CAUCHO

RESIN APPLICATION SERVER JAVA EE 6 WEB PROFILE

White paper
By Reza Rahman

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

CAUCHO

TABLE OF CONTENTS

Table of Contents **2**

Abstract **3**

Introduction **3**

Java EE 6 Web Profile **4**

Resin and Java EE **6**

Resin and Servlet Containers **7**

The Resin Java EE 6 Web Profile Implementation **8**

- Contexts and Dependency Injection 9
- Servlet 3 9
- EJB 3.1 Lite 10
- Unit Testing/Embedded Containers 11
- Hessian Remoting 12
- Seam 3 Support 13
- Spring Support 13

Summary **13**

About the Author **13**

About Caucho Technology **14**

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

ABSTRACT

Caucho Technology recently passed the Java EE 6 Web Profile Compatibility Test Kit (TCK) with the Resin 4 application server en route to official certification from Oracle. This whitepaper discusses the Java EE 6 Web Profile and how it fits with the development philosophy of Resin as well as the details of our implementation including Resin extensions to the Java EE 6 Web Profile.

The Java EE 6 Web Profile is composed of a core subset of Java EE APIs geared towards a majority of modern web applications. Resin's implementation is focused on providing high quality implementations of CDI, Servlet 3 and EJB 3.1 Lite. In addition to the Web Profile APIs like JSF 2, Servlet 3, CDI, EJB 3.1 Lite, JPA 2 and bean validation, Resin includes a lightweight JMS server, Hessian based remoting, a JTA transaction manager, database connection pooling, built-in authentication providers, security, clustering as well as an administration console.

CanDI, Caucho's independent implementation of the CDI standard for next generation dependency injection, serves as the foundation for Resin itself and enables many of the features that go beyond the Java EE 6 APIs. This includes powerful integration with popular third-party APIs like JUnit, Struts 2, Wicket, iBATIS, Quartz and Spring. CanDI also enables the use of all EJB annotations like @TransactionAttribute, @Schedule and @Asynchronous outside EJBs in POJO beans – a feature we hope to get standardized in Java EE 7.

INTRODUCTION

Resin enjoys a solid reputation for being one of the fastest, lightest and most stable application servers in the industry supporting some of the most high traffic web sites in the world including Salesforce, CNET and the Toronto stock exchange. The Java EE 6 Web Profile enables Caucho to create a truly lightweight standards-based runtime that focuses on ease-of-use for web application development. Indeed, Resin is the only major application server solely focused on the Web Profile.

The first section of this document covers the Java EE 6 Web Profile itself. The next section outlines our perspectives on Java EE as well as Servlet containers like Tomcat and Jetty, focusing on our vision for Resin 4. The final section covers our Java EE 6 Web Profile implementation including extended features and APIs.

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

JAVA EE 6 WEB PROFILE

Much like other mainstream development platforms like .NET, Java EE attempts to provide a comprehensive API set covering as many applications and use-cases as possible. As a result Java EE has had an ever-expanding set of APIs gradually added to it over time. The problem is that until the introduction of Profiles, Java EE has been a monolithic API set. This one-size-fits-all approach has meant that most applications do not use a large number of APIs in Java EE. For example, most web applications do not use remoting but do use the web tier APIs like Servlet, JSP and JSF. Similarly, SOA centric "headless" applications might use features like messaging, remoting, SOAP and REST but have no use for JSF or JSP. The heavyweight footprint of application servers that implement the entire Java EE API set is a symptom of this underlying problem.

Profiles in Java EE 6 are designed to address this problem by defining sub-sets of APIs geared towards particular types of applications. Following the previous examples, profiles geared towards web applications and SOA applications might make sense. In fact, the Web Profile is the first and only profile defined in Java EE 6 (the expectation is that other profiles will be defined going forward). The Web Profile is composed of a complete set of Java EE APIs that is needed for a majority of modern web applications. This includes APIs for the presentation tier, business tier, persistence tier, transactions, security, context management, dependency injection, cross-cutting logic, constraint management and testing. The following table outlines the specific APIs included in the Java EE 6 Web Profile and their intended purpose:

API	PURPOSE
JSF 2, Facelet, JSP, Servlet 3	Web tier
CDI, managed beans, interceptors	Dependency injection, context management, cross-cutting logic, events, extensibility, integration, testing
EJB 3.1 Lite	Business tier, declarative and programmatic transactions, declarative and programmatic security, testing
JTA	Transaction management
JPA 2	Persistence tier
Bean validation	Constraints management

Table 1. Java EE 6 Web Profile APIs

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

Like Profiles, EJB 3.1 Lite is a sub-set of the full EJB API. It includes stateless session beans, stateful session beans, singleton beans, declarative and programmatic transactions, declarative and programmatic security as well as an embedded container geared towards out-of-container testing. EJB 3.1 Lite does not have support for message driven beans, asynchronous processing, scheduling, REST (JAX-RS) end-points, SOAP (JAX-WS) end-points, RMI/CORBA based remoting as well as backwards compatibility requirements for EJB 2.x.

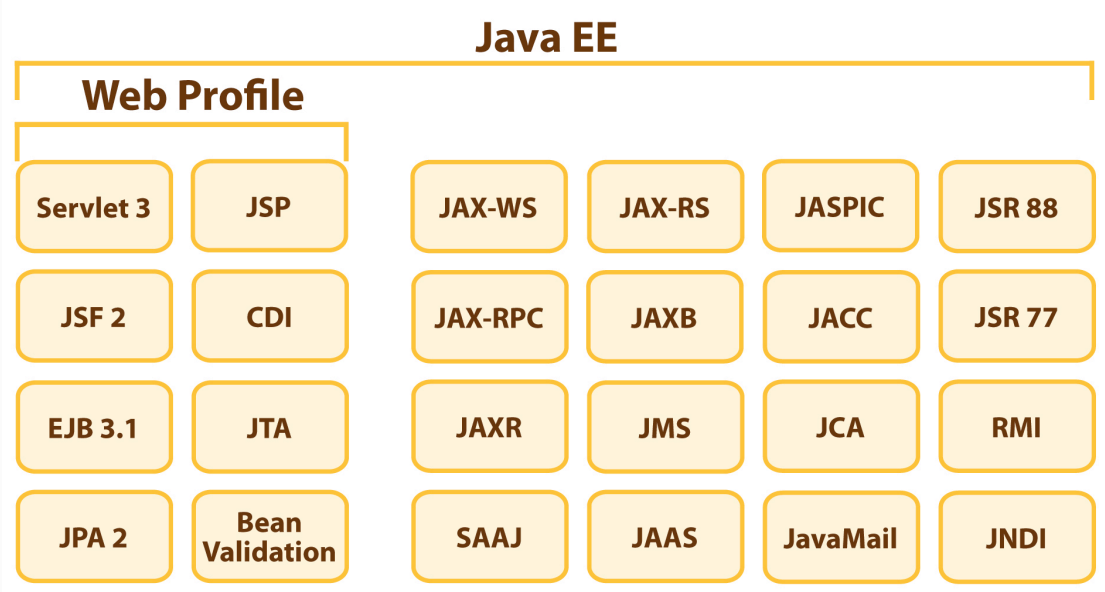

Figure 1. Java EE and the Web Profile

The Java EE 6 Web Profile leaves out a number of APIs that are not used often in web applications such as JAX-WS, JAX-RPC, JAXR, SAAJ, JAX-RS, JAXB, JMS, JAAS, JASPIC, JACC, JCA, JavaMail, the Java EE Management Specification (JSR 77) and the Java EE Deployment Specification (JSR 88). The Web Profile also only includes support for WAR files and not EAR files. Note Profiles do not stop a vendor from adding APIs and features as they see fit. As we will discuss shortly, we have chosen to add a very small number of Java EE APIs and features on top of the Web Profile. Specifically, we see value in adding support for scheduling, asynchronous processing, messaging, message driven beans and Hessian based remoting.

