**重点在：hibernate逆向工程**

# 项目介绍

## 项目背景

国税协同办公平台包括了行政管理、后勤服务、在线学习、纳税服务、我的空间等几个子系统；我们本次主要的开发功能是纳税服务子系统的部分模块和基于纳税服务子系统信息的个人工作台首页。纳税服务子系统是办税PC前端或移动端的后台管理系统，主要包括的功能有系统角色管理、用户管理、信息发布管理、投诉受理、纳税咨询、易告知、服务预约、服务调查等模块。





## 项目前期

项目前期；一般是由客户经理从客户那边了解到有关该项目的招标信息，然后开发公司再组织竞标；编写标书和详细的项目建设方案。这个阶段一般由商务部门和开发部门的管理层参与。建设方案由部门经理或者项目经理协助完成。项目建设方案中一般包括初步的系统功能、模块设计、软硬件设备、开发组人员介绍等。提供这些信息给客户后他们再评估相应开发公司是否有承建系统能力。

## 需求分析

当项目中标后，公司的客户经理、项目经理再到客户环境进行详细的需求调研，完成调研后将初步的调研结果转化成正式的需求文档。需求文档是所有后续流程的基础和非常重要的文档，开发和测试都需要按照需求来开发测试。详细需求见《国税协同平台-纳税服务需求规格说明书.docx》

## 设计

在这个阶段主要是针对需求进行详细的开发前的分析和对具体模块的设计。在设计过程中可能包括到的有系统原型设计的完善、界面美工设计切图、数据库设计（概念、物理模型）等；最终需要出详细设计说明书。见《国税协同平台-纳税服务详细设计说明书.docx》

## 制定开发计划

在需求和详细设计都已经确定的情况下；项目需要根据开发的时间和任务数来制定开发计划；计划中需要描述所有的开发阶段，从需求阶段到部署上线期间所有要做的任务，每个任务对应具体的开始时间、结束时间，前置任务，工作时长，资源名称。后续的开发都根据该计划进行在每周开周例会上确认开发进度是否和计划保持一致，如果不一致则需要预测好相应方案；保证能如期提交系统给到客户。

## 开发

根据开发计划，项目经理将组织开发团队完成具体的开发任务，每周向公司部门、客户主管人员汇报项目进度。对应具体的开发人员则熟悉、理解需求，明白设计文档；按照计划中时间点开发功能，提交代码。

# 框架整合

## 新建数据库及web项目

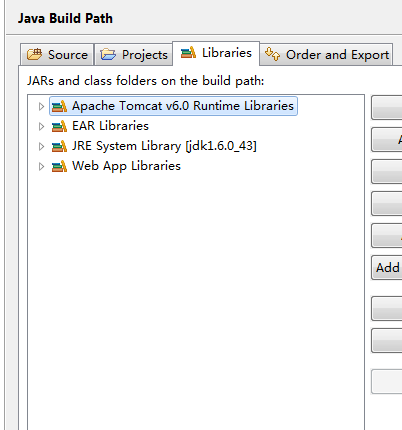
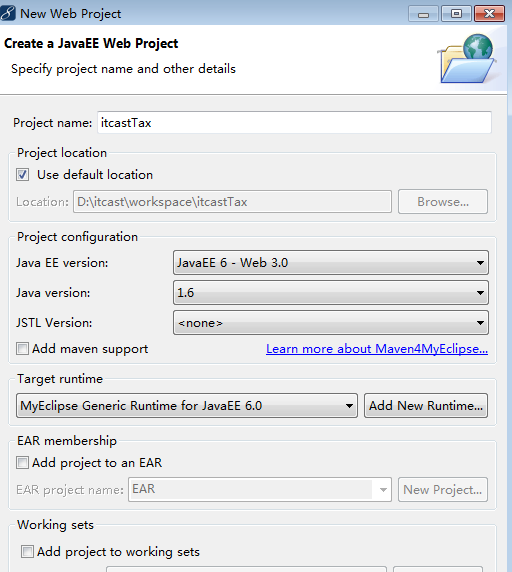
### 创建itcastTax数据库

-- 创建数据库

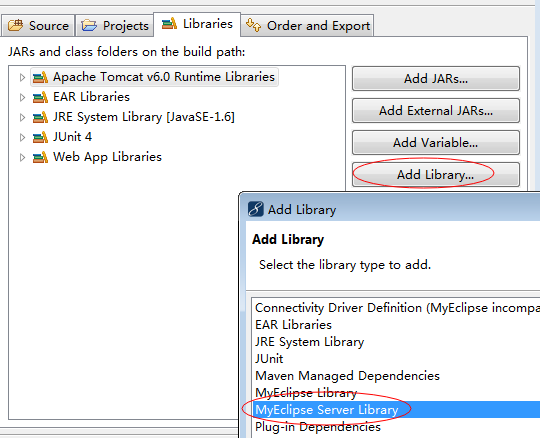
CREATE DATABASE itcastTax DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;

### 新建web项目

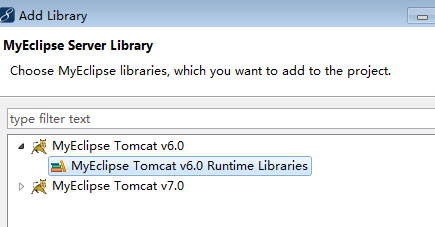
新建工作空间指定项目编码（或工作空间编码）为utf-8，再建 web project，配置buildpath



引入tomcat 的包：

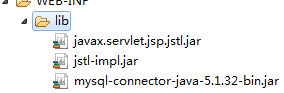


如若对应的MyEclipse版本没有上述的“MyEclipse Server Library”；可以选择“My Eclipse Library”然后选择ee的包。



选择对应配置的tomcat版本即可。

添加jstl jar包和mysql驱动包；



## 框架整合

### 添加struts2的jar包和配置文件

**添加jar包**：

commons-fileupload-1.3.1.jar，commons-io-2.2.jar，commons-lang-2.4.jar ，commons-lang3-3.2.jar，freemarker-2.3.19.jar，ognl-3.0.6.jar，struts2-core-2.x.jar

，struts2-spring-plugin-2.x.jar，xwork-core-2.x.jar 到web-inf/lib目录下。

**添加struts.xml**到src目录下。可在“struts-2.x\apps\struts2-blank\WEB-INF\classes”下复制。

在struts.xml中添加几个常用属性：

|  |
| --- |
| <!-- 禁用动态方法访问 -->  <constant name="struts.enable.DynamicMethodInvocation" value="false" />  <!-- 配置成开发模式 -->  <constant name="struts.devMode" value="true" />  <!-- 配置拓展名为action -->  <constant name="struts.action.extention" value="action" />  <!-- 把主题配置成simple -->  <constant name="struts.ui.theme" value="simple" /> |

**配置web.xml**：添加struts2 过滤器：

|  |
| --- |
| <filter>  <filter-name>struts2</filter-name> <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>struts2</filter-name>  <url-pattern>\*.action</url-pattern>  </filter-mapping> |

### 添加hibernate的jar包和配置文件

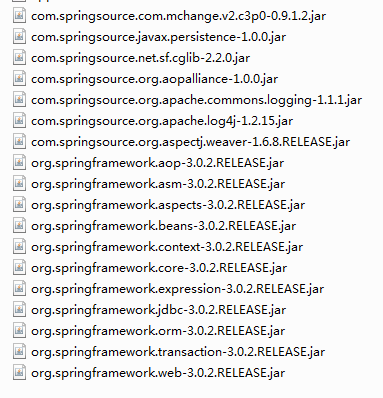
**添加hibernate jar包**：

hibernate3.jar，lib/required/\*.jar，lib\jpa\hibernate-jpa-2.0-api-1.0.0.Final.jar，lib\bytecode\cglib\cglib-2.2.jar到web-inf/lib目录下。

至于hibernate.cfg.xml文件，因项目使用spring来整合管理实体和数据库的连接等hibernate原本的工作，所以这个配置文件不再需要。

### 添加spring的jar包和配置文件

**添加spring3.0.2中的jar包**：



**添加spring配置文件**applicationContext.xml 到src目录下；

**在web.xml中注册spring监听器**，启动spring容器：

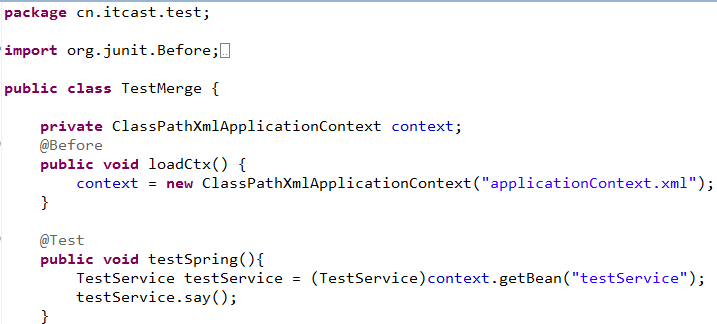
|  |
| --- |
| <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext.xml</param-value>  </context-param> |

## 整合测试项目

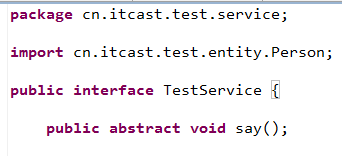
### 整合struts 和 spring

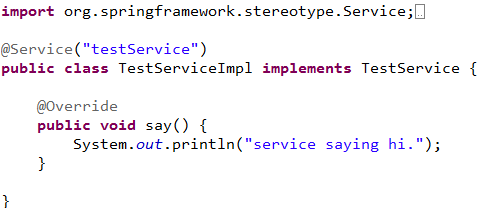
预期：如果可以在action中能够正确调用service里面的方法执行并返回到一个页面中；那么我们认定struts和spring的整合是成功的。

编写JUnit测试类，测试spring加载是否正确：

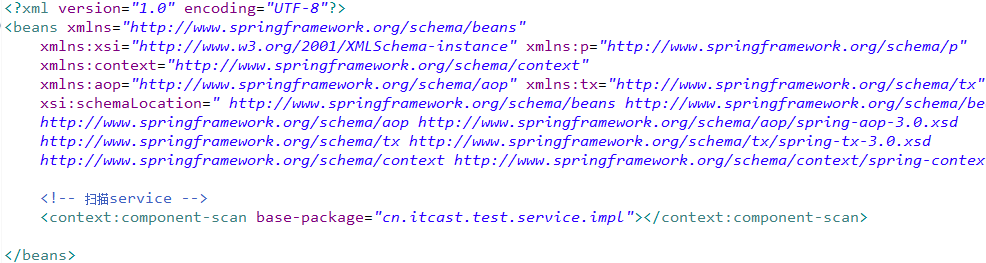


编写 TestService 接口 和实现类 TestServiceImpl





在applicationContext.xml中添加bean扫描配置信息；这边使用导入配置文件的方式配置。①首先在cn.itcast.test.conf中建立test-spring.xml，里面内容：



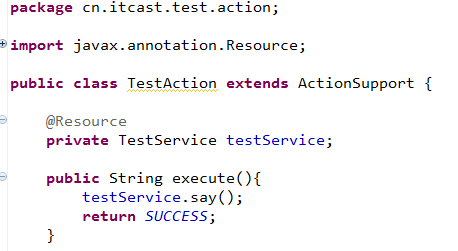
|  |
| --- |
| <!-- service -->  <context:component-scan base-package="cn.itcast.test.service.impl" /> |

