

基于 Apusic Studio 平台的 JSF 开发研究

吴教育¹, 熊君丽¹, 朱小平¹, 钟雷², 张勇²

(1. 广东科学技术职业学院 计算机工程技术学院, 广东 珠海 519090;

2. 深圳市金蝶中间件有限公司, 广东 深圳 518100)

摘要:JavaServer Faces(JSF)是一种新兴的 Web 表现层框架,它超越了 JavaServer Pages (JSP),在页面内提供了真正的服务器端事件处理,并提供了基于组件的、可以跨多个服务器请求生存的页面。Apusic Studio 作为 J2EE 的集成开发平台,对 JSF 实现了全面支持,包括界面导航、拖放式设计、事件管理机制,以及部署文件的自动生成等,另配以语法加亮、代码导航、断点调试、可视化的设计等辅助功能。

关键词:JSF; Apusic Studio; 事件管理机制; 定制 UI 组件

中图分类号:TP311

文献标识码:A

文章编号:1673-629X(2008)02-0108-04

JSF Development and Research on Basis of Apusic Studio Platform

WU Jiao-yu¹, XIONG Jun-li¹, ZHU Xiao-ping¹, ZHONG Lei², ZHANG Yong²

(1. Sch. of Eng. and Techn., Guangdong Inst. for Techn., Zhuhai 519090, China;

2. Shenzhen Kingdee Middleware Co., Ltd., Shenzhen 518100, China)

Abstract:JavaServer Faces(JSF) is a new kind frame of Web presentation. It exceeds JavaServer Pages(JSP) and provides genuine incident process based on server in the page, also the page component-based of which requests can survival across multiple servers. Being a stage of intergration and development platform for J2EE, Apusic Studio implements a complete support for JSF, including interface navigation, drag-and-drop design, incident management mechanism and automatic generation of deployment document, furthermore assistant functions including expression highlight, code navigation, breakpoint debugging, visual design and so on.

Key words:JSF; Apusic Studio; incident management mechanism; custom UI component

0 前言

J2EE 下一代规范 Java EE 5.0 中, SUN 已经提议将 JSF(JavaServer Faces)作为构建 JavaWeb 的首选技术。J2EE 社区将通过 JSF 来统一 Web 应用的开发模式与方法。JSF 通过提供模型-视图-控制器设计模式的一个简洁实现,在不牺牲开发能力和灵活性的前提下提供高效的以组件为中心的开发,解决了 Java Web 开发的许多历史问题。此外,JSF 是一种 Java 标准,因此多个软件供应商将继续提供始终高效的开发环境,这些开发环境毫无疑问将达到或很可能超过专有的可视化开发环境。

基于 JSP/Servlet 的开发模型,JSF 通过提供下列特性解决了 Web 开发过程中的许多实际问题:可扩展

展、可定制的丰富的 UI 组件,方便的页面导航,良好的事件响应机制,JavaBean 管理,表单数据的自动转换与验证,方便的错误处理,国际化支持等等。

相对于提高用户体验的 Ajax (Asynchronous JavaScript and XML) 技术来说,JSF 保留了它的优势,并且减少了数据传输量,保持了系统的敏捷高效性,同时提供了可扩展的框架级解决方案^[1]。

业界常见的开发 JSF 的组件及开发构建 JSF 应用的辅助工具普遍存在着一些不足之处,是常规的实现机制所带来的问题。这些缺陷可能对 JSF 成为日后 Web 应用主流开发技术带来一定的负面影响,如:运行期性能并不是非常理想;缺少良好的工具支持等。

作为 JCP 组织成员,Apusic Studio 的 JSF 平台依赖于 Web Tools Platform(WTP)以及 Eclipse JDT,对 JSF 技术进行了全面支持。包括管理 JSF 特定的配置文件,编辑 HTML, JSP 和 XML 文件,并提供有语法加亮、代码导航等辅助功能。

收稿日期:2007-05-08

基金项目:粤港关键领域重点突破招标项目(A10207008)

