对基于J2EE的MVC模式视图部分改进

杨 刚,顾宏斌,赵芷晴
(南京航空航天大学 民航学院,江苏 南京 210016)

摘 要:为了降低大型应用系统的开发成本和维护成本,文中在提出了基于J2EE的MVC(Model View Controller)模式中的一些局限性后,对传统模式中的各部分进行适当的改变,构造了一个改进后的MVC模式。利用XSLT转换XML取代了原来较为复杂的JSP视图(View)部分,由于视图全部是由XML文件所组成的静态页,而且转化过程也是在客户端的浏览器中完成的,因此当用户访问量很高的时候可以大大减轻服务器压力。此外,有效地降低了开发的复杂性,缩短开发周期,并提高了系统的可靠性、可维护性和可扩展性。

关键词:视图;可扩展标记语言;java 2平台企业版
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2012)03-0103-03

MVC Mode View Improvement Based on J2EE

YANG Gang, GU Hong-bin, ZHAO Zhi-qing
(College of Civil Aviation, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract: In order to reduce the large application system development and maintenance costs, it proposes some limitations based on the J2EE MVC (Model View Controller) model, makes all part of the proper change on the traditional mode, constructs an improved MVC model. Using XSLT to convert XML to replace the original complex JSP view part, because the view is by all XML files consisting of a static page, and the transformation process is completed in the client browser, therefore, when a large number of users visit can significantly reduce the server pressure. In addition, can effectively reduce the complexity of the development, shorten the development cycle, and improve system reliability, maintainability and scalability.

Key words: view; XML; J2EE

0 引 言

MVC设计模式是在八十年代为编程语言发明的一种软件设计模式,至今已被广泛的应用[1]。现在利用Java开发的大型Web应用程序中,常用的MVC模式则是利用jsp作为视图,servlet作为控制,javabean(ejb)作为模型。使用MVC模式,由于分离了视图与业务逻辑,使得开发时间得到大幅度的缩减,它使程序员集中精力于业务逻辑,界面程序员集中精力于表现形式上。因多视图能共享一个模型,所以无论用户想要Flash界面或是WAP界面,用一个模型就能处理。

1 传统的基于EJB3.0的MVC设计模式

1.1 MVC模式简单介绍

视图(view):此部分是由JSP(JavaServer Pages)页面组成,负责展示界面给用户。只是作为一种数据输出并允许用户操纵的方式。

模型(model):模型是包含核心功能和数据的。封装了应用程序的状态,响应状态查询,应用程序功能以及通知视图的改变。

控制器(controller):负责同步视图与模型,将用户动作映射成模型更新相应的视图。

1.2 JSP技术

JSP(JavaServer Pages)是由Sun Microsystems公司倡导、许多公司参与一起建立的一种动态网页技术标准。它是在传统的网页HTML文件中插入Java程序段(Scriptlet)和JSP标记,从而形成JSP文件(+.jsp)。其开发的Web应用是跨平台的,即能在Linux下运行,也能在其他操作系统上运行[2,3]。同时,JSP将网页逻辑与网页设计和显示分离,支持可重用的基于组件的设计,使基于Web的应用程序的开发变得迅速和容易。当Web服务器在遇到访问JSP网页的请求时,首先执行其中的程序段,然后将执行结果连同JSP文件中的HTML代码一起返回给客户,因此客户端只要有浏览器就能浏览。

收稿日期:2011-07-22;**修回日期**:2011-10-27
基金项目:中国民航总局专项科技基金支持项目(E9905)
作者简介:杨 刚(1982-),男,江苏南京人,硕士研究生,CCF会员,研究方向为载运工具运用工程;顾宏斌,教授,博导,研究方向为计算机应用。

因此传统的视图部分,基本采用 JSP 技术。虽然 JSP 相对于过去能有效地提升程序员的工作效率,但缺点也逐渐显露出来。首先,因为采用 JSP 技术,会导致 HTML 代码与 Java 代码混在一起,从而不易于管理,而且代码查看也非常麻烦,基本不能使用"面向对象的"角度去看代码。其次,调试非常麻烦,编译器只能报出 jsp 页面中对应的 servlet 错误,很难找出错误地方。再次,采用动态编译,必须访问每个 jsp 页面,这样浪费内存,也增加读取时间,对访问者非常不友好。