里面的配置就是普通的bean扫描，只是将扫描范围缩小了。

②将test-spring.xml导入到applicationContext.xml中如下：

|  |
| --- |
| <import resource="classpath:cn/itcast/\*/conf/\*-spring.xml" /> |

编写TestAction类



在test的conf文件夹下新建test-struts.xml中配置TestAction ：

|  |
| --- |
| <struts>  <package name="test" namespace="/" extends="struts-default">  <action name="test\_\*" class="cn.itcast.test.action.TestAction" method="{1}">  <result name="success">/WEB-INF/jsp/test/test.jsp</result>  </action>  </package>  </struts> |

将test-struts.xml导入到struts.xml文件中。

|  |
| --- |
| <include file="cn/itcast/test/conf/test-struts.xml"/> |

在webRoot目录下新建test/test.jsp

在浏览器中输入：<http://localhost:8080/itcastTax/test.action> 查看后台是否能输入service中的打印信息。

### 整合hibernate 和 spring

在applicationContext.xml中配置如下原本在hibernate.cfg.xml中需要配置的信息，在spring中配置后hibernate.cfg.xml 可删除。

1. 配置c3p0数据库连接源：

|  |
| --- |
| <!-- 导入外部的properties配置文件 -->  <context:property-placeholder location="classpath:db.properties" />  <!-- 配置c3p0数据源 -->  <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource" destroy-method="close">  <property name="jdbcUrl" value="${jdbcUrl}"></property>  <property name="driverClass" value="${driverClass}"></property>  <property name="user" value="${user}"></property>  <property name="password" value="${password}"></property>  <!--初始化时获取三个连接，取值应在minPoolSize与maxPoolSize之间。Default: 3 -->  <property name="initialPoolSize" value="${initialPoolSize}"></property>  <!--连接池中保留的最小连接数。Default: 3 -->  <property name="minPoolSize" value="3"></property>  <!--连接池中保留的最大连接数。Default: 15 -->  <property name="maxPoolSize" value="${maxPoolSize}"></property>  <!--当连接池中的连接耗尽的时候c3p0一次同时获取的连接数。Default: 3 -->  <property name="acquireIncrement" value="3"></property>  <!--最大空闲时间，1800秒内未使用则连接被丢弃，若为0则永不丢弃。Default: 0 -->  <property name="maxIdleTime" value="1800"></property>  </bean> |

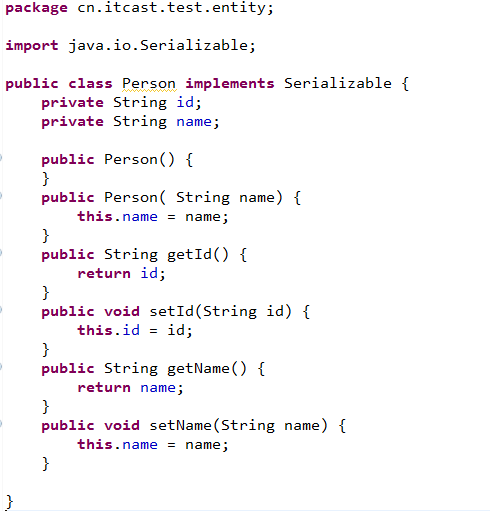
1. db.properties

|  |
| --- |
| jdbcUrl=jdbc:mysql://localhost:3306/itcastTax?useUnicode=true&characterEncoding=utf8  driverClass=com.mysql.jdbc.Driver  user=root  password=root  initialPoolSize=10  maxPoolSize=30 |

1. 配置sessionFactory，并将dataSource指向c3p0创建的dataSource：

|  |
| --- |
| <bean id="sessionFactory"  class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">  <property name="dataSource" ref="dataSource"></property>  <property name="hibernateProperties">  <props>  <prop key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>  <prop key="hibernate.show\_sql">true</prop>  <prop key="hibernate.hbm2ddl.auto">update</prop>  <prop key="javax.persistence.validation.mode">none</prop>  </props>  </property>  <property name="mappingLocations">  <list>  <value>classpath:cn/itcast/nsfw/\*/entity/\*.hbm.xml</value>  <value>classpath:cn/itcast/test/entity/\*.hbm.xml</value>  </list>  </property>  </bean> |
|  |

编写实体类Person和对应的映射文件Person.hbm.xml：

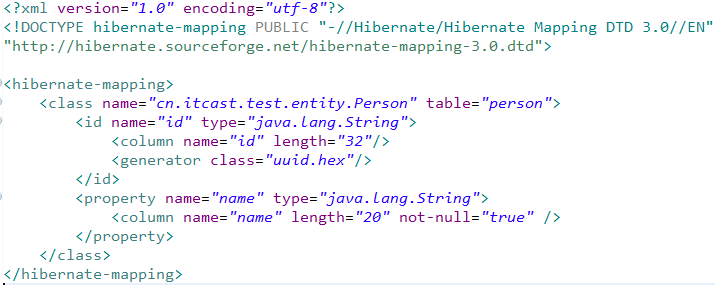


映射文件的头部信息：

<?xml version="1.0" encoding="utf-8"?>

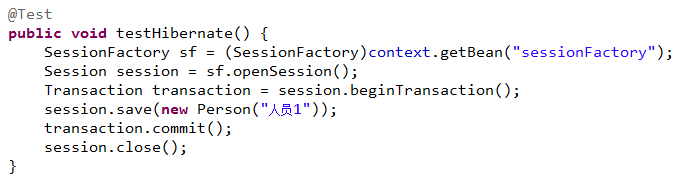
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">



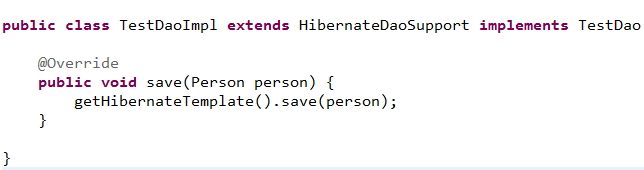
编写完实体映射文件后，用JUnit测试hibernate和spring的整合，在测试用例中启动spring容器的时候将扫描Person类根据其创建数据库表，并在测试时将向表插入一条数据。

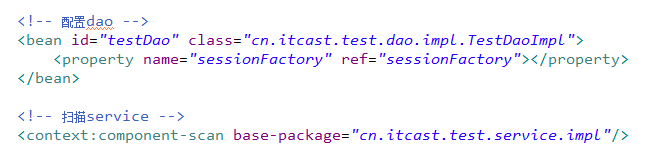
测试hibernate，添加一个人员

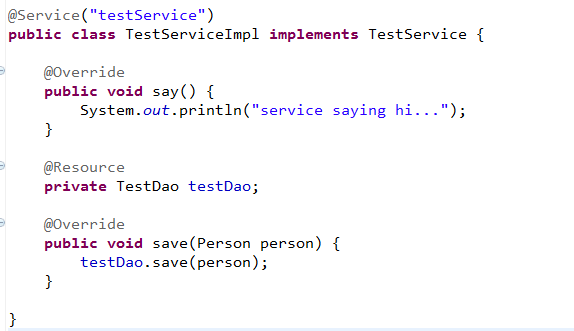


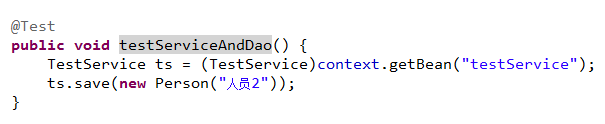
测试框架分层的整合(service 与 dao)

TestDao 中新增方法 save ，在TestService中通过调用testDao来保存人员信息。









### 配置spring事务管理

|  |
| --- |
| <!—事务管理-->  <bean id="txManager"  class="org.springframework.orm.hibernate3.HibernateTransactionManager">  <property name="sessionFactory" ref="sessionFactory" />  </bean>  <!—事务通知-->  <tx:advice id="txAdvice" transaction-manager="txManager">  <tx:attributes>  <tx:method name="find\*" read-only="true" />  <tx:method name="get\*" read-only="true" />  <tx:method name="load\*" read-only="true" />  <tx:method name="list\*" read-only="true" />  <tx:method name="search\*" read-only="true" />  <tx:method name="\*" rollback-for="Throwable" />  </tx:attributes>  </tx:advice>  <!—配置需要进行事务控制的类 -->  <aop:config>  <aop:pointcut id="serviceOperation" expression="bean(\*Service)" />  <aop:advisor advice-ref="txAdvice" pointcut-ref="serviceOperation" />  </aop:config> |

【注意：上面的pointcut expression 表示拦截以Service结尾的bean，或者可写成

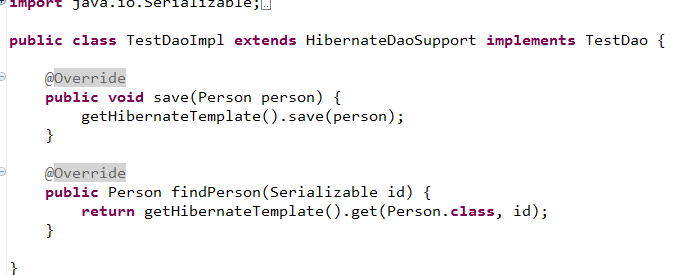
execution(\* cn.itcast..service.impl.\*.\*(..))】

完善 TestService接口和TestServiceImpl；利用service中的操作来验证上面配置的事务管理是否生效。

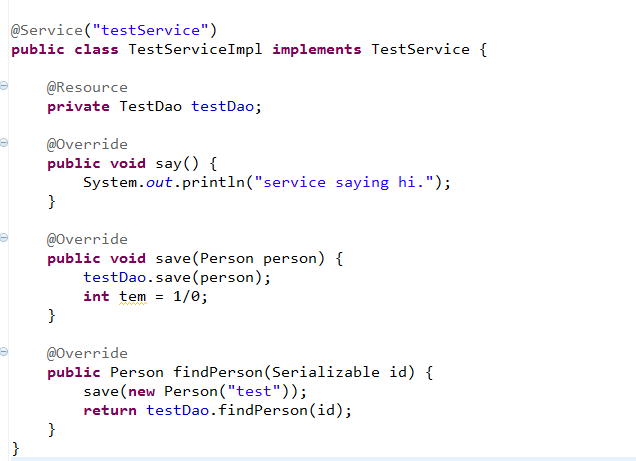
测试方法



Dao中

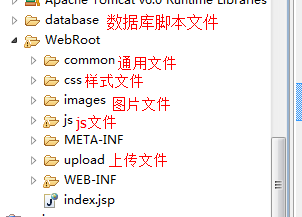


Service中



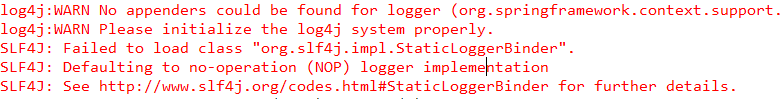
## 资源文件分类

1. 将配置文件归类到新建config文件夹；
2. 源代码目录按照功能模块进行划分：cn.itcast.子系统.功能模块.\*
3. Jsp放置到WEB-INF目录下；
4. 其它：



