# 系统开发指南

目录

[系统开发指南 1](#_Toc355942403)

[1. 系统架构概述 2](#_Toc355942404)

[1.1分层架构模型 2](#_Toc355942405)

[1.2组件架构模型 2](#_Toc355942406)

[1.3数据流转模型 3](#_Toc355942407)

[2.框架目录介绍 3](#_Toc355942408)

[2.1Cms包介绍 3](#_Toc355942409)

[2.2Common包介绍 3](#_Toc355942410)

[2.3Core包介绍 3](#_Toc355942411)

[2.4页面资源介绍 3](#_Toc355942412)

[3. 开发流程指导 3](#_Toc355942413)

[3.1展现层 3](#_Toc355942414)

[3.1.1Freemarer介绍 3](#_Toc355942415)

[3.1.2后台页面 3](#_Toc355942416)

[3.1.3前台模版页面 3](#_Toc355942417)

[3.2控制层 3](#_Toc355942418)

[3.2.1SpringMVC简介 3](#_Toc355942419)

[3.2.2控制层实现 3](#_Toc355942420)

[3.3逻辑层 3](#_Toc355942421)

[3.3.1类存放包介绍 3](#_Toc355942422)

[3.3.2service类代码编写样例 3](#_Toc355942423)

[3.3.3配置service bean 3](#_Toc355942424)

[3.4持久层 3](#_Toc355942425)

[3.4.1Hibernate3简介 3](#_Toc355942426)

[3.4.2DAO类代码编写样例 3](#_Toc355942427)

[3.4.3配置DAO bean 3](#_Toc355942428)

[3.4.4POJO 3](#_Toc355942429)

[3.5自定义标签 3](#_Toc355942430)

[3.5.1定义标签类 3](#_Toc355942431)

[3.5.2配置标签 3](#_Toc355942432)

[3.5.3标签应用 3](#_Toc355942433)

[3.6系统间交互 3](#_Toc355942434)

[3.6.1 xfire支持 3](#_Toc355942435)

[4.开发工具使用 3](#_Toc355942436)

[4.1基础代码生成步骤 3](#_Toc355942437)

[4.1.1将hibernate mapping file插件引入到myeclipse 3](#_Toc355942438)

[4.1.2插件的使用说明 3](#_Toc355942439)

[4.2代码生成器生成Dao，Service，Action后台页面步骤 3](#_Toc355942440)

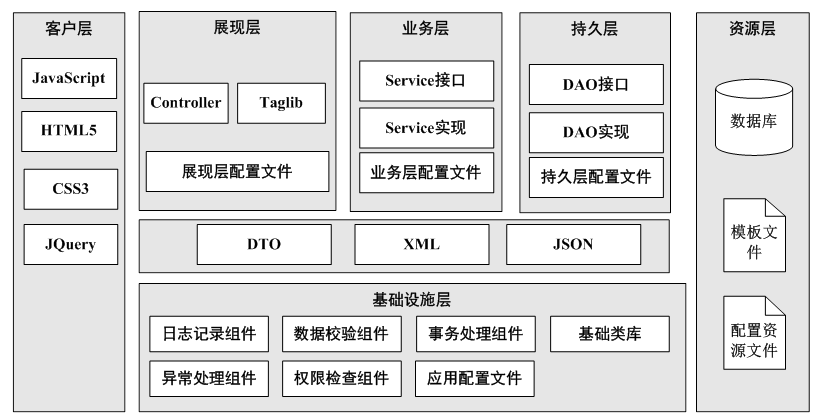
[4.2.1配置properties文件 3](#_Toc355942441)

[4.2.2运行相关的java文件 3](#_Toc355942442)

# 系统架构概述

本系统核心架构为FreeMarker+hibernate+Spirng的mvc分层架构。

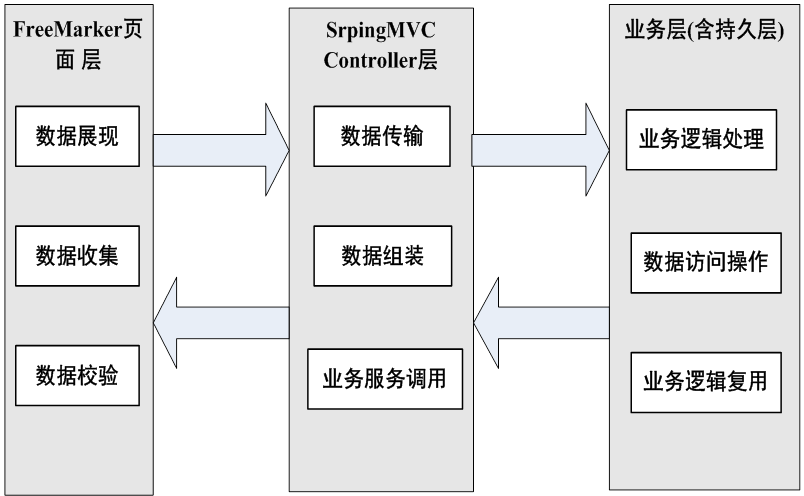
## 1.1分层架构模型



## 1.2组件架构模型



## 1.3数据流转模型



# 2.框架目录介绍

## 2.1Cms包介绍

|  |  |
| --- | --- |
| 具体包名 | 相关功能描述 |
| com\jspgou\cms | 常量包 |
| com\jspgou\cms\action\admin | 后台欢迎页的action |
| com\jspgou\cms\action\admin\main | 后台核心功能包 |
| com\jspgou\cms\action\directive | cms自定义标签包,公用部分继承于abs包 |
| com\jspgou\cms\action\directive\abs | 公用部分自定义标签的抽象包 |
| com\jspgou\cms\action\front | 前台action包 |
| com\jspgou\cms\action\front\ utils | 前台图片辅助包 |
| com\jspgou\cms\action\member | 会员中心包 |
| com\jspgou\cms\dao\assist | 辅助dao的接口层 |
| com\jspgou\cms\dao\assist\impl | 辅助dao的接口实现层 |
| com\jspgou\cms\dao | 核心的dao接口层 |
| com\jspgou\cms\dao\impl | 核心的dao接口实现层 |
| com\jspgou\cms\entity | cms核心实体类子类 |
| com\jspgou\cms\entity\base | cms核心实体类基类 |
| com\jspgou\cms\entity\hbm | hibernate核心实体类的关系映射文件 |
| com\jspgou\cms\lucene | 搜索引擎封装包 |
| com\jspgou\cms\manager | 核心的service接口层 |
| com\jspgou\cms\manager\impl | 核心的service接口实现层 |
| com\jspgou\cms\service | service封装接口层 |
| com\jspgou\cms\service\impl | service封装接口实现层 |
| Com\jspgou\cms\template | 代码生成器包 |
| com\jspgou\cms\web | 拦截器包 |
| com\jspgou\cms\web\threadvariable | 线程包 |