作者简介:吴教育(1964-),男,江西都昌人,硕士,副教授,研究方向为软件工程、计算机安全与维护。

1 设计中的辅助功能支持

1.1 页面导航功能

Apusic Studio 增加了对 JSF 配置文件 `faces-config.xml` 进行编辑和图形化设计的编辑器,这个编辑器包含了三个标签:页面导航标签,Managed Bean 标签,源代码标签。

页面导航标签中提供了对 JSF 页面流转的设置,可以从左边的导航器拖动页面到编辑器,制定页面导航方向以及导航规则,并可以通过右键菜单和属性视图来编辑页面或者导航线。在 JSF 中不仅使用了 POJO 技术,而且还使用了类似 Spring 的控制反转 (IoC) (或称为依赖注入 - DI) 技术,在 JSF 的 Backing Bean 中,可以把视图所需要的数据和操作放进一个 Backing Bean 中。同时得益于 JSF 使用的 DI 技术,可以在配置文件中初始化 Managed Bean,同时也可以通过这样的技术很方便地和使用类似技术的 Spring 进行整合^[2]。Managed Bean 标签提供对 Managed Bean 的设置,包括:名称(将在 Web 页面中被使用)、引用类、作用域。源代码标签提供对应用配置文件 `faces-config.xml` 源码的修改并且自动与其余两个编辑器保持同步。Apusic Studio 提供了非常丰富的标签库,通过这些标签库,可以生成各种客户端模型,如 HTML, WML, XML 以及 JavaScript 等。通过这些标签,很容易建立大规模的客户端模型,并由这些标签自动处理客户端请求。如图 1 中显示标签库 JSF Core, JSF Facelets, JSF HTML^[2]。

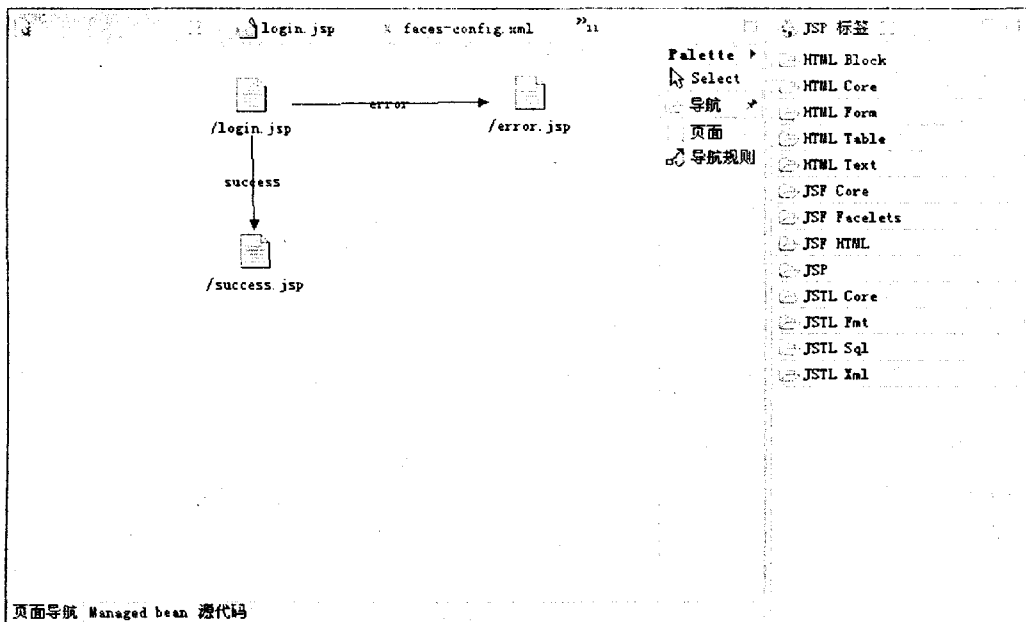


图 1 Apusic Studio JSF 设计器

1.2 使用拖放式组件

Apusic Studio 提供了丰富的可自定义的 HTML,

JSF, JSP 以及 JSTL 的拖拉式组件。通过拖拉操作将 JSP 标签工具栏中的组件加入页面代码中。使用属性视图图形化设置标签属性值,值绑定时可以选择 Managed Bean 中属性并且实现数类型过滤匹配。即使前台 JSP 页面与 Managed Bean 之间的事件驱动也可以通过代码导航来实现(见图 1)。