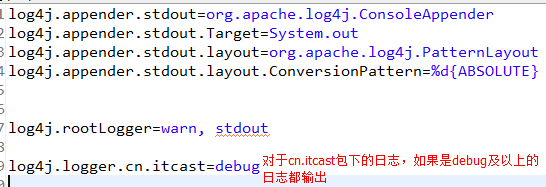
## 日志工具配置

在控制台会报出日志log4j没有配置好配置文件的信息。

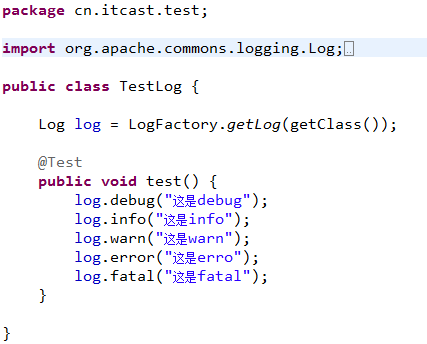


Slf4j 接口jar（slf4j-log4j12-1.6.1.jar）

log4j的jar包（com.springsource.org.apache.log4j-1.2.15），配置log4j.properties文件。

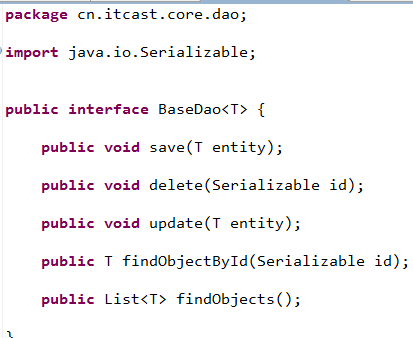


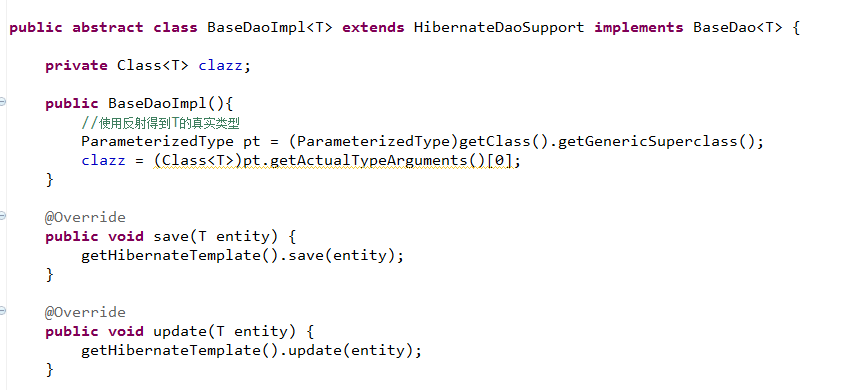
测试：



## 编写基类BaseDao

DAO基类中配备增删改查的操作。





获取泛型类型：

// 使用反射得到T的真实类型

ParameterizedType pt = (ParameterizedType) this.getClass().getGenericSuperclass(); // 获取当前new的对象的 泛型的父类 类型

this.clazz = (Class<T>) pt.getActualTypeArguments()[0]; // 获取第一个类型参数的真实类型

}

# 用户管理

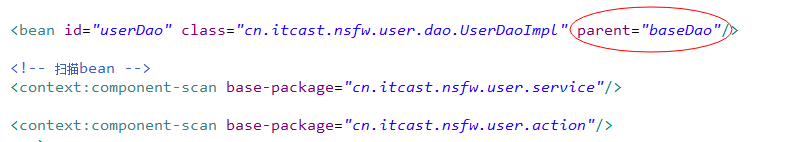
## 实现CRUD的步骤及用户实体映射文件

1. 用户实体类及映射文件：Usre.java 和 User.hbm.xml
2. 实现UserDao、UserDaoImpl，直接继承BaseDao、BaseDaoImpl；
3. 编写UserService、UserService类，里面包括baseDao中的基本方法；
4. UserAction中要实现增删改查，需要6个方法；我们约定需要跳转到页面的方法以UI结尾，如果addUI 则说明这是跳转到添加页面，需要对应一个addUI.jsp。在删除、修改、添加完成后应该返回列表页面，这时为了避免重复提交返回使用重定向。

配置user-spring.xml，并将其导入到applicationContext.xml中，然后在applicatoinContext.xml中设置所有业务dao的父类dao，为baseDao中的HibernateDaoSupport注入sessionFactory。

|  |
| --- |
| <!-- 所有dao的parent -->  <bean name="baseDao" abstract="true">  <property name="sessionFactory" ref="sessionFactory"/>  </bean> |

在user-spring.xml中的内容：



## 实现用户管理的CRUD方法及显示列表

在UserAction中至少需要有的6个方法：

listUI() --- 返回到列表页面对应listUI.jsp

addUI() --- 返回到添加页面对应addUI.jsp

add() --- 重定向到listUI

edit() --- 重定向到listUI

editUI() --- 返回到编辑页面对应editUI.jsp

delete() --- 删除用户完后重定向到listUI

设置完action后，配置user-struts.xml文件，里面对应各个方法对应的页面；最后将user-struts.xml包含到struts.xml文件中。

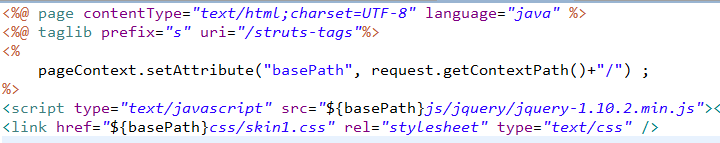
接下来引入美工设计好的页面；将数据与页面结合展示。

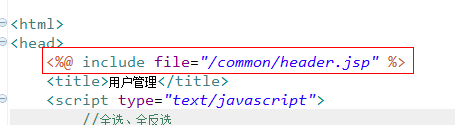
## 完成美工设计页面显示

将美工设计好的页面复制到项目中：

将js、css、image、分别放置到WebRoot目录下；其它的jsp对应复制。复制完成后针对jsp页面填充动态数据。

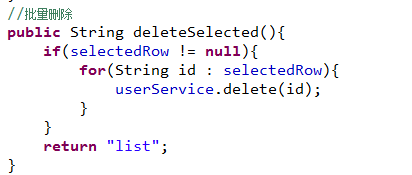
对应大多数页面样式和jquery都是需要的，所以我们把公共样式和jquery、struts标签库等信息抽出到一个header.jsp文件中，将此文件放入common公共文件目录。





## 批量删除

在用户列表listUI.jsp中设置好复选框<input type="checkbox" name="selectedRow" value='<s:property value="id"/>'/> ，点击操作栏中的删除执行js脚本提交用户列表表单到后台UserAction，在action中使用数组接收勾选的用户id，然后在对该数组进行遍历删除。



Js脚本：



## 日期组件wdatepicker

**导入**WdataPicker文件包到项目的js文件夹下；

**在用户管理中的添加、编辑jsp页面对生日表单项引入**日期组件：

|  |
| --- |
| **<script** type="text/javascript" src="${basePath}js/datepicker/WdatePicker.js">**</script>** |

**用法**：

|  |
| --- |
| <s:textfield id="birthday" name="user.birthday" readonly="true"  onfocus="WdatePicker({skin:'whyGreen', el:'birthday',dateFmt:'yyyy-MM-dd'})"/> |

回显时，注意设置好显示格式；可以如下设置：

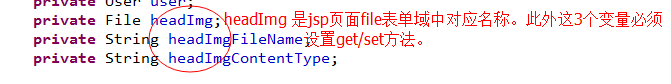
|  |
| --- |
| <s:textfield id="birthday" name="user.birthday" readonly="true"  onfocus="WdatePicker({skin:'whyGreen', el:'birthday',dateFmt:'yyyy-MM-dd'})">  <s:param name="value"><s:date name='user.birthday' format='yyyy-MM-dd' /></s:param>  </s:textfield> |

