基于 Ajax 技术的 MVC 模式在远程控制实验室中的应用①

Application of MVC Pattern to Remote Control Laboratory Based on Ajax Technology

陈 武 阳春华 吴同茂 (中南大学 信息科学与工程学院 湖南 长沙 410083)

摘 要： 针对传统的同步交互模式，远程用户难以实时获取远程实验室中设备运行状态的问题，利用 AJAX (Asynchronous JavaScript and XML)技术定时发送查询请求，实现了远程设备运行状态的实时显示。为了方便代码的维护和调试，引入了 MVC（Model-View-Control）设计模式。实际使用表明，引入 AJAX 技术，能有效的减小服务器的响应时间和网络流量。

关键词： AJAX MVC 远程控制实验室 JNDI

1 引言

远程控制实验室允许远程用户通过网络对真实的设备操作并完成自己预定的实验[1]。传统的同步交互模式下，远程用户要知道远程实验室中设备运行状态需要刷新页面或点击按钮来提交请求[2]。当有多用户并发操作时，用户就要花较多时间等待服务器处理和返回整个页面。为了提高实验操作的流畅和实时性以及整个软件开发过程中实现具体清晰的逻辑划分，本文引入了基于 AJAX 技术的 MVC 设计模式来实现远程实验室设备状态的实时显示。

2 Ajax技术与MVC模式介绍

与传统的 Web 应用不同，Ajax 采用异步交互过程[3]，如图 1 所示。

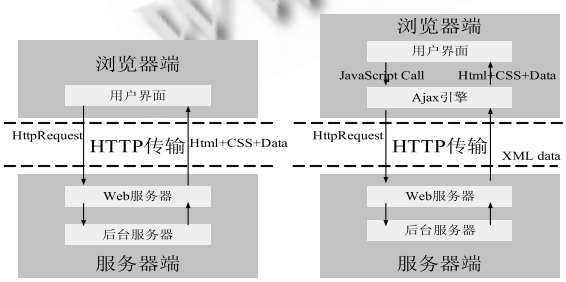


图 1 传统 Web 应用(左)和 Ajax 模型(右)

在用户界面和服务器之间添加了一个 Ajax 引擎，由它负责处理用户界面与服务器之间的交互。而 JavaScript 调用 Ajax 引擎来代替产生一个 HTTP 的用户动作和服务器交互，并部分更新用户相关界面，因此不需要重新载入整个页面，这样操作过程就更加流畅。

MVC 模式，即模型-视图-控制器模式，其核心思想是将整个程序代码分成相对独立而又能协同工作的三部分[4]。系统整体架构如图 2 所示。

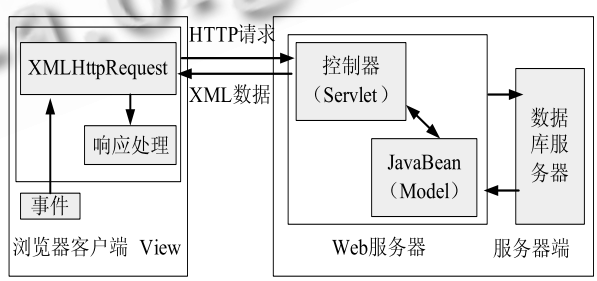


图 2 基于 AJAX 的 MVC 模式结构图

MVC 模式实现了数据层、显示层、业务层的分离，层次清晰而独立，有利于开发中的分工，并且提高了代码的可重用性及可扩展性。本文采用 JSP 页面来承担表示层的功能，负责显示远程设备的运行状态，服

① 基金项目:国家自然科学基金(60634020)
收稿时间:2008-12-05

务器端的 Servlet 程序充当控制层，而 JavaBean 则负责对数据库的操作。

3 基于Ajax技术的MVC模式的应用

3.1 整体架构与功能描述

远程控制实验室中，设备的运行状态由服务器端的实验管理程序在 MySQL 数据库中实时地记录下来。远程客户要获取设备的运行状态就要读取数据库中相应的记录项。这就需要浏览器端定时的异步向服务器发送查询设备状态的请求，服务器端对应的 Servlet 程序获取客户端发送过来的参数，调用 JavaBean 程序来相应的对数据库进行查询，并将查询结果返回给客户端。客户端的 AJAX 引擎当获取到服务器返回的数据后，利用 DOM 技术局部改变页面来实现无闪显示设备的状态。

