基于 Struts 与 JSF 的 J2EE 集成开发模式的应用研究

敬宗儒，余青松，周 云
（华东师范大学 信息学院，上海 200062）

摘 要：Struts 框架经过数年的发展，已经成为程序员开发 J2EE Web 应用程序的一种事实上的标准。而 JSF 作为 Web 应用程序框架的新贵，其灵活的开发机制和强大的协议无关的用户界面组件呈现方法，正受到越来越多程序员的青睐。通过对这两种技术的分析比较，归纳出利用 Struts－Faces 在 J2EE 框架中整合 Struts 和 JSF 的方法，并通过实例详细阐述了基于 Struts 和 JSF 的 Web 框架开发流程。

关键词：Web 框架；Struts；JSF；Struts－Faces
中图分类号：TP311.52 **文献标识码**：A **文章编号**：1673－629X(2007)03－0214－04

An Integrated J2EE Pattern Based on Struts and JSF: An Application Investigation

JING Zong-ru, YU Qing-song, ZHOU Yun
(Information Institute, East China Normal University, Shanghai 200062, China)

Abstract: For several years, the Struts framework has become the de facto standard that developers build J2EE Web application. And JSF, the new kid in the Web－application－framework family, which provides flexible development mechanisms and a powerful protocol－independent way of rendering UI components, is becoming more and more popular. By analyzing and comparing both technologies, this article concludes a method of integrate Struts and JSF with Struts－Faces integration library, and demonstrates the Web framework development flow based on Struts and JSF.

Key words: Web framework; Struts; JSF; Struts－Faces

0 引 言

近年来，基于 MVC 模式的 J2EE 框架被广泛应用在了 Web 应用程序的开发中。其中，Apache Struts framework 在业界得到了很高的认可，其强大的控制器和业务逻辑层接口为企业应用程序提供了成熟可靠的后台支持。而随着 Web 应用系统的日益复杂化，开发精致的前台用户界面变得越来越困难，开发者开始怀念昔日使用 GUI framework 的美好时光。在此背景下，JSF(JavaServer Faces)应运而生，依靠简洁美观的用户界面开发方式和更加模块化的程序代码，大幅度降低了 Web 应用程序的开发和维护难度，具有十分美好的前景。那么，这两种框架有什么异同呢？如何利用 JSF 强大的前端处理功能，又不摒弃现有的大型 Struts 应用程序，做到各取所长呢？这将是下面要讨论的问题。

收稿日期：2006－06－24

作者简介：敬宗儒(1982－)，男，四川阆中人，硕士研究生，研究方向为 Web 应用技术；余青松，高级工程师，研究方向为计算机系统分析与集成、Web 应用技术。

1 J2EE Web 应用程序框架

1.1 Web 应用程序框架

开源软件的兴起，使得各种各样的框架纷纷出现。在设计模式中，Gamma 等人为框架给出了一个定义："框架就是一组协同工作的类，它们为特定类型的软件构筑了一个可重用的设计。"在 Web 应用中，大部分框架是基于 MVC 模式进行设计的。

MVC 是现今应用非常广泛的一种设计模式，对 MVC 模式的准确定义虽然有很多不同的看法，但基本概念是相同的。它主要包括以下 3 个组件。

(1)模型(model)，负责业务领域的状态知识，包含应用程序的核心功能。一个模型可以为多个视图提供数据，提供了应用的可重用性。

(2)视图(view)，负责业务领域的表示视图。视图向用户显示相关的数据，并能接收用户的输入数据，但并不进行任何实际的业务处理。视图可以向模型查询业务状态，但不能改变模型。

(3)控制器(controller)，负责控制用户输入的流程和状态。控制器接受用户的输入并调用模型和视图去

完成用户的需求。

1.2 Struts 框架

Struts 是在 JSP Model2 的基础上实现的一个 MVC 框架。在 Struts 框架中,模型通常由实现业务逻辑的 JavaBean 或 EJB 组件构成,在模型层,业务逻辑被封装在独立的组件里,并向 Action 类提供接口,这样有助于重用。

