文章编号:1671-4598(2018)10-0122-06 DOI:10.16526/j.cnki.11—4762/tp.2018.10.027 中图分类号:TP311.52 文献标识码:A

SSH 框架在 Web 项目开发中的设计与实现

卢肖霞
(广州医科大学 信息技术教研室,广州 511436)

摘要:在 Web 项目开发中,一个良好的框架可以加快开发速度,降低成本,减少工作量,同时可以使 Web 项目具有良好的扩展性和移植性;基于 Spring MVC+Spring+Hibernate (SSH) 的项目以其快速的开发效率和良好的扩展性,成为中小企业应用主流的框架组合;Spring MVC 是一个基于 MVC 设计模式的框架,是 Spring 的子项目,主要负责表现层和控制层的功能,例如响应请求等;Spring 是一个实现 IoC 和 AOP 的容器,降低组件间的耦合度,可以整合和支持 Spring MVC 和 Hibernate 等主流框架,使业务逻辑更加清晰;Hibernate 负责管理数据持久化,实现与数据库相关的 CRUD 操作;以购书网站的后台管理系统为例,重点说明 SSH 框架在 Web 应用系统的后台开发中的应用。

关键词:Spring MVC;Spring;Hibernate;SSH;Web 项目开发

Design and Implementation of SSH Framework in Web Project Development

Lu Xiaoxia
(Information Technology Teaching and Research Office, Guangzhou Medical University, Guangzhou 511436, China)

Abstract: In the development of Web projects, a good framework can speed up development, reduces costs and reduces workload, and enables Web projects to have good scalability and portability. The project based on Spring MVC+Spring+Hibernate (referred to as SSH) has become a mainstream portfolio framework for small and medium-sized enterprises because of its rapid development efficiency and good scalability. Spring MVC is a framework based on MVC design patterns, as a sub project of Spring, which is responsible for the functions of the presentation layer and the control layer, such as response requests. Spring is a container to realize IoC and AOP, reduces coupling between components, integrates and supports main frameworks such as Spring MVC and Hibernate, and makes business logic clearer. Hibernate is responsible for managing data persistence and implementing CRUD operations related to the database. Taking the background management system of the book purchase website as an example, the application of the SSH framework in the background development of the Web application system is emphasized.

Keywords: Spring MVC; Spring; Hibernate; SSH; web project development

0 引言

随着 Web 技术在互联网上的发展,为解决 web 大型应用开发的复杂性和提高效率等,许多轻量级的 web 框架技术应运而生,著名的有 Struts2,Spring、Hibernate、MyBatis 等[1]。在这些框架中,许多框架专注于为某一层提供解决方案,例如表示层或持久层。Spring 框架致力于简化 Java EE 应用程序的开发[2],将很多高质量的开源框架整合在一起,为企业应用开发提供一个全面的解决方案,降低企业开发的难度,提高开发效率。以购书网站后台管理系统的用户管理为例,阐述基于 Spring 整合 Spring MVC 和 Hibernate 的 SSH 框架在 Java Web 项目开发中的应用。

收稿日期:2018-02-28; **修回日期**:2018-04-10。
基金项目:国家自然科学基金项目(61603106);国家自然科学基金项目(61603106)。
作者简介:卢肖霞(1977-),女,广西陆川人,硕士研究生,讲师,主要从事软件工程、Web 开发技术方向的研究。

1 MVC 设计模式

MVC (Model-View-Controller) 是软件工程中的一种软件设计模式,把软件系统分为模型 (Model)、视图 (View) 和控制器 (Controller) 3 个部分[3]。图 1 显示 MVC 设计模式的关系模型。Model 模型用于封装与业务逻辑相关的数据和处理方法;视图 View 是数据的呈现,与用户交互的界面,其组件一般是 JSP 或 HTML;控制器 Controller 是接收请求,调用 Model 实现业务,调用 View 显示数据,最终完成用户请求。MVC 设计模式将包含业务数据的模块与显示模块的视图解耦,不仅使得代码复用性和组织性更好,而且使 Web 应用的配置性和灵活性更好。

2 Spring MVC

Spring MVC 是一个实现了 MVC 设计模式的 Web 表示层框架,即用 MVC 设计模式的思想,将 web 层进行职责解耦,目的就是简化 web 开发。Spring MVC 提供了构建 Web 应用程序的全功能 MVC 模,有清晰的角色划分,包括前端控制器 (DispatcherServlet)、映射处理器 (HandlerMap-

