# 实验二 基于Struts2的CRUD

一、实验目的：

熟悉Struts2 Action类的基本用法，掌握Struts2 Action类在MVC程序开发中的编写和应用，实现CRUD编程。

二、实验内容：

在MyEclipse环境下运用Struts2技术，完成数据库CRUD功能，调试运行程序。

三、实验要求：

1. 掌握运用Struts2 开发MVC程序的基本步骤；

2. 掌握Struts2 Action接收请求参数、调用业务逻辑、封装结果数据并转向结果页面的用法

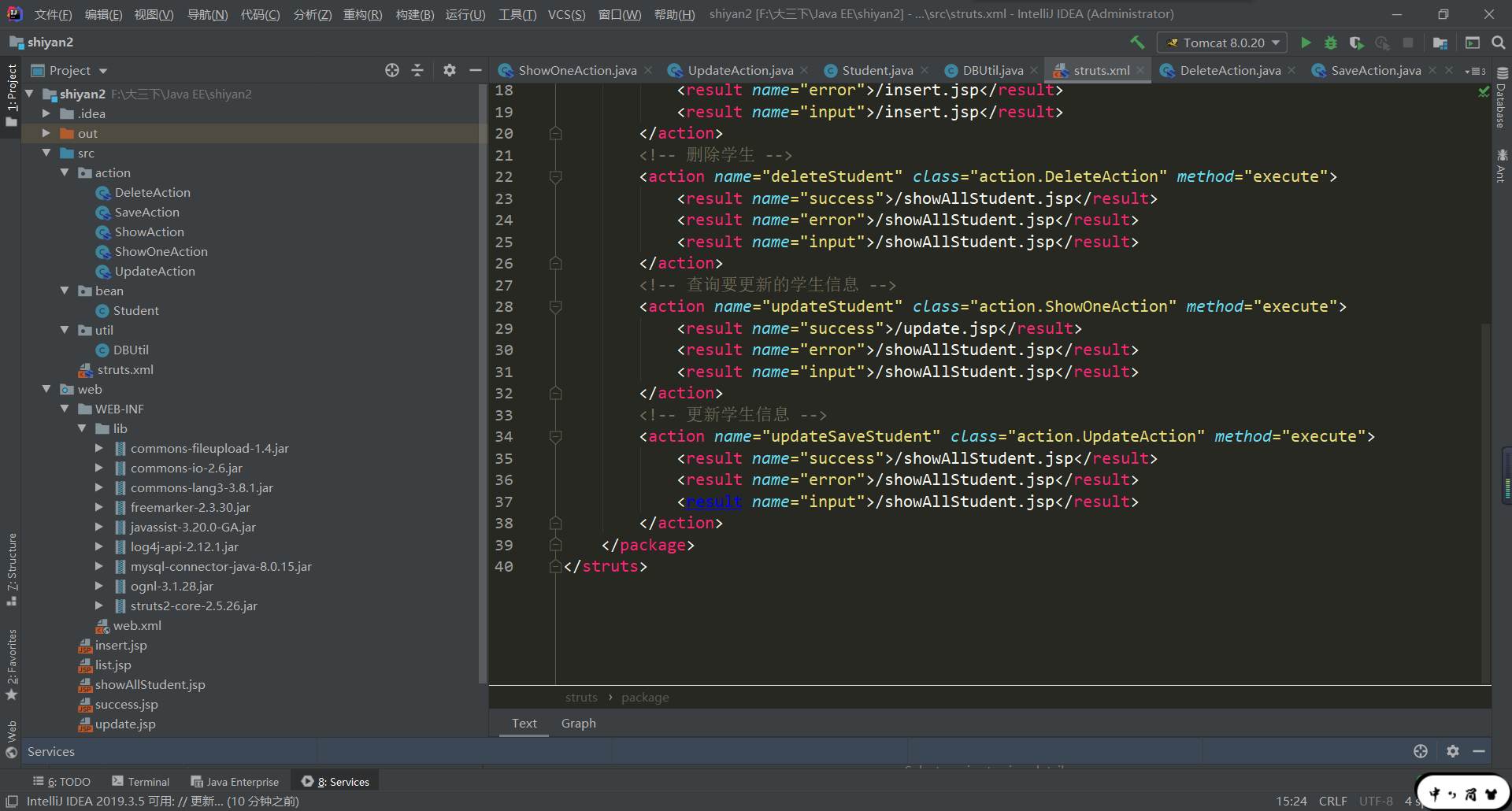
3. 掌握在struts.xml文件中对Struts2 Action的基本配置；

