scoreManagement说明

1. 总体设计

## 1.1 数据库设计

该数据库中一共有两张表，一张是学生成绩表，一张是班级表。班级表(t\_class)包括：班级编号(classNo)、班级名(className)。其中班级编号为主键。学生成绩表(t\_score)包括：学号(stuNo)、姓名(stuName)、平时成绩(dailyScore)、考试成绩(examScore)、班级编号(clsNo)。其中学号为主键，班级编号为外键。如下表：

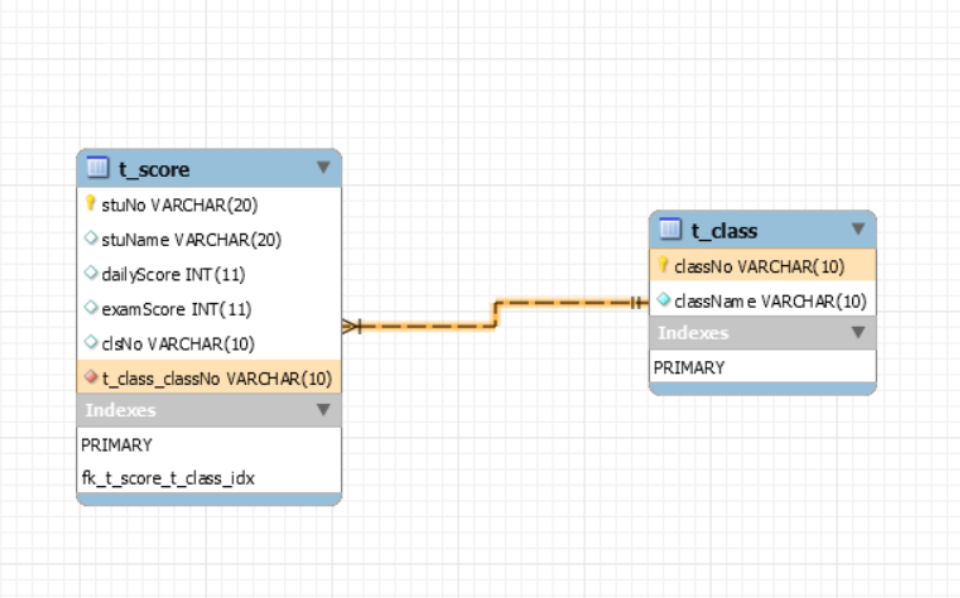
表1 班级表（t\_class）

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 含义 | 数据类型 | 约束 |
| classNo | 班级编号 | varchar(10) | 主键 |
| className | 班级名 | varchar(10) | 一班、二班、三班 |

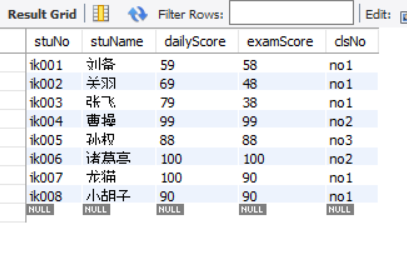
表2 学生成绩表（t\_score）

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 含义 | 数据类型 | 约束 |
| stuNo | 学号 | varchar(20) | 主键 |
| stuName | 学生姓名 | varchar(20) |  |
| dailyScore | 平时成绩 | int |  |
| examScore | 考试成绩 | int |  |
| clsNo | 班级编号 | varchar(10) | 外键，参照t\_class表主键 |

