **Create** ► **Workflow & Approvals** ► **Workflow Rules**.
17. Click **Activate** next to the `Assign Position to Recruiter` workflow rule.

Look At What We've Done

Great! We've now got a workflow rule that moves position records created by non-recruiters to a queue of unclaimed positions, and if not claimed by a recruiter in three days, assigns a task to the recruiting manager.

To try it out, simply make sure that you're logged in as a hiring manager and create a new position. As soon as you return to the detail page for the position, you'll see that the Unclaimed Positions Queue has automatically been assigned as the record owner, and that any user assigned to the Recruiter or Recruiting Manager role will have received an email notification.

To actually view the contents of the queue, click the Positions tab, and choose Unclaimed Positions Queue from the View drop-down list. Any recruiter or recruiting manager can click the **Accept** button on this page and take ownership of the new position.

Although our Assign Unclaimed Position Record to Recruiter workflow task won't be activated for another three days, we can see that it's currently scheduled to fire by checking out the *workflow queue*. This queue lists all of the time-dependent workflow actions that are scheduled for the future. To view it:

1. Log back in to your organization as an administrator.
2. Click *Your Name* ► Setup ► Monitoring ► Time-Based Workflow.
3. Click Search.

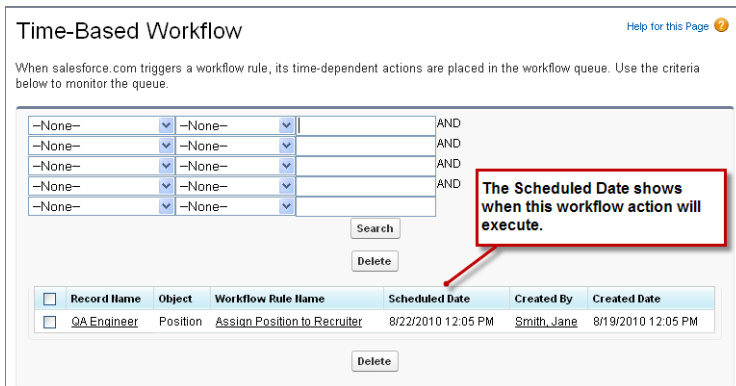


Figure 75: The Monitor the Workflow Queue Page

As soon as a recruiter takes ownership of the new position record, this task is deleted from the workflow queue. Pretty slick, isn't it?

Now let's build one final workflow rule so we can see how to create a workflow email alert.

Creating a Workflow Rule That Sends Email Alerts

For our final workflow use case, let's create a workflow rule and email alert that sends a notification whenever a new position is created. We want all the employees at Universal

Containers to know when there are new positions available so they have the best opportunity to bring in referrals.

For this workflow rule, there's a step that we'll need to handle first: we need to design a template for what the email alert should look like.

Introducing Email Templates

Just as the platform includes built-in tools for setting security permissions, tracking events and tasks, and building business logic with workflow, it also provides a built-in tool for writing emails to users and contacts in your organization. *Email templates* allow you to create form emails that communicate a standard message, such as a welcome letter to new employees or an acknowledgement that a customer service request has been received.

To personalize the content of an email template, we can use *merge fields* to incorporate values from records that are stored in the system. For example, if we wanted to address an email recipient by their first name, we could write an email template as follows:

```
Dear {!Contact.FirstName},
...

```

In this example, `{!Contact.FirstName}` is a merge field that brings in the first name of the contact to whom the email is addressed, so an email to John Smiley would read:

```
Dear John,
...

```

For our workflow alert, we can build an email template to notify users of new positions that have been added to the system. We can use merge fields to include information from the position record, such as its title and the required skills. Let's go do that now, and then we can get back to finishing up our final workflow rule.

Try It Out: Building an Email Template

To build a new email template, we have to go to the Administration Setup area:

1. Click **Your Name** ► **Setup** ► **Communication Templates** ► **Email Templates**.

Here you should see a list of all the email templates that have already been defined for your organization, including several sample templates from salesforce.com.

2. Click **New Template**.

We can choose to create a text, HTML, or custom email template. HTML and custom email templates are the same except that HTML templates allow you to specify a letterhead to give your email the same look and feel as other emails from the same source.



Note: A fourth option, Visualforce, lets developers create email templates using salesforce.com's tag-based markup language. We'll touch on Visualforce in Chapter 10.

To keep things simple, we'll just stick with a plain text email for now.

3. Select **Text**, and click **Next**.

Edit Text Email Template Help for this Page ?

Recruiting App: New Position Alert

Use merge fields to personalize your email content. You can add substitute text to any merge field. Substitute text displays only if the merge record does not contain data for that field. Enter substitute text after a comma in the merge field, for example, {!Contact.FirstName,Sir or Madam}. When you save the template, the merge field will appear in the email body of the template with the following syntax: {!NullValue(Contact.FirstName,"Sir or Madam")}. Click on the link below to see a sample email template.

[View Sample Template](#)

Note that the Description field is for internal use only. It will be listed as the title of any email activities you log when sending mass email.

Available Merge Fields

Select Field Type: Position Fields | Select Field: Position Title | Copy Merge Field Value: {!Position__c.Name}

Copy and paste the merge field value into your template below.

Email Template Edit Save Save & New Cancel

Email Template Information |= Required Information

Folder: Unfiled Public Email Templates

Available For Use:

Email Template Name: Recruiting App: New Po

Template Unique Name: Recruiting_App_New_P

Encoding: General US & Western Europe (ISO-8859-1, ISO-LATIN-1)

Description: Send update email to all Universal Containers employees.

Subject: New Position Alert: {!Position__c.Name}

Email Body:


```
{!Position__c.Responsibilities__c}

Skills Required
{!Position__c.Skills_required__c}

Educational Requirements
{!Position__c.Educational_Requirements__c}

If you know of anyone great who might be able to fill this role, please
contact the hiring manager, {!Position__c.Hiring_Manager__c}
```

Paste your merge field codes into the Subject and Email Body areas of your email template.

Save Save & New Cancel

Figure 76: Defining an Email Template

The New Template page allows us to define the email template itself. The gray area near the top is where we'll generate the merge field codes for the fields in the email template below it, so let's skip past it for now and start with the Folder drop-down list.

4. In the Folder drop-down list, choose `Unfiled Public Email Templates`.

The Unfiled Public Email Templates folder is a standard, public folder available in every organization. By keeping the email template in a public folder, it'll be available to other users who have permission to view and edit email templates.

5. Select the `Available For Use` checkbox.

This option will make our email template available when we create our workflow alert.

6. In the Email Template Name text box, enter `Recruiting App: New Position Alert`.



Tip: To help keep your email templates organized, it's a good idea to preface any template name with the name of the app that uses it. Or, even better, you can create a public email template folder with the name of the app, such as `Recruiting App Templates`, and file all the relevant email templates in it.

7. In the Encoding text box, accept the default of `General US & Western Europe (ISO-8859-1, ISO-Latin-1)`.
8. In the Description text box, enter `Send update email to all Universal Containers employees`.

Now we get to the heart of our email template—the email's subject and body text.

9. In the Subject text box, enter `New Open Position Alert:`.

We want to put the title of the new position in the subject of our email, so we'll need to use our first merge field here, just after the colon in our subject. To get the proper code, we'll have to go back to the gray merge field area near the top of the page.

10. In the Select Field Type drop-down list, choose `Position Fields`.

Although there are many objects to choose from in the Select Field Type drop-down list, because we're creating an email template for a workflow rule, we're limited to the fields for the object that will be associated with that workflow—in our case, `Position`. That's because the workflow rule that uses this email template won't know about any individual records other than the position record that triggered the rule's execution. If we put in fields from another object, they'd be left blank in our email, because there wouldn't be a record from which to pull the values.

Now let's grab the field we want.

11. In the Select Field drop-down list, choose `Position Title`.

In the `Copy Merge Field Value` text box, a merge field code appears for `Position Title`. We can cut and paste it to the end of our subject line so the subject now looks like this: `New Open Position Alert: {!Position__c.Name}`. When an email is generated from this template, `{!Position__c.Name}` will be replaced with the relevant position title.

Easy, right? Now let's finish the remainder of our email.

12. In the text area just below the `Subject` text box, enter the following text:

```
There's a new position open at Universal Containers!

Title: {!Position__c.Name}
Functional Area: {!Position__c.Functional_Area__c}
Location: {!Position__c.Location__c}

Job Description
{!Position__c.Job_Description__c}

Responsibilities
{!Position__c.Responsibilities__c}

Skills Required
{!Position__c.Skills_Required__c}

Educational Requirements
{!Position__c.Educational_Requirements__c}

If you know of anyone great who might be able to fill this role, please
contact the hiring manager, {!Position__c.Hiring_Manager__c}.

Thanks!
```

13. Click **Save**.

That's it for our email template. Now that it's done, we're ready to create our `New Position` workflow rule and alert.

Try It Out: Creating the New Position Workflow Rule and Alert

Now that we've built our email template, we're ready to build the workflow rule and email alert that use it. By now this procedure should be very familiar to you:


1. Click *Your Name* ► **Setup** ► **Create** ► **Workflow & Approvals** ► **Workflow Rules**.
2. Click **New Rule**.

3. In the Select object drop-down list, choose `Position` and click **Next**.
4. In the Rule Name text box, enter `Email New Position Alert`.
5. In the Description text box, enter `Send an email to everyone whenever a position record is opened`.

We only want this rule to execute once, whenever a position record's status is set to `Open - Approved`.

6. In the Evaluation Criteria section, select `When a record is created, or when a record is edited and did not previously meet the rule criteria`.
7. In the first row of rule criteria:
 - Set the Field to `Status`.
 - Set the Operator to `equals`.
 - Set the Value to `Open - Approved`. Although `New Position` is the default value of the `Status` picklist, we only want to publicize those positions that have been approved for hire by an executive.
8. Click **Save & Next**.

Now let's create the email alert for this workflow rule:

9. In the Immediate Workflow Actions area, click **Add Workflow Action**, and select **New Email Alert**.
10. In the Description text box, enter `Email New Position Alert`.
11. Next to the Email Template field, click the lookup icon () and select **Recruiting App: New Position Alert**.

We want to send this email to everyone at Universal Containers, but for this workflow rule there's no obvious way of doing that. We can work around this by relying on our role hierarchy and sending the email to everyone in the `CEO` role and its subordinates.

12. In the Recipient Type Search field, choose `Role and Subordinates`.
13. In the Available Recipients list, select `Role and Subordinates: CEO` and click **Add**.
14. Click **Save**.
15. Click **Done**.
16. Click **Activate**.

And that's all there is to it! To test out this workflow rule, all you need to do is create a new position record with a status of `Open - Approved`. Within a few seconds, all users within your

organization will receive an email letting them know that a position has just been created. Go ahead—try it!

Beyond the Basics

Did you know you can use the System Log Console to troubleshoot Apex, workflow rules, and validation logic?

Say you want to debug a workflow rule that isn't working correctly. You can use the console to step through the execution log of everything that occurred during the request, line by line, until you find the problem.

To find out more, see “About the System Log Console” in the Salesforce online help.

Introducing Approvals

Now that we've just made a handful of workflow rules, let's take a look at another business logic tool that the platform provides: *approval processes*.

Approval processes allow you to specify a sequence of steps that are required to approve a new record. Each step allows one or more designated approvers to accept or reject a record. The steps can apply to all records included in the process, or just to records that meet certain requirements. Like workflow, approval processes also allow you to specify actions—like sending an email alert, updating a field value, or assigning a task—that should occur whenever a record is approved, rejected, first submitted for approval, or recalled.

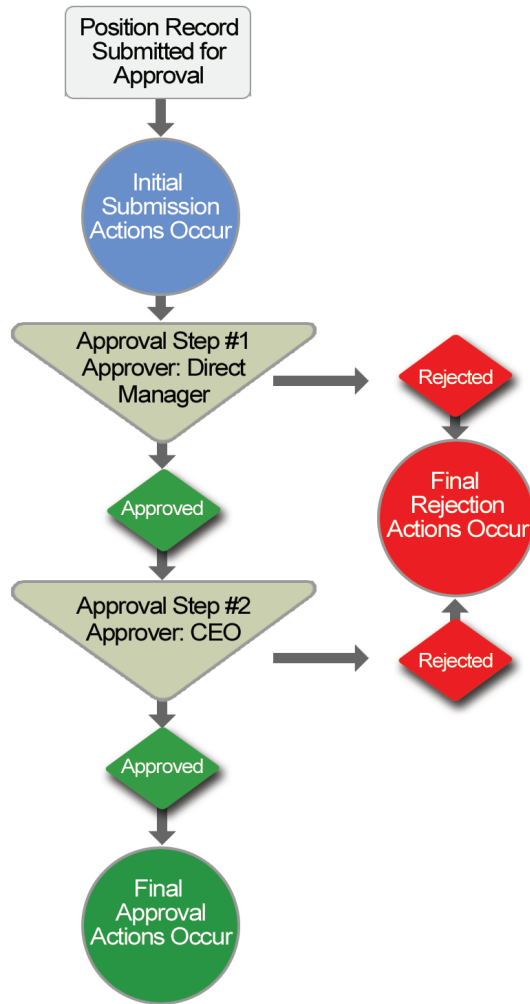


Figure 77: Approval Processes Consist of Steps and Actions

For example, suppose your organization has a three-tier process for approving expenses: submitted expenses that are less than \$50 are automatically approved, those over \$50 must be approved by a manager, and those over \$5,000 must also be approved by a Vice President. For this example, you can define an approval process that specifies the following:

- If an expense record is submitted for approval, lock the record so that users cannot edit it and change the status to “Submitted.”
- If the amount is \$50 or less, automatically approve the request.
- If the amount is greater than \$50, send an approval request to the direct manager.

- If the amount is greater than \$5,000 and the first approval request is approved, send an approval request to the Vice President.
- If all approval requests are approved, change the status to “Approved” and unlock the record.
- If any approval requests are rejected, change the status to “Rejected” and unlock the record.

For our recruiting app, we're going to define a similar approval process to submit new positions for approval. We want to make sure that a manager approves any position that his or her employee creates, and that any position with a minimum salary of more than \$150,000 is approved by the CEO. Let's get started.

Planning for Approval Processes

In most cases you'll find that you have to do a little advance planning before implementing an approval process on your own. The *Getting Started with Approval Processes* checklist available from the Tips and User Guides area of the online help outlines the things that you should think about and the components you should build before diving in.

For this approval process, the only preliminary step we need to do is define an email template that can be used to notify the designated approver that he or she has a pending approval request.

Try It Out: Creating an Email Template for Approvals

Since we've already talked about email templates in [Introducing Email Templates](#) on page 219, we'll simply include the values that we want to use for the template in the following table. You can build it from **Your Name** ► **Setup** ► **Communication Templates** ► **Email Templates**.

Table 38: The “Recruiting App: New Position Requires Approval” Email Template

Parameter	Value
Template Type	Text
Available For Use	Selected
Email Template Name	Recruiting App: New Position Requires Approval
Encoding	General US & Western Europe (ISO-8859-1, ISO-LATIN-1)
Description	Send notification email to designated approver when new position record requires approval.
Subject	New Position Requires Approval

Parameter	Value
Email Body	<p>A new position record has been submitted for your approval. Please visit the link below and either approve or reject it.</p> <p>{!Position__c.Link}</p> <p>Thanks!</p>

Try It Out: Creating an Approval Process

Now that we've finished our preparation, we're ready to define the approval process itself. The approval process definition acts as a framework for the approval steps and actions that we'll define later:

1. Click ***Your Name*** ► **Setup** ► **Create** ► **Workflow & Approvals** ► **Approval Processes**.
2. From the Manage Approval Processes For drop-down list, choose `Position`.

There are two different wizards that we can use to create a new approval process: a Jump Start Wizard and the Standard Setup Wizard. The Jump Start Wizard sets several default values for us and only requires input for the most crucial fields: the approval assignment email template, filter criteria to enter the approval process, and the designated approvers. The Standard Setup Wizard, on the other hand, allows us to configure every possible option for our approval process. We'll stick with it for now so we can take a look at all of the options that are available.

3. From the **Create New Approval Process** drop-down button, choose **Use Standard Setup Wizard**.
4. In the `Process Name` field, enter `Approve New Position`.
5. In the `Description` field, enter `Ensure that a manager approves any position that his or her employee creates, and that any position with a minimum salary of more than $150,000 is approved by the CEO.`
6. Click **Next**.

After entering the name and description, our next step is to specify the conditions that should be used to determine which positions need approval. As with workflow rules, we can do this by either defining a set of criteria or creating a formula. Let's define the criteria so that all positions created by a user other than the CEO must be approved by at least a direct manager.

7. In the first row of filter criteria, select `Current User: Role not equal to CEO`.
8. Click **Next**.
9. From the `Next Automated Approver Determined By` drop-down list, select `Manager`.

The `Manager` field is a standard field on the `User` object that designates the user's manager. The field establishes a hierarchical relationship between users, which prevents you from selecting a user that directly or indirectly reports to his or herself. This manager will be the designated approver for the first step of our approval process.

Alternatively, you could have selected the `Create New Hierarchical Relationship Field` option in the drop-down list to define a new custom *hierarchical relationship* lookup field on the fly. The hierarchical relationship field type is specifically designed for the `User` object, and mimics the behavior of the standard `Manager` field by associating one user with another without indirectly associating that user to him or herself. For this approval process, though, the standard `Manager` field is perfect, so let's move on.

Figure 78: Specifying the Approver Field

10. Select the `Use Approver Field of Position Owner` checkbox.

The `Use Approver Field of Position Owner` checkbox becomes editable when you select `Manager` in the `Next Automated Approver Determined By` drop-down list. When you select this checkbox, the approval request is routed to the user specified in the `Manager` field on the record owner's user record. If you don't select this checkbox, the approval request is routed to the manager of the user submitting the record. In our case, we want to obtain approval from the position owner's manager, so select this checkbox.

11. In the `Record Editability Properties` area, choose `Administrators ONLY can edit records during the approval process`.

Record editability allows you to specify whether a record that's been submitted for approval can be edited by the approver before being approved. Since we don't want managers to change the positions that a hiring manager or recruiter creates without alerting the owner, we'll only let administrators perform edits while a record is in our approval process.

12. Click **Next**.

13. In the Approval Assignment Email Template lookup field, select **Recruiting App: New Position Requires Approval**.
14. Click **Next**.

Our next step in defining the approval process is specifying which fields should be displayed on the Approver page layout, which the approver sees when he or she approves or rejects a record. Each approval process has its own page layout, and, unlike other page layouts, the Approver page layout can only be configured from within its approval process.

We can display any Position object fields on the Approver page layout, but since we've restricted the editability of the position record while it's in the approval process, these fields can't be edited by the approver until the record is approved or rejected.

15. Move the following fields from Available Fields to Selected Fields:

- Position Title
- Owner
- Hiring Manager
- Type
- Location
- Hire By
- Job Description
- Min Pay
- Max Pay

Figure 79: Defining the Record Approval Page Layout

On this page we can also specify whether approval history information should be displayed on the Approver page layout. This information shows whether this record was submitted for approval in the past, who were the designated approvers, and whether it was approved or rejected.

16. Select **Display approval history information in addition to the fields selected above.**

Finally, before leaving this page, we can specify security settings to determine whether an approver can approve or reject records from a wireless-enabled mobile device. Unless it's a mandatory requirement for your approvers, it's better not to choose this option because it prevents a user from manually selecting an appropriate approver for a record. We'll leave the default choice selected for now.

17. Click **Next.**

The last page of the New Approval Process wizard allows us to choose who should be allowed to submit position records for approval. Again, we'll just leave the default Record Owner selected, because there's no reason for another user to have this power.

The last two options on this page allow us to place the Approval History related list on all Position page layouts and give users the ability to recall pending approval requests after they've submitted them. The Approval History related list is the same history related list that we included on the Approver page layout, so we'll also include it on the Position detail page. From this related list users can also click the **Recall Approval Request** button to withdraw their pending approval requests. If we didn't enable this last option, only administrators would have access to the **Recall Approval Request** button.

18. Select Add Approval History Related List to All Position Page Layouts.

19. Select Allow submitters to recall approval requests.

20. Click **Save**.

Phew! We've finished defining the framework for our approval process, but we won't be able to activate it until we've given it some steps and some actions to fire when records are actually approved or rejected. Let's move on to those now.

21. Select Yes, I'd like to create an approval step now.

22. Click **Go!**

Try It Out: Creating Approval Steps

As we said earlier, every approval process consists of a set of steps that are required to approve the creation of a new record, and each step allows one or more designated approvers to accept or reject the submitted record. In other words, every round of “signatures” that you need to get a record approved must have a corresponding step in the approval process. An approval step consists of:

- One or more designated approvers
- Optional filter criteria, so that only records that meet certain conditions will require approval at that step
- Optional step approval actions that execute regardless of the outcome of the whole approval process
- Optional step rejection actions that execute regardless of the outcome of the whole approval process
- Optional step recall actions that execute if the record is recalled

For our New Position approval process, we'll need to define two steps—one that requires approval from the record submitter's manager for all new position records, and one that requires additional approval from the CEO for position records with minimum salaries in excess of \$150,000. Let's define the first step for all new position records now.

Because we selected `Yes, I'd like to create an approval step now` at the end of the Standard Approval Process wizard in the last section, we're already at the beginning of the New Approval Step wizard. If we weren't, we could return to the same wizard with the following steps:

1. Click **Your Name** > **Setup** > **Create** > **Workflow & Approvals** > **Approval Processes**.
2. In the Inactive Approval Processes related list, click **Approve New Position**.
3. In the Approval Steps related list, click **New Approval Step**.

In this first step, we want the approval request to go to the Position owner's manager:

4. In the `Name` field, enter `Manager Approval`.
5. In the `Description` field, enter `Every new position record must be approved by the Position owner's manager`.
6. In the `Step Number` field, enter `1`.

The `Step Number` field specifies the order in which multiple steps should be processed. By assigning this as `Step 1`, it will be the first to execute when the approval process is triggered.

7. Click **Next**.

The Specify Step Criteria area allows us to define the criteria or create a formula that filters the records that require approval during this step. Because we've already filtered out position records that are owned by the CEO from the whole approval process, this step does not need any additional filtering.

8. Click **Next**.

Finally, we have to select the assigned approver for this step, and specify whether his or her delegate is allowed to approve the request as well. Because this is the `Manager Approval` step, it clearly makes sense to accept the default option of `Automatically assign using the custom field selected earlier. (Manager)`. However, because position records aren't particularly sensitive, it's okay for managers to assign delegated approvers. So managers who go on vacation, or who receive large quantities of approval requests, can share their work with another employee.

9. Select `The approver's delegate may also approve this request`.
10. Click **Save**.

Having completed our first approval step, we're faced with another choice to create optional approval or rejection actions for this step, or to return to the approval process detail page. While we ultimately need to specify *final* approval and rejection actions that occur when the approval process ends one way or the other, there's nothing in particular that needs to happen after this first step that we can't specify elsewhere. Let's return to the detail page for our approval process and define our second approval step for positions with minimum salaries of more than \$150,000.

11. Select **No, I'll do this later**. Take me to the approval process detail page to review what I've just created.
12. Click **Go!**.
13. In the Approval Steps related list, click **New Approval Step**.

Once again we're back in the New Approval Step wizard, but this time it includes a summary of the previous step that we created. This helps us to remember where we are in our approval process.

New Approval Step Help for this Page ?

Step 1. Enter Name and Description Step 1 of 3

Enter a name, description, and step number for your new approval step.

Previous Approval Step Information

Step Number:	1
Name:	Manager Approval
Criteria:	
Assign To:	Manager

Enter Name and Description Required Information

Approval Process Name: Approve New Position

Name:

Unique Name:

Description:

Step Number:

Callout 1: A summary of the previous step reminds us of where we are in the approval process.

Callout 2: Step Number indicates the order in which approval steps will execute.

Next Cancel

Figure 80: Defining a Second Approval Step

14. In the Name field, enter **CEO Approval**.
15. In the Description field, enter **Every new position record with a minimum salary over \$150,000 must be approved by the CEO**.

16. In the `Step Number` field, enter 2.
17. Click **Next**.

For this approval step, we only want to send positions with a minimum salary over \$150,000 to the CEO. Additionally, we want to exclude any records that the CEO has already approved (for example, because one of the CEO's direct reports created the record).


18. Select the `Enter this step if the following criteria are met` from the drop-down list.
19. In the first row of filters, enter `Min Pay greater or equal 150000`.



Tip: As a short cut, you can specify 150,000 as 150k.

20. In the second row of filters, enter `Current User: Manager not equal to Cynthia Capobianco` (the acting CEO in [Universal Containers Role Hierarchy](#) on page 163).
21. Click **Next**.

Finally, we need to select the approver (the CEO), and specify what should happen if he or she rejects this request.

22. Select the `Automatically assign to approver(s)` radio button.
23. In the drop-down list below the radio button, select `User`, click the lookup icon () , and choose the name of the CEO in your organization (Cynthia Capobianco).

We're keeping this approval process fairly simple, but if we wanted to, we could use the **Add Row** and **Remove Row** links to send the approval request to multiple approvers in this step. We could also select `Related Users` in the first drop-down list in the row to add an approver who is listed in fields on the submitted record. For example, since this is an approval process for Position records, we could add the position's hiring manager to the list of approvers.

24. Below the **Add Row** and **Remove Row** links, select `Approve or reject based on the FIRST response`.

If this step was requesting approval from multiple users, the radio buttons below the **Add Row** and **Remove Row** links would determine whether the approve request needed unanimous approval, or if the record would be approved or rejected based on the first user to respond to the request.

25. Select `The approver's delegate may also approve this request`.

The next section allows us to specify what to do with the record if it's rejected at this step. Because the position record is locked from editing during the approval process, it makes the most sense to perform the final rejection.

26. Select `Perform all rejection actions for this step AND all final rejection actions.` (Final Rejection).

27. Click **Save**.

Once again, we're faced with a choice to define approval or rejection actions for this particular step. Let's circumvent those, and return to the approval process detail page to define our initial submission, final approval, and final rejection actions for the whole process.

28. Select `No, I'll do this later. Take me to the approval process detail page to review what I've just created.`

29. Click **Go!**

Try It Out: Creating Approval Actions

Now that we've finished defining our approval process steps, we're nearly done. All that remains is to specify the *approval actions* that should occur whenever a record is initially submitted, or when it's finally approved or rejected.

Just like workflow actions, approval actions allow you to create and assign tasks, update fields, and send email updates and outbound messages. They can either be associated with the approval process as a whole or with individual approval steps.

Approval Processes Help for this Page

Position: Approve New Position

[← Back to Approval Process List](#)

Process Definition Detail
[Edit](#) [Clone](#) [Deactivate](#) [View Diagram](#)

Process Name	Approve New Position	Active	✓
Unique Name	Approve_New_Position	Next Automated Approver Determined By	Manager of Record Owner
Description	Ensure that a manager approves any position that his or her employee creates, and that any position with a minimum salary of more than \$150,000 is approved by the CEO.		
Entry Criteria	Current User: Role NOT EQUAL TO CEO		
Record Editability	Administrator ONLY	Allow Submitters to Recall Approval Requests	✓

Approval Assignment: [Recruitment App: New Position Requires Approval](#)

Email Temp: [Create](#)

Initial Submit: [Create](#)

Modified By: [Jane Smith](#), 8/19/2010 2:36 PM

Initial Submission Actions
[Add Existing](#) [Add New](#)

Action	Type	Description
Edit Remove	Record Lock	Lock the record from being edited
Edit Remove	Field Update	Set Status to Pending Approval

Approval Steps
[Add Existing](#) [Add New](#)

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Show Actions Edit	1	Manager Approval	Every new position record must be approved by the Positions owners manager.			Final Rejection
Show Actions Edit	2	CEO Approval	Every new position record with a minimum salary over \$150,000 must be approved by the CEO.	(CURRENT_USER: Manager: NOT EQUAL TO Jane Smith)		Final Rejection

Final Approval Actions
[Add Existing](#) [Add New](#)

Action	Type	Description
Edit Remove	Record Lock	Lock the record from being edited
Edit Remove	Field Update	Set Status to Open Approval

Final Rejection Actions
[Add Existing](#) [Add New](#)

Action	Type	Description
Edit Remove	Record Lock	Unlock the record for editing
Edit Remove	Field Update	Set Close Date to Today
Edit Remove	Field Update	Set Status to Closed - Not Approved

Recall Actions
[Add Existing](#) [Add New](#)

Action	Type	Description
Edit Remove	Record Lock	Unlock the record for editing

Figure 81: The Approval Process Detail Page

Because defining an approval action is almost identical to the way we created workflow actions, we'll quickly step through the process of updating the status field to Pending Approval when a position is initially submitted and then leave our other approval actions as exercises:

1. If you're not on the approval process detail page already, click **Your Name** ► **Setup** ► **Create** ► **Workflow & Approvals** ► **Approval Processes**, and then click **Approve New Position**.
2. In the Initial Submission Actions related list, click **Add New**, and select **Field Update**.