控制器由 ActionServlet 和 Action 来实现。ActionServlet 是 Struts 框架的集中控制点,接收所有的客户请求进行最初的处理,并将其映射到适当的 Action 进行处理。Action 类是 Struts 框架的心脏,也是客户请求和业务操作的桥梁。它根据客户请求调用相应的业务逻辑,并根据执行的结果决定程序的去处,以 ActionForward 对象的形式返回给 ActionServlet。

视图通常由 HTML、数据传输对象、Struts ActionForm、JSP 页面、自定义标记和 Java 资源包构成[1]。其中,结合自定义标记库的 JSP 页面构成了视图组件的主体。在 Struts 中共有 5 个定制标记库,它们代替了 Struts 项目在 JSP 页面中大量 Java 代码的逻辑表示,大大简化了 JSP 页面的复杂度。ActionForm 对象用于在用户和业务之间传输客户的输入数据,并提供了对 HTML 表单数据进行验证以及重新设置其属性值为默认值的方法。Struts 框架基本架构如图 1 所示。

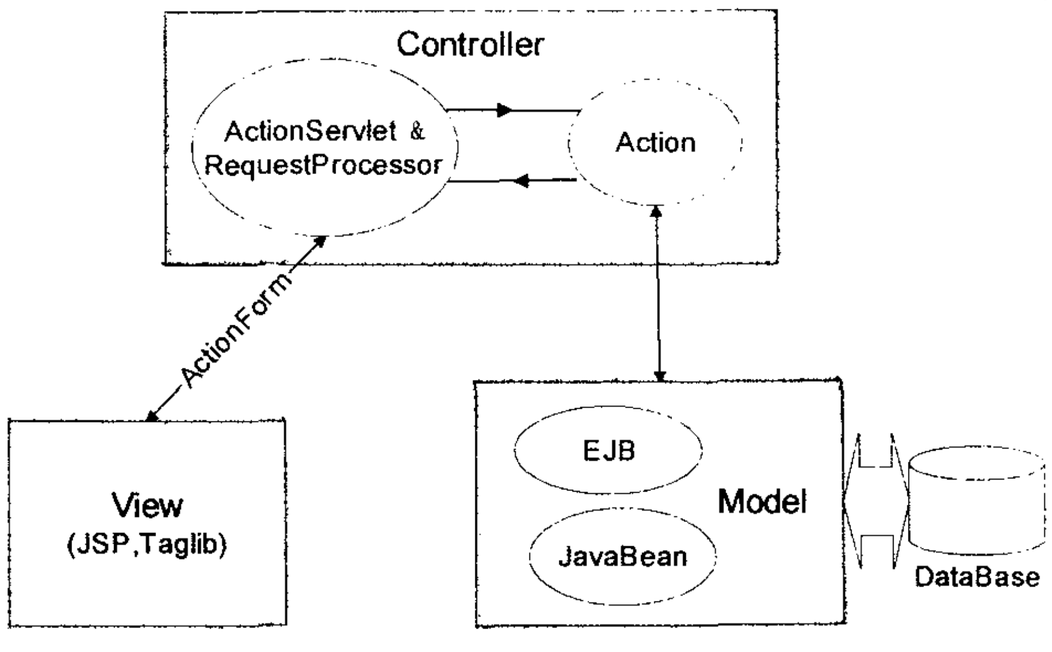

图 1　Struts 框架基本架构

1.3 JSF 技术简介

JavaServer Faces 是由 Java Community Process (JCP)团队开发的类似于传统 GUI framework 的 Web application framework。如果你曾经使用过 Java AWT 或者 Swing API,你将会对 JSF 的 UI 组件比较熟悉。JSF 规范定义了一组 UI 组件以及一组标准的应用程序接口(API)。API 用于扩充原有组件或者开发全新的组件。连接在组件上的 Validator 用于验证用户的输入数据并将其传递给应用对象。用户的动作会触发 event handler,而 handler 可以改变其它组件的状态,或是运行某段后台程序。借由一个可插入的 navigation handler, event handler 可以控制下来要显示哪一个网页[2]。