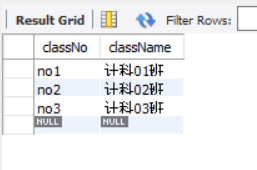
数据库ER-图为：



建成数据库t\_score表：

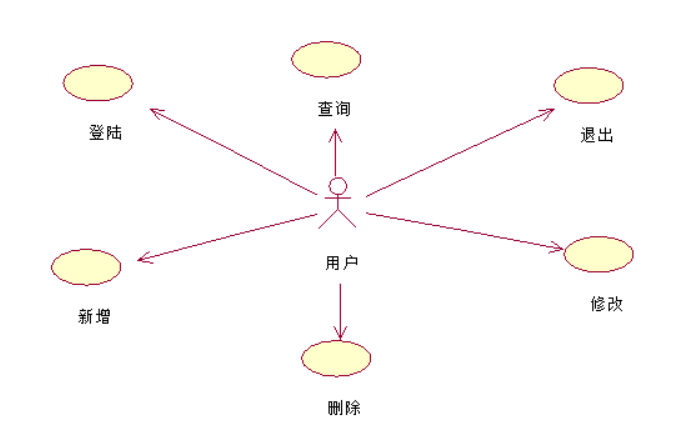


建成数据库t\_class表：

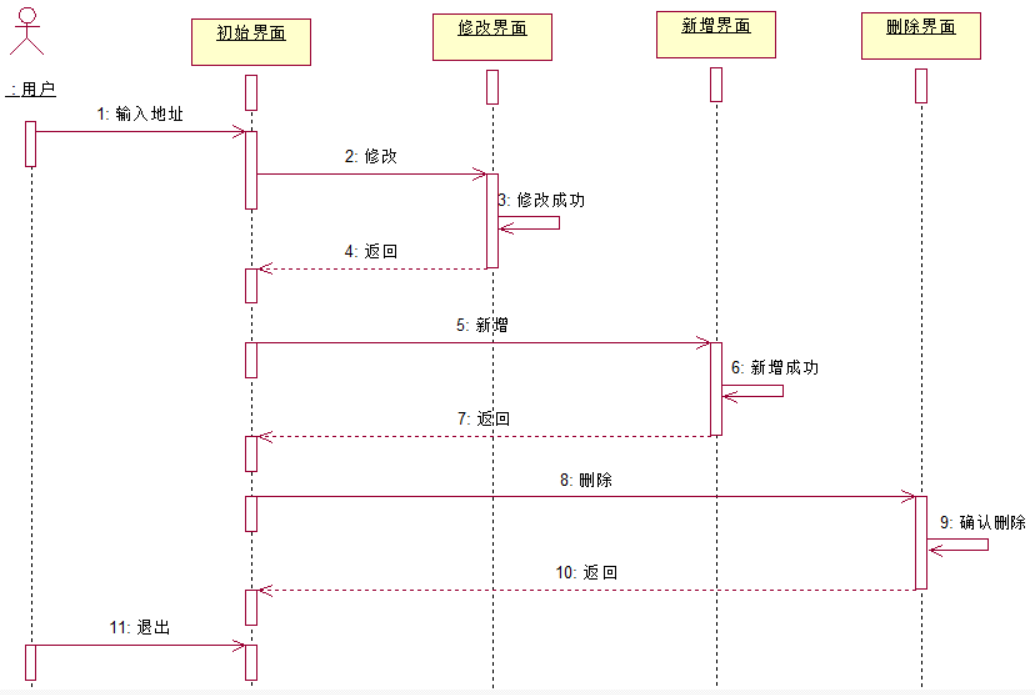


## 1.2 项目过程设计

1. 用户可以查询、修该、删除、新增学生成绩信息。其用例图如下：



2. 具体操作如下：



1. 详细设计

## 2.1 环境搭建及素材准备

1. 素材准备

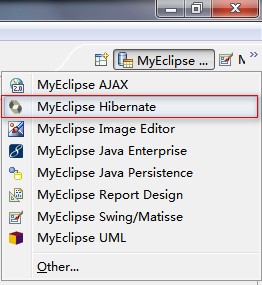
选取几张.jpg格式的图片放到一个images文件夹中，待工程建好后，将文件夹放到WebRoot文件夹下，等待调用。主要是为了页面美观。

2. 环境搭建

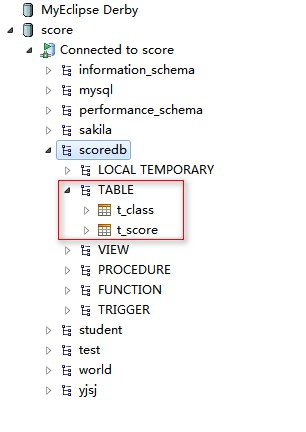
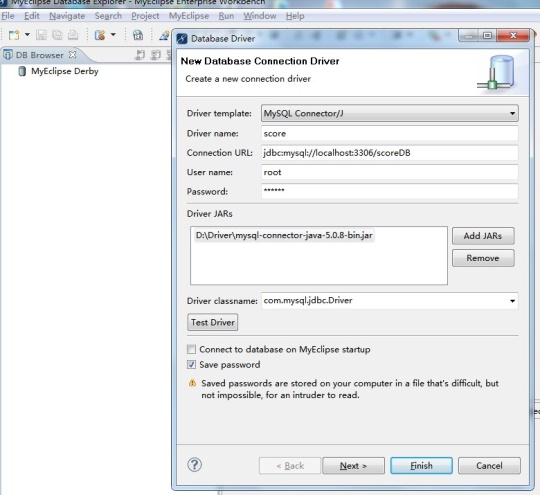
在数据库创建好后：

