摘　要

随着 Internet 的飞速发展和普及，Web 应用呈现出多功能化发展趋势，同时对 Web 应用框架支持多界面性、可扩展性、易维护性提出越来越高的要求。然而，目前面向过程的 Web 应用技术很难适应新的变化。本文针对目前 Web 应用框架存在的上述不足，基于 Struts 框架提出了一种改进方案，并提出了一个方便的 Web 应用框架配置文件管理器。

本文的主要工作如下：

(1)针对 Struts 框架的不足，在 Web 应用框架中应用面向对象技术及 MVC 设计模式，并对框架实现的关键技术作了详细阐述。

(2)分析了 Struts 框架的优点和不足及多类型用户界面的支持机制，采用 XML 和 XSL 技术，对 Struts 控制器的视图支持部分进行改进，初步实现了控制器对多类型用户界面的支持。

(3)针对 Struts 配置文件不易配置的不足，设计并实现了一个基于 JMX 技术的 Web 应用框架的配置文件管理器，此管理器具有可动态热插拔及灵活方便等特点，使对复杂配置文件的修改操作得以简化。

(4)开发了一个网上数码冲印系统，验证了本文的研究结论。

【关键词】MVC，Web 应用框架，多类型用户界面，控制器，JMX

ABSTRACT

With the development and the popularization of Internet at very fast speed, web application presents the multi-purpose tendency, and that puts forward the more and more high demand to the support of multi-view user interface, the expandability and easy maintenance of web application framework. However at present the procedure-oriented programming of web application techniques are hard to adapt to the new changes. Aiming at the problem of web's application framework presently, this thesis proposed the improving solutions based on Struts framework and a convenient configuration file's manager of web application framework.

The main work of this thesis as follows:

(1)In view of Struts framework's deficiency, applying the Object Oriented programming technique and MVC design pattern, analyzing detailedly the every tiers' key realization technique of framework.

(2)Analysing the merit and deficiency of Struts frame and support mechanism of multi-view user interface, adopting XML and XSL techniques, improving on the view part of Struts controller, realizing support of multi-view user interface in the controller.

(3)Aiming at the hard configured deficiency of Struts' configuration file, designing and realizing a configuration file's manager based on JMX technique, it has characteristic of dynamic remove and installation and convenient flexible management , which convert the complicated modification to the configuration file into the simpleness.

(4)Designing a system of digital photocopy in the internet and verifying the research conclusion of this thesis.

Keyword: MVC, Web application frame, multi-view user interface, controller, JMX

第一章 绪论

1.1 课题研究的背景

随着 Internet 的出现和飞速发展，信息技术发生了翻天覆地的变化，计算机应用的范围也在不断的扩展。

Web 应用是当前 Internet 上使用极其广泛的应用开发技术，它结合了数据库的运用，支持实时的信息发布，动态的用户交互以及与后台系统灵活安全的链接。因此，如何构造一个既能够响应大量的客户端用户，又能够安全稳定地运行，同时功能强大，应用灵活，开发简便的 Web 应用程序是当前 Internet 技术发展的热点之一。

目前，应用较为广泛的 Web 应用程序开发技术是在传统的 Web 服务器端增加一个数据库服务器，通过应用程序或脚本程序根据用户请求从数据库中提取相应的信息。一些先进的 Web 应用引入了模型－视图－控制器(Model-View-Controller，MVC)设计模式，以分离控制和模型混合的代码。

然而，现有的 Web 应用技术尽管有很多优点，但也存在以下问题：

(1) 现有的 Web 应用技术欠缺对多类型客户端应用的集成。无线网络的应用和 Web Services 技术在飞速的发展，现有的 Web 应用并没有集成这些应用，需要改进其对多类型客户端应用的集成，以满足日渐突出的需求。

(2) 程序不易维护和扩展。Web 应用的更新速度加快，应用中的信息也相应更新频繁，且涉及多个方面和层次，而代码的不分离性降低了程序的可维护性、可扩展性。

(3) 对 Web 应用程序配置文件的管理较复杂。目前一些先进的 Web 应用技术引进了 MVC 设计模式，实现了 Web 应用程序的复用和控制视图代码的分离，并使用配置文件方便了对 Web 应用视图和模型文件的集中管理，但随之而来的是配置文件的庞大，难于管理。

综上所述，对现有的 Web 应用框架进行改进，分离 Web 应用中的界面表示、流程控制和业务逻辑的实现，支持多类型客户端的需求，增加对配置文件的易管理性，变得日益迫切起来。如何规划 Web 应用框架，以及如何开发 Web 应用已成为开发者关注的焦点。

1.2 本文解决的问题

由于对 Web 应用可扩展性、易维护性及支持多类型用户界面的需求日渐突

出，本文分析研究了 MVC 设计模式及其应用于 Web 应用的优势，并在基于 MVC 模式的 Web 应用框架 Struts 框架控制器的基础上加以改进，使其能检测多种客户端的请求，并支持多类型用户界面的显示。扩大了 Web 应用框架的灵活性。

基于 Struts 配置文件的实现机制，加入了对框架视图文件的集中配置。并基于 JMX 的简单灵活性，创建了配置文件的管理器。此管理器可动态实现热插拔，对原有代码无需改动，即可实现对配置文件资源的管理，使管理系统与被管理系统做到了很好的隔离，增加了配置文件的可管理性及易维护性。

1.3 本文的主要研究内容及组织

1.3.1 本文的主要研究内容

本文的研究内容主要有两点：

首先分析了 Struts 框架的优点和不足，针对 Web 应用出现的新特点，在目前流行的基于 MVC 模式 Struts 框架控制器的基础上研究了支持多类型用户界面的实现方法，扩展了 Struts 框架对于多类型用户界面的支持。使控制器能够根据用户请求类型，自动调用相应的样式表生成结果视图文件响应给用户。

其次基于 Struts 配置文件的实现机制，在控制器中通过配置文件实现 XML 文件的集中配置，但随着 Web 应用规模的扩大，配置文件也会变得极其庞大，难于配置和管理，针对此问题，本文提出并创建了一个基于 JMX 技术的配置文件的管理系统，此管理系统可动态实现热插拔，不用修改原有代码，即可实现对配置文件的灵活方便的管理，使对复杂的配置文件的改动变得简单方便。

最后设计开发了一个电子商务的实例－网上数码冲印系统，验证了本文的研究。

1.3.2 论文组织

论文分为五个部分来阐述：

第一章 绪论。论题的研究背景、主要研究内容及全文的组织结构。

第二章 基于 MVC 的 Struts 框架分析。分析 Web 应用构架及 MVC 设计模式，探讨 Struts 框架的实现及其优缺点，提出本文的改进思想。

第三章 Struts 框架控制器的改进。分析多类型用户界面的特点，指出多类型用户界面的支持机制。在 Struts 框架控制器的基础上改进控制器的设计，实现控制器对多类型用户界面的支持，在 ViewRouter 中实现了检测用户请求类型并调用相应的样式表进行显示。

第四章 基于 JMX 的 Web 应用框架配置文件的管理。针对配置文件存在的缺点，提出并创建了一个基于 JMX 技术的管理器，此管理器可动态实现热插拔，不用修改原有代码，即可实现对配置文件的灵活方便的管理，使对复杂的配置文件的改动变得简单方便。

第五章 Web 应用框架的实现：网上数码冲印系统。以网上冲印系统为例，设计并实现本文的研究结果。

第六章 总结和展望。对全文进行总结，说明有待进一步研究的课题，并对未来的技术发展趋势进行讨论。

第二章　基于 MVC 的 Struts 框架分析

随着 Web 应用的发展，Web 应用模型经历了三个发展阶段。本章首先分析 Web 应用模型各个发展阶段的优缺点，探讨了 MVC 设计模式及组件技术应用于 Web 的优势，然后分析了 Struts 框架的实现机制及其优缺点，重点分析了 Struts 框架对多类型界面支持的不足及配置文件难于管理的缺点，指出了需要改进之处。

2.1Web 应用模型与框架

2.1.1Web 应用模型

由于本文关注 Web 应用框架的设计与实现，首先讨论 Web 应用技术发展过程中各阶段的模型。

(1)静态的 Web 应用模型。服务器端基本上只有 Web 服务器构成，服务器要发布的内容以文件的形式保存在 Web 服务器上，只能通过 HTML 文件提供静态的 Web 内容，所有的服务内容必须预先定义编辑好，用户可以通过 URL 直接定位到 HTML 文件进行存取。这一模型比较简单，可靠性较高，实现起来较容易，但是提供的内容比较单调，并且时效性及可维护性较差，现在大的网站系统已很少采用。

(2) 交互式的 Web 应用模型。此阶段分为两种应用模型：基于两层应用的交互式应用模型和基于多层应用的交互式应用模型。

基于两层应用的交互式应用模型如图 2-1 所示。它在服务器端增加了一台数据库服务器，将要发布的信息分类保存到数据库服务器，然后通过应用程序或脚本程序根据用户请求提取相应的信息。动态发布模型能为用户提供动态信息的即时发布服务，保证信息的时效性。但是，支持数据库服务器，动态发布模型的 Web 服务的负担过重，降低了 Web 服务器的稳定性。此种模型可通过 ASP、JSP、PHP 等脚本语言来实现。它是当前应用比较多的一种模型。

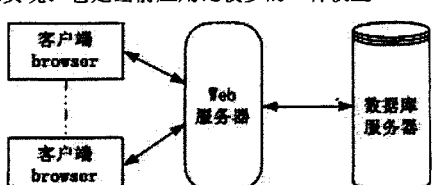


图 2-1　基于数据库服务器的动态发布模型

基于多层应用的交互式应用模型是一种分布式的 Web 应用，其在 Web 服务器和后台数据库服务之间增加了中间层应用服务器，这种比较先进的模型被广泛应用于大型网站。它优点是能够动态地发布信息，保证了信息的时效性，并且由于增加了中间应用服务器，可以将一些复杂的企业逻辑及数据库的连接服务封装到中间层的应用服务器上，通过中间层来完成发布，减轻了 Web 服务器的负担。该模型通过 ASP 脚本结合 COM / COM+或者 JSP 及 Javabean 来实现。如图 2-2 所示。

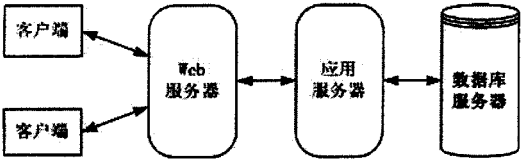


图 2-2 基于多层应用的动态发布模式

到目前为止，Web 应用主要停留在交互式模型上。Web Services 技术的出现促进了 Web 应用向第三个阶段转化：动态式的 Web 应用。

(3) 动态式的 Web 应用。动态式的 Web 应用是指企业对企业（B2B，Business to Business）的应用集成与信息的交换。由于 Web Services 是自描述的、自包含的模块应用。企业的应用程序封装成 Web Services，只要通过制定的 Web Services 接口，其它系统可以随时与这些 Web Services 连接，完成 B2B 的应用集成，达到动态电子商务的目的。动态式的 Web 应用如下图所示。

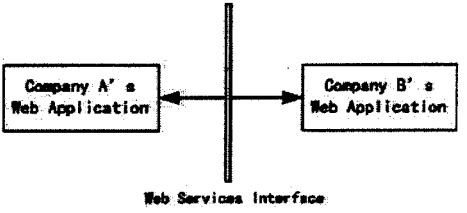


图 2-3 动态式的 Web 应用[14]

动态式的 Web 应用主要目的是简化系统的集成。动态系统集成主要分为连接和交易处理两大类。连接解决应用程序集成的问题，这类集成可以直接使用 Web Services 去解决程序与程序之间的数据交换问题。交易处理能够建立一个完整的企业对企业（B2B）的应用集成。连接和交易处理满足了动态系统集成的要求。

综上所述，Web 应用发展出现了新的特点，需要集成 Web Services 处理企业间的请求。这就对 Web 应用模型支持 SOAP 请求提出了要求。

2.1.2 框架

框架是为了解决某类软件问题而设计的一系列类与接口的集合[6]。框架有如下特性：

(1)框架由多个类和组件构成，每个类或组件能提供一个对某些具体问题的抽象。

(2)框架定义了这些抽象的类或组件如何协同工作解决整个问题。

(3)框架组件是可重用的。

框架一般为多层体系结构，典型的情况分为客户层、Web 层、中间层、EIS(Enterprise Information System)层。

客户层：实现用户与应用程序交互的功能，包括发送请求和接收中间层的响应。客户层有多种类型，除了传统的 Web 界面，还可以是通过 Web Services 接口访问的 SOAP 消息、移动浏览器和 Java Applets。

Web 层：提供客户层与应用逻辑的交互，映射客户端请求到中间层的业务逻辑并产生表示逻辑，可以实现与数据库、应用服务器的交互，是客户端和业务模型联系的纽带。

中间层：也称应用层或服务层，EJB 组件位于此层，Web 层通过 RMI(Remote Method Invocation)或 IIOP(Internet Inter-ORB Protocol)协议访问中间层，中间层实现了表现逻辑与业务逻辑的分离，增加 Web 应用框架的容错性和可靠性。但小规模的 Web 应用可以没有此层，Web 层直接访问数据库或其它数据源。

EIS 层：包含企业现有的数据和服务，提供对企业资源如数据库管理系统，客户关系管理系统（CRM，Customer Relationship Management），企业资源计划（ERP，Enterprise Resource Planning）等的访问。中间层对 EIS 层组件的访问有多种方法，如使用 JDBC(Java Database Connectivity)驱动访问关系型数据库，通过适配器访问 ERP 系统等。

在 Web 应用中需要体现框架的设计方法，保证 Web 应用程序的模块性。

2.2 MVC 设计模式及主要开发技术

应用框架中模式的应用对于框架的可靠性及执行效率都有很大的影响，有必要研究模式及 MVC 设计模式。

在构建基于 MVC 设计模式的控制器中，有多种可供选择的开发技术。由于 Java 语言的面向对象和跨平台的特性，控制器部分采用 Servlet 技术，为支持多类型用户界面，采用 XML 和样式表技术。下面分别介绍各种技术。

一、模式

模式是解决特定问题的经验，是在软件开发过程中整理和记录的经常重复出现问题的成功解决方案[1]。

模式有四个要素[2]：

(1)模式名称：模式名称用简短的词语描述出该模式所要解决的方案和效果，以帮助设计和交流。

(2)问题：描述何时使用该模式以及使用该模式所必须满足的条件等内容。

(3)解决方案：描述了设计的组成成分及成分间的关系及各自的职责，成分之间如何进行协作。提供设计问题的抽象描述和怎样用具有一般意义的元素组合（类或对象组合）来解决问题。