2 XSL 介绍

XSL 代表可扩展样式表语言,它定义了描述 XML 样式的词汇集,也是 W3C 在 1999 年 11 月 16 日作为"推荐书"发布的 XML 结构和数据表现技术之一。XSL 主要包含 XSLT(XSL Transformations,可扩展样式表 XML 转换语言)和 XSL-FO(XSL Formatting Object,可扩展样式表格式化对象语言)。文中主要用到 XSLT。

XSLT 用于将一种 XML[4,5] 文档转换为另外一种 XML 文档,或者可被浏览器识别的其他类型的文档,比如 HTML 和 XHTML。通常,XSLT 是通过把每个 XML 元素转换为(X)HTML 元素来完成这项工作的。XSLT 使用 XPath 在 XML 文档中查找信息。XPath 被用来通过元素和属性在 XML 文档中进行导航。在转换过程中,XSLT 使用 XPath 来定义源文档中可匹配一个或多个预定义模版的部分。一旦匹配被找到,XSLT 就会把源文档的匹配部分转换为结果文档[6]。

利用 XSLT 的转化功能,可以简化视图部分代码。更主要的,XSLT 转化是在浏览器里完成的,能大量减轻服务器压力,降低维护成本。

3 MVC 模式部分的改进

本方法主要是基于 EJB3.0[7],定义一个接口。此接口继承 java. io. Serializable。接口只有一个方法,返回 xml 形式的字符串。这样,所有的实体 bean[8] 都需要继承这个接口。视图部分采用 xsl 标签来写,之后只需要用 javascript 匹配对应的 xml 即可。

以一个利用 MVC 模式编写的工程管理系统为例,如图 1 所示。

DRProManager 是一个 j2ee 工程,DRProManagerClient 为前台展示页面,相当于 MVC 模式的视图。DRProManagerEJBClient 为接口,定义了所有实体模型的接口,负责定义实体模型的基本行为。DRProManagerEJB 则是实现了 DRProManagerEJBClient 接口,相当于 MVC 模式的控制器。DRProManagerEJBJPA 负责持久化实体 bean,相当于 MVC 模式的模型(model)。DRProManagerUtils 是自定义的一些工具类。

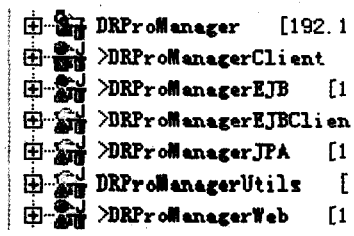

图 1 工程管理系统

3.1 控制器与模型部分创建

为了利用 XSLT 解析功能[9,10],在定义实体 bean 时,为所有类要实现 toXML 方法,此方法返回一个 XML 形式的字符串。以 User 为例,代码如下:

```
public String toXML() {
StringBuffer sb = new StringBuffer();
sb. append("<User>");
sb. append("<ID>");
sb. append(getId());
sb. append("</ID>");
sb. append("<User-Name>");
sb. append("<! [CDATA[");
sb. append(getUsername());
sb. append("]]>");
sb. append("</User-Name>");
sb. append("<Name>");
sb. append("<! [CDATA[");
sb. append(getName() = = null ? "" : getName());
sb. append("]]>");
sb. append("</Name>");
sb. append(Department. toXML());
sb. append(Company. toXML());
sb. append("</User>");
return sb. toString();
}
```

Departmen,Company 也是实体 bean。可以看出,toXML 方法中的数据都是以对象的方式获得。当有请求时,所请求的底层数据,都以 xml 的方式显示。

编写实体 bean 请求的 servlet,同样以 User 为例,有如下类,UserFunctionFactory. java,用来根据参数返回具体的 function 类。UserPermissionFactory. java,用来进行权限控制。UserQueryConditionFactory. java,用来做查询。

当视图部分有访问时,servlet 先实例化,然后根据 post 请求或 get 请求自动执行 doPost 或是 doGet 方法。servlet 只在第一次调用的时候实例化,以后就常驻内存。当有 servlet 对象后,通过 servlet 对象里的 getFactory 方法,获取 functionfactory。然后根据请求的 function 参数获取对应的 Function 类,处理请求,并获得实