## 2.2Common包介绍

|  |  |
| --- | --- |
| 具体包名 | 相关功能描述 |
| com\jspgou\common\checkcode | 验证码封装包 |
| com\jspgou\common\developer | 代码生成器辅助包 |
| com\jspgou\common\fck | fck编辑器封装包 |
| com\jspgou\common\file | 文件操作封装包 |
| com\jspgou\common\file\lucene | 收索引擎的基础包 |
| com\jspgou\common\hibernate3 | hibeinate封装包 |
| com\jspgou\common\image | 图片的封装包 |
| com\jspgou\common\page | jspgou分页封装包 |
| com\jspgou\common\security | 安全认证相关的包 |
| com\jspgou\common\security\annotation |
| com\jspgou\common\security\encoder |
| com\jspgou\common\security\rememberme |
| com\jspgou\common\security\userdetails |
| com\jspgou\common\upload | 上传封装包 |
| com\jspgou\common\util | 工具包 |
| com\jspgou\common\web | 常量包 |
| com\jspgou\common\web\freemarker | freemarker视图封装包 |
| com\jspgou\common\web\session | session包 |
| com\jspgou\common\web\session\cache | session缓存包 |
| com\jspgou\common\web\springmvc | springMVC的简单封装 |

## 2.3Core包介绍

|  |  |
| --- | --- |
| 具体包名 | 相关功能描述 |
| com\jspgou\core\cache | 缓存接口层 |
| com\jspgou\core\cache\impl | 缓存接口实现层 |
| com\jspgou\core\dao | 核心dao的接口 |
| com\jspgou\core\dao\impl | 核心dao的接口实现类 |
| com\jspgou\core\entity | 登录认证和核心用户的实体子类 |
| com\jspgou\core\entity\base | 登录认证和核心用户的实体基类 |
| com\jspgou\core\entity\hbm | 相关的hibernate的映射文件 |
| com\jspgou\core\manager | 核心的service接口层 |
| com\jspgou\core\manager\impl | 核心的service接口实现层 |
| com\jspgou\core\resolver | 获得文件路径包 |
| com\jspgou\core\security | 用户没有激活异常包 |
| com\jspgou\core\tpl | 模板接口和相关service层 |
| com\jspgou\core\web | 定义内容显示的接口，工具类，和错误页面的指定 |
| com\jspgou\core\web\admin | 后台（管理员）本地化信息拦截器包 |
| com\jspgou\core\web\front | uri帮助类 |

## 2.4页面资源介绍

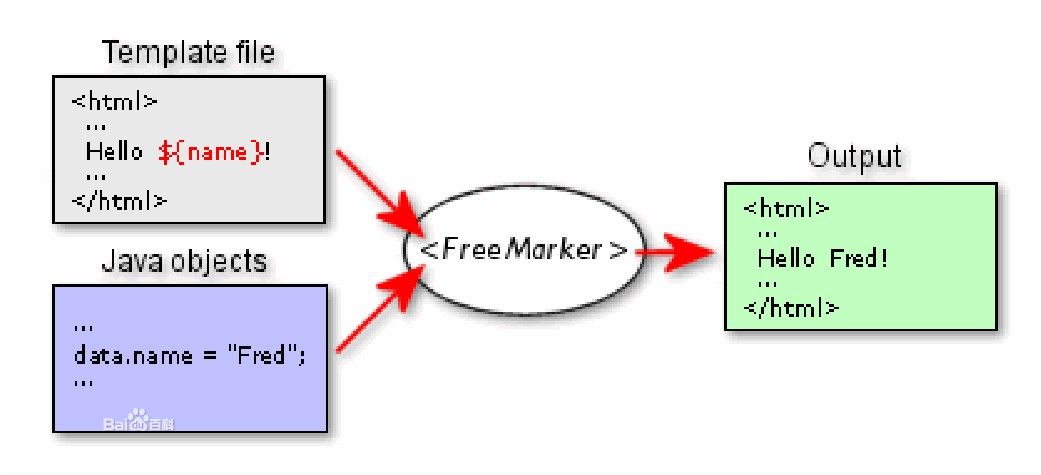
|  |  |  |
| --- | --- | --- |
| 具体包名 | 相关功能描述 | |
| WebContent | | |
| r | 前台资源文件，如css、img、js等 | |
| res | 后台资源文件 | |
| thirdparty | 第三方插件（fck编辑器、swf上传、My97DatePicker日期选择） | |
| WebContent/WEB-INF | | |
| common | 通用页面 | |
| config | 核心的配置文件，如果action，service,manager,dao,bean等 | |
| jspgou\_sys | 后台管理页面 | |
|  | address | 地区 |
|  | admin | 管理员 |
|  | adspace | 广告版位 |
|  | advertise | 广告 |
|  | aftersale | 售后管理 |
|  | aftersaleDemand | 售后服务要求 |
|  | aftersaleType | 售后服务类型 |
|  | article | 文章管理 |
|  | balance | 结算管理 |
|  | brand | 品牌 |
|  | category | 商品类别 |
|  | channel | 栏目 |
|  | config | 系统配置 |
|  | consult | 顾客咨询 |
|  | coupon | 优惠劵 |
|  | discuss | 商品评论 |
|  | frame | 框架集 |
|  | fullGive | 满就送 |
|  | gift | 礼品 |
|  | group | 会员组 |
|  | member | 会员 |
|  | order | 订单 |
|  | orderReturn | 退款订单 |
|  | payment | 支付方式 |
|  | poster | 首页海报 |
|  | product | 商品 |
|  | productColor | 商品颜色规则 |
|  | promote | 打折 |
|  | refund | 退款管理 |
|  | report | 举报投诉 |
|  | reportType | 举报投诉类型 |
|  | resource | 资源管理 |
|  | role | 角色 |
|  | sale | 销售统计 |
|  | shipping | 配送方式 |
|  | shopDictionary | 数据字典 |
|  | shopDictionaryType | 数据字典类型 |
|  | shopMoney | 账户记录 |
|  | shopPurchase | 购买 |
|  | shopScore | 积分 |
|  | standard | 规格 |
|  | standardType | 规格类型 |
|  | store | 店铺管理 |
|  | storeCategory | 店铺类别 |
|  | storeExt | 店铺附属 |
|  | storeModel | 店铺模板 |
|  | storeTxt | 店铺内容 |
|  | tag | Tag关键词 |
|  | template | 模版 |
|  | type | 商品类型 |
|  | typeProperty | 商品属性 |
| languages | 国际化配置 | |
|  | fckeditor | 编辑器 |
|  | jeecore\_admin | 后台 |
|  | jeecore\_front | 前台 |
|  | jspgou\_admin | 后台 |
|  | jspgou\_front | 前台 |
| lib | 存放jar | |
| t/gou | 前台模板页面 | |
| t/gou\_sys\_defined | 前台模板分页 | |