(4)效果：描述了模式使用后的效果和使用此模式所必须考虑到的利弊。因为复用是面向对象设计的要素之一，所以模式效果包括它对系统的灵活性、扩充性或可移植性的影响。

二、MVC 设计模式

MVC 是一种目前广泛流行的软件设计模式。近来，随着 J2EE(Java 2 Enterprise Edition)的成熟，它正成为 J2EE 平台上推荐的一种设计模式。随着网络应用的快速增加，MVC 模式的设计思想也影响着 Web 应用的开发，它为开发者理解分析应用模型提供最基本的分析方法、清晰的设计框架和规范的依据。

MVC 把一个应用的输入、处理、输出流程按照视图、控制、模型的方式进行分离，将 Web 应用分为三层：控制层、模型层、视图层，也称为控制器、模型、视图，如下图所示。

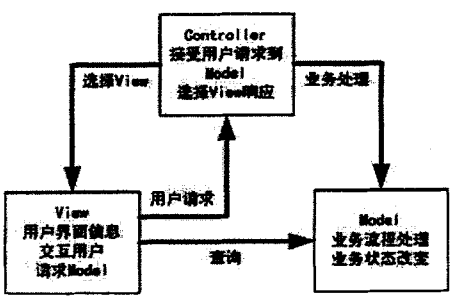


图 2-4 MVC 设计模式[1]

(1) 控制器(Controller)：控制器解释用户的输入并将其映射为模型的操作，同时定义应用程序的行为，分派用户的请求并选择恰当的视图用于显示。由于不同用户的视图交互和选择是不同的，一个应用程序可以基于一组相关功能设定一个控制层模块，也可以根据不同的用户类型设定多个控制层模块。

(2)模型(Model)：模型是业务逻辑的处理以及业务规则的制定。模型接受视

图请求的数据，并返回最终的处理结果。模型分为业务模型和数据模型。业务模型的设计是 MVC 的核心，它封装了业务逻辑的处理。数据模型主要指实体对象的数据保存，它是数据的对象化。目前流行的 EJB 组件是一个典型的构造模型的组件，它从应用技术实现的角度对模型做了进一步的划分，充分利用了现有的组件。

(3) 视图(View)：视图代表用户交互界面。对于 Web 应用，视图即为 HTML、XHTML(eXtensible HyperText Markup Language)、XML(Extensible Markup Language)和 Applet 等界面。随着应用的复杂性和规模性的增加，界面的处理也变得越来越复杂，一个 Web 应用可能有很多不同的视图，对视图的设计提出较高的要求。MVC 设计模式对于视图的处理仅限于视图数据的采集处理，不包括视图业务流程的处理，业务流程交给模型处理。比如，一个订单视图接收来自模型的数据并显示给用户，并将用户界面输入的数据和请求传递给控制器和模型。

MVC 设计模式的优点：

(1)允许多种用户界面的扩展：在 MVC 模式中视图与模型没有必然的联系，都是通过控制器发生关系，这样如果要增加新类型的用户界面，只需要改动相应的视图和控制器即可，而模型则无需发生改动。

(2)模块的有效性：控制器、模型、视图的程序改变不会影响到其他各层及组件，不同的组件开发能够同时进行。

(3)代码和设计的复用性：广泛采用可复用的组件，使开发小组之间易于沟通，设计的系统易于理解，同时也降低项目的开发成本。

(4)易于维护：控制器和视图可以随着模型的扩展而进行相应的扩展，只要保持一种公共的接口，控制器和视图的旧版本也可以继续使用。

(5)功能强大的用户界面：用户界面与模型方法调用组合起来，使程序使用更清晰友好的界面发布给用户。

但是 MVC 设计模式的设计要求较高。

综合上述，MVC 是构建应用框架的一个较好的设计模式，可以将业务处理与显示分离，将应用分为控制器、模型和视图，增加了应用的可拓展性、强壮性及灵活性。基于 MVC 的优点，目前比较先进的 Web 应用框架都是基于 MVC 设计模式的。本文也是在基于 MVC 模式的 Web 应用框架 Struts 控制器基础上的改进。

三、Servlet 技术

Servlet 是由容器管理的 Web 组件，是一种小型的、与平台无关的 Java 类。Servlet 通过由 Servlet 容器实现的请求－响应模型与 Web 客户机进行交互。Servlet 可以看作是运行在 Request、Response 请求服务器上的模块[3]。当客户向能调用 Servlet 的服务器发送请求时，服务器首先查看是否已加载 Servlet，如果

没有，则加载 Servlet 类，并创建新实例；然后调用 init()方法初始化 Servlet，接着调用它的 service()方法。下图显示了 Servlet 的工作方式。

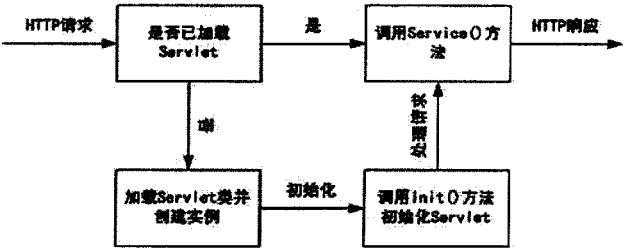


图 2-5 Servlet 实例化、初始化以及处理请求

Servlet 的主要优势在于：

(1)具有 Java 语言的优点。由于使用 Java 语言开发，具有 Java 语言的所有特性，如面向对象、可移植性、包括联网支持在内的大量应用程序接口(API, Application Programming Interface)和易用性。

(2)执行效率高。一个 Servlet 可以同时处理多个请求，每个请求将生成一个新的线程，而不是一个完整的进程，多个客户能够在同一个进程中同时得到服务。

(3) 构造的控制器功能强。Servlet 是模块化的，每个都执行一个特定的任务，也可以协同一起进行工作，构成功能更强的“Servlet 链”。

本文在构造 MVC 模式的控制器中采用 Servlet 技术。

四、XML 技术

可扩展标记语言(XML，Extensible Markup Language)是标准通用标记语言(SGML，Standard Generic Markup Language)的一个子集，其设计目标是能够以目前 HTML 可能实现的方式在 Web 上使用、接收和处理通用 SGML。XML 具有以下特性[4]：

(1)自描述性：基于 XML 的文档有关于文档的内容描述，还有关于文档内容间互相关系的描述。

(2)结构良好性：XML 文档由 XML 元素构成。一个 XML 元素由开始标记、数据和结束标记构成，标记用以说明数据的含义。标记间的数据可以包含另一个 XML 元素，这样就形成层次结构，且标记具有不能交错嵌套的结构良好性。

(3)有效性：XML 文档的标记可以是任何类型的标记。为使 XML 文档中的标记含义已知，满足标记语义限制，引入 XML Schema 或 DTD(Document Type Definition)，XML Schema 或 DTD 的 XML 文档称为有效的 XML 文档。

(4)内容与形式分离：XML 把数据内容与表现形式分开，这样既可以只关心

数据的逻辑结构，也可以通过样式表来格式化数据的表现。

由于 XML 具有以上的特性，其应用领域不断地被拓展，下面是与 XML 相关的主要应用：

(1)内容管理发布：基于 XML 的可自定义、可扩展的能力，使用 XML 描述 Web 应用上多种类型、多种样式的数据内容。基于 XML 一次描述，多次表现的特性，使得 XML 描述的内容能够以多种形式进行信息发布。

(2)电子商务应用：XML 承担了以往 EDI(Electronic Data Interchange)所承担的角色，使用 XML 描述交换商务事务信息，实现分布式的电子商务应用的交互。

(3)数据层集成：由于 XML 的结构良好性，很适合应用于数据集成，数据交换领域。

(4)应用层集成：应用层集成主要是以 XML 为技术基础的 Web Services 系列技术，完成在业务层或者函数层上的系统互联。

(5)系统配置信息描述：随着 XML 在各种各样应用开放中的延伸，原先以文本文件和 INI 形式文件存在的系统应用配置文件逐渐被以 XML 文档管理方式存在的配置文件所代替，如 Tomcat、Struts 中的配置文件。

五、EJB 组件技术

目前，组件技术主要有三种代表性的主流技术，即 Microsoft 的 COM/DCOM 技术，Sun 公司的 EJB(Enterprise JavaBean)技术和 OMG(Object Management Group)的 CORBA(Common Object Request Broker Architecture)技术。由于 Java 技术的良好移植性及跨平台性，本文的应用框架采用 EJB 技术。

EJB 体系结构或称 EJB 组件技术是标准的组件体系结构[5]，使用 Java 编程语言构建分布式的面向对象的商务应用程序。通过把使用不同供应商提供的工具及其组件组合在 EJB 体系结构中，可以构建分布式的应用程序。使用 EJB 编程，应用程序开发人员不必了解低层次的事务和状态管理的细节，程序开发相对容易。EJB 应用程序遵循 Java 编程语言的“一次编写，到处运行”的原则，使 EJB 组件只需开发一次，然后在多个平台上部署，而不需要重新编译或修改源代码，EJB 体系结构处理企业应用程序生命周期中的开发、部署和运行等方面。EJB 体系结构可以使多个供应商提供的工具能够开发并部署组件。EJB 体系结构与现有的服务器平台兼容，供应商能够扩展它们的现有产品，以支持 EJB 组件。EJB 体系结构还提供 EJB 组件和非 Java 编程语言应用程序之间的互操作。

EJB 容器是管理一个或多个 EJB 类或实例的抽象。它通过规范中定义的接口使 EJB 类访问所需的服务。容器厂商也可以在容器或服务器中提供额外服务的接口。EJB 服务器是管理 EJB 容器的应用程序，提供对系统服务的访问。一个 EJB 服务器必须提供对可访问 JNDI(Java Naming and Directory Interface)的名字服务和事务服务支持。EJB 服务器也提供厂商自己的特性，如优化的数据库访问接

口，对其它服务（如 CORBA 服务）的访问。下图展示了 EJB 的体系结构。

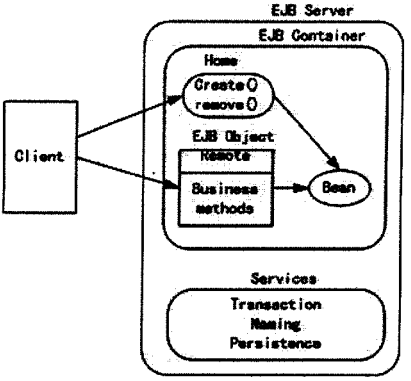


图 2-6　EJB 体系结构[5]

EJB 组件技术的优势在于：

(1)EJB 组件使应用程序的编写简单化。尽管 EJB 体系结构复杂，但应用程序开发人员一般都不必再编写用于访问系统服务的代码，EJB 容器管理一切。

(2) EJB 组件容易移植。因为在 EJB 模型中，各个软件组件都是严格分离的，用 EJB 组件构建的应用程序可以从一个服务器移植到另一个服务器。

(3)软件重用性高。可以从现有的软件组件装配出服务器端应用程序，从而使软件能够重用。

(4)内置了对典型企业级系统服务的支持。包括分布式对象、事务处理、数据库、安全和全局命名。

(5)有多种 EJB 产品可以选择。多家 IT 供应商都采纳 EJB 体系结构，客户可以随意选购软件组件，如 EJB 组件、容器及 EJB 服务器，只要符合 EJB 体系结构，就可以互操作。

(6)EJB 体系结构保障原有的 IT 投资。因为 EJB 体系结构可以将现有的信息系统包装起来使用，而不要求客户更换现有技术。

本文的实例中模型层采用 EJB 技术实现。

2.3 Struts 框架的分析

2.3.1 概述

为了适应 Web 应用发展出现的新特点，提高 Web 应用框架的可扩展性、易维护性和高度灵活性，国内外出现了许多基于 MVC 设计模式的 Web 应用框架，

其中最为著名的是 Struts 框架、Enhydra 框架和 Turbine 框架。它们都是基于 MVC 模式的 Web 应用框架，对实现 MVC 模式的方式上大致相同，但侧重点不同。Struts 采用 JSP 技术，是基于 MVC 模式的 JSP，是最好的 JSP 框架。Enhydra 框架侧重于完整的应用平台的开发。Turbine 是基于 Servlet 的 Web 应用开发，它可以通过创建使用特定服务来处理模板的 Screen，来集成现有的模板技术（如 Velocity、Webmacro、JSP、FreeMarker、Cocoon）。这些框架都具有很多优点，但对多类型用户界面的支持的研究稍显不足，特别是无线应用的普及，Web Services 技术的发展，使 Web 应用向动态式 Web 应用转化，对 Web 应用框架支持多类型用户界面的需要也日益迫切起来。

本文针对 Web 应用出现的新特点，参考 Struts 实现 MVC 框架的原理，在 Struts 框架的基础上改进其控制器的设计，摒弃 Struts 的标签技术，采用 XML 技术实现对 WAP 和 SOAP 消息的支持。下面分析 Struts 实现 MVC 模式的机制，并提出本文的改进地方。

2.3.2 Struts 框架的实现机制

Struts 是 Apache 组织的一个项目，提供了一个实现 MVC 框架的高度自动化的方式。采用的主要技术是 Servlet，JSP 和 custom tag library。作为一个 MVC 的框架，Struts 对 Model、View 和 Controller 都提供了对应的实现组件。下面对控制器、模型、视图及配置文件的实现作详细的分析。

(1)Controller(控制器)：Struts 的控制器是由 Java servlet 实现的。控制器是 Web 应用的控制中心，它映射用户请求为业务逻辑并选择相应的视图进行显示。控制器也负责用相应的请求参数填充 Action From（通常称之为 FromBean）,并传给动作类（通常称之为 ActionBean）。动作类实现核心商业逻辑，它可以访问 java bean 或调用 EJB。最后动作类把控制权传给后续的 JSP 文件，后者生成视图。Struts 的控制组件主要是由三个类组成：ActionServlet、Action 和 RequestProcesser。

①ActionServlet：ActionServlet 继承了 javax.servlet.http.HttpServlet 类，是 Struts 应用的核心控制组件。它接收客户端请求并执行 Struts 控制组件中的 Action 类进行业务处理。ActionServlet 包括一组基于配置的 ActionMapping 对象，每个 ActionMapping 对象实现了一个请求到一个具体的 Action 处理器对象之间的映射。类 ActionMapping 中有 Action 对象的名字和地址的描述。当有请求传入 controller 时，它把请求的路径映射到 Action 的地址，并把请求传给那个 Action。

ActionServlet 类包括 HttpServlet 类的几个主要方法：init(),doGet(),doPost() 和 destroy()方法。

ActionServlet 类的执行流程是首先根据是链接还是表单请求，调用 doGet()或 doPost()方法接收 request 请求并调用 process()方法，在 process()方法里调用 RequestProcessor（在后文作详细介绍）的 process 方法，从配置文件里获取相应的用户请求映射信息，调用 Action 的 execute 方法处理业务逻辑，并返回 ActionForward 对象选择视图进行显示。一个请求周期完成。