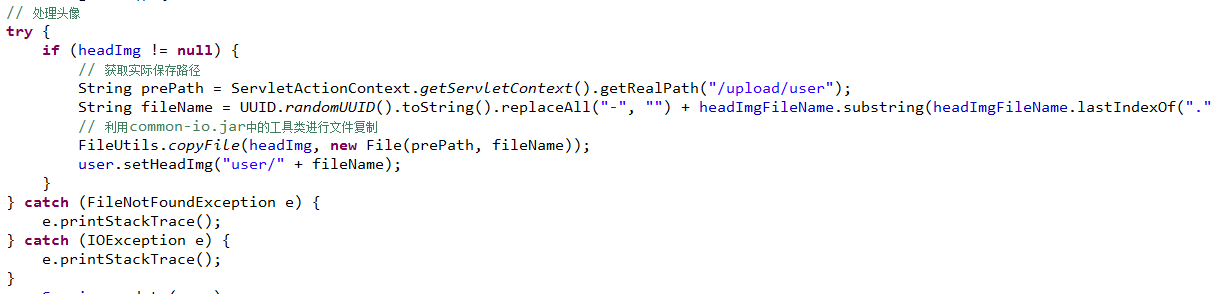
## 头像上传

在用户管理的添加、编辑页面中，可以上传用户的头像；页面中对头像字段引用type=file的文本域。

<input type="file" name="headImg"/>

之后UserAction接收文件需要配置的3个基本属性变量：



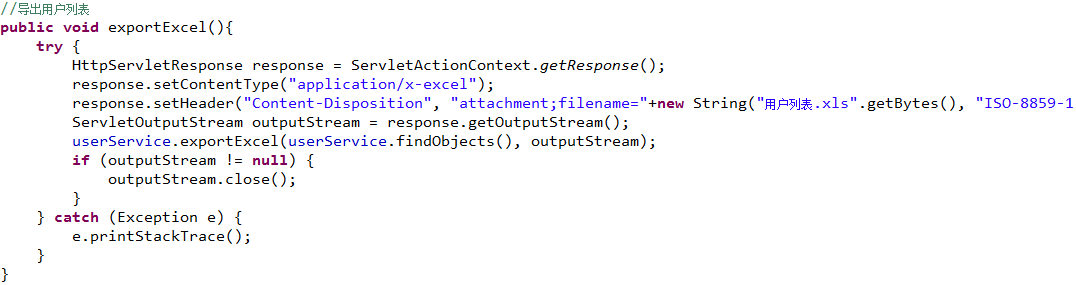


将头像文件保存在服务器的upload/user目录下，文件的名称以uuid重命名。（必须重命名：①重名问题②特殊字符在显示时的问题）

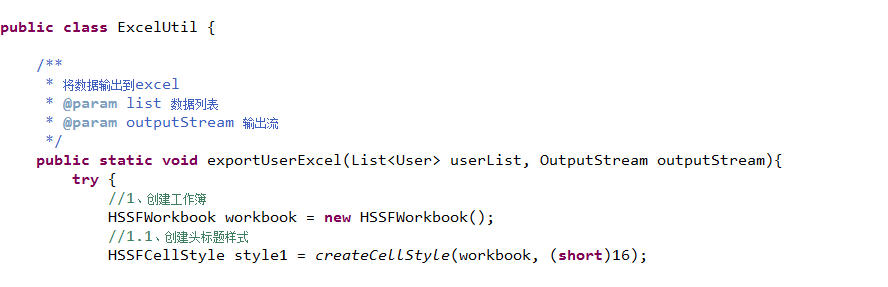
## 批量导入、导出

查阅《POI操作excel.doc》 先了解POI，使用POI工具类导出用户数据到excel中。

**导出：**在用户管理页面中点击“导出”，将所有用户记录导出到excel中。

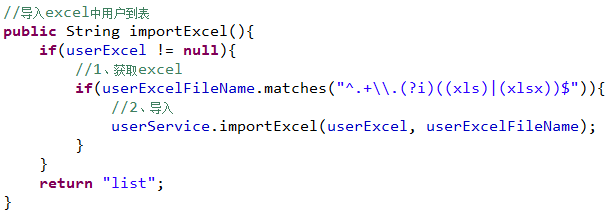


在userService中将调用导出工具类ExcelUtil导出数据。

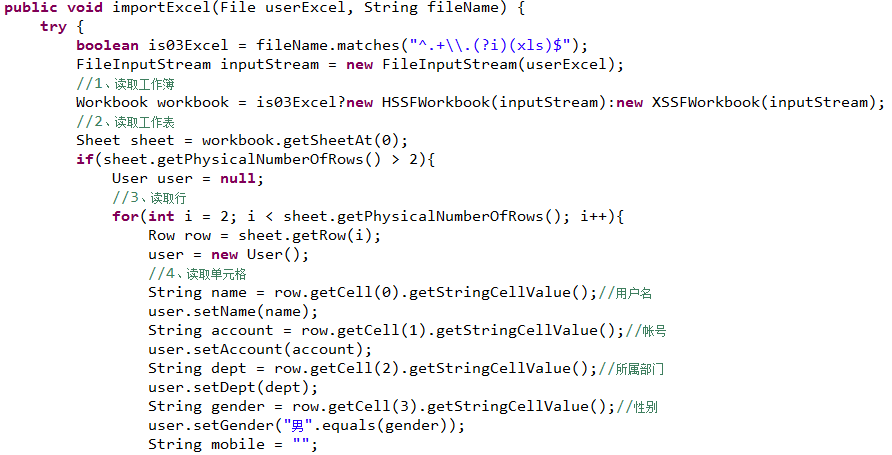


**导入：**在用户管理页面中，上传包含用户列表的excel文档，然后点击“导入”将excel中的用户记录插入的用户表中，并显示在列表上。

* 1. UserAction 中接受导入文件



* 1. UserServiceImpl 中具体处理导入文件



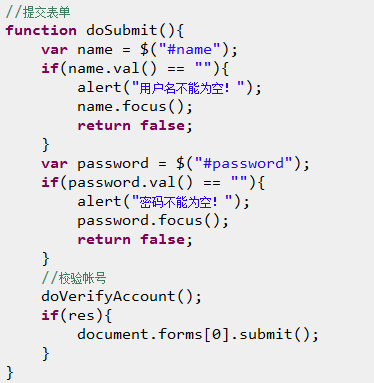


## 表单校验

在用户的新增、编辑中对帐号、密码进行非空校验和帐号的唯一性校验。需要注意的是在编辑页面中帐号的唯一性校验要排除当前编辑的帐号，不然将出现无法保存的情况。

1. 在jsp页面中，分别进行两个时机的校验，①是在编辑帐号的表单域时需要到服务器中校验是否已经存在输入的帐号；②在提交表单时再次进行校验，避免在填写表单时帐号已被使用的情况；

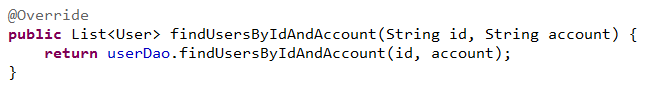


1. 在action中，根据传入的帐号和用户id到用户表中查找用户，如果出现有任何记录说明该帐号已经被使用。

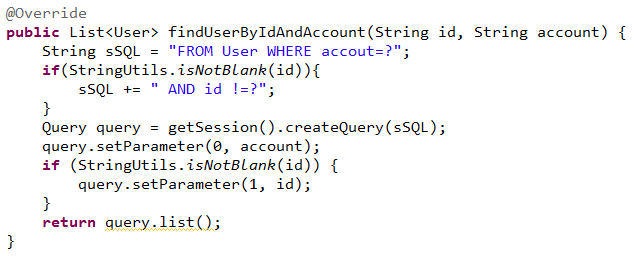


1. 在service、dao中的代码分别如下：

UserServiceImpl 中实现的方法：



UserDaoImpl中需要对传入的id进行判断实现的方法：



# 抽取BaseAction

在action中继承了ActionSupport和其它一些公共属性，如selectedRow等；可能以后还会产生更多公共的内容，所以应该把这些共有的抽取出来，放入到一个基本action中，我们命名为BaseAction，让它去继承ActionSupport和其它公共属性，其它的action只要继承它就可以了。

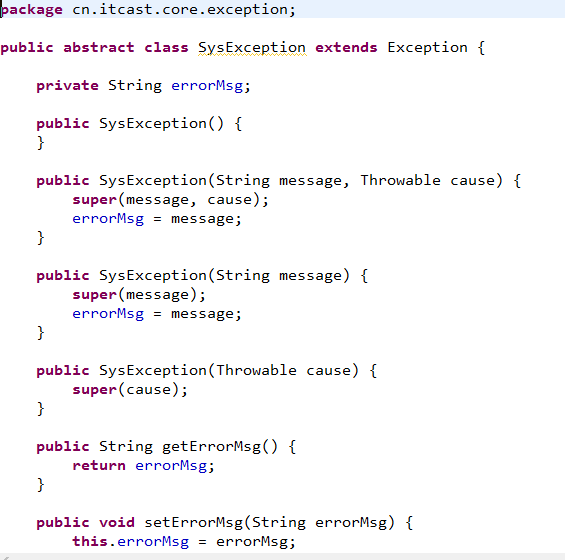
# 系统异常处理

当在页面中发送了请求，请求被系统处理时出现错误时，如果没有特别处理在请求过程中出现的异常信息时，那么返回给用户的则是后台的异常信息页面；为避免直接将这报错的页面呈现给用户，可对这报错进行处理。

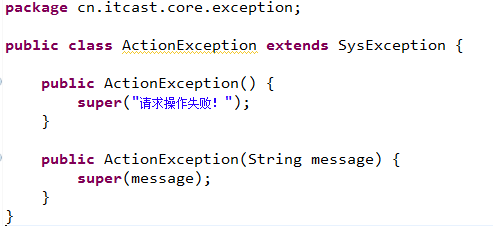
## 分层异常

在本系统中的后台处理有分3个层，分别是action、service和dao层，每个层次都有可能出现相应错误。① Action层可能出现解析请求参数、返回结果有问题；② Service 层则可能出现请求中要做的业务操作出现问题；出现了问题要根据实际情况判断是否会影响本次操作结果，action中要根据异常信息进行判断然后确定是否操作成功；③ dao层也可能出现在操作数据库时出现错误；而此种错误一般都是致命的会影响操作结果。以此；在3个层次中至少要有两种类型的异常信息来标识。

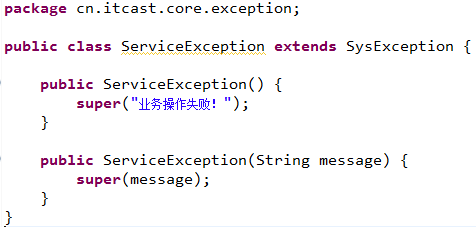
制定总的系统异常类：



Action层异常类：



Service层异常类：

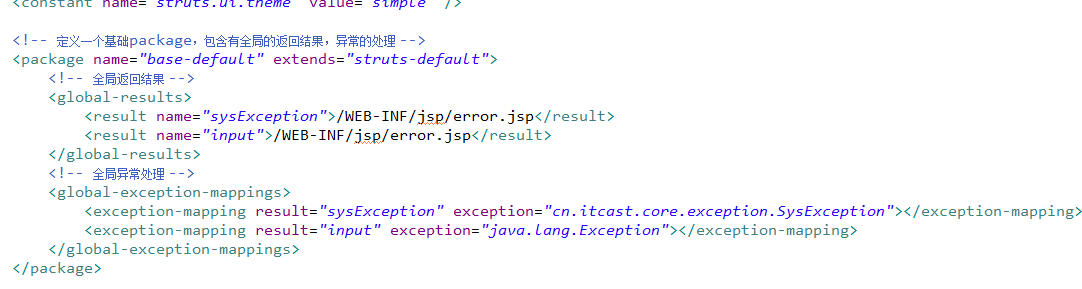


## 异常处理

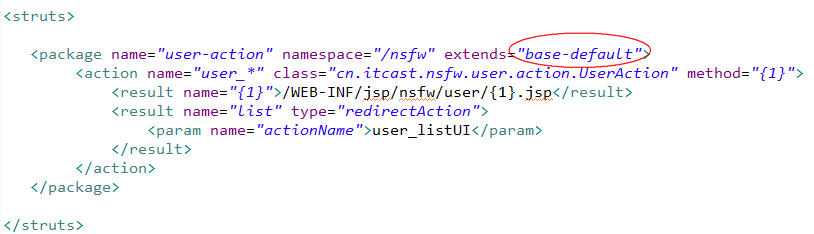
### 全局异常映射

在struts总配置文件（struts.xml）中配置一个包括 全局异常映射和全局结果的package，然后让其它所有的业务模块的struts配置文件（\*-struts.xml）继承后便可处理由后台抛出的特定异常并处理。

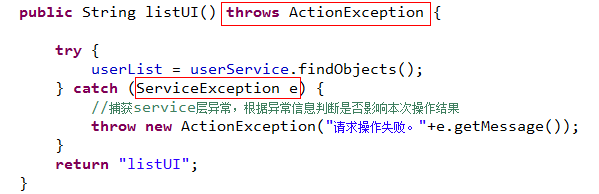
在总配置文件struts.xml中：



在子业务模块中的\*-sturts.xml中继承总配置文件中的package：



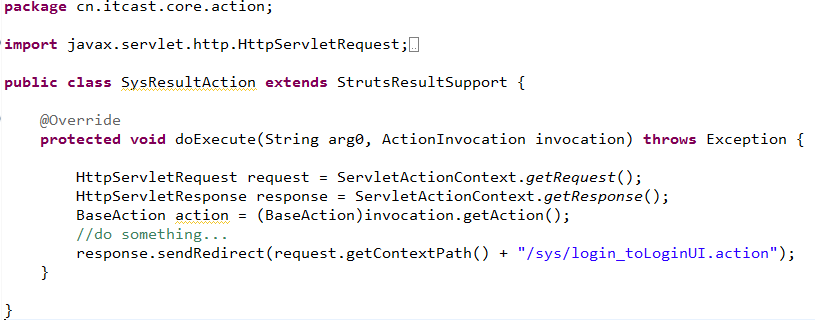
在具体的后台处理代码中抛出系统异常：



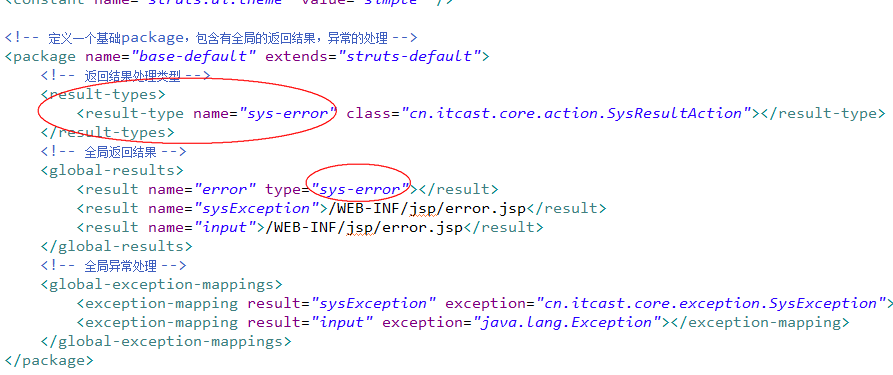
如果在action中只抛出了ActionException，那么非此异常出现时，如果在全局异常映射中配置了java.lang.Exception 异常类的映射，同样可以捕获。

### 制定返回类型（StrutsResultSupport）

在有特殊情况时；如果没有异常信息，但是有错误并且有错误信息等内容；此时也需要进行友好的错误处理的话，那么可以借助StrutsResultSupport 返回结果类型来实现特定处理。此种方式先需要继承StrutsResultSupport ，然后可以在子类中获取本次请求的相关信息，再根据相关信息进行结果处理：