4. 运用相关技术（Struts2以及JDBC等）完成规定功能；

5. 写出实验报告。

四、实验步骤：

1．进入MyEclipse环境，新建一个Web Project。



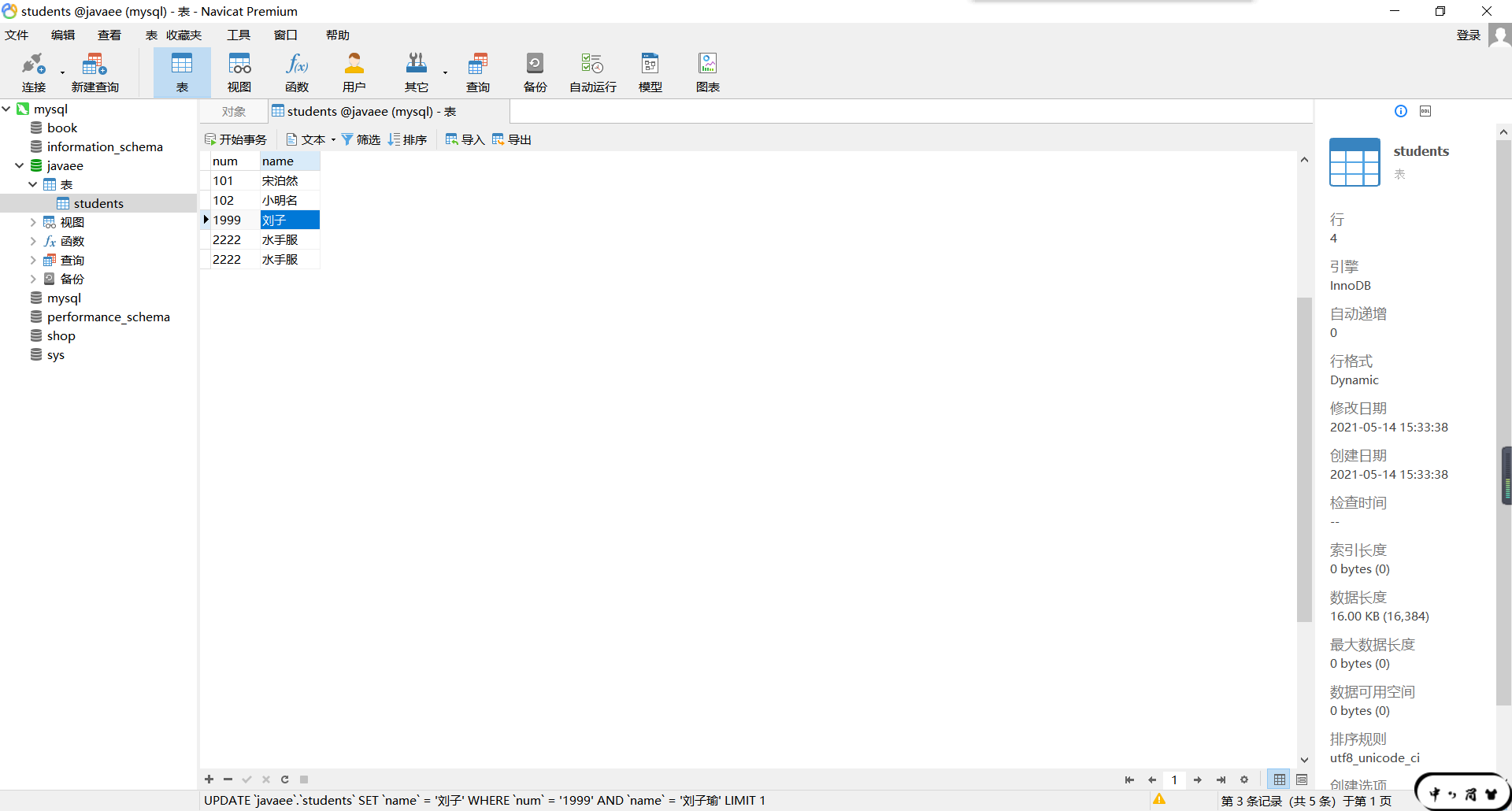
1. 设计数据库表结构。

CREATE TABLE `students` (

