(54) 发明名称
一种通过数据渲染移动应用界面的方法

(57) 摘要
本发明涉及通过数据渲染移动应用界面的方法，包括以下步骤：S1→S3。其中：S1 为：在服务器植入页面模板 html 与控件 xml，并解析形成页面数据以及控件数据的 Json 数据包；S2 为：客户端向服务器发送请求并接收该 Json 数据包，将数据转化为页面模型；S3 为：将该页面模型数据进行处理生成页面事件、页面控件，同时渲染页面布局。本发明的有益效果为：能够通过数据渲染出移动的 web 页面，可以使前端程序员不必过于关注页面的控件实现细节，不仅节约了开发时间，而且增加了移动应用之间的页面复用度。
1. 一种通过数据渲染移动应用界面的方法，其特征在于，包括以下步骤；
S1: 抽取页面交互特性，分离数据、布局以及控件，在服务器植入页面模板 html 与控件 xml，并解析形成页面数据以及控件数据的 Json 数据包；
S2: 客户端向服务器发送请求并接收该 Json 数据包，将其转化为页面模型；
S3: 将该页面模型数据进行处理生成页面事件、页面控件，同时渲染页面布局。
2. 根据权利要求 1 所述的方法，其特征在于，进一步包括：
客户端检查 Json 数据包的完整性；
将页面事件绑定至页面控件上，以便两者进行集中管理。
3. 根据权利要求 1 或 2 所述的方法，其特征在于，在步骤 S2 中，进一步包括：
通过浏览器启动指定的初始页面，并读入指定的配置文件；
读取配置文件中的信息并获取服务 URL，开启构建请求服务器的 Json 数据包；
通过 HTTP 协议向服务器发送 Ajax 请求；
获取响应的 Json 数据，验证其状态值，判断响应是否成功；
将成功响应的 Json 数据转化为页面模型数据。
4. 根据权利要求 3 所述的方法，其特征在于，通过 Ajax 请求获得服务器的响应数据，包括以下步骤：
   通过 Ajax 请求指定的服务器地址获得页面 Json 数据；
   解析获得的 Json 数据，验证数据的状态。
5. 根据权利要求 3 所述的方法，其特征在于，将请求的页面 Json 数据转化为页面模型，包括以下步骤：
   将请求获得的页面数据进行分割；
   将分割获得的页面数据按照不同的类型封装为不同的数据模型；
   将数据模型搜集在不同的 collection 中。
6. 根据权利要求 5 所述的方法，其特征在于，在步骤 S3 中，进一步包括以下步骤：
   从不同类型的 collection 中将数据模型还原；
   将数据模型分解为控件模型与事件模型；
   按照控件模型的类型将其渲染成不同的控件；
   通过页面模型的所属 ID 将其绑定在指定的控件模型上。
7. 根据权利要求 2 或 6 所述的方法，其特征在于，进一步还包括以下步骤：
   将页面事件、页面控件进行集中管理，包括有：
   在生成页面控件和页面事件时将其放入页面容器中；
   在页面数据进行切换时将指定的页面控件和页面事件清除。
8. 根据权利要求 1 所述的方法，其特征在于，配置文件定义的内容包括：页面响应服务器的地址；应用所属的 ID；应用初始化页面；事件的响应地址。
9. 根据权利要求 1 所述的方法，其特征在于，进一步包括：
   页面事件响应产生页面服务时，搜集页面控件信息，构造请求的数据包信息；
   获取服务器的响应 Json 数据，重新使用以上方法渲染新的页面。
10. 根据权利要求 1 所述的方法，其特征在于，服务器响应的 Json 数据格式包括：数据包的状态值、页面的模版数据、页面的控件描述信息集合、页面事件的定义集合、应用认证
信息以及页面签名信息中的一种或者几种。
说明书