JSF 对于表示层的包容性非常大,它不仅支持现今非常流行的 JSP 表示层技术,也支持其它的表示层技术,比如说可以用纯 Java 程序来制作的 JSF 视图和以纯 HTML 模板搭配 XML 文件所定义的 JSF 视图。除此之外,JSF 将 UI 组件的表示逻辑和事务逻辑分离,通过在 JSP 页面中使用 JSF 标记,可以将 Render 和 UI 组件关联在一起。组件的呈现方式决定于呈现器(renderer)。renderer 独立于 UI 组件之外,同一个 UI 组件,搭配不同的 renderer,可产生不同的输出。UI 组件运行在服务器端,并响应用户动作产生的 events。

JSF 框架基本架构如图 2 所示。

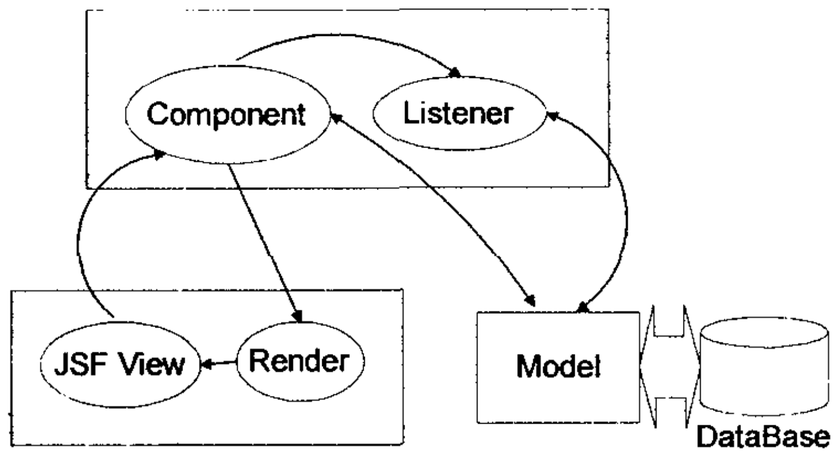

图 2　JSF 框架基本架构

1.4 Struts 与 JSF 的比较

1.4.1 相似点

值得大家注意的一个有趣的现象是,Struts 的开发者 Craig McClanahan 同样是 JSF 技术的主要负责人,因此,这两个应用框架注定有很多的相似点[3]。

第一,在表示层的处理上,两者都使用了 Taglib,都在 JSP 页面中采用一套标记库来处理表示层和模型层之间的交互。

第二,都采用了 JavaBean 来保存 JSP 页面的数据并对其进行验证。在 Struts 中表现为 ActionForm,在 JSF 中表现为 Backing Beans。

第三,都采用 Servlet 作为控制层,来管理页面请求处理周期。在 Struts 中采用 ActionServlet 来作为基于 MVC 模式的控制器,在 JSF 中采用 FacesServlet 来处理用户界面和组件的交互。

第四,都采用 XML 文件来配置 JavaBean 和处理页面导航,增加了系统的灵活性。

第五,都采用资源文件来处理国际化问题。

1.4.2 不同点

两个框架也存在着诸多的不同。

第一,从大的方面讲,两个框架的着重点不一样。Struts 框架的目标在于支持完整应用系统的开发,着重于大结构。它就像一个交通枢纽,接受各种 HTTP request,根据符号名称和各类型应用组件的对应关系

进行映射,再将 view request 传送给负责产生响应视图的程序。Struts 并不在乎如何绘制响应视图,也不管用户的操作是否只会影响视图,因为这类事件不影响后台程序的处理。

而 JSF 的重点放在用户界面的细节,而不在乎应用系统的其它部分是如何运作的。JSF 为 UI 组件定义了一组详细的 API,规范了用户怎样的操作会引发怎样的 UI 事件,而这些事件又应该如何处理,组件如何与它们所要描绘的事务数据链在一起等。

第二,从细节方面讲,主要差异在于 request 的处理控制流。Struts 总是先调用应用逻辑来处理 request,然后才调用个别的表示层片断(JSP 网页)将处理结果转换成 UI widget(也就是绘制相应视图)。另一方面,JSF 则是先调用 UI widget(构成 view 的各个组件)来触发事件,接着由 event handler 调用应用逻辑,然后由 widget 以它们的新值来绘制它们自己[2]。