3.2 客户端程序的具体实现过程

3.2.1 创建 XMLHttpRequest 对象

若 XMLHttpRequest 对象可以在不刷新的情况下发送请求并获取服务器的响应。通过下面的语句可以建立一个在大部分浏览器下都能正常工作的 XMLHttpRequest 对象。

```
var xhr=function(){
try {return newActive XObject ("Msxml2.X－MLHTTP" );} catch(e){}        // IE 高级版本
try{return new ActiveXObject("Microsoft.XMLH-TTP");}
catch(e){}        //IE 低级版本
try{return new XMLHttpRequest();}
catch(e){}        // Mozilla 浏览器
alert("XMLHttpRequest not supported");
//浏览器不支持 XMLHttpRequest 对象
return null; }
```

3.2.2 发送查询设备状态的请求

浏览器端通过异步方式向服务器发送设备状态请求，XMLHttpRequest 对象向服务器发送请求的代码如下：

```
xhr.open("GET","GetPlantStatus?PlantID=<%= PlantID%>,true); xhr.send(null);
```

open 是 XMLHttpRequest 对象的方法，第一个参数是 HTTP 请求的方式，第二是目标 URL 字符串，其中 PlantID 是向服务器传送的参数。第三个是指定在等待服务器返回信息的时间内是否继续执行下面的代码。默认为 true，表示继续执行。GET 请求方式时 send 的参数为 null。

3.2.3 指定并实现处理返回信息的函数

指定服务返回信息后客户端所调用的函数，就是将函数名赋给 XMLHttpRequest 对象的 onreadys－tatechange 属性，可以用如下代码处理返回的信息：

```
xhr.onreadystatechange=function ()
{if(xhr.readyState!=4){return;}        //请求未成功则返回
if(xhr.responseText=='No')
{document.getElementById(("PlantStatus").innerHTML="<font color=red>Plant is busy</font> " ; }
else
{document.getElementById(("PlantStatus").innerHTML="<font color=blue>Plant is ready </font> ";}}
```

当 XMLHttpRequest 对象的 readyState 值为 4 时开始处理返回的信息，如果设备忙，则红色显示，否则是显示蓝色的“ Plant is ready ”。

3.3 服务器端程序的实现过程

3.3.1 模型 JavaBean 的设计

远程实验室中设备的状态信息实时的保存在服务器端的数据库中，因此可以建立 JavaBean 程序封装对数据库进行操作的业务逻辑。为了提高访问 MYSQL 数据库的效率，结合 JDBC(Java DataBase Connectivity) 的 DataSourse 接口和 Connec-tionpool 连接池，使用 JNDI(Java Naming and Dire-ctory Interface)的 API 来访问数据库。在 JNDI API 中，Javax.naming 包提供了 Context 接口，该接口提供了将对象和名字绑定，以及通过名字检索对象的方法。InitialContext 类实现了 Context 接口，因此可使用下面代码来从连接池中获得一个数据源对象。

```
Context ctx = new InitialContext();
DataSource ds=(DataSource)ctx.lookup("jd-bc/ centreDB")
```

这就不同于 JDBC 使用 Driver 的方式来连接数据库，不用每次访问都要实例化一个对象，在 DataSource 中事先建立了多个数据库连接，这些数据库连接保存在连接池中。Java 程序访问数据库时，