一种通过数据渲染移动应用界面的方法

技术领域
[0001] 本发明涉及一种能够增加移动应用页面复用度的通过数据渲染移动应用界面的方法。

背景技术
[0002] 目前移动应用对于跨平台的解决方案，大多是通过使用 web 技术来解决的，而对于 web 页面编写通常是前端程序员设计成固定的页面而完成的。这种方式创建的应用的页面并不能在多个应用间进行复用，因而需要花费巨大的人力成本。
[0003] 目前 web 界面的渲染主要为两种技术：一种是网页跳转方式，客户端直接请求新页面替换当前页，另一种是通过交互的数据来渲染页面，通过程序解析交互的数据，再渲染页面。传统网页技术一般使用跳转的方式请求新的页面，但是一些新的应用开始使用 AJAX 技术，通过数据交互来渲染部分页面。
[0004] 在现有技术中，移动界面的设计实现大多基于 HTML 或者 XML 的配置式进行构造，比如 Android 的页面设计基于固定页面 XML 布局模板，开发人员通过编写 XML 文件来描述某个 Activity 的内容，程序会对布局模版进行解析，然后渲染产生页面。而基于传统的 WEB 方法的 JqueryMobile 等都是基于 HTML 文本进行描述。本发明是在抽取出了页面交互特性的基础上，将数据、布局和控件分离，用单独的配置文件和各自特定的的标签来描述页面，并且将常用的组件进行抽象分离，使其能够更加多样化，适用于多种移动环境下，保证了不论在 PC 端和移动端（比如 Android, iOS)，只要是支持 HTML5 的环境中都能得到同样的应用界面渲染效果。

发明内容
[0005] 针对现有技术中存在的不足，本发明的目的在于提供一种通过数据渲染移动应用界面的方法，能够通过数据渲染出移动的 web 页面，极大的提高了用于构建应用页面的复用度。
[0006] 本发明的技术方案如下：
[0007] 一种通过数据渲染移动应用界面的方法，包括以下步骤：
[0008] S1: 在服务器植入画面模板 html 与控件 xml, 并解析形成页面数据以及控件数据的 Json 数据包；
[0009] S2: 客户端向服务器发送请求并接收该 Json 数据包，将其转化为页面模板；
[0010] S3: 将该页面模板数据进行处理生成页面事件、页面控件，同时渲染页面布局。
[0011] 上述的方法，进一步包括：
[0012] 客户端检查 Json 数据包的完整性；
[0013] 将页面事件绑定至页面控件上，以使两者进行集中管理。
[0014] 上述的方法，在步骤 S2 中，进一步包括：
[0015] 通过浏览器启动指定的初始页面，并读入指定的配置文件；
[0016] 读取配置文件中的信息并获取服务 Url, 开始构建请求服务器的 Json 数据包；
[0017] 通过 http 协议向服务器发送 Ajax 请求；
[0018] 获取响应的 Json 数据，验证其状态值，判断响应是否成功；
[0019] 将成功响应的 Json 数据转化为页面模型数据。
[0020] 上述的方法，通过 Ajax 请求获得服务器的响应数据，包括以下步骤：
[0021] 通过 Ajax 请求指定的服务器地址获得页面 Json 数据；
[0022] 解析获得的 Json 数据，验证数据的状态。
[0023] 上述的方法，将请求的页面 Json 数据转化为页面模型，包括以下步骤：
[0024] 将请求获得的页面数据进行分割；
[0025] 将分割获得的页面数据按照不同的类型封装为不同的数据模型；
[0026] 将数据模型搜集在不同的 collection 中。
[0027] 上述的方法，在步骤 S3 中，进一步地包括以下步骤；
[0028] 从不同类型的 collection 中将数据模型还原；
[0029] 将数据模型分为控件模型与事件模型；
[0030] 按照控件模型的类型将数据渲染成不同的控件；
[0031] 通过页面模型的所属 ID 将其绑定在指定的控件模型上。
[0032] 上述的方法，进一步还包括以下步骤：
[0033] 将页面事件、页面控件进行集中管理，包括有：
[0034] 在生成页面控件和页面事件时将其放入页面管理器中；
[0035] 在页面数据进行切换时将指定的页面控件和页面事件清除。
[0036] 上述的方法，配置文件定义的内容包括：页面响应服务器的地址；应用所属的 ID；
应用初始化页面；事件的响应地址；
[0037] 上述的方法，进一步包括：
[0038] 页面事件响应产生页面迁移到，搜集页面控件信息，构造请求的数据包信息；
[0039] 获取服务器的响应 Json 数据，重新使用以上方法渲染新的页面。
[0040] 上述的方法，服务器响应的 Json 数据格式包括：数据包的状态值，页面的模版数据、
页面的控件描述信息集合，页面事件的定义集合，应用认证信息以及页面签名信息中的一种或者几种。
[0041] 本发明的主要效果是：
[0042] (1) 能够通过数据渲染出移动的 web 页面，可以使前端程序员不必过于关注页面的
控件实现细节，不仅节约了开发时间，而且增加了移动应用之间的页面复用度；
[0043] (2) 使用数据来渲染界面，减少了前端开发人员的工作量，同时使得应用程序升级
变得方便；
[0044] (3) 本发明运用了 AJAX 技术，不但加快了页面的渲染速度，减少了数据的交互量，
而且在传统 AJAX 交互的基础上，采用了纯 Json 数据交互的方式，进一步降低了交互的数据量，提升
了在移动环境下的使用性能；
[0045] (4) 在提取出了页面交互特性的基础上，将数据、布局和控件分离，用单独的配置
文件和各自特定的的标签来描述页面，并且将常用的组件进行抽象分离，使其能够更加多
样化，适用于多种移动环境下，保证了不论在 PC 端和移动端（比如 Android，iOS），只要是
支持 HTML5 的环境中都能得到同样的应用界面渲染效果。