体bean对象,每个实体bean对象有toXML方法,返回xml对象。

在本例中有如下Function:

```
private static final String FUNCTION_USER_LOGIN = "function_user_login";
private static final String FUNCTION_ADD_USER = "function_add_user";
private static final String FUNCTION_QUERY_USER = "function_query_user";
private static final String FUNCTION_GET_USER_BY_ID = "function_get_user_by_id";
private static final String FUNCTION_MODIFY_USER = "function_modify_user";
private static final String FUNCTION_DELETE_USER = "function_delete_user";
```

3.2 视图部分改进

利用xsl标签,编写视图部分代码,取代原来的jsp。部分代码如下:

```
<? xml version="1.0" encoding="UTF-8"? >
<xsl:stylesheet version="2.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/ User">
<form id="form_sensor_list">
<table cellspacing="0" id="sensor_grid">
<thead>
<tr>
<th>姓名</th>
<th>性别</th>
<th>部门</th>
<th>公司</th>
</tr>
</thead>
<xsl:for-each select=" User ">
<tr>
<td rowspan="{../@total}">
<xsl:value-of select="../../Name" />
</td>
<td rowspan="{../@total}">
<xsl:value-of select="../../Sex" />
</td>
<td rowspan="{../@total}">
<xsl:value-of select="../../ Department " />
</td>
<td rowspan="{../@total}">
<xsl:value-of select="../../ Company " />
</td>

</tr>
</xsl:for-each>
</table>
</form>
</xsl:template>
</xsl:stylesheet>
```

在客户端,通过使用JavaScript,可以进行浏览器的确认测试,根据浏览器来使用不同的样式[11,12]。所以,用JavaScript来转换xml文档是比较合适的。

编写JavaScript,转换匹配的xml文档,JavaScript代码如下:

```
function parse(xmlDoc, xslDoc) {
if (xmlDoc == null)
return null;
if (xslDoc == null)
return null;
try {
if (isIE()) {
return xmlDoc.transformNode(xslDoc);
} else {
var processor = new XSLTProcessor();
processor.importStylesheet(xslDoc);
var resultDocument = processor.transformToDocument(xmlDoc);
var serializer = new XMLSerializer();
return serializer.serializeToString(resultDocument);
}
} catch (e) {
return null;
}
};
function buildGrid() {
var xmlDoc = loadXML(REQUEST_REMOTE
+ '/UserServlet? function = function_query_user&limit = 20&start=0');
var xslDoc = loadXML("../xslt/User.xsl");
var htmlStr = parse(xmlDoc, xslDoc);

if (! htmlStr)
return;
return htmlStr;
}
```

其中xmlDoc为实体bean中toXML的结果。loadXML函数提出function请求,请求包含的function已经在控制器部分编写完成。在上例中则通过function_query_bases这个常量表示需要查询请求。至此整个MVC框架已经搭成。当需要新的页面时,只需增加新的XSL文件,然后匹配需要的持久层数据对象即可。如果需对视图部分改动,只需改动XSL文件。如对控制器改动,则修改function类。视图与控制分离,大大减少改动的代码。对后期的维护有很大的作用。

(下转第109页)

表中前两列为随机参数训练 HMM 实验结果,五个域的精确率和召回率较低,最明显的是 author 域的精确率仅为 0.465,booktitle 域的召回率为 0.582,REC 和 PRE 平均值仅为 0.726 和 0.710,实验结果不太理想。基于 SA-HMM、GA-HMM 的实验效果有所改善,表中最后两列是基于 SGA-HMM 的实验结果,其精确率和召回率都有明显提高,如 date 域的召回率达到了 0.954,publisher 域的精确率达到了 0.961,平均 REC 和 PRE 高达 0.897 和 0.913。

6　结束语

实验结果表明将遗传退火算法(SGA)训练隐马尔可夫模型,利用模拟退火算法能以一定概率跳出局部极值区域从而增大了找到全局极值的概率的优点,克服了遗传算法过早收敛的缺点,遗传算法又弥补了模拟退火算法随机漫游、全局搜索能力不强、运行速度慢的缺陷,该算法快速有效地找到最优初始参数,从而极大地改善了实验效果,在 Web 挖掘中提高了域提取的精确率和召回率。

参考文献:

[1] Salzenstein F, Pieczynski W. Parameter Estimation in Hidden Fuzzy Markov Random Fields and Image Segmentation[J]. Graphical Models and Image Processing, 1997, 59(4): 205-220.