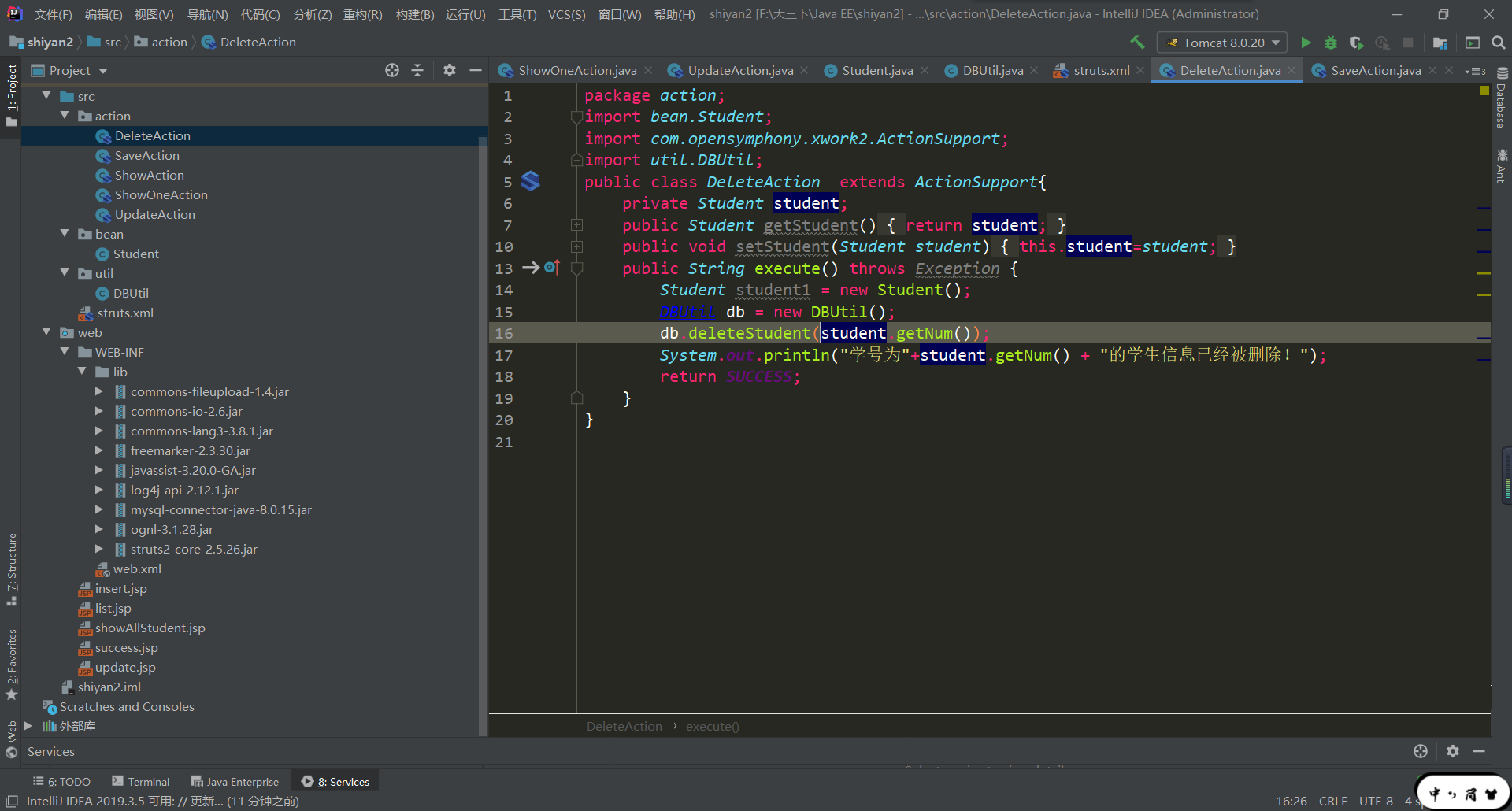
`num` varchar(255) NOT NULL,

`name` varchar(255) NOT NULL