2 JSF 事件管理机制

JSF 应用是事件驱动的表现层框架,是一种事件驱动型的组件模型^[3]。它既符合模型 - 视图 - 控制器 (MVC) 模式又遵循 Java2 事件模型,该模型中涉及三个参与者即事件源、事件对象和事件监听器。事件源通过广播将发生的事件对象发给感兴趣的事件监听器。在 JSF 应用中,UIInput 或 UICommand 组件都可以作为事件源。

JSF 的事件可以分成两种类型:阶段事件 (Phase Event) 和 Faces 事件 (Faces Event)。阶段事件用来处理请求生命周期的特定阶段中发生的事件。Faces 事件则是负责处理组件事件的。因为 Web 通过没有状态的 HTTP 协议来进行请求和响应,所以 JSF 使用阶段事件来处理。JSF 生命周期中有 6 个阶段:Restore View, Apply Request Value, Process Validators, Update Model Values, Invoke Application, Render Response。阶段事件监听器需在配置文件定义,实现 PhaseListener 接口的 `afterPhase`, `beforePhase`, `getPhaseId` 三个方法。`afterPhase` 在执行某个阶段之后被调用, `beforePhase` 方

法则反之, `getPhaseId` 被 JSF 调用以决定在哪个阶段执行它们。

Faces 事件可以分成两种类型:值变事件 (Value Change Event) 和动作事件 (Action Event)。值变事件关注的是组件属性值的变化(例如展开树节点,改变输入栏中的内容等等)。事件监听器类实现 Value Change Listener 接

口监听发生的值变事件,触发该接口的 `process Value Changed` 方法。事件源 UIInput 组件要实现监听,可以

拖拉放置 JSF Core 标签库中 value Change Listener 标签成该组件标签的子标签,对 type 属性进行设置,如图 2 所示。

动作事件主要关注组件的激活,其实就是组件的状态(例如按钮点击,超链接点击等等)。同值变事件类似,事件监听器类实现 ActionListener 接口监听发生的值变事件,触发该接口的 processAction 方法。事件源 UICommand 组件要实现监听,使用 JSF Core 标签库中 actionListener 标签,对 type 属性进行设置。

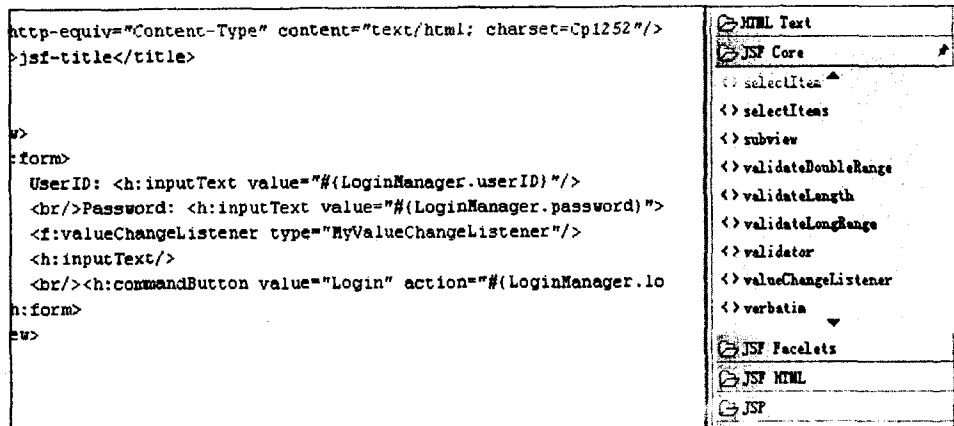


图 2 值变事件设计

对于简单的封装业务逻辑的 javabean,可以无需构造复杂繁琐的监听器类,直接将 UIOutput 组件与 javabean 的方法绑定实现监听。

3 定制 UI 组件

JSF 拥有一个与 AWT 的 GUI 组件模型类似的组件模型,可以用 JSF 创建可重用组件和复合组件。设计内容非常简洁,由三个因素构成:接近于组件功能的标准 UIComponent 的子类、呈现子类,以及用于驱动呈现响应阶段的标记处理类。并在标记库描述符文件和包含组件相关信息的元数据文件中说明定制组件^[4]。