②Action：Action 是 Struts 控制器中又一个重要的组件。Action 对象继承 org.apache.struts.action.Action 类并实现了 Web 应用的具体功能。Action 对象主要的方法是 execute()，在 execute()方法中涉及到了具体的请求处理，比如用户登录时用户名和密码的验证，对于数据库中数据的调用等。Struts 中的 Action 是 ActionServlet 从用户请求中截取相应的请求信息根据配置文件进行选择调用的，此信息在 struts-config.xml 文件的<action>元素中进行配置，如下代码所示：

```
<action      path="/editRegistration"
        type="org.apache.struts.webapp.example.EditRegistrationAction"
        attribute="registrationForm"
        scope="request"
        validate="false">
        <forward name="success" path="/registration.jsp"/>
</action>
```

上面代码显示了如果用户请求路径是 editRegistration，ActionServlet 将调用 org.apache.struts.webapp.example.EditRegistrationAction 类进行业务处理，并转向 registration.jsp 视图页面进行显示。

③ RequestProcesser：主要功能是完成用户请求信息的提取。在 Struts1.1 以前，RequestProcesser 的功能在 ActionServlet 里实现，Struts1.1 版本把此功能单独划分出来由 RequestProcesser 类实现，以方便 Web 应用的子应用请求的分类管理，RequestProcesser 中由 ActionServlet 调用的方法是 process()，它首先调用 processPath 方法获取请求路径，接着通过 processLocal()方法获取客户端浏览器的地域类型，以实现整个应用的国际化，然后通过调用 processMapping()方法获取配置文件中的 ActionMapping 的 Action 配置信息使得程序转向相应的 Action 对象。

(2)Model：Struts 框架本身并没有提供模型组件，从这一点上说，Struts 并不是完全意义上的基于 MVC 的框架。但 Struts 借助现有的其它框架和模型，为使用 Struts 框架的开发者提供了更为灵活丰富的可选模型组件，其中包括 EJB 组件，Java Data Object，JavaBean 和一些对象关系映射框架(Object-to-Relation Mapping framework，ORM)，Struts 框架允许开发者自由选择模型组件。

(3)View：Struts 的 View 组件包括 JSP、Struts 标记库和 ActionForm 对象。在 Struts 应用中可以直接采用 JSP 和 JSP 标记库开发 View 组件，并为了避免视图硬编码在程序里，在配置文件里可以配置 JSP 文件的逻辑名和实际文件名。为

了增加 View 组件的功能和尽可能的减少视图文件中的代码，Struts 本身设计了标记库，包括 Bean 标记库、HTML 标记库、Logic 标记库和 Template 标记库，Bean 标记库提供了一组封装了访问和操作 JavaBean 的标记，HTML 标记库提供了用于表单输入和创建基于 HTML 的用户接口标记，Logic 标记库主要实现了对于返回的结果的判断、迭代及视图的重定向，Template 标记库提供了可重用的模板视图的定义和使用。通过这些标记库的使用，实现了 Struts 视图中的零代码及功能的丰富，分开了表现逻辑和程序逻辑。

Struts 的 ActionForm 对象是针对封装客户表单输入的信息而设计的。当用户提交表单的输入信息时，一般情况下是随着 HTTP 请求被发送到服务器，然后开发者通过编程获取用户提交的信息，验证输入的有效性，并把此信息传输给业务逻辑处理组件。若用户输入错误，则应返回到原来页面，并显示错误信息提示登录失败。这个过程如果手动编程会显得很烦琐，Struts 提供了 ActionForm 对象，ActionForm 对象能自动获取 HTML 表单的输入数据，传输给 Action 类，并能自动执行表单数据验证功能。ActionForm 对象还可以接收 Action 类返回的数据回送给表单。一般情况下，一个 HTML 表单对应一个 ActionForm，但也可以多个 HTML 表单对应一个 ActionForm，ActionForm 使得获取用户提交的信息变得更为方便简单。

Struts 的实现机制如下图[6]。

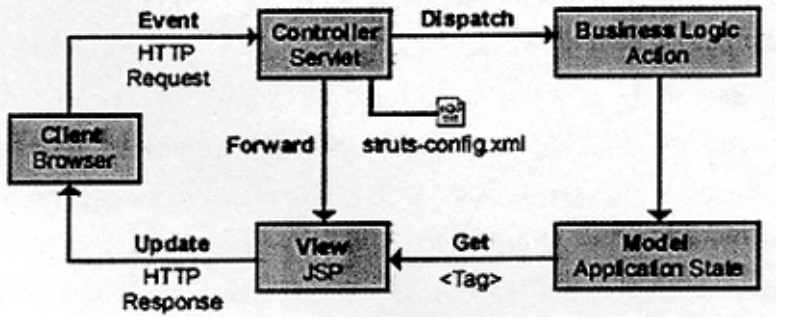


图 2-7　Struts 实现 MVC 模式的机制

(4)Struts 的配置文件

由于本文涉及到了对 Struts 视图设计的改进，并参考 Struts 的 struts-config.xml 配置文件，实现控制器选择的 xml 文件的集中配置，避免在控制器中硬编码视图文件名，所以有必要分析 Struts 的配置文件的实现机制。

Struts 中第一个需要配置的是 web.xml 文件，它描述了系统的 ActionServlet 对象并指定其初始化参数及 struts-config.xml 文件，并定义了网站首页。下面代码显示了 web.xml 的部分代码[6]。

```
<servlet>
        <servlet-name>action</servlet-name>
```

```
        <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
        <init-param>
            <param-name>config</param-name>
            <param-value>/WEB-INF/struts-config.xml</param-value>
        </init-param>
        <init-param>
            <param-name>debug</param-name>
            <param-value>2</param-value>
        </init-param>
</servlet>
//首页定义
<welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>index.html</welcome-file>
</welcome-file-list>
```

struts-config.xml 文件是一组响应用户请求的 ActionMapping 对象集合，提供了模型、视图和控制的集中映射。每一个用户请求都有一个 ActionMapping 对象与之对应。它需要应用开发者配置与用户请求相对应的 Actionform 数据属性、是否需要进行数据校验与回写、Action 数据处理、输入页面，以及处理完成后可能跳转的页面等一组属性信息，这组属性信息构成一个 ActionMapping 对象，它决定了该项请求的处理过程。ActionMapping 配置示例如下[6]。

```
//输入页面及跳转页面的配置信息
<action-mappings>
<action   path="/loginout/findPassword"
  type="com.hjc.homepage.action.FindPasswordAction"
   input="idInputView" name="findPasswordForm"
   scope="request" validate="false">
            <forward name="idInput" path="idInputView"/>
            <forward name="finish" path="loginView"/>
</action>
<action    path="/vote/vote"    type="com.hjc.homepage.action.VoteAction"    input="voteView"
   name="voteForm" scope="request" validate="true">
            <forward name="loginPage" path="loginView"/>
            <forward name="success" path="voteView"/>
</action>
</action-mappings>
```

上面两个的配置文件的调用是由 ActionMapping 对象实现的。具体实现流程是：Struts 框架的调度控制中心 ActionServlet 类依据 ActionMapping 对象，对用户配置文件进行启动时加载，读取配置信息，进行相应的业务逻辑处理，并根据处理结果返回一个 ActionForward 对象，再根据 struts-config.xml 文件的映射，定位到相应的 JSP 页面进行显示。

由上面代码可以看出，Struts 对于输入及跳转页面的配置信息较多，特别是应用于大型网站时，这样的配置信息会急剧增多，虽然配置文件的好处是显然的，但配置的信息增多也会增加维护的难度，本文将使用 JMX 技术对本文设计的框架的配置文件进行管理，在不改变框架任何模块的情况下，实现对框架配置文件的简单方便的管理。第四章将加以详细分析及实现。

2.3.3 Struts 框架的不足

Struts 框架的控制器 ActionServlet 基于配置的 ActionMapping 对象，根据具体请求调用模型层相应的 Action 对象，完成业务逻辑的处理和数据的查询与更新，并根据 struts-config.xml 配置文件在 ActionForward 对象中定位相应的 JSP 页面响应给客户端。请求、处理、跳转视图流程清晰，但其欠缺对 WAP 和 Web Services 请求的支持。Struts 的 View 组件主要是应用标签技术减少视图中的代码，但 Struts 标签繁杂，难于学习且对用户界面的支持只是单一的 Web 用户，没有考虑到无线用户和 Web Services 界面，不易扩展对其它界面的支持。所以本文将对 Struts 框架在支持多类型用户界面进行改进，摒弃其标签技术，使用 XML 及样式表支持多类型用户界面，改进其控制器的设计，实现基于相同的内容源文件选择相应的样式表产生视图响应给客户端。

2.4 Struts 框架与其它框架的比较

2.4.1 Enhydra 框架

Enhydra 是 ObjectWeb 组织的开源项目[8]，它作为一个应用服务的开发平台，是基于 J2EE 标准，建立在 Java 虚拟机之上的。Enhydra 运用了已经在 Java 中比较成熟的面向对象的技术来设计它本身的层次结构。

Enhydra 应用程序框架分为三个独立的层次：表示层（The presentation layer）、商务层（The business layer）、数据层（The data layer）。

Enhydra 提供了很多应用工具，其中最常用的是 XMLC(The Extensible Markup Language Compiler)和 DODS (The Data Object Design Studio)。其中 XMLC 用来建立一个代替 XML 文档结构的 Java 对象，DODS 用来建立数据模型的图形化操作工具，它把对数据库直接的操作都封装在工具包里面。Enhydra 由于其开源项目的优势，正日益被越来越多的人研究应用，但 Enhydra 的初始学习曲线有点陡峭，并且还没有实现完全的松耦合结构。

2.4.2 Cocoon 框架

Cocoon 是一项 XML 发布框架。允许定义 XML 文档和文档的转换，并最终生成所需的表示形式，Cocoon 的核心技术是 XSP，XSP 和 JSP 非常相似，都是基于 Java 的技术，XSP 实际上就是动态的 XML 文档，是在 XML 文件中嵌入了动态的数据内容，XSP 分为三种类型：嵌入式的 XSP、嵌入逻辑单的 XSP 和带 taglib 标记库的 XSP，Cocoon 主要是侧重于视图显示的研究。

2.4.3 各种应用框架的优缺点

由于每个 Web 应用框架重点解决的问题不同，因此各有优缺点，Struts 专注于统一控制器的实现和用户提交的表单自动映射为 Javabean，使用户提交的请求和数据能够自动获取，减少编程的复杂度。Enhydra 则专注于开发平台和运行平台的开发，Cocoon 主要是侧重于标签技术的研究。但是，它们在支持多类型用户界面上没有做深一步的细化。本文则对此问题提出自己的见解和设计思想。

2.5 本章小结

本章首先论述了 Web 应用框架发展的三个阶段，指出了 Web 应用框架发展的新特点。然后分析了 MVC 设计模式应用于 Web 应用框架的优势，并研究了基于 MVC 模式的 Struts 框架的实现机制，分析了 Struts 框架的优点及对多类型用户界面支持的不足，在此基础上针对 Struts 框架的不足，提出了一种改进方案，以实现控制器对多类型用户界面的支持。

第三章 Struts 框架控制器的改进

3.1 多类型用户界面及其特征

3.1.1 简介

随着企业应用的多样化和实现技术的日益成熟，单一的用户界面越来越显示出表达的不足和功能上的欠缺，开发能提供多类型用户界面，以更好的服务满足不同类型客户需求的企业应用系统成为必然的趋势。

面对相同的企业信息系统和数据，一般用户希望通过 Web 浏览器以 HTML 形式访问；移动用户希望通过无线应用协议(WAP,Wireless Application Protocol)以无线标记语言(WML,Wireless Markup Language)形式访问；企业的 B2B(Business To Business)伙伴则希望使用基于 XML 的 Web Services 交换数据；企业内部用户出于管理的需要，则会以图形用户界面形式的客户端访问[12]。下面将分别介绍各种类型客户端的特点及实现机制。

3.1.2 WAP 应用的特点

WAP 即无线应用协议[9]，是指在数字移动电话、Internet、个人数字助理、计算机和应用设备之间进行通讯的开放全球标准。WAP 是由一系列协议组成，用来对无线通信设备进行标准化建设。WAP 服务可用于 Internet 访问，包括收发电子邮件、访问 WAP 网站等。WAP 将移动网络和 Internet 以及公司的局域网紧密地联系起来，提供一种与网络类型、运营商和终端设备都独立的移动增值业务。

由于无线应用的普及，越来越多的无线通讯设备通过 WAP 访问 Internet，基于 WAP 的应用为网站提供了非常好的增值服务，企业现有的 Web 系统对集成 WAP 服务的要求也日益迫切。

为了使 Web 应用能够集成 WAP，有必要分析 WAP 的编程模型，如下图。

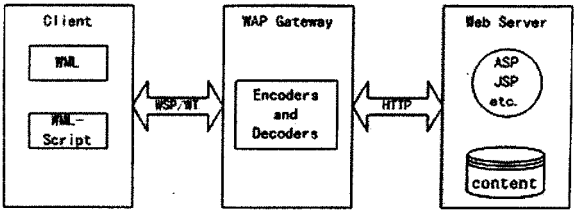


图 3-1 WAP 编程模型[37]

WAP 编程模型由三部分组成，WAP 终端、WAP 网关和 WAP 内容服务器。

WAP 终端上集成了一个结构相对简单的微浏览器，通过这个内置的微浏览器，利用 WML，可发出带有各种参数的访问请求，网关通过解析，发出相应的请求到 WAP 服务器上，服务器通过检索参数生成相应的结果，并返回给网关，然后网关进行解码和编码，再把结果送回移动终端，从而实现一个会话全过程。网关与服务器之间通过 HTTP 协议进行通信，WAP 网关起着协议的“翻译”作用。WAP 内容服务器存储着大量的信息，以提供 WAP 手机用户来访问、查询、浏览等。

由于无线通讯设备与普通的个人计算机相比，屏幕较小，设备的内存有限，且其 CPU 性能也有限，通讯带宽较窄，时延较长，所以采用 WML 编码显示数据。无线标记语言 WML 是扩展标记语言 XML 的子集，是专为移动电话，呼机和个人数字助理(PDA,Personal Digital Assistants)等无线设备用户提供交互界面而设计的。它像一盒卡片，一个单一的 HTML 文档显示成一个单一的 Web 文档，而一个单一的 WML 文档可以包含很多卡片，WAP 设备的屏幕一次只能显示一张卡片。

根据 WAP 的特点，本文改进的控制器中对 WAP 输入的检测通过用户请求代理程序中的参数实现，产生 WML 语言的输出则通过针对 WML 语言的样式表格式化 XML 内容文件来实现。

3.1.3 独立桌面客户端的实现模型

由于桌面客户端的特殊性，不通过控制层直接和业务层 EJB 进行交互，客户端应用程序使用 JNDI 查找服务器方业务层的 EJB 组件。例如，客户端应用程序查找 JNDI 中的 EJB 引用并收到一个 EJB 客户端代理，客户端在后面将使用这个代理来访问 EJB 组件。

