# 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

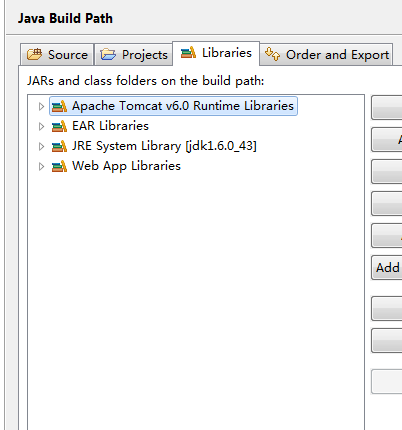
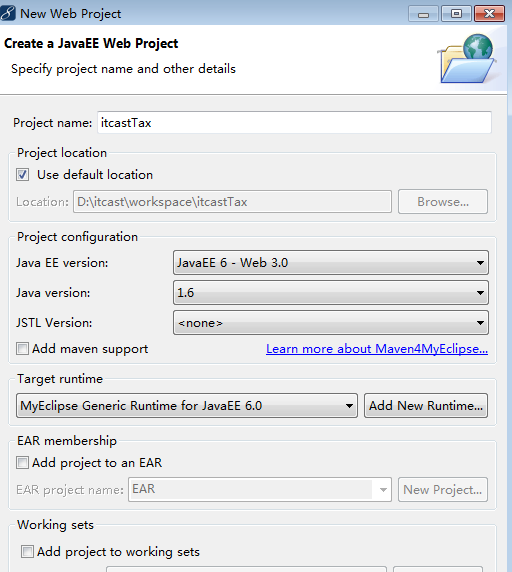
### 创建itcastTax数据库

-- 创建数据库

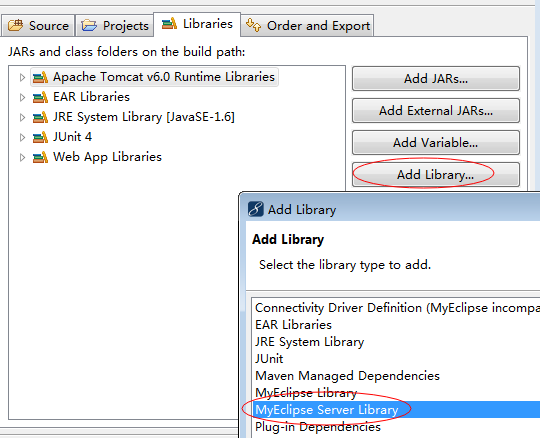
CREATE DATABASE itcastTax DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;

### 新建web项目

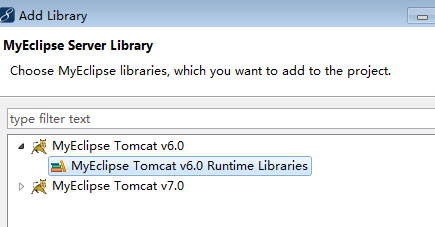
新建工作空间指定项目编码（或工作空间编码）为utf-8，再建 web project，配置buildpath



引入tomcat 的包：

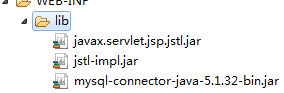


如若对应的MyEclipse版本没有上述的“MyEclipse Server Library”；可以选择“My Eclipse Library”然后选择ee的包。



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

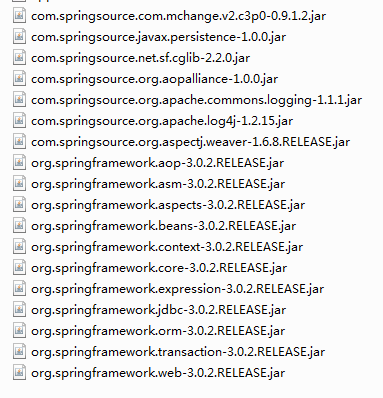
**添加hibernate jar包**：

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：

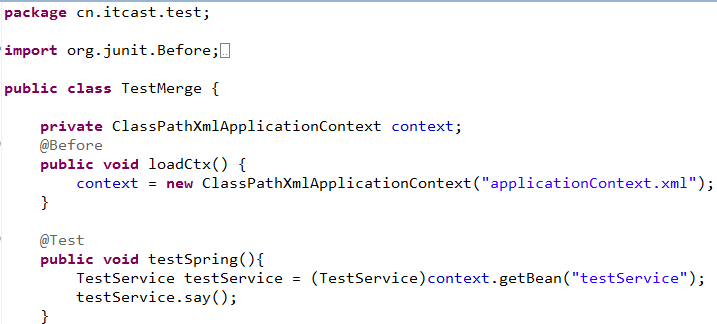
|  |
| --- |
| <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext.xml</param-value>  </context-param> |