在struts.xml总配置文件中定义全局处理类型和全局处理结果：



# 角色管理

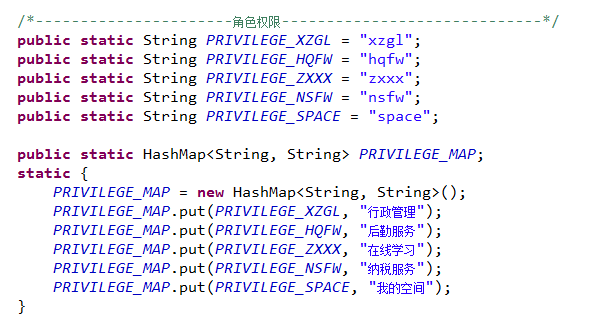
## 角色与权限说明

### 角色与权限的关系

系统中可以存在多个角色，每个角色可以自由的组合系统定义的权限集合。即角色和权限的关系是多对多的关系。为了保存这种多对多关系，需要一个角色权限表来保存。角色与角色权限的关系是一对多的关系；而权限与角色权限的关系也为一对多关系。

### 定义系统权限集

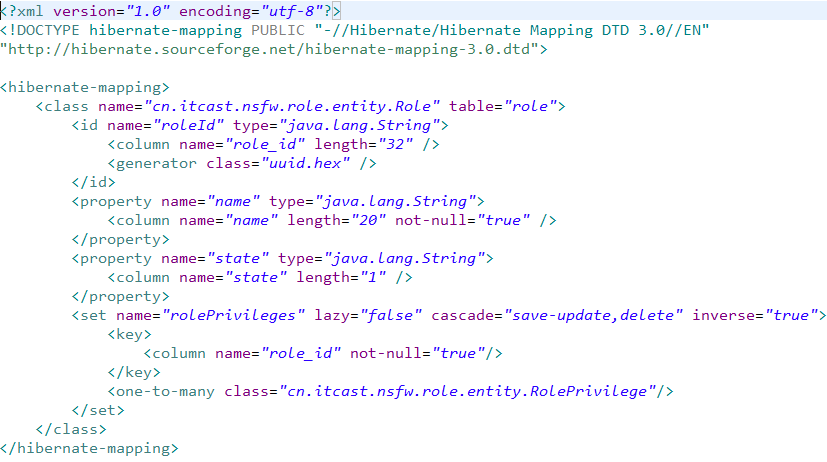
将系统中需要使用到的权限先定义出来：粗粒度的分为各个子系统的访问权限；这些权限可以定义在常量文件中。



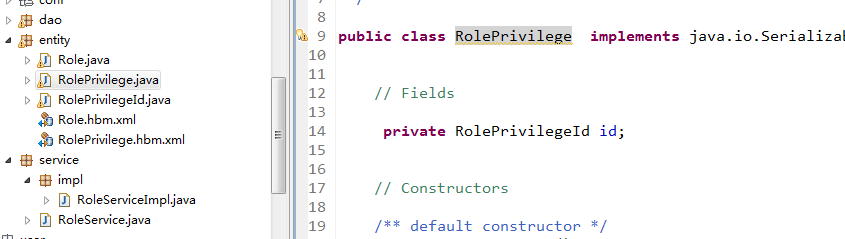
## 角色管理CRUD

### 实体类及映射文件

1. Role/Role.hbml.xml



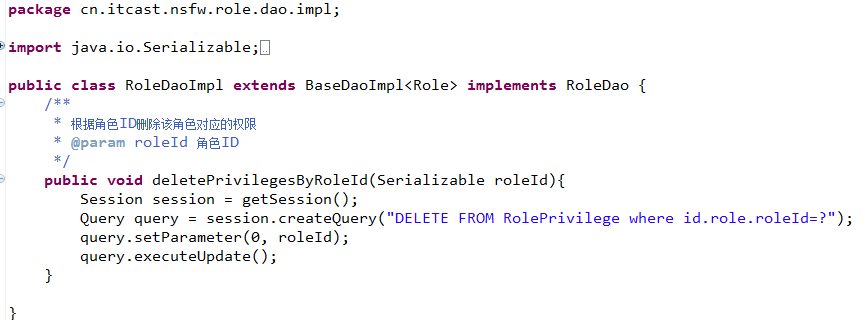
1. RolePrivilege/RolePrivilegeId，RolePrivilege.hbm.xml



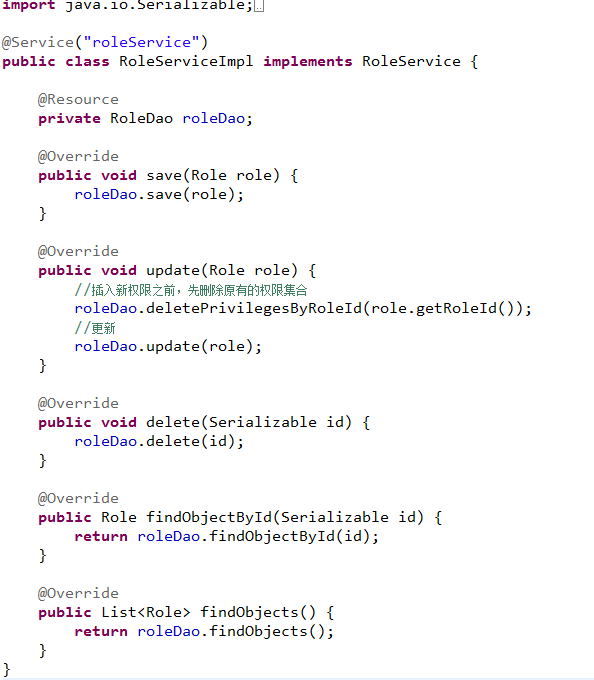


### dao、service层主要操作方法

dao中主要方法：



service中主要方法：



### action中主要方法





### 配置文件

配置role-spring.xml及role-struts.xml，并将role-struts.xml加入到struts.xml总配置文件中。

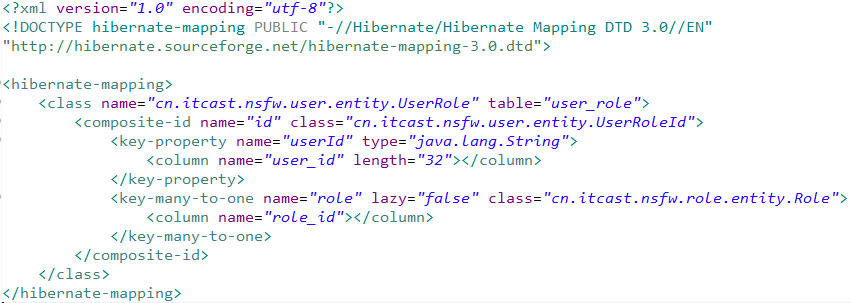
## 用户角色说明

一个用户可以对应多个角色，一个角色可以对应多个用户。用户与角色的关系也是多对多的关系。在页面中应该体现出在添加、编辑用户时可以选择多个角色；并且用户不直接关联系统的权限，用户的权限都是通过角色来关联实现。

## 改造用户管理

### 用户表实体类、映射文件的改造

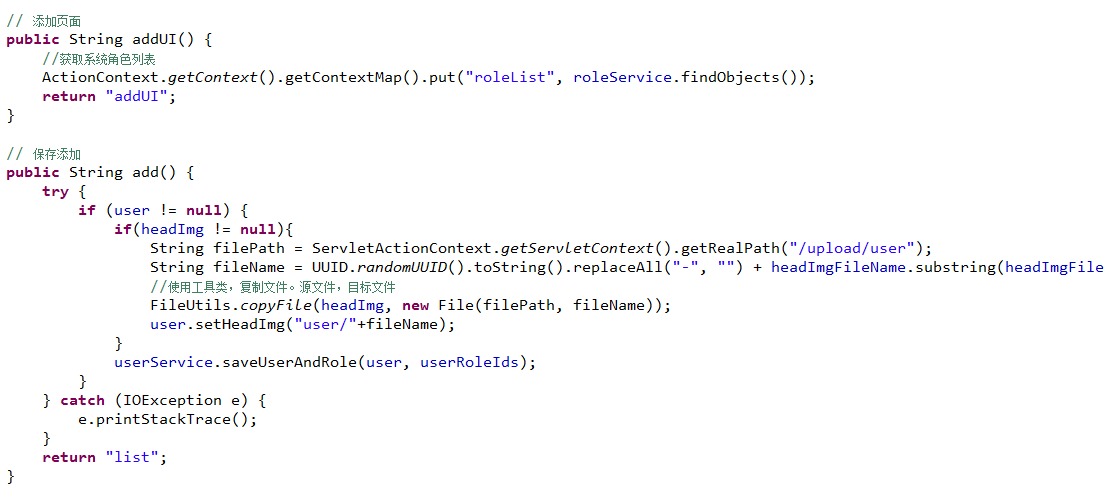
新增“用户角色”实体及映射文件：



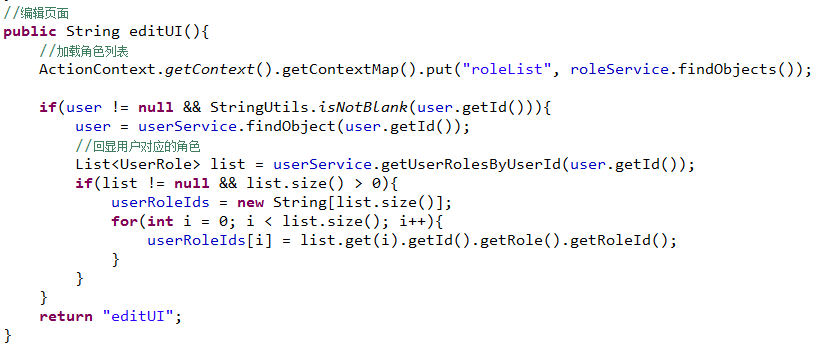
### 用户保存、更新、删除方法改造

1. UserAction：

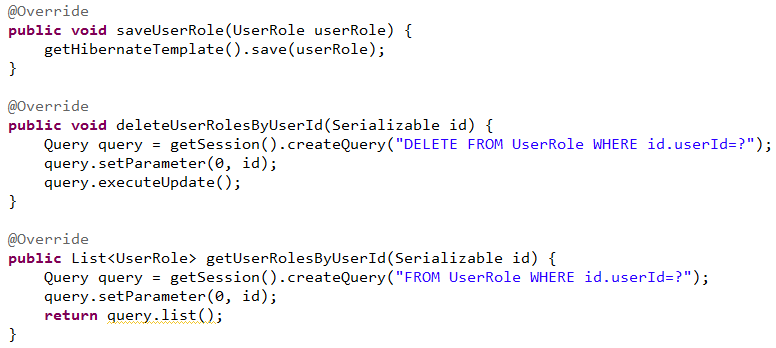
在跳转到addUI时应该初始化系统的所有角色列表提供添加页面进行选择：



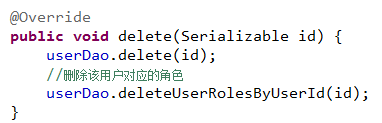
更新方法：

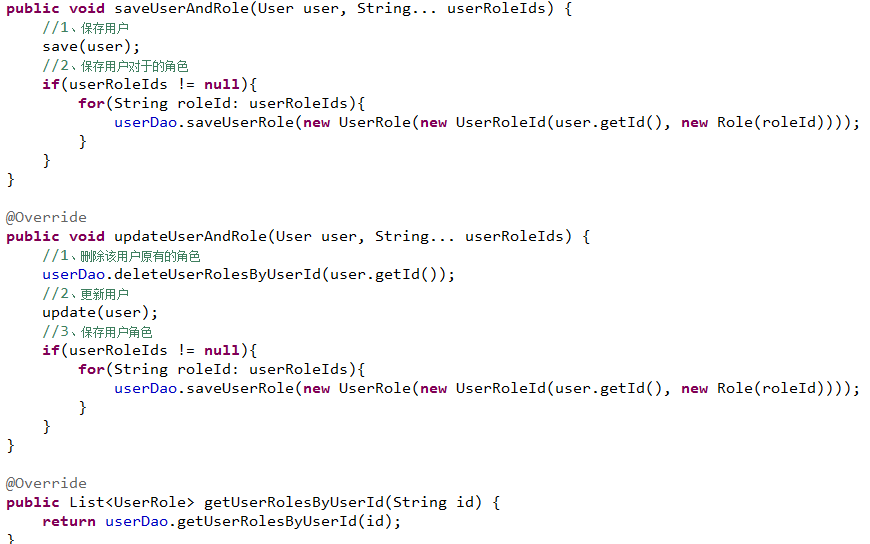


在UserDaoImpl中：



UserServiceImpl类中：





1. 在jsp页面中将角色选择改造为多选：



# 权限验证

## 系统首页

1. 编写首页cn.itcast.home.HomeAction

类的内容就一个跳转到首页的方法：

|  |
| --- |
| public class HomeAction extends ActionSupport {    //系统首页  public String execute(){  return "home";  }  } |

