

文章编号:1001-9081(2006)04-0840-04

基于 Spring MVC 与 iBATIS 的轻量级 Web 应用研究

刘 军,戴金山

(武汉理工大学 计算机科学与技术学院,湖北 武汉 430070)

(daijinshan@mail.whut.edu.cn)

摘 要:针对基于 Java 的 Web 应用系统设计与开发,根据中小型企业实际业务环境,探讨了 Web 应用系统的技术框架,对 Spring 与 iBATIS 开源框架及相关核心技术反向控制机制(IoC)进行了研究。结合 JSP 与 JSTL 技术,实现了基于 Spring MVC 与 iBATIS 框架的轻量级 Web 应用系统的设计方案。在实际试运行期间,经不同终端的测试,服务器始终保持稳定,显示该系统框架在安全性、稳定性和健壮性上都有良好的改进。

关键词:Spring; iBATIS; MVC; 反向控制; JSP 标准标签库

中图分类号: TP311.5 **文献标识码:** A

Research of lightweight Web application based on Spring MVC and iBATIS frameworks

LIU Jun, DAI Jin-shan

(School of Computer Science and Technology, Wuhan University of Technology, Wuhan Hubei 430070, China)

Abstract: To describe the design and development of Java-based Web application systems under the actual business environment of large or small scale, the frameworks of Web application systems, the open frameworks Spring and iBATIS and the related IoC mechanism was discussed. Then a solution for light weight Web application systems based on Spring MVC and iBATIS frameworks were proposed. The JSP and JSTL technologies were also introduced into this solution. The good performance of application systems in the test and running phases manifests a remarkable improvement of the proposed system framework in security, stability and robustness issues.

Key words: Spring; iBATIS; MVC; Inversion of Control(IoC); Java Server Pages Standard Tag Library(JSTL)

0 引言

在基于 Web 的 Java 应用开发中,服务器端的容器组件是关键的一环。复杂的业务应用系统需要重量级的容器,如集成众多企业级服务的 EJB 容器。而在实现全面而完整的服务策略的同时,重量级容器也带来了许多的负面效果,例如部署复杂,运行缓慢,内在服务多,启动慢,规则特多,空间很小,难以测试或者调试等。这些问题在中小企业的开发中应该避免。针对以上问题提出的轻量级 Java 开发主要指两个方面:简化的编程模型和更具响应能力的轻量级容器。它旨在消除与传统 J2EE API 有关的不必要的复杂性和限制,也将缩短应用程序的部署时间。这对于支持开发最佳实践(比如频繁单元测试)非常重要。轻量级 Java 开发的另一个关键特征是,它不会强迫业务对象遵循平台特定接口。这允许开发人员在简单 Java 对象(Plain Old Java Object, POJO)中实现业务逻辑,从而提高生产率。所以在中小型且周期较短的项目中更需要“可选择型”的服务容器。轻量级容器通过反向控制(Inversion of Control, IoC)让容器具有主动权,去管理所有符合标准的组件。

基于 Web 的 MVC framework 在 J2EE 应用开发中已经得到广泛的应用。MVC 模式实现了业务逻辑和数据显示的分离,使得应用系统结构清晰,同时维护性、可扩展性及可移植

性得到了很大提高^[1]。而这种组件化的开发更易于实现对大规模系统的开发控制和管理。但是开发 MVC 系统比简单的 JSP 开发要复杂许多,所以在采用 MVC 实现 Web 应用时,最好选用一个现成的 MVC 框架,并在其基础上进行开发。它能够通过内部定制而具备良好的扩展性,并且有强大的用户组支持。目前较为稳定的 MVC 框架有 Struts、Webwork 等。这些框架都提供了较好的分层能力,但在扩展性、设计的精密性及可测试性等技术特性及开发效率上,这些框架已经落后于新兴的 MVC 框架(如 Spring MVC、Tapestry、JSF 等)。在项目开发中选择一个好的框架,对于项目的开发效率和可重用是至关重要的。

本文以某信息网和基于 Web 的某 CRM (Customer Relationship Management) 系统等项目的实际开发为背景,针对中小型企业信息网现状和问题,结合企业的实际业务流程,探讨如何利用现有开源技术框架实现 Web 应用松耦合的目的,探讨轻量级 Web 应用的开发过程及在 Spring 框架中如何实现面向接口编程。在阐述系统实现的框架技术的基础上,进行了基于 Spring MVC 与 iBATIS 框架的轻量级 Web 应用分析与设计。系统框架基于 Spring Web MVC 框架的 MVC 实现机制,表现层采用 JSP 与 JSTL 模板技术的组合;业务逻辑层使用 Spring 框架的 IoC 机制进行整合;持久化层使用 iBATIS 框架(由 Spring ORM 模块提供)实现映射。当前,本文所涉及