组件的功能通常围绕着两个动作:解码和编码数据。解码把进入的请求参数转换成组件的值,在 Apply Request Value 阶段执行。编码在 Render Response 阶段把组件的当前值转换成对应的标记。JSF 框架提供了两个选项用于编码和解码数据。使用直接实现方式,组件自己实现解码和编码。使用委托实现方式,组件委托渲染器进行编码和解码。如果选择委托实现,可以把组件与不同的渲染器关联,会在页面上以不同的方式渲染组件。因此,JSF 组件由两部分构成:组件和渲染器。JSF 组件类定义 UI 组件的状态和行为,渲染器定义如何从请求读取组件、如何显示组件——通常通过 HTML 渲染。渲染器把组件的值转换成适当

的标记。所有 JSF 组件的基类是 UIComponent。在开发自己的组件时,需要继承 UIComponentBase,它扩展了 UIComponent 并提供了 UIComponent 中所有抽象方法的默认实现,另需保存组件状态且在 faces-config.xml 登记组件。然后定义渲染器或者内联地实现它,覆盖 encode 和 decode 方法,用 faces-config.xml 登记渲染器。最后创建定制标记,继承 UIComponent-Tag,返回渲染器类型和组件类型,设置可能使用 JSF 表达式的属性^[4]。

呈现子类有两个作用。第一,呈现程序负责发送适当的 HTML 程序段,该程序段能在客户端中呈现组件。通常情况下,这个 HTML 程序段由一些适于呈现整个 Web 浏览器的 HTML 标签组成。此 JSF 生存周期称作编码阶段或呈现—响应阶段。该响应阶段还能发送增强客户端

交互性的 JavaScript 代码。呈现程序的第二个作用是解析来自客户端的数据,从而对服务器端的组件状态进行更新(如用户在文本字段输入的文本)。标准呈现程序软件包具有强制性,但也可以提供其他呈现程序软件包,用于提供可替换的客户端表示法或 SVG 之类的语言。通过检验组件的连接属性,您实现的呈现程序将选择在 HTML 页面中发送的 CSS 样式。

Apusic Studio 支持定制 UI 组件功能,UI 组件可以采用自我呈现和呈现委托两种实现方式。JSF 引擎能正常解码和编码定制 UI 组件。

4 容器级别的 Ajax 支持

Ajax 提供与服务器异步通信的能力,从而使用户从请求/响应的循环中解脱出来。Ajax 的核心是 JavaScript 对象 XmlHttpRequest,该对象是一种支持异步请求的技术。JSF 有三个主要的 Ajax 集成策略。一是把 Ajax 支持加到现有组件中,二是把 Ajax 支持直接集成到 JSF 组件中,三是对现有 Ajax 控件封装成一个 JSF 组件。Apusic 采用 JSF 容器处理 Ajax 请求。Faces Request 发出一个 Ajax 请求,该请求到达服务器端以后,Apusic 服务器端所产生的 Faces Response 的 Content-Type 不再是 text/html,而是了 text/javascript。同时,响应的主体也不再是 html 文本,而是一堆 script 脚本。浏览器在接收到响应以后,再也

不需要进行整个页面的渲染与刷新,而仅仅需要执行这段脚本内容,将页面的控件进行更新即可^[4]。

而且所有的标准组件都支持 Ajax,对于不支持 Ajax 的第三方组件,可以使用<ajax:renderGroup>的标签转换。Apusic JSF 引擎实现了一个 Ajax Render Kit,在 HTML 文档中嵌入 Java Script 代码来实现 Ajax 特性呈现,而替换 Render Kit 只需要修改配置文件^[5]。

Apusic JSF 实现了一个<ajax:status>的标签,用于接受发送和完成 Ajax 请求时触发的事件。缺省的实现是在发送 Ajax 请求之前显示一个 HTML 片段,在完成 Ajax 请求之后显示另一个 HTML 片段。开发人员可以设置<ajax:status>标签的 onStart 和 onStop 属性,来自定义修改执行的 Java Script 代码以实现更复杂的效果。此外,还实现了一个<ajax:invoke>标签,能采用 RPC(Remote Procedure Call)方式调用服务器端 Java 对象的方法^[5]。