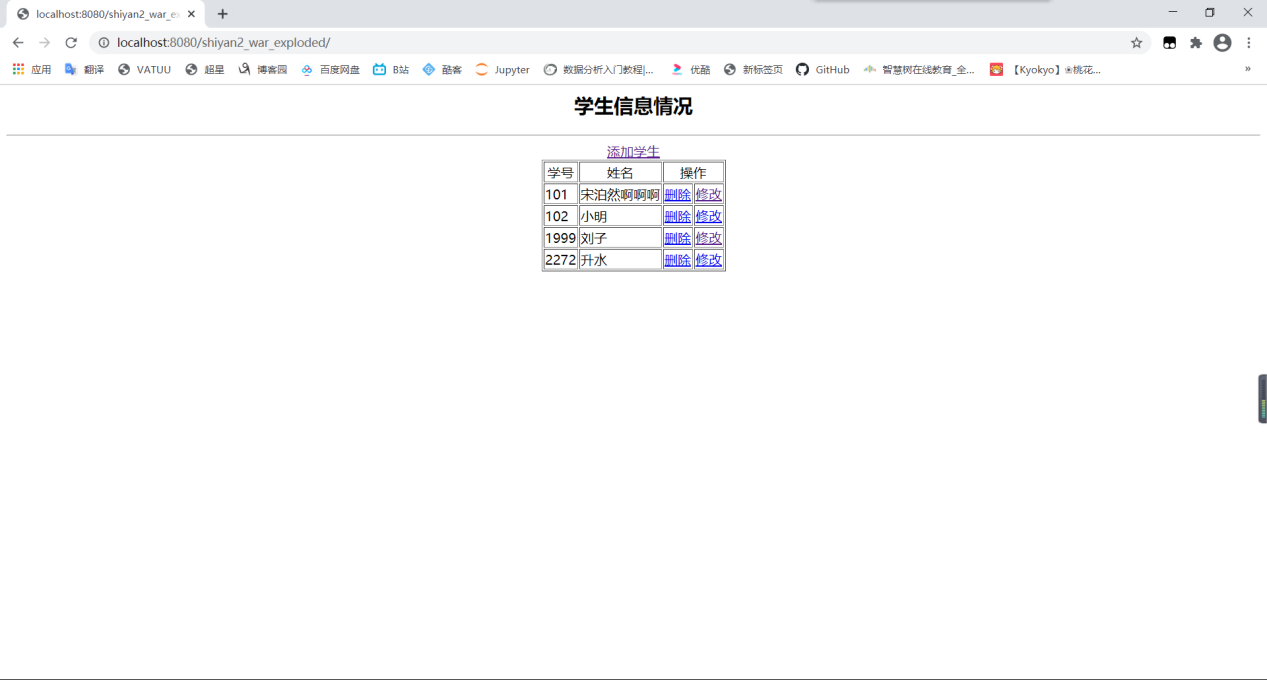
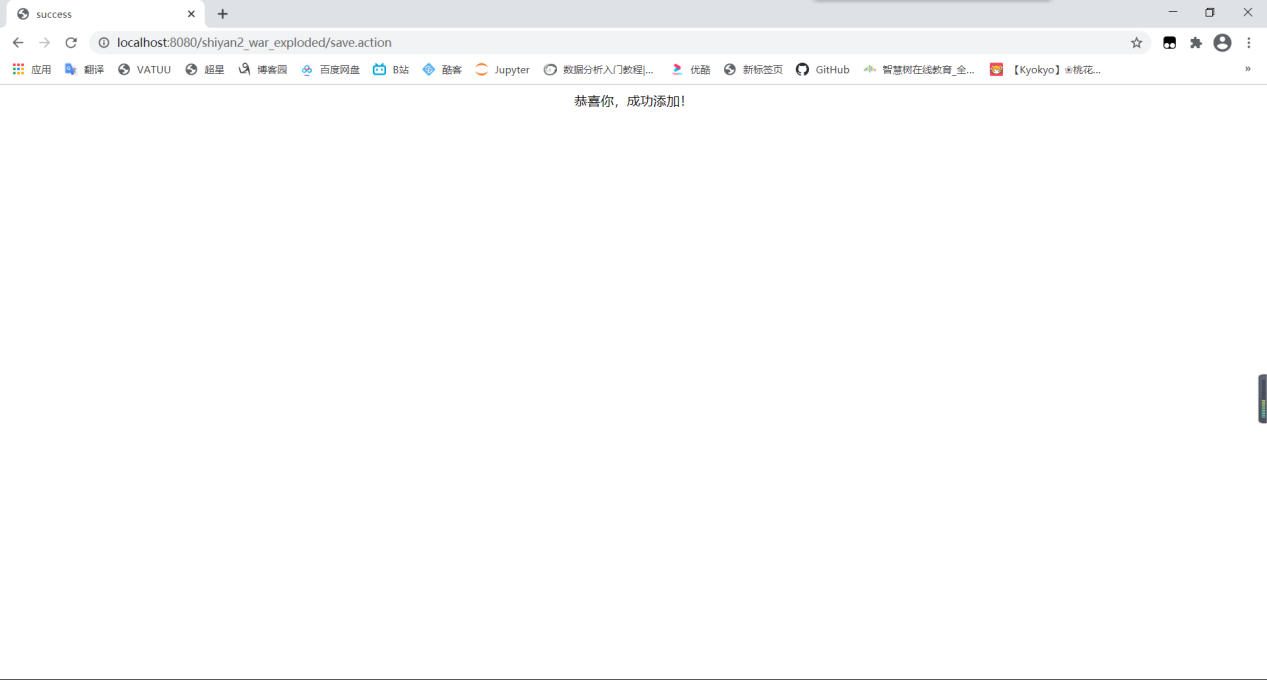
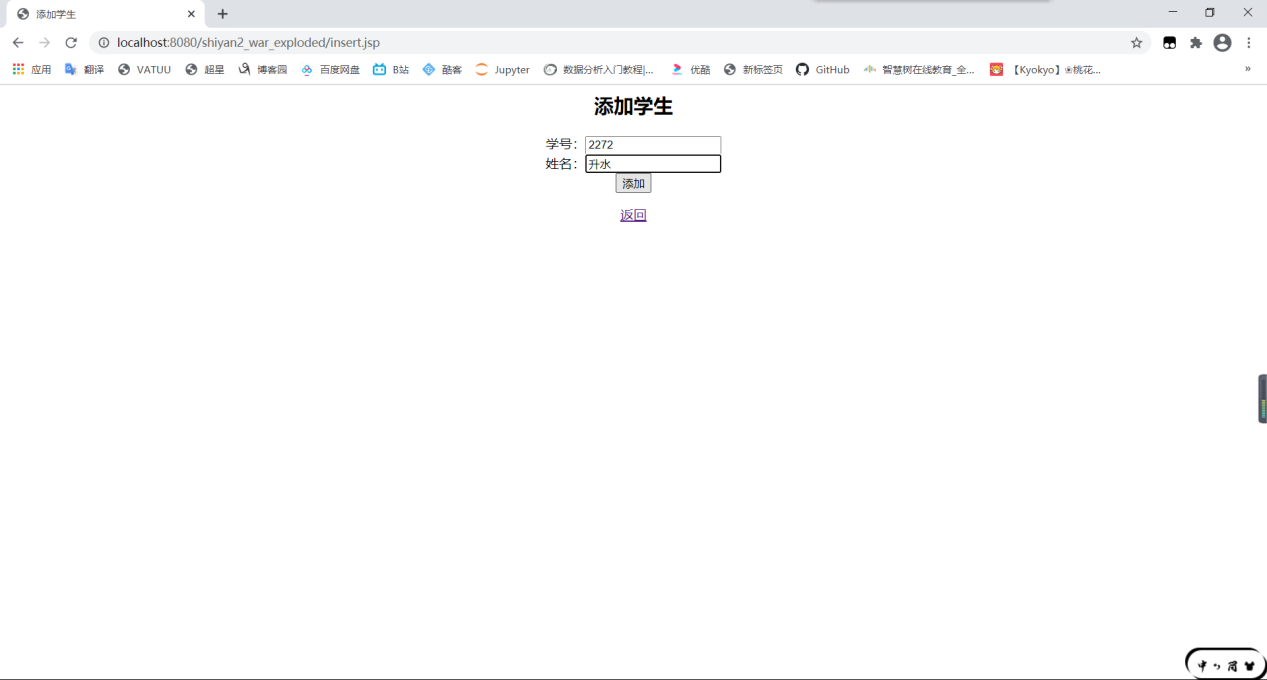