Does the New Field Update Action page look familiar? That's because we've seen it before—all approval tasks, field updates, email alerts, and outbound messages use the same editing interface as workflow actions. In fact, every workflow and approval action that you create can be used interchangeably on both workflow and approval processes.

3. In the Name field, enter `Set Status to Pending Approval`.
4. In the Description field, enter `While a position is in an approval process, its status should be set to Pending Approval`.
5. In the Field to Update drop-down list, select `Status`.
6. In the Picklist Options area, select `A specific value` and choose `Pending Approval` from the list.
7. Click **Save**.

Now, to finish up the rest of the approval process, define the remaining approval actions on your own according to the values in the following table.

Table 39: Additional Approval Actions

Category	Type	Values
Final Approval Actions	Field Update	Name: <code>Set Status to Open Approved</code> Field to Update: <code>Status</code> A specific value: <code>Open - Approved</code>
Final Rejection Actions	Field Update	Name: <code>Set Status to Closed - Not Approved</code> Field to Update: <code>Status</code> A specific value: <code>Closed - Not Approved</code>
Final Rejection Actions	Field Update	Name: <code>Set Close Date to Today</code> Field to Update: <code>Close Date</code> Use a formula to set the new value: <code>TODAY ()</code>

Try It Out: Activating Our Approval Process

We're all done with defining our Approve New Position approval process, but in order to test it one final step remains: we've got to activate it. Before we do so, however, be careful! Once an approval process has been activated, its approval steps can no longer be edited, even if you deactivate it. The only way to make changes is to clone the existing approval process and make edits in the cloned one.

Once you're ready to test the approval process:

1. Click **Your Name** ► **Setup** ► **Create** ► **Workflow & Approvals** ► **Approval Processes** to return to the approval process list page.
2. Click **Activate** next to the Approve New Position approval process.

The Approve New Position approval process automatically moves to the Active list, and a new field is displayed: `Process Order`. This field is important if we're trying to use more than one approval process at a time because it tells us the order in which each approval process will be evaluated.

Look At What We've Done

As we've seen, approval processes are inherently complex and take a bit of work to set up. You have to walk through multiple screens and set several parameters, making it difficult to visualize and understand the approval process holistically.

Fortunately, the Force.com platform has a Process Visualizer that renders each approval process as a flowchart. The flowchart contains all of the critical details for each approval process, including the steps necessary for a record to be approved, the designated approvers for each step, the criteria used to trigger the approval process, and the actions that take place when a record is approved, rejected, recalled, or first submitted for approval.

To access the Process Visualizer, click the **View Diagram** button at the top of any approval process detail page.

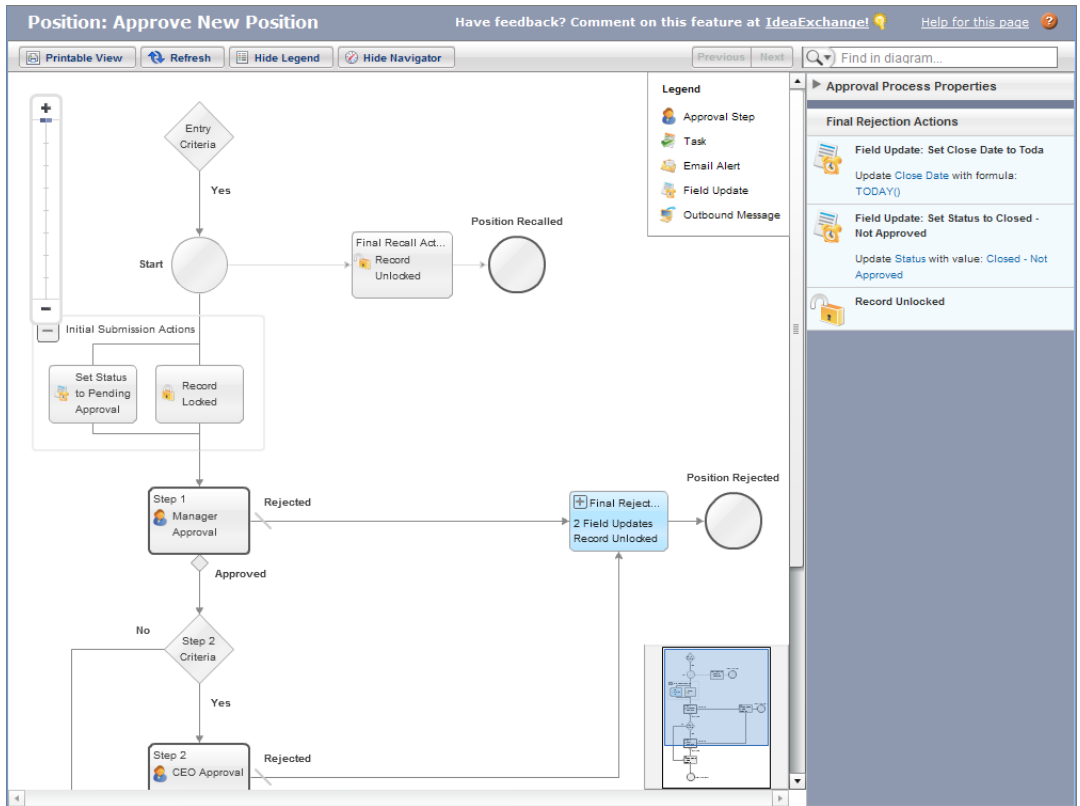


Figure 82: The Process Visualizer

Now let's test our approval process:

1. Click ***your Name*** ► **Setup** ► **Manage Users** ► **Users** and edit the user record for your Recruiting Manager to fill in the **Manager** field so the approval chain is properly set up.
2. Log in to your app as a Recruiting Manager and create a new position.

Notice that after clicking **Save**, the detail page displays a **Submit for Approval** button in the new Approval History related list.

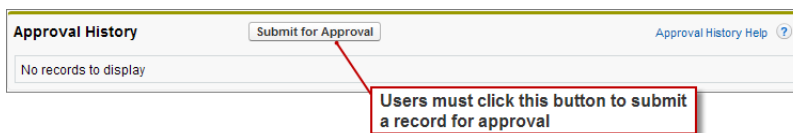


Figure 83: The Submit for Approval Button

3. Click **Submit for Approval** and then click **OK**.

Clicking the **Submit for Approval** button causes several things to happen. First, the record is locked from editing, as shown by the lock icon at the top of the page. Additionally, two new entries appear in the Approval History related list showing who submitted the record and the current approval assignment. Finally, the manager of the position owner receives an email reporting that there's a new position to approve.

The screenshot shows a 'Test Position' record with the following details:

- Position Detail:** Position Title: Test Position; Status: Pending Approval; Type: Part Time; Functional Area: Human Resources; Job Level: HR-200; Travel Required: [checkbox]; Owner: George Abitbol; Location: San Francisco, CA; Open Date: 6/4/2010; Hire By: 9/2/2010; Created By: George Abitbol (6/4/2010 4:11 PM); Last Modified By: George Abitbol (6/4/2010 4:12 PM).
- Compensation:** Min Pay: \$151,000.00; Max Pay: \$160,000.00.
- Description:** Job Description, Responsibilities, Skills Required, Educational Requirements.
- Required Languages:** Java, JavaScript, C#, Apex.
- Job Applications:** No records to display.
- Open Activities:** No records to display.
- Activity History:** No records to display.
- Notes & Attachments:** No records to display.
- Employment Websites:** No records to display.
- Approval History:**

Action	Date	Status	Assigned To	Actual Approver	Comments	Overall Status
Step: Manager Approval (Pending for first approval)						
Reassign Approve / Reject	6/4/2010 4:12 PM	Pending	Cynthia Casobianca	Cynthia Casobianca		Pending
Approval Request Submitted						
	6/4/2010 4:12 PM	Submitted	George Abitbol	George Abitbol		

Red callout boxes highlight:

- A lock icon above the 'Unlock Record' button with the text: "This icon shows that users cannot edit the record."
- Two entries in the Approval History list with the text: "Two new entries appear in the Approval History related list"

Figure 84: A Submitted Position Record

When the manager next logs in and visits the record, an **Approve/Reject** button is visible on the Approval History related list. He or she can click this button to see the approval request and approve or reject the record, with comments.

4. Log in as the direct manager who is responsible for approving the request.
5. Click **Approve/Reject** in the Approval History related list on the record, enter any optional comments, and then click **Approve**.



Tip: To make accepting and rejecting approval requests more convenient, consider adding the Items to Approve related list to the default Home page layout:

1. Click **Your Name** ► **Setup** ► **Customize** ► **Home** ► **Home Page Layouts**.
2. Click **Edit** next to the Dashboard Home Page Default.
3. Select **Items to Approve** and click **Next**.
4. Arrange the **Items to Approve** component on the page layout by moving it up and down the Wide (Right) Column list as desired.
5. Click **Save**.

Approval Request
Help for this Page ?

Position: Test Position

[« Back to Position: Test Position](#)

Approve/Reject Approval Request

Position Title: Test Position

Owner: [George Abitbol](#)

Hiring Manager:

Type: Part Time

Location: San Francisco, CA

Hire By: 9/2/2010

Job Description:

Min Pay: \$151,000.00

Max Pay: \$160,000.00

Comments:

Approvers can add comments along with their acceptance or rejection.

Approval History Approval History Help ?

Date	Status	Assigned To	Actual Approver	Comments	Overall Status
Step: Manager Approval (Pending for first approval)					
Pending					
6/4/2010 4:12 PM	Pending	Cynthia Capobianca	Cynthia Capobianca		
Approval Request Submitted					
6/4/2010 4:12 PM	Submitted	George Abitbol	George Abitbol		

Figure 85: The Approval Request Page

If the approver accepts the record, it progresses to the next step of the approval process (if its `Min Pay` field is greater than \$150,000 and the CEO still hasn't approved it), or else the position's `Status` field is set to `Open - Approved`. The record details remain locked to protect them from being changed, but recruiters can still associate the position with job applications, tasks, or other activities. If the record is rejected, its status is set to `Closed - Not Approved`, the `Close Date` field is set to today's date, and the record is unlocked in case it just needs a simple edit before it reenters the approval process. With just a few minutes of work, we've built an effective business process that will make all of Universal Containers' users more effective.

Summing Up

Check out our Recruiting app now! By leveraging the platform's built-in workflow and approval process tools, we've transformed our app from a glorified database into a fully functional application that provides real value to its users.

Next we'll tackle something that provides real value to our executive users: reports and dashboards that give our users a complete analytical picture of how the recruiting program at Universal Containers is going.

Chapter 11

Analyzing Data with Reports and Dashboards

In this chapter ...

- [Introducing Reports](#)
- [Introducing Dashboards](#)
- [Introducing Custom Report Types](#)
- [Look At What We've Done](#)

We've come a long way with our Recruiting app—not only do we have custom objects to store our data, we also defined security and sharing rules to protect that data, while making it easier for the hiring team to collaborate. And we added automation by implementing several business processes with workflow and approvals. We've built a functional application in the cloud, and we haven't even written a single line of code!

Now it's time to turn our attention to the needs of the Universal Containers managers and executive staff. Because they need to keep track of many different aspects of the business, we need a way to give them a bird's-eye view of the company's recruiting activity without forcing them to delve into piles and piles of data. To do this, we'll create a set of custom reports for our Recruiting app and then build a dashboard that allows users to view summaries of key Recruiting app statistics every time they log in.

Introducing Reports

We can help users monitor and analyze the data that's being generated in their organization by building *reports*. Reports are summaries of the data that's stored in an app. They consist primarily of a table of data, but can also include data filters, groupings, and a customized graph.

Help for this Page ?

Positions Open Longer Than 90 Days

Report Generation Status: Complete

Report Options: Optional data filters

Summarize information by: Location View: All positions

Time Frame
 Columns: Close Date Duration: Custom
 Start Date: End Date:

Run Report Hide Details Customize Save Save As Delete Printable View Export Details

Filtered By: Edit
Days Open greater or equal 90 Clear

Position	Position Title	Position: Owner Name	Days Open	Hire By	Open Date	Status
Location: San Francisco, CA (6 records)						
Functional Area: - (1 record)						
Sr. Developer		Cynthia Capobianca	126	5/3/2010	2/2/2010	New Position
Functional Area: Human Resources (1 record)						
Benefits Specialist		Cynthia Capobianca	125	5/4/2010	2/3/2010	New Position
Functional Area: Information Technology (4 records)						
Documentation Writer		Cynthia Capobianca	1,221	5/9/2007	2/3/2007	New Position
Sr. QA Engineer		Cynthia Capobianca	1,222	5/7/2007	2/2/2007	New Position
SW Engineer		Cynthia Capobianca	1,223	5/6/2007	2/1/2007	Open - Approved
Sr. UI Designer		Cynthia Capobianca	1,226	5/3/2007	1/29/2007	Open - Approved
Location: Austin, TX (1 record)						
Functional Area: Information Technology (1 record)						
DBA		Cynthia Capobianca	1,224	5/6/2007	1/31/2007	Open - Approved
Location: London, England (1 record)						
Functional Area: Retail Operations (1 record)						
Account Executive I		Cynthia Capobianca	1,220	8/9/2007	2/4/2007	Open - Approved
Location: Mumbai, India (1 record)						
Functional Area: Retail Operations (1 record)						
Account Executive III		Cynthia Capobianca	1,218	5/11/2007	2/6/2007	Open - Approved
Location: Sydney, Australia (1 record)						
Functional Area: Retail Operations (1 record)						
Account Executive IV		Cynthia Capobianca	1,217	5/12/2007	2/7/2007	Open - Approved
Grand Totals (10 records)						

Check rows above to filter to just those rows, then drill down by: -None- Drill Down

Edit Large | Medium | Small

Positions Open Longer Than 90 Days

Record Count

Optional graph

Figure 86: A Sample Report

While a comprehensive set of reports is included with every organization to provide information about standard objects such as contacts and accounts, we can also build custom reports that highlight interesting metrics about the data stored in our custom objects.

For example, an executive at Universal Containers might have the following questions about recruiting:

- On average, how many days does it take for each recruiter to fill a position?
- Which functional areas have the most open positions?
- Which positions have been open for more than 90 days?
- Which positions are getting the most candidates?
- Which employees conduct the most interviews?
- What does the job application pipeline look like for each open position?
- Who have we hired in the last 90 days?

We can answer all of these questions and more by creating custom reports in the Reports tab of the app. Although this tab isn't visible by default in our Recruiting app, any user can click the arrow tab on the right side of the tab bar to display all available tabs, and then navigate to the reports area by clicking **Reports**.

Report Formats

The platform supports three different report formats, each with varying degrees of functionality and complexity:

- *Tabular reports* are the simplest and fastest way to look at your data. Similar to a spreadsheet, they consist simply of an ordered set of fields in columns, with each matching record listed in a row. While easy to set up, they can't be used to create groups of data or graphs. Consequently, they're best used just for tasks like generating a mailing list.



Tip: Use tabular reports when you want a simple list or a list of items with a grand total.

- *Summary reports* are similar to tabular reports, but also allow users to group rows of data, view subtotals, and create charts. For example, in the sample Employee Interviewer reports that appear in the following screenshot, the summary report groups the rows of reviews by the possible values of the `Owner Name` field, allowing us to see at a glance subtotals of how many times the two interviewers have talked to candidates and entered reviews for them.

While a little more time-consuming to set up, summary reports give us many more options for manipulating and organizing the data, and, unlike tabular reports, they can be used in dashboards. Summary reports are the workhorses of reporting—you'll find that most of your reports tend to be of this format.



Tip: Use summary reports when you want subtotals based on the value of a particular field or when you want to create a hierarchically grouped report, such as sales organized by year and then by quarter.

- *Matrix reports* are the most complex kind of report available, allowing you to group records both by row and by column. For example, in the following sample Employee Interviewer reports, the matrix report groups the review rows by the possible values of the `Owner Name` field, and also breaks out the possible values of the `Position` field into columns. Consequently, the report gives us summarized information such as the number of times an interviewer has interviewed candidates and entered reviews for a particular position. These reports are the most time-consuming to set up, but they also provide the most detailed view of our data. Like summary reports, matrix reports can have graphs and be used in dashboards.



Tip: Use matrix reports when you want to see data by two different dimensions that aren't related, such as date and product.

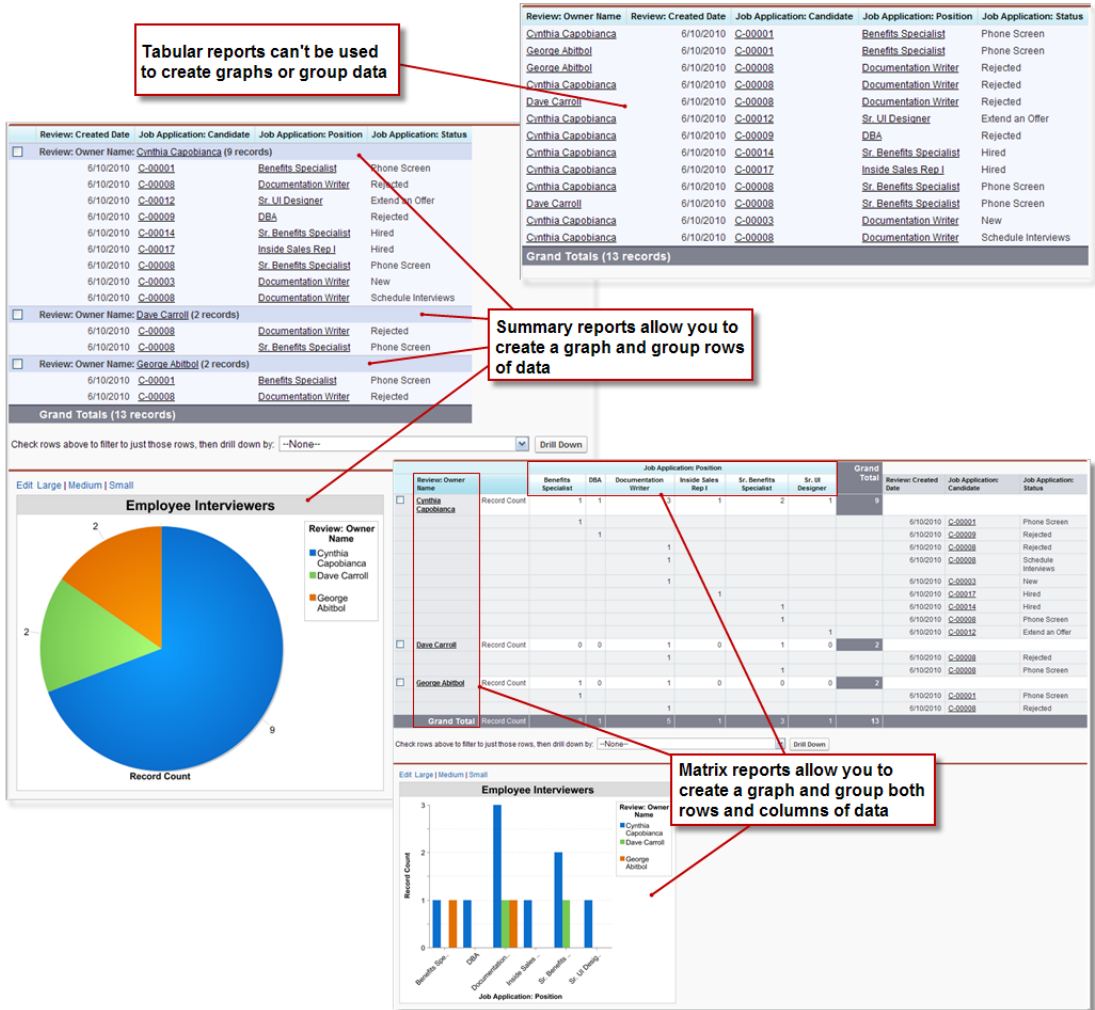


Figure 87: Tabular, Summary, and Matrix Reports Offer Different Options for Viewing the Same Data

Setting Up the Recruiting App for Reports

Before we get started building reports, we first need to take care of a couple tasks that will make our reports easy to find in the Recruiting app. We need to add the Reports tab to the default tab display along with our Positions, Candidates, and Job Applications tabs. We'll also need to create a folder for storing all the reports that we create. While both of these tasks are purely optional, they'll go a long way towards improving the experience of our Recruiting app users.

Try It Out: Adding the Reports Tab

First we'll start by adding the Reports tab to the set of default tabs that are displayed for every Recruiting app user. To do so, we need to revisit the Recruiting app that we created way back in [Building a Simple App](#) on page 29:

1. Click **Your Name** ► **Setup** ► **Create** ► **Apps**.
2. Click **Edit** next to the Recruiting app.
3. In the Choose the Tabs section, add Reports to the Selected Tabs list.
4. Optionally, select **Overwrite users' personal custom app customizations**.

If you choose this option, the Reports tab is automatically added to the tab bar by default for all users. If you've already deployed your app and you'd rather not overwrite existing users' changes, leave this option unchecked. Users can manually add the Reports tab to their personal tab bar by clicking the + tab.

5. Click **Save**.

Perfect! Now let's visit the Reports tab to perform our next task: creating a folder for Recruiting reports.

Try It Out: Creating a Recruiting Reports Folder

When you go to the Reports tab, you'll see that it displays a long list of all the standard and custom reports that are defined for your organization. As you can see by their headings, they're divided into categories to help users find what they're looking for. You can also quickly jump to a particular category by choosing it from the Folder drop-down list.

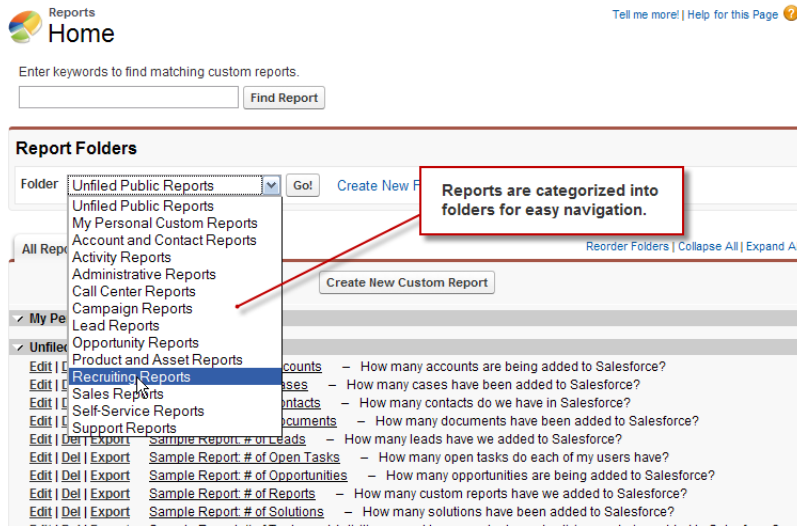


Figure 88: The Reports Tab and the Folder Drop-Down List

Because every organization already comes with over fifty standard reports and around a dozen report categories, it's important to create a new category whenever you're defining a new set of reports. It's an easy way of making sure that your users can always find the reports they need. To define a new report category, we simply have to define a new report folder:

1. In the Reports tab, click **Create New Folder** next to the Folder drop-down list.
2. In the Report Folder Label field, enter `Recruiting Reports`. The Folder Unique Name field autopopulates with `Recruiting_Reports`. Leave this default value.
3. In the Public Folder Access drop-down list, choose `Read Only`.

By choosing `Read Only` for this folder, only administrator users can modify the reports the folder contains, or save new reports to it. However, because all users can create their own reports, this won't hinder your users from modifying a report that we create and saving it to a personal folder of their own.

4. Select `This folder is accessible only by the following users`.
5. In the Search drop-down list, select `Roles and Subordinates` and then add `Role and Subordinates: CEO` to the Shared To list. This allows all users in your company to view this folder and the reports in it.
6. Click **Save**.

Great—we're now ready to create our first report.

Creating a Summary Report

To get started, we'll create a summary report that answers the question, "Which functional areas have the most new or open positions?" This report will include the title, hiring manager, location, and status of each position, as well as a pie chart that shows a visual representation of the data. As we do this, we'll skip over some of the more complicated reporting options so we can quickly create a report that fits our requirements. Our second example will then define a more complex matrix report that shows off some of the more advanced reporting features.

Because this report is going to use so many different reporting features, we'll break down the procedure into three parts:

1. Creating a summary report
2. Adding columns and filters
3. Adding a chart

Try It Out: Creating a Summary Report

To create our summary report, we'll start by opening the report builder, a powerful visual editor for reports:

1. On the Reports tab, click **Create New Custom Report**.

The first step in creating a report is choosing the right report type. A *report type* defines the set of records and fields available to a report based on the relationships between a primary object and its related objects. Reports display only records that meet the criteria defined in the report type. Your administrators may have set up custom report types for you, or you can select from the available standard report types.

To help with navigation in this screen, all objects and relationships are grouped in categories like Accounts & Contacts or Customer Support Reports. The custom objects and relationships that we built for our Recruiting app can be found in Other Reports.



Note: The Other Reports category contains all reports based on just custom objects. If you've built a custom object that's related to a standard object, such as Account or Contact, you'll also be able to report on your custom object in the standard object's category.

2. From the Create New Report page, select the Other Reports category.

Objects that have a many-to-one relationship with another object, like Job Applications and Positions, can either be selected on their own or in the context of their relationship with the

other object. For example, if we select Job Applications with Position, our report will count job application records, but can filter, group, or display fields from the related position records as well. This will come in handy a little later when we build reports that count job application records. But because we need to count position records in our report and positions aren't on the many side of a relationship, we'll stick with a standalone positions report for now.

3. Select Positions.
4. Click **Create**.

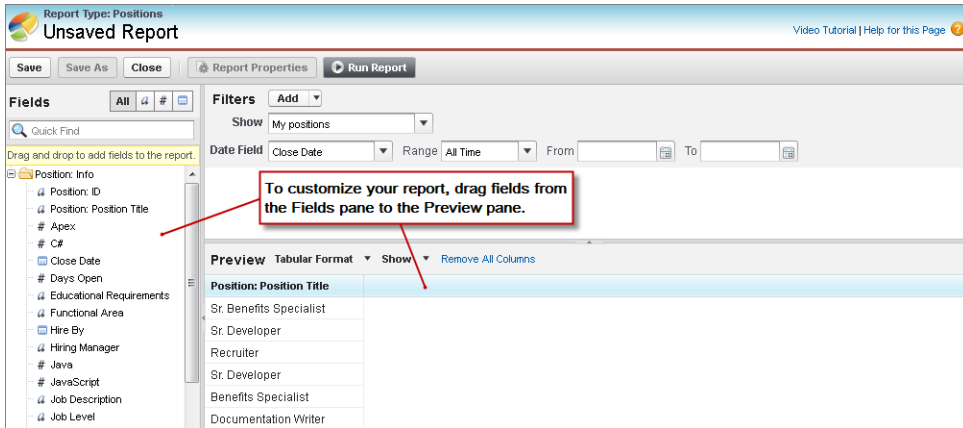


Figure 89: Custom Report in Report Builder

Now that we've chosen the report type, we can customize the report. Note that we can't go back and change the report type. To do that, you have to start with a whole new report. The report builder's Preview pane displays a limited set of data; we can also run the report at any time to see if we have the results we're looking for. For example, let's see what a baseline positions report looks like without any customizations.

5. Click **Run Report**.

Position Report Help for this Page ?