收稿日期:2005-10-28;修订日期:2006-01-13

作者简介:刘军(1966-),女,湖北武汉人,副教授,硕士,主要研究方向:软件工程、网络数据库;戴金山(1978-),男,福建莆田人,硕士研究生,主要研究方向:网络数据库、网络安全、工作流管理。

的企业信息网及 CRM 系统已开发完毕并试运行。该技术框架组合在可扩展性、安全性、稳定性、移植性及代码清晰度等多方面已初步取得良好的效果。

1 核心技术及框架介绍

1.1 Spring 框架制与 IoC 反向控制机

Spring 框架是一个开源且基于 POJO 的轻量级 J2EE 应用框架,它是为了解决企业应用程序开发复杂性而创建的。通过使用基本的 JavaBeans 来完成以前只可能由 EJB 才可能完成的事情。从应用开发的实际角度看, Spring 是一个从实际项目开发经验中抽取的,可调试重用的应用框架^[2]。

Spring 框架各模块构建在核心容器(Spring Core Bean Container)之上。核心容器定义了创建、配置和管理 bean 的方式,提供了 Spring 框架的基本功能。其主要组件是 BeanFactory,它实现了工厂模式,是个轻量级组件。Spring 框架通过该组件加载所有类。Spring 框架内含了许多 ORM 框架,从而提供了 ORM 映射工具,其中包括 JDO、Hibernate 和 iBATIS SQL Maps。Spring Web MVC 框架是一个全功能的构建 Web 应用的 MVC 实现。通过接口, MVC 框架变成了高度可配置的 MVC,容纳了大量视图技术,其中包括 JSP、Velocity、Tiles 等。

框架的主要优势之一就是其分层架构,这使组件式开发成为可能。作为一个分层架构, Spring 涵盖了应用系统开发所涉及的大多数技术范畴,包括 MVC、ORM 和 Remote Interface 等,这些技术贯穿了大多数应用系统的开发过程。Spring 从开发者的角度对这些技术内容进行了进一步的封闭和抽象,使得应用开发更为简便。在实际开发工作中,借助于 Spring 提供的丰富类库,相对于传统开发模式,可节省大量编码。

Spring 框架中目前最引人注目的是 IoC 反向控制或者依赖注入(Dependence Injection, DI)的设计思想。

通常情况下,要实现对其他类的引用,应用代码需要告知容器或框架,让它们找到自身所需要的类,然后再由应用代码创建待使用的对象实例。而在 IoC 模式中,创建对象实例的任务将由 IoC 容器或框架实现(实现了 IoC 设计模式的框架也被称为 IoC 容器),即在运行期由容器将依赖关系注入到组件之中,这使得应用代码只需要直接使用实例。Spring IoC 容器实现了 IoC 设计模式。Spring 根据配置文件,将其他对象的引用通过组件提供的 setter 方法进行设定^[3]。图 1 显示了两者的差异:在图 1(a)所示情况下,对象 A 直接创建所需对象 B 和 C,这将导致代码的紧耦合,对象 B 和 C 的任何改动都将导致对象 A 的重新编译;在图 1(b)所示情况下,使用 IoC 机制,对象在更高层创建并传递到需要使用它们的其他对象中。

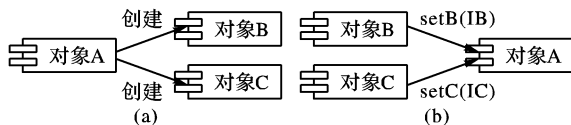


图 1 不使用 IoC 机制与使用 IoC 机制比较

Spring 通过依赖注入机制,可以在运行期为组件配置所需要资源,而无需在编写组件代码时就加以指定,从而在相当程度上降低了组件之间的耦合,实现了组件真正意义上的即插即用。而组件间依赖关系的减少也将极大改善代码的可重用性。这也是 Spring 最具价值的特性之一。此外,应用的组