3.1.4 基于 XML 的 Web Services 客户端

随着 Web 应用和基于 Internet 的 B2B 电子商务的不断发展，各种不同应用系统之间的系统集成、相互访问和对接的问题日益突出，基于 XML 的 Web Services 技术正是针对这一问题的解决方案。Web Services 的主要目标是在现有的各种异构平台的基础上构筑一个通用的与平台无关、语言无关的技术层，各种平台之上的应用依靠这个技术层来实施彼此的连接和集成。既然 Web Services 如此重要，有必要了解 Web Services 技术及如何与 Web 应用集成。

Web Services 的定义[10]：Web Services 通过简单对象访问协议(SOAP，Simple Object Access Protocol)为 Web 用户提供有用的功能；Web Services 提供了一种描述接口的方式即 Web 服务描述语言（WSDL，Web Services Description Language），允许用户建立一个客户机应用程序与之交流；Web Services 通过使用统一发现、描述及集成（UDDI，Universal Discovery，Description and Integration）在目录中进行注册，从而使潜在的用户可以很容易找到。Web Services 中提到的概念有三个：SOAP、WSDL、UDDI。

SOAP 是一个轻型的协议，用于在非集中的分布式环境中交换信息，包含三部分：封套（定义一个框架来描述消息的内容以及如何处理消息）、编码规则（用于表达应用程序定义的数据类型的实例）以及约定（用于表示远程过程调用和响应）。

WSDL 是 Web Services 描述语言，描述了 Web Services 的位置、用于访问这个 Services 的通信协议、SOAP 消息以及如何交换这些消息等。

UDDI 定义了服务描述与发现的标准规范，使得商业实体能够彼此发现，实施其电子化的商业贸易，并在一个全球的注册体系架构中共享信息。下图展示了 Web Services 的体系结构[13]。

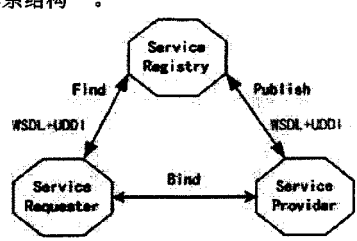


图 3-2 Web Services 面向服务的体系结构[8]

Web Services 的工作流程如下，开发人员将自身待集成并提供服务的应用包装成 Web Services，使用 WSDL 描述这些包装好的 Web Services，并按需要将这些 Web Services 及其描述发布到 UDDI 注册中心以供查询。所有的这些工作都可

以使用支持规范的工具来完成。此时，企业之间的集成就转变为 Web Services 的对接。开发人员可以通过 UDDI API 来查询 UDDI 注册中心或与业务伙伴的技术人员进行交流，获取对方的 Web Services 的 WSDL 描述文档，通过平台工具自动将 WSDL 描述文档装载到自己的开发平台中，并生成相应的接口。开发人员可以使用 XML Schema 工具理解数据结构，在自己的应用中引入使用平台工具生成的调用接口和数据结构，利用 SOAP 技术与对方的 Web Services 进行交互，从而完成 B2B 的集成。

Web Services 和 Web 应用技术存在着差异，Web 应用解决如何让人来使用 Web 应用所提供的服务，Web Services 解决如何让计算机系统来使用 Web 应用所提供的服务。

本文改进的控制器对于 Web Services 的支持是通过包装控制器为 Web Services，以提供控制器对 SOAP 消息的响应。

3.2 多类型用户界面的支持机制

3.2.1 XML 和 XSL 的支持

在最初的 Web 网站发布中，只提供 HTML 页面的显示，开发者可以使用脚本语言和 HTML 标记结合显示动态内容，必要时也可使用层叠样式表统一控制视图的显示样式。但 HTML 文件结构和显示混合在一起，显示的内容与显示的格式不能分离，使其难于扩展对多界面的支持，这也是发展 XML 的最主要的原因之一。

XML 的最大优点是其数据存储格式不受显示格式的制约，把显示格式从数据内容中独立出来，保存在样式表文件中。如果需要改变文档的显示方式，只要修改样式表文件即可。所以本文在扩展 Struts 框架对于多类型界面的支持时采用 XML 和 XSL 技术，实现对于 HTML、WAP 和 Web Service 的支持。

XSL 包括两个方面：XSL 转换(XSL Transformation，XSLT)和 XSL 格式化对象(XSL Formatting Object，XSLFO)。XSLT 可以使用 XSL 文件，将 XML 文档转换为 HTML 页面、WML 页面或者甚至可以是另外一个 XML 文档，可以选择要显示的内容，添加内容，重新排列内容等。XSLFO 主要是用来转换 XML 为格式化文档如 PDF 等文档，它定义了制作高质量可打印文档时非常重要因素，如有关页面、页边距和字体的信息等，本文不多加介绍。

使用 XML 和 XSL 支持多类型界面的关键是为同一内容源文件 XML 创建不同的 XSL 文件。下面介绍 XSL 文件。

(1)模板：XSL 文件本身是一个 XML 格式的文档，XSL 用模板来描述如何

输出 XML，XSL 使用一个或多个模板来定义如何输出 XML 元素，用一个匹配属性来将模板与一个 XML 元素联系起来，还可以用匹配属性来为 XML 文档的一个完整分支来定义模板。

(2)Xpath：XPath 是用来帮助 XSLT 在 XML 源文档中查找定位信息的语言，它的语法如代码：<xsl:value-of select="."/>(选择当前节点)，<xsl:for-each select="news//title">(选择 news 元素下所有的 title 子元素)，Xpath 还包括运算符和功能函数等，通过使用 Xpath ，XSLT 可以实现复杂的查询和操作。

(3)循环和排序：实现了所需内容的循环和有序的输出，如代码：

```
<xsl:for-each select=" //news" >
   <h2><xsl:value-of select=" title" /><h2>
<xsl:for-each>
```

将输出一列新闻的标题，当需要对所输出的标题进行排序时，可以在第二行加代码<xsl:sort select=" title" />，新闻标题即可按字母顺序排序。

(4)条件句：可以基于数据自身格式化数据，如给 news 设定首页显示的优先权(prior)，首页只显示优先权为 yes 的新闻。则若 news 的 prior 属性为 yes 时，显示 news 元素所有的信息，若为 no 时，则不显示 news 元素的任何信息，则可用如下代码描述：

```
<xsl:if test=" @prior!='no' " />
//选择新闻标题
<xsl:if >
```

综上所述，在 XSL 里选择所需内容，并加入针对 HTML 和 WML 标记，即可将 XML 转换为相应的 HTML、WML 页面输出。若不加任何标记语言，则是对原 XML 文件的选择并仍转换为了 XML 文件，适应了 Web Services 的 SOAP 消息的输出。

3.2.2 Servlet 的支持

通过 XML 和 XSL 可以显示多类型界面，但控制器需要知道用户请求是何种类型，才能自动调用相应的 XSL 进行显示，所以 Servlet 必须增加检测用户类型的机制。Servlet 支持多类型界面的步骤如下：

(1)检测浏览器类型：用户的 HTTP 请求被提交后，控制器可以从 User-Agent 标题中检测出用户所使用的浏览器类型，如在 Servlet 控制器中加入代码：String target=request.getHeader("User-Agent")，target 变量就是请求用户的浏览器类型。

(2)资源包：建立一个资源包文件，列出浏览器类型串和将串赋给组的名字，如 MSIE=explorer Nokia=nokia UP.Browser=up HandHTTP=handweb 等，在 Servlet 中读取资源包并查询出用户请求浏览器类型串所对应的组的名字。

(3)Servlet 自动获取相应的样式表：在内容源文档 XML 中关联各种媒体的样式表，并根据 Servlet 查询到的组的名字自动调用相应的样式表进行转换。关联样式表的代码如：<?xml-stylesheet href="wap.xsl" type="text/xsl"?>。

Servlet 中调用的代码如：

InputSource[] XSLSheet=new InputSource(("d:/xml/news.xml"),mediaStr,null,null);

可根据 mediaStr 的值调用相应的 XSL。

以上是 Servlet 对于多类型界面的一般支持，下面将对 Struts 控制器进行此方面的改进。

3.3 Struts 框架控制器的改进

3.3.1 Web 应用的整体框架

本文在 Struts 控制器的基础上，对控制器的实现进行改进，以使其支持多类型用户界面。下图是 Web 应用框架的整体图，体现了支持的用户界面及框架的整体结构。

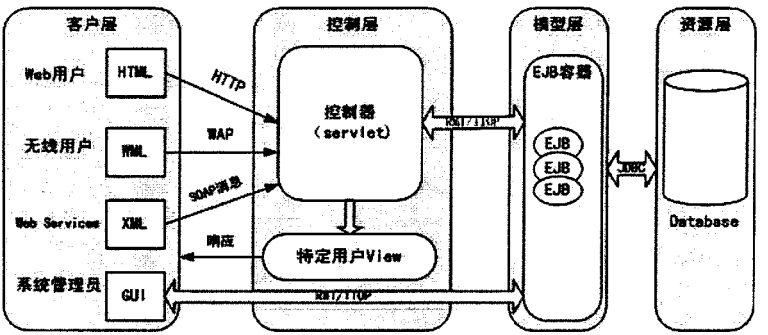


图 3-3　基于 MVC 模式的 Web 应用的整体框架

整个框架图分为客户层、控制层、模型层、资源层。客户层为多种类型客户端，包括无线用户、Web Services、无线用户和图形用户界面。控制层是论文设计的重点，是在 Struts 框架基础上的改进，可以统一接收用户请求，检测用户请求浏览器类型，并调用相应的业务处理模块，获取结果，并由结果文件根据检测到的浏览器类型调用相应的样式表生成视图文件，响应给客户端。模型层由 EJB 组件构成，主要完成业务逻辑的处理和与资源层的交互。资源层主要是数据库资源。本章的改进主要集中在控制器，下一节将具体介绍。

3.3.2 控制器的结构改进

Struts 的控制器针对 Web 用户界面，由控制器统一接收用户请求，根据配置文件调用相应的 Action 对象，完成业务逻辑的处理，并返回给 Action Servlet 视图文件(JSP 文件)，由控制 Servlet 响应给 Web 客户端，如下图所示。

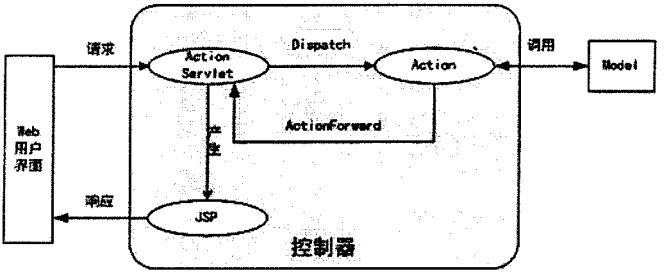


图 3-4 Struts 控制器

本文对控制器的改进是在其 Action Servlet 的基础上的，将 ActionForward 对象包装在自己创建的对象 ViewRouter 里，在 ViewRouter 对象里实现针对特定用户界面调用不同样式表对源文件进行转换。如下图控制器的改进。

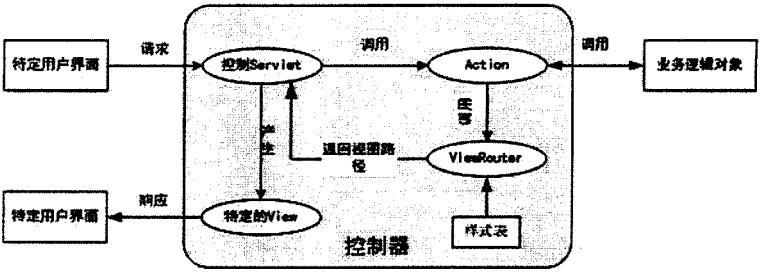


图 3-5 控制器的改进模型

下面是改进的控制器的工作流程：客户端向 Web 服务器发出请求，由配置文件把此请求映射至控制 Servlet 统一处理，根据请求的类型，控制 Servlet 从 Action 库中检索相应的操作类型并获取。

由于 Web 应用要处理大量的请求，所以操作库中为每个操作类型保持一个操作实例。对于给定的请求类型，这些操作是可重用的。对于多个请求，重用单个操作可以大幅减少框架必须实例化所需的操作数量。

控制 Servlet 从操作库中获取一个操作后，便调用此操作的 perform 方法，perform 方法实现对业务逻辑的调用及数据对象的更新，返回用户需要的数据，

并根据从模型层返回的数据生成 XML 文件。接着 perform 方法返回一个 ViewRouter 对象。

Controller Servlet 调用 ViewRouter 对象的 route 方法，route 方法获得用户请求 USER-AGENT 中的信息，根据请求的媒体类型不同调用相应的样式表对 XML 文件进行转换，将转换结果返回给控制 Servlet，由控制 Servlet 把结果文件响应给相应的客户，一个请求周期完成。参照下面的序列图。

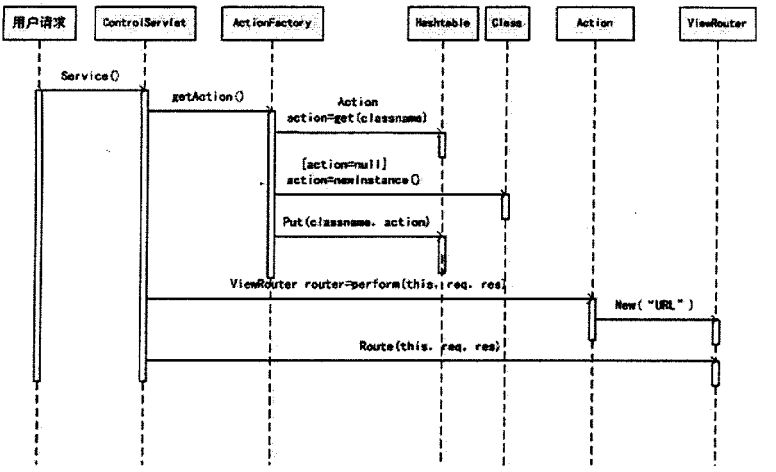


图 3-6 控制器序列图

对其的改进体现在由 Action 完成业务逻辑的处理后将要显示的结果生成 XML 文件，为 ViewRouter 对象提供内容源文件。并在 ViewRouter 对象里检测用户请求类型，自动选择相应的样式表对已产生的内容源 XML 文件进行显示，这样不仅避免了在视图文件里加入 Struts 标签，并可以编写多种样式表，支持多类型的用户界面。

下面是 ViewRouter 类的 Route 方法的详细实现。

3.3.3 改进控制器的技术实现

由上节可知，对于多类型用户界面的选择是在视图选择器中的 route 方法完成的，本文中对多类型用户界面的判断是针对三种类型的： Web 用户界面、移动设备的 WAP 界面和 Web Services 的 SOAP 请求。

具体实现流程：首先分别编写不同的用户界面的样式表，其次向内容源文件中添加带有 media 属性的各种样式表伪指令，最后在 Route 方法里进行转换，输

出相应的视图。

(1)编辑不同的样式表，针对不同的媒体转换内容源的格式，向 XML 文件中添加带有 media 属性的各种样式表伪指令，如下所示：

```
//xml 文件
<?xml-stylesheet href=http://localhost:7001/moblile.xsl
    type="text/xsl" media="mobile"?>
<?xml-stylesheet href=http://localhost:7001/browser.xsl
    type="text/xsl" media="general"?>
……
```