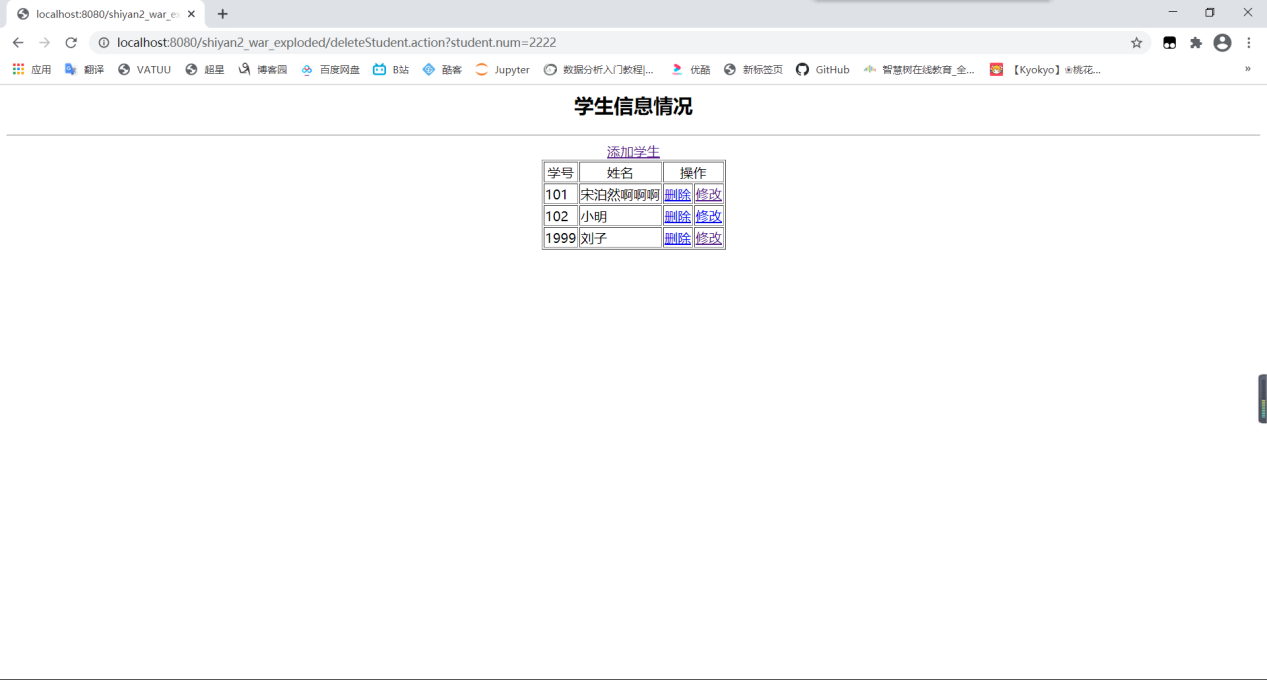
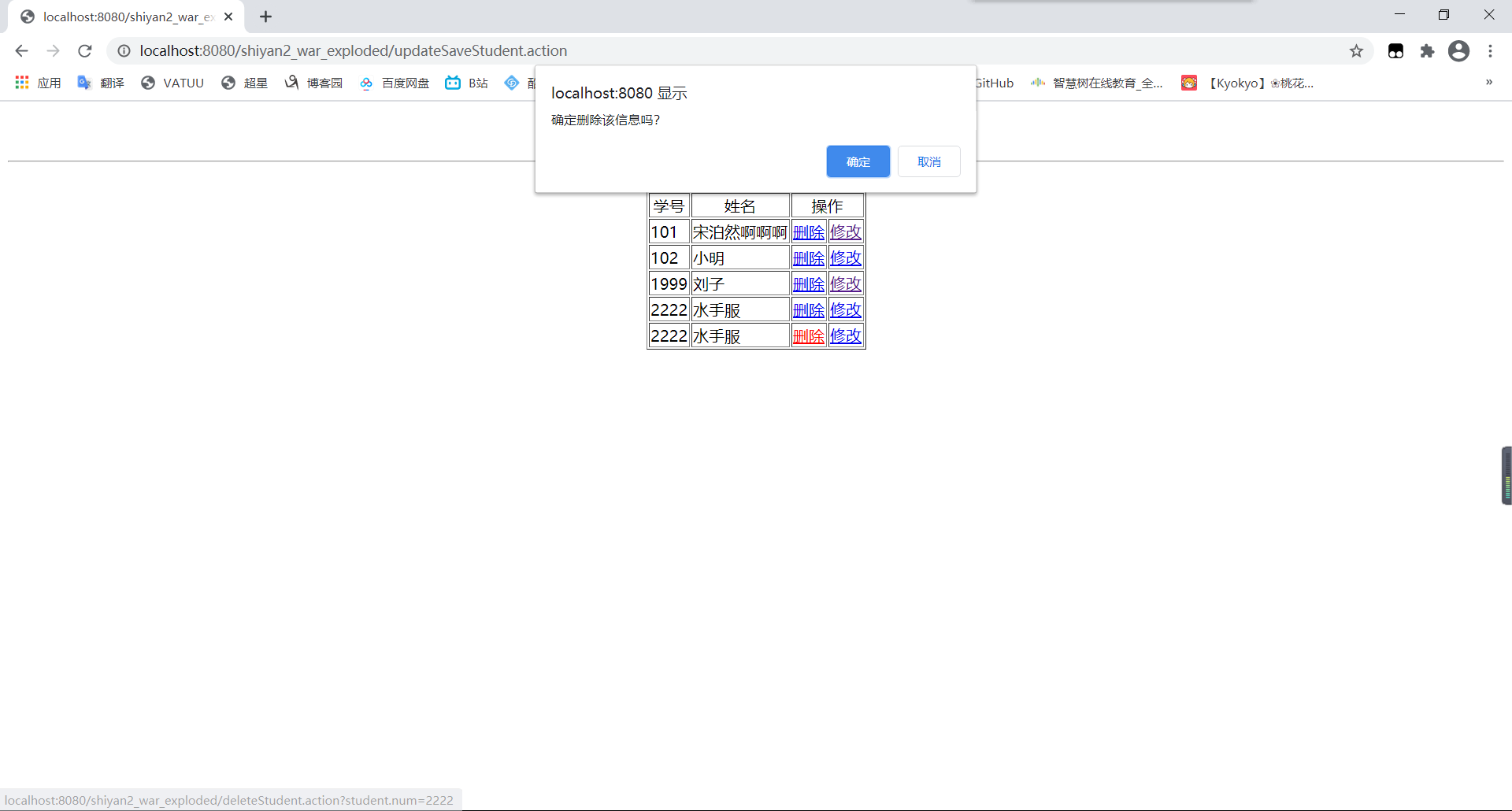