(1) 打开MyEclipse-->File-->New-->Web Project

(2) 切换到Hibernate面板

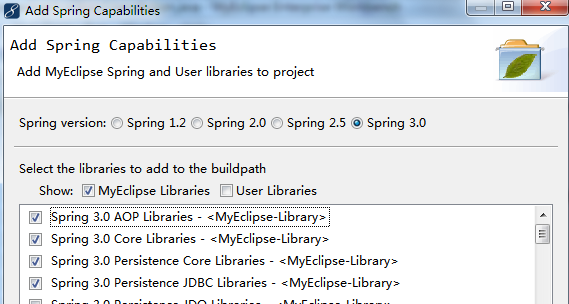
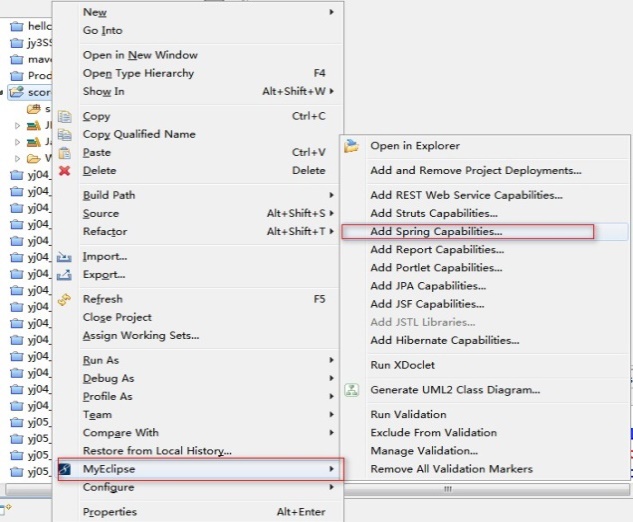


(3)在DB Browser中右键新建一个数据源



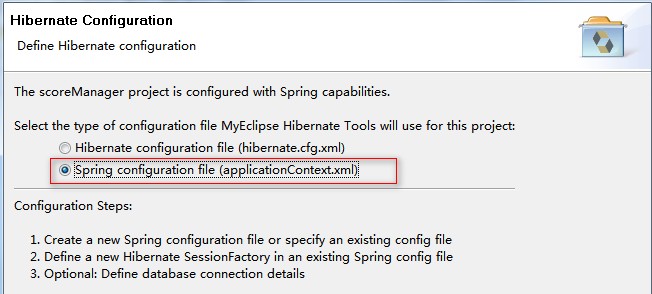
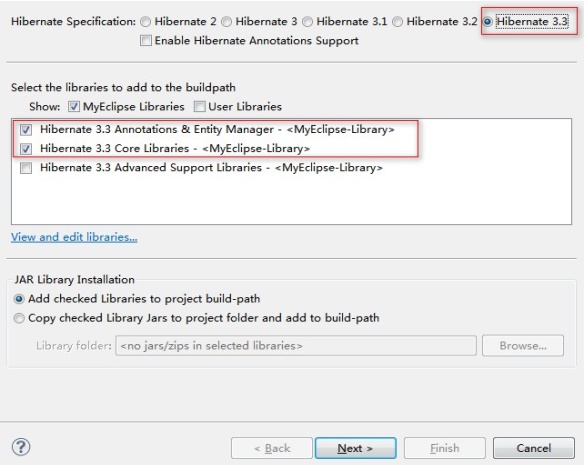
(4)Struts2,Hibernate,Spring整合

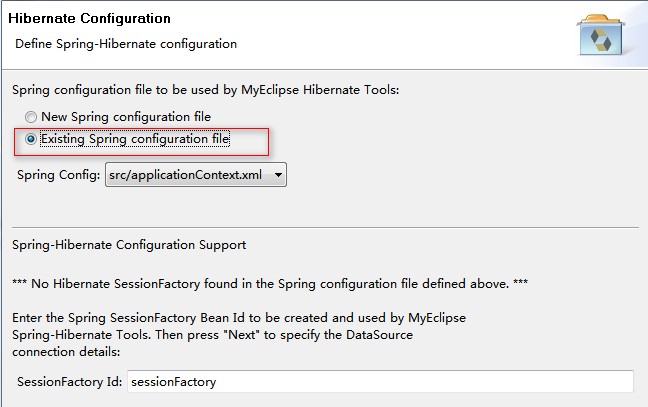
导入Spring (注意：注意需要导入的包，如下图)