(2)Route 方法根据以上样式表伪指令，为不同的用户界面类型调用不同的样式表进行转换，输出相应的视图。

下面是视图选择器的 route 方法示意代码：

```
//ViewRouter 的 route 方法
  Public void route(GenericServlet servlet,HttpServletRequest req,
HttpServletResponse res)throws ServletException,IOException{
//创建内容源、样式源和结果目标
StreamSource   contentSource = new StreamSource(……);
StreamSource   styleSource =
      transformerFactory.getAssiociatedStylesheet(contentSource,"general",null,null);
StreamResult   result = new StreamResult(out);
//创建 Transformer 对象、执行转换
Transformer transformer = fransformerFactory.newTransformer(styleSource);
Transformer.transform(contentSource,result);
}
```

Route 方法根据样式表伪指令基于用户代理程序进行选择。USER-AGENT 头中包含了用户上网时的 IP 地址和所用浏览器的类型，该信息以请求头的形式被发送至服务器。控制器检测 USER-AGENT 信息，若浏览器类型为 WAP，则选择 media="mobile",否则 media="general"，并在创建样式表源时加入相应的参数，这样就实现了根据浏览器类型选择相应的样式表进行显示。

但是，将浏览器代理程序的每种可能组合硬编码到应用程序会不利于维护和扩展，可以创建一个表示浏览器类型和样式表媒体类型对应信息的属性文件，控制器读取并分析此属性文件，然后选择相应样式表。下面是 browser.properties 属性文件的可能内容：

MSIE=general

UP.=mobile

在控制器中使用 getMedia 方法创建 Properties 对象并装入 browser.properties 属性文件，然后查找每个可用的属性，查找与用户代理程序匹配的属性并返回 media 值。

(3)使用配置文件

Struts 配置文件思想的引入使控制器中调用视图文件名不用硬编码在程序中，而只需要一个逻辑名，由配置文件对逻辑文件名进行实际文件名的映射，极大的降低了文件名和程序代码的耦合。本章改进的控制器对跳转视图文件名的调用也采用 Struts 配置文件的技术，实现对视图文件的集中管理，并在第四章对配置文件管理上的不足进行分析，提出一个解决方案。

下面是本文改进的控制器的实现代码：

```
//用户发出请求
public class ControlServlet extends HttpServlet{
//初始化操作库
public void service(HttpServletRequest req,HttpServletResponse res)
    throws java.io.IOException,javax.servlet.servlet,ServletException{
  Action action=factory.getAction(getClassname(req),getClass().getClassLoader());
//由操作库中获取用户请求的相应操作
  ViewRouter router=action.perform(this,req,res);
//调用相应操作的 perform 方法，执行业务代码，调用帮助类的 toXML 方法，把需要返回的数据转换为 XML 格式，返回视图选择器。
  Router.route(this,req,res);
//调用 route 方法，根据请求的类型调用相应的样式表显示结果数据
  Private String getClassname(HttpServletRequest req){
//获取对 servlet 路径的引用，并提取斜线到句点之间的字符串，将 URL 映射到操作类名
```

综上所述，本文设计的控制器能够检测到传统和移动浏览器发出的请求，并自动选择样式表进行响应。下面是控制器对于 Web Service 请求的支持。

3.3.4 改进控制器对 Web Services 的支持

创建 Web Services 可以使用工具开发。本文使用 WebLogic 来包装控制器为 Web Services，创建了一个控制器包装程序的代理（proxy）。代理接收 SOAP 消息并进行解码，确定正在请求的方法。随后，它执行该方法并将结果编码在 SOAP 响应中。

下面为 Web Services 创建新的样式表，如下所示代码：

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="/">
    <xsl:if test="$news='all'">
        <xsl:apply-templates select="/news/news"/>
    </xsl:if>
    </xsl:template>
    <xsl:template match="text()">
    <xsl:apply-templates/>
    </xsl:template>
```

```
        <xsl:template match="news_title">
        <xsl:value-of select="." />
        </xsl:template>
        <xsl:template match="news_time">
        <xsl:value-of select="." />
        </xsl:template>
</xsl:stylesheet>
```

接下来将新的样式表也添加到 XML 文件：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="http://localhost:7001/ webservice.xsl"
type="text/xsl"
media="webservice" alternate="yes"?>
<?xml-stylesheet href="http://localhost:7001/mobile.xsl"
type="text/xsl"
media="mobile" alternate="yes"?>
<?xml-stylesheet href="http://localhost:7001/browser.xsl"
type="text/xsl"
media="general" ?>
<news>
...
```

然后，就可以把控制器包装成 Web Service，使用工具为控制器创建代理服务，并生成测试客户机调用控制器的 Web Services，控制器的所有可用方法都可以展现给客户端。由此，控制器完成了对 SOAP 消息的响应。

本文在 Struts 控制器的基础上，实现了控制器对 WAP 和 Web Services 的支持，并通过配置文件的映射，使 ViewRouter 对 XML 文件的逻辑调用转换为实际文件名的调用，并结合样式表，产生输出页面，如下图所示。

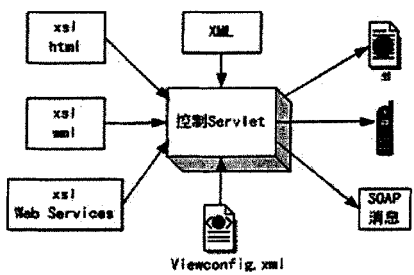


图 3-7 控制器对多类型用户界面的支持

3.4 本章小结

本章首先详细的分析了多类型用户界面的特点，指出 Struts 框架在支持多

类型用户界面上的不足。然后分析了多类型用户界面的支持机制，在 Struts 框架控制器的基础上提出了对其控制器的改进方法，引入 XML 和样式表，在 ViewRouter 中实现了检测用户请求类型并调用相应的样式表进行显示，避免了在视图文件中嵌入 Struts 标签。并且由于 XML 和样式表的优点，能灵活的产生用户需要的视图类型。

改进前的 Struts 控制器只能支持单一的 Web 用户界面，对于大型电子商务平台中需要的无线移动设备及平台间互相提供服务支持不足，不能满足 Web 应用发展出现的新的特点。改进后的控制器扩展了 Struts 对于多类型用户界面的支持，适应了目前 Web 应用的要求。

本章还参考 Struts 的配置文件实现原理，对用作视图输出的 XML 文件进行集中控制。配置文件思想的引入使控制器中调用视图文件名不用硬编码在程序中，而只需要一个逻辑名，由配置文件对逻辑文件名进行实际文件名的映射，极大的降低了文件名和程序代码的耦合，但如果 Web 应用规模加大，配置文件将变得十分庞大，难于管理和配置，下面将进一步增加对配置文件的管理，使用 JMX 技术为配置文件创建管理器，增加配置文件的可管理性。

第四章 基于 JMX 的 Web 应用框架配置文件的管理

本章参考 Struts 的配置文件的实现原理，在对基于 MVC 模式的 Struts 框架进行支持多类型用户界面的改进时，建立一个配置文件 viewconfig.xml，实现对 xml 文件选择输出的集中管理，避免在视图选择器中的硬编码，但随着 Web 应用规模的扩大，带来了配置文件庞大且难于配置的不足。基于 JMX 的简单灵活和可动态实现热插拔的特性，本章创建了一个基于 JMX 的配置文件管理系统，实现对配置文件的灵活方便的管理，使管理系统与被管理系统很好的实现了隔离。下面先了解 JMX 的体系结构和功能。

4.1 JMX 的体系结构和功能

JMX(Java Management Extensions) 是受欢迎的新的 Java 平台标准扩展，它在可管理性方面的特性扩展了 Java 平台，定义了一个包含若干中心组件的管理体系结构，是一套标准的代理和服务，用户可以在任何 Java 应用程序中使用这些代理和服务实现管理。支持通过现代网络管理系统（Network Management System）或企业管理系统（Enterprise Management System）对设备、应用程序和服务进行管理、控制和监控[18]。

JMX 体系结构分为三个层次：设备层(Instrumentation Layer)、代理层(Agent Layer)和分布服务层(Distributed Layer)，如下图所示[20]。

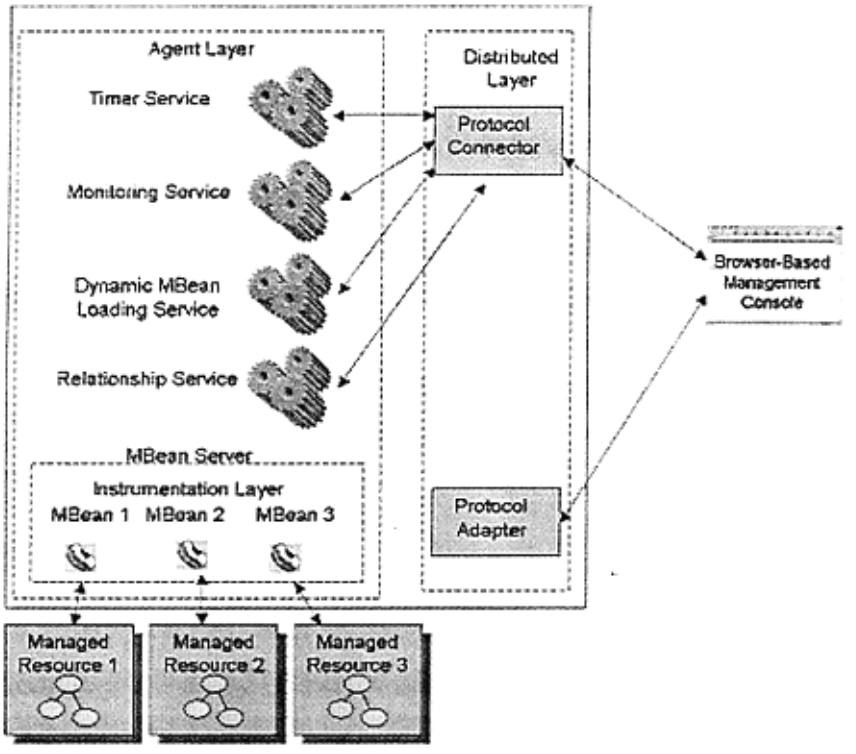


图 4-1 JMX 体系结构图

4.1.1 设备层(Instrumentation Layer)

设备层主要包括了一系列的接口定义和描述如何开发 MBean 的规范。MBean 对可管理资源进行封装和抽象，可管理资源可以是应用程序、设备，也可以是某种服务和策略的软件实现。一个管理资源由一个或多个 MBean(Managed Bean)组成，MBean 是从 JavaBean 的思想演化而来的，具有封装、重用和继承等特点。因此，MBean 必须遵循 JavaBean 组件规范，这样就提供了一个到 JavaBean 组件的直接映射。封装后的 MBean 对外提供统一的操作接口：属性、操作和通告，用 getter/setter 来进行每个属性的读写。

4.1.2 代理层(Agent Layer)

代理层直接控制可管理资源并使它们能被远程的管理者访问，它由 MBean 服务器和代理服务两个组件组成，通常和它们所控制的资源运行在同一台机器上。

MBean 服务器是代理层的关键组件，设备层的 MBean 只有在 MBean 服务器上注册，才能由管理者对它进行管理。MBean 服务器把 MBean 所实现的管理接

口中的内容暴露给外界管理者，使外界管理者可以管理 MBean，但不能直接引用 MBean。

代理服务组件实现对注册在 MBean 服务器的 MBean 的管理操作和提供服务，它本身也是以 MBean 的形式注册在 MBean 服务器中的，因此外界可以通过 MBean 服务器控制它们。在 JMX 规范中，定义了四种代理服务[21]：

(1)动态类载入：通过这种服务可以实例化从网络下载的类或本地库文件。

(2)监控：当有关 MBean 的一些特性发生改变时，可以根据特定条件把这些改变通知给监听者。

(3)定时：按设定的时间发送消息，可用于调度。

(4)关系：定义 MBean 之间的相互联系，并使之保持一致。

4.1.3 分布服务层

分布服务层定义了外界访问代理层的接口和组件，最重要的和必备的组件是连接器和适配器，它把管理组件的管理指令映射为代理层可以接受的指令，把代理层的信息传送给管理组件。各种管理组件可以分布在网络中的各个位置，它们之间相互配合，提供了分布式、规模化的管理功能。该层定义的接口和组件提供了以下功能：

(1)使管理者能通过连接器透明地和代理层及其 MBean 交互。

(2)把代理层及其 MBean 暴露给外界的接口内容通过协议适配器映射到数据传输协议(如 HTML 或 SNMP)。

(3)把上层管理平台的管理信息发布给很多的代理服务组件。

(4)把从各种代理服务组件得到的管理信息组合成逻辑视图信息再和最终用户交互。

4.1.4 JMX 的优点

JMX 具有以下的作为管理系统的优势：

(1)简单灵活。JMX 体系结构依赖于一个作为管理代理的核心 MBean 服务器，该服务器能运行在大部分的 JAVA 使能的设备上。

(2)伸缩性强。JMX 的管理构件可根据管理应用的需要随时插入到 JMX 代理中来实现管理服务，可实现热插拔。

(3)基于管理目的而产生。JMX 是专为管理而产生的，具有很强的针对性。JMX 提供的一系列服务的目的是为了管理网络、系统和应用程序。

4.2 配置文件的使用

参考 Struts 配置文件的实现原理，本节实现两个配置文件 web.xml 和 viewconfig.xml，web.xml 映射用户请求到控制 Servlet，viewconfig.xml 则实现 route 方法中选择的 XML 文件的软编码配置，下面加以介绍。

为了将用户请求映射到控制 Servlet，需要在 web.xml 部署描述信息把以. do 结尾的 URL 映射到控制 Servlet，如下所示的配置文件(web.xml)的示例代码。

```
//WEB-INF/web.xml
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>ControlServlet</servlet-class>
<servlet>
<servlet-mapping>
   <servlet-name>action</servlet-name>
   <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

用来被转换为视图输出界面的 XML 文件在 viewconfig.xml 中进行配置，方便 route 方法对跳转页面的集中控制，如下示例代码：