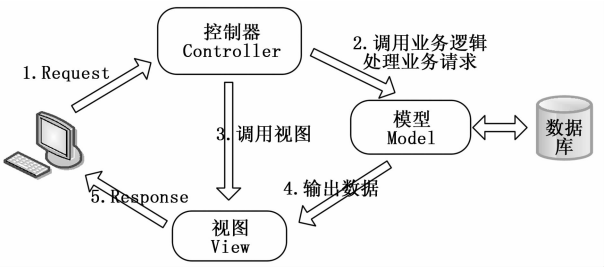

图 1　MVC 关系模型

ping)、映射适配器（HandlerAdapter)、视图解析器（ViewResolver)、控制器（Controller)、验证器（Validator)、命令对象（Command Object)、表单对象（Form Object)等，其中 DispatchServlet 是 SpringMVC 的核心组件。SpringMVC 的工作流程如图 2 所示，Dispatcher Servlet 接收到请求后，将请求委派为映射处理器，映射处理器和映射适配器根据请求所带的 URL 信息进行匹配选择控制器，然后请求发送给选中的控制器。控制器在完成逻辑处理后，通常产生一些需要返回给浏览器显示的信息，这些信息称为模型（Model)，并且需要用界面友好的视图（通常是 JSP)来显示[4]。控制器把模型数据和用于渲染输出的逻辑视图名，连同请求一起发送回 DispatcherServlet。DispatcherServlet 使用视图解析器将逻辑视图名匹配为一个特定的物理视图，然后将模型数据交付给物理视图来渲染输出。

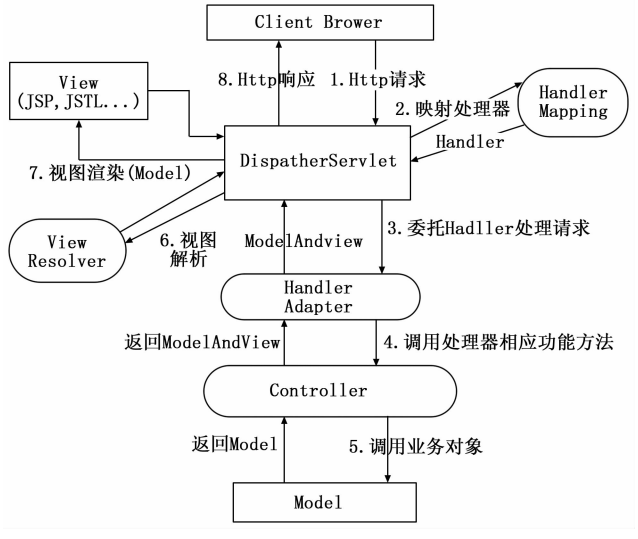

图 2　Spring MVC 工作流程

Spring MVC 虽然和 Struts2 功能相似，但 Struts2 是类级别的控制拦截，一个 Action 类对应一个请求 URL 上下文，而 Spring MVC 是方法级别的拦截，一个方法对应一个请求 URL 的上下文。

3　Spring

Spring 是一个轻量级的企业级开源框架，是为了解决企业级应用程序开发的复杂性而创建[5]。Spring 框架的核心是一个 IoC 容器，在这基础上提供了大量实用的服务，将很多高质量的开源项目集成起来。Spring 框架大约由 20 个功能模块组成，这些模块分为 7 个部分：分别是 CoreContainer、DataAccess/Integration、Web、AOP（Aspect Oriented Programming)、Instrumentation 及 Test，如图 3 所示。

