(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 113127361 A
(43)申请公布日 2021.07.16

(21)申请号 202110445286.4

(22)申请日 2021.04.23

(71)申请人 中国工商银行股份有限公司
地址 100140 北京市西城区复兴门内大街55号

(72)发明人 赵海强 罗涛 高见

(74)专利代理机构 中科专利商标代理有限责任公司 11021
代理人 周天宇

(51)Int.Cl.
G06F 11/36(2006.01)

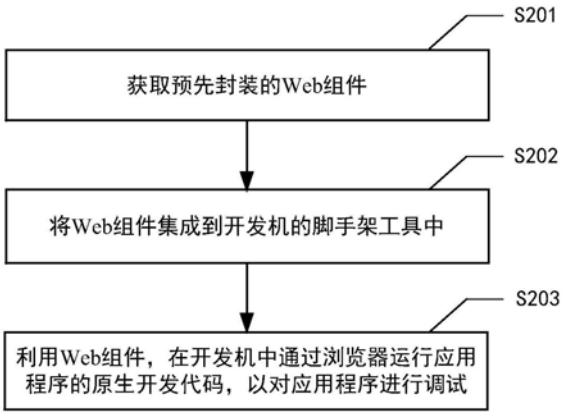
权利要求书2页 说明书10页 附图4页

(54)发明名称

应用程序的开发方法、装置、电子设备和存储介质

(57)摘要

本公开提供了一种应用程序的开发方法，包括：获取预先封装的Web组件，Web组件提供原生开发代码在浏览器中运行的环境；将Web组件集成到开发机的脚手架工具中；利用Web组件，在开发机中通过浏览器运行应用程序的原生开发代码，以对应用程序进行调试。本公开还提供了一种应用程序的开发装置、电子设备以及计算机可读存储介质。



CN 113127361 A

1.一种应用程序的开发方法,包括:

获取预先封装的Web组件,所述Web组件提供原生开发代码在浏览器中运行的环境;

将所述Web组件集成到开发机的脚手架工具中;

利用所述Web组件,在所述开发机中通过浏览器运行应用程序的原生开发代码,以对应用程序进行调试。

2.根据权利要求1所述的方法,其中,所述将所述Web组件集成到开发机的脚手架工具中,包括:

获取所述Web组件的配置文件;

基于所述配置文件,在所述开发机的React Native开发框架中集成Web依赖环境、调试环境、cli命令、Web端入口文件和Web端打包文件。

3.根据权利要求2所述的方法,其中,所述将所述Web组件集成到开发机的脚手架工具中,还包括:

将所述开发机的React Native开发框架的原生调试接口转换成Web端的调试接口,所述调试接口包括界面调试接口和页面样式调试接口。

4.根据权利要求2所述的方法,其中,基于所述配置文件,在所述开发机中集成Web依赖环境,包括:

对所述配置文件过滤,得到目标配置文件;

将所述目标配置文件转换成与Web端适配的文件格式;

在所述开发机中启动转换文件格式后的目标配置文件,以建立Web依赖环境。

5.根据权利要求2所述的方法,其中,基于所述配置文件,在所述开发机中集成Web端入口文件,包括:

基于所述配置文件,构建运行容器;

在所述运行容器中构建注册入口,以通过所述注册入口运行业务项目。

6.根据权利要求2所述的方法,其中,基于所述配置文件,在所述开发机中集成Web端打包文件,包括将所述开发机中原始的打包文件转换为Web端打包文件。

7.一种应用程序的开发装置,包括:

获取模块,用于获取预先封装的Web组件,所述Web组件提供原生开发代码在浏览器中运行的环境;

集成模块,用于将所述Web组件集成到开发机的脚手架工具中;

运行模块,用于利用所述Web组件,在所述开发机中通过浏览器运行应用程序的原生开发代码,以对应用程序进行调试。

8.根据权利要求7所述的装置,其中,所述集成模块包括:

获取单元,用于获取所述Web组件的配置文件;

集成单元,用于基于所述配置文件,在所述开发机的React Native开发框架中集成Web依赖环境、调试环境、cli命令、Web端入口文件、Web端打包文件。

9.根据权利要求7或8所述的装置,其中,所述集成模块,还包括:

第一转换单元,用于在所述开发机的React Native开发框架的原生调试接口转换成Web端的调试接口,所述调试接口包括界面调试接口和页面样式调试接口。

10.根据权利要求8所述的方法,其中,所述集成单元,包括:

过滤单元,用于对所述配置文件过滤,得到目标配置文件;

第二转换单元,用于将所述目标配置文件转换成与Web端适配的文件格式;

启动单元,用于在所述开发机中启动转换文件格式后的目标配置文件,以建立Web依赖环境。

11.根据权利要求8所述的方法,其中,所述集成单元,包括:

第一构建单元,用于基于所述配置文件,构建运行容器;

第二构建单元,用于在所述运行容器中构建注册入口,以通过所述注册入口运行业务项目。

12.根据权利要求8所述的方法,其中,所述集成单元,包括:

第三转换单元,用于将所述开发机中原始的打包文件转换为Web端打包文件。

13.一种电子设备,包括:

一个或多个处理器;

存储器,用于存储一个或多个程序,

其中,当所述一个或多个程序被所述一个或多个处理器执行时,使得所述一个或多个处理器实现权利要求1至6中任一项所述的方法。

14.一种计算机可读存储介质,其上存储有可执行指令,该指令被处理器执行时使处理器实现权利要求1至6中任一项所述的方法。

应用程序的开发方法、装置、电子设备和存储介质

技术领域

[0001] 本公开涉及计算机技术领域，更具体地，涉及一种应用程序的开发方法、装置、电子设备和存储介质。

背景技术

[0002] React Native是目前常见的跨端研发框架。其上层编写代码使用JavaScript语言，底层采用安卓、iOS原生渲染的方式，实现了一次代码编写，多端运行，且底层采用原生渲染的方式保证了页面的渲染效率及性能体验。