1. 配置home-struts.xml文件

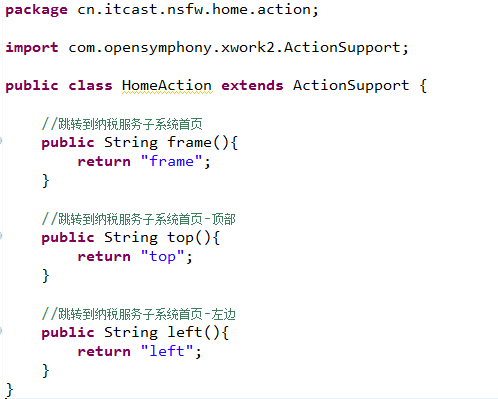
|  |
| --- |
| <struts>  <package name="home" namespace="/sys" extends="struts-default">  <action name="home\_\*" class="cn.itcast.home.action.HomeAction" method="{1}">  <result name="home">/WEB-INF/jsp/home/home.jsp</result>  <result name="{1}">/WEB-INF/jsp/home/{1}.jsp</result>  </action>  </package>  </struts> |

将home-struts.xml加入到struts.xml中

将home.jsp 加入到项目的WEB-INF/jsp/home目录下，修改进入纳税服务首页的链接。

## 纳税服务首页

新增纳税服务的首页对应的cn.itcast.nsfw.home.action.HomeAction，里面就只包含3个方法，分别定向frame.jsp、top.jsp、left.jsp的方法：



配置home-struts.xml 并包含到struts.xml中。

之后引入welcome.jsp、bg.jsp到common目录下，frame.jsp、top.jsp、left.jsp三个jsp文件到WEB-INF/jsp/nsfw目录下并修改这3个页面中的链接和相关信息。

## 登录与注销

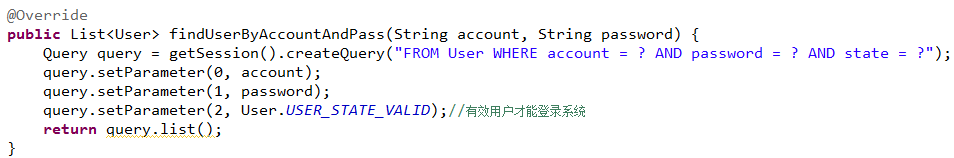
### 登录与注销

#### 编写LoginAction

在这个action中；主要处理登录和注销操作；方法应该包括：

1. 跳转到登录页；toLoginUI()——loginUI
2. 当没有权限时，跳转到没有权限的提示页面；toNoPermissionUI()——noPermissionUI
3. 登录方法；验证登录。需要具体的userService，userDao方法。

userDaoImpl类中具体的实现方法为：



1. 注销登录；logout()——tologinUI

LoginAction类：



配置login-struts.xml文件并引入struts.xml中。

#### 项目首页重定向到登录页

改写WebRoot下面的index.jsp，在里面直接重定向到登录页。

|  |
| --- |
| <%@ page language="java" import="java.util.\*" pageEncoding="utf-8"%>  <%  String path = request.getContextPath();  response.sendRedirect(path+"/sys/login\_toLoginUI.action");  %> |

## 权限验证说明

在系统中，① 需要验证用户的登录，只有登录后才能访问系统的资源。② 纳税服务子系统只有角色拥有“纳税服务”权限的才可以操作。

对于上述2种的权限验证，在此采用过滤器进行处理。编写LoginFilter过滤器，在过滤器中对用户访问的url进行登录验证，在登录的前提下再次验证如果访问的是纳税服务命名空间下的资源则利用权限验证接口PermissionCheck校验用户是否有“纳税服务”。

### 登录过滤器

编写过滤器cn.itcast.core.filter.LoginFilter 类；实现javax.servlet.Filter接口：



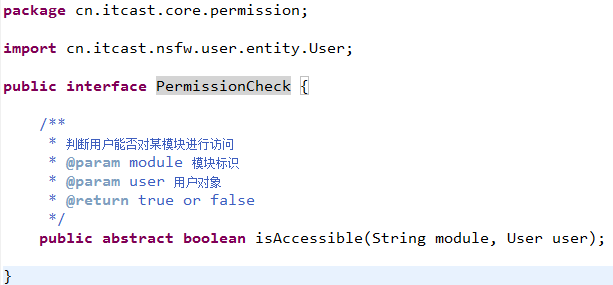
在web.xml中配置该过滤器：

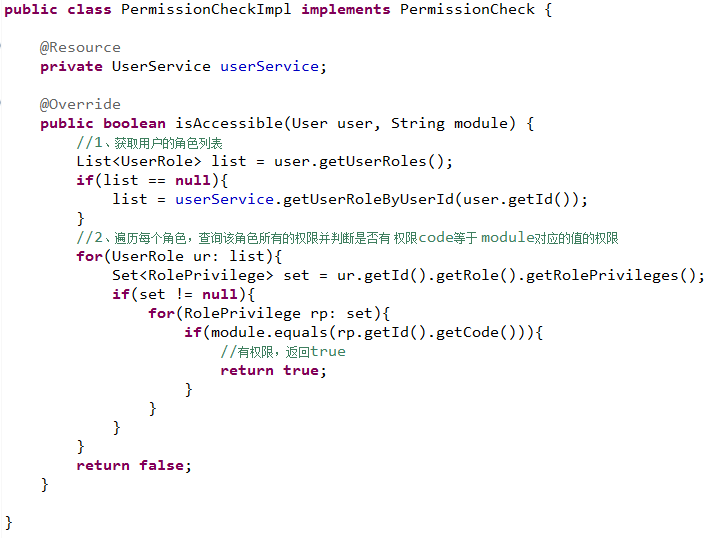
|  |
| --- |
| <filter>  <filter-name>loginFilter</filter-name>  <filter-class>cn.itcast.core.filter.LoginFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>loginFilter</filter-name>  <url-pattern>\*.action</url-pattern>  </filter-mapping> |

### 鉴定权限

在过滤器中加入权限的鉴定；当用户登录的情况下如果访问特定模块需要验证用户是否有特定权限。由于权限常用于在用户使用系统时权限的鉴定，所以可以在用户实体中加入一个用户权限的集合来保存用户的角色集合。新增用户类中角色列表属性userRoles用于存放用户的角色列表。

PermissionCheck/PermissionCheckImpl





在applicationContext.xml中需要注册PermissionCheck Bean，主要是为了方便在使用鉴定权限时可以注入其它Service。

## 解决细节-登录页面嵌套

在纳税服务的首页中使用了frameset，当前用户的系统登录信息失效后；如果再点击左边的菜单，那么在右边的显示登录页面，而正确的应该是整个页面返回到登录页。

解决这个问题：应该在跳转到登录页面中使用js脚本判断，是否当前页面在框架内，即当前页面的窗口是否是顶级窗口，如果是子窗口的话；可以直接刷新父窗口的地址则会自动地整个页面跳转为登录页。

|  |
| --- |
| <script type="text/javascript">  if(window != window.parent){  window.parent.location.reload(true);  }  </script> |

# 信息发布管理

## 实现信息发布管理CRUD

### 编写信息实体映射文件

编写信息实体类Info 及其映射文件 Info.hbm.xml

设置“信息分类”、“状态”的常量信息，其中信息分类需要设置一个静态MAP包括信息分类用于在页面的显示。

### 实现信息发布管理

1. 编写InfoDao/InfoDaoInfo
2. 编写InfoService/InfoServiceImpl
3. 编写 InfoAction
4. 编写info-struts.xml 和 info-spring.xml，并将info-struts.xml引入struts.xml
5. 引入美工页面并修改
6. 实现页面中添加、编辑、删除

## 富文本编辑器Ueditor

ueditor下载地址：

<http://ueditor.baidu.com/> 下载1.4.3 –utf8-Jsp版本。完整demo可参考下载文件中的index.html

**导入** ueditor 到项目中；将ueditor导入到项目的js目录下。导入ueditor/jsp/lib目录中的“commons-codec-1.9.jar”、“json.jar”、“ueditor-1.1.1.jar”这几个jar包到项目的web-inf/lib目录中。

**配置** ueditor 中图片上传前缀和路径；打开“ueditor/jsp/config.json”



**注意**：修改web.xml中struts过滤器的过滤规则，将/\*改为 \*.action 。避免引起struts过滤器把ueditor的图片等资源上传jsp给拦截而导致上传图片等失败。

|  |
| --- |
| <filter-mapping>  <filter-name>struts2</filter-name>  <url-pattern>\*.action</url-pattern>  </filter-mapping> |

**引入**ueditor到jsp页面，在ueditor的ueditor.config.js文件中，要求我们需要配置好ueditor的根目录地址；在我们页面引用时也需要在js中制定 UEDITOR\_HOME\_URL的路径。运用到页面时我们只需要将一个textarea的表单项的id和euditor实例化时的id一致即可。

将下面脚本内容引入到jsp页面中：

|  |
| --- |
| <script type="text/javascript" charset="utf-8" src="${basePath}js/ueditor/ueditor.config.js"></script>  <script type="text/javascript" charset="utf-8" src="${basePath}js/ueditor/ueditor.all.min.js"> </script>  <script type="text/javascript" charset="utf-8" src="${basePath}js/ueditor/lang/zh-cn/zh-cn.js"></script>  <script>  //配置ueditor的根路径  var UEDITOR\_HOME\_URL = "${basePath}js/ueditor/";  var ue = UE.getEditor('editor');  </script> |

|  |
| --- |
| <s:textarea id="editor" name="info.content" cssStyle="width:90%;height:160px;" /> |