附图说明
[0046] 图 1 为本发明的流程框图；
[0047] 图 2 为客户端渲染页面的工作过程示意图；
[0048] 图 3 为定义文件、服务器、客户端关系图。

具体实施方式
[0049] 以下描述和附图充分地示出本发明的具体实施方案，以使本领域的技术人员能够
实践它们。本发明的实施方案的范围包括权利要求书的整个范围，以及权利要求书的所有
可获得的等同。在本文中，本发明的这些实施方案可以被单独地或总地用术语“发明”来表
示，这仅仅是为了方便，并且如果事实上公开了一个的发明，不是要自动地限制该应用
的范围为任何单个发明或发明构思。
[0050] 图 1 说明了本发明实施例所述的通过数据渲染移动应用界面的方法的总体流程
图，如图 1 所示，一种通过数据渲染移动应用界面的方法，包括以下步骤 S1 ～ S3，其中，S1
为：在服务器植入页面模板 html 与控件 xml，并解析形成页面数据以及控件数据的 Json 数
据包；S2 为：客户端与服务器发送请求并接收该 Json 数据包，将其转化为页面模型；S3 为：
将该页面模型数据进行处理生成页面事件、页面控件，同时渲染页面布局。
[0051] 在本实施例中，页面模板 html 为：
[0052]
```html
  <div id="content">
    <div class="div-login-1">
      <img class="img-responsive" src="http://10.66.4.69:8080/SmartX/logo.png">
    </div>
    <div id="input" class="div-login-2">
```
[0053]
<div>
  <div id="btn" class="div-login-3">
  </div>
</div>

控件 xml 为:

<?xml version="1.0" encoding="utf-8"?>

<page code="login" title="登录" path="root">
  <data id="user" import="MemAccountBalance" iterator="object">
    <parameter name="where" value="relMemInfo.miName=@{request.get('userid')}" and mabPassword=@{request.get('password')}"/>
  </data>
</page>

<layout file="guodong/layout/main/login.html"/>

[0054] 本发明的方法应用在至少一个客户端和服务器之间，即，移动客户端的个数可以为多个，且各移动客户端均可通过访问接收服务器的数据进行自身用的应用界面渲染，以达到节省人力成本之目的。

[0055] 开发人员事先将页面模板 html 与控件 xml 放入服务器中，由服务器进行解析，将页面模板数据与控件数据形成的 Json 数据包存储到服务器的数据库中，任何一移动的客户端请求到这些数据后即可完成整个移动应用界面的渲染。

[0056] 其中，所述的客户端具有一移动前端的浏览器，浏览器的作用是：点击运行该浏览器，由浏览器中输入一指定的 Url 将指定的页面打开，以启动指定的初始化页面，从而在该初始化页面中读入一指定的配置文件，通过读取配置文件中的信息获取服务 Url，开始构建请求服务器的 Json 数据包。