[0003] 在实现本公开构思的过程中，发明人发现相关技术中至少存在如下问题：在React Native中采用原生渲染，需要采用模拟器或者真机(安卓端和iOS端的真机)进行页面预览。采用模拟器预览时，模拟机占用内存较多，很容易造成开发机的性能不足；采用真机预览时，涉及的设备较多，调试和开发过程中需要频繁地切换设备，影响研发体验。

发明内容

[0004] 有鉴于此，本公开提供了一种应用程序的开发方法和装置。

[0005] 本公开的一个方面提供了一种应用程序的开发方法，包括：获取预先封装的Web组件，所述Web组件提供原生开发代码在浏览器中运行的环境；将所述Web组件集成到开发机的脚手架工具中；利用所述Web组件，在所述开发机中通过浏览器运行应用程序的原生开发代码，以对应用程序进行调试。

[0006] 根据本公开的实施例，所述将所述Web组件集成到开发机的脚手架工具中，包括：获取所述Web组件的配置文件；基于所述配置文件，在所述开发机的React Native开发框架中集成Web依赖环境、调试环境、cli命令、Web端入口文件和Web端打包文件。

[0007] 根据本公开的实施例，所述将所述Web组件集成到开发机的脚手架工具中，还包括：将所述开发机的React Native开发框架的原生调试接口转换成Web端的调试接口，所述调试接口包括界面调试接口和页面样式调试接口。

[0008] 根据本公开的实施例，基于所述配置文件，在所述开发机中集成Web依赖环境，包括：对所述配置文件过滤，得到目标配置文件；将所述目标配置文件转换成与Web端适配的文件格式；在所述开发机中启动转换文件格式后的目标配置文件，以建立Web依赖环境。

[0009] 根据本公开的实施例，基于所述配置文件，在所述开发机中集成Web端入口文件，包括：基于所述配置文件，构建运行容器；在所述运行容器中构建注册入口，以通过所述注册入口运行业务项目。

[0010] 根据本公开的实施例，基于所述配置文件，在所述开发机中集成Web端打包文件，包括将所述开发机中原始的打包文件转换为Web端打包文件。

[0011] 本公开的另一个方面提供了一种应用程序的开发装置，包括：获取模块，用于获取预先封装的Web组件，所述Web组件提供原生开发代码在浏览器中运行的环境；集成模块，用于将所述Web组件集成到开发机的脚手架工具中；运行模块，用于利用所述Web组件，在所述

开发机中通过浏览器运行应用程序的原生开发代码,以对应用程序进行调试。

[0012] 根据本公开的实施例,所述集成模块包括:获取单元,用于获取所述Web组件的配置文件;集成单元,用于基于所述配置文件,在所述开发机的React Native开发框架中集成Web依赖环境、调试环境、cli命令、Web端入口文件、Web端打包文件。

[0013] 根据本公开的实施例,所述集成模块,还包括:第一转换单元,用于在所述开发机的React Native开发框架的原生调试接口转换成Web端的调试接口,所述调试接口包括界面调试接口和页面样式调试接口。

[0014] 根据本公开的实施例,所述集成单元,包括:过滤单元,用于对所述配置文件过滤,得到目标配置文件;第二转换单元,用于将所述目标配置文件转换成与Web端适配的文件格式;启动单元,用于在所述开发机中启动转换文件格式后的目标配置文件,以建立Web依赖环境。

[0015] 根据本公开的实施例,所述集成单元,包括:第一构建单元,用于基于所述配置文件,构建运行容器;第二构建单元,用于在所述运行容器中构建注册入口,以通过所述注册入口运行业务项目。

[0016] 根据本公开的实施例,所述集成单元,包括:第三转换单元,用于将所述开发机中原始的打包文件转换为Web端打包文件。

[0017] 本公开的另一方面提供了一种电子设备,所述电子设备包括一个或多个处理器;以及存储器,用于存储一个或多个程序,其中,当所述一个或多个程序被所述一个或多个处理器执行时,使得所述一个或多个处理器实现权利要求上述任一项所述的方法。

[0018] 本公开的另一方面提供了一种计算机可读存储介质,存储有计算机可执行指令,所述指令在被执行时用于实现如上所述的方法。

[0019] 根据本公开的实施例,因为采用了在开发机的原生开发平台中集成一个Web端开发环境的技术手段,所以至少部分地克服了在开发机中无法直接运行预览和调试应用程序的技术问题,进而达到了简化开发过程,提高开发效率技术效果。

附图说明

[0020] 通过以下参照附图对本公开实施例的描述,本公开的上述以及其他目的、特征和优点将更为清楚,在附图中:

[0021] 图1示意性示出了可以应用本公开的用于应用程序的开发方法的方法和装置的示例性系统架构;

[0022] 图2示意性示出了根据本公开实施例的应用程序的开发方法的流程图;

[0023] 图3A示意性示出了根据本公开另一实施例的应用程序的开发方法的流程图;

[0024] 图3B示意性示出了根据本公开另一实施例的应用程序的开发方法的流程图;

[0025] 图4示意性示出了根据本公开实施例的应用程序的开发装置的框图;

[0026] 图5A示意性示出了根据本公开另一实施例的应用程序的开发装置的框图;

[0027] 图5B示意性示出了根据本公开另一实施例的应用程序的开发装置的框图;以及

[0028] 图6示意性示出了根据本公开实施例的适于实现应用程序的开发装置的电子设备的框图。

具体实施方式

[0029] 以下，将参照附图来描述本公开的实施例。但是应该理解，这些描述只是示例性的，而并非要限制本公开的范围。在下面的详细描述中，为便于解释，阐述了许多具体的细节以提供对本公开实施例的全面理解。然而，明显地，一个或多个实施例在没有这些具体细节的情况下也可以被实施。此外，在以下说明中，省略了对公知结构和技术的描述，以避免不必要地混淆本公开的概念。

[0030] 在此使用的术语仅仅是为了描述具体实施例，而并非意在限制本公开。在此使用的术语“包括”、“包含”等表明了所述特征、步骤、操作和/或部件的存在，但是并不排除存在或添加一个或多个其他特征、步骤、操作或部件。