件并不需要实现框架指定的接口,因此可以轻松地将组件从 Spring 中脱离,甚至无需任何修改,这在基于 EJB 框架实现的应用中是难以想象的^[2]。

1.2 iBATIS 及 iBATIS SQLMaps 框架

iBATIS 框架主要包含两类组件,即 SQL Maps 组件和 DAO 组件。SQL Maps 组件允许开发人员在不使用 JDBC API 以及不耦合 Java 代码与 SQL 语句的情况下实现 Java 对象和关系型 DBMS 系统的读写操作^[4]。SQL Maps 组件有以下特点:1)该组件基于 XML 配置文件,使用简单 XML 配置文件将 Java Bean 映射成 SQL 语句。它不同于其他如 hibernate 等 ORM 工具,后者将 SQL 语句映射成 Java 对象^[5]。iBATIS 以 SQL 开发的工作量和数据库移植性上的让步,为系统设计提供了更大的自由空间;2)SQL 语句的输入和输出参数可以是基本类型的包装类,或者简单类(如 Integer、String 等),或者 HashMap,也可以直接使用应用中更为复杂的类(例如值对象 VO、数据传输对象 DTO 等);3)可针对特定的数据库操作设置特定的 Java 属性/SQL 字段列值映射;4)相比于其他 ORM 框架,SQL Map 框架简单易学,且编程代码简练代码。图 2 显示了 JDBC 编程与 iBATIS 框架的流程比较。从中可看出,使用 SQL Map 框架能够大大减少访问关系数据库的代码。

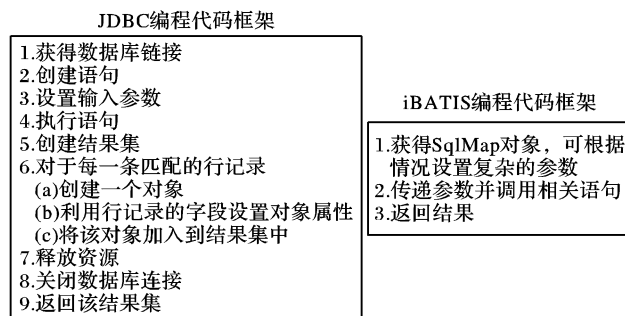


图 2 JDBC 与 iBATIS 程序框架的比较

使用 iBATIS SqlMaps 涉及创建包含语句和结果对应关系的配置文件。Spring 将通过 SqlMapClientFactoryBean 读取配置文件。

2 Web 应用项目具体实现方案

本 Web 应用系统利用 Spring MVC 作为 MVC 开发模式,利用 iBATIS 作为 ORM 数据持久层框架,结合 JSP 与 JSTL1.1 作为表现层实现技术。MVC 框架的控制器将以 AbstractController 抽象控制器(org. springframework. web. servlet. mvc. AbstractController)为主;数据封装以 Map、HashMap、List 和 ArrayList 等类为主;数据访问使用 iBATIS 中的 SqlMapClient 接口;SQL 语句利用映射文件加载;日志产生是基于 Apache 的 Log 日志对象技术;JSP 页面使用 JSTL 模板技术;页面样式使用 CSS(级联样式表)与 HTML 组件技术(即 htc, html component)。

2.1 权限与逻辑控制方案

在该项目中,权限与逻辑控制主要通过以下三类控制器实现:逻辑控制器、权限控制器与业务控制器。三类控制器的继承关系如图 3 所示。

(1) 抽象权限控制器。该类为 com. framework. purview. PurviewController。在该控制器中实现了 AbstractController 控制器的 handleRequestInternal 方法,该方法对权限进行了初步判断(包括用户是否登录,是否为管理员,及请求动作等详细

权限判断)。登录成功后执行该控制器中的请求抽象方法 `handleRequestLogic`, 具体的实现由继承该控制器的子类即各业务控制器类实现。在抽象权限控制器中还设置了一些常用的属性对象, 如访问用户的权限路径、用户的值对象 (VO)、权限判断标志变量等, 在执行 `handleRequestInternal` 方法时将获得这些对象的实例, 并可被所有继承该控制器的子类直接使用。该实现方案可精简业务代码, 同时, 整个系统框架都是在 `AbstractController` 控制器基础上实现。