Report Generation Status: Complete

Report Options:

Summarize information by: Show
 -None- My positions

Time Frame

Date Field Range
 Close Date Custom
 From To

Run Report Hide Details Customize Save As Printable View Export Details

Position: Position Title

- Sr. Benefits Specialist
- Sr. Developer
- Recruiter
- Sr. Developer
- Benefits Specialist
- Documentation Writer
- Grand Totals (6 records)**

Figure 90: A Positions Report with No Customizations



Note: The Show drop-down list defaults to My Positions. If you are logged in as a user who does not own any positions, select All Positions before you click **Run Report**. The report will then display all the positions to which you have access.

As you can see, without specifying any details, we already have a list of position records with a Grand Total at the bottom. This is the equivalent of what we might see if we were creating a tabular report without additional columns. Now let's take this basic report to the next level.

6. Click **Customize to return to the report builder.**

When creating a new report, we first need to choose the format of the report that we want to create. The default is tabular. Because we want to group rows of open positions by functional area, we'll create a summary report.

7. Click **Tabular Format and select **Summary** to change the format.**

Now we want to group our rows of data by the `Functional Area` field.

8. Find `Functional Area` in the Fields pane and drag it to the grouping drop zone in the Preview pane. When you have lots of fields, using **Quick Find is usually fastest.**

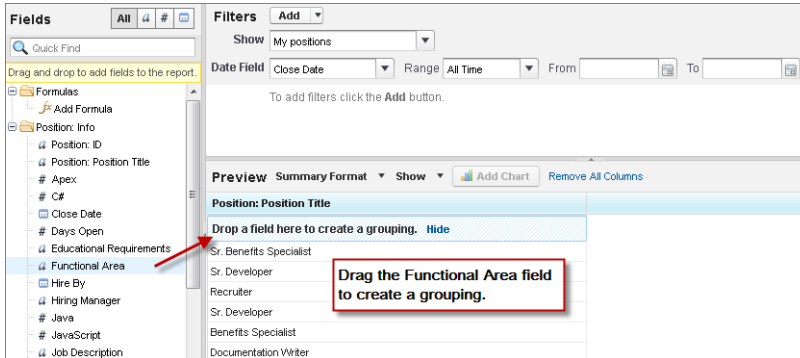


Figure 91: Adding a Grouping for a Report

We're closer now! Our report still only lists position titles, but now they're grouped by the functional areas to which they belong. Also, each grouping has a record count subtotal in parentheses. These counts will be the basis of the pie chart that we'll add later.

Try It Out: Adding Columns and Filters

Now, let's select and order the fields shown, and set the filters that restrict the set of records included in the report.

In addition to `Position Title`, which is already selected by default, we also want to display the `Hiring Manager`, `Location`, and `Status` fields for each record. To select multiple fields or columns, press **CTRL** (Windows) or **Command** (Mac).

1. In report builder, drag the following fields into the preview:

- `Hiring Manager`
- `Location`
- `Status`

You can also double-click fields to add them to the end of the report.

2. Reorder columns by dragging them.

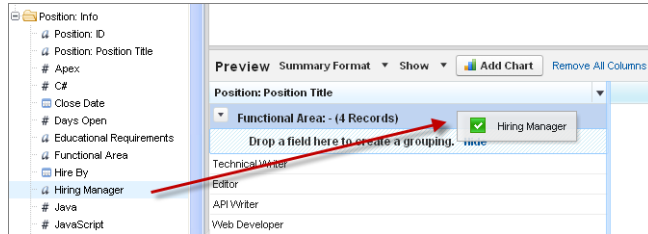


Figure 92: Adding a Column Using Drag-and-Drop

Let's run our report again.

3. Click **Run Report**.

Position	Position Title	Hiring Manager	Location	Status
Functional Area: - (2 records)				
Sr_Developer		Jane Teeole	-	-
Sr_Developer			San Francisco, CA	New Position
Functional Area: Human Resources (3 records)				
Benefits Specialist		Jane Teeole	San Francisco, CA	New Position
Recruiter		Jane Teeole	San Francisco, CA	Closed - Canceled
Sr_Benefits Specialist		Jane Teeole	San Francisco, CA	Closed - Filled
Functional Area: Miscellaneous (1 record)				
Documentation Writer		Jane Teeole	San Francisco, CA	New Position

Figure 93: A Positions Report with Grouping and Additional Columns


We're even closer to our goal. However, the `Status` field shows us that the report includes some position records that have already been filled. Since we only want to view the new and open ones, we need to set some filters.

4. Click **Customize** to return to the report builder.

Now, let's set up the filters that should be applied to this report. Filters define the set of records to be included in the report; for example, you can include only records created this month or only records that belong to a certain user. With standard filters, you can quickly filter by record owner or date field. With field filters, you can filter on any field in the report. Because we want to view open positions across the entire organization and not just positions that we own (by default, all custom reports include "My" records only), we need to set two filters:

5. In the Show drop-down, select All Positions.
6. Click **Add ► Field Filter**.
7. Define a filter for `Status equals New Position, Pending Approval, Open - Approved`.



Tip: Notice that whenever you choose a checkbox field or a picklist field, like `Status`, in your filter, a lookup icon () shows up next to the filter row. Click the icon to view valid values for that field and quickly insert the ones by which you want to filter.

Using this filter of `Status equals New Position, Pending Approval, Open - Approved` means that our report will include only those position records with one of these three statuses. Note that the comma between the three `Status` values is treated as an OR function, so this one filter is the same as using these three filters:

- `Status equals New Position, OR`
- `Status equals Pending Approval, OR`
- `Status equals Open - Approved`

Try It Out: Adding a Pie Chart

The final step is to specify the chart to display with the report. In our case, we want a pie chart:

1. Click **Add Chart** and choose the pie chart type.

The chart builder automatically knows that we want the values to be the record count, and the wedges to be the different functional areas. That's because chart values correlate to a report's summary fields, and pie chart wedges correlate to a report's groupings. Had we grouped our data with an additional column (like `Position Owner`), we would have had a choice of which field values to display in the wedges.

Now let's finish up our chart and generate our final report:

2. Click **OK**.

Position Report Help for this Page

Report Generation Status: Complete

Report Options:

Summarize information by: Show
 Functional Area: My positions

Time Frame
 Date Field: Close Date
 Range: Custom
 From: To:

Run Report Hide Details Customize Save As Printable View Export Details

Position	Position Title	Hiring Manager	Location	Status
<input type="checkbox"/> Functional Area: - (2 records)				
	Sr_Developer	Jane Teagle	-	-
	Sr_Developer	Jane Teagle	San Francisco, CA	New Position
<input type="checkbox"/> Functional Area: Human Resources (3 records)				
	Benefits Specialist	Jane Teagle	San Francisco, CA	New Position
	Recruiter	Jane Teagle	San Francisco, CA	Closed - Canceled
	Sr_Benefits Specialist	Jane Teagle	San Francisco, CA	Closed - Filled
<input type="checkbox"/> Functional Area: Miscellaneous (1 record)				
	Documentation Writer	Jane Teagle	San Francisco, CA	New Position

Figure 94: The Final “Open Positions by Functional Area” Summary Report

Terrific! Because our report meets all of our criteria, let's save it to the Recruiting Reports folder.

3. Near the top of the report, click **Save As**.
4. In the Report Name field, enter `Open Positions by Functional Area`.
5. In the Report Description field, enter `Which functional areas have the most new or open positions?`
6. In the Report Unique Name field, enter `Open_Positions_by_Functional_Area` if that's not already the value.
7. From the Report Folder drop-down list, select `Recruiting Reports`.
8. Click **Save**.

Now if we view the Recruiting Reports folder, we can see our new summary report. Next, we'll make a matrix report that takes advantage of some of the more advanced reporting features.

Creating a Matrix Report with Summary Fields, Time-Based Filters, and Conditional Highlighting

For our next report, we'll answer the question, “On average, how many days does it take each recruiter to fill a position with or without required travel?” This report will use a matrix format to highlight the difference that mandatory travel makes in how long it takes for positions to be filled. Looking at data only for this year and the last, the report will include:

- Record counts for each position a recruiter owns

- A custom summary formula for the percentage of a recruiter's positions that require travel
- A time-based filter that restricts the data only to positions that were created this year or last
- A color-coded field summary for the average number of days positions remain open:
 - Averages of less than 30 days are color-coded green
 - Averages of between 30 and 60 days are color-coded yellow
 - Averages of more than 60 days are color-coded red

Because this report is going to use so many different reporting features, we'll break down the procedure into four parts:

1. Creating a matrix report
2. Adding custom summary fields
3. Adding columns and filters
4. Adding a chart and conditional highlighting

Try It Out: Creating a Matrix Report

To create the matrix report, we'll use the report builder again, this time highlighting steps that are different from how we defined the summary report.

1. In the Reports tab, click **Create New Custom Report**.

Once again, we'll be creating a report that counts position records.

2. From the list, choose Other Reports.
3. Select Positions, and click **Create**.

Because we want to directly compare individual recruiter performance for positions that do and do not require travel, we'll use a matrix report. That way we can group the rows of positions by recruiter, and columns of positions by whether or not they require travel.

4. From the Format drop-down, select `Matrix`.

The screenshot shows a report preview interface with a table. The table has three columns: 'Record Count', 'Grand Total', and 'Position: Position Title'. The 'Record Count' column contains the value '1' for each row. The 'Grand Total' column contains the value '18' for the bottom row. The 'Position: Position Title' column lists various job roles. The interface includes a 'Preview' button, a 'Matrix Format' dropdown, a 'Show' dropdown, an 'Add Chart' button, and a 'Remove All Columns' button. There are also two drop zones: 'Drop a field here to create a column grouping.' and 'Drop a field here to create a row grouping.'. Below the table, a note states: 'This preview shows a limited number of records. Run the report to see all results.'

Record Count	Grand Total	Position: Position Title
1		Developer
1		Sr. Benefits Specialist
1		Sr. UI Designer
1		Inside Sales Rep I
1		DBA
1		SW Engineer
1		Sr. SQA Engineer
1		Documentation Writer
1		Account Executive I
1		Account Executive II
1		Account Executive III
1		Account Executive IV
1		Recruiter
1		Technical Writer
1		Editor
1		QA Lead
1		API Writer
1		Web Developer
Grand Total	18	

Figure 95: Specifying Groupings in a Matrix Report

Now let's group by both rows and columns. Notice the two sets of drop zones for matrix reports.

5. From the Fields pane, drag `Position: Owner Name` to the row grouping drop zone.
6. Now, drag `Travel Required` to the column grouping drop zone.

Our report now breaks out the possible values for the `Travel Required` field in the columns dimension. Recruiters are also broken out in the row dimension, but because all custom reports query just the report creator's data by default (that is, “My” records only), only one recruiter is listed in the report so far. Let's keep going.

Try It Out: Adding Summary Fields

In the next step, we want to specify which numerical or checkbox field values to include in our report, and how each of them should be summarized in subtotals and grand totals. Though record count is always summarized as a sum total, we can summarize other numerical and checkbox fields in different ways.

For example, it doesn't make sense to sum the values of the `Days Open` column—the resulting total wouldn't provide much value. However, if we calculate the average for `Days Open`, we'd know roughly how long positions stay open.

For our report, we need to include three different types of summaries: record count, average days open, and a formula that calculates the percentage of records requiring travel. While the first two are standard summary fields, the third will require a visit to the Custom Summary Formula editor. Let's start with the first two. By default, the report already adds Record Count, so let's add the average for the number of days a position stays open.

1. Find the Days Open field and drop it just above the Record Count row in the Preview pane.
2. In the Summarize dialog, select Average.

Now, let's create that formula to calculate the percentage of records requiring travel.

3. Double-click **Add Formula**. You'll see the Custom Summary Formula editor.

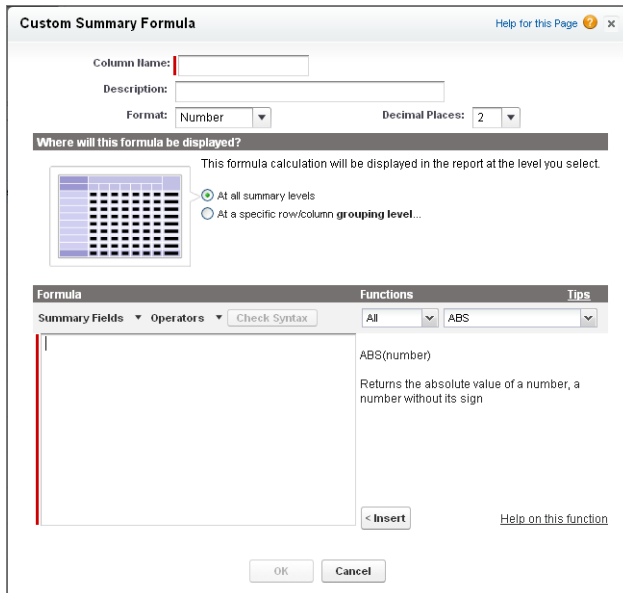


Figure 96: The Custom Summary Formula Builder

The formula editor lets us define a new formula based on the summarizable fields in the report. In our case, we want to include a summary that shows the percentage of position records that require travel in any given segment. To make this calculation we need to divide the sum of records that require travel by the sum of all records:

4. In the Column Name field, enter Travel Required Percentage.
5. In the Description field, enter The percentage of total position records that require travel.
6. In the Format drop-down list, choose Percent.

7. In the `Decimal Places` drop-down list, choose 0.
8. For `Where will this formula be displayed?`, choose `At all summary levels`.

Now, let's write our formula. Similar to other formula editors in the platform, this provides tools to make make it easier.

9. In the `Formula` section, click **Summary Fields** and select `Travel Required`, then select `Sum`.

The formula editor displays the following API representation of those values:

```
Position__c.Travel_Required__c:SUM
```

10. Click the **Operators** drop-down and select **Divide**.
11. Click **Summary Field** and select `Record Count`.

The final formula looks like this:

```
Position__c.Travel_Required__c:SUM / RowCount
```

We can quickly verify that the formula is correct by checking its syntax before saving.

12. Click **Check Syntax**.
13. Click **OK**.

Try It Out: Adding Columns and Filters

In the next steps, we'll select report columns, then define our filters. Because we're already familiar with adding columns, let's zip through that step first.



Note: Before you can add fields to a matrix report, make sure to select **Show ► Details**. If details aren't shown, you can only add summary fields.

1. Add the following report columns by double-clicking them:

- `Position: Position Title`
- `Functional Area`
- `Status`

`Days Open` and `Travel Required` should already be part of your report.

For our report, we want to define three filters: one to include all positions, one to include only those positions created in the last year, and one to include those with a Status of Open - Approved or Closed - Filled.

2. In the Show drop-down list, choose All Positions.
3. In the Date Field filter select Position: Created Date.

Notice that all other date fields defined on the Position object are also available in the Date Field filter, including Close Date, Hire By, and Open Date.

4. In the Range drop-down list, choose Current and Previous CY (meaning this and last calendar year). The start and end dates are populated automatically.

Now let's add a field filter based on the status:

5. Click **Add ► Field Filter**.
6. Create the following filters:
`Status equals Open - Approved, Closed - Filled`
7. Click **OK**.

Try It Out: Adding a Chart and Conditional Highlighting

We're almost done. Now, in addition to creating a horizontal bar chart that shows the average number of days open, recruiters, and whether the position requires travel, we also want to define some conditional highlights to help us quickly analyze which recruiters are performing well and which need to work on filling their positions faster.

1. Click **Add Chart**.
2. Select the Vertical Bar Chart type.
3. In the Y-Axis drop-down list, choose Average of Days Open.
4. In the X-Axis drop-down list, choose Position: Owner Name.
5. In the Group-By drop-down list, choose Travel Required and keep the side-by-side grouping format.
6. Leave the Plot additional values checkbox deselected.

A *combination chart* plots multiple sets of data on a single chart. Each set of data is based on a different field, so values are easy to compare. You can also combine certain chart types to present data in different ways in a single chart. The report we're building doesn't need a combination chart, but they are extremely useful when comparing data, charting trends, and so forth.

7. Click the **Formatting** tab.
8. In the **Chart Title** field, enter `Avg Days to Hire With and Without Travel`.
9. Click **OK**.

All three of our summary fields are available to highlight, but we just want to emphasize one: `Average Days Open`. That's because we want to highlight which recruiters are closing positions in less than 30 days, less than 60 days, or more than 60 days.

10. Click the **Show** drop-down list and select **Conditional Highlighting**.
11. In the first row, set the **Select Field** drop-down list to `Average Days Open`.
12. Set the **Low Color** to a shade of green using the color picker.
13. In the **Low Breakpoint** field, enter 30.
14. Keep the **Mid Color** the default shade of yellow.
15. In the **High Breakpoint** field, enter 60.
16. Set the **High Color** to a shade of red.
17. Click **OK**.
18. Click **Run Report**.

Our report succeeds in showing how well our recruiters fill positions and provides at-a-glance insight into how long, on average, positions stay open. Conditional highlighting shows which positions are taking longer to fill—generally those requiring travel. Let's quickly save this report before moving on.

19. Near the top of the report, click **Save As**.
20. In the **Report Name** field, enter `Avg Days to Hire With and Without Travel`.
21. In the **Report Description** field, enter *On average, how many days does it take each recruiter to fill a position with or without required travel?*
22. In the **Report Unique Name** field, enter `Avg_Days_to_Hire_With_and_Without_Travel` if that's not already the value.
23. From the **Report Folder** drop-down list, select **Recruiting Reports**.
24. Click **Save**.

As we've seen, custom reports can provide a lot of interesting data, giving insight into the challenges that an organization faces. However, unless a user visits these reports on a regular basis, much of their benefit remains untapped. How can we give users a way of keeping tabs on the information in reports without wasting their time? The answer, as we'll see next, lies with dashboards.

Introducing Dashboards

A *dashboard* shows data from source reports as visual components, which can be charts, gauges, tables, metrics, or Visualforce pages. They provide a snapshot of key metrics and performance indicators for your organization. Each dashboard can have up to 20 components. Users can view any dashboard available in a public folder in their organization, such as Company Dashboards, and can select a favorite, whose first three components display on the Home tab.

To put it mildly, users *love* the summarized views they get with dashboards, and no good Force.com app is complete without at least one.

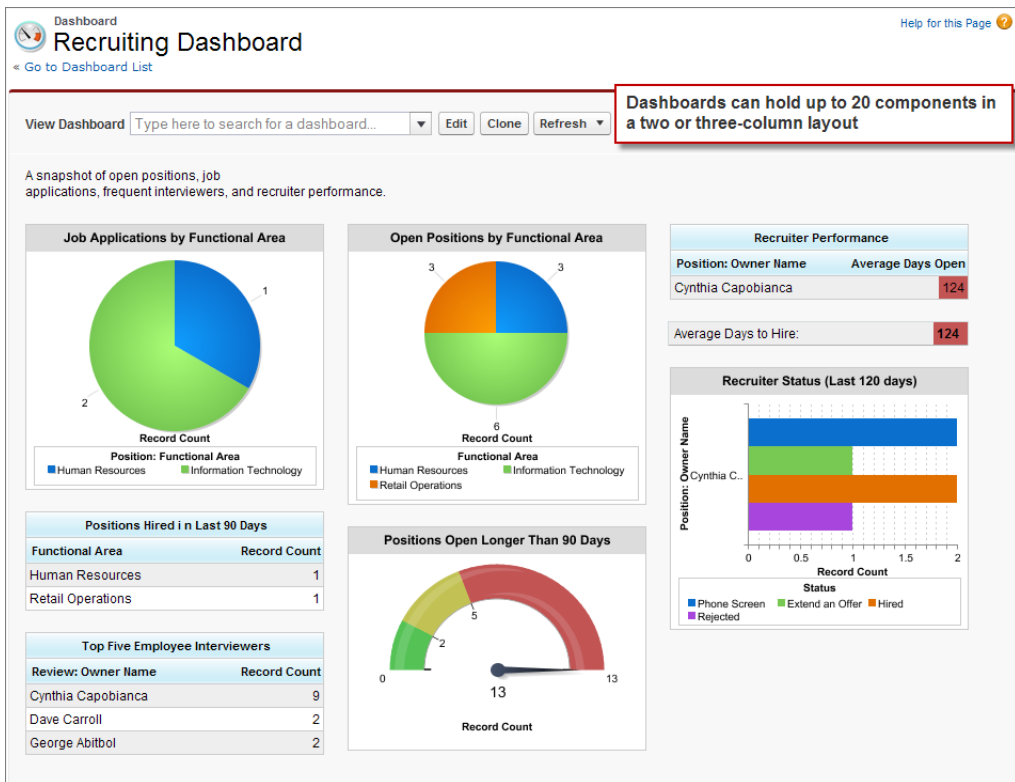


Figure 97: A Sample Recruiting Dashboard

Try It Out: Creating Additional Reports

We'll be creating a small-scale recruiting dashboard in this chapter, but before we do, let's use our new report-building skills to create a few more reports. Because we're already familiar with

how report builder works, we'll just outline the specifics of a few reports in the following table—you can either create them on your own, or just familiarize yourself with the options.



Tip: Want to click along with our dashboard instructions but don't want to spend your time creating all these new reports? Just create the Positions Open Longer Than 90 Days report in the first row. However, if you're interested in recreating the sample dashboard displayed here, you'll need to create the other four as well.

Table 40: Additional Recruiting Report Specifications

Report Name	Question	Report Type	Options
Positions Open Longer Than 90 Days	<i>Which positions have been open for more than 90 days?</i>	Positions	Format: Summary Report Summarize information by: Location and then by Functional Area Columns: Position: Position Title, Position: Owner Name, Status, Open Date, Hire By, Days Open Filters: View All Positions; Days Open greater or equal 90 Chart Type: Pie Chart Wedges: Functional Area Chart Title: Positions Open Longer Than 90 Days
Job Applications by Functional Area	<i>Which positions are getting the most candidates?</i>	Job Applications with Position	Format: Summary Report Summarize information by: Position: Functional Area and then by Position: Position Title Columns: Job Application Number, Job Application Status

Report Name	Question	Report Type	Options
			<p>Filters: View All Job Applications; Position: Status equals Open - Approved</p> <p>Chart Type: Vertical Column</p> <p>X-Axis: Position: Functional Area</p> <p>Chart Title: Job Applications by Functional Area</p>
Employee Interviewers	<i>Which employees conduct the most interviews?</i>	Job Applications with Reviews	<p>Format: Summary Report</p> <p>Summarize information by: Review: Created By</p> <p>Columns: Job Application: Job Application Number, Job Application: Position, Job Application: Status, Review: Created By, Review: Created Date</p> <p>Filters: View All Job Applications</p> <p>Chart Type: Pie</p> <p>Chart Title: Employee Interviewers</p>
Recruiter Status	<i>What does the job application pipeline look like for each recruiter and open position?</i>	Job Applications with Position	<p>Format: Matrix Report</p> <p>Subtotal Rows by: Position: Owner Name and then by Position: Position Title</p> <p>Subtotal Columns by: Job Application Status</p>

Report Name	Question	Report Type	Options
			Columns: Job Application Number, Position: Position Title Filters: View All Job Applications; Job Application: Last Modified Date Last 120 Days; Position: Status equals Open - Approved, Closed - Filled
Positions Hired in Last 90 Days	<i>Who have we hired in the last 90 days?</i>	Positions	Format: Summary Report Summarize information by: Functional Area Columns: Position: Position Title, Position: Owner Name, Hiring Manager, Functional Area, Job Level, Location, Close Date Filters: View All Positions; Close Date Last 90 Days; Status equals Closed - Filled

Try It Out: Creating a Dashboard

Now that we've got a set of reports to reference, we're ready to create a small Recruiting dashboard. To do so, we'll be working in the Dashboards tab. You can either access it by clicking the arrow tab and selecting it from the list of all available tabs, or by adding it to the list of visible tabs in your Recruiting app, as outlined in [Try It Out: Adding the Reports Tab](#) on page 249.

1. Click the Dashboards tab and click **Go to Dashboard List** near the top of the screen.



Note: Unlike other tabs, opening the Dashboards tab always displays the last dashboard that you viewed. If you've never visited the tab before, it displays a sample dashboard that comes by default with every organization.

2. Click **New Dashboard**.

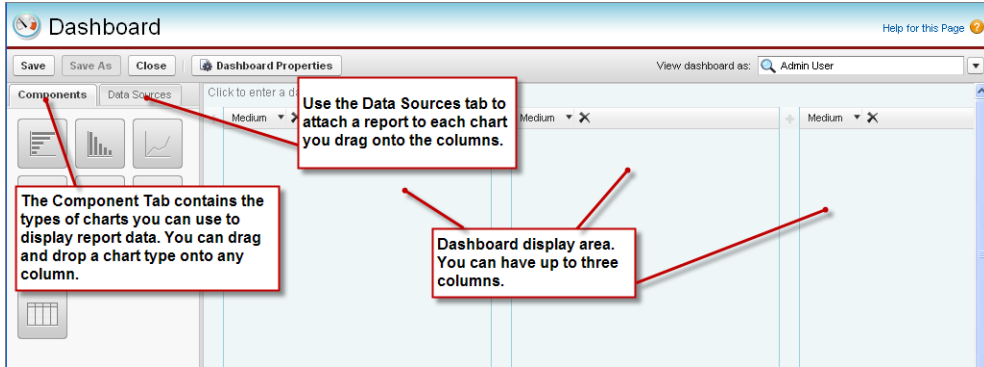


Figure 98: The Dashboard Edit Page

Once we specify the dashboard properties, we can add components.

3. Click **Dashboard Properties**.

4. In the `Title` field, enter `Recruiting Dashboard`.
5. In the `Dashboard Unique Name` field keep the default value of `Recruiting_Dashboard`.
6. Select the `Company Dashboards` folder to save it in.
7. Click **OK**.
8. For the dashboard description—the editable field near the top of the page—enter `A snapshot of open positions, job applications, frequent interviewers, and recruiter performance`.

Each dashboard has a running user, whose security settings determine which data to display in a dashboard. All users with access to the folder see the same data, regardless of their own personal security settings.

The running user's security settings only apply to the dashboard view. Once a user drills down into a source report or detail page off the dashboard, the user will view the data based on his or her normal security settings.

For example, suppose a system administrator with the "Modify All Data" permission is the running user for our recruiting dashboard. In this case, every recruiting-related record is counted in all of the report totals on our dashboard, including users who'd normally be restricted from viewing certain records (like those assigned to the Standard Employee profile). Although those users would be able to see the summary data for all records in the dashboard, if they navigated to the source reports, they'd see just the records they have access to.

When you're designing a dashboard, keep the dashboard's audience in mind. Ask whether any of the information is sensitive and how much you want them to see. If you do give a user access to dashboards that include more data than he or she normally has permission to view, be sure to communicate that they might see different values when they click through the dashboard to view the associated reports. And if you need to restrict a dashboard from certain users, just save it to a restricted-access folder.

Beyond the Basics

Did you know you can set up a *dynamic dashboard* that shows users data according to their own security settings?

Say you want to show the same set of dashboard components to different sets of users, each with a different level of visibility. You'd potentially have to set up dozens of dashboards with the right running user for each, and store them in separate folders. With dynamic dashboards, administrators can accomplish the same thing without having to create and maintain all those extra dashboards and folders. A single dynamic dashboard can display a standard set of metrics across all levels of your organization.

To find out more, see “Dynamic Dashboards Overview” in the Salesforce online help.

For our recruiting dashboard, the data that we'll be showing in the dashboard isn't particularly sensitive. Consequently, we'll choose a system administrator as the running user, and save the dashboard to a public folder.

9. In the `View dashboard as` field in the upper right of the screen, enter the name of a user with system administrator privileges. This sets the running user for the dashboard.

We now have an empty dashboard that's ready to be filled with components.

Adding Dashboard Components

We haven't yet defined any components, but it's easy to add and reorder them in dashboard builder.

Components come in five varieties:

- *Charts*—Displays a bar, column, line, pie, donut, or funnel chart—or any chart contained in a report.
- *Tables*—Displays a table that contains values and totals from columns in the report.
- *Metrics*—Displays the grand total from a report, along with a label that you enter.

- *Gauges*—Shows the grand total of a report as a point on a “fuel-tank” type of scale.
- *Visualforce Pages*—Displays any Visualforce dashboard component in your organization.

Try It Out: Adding a Chart Component

Let's start by adding a chart that shows the number of open positions by functional area:

1. Drag a pie chart component onto the left column of your dashboard.
2. Click “Edit Title” and enter `Open Positions by Functional Area`.
3. Click the Data Sources tab and find and select the `Open Positions by Functional Area` report.
4. Drag the report and drop it onto the pie chart component.