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

RESIN AND JAVA EE

The Resin team has always focused on delivering a lightweight, fast, reliable and easy-to-use application server. We have also always respected the value in standardization, developer choice, multilateral collaboration and having competing but compatible products. These ideas are clearly factors in the success of server-side Java that builds on the good foundations of a portable, scalable, type-safe programming language and leads to a lively industry ecosystem enriched by wide adoption, open source, innovation, disciplined development practices and enterprise credibility. However, before the Java EE 6 Web Profile, it was difficult to reconcile these concepts in Resin. The choices were really split between either creating a lightweight Java application server or aiming for full standards compliance. Resin has historically chosen the lightweight implementation route along the same lines as Tomcat and Jetty, with some important differences we will discuss shortly.

The problems with earlier versions of Java EE are now well understood. They included an over-emphasis on CORBA/remoting in EJB 1.x, the heavyweight programming paradigm in EJB 2.x, the flawed persistence model in EJB 2.x entity beans, serious problems in portability as well as the complexity in XML deployment descriptors. While Java EE 5 solved the issues around ease-of-use, we still saw a number of practical roadblocks in the way of creating a really lightweight implementation that Resin developers would find compelling.

Because of backwards compatibility requirements in EJB 3.0, it was still necessary to fully implement EJB 2.x and RMI/CORBA remoting. Due to the absence of profiles, we would have been forced to support bloated APIs like JAX-WS, JAX-RPC, JAXR, SAAJ, JACC, JAAS, JASPIC, JAXB, JCA, the Java EE Management Specification (JSR 77) and the Java EE Deployment Specification (JSR 88). As a result, we continued to primarily focus on the Servlet API instead of pursuing Java EE 5 compliance while still supporting the annotation-driven programming model in EJB 3/JPA as well as maintaining lightweight alternatives to RMI/CORBA such as Hessian.

With the Java EE 6 Web Profile, we are confident that we can deliver a fully standards compliant version of Resin that is really on the mark in terms of features and usability. We are excited in creating a very lightweight Java EE application server perhaps more compelling than any other server-side Java development option with a great "just-works-out-of-the-box" development experience.

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

RESIN AND SERVLET CONTAINERS

While we are firm believers in lightweight development, we feel that taking the extreme approach of simply implementing the Servlet specification and nothing else is the wrong set of trade-offs to make. While this minimalist option seems good in theory, it quickly leads to a lot of easily avoidable complexity for a majority of cases. The fundamental problem is that modern web applications use a lot more than the Servlet API. In fact, there are very few non-legacy applications that use the Servlet API directly. Most applications need a transaction manager, persistence mechanism and higher-level presentation layer API, not to mention dependency injection, security, scheduling and crosscutting logic. Some applications also need support for messaging as well as remoting via an efficient binary protocol like Hessian.

As a result, development shops that go the plain Servlet engine route end up configuring numerous third-party APIs on top of the minimal container to get these features. Even while using the most efficient integration solution, configuring and maintaining these third-party APIs become tasks in their own right, especially compounded by the number of applications that need to be configured in an ad-hoc fashion. The complexity involved in such configuration tasks have very little to do with solving business domain problems. In reality the configuration task is really the domain of an administrator or integrator working at the system level rather than a developer working at the application level. This complex development model is in stark contrast to platforms like Ruby on Rails which effectively promote ease-of-use and convention-over-configuration. Indeed this development model is likely sensible only in contrast to using a full-scale Java application server that is very heavy-weight or for systems that are really very specialized.

Besides development APIs, a vast majority of enterprise applications also need scalability features like resource/connection pools, security authentication providers, thread/process management, bandwidth throttling, caching, clustering, load-balancing and distributed transactions as well as management, monitoring and administration facilities. It is impossible or very difficult to add such features to a plain Servlet container as a third-party extension. Moreover, adding such foundational features on an a-la-carte basis takes away from the performance benefits to be achieved by having intelligent centralized coordination and resource sharing at the application server level.

Lastly, having a cohesive runtime that "just works" in a majority of cases allows us to create a simple vision covering all major application life-cycle stages including prototyping, development, testing, deployment, performance tuning, upgrades, maintenance and support, much like Ruby on Rails.