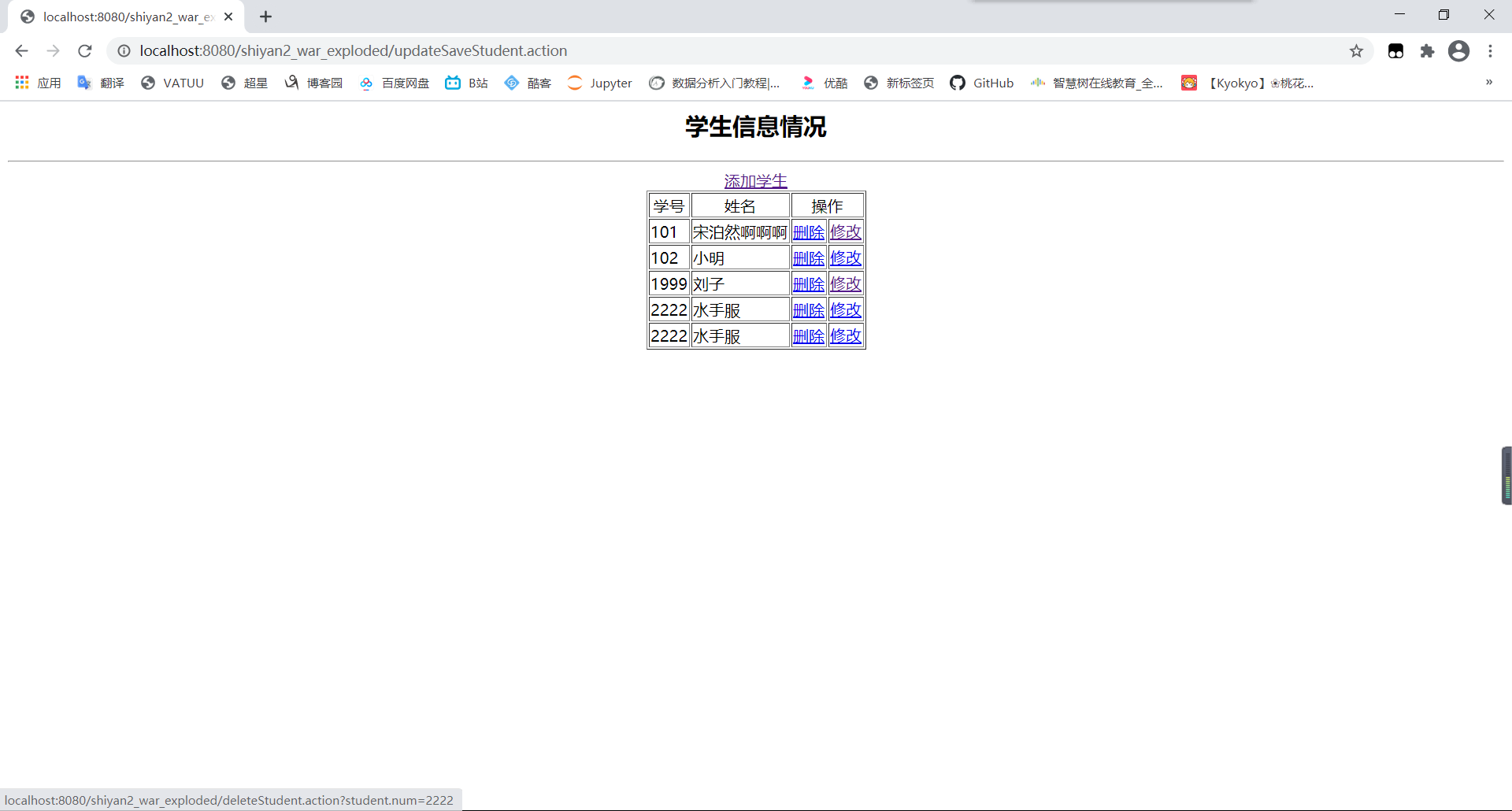
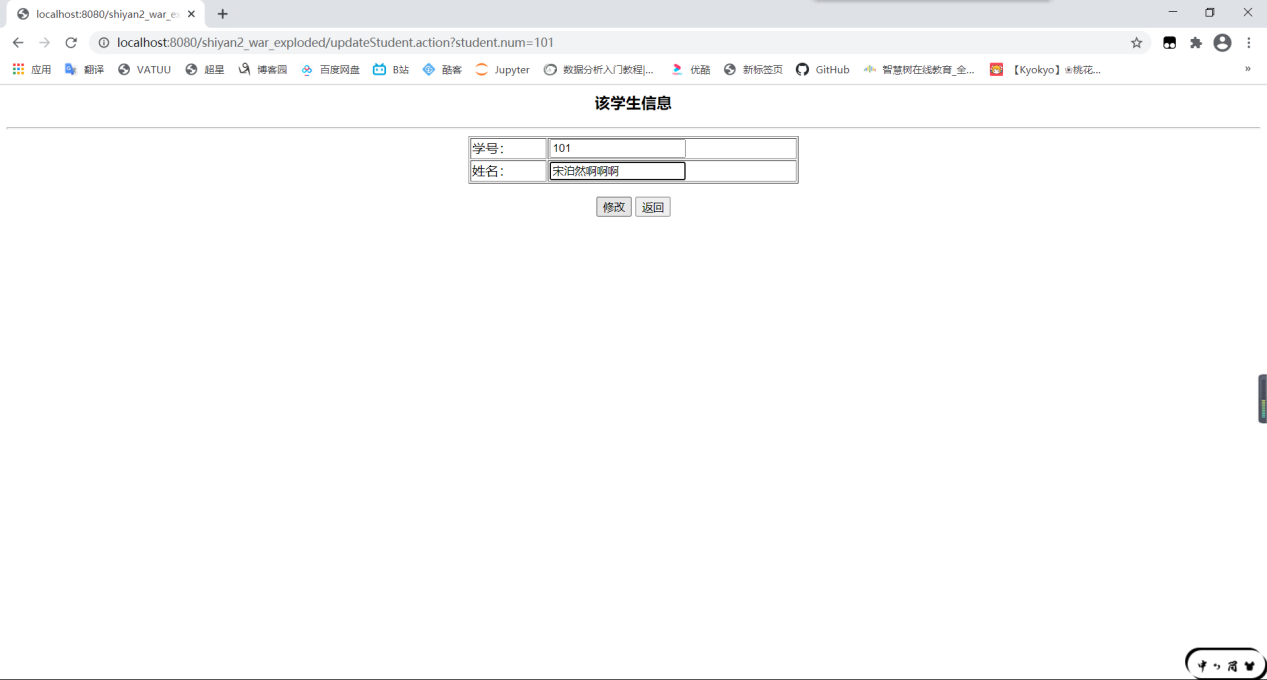