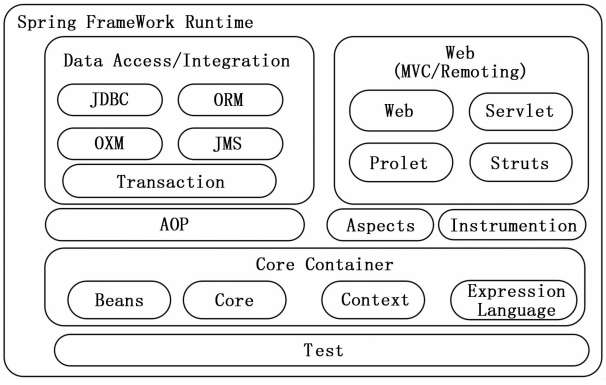

图 3　Spring 框架结构

Spring 的核心是控制反转 IoC 和 AOP 面向切面编程技术[6]，Spring 核心容器定义了创建、配置和管理 Bean 的方式，管理各个 Bean 对象之间的依赖关系，能够有效避免由于硬性编码所造成的耦合度过于紧密的状况；AOP 功能可以轻松实现业务逻辑与系统服务（例如日志、事务等）的分离，使开发人员更加专注于业务逻辑实现。Spring 是一个全面的解决方案，实现了表现层、业务层和持久层的整合，其主要目标是让现有的技术更加易用，绝不做重复实现，只是对现有方案提供支持，使之更易用。

4　Hibernate

Hibernate 是一个优秀的 Java 持久层解决方案，是当今主流的对象—关系映射（ORM，Object Relational Mapping）工具[7]。Hibernate 的工作流程如图 4 所示。

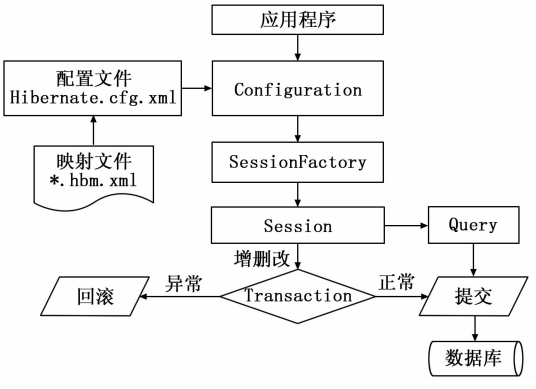

图 4　Hibernate 工作流程

Configuration 首先读取并解析配置文件 hibernate. cfg. xml，创建 SessionFactory，SessionFactory 是单例模式，只创建一次；接着 SessionFactory 创建 Session 接口，Session

接口提供了操作数据库的各种方法：save（）、delete（）、load（）等，Session 方法是非线程安全，每次执行一次数据库事务，都需要创建 Session 对象；如果是增删改操作，则由 Session 对象开启一个事务对象 Transaction，保证数据库操作的完整性如果数据库操作正常则提交事务，否则回滚；如果是查询操作，则由 Session 创建一个 Query 接口对象来执行查询，查询不需要开启事务。数据库访问结束，需要关闭 Session 对象，如果开启 Transaction 对象，也要关闭。在查询中，Query 接口用于 HQL 查询，HQL（Hibernate Query Language）是 Hibernate 中最常用的一种面向对象的查询语言，可以理解继承、多态和关联之类的概念[8]。

Hibernate 的优势在于数据库操作对象化，使用数据库方言机制和 HQL 查询语言，无须关心数据库类型，便于移植[9]；Hibernate 封装了 JDBC，还封装了底层数据库事务处理，极大提高了开发效率；为了减少访问物理数据库的次数，还使用了缓存机制，从而提高了程序的运行性能。

5 SSH 的系统结构

这里的 SSH 是 Spring MVC、Spring 和 Hibernate 的简称，这是当前主流的 Web 应用项目架构。Java EE 经过多年的发展，已经形成一套成熟的系统结构，系统一般分为四层：表示层、控制层、业务层和数据持久层。在 SSH 框架下，其系统的程序结构如图 5 所示。

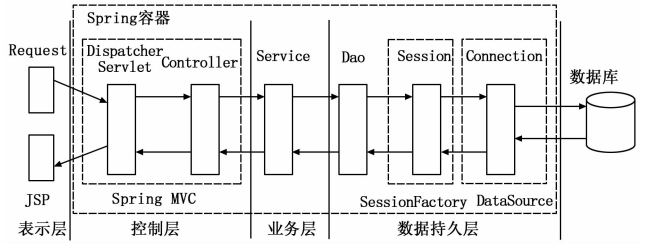

图 5 SSH 系统的程序结构

表示层一般为 JSP 页面，返回用户请求响应；控制层由 Spring MVC 框架负责，转发请求并调用业务层方法处理请求；业务层即 Service 层，是系统架构中体现核心价值的部分，它主要集中在业务规则的制定、业务流程的实现等与业务需求有关的系统设计上[10]。业务层处在控制层和持久层中间，对持久层而言，它是调用者，对控制层而言，它是被调用者。数据持久层由 Hibernate 框架实现。