[0031] 在此使用的所有术语(包括技术和科学术语)具有本领域技术人员通常所理解的含义，除非另外定义。应注意，这里使用的术语应解释为具有与本说明书的上下文相一致的含义，而不应以理想化或过于刻板的方式来解释。

[0032] 在使用类似于“A、B和C等中至少一个”这样的表述的情况下，一般来说应该按照本领域技术人员通常理解该表述的含义来予以解释(例如，“具有A、B和C中至少一个的系统”应包括但不限于单独具有A、单独具有B、单独具有C、具有A和B、具有A和C、具有B和C、和/或具有A、B、C的系统等)。在使用类似于“A、B或C等中至少一个”这样的表述的情况下，一般来说应该按照本领域技术人员通常理解该表述的含义来予以解释(例如，“具有A、B或C中至少一个的系统”应包括但不限于单独具有A、单独具有B、单独具有C、具有A和B、具有A和C、具有B和C、和/或具有A、B、C的系统等)。

[0033] 本公开的实施例提供了一种应用程序的开发方法和装置。该方法包括获取预先封装的Web组件，Web组件提供原生开发代码在浏览器中运行的环境。将Web组件集成到开发机的脚手架工具中，使得开发机中具有Web运行环境。进一步在开发机中通过浏览器运行应用程序的原生开发代码，以对应用程序进行调试。

[0034] 图1示意性示出了根据本公开实施例的可以应用应用程序的开发方法和装置的示例性系统架构100。需要注意的是，图1所示仅为可以应用本公开实施例的系统架构的示例，以帮助本领域技术人员理解本公开的技术内容，但并不意味着本公开实施例不可以用于其他设备、系统、环境或场景。

[0035] 如图1所示，根据该实施例的系统架构100可以包括终端设备101、102、103，网络104和服务器105。网络104用以在终端设备101、102、103和服务器105之间提供通信链路的介质。网络104可以包括各种连接类型，例如有线和/或无线通信链路等等。

[0036] 用户可以使用终端设备101、102、103通过网络104与服务器105交互，以接收或发送消息等。终端设备101、102、103上可以安装有各种通讯客户端应用，例如购物类应用、网页浏览器应用、搜索类应用、即时通信工具、邮箱客户端和/或社交平台软件等(仅为示例)。

[0037] 终端设备101、102、103可以是具有显示屏并且支持网页浏览的各种电子设备，包括但不限于智能手机、平板电脑、膝上型便携计算机和台式计算机等等。

[0038] 服务器105可以是提供各种服务的服务器，例如对用户利用终端设备101、102、103所浏览的网站提供支持的后台管理服务器(仅为示例)。后台管理服务器可以对接收到的用户请求等数据进行分析等处理，并将处理结果(例如根据用户请求获取或生成的网页、信息或数据等)反馈给终端设备。

[0039] 需要说明的是，本公开实施例所提供的应用程序的开发方法一般可以由服务器105执行。相应地，本公开实施例所提供的应用程序的开发装置一般可以设置于服务器105中。本公开实施例所提供的应用程序的开发方法也可以由不同于服务器105且能够与终端设备101、102、103和/或服务器105通信的服务器或服务器集群执行。相应地，本公开实施例所提供的应用程序的开发装置也可以设置于不同于服务器105且能够与终端设备101、102、103和/或服务器105通信的服务器或服务器集群中。或者，本公开实施例所提供的应用程序的开发装置也可以由终端设备101、102或103执行，或者也可以由不同于终端设备101、102或103的其他终端设备执行。相应地，本公开实施例所提供的应用程序的开发装置也可以设置于终端设备101、102或103中，或设置于不同于终端设备101、102或103的其他终端设备中。

[0040] 例如，Web组件可以原本存储在终端设备101、102或103中的任意一个(例如，终端设备101，但不限于此)之中，或者存储在外部存储设备上并可以导入到终端设备101中。然后，终端设备101可以在本地执行本公开实施例所提供的应用程序的开发方法，或者将Web组件发送到其他终端设备、服务器或服务器集群，并由接收该Web组件的其他终端设备、服务器或服务器集群来执行本公开实施例所提供的应用程序的开发方法。

[0041] 应该理解，图1中的终端设备、网络和服务器的数目仅仅是示意性的。根据实现需要，可以具有任意数目的终端设备、网络和服务器。

[0042] 图2示意性示出了根据本公开实施例的应用程序的开发方法的流程图。

[0043] 如图2所示，该方法包括操作S201～S203。

[0044] 在操作S201，获取预先封装的Web组件。其中，Web组件提供原生开发代码在浏览器中运行的环境。

[0045] 在操作S202，将Web组件集成到开发机的脚手架工具中。

[0046] 在操作S203，利用Web组件，在开发机中通过浏览器运行应用程序的原生开发代码，以对应用程序进行调试。

[0047] 原生React Native开发框架提供了react-native run-ios(android)命令，因此，可启动iOS端和/或安卓端的模拟器对原生开发代码进行调试，但不支持在Web端对原生开发代码进行调试。

[0048] 在本公开实施例中，在原生的React Native开发框架上单独部署iftide-web的环境，实现“三端融合”的代码调试方案，即开发人员可在Web端上对原本运行在iOS端和安卓端的原生开发代码进行调试。由于iftide-web依赖较多，部署复杂，故需要将该功能集成到脚手架工具中。

[0049] 常见的开发机为PC机，开发人员在PC机中进行原生开发代码编写，并在具有安卓端和iOS端的移动设备中渲染运行。现将Web组件集成到开发机的脚手架工具中，使得开发机中具有Web端的运行环境。因此，开发人员可通过PC机中现有的浏览器直接对原生开发代码进行运行预览和调试，不再依赖外部真机或者模拟机，简化应用程序的开发调试过程，也减少开发设备的使用，提高开发机的运行效率。

[0050] 下面参考图3A和3B图，结合具体实施例对图2所示的方法做进一步说明。

[0051] 图3A示意性示出了根据本公开另一实施例的应用程序的开发方法的流程图。

[0052] 如图3A所示，将Web组件集成到开发机的脚手架工具中，包括操作S301～S302。

[0053] 在操作S301,获取Web组件的配置文件。

[0054] 在操作S302,基于配置文件,在开发机的React Native开发框架中集成Web依赖环境、调试环境、cli命令、Web端入口文件和Web端打包文件。