# 3. 开发流程指导

## 3.1展现层

### 3.1.1Freemarer介绍

FreeMarker是一个用Java语言编写的模板引擎，它基于模板来生成文本输出。FreeMarker与Web容器无关，即在Web运行时，它并不知道Servlet或HTTP。它不仅可以用作表现层的实现技术，而且还可以用于生成XML，JSP或Java 等。

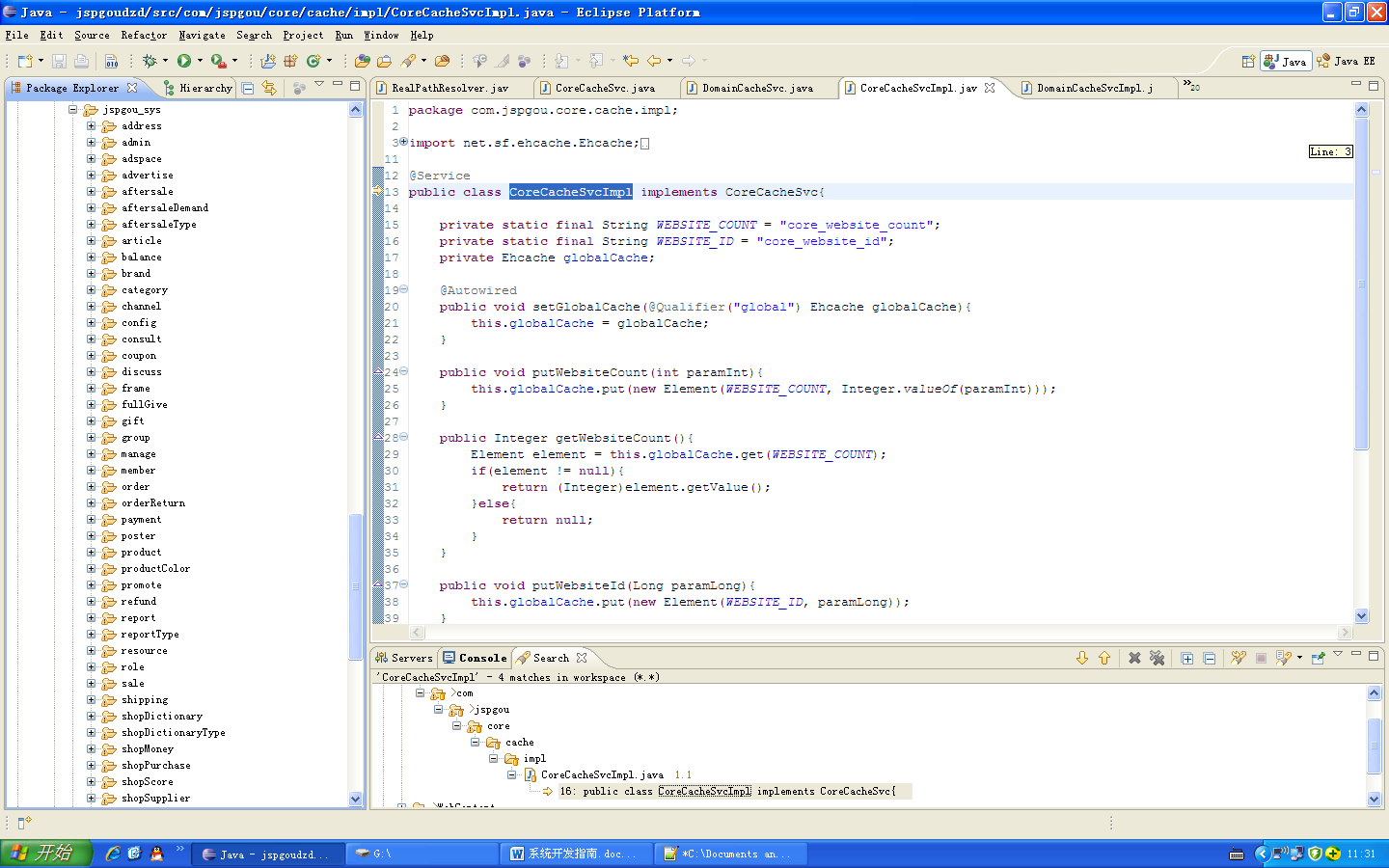


### 3.1.2后台页面

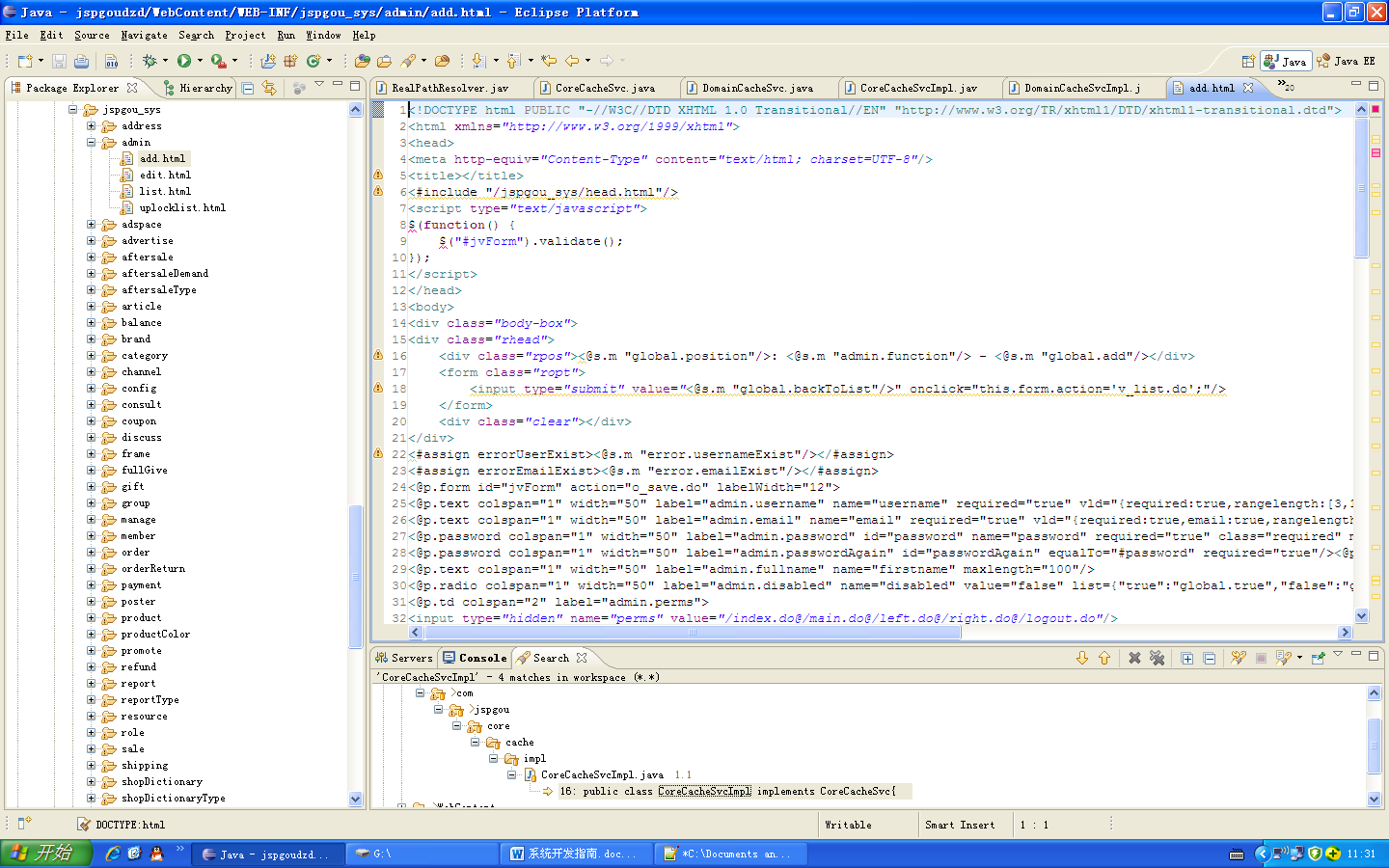
后台页面需要的文本信息在languages/jspgou\_admin下messages\_zh\_CN.properties中配置国际化属性

cms.function.assistant=辅助

spring中配置好资源文件，使spring mvc在html文件能够读到资源文件信息。

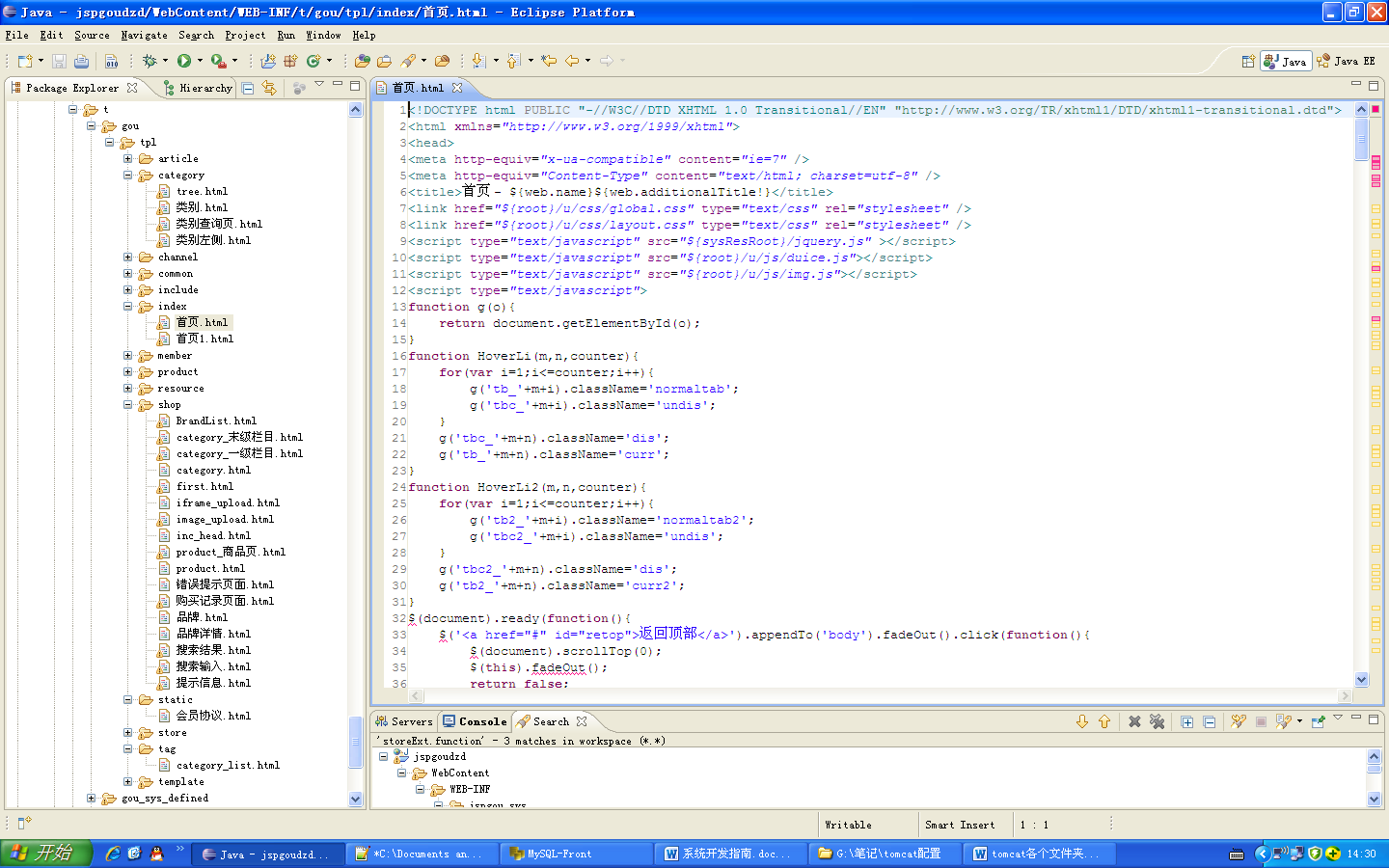


在jspgou\_sys文件夹中添加功能板块文件夹和相应的管理页面



读取控制层的属性非常简单，freemark只需用类似jsp的el表达式输出普通对象。对于控制层传输过来的集合对象可用freemark内置对象list来循环出来

### 3.1.3前台模版页面



在red目录相应的文件夹添加html，在html文件中引入freemark机制控制control层传输对象的显示。

## 3.2控制层

### 3.2.1SpringMVC简介

Spring 框架提供了构建 Web 应用程序的全功能 MVC 模块。使用 Spring 可插入的 MVC 架构，通过策略接口，Spring 框架是高度可配置的，而且包含多种视图技术，例如 FreeMarker、JavaServer Pages（JSP）技术、Velocity、Tiles、iText 和 POI。Spring MVC 框架并不知道使用的视图，所以不会强迫您只使用 JSP 技术。Spring MVC 分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。