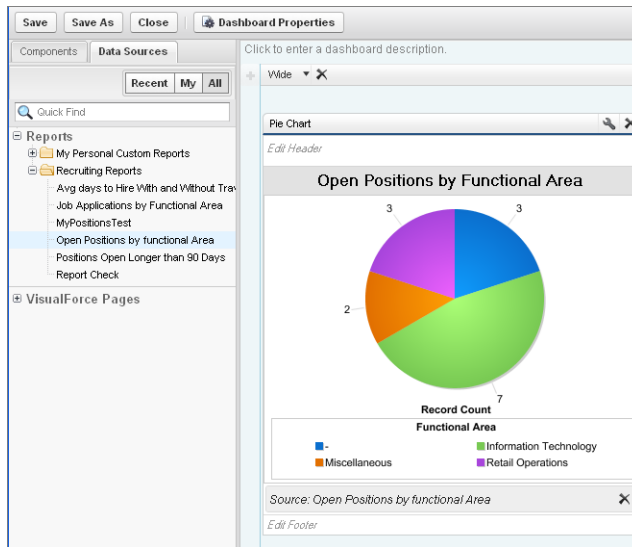


Figure 99: The Dashboard Component Edit Page

Ta-da! We now have a pie chart in our dashboard! If a user clicks on the chart, they're taken to the report from which the chart was generated.


Try It Out: Adding a Gauge Component

Now let's create a gauge component that shows us how we're doing with positions that have been open for longer than 90 days:

1. Drag a gauge component onto the middle column of your dashboard.

2. Click “Edit Title” and enter `Positions Open Longer Than 90 Days`.
3. Click the Data Sources tab and find and select the `Positions Open Longer Than 90 Days` report.
4. Drag the report and drop it onto the gauge component.

Now, let's change how our gauge looks. Just like the conditional highlighting in our matrix report, this gauge can display different colors depending on the total count of positions that have been open too long. We want a green color if there are fewer than two positions, yellow if there are between two and five, and red if the value is over five.

5. Click  on the gauge to open the component editor.
6. Click the Formatting tab and set the following fields:
 - For `Minimum`, enter 0.
 - For `Low Range Color`, pick a shade of green.
 - For `Breakpoint 1`, enter 2.
 - For `Breakpoint 2`, enter 5.
 - For `High Range Color`, pick a shade of red.
 - For `Maximum`, enter 10.
7. Click **OK**.


Perfect! Our recruiting dashboard now contains two components.

Try It Out: Adding a Table Component

Now let's create a table component that shows us the average number of days it takes each recruiter to fill a position:

1. Drag a table component onto the right column of your dashboard.
2. Click “Edit Title” and enter `Recruiter Performance`.
3. Click the Data Sources tab and find and select the `Avg Days to Hire With and Without Travel` report.
4. Drag the report and drop it onto the gauge component.

Just like its underlying source report, we can add conditional highlighting to our table component to highlight recruiter performance levels. We want a green color if the recruiter needs fewer than 45 days, yellow if he or she needs between 45 and 75 days, and red if the value is over 75.

5. Click  on the table to open the component editor.
6. Click the Formatting tab and set the following fields:

- For `Low Range Color`, pick a shade of green.
 - For `Breakpoint 1`, enter 45.
 - For `Breakpoint 2`, enter 75.
 - For `High Range Color`, pick a shade of red.
7. Click **OK**.

Try It Out: Adding a Metric Component

Finally, let's create a metric component that shows us the average number of days it takes all recruiters to fill a position:

1. Drag a metric component onto the right column of your dashboard and drop it below the table we just created.
2. Click the `Data Sources` tab and find and select the `Avg Days to Hire With and Without Travel` report.
3. Drag the report and drop it onto the metric.


Because metric components consist of a single value, they don't need a title. Instead, we give them a label much like any other field that you see in the platform:

4. In the label field, enter `Average Days to Hire`.



Note: Did you notice that we're using the same report that we used for our table? This metric is just another visualization of the data in that report.

Finally, let's add more conditional highlighting to showcase values:

5. Click  on the metric to open the component editor.
6. Click the `Formatting` tab and set the following fields:
 - For `Low Range Color`, pick a shade of green.
 - For `Breakpoint 1`, enter 45.
 - For `Breakpoint 2`, enter 75.
 - For `High Range Color`, pick a shade of red.
7. Click **OK**.

We've now got a simple, four-component dashboard for our Recruiting app! (Your dashboard might look slightly different.) Let's save our dashboard:

8. Click **Close**, then click **Save & Close**. You can also click **Save** if you want to update the title or description, or change the folder.

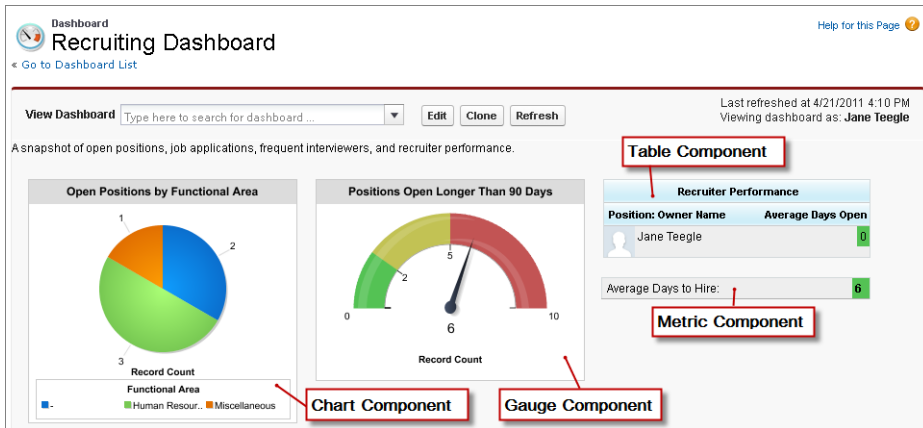


Figure 100: Four Components on the Recruiting Dashboard

If you click the Home tab, you can add the dashboard's first row to your home page. Be sure to add the dashboard section to your home page layout and then follow these steps:

1. On the Home tab, click **Customize Page** in the upper right corner of the Dashboard component.
2. Select Recruiting Dashboard from the Dashboard Snapshot drop-down list.
3. Click **Save**.

And because we saved the dashboard in a public folder, any user can add it to his or her Home tab, too.

As you can see, once we created the reports for our Recruiting app, adding them to a dashboard was a piece of cake. It's so easy, that we'll leave the remainder of the components as an exercise for you to try on your own—see how close you can come to recreating [A Sample Recruiting Dashboard](#) on page 264.

Refreshing Dashboards

Dashboards reflect a snapshot of your data at a specific time. On the Home tab, that time is indicated in the upper left corner on the Dashboard component; on the Dashboard tab, it's indicated in the upper right corner. You can refresh your dashboards on either tab by clicking **Refresh**.

If you are using Enterprise or Unlimited Edition, you can schedule dashboards to refresh automatically at specific times, and upon completion, receive an email notification that includes the refreshed dashboard.

Scheduling a dashboard refresh is easy. On the dashboards tab, simply click the down arrow on the **Refresh** button and select **Schedule Refresh...**, then, indicate who should receive the notification email and the time you want the refresh to occur.

Introducing Custom Report Types

While dashboards are an excellent way of rendering data, many users will need information that's more granular and unique to their job function, so they'll need to create their own reports. You'll want to control which data they can access for their reports, and make it easy for them to create reports that are useful. You can do both of these by creating custom report types.

Custom report types define the criteria from which your users can run and create reports. When you create a report type, you specify the objects, relationships, and fields that users can select for their reports.

How are custom report types useful in our Recruiting app? Well, our recruiters will appreciate it if we give them an easy way to scan for positions to which candidates have applied. In addition, the recruiters will probably want to see which of those job applications have reviews. That way, they'll know if any positions are on the verge of closing.

Creating Report Types

We've spent most of this chapter on the Dashboards and Reports tabs. Now it's time to return to the **Your Name > Setup** menu.

1. Click **Your Name > Setup > Create > Report Types**.
2. If you see an introductory splash page, just click **Continue**.
3. Click **New Custom Report Type**.

The custom report type that we're creating will include data from three different objects: Position, Job Application, and Review. Positions will be the focus of the recruiters, though, so let's make Position the primary object in this report type.

4. In the **Primary Object** drop-down list, select **Positions**.

Next, we'll give this report type an intuitive name and categorize it so it's easy for users to find it. We'll also enter a description so users who select the report type will know its function.

5. In the **Report Type Label** field, enter **Positions with Reviewed Job Applications**.

6. In the **Report Type Name** field, enter `Positions_with_Reviewed_Job Applications` if it is not there already.
7. In the **Description** field, enter `Which positions have job applications that have been reviewed?`
8. In the **Store in Category** drop-down list, select `Other Reports`.

When creating a report type, set its status to `In Development` if you want to test it before making it available to all your users; however, the report type we're creating does not require much testing, so let's deploy it:

9. Select `Deployed`.
10. Click **Next**.

The platform uses a graphical hierarchy and set diagrams (also called Venn diagrams) to let us easily specify which related records from other objects we want to include in the report results.

11. Click the white box under box A (Positions). Box B appears.
12. On box B, select `Job Applications` from the drop-down list, and leave `Each "A" record must have at least one related "B" record.` record must have at least one related "B" record selected.

Figure 101: Report Type Hierarchy and Set Diagram for Positions and Job Applications

By leaving this option selected, we've indicated that we want this report type to only include position records for which there are job applications. Notice how the set diagrams on the right change to reflect our selection.

13. Click the white box under box B (Job Applications). Box C appears.

14. On box C, select Reviews from the drop-down list, and leave Each "B" record must have at least one related "C" record selected.

The screenshot shows the 'Step 2. Define Report Records Set' configuration page. It includes a set diagram with three overlapping circles labeled A, B, and C. The intersection of circles A and B is shaded. Below the diagram are three columns of data representing the objects. The 'A to B Relationship' is defined as 'Each "A" record must have at least one related "B" record.' The 'B to C Relationship' is defined as 'Each "B" record must have at least one related "C" record.'

Figure 102: Report Type Hierarchy and Set Diagram for Positions, Job Applications, and Reviews

We've further narrowed the scope of our report to only include job applications that have reviews.

15. Click **Save**.

Our custom report type is nearly finished. In fact, we could technically say it's done now, but there's something we can do to make it even more convenient for our users: we can reorganize how the fields display in the Fields pane of the report builder.

This report type incorporates three objects (Position, Job Application, and Review), each containing many fields. A user creating a report using this custom report type will likely be overwhelmed by all of those fields, so let's remove the ones they won't need, and move the important ones to the top. We can also specify which ones are selected by default.

16. On the Custom Report Type detail page, scroll down to the Fields Available for Reports related list and click **Edit Layout**.

The page that appears is similar to the page layout editor we used in [Enhancing the Simple App with Advanced Fields, Data Validation, and Page Layouts](#) on page 53. You can reorder fields by dragging them, create and delete sections, and so forth. You can also double-click the fields to change their label and specify whether they should be checked by default.



Important: If a field isn't in a section, it won't be available to users when they generate reports from this report type.

Let's reorganize the page as follows:

17. Create a new section called `Position`, `Job Application`, and `Review Fields`.
18. In the `Positions` section, change the label of the following fields by double-clicking the field name and editing it in the dialog that opens:
 - `Created By` to `Position Created By`
 - `Created Date` to `Position Created Date`
 - `Status` to `Position Status`
19. In the `Job Applications` section, change the label of the following fields by double-clicking the field name and editing it in the dialog that opens:
 - `Created By` to `Job Application Created By`
 - `Created Date` to `Job Application Created Date`
 - `Status` to `Job Application Status`
20. In the `Reviews` section, change the label of the following fields by double-clicking the field name and editing it in the dialog that opens:
 - `Created By` to `Review Created By`
 - `Created Date` to `Review Created Date`
21. Move the following fields from the `Positions` section into the `Position`, `Job Application`, and `Review Fields` section:
 - `Position Title`
 - `Created By`
 - `Created Date`
 - `Days Open`
 - `Functional Area`
 - `Hire By`
 - `Hiring Manager`
 - `Open Date`
 - `Status`
 - `Travel Required`

22. Move the following fields from the Job Applications section into the Position, Job Application, and Review Fields section:

- Job Application Number
- Average Rating
- Created By
- Created Date
- Number of Reviews
- Status

Notice that we didn't include the `Candidate` field. We left it out because this report type is available to all users, including hiring managers. As mentioned in the previous chapter, a hiring manager might try to poach candidates that apply for other jobs, so it's best not to reveal candidate names in reports.

23. Move the following fields from the Reviews section into the Position, Job Application, and Review Fields section:

- Review Number
- Created By
- Created Date
- Rating

24. Make the following fields checked by default:

- Days Open
- Hiring Manager
- Job Application Number
- Open Date
- Position Title
- Review Number

25. Delete the individual Positions, Job Applications, and Reviews sections.

26. Click **Save**.

Your custom report type is ready! To try it out, go to the Reports tab, click **Create New Custom Report**, and select the Other Reports report type category. The Positions with Reviewed Job Applications report type is in the list below.

Look At What We've Done

Check out our Recruiting app now! We've met almost all of the requirements we talked about at the beginning of the book, but we're not there yet. In the next chapter, we're going to move beyond the platform's point-and-click tools, like security and workflow, and introduce Visualforce, the Web services API, and public sites. These features provide the key to incorporating functionality from all over the Web and will help us create a truly powerful application in the cloud. In fact, once you've mastered the tools available in the platform, these tools will be the way that you can let your creativity soar—the functionality of any app you build on the platform will be limited only by the Web itself!

Chapter 12

Moving Beyond Point-and-Click App Development

In this chapter ...

- [Introducing Mash-Ups and Web Services](#)
- [Introducing Visualforce](#)
- [Implementing the Candidate Map](#)
- [Implementing the Mass Update Status Button](#)
- [Taking Your App Public with Sites](#)

Up to this point, we've built a compelling app using various parts of the platform. We've created a data model to store our recruiting information, put workflow and approval logic in place to help manage the data, and built reports and a dashboard to help share the data. The fact that we were able to quickly put all of these pieces together without writing any code is a testament to the power of the Force.com platform.

A tremendous amount of research and thought has gone into the design of the platform. Salesforce.com has strived to anticipate the various business application needs of 21st century companies, and has addressed many of those needs with simple yet powerful declarative, point-and-click tools that nontechnical users can use to achieve unique business goals. However, it is impossible for one company to provide a single solution that's a perfect fit for everyone. That's why salesforce.com has made it easy for developers to write code that builds upon the Force.com platform's point-and-click functionality. Once you learn how to program for the Force.com platform, you'll find that you can create apps for the Cloud that do just about anything you can dream up. The sky's the limit!

“Programming-phobia” is common for people without computer science backgrounds, and it's likely that many readers will want to put this book down as soon as they see the words “code” and “variable.” Relax! When you read this chapter, you'll discover just how fun and easy programming for the Force.com platform can be. Also, bear in mind that the intent of this chapter isn't to make you a programmer

(although it's a great place to start). Instead, this chapter will just give you a taste of what it's like to enhance your app with some very simple programming. If you don't feel like typing any code yourself, just copy and paste the code samples from the files included in the `RecruitingApp-5_0.zip` file, which you downloaded while reading the [Expanding the Simple App Using Relationships](#) chapter.

With the sample code and a few mouse clicks, we'll be able to swiftly enhance our app by adding the following features:

- **Candidate Map**—An interactive map that shows the locations of the candidates that have applied for a particular position
- **Mass Update Status**—The ability to update the `Status` field on multiple job applications at the same time
- **Public Careers Site**—A public Web site that anyone on the Internet can access to see what positions are open at Universal Containers

Let's begin!

Introducing Mash-Ups and Web Services

If we revisit the requirements of our Recruiting app, we'll see that we still need to implement an interactive map that shows the locations of the candidates that have applied for a particular position. Clearly no such functionality is available by default in the platform, and we certainly don't have time to build our own mapping engine. Is this type of functionality even possible?

Of course it's possible! Because the Force.com platform runs on the Web, we can leverage the power of other websites to implement features that would never be available just through our platform alone. This means that with a little code, we can *mash up* our own recruiting data with an interactive map website, such as Yahoo! maps, and place this functionality in our own app.

To implement a mash-up, we'll first need to understand a little about the technology that makes it possible: Web services. A *Web service* is the mechanism by which two applications that run on different platforms, that were written in different languages, and that are geographically remote from each other, can exchange data using the Internet. Web services makes data exchange between two such applications as straightforward as two processes exchanging data on a single computer.

The way that data is exchanged between two Web services is similar to the way data is exchanged between a Web browser like Microsoft Internet Explorer and a Web server. Just as a Web browser uses a common network protocol (HTTP over TCP/IP) to download HTML files hosted on a Web server, a Web service can also use this same network protocol to download data from another Web service. The key difference is the actual data that is sent and received—Web services use XML instead of HTML.

The online world has a vast array of Web services, many of which are free. For our Candidate Map feature, we'll utilize the free Yahoo! Maps Web Services, which let you easily embed interactive maps in your apps. If we can find a way to pass candidate addresses from our app to the Yahoo! Maps Web Services, Yahoo! will take care of all the mapping functionality, saving us from worrying about how our app will render an interactive map. We'll just need to figure out how to pull that rendered map into our app.



Note: The Force.com platform also has its own powerful Web services: the Web services API. With the Force.com Web Service API, you can customize and integrate your Salesforce organization using the language and platform of your choice. The API defines a Web service that enables full, reliable access to all of the data in your organization, including the ability to read, create, update, and delete records.

Because the Web services API is only used behind the scenes for our Candidate Map feature, this book does not go into detail about it; however, you can find Web services API documentation and code samples on Developer Force at developer.force.com.

Introducing Visualforce

So how do we pass our candidate's addresses to the Yahoo! Maps Web Services? And after Yahoo!Maps Web Services renders an interactive map that plots those addresses, how do we pull that map into our app?

These requirements may seem intimidating, but they're actually quite easy to meet, thanks to Visualforce. Visualforce is a powerful and flexible framework for customizing your app's user interface far beyond what's available using the platform's point-and-click tools. It's the most efficient way to combine data from multiple Force.com objects, blend data from Web services into your apps, or customize the logic that dictates the behavior of your app's user interface. When you use Visualforce, you'll see your productivity increase, and you'll find that you can create just about any type of browser-based user interface you can imagine.

The Visualforce framework consists of a tag-based markup language, similar to HTML. In the Visualforce markup language, each Visualforce tag corresponds to a user interface component. Need a related list? Simply add the `<apex:relatedList>` component tag. Want to display a record's details? Just use the `<apex:detail>` tag.

The following graphic shows a few of the most commonly used Visualforce tags and how they correspond to user interface components. Over seventy tags exist, ranging from large components, such as a detail section of a standard page, to small components, like a single field or link. You can learn about them all in the *Visualforce Developer's Guide*, at www.salesforce.com/us/developer/docs/pages/index.htm.

The screenshot shows a Salesforce interface with several components highlighted by red boxes and arrows pointing to their corresponding Visualforce tags:

- <apex:page>**: Points to the overall page structure.
- <apex:pageBlock>**: Points to the main content area.
- <apex:commandLink>**: Points to the search bar and 'Create New...' button.
- <apex:dataTable>**: Points to the table listing favorite contacts.
- <apex:image>**: Points to the profile picture of Doug Chasman.
- <apex:relatedList>**: Points to the 'Recent Items' list on the left sidebar.
- <apex:detail>**: Points to the detailed contact information for Doug Chasman.

The table of favorite contacts is as follows:

Name	Account Name
Doug Chasman	Grand Hotels & Resorts Ltd
Greg Salomon	University of Arizona
Jill Webster	Burlington Textiles Corp
Andrew White	Burlington Textiles Corp
Craig Weissman	Pyramid Construction Inc.

The contact detail for Doug Chasman includes:

- Contact Owner: Caroline Roth (Change)
- Name: Doug Chasman
- Account: Grand Hotels & Resorts Ltd
- Title: President and CEO
- Department: [Blank]
- Birthdate: [Blank]
- Reports To: [Blank]
- Lead Source: [Blank]
- Mailing Address: 2334 N. Michigan Avenue, Suite 1500, Chicago, IL 60601, USA, Chicago, IL
- Languages: [Blank]
- Created By: Caroline Roth 8/9/2007 2:28 PM
- Description: [Blank]

Figure 103: Sample Visualforce Components and Their Corresponding Tags

The behavior of Visualforce components can either be controlled by the same logic that is used in standard Salesforce pages, or you can associate your own logic written in Apex. Apex is salesforce.com's programming language that runs in the cloud on Force.com servers.

Don't panic! You won't need to learn how to write Apex to create any of the features described in this book. Visualforce comes with a rich component library that allows you to quickly build pages without having to code a lot of functionality yourself. And because Visualforce markup is ultimately rendered into HTML, you can use Visualforce tags alongside standard HTML, JavaScript, Flash, AJAX, or any other code that executes within an HTML page. This means that we can create our Candidate Map by simply creating a Visualforce page that uses some basic Visualforce markup and JavaScript to pass our candidate addresses to Yahoo! Maps Web services.



Note: To learn more about Apex, see the *Force.com Apex Code Developer's Guide*, at

www.salesforce.com/us/developer/docs/apexcode/index.htm.

Introducing Visualforce Development Mode

Creating Visualforce pages is quick and easy, and there are two ways to do it: through the setup area by clicking **Your Name** ► **Setup** ► **Develop** ► **Pages**, or by enabling Visualforce development mode and navigating to a “blank” page.

In this book, we'll use Visualforce development mode to create and edit our Visualforce pages because it has several features that are quite handy. One of those is the special development footer on every Visualforce page. The footer lets you access a page markup editor that includes tools you can use to search for values, jump to a certain line in your code, and undo and redo changes. The page editor also offers highlighting and auto-suggest for component tags and attributes, and has a link to the component reference documentation, which includes descriptions and examples for every Visualforce component.

While you're in Visualforce development mode, you can create a new page just by entering a unique URL in your browser's address bar. And as you add code to your Visualforce page, you'll be able to save it and see your changes instantly rendered in your browser!

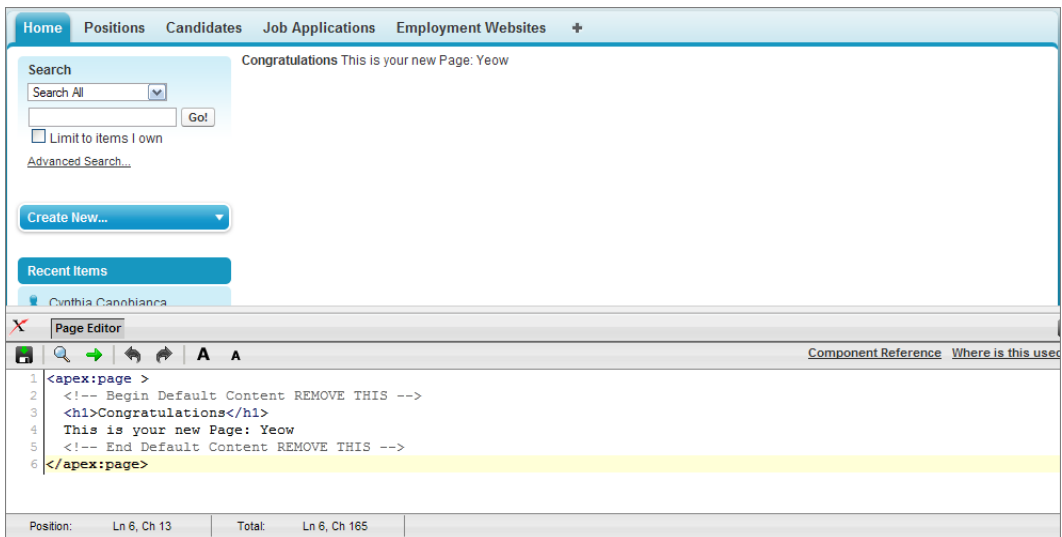


Figure 104: Visualforce Development Mode

Try It Out: Enabling Visualforce Development Mode

Enable Visualforce development mode as follows:

1. Click **Your Name** ► **Setup** ► **My Personal Information** ► **Personal Information**, and click **Edit**.
2. Select the `Development Mode` checkbox.
3. Click **Save**.

Implementing the Candidate Map

Now that we've been introduced to Web services and Visualforce, we're ready to implement the mash-up piece for our Recruiting app: the Candidate Map. Our requirements state that we need to generate a map for all the candidates that have applied for a particular position so that we can better understand potential relocation costs associated with a new hire. Since Universal Containers has offices all over the country, this map will also help us assign candidates to an office if a particular position is open in more than one location.

Try It Out: Creating a Visualforce Page

Now that we're in development mode, let's create a Visualforce page called `CandidateMap` by simply modifying the URL in our browser.

Salesforce URLs typically look something like this: `https://na1.salesforce.com/001/o`. You can create a new Visualforce page by removing everything to the right of the `salesforce.com/` part of the URL and replacing it with `apex/` followed by the name of the page you want to create.

1. In the address bar of your browser, replace everything to the right of `salesforce.com/` with `apex/CandidateMap`.



Caution: Don't change anything to the left of `salesforce.com/`, as this information is specific to the instance of the Force.com platform that you're using, and changing it will prevent you from creating the Visualforce page.

The resulting URL should look something like this:

`https://na1.salesforce.com/apex/CandidateMap`. Note that your URL may have an instance name other than `na1` preceding the `salesforce.com/` part. That's okay.

2. Press **Enter**.

The following Visualforce error page appears indicating that the page doesn't exist yet. Again, that's okay—this gives us a chance to exercise one of the other handy features of development mode, the *quick fix*. A quick fix is a way of creating something on the fly, right when we need

it. In this case, even though the CandidateMap page doesn't exist yet, development mode gives us a quick fix link to create it on the fly. Clicking the link is the equivalent of going to the setup area, navigating to the Visualforce page section, clicking **New**, entering the name of the page, and clicking **Save**.

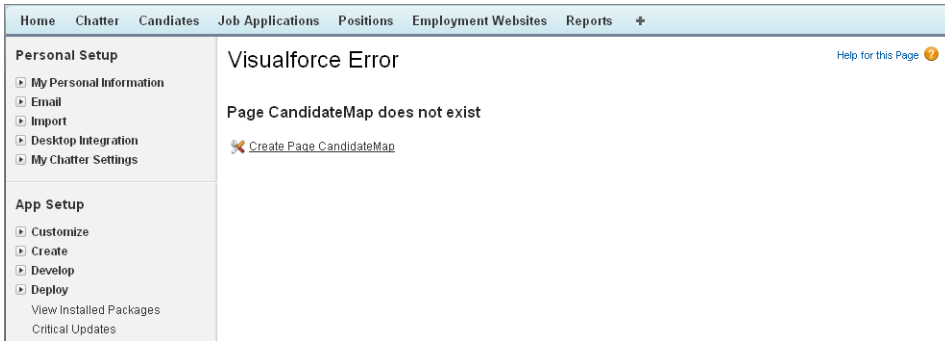


Figure 105: Visualforce Error Page

3. Click the **Create Page CandidateMap** link.

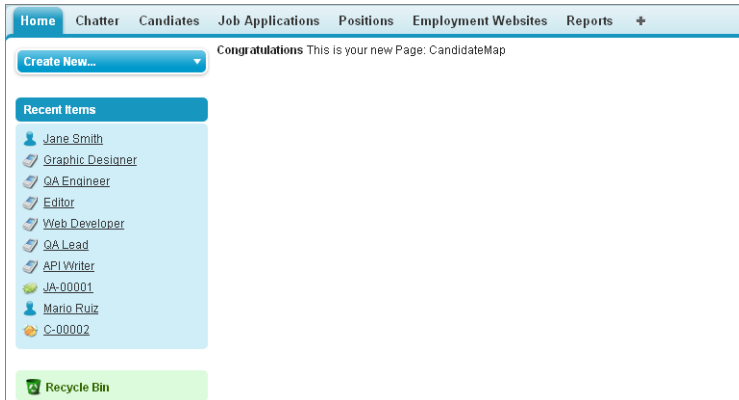


Figure 106: CandidateMap Visualforce Page

Congratulations! You've created your first Visualforce page! Now it's time to add some Visualforce markup so it displays our candidate map.