[0055] 在开发机中集成的Web运行环境包括:Web依赖环境、Web调试环境、cli命令、Web段入口文件和Web端打包文件等。

[0056] 例如,在开发机中集成Web依赖环境,包括对配置文件过滤,得到目标配置文件;将目标配置文件转换成与Web端适配的文件格式;在开发机中启动转换文件格式后的目标配置文件,以建立Web依赖环境。

[0057] 在集成Web依赖环境的过程中,首先从工程文件中根据存储路径获取到集成环境所需的配置文件,配置文件可以包括入口文件、业务功能文件、原型文件、模型文件等。例如,入口文件可以是web/index.web.js文件;业务功能文件可以是app文件和WapBank文件;原型文件可以是mydemo文件;模型文件可以是node_moduLes/react-native-uncompiled文件等等。

[0058] 本公开实施例提供其中一种集成Web依赖环境的实施方法:Const babelLoaderConfiguration=｛

[0059] test:/\.js$/,

[0060] //Add every directory that needs to be compiled by Babel during the build.

[0061] include:[

[0062] path.resoLve(appDirectory,'web/index.web.js'),

[0063] path.resolve(appDirectory,'app'),

[0064] path.resoLve(appDirectory,'WapBank'),

[0065] path.resoLve(appDirectory,'mydemo'),

[0066] path.resoLve(appDirectory,'node_moduLes/react-native-uncompil ed').

[0067] 其次,在开发机中集成Web调试环境和cli命令。集成Web调试环境提供运行的基础环境,使应用程序的原生开发代码可以运行生成访问页面,cli命令为运行指令。

[0068] 本公开实施例提供其中一种集成Web调试环境和cli命令的实施方法:

[0069] "webpack-cli":"^2.0.12",

[0070] "webpack-dev-server":"3.1.1"

[0071] "webstart":"./node_moduLes/.bin/webpack-dev-server-d--debug--devtooLevaL-source-map--config./web/webpack.config.js--inLine--hot--colors"

[0072] 在开发机中集成Web端入口文件,包括:基于配置文件,构建运行容器;在运行容器中构建注册入口,以通过注册入口运行业务项目。通过入口文件,开发人员可为应用程序添加不同的目标业务功能。

[0073] 本公开实施例提供其中一种集成Web端入口文件的实施方法:

```
<!DOCTYPE html>
<meta charset="utf-8">
<title>React Native for Web</title>
<meta          name="viewport"          content="width=device-width,
initial-scale=1">
<div id="react-app"></div>
<script src="/bundle.web,js"></script>
import ...
AppRegistry.registerComponent('App', ()=>Fund)
AppRegistry.runApplication('App', {
    initialProps:{},
    rootTag:document.getelementByid(('react-app')
});
```

[0075] 在开发机中集成Web端打包文件,包括将开发机中原始的打包文件转换为Web端打包文件。在原生的React Native开发框架中,打包文件的入口为iOS入口和安卓入口,为实现对文件的加载和打包,需要将入口替换为Web入口。

[0076] 本公开实施例提供其中一种集成Web端打包文件的实施方法:

```
// webpack.config.js
module.exports = {
  //...the rest of your config
  resolve: {
    alias: {
      'react-native':'react-native-web'
    }
  }
}
```

[0078] 图3B示意性示出了根据本公开另一实施例的应用程序的开发方法的流程图。

[0079] 如图3B所示,将Web组件集成到开发机的脚手架工具中,还包括操作S303。

[0080] 在操作S303,将开发机的React Native开发框架的原生调试接口转换成Web端的调试接口,调试接口包括界面调试接口和页面样式调试接口。

[0081] React Native开发框架的原生调试接口适用于iOS端和安卓端,且面对不同类型的终端,需要相对应地为每个类型的终端均提供一个调试接口。在调试的过程中,开发人员需要频换切换接口进行相应的调试,增加调试难度。通过本公开实施例,采用Web端的浏览器对原生开发代码进行直接预览和调试,简化调试过程,提高开发效率。

[0082] 调试接口均构建于Web的运行环境中,集成为一个开发工具,即chrome-devTools。例如,开发人员对应用程序进行UI调试时,调用electron提供的mainWindow.webContents.openDevTools()函数,即可调用展示chrome-devtools中的UI调试功能。界面调试接口实现功能包括:JavaScript断点调试,堆栈信息展示,Pretty Print等调试功能;源代码查看及sourcemap的自动转换;console控制台以及网络监控及JS性能监控等等。

[0083] 通过原生的调试接口,开发人员只能对react对运行后的页面元素及样式进行检查功能,转换后的Web端调试接口,可以实现对react运行前的组件层次的检查,还可以查看各个组件Props、State等信息。

[0084] 作为一种可选实施例,本申请集成Web运行环境后的React Native开发框架,还提

供Mock功能。由于iOS端和安卓端等移动端的特殊性，在PC机上运行的Web端无法实现移动端的全部功能，因此，提供模拟数据生成器(Mock)，对应用程序在iOS端和安卓端上可执行的功能进行模拟运行，以提供全面的调试功能。

[0085] 例如，iOS端和安卓端等移动端可进行扫描二维码等操作，但PC机上运行的Web端无法进行扫描二维码的操作。因此通过Mock功能生成二维码的模拟字符串，以通过Web端的浏览器模拟字符串进行识别，实现对扫描二维码这一功能的测试。

[0086] 常见的，对于大多数开发人员，需要PC机和模拟机或者PC机和真机两种不同的终端进行应用程序的开发。其中涉及的设备较多，且调试及开发过程中需要频繁地切换不同的设备，界面调试与样式调试还在两个不同的PC工具中完成，影响研发体验。若使用真机进行原生开发代码的预览和调试，则需要保证PC机与真机在同一个局域网内，开发过程存在网络限制。若使用模拟器进行原生开发代码的预览个调试，模拟器占用内存较多，很容易造成PC机的性能不足，影响开发体验。