## 整合测试项目

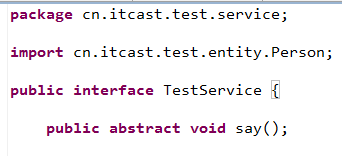
### 整合struts 和 spring

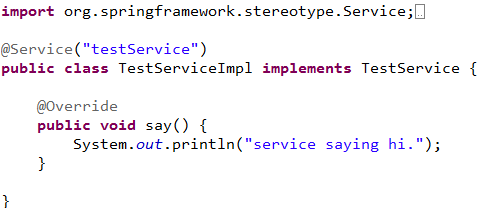
预期：如果可以在action中能够正确调用service里面的方法执行并返回到一个页面中；那么我们认定struts和spring的整合是成功的。

编写JUnit测试类，测试spring加载是否正确：

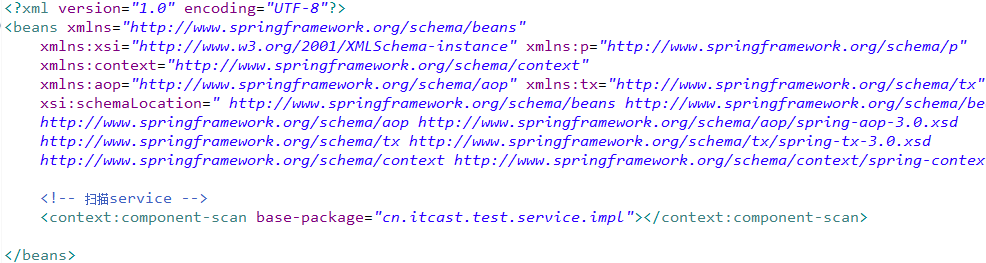


编写 TestService 接口 和实现类 TestServiceImpl





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：



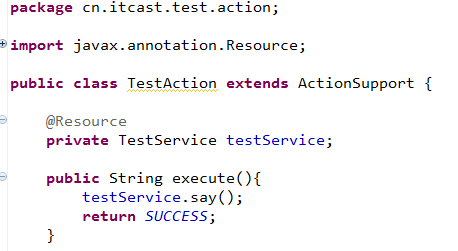
|  |
| --- |
| <!-- service -->  <context:component-scan base-package="cn.itcast.test.service.impl" /> |

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> |

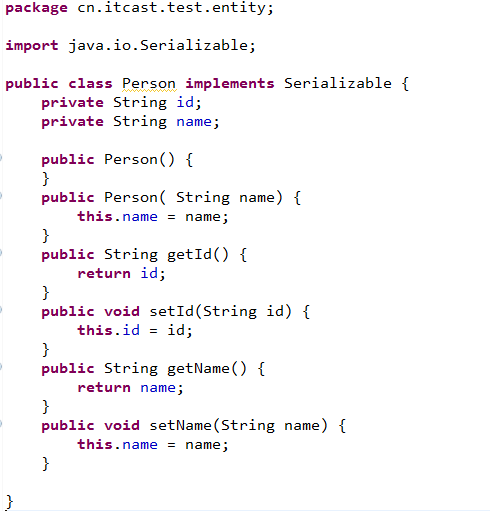
1. db.properties

|  |
| --- |
| 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：

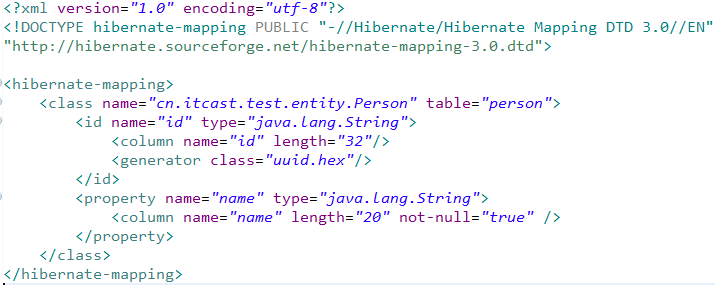


映射文件的头部信息：

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

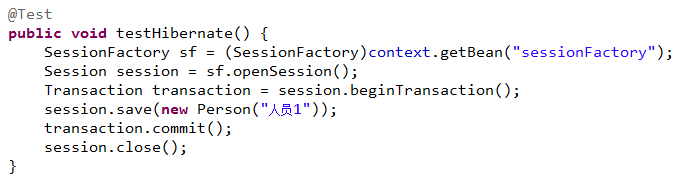
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"