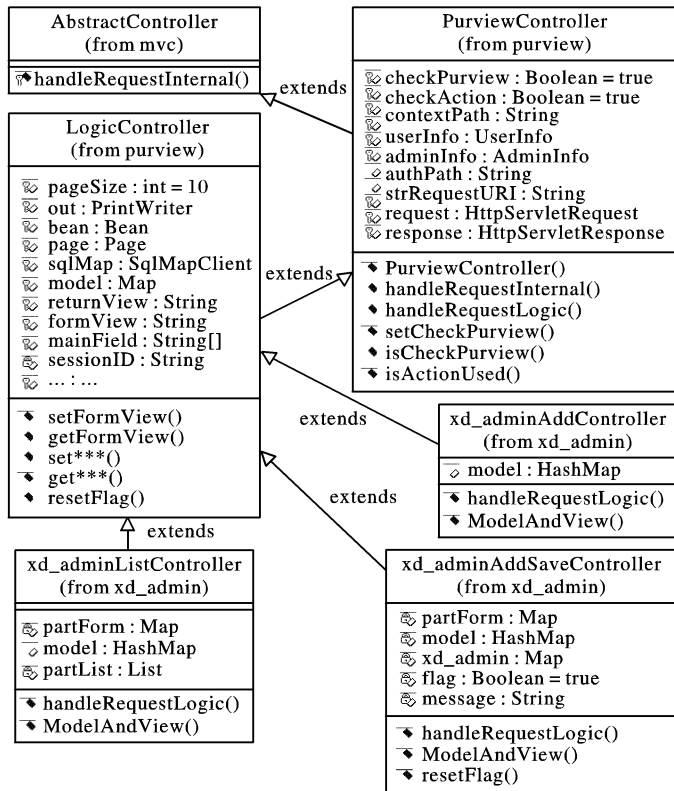


图3 系统控制器继承关系图

(2) 抽象逻辑控制器。该类为 `com.framework.purview.LogicController`。对于 Spring 的控制器, 大部分业务逻辑控制都要用到一些公用的属性、方法和对象。如果每个业务控制器都定义这些属性及其相应的 `set/get` 方法对, 即便通过复制和粘贴脚本完成, 仍存在重复代码过多, 维护代价高, 效率较低。基于上面的问题, 提出了抽象逻辑控制器, 它继承自权限控制器 `PurviewController`, 在该控制器对各模块中对公用属性和方法统一进行定义, 这样既验证了权限, 又精简了业务代码, 从而提高代码编写的效率。该控制器定义的公用属性和方法包括: 控制器返回视图 `Model`, 封装表单字段, 公用数据封装接口 `bean`, `iBATIS` 核心对象接口和公用的方法实现等。该控制器没有实现权限控制器中的 `handleRequestLogic` 方法。作为抽象的方法, 它由继承自该控制器的各业务控制器实现。

(3) 业务控制器。这些业务逻辑模块实现系统各模块中的请求动作。业务控制器统一继承自 `LogicController`, 实现其中的 `handleRequestLogic` 方法。它们可以直接调用 `PurviewController` 和 `LogicController` 中的各种属性和方法, 不必重复定义。对于所有的业务请求, 业务控制器都会先执行父类中的权限控制, 然后再执行各自业务请求, 这样对所有请求动作进行统一的控制, 提高代码重用率, 同时也实现了权限管理模块与业务模块分离, 实现公共的权限管理框架, 达到组

件重用目的。

(4) 系统权限控制方案。文中 Web 应用系统的权限管理基于用户—角色—功能点的方式实现。一个管理员可对应于多个角色, 一个角色对应于与系统特定模块相关的多个功能点。用户或管理员在登录时系统将其权限信息保存在其相应的会话中, 以达到权限的初步控制。同时, 为了防止用户访问其无权使用的模块及功能点, 系统将对所有请求动作进行权限控制。如前所述, 抽象权限控制器 `PurviewController` 继承自 `AbstractController` 控制器, 在该类中统一对权限进行初步控制, 同时提供处理请求的抽象方法。抽象逻辑控制器 `LogicController` 抽象出业务控制器的公共属性与方法, 统一定义公用接口与方法的实现。然后所有业务模块控制器都继承自该抽象逻辑控制器, 实现权限控制提供的抽象方法。

2.2 系统的配置文件及 IoC 依赖注入的实现

(1) 系统的配置文件

业务控制器按模块划分, 针对各模块生成相应的 Spring 配置文件, 并以模块名作为前缀, 统一置于 `\web\WEB-INF\config` 下。各模块的配置文件都包含了对 `applicationContext.xml` 配置文件中定义的 `bean` 和 `sqlMap` 对象的引用。