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

For these reasons, we believe a lightweight Resin implementation of the Java EE 6 Web Profile makes the right set of trade-offs by providing features that a majority of web applications need out-of-the-box, minimizing the configuration burden and focusing on ease-of-use.

THE RESIN JAVA EE 6 WEB PROFILE IMPLEMENTATION

The basic Resin strategy to supporting the Java EE 6 Web Profile is to provide Caucho implementations for core APIs and integrate best of breed pluggable open source implementations developed by other responsible, reputable organizations where it best makes sense. This strategy allows us to focus on delivering very high quality implementations where we can best add value to developers while not needlessly duplicating effort and leveraging the Java EE open source ecosystem in a sensible way.

In line with this philosophy, we are providing independent Caucho implementations for managed beans, interceptors, CDI, Servlet 3/JSP and EJB 3.1 Lite. Resin also includes its own high performance JTA compatible transaction manager, a very lightweight JMS implementation as well as the Hessian binary remoting protocol that offers better performance than RMI/CORBA and works over HTTP. Building on these core APIs, Caucho will integrate the reference implementations for JSF 2, Facelets, JPA 2 and bean validation. All of these implementations have been independently certified for Java EE 6 as part of the GlassFish application server.

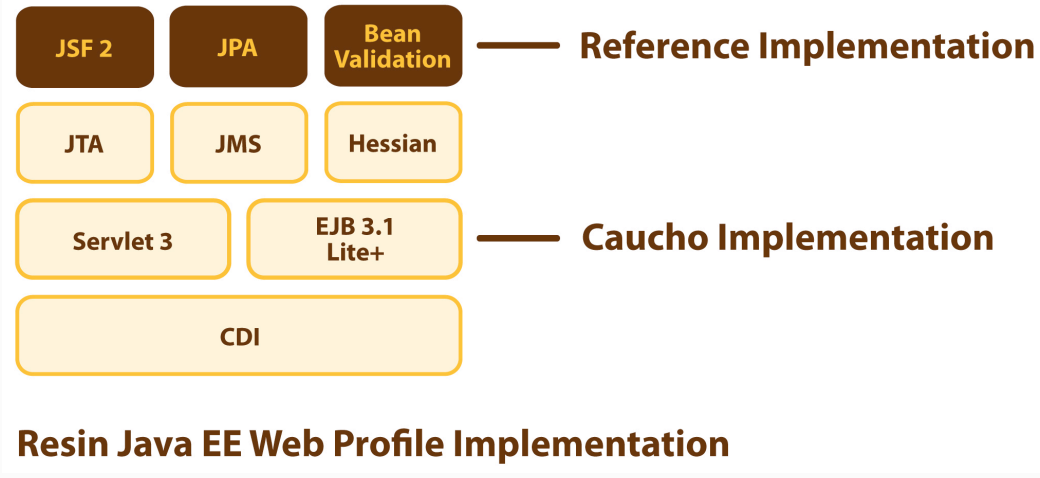

Figure2. Resin Java EE 6 Web Profile implementation strategy.

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

Contexts and Dependency Injection

The Contexts and Dependency Injection for Java EE (CDI) API is one of the key parts of the Web Profile. It provides robust context management including support for conversations, type-safe next-generation generic dependency injection, stereotypes, interceptors, decorators, lightweight events as well as a powerful SPI intended for building portable extensions. As active participants in JSR 299, the Resin team played an important part in providing visionary support for the CDI API. We are very proud to be offering a very high quality independent implementation of CDI. Our implementation, CanDI is one of the three major implementations of CDI, along with Apache's OpenWebBeans and Weld, the reference implementation from JBoss. CanDI is the centerpiece for the Caucho implementation of the Java EE 6 Web Profile. Indeed, many parts of the Resin application server itself have been written using CanDI.

A number of additions to CDI are included in CanDI. For example, we support a custom @TransactionScoped/@ThreadScoped in addition to the standard @ApplicationScoped, @SessionScoped, @RequestScoped and @ConversationScoped. We also plan to provide portable extensions for integrating popular open source and standard tools that developers will find useful. The possibilities include integration/ease-of-use CDI portable extensions for JMS, JDBC, JavaMail, Struts 2, iBATIS and Quartz. We will also fully support all portable extensions developed by the Apache and JBoss CDI projects on CanDI/Resin.