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



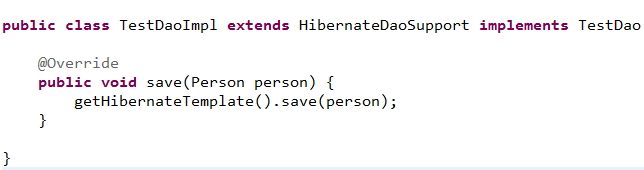
编写完实体映射文件后，用JUnit测试hibernate和spring的整合，在测试用例中启动spring容器的时候将扫描Person类根据其创建数据库表，并在测试时将向表插入一条数据。

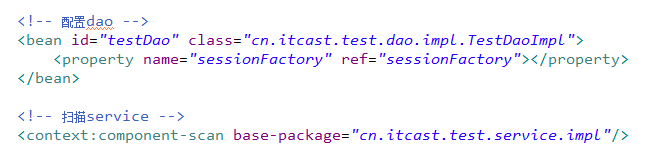
测试hibernate，添加一个人员

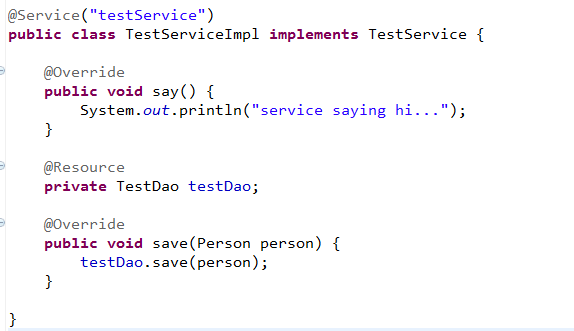


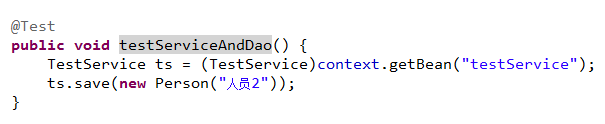
测试框架分层的整合(service 与 dao)

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

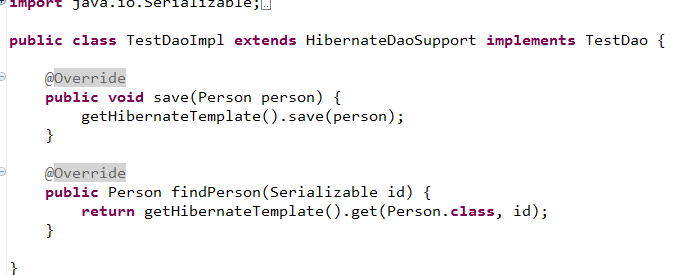
execution(\* cn.itcast..service.impl.\*.\*(..))】