```
<action path="/test/login" type="com.webapp.loginAction" input="addBackView" >
        <forward name="success" path="view/loginSuccess.xml"/>
</action>
<action   path="/test/showdetail" type="com.webapp.showDetail" >
        <forward name="success" path="view/showDetail.xml"/>
</action>
```

由上代码可以看出，viewconfig.xml 主要是对要跳转页面的实际文件名和逻辑文件名的映射。如果 Web 应用规模增大，要显示的视图文件必然会增加，viewconfig.xml 文件将会变得很庞大。如何有效地管理配置文件将会成为开发者需要考虑的问题。基于上节介绍的 JMX 的简单灵活和可动态实现热插拔的特性，下面创建一个基于 JMX 的配置文件管理系统，实现对配置文件的灵活方便的管理，使管理系统与被管理系统很好的实现了隔离。

4.3 Web 应用框架配置文件的 JMX 管理器

4.3.1 创建 JMX 管理器

由于 JMX 作为管理系统的优势，采用 JMX 技术管理配置文件。使用 JMX 为配置文件创建管理器，可以分为以下四个步骤：

(1)创建管理配置文件的 MBean

根据 JMX 的规范，MBean 有标准的和动态的两种主要类型。

标准 MBean 的设计和实现比较简单，其管理接口通过方法名来描述。与标准 MBean 相关联的所有属性、操作和事件都在其接口中静态地指定。它们是固定的，不随时间变化而变化。标准 MBean 的实现依靠一组命名规则，称之为设计模式。这些命名规则定义了属性和操作。检查标准管理构件接口和应用设计模式的过程被称为内省（Introspection）[18]。JMX 代理通过内省来查看每一个注册在 MBean 服务器上的标准 MBean 的方法，辨认出它的属性和操作。因此对于标准管理构件，JMX 代理通过内省机制来获取该构件的管理接口。

动态 MBean 提供了更大的灵活性，它可以在运行期暴露自己的管理接口。所有动态 MBean 都实现 javax.management.DynamicMBean 接口。通过使用 DynamicMBean 接口，与 MBean 相关联的一组属性、操作和事件直到运行时才是确定的。这满足了拥有可能随时间改变而改变的属性、操作和事件的可管理 JMX 资源的需要。

动态 MBean 有两种典型的 MBean：模型 MBean 和开放 MBean，模型 MBean 是预制的、通用的 MBean ，已经包含了所有必要缺省行为的实现，并允许在运行时添加或覆盖需要定制的实现。JMX 规范规定该类必须实现为 javax.management.modelmbean.RequiredModelMBean，管理者要做的就是实例化该类，并配置该构件的默认行为并注册到 JMX 代理中，即可实现对资源的管理。

在 Web 应用中，资源是多元化的，如运行实例，静态文件，设备或者是硬件状态，可以把这些资源分为两类：需要进行动态管理监控的资源（如登陆用户数，数据库连接监控，即时日志等）和只需进行静态管理的资源（如系统配置文件，日志级别等），要管理这些不同类型的资源，这就要求 MBean 一方面能够直接调用运行实例提供的接口，另一方面又希望 MBean 自身能够提供静态资源的管理，模型 MBean 能够很好的满足这样的要求，并且对于新增需要管理的资源，提供了很好的灵活性和可扩展性，因此本文使用模型 MBean 管理配置文件。

ViewConfig 是本文对配置文件 viewconfig.xml 创建的 MBean，它的作用是对 viewconfig.xml 封装成 MBean，该 MBean 包括四个属性及其相关的 getter 和 setter 和一个 save 方法，其中 save 方法的作用是属性存入配置文件，从而实现了通过 MBean 直接对配置文件进行操作。首先创建一个 ViewConfigMBean 接口，如下示例代码：

```
public interface ViewConfigMBean {
    public String getPath();
    public void setPath(String path);
    public String getType();
    public void setType(String type);
    public String getName();
    public void setName(String name);
```

```
    public String getPathName();
    public void setPathName(String pathname);
    public String save()
}
```

然后建立 ViewConfigMBean 的实例，如下代码：

```
public class ViewConfig implements ViewConfigMBean {
    private String path;
    private String type;
    private String name;
    private String pathname;
    private static final String
        CONFIG_FILEPATH = "http://localhost:7001/WEB-INF/viewconfig.xml";
        private void init() {
        InputStream in = null;
    //读入配置文件，并读取属性名
    //get 和 set 方法设置配置文件属性
    }
    public String save() {
    //将修改后的属性存入配置文件
    }
}
```

由此，创建了管理配置文件的 MBean。

(2)创建 MBean 描述文件

将 MBean 注册到 MBean Server 中必须先创建 MBeanInfo，MBean 的 setModelMBeanInfo()用来将 MBeanInfo 设置到 MBean 中。使用 MBean 描述文件的方式创建 MBean，下面是描述文件的示例代码：

```
<mbean-list>
<mbean    name="ViewConfigMBean"
            className="com.myApp.jmx. ViewConfigMBean"
                domain="myapp">
    <attribute    name="path"
            type="java.lang.String"
                writeable="false"/>
    <attribute    name="type"
            type="java.lang.String"/>
    <attribute    name="name"
        description="output file name"
                type="java.lang.String"/>
    <attribute    name="pathname"
            type="java.lang.String"/>
    <operation    name="save"
        description="save the configuration"
                impact="ACTION"
```

```
            returnType="java.lang.String">
        </operation>
    </mbean>
</mbean-list>
```

(3)创建 MBean 服务器。

通过 JMX API 中 MBeanServerFactory 类的 createMbeanServer()的方法创建 MBeanServer 的实例，以下是创建的语句：

```
MBeanServer server = MBeanServerFactory.createMBeanServer();
```

(4)注册 MBean

将上文创建的 ViewConfig MBean 注册到 MBean 服务器中，下面是注册代码：

```
//注册 mbean 到 MBean Server 中
MBeanServer serv = getMBeanServer();
ViewConfigMBean viewConfig1 = new ViewConfig();
ViewConfigMBean viewConfig2 = new ViewConfig();
ObjectName viewObjectName1 = new ObjectName(
"HelloServer:type=ViewConfig,name=viewConfig1");
server.registerMBean(viewConfig1, viewObjectName1);
ObjectName viewObjectName2 = new ObjectName(
"HelloServer:type=ViewConfig,name=viewConfig2");
server.registerMBean(viewConfig2, viewObjectName2);
```

(5)通过客户端访问 MBean

JMX 提供了 HTMLAdapter，本例设置端口号为 8082，使用户可以通过浏览器访问 MBean，如下代码：

```
HtmlAdaptorServer adapter = new HtmlAdaptorServer();
ObjectName adaptorObjectName = new ObjectName(
"HelloServer:type=htmladaptor,port=8082");
adaptorServer.setPort(8082);
server.registerMBean(
adaptorServer, adaptorObjectName);
adaptorServer.start();
```

在浏览器输入 http://localhost:8082，显示如下页面。

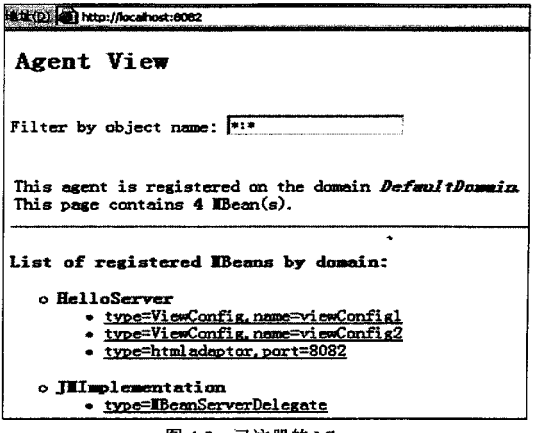


图 4-2　已注册的 Mbean

可以配置 ViewConfig1 的界面如下图。

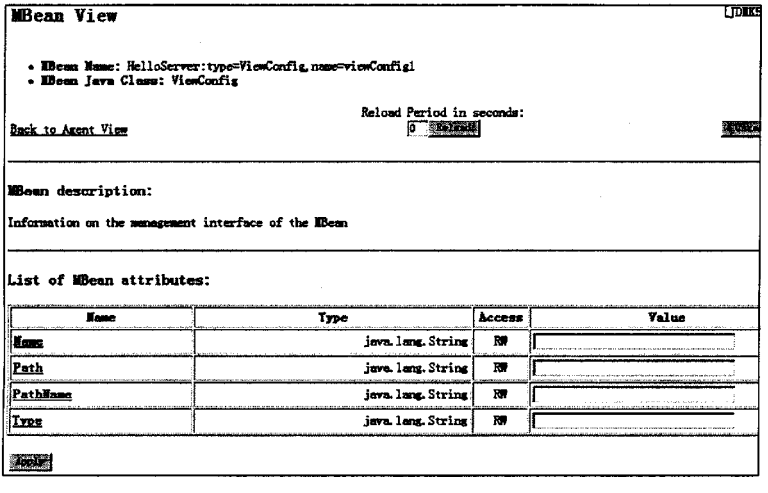


图 4-3　对于配置文件管理的界面

4.3.2 管理器的优点

综上所示，在增加新的子应用时，配置文件的修改需要编辑原配置文件，手动的在其中加入各种配置信息，如用户请求的路径映射到那个 Action，Action 所

对应的 ActionForm、forward 和 input 等信息。这种修改方式界面不友好并容易出错，开发者需要记忆各种参数及配置格式。特别是 Web 应用规模增大时，需要配置的 Action 增多，导致配置文件增大，不易管理和修改。而使用基于 JMX 的配置文件管理器，用户界面友好，基于 GUI 方式而不是基于字符的，需要配置的 Action 的信息、参数及其类型一目了然，不用记忆配置格式，只需填入数据即可。并且此管理器的存在与否对 Struts 的配置文件无任何影响，开发者仍可以不使用此管理器直接修改原配置文件以字符方式进行配置，就好像管理器不存在一样，实现了管理器的动态热插拔。因此，此管理器具有方便、灵活及动态热插拔的优点。

4.4 本章小结

由于 Web 应用规模的增大，配置文件也变得日益庞大。而对配置文件的管理还是采用字符方式，在记事本里增加对于各种参数的配置，需要记忆各种参数及格式，管理用户界面不友好，难于配置和管理的问题日益突出。本章采用 JMX 技术，实现了对配置文件的方便灵活的管理，并且此管理器具有可动态实现热插拔的优点，不用修改原有代码，即可实现对配置文件的管理，使管理系统与被管理系统做到了很好的隔离。下面结合具体的实例，验证本文改进的 Web 应用框架的各种优点。

第五章 Web 应用框架的实现：网上数码冲印系统

随着 Internet 宽带网络的普及以及网上信用体系的逐渐健全，电子商务得到了很大的发展，电子商务的成交额也在不断的激增，对电子商务平台的开发也提出了越来越高的要求，本文以电子商务的一个应用—网上数码冲印系统为例，实现一个电子商务系统，以验证本文的基于MVC模式的Web应用框架的可扩展性、易维护性及支持多类型用户界面的优势。

网上数码冲印，是指用户在网上直接将数码照片资料传给专业的数码冲印网站，并根据网上的报价选择自己需要的服务，冲印好以后再将照片寄给用户。使用此服务，用户只需操纵电脑，即可完成全部冲印过程。

网上数码冲印有许多吸引用户的优势：

(1)价格优惠。由于省去了店面上的开销，网上数码冲印所提供的价格往往比数码冲印店更加优惠。

(2)多元化的服务。除了可以在线冲印数码照片外，网上冲印店还可提供各种多元化、个性化的数码增值服务，包括使用数码照片制作马克杯、马克盘、台历和挂历等。

(3)方便快捷。用户可以足不出户即可完成送资料、取相片的工作，多种付费方式、多种配送模式以及电子邮件通知发货等服务能为用户最大限度地节约时间。

网上冲印的诸多优势使其迅速的发展起来，对网上数码冲印系统的健壮性，灵活性，易维护性的要求也越来越高。

5.1 需求分析概述

5.1.1 客户端模块

下面是客户端功能模块。

(1)会员注册。网上冲印系统实行会员制，每一位用户拥有唯一的会员号和唯一的相册名，用户只需花一两分钟填写完必要的个人资料便可以完成注册，不会有冗长繁杂的选项。

(2)找回密码。如果会员忘记了自己登录密码，可以通过系统“找回密码”功能回答出注册时设置的“忘记密码问题”的答案，然后重新设置便于自己记住的密码；如果连答案也记不住的话，那只好到冲印店凭身份证及电话号码（手机

号码）修改密码了。

(3)修改资料。本系统只认准唯一的会员号，用户的个人信息如：电话、地址、邮箱等有变动，可直接在会员专区进行修改。

(4)发送订单（传送图片及其相应的订单信息）。发送订单是本系统的一个重要流程，用户通过此流程将需要冲印的数码相片及其规格、数量、相纸、附加服务、尺寸处理方式、图片亮度对比度处理方式、配送方式等所有信息一起发送到服务器端供冲印店操作员处理，系统同时将这一订单的总费用保存到用户及服务器端的历史记录中。如下图所示：

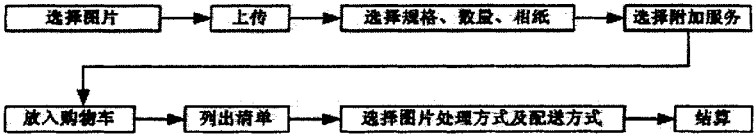


图 5-1　网上数码冲印流程

(5)查看订单。订单发送完成，用户可在会员专区“查看订单”页中查询所有订单记录，便于日后的收件核对、资金对帐等。

(6)资金查询。可查询每笔订单的订单号、交易金额、发生日期、支出记录、退款记录及存入总金额、消费总金额、余额总数，明细帐一目了然。

(7)充值记录。用户可查看通过现金缴款、网上支付、银行转帐、邮寄等方式的充值记录，以便资金核对。

(8)网上支付。可绑定网上支付平台如招商银行的“一网通网上支付系统”等。

(9)发票开具。系统提供快捷式发票开具功能，会员只需简单填写发票开具要求，服务器端便会收阅,并做出处理。

5.1.2 服务器端模块

服务器端实现的各功能模块如下：

(1)订单管理

最新订单：操作员一进入后台便会看到最新订单的醒目提示，操作员根据订单信息用 FTP 软件下载相片备份并打印出订单转交给图像处理员进行相片修饰调整，出片后再根据订单上的配送方式交至客户手中。

订单列表：可查询所有订单信息，便于日后管理及核对；

作废订单：一些作废订单、垃圾图片（如像素不够、图片出错）的回收站，

同时将作废订单或图片的相应款项退给会员；

订单统计：对当前所有订单数据的日、月、季度、年度统计，便于冲印店的帐务管理，经营状况的分析；

(2)会员管理

会员管理：会员资料查询、核对、备份，会员密码查询，会员操作权限的锁定、解锁，会员删除等功能；

会员充值：将会员的真实入帐（如现金支付、银行转帐、邮政汇款）充入到系统会员帐号中，供会员进行网上消费；

(3)资金管理

资金统计：详细记录当前所有会员、单个会员的存入总金额、消费总金额，余额总数及每个会员的资金记录；

退款记录：会员作废订单、作废图片的相应退款记录，便于日后查询核对；

充值记录：当前所有会员的充值记录，便于日后查询核对；

银行支付记录：单独记录当前所有会员的网上支付记录，便于日后查询核对；

(4)公告版

添加公告：发布冲印店服务公告、打折公告、通知、新闻等信息；

公告管理：公告的修改、删除功能；

(5)发票管理

最新发票：查看最新发票开具的要求，并对照其要求进行发票填写并配送；

已开发票：查看已开发票的历史记录，以便日后查询核对；

(6)疑难解答

管理留言：管理当前会员的疑难、留言及建议等信息，及时发现问题、解决问题以提高服务质量；

(7)配送方式

修改冲印店所提供的配送方式、配送价格及送货上门的消费金额底限；

(8)系统说明

添加说明：可将系统使用中常见问题整理发布，供用户参考学习；

说明管理： 修改、删除系统说明记录；

(9)公司信息

添加信息：添加公司简介、联系方式、银行帐号、客户服务信息，便于用户联系；

信息管理：公司信息的修改删除；

(10)系统管理

管理规格：管理各冲印店所能提供的冲印规格信息，此项设置同样必须在系统对外发布前设置完成，系统运行期间不能删除，只能对其进行修改或添加；

管理价格：管理每种规格相应的附加服务、价格及像素要求，此项设置同样必须在系统对外发布前设置完成，系统运行期间不能删除，只能对其进行修改或添加；用户订单中的图片清晰度便是通过价格表中的像素要求来判断的；

添加价格：与添加规格的同时添加其相应的价格；

用户管理：管理服务器端的用户，如修改管理员密码，添加、修改、删除操作员等功能；

5.2 系统实现

5.2.1 系统平台实现

本系统采用 B/S 模式，用户可以通过不同类型的浏览器输入请求信息，Web 服务器和 EJB 容器选用 BEA WebLogic platform 8.1，系统服务器端程序存放于此。数据库服务器选用 MYSQL 数据库，系统的数据包括用户注册信息、冲印店所发布的信息、用户的订单记录均存放在数据库服务器上，以实现管理员的动态管理。下面是平台配置情况：

(1)服务器：操作系统采用 Windows2000，应用服务器为 BEA WebLogic platform 8.1，数据库采用 MYSQL

(2)客户端：Windows2000/98,IE4.0 以上，UPSDK 4.0 手机模拟器

(3)开发工具：Jbuilder9.0，Dreamwaver Ultradev4.0

体系结构设计基于本文的多类型的 Web 应用的体系结构，整个系统分为三层，客户端发出 HTTP 或者是 WAP 请求，控制器接收请求，由控制 servlet 统一转发请求，调用相应的 Action，访问模型层的业务逻辑，由处理业务逻辑的 EJB 必要时访问实体 bean，由此可修改和提取数据库的相应信息，然后转交给 ViewRouter 进行必要的转换，由此产生显示给特定客户端的页面，并由控制 Servlet 响应给客户端。

5.2.2 数据库的实现

基于以上需求分析，设计如下数据表：

订单表：订单号、数量、规格、附加服务、配送方式、是否已处理标志。

会员信息表：会员 ID 号、会员昵称、会员密码、会员 Email。

新闻表：新闻标题、新闻时间、新闻内容、新闻起源、新闻发布者。

留言表：留言标题、留言时间、留言内容、留言作者。

下图为其中的订单表和新闻表的 Entity Bean 的设计视图

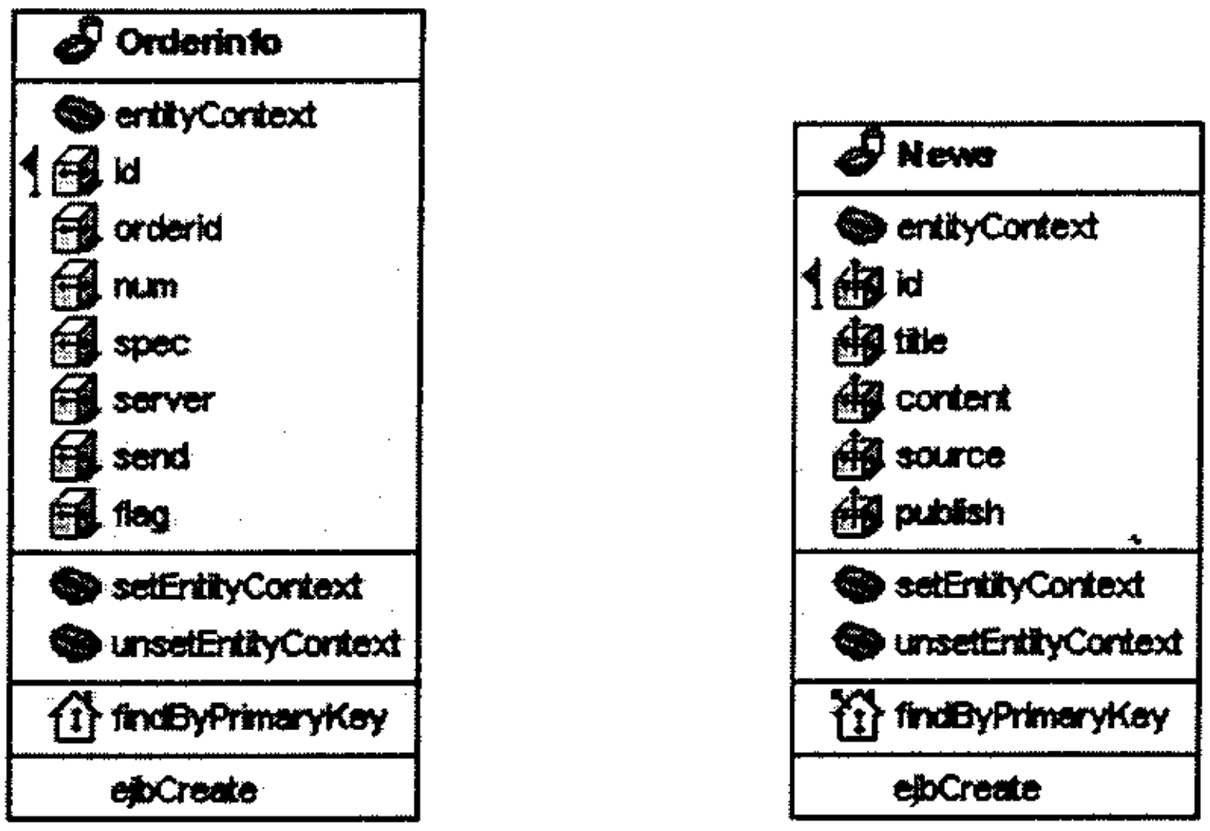


图 5-2　订单表和新闻表的 EJB 设计图

5.3 网上数码冲印系统的详细实现

5.3.1 整体设计体系结构

整个系统设计为视图层、控制层、模型层和资源层，如图 5-3 所示。其中控制层和模型层放在应用服务器上实现。整个系统使用统一的控制器，通过调用不同的样式表，产生响应客户端的视图。

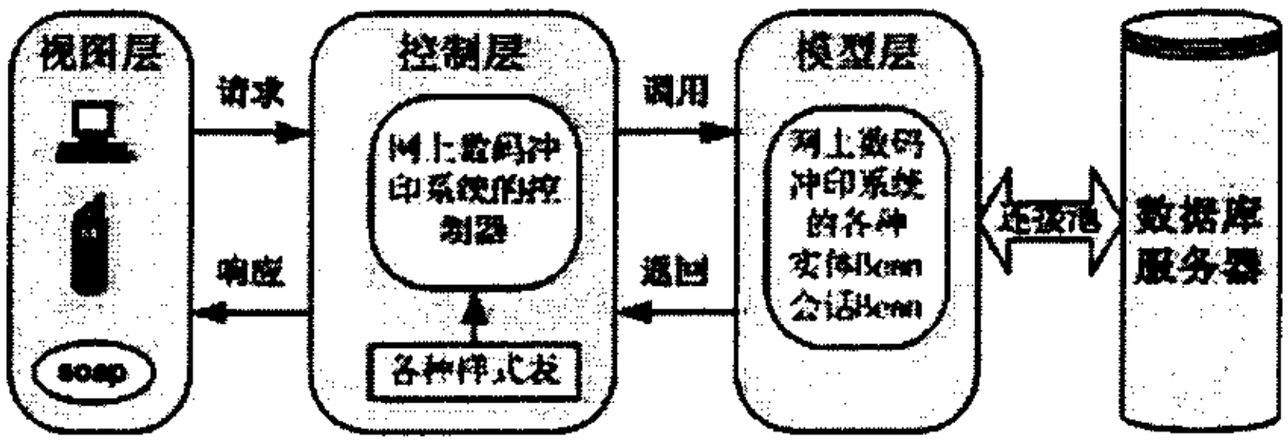


图 5-3　网上数码冲印系统体系结构

5.3.2 控制器的实现

(1)控制器的核心是控制 Servlet 的设计，控制 Servlet 的控制流程第三章已介绍过，下面列出它的主要实现代码：