Servlet 3

Servlet 3 brings a major overhaul to this foundational Java EE API. The changes include full support for annotations, pluggable web.xml fragments, programmatic addition of Servlets, Filters and Listeners at runtime as well as asynchronous processing. While annotation support is geared towards ease-of-use and the asynchronous processing capabilities add better support for emerging paradigms like the real-time web, the rest of the changes are critical in improving plug-ability for the rich set of third-party tools and frameworks that build on Servlets such as JSF, Facelets, Wicket, Struts 2 and the like. Taken as a whole, these changes will likely enable stronger innovation in the web tier based on the new capabilities in Servlet 3. We have long focused on excellent support for the Servlet API and will continue to do so with Servlet 3. In fact, it is one of our first APIs to pass the Compatibility Test Kit (TCK).

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

EJB 3.1 Lite

EJB 3.1 has both further ease-of-use elements as well as core features frequently requested by developers. The changes include session bean optional interfaces, singleton beans with concurrency control, cron-style declarative and programmatic timers, asynchronous bean invocation, support for packaging EJBs in WARs, embedded container support for unit testing, standardized global JNDI naming as well as the definition of EJB Lite. In addition to basic support for EJB Lite, Resin includes support for EJB scheduling (@Schedule, @Timeout), asynchronous processing (@Asynchronous), message driven beans (@MessageDriven) and Hessian based remoting (@Remote).

In our view, one of the most important changes in EJB 3.1 is the redefinition of EJBs as simple managed bean POJOs with additional services. We have taken this realignment to the logical next step by allowing developers to use EJB annotations in CanDI managed beans in addition to EJBs including the @TransactionAttribute, @Schedule, @Asynchronous, @RolesAllowed, @PermitAll, @DenyAll, @RunAs, @Lock , @Startup and @Remote. Our hope is that this capability will be standardized in Java EE 7. Below are two examples of this capability:

```
@ApplicationScoped
@TransactionAttribute(REQUIRED)
public class BidDao {
  // Resin thread-safe EntityManager proxy.
  @PersistenceContext
  private EntityManager entityManager;

  public void addBid (Bid bid) {
    entityManager.persist(bid);
  }
}
```

Figure 2. EJB Lite sample code 1.

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

```
@Startup
@ApplicationScoped
public class NewsLetterGenerator {

  @Schedule(dayOfMonth="L", month="*") // Last day of every month.
  public void generateMonthlyNewsLetter() {
    // Code to generate the monthly news letter goes here.
  }
}
```

Figure 3. EJB Lite sample code 2.

Unit Testing/Embedded Containers

Resin provides excellent out-of-container unit/integration testing support for JUnit 4. Testing support will be based on the Resin embedded container built around CanDI, managed beans and EJB 3.1 Lite. Using JUnit bootstrap mechanisms, the Resin test tools will inject components under test directly into testing artifacts that can be run from the command-line or IDE. The following code example shows these capabilities:

```
@RunWith(ResinBeanContainerRunner.class)
// Deployment overrides for the test.
@ResinBeanConfiguration(beansXml="beans-test.xml")
public class AccountServiceTest {

  @Inject
  private AccountService accountService;

  @Test
  public void testGetAccount() throws Exception {
    Account account = accountService.getAccount(1007);
    assertNotNull(account);
  }
}
```

Figure 4. Unit/integration testing with the Resin embedded container

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

While the Resin bean container supports CDI, managed beans, EJB and JPA, it does not start the Servlet container for performance reasons. If you need support for JUnit based testing for Servlets, JSP or JSF, it is possible to use full resin embedded version with JUnit as well – the configuration is very similar to the example above with the ResinBeanContainerRunner being replaced by ResinEmbedRunner. In addition, Resin will also support JBoss Arquillian for testing.