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。

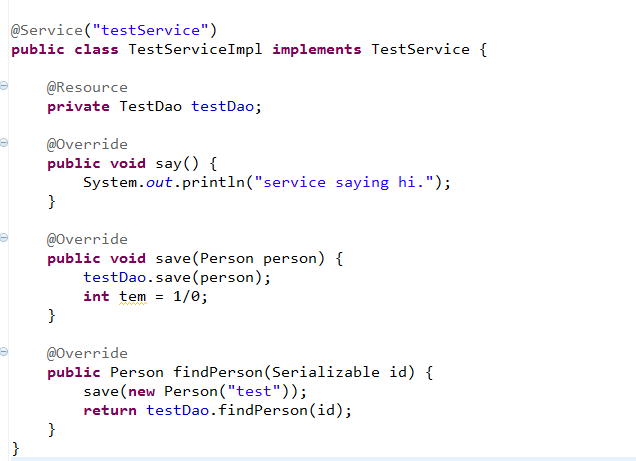
测试方法



Dao中

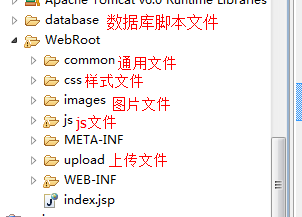


Service中



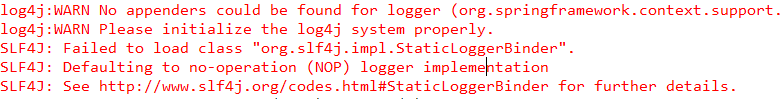
## 资源文件分类

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：



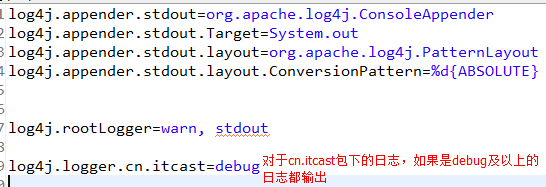
## 日志工具配置

在控制台会报出日志log4j没有配置好配置文件的信息。

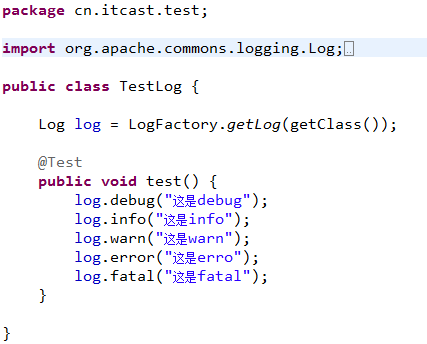


Slf4j 接口jar（slf4j-log4j12-1.6.1.jar）

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

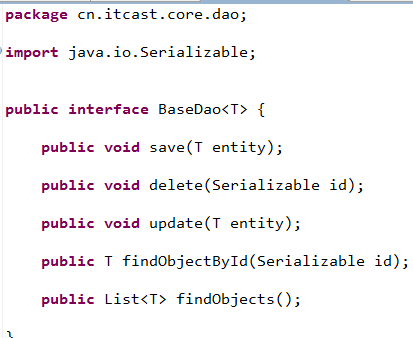


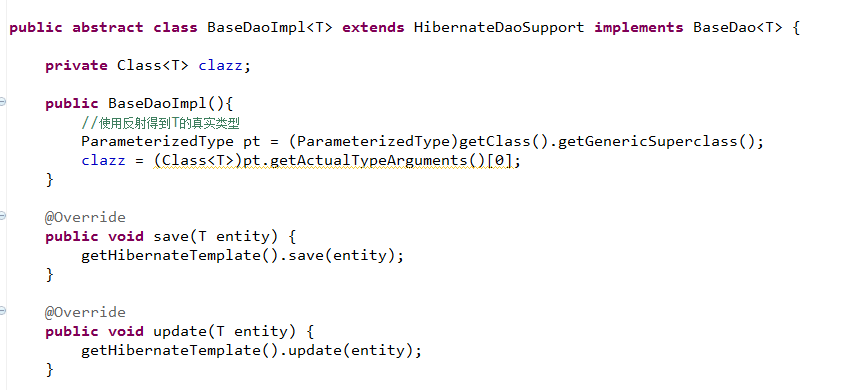
测试：



## 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

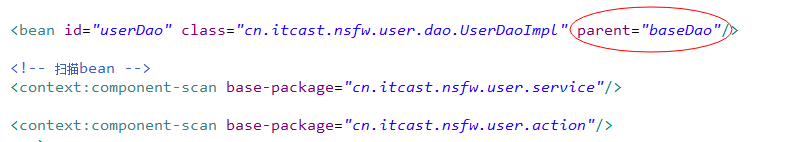
## 实现CRUD的步骤及用户实体映射文件

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。

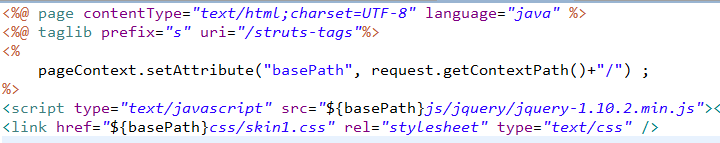
接下来引入美工设计好的页面；将数据与页面结合展示。

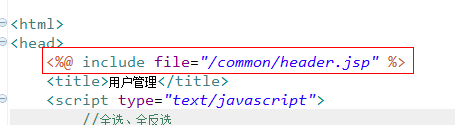
## 完成美工设计页面显示

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。

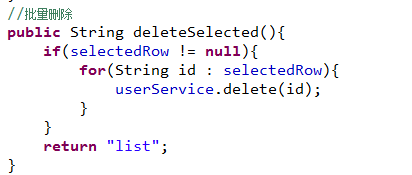
对应大多数页面样式和jquery都是需要的，所以我们把公共样式和jquery、struts标签库等信息抽出到一个header.jsp文件中，将此文件放入common公共文件目录。