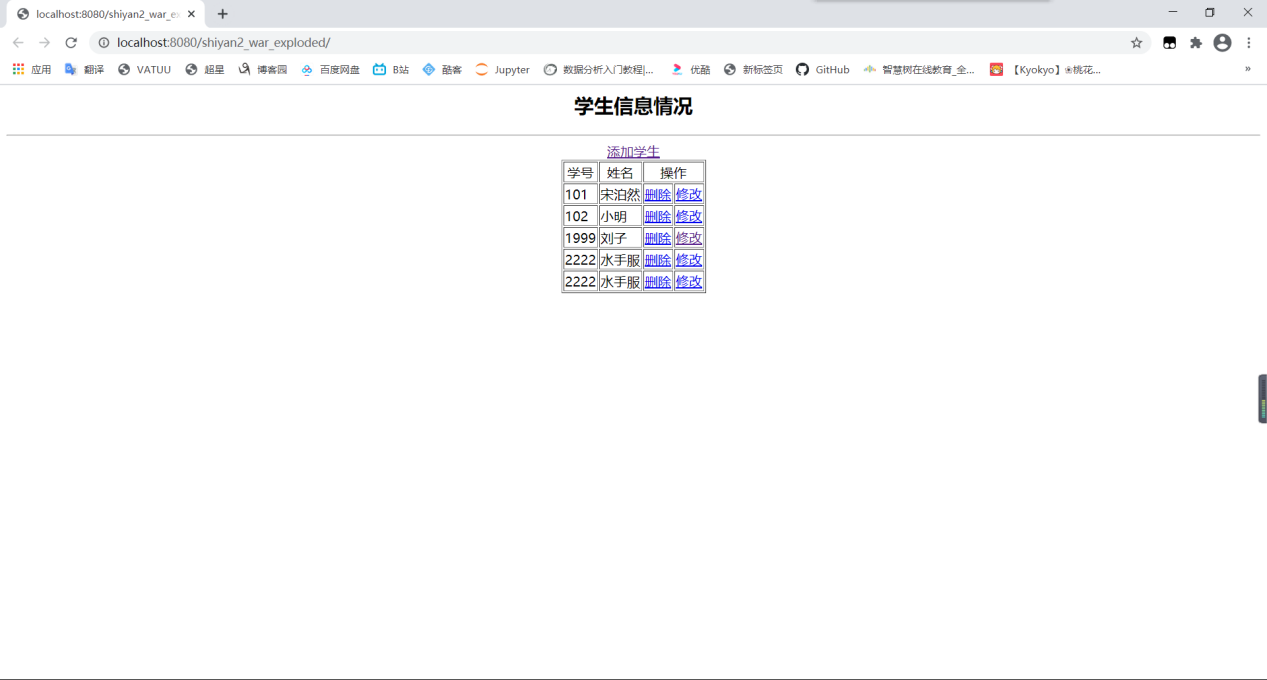
) ;



1. 采用Struts2技术完成数据库的CRUD。

Action：

1. 程序运行截图。



源代码