[0087] 通过本公开实施例，在PC机中集成web环境，使得开发人员直接利用PC机中已安装的浏览器，对原生开发代码进行预览和调试。在简化应用程序开发过程的同时，也降低了对开发设备数量的需求，提高了PC机的运行效率，降低了开发人员的开发难度。

[0088] 图4示意性示出了根据本公开的实施例的应用程序的开发装置的框图。

[0089] 如图4所示，装置400包括获取模块410、集成模块420和运行模块430。

[0090] 获取模块410，用于获取预先封装的Web组件，Web组件提供原生开发代码在浏览器中运行的环境。

[0091] 集成模块420，用于将Web组件集成到开发机的脚手架工具中。

[0092] 运行模块430，用于利用Web组件，在开发机中通过浏览器运行应用程序的原生开发代码，以对应用程序进行调试。

[0093] 通过本公开实施例，将提供原生开发代码在浏览器中运行的环境的Web组件集成到开发机的脚手架工具中，使得开发机中具有Web运行环境。开发人员可通过开发机中现有的浏览器直接对原生开发代码进行运行和预览，不再依赖外部真机或者模拟机，简化应用程序的开发调试过程，也减少开发设备的使用，提高开发机的运行效率。

[0094] 根据本公开的实施例的模块、子模块、单元、子单元中的任意多个或其中任意多个的至少部分功能可以在一个模块中实现。根据本公开实施例的模块、子模块、单元、子单元中的任意一个或多个可以被拆分成多个模块来实现。根据本公开实施例的模块、子模块、单元、子单元中的任意一个或多个可以至少被部分地实现为硬件电路，例如现场可编程门阵列(FPGA)、可编程逻辑阵列(PLA)、片上系统、基板上的系统、封装上的系统、专用集成电路(ASIC)，或可以通过对电路进行集成或封装的任何其他的合理方式的硬件或固件来实现，或以软件、硬件以及固件三种实现方式中任意一种或以其中任意几种的适当组合来实现。或者，根据本公开实施例的模块、子模块、单元、子单元中的一个或多个可以至少被部分地实现为计算机程序模块，当该计算机程序模块被运行时，可以执行相应的功能。

[0095] 例如，获取模块410、集成模块420和运行模块430中的任意多个可以合并在一个模块/单元/子单元中实现，或者其中的任意一个模块/单元/子单元可以被拆分成多个模块/单元/子单元。或者，这些模块/单元/子单元中的一个或多个模块/单元/子单元的至少部分功能可以与其他模块/单元/子单元的至少部分功能相结合，并在一个模块/单元/子单元中

实现。根据本公开的实施例，获取模块410、集成模块420和运行模块430中的至少一个可以至少被部分地实现为硬件电路，例如现场可编程门阵列(FPGA)、可编程逻辑阵列(PLA)、片上系统、基板上的系统、封装上的系统、专用集成电路(ASIC)，或可以通过对电路进行集成或封装的任何其他的合理方式等硬件或固件来实现，或以软件、硬件以及固件三种实现方式中任意一种或以其中任意几种的适当组合来实现。或者，获取模块410、集成模块420和运行模块430中的至少一个可以至少被部分地实现为计算机程序模块，当该计算机程序模块被运行时，可以执行相应的功能。

[0096] 图5A示意性示出了根据本公开另一实施例的应用程序的开发装置的框图。

[0097] 如图5A所示，集成模块420包括：获取单元421，用于获取Web组件的配置文件。集成单元422，用于基于配置文件，在开发机的React Native开发框架中集成Web依赖环境、调试环境、cli命令、Web端入口文件、Web端打包文件。

[0098] 其中，集成单元422，包括：过滤单元4221，用于对配置文件过滤，得到目标配置文件；第二转换单元4222，用于将目标配置文件转换成与Web端适配的文件格式；启动单元4223，用于在开发机中启动转换文件格式后的目标配置文件，以建立Web依赖环境。

[0099] 集成单元422，还包括：第一构建单元4224，用于基于配置文件，构建运行容器；第二构建单元4225，用于在运行容器中构建注册入口，以通过注册入口运行业务项目。

[0100] 集成单元422，还包括：第三转换单元4226，用于将开发机中原始的打包文件转换为Web端打包文件。

[0101] 图5B示意性示出了根据本公开另一实施例的应用程序的开发装置的框图。

[0102] 如图5B所示，集成模块420还包括：第一转换单元423，用于在开发机的React Native开发框架的原生调试接口转换成Web端的调试接口，调试接口包括界面调试接口和页面样式调试接口。

[0103] 需要说明的是，本公开的实施例中应用程序的开发装置部分与本公开的实施例中应用程序的开发方法部分是相对应的，应用程序的开发装置部分的描述具体参考应用程序的开发方法部分，在此不再赘述。

[0104] 图6示意性示出了根据本公开实施例的适于实现上文描述的方法的电子设备的框图。图6示出的电子设备仅仅是一个示例，不应对本公开实施例的功能和使用范围带来任何限制。

[0105] 如图6所示，根据本公开实施例的电子设备500包括处理器501，其可以根据存储在只读存储器(ROM)502中的程序或者从存储部分508加载到随机访问存储器(RAM)503中的程序而执行各种适当的动作和处理。处理器501例如可以包括通用微处理器(例如CPU)、指令集处理器和/或相关芯片组和/或专用微处理器(例如，专用集成电路(ASIC))，等等。处理器501还可以包括用于缓存用途的板载存储器。处理器501可以包括用于执行根据本公开实施例的方法流程的不同动作的单一处理单元或者是多个处理单元。

[0106] 在RAM 503中，存储有系统500操作所需的各种程序和数据。处理器501、ROM 502以及RAM 503通过总线504彼此相连。处理器501通过执行ROM 502和/或RAM 503中的程序来执行根据本公开实施例的方法流程的各种操作。需要注意，所述程序也可以存储在除ROM 502和RAM 503以外的一个或多个存储器中。处理器501也可以通过执行存储在所述一个或多个存储器中的程序来执行根据本公开实施例的方法流程的各种操作。