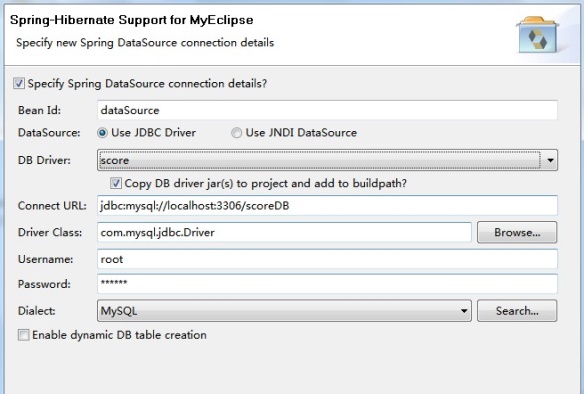
spring2.png

导入Hibernate (注意：如果需要使用Spring配置文件来统一管理如图选择)

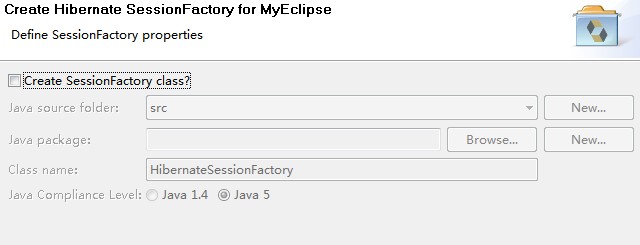




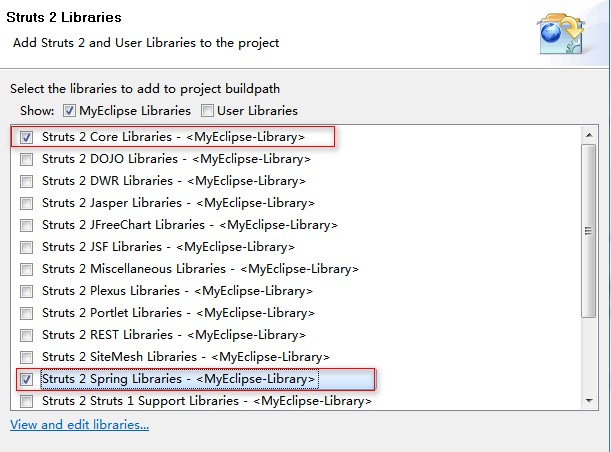
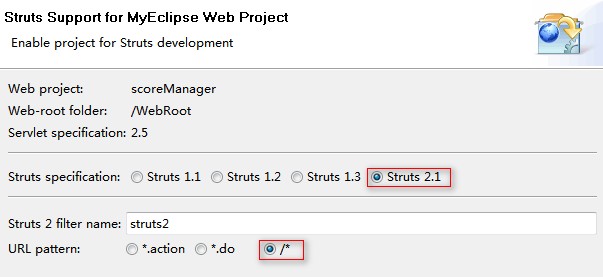
选择数据源



取消创建SessionFactory



导入Struts2 (注意：需要导入的包如下图)



解决冲突包，由于三个框架的整合要解决几个框架中冲突的JAR包

Windows --- preferences --- 在文本框中搜索struts 2(中间有空格)---选择struts 2---选择antlr-2.7.2.jar--- Remove --- OK