在图 5 中，来自客户端的 Request 请求被由 DispatcherServlet 前端控制器接收，DispatcherServlet 把请求派发给 Controller 控制器，Controller 调用业务方法 Service 来实现业务，如需要访问数据库，业务类就调用 Dao（Data Access Object，数据访问对象）数据操作类访问数据库。而 Dao 类则需要使用 Hibernate 的 SessionFactory（会话工厂）提供的 Session 会话来实现具体操作，Session 通过 Connection 来实现增删改擦操作，而 Connection 由数据源（DataSource）来提供。

Controller、Service、Dao 都纳入 Spring 容器管理，Session 由 SessionFactory 创建并管理，Connection 由 DataSource 创建并管理，所以也需要把 SessionFactory 和 DataSource 加入 Spring 容器管理。

6 基于 SSH 框架的购书网站后台管理系统的设计与实现

6.1 系统功能模块设计

购书网站主要包括两个部分：前台部分和后台管理部分。前台部分主要服务于普通用户，用户可以查看图书信息，登录，注册，购买和进行个人信息管理、购物车管理、下单等操作；后台管理模块则是管理员对书店进行业务管理，其后台系统的基本功能模块设计如图 6 所示。

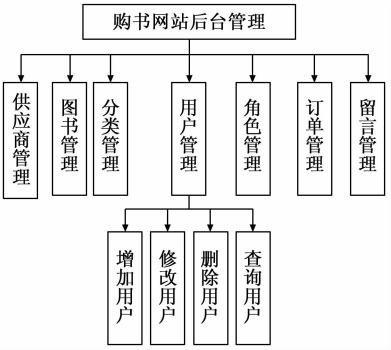

图 6 购书网站后台管理的基本功能结构

6.2 SSH 框架整合

首先进行系统的 SSH 框架整合，整合工作主要是 Spring 与 Hibernate 整合和 Spring 与 SpringMVC 的整合，整合以后，所有实例对象纳入 Spring 容器管理。整合分三步：

第一步，整合 Spring 和 Hibernate。在 Spring 配置文件 applicationContext. xml 中，把外部导入的 Properties 属性文件、Hibernate 的数据源对象、sessionFactory 对象、事务管理等配置进来，Hibernate. cfg. xml 就不再需要。接下来配置持久类映射文件的路径和 Dao 以及 Service。为了简化配置工作，Dao 和 Service 可以采用注解方式注入 Spring 容器。为了使用注解，需要在配置文件中加入两个配置，见下列代码。

```
<! —启用注解和自动扫描带注解的基础包,但不包括使用了@Controller 注解的类所在的包——>
<context: annotation—config />
<context: component—scan base—package="com. * "/>
```

上述配置的代码，可以自动扫描 com. * 包下的所有注解类。

第二步，Spring 整合 Spring MVC。需要单独使用一个配置文件 springMVC. xml，该文件放在 src 下或 WEB—

INF 下均可。Spring MVC 的配置文件通常有三个常用的配置，分别是：

1）＜context：annotation－config /＞，表示向 Spring 容器注册使用了@controller 注解的类，使 SpringMVC 认为使用@controller 注解的类是控制器。

2）＜context：component－scan base－package＝"com. ＊"＞表示自动扫描所有 com 包下的带@Controller 注解的类。

3）＜mvc：annotation－driven/＞表示注册映射处理器（HandlerMapping）和映射适配器（HandlerAdapter），并且只有配置了＜mvc：annotation－driven/＞，才能使用@Controller 注解，否则 DispatcherServlet 无法找到控制器 Controller 并把请求派发到控制器上。

此外，spring MVC 的配置文件还需要配置视图解析器 ViewResolover，表示把用户响应指定给某个表现层技术，例如 JSP、JSTL，XLST 和 velocity 等，通常设定为 JSP。如果使用 JSP，则配置的视图解析器是 InternalResourceViewResolver；如果 JSP 页面需要使用 JSTL 标签，则增加＜property＞节点来配置 JstlView 类。如果需要使用文件上传下载功能，需要配置 multipartResolver，其视图解析器是 CommonsMultipartResolver。