### 3.2.2控制层实现

#### 3.2.2.1配置SpringMVC支持

Web.xml配置Spring后台核心调度器

<servlet>

<servlet-name>JspGouAdmin</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/config/jspgou-servlet-admin.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

Web.xml配置Spring后台核心调度器匹配路径

<servlet-mapping>

<servlet-name>JspGouAdmin</servlet-name>

<url-pattern>/jeeadmin/jspgou/\*</url-pattern>

</servlet-mapping>

Web.xml配置Spring前台核心调度器

<servlet>

<servlet-name>JspGouFront</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/config/jspgou-servlet-front.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

Web.xml配置Spring前台核心调度器匹配路径

<servlet-mapping>

<servlet-name>JspGouFront</servlet-name>

<url-pattern>\*.jhtml</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>JspGouFront</servlet-name>

<url-pattern>\*.htm</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>JspGouFront</servlet-name>

<url-pattern>\*.jsp</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>JspGouFront</servlet-name>

<url-pattern>\*.jspx</url-pattern>

</servlet-mapping>

#### 3.2.2.2Spring Controller

com.jspgou.cms.action.admin.main核心管理模块Controller

/\*\*

@Controller标志该类是Srping Controller

\*\*/

@Controller

public class WelcomeAct {

/\*

\*/index.do该地址将由Spring DispatchServlet负责调度给该方法处理

\*/

@RequestMapping("/index.do")

public String index() {

return "index";

}

@RequestMapping("/top.do")

public String top(HttpServletRequest request, ModelMap model) {

// 需要获得站点列表

List<CmsSite> siteList = cmsSiteMng.getList();

CmsSite site = CmsUtils.getSite(request);

CmsUser user = CmsUtils.getUser(request);

// 传输对象到view层

model.addAttribute("siteList", siteList);

model.addAttribute("site", site);

model.addAttribute("siteParam", AdminContextInterceptor.SITE\_PARAM);

model.addAttribute("user", user);

return "top";

}

/\*\*

\*装配service层对象，调用业务逻辑层

\*/

@Autowired

private CmsSiteMng cmsSiteMng;

}

#### 3.2.2.3SpringMVC扩展配置

jspgou-admin-servlet.xml 后台action配置文件

jspgou-front-servlet.xml前台action配置文件

配置action bean

<bean id="welcomeAct" class="com.jspgou.cms.action.admin.WelcomeAct"/>

## 3.3逻辑层

### 3.3.1类存放包介绍

com.jspgou.cms.manager.impl核心service层

### 3.3.2service类代码编写样例

@Service

@Transactional

public class CmsConfigMngImpl implements CmsConfigMng {

@Transactional(readOnly = true)

public CmsConfig get() {

CmsConfig entity = dao.findById(1);

return entity;

}

public void updateCountCopyTime(Date d) {

dao.findById(1).setCountCopyTime(d);

}

public void updateCountClearTime(Date d) {

dao.findById(1).setCountClearTime(d);

}

public CmsConfig update(CmsConfig bean) {

Updater<CmsConfig> updater = new Updater<CmsConfig>(bean);

CmsConfig entity = dao.updateByUpdater(updater);

entity.blankToNull();

return entity;

}

public MarkConfig updateMarkConfig(MarkConfig mark) {

get().setMarkConfig(mark);

return mark;

}

public void updateMemberConfig(MemberConfig memberConfig) {

get().getAttr().putAll(memberConfig.getAttr());

}

private CmsConfigDao dao;

//自动装配dao层

@Autowired

public void setDao(CmsConfigDao dao) {

this.dao = dao;

}

}

需要@Service@Transactional

注解标志该类为业务逻辑层，所有的service层均采用接口开发模式

@Transactional(readOnly = true)只读事物

### 3.3.3配置service bean

jspgou-context.xml中增加service bean的配置

<bean id="cmsConfigMng" class="com.jspgou.cms.manager.impl.CmsConfigMngImpl"/>

## 3.4持久层

### 3.4.1Hibernate3简介

Hibernate是一个开放源代码的对象<http://baike.baidu.com/view/2387.htm>关系映射框架，它对JDBC进行了非常轻量级的对象封装，使得Java程序员可以随心所欲的使用对象编程思维来操作数据库。 Hibernate可以应用在任何使用JDBC的场合，既可以在Java的客户端程序使用，也可以在Servlet/JSP的Web应用中使用

### 3.4.2DAO类代码编写样例

持久层采用Hibernate3，缓存采用Ehcache

com.jspgou.cms.dao.impl 核心DAO层

@Repository

public class CmsConfigDaoImpl extends HibernateBaseDao<CmsConfig, Integer>

implements CmsConfigDao {

public CmsConfig findById(Integer id) {

CmsConfig entity = get(id);

return entity;

}

//重写getEntityClass方法

@Override

protected Class<CmsConfig> getEntityClass() {

return CmsConfig.class;

}

}

@Repository注解标志该类是DAO层组件，可以选择继承HibernateBaseDao基础类，需要实现接口。

### 3.4.3配置DAO bean

jspgou-context.xml中增加dao bean的配置

<bean id="cmsConfigDao" class="com.jspgou.cms.dao. impl.CmsConfigDaoImpl"/>

### 3.4.4POJO

com.jspgou.cms.entity核心功能包的pojo

com.jspgou.cms.entity.base 辅助功能包的pojo基础类

com.jspgou.cms.entity.hbm 辅助功能包Hibernate实体映射文件

## 3.5自定义标签

### 3.5.1定义标签类

自定义标签类所属包com.jspgou.cms.action.directive

标签类需要实现Freemarker内置接口TemplateDirectiveModel

获取标签参数可以用DirectiveUtils工具类获取

public class ChannelDirective implements TemplateDirectiveModel {

/\*\*

\* 输入参数，栏目ID。

\*/

public static final String PARAM\_ID = "id";