```
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ActionServlet extends HttpServlet {
//产生操作库
```

```
private ActionFactory factory = new ActionFactory();
public void init(ServletConfig config) throws ServletException{
//初始化 Servlet，从属性文件中创建资源包
    }
public void service(HttpServletRequest req, HttpServletResponse res)
  throws java.io.IOException, ServletException {
        try {
    String actionClass = getActionClass(req);
//从操作库中获取操作
    Action action = factory.getAction(actionClass,
                                      getClass().getClassLoader());
//调用操作的 perform 方法，完成业务逻辑处理，并返回视图选择器
    ViewRouter router = action.perform(this,req,res);
//调用视图选择器的 route 方法，产生针对特定用户请求的响应视图
    router.route(this, req, res);
      }
       catch(Exception e) {
           throw new ServletException(e);
      }
    }
// getActionClass 方法提取斜线到句点间的字符串，获得用户请求的操作类名
private String getActionClass (HttpServletRequest req) {
       String path = req.getServletPath();
       int slash = path.lastIndexOf("/"),
           period = path.lastIndexOf(".");
       if(period > 0 && period > slash)
        path = path.substring(slash+1, period);
       return path;
}
}
```

Action 的主要功能是根据具体的请求实现一定的业务功能，下面是 Ation 的接口代码，它定义了一个 perform 方法，向控制 Servlet 传递引用：

```
package actions;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public interface Action {
    public ActionRouter perform
        (HttpServlet servlet,HttpServletRequest req, HttpServletResponse res)
        throws java.io.IOException,javax.servlet.ServletException;
}
```

操作库中可以保持操作的哈希表，若需调用的操作未存储在操作库的哈希表中，操作库首先创建它，然后将其存储在哈希表中，以后再遇到相同操作的请求，

操作库将仅仅从哈希表中返回一个引用，下面是示例代码：

```
package actions;
import java.util.Hashtable;
public class ActionFactory {
    private Hashtable actions = new Hashtable();
    public Action getAction(String classname, ClassLoader loader)
    throws ClassNotFoundException, IllegalAccessException,
                InstantiationException {
    Action action = (Action)actions.get(classname);
    if(action == null) {
            Class klass = loader.loadClass(classname);
            action = (Action)klass.newInstance();
            actions.put(classname, action);   }
        return action;
    }
}
```

ViewRouter 用于调用相应的样式表产生结果文件，下面是示例代码：

```
package actions;
import java.util.ResourceBundle;
import javax.servlet.GenericServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class ViewRouter {
// getTransform 方法根据 media（用户界面类型），采用相应的样式表，输出结果文件
public static String getTransform(String media, String thisStory)
// getMedia 方法用于获取客户端的界面类型
public static String getMedia(String agent){
    String media = "general";
    Properties browserProps = new Properties();
    try {
        browserProps.load(new java.io.FileInputStream("h:\\browser.properties"));
        Enumeration allBrowsers = browserProps.propertyNames();
        while (allBrowsers.hasMoreElements()){
            String thisBrowser = allBrowsers.nextElement().toString();
            if (agent.indexOf(thisBrowser) > 0) {
                media = browserProps.getProperty(thisBrowser);
                break;
            }    }
    } catch (FileNotFoundException e) {
            e.printStackTrace();
    } catch (IOException e) {
            e.printStackTrace();
    }
return media;
```

```
  }
// route 方法被控制 Servlet 调用
public synchronized void route(GenericServlet servlet, HttpServletRequest req,
HttpServletResponse res) throws java.io.IOException, javax.servlet.ServletException {
ResourceBundle bundle = (ResourceBundle)servlet. getServletContext().
        getAttribute("action-mappings");
String url = (String)bundle.getObject(key);
String media = getMedia(req.getHeader("USER-AGENT"));
String thisStory = req.getParameter("storyid");
if(isForward) {
servlet.getServletContext().getRequestDispatcher(
getTransform(media, thisStory)).forward(req, res);
        }
 else {
res.sendRedirect(getTransform(media, thisStory)); }
 }
}
```

下面举例说明一个登录操作的实现代码：

```
package actions;
import javax.servlet.ServletContext;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import beans.LoginDB;
import beans.User;
public class LoginAction implements Action {
    public ActionRouter perform(HttpServlet servlet,
                                HttpServletRequest req,
                                HttpServletResponse res)
                          throws java.io.IOException,
                                 javax.servlet.ServletException {
        LoginDB loginDB = getLoginDB(servlet.getServletContext());
        User user = loginDB.getUser(req.getParameter("userName"),
                                    req.getParameter("password"));
        if(user != null) { // user is in the login database
            req.getSession().setAttribute("user", user);
            return new ActionRouter("welcome-page");
        }
        else { // store userName request parameter in session
            req.getSession().setAttribute("userName",
                                  req.getParameter("userName"));
            return new ActionRouter("login-failed-page");        }
    }
    private LoginDB getLoginDB(ServletContext context) {
```

```
        LoginDB loginDB = (LoginDB)context.getAttribute("loginDB");
        if(loginDB == null)
            context.setAttribute("loginDB", loginDB = new LoginDB());
        return loginDB;
    }
}
```

5.3.3 模型的实现

(1)实体 bean 的设计。

根据需求分析，设计主要的实体 bean 如下：

订单 bean(OrderinfoBean)、新闻 bean(NewsBean)、会员信息 bean(UserBean)、留言 bean(AdviseBean)，分别代表订单表、新闻表、会员信息表、留言表。

(1) 会话 bean 的设计

本文主要是开发一个有状态的会话 Bean(orderCartBean)实现订单的提交和一个会话外观 Bean（facadeBean）实现对实体 Bean 访问的封装，下面是会话外观的示例代码：

```
public class facadeBean implements SessionBean {
  SessionContext sessionContext;
   private newsItem newsItem=null;
   private newsItemHome newsItemHome=null;
   private Orderinfo orderInfo=null;
   private OrderinfoHome orderInfoHome=null;
   public void ejbCreate() throws CreateException {
     try{
     Context context = new InitialContext();
     Object ref = context.lookup("java:/comp/env/news");
     newsItemHome= (newsItemHome) ref;
     ref=context.lookup("java:/comp/env/Orderinfo");
     orderInfoHome= (OrderinfoHome) ref;
     }catch(Exception e){e.printStackTrace();}
   }
    public void ejbActivate()
//访问 Bean news 获取公告信息
public java.util.Collection getAllnews() {
 Collection result=new Vector();
 try{
 Collection c1 = newsItemHome.findAllnews();
       System.out.println(c1.size());
     Object o1[]=c1.toArray();
for(int i=0;i<o1.length;i++)
{
```