根据以上的分析,JSF 是能够搭配 Struts 构成更加完善的 J2EE 框架的,在这个框架中,它们发挥各自所长,既为用户界面的设计提供了更富有弹性和便捷的方式,又保持了 Struts 成熟的后台机制。在这种混合模式的开发框架中,基本上,request 都是先交由 JSF 处理,直到确定应该由后台程序接手时,才将控制权交给 Struts,最后再回到 JSF 来绘制响应视图。

2 利用 Struts－Faces 整合 Struts 和 JSF 的 Web 开发流程

2.1 整合所面临的问题

将 Struts 应用与 JSF 整合的挑战主要来自两个方面。

第一,Struts 标签并不适用于 JSF,换句话说,它们并不像 JSF 规范要求的那样继承 UIComponentTag,因此,JSF 不能识别它们并把它们和 UI 组件与 Render 结合起来。

第二,在 FacesServlet 与 Struts 的 RequestProcessor 之间并没有联系的桥梁。在 Struts 应用程序中,RequestProcessor 管理 ActionForm 和 Action 类中的回调方法(callback method)。在 ActionForm 中的回调方法是 Getter,Setter 方法和 Validate()方法。Action 中的回调方法是 execute()方法。除非 RequestProcessor 调用它们,否则这些回调方法将不会有任何机会参与到事务逻辑中去。

2.2 Struts－Faces 简介

此时你也许会想知道是否有什么软件可以帮助整合 Struts 和 JSF,还是需要你自己来写这个整合软件。这里有一个好消息,那就是这种软件已经存在了,那就是 Struts－Faces integration library。它是由 Struts 的创造者 Craig McClanahan 编写的,目的是使将现存的 Struts 应用程序移植到 JSF 变得容易,这样便可最大限度地保留现有的 Struts 程序的价值。Struts－Faces 同时致力于使用 JSF 进行干净的整合,这样 JSF 可被用于前端,而后台仍可保留人们所熟悉的 Struts 组件。

以下是 Struts－Faces 的主要组件[4]:

* FacesRequestProcessor 类,它用于处理所有的 faces 请求。它继承了通常的 Struts 的 RequestProcessor 类并处理 faces 请求。Non－faces 请求被委托给它的父类 RequestProcessor 来处理。

* ActionListenerImp 类,用于处理 ActionEvents,比如说提交表单或者点击链接等。这个类被用来代替 JSF－RI 提供的 ActionListener 的默认实现。只要在 faces 请求中产生了 ActionEvent,则 ActionListenerImp 的 processAction()方法则被调用,随之 ActionEvent 被转发至 FacesRequestProcessor。这点很有趣,因为 RequestProcessor 通常仅由 Struts 的 ActionServlet 来调用处理 HTTP 请求。

* FormComponent 类,继承至 JSF 的 Form 组件,但它是在 Struts 生命周期内被调用的。

* FormComponent 的 renderer 和标签。

* 仅用于显示输出数据的标签和 renderer,这里不需要分离组件。比如,ErrorTag 和 ErrorRenderer 就被用来显示 HTML 的错误信息。

* 名为 LifeCycleListener 的 ServletContextListener 实现,被用于在初始化时注册相应的 RequestProcessor。

* Faces－config.xml 文件。它已经被绑定在了 struts－faces.jar 文件中。

需要注意的是,Struts 与 JSF 之间有部分功能重叠(比如两者都提供了 navigation 与 validation 的机制),但是由于两者都采用弹性设计,所以用户可以自己任选其中之一来承担这些工作。