[0107] 根据本公开的实施例，系统500还可以包括输入/输出(I/O)接口505，输入/输出(I/O)接口505也连接至总线504。系统500还可以包括连接至I/O接口505的以下部件中的一项或多项：包括键盘、鼠标等的输入部分506；包括诸如阴极射线管(CRT)、液晶显示器(LCD)等以及扬声器等的输出部分507；包括硬盘等的存储部分508；以及包括诸如LAN卡、调制解调器等的网络接口卡的通信部分509。通信部分509经由诸如因特网的网络执行通信处理。驱动器510也根据需要连接至I/O接口505。可拆卸介质511，诸如磁盘、光盘、磁光盘、半导体存储器等等，根据需要安装在驱动器510上，以便于从其上读出的计算机程序根据需要被安装入存储部分508。

[0108] 根据本公开的实施例，根据本公开实施例的方法流程可以被实现为计算机软件程序。例如，本公开的实施例包括一种计算机程序产品，其包括承载在计算机可读存储介质上的计算机程序，该计算机程序包含用于执行流程图所示的方法的程序代码。在这样的实施例中，该计算机程序可以通过通信部分509从网络上被下载和安装，和/或从可拆卸介质511被安装。在该计算机程序被处理器501执行时，执行本公开实施例的系统中限定的上述功能。根据本公开的实施例，上文描述的系统、设备、装置、模块、单元等可以通过计算机程序模块来实现。

[0109] 本公开还提供了一种计算机可读存储介质，该计算机可读存储介质可以是上述实施例中描述的设备/装置/系统中所包含的；也可以是单独存在，而未装配入该设备/装置/系统中。上述计算机可读存储介质承载有一个或者多个程序，当上述一个或者多个程序被执行时，实现根据本公开实施例的方法。

[0110] 根据本公开的实施例，计算机可读存储介质可以是非易失性的计算机可读存储介质。例如可以包括但不限于：便携式计算机磁盘、硬盘、随机访问存储器(RAM)、只读存储器(ROM)、可擦式可编程只读存储器(EPROM或闪存)、便携式紧凑磁盘只读存储器(CD-ROM)、光存储器件、磁存储器件，或者上述的任意合适的组合。在本公开中，计算机可读存储介质可以是任何包含或存储程序的有形介质，该程序可以被指令执行系统、装置或者器件使用或者与其结合使用。

[0111] 例如，根据本公开的实施例，计算机可读存储介质可以包括上文描述的ROM 502和/或RAM 503和/或ROM 502和RAM 503以外的一个或多个存储器。

[0112] 附图中的流程图和框图，图示了按照本公开各种实施例的系统、方法和计算机程序产品的可能实现的体系架构、功能和操作。在这点上，流程图或框图中的每个方框可以代表一个模块、程序段或代码的一部分，上述模块、程序段或代码的一部分包含一个或多个用于实现规定的逻辑功能的可执行指令。也应当注意，在有些作为替换的实现中，方框中所标注的功能也可以以不同于附图中所标注的顺序发生。例如，两个接连地表示的方框实际上可以基本并行地执行，它们有时也可以按相反的顺序执行，这依所涉及的功能而定。也要注意的是，框图或流程图中的每个方框，以及框图或流程图中的方框的组合，可以用执行规定的功能或操作的专用的基于硬件的系统来实现，或者可以用专用硬件与计算机指令的组合来实现。

[0113] 本领域技术人员可以理解，本公开的各个实施例和/或权利要求中记载的特征可以进行多种组合和/或结合，即使这样的组合或结合没有明确记载于本公开中。特别地，在不脱离本公开精神和教导的情况下，本公开的各个实施例和/或权利要求中记载的特征可

以进行多种组合和/或结合。所有这些组合和/或结合均落入本公开的范围。

[0114] 以上对本公开的实施例进行了描述。但是，这些实施例仅仅是为了说明的目的，而并非为了限制本公开的范围。尽管在以上分别描述了各实施例，但是这并不意味着各个实施例中的措施不能有利地结合使用。本公开的范围由所附权利要求及其等同物限定。不脱离本公开的范围，本领域技术人员可以做出多种替代和修改，这些替代和修改都应落在本公开的范围之内。