```
 newsItem=(newsItem)o1[i];
result.add(new news(newsItem.getnewsname(),newsItem.getAuthor(),newsItem.getContent());
}
}catch(Exception e){e.printStackTrace();}
return result;
}
    private   Context getInitialContext() throws Exception {
       //设置相应于 weblogic 服务器的上下文
     }
  public void deleteOrder(String orderid) {
    try{
      orderInfoHome.remove(orderid);
    }catch(Exception e){e.printStackTrace();}
      }
}
```

5.3.4 视图的实现

视图的实现主要是编写各种类型的样式表，由于手机拍照并通过无线网络通过彩信方式上传照片需要专门软件的支持，本实例对于 Web Services 和无线设备的支持通过简单的实例体现，使它们可以查看网上数码冲印服务的公告栏。

第三章已列出了适应 Web Services 请求的公告栏的 XSL 样式表，下面是基于无线设备界面的公告栏样式表：

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
<wml>
   <card title="公告栏">
       <ul>
       <xsl:if test="$news='all'">
          <xsl:apply-templates select="/news/news"/>
       </xsl:if>
       <xsl:for-each select="//news">
         <h2><xsl:value-of select="news_title"/>
             <xsl:value-of select="news_time"/>
         </h2>
     </ul>
   </card>
</wml>
<xsl:apply-templates/>
</xsl:template>
</xsl:stylesheet>
```

5.3.5 系统的组装与部署

应用程序可以按以下步骤进行部署：

(1)配置 Weblogic 域，创建 SERVER

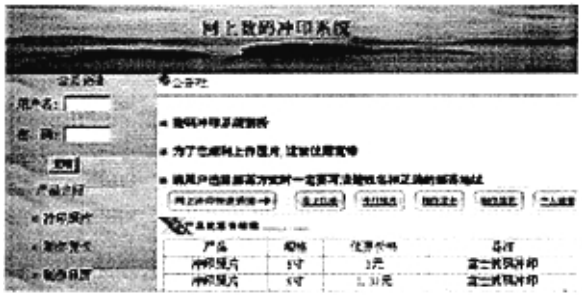
(2)在 JB 中配置 SERVER 信息

(3)配置 JDBC 连接池

(4)创建 EJB，部署 EJB 到 Weblogic 服务器中

5.3.6 运行与测试

(1)Web 浏览的界面，在浏览器的地址栏中输入 http://127.0.0.1:7001/index.htm，出现以下界面：



(2)移动浏览界面，采用 UP.SDK 4.0 手机模拟器，在浏览器的地址栏中输入 http://127.0.0.1:7001/index.wml，显示以下界面：

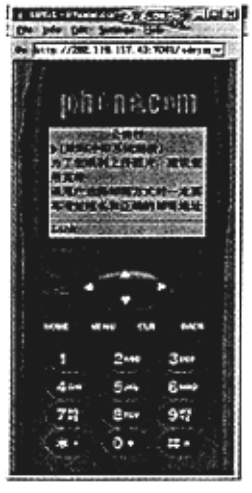


图 5-5 手机输出界面

(3)代理调用 Web Services 的界面。首先使用 Weblogic 创建调用 Web Services 的客户端，然后调用 Web Services，如下图所示。

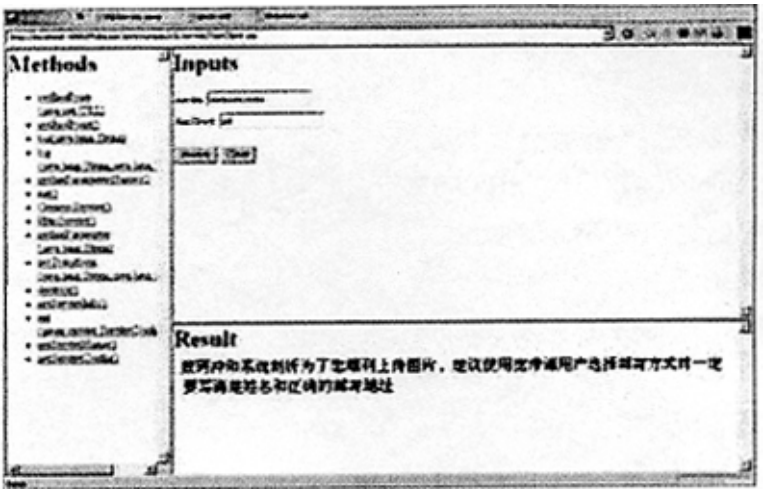


图 5-6　Web Services 输出的结果

5.4 本章小结

本章主要是依据本文改进的应用框架在网上数码冲印系统上做了具体实现，对网上数码冲印系统做了具体的需求分析，并对系统实现的主要模块做出了示例代码。

整个系统实现了三层的分离，并采用论文改进的控制器和视图实现技术，实现了整个系统可以灵活的响应多类型客户端的请求，体现了本文改进的应用框架对多类型用户界面的支持。

一般的构建网上数码冲印系统的技术是使用 JSP 技术，用户请求的接收、业务逻辑的处理、数据的查询与存储和视图的显示全部采用 JSP，程序混杂着代码和显示标记，不易扩展和维护。采用此框架，由于整个系统分为三层，很好实现了网页设计和程序开发的分离，使得整个系统流程清晰，易于扩展和维护。并且随着无线设备的普及，网上数码冲印系统对于无线设备的支持更增加了用户冲印请求的方便快捷。

第四章已针对网上数码冲印系统实现了 viewconfig.xml 配置文件的管理器，方便了开发都对于配置文件的管理，本章没有再加以说明。

第六章　总结和展望

6.1 总结

随着 Internet 的飞速发展和普及，Web 应用越来越显示出多功能化发展趋势，对 Web 应用框架的可扩展性、易维护性、支持多界面性提出越来越高的要求。

本文对 Web 应用框架及其使用的关键技术进行分析，并重点探讨目前流行的基于 MVC 的 Web 应用框架 Struts 框架及其重要技术之一的配置文件的实现，结合 Web 应用的新特点，在 Struts 框架的基础上对其控制器进行改进，实现了控制器对多类型用户界面的支持。并针对配置文件难于管理的问题，基于 JMX 技术创建了一个配置文件的管理系统，实现了对配置文件方便灵活的管理。并在实例中做了具体实现。

本文的主要工作如下：

(1)针对 Struts 框架的不足，在 Web 应用框架中应用面向对象技术及 MVC 设计模式，并对框架实现的关键技术作了详细阐述。

(2)分析了 Struts 框架的优点和不足及多类型用户界面的支持机制，采用 XML 和 XSL 技术，对 Struts 控制器的视图支持部分进行改进，初步实现了控制器对多类型用户界面的支持。

(3)针对 Struts 配置文件不易配置的不足，设计并实现了一个基于 JMX 技术的 Web 应用框架的配置文件管理器，此管理器具有可动态热插拔及灵活方便等特点，使对复杂配置文件的修改操作得以简化。

(4)开发了一个网上数码冲印系统，验证了本文的研究结论。

6.2 工作进一步的展望

本文对多种类型用户界面的支持得出了一个可行的办法，但考虑的还很不完善， Web 应用框架如何寻找，发现其它 Web Services，Web 应用门户网站提供的 Web Services 如何以可视的，面向用户的方式提供给用户都是需要进一步研究和实现的。并且随着语音识别和自动文语转换的技术的进步，控制器需要进一步扩展对语音浏览器的支持。

配置文件的管理器如何以一个更友好的界面显示给用户也需要进一步的探讨和研究。

本文仅对电子商务的一个实例“网上数码冲印系统”做了基本功能的实现。但是，由于新技术的飞速发展，企业信息化水平也明显改善，企业陆续采用 CRM、

ERP、SCM(Supply Chain Manage)等软件应用系统，随着电子商务的发展，如何开发整合这些系统，建立生产、销售、财务和客户关系管理为一体的电子商务平台，是本文需要进一步研究的课题。

参考文献

[1] 阎宏，Java 与模式，电子工业出版社，2002.10

[2] 胡文华　李建民　胡振鹏，模式与设计模式，计算机与现代化，2002.12

[3] Stefan Zeiger，Servlet Essentials，http://www.novocode.com/doc/servlet-essentials/
Web Service for J2EE, Jim Knutson Heather Kreger

[4] 黎升洪，基于 XML 的动态网页方法，计算机与现代化，2001.6

[5] Ed Roman，刘晓华等译，精通 EJB（第二版），电子工业出版社，2003.9

[6] Chuck Cavaness，ProgrammingJakartaStruts

[7] James Goodwill，MasteringJakartaStruts，Wily Publishing，Inc. 2002.12

[8] Enhydra Documentation，http://enhydra.objectweb.org/doc/index.html

[9] 宋磊 任立红 丁永生，基于 WAP 的移动电子商务系统的设计与实现，计算机工程与应用，2003.01

[10] 柴晓路 梁宇奇，Web Services 技术、架构和应用，电子工业出版社，2003.1

[11] Harry M. Sneed，Using XML to Integrate existing Software Systems into the Web，IEEE SOTEWARE，2002.2

[12] 邓贵仕 易峰 李文立，支持 Multi-interface r MVC 设计模式研究，计算机应用，2002.6

[13] Angel Luis Diaz　Peter Fischer　Carsten Leue　Thomas Schaeck，远程门户网站 Web 服务（WSRP），
http://www-900.ibm.com/developerWorks/cn/webservices/ws-wsrp/index.shtml，2002.1

[14] 李劲，动态电子商务的 Web 服务，清华大学出版社，2002.11

[15] Nicholas Chase，用 XSLT 创建多用途 Web 内容，
http://www-900.ibm.com/developerWorks/cn，2003.8

[16] Alan Knight　Naci Dai，Objects and the Web，IEEE SOTEWARE,，2002.4

[17] 刘芳珠　潘亦　潘金贵，基于三层网络架构及 DOM 的 XML 系统模型，小型微型计算机系统，2001.12

[18] Java™ Management Extensions (JMX) v1.2 Specification

[19] Greg Barish，林琪 英宇译，J2EE Web 应用高级编程，清华大学出版社，2002.10

[20] J. Steven Perry :Java Management Extensions.O'Reilly.2002

[21] 闫波，基于 JMX 的应用服务器构件化设计，计算机工程，2003.12

[22] 唐海涛，MVC 在多层 Web 体系结构中的应用研究，2003.3

[23] Jim Knutson Heather Kreger，Web Services for J2EE，
http://www.ibm.com/developerworks/library/ws-jsr109-proposed/

[24] Mario GuillenR.　Ma.del Rosario Vazquez A.，GARP：A Tool for Creating Dynamic Web Reports Using XSL and XML Technologies，IEEE SOTEWARE，2003.2

[25] 何成万　余秋惠，MVC 模型 2 及软件框架 Struts 的研究，计算机工程，2002.6

[26] 邓玉龙，MVC 设计模式在电子商务系统中的研究与应用，南京邮电学院学报，2002.6

[27] 王君　樊治平，一种基于基于 Web 的企业知识管理系统的模型框架，东北大学学报，2003.2

[28] Eckel.B.著，京京工作室译，Java 编程思想（Thinking in Java），北京:机械工业出版社,1999.4

[29] 张波，冯玉琳，黄涛，基于对象视图模型 WebView 的 Web 应用框架，软件学报，2002.10

[30] 熊光彩　莫蓉　赵歆波　张定华，XML 文档对象模型研究与应用，计算机工程与设计，2002.5

[31] 黄骏　张凌　宁国宁，利用 JMX 实现层次化网络管理架构

[32] 刘永峰，JMX 技术在网络管理中的应用研究，2003.3

[33] 袁梅冷　黄烟波　黄家林　翁艳彬，J2EE 应用模型中 MVC 软件体系结构的研究与应用，计算机应用技术，2003.2

[34] The Struts User's Guide，http://Jakarta.apache.org

[35] 贾晓琳，基于 J2EE 的企业级 Web Service 体系结构，计算机工程，2003.11

[36] 林天峰，JMX 资源管理编程，电脑与信息技术，2003.6

[37] 李蜀青　李元萍　李元良，WAP 网站建设的研究，湖南大学学报，2001.6

[38] 孟小峰，Web 信息集成技术研究，计算机工程，2003.8

[39] 孙明　周明天　詹瑾瑜，JMX 动态管理服务的研究与设计，电子科技大学学报，2002.10

附录

下面是论文中出现的英文缩略语对照表，按照字母顺序排列

API：Application Programming Interface

ASP：Active Server Pages

B2B：Business To Business

B2C：Business-to-Customers

CORBA：Common Object Request Broker Architecture

COM/DCOM：Component Object Model /Distributed Component Object Model

DTO：Data Transfer Object,

ERP：Enterprise Resource Planning

EDI：Electronic Data Interchange

EIS：Enterprise Information System

EJB：Enterprise JavaBean

GSM：Global System for Mobile communications

HTTP：Hypertext Transfer Protoco

IIOP：Internet Inter-ORB Protocol

JDBC：Java Database Connectivity

JMX：Java Management Extensions

JNDI：Java Naming and Directory Interface

JSP：Java Server Pages

J2EE：Java 2 Enterprise Edition

MVC：Model-View-Controller

OMG：Object Management Group

PDA：Personal Digital Assistants

RMI：Remote Method Invocation

SGML：Standard Generic Markup Language

SOAP：Simple Object Access Protocol

UDDI： Universal Discovery，Description and Integration

WAP：Wireless Application Protocol

WML：Wireless Markup Language

WSDL：Web Services Description Language

XHTML：Extensible HyperText Markup Language

XML：Extensible Markup Language

XSL：eXtensible Stytlesheet Language

致 谢

本文是在导师江红副教授的悉心指导下完成的。衷心感谢她在我研究生阶段的三年中所给予的悉心关怀和耐心指导，帮助我正确地确立了研究方向和科学的研究方法，并为我提供了良好的学习、研究和开发的环境和机会。江老师严谨、求实的工作作风、缜密的思维以及广博的学识将使我受益终生。同时也要感谢陈晓坤老师所给予的关心。

衷心感谢导师宗平教授。在我的学习过程中，导师所教予我的不仅仅是他严谨科学的治学态度和不懈探索的治学精神，他坦荡的胸怀，谦和的为人，积极乐观的生活态度更是令我难忘。

感谢师兄刘永峰、程国青、顾新征对研究工作的指导。

感谢多年来给予我鼓励和帮助的河海大学网络中心的各位老师，以及计算机信息工程学院的各位领导和老师。

感谢一起工作和生活的同学：麻春艳、史宇清、高军、潘洁、吉建峰、朱辉、温泉等，文中的许多思想来自与他们长期以来相关讨论时所受的启发，在此谨对他们表示诚挚的谢意。

感谢同宿舍的李静燕、吕小燕等同学，是你们和我一起度过与分享了美好而难忘的研究生时光。

感谢钱永强对我的关心和支持。

最后感谢我父母、弟弟以及亲友多年来对我学习和生活等各方面无微不至的关心和支持，我成长的每一步都凝聚了他们的心血。