[2] Phan Xuan-Hieu, Horiguchi S, Ho Tu-Bao. Automated data extraction from the web with conditional models[J]. Int J Business Intelligence and Data Mining, 2005, 1(2): 210-228.

[3] Freitag D, McCallum A, Pereira F. Maximum Entropy Markov Models for Information Extraction and Segmentation[J]. Proceeding of ICML, 2000, 1(1): 591-598.

[4] 林慧君,彭　宏. 模拟退火算法在全局查询优化中的应用[J]. 计算机技术与发展,2006,16(4):155-157.

[5] 贾德香,唐国庆,韩　净. 基于改进模拟退火算法的电网无功优化[J]. 继电器,2004,32(4):32-36.

[6] 洪沛霖,张佑生,邢　燕. 基于改进模拟退火算法的手写体数字识别[J]. 计算机技术与发展,2007,17(9):15-17.

[7] 庞　峰. 模拟退火算法的原理及算法在优化问题上的应用[D]. 长春:吉林大学,2005.

[8] 周　明,孙树栋. 遗传算法原理及其应用[M]. 北京:国防工业出版社,2000.

[9] Gao Shan, Shan Yuanda. Advanced genetic algorithm approach to unit commitment wish searching optimization[J]. Proceedings of the CSEE, 2001, 21(3): 45-48.

[10] 高经纬,张培林,李　峰,等. 遗传算法和神经网络在铁谱图象识别中的应用[J]. 润滑与密封,2004(1):16-18.

[11] 牛志华,李乃成,肖国镇. 一种新的求解多目标优化问题的混合遗传算法[J]. 计算机工程,2003,29(18):64-66.

[12] 任传祥,张　海,范跃祖. 混合遗传模拟退火算法在公交智能调度中的应用[J]. 系统仿真学报,2005, 17(9):75-78.

[13] Liu Jianghua, Cheng Junshi, Chen Jiapin. Optimization of HMM parameters based on chaos and genetic algorithm for hand gesture recognition[J]. Journal of Systems Engineering and Electronics, 2002, 3(4): 79-84.

(上接第 105 页)

4　结束语

由于工程代码过多,只给出相关关键性的代码。文中利用 XSLT 技术,替代了传统的 MVC 模式中的 JSP 页面。该方案采用完全的静态页作为视图页面,能很大程度减少服务器压力。此外,可以让视图部分的代码看的很清晰,基本都是以“面向对象的”角度。虽然开始需要一些额外的工作,但对于企业级应用来说,之后所带来的便利是肯定的。

参考文献:

[1] 潘海兰,吴翠红,葛　晓. XML 及其在 MVC 模式中的应用[J]. 计算机技术与发展,2010,20(2):203-205.

[2] 周彩兰,孙　琳,李素芬. 基于 JSP 的网络数据库连接技术[J]. 计算机技术与发展,2006,16(4):209-214.

[3] 樊振宇. 深入理解 SERVLET 和 JSP 原理[J]. 电脑知识与技术,2011,7(11):2570-2573.

[4] 周从军. XML 程序设计[M]. 天津:天津大学出版社,2010.

[5] McLaughlin B. Java and XML[M]. [s. l.]: O'Reiiiy & Assoiates, Inc., 2000.

[6] 陈一明,欧兴灿. 基于 XSL 的 XML 文档转换技术及应用[J]. 湖南科技学院学报,2009,30(8):102-104.

[7] 陈立岩. EJB 组件技术及应用[J]. 计算机技术与发展,2007,17(3):62-64.

[8] Sun Microsystem. Enterprise JavaBeans Specification Version 3.0[R]. [s. l.]: Sun Microsystem, 2005.

[9] 周　强,李　宇,许雁冬. 基于 dom4j 转换 XML 为 XHTML 页面的方法[J]. 计算机技术与发展,2010,20(1):43-45.

[10] 杨少波. J2EE Web 核心技术-XHTML 与 XML 应用开发[M]. 北京:清华大学出版社,2009.

[11] Yu Dachuan, Chander A, Islam N. Javascript instrumentation for browser security[J]. ACM SIGPLAN Notices, 2007, 42(1): 237-249.

[12] 何俊斌. JavaScript 实例精通[M]. 北京:机械工业出版社,2009.