## 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：

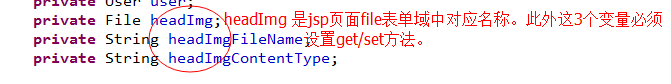
|  |
| --- |
| <s:textfield id="birthday" name="user.birthday" readonly="true"  onfocus="WdatePicker({skin:'whyGreen', el:'birthday',dateFmt:'yyyy-MM-dd'})">  <s:param name="value"><s:date name='user.birthday' format='yyyy-MM-dd' /></s:param>  </s:textfield> |

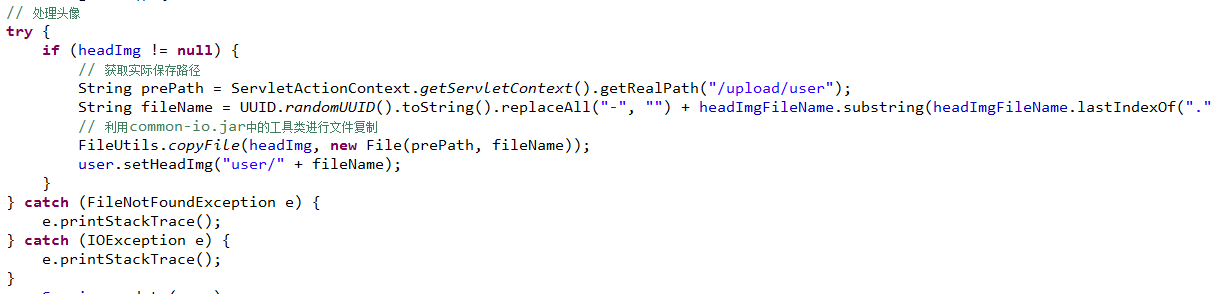
## 头像上传

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：



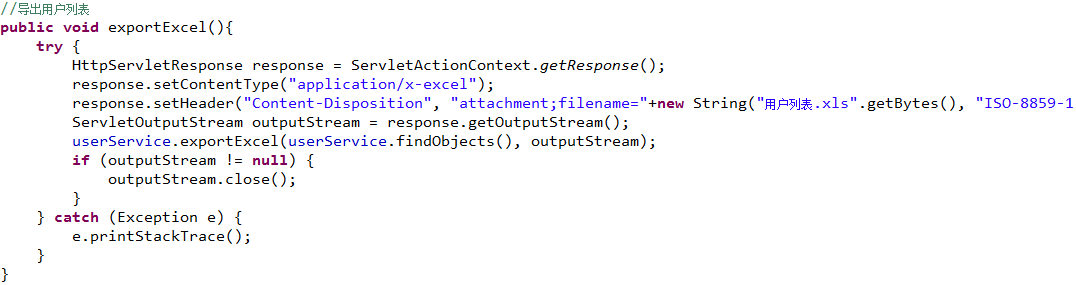


将头像文件保存在服务器的upload/user目录下，文件的名称以uuid重命名。（必须重命名：①重名问题②特殊字符在显示时的问题）

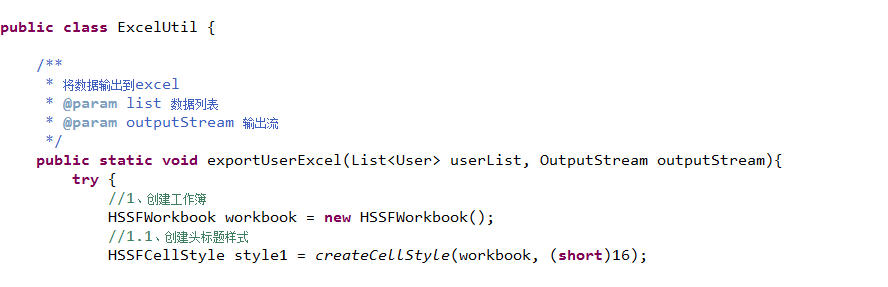
## 批量导入、导出

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。

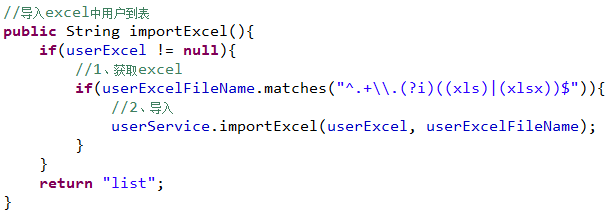


在userService中将调用导出工具类ExcelUtil导出数据。



**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