2.3 Struts－Faces 应用实例

下面将举出一个网上书店书籍查询的例子,以便读者更深刻地理解 Struts 和 JSF 的整合过程。在此集

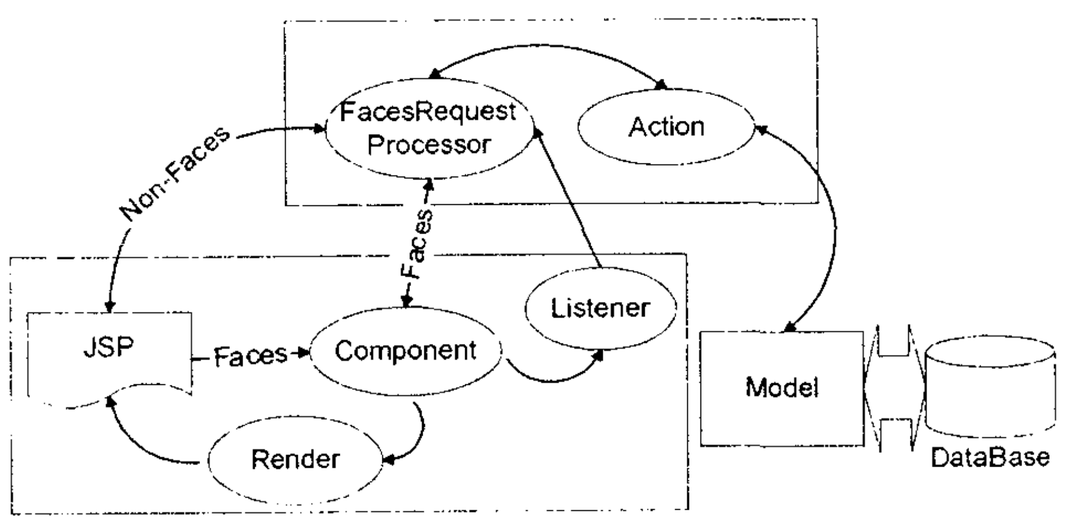

图 3 整合框架的处理流程

成框架中,程序处理流程如图 3 所示。

为简便起见,仅列出主要页面 bookQuery. jsp,在此页面中可以通过输入书名关键字或作者或书籍种类来查询相关书籍列表。首先列出 bookQuery. jsp 的 Struts 版本,如图 4 所示。

```
<%@ page contentType="text/html, charset=Shift_JIS" %>
<%@ taglib uri="/tags/struts-bean" prefix="bean" %>
<%@ taglib uri="/tags/struts-html" prefix="html" %>

<html.html>
    <head>
        <meta http-equiv="Content-Type" content="text/html, charset=Shift_JIS"/>
        <title><bean.message key="bookQuery.title"/></title>
    </head>
    <body>
        <html:form method="POST" action="/demo/bookQuery.do">
            <bean:message key="bookQuery.keyWord"/>
            <html:text property="keyWord"/>
            <bean:message key="bookQuery.author"/>
            <html:text property="author"/>
            <bean:message key="bookQuery.category"/>
            <html:text property="category"/>
            <html:submit ><bean:message key="bookQuery.submit"/></html:submit>
            <html.reset ><bean:message key="bookQuery.reset"/></html:reset>
        </html:form>
        <html:errors/>
    </body>
</html:html>
```

图 4　bookQuery. jsp 页面的 Struts 版本

经过一系列修改,得到了此页面的 Struts－Faces 版本,如图 5 所示。

```
<%@ taglib uri="http://java.sun.com/jst/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-faces" prefix="s" %>

<HTML>
    <f:view>
        <f:loadBundle basename="demo.bundle.Messages" var="Message"/>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=Shift_JIS"/>
            <title>
                <h:outputText value="#{Message.bookQuery.title}" />
            </title>
        </head>
        <body>
            <s:form action ="/demo/bookQuery">
                <h:outputText value="#{Message.bookQuery.keyWord}" />
                <h:inputText value="BookQueryForm.keyWord" />
                <h:outputText value="#{Message.bookQuery.author}" />
                <h:inputText value="BookQueryForm.author" />
                <h:outputText value="#{Message.bookQuery.category}" />
                <h:inputText value="BookQueryForm.category" />
                <h:commandButton  value="#{Message.bookQuery.submit}" type="submit"/>
                <h:commandButton  value="#{Message.bookQuery.reset}" type="reset"/>
            </s:form>
            <s:errors/>
        </body>
    </f:view>
</HTML>
```

图 5　bookQuery. jsp 页面的 Struts－Faces 版本

通过比较,发现两个版本的主要不同集中在以下几点:

第一,引入了一个新标记库 tags－faces。这个标记库定义声明这些标记由 Struts－Faces API 使用,添加了<f:view>标记,它容纳了 view 中其它组成组件。必须确定它包含了网页中所有的其他 JSF 标记。另外,<html:html>标记改为<html>标记。

第二,修改了消息资源包的声明。在 Struts 程序中,由<bean:message>引入资源包中的相关信息,但 JSF 组件并不支持这一标签。故加入<f:loadBundle>来加载资源附件,它的 basename 属性指定了相关的资源包。随后,用<h:outputText>取得资源包中信息。除此之外,还可以直接引用<s:message>标签来直接引用消息资源包的资源。

第三,用<s:form>标签替代原来的<html:html>。注意,属性 action="/demo/bookQuery" 既不同于 Struts 版本的 action="/demo/bookQuery.do",又不同于 JSF 通常的制定表单名的形式。这是因为 Struts－Faces 使用表单 action 本身作为表单名(它还应该与 Struts 配置文件中的 ActionForm 名相匹配)。而在 JSF 中,表单名只是指定给 UI Form 的名字而没有更多的意义。通过图 6[5]可以看出,ActionForm (BookQueryForm)本身就是 Struts－Faces 中的 html 表单模型,它的 action 间接地指向这个模型。这与 JSF 的每个表单都有一个对应模型的要求相符合[4]。

第四,对 form 中每个字段都使用值绑定表达式。在 Struts 中,form 中的字段与 ActionForm 的属性一一对应,而 JSF 不知道如何找到 Form Bean 的定义,故必须在值绑定表达式中明确地写出来,如<h:inputText value="BookQueryForm.keyWord" />。

第五,应该注意到,在 Struts－Faces 版本中,并没有使用 JSF validation 标记,这是因为在 Struts 中,验证是在 ActionForm 类中的 validate 方法中进行的,也有可能是使用 commons－validator。<s:errors>标记类似于 Struts 错误标记并用于显示在验证时出现的错误消息。

第六,将 Struts 按钮标签改成相应的 JSF 标签。但是注意到,没有 ActionListener 显式地与提交按钮相关联。这是因为在 Struts－Faces 中已经提供了 ActionListener 并且总是将 faces 请求与 ActionEvents 一同转交给 FacesRequestProcessor,在那里根据 struts－config. xml 文件将请求分派给相应的 Action 类。

第七,除此之外,还应该在 web. xml 的适当位置添加 FacesServlet 和 servlet－mapping 声明。

```
<!– struts-config.xml –>
<form-bean  name="bookQueryForm "  type="demo.BookQueryForm "/>
<action-mappings>
    <action path="/demo/bookQuery "
            type="demo.bundle.BookQueryAction"
            name="bookQueryForm "
            scope="request"
            validate=" true "
            input=" /faces/bookQuery.jsp " >
        <forward name ="success" path="/faces/bookQueryList.jsp"/>
    </action>
</action-mappings>
```

```
<!– web.xml –>
<servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping >
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
</servlet-mapping>
```

图 6　Struts－Faces 版本的相关配置信息

经过以上的改动,在不变动 Struts 框架的 ActionForm 和 Action 的情况下,新的集成了 JSF 的混合框架

(下转第 221 页)

3　ASI 索引结构性能对比分析

为了检测 ASI 的性能指标,测试方案使用 IBM 开发的城市模拟器(City Simulator)[11]生成测试数据。城市模拟器是一个可扩展的、可以模拟三维城市的工具,使用 XML 文件描述公路、建筑物、公园等地形地物信息,该模拟器最大的特点是其交通流量控制模型可以产生较真实的流量数据,比较适合测试在日常交通环境中各类索引算法的效率[11]。如表 1 所示,受篇幅所限仅仅给出各种索引类型在不同查询区域比例下的空间查询响应时间数据,从表中可以看出 ASI 树和其他空间索引结构相比具有明显的性能优势,尤其体现在查询范围比较大的时候。

表 1　多种索引结构查询响应时间对比(ms)

索引类型 / 响应时间 / 查询区域比例	ASI	TB	STR	3DR	MV3R	HR
5%	71	605	712	510	110	652
10%	82	802	810	620	135	807
15%	156	1611	1501	1302	155	1532
20%	211	2102	1800	1425	281	1723