Try It Out: Writing Visualforce Markup

In the footer at the bottom of your new Visualforce page, click **Page Editor** to display the Visualforce development mode page editor. This editor displays all the markup for the page you're currently viewing.

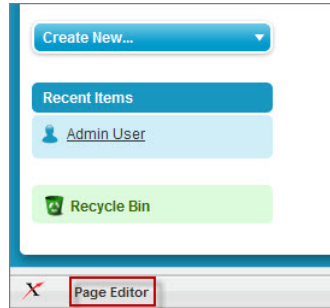


Figure 107: Visualforce Development Mode

You'll notice that the page editor has the following default content:

```
<apex:page >
  <!-- Begin Default Content REMOVE THIS -->
  <h1>Congratulations</h1>
  This is your new Page: CandidateMap
  <!-- End Default Content REMOVE THIS -->
</apex:page>
```

The default content contains the Visualforce component tag `<apex:page>` on the first line and the closing tag, `</apex:page>`, on the last line. Just like elements in other markup languages, Visualforce component tags have a start tag, such as `<apex:page>`, and an end tag that is identical to the start tag except that it has a forward slash, such as `</apex:page>`.

The `<apex:page>` tag represents a single Visualforce page. All of the other content you want displayed on a page must be wrapped inside the start and end `<apex:page>` tags. As discussed earlier, content can be other Visualforce tags, plain text, merge fields, HTML, JavaScript, and so forth. For example, in the default content there are comments marked by the `<!--` and `-->` symbols, the HTML `<h1>` header tag, and plain text.

Let's change the look and feel of the page so it matches the style of the Position object, the object that's most closely tied to our candidate map. We can do this just by setting an attribute on the `<apex:page>` tag. As with HTML tags, attributes on Visualforce tags configure the style or behavior of what the tag represents.

The attribute we need to set is `standardcontroller`. In Visualforce, controllers determine the style of the page, how the page acts when a user clicks a button, and the data that should be displayed in a page. Simply by setting value of the `standardcontroller` attribute to the Position object, we're configuring our page to look and behave like the position pages, and display position data.

1. Place your cursor just inside the closing bracket of the `<apex:page>` tag, press the spacebar, and type `standardcontroller="Position__c"`. The result should look like this:

```
<apex:page standardcontroller="Position__c">
```



Tip: You can develop your own controllers, but the Force.com platform also includes a standard controller for every object, including custom objects that you create. It's another one of the benefits of Force.com!

Note that the format for setting an attribute is the name of the attribute followed by the equals sign (=), then the value of the attribute enclosed in quotes. Also, remember that the unique API names of custom objects have two underscores (__) and the letter *c* at the end.

Let's save our work so we can see our changes applied.

2. Click the save icon (📁) on the page editor, or press CTRL+S.

When you save your markup, the Force.com platform checks to make sure it's valid and lets you know if there are errors. If the markup is valid, the new version of your Visualforce page is saved and rendered in your browser.

Can you see how your page is different from before? Setting the `standardcontroller` attribute changed the look and feel of the default Visualforce page so that the Positions tab is selected.

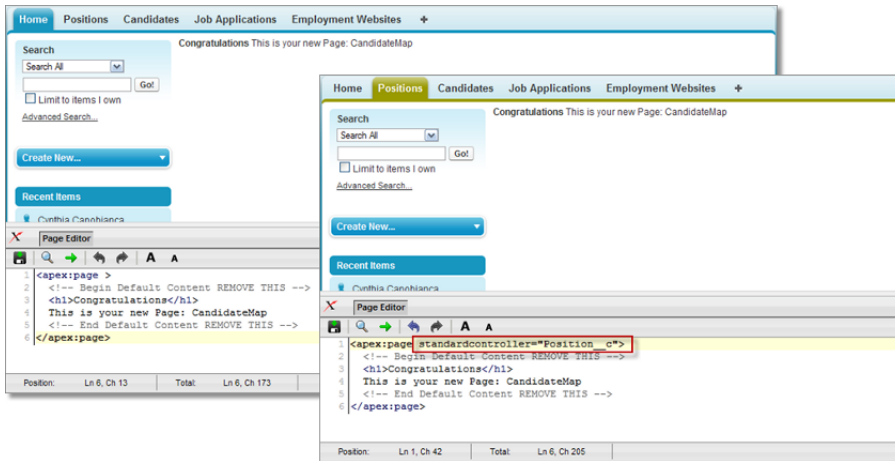


Figure 108: Before and After Setting the `standardcontroller` Attribute

Next, let's remove the default HTML on our Visualforce page and replace it with something relevant to our Candidate Map, such as a brief description of the page just in case the purpose of the map isn't immediately obvious to users.

3. Delete the following markup:

```
<!-- Begin Default Content REMOVE THIS -->
<h1>Congratulations</h1>
This is your new Page: CandidateMap
<!-- End Default Content REMOVE THIS -->
```

4. Enter the following markup between the start and end `<apex:page>` tags.

```
This map shows the locations of candidates who have applied
for the <b>{!Position__c.Name}</b> position.
```

The result should look like this:

```
<apex:page>
This map shows the locations of candidates who have applied for the
<b>{!Position__c.Name}</b> position.
</apex:page>
```

The description includes the `{!Position__c.Name}` merge field enclosed in the `` HTML tag. We've used merge fields quite a bit while building our app. They were in our email template and some of our formulas. Now, we're using them once again, this time side by side with HTML and Visualforce component tags.

5. Click the save icon () on the page editor.

Right now, our merge field isn't displaying because the Candidate Map is out of *context*, that is, there's no specific position record from which it can grab the data. Don't worry about that for now—we'll learn how to set context later. For now just trust that the `{!Position__c.Name}` merge field will render the position name, and the `` HTML tag will make it bold.

Our Visualforce page is now ready for us to add our interactive map!

Try It Out: Adding the Map to Our Visualforce Page


The interactive map functionality that is essential to our Candidate Map feature can be achieved with a blend of Visualforce markup and JavaScript that accesses the Yahoo! Maps Web Services. As promised, you don't have to learn JavaScript to get the Candidate Map working. Instead, we'll use the sample code in the `RecruitingApp-5_0.zip` file you downloaded from Developer Force in [Expanding the Simple App Using Relationships](#).

1. In the `RecruitingApp-5_0.zip` file, locate the file `CandidateMapSample` and open it in your favorite text editor.



Important: Make sure that Word Wrap or any feature that might add line breaks to the code is turned off in your text editor. You must preserve the original line breaks in the code sample. New line breaks caused by word wrapping features in text editors can break the code and prevent the code sample from working correctly.

Also, be aware that copying code from Adobe PDF files can also cause code to break, so avoid copying code from PDF versions of this book.

2. Select the entire content of the file, and copy and paste it into the page editor at the bottom of your CandidateMap Visualforce page. The `CandidateMapSample` file contains the Visualforce markup you added in the steps above, so just replace everything in the page editor with the entire sample in the file.
3. Click the save icon () on the page editor.

The interactive map should appear on your CandidateMap Visualforce page, but with an error message and without plots. This is because the Candidate Map is still out of context and the page can't figure out which candidates to show. We can fix this by simply adding the CandidateMap Visualforce page to our position page layouts.

Try It Out: Adding the Candidate Map to Position Page Layouts

Adding Visualforce pages to page layouts is as simple as adding fields or sections to page layouts. And once we add the CandidateMap Visualforce page to our position page layouts, the map will automatically plot the locations of the candidates who apply to each position. It'll work like magic!

We have two page layouts for the Position object and we'll want the map to appear on both. Let's start by adding the page to the original position page layout (Position Layout).

If you have any standard position records that use the Position Layout, navigate to one of those positions and click the **Edit Page Layout** link in the upper right corner of the record. If you don't have any standard position records:

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In the Page Layouts related list, click **Edit** next to Position Layout.

The Position Layout is ready to edit. Let's create a section on the page layout for the CandidateMap Visualforce page.

4. In the palette, select the Fields category.
5. Drag the Section user interface element from the palette to just below the Description section on the page layout. The Layout Properties popup window appears.
6. In the Section Name text box, enter Candidate Map.
7. In the Display Section On Header area, select both the Detail Page and Edit Page checkboxes.
8. In the Layout drop-down list, choose 1-Column.
9. Click **Ok**.

Now let's move the CandidateMap Visualforce page into our new section.

10. Select the Visualforce Pages category on the palette.

Notice that the palette lists our CandidateMap Visualforce page as a user interface element.

11. Drag the CandidateMap user interface element from the palette to the Candidate Map section on the page layout.

The Candidate Map JavaScript generates a map that is 400 pixels high and spans the width of the screen. This is just large enough to be useful without consuming too much room on the screen. The default properties of the Visualforce page user interface element need to be adjusted to accommodate the map. Otherwise, there might be odd spacing around the map and distracting scrollbars.

Let's fix the Visualforce page user interface element properties to accommodate the size of the generated map, as well as the text we added to the top of the CandidateMap Visualforce page. (In case you don't remember, the text is “This map shows the locations of candidates who have applied for the {!Position__c.Name} position.”)

12. Double-click the CandidateMap element to access its properties.
13. Set the width to 100%.
14. Set the height to 405.

The height is always in pixels, while the width can be specified either in pixels or as a percentage. Setting the height to 405 pixels allows 400 pixels for the map and five more pixels for the text. Perfect!

15. Leave the Show scrollbars and Show label checkboxes deselected.
16. Click **OK** to exit the user element properties.
17. Click **Save** on the page layout.

Repeat all of the above steps for your other Position object page layout, the IT Position Layout. Once you finish, it's time to test the Candidate Map!

Try It Out: Testing the Candidate Map

Navigate to any position record with one or more candidate applications. (If you haven't created any job applications in your app, now would be a good time.) After you bring up a position record, scroll down to see the Candidate Map in action!

Responsibilities	Design, create, update, and deliver documentation for UCI's applications
Skills Required	<ul style="list-style-type: none"> * At least 5+ years writing technical documentation for a software development company * Exceptional technical writing and editing skills * Proven ability to write documentation for both non-technical business users and administrators * Ability to consistently meet deadlines and deliver high-quality materials * Proven track record of innovation for making online documentation more usable and useful for customers * Strong experience using XML tools to write single-source documentation for both online help and PDF distributio * Ability to learn quickly and adjust priorities in a dynamic environment with short release cycles * Demonstrated ability to adhere to existing styles and processes, and contribute towards improving them as necessary * Strong team player and able to work with documentation management to establish goals and priorities * Bachelor's degree
Educational Requirements	- Bachelor's Degree

▼ Candidate Map

This map shows the locations of candidates who have applied for the Documentation Writer position.

Figure 109: Our Working Candidate Map

Notice how the Candidate Map Visualforce page is embedded within our Recruiting app just like the other functionality we built using the platform's declarative, point-and-click tools. Also notice that the map automatically displays only candidates who applied to this position.

With very little programming and a few clicks, we've mashed up Yahoo Maps Web Services with Force.com and made the result look as if it were made expressly for our Recruiting app. Couple the Force.com Web Services API with the thousands of Web services that are available on the Internet today, and the possibilities are limited only by our imaginations!

Implementing the Mass Update Status Button

The Candidate Map feature we just added is both useful and flashy. It'll help recruiters do their jobs more efficiently while adding an eye-catching graphic to an otherwise plain collection of data. The next feature we're going to implement is not as aesthetically appealing, but is much more useful and likely to save your users an incredible amount of time.

Picture yourself as a recruiter at Universal Containers. You're working on filling a highly-coveted position that's garnered hundreds of job applications and you've finally zeroed in on the three top candidates. However, before you can go further you need to reject the job applications submitted by all of the other candidates for the position—a task that could take hours if you have to open every one of those job application records and set the `Status` to rejected. What can you do?

Fortunately, creating tools that perform mass actions on data is yet another of the innumerable ways you can build upon the platform's functionality. With a few lines of Visualforce and the platform's point-and-click tools, we can enhance the Recruiting app in a way that will dramatically increase our users' productivity. When we're done, the daunting task of updating the `Status` field on multiple job applications will be reduced from hours of tedium to a few seconds of easy mouse clicking.

Planning the Mass Update Status Feature

The objective is simple: create a way to update the `Status` field on multiple job applications in a single operation. We just need to work out a few logistics to make sure the implementation goes smoothly. For example, where in our app will users go to access the Mass Update Status functionality? And how will they perform the actual update?

When designing a new feature, it's important to consider the context in which users will access it. Since the purpose of the Mass Update Status feature is to update the `Status` field on multiple job applications for a single position, it's most likely that users will want to perform this operation while viewing a position record. Therefore, it makes the most sense to provide access to the Mass Update Status feature from position records.

So where exactly on a position record should we provide access to the feature? Well, back in [Expanding the Simple App Using Relationships](#) on page 85, we added a Job Applications related list to our position records that made it easy for users to quickly identify all of the job applications submitted for a position. We could leverage that list by adding a checkbox next to each job application so that users can select the group of job applications they want to update.

Then we could add an **Update Status** custom list button that lets users update the `Status` fields on all of the selected job applications in one fell swoop.

Once we're done, the feature will work something like this:

1. Open a position record and scroll down to the Job Applications related list.
2. In the Job Applications related list, select the checkboxes next to the job applications you want to update.

Action	Job Application Number	Candidate	Status	Created Date	Owner First Name	Owner Last Name
<input type="checkbox"/> Edit Del	JA-00002	C-00008	Rejected	2/4/2010	Cynthia	Capobianca
<input type="checkbox"/> Edit Del	JA-00008	C-00003	New	6/10/2010	Cynthia	Capobianca
<input type="checkbox"/> Edit Del	JA-00009	C-00008	Schedule Interviews	6/10/2010	Cynthia	Capobianca

Figure 110: Job Applications Related List

3. Click the **Update Status** button.

The Mass Update Status page appears:

Mass Update the Status of Job Applications

Save Cancel

Change

Status: --None--

Selected Applications

Job Application Number	Position Title	Candidate Name	Status
JA-00008	Documentation Writer	Chris McGuire	New
JA-00009	Documentation Writer	George Schnell	Schedule Interviews
JA-00002	Documentation Writer	George Schnell	Rejected

Save Cancel

Figure 111: Mass Update Status Page

4. Choose a value for the `Status` field.
5. Click **Save**.

It may sound like this will take an astounding effort to implement, but it's really quite simple. All you need to do is create the Mass Update Status page with some basic Visualforce markup, and add a custom list button to the Job Applications related list on position records. You'll be done before you know it!

Try It Out: Creating the Mass Update Status Page

We'll begin implementing the Mass Update Status feature by creating the Visualforce page that lets users choose the value with which the `Status` field is updated. To make this page even more usable, we'll include a table that shows the `Job Application Number`, `Position Title`, `Candidate Name`, and `Status` of each selected job application.

1. In the address bar of your browser, replace everything to the right of `salesforce.com/` with `apex/MassUpdateStatus`.

The resulting URL should look something like this:

`https://na1.salesforce.com/apex/MassUpdateStatus`. Remember to only modify the part of the URL after `salesforce.com/`.

2. Press **Enter**.
3. Click the **Create Page MassUpdateStatus** link.

We now have a new Visualforce page called `MassUpdateStatus`. Next, we'll add the Visualforce markup that implements the Mass Update Status functionality.

To enter our markup, we'll use the Visualforce development mode page editor just as we did before. But this time around, we won't need any JavaScript; Visualforce will be able to do it all!

4. In the footer at the bottom of the `MassUpdateStatus` Visualforce page, click **Page Editor** to display the Visualforce development mode page editor.
5. Delete all of the default markup in the Visualforce development mode page editor and replace it with the following markup. Remember that instead of typing, it's easiest to copy and paste the code from the `MassUpdateStatusSample` file located in the `RecruitingApp-5_0.zip` file you downloaded from developer.force.com/books/fundamentals.

```
<apex:page standardController="Job_Application__c"
    recordSetVar="applications">
  <apex:sectionHeader title="Mass Update the Status
    of Job Applications"/>
  <apex:form>
    <apex:pageBlock>
      <apex:pageMessages />
      <apex:pageBlockButtons>
        <apex:commandButton value="Save"
          action="{!save}"/>
        <apex:commandButton value="Cancel"
          action="{!cancel}"/>
      </apex:pageBlockButtons>
      <apex:pageBlockSection title="Status Update"
        collapsible="false">
        <apex:inputField value=
```

```

        "{!Job_Application__c.Status__c}"/>
    </apex:pageBlockSection>
    <apex:pageBlockSection title="Selected Job
    Applications" columns="1">
        <apex:pageBlockTable value="{!selected}"
        var="application">
            <apex:column value="{!application.name}"/>
            <apex:column value=
            "{!application.position__r.name}"/>
            <apex:column headerValue="Candidate Name">
                <apex:outputText value=
                "{!application.candidate__r.
                First_Name__c & ' ' & application.
                candidate__r.Last_Name__c}"/>
            </apex:column>
            <apex:column value=
            "{!application.Status__c}"/>
        </apex:pageBlockTable>
    </apex:pageBlockSection>
</apex:pageBlock>
</apex:form>
</apex:page>

```



Tip: In this sample code, notice that some components don't have close tags. If there are no other components nested within a component, you can “close” the tag by putting a forward slash at the end of the start tag, like this: `<apex:sectionHeader/>`.

6. Click **Save**.

Understanding the MassUpdateStatus Visualforce Markup

Let's take a moment to discuss the main Visualforce tags we just added to our MassUpdateStatus page. Although we won't go into all of the use cases for each component or discuss every attribute in depth, you'll get a better understanding of how Visualforce works.



Tip: To see general descriptions and examples for all of the Visualforce components and their attributes, click the **Component Reference** link in the upper right corner of the Visualforce development mode page editor.

```
<apex:page>
```

As with all Visualforce pages, the MassUpdateStatus page must begin with an `<apex:page>` component. Notice the tag has the same `standardController` attribute used in our interactive Candidate Map feature, although this time it is set to the Job Application object (`Job_Application__c`). This makes sense because the Mass Update Status feature updates a field on job application records, not position records.

The component also has a `recordSetVar` attribute. We use this attribute to change the `standardController` so that it accommodates a set of records rather than a single record.

`<apex:sectionHeader>`

The `<apex:sectionHeader>` component adds a header to the top of the page. The component's `title` attribute determines the text in the header.

`<apex:form>`

The `<apex:form>` component establishes a section on the page in which users can enter data and submit it by clicking a button or link. It's like an invisible container, similar to a `<form>` element in HTML.

`<apex:pageBlock>`

The `<apex:pageBlock>` component designates an outlined area on the page similar to the areas on detail pages that contain sections.

`<apex:pageMessages>`

The `<apex:pageMessages>` component allocates space for standard system messages (such as those that notify users when a file is being saved) and validation rule errors. These messages already exist in the Force.com platform, so you don't have to create them—you just have to use this component to make room for them in case the platform needs to display them.

`<apex:pageBlockButtons>`

The `<apex:pageBlockButtons>` component allocates space for a set of buttons on the page. Its subcomponents specify what the buttons do and how they are labeled.

`<apex:commandButton>`

Each `<apex:commandButton>` component creates an individual button inside the `<apex:pageBlockButtons>` component. The Mass Update Status page uses two `<apex:commandButton>` components: one to create a **Save** button and a second to create a **Cancel** button. The buttons are styled like standard Salesforce buttons.

The `value` attribute on the `<apex:commandButton>` component determines the words that appear on the button (such as “Save” or “Cancel”), while the `action` attribute determines the operation that occurs when the button is clicked. When setting the `action` attribute, you must use merge field syntax. For example, to configure the button to save the data entered on the page, set the `action` attribute to `{!save}`.

Each button appears twice on our MassUpdateStatus page—once at the top of the area allocated by the `<apex:pageBlock>` component and once at the bottom. This is a precautionary

measure built into the `<apex:pageBlockButtons>` component to ensure that the button functionality is apparent to users, even if the page block is large.

`<apex:pageBlockSection>`

The `<apex:pageBlockSection>` component can be used within `<apex:pageBlock>` components to create a section on a page similar to the sections found on page layouts. On this page, the `<apex:pageBlockSection>` component is used twice.

The first instance of the `<apex:pageBlockSection>` component has a `title` attribute that's set to "Status Update." This text will appear at the top of the section. It also has a `collapsible` attribute that determines whether users can collapse and expand the section by clicking an arrow to the left of the title. We don't want users to accidentally collapse this page block section, so the attribute is set to "false."

The second `<apex:pageBlockSection>` component creates a section that has a table showing the job applications selected for updating. Its `title` attribute is set to "Selected Job Applications." Also, its `columns` attribute is set to "1."



Tip: Unlike page layouts, a section on a Visualforce page can have more than two columns. However, the platform's stylesheets are optimized to accommodate one or two columns, so it's best not to exceed that limit.

`<apex:inputField>`

The `<apex:inputField>` component renders the `Status` field from the Job Application object on our page. Use `<apex:inputField>` components to create HTML input elements for any Salesforce field. All you need to do is set the component's `value` attribute to the API name of the Salesforce object and field.

`<apex:pageBlockTable>`

The `<apex:pageBlockTable>` component renders a table containing field values from multiple records of a specific object. For our feature, we need to set two of this component's attributes: `value` and `var`.

The `value` attribute tells the table which set of records contains the values to display. In this instance, we set the attribute to the expression `{!selected}` to enable the table to display values from the selected job applications.

The second attribute, `var`, creates a name that components within the table can use to reference individual records in the record set without actually referring to each record by name.

<apex:column>

The <apex:column> components inside the <apex:pageBlockTable> determine the columns of the table and the job application fields each column displays. For three of the four <apex:column> components, we just need to set the value attribute to an expression that references the field using the value of the <apex:pageBlockTable> component's var attribute followed by the API name of the field. For example, the following expression displays the values of the Job Application Number field:

```
{!application.name}
```

For the Candidate Name field, though, we need to do a bit more because the field is actually a combination of the First Name and Last Name fields from the Candidate object. To combine these fields, we use an <apex:outputText> component inside an <apex:column> component and set its value attribute to an expression that combines the First Name and Last Name fields from the Candidate object.

```
<apex:column headerValue="Candidate Name">
  <apex:outputText value="{!application.
    candidate__r.First_Name__c & ' ' &
    application.candidate__r.Last_Name__c}"/>
</apex:column>
```

When you generate column fields in this manner, the value attribute on the <apex:column> component is not set, so the table doesn't know what to use as the column header. Rectify this by setting the headerValue attribute on this <apex:column> component.

Beyond the Basics

Did you know you can add a Chatter feed to a Visualforce page?

Say you want to add the Chatter feed for a position to its detail page. You can simply use the <chatter:feed> standard component:

```
<apex:page standardController="Position__c">
  <chatter:feed entityId="{!Position__c.id}">
    <apex:detail />
  </chatter:feed>
</apex:page>
```

To find out more, see the *Visualforce Developer's Guide* at wiki.developerforce.com/index.php/Documentation#Reference_Guides.

Try It Out: Creating a Custom List Button

Now that our Mass Update Status Visualforce page is complete, we're ready to create the button that users will click to access it.

The platform gives us the option of creating two types of buttons:

Detail Page Buttons

Buttons that appear on detail pages in the Button Section of the page layout.

List Buttons

Buttons that appear in list views, search result layouts, or related lists.

We need a button that users can click in the Job Applications related list on position records, so let's create a list button.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Job Application**.

You might be wondering why we chose the **Job Application** object instead of the **Position** object, given that the button will appear on position records. The reason is that you create list buttons on the object that is being listed.

3. In the Custom Buttons and Links related list, click **New**.

The screenshot shows the 'Custom Button or Link Edit' interface. The 'Label' field contains 'Update Status', the 'Name' field contains 'Update_Status', and the 'Description' field contains 'Updates the Status field on multiple job applications'. Under 'Display Type', 'List Button' is selected. The 'Behavior' dropdown is set to 'Display in existing window with sidebar'. The 'Content Source' dropdown is set to 'Visualforce Page'. A 'Quick Tips' panel on the right lists 'Getting Started', 'Sample Buttons & Links', and 'Operators & Functions'. Two red callout boxes provide additional context: one points to the Behavior field stating 'Behavior defines what will happen when the custom link is clicked', and another points to the Content Source field stating 'Content Source is the "target" of the custom link'.

Figure 112: Custom Button Edit Page

The custom button edit page should look familiar, since it closely resembles the formula editor that we saw previously.

4. In the Label text box, enter Update Status.
5. In the Name text box, accept the default value of Update_Status.

6. In the `Description` text box, enter `Updates the Status field on multiple job applications`.

We now have the option to specify whether we are creating a detail page link, detail page button, or list button. We've already decided to create a list button, but keep these other options in mind for future apps.

7. Select `List Button` as the `Display Type`.

When we selected the `List Button` option, did you notice the `Display Checkboxes (for Multi-Record Selection)` checkbox that appeared beneath it? By leaving this option selected, the platform knows to put checkboxes next to the records in the related list.

8. Leave the `Display Checkboxes (for Multi-Record Selection)` checkbox selected.

Next, the `Behavior` picklist lets you choose what happens when a user clicks the button. We know we want the button to open the `Mass Update Status` page, but we can specify whether the page opens in a new window or in the existing window, and whether or not it has a sidebar and header. You can even configure the button to execute some JavaScript if necessary. For our app, it makes the most sense to configure the button to open the `Mass Update Page` in the existing window, and to leave the sidebar and header there as well.

9. In the `Behavior` picklist, select `Display in existing window with sidebar`.

Now we need to specify the content we want to display. To do this, we'll first need to indicate the type of content we want to display in the `Content Source` picklist.

10. In the `Content Source` picklist, select `Visualforce Page`.

When you select the `Visualforce Page` option, the bottom section of the custom list button edit page displays a `Content` picklist. The picklist contains the `Visualforce` pages in your organization that have a standard controller set to the object for which you are creating the button.

11. In the `Content` drop-down list, select `MassUpdateStatus [MassUpdateStatus]`.

12. Click **Save**.

After clicking **Save**, you are reminded that no users will be able to access the button until it is added to a page layout. This is because creating a custom button is similar to adding a custom field to an object—even if it's defined in the database, nobody will be able to see it until you explicitly add it to a page layout.

Try It Out: Adding a Custom List Button to a Page Layout

Let's finish up our Mass Update Status feature by adding the Update Status list button to our Position page layouts.

1. Click **Your Name** ► **Setup** ► **Create** ► **Objects**.
2. Click **Position**.
3. In the Page Layouts related list, click **Edit** next to the page layout you want to edit first.
4. Click the wrench icon (🔧) in the Job Application related list to edit its properties.
5. In the Related List Properties window, click the **Buttons** bar at the bottom to expand the Buttons section.

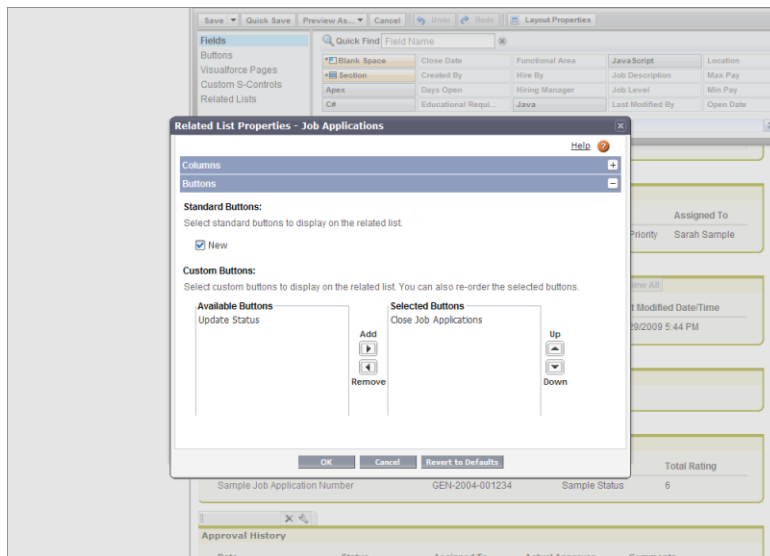


Figure 113: Custom Button Edit Page

Here we can specify the standard and custom buttons that the related list displays. The Update Status button we just created should appear in the Available Buttons list.

6. Select the Update Status button and click **Add** to move it to the Selected Buttons list.
7. Click **OK** to exit the Related List properties window.
8. Click **Save**.