1) `applicationContext.xml` 文件对系统框架中使用的公共 `Bean` 进行了设置, 定义表现层资源的前缀和后缀, 加载 `properties` 资源文件、异常监听组件、`iBATIS` 核心对象、数据封装对象和分页生成类等。该文件在站点启动时通过 `web.xml` 文件加载, 生成在该其中定义的各种 `bean` 的实例并将它们放置在 `bean` 工厂中, 各业务模块通过简单的设置就可以获得实例;

2) `log4j.properties` 是系统运行日志的配置文件;

3) `purview-config.xml` 是权限控制和系统管理模块配置文件, 对应的请求路径是“/* .do”, 即根目录下的所有操作。其他模块则以 `/model/* .do` 作为请求路径前缀, 以针对各业务模块按不同的配置文件进行配置;

`xd_admin-config.xml` 及其他业务模块配置文件为各具体业务模块的配置文件, 在其中实现请求映射及相应控制类的设置;

4) `sqlMapConfig.xml` 是设置 `iBATIS` 框架的配置文件, 由 `applicationContext.xml` 配置文件加载, 加载时生成 `sqlMap` 实例以供各业务模块调用。

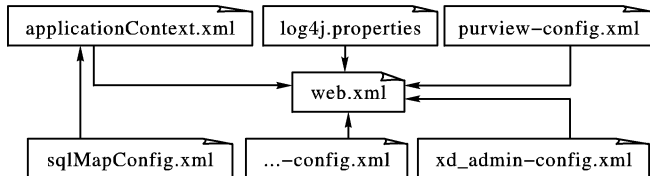


图4 Web系统及各模块的配置文件

(2) IoC 依赖注入的实现

以系统管理员管理模块为例, 在配置文件 `xd_admin-config.xml` 中, 需要利用 Spring 框架的 IoC 机制实现对数据封装接口 `bean`、分页查询接口 `page` 及 `iBATIS` 核心对象接口等对象的引用, 实现方法如下面程序段中的第二至第四行代码所示。这样即可在运行期动态注入对象间的依赖性, 在对象代码发生变化的情况下无需重新编译便可保证系统的运行。业务控制器配置文件中对 `bean` 等接口的引用代码为:

```

<!-- 设置系统管理员列表逻辑的控制器 -->
<bean id = "xd_adminListController" class = "xinde.xd_admin.xd_
adminListController" >
    <property name = "bean" > <ref bean = "bean"/ > </property>
        <!-- 引用数据封装接口 bean -->
    <property name = "page" > <ref bean = "page"/ > </property>
        <!-- 引用分页查询接口 page -->
    <!-- 引用 iBATIS 核心对象接口 sqlMap -->
    <property name = "sqlMap" > <ref bean = "sqlMap"/ >
        </property>
    <!-- 定义数据表单视图数据页面 -->
    <property name = "formView" > <value>/WEB-INF/jsp/xd_
admin/xd_adminList.jsp</value> </property>
    <!-- 定义待封装表单字段 -->
    <property name = "mainField" >
        <list>
            <value>ADMI_NAME</value>
            <value>ADMI_PWD</value>
            <value>ADMI_REALNAME</value>
            <value>ADMI_ROLEID</value>
            <value>pageNumber</value>
        </list>
    </property>
</bean>

```

2.3 数据的封装

数据在各层之间是通过容器封装并实现传递的。考虑到数值对象的编写和维护比较费时,因此在本系统中大量使用 Map、HashMap、List 等对象作为数据封装的容器,而对于少数公用 bean,如用户和管理员登录的信息对象等,则使用具体值对象。

在本项目技术框架中已经实现了数据封装 bean 的接口和实现,分别为 com.framework.beanutils.Page 类与 com.framework.beanutils.PageImpl 类。在抽象逻辑控制器 LogicController 中对该接口进行定义,因此在各业务控制器配置中只需加入以下 Bean 引用代码及请求封装的字段即可直接调用了。

```
<property name = "bean" > <ref bean = "bean"/ > </property>
```

这也体现了 Spring 的 IoC 依赖注入机制以及 Spring 框架所倡导的接口与实现分离、面向接口而非实现进行编程的思想。Spring 对于面向接口设计的意义,在于它为面向接口编程提供了一个更为自然的平台^[2],使得接口的定义和使用不再像传统编码过程中那么繁琐,而传统编码过程中,引入一个接口往往也意味着要引入一个 Factory 类,甚至还必须额外的配置文件及其读写代码。