4　结束语

文中所做的主要工作有:

①提出了一个适合对存储和索引移动空间对象的时间片、时间段以及路径查询的数据结构;

②采用的复用技术有效地减少了磁盘存储空间的需求;

③基于 ASI 索引结构的查询速度比基于类 R 树的层次空间索引结构有所提高。

接下来需要完成的工作是使用预先提取数据的策略进一步解决系统查询的效率,目前 CELL 网格内存储是关于移动对象在某个时间点的位置信息,通过磁盘存储结构的设计以进一步存储折线段数据,这样可以大大提高支持在采样时间间隔的"瞬时"查询类型。

参考文献:

[1] 吴信才.地理信息系统原理与方法[M].北京:电子工业出版社,2002.

[2] Guttman A. R－trees: A Dynamic Index Structure for Spatial Searchin[J]. Proc ACMSIGMOD, 1984(6):47－57.

[3] 郭仁忠.空间分析[M].武汉:武汉测绘科技大学出版社,1997.

[4] 顾　军,吴长彬.常用空间索引技术的分析[J].微型电脑应用,2001,17(12):40－42.

[5] 徐少平,徐少文,罗　洁.XBR 树:一种基于四叉树的空间移动对象移动路径索引结构[J].现代计算机,2005(6):9－12.

[6] Song Zhexuan, Roussopoulos N. Hashing Moving Objects [C]//In Proceedings of the 2nd International Conference on Mobile Data Management. Hong Kong, China:[s. n.],2001: 161－172.

[7] Tayeb J, Wolfson O. A quadtree－based dynamic attribute indexing method[J]. The Computer Journal, 1998, 41(3):185－200.

[8] Tao Y, Papadias D. Mv3r－tree: a spatio－temporal access method for timestamp and interval queries[C]//In Proc. of the Intl. Conf. on Very Large Data Bases, VLDB. Rome, Italy: [s. n.],2001:431－440.

[9] Saltenis S, Jensen C, Leutenegger S, et al. Indexing the position of continuously moving objects[C]//Proc. ACM SIGMOD Intl. Conf. on Management of Data. Dallas, Texas, USA: [s. n.],2000.

[10] Prabhakar S, Xia Y, Kalashnikov D, et al. Query indexing and velocity constrained indexing: scalable techniques for continuous queries on moving objects[J]. IEEE Transaction on Computers, 2002, 51(10):1124－1140.

[11] Myllymaki J, Kaufman J. A Testbed for Dynamic Spatial Indexing[M]. Almaden: IBM Almaden Research Center, 2002.

(上接第 217 页)

程序已经能够正确运行了。

3　总　结

Web 应用经过数年的发展,已经涌现出了许多优秀的框架,然后,如果在各个框架中取长补短,也是需要面临的问题。经过相对较小的修改,我们成功地利用 Struts－Faces 集成了 Struts 和 JSF,并保留了前期在 Struts 项目上所做的投资。通过演示,证明了可以将 JSF 强大的前端能力和 Struts 控制器层的灵活性结合在一个包中,使得创建一个 J2EE Web 应用程序更加快捷、规范。

参考文献:

[1] Cavaness C. Programming Jakarta Struts[M]. 2nd ed. [s. l.]:O'Reily Media, Inc, 2005:179－181.

[2] Bergsten H. JavaServer Faces 交互式网站界面设计[M].[s. l.]:O'Reily Media, Inc, 2006:7－15, 243－250.

[3] 杨　浩.基于 Struts 和 JSF 技术的中间件的研究与设计[C/OL].In:中国优秀博硕士学位论文全文数据库.[出版地不详]:[出版者不详],2005:1－43.

[4] Shenoy S, Mallya N. Integrating Struts, Tiles and JavaServer Faces[C]// In World Wide Web: developerWorks. [s. l.]: [s. n.],2003:7－9.

[5] 王冠宇,赵冬生.在 J2EE 应用程序中整合 JSF 与 STRUTS[J].微计算机信息(管控一体化),2006,22(2－3):3－4.