/\*\*

\* 输入参数，栏目路径。

\*/

public static final String PARAM\_PATH = "path";

/\*\*

\* 输入参数，站点ID。存在时，获取该站点栏目，不存在时获取当前站点栏目。

\*/

public static final String PARAM\_SITE\_ID = "siteId";

@SuppressWarnings("unchecked")

public void execute(Environment env, Map params, TemplateModel[] loopVars,

TemplateDirectiveBody body) throws TemplateException, IOException {

CmsSite site = FrontUtils.getSite(env);

//get required params from directive

Integer id = DirectiveUtils.getInt(PARAM\_ID, params);

Channel channel;

if (id != null) {

channel = channelMng.findById(id);

} else {

String path = DirectiveUtils.getString(PARAM\_PATH, params);

if (StringUtils.isBlank(path)) {

// 如果path不存在，那么id必须存在。

throw new ParamsRequiredException(PARAM\_ID);

}

Integer siteId = DirectiveUtils.getInt(PARAM\_SITE\_ID, params);

if (siteId == null) {

siteId = site.getId();

}

channel = channelMng.findByPathForTag(path, siteId);

}

Map<String, TemplateModel> paramWrap = new HashMap<String, TemplateModel>(

params);

//put result to view

paramWrap.put(OUT\_BEAN, DEFAULT\_WRAPPER.wrap(channel));

Map<String, TemplateModel> origMap = DirectiveUtils

.addParamsToVariable(env, paramWrap);

//render result to response

body.render(env.getOut());

DirectiveUtils.removeParamsFromVariable(env, paramWrap, origMap);

}

//装配所需service

@Autowired

private ChannelMng channelMng;

}

### 3.5.2配置标签

1.jspgou-context.xml配置Spring bean

<bean id="cms\_content\_list" class="com.jspgou.cms.action.directive.ContentListDirective"/>

2.jspgou-context 文件中freemarkerVariables变量中引入自定义标签bean

<entry key=" cms\_content\_list " value-ref=" cms\_content\_list "/>

3. jspgou-front-servlet.xml配置文件中freemarkerVariables变量中引入自定义标签bean

<entry key=" cms\_content\_list " value-ref=" cms\_content\_list "/>

### 3.5.3标签应用

[@cms\_content\_list count='9' titLen='15' orderBy='8' channelOption='1' channelId='1']

[#list tag\_list as a]

<li><a href="${a.url}" title="${a.title}" target="\_blank">[@text\_cut s=a.title len=titLen append=append/]</a></li>

[/#list]

[/@cms\_content\_list]

cms\_content\_list标签名称count、titLen、orderBy、channelOption、channelId标签参数

tag\_list标签结果

a循环变量

${a.url}输出a对象的url属性

## 3.6系统间交互

本系统提供了webservice系统间交互的方式，webservice集成xfire框架实现。

### 3.6.1 xfire支持

本系统xfire提供作为webservice服务器端供其他系统调用的接口，并可作为客户端调用相关系统的webservice接口以获取服务。

##### 服务器端

Xfire作为服务器端开发流程为：

1. 编写服务接口，服务接口package为com.jspgou.cms.webservice；示例如下：

public interface WebServiceTest {

public Integer add(int a,int b );

}

1. 编写接口实现类，实现类package为com.jspgou.cms.webservice.impl；示例如下：

public class WebServiceTestImpl implements WebServiceTest {

public Integer add(int a, int b) {

return a+b;

}

}

1. 配置WEB-INF/xfire-servlet.xml以发布服务，示例如下：

<beans>

<!-- add by wangguowei -->

<bean

class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">

<property name="urlMap">

<map>

<entry key="/WebServiceTest">

<ref bean="webServiceTestBean" />

</entry>

</map>

</property>

</bean>

<bean id="webServiceTestBean" class="org.codehaus.xfire.spring.remoting.XFireExporter">

<property name="serviceFactory">

<ref bean="xfire.serviceFactory" />

</property>

<property name="xfire">

<ref bean="xfire" />

</property>

<property name="serviceBean">

<ref bean="webServiceTest" />

</property>

<property name="serviceClass">

<value>com.jspgou.cms.webservice.WebServiceTest</value>

</property>

</bean>

1. 测试。

##### 客户端

Xfire作为客户端调用webservice示例代码如下：

WebServiceTest webServiceTest=null;

Service srvcPrintService = new ObjectServiceFactory().create(WebServiceTest.class);

XFireProxyFactory factory = new XFireProxyFactory(XFireFactory.newInstance().getXFire());

try {

webServiceTest = ((WebServiceTest)factory.create(srvcPrintService, "http://localhost:7001/WebContent/services/WebServiceTest"));

}

catch (Exception e) {

e.printStackTrace();

}

System.out.println("返回值=="+webServiceTest.add(1, 3));