Repeat the above procedure for the other Position page layout. Once that's done, the Mass Update Status feature is ready to test!

Try It Out: Testing the Mass Update Status Feature

To test the Mass Update Status feature, navigate to a position record to which multiple candidates have applied, and scroll down to the Job Applications related list. Notice that the **Update Status** button is there, and checkboxes appear next to the job application numbers, just as we planned.

Now select a few job applications and click the **Update Status** button. Our Mass Update Status Visualforce page appears. Notice that the lower half of the page lists the job application we selected on the previous page.

Job Application Number	Position Title	Candidate Name	Status
JA-00008	Documentation Writer	Chris McGuire	New
JA-00009	Documentation Writer	George Schnell	Schedule Interviews

Figure 114: Mass Update Status Page Displaying Selected Applications

In the **Status** picklist, choose the new status for the selected records, and click **Save**. Salesforce updates all of the selected records with the new value.

The feature is a success!

Taking Your App Public with Sites

We've looked extensively at what the Force.com platform can do for employees at Universal Containers, and how it's possible to build upon that power with Web services and Visualforce. Still, there's a key part to a company's success that's missing from the equation: the public. In today's world, it's critical for companies to exchange information not just with their own employees, but with customers, community members, developers, bloggers, and so forth. And given the lightening-fast pace of business in the 21st century, this information must be shared with ease and efficiency.

While most of the data in our Recruiting app is highly sensitive and should only be accessible to users with the appropriate permissions, there is one piece of data that should definitely be

publicized: open positions. By enabling job seekers outside of the company to view the positions that are available, we expand the pool of candidates and increase the odds of hiring the best person available. This is central to the success of not only the Human Resources department at Universal Containers, but the entire company.

In the past, to make Force.com data available to the general public, you had to set up a Web server, create custom Web pages (JSP, PHP, or other), and use the API to integrate Force.com apps with an external website. This is no longer the case, thanks to Sites!

Introducing Sites

With Sites, you can create public websites that are directly integrated with your Force.com organization. Anyone on the Internet can access these sites without logging in with a username and password.

Use sites to selectively expose a subset of your Force.com data through pages that are branded to match the look and feel of your company's website. For example:

- Host a public community forum for sharing and voting on ideas about your company, services, or products.
- Provide support information on a public website where customers can search for solutions to their issues.
- Add a public tool to your portal that helps customers find stores in their area.
- List all of your company's products on a public website, with model numbers, current prices, and product images pulled dynamically from your organization.

Sites are hosted by salesforce.com on the same servers that host the Force.com Web domain, so there are no data integration issues. And because you build sites on the Force.com platform with Visualforce pages, the Force.com platform handles all of the validation on the data collected through the site.

You can build multiple Force.com sites for your organization. For example, you can create one site for your Marketing department, a separate one for a community forum, and another for an online catalog.

In this chapter, we're going to build a very basic site called Career Resources that will allow any Web user to view the open positions at Universal Containers. While it's possible to add just about any kind of functionality to this site, such as search options and electronic job application forms, we're going to keep it simple to limit the amount of Visualforce needed to get the job done. By the time you're done with this chapter, though, anyone on the Internet will be able to see the positions that are open at Universal Containers, and you'll have a solid understanding of the power of Sites. Let's begin!

Introducing Force.com Domain Names

As mentioned above, sites are hosted on the Force.com domain. This saves you the hassle of dealing with a third-party Web hosting service, and allows for seamless integration with the Force.com platform. Each Salesforce account that uses sites has its own reserved section on the Force.com domain servers.

Before you begin creating sites, you must register a unique Force.com domain name that will identify your account's space on the Force.com domain. Your Force.com domain name is a URL that consists of a prefix followed by a string of characters that Salesforce provides.

Prior to creating your first site, you must register your Force.com domain name. You only have to do this one time, though. Once your Force.com domain name is registered, you never have to repeat this part of the process again.

Try It Out: Registering Your Force.com Domain Name

To register your Force.com domain name:

1. Click ***Your Name*** ► **Setup** ► **Develop** ► **Sites**.

The sites registration page appears. Near the bottom of the page is a text box in a URL. This is where you will enter the prefix for your Force.com domain name. Before you do, though, consider the following:

- The Force.com domain name can contain only underscores and alphanumeric characters (no spaces or punctuation), and must begin with a letter. Also, it can't end with an underscore or contain two consecutive underscores.
- Your Force.com domain name must be unique throughout the entire Force.com domain. To avoid any duplicate entries, consider using your company's name or a variation of your company's name, such as `mycompanysite`.
- Salesforce combines the prefix you enter with the rest of the URL on the page to create your Force.com domain name. For example, if you enter `mycompanysite` as your prefix and the rest of the URL is `-developer-edition.na1.force.com`, your full Force.com domain name will be `www.mycompanysite-developer-edition.na1.force.com`.

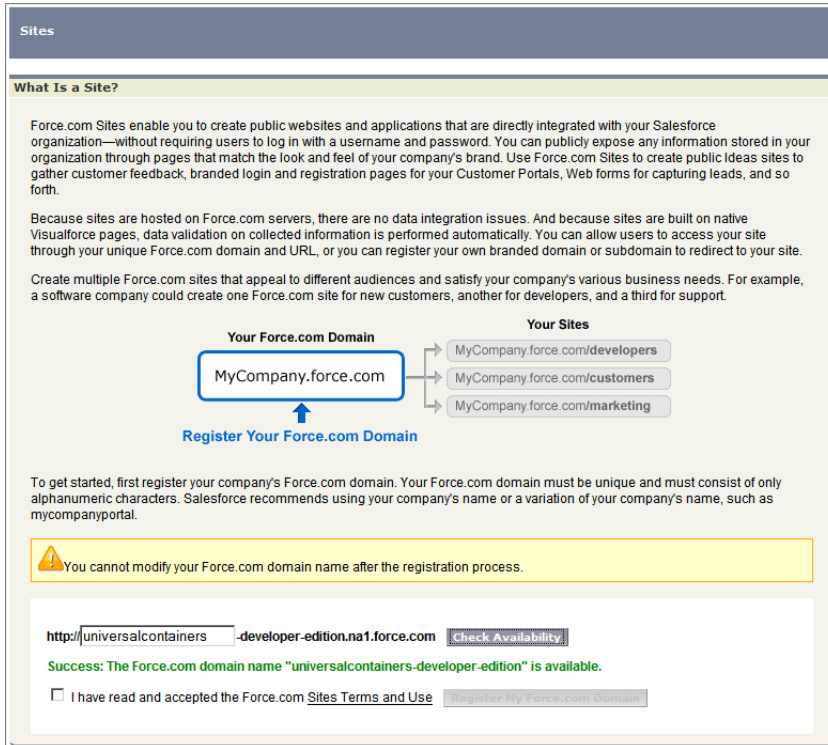


Figure 115: Sites Registration Page

2. In the text box, enter a prefix for your Force.com domain name.
3. Click **Check Availability**.

Salesforce quickly scans all of the other Force.com domain names to make sure the one you entered is unique. If your Force.com domain name is unique, you'll see a message indicating it's available. If it's not unique, you'll have to try another name.

4. Click the link to review the Sites Terms of Use. Acknowledge your acceptance by selecting the checkbox.
5. Click **Register My Force.com Domain**.

A prompt appears warning that you can't change your Force.com domain name after you register it.

6. Click **OK**.

Nice! You're now ready to create your first Force.com site.

Try It Out: Creating a Force.com Site

After registering your Force.com domain name, you should still be on the sites introductory page. If you navigated away from the page, click **Your Name > Setup > Develop > Sites** to return.

1. Click **New**.

The Sites edit page appears. The first field, `Site Label`, is the name of the site as it appears in the administration setup area of the Force.com platform user interface.

2. In the `Site Label` text box, enter `Career Resources`.

When you click out of the `Site Label` text box, notice that the value `Career_Resources` appears in the `Site Name` text box. `Site Name` is the name used when referencing the site in the API.

3. Leave `Career_Resources` as the value of the `Site Name` text box.
4. In the `Site Description` field, enter `A public site for employment-related information at Universal Containers`.

Every site must have a designated user who can receive site-related communications from the site's visitors and salesforce.com.

5. In the `Site Contact` field, click the lookup icon and choose yourself if your name is not already the default value.

The next field, `Default Web Address`, displays your Force.com domain name followed by a text box. In the text box, enter a unique domain suffix using alphanumeric characters. The Force.com domain name and suffix constitute the complete default Web address for this site. For example, if your Force.com domain name is `www.mycompanysite-developer-edition.na1.force.com` and the suffix is `careers`, the default Web address for this site is `www.mycompanysite-developer-edition.na1.force.com/careers`.

After we set up and activate the site, anyone on the Web will be able to access the site by entering the default Web address.

6. In the `Default Web Address` text box, enter `careers`.

As with any Web page, you can allow your site to be accessed via an alternative URL, one that is perhaps shorter or easier to remember. To do this, you have to set up a custom Web address through a domain name registrar and configure it to redirect users to the default Web address specified above. If you are building the Recruiting app in your sandbox or a version of Salesforce other than Developer Edition, you'll see a `Custom Web Address` field in which you can

enter that custom Web address. For now, leave this field blank; however, if you create a custom Web address in the future for this site, return to this page to enter the address in this field.

7. Leave the `Custom Web Address` text box blank.

A site is not available to the public until it is activated. If you were creating a site for a public company or working on a complex site that took several days to develop, it would make sense to create and test the site before making it available to everyone. Since we're creating a simple site for a fictitious company, it's OK for our site to be immediately available when we save this page.

8. Select the `Active` checkbox.

The `Active Site Home Page` field lets you specify which Visualforce page people see if they navigate to your site when it's active. Ultimately, we'll want this to be a Visualforce page that displays all of the open positions at Universal Containers page, but we haven't created that page yet. For now, let's just use the default `UnderConstruction` page included with the Force.com platform.

9. In the `Active Site Home Page` field, leave `UnderConstruction` selected.

Even though we opted to make our site immediately active, there still may be times in the future in which we'll want to deactivate the site for maintenance or redesigning. When the site is inactive, users are redirected to the Visualforce page listed in the `Inactive Site Home Page` field. Again, the Force.com platform includes a default Visualforce page for this purpose.

10. In the `Inactive Site Home Page` field, leave `InMaintenance` selected.

We haven't begun designing the pages of our site yet, but while we're setting up our site's basic configuration, we can specify a site template to make the construction of our site's pages much easier. A *site template* is a Visualforce page that lets you dictate the style and layout of any of your site's pages that don't reference a specific site template. That is, every page on your site will inherit the site template's look and feel unless you specify another template on the page itself. With site templates, you can control the visual styling of your entire site from a single file!

If you have multiple sites, you can use the same site template for each site to preserve a uniform style throughout your company's entire Web presence. Alternatively, you can create different site templates for different sites so that visitors instantly recognize a change in context. For our `Careers Resources` site, we're going to use the default Visualforce page provided with all organizations called `SiteTemplate`.

11. Leave `SiteTemplate` selected in the `Site Template` field.

The next few fields and checkbox are a bit technical and don't affect our site's functionality, so we're going to leave them blank. It's useful to understand what they can do, however, since it's quite likely you'll want to utilize them when creating sites in the future:

Site Robots.txt

This file determines which parts of your public site are accessible to Web crawlers. A *Web crawler* is a computer program that browses the Web and retrieves current data. The data is primarily used by search engines for indexing purposes, and for automating maintenance tasks on Web sites, such as checking for broken links. However, there are reasons to deny Web crawlers access to your site. For example, some Web crawlers are maliciously designed to gather contact information from Web pages, such as email addresses, for spam. Also, Web crawlers can consume networking resources, impeding download rates for legitimate Web users.

Site Favorite Icon

This icon appears in a browser's address field when a user visits the site. You can use this field to set the favorite icon for your entire site, or you can specify an icon for each individual page.

Analytics Tracking Code

This code lets you easily integrate your site with Google Analytics. By integrating your site with Google Analytics, you can track the usage of your site and site pages, including the number of visits and page views over a specific period of time, the average time visitors spent on the site, and more.



Note: This option may not be available in all organizations.

URL Rewriter Class

After setting up your site, you can create an Apex class to rewrite your URLs to make them more user-friendly. Enter the class in this field to apply it to your site. To find out more, see “[Site.UrlRewriter](#)” in the *Force.com Apex Code Developer's Guide*.

Enable Feeds

This option displays the Syndication Feeds related list, where you can create and manage syndication feeds for users on your public sites. *Syndication feeds* give users the ability to receive updates in external news readers when there are changes to a site. This field is visible only if you have the feature enabled for your organization.

12. Click **Save**. The Site detail page appears.



Figure 116: The Site Detail Page

Your first site is live! If you go to the URL you entered in the Default Web Address field or just click the link in the Default Web Address field on the Site detail page, you'll see the UnderConstruction Visualforce page that we selected as our Active Site Home Page.



Note: Sometimes it takes a few minutes for changes to your sites to appear. If the changes don't appear immediately, check back a few minutes later.

Notice that the **email us** link opens an email addressed to the site contact.

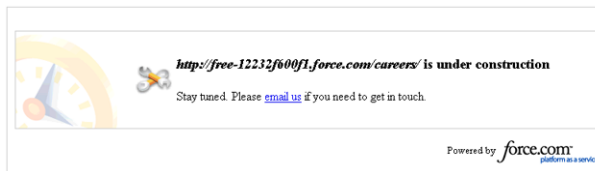


Figure 117: The UnderConstruction Page

If you go back to the Site detail page and scroll down, you'll see several related lists that display information such as the site's usage and change history. One of these related lists contains the various Visualforce pages associated with the site. These are default pages included with the platform though, and none of them display the positions open at Universal Containers, which is the whole reason we're building the site. We'll have to create that page ourselves, but that won't be hard since we already picked up a few Visualforce tricks earlier in this chapter.

Try It Out: Creating the Public Jobs Page

Now that we're ready to create our Public Jobs Visualforce page, let's iron out the details so we can understand precisely what we're creating.

First, we know that we want the page to display some of the data on position records, such as each position's title, location, and description. Those are essential bits of information to job seekers. We probably shouldn't make the pay ranges public, though, since that information could give candidates the upper hand when negotiating salaries. Also, keep in mind that the page is going to list all open positions, and large scale companies might have hundreds or thousands of open positions. For that reason we'll want to restrict the information we display to just the three essential fields mentioned above.

Next, we have to decide how the page will look. What colors will we use? What format will we use to render the data? As mentioned before, we want to keep things simple, so let's give the page the same look and feel that position records have in our app. We'll render the data in a table, and we'll add a brief introductory blurb above the table that will orientate the site's visitors by describing the contents of the page and how they should use it. Let's begin!

1. In the address bar of your browser, replace everything to the right of `salesforce.com/` with `apex/PublicJobs`.

The resulting URL should look something like this:

`https://na1.salesforce.com/apex/PublicJobs`. Remember to only modify the part of the URL after `salesforce.com/`.

2. Press **Enter**.
3. Click the **Create Page PublicJobs** link.

We now have a new Visualforce page called `PublicJobs`. The next step is to add the Visualforce code to the page that will display open positions.

4. In the footer at the bottom of the `PublicJobs` Visualforce page, click **Page Editor** to display the Visualforce development mode page editor.
5. Delete the default markup in the Visualforce development mode page editor and enter the following:



Tip: You can cut and paste this markup from a file in the `RecruitingApp-5_0.zip` file you downloaded from developer.force.com/books/fundamentals.

```
<apex:page standardController="Position__c"
  recordSetVar="positions"
  sidebar="false" showHeader="false">
```

```

<apex:pageBlock>
<h1>Welcome to the Universal Containers Careers
  Home Page!</h1>
<p>Universal Containers is an industry leader,
  but to stay ahead of the pack, we need to grow!
  We are currently seeking bright and talented
  professionals to join our winning team. Browse
  the current openings below and email your resume
  to apply.
</p>
<br/>
<apex:pageBlockTable value="{!positions}"
  var="position">
  <apex:column value="{!position.name}"
    rendered="{!IF(position.Status__c ==
  'Open - Approved', true, false)}"/>
  <apex:column value="{!position.Location__c}"
    rendered="{!IF(position.Status__c ==
  'Open - Approved', true, false)}"/>
  <apex:column value="{!position.Job_Description__c}"
    rendered="{!IF(position.Status__c ==
  'Open - Approved', true, false)}"/>
</apex:pageBlockTable>
</apex:pageBlock>
</apex:page>

```

6. Click **Save**.

The public jobs page appears:

Welcome to the Universal Containers Careers Home Page!		
Universal Containers is an industry leader, and to stay ahead of the pack, we need to grow! We are currently seeking bright and talented professionals to join our winning team. Browse the current openings below, and send in your resume via email to apply today.		
Account Executive I	London, England	Do you have what it takes to be with the best-of-the best? Do you thrive on doing deals? Working in Sales at UCI provides a compelling opportunity for Sales Professionals with a proven track record exceeding quota in technology sales. With over 22,700 loyal, global customers, a Marketing focus on lead generation and sales-driven culture, it is no wonder that last year the majority of our Sales organization exceeded quota. We are seeking proven, quota-carrying sales performers to help us grow our loyal customer base for medium-to-large markets.
Account Executive III	Mumbai, India	Do you have what it takes to be with the best-of-the best? Do you thrive on doing deals? Working in Sales at UCI provides a compelling opportunity for Sales Professionals with a proven track record exceeding quota in technology sales. With over 22,700 loyal, global customers, a Marketing focus on lead generation and sales-driven culture, it is no wonder that last year the majority of our Sales organization exceeded quota. We are seeking proven, quota-carrying sales performers to help us grow our loyal customer base for medium-to-large markets.
Account Executive IV	Sydney, Australia	Do you have what it takes to be with the best-of-the best? Do you thrive on doing deals? Working in Sales at UCI provides a compelling opportunity for Sales Professionals with a proven track record exceeding quota in technology sales. With over 22,700 loyal, global customers, a Marketing focus on lead generation and sales-driven culture, it is no wonder that last year the majority of our Sales organization exceeded quota. We are seeking proven, quota-carrying sales performers to help us grow our loyal customer base for medium-to-large markets.
DBA	Austin, TX	Senior Oracle 11i Apps DBA to support an 11i Financials implementation. Must have experience on multiple 11i implementation projects, preferably supporting large customers.
Sr. UI Designer	San Francisco, CA	As a designer of the UCI interface, you will own and control the user interaction experience and information flow in architecture of the application, as well as the visual interface design and look-and-feel of UCI. Your mission will be to create a user experience that results in successful usage of UCI features and functions for both business users and systems administrators.
SW Engineer	San Francisco, CA	As a member of the development team, design, develop, test and debug computer software applications, including building highly scalable web applications using Java and Oracle technologies. Write object oriented software in Java and database code in SQL. Fine tune application code in Java and SQL to improve system performance. Build new software functionality and resolve technical issues of existing software. Analyze and design and develop test cases and implement automated test suites while maintaining documentation test results and recommending corrective action. Analyze tests, resolve errors, implement solutions and perform functionality and regression testing for web-based applications.

Figure 118: The Public Jobs Page

You probably recognize the components in the markup since we used all of them before in our Mass Update Status feature. The page begins with the familiar `<apex:page>` and `<apex:pageBlock>` components, followed by a few elementary HTML tags that add a header and some introductory text. Then the `<apex:pageBlockTable>` component and a few `<apex:column>` components create a table that displays fields from a set of records. Simple!

While all of the components are familiar, the markup uses a few attributes that we haven't seen yet. Let's take a look at them.

First, in the `<apex:page>` component, we set the `sidebar` and `showHeader` attributes to `false`. These attributes specify whether the page displays the standard sidebar and tab header you see on your app's pages. If we set these values to `true` or don't set them at all, the sidebar and header are visible. However, the search functionality in our app is restricted to registered users, so having the sidebar appear might mislead our site's visitors into believing they can search for positions when in fact their search attempt will result in an error message. As for the tabs, they don't really make sense on a single-page site like the one we're building, so we shouldn't display them. Therefore, both of these attributes are set to `false`.

The other attribute that we haven't covered yet is the `rendered` attribute on the `<apex:column>` component. If the value of this attribute is `true`, the field specified in the `value` attribute appears in the table; if the value is `false`, the field doesn't appear. Since this value should change depending on the status of the position we're displaying, we've set the attribute to an expression that *resolves* to either `true` or `false`. That way, the page can dynamically determine whether or not to display fields.

For example, the `rendered` attributes in our markup are set to the following expression:

```
{!IF(position.Status__c == 'Open - Approved', true, false)}
```

The expression uses a simple `IF()` function that only resolves to `true` if the `Status` field on a position record is set to “Open — Approved.” This means that our `<apex:column>` components will only display fields from position records that are open, which is perfect because we only want open positions to appear on our Public Jobs page.

That's all the Visualforce markup we'll need for our site! We just have a few more clicks left to finish up our app.

Setting Your Active Home Page

When we created our site, we set its `Active Site Home Page` field to the `UnderConstruction` page because we had yet to create our Public Jobs page. Now that our public jobs page is ready to go, let's set it as the site's home page.

1. Click **Your Name** ► **Setup** ► **Develop** ► **Sites**.
2. Click **Edit** next to the name of your site.
3. Use the lookup field to find and select the `PublicJobs` page for the `Active Site Home Page` field.
4. Click **Save**.

The Public Jobs page is now the home page for your Career Resources site; however, if you launch the default Web address in your browser, the Public Jobs page is not there. Instead, you see a Web page with a message saying that authorization is required. Why would authorization be necessary to access a public site?

It turns out there's still one step left. We have to grant public access to the position object.

Granting Public Access Settings

Public access settings control what public users can do on each site. These settings are similar to the profile settings we set in [Expanding the Simple App Using Relationships](#), only this time

we're setting them specifically for this site and not the entire app. In fact, the settings only apply to Internet users who do *not* have a Recruiting App login.

For our site, we need to give the public “Read” access to the Position object. If we don't, the platform will not render the Public Jobs page because the page contains position data to which the public doesn't have access.

1. From the Site Details page, click **Public Access Settings**.
2. Click **Edit**.
3. Under Custom Object Permissions, enable the “Read” permission on Positions.
4. Click **Save**.

We're done! Let's test our site.

Testing Your Site

Testing your site is easy—simply point your browser to the site's default Web address. You can do this by entering the default Web address in your browser's address bar, or by clicking the `Default Web Address` link on your site's detail page.

Once the Public Jobs page loads, you can see the title, location, and description of each open position at Universal Containers, even if you aren't logged in to the Recruiting App. We've successfully published a subset of our data while keeping the rest of it secure!

We're going to stop here since we achieved our targeted goal of creating a public site. However, it's good to think about the additional functionality that you could build into your site to improve its usability. For instance, career websites like ours typically have search capabilities that let visitors view positions based on specified criteria. Also, it's nice to give users a more efficient way to apply online rather than sending a resume in an email.

All this and so much more can be implemented using advanced Visualforce markup and Apex. Those topics are out this book's scope, but they are thoroughly covered in the various resources available at developer.force.com. If you go on to learn more about Visualforce and Apex, you can become a true master of cloud computing!

Chapter 13

Learning More

In this chapter ...

- [Developer Force](#)
- [Help and Training Options](#)
- [Multimedia](#)
- [AppExchange Partner Program](#)
- [What Do You Think?](#)

This book has introduced some of the native technologies associated with the Force.com platform. We've created a fully functional Recruiting app, and we've introduced Visualforce and the Web services API to show you how it can be used to build composite apps. But there's only so much we can cover in a single book—we skipped over many other powerful tools and options, and didn't talk about how you can share your own apps with others on the AppExchange. Indeed, we'd be surprised if you didn't have more questions about how you can take advantage of all that the platform has to offer!

Fortunately there are a number of ways you can learn more about the platform.

Developer Force

The Developer Force website is the place to be for Force.com developers. It provides a full range of resources including sample code, toolkits, an online developer community, and the test environments necessary for building apps. It includes an online version of this book (written by Developer Force staff members) and has information about the Dreamforce event that we hold every year for Force.com platform developers. If you need more info, have a question to ask, are seeking a toolkit or sample, or just want to dig a little deeper into Force.com platform development, the Developer Force website is the first place you should go.

Let's take a quick tour:

- **Get a Developer Edition account.** Take your application-building skills to the next level. A Developer Edition account is a free, fully-functional Salesforce account that allows you to get hands-on experience with all aspects of the platform in an environment designed for development. If you haven't done so already, go to developer.force.com/join and create a Developer Edition account for yourself. Then use it to try out new application ideas and to test various Force.com platform tools and technologies.
- **Get the resources you need on the Developer Force wiki.** Of course, a big part of Developer Force's mission is to ensure that you and every other Force.com platform developer has the support necessary to build great apps. To this end, you'll want to know about the Wiki section of the Developer Force site—wiki.developerforce.com. This is the place to turn for tools, documentation, and reference information for developers, whether you're building native apps like the Recruiting app, composite apps that use the Web services API, writing Apex, or getting ready to distribute your app on the AppExchange. And because it's all maintained on a wiki, you'll be able to see tips, tools, and best practices from other Developer Force members, as well as contribute your own! By developers and for developers, the wiki is the heart of the Developer Force website.
- **Want to learn more?** Build as you learn with salesforce.com's workbooks! The *Force.com Workbook* gets you started building an app in the cloud with a set of 30-minute tutorials that introduce various Force.com platform features. The *Chatter Workbook* takes you through a series of tutorials on developing with Chatter. And the *Visualforce Workbook* gets you started building custom user interface Visualforce pages. You can find them all at wiki.developerforce.com/index.php/Documentation#Getting_Started.

Another great resource worth checking out is Jason Ouellette's book *Development with the Force.com Platform: Building Business Applications in the Cloud*.

- **Take advantage of community code samples.** The *Force.com Cookbook* collaborative site contains dozens of “recipes” for both novice and expert developers. Learn how to add even more exciting functionality to your apps by accessing the Force.com API, developing Apex

code, and creating Visualforce pages. You can read and comment on existing recipes, or submit your own recipes, at developer.force.com/cookbook.

- **Grab these quick reference guides.** The Force.com developer cheat sheets are condensed, one- to four-page documents with the most commonly used reference material for a particular subject. Print them out and hang them on your wall for a handy quick reference. There are cheat sheets on Apex code, formulas, Visualforce, the Web services API, and Chatter. You can find them at wiki.developerforce.com/index.php/Documentation#Developer_Cheat_Sheets.
- **Keep up with the Developer Force community.** Your fellow developers of all levels of sophistication are maintaining lively conversations every day on the Developer Force discussion boards at community.salesforce.com/sforce?category.id=developers. You're welcome to join in by posting questions and comments of your own, or you can just read along and learn. The free monthly Developer Force newsletter provides a recap of the most interesting news and information, delivered right to your email inbox.
- **Get the latest Force.com platform news.** The Developer Force blog at blog.sforce.com/ is where the Developer Force team makes announcements and shares some of what it has learned in the process of creating and managing more than 300 apps on the AppExchange directory. This blog thrives on feedback and the cross-pollination of ideas. We welcome your comments.
- **Take part in Developer Force events and activities.** Developer Force also sponsors various special events and activities, like the very popular Developer Force Seminars tour, training classes, and our biggest event of the year, Dreamforce—the conference within a conference for Force.com application developers. Information on upcoming events is posted on the Developer Force site at developer.force.com and is included in the Developer Force newsletter.

Help and Training Options

In addition to the Developer Force website, the platform itself offers lots of help and training options:

- **Find answers to your questions.** Click **Help** at the top of any page in the application. Enter your keywords in the Search box and click **Go!**. The search returns online help topics, knowledge base solutions, and recommended training classes that match the keywords you entered.
- **Take online training.** Select the Training tab of the Help & Training window, choose your role and geographic location, and click **View Classes!** to find free online training classes. More than fifteen online training classes are available to you on-demand, 24/7!

- **Attend a class.** A number of examples in this book have been provided by Salesforce.com Training & Certification and are drawn from the expert-led training courses available around the world. Salesforce.com training courses provide an opportunity to get hands-on experience with the Force.com platform and Salesforce applications, and prepare you to become Salesforce Certified. To learn more or register for a class, visit www.salesforce.com/training.
- **Download tip sheets and best practice guides.** Select the Help tab of the Help & Training window and click **Tips** in the task bar to view and download tip sheets, implementation guides, and best practices for specific features.