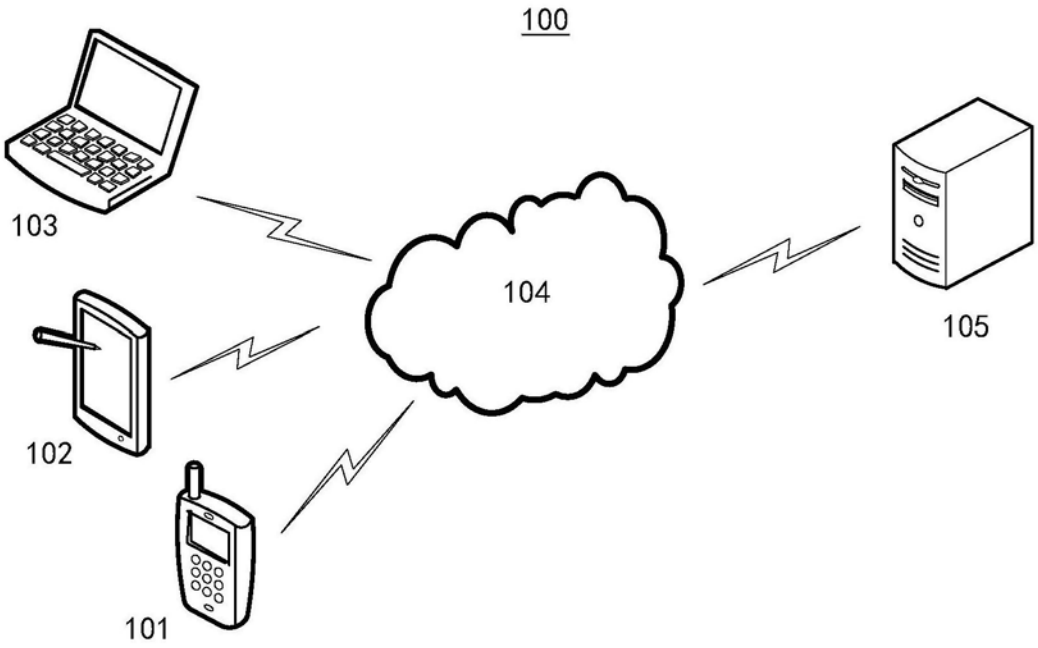

100
103
104
105
102
101

图1

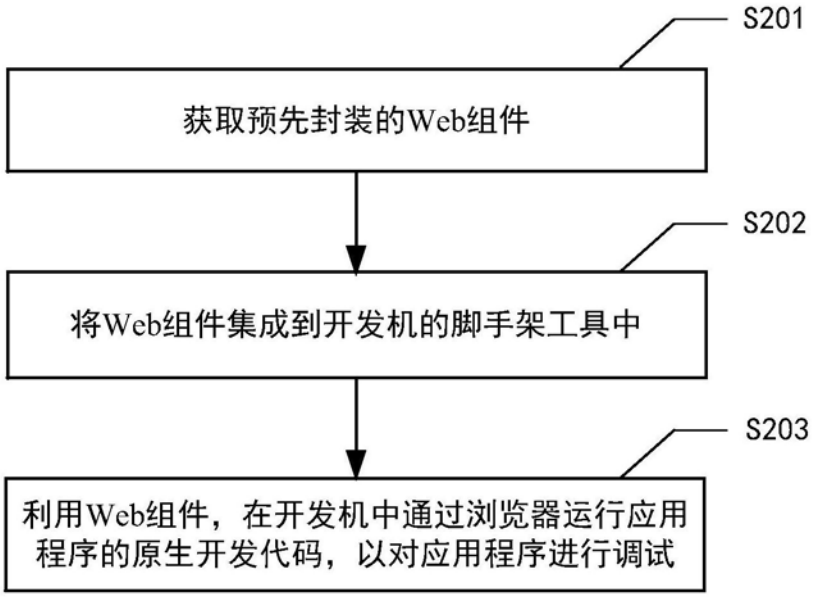

S201
获取预先封装的Web组件
S202
将Web组件集成到开发机的脚手架工具中
S203
利用Web组件，在开发机中通过浏览器运行应用程序的原生开发代码，以对应用程序进行调试

图2

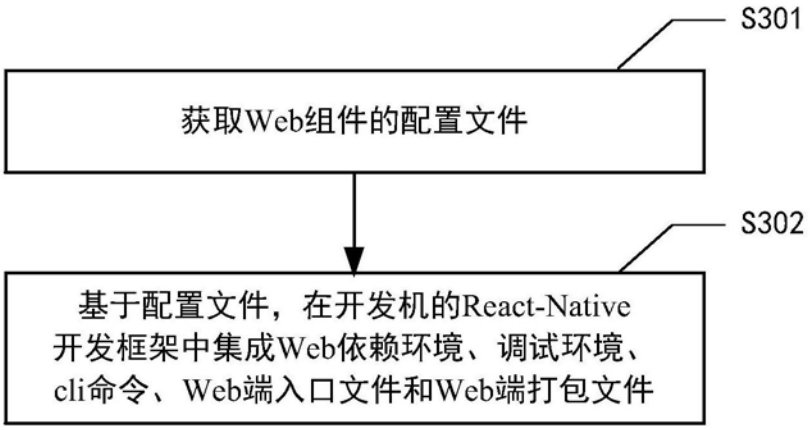

S301
获取Web组件的配置文件
S302
基于配置文件，在开发机的React-Native开发框架中集成Web依赖环境、调试环境、cli命令、Web端入口文件和Web端打包文件