第三步，在 Web. xml 中配置 Spring 监听器 ContextLoaderListener 和 SpringMVC 的核心 DispatcherServlet，以及其他一些参数，例如过滤器 filter 和上下文参数 context－param。这样 Spring MVC 就可以监听来自客户端的请求，并把请求转发给 DispatcherServlet 处理。

6.3　系统实现

后台管理系统的功能主要是与后台数据库交互进行管理业务数据，与后台数据库的交互主要是数据库增加、修改、删除和查询等操作。所以这里以用户管理为例，主要讲述在 SSH 框架下用户管理功能的实现。

6.3.1　持久化类实体层

使用 Hibernate 来实现数据访问持久化，首先要建立持久化类以及配置持久化类和数据库表的一一映射关系。本系统的用户管理涉及两个数据表：User 用户表和 Role 角色表，因此先创建与数据库表字段一一对应的持久化类 User 和 Role，然后配置持久化类与数据库表之间的映射关系。在 Hibernate 中，有两种方式来实现该映射关系，一种是使用映射配置文件，即建立 User. hbm. xml，还在改文件中表与表之间的关联关系；另一种是使用注解方式。本文采用注解来实现数据库表与持久类的映射关系，以及 User 持久化类和 Role 持久化类之间多对一关联关系。具体见如下代码：

```
@Entity //声明 User 类为 POJO 类
@Table(name="user") //和 User 数据库表建立映射关系
public class User {
    //省略其他属性和注解
    @ManyToOne(cascade=CascadeType. ALL,optional=true)
//多对一关系的注解
    @JoinColumn(name="role_id")  //User 表的外键 role_id
    private UserRole userRole=new UserRole();
    //省略属性的 getter/setter 方法
}
```

与创建 hbm. xml 映射配置文件相比，使用注解可以减少代码工作量，其他相关注解的说明在此不再赘述。

6.3.2　Dao 数据访问层

用户管理模块的 Dao 数据访问层是通过 Hibernate 直接访问数据库，定义并实现用户增加、修改、删除、查询和修改角色等 Dao 方法。系统面向接口编程，采用基于泛型＜T＞Dao 接口模式来设计数据访问层。

首先编写泛型的 BaseDao＜T＞接口及 BaseDaoImpl＜T＞抽象实现类，然后编写继承 BaseDao＜T＞接口的 UserDao 接口，最后再编写继承抽象类 BaseDaoImpl＜User＞和实现 UserDao 接口的 UserDaoImpl 实现类。基于泛型的 BaseDao 基类封装了所有 Dao 访问数据库共同的 CRUD 操作，提高了代码复用性。用户管理模块所包含的 Dao 接口和实现类之间的依赖关系如图 7 所示。

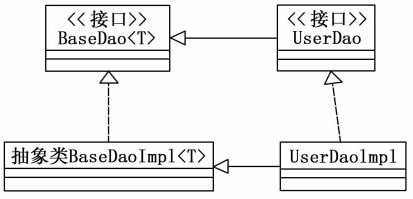

图 7　基于泛型 Dao 的类图

由于 Spring 框架提供了对 Dao 组件的支持，支持用 HibernateTemplate 模板进行数据库持久化访问，所以在 BaseDaoImpl＜T＞实现类中只需要一个 sessionFactory 对象，就可以使用 HibernateTemplate 提供的相关方法对数据库进行增删改查操作[10]。部分代码如下所示：

```
public abstract class BaseDaoImpl＜T＞ extends HibernateDaoSupport implements BaseDao＜T＞ {
    protected SessionFactory sessionFactory;
    private Class＜T＞ entityClass;
    public BaseDaoImpl(){
    //当 BaseDaoImpl＜T＞被继承时,使用该代码可以获取 T 的类型
    entityClass＝(Class＜T＞)((ParameterizedType)getClass().getGenericSuperclass()).getActualTypeArguments()[0];
    }
    public void save(T instance){
    getHibernateTemplate().save(instance);  }
  //省略其他方法
}
```

使用 HibernateTemplate 的好处显而易见，只要传进实体类，不需要编写太多的 SQL 语句，就可以进行数据库持久化操作，再加上泛型的使用，便大大提高了程序的开发效率。