Multimedia

Thanks to the hard work of our colleagues on the Salesforce.com Community and Force.com websites, we have an impressive number of podcasts and videos available. Our podcasts and videos keep the Force.com platform community connected by providing free access to a wide range of best practices, case studies, and product- and platform-focused digital audio content.

Find podcasts on our iTunes channel by searching for “salesforce” in the iTunes Music Store. To find our videos, go to the <http://developer.force.com/content/type/Tech+Talk> on the Developer Force website.

Access to expert Force.com platform and CRM-related luminary interviews, thought-leadership presentations, roundtable discussions, and best practices are just a click away. Happy listening and watching!

AppExchange Partner Program

With the emergence of The Business WebTM, companies can offer their services and applications to businesses over the Internet as easily as retailers and auctioneers can connect with online consumers. As AppExchange partners, more than 150 companies are already participating in this new chapter of computing by making their offerings available on The Business Web via the AppExchange. The AppExchange partner program makes it easy for both new and established businesses to join this growing community of cloud computing providers. Visit the Partner Program website at www.salesforce.com/partners/ and join the Force.com AppExchange partner program today!

What Do You Think?

Well that about sums it up. Did you like what you've read? Has it inspired you to create your own on-demand apps and enter the world of cloud computing? We certainly hope so! We welcome any comments you might have—indeed, we count on your feedback and ideas. Go to the Developer Force discussion boards at community.salesforce.com/sforce?category.id=developers or email us at developerforce@salesforce.com and let us know what you think!

Glossary

This glossary defines terms that appear in this guide.

A

Activity (Calendar Events/Tasks)

Planned task or event, optionally related to another type of record such as an account, contact, lead, opportunity, or case.

Administrator (System Administrator)

One or more individuals in your organization who can configure and customize the application. Users assigned to the System Administrator profile have administrator privileges.

Advanced Function

A formula function designed for use in custom buttons, links, and s-controls. For example, the `INCLUDE` advanced function returns the content from an s-control snippet.

Analytic Snapshot

An analytic snapshot lets you report on historical data. Authorized users can save tabular or summary report results to fields on a custom object, then map those fields to corresponding fields on a target object. They can then schedule when to run the report to load the custom object's fields with the report's data.

Analytic Snapshot Running User

The user whose security settings determine the source report's level of access to data. This bypasses all security settings, giving all users who can view the results of the source report in the target object access to data they might not be able to see otherwise.

Analytic Snapshot Source Report

The custom report scheduled to run and load data as records into a custom object.

Analytic Snapshot Target Object

The custom object that receives the results of the source report as records.

Apex

Force.com Apex code is a strongly-typed, object-oriented programming language that allows developers to execute flow and transaction control statements on the Force.com platform server in conjunction with calls to the Force.com API. Using syntax that looks like Java and acts like database stored procedures, Apex code enables developers to add business logic to most system events, including button clicks, related record updates, and

Visualforce pages. Apex scripts can be initiated by Web service requests and from triggers on objects.

Apex Controller

See Controller, Visualforce.

Apex Page

See Visualforce Page.

App

Short for “application.” A collection of components such as tabs, reports, dashboards, and Visualforce pages that address a specific business need. Salesforce provides standard apps such as Sales and Call Center. You can customize the standard apps to match the way you work. In addition, you can package an app and upload it to the AppExchange along with related components such as custom fields, custom tabs, and custom objects. Then, you can make the app available to other Salesforce users from the AppExchange.

App Menu

See Force.com App Menu.

AppExchange

The AppExchange is a sharing interface from salesforce.com that allows you to browse and share apps and services for the Force.com platform.

Application Programming Interface (API)

The interface that a computer system, library, or application provides to allow other computer programs to request services from it and exchange data.

Approval Action

See Workflow and Approval Actions.

Approval Process

An approval process is an automated process your organization can use to approve records in Salesforce. An approval process specifies the steps necessary for a record to be approved and who must approve it at each step. A step can apply to all records included in the process, or just records that have certain attributes. An approval process also specifies the actions to take when a record is approved, rejected, recalled, or first submitted for approval.

Auto Number

A custom field type that automatically adds a unique sequential number to each record. These fields are read only.

B**Boolean Operators**

You can use Boolean operators in report filters to specify the logical relationship between two values. For example, the AND operator between two values yields search results that include both values. Likewise, the OR operator between two values yields search results that include either value.

C**Class, Apex**

A template or blueprint from which Apex objects are created. Classes consist of other classes, user-defined methods, variables, exception types, and static initialization code. In most cases, Apex classes are modeled on their counterparts in Java.

Clone

Clone is the name of a button or link that allows you to create a new item by copying the information from an existing item, for example, a contact or opportunity.

Cloud Computing

A model for software development and distribution based on the Internet. The technology infrastructure for a service, including data, is hosted on the Internet. This allows consumers to develop and use services with browsers or other thin clients instead of investing in hardware, software, or maintenance.

Combination Chart

A combination chart plots multiple sets of data on a single chart. Each set of data is based on a different field, so values are easy to compare. You can also combine certain chart types to present data in different ways on a single chart.

Component, Visualforce

Something that can be added to a Visualforce page with a set of tags, for example, `<apex:detail>`. Visualforce includes a number of standard components, or you can create your own custom components.

Component Reference, Visualforce

A description of the standard and custom Visualforce components that are available in your organization. You can access the component library from the development footer of any Visualforce page or the *Visualforce Developer's Guide*.

Controller, Visualforce

An Apex class that provides a Visualforce page with the data and business logic it needs to run. Visualforce pages can use the standard controllers

that come by default with every standard or custom object, or they can use custom controllers.

Controller Extension

A controller extension is an Apex class that extends the functionality of a standard or custom controller.

Controlling Field

Any standard or custom picklist or checkbox field whose values control the available values in one or more corresponding dependent fields.

Custom App

See App.

Custom Controller

A custom controller is an Apex class that implements all of the logic for a page without leveraging a standard controller. Use custom controllers when you want your Visualforce page to run entirely in system mode, which does not enforce the profile-based permissions and field-level security of the current user.

Custom Field

A field that can be added in addition to the standard fields to customize Salesforce for your organization's needs.

Custom Help

Custom text administrators create to provide users with on-screen information specific to a standard field, custom field, or custom object.

Custom Links

Custom URLs defined by administrators to integrate your Salesforce data with external websites and back-office systems. Formerly known as Web links.

Custom Object

Custom records that allow you to store information unique to your organization.

Custom Report Type

See Report Type.

Custom Settings

Custom settings are similar to custom objects and enable application developers to create custom sets of data, as well as create and associate custom data for an organization, profile, or specific user. All custom settings data is exposed in the application cache, which enables efficient access without the cost of repeated queries to the database. This data can then be used by formula fields, validation rules, Apex, and the Web services API.

See also Hierarchy Custom Settings and List Custom Settings.

Custom View

A display feature that lets you see a specific set of records for a particular object.

D**Dashboard**

A *dashboard* shows data from source reports as visual components, which can be charts, gauges, tables, metrics, or Visualforce pages. They provide a snapshot of key metrics and performance indicators for your organization. Each dashboard can have up to 20 components.

Database

An organized collection of information. The underlying architecture of the Force.com platform includes a database where your data is stored.

Database Table

A list of information, presented with rows and columns, about the person, thing, or concept you want to track. See also Object.

Data Loader

A Force.com platform tool used to import and export data from your Salesforce organization.

Decimal Places

Parameter for number, currency, and percent custom fields that indicates the total number of digits you can enter to the right of a decimal point, for example, 4.98 for an entry of 2. Note that the system rounds the decimal numbers you enter, if necessary. For example, if you enter 4.986 in a field with `Decimal Places` of 2, the number rounds to 4.99. Salesforce uses the round half-up rounding algorithm. Half-way values are always rounded up. For example, 1.45 is rounded to 1.5. -1.45 is rounded to -1.5.

Delegated Administration

A security model in which a group of non-administrator users perform administrative tasks.

Delegated Authentication

A security process where an external authority is used to authenticate Force.com platform users.

Dependency

A relationship where one object's existence depends on that of another. There are a number of different kinds of dependencies including mandatory fields, dependent objects (parent-child), file inclusion (referenced images, for example), and ordering dependencies (when one object must be deployed before another object).

Dependent Field

Any custom picklist or multi-select picklist field that displays available values based on the value selected in its corresponding controlling field.

Detail

A page that displays information about a single object record. The detail page of a record allows you to view the information, whereas the edit page allows you to modify it.

A term used in reports to distinguish between summary information and inclusion of all column data for all information in a report. You can toggle the **Show Details/Hide Details** button to view and hide report detail information.

Developer Edition

A free, fully-functional Salesforce organization designed for developers to extend, integrate, and develop with the Force.com platform. Developer Edition accounts are available on developer.force.com.

Development as a Service (DaaS)

An application development model where all development is on the Web. This means that source code, compilation, and development environments are not on local machines, but are Web-based services.

Development Environment

A Salesforce organization where you can make configuration changes that will not affect users on the production organization. There are two kinds of development environments, sandboxes and Developer Edition organizations.

Developer Force

The Developer Force website at developer.force.com provides a full range of resources for platform developers, including sample code, toolkits, an online developer community, and the ability to obtain limited Force.com platform environments.

Document Library

A place to store documents without attaching them to accounts, contacts, opportunities, or other records.

E**Email Alert**

Email alerts are workflow and approval actions that are generated using an email template by a workflow rule or approval process and sent to designated recipients, either Salesforce users or others.

Email Template

A form email that communicates a standard message, such as a welcome letter to new employees or an acknowledgement that a customer service

request has been received. Email templates can be personalized with merge fields, and can be written in text, HTML, or custom format.

Enterprise Application

An application that is designed to support functionality for an organization as a whole, rather than solving a specific problem.

Enterprise Edition

A Salesforce edition designed for larger, more complex businesses.

Entity Relationship Diagram (ERD)

A data modeling tool that helps you organize your data into entities (or objects, as they are called in the Force.com platform) and define the relationships between them. ERD diagrams for key Salesforce objects are published in the *Web Services API Developer's Guide*.

Event

An event is an activity that has a scheduled time. For example, a meeting, or a scheduled phone call.

F**Field**

A part of an object that holds a specific piece of information, such as a text or currency value.

Field-Level Security

Settings that determine whether fields are hidden, visible, read only, or editable for users based on their profiles. Available in Enterprise, Unlimited, and Developer Editions only.

Field Dependency

A filter that allows you to change the contents of a picklist based on the value of another field.

Field Update

Field updates are workflow and approval actions that specify the field you want updated and the new value for it. Depending on the type of field, you can choose to apply a specific value, make the value blank, or calculate a value based on a formula you create.

Filter Condition/Criteria

Condition on particular fields that qualifies items to be included in a list view or report, such as “State equals California.”

Folder

A *folder* is a place where you can store reports, dashboards, documents, or email templates. Folders can be public, hidden, or shared, and can be set to read-only or read/write. You control who has access to its contents based on roles, permissions, public groups, and license types.

Force.com

The salesforce.com platform for building applications in the cloud. Force.com combines a powerful user interface, operating system, and database to allow you to customize and deploy applications in the cloud for your entire enterprise.

Force.com App Menu

A menu that enables users to switch between customizable applications (or “apps”) with a single click. The Force.com app menu displays at the top of every page in the user interface.

Force.com IDE

An Eclipse plug-in that allows developers to manage, author, debug and deploy Force.com applications in the Eclipse development environment.

Web Services API

A SOAP-based Web services application programming interface that provides access to your Salesforce organization's information. See also Bulk API.

Foreign key

A field whose value is the same as the primary key of another table. You can think of a foreign key as a copy of a primary key from another table. A relationship is made between two tables by matching the values of the foreign key in one table with the values of the primary key in another.

Formula Field

A type of custom field. Formula fields automatically calculate their values based on the values of merge fields, expressions, or other values.

Function

Built-in formulas that you can customize with input parameters. For example, the DATE function creates a date field type from a given year, month, and day.

G**Global Variable**

A special merge field that you can use to reference data in your organization.

A method access modifier for any method that needs to be referenced outside of the application, either in the Web services API or by other Apex scripts.

Group

A group is a set of users. Groups can contain individual users, other groups, or the users in a role. Groups can be used to help define sharing access to data or to specify which data to synchronize when using Connect for Outlook or Connect for Lotus Notes.

Users can define their own personal groups. Administrators can create public groups for use by everyone in the organization.

Group Edition

A product designed for small businesses and workgroups with a limited number of users.

H

Hierarchy Custom Settings

A type of custom setting that uses a built-in hierarchical logic that lets you “personalize” settings for specific profiles or users. The hierarchy logic checks the organization, profile, and user settings for the current user and returns the most specific, or “lowest,” value. In the hierarchy, settings for an organization are overridden by profile settings, which, in turn, are overridden by user settings.

Home Tab

Starting page from which users can choose sidebar shortcuts and options, view current tasks and activities, or select another tab.

Hover Detail

Hover details display an interactive overlay containing detailed information about a record when users hover the mouse over a link to that record in the Recent Items list on the sidebar or in a lookup field on a record detail page. Users can quickly view information about a record before clicking **View** for the record's detail page or **Edit** for the edit page. The fields displayed in the hover details are determined by the record's mini page layout. The fields that display in document hover details are not customizable.

I

ID

See Salesforce Record ID.

Integrated Development Environment (IDE)

A software application that provides comprehensive facilities for software developers including a source code editor, testing and debugging tools, and integration with source code control systems.

Immediate Action

A workflow action that executes instantly when the conditions of a workflow rule are met.

Import Wizard

A tool for importing data into your Salesforce organization, accessible from Setup.

Instance

The cluster of software and hardware represented as a single logical server that hosts an organization's data and runs their applications. The Force.com platform runs on multiple instances, but data for any single organization is always consolidated on a single instance.

J**Junction Object**

A custom object with two master-detail relationships. Using a custom junction object, you can model a “many-to-many” relationship between two objects. For example, you may have a custom object called “Bug” that relates to the standard case object such that a bug could be related to multiple cases and a case could also be related to multiple bugs.

K

No Glossary items for this entry.

L**Layout**

See Page Layout.

Length

Parameter for custom text fields that specifies the maximum number of characters (up to 255) that a user can enter in the field.

Parameter for number, currency, and percent fields that specifies the number of digits you can enter to the left of the decimal point, for example, 123.98 for an entry of 3.

Letterhead

Determines the basic attributes of an HTML email template. Users can create a letterhead that includes attributes like background color, logo, font size, and font color.

List View

A list display of items (for example, accounts or contacts) based on specific criteria. Salesforce provides some predefined views.

In the Console tab, the list view is the top frame that displays a list view of records based on specific criteria. The list views you can select to display in the console are the same list views defined on the tabs of other objects. You cannot create a list view within the console.

Locale

The country or geographic region in which the user is located. The setting affects the format of date and number fields, for example, dates in the

English (United States) locale display as 06/30/2000 and as 30/06/2000 in the English (United Kingdom) locale.

In Professional, Enterprise, Unlimited, and Developer Edition organizations, a user's individual `Locale` setting overrides the organization's `Default Locale` setting. In Personal and Group Editions, the organization-level locale field is called `Locale`, not `Default Locale`.

Long Text Area

Data type of custom field that allows entry of up to 32,000 characters on separate lines.

Lookup Dialog

Popup dialog available for some fields that allows you to search for a new item, such as a contact, account, or user.

Lookup Field

A type of field that contains a linkable value to another record. You can display lookup fields on page layouts where the object has a lookup or master-detail relationship with another object. For example, cases have a lookup relationship with assets that allows users to select an asset using a lookup dialog from the case edit page and click the name of the asset from the case detail page.

Lookup Relationship

A relationship between two records so you can associate records with each other. For example, cases have a lookup relationship with assets that lets you associate a particular asset with a case. On one side of the relationship, a lookup field allows users to click a lookup icon and select another record from a popup window. On the associated record, you can then display a related list to show all of the records that have been linked to it. A lookup relationship has no effect on record deletion or security, and the lookup field is not required in the page layout.

M

Manual Sharing

Record-level access rules that allow record owners to give read and edit permissions to other users who might not have access to the record any other way.

Many-to-Many Relationship

A relationship where each side of the relationship can have many children on the other side. Many-to-many relationships are implemented through the use of junction objects.

Master-Detail Relationship

A relationship between two different types of records that associates the records with each other. For example, accounts have a master-detail relationship with opportunities. This type of relationship affects record deletion, security, and makes the lookup relationship field required on the page layout.

Master Picklist

A complete list of picklist values available for a record type or business process.

Matrix Report

Matrix reports are similar to summary reports but allow you to group and summarize data by both rows and columns. They can be used as the source report for dashboard components. Use this type for comparing related totals, especially if you have large amounts of data to summarize and you need to compare values in several different fields, or you want to look at data by date *and* by product, person, or geography.

Merge Field

A field you can put in an email template, mail merge template, custom link, or formula to incorporate values from a record. For example, `Dear {!Contact.FirstName}`, uses a contact merge field to obtain the value of a contact record's `First Name` field to address an email recipient by his or her first name.

Metadata

Information about the structure, appearance, and functionality of an organization and any of its parts. Force.com uses XML to describe metadata.

Mini Page Layout

A subset of the items in a record's existing page layout that administrators choose to display in the Console tab's Mini View and in Hover Details. Mini page layouts inherit record type and profile associations, related lists, fields, and field access settings from the page layout.

Multitenancy

An application model where all users and apps share a single, common infrastructure and code base.

N**Notes**

Miscellaneous information pertaining to a specific record.

O

Object

An object allows you to store information in your Salesforce organization. The object is the overall definition of the type of information you are storing. For example, the case object allow you to store information regarding customer inquiries. For each object, your organization will have multiple records that store the information about specific instances of that type of data. For example, you might have a case record to store the information about Joe Smith's training inquiry and another case record to store the information about Mary Johnson's configuration issue.

Object-Level Help

Custom help text that you can provide for any custom object. It displays on custom object record home (overview), detail, and edit pages, as well as list views and related lists.

Object-Level Security

Settings that allow an administrator to hide whole tabs and objects from a user so that he or she does not know that type of data exists. On the platform you set object-level access rules with object permissions on user profiles.

One-to-Many Relationship

A relationship in which a single object is related to many other objects. For example, an account may have one or more related contacts.

Organization

A deployment of Salesforce with a defined set of licensed users. An organization is the virtual space provided to an individual customer of salesforce.com. Your organization includes all of your data and applications, and is separate from all other organizations.

Organization-Wide Defaults

Settings that allow you to specify the baseline level of data access that a user has in your organization. For example, you can make it so that any user can see any record of a particular object that is enabled in their user profile, but that they need extra permissions to edit one.

Outbound Message

An outbound message is a workflow, approval, or milestone action that sends the information you specify to an endpoint you designate, such as an external service. An outbound message sends the data in the specified fields in the form of a SOAP message to the endpoint. Outbound messaging is configured in the Salesforce setup menu. Then you must configure the external endpoint. You can create a listener for the messages using the Web services API.

Overlay

An overlay displays additional information when you hover your mouse over certain user interface elements. Depending on the overlay, it will close when you move your mouse away, click outside of the overlay, or click a close button.

Owner

Individual user to which a record (for example, a contact or case) is assigned.

P**Page Layout**

The organization of fields, custom links, and related lists on a record detail or edit page. Use page layouts primarily for organizing pages for your users. In Enterprise, Unlimited, and Developer Editions, use field-level security to restrict users' access to specific fields.

Picklist

Selection list of options available for specific fields in a Salesforce object, for example, the `Industry` field for accounts. Users can choose a single value from a list of options rather than make an entry directly in the field. See also Master Picklist.

Picklist Values

Selections displayed in drop-down lists for particular fields. Some values come predefined, and other values can be changed or defined by an administrator.

Platform as a Service (PaaS)

An environment where developers use programming tools offered by a service provider to create applications and deploy them in a cloud. The application is hosted as a service and provided to customers via the Internet. The PaaS vendor provides an API for creating and extending specialized applications. The PaaS vendor also takes responsibility for the daily maintenance, operation, and support of the deployed application and each customer's data. The service alleviates the need for programmers to install, configure, and maintain the applications on their own hardware, software, and related IT resources. Services can be delivered using the PaaS environment to any market segment.

Platform Edition

A Salesforce edition based on either Enterprise Edition or Unlimited Edition that does not include any of the standard Salesforce CRM apps, such as Sales or Service & Support.

Primary Key

A relational database concept. Each table in a relational database has a field in which the data value uniquely identifies the record. This field is called the primary key. The relationship is made between two tables by matching the values of the foreign key in one table with the values of the primary key in another.

Printable View

An option that displays a page in a print-ready format.

Private Sharing

Private sharing is the process of sharing an uploaded package by using the URL you receive from Salesforce. This URL is not listed in the AppExchange. Using the unlisted URL allows you to share a package without going through the listing process or making it public.

Process Visualizer

A tool that displays a graphical version of an approval process. The view-only diagram is presented as a flowchart. The diagram and an informational sidebar panel can help you visualize and understand the defined steps, rule criteria, and actions that comprise your approval process.

Production Organization

A Salesforce organization that has live users accessing data.

Professional Edition

A Salesforce edition designed for businesses who need full-featured CRM functionality.

Profile

Defines a user's permission to perform different functions within Salesforce. For example, the Solution Manager profile gives a user access to create, edit, and delete solutions.

Q**Queue**

A holding area for items before they are processed. Salesforce uses queues in a number of different features and technologies.

R**Read Only**

One of the standard profiles to which a user can be assigned. Read Only users can view and report on information based on their role in the organization. (That is, if the Read Only user is the CEO, they can view all data in the system. If the Read Only user has the role of Western Rep, they can view all data for their role and any role below them in the hierarchy.)

Recent Items

List of links in the sidebar for most recently accessed records. Note that not all types of records are listed in the recent items.

Record

A single instance of a Salesforce object. For example, “John Jones” might be the name of a contact record.

Record ID

See Salesforce Record ID.

Record-Level Security

A method of controlling data in which you can allow a particular user to view and edit an object, but then restrict the records that the user is allowed to see.

Record Name

A standard field on all Salesforce objects. Whenever a record name is displayed in a Force.com application, the value is represented as a link to a detail view of the record. A record name can be either free-form text or an autonumber field. *Record Name* does not have to be a unique value.

Record Type

A field available for certain records that can include some or all of the standard and custom picklist values for that record. Record types are special fields that you can associate with profiles to make only the included picklist values available to users with that profile.

Recycle Bin

A page that lets you view and restore deleted information. Access the Recycle Bin by using the link in the sidebar.

Related List

A section of a record or other detail page that lists items related to that record. For example, the Stage History related list of an opportunity or the Open Activities related list of a case.

Related List Hover Links

A type of link that allows you to quickly view information on a detail page about related lists, by hovering your mouse over the link. Your administrator must enable the display of hover links. The displayed text contains the corresponding related list and its number of records. You can also click this type of link to jump to the content of the related list without having to scroll down the page.

Related Object

Objects chosen by an administrator to display in the Console tab's mini view when records of a particular type are shown in the console's detail

view. For example, when a case is in the detail view, an administrator can choose to display an associated account, contact, or asset in the mini view.

Relationship

A connection between two objects, used to create related lists in page layouts and detail levels in reports. Matching values in a specified field in both objects are used to link related data; for example, if one object stores data about companies and another object stores data about people, a relationship allows you to find out which people work at the company.

Report

A *report* returns a set of records that meets certain criteria, and displays it in organized rows and columns. Report data can be filtered, grouped, and displayed graphically as a chart. See Tabular Report, Summary Report, and Matrix Report.

Report Type

A *report type* defines the set of records and fields available to a report based on the relationships between a primary object and its related objects. Reports display only records that meet the criteria defined in the report type. Salesforce provides a set of pre-defined standard report types; administrators can create custom report types as well.

Role Hierarchy

A record-level security setting that defines different levels of users such that users at higher levels can view and edit information owned by or shared with users beneath them in the role hierarchy, regardless of the organization-wide sharing model settings.

Roll-Up Summary Field

A field type that automatically provides aggregate values from child records in a master-detail relationship.

Running User

Each dashboard has a *running user*, whose security settings determine which data to display in a dashboard. If the running user is a specific user, all dashboard viewers see data based on the security settings of that user—regardless of their own personal security settings. For dynamic dashboards, you can set the running user to be the logged-in user, so that each user sees the dashboard according to his or her own access level.

S

SaaS

See Software as a Service (SaaS).

Salesforce Record ID

A unique 15- or 18-character alphanumeric string that identifies a single record in Salesforce.

Sandbox Organization

A nearly identical copy of a Salesforce production organization. You can create multiple sandboxes in separate environments for a variety of purposes, such as testing and training, without compromising the data and applications in your production environment.

Save As

Option on any standard, public, or custom report to save the parameters of the report without altering the original report. It creates a new custom report with your saved changes.

Save & New

Alternative “save” on most pages with which you can save your current changes and create a new entry.

Search

Feature that lets you search for information that matches specified keywords. If you have Sidebar Search, enter search terms in the Search section of the sidebar or click **Advanced Search...** for more search options. If you have Global Search, enter search terms in the search box in the header.

Search Layout

The organization of fields included in search results, in lookup dialogs, and in the key lists on tab home pages.

Setup

An administration area where you can customize and define Force.com applications. Access Setup through the **Your Name** ► **Setup** link at the top of Salesforce pages.

Sharing

Allowing other users to view or edit information you own. There are different ways to share data:

- **Sharing Model**—defines the default organization-wide access levels that users have to each other’s information and whether to use the hierarchies when determining access to data.
- **Role Hierarchy**—defines different levels of users such that users at higher levels can view and edit information owned by or shared with users beneath them in the role hierarchy, regardless of the organization-wide sharing model settings.
- **Sharing Rules**—allow an administrator to specify that all information created by users within a given group or role is automatically shared to the members of another group or role.
- **Manual Sharing**—allows individual users to share a specific account or opportunity with other users or groups.

- Apex-Managed Sharing—enables developers to programmatically manipulate sharing to support their application’s behavior. See Apex-Managed Sharing.

Sharing Model

Behavior defined by your administrator that determines default access by users to different types of records.

Sharing Rule

Type of default sharing created by administrators. Allows users in a specified group or role to have access to all information created by users within a given group or role.

Show/Hide Details

Option available for reports that lets you show/hide the details of individual column values in report results.

Sidebar

Column appearing on the left side of each page that provides links to recent items and other resources.

Sites

Force.com sites enables you to create public websites and applications that are directly integrated with your Salesforce organization—without requiring users to log in with a username and password.

SOAP (Simple Object Access Protocol)

A protocol that defines a uniform way of passing XML-encoded data.

Software as a Service (SaaS)

A delivery model where a software application is hosted as a service and provided to customers via the Internet. The SaaS vendor takes responsibility for the daily maintenance, operation, and support of the application and each customer's data. The service alleviates the need for customers to install, configure, and maintain applications with their own hardware, software, and related IT resources. Services can be delivered using the SaaS model to any market segment.

Source Report

A custom report scheduled to run and load data as records into a target object for an analytic snapshot.

Standard Object

A built-in object included with the Force.com platform. You can also build custom objects to store information that is unique to your app.

Summary Report

Summary reports are similar to tabular reports, but also allow users to group rows of data, view subtotals, and create charts. They can be used as the source report for dashboard components. Use this type for a report to

show subtotals based on the value of a particular field or when you want to create a hierarchical list, such as all opportunities for your team, subtotaled by *Stage* and *Owner*.

System Administrator

See *Administrator (System Administrator)*.

T**Tab**

A tab is an interface component that allows you to navigate around an app. A tab serves as the starting point for viewing, editing, and entering information for a particular object. When you click a tab at the top of the page, the corresponding tab home page for that object appears. A tab can be associated with an object, a Web page, or a Visualforce page.

Tabular Report

Tabular reports are the simplest and fastest way to look at data. Similar to a spreadsheet, they consist simply of an ordered set of fields in columns, with each matching record listed in a row. Tabular reports are best for creating lists of records or a list with a single grand total. They can't be used to create groups of data or charts, and can't be used in dashboards unless rows are limited. Examples include contact mailing lists and activity reports.