[0057] 通过指定的配置文件构建数据请求，包括以下步骤；

[0058] a) 配置指定路径下的配置文件；

[0059] b) 导入并解析该配置文件信息；

[0060] c) 通过配置文件定义的服务地址与主页信息构建 Url；

[0061] d) 开始对解析获得的 Url 进行数据请求。

[0062] 以上步骤中所述的配置文件构建数据的请求是通过 HTTP 协议完成的。

[0063] 通过 http 协议向服务器发送 Ajax 请求，以获取响应的 Json 数据，包括以下步骤；

[0064] a) 通过 Ajax 请求指定的服务器地址获得页面 Json 数据；

[0065] b) 解析获得的 Json 数据，验证数据的状态，通过验证得到的 Json 数据的状态值来判断响应是否成功。

[0066] 其中，服务器响应的 Json 数据格式包括：数据包的状态值、页面的模板数据、页面的控件描述信息集合、页面事件的定义集合、应用认证信息以及页面签名信息等，在客户端进行数据请求时，请求的 Json 数据定义的内容至少包括有页面事件及页面控件的描述信息、请求状态信息以及页面模板信息。

[0067] 对于响应成功的 Json 数据转化为页面模型数据，其过程如下：
a) 将请求获得的页面数据进行分割；
b) 将分割获得的页面数据按照不同的类型封装为不同的数据模型；
c) 将数据模型搜集在不同的 collection（数据模型集合）中，以形成页面模型数据。

进一步形成的页面模块经过以下处理生成页面事件以及页面控件：

a) 从不同类型的 collection 中将数据模型还原；
b) 将数据模型分为控件模型与事件模型；
c) 按照控件模型的类型将其渲染成不同的控件；
d) 通过页面模型的所属 ID 将其绑定在指定的控件模型上。

通过以下步骤，对页面事件和页面控件进行集中管理：

a) 在生成页面控件和页面事件时将其放入页面管理器中；
b) 在页面数据进行切换时将指定的页面控件和页面事件清除。

具体地，本实施例中的页面模板 html 和控件 xml 的基础上的代码如下：

```html
<!-- 导航栏 -->

<widget id="navtoppanel" type="navtoppanel" container="navigator" bind=""/>

<navtoppanel></navtoppanel>
```

[0081]
<widget id="headertext" type="textview" container="navtoppanel" bind="登录" class="nav-title">
    <textview></textview>
</widget>

<widget id="register" type="iconbutton" container="navtoppanel" bind="" class="btn nav-btn nav-btn-right">
    <iconbutton iconclass="glyphicon glyphicon-user"></iconbutton>
</widget>

<!-- 主面板 -->
<widget id="userid" type="text" container="input" bind="">
    <text placeholder = "用户名"></text>
</widget>

<widget id="password" type="password" container="input" bind="">
    <password placeholder = "密码" md5 = "true"></password>
</widget>

<widget id="login" type="button" container="btn" bind="确认">
    <button></button>
</widget>

<event when="submit">
    if(request.get('x_source') == "login"){
        if(isspace(user)){
            if(isspace(session)){
                context.removeSession("user");
            }
            var session = context.createSession();
            context.prompt("OK");
            session.addSession("user",user);
        }
        else{
            context.error("用户名或密码错误！");
        }
    }
</event>

[0082]
<event/>

<action type="click" target="login" do="submit"/>
<action type="click" target="register" do="submit"/>
<transfer on="request.get('x_source') === 'register'" to="register"/>
<transfer on="isdef user" to="categorylist"/>
</page>

[0083] 图 2 为数据渲染界面的工作过程示意图，参见图 2，该过程包括:
[0084] 步骤 1): 客户端应用打开，发送请求给服务器，向服务器请求初始页面数据。在
本实施例中，请求格式为 http://10.66.4.69:8080/SmarX/webapi/adapterx/page/get/guodong/login。
[0085] 步骤 2): 服务器接收到请求，从数据库中读取数据，然后发送给客户端。在本实
施例中服务器发给客户端的 json 数据为:
[0086]
{
  "code": "OK",
  "msg": null,
  "extra": null,
  "refUrl": null,
  "token": null,
  "signature": "guodong__86e43707f6554515ae1066e28755aa7d",
  "datas": {
    "code": "login",
    "title": "登录",
    "reload": null,
    "layout": "<DIV id="content">
      <DIV class="div-login-1">
        <IMG class="img-responsive" src="http://10.66.4.69:8080/SmarX/logo.png"/>
      </DIV>
      <DIV class="div-login-2" id="input">
        <DIV class="div-login-3" id="btm">
          <DIV class="div-login-4" id="input">
            <SCRIPT id="itemview" type="text/template">
              </SCRIPT>
            "navigator": 
            "<DIV id="navigator"/>
            "pagewidgets": [
            