6.3.3 业务逻辑层

用户管理的业务逻辑层主要是实现与用户和角色相关的业务逻辑操作，包括复杂的业务流程处理，也可以调用底层的 Dao 方法来实现之。在业务逻辑层的实现上，系统也采用泛型模式建立泛型接口的 BaseService＜T＞，将所有模块的业务层共同的操作封装起来，便于复用代码。部分关键代码如下：

```
public interface BaseService＜T＞ {
  //声明增删改查的业务方法。
  public void save(T instance);
  public List ＜ T ＞ findByHql ( String hql, Object... params);
  //省略其他方法
  }
public abstract class BaseServiceImpl＜T＞ implements BaseService＜T＞ {
  private BaseDao＜T＞ baseDao;//声明泛型接口
  public void setBaseDao(BaseDao baseDao) {
  this.baseDao = baseDao;
  }
  public void save(T instance) {
  baseDao.save(instance);//实现 save 方法
  }
  //省略其他方法
}
public interface IUserService extends BaseService＜User＞ {
  //声明独有的方法
  public void login(){ }
}
@Service("userService")
public class UserService extends BaseServiceImpl＜User＞ implements IUserService {
@Autowired
  private UserDao userDao;
  public void setUserDao(UserDao userDao) {
//为泛型父类注入具体的 userDao
  super.setBaseDao(userDao);
  this.userDao = userDao;
  }
  public void login() {
  //代码略
  }
}
```

在复杂大型的实际应用中，还可以把泛型的 Dao 和 Service 独立成一个核心包，其他的模块例如用户管理、订单管理等可以直接继承核心包里的泛型 BaseDao 和泛型 BaseService，这样项目的程序结构更加清晰明了。基于泛型接口的 Dao 层和 Service 层搭建好，具体 Dao 实现类和 Service 实现类的代码量少很多，很大程度上降低了程序员的工作量，使程序员集中精力解决业务逻辑问题，加快系统的开发速度。

6.3.4 Web 控制层

Web 控制层由 Spring MVC 负责。系统为用户管理模块编写了一个 UserController 控制器类，包括用户的登录、注册、用户增加、修改、查询、角色修改等业务请求处理方法。来自客户端的用户请求由 Spring MVC 核心——DispatcherServlet 接收，并把请求派发到 Controller 控制器的相应的处理用户请求的方法上，该方法调用业务逻辑层的 Service 方法来处理用户请求，然后把 ModelAndView 返回给 DispatcherServlet。由于 DispatcherServlet 已经配置在 Web. xml 中，开发人员只需编写 Controller 类即可。UserController 类的部分代码示例如下：

```
@Controller //注册为控制器类
//表示"/user"路径下的所有 URL 请求都交由 UserController 处理
@RequestMapping(value="/user")
public class UserController {
  @Autowired
  private UserService userService;
  @Autowired
  private UserRoleService userRoleService;
  //表示"/user/admin"的 URL 请求交给 gotoadmin 方法处理。
  @RequestMapping(value="/admin")
  public String gotoadmin() {
  return "admin－index";
  }
}
```

编写 Controller 类，一是需要声明所需要的 Service 接口，二是通过注解注册 Controller 类和请求地址映射及参数绑定。其中注解@Controller 来将该类注册为 Controller 类，注解@RequestMapping 来处理请求地址映射，注解@PathVariable 或@RequestParam 等进行参数绑定，三是编写具体操作方法，至此可实现控制层功能。

6.3.5 表示层

Spring MVC 有一个标签库，其标签库的 jar 包为 spring. jar，需要加入项目库文件中。使用 Spring MVC 标签可以将请求参数绑定到命令对象上，由于系统已经在 Spring MVC 的配置文件中的配置了 JSP 作为视图响应技术，可以容易地使用 Spring MVC 标签或 JSTL 标签进行含有数据绑定的 JSP 页面开发。

本系统采用 Bootstrap 作为前端响应技术。Bootstrap 是目前很受欢迎的前端开发工具，内置很多漂亮的样式，代码非常简洁，易于修改，具有灵活的响应式栅格系统和良好的浏览器兼容性支持，可以支持桌面浏览器和移动浏览器，本系统采用 Bootstrap 框架，就实现了后台运行一份代码，用户就可以根据需要随时在移动端、PC 端进行交易。图 8 展示了桌面浏览器的用户管理界面。图 9 展示了移动端的图书列表界面。