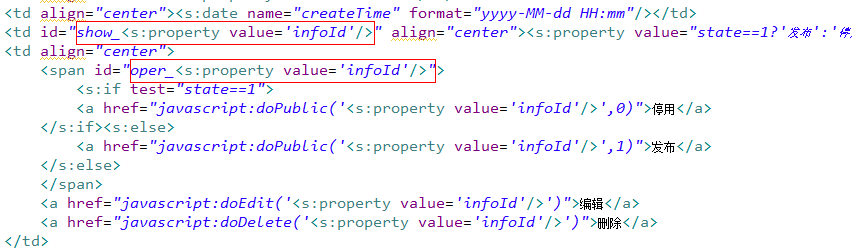
## 异步信息发布

在InfoAction中新增方法 publicInfo，主要用于修改信息的状态。



将新加的未发布的信息，在操作栏点击“发布”后，通过ajax将信息状态改为发布状态并更新列表中对应信息的状态。

在listUI.jsp中需要对状态列、和操作栏中的发布链接新增id来标识，当处理成功后根据id修改对应的值。



新增如下js方法。

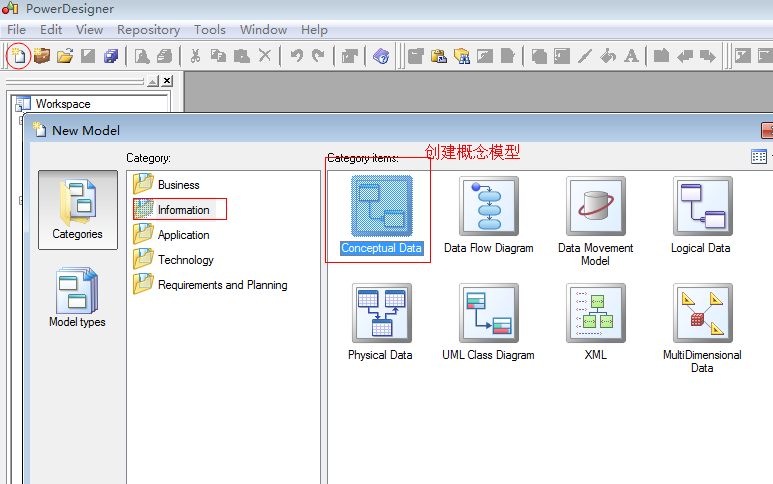


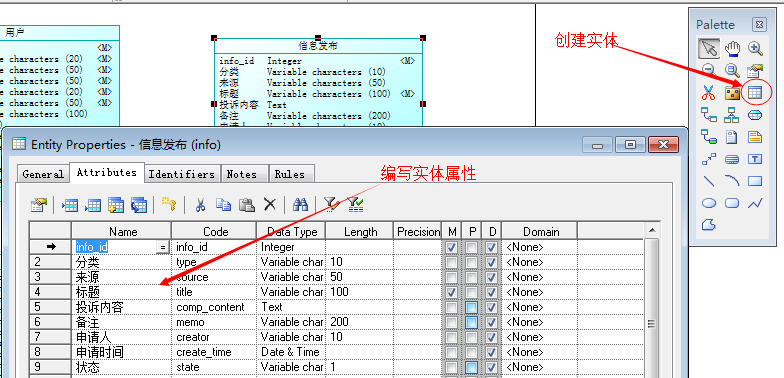
# hibernate逆向工程

## hibernate逆向工程生成实体

介绍一个模型设计工具PowerDesigner，这个是j2ee开发必要的一个工具。一般在开发中先使用PowerDesigner 创建实体关系图即概念模型。建立了概念模型后生成特定数据库类型的物理模型，然后根据物理模型的创建表脚本创建库表；最后使用MyEclipse的hibernate特性针对表逆向生成实体类和映射文件。这样免去了编写实体和映射文件及其实体关系的麻烦。

## 设计概念模型





relationship 一般的实体关系都可以使用，如果实体之间的关系指定为多对多，那么在生成pdm时则自动会生成一个以2个实体主键联合起来作为联合主键的一张新的关系表

inheritance 实体之间的继承关系，如：员工——领导

association 关系，如角色与人员存在多对多关系叫做：人员角色，这个关系里面还可能有是否、状态等。

association link 用于连接实体和关系

link/extended dependency 实体或关系的额外信息，信息比较独立；不会产生主外键关系

例如；设计一个人员组织架构：有机构、部门、员工、领导、角色、权限。

一个机构有多个部门

一个部门有多个员工

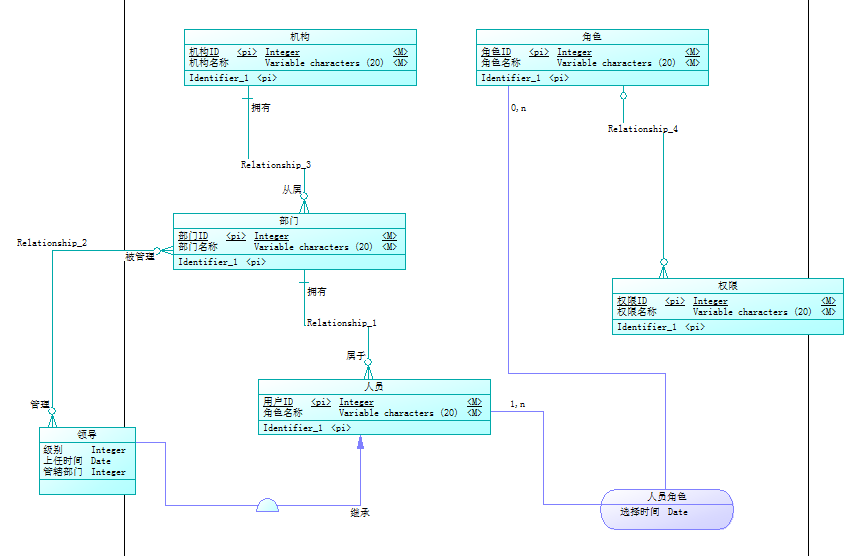
领导可以管理多个部门，同时领导他自己也是员工

一个员工可以有多个角色

一个角色可以分配给多个人

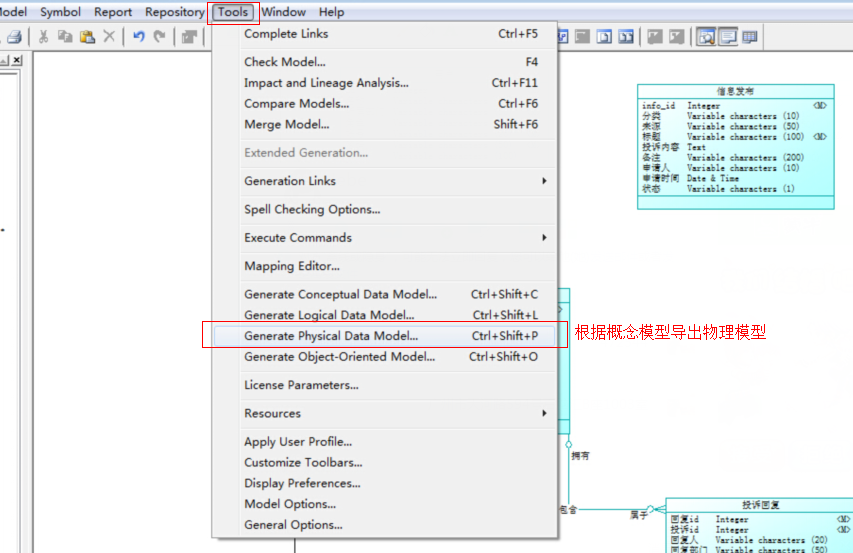
人员角色分配后可以设置是否有效，分配时间等

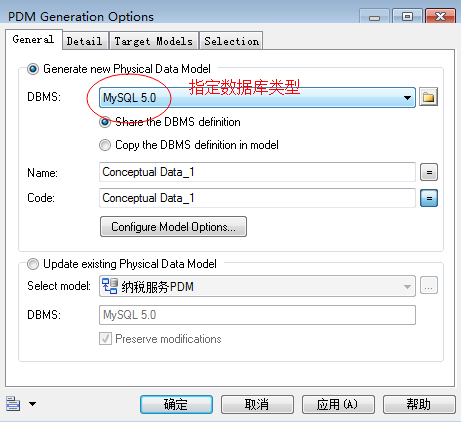
一个角色有多个权限

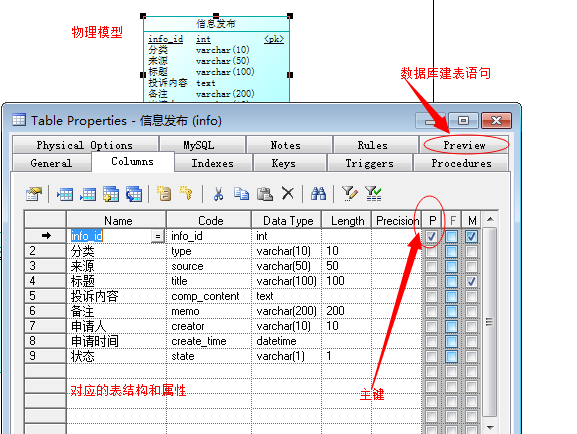


在概念模型阶段直接转成物理模型后，转换后的物理模型可以根据项目实际情况再调整相关信息。

## 导出物理模型

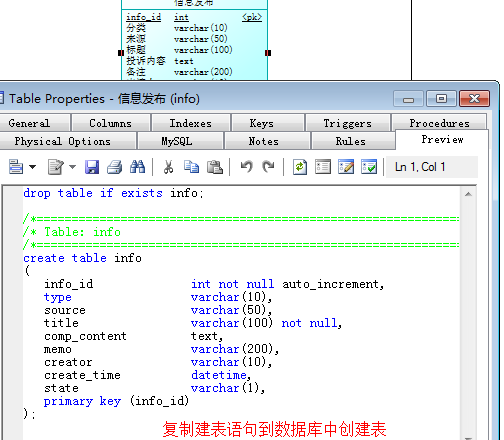






在物理模型指定或修复主外键关系，一般情况下只需要制定主键的生成策略，外键关系会从概念模型中保留下来。如果对其自动生成外键关系不满意可以在物理模型中修改。在物理模型中的任何修改都会体现在SQL语句中，所以建表时直接复制其语句即可。