Task

Assigns a task to a user you specify. You can specify the *Subject*, *Status*, *Priority*, and *Due Date* of the task. Tasks are workflow and approval actions that are triggered by workflow rules or approval processes. For Calendar-related tasks, see *Activity (Calendar Events/Tasks)*.

Text

Data type of a custom field that allows entry of any combination of letters, numbers, or symbols, up to a maximum length of 255 characters.

Text Area

A custom field data type that allows entry of up to 255 characters on separate lines.

Text Area (Long)

See *Long Text Area*.

Time-Dependent Workflow Action

A workflow action that executes when the conditions of a workflow rule and an associated time trigger are met.

Time Trigger

An event that starts according to a specified time threshold, such as seven days before an opportunity close date. For example, you might define a

time-based workflow action that sends email to the account manager when a scheduled milestone will occur in seven days.

U

Unlimited Edition

Unlimited Edition is salesforce.com's flagship solution for maximizing CRM success and extending that success across the entire enterprise through the Force.com platform.

URL (Uniform Resource Locator)

The global address of a website, document, or other resource on the Internet. For example, <http://www.salesforce.com>.

User Interface

The layouts that specify how a data model should be displayed.

V

Validation Rule

A rule that prevents a record from being saved if it does not meet the standards that are specified.

Visualforce

A simple, tag-based markup language that allows developers to easily define custom pages and components for apps built on the platform. Each tag corresponds to a coarse or fine-grained component, such as a section of a page, a related list, or a field. The components can either be controlled by the same logic that is used in standard Salesforce pages, or developers can associate their own logic with a controller written in Apex.

Visualforce Controller

See Controller, Visualforce.

Visualforce Page

A web page created using Visualforce. Typically, Visualforce pages present information relevant to your organization, but they can also modify or capture data. They can be rendered in several ways, such as a PDF document or an email attachment, and can be associated with a CSS style.

W

Web Links

See Custom Links.

Web Service

A mechanism by which two applications can easily exchange data over the Internet, even if they run on different platforms, are written in different languages, or are geographically remote from each other.

Web Tab

A custom tab that allows your users to use external websites from within the application.

Wizard

A user interface that leads a user through a complex task in multiple steps.

Workflow and Approval Actions

Workflow and approval actions consist of email alerts, tasks, field updates, and outbound messages that can be triggered by a workflow rule or approval process.

Workflow Action

An email alert, field update, outbound message, or task that fires when the conditions of a workflow rule are met.

Workflow Email Alert

A workflow action that sends an email when a workflow rule is triggered. Unlike workflow tasks, which can only be assigned to application users, workflow alerts can be sent to any user or contact, as long as they have a valid email address.

Workflow Field Update

A workflow action that changes the value of a particular field on a record when a workflow rule is triggered.

Workflow Outbound Message

A workflow action that sends data to an external Web service, such as another cloud computing application. Outbound messages are used primarily with composite apps.

Workflow Queue

A list of workflow actions that are scheduled to fire based on workflow rules that have one or more time-dependent workflow actions.

Workflow Rule

A workflow rule sets workflow actions into motion when its designated conditions are met. You can configure workflow actions to execute immediately when a record meets the conditions in your workflow rule, or set time triggers that execute the workflow actions on a specific day.

Workflow Task

A workflow action that assigns a task to an application user when a workflow rule is triggered.

X**XML (Extensible Markup Language)**

A markup language that enables the sharing and transportation of structured data. All Force.com components that are retrieved or deployed through the Metadata API are represented by XML definitions.

Y

No Glossary items for this entry.

Z**Zip File**

A data compression and archive format.

A collection of files retrieved or deployed by the Metadata API. See also Local Project.

Index

__c suffix 36, 47

* search wildcard 116

&& function 73

|| function 73

A

About this book 2

Actions, approval

See Approval actions 235

Actions, workflow

See Workflow actions 201

Activating

approval processes 238

time-dependent workflow 217

workflow rules 208

Activities, enabling for custom objects 38

Administration

delegating 184

setup area 31

Administrative permissions

global 184

Advanced formula editor 65

Alerts, workflow

See Workflow email alerts 202

Amazon.com 2

Analytics

See Reports 244

AND() function 73

Apex

about 14, 285

API

__c suffix 36, 47

about 13, 283

documentation 320

field labels vs. names 47

labels vs. names, field 47

labels vs. names, object 36

name 63

object labels vs. names 36

App setup area 31

AppExchange 16

partner program 322

Approval actions

about 235

creating 236

See also Workflow actions 235

Approval History related list 231, 240

Approval processes 10, 224

actions 224

activating 238

approving and rejecting records 241

creating 227

delegate approvers 232

history information 230

Items to Approve related list 241

jump-start vs. standard setup 227

multiple approvers 234

page layouts 229

planning 226

record editability 228

Recruiting app 20

related users 234

Submit for Approval button 239

testing 238

visualizer 238

wireless devices 230

Approval steps

about 231

approval and rejection actions 233

creating 232

designating an approver 232

ordering 232

Approving records 241

Apps

about 31

basic elements 8

benefits 9

building iteratively 34

cloud computing 1

collaborative 10

creating 32

custom app wizard 32

data-centric 9

databases 23

debugging 34

default tab display 143

detail pages 8

Index

Apps (*continued*)

- distributing on AppExchange 16
 - edit pages 8
 - fields 26
 - forms 8
 - logos 33
 - metadata-driven 12
 - multitenant 11
 - navigation 8
 - objects 24
 - profiles and 33
 - setting a default 143
 - tabs 8, 33, 39
- Architecture, multitenant 11
- Areas, page layout field 78
- Assigning page layouts 179
- Attachments, enabling for custom objects 38
- Attributes, Visualforce 289
- Audience, book 3
- Auto-number data types 37
- Averages, report 259

B

- Blank spaces, page layouts 80
- Blog, Developer Force 321
- Boards, Developer Force discussion 321
- Business logic 200, 224
- Button layout, list view 102
- Buttons
 - adding to page layouts 304
 - creating custom list 302
 - Submit for Approval 239

C

- Candidate custom object 90
 - creating 90
- Candidate Map
 - adding to page layouts 292
 - code 291
 - creating 287
 - testing 294
- Case queues 211
- Channel, Force.com iTunes 322
- Charts
 - dashboard 269, 270
 - See Reports 244
- Chatter
 - about 192

Chatter (*continued*)

- adding elements to Visualforce pages 301
 - feed tracking 195
 - following 195
 - workbook 320
- Chatter Desktop 196
- Chatter mobile app 196
- Checkbox fields 49
- Child relationship 98
- Cleansing data 69
- Cloning profiles 142
- Cloud computing
 - apps 1
 - databases and 23
 - platforms 2
- Code
 - Apex
 - about 14
 - Force.com platform 281
- Code, Candidate Map 291
- Collaborative apps 10
- Colors, tab 40
- Combination chart 262
- Communication templates 219
- Company Dashboards folder 269
- Components, dashboard 269
- Concatenate 110
- Conditional highlighting 262
- Conditions, validation rule error 71
- Considerations, recruiting app 18
- Contents, book 3
- Context 295
- Contract Manager profile 140
- Controllers, Visualforce 289
- Controlling picklist fields 57
- Crawlers, Web 311
- Create New drop-down list 42
- Cross-object formula fields 107
- Currency fields 49
- Custom app wizard 32
- Custom email templates 220
- Custom formula fields
 - See Formula fields 62
- Custom objects 25
 - detail page 44
 - Grant access using hierarchies 164
 - queues 211
 - See also Objects 19, 35
- Custom tab wizard 38

D

- Dashboard tab 267
- Dashboards
 - about 264
 - adding a chart component 270
 - adding a gauge component 270
 - adding a metric component 272
 - adding a table component 271
 - adding to the Home tab 272
 - components 269
 - creating 267
 - Recruiting app 22
 - refreshing 273
 - running user 268
 - scheduling 273
 - security 268
- Data cleansing 69
- Data integrity 175
- Data types 45
 - auto-number 37
 - changing field 47
 - text 37
- Data values 26
 - restricting with page layouts 76
 - short cut for entering numbers 234
- Data-centric apps 9
- Data, importing 127
- Database concepts 28
- Databases 23
 - data values 26
 - definition 24
 - entities 24
 - fields 26
 - foreign keys 28
 - objects 24
 - primary keys 28
 - records 25
 - relational 26
 - tables 24
- Date fields 50
- Days Open field, calculating the 63
- Debugging apps 34
- Default app, setting a 143
- Default field values 68
- Default workflow user 217
- Defaults, organization-wide 157
- Delegated administration groups 186
 - defining 187
 - verifying 188
- Delegates, approval process 232

- Delegating administration
 - about 184
 - groups 186
 - groups, defining 187
 - groups, verifying 188
 - object-level permissions 185
- Demoing apps 16
- Dependency matrix, field 58
- Dependent lookup relationship fields
 - creating 89
- Dependent picklist fields 57
 - creating 58
- Deployment status 38
- Detail pages 8, 43
- Developer Edition 5, 320
- Developer Force 320
 - about 5
 - blog 321
 - discussion boards 321
 - events 321
 - wiki 320
- Developer resources 319
- Development mode, Visualforce 286
- Development, metadata-driven 12
- Directory, AppExchange 16
- Discussion boards, Developer Force 321
- Documentation, Force.com platform 320, 321
- Dreamforce event 320

E

- eBay 2, 12
- Edit pages 8, 43
- Editability, record 228
- Editor, advanced formula 65
- Education services, Force.com platform 321
- Email alerts, workflow
 - See Workflow email alerts 202
- Email templates 219
 - creating 219, 226
 - merge fields 219
 - naming 221
 - selecting merge field type 221
 - types 220
- Employment Website custom object 118
- Entities, database 24
- Entity relationship diagram, Recruiting app 125
- Error checking records 69
- Error conditions, validation rule 71

Index

- Events
 - enabling for custom objects 38
- Exercises, book 5
- External ID fields 91

F

- Favorite icons, site 311
- Feedback, sending book 5
- Feeds 311
- Field accessibility 151
 - defining field-level security in 154
- Field dependency matrix 58
- Field history tracking
 - enabling for custom objects 38
- Field updates, workflow
 - See Workflow field updates 202
- Field-level security 138, 149
 - defining in profiles 151
 - defining in the Field Accessibility area 154
 - vs. page layouts 150
- Fields
 - __c suffix 47
 - adding to lookup dialogs 102
 - adding to related lists 102
 - adding to related lists on objects without a tab 115
 - adding to search results 102
 - adding to tab home pages 102
 - advanced 54
 - changing data types 47
 - creating checkbox 49
 - creating currency 49
 - creating date 50
 - creating dependent lookup relationship 89
 - creating dependent picklist 58
 - creating formulas 64
 - creating lookup relationship 88
 - creating master-detail relationship 106
 - creating multilevel master-detail relationships 107
 - creating picklist 54
 - creating text 46
 - data type 45
 - default values 68
 - definition 26, 45
 - dependent picklist 57
 - external ID 91
 - foreign keys 28
 - formula 62

- Fields (*continued*)
 - hierarchical relationship 228
 - indexed 91
 - lookup relationship 87
 - master-detail relationship 87
 - merge 62, 219
 - Owner 42
 - page layouts 76
 - picklist, multi-select 54
 - picklist, standard 54
 - primary keys 28
 - properties 81
 - read-only 81
 - required 32, 81
 - restricting with page layouts 76
 - searchable 91
 - standard vs. custom 26
 - summary report 259
 - validation rules 69
- Filters
 - adding fields to search 102
 - lookup 90
 - setting report 255, 262
- Folders
 - Company Dashboards 269
 - report 248, 249
 - Unfiled Public Email Templates 221
- Force.com
 - cookbook 320
 - developer cheat sheets 321
 - workbook 320
- Force.com platform
 - domain name registering 307
 - about 15
 - API 13
 - app menu 34
 - AppExchange directory 16
 - checkout 31
 - coding on 281
 - collaborative apps 10
 - data-centric apps 9
 - databases 23
 - documentation 320
 - domain name 307
 - help and training 321
 - integrating with 283
 - introduction 7
 - Metadata API 13
 - metadata-driven development 12
 - mobile 14
 - multitenancy 11

- Force.com platform (*continued*)
 - objects 24
 - partners 322
 - podcasts 322
 - records 25
 - setup area 30
 - structured information 9
 - supporting technologies 10
 - videos 322
 - Force.com Platform Fundamentals book
 - audience 3
 - contents 3
 - following exercises in 5
 - online version 3
 - screenshots 4
 - sending feedback 5
 - Foreign keys 28, 118
 - Forms 8
 - Formula fields
 - about 62
 - advanced editor 65
 - checking syntax 67
 - creating 64
 - custom report summary 260
 - IF() function 64
 - ISBLANK() function 64
 - merge fields in 62
 - sample formulas 63
 - TODAY() function 63
 - Formulas
 - cross-object 107
 - default value 68
 - spanning 107
 - Functionality, recruiting app 18
 - Functions
 - && 73
 - || 73
 - AND() 73
 - HYPERLINK 109
 - IF() 64
 - ISBLANK() 64, 72
 - ISPICKVAL() 72
 - OR() 73
 - TODAY() 63
 - Grant Access Using Hierarchies checkbox 164
 - Graphs
 - See Reports 244
 - Grouping report records 253, 259
 - Groupings, page layout field 78
 - Groups, public 170
- ## H
- Help and training options 321
 - Hierarchical relationship fields 228
 - Hierarchies, role
 - See Role hierarchies 138
 - Highlighting, conditional
 - See Conditional highlighting 262
 - Hiring Manager profile 146
 - History tracking, field
 - enabling for custom objects 38
 - History, approval process 230
 - Home tab 33, 264
 - adding a dashboard 272
 - HTML email templates 220
 - HYPERLINK 109
 - Hyperlinks
 - See Links 8
- ## I
- Icons
 - lookup 88
 - site favorite 311
 - tab 40
 - ID fields, external 91
 - IF() function 64
 - Importing data 127
 - viewing the import queue 129
 - Indexed fields 91
 - Information, structured 9
 - Installing apps 16
 - Introductory splash pages 32
 - ISBLANK() function 64, 72
 - ISPICKVAL() function 72
 - Items to Approve related list 241
 - iTunes, Force.com platform podcasts on 322
- ## J
- JavaScript, with Visualforce 291
 - Job Application custom object 95
 - creating 96

Index

- Job applications
 - Mass Update Status page 297
 - Mass Update Status, about 295
- Job Posting custom object 117, 120
- Junction objects 117

K

- Keys
 - foreign 118
 - primary and foreign 28

L

- Labels vs. names
 - field 36, 47
 - object 36, 47
- Layout, list view button 102
- Layouts, page
 - See Page layouts 76
- Layouts, search
 - See Search layouts 100
- Lead queues 211
- Links 8
- List Buttons
 - creating custom 302
- List pages 32, 43
- List view button layout 102
- List view, role hierarchy 166
- List views 8
- Lists, related 44, 86
- Locked records 240
- Logic, business 200
- Logos, app 33
- Lookup dialogs
 - * search wildcard 116
 - adding fields to 102
- Lookup filters 90
- Lookup relationship fields 87
 - creating 88
 - icon 88

M

- Manager field 228
- Manual sharing 139, 174
 - defining 174
- Many-to-many relationships 117
- Map feature, Recruiting app
 - creating 287

- Marketing User profile 140
- Markup, Visualforce 284
 - using 288
- Mash-ups
 - about 2, 283
 - implementing 287
- Mass Update Status
 - about 295
 - page 297
 - planning 295
 - testing 305
- Master-detail relationship
 - primary 121
 - secondary 122
 - secondary, sharing implications 173
- Master-detail relationship fields 87
 - creating 106
 - creating multilevel 107
 - organization-wide defaults 162
- Matrix reports 247
- Matrix, field dependency 58
- Menu, Force.com platform
 - about 34
- Merge fields 62, 219
 - name 63
- Messages, workflow outbound
 - See Workflow outbound messages 202
- Metadata API 13
 - about 13
- Metadata-driven development 12
- Metrics, dashboard 269, 272
- Microsoft 9
- Mobile app, Chatter 196
- Mobile, Force.com platform 14
- Model, metadata-driven development 12
- Models, sharing
 - See Organization-wide defaults 157
- Monitoring the workflow queue 218
- Multi-select picklist fields 54
- Multiple users, supporting 10
- Multitenant architecture 11

N

- Names vs. labels
 - field 36, 47
 - object 36, 47
- Navigation 8, 39
- Notes, enabling for custom objects 38
- Number values, short cut for entering 234

O

- Object-level permissions 185
- Object-level security 138, 140
 - vs. field-level security 145
- Objects 24
 - __c suffix 36
 - about 35
 - adding checkbox fields 49
 - adding currency fields 49
 - adding date fields 50
 - adding dependent lookup fields 89
 - adding dependent picklist fields 58
 - adding formula fields 64
 - adding lookup fields 88
 - adding master-detail fields 106
 - adding multilevel master-detail fields 107
 - adding picklist fields 54
 - adding text fields 46
 - auto-numbered 37
 - creating 35
 - creating a tab 39
 - custom 19
 - deployment status 38
 - enabling activities 38
 - enabling field history tracking 38
 - enabling notes and attachments 38
 - enabling reports 38
 - junction 117
 - labels vs. names 37
 - page layouts 76
 - queues 211
 - related lists 44, 86
 - relationships 20, 27, 86, 117
 - reporting on 251
 - setting object-level permissions 145
 - standard User 88
 - standard vs. custom 25, 35
 - validation rules 69
 - workflow actions and 206
 - workflow and 204
- One-to-many relationships 117
- Online book version 3
- Operators
 - Concatenate 110
- OR() function 73
- Oracle 9
- Organization-wide defaults 138, 157
 - determining 158
 - master-detail relationship fields 162
 - setting 161

- Organizations 32
- Organizing fields on pages 76
- Other Reports category 251
- Outbound messages, workflow
 - See Workflow outbound messages 202
- Owner default field 42

P

- Page editor, Visualforce 286
- Page layouts 76
 - adding custom list buttons 304
 - adding Visualforce page 292
 - Approver 229
 - Assigning 179
 - blank spaces 80
 - Candidate Map 292
 - creating 178
 - Edit Layout link 94
 - editor 76
 - palette 77
 - quick save 81
 - Read-only fields 81
 - related list properties 115
 - Required fields 81
 - restricting field access with 76
 - saving 81
 - sections 78
 - vs. field-level security 150
- Pages
 - detail vs. edit 43
 - list 32, 43
 - page layout editor 76
 - Sharing Settings 161
 - splash 32
- Palette, page layout 77
- Partner program 322
- Permissions
 - administrative 184
 - object- vs. record-level 145
- Personal setup area 31
- Picklist fields 54
 - creating 54
 - creating dependent 58
 - dependent vs. controlling 57
- Planning, Mass Update Status feature 295
- Platform, Salesforce
 - See Force.com platform 7
- Platforms, cloud computing 2
- Podcasts 322

Index

- Position custom object 35
- Primary keys 28
- Process Visualizer 238
- Processes, approval
 - See Approval processes 224
- Profiles
 - about 140
 - apps and 33
 - cloning 142
 - creating 142
 - defining field-level security in 151
 - field-level security 138, 149
 - object-level security 138, 140
 - setting a default app 143
 - standard 140
 - tabs and 41
 - vs. roles 164
- Properties, field 81
- Public access
 - about 15
 - settings 316
- Public groups 170
- Public Jobs page 313

Q

- Questions to determine org-wide defaults 158
- Queues 210
 - about 211
 - creating 211
 - import 129
 - notifying members 211
 - time-dependent workflow 218
 - viewing 217

R

- Read Only profile 140
- Read-only fields 81
- Record ID 110
- Record type field 179
- Record types
 - about 176
 - creating 176
- Record-level security 138
 - about 157
 - vs. object-level security 145
- Records 25
 - approving and rejecting 241
 - editability in approval processes 228

- Records (*continued*)
 - grouping in a report 253, 259
 - locked 240
 - submitting for approval 239
 - validating before saving 69
 - viewing in queues 217
- Recruiter profile 142
- Recruiting app
 - calculating the Days Open field 63
 - Candidate Map 287
 - Candidate object 90
 - coding for 281
 - controlling data access 132
 - custom objects 19
 - custom profiles 141
 - design overview 19
 - Employment Website object 118
 - entity relationship diagram 125
 - Hiring Manager profile 146
 - importing sample data 127
 - introduction 17
 - Job Application object 95
 - Job Posting custom object 117
 - Job Posting object 120
 - objects 24
 - Position custom object 35
 - reports and dashboards 22
 - requirements 18
 - Review object 104
 - role hierarchy 163
 - security and sharing rules 20
 - Standard Employee profile 146
 - tabs 19
 - Visualforce 22
 - workflow and approvals 20
- Refreshing dashboards 273
- Registering a Force.com domain name 307
- Rejecting records 241
- Related lists 44, 86, 97
 - adding fields to 102
 - adding fields to objects without a tab 115
 - Approval History 231, 240
 - Items to Approve 241
 - properties 115
- Relational databases 26
- Relationship fields 87
- Relationship, child 98
- Relationships 20, 27
 - about 86
 - hierarchical 228
 - junction objects 117

- Relationships (*continued*)
 - lookup custom fields 87
 - many-to-many 117
 - master-detail custom fields 87
 - overview 87
 - Recruiting app entity relationship diagram 125
 - reporting on 251
 - Report builder
 - custom report 251
 - Report types, custom
 - about 274
 - creating 274
 - Reports
 - about 244
 - charts 256, 262
 - Combination chart 262
 - conditional highlighting 262
 - creating matrix 257
 - creating summary 251
 - dashboards 264
 - enabling for custom objects 38
 - filtering data 255, 262
 - folders 248, 249
 - formats 246
 - grouping records 253, 259
 - objects in 251
 - Other Reports category 251
 - Recruiting app 22
 - summary fields 259
 - tab 244, 248, 249
 - Required fields 32, 81
 - Requirements, recruiting app 18
 - Resources, developer 319
 - Restricting fields with page layouts 76
 - Review custom object 104
 - creating 105
 - Reviewing apps 16
 - Role hierarchies 138
 - about 163
 - defining 165
 - Recruiting app 163
 - views 166
 - vs. an org chart 163
 - vs. profiles 164
 - Roles
 - assigning to workflow tasks 207
 - Roll-up summary fields
 - about 112
 - creating 112
 - Rules, sharing
 - See Sharing rules 139
 - Rules, validation
 - See Validation rules 69
 - Rules, workflow
 - See Workflow rules 201
 - Running user, dashboard 268
- ## S
- Salesforce Mobile 14
 - salesforce.com
 - Training & Certification 6
 - Sample formulas 63
 - Sandbox 5
 - Scheduling dashboards 273
 - Screenshots, book 4
 - Search layouts 100
 - adding fields to 101
 - Searchable fields 91
 - Searching records in lookups 116
 - Sections, page layout 78
 - Security
 - criteria based sharing 173
 - Security and sharing 10
 - designing for your organization 137
 - field-level security 138, 149
 - manual sharing 139, 174
 - object-level security 138, 140
 - organization-wide defaults 138, 157
 - overview 138
 - profiles vs. roles 164
 - record types 176
 - record-level security 138, 157
 - Recruiting app 20
 - role hierarchies 138, 163
 - setting object-level permissions 145
 - sharing rules 139, 168
 - single sign-on 189
 - Sending book feedback 5
 - Services, Web
 - See Web services 283
 - Setup area 30
 - detail pages vs. edit pages 43
 - Field accessibility 151
 - Sharing
 - criteria-based 173
 - overriding 185
 - Sharing apps 16
 - Sharing models
 - See Organization-wide defaults 157

Index

- Sharing rules 139
 - about 168
 - creating 171
 - public groups 170
- Sharing settings page 161
- Sharing, manual
 - See Manual sharing 139
- Short cut for entering number values 234
- Site templates 310
- Sites
 - about 306
 - active site home page 316
 - creating 309
 - favorite icon 311
 - Force.com domain names, about 307
 - public access settings 316
 - Public Jobs page 313
 - registering a Force.com domain name 307
 - See Sites 306
 - testing 317
- Sites, Force.com platform 15
- Solution Manager profile 140
- Sorted list view, role hierarchy 166
- Spanning formula fields 107
- Specifications, recruiting app 18
- Splash pages 32
- Standard Employee profile 146
- Standard objects 25, 35
 - User 88
- Standard picklist fields 54
- Standard profiles 140
 - editing 141
- Standard User profile 140
- Status
 - Mass Update Status page 297
 - Mass Update Status, about 295
- Structured information 9
- Styles, tab 40
- Submit for Approval button 239
- Suffix, __c 36, 47
- Summary fields, report
 - about 259
 - creating custom 260
- Summary reports 246
- Sums, report 259
- Syndication feeds 311
- Syntax, checking formula 67
- System Administrator profile 140
 - permissions 184
- System Log Console 224

T

- Tab bar 31
- Tables
 - dashboard 269, 271
 - database 24
- Tabs
 - about 39
 - adding fields to tab home pages 102
 - appending to users' customizations 41
 - creating 39
 - Dashboard 267
 - display defaults 143
 - Home 33, 264
 - introduction 8
 - launching custom tab wizard 38
 - profiles and 41
 - recruiting app 19
 - Reports 244, 248, 249
 - setting default 249
 - style 40
- Tabular reports 246
- Tags, Visualforce 284
- Tasks
 - enabling for custom objects 38
- Tasks, workflow
 - See Workflow tasks 201
- Technologies, supporting 10
- Templates, email
 - See Email templates 219
- Templates, site 310
- Testing
 - Mass Update Status feature 305
- Text data types 37
- Text email templates 220
- Text fields 46
- Time-dependent workflow 206
 - about 215
 - activating 217
 - creating 215
 - default workflow user 217
 - queue 218
 - time triggers 216
- TODAY() function 63
- Tracking, field history
 - enabling for custom objects 38
- Training & Certification 6
- Training options 321
- Tree view, role hierarchy 166
- Triggers
 - time-dependent workflow 216

Triggers (*continued*)
 workflow rule 204

U

Unfiled Public Email Templates folder 221
 Universal Containers, about 18
 Updates, workflow field
 See Workflow field updates 202
 URL Rewriter 311
 Usability 175
 User interface design
 Visualforce 14
 User object
 Manager field 228
 User standard object 88
 User, default workflow 217
 Users
 dashboard running user 268
 defining 181
 hierarchical relationship fields 228
 supporting multiple 10

V

Validation rules 69
 && function 73
 || function 73
 AND() function 73
 creating 70
 error conditions 71
 ISBLANK() function 72
 ISPICKVAL() function 72
 OR() function 73
 sample 70
 testing 74
 Values, data 26
 Videos 322
 Viewing queue contents 217
 Views, defining 182
 Visualforce 14
 about 284
 adding to page layouts 292
 attributes 289
 controllers 289
 creating custom buttons 302
 creating pages 287
 dashboard 269
 development mode 286
 JavaScript 291

Visualforce (*continued*)
 markup 284
 markup, using 288
 page editor 286
 page editor, displaying 288
 Page tag 289
 Recruiting app 22
 workbook 320
 Visualforce email templates 220
 Visualforce pages
 Mass Update Status page 297
 Public Jobs page 313

W

Web crawlers 311
 Web services
 about 283
 Web services API
 See API 13
 Wiki, Developer Force 320
 Wildcard, * search 116
 Wireless devices, approval processes and 230
 Wizards
 approval process jump-start 227
 approval process standard setup 227
 custom app 32
 custom formula field 64
 custom tab 38, 39
 import 128
 new approval step 232
 workflow rule 203
 Workflow
 about 200
 Recruiting app 20
 Workflow actions 201
 objects and 206
 time triggers 216
 time-dependent 206, 215
 Workflow email alerts 202, 218
 creating 222
 Workflow field updates 202
 creating 210, 214, 237
 Workflow outbound messages 202
 Workflow queue, time-dependent 218
 Workflow rules 10
 about 201
 activating 208
 activating time-dependent 217
 creating 203, 214, 222

Index

Workflow rules (*continued*)
 default workflow user 217
 evaluation criteria 204
 objects and 204
Workflow tasks 201
 assigning to roles 207
 creating 205
 testing 209

Workflow tasks (*continued*)
 time-dependent 216

Y

Yahoo! 2, 12
Yahoo! Maps
 integrating with 283, 287

<http://developer.force.com>

salesforce.com®

ISBN: 978-0-9789639-3-4



9 780978 963934

5 2 9 9 9