5 其它特性

Apusic JSF 还包含其它特性,如控件的换肤功能,控件对 IE、Mozilla(Firefox)、Opera 等多浏览器的支持,以及强大的布局功能等^[5]。

6 结束语

JSF 事实上与 MVC 模式有所不同,它是 Web 开发。编程人员的开发理念更为轻松。该模型使应用程序开发人员能够设计应用程序的页面流。与 Struts 的

方式类似,所有的页面流信息都定义在 JSF 配置 XML 文件(faces-config.xml)中,而非硬编码在应用程序中。这很大程度简化了开发人员开发难度,简化了应用程序的开发。

JSF 自定义组件由 java 代码和 tag 库文件组成,开发难度应该与现有 J2SS 组件开发的难度基本一致。自定义组件通过自定义标记构造页面,在页面上增加组件的数量会对性能有较大的影响。在定制组件功能上,Apusic Studio 可以考虑增加设计导航功能,并使应用配置文件与组件功能和呈现设计保持同步。

JSF 是可高度抽象、可扩展的 Web 表现层框架级解决方案。但是受客户端浏览器技术的限制,并未达到无缝、快捷的用户体验。JSF 的组件和事件模型与 Ajax 的异步通讯方式相配合,优势互补,会使互联网客户端应用环境界面更加友好便捷。

参考文献:

- [1] Crane D. Ajax 实战[M]. 北京:人民邮电出版社,2006.
- [2] Harrop R, Machacek J. Pro Spring 中文版[M]. 夏 昕译. 北京:电子工业出版社,2006.
- [3] Geary D. A first look at JavaServer Faces, Part 1[EB/OL]. 2002-11-29. <http://www.javaworld.com/javaworld/jw-11-2002/jw-1129-jsf.html?page=1>.
- [4] Kurniawan B. JavaServer Faces Programming[M]. 北京:清华大学出版社,2005.
- [5] 袁红岗. JavaEE without Ajax[EB/OL]. 2007-05-28. <http://www.apusic.com/article/article19.htm>.

(上接第 107 页)

以达到效率和准确性的平衡。

参考文献:

- [1] Casati F, Llnicki S, Jin L, et al. Adaptive and Dynamic Service Composition in eFlow[C]//Proceedings of the 12th International Conference on Advanced Information Systems Engineering, 2000. London: Springer Verlag, 2000: 13-31.
- [2] Zeng L, Benatallah B, Nguyen P, et al. AgFlow: Agent-based Cross Enterprise Workflow Management System[C]//Proceedings of the 27th Intl. Conf. on Very Large Data Bases, 2001. Italy: Morgan Kaufmann Publishers Inc, 2001: 697-698.
- [3] Benatallah B, Dumas M, Sheng Q. The Self-Serv Environment for Web Services Composition[J]. IEEE Internet Computing, 2003, 7(1): 40-48.
- [4] 蒋运承, 杨 庸. 服务组合的质量估计模型[J]. 小型微型计算机系统, 2006, 27(8): 1519-1524.
- [5] Berardi D, Calvanese D, Giacomo G, et al. Automatic Composi-

tion of E-Services That Export Their Behavior[C]//Proceedings of the 1st Intl Conf on Service-Oriented Computing, 2003. Berlin: Springer Verlag, 2003: 43-58.

- [6] Benatallah B, Dumas M. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services[C]//Proceedings of the 18th Intl. Conf. Data on Engineering, 2002. Washington: IEEE Computer Society, 2002.
- [7] Hamadi R, Benatallah B. A Petri net-based model for web service composition[C]//Proceedings of the 14th Australasian Database Conf, 2003. Darlinghurst: Australian Computer Society, 2003: 191-200.
- [8] Narayanan S, McIlraith S A. Simulation, Verification and Automated Composition of Web Services[C]//Proceedings of the 11th intl Conf on World Wide Web, 2002. New York: ACM Press, 2002: 77-88.
- [9] Hashemian S V, Mavaddat F. A Graph-Based Approach to Web Services Composition[C]//Proceedings of the 2005 Symposium on Applications and the Internet, 2005. Washington: IEEE Computer Society, 2005: 183-189.