# 4.开发工具使用

## 4.1基础代码生成步骤

### 4.1.1将hibernate mapping file插件引入到myeclipse

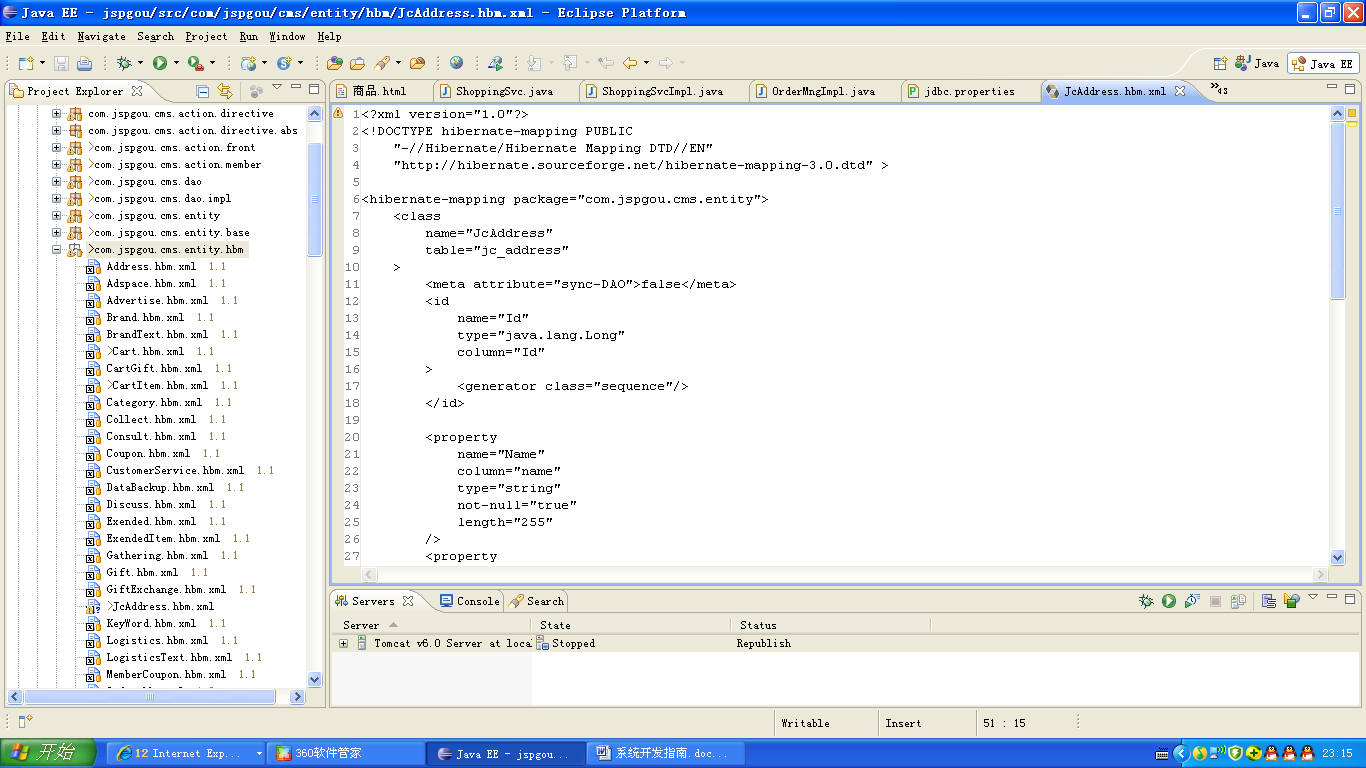
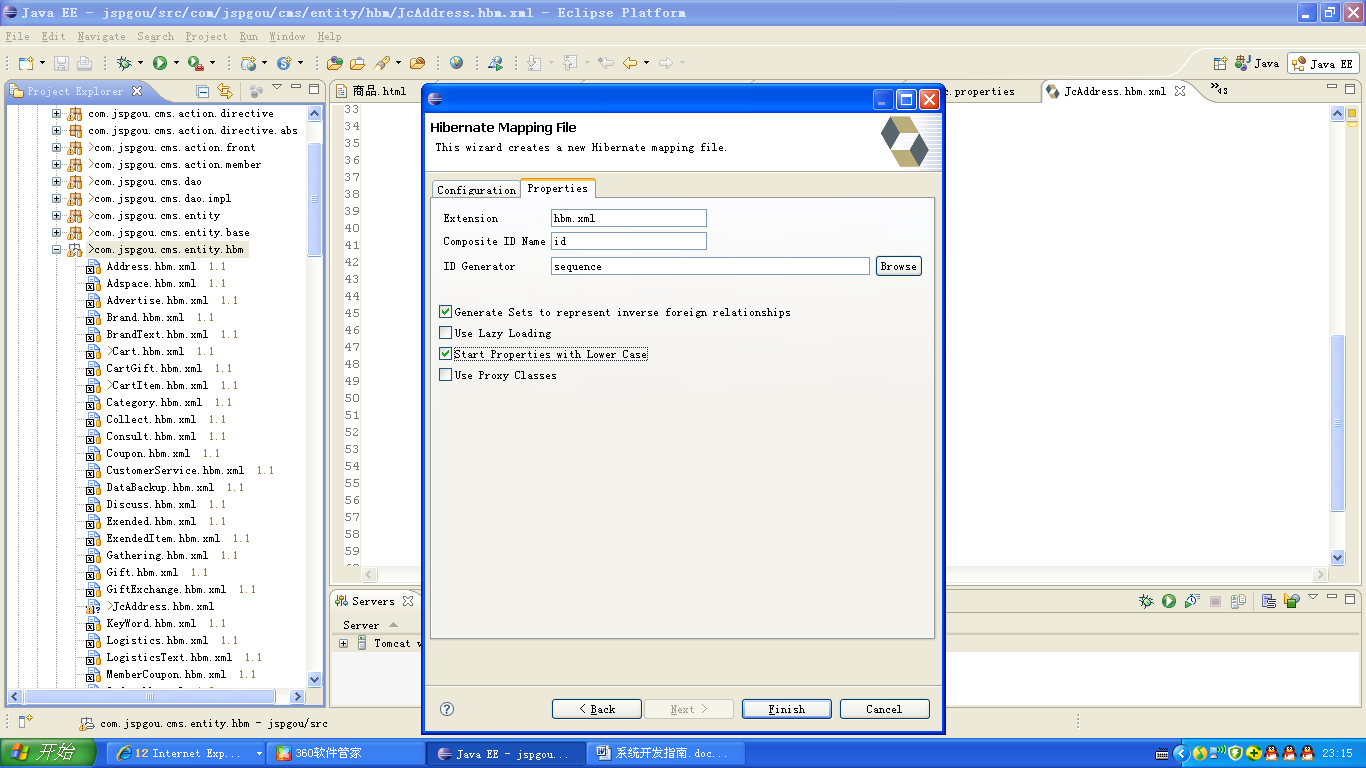
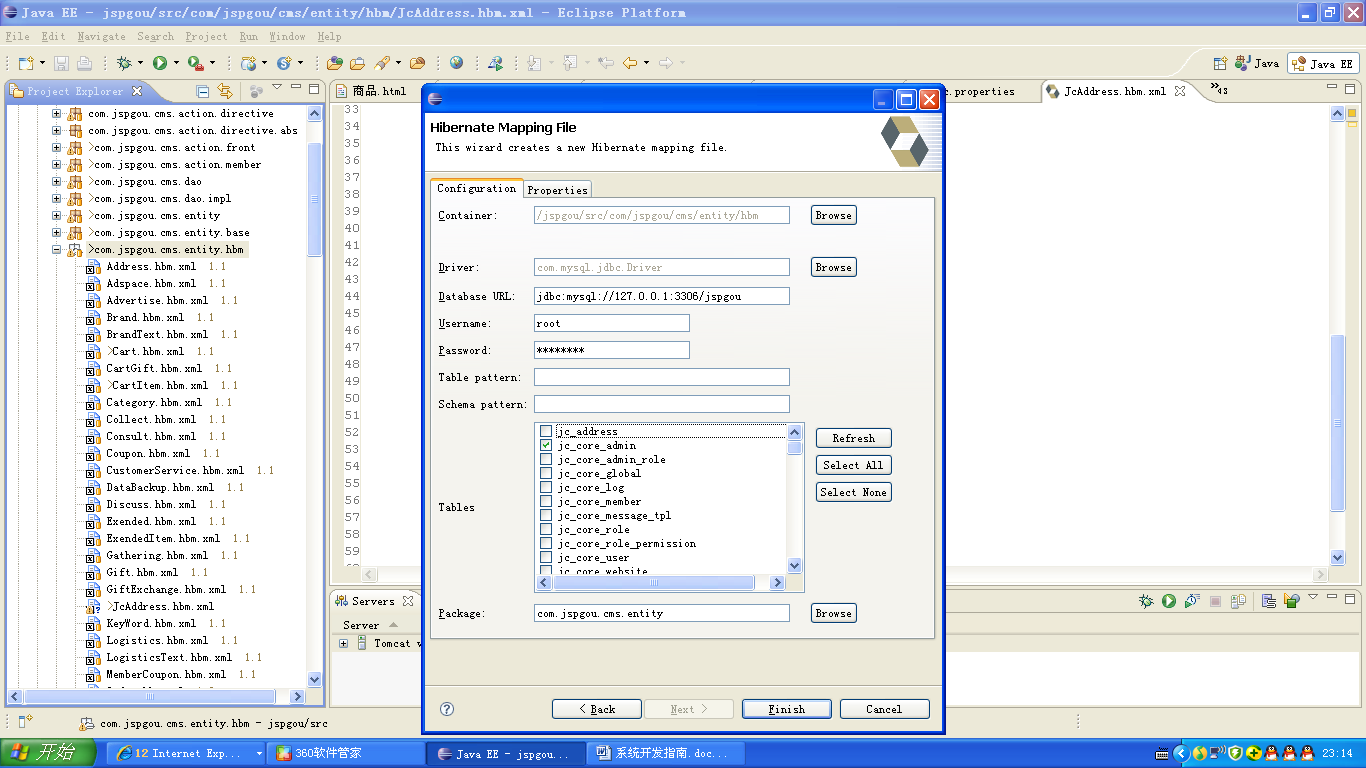
注意事项：1.myeclipse版本最好为6.5或者6.0版本