            [0087]
"wgttype": "navtoppanel",
"wgtid": "navtoppanel",
"wgtcontainer": "navigator",
"wgttitle": null,
"wgtvalue": "",
"wgtclass": "x_navtoppanel",
"wgtrequired": "false",
"wgtreadonly": false,
"wgtcontentpage": null,
"wgtvisible": true,
"interval": 0
},
{
"wgttype": "textview",
"wgtid": "headertext",
"wgtcontainer": "navtoppanel",
"wgttitle": null,
"wgtvalue": "登录",
"wgtclass": "nav-title",
"wgtrequired": "false",
"wgtreadonly": false,
"wgtcontentpage": null,
"wgtvisible": true,
"interval": 0
},
{
"wgttype": "iconbutton",
"wgtid": "register",
"wgtcontainer": "navtoppanel",
"wgttitle": null,
"wgtvalue": "",
"wgtclass": "btn nav-btn nav-btn-right",
"wgtrequired": "false",
"wgtreadonly": false,
"wgtvisible": true,
"interval": 0
}
"wtcontentpage": null,
"wtvisible": true,
"interval": 0,
"iconclass": "glyphicon glyphicon-user"
},
{
  "wttype": "text",
  "wtid": "userid",
  "wtcontainer": "input",
  "wttitle": null,
  "wtvalue": "",
  "wtclass": "x_text",
  "wtrequired": "false",
  "wtreadonly": false,
  "wtcontentpage": null,
  "wtvisible": true,
  "interval": 0,
  "placeholder": "用户名"
},
{
  "wttype": "password",
  "wtid": "password",
  "wtcontainer": "input",
  "wttitle": null,
  "wtvalue": "",
  "wtclass": "x_password",
  "wtrequired": "false",
  "wtreadonly": false,
  "wtcontentpage": null,
  "wtvisible": true,
  "interval": 0,
  "placeholder": "密码",
  "md5": "true"
}
步骤 3) 客户端接收到服务器发送的数据，检查数据包的完整性，确认数据在发送过程中没有错误。

步骤 4) 客户端解析接收到的数据，读取 datas 中的 layout 内容，并将这些内容加入至 html 页面 id 为 pageviews 的 div 中。

步骤 5) 客户端根据 datas 中的 pagewidgets 的若干个控件数据来渲染控件。
些数据中存储了控件的类型、id、父容器、内容、类名等信息，以及一些特定控件包含的信息如只读属性、循环属性等信息。

[0093] 步骤6）客户端根据datas中的pageactions的若干行为数据来绑定事件，这些数据中存储了行为的类型、行为的所属控件、行为的内容。

[0094] 上述实施例仅仅是为清楚地说明本发明创造所作的举例，而并非对本发明具体实施方式的限定。对于所属领域的普通技术人员来说，在上述说明的基础上还可以做出其它不同形式的变化或变动。这里无需也无法对所有的实施方式予以穷举。凡在本发明的精神和原则之内所引伸出的任何显而易见的变化或变动仍处于本权利要求的保护范围之中。
抽取页面交互特性，分离数据、布局以及控件，在服务端植入页面模板html与控件xml，并解析形成页面数据以及控件数据的Json数据包

客户端向服务端发送请求并接收该Json数据包，将其转化为页面模型

将该页面模型数据进行处理，生成页面事件、页面控件，同时渲染页面布局

图1
客户端发送页面请求给服务端

服务端接受请求，处理完毕后发送数据给客户端

客户端检查数据的完整性

客户端解析数据，渲染页面的布局

客户端渲染控件，并将控件加入到布局中

客户端绑定控件的事件

图 2
图 3