在web.xml中编写监听器，以及applicationContext路径

<!--监听器-->

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

<!--配置applicationContext读取路径-->

<context-param>

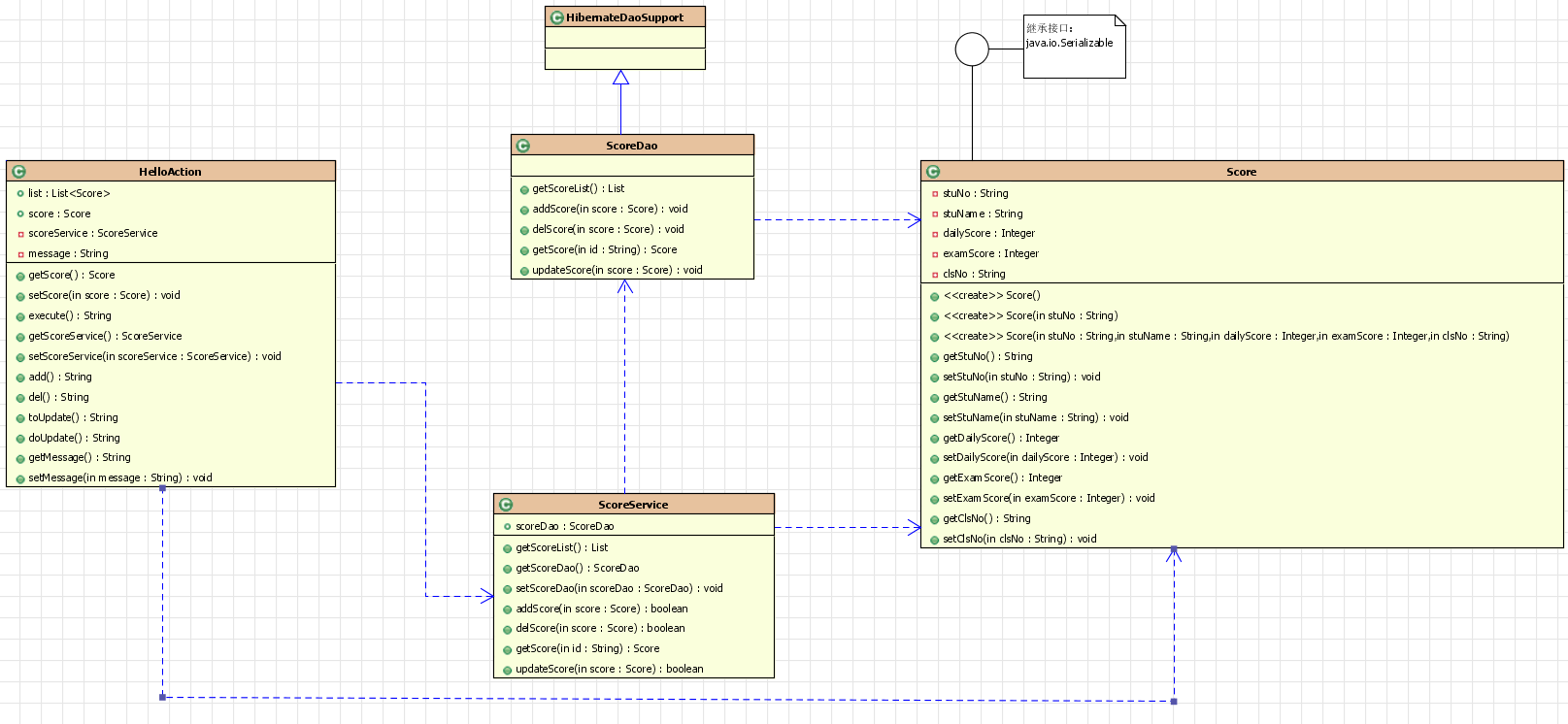
<param-name>contextConfigLocation</param-name>

<param-value>classpath\*:applicationContext\*.xml</param-value>

</context-param>

## 2.2 实现过程

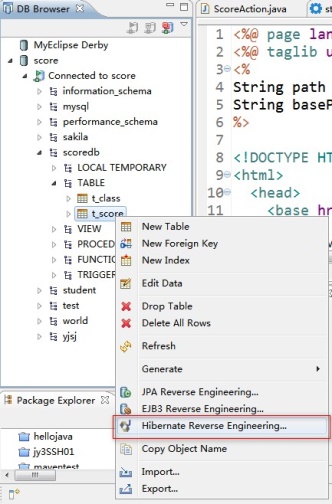
类图：



1.学成绩信息显示

(1) 在src中建edu.rg.score.entity、edu.rg.score.action、edu.rg.score.dao、edu.rg.score.service四个package。

(2)切换为Hibernate面板用t\_class表做逆向生成类。



(3) 在edu.rg.score.dao包中，新建ScoreDao类，在其中添加查询学生成绩信息的代码：

//查询学员列表

**public** List<Score> getScoreList(){

**return** **this**.getHibernateTemplate().find("from Score");}

(4) 在edu.rg.score.service包中，新建ScoreService类，添加将dao层注入的service层代码：

**public** ScoreDao scoreDao;//使用依赖注入

**public** List<Score> getScoreList(){

**return** scoreDao.getScoreList();}

**public** **void** setScoreDao(ScoreDao scoreDao) {

**this**.scoreDao = scoreDao;}

(5) 在edu.rg.score.action包中，新建HelloAction类，添加代码：

**public** List<Score> list;

**private** ScoreService scoreService;//使用依赖注入

**public** String execute(){

list = scoreService.getScoreList();