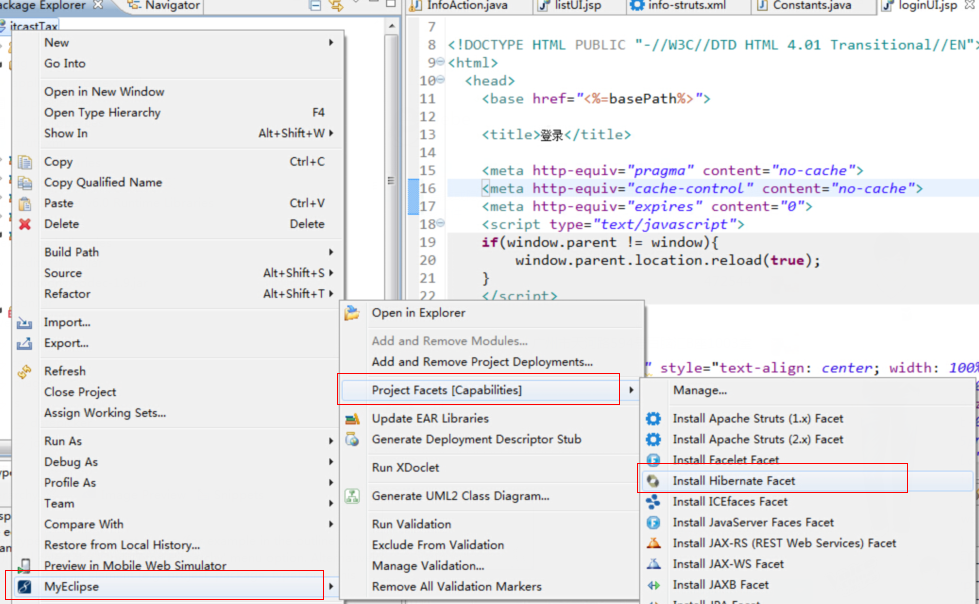
## 物理模型 –生成库表

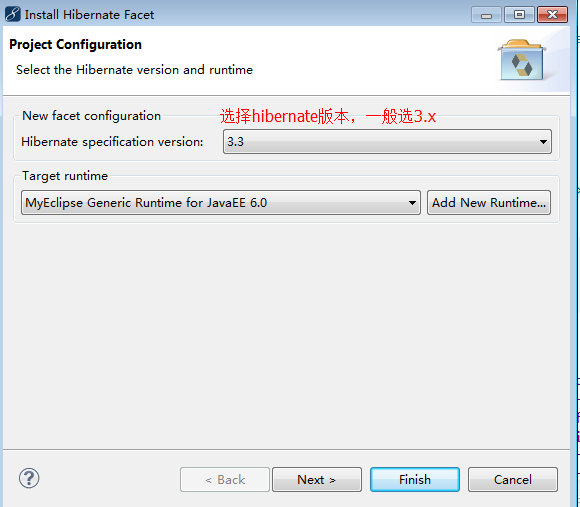


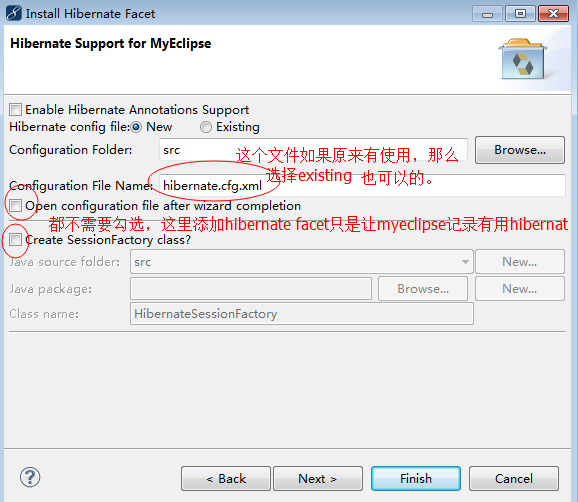
## 逆向生成实体及映射文件

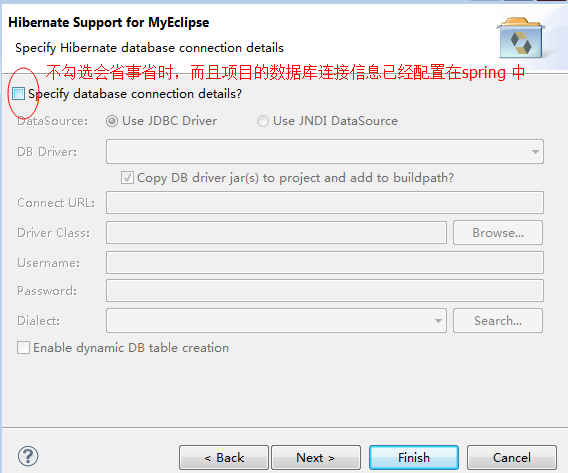
1. 添加hibernate facet到项目

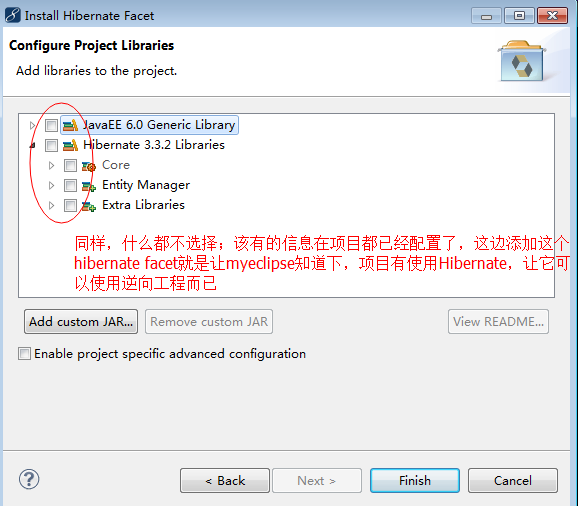
右击项目——MyEclipse——Project Facets[compabilities]——install Hibernate facet）



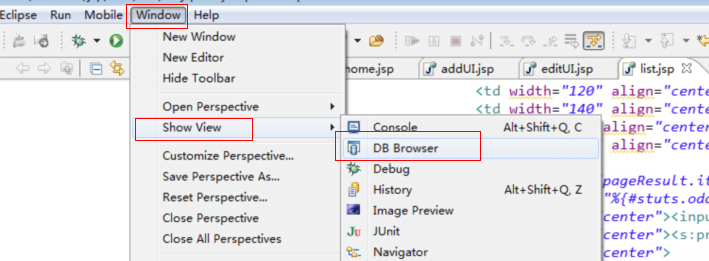


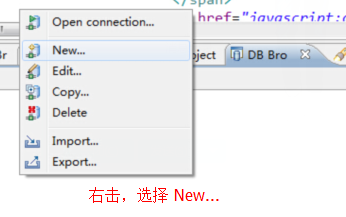


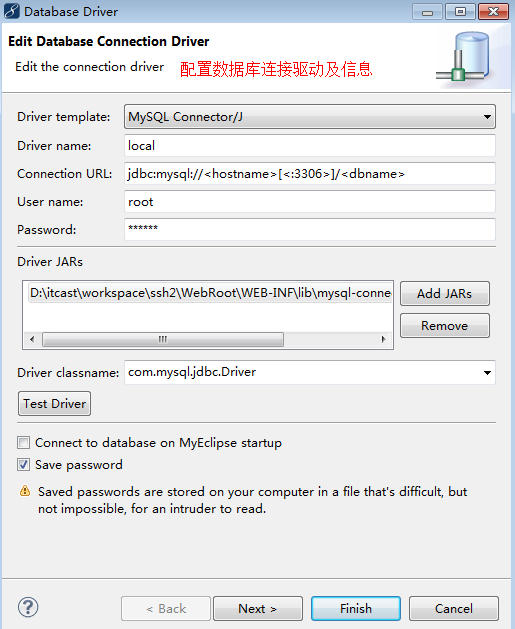


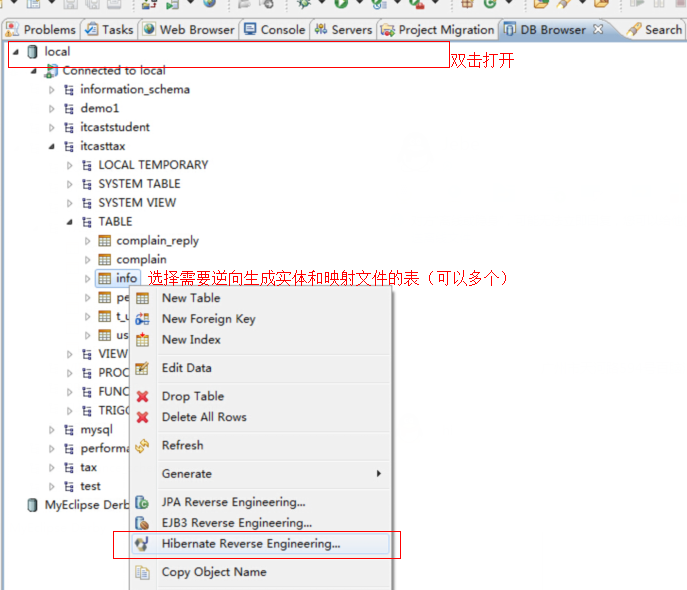


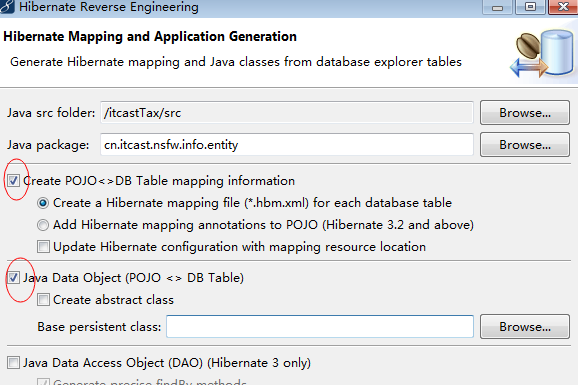
1. 在MyEclipse中打开DB Browser视图：

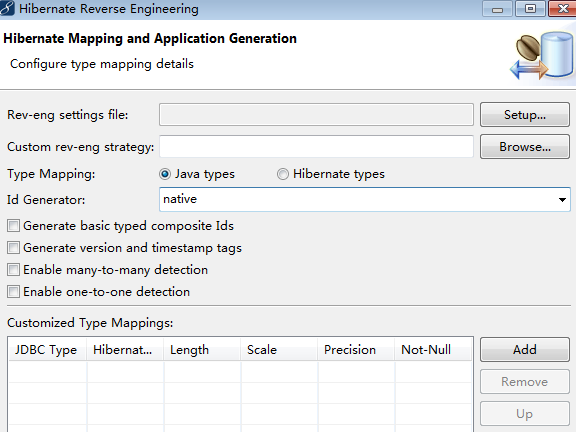




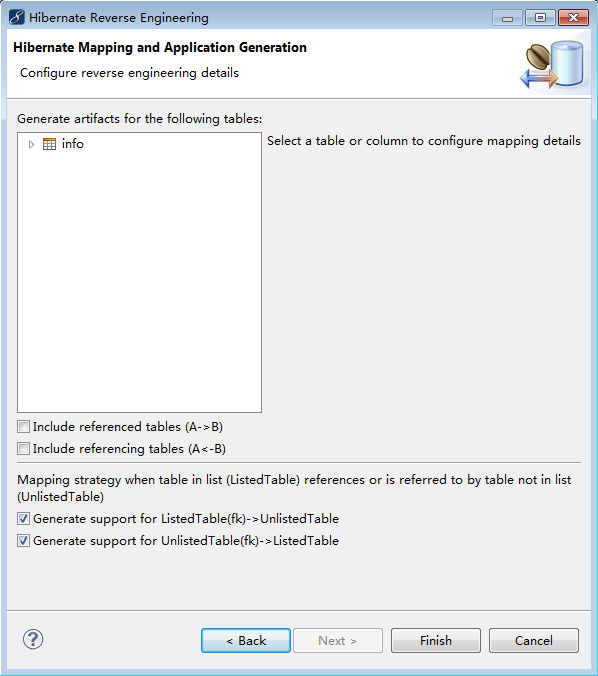








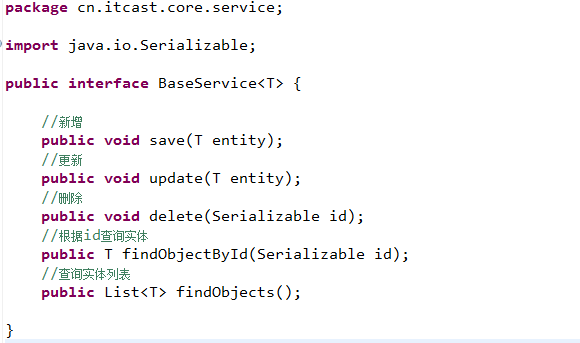
在 Id Generator 中，如果是mysql 中需要自增的可选择native，如果是uuid的可选择uuid.hex或uuid



完成。在图中设置的对应的目录下则会生成实体和映射文件。

# 抽取BaseService

由于在每个业务service中都存放了同样的基类方法，如增删改查；应写一个BaseService来实现这些基础方法并用BaseServiceImpl来实现其操作。这样具体业务Service可以直接继承这些内容。在BaseServiceImpl中只需要操作基础方法，所以在与数据库的操作交互中，只要能够获取到BaseDao对象即可。这个BaseDao可以从具体的业务ServiceImpl 中传递自身业务Dao给BaseServiceImpl便可。



BaseServiceImp类：



子Service类；可如下实现对BaseServiceImpl类中的BaseDao对象的实例化：