2.4 数据的访问

数据访问统一调用 iBATIS 核心对象 SqlMapClient 接口。在抽象逻辑控制器 LogicController 中已经对该接口的调用进行了定义,在继承 LogicController 控制器的各业务控制器中通过配置即可注入 sqlMap 对象。

```
<property name = "sqlMap" > <ref bean = "sqlMap"/ > </property>
```

如果要进行多次数据库操作,则应该通过事务来控制,这样可保证数据库访问在一次连接内完成。

2.5 分页查询的实现

分页查询在 Web 应用中是最常用,也是比较复杂的技术。利用 Spring 框架和 iBATIS 框架的现有功能,本系统框架中已经实现了公用分页控制器。该控制器结合了 Spring 中 Bean 的重用机制及 iBATIS 中对动态查询条件的配置。

在 Spring 中一个 Bean 对象是可以通过不同的 Bean ID 来配置不同的实例,这样就可以为公共分页类配置不同 ID 和属性,来执行不同的分页查询。此外,在 iBATIS 中进行查询时可以将查询条件的封装交由 sqlMap 对象处理,iBATIS 可根据 SQL 映射文件中的 SQL 语句配置及传入的查询条件对象中的字段动态拼装 SQL 语句并执行。

2.6 视图表现层的实现

在本技术框架中 JSP 技术只用作表现层,因此,在 JSP 页面上避免了使用类似于 <% = request... % > 的语法,而统一使用 JSTL1.1 标签来输出。JSP 页面通过 <% @ include file = "/WEB-INF/jsp/inc/tld.jsp" % > 来载入 JSTL1.1 标签^[6]。此外,基于安全性考虑,JSP 文件统一置于目录/WEB-INF/jsp 下,防止用户通过请求直接访问 JSP 页面,强制用户通过系统的权限控制方案来访问访问信息。JSP 页面中 JavaScript 脚本以 <script src = "/js/sell/ *.js" > </script> 载入,这样避免了在 JSP 页面上直接编写,保持页面的简洁,提高脚本的重用性,有利于脚本的调试。JavaScript 脚本按不同的模块划分,存放在/js/下的不同目录。

3 结语

本文介绍了轻量级 Web 应用开发中的相关技术和架构知识,主要思想是通过合理的应用程序分层,采用基于开源框架的 MVC 架构,利用 IoC 依赖注入机制,实现面向接口编程,以达到降低耦合度,减少代码量的目的。通过 Spring 框架提供的依赖注入机制,在运行期间为组件配置所需的资源,降低组件间的耦合度;通过 iBATIS SqlMap 框架使得在不借助于 JDBC API 及不耦合 Java 代码与 SQL 语句的情况下 Java 对象和关系型 DBMS 系统的读写操作,极大减少访问关系数据库的代码量;通过采用 JSP2.0 和 JSTL1.1 标签库,可使 JSP 页面基本上避免了脚本元素,页面简洁且可读性强。在实际试运行期间,经不同终端的测试,服务器始终保持稳定,显示该系统框架在安全性、稳定性和健壮性上都有良好的改进。上述特性和结论表明,本文所采用的核心技术及系统框架,在轻量级 Web 信息系统的设计与开发中有一定的优势与广泛地应用前景。

参考文献:

- [1] 徐长盛,戴超.一种快速开发 Web 应用程序方法的研究[J].计算机工程与设计,2004,25(12):2237-2239.
- [2] 夏昕. Spring 开发指南[EB/OL]. <http://www.matrix.org.cn/resource/down/579.html>, 2004.
- [3] 罗时飞.精通 Spring[M].北京:电子工业出版社,2005.29-47.
- [4] 夏昕. iBATIS 2.0 开发指南[EB/OL]. <http://www.matrix.org.cn/resource/down/581.html>, 2004.
- [5] MUKHAR K, LAUINGER T. Java 数据库应用程序编程指南[M].北京:电子工业出版社,2002.448-520.
- [6] COOK T. JSP 从入门到精通[M].北京:电子工业出版社,2003.437-469.
- [7] 吴高峰.基于 Struts + Spring + iBATIS 的轻量级 J2EE 开发[EB/OL]. <http://www-128.ibm.com/developerworks/cn/java/j-s-s-i/>, 2005.
- [8] Apache Software Foundation. iBATIS SqlMaps Developer Guide[EB/OL]. <http://prdownloads.sourceforge.net/ibatisdb/>, 2005.
- [9] 夏昕.深入浅出 Hibernate[M].北京:电子工业出版社,2005.467-512.