图3A

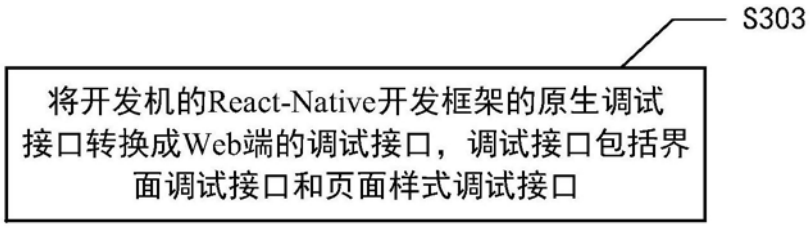

S303
将开发机的React-Native开发框架的原生调试接口转换成Web端的调试接口，调试接口包括界面调试接口和页面样式调试接口

图3B

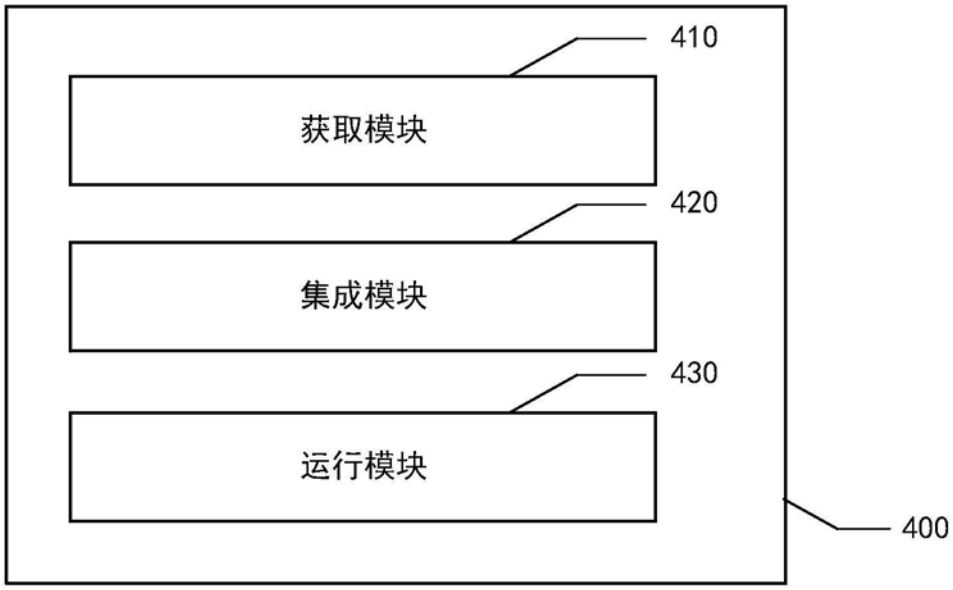

410
获取模块
420
集成模块
430
运行模块
400

图4

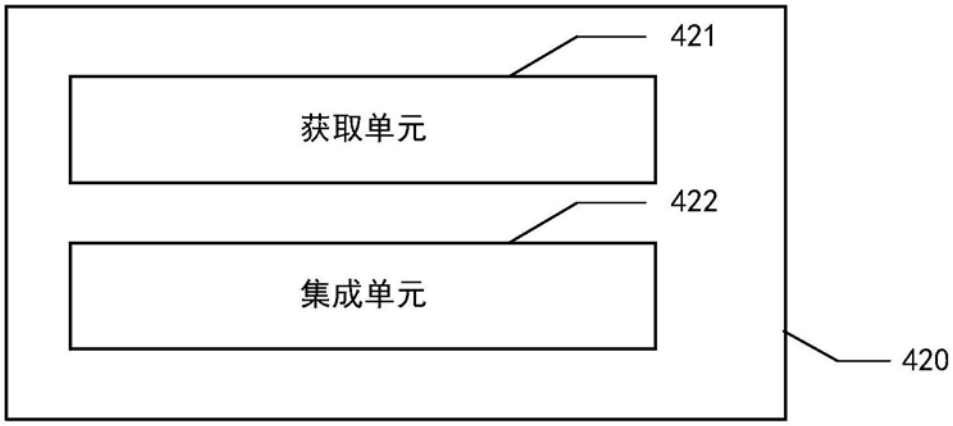


图5A

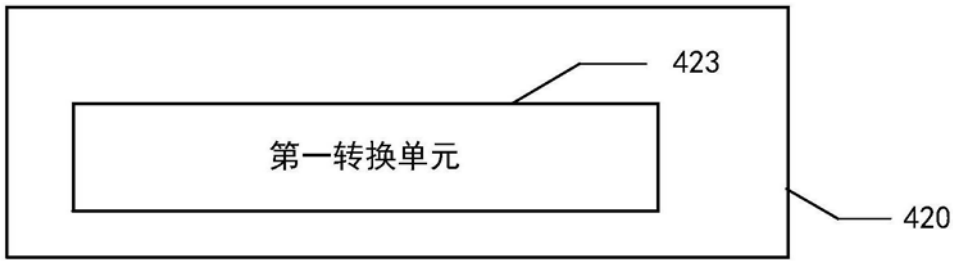


图5B

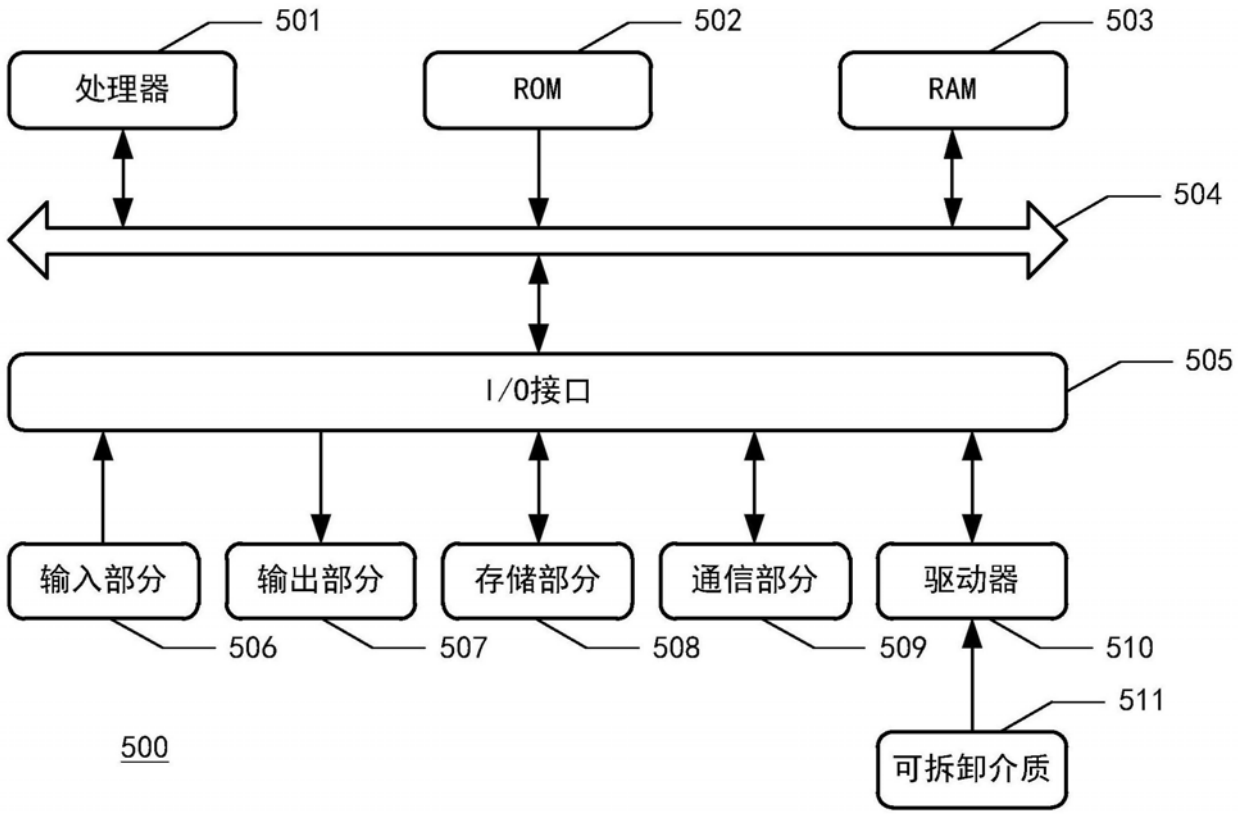

501
处理器
502
ROM
503
RAM
504
505
I/O接口
输入部分
输出部分
存储部分
通信部分
驱动器
506
507
508
509
510
511
可拆卸介质
500

图6