**return** "success";}

(6)在applicationContext.xml中对这三个类进行注入，举一个例子(dao注入)：

<!-- dao注入 -->

<bean id=*"scoreDao"* class=*"edu.rg.score.dao.ScoreDao"*>

<property name=*"sessionFactory"* ref=*"sessionFactory"*></property>

</bean>

(7)在index.jsp中引入指令和添加迭代集合数据的代码：

<%@ page language=*"java"* import=*"java.util.\*"* pageEncoding=*"utf-8"*%>



(8)在struts.xml中配置：

<struts>

<package name=*"default"* namespace=*"/"* extends=*"struts-default"*>

<action name=*"score"* class=*"edu.rg.score.action.HelloAction"*>

<result name=*"success"*>index.jsp</result>

</action>

</package>

</struts>

2.学生成绩信息添加

(1)在ScoreDao类中添加增加学生信息

//添加学生信息

**public** **void** addScore(Score score){

**this**.getHibernateTemplate().save(score);}

(2)在ScoreService类中编写调用dao的方法

//新增

**public** **boolean** addScore(Score score){

**try** {scoreDao.addScore(score);

**return** **true**;

} **catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();

}**return** **false**;}

(3)在HelloAction类中引入dao实体

//新增学员信息

**public** String add(){

**boolean** flag=scoreService.addScore(score);

**if**(flag){

message="<script>alert('新增成功')</script>";

**return** execute();//如果添加成功，则刷新列表

}**else**{

message="<script>alert('新增失败')</script>";

**return** "false";}}

(4)新建add.jsp，作为添加页面，单标签action的地址

提交元素的name属性

<body style="background: *url(images/6.jpg)*">

<center>

<form action=*"score!add"* method=*"post"*>

<p>添加学生考试信息</p>

<p>学号：<input type=*"text"* name=*score.stuNo*></p>

<p>姓名：<input type=*"text"* name=*score.stuName*></p>

<p>平时：<input type=*"text"* name=*score.dailyScore*></p>

<p>考试：<input type=*"text"* name=*score.examScore*></p>

<p>班级：

<select name=*"score.clsNo"*>

<option value=*"no1"*>计科01班</option>

<option value=*"no2"*>计科02班</option>

<option value=*"no3"*>计科03班</option>

</select>

</p>

<p><input type=*"submit"* value=*"提交"*></p>

</form>

</center>

</body>

(5)在index.jsp中添加页面跳转

<input align=*"middle"* onclick="location.href='add.jsp'" type=*"button"*

value=*"新增学员考试信息"*>

3.学生成绩信息修改与删除

(1)在ScoreDao类中添加个根据id查询学生成绩信息、修改学生信息及删除学生信息

//删除学生信息

**public** **void** delScore(Score score){

**this**.getHibernateTemplate().delete(score);}

//通过ID查询学生信息

**public** Score getScore(String id){

**return** **this**.getHibernateTemplate().get(Score.**class**, id);}

//通过ID修改学生信息

**public** **void** updateScore(Score score){

**this**.getHibernateTemplate().bulkUpdate("update Score set stuName

= ?,dailyScore = ?,"+"examScore = ?,clsNo=? where stuNo=?",

score.getStuName(),score.getDailyScore(),score.getExamScore(),score.getClsNo(),score.getStuNo());}

(2)在Score1Service类中添加调用dao的方法

//删除

**public** **boolean** delScore(Score score){

**try** {scoreDao.delScore(score);

**return** **true**;} **catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();}

**return** **false**;}

//根据ID查对象

**public** Score getScore(String id){Score score=**null**;

**try** {score = scoreDao.getScore(id);

**return** score;} **catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();}

**return** score;}

//修改学生考试成绩

**public** **boolean** updateScore(Score score){

**try** {scoreDao.updateScore(score);

**return** **true**;} **catch** (Exception e) {

// **TODO**: handle exception

e.printStackTrace();}

**return** **false**;}

(3)在HelloAction类中添加执行

//删除学员信息

**public** String del(){

**boolean** flag=scoreService.delScore(score);

**if**(flag){message="<script>alert('删除成功')</script>";

**return** execute();//如果添加成功，则刷新列表}**else**{

message="<script>alert('删除失败')</script>";

**return** "false";}}

//根据ID查到这个学生成绩对象，把该对象转发到edit页面进行修改

**public** String toUpdate(){

score = scoreService.getScore(score.getStuNo());

**if**(score!=**null**){**return** "edit";

}**else**{**return** execute();}}

//修改学生成绩

**public** String doUpdate(){

**boolean** flag=scoreService.updateScore(score);

**if**(flag){message="<script>alert('修改成功')</script>";

**return** execute();//如果添加成功，则刷新列表

}**else**{message="<script>alert('修改失败')</script>";

**return** "edit";}}

(4)新建edit.jsp,为修改页面

<body>

<center>

<form action=*"score!doUpdate"* method=*"post"*>

<!-- 将不用修改的score.stuNo放入隐藏作用域 -->

<input type=*"hidden"* name=*"score.stuNo"* value=*"*${score.stuNo}*"*>

<p>编辑学生考试信息</p>

<p>姓名：<input type=*"text"* name=*"score.stuName"* value=*"*${score.stuName}*"* ></p>