Hessian Remoting

Hessian is an HTTP based binary communication protocol developed by Caucho. Hessian offers better performance than RMI/CORBA, SOAP or REST. Since it is HTTP based, it can work across firewalls much like web services. Because of these characteristics, we believe it is the ideal communication protocol for remoting using Resin and Java EE 6. Resin will support Hessian remoting through the EJB @Remote annotation as in the example below:

```
@Remote
public interface BidService {
  public void addBid(Bid bid);
}

@ApplicationScoped
public class DefaultBidService implements BidService {
  ...
}
```

Figure 5. Hessian Remoting example.

In addition to Hessian based remoting Resin also fully supports running Jersey, RESTEasy, Metro or Apache CXF as third-party JAX-RS and JAX-WS web services frameworks.

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

Seam 3 Support

Seam 3 by JBoss is one of the most promising CDI based projects. Seam 3 includes CDI portable extensions for XML configuration, JPA enhancements, JSF enhancements, Servlet enhancements, JMS enhancements, REST enhancements, JavaScript remoting, declarative exception handling, internationalization/localization, security, mail, cron, document generation, GWT, Drools, jBPM, JBoss ESB and many others. With a strong focus on performance and stability CanDI and Resin 4 are excellent platforms for running Seam 3.

Spring Support

A large number of Resin customers currently use the popular Spring framework. Caucho has always provided and will continue to provide outstanding runtime support for the framework. Indeed, we believe great opportunities for even better Spring framework integration exist particularly via the common Dependency Injection for Java (JSR 330) annotations shared by both CDI and Spring IoC. We will explore such innovative integration possibilities going forward.

Summary

Along with GlassFish and JBoss, Resin provides an excellent early implementation for Java EE 6. Unlike other major application servers, Resin will not have an offering that implements the full Java EE platform that we feel is heavyweight for the requirements of most server-side Java applications. Instead, we will maintain our traditionally lightweight development philosophy and focus on the Web Profile while still providing the ease-of-use, ease-of-configuration, ease-of-administration, usability, scalability and performance that is often compromised with commodity Servlet containers. Visionary implementations like CanDI enables us to go beyond the Java EE standard and fully leverage the Java open source ecosystem as part of the evolution of the Resin developer community.

About the Author

Reza Rahman is a Caucho engineer focusing on Resin.

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com

Reza has over a decade of experience with consulting, enterprise architecture, technological leadership and application development. He has been working with Java EE since its inception, developing on almost every major application platform ranging from Tomcat to JBoss, GlassFish, WebSphere and WebLogic. He has developed enterprise systems for companies like Motorola, Comcast, Nokia, Guardian Life, Prudential, Independence Blue Cross, Citigroup, Accenture and GMAC using EJB 2, Spring, EJB 3 and Seam. He is particularly interested in distributed systems, messaging, middleware, persistence and machine learning.

Reza is the author of EJB 3 in Action <http://www.ejb3inaction.com> from Manning Publishing. He is a frequent speaker at conferences and Java user groups including JavaOne and TSSJS. Reza is an independent member of the Java EE 6 and EJB 3.1 expert groups.

About Caucho Technology

Caucho's relentless quest for performance and reliability paved the way for Resin® to be a leading global Open Source Java application server since 1998. Our engineers' dedication to the development, support and evolution of the Resin Java EE 6 Web Profile continues to uphold our reputation for quality, performance and manageability. We've helped organizations worldwide including start-ups, governments and Fortune 500 companies build and grow their business with one of the most flexible, rock-solid and powerful application servers, Resin.

About Resin Application Server

Delivering the most lightweight and scalable software is just the beginning. By employing agile development with rigorous planning and testing, Caucho continuously aligns our product delivery to meet our customers' needs. Our focus is on the advancement of the Resin Java EE 6 Web Profile and beyond to provide developers with a highly functional yet lightweight tool: an application server with built-in Caching, Cloud Computing Support, Messaging and Clustering. Resin is a fast and clear-cut Web Profile application server designed for deployment and management of both web-tier and service oriented applications. From WebSocket to SOA, Resin provides current and next generation enterprise features.

Caucho Technology, Inc.
PO Box 9001
La Jolla, CA 92038 USA
Tel: 858.456.0300 / Int'l +1.858.456.0300
Fax: 858.777.3636
Email: sales@caucho.com
Web: http://caucho.com