2.将插件直接放入到myeclipse安装包的plugins包下面

3.重启服务

### 4.1.2插件的使用说明

#### 4.1.2.1选中需要生成映射文件对用的文件夹

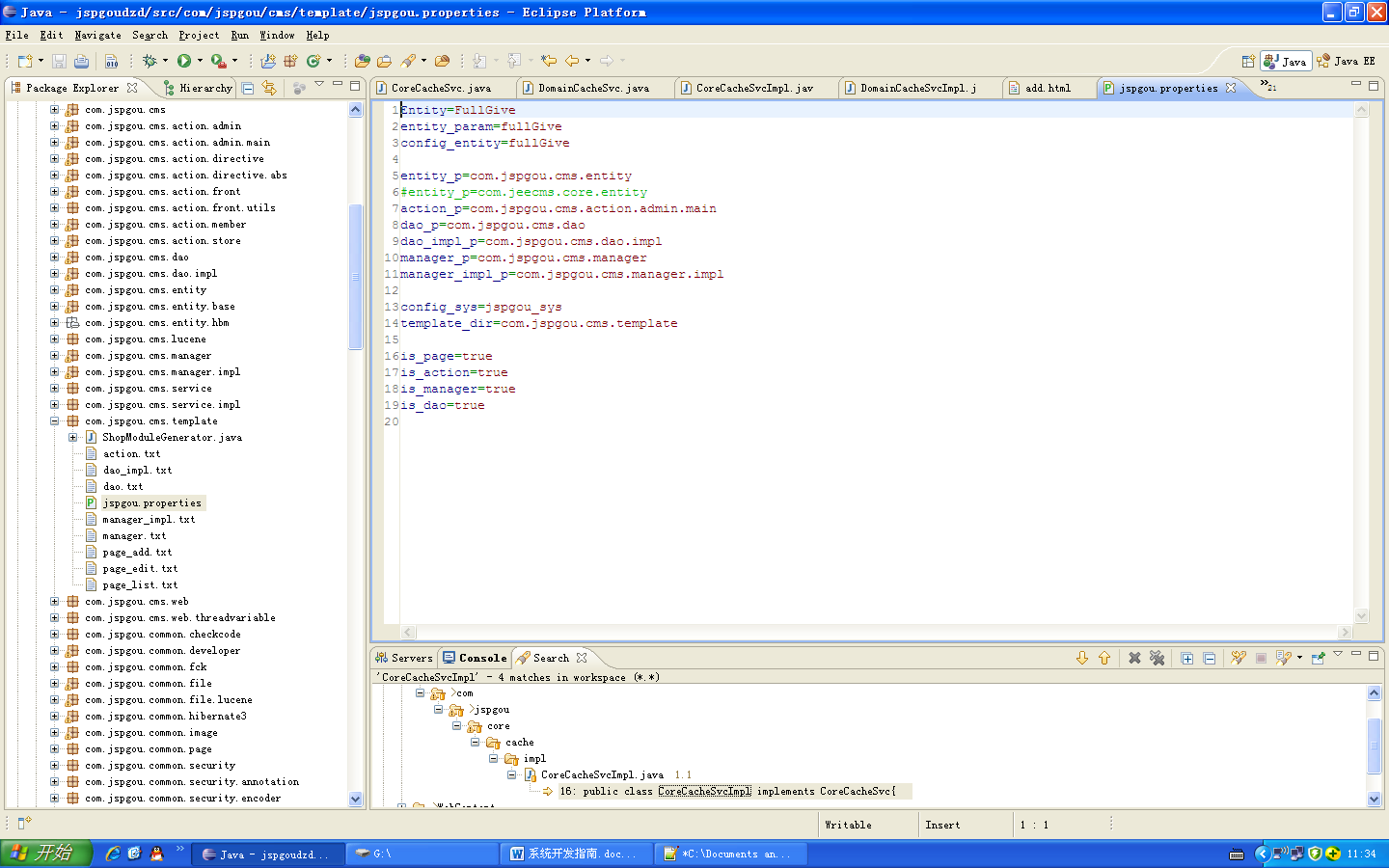


#### 4.1.2.2生成entity层

将修改完的生映射文件进行保存，会自动生成entity层（包括base层）

## 4.2代码生成器生成Dao，Service，Action后台页面步骤

### 4.2.1配置properties文件



### 4.2.2运行相关的java文件

com.jspgou.cms.template.ShopModuleGenerator.java打开后Run As—java application