<p>平时：<input type=*"text"* name=*"score.dailyScore"* value=*"*${score.dailyScore}*"*></p>

<p>考试：<input type=*"text"* name=*"score.examScore"* value=*"*${score.examScore}*"*></p>

<p>班级：

<select name=*"score.clsNo"*>

<option value=*"no1"*>计科01班</option>

<option value=*"no2"*>计科02班</option>

<option value=*"no3"*>计科03班</option>

</select>

</p>

<p><input type=*"submit"* value=*"提交"*></p>

</form>

</center>

</body>

(5)在index.jsp中添加页面跳转

<td><a href=*"javascript:void(0)"* onclick=*"if(confirm('你确定要删除吗？')){location.href ='score!del?score.stuNo=*${score.stuNo}*'}"*>删除</a>/

<a href=*"score!toUpdate?score.stuNo=*${score.stuNo}*"*>修改</a></td>

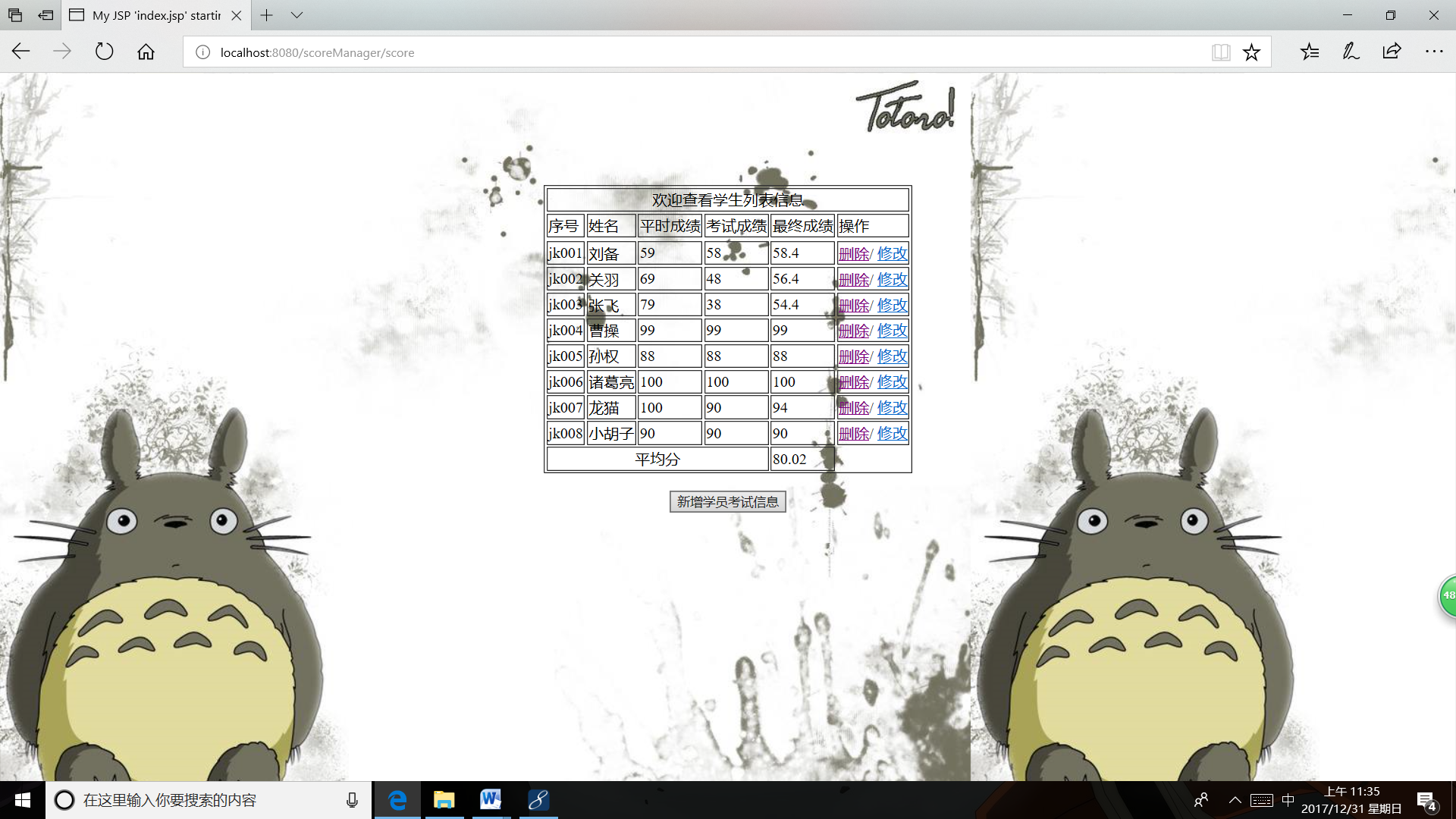
(6)在struts.xml中添加响应

<result name=*"false"*>index.jsp</result>

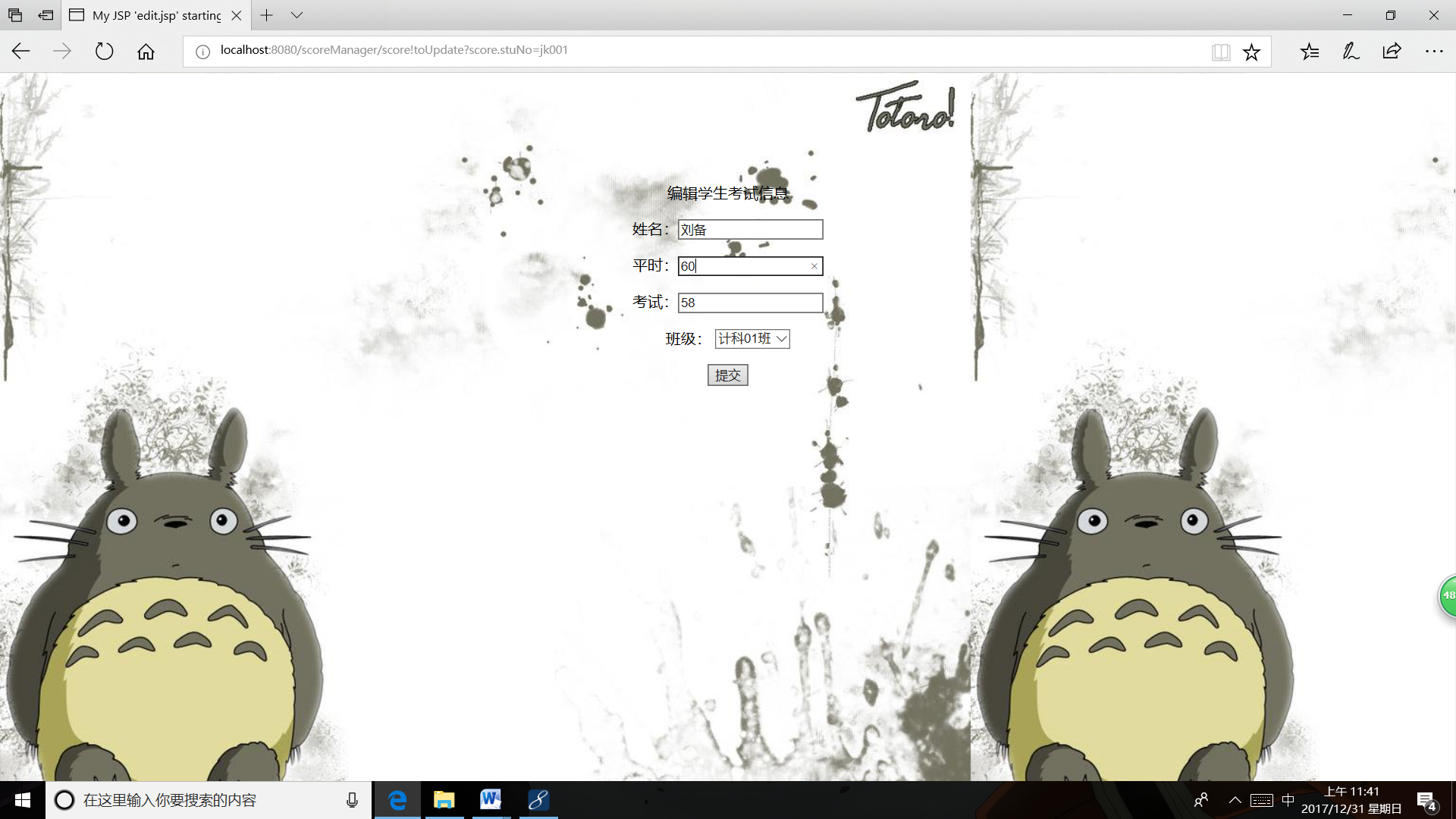
<result name=*"edit"*>edit.jsp</result>

## 2.3 实现结果

1.初始界面



2.修改界面



3.添加界面