只需要从连接池中取出空闲状态的数据库连接，当程序访问数据库结束，再将数据库连接放回连接池[5]。由于在 JNDI 中注册了数据源对象，而在程序中只需使用一个逻辑名称来引用它，JNDI 会自动根据你给出的名称找到与这个名称绑定的 DataSource 对象。然后就可以使用这个 DataSource 对象来建立和具体数据库的连接了。建立数据库连接池前，需要 XML 文件配置，配置内容如下：

```
<Context path="/ncslab">
<Resource name="jdbc/centraldb"
driverClassName="com.mysql.jdbc.Driver"
username="root"
password="123456"
url="jdbc:mysql://localhost/central"
...
/> </Context>
```

配置指明了要建立的 Web 站点 ncslab，JNDI 要查找的名字，以及连接池的驱动程序类名字、最大激活连接数、最大空闲连接数等属性。

3.3.2 控制层 Servlet 程序的开发

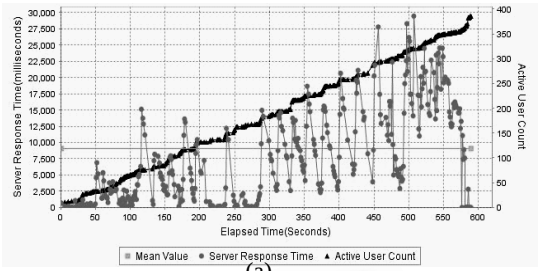
Servlet 的程序先获取客户端发送的设备号，然后调用 JavaBean 得到连接接口并创建 Statement 对象，再根据设备号执行该对象来查询数据库中设备项的记录，根据返回的结果向客户端发送设备运行的信息。最后依次关闭结果集、对象、输出流等，主要代码如下：

```
String sql="SELECT status from plants where ID ="+PlantID;
ResultSet set=stmt.executeQuery(sql);
set.next();
String status=set.getString("plant.status");
if (status == ”no” ) {out.print("no");}
else {out.print("yes");}
```

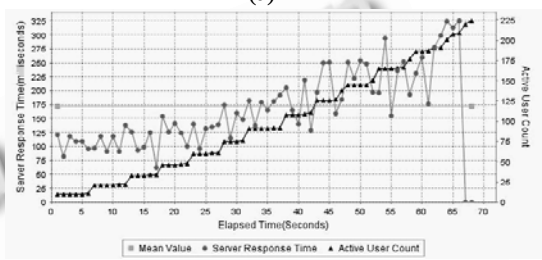
4 引入AJAX技术前后系统性能比较

为了测试 Web 性能，本文选择了强大易用的 Web 性能测试工具——AdventNet QEngine 6 来进行 Web 负载测试，测试条件如下：测试类型是 Ramp-up，每 5 秒虚拟用户增加 1 个。初始用户设为 10 个。

测试结果显示未引入 Ajax 技术的 WEB 程序如图 3(a)，服务器响应时间均值为 9067 毫秒，而在引入 AJAX 技术后如图 3(b)所示，服务器响应时间均值仅为 172 毫秒。



(a)



(b)

图 3 远程实验室页面响应时间图

可见 AJAX 技术的引入，提高了页面的响应速度。同时，测试结果中网络数据流量从 355kb/s 减少到了 62kb/，还有下载每个页面的平均时间也从 12445 毫秒减少到了 211 毫秒。

5 结论

引入 Ajax 技术的远程实验室，减少了服务器的响应时间，再加上不用等待页面刷新，用户操作起来感觉更加流畅。同时由于减少了不必要的请求，减少了服务器的数据吞吐量，从而减轻了网络和服务器的负担，给服务器和网络腾出更多的空间来处理大量的实验数据。

参考文献

1 吕为工,杨书华.远程实验室之网络实验管理系统的设与实现.计算机工程与应用, 2004,(3):174－176.

2 意大利锡耶纳大学远程自动化控制实验室网站. http://www.dii.unisi.it/~co- ntrol/－act/.

3 Garrett JJ. Ajax: A New Approach to Web Applications, February18, 2005.

4 王东,孙彬.基于 Ajax 的 MVC 框架的改造分析.计算机应用, 2007,(6):293－295.

5 赵天海,沈钧毅,齐勇,蒋涛.基于 JNDI 的高可靠命名服务的研究.计算机工程, 2006,05:44－47.