图 8　桌面端的后台用户管理

图 9　移动端图书管理界面

7　结束语

本文介绍了 Spring MVC、Spring 和 Hibernate 的工作原理和特点，Spring MVC 充当 web 层的控制器，Spring 容器管理 SpringMVC 和 Hibernate，进行业务逻辑处理，Hibernate 负责与后台数据库进行交互实现持久层。以及介绍了基于 SSH 框架的购物网站后台管理基本模块的设计，并以后台系统的用户管理模块为例，简要阐述了在 SSH 框架下的 Web 项目开发的设计与实现。基于 SSH 架构的 Web 应用程序会具有良好的扩展性和复用性，能充分体现基于 MVC 设计模式的三层体系结构的特点和优点，并且层与层之间的高度解耦，使系统更加容易更新与维护，适应快速变化的用户需求。

参考文献：

[1] 张宜浩，涂　飞，刘小洋. Java Web 项目整合开发渐近式教学探索与实施 [J]. 软件工程，2016，19 (8)：51－52.
[2] 胡　杰，周鹏飞，郭乔进. 基于 MVC 设计模式的 SSH 框架的研究 [J]. 信息化研究，2016 (1)：17－22.
[3] 杨瑞东，王云峰，张海英. Spring 新特性之 Java Config 在 Web 开发中的应用 [J]. 微型机与应用，2017，36 (18)：26－29.
[4] 陶　晶，李　娜. 借助 WEB 技术将 IPD 理念运用于企业研发项目的信息管理 [J]. 项目管理技术，2017，15 (3)：108－112.
[5] 王　崟，陆莉莉. Java web 课程项目驱动式教学中的几点思考 [J]. 电脑知识与技术，2017，13 (18)：140－141.
[6] 张小龙，孔勇强，胡志明，等. 基于 Extjs＋SSH 框架的电子商务系统 [J]. 中国科技信息，2017 (10)：65－67.
[7] 王　磊，刘　娜，马晓明. 基于 SSH 架构的安防系统设计与实现 [J]. 电子科技，2016，29 (2)：89－91.
[8] 黄　涛. 面向项目的“Web 应用开发”教学实践与研究 [J]. 无线互联科技，2017 (6)：92－93.
[9] 巩宇航，于晓明. 基于 SSH 框架的教务管理系统模型实现 [J]. 信息通信，2016 (4)：124－125.
[10] 李超鹏，卜智勇. 基于实时计算时延扩展的 LMMSE 信道估计算法 [J]. 电子设计工程，2017，25 (1)：82－85.

(上接第 109 页)

观察，极大地方便了操作者对产品测试结果的进一步分析和故障定位。

参考文献：

[1] 赵　敏，吴卫山. 弹载雷达导引技术发展趋势及其关键技术 [J]. 飞航导弹，2017 (1)：80－84.
[2] 白志刚，吴　斌，刘书信，等. 基于 PC 平台的图像导引头操控测试系统设计 [J]. 中国测试，2015，41 (Z1)：96－100.
[3] 杜　江，李　淼，王天辉. 基于测试诊断一体化技术的雷达导引头技术支援系统设计 [J]. 计算机测量与控制，2014，22 (3)：775－777.
[4] 张　磊. 航空 ARINC429 总线测试分析仪的研究与实现 [D]. 天津：中国民航大学，2010. 4.
[5] 祖先锋，韩玉芹，李　猛，等. 空空导弹制导软件实时监控调试系统 [J]. 中国测试，2015，41 (4)：110－113.
[6] 吴琼之，南　方，张　峰. 并行 DSP 系统软件调试器设计与实现 [J]. 北京理工大学学报，2011，31 (7)：855－858.
[7] 王白江，蔡惠智，冯欣欣，等. 基于总线的多 DSP 交叉调试器的设计与实现 [J]. 计算机工程，2007，33 (20)：239－24.
[8] 高　山，翟龙君，曲洪东，等. 一种基于 DRFM 和数字信道化技术的宽带雷达目标干扰模拟器设计 [J]. 海军航空工程学院学报，2017，32 (2)：205－214.
[9] 毕锐锐. 宽带 PD 雷达回波模拟器设计与性能验证 [D]. 北京：北京理工大学，2015.
[10] 王子龙，路景泽. 基于 CPCI 总线的雷达导引头测试系统设计与实现 [J]. 计算机测量与控制，2016，24 (7)：141－143.
[11] 张海丽，苏淑靖. 基于 LabWindows/CVI 的测试仪软件设计 [J]. 测控技术，2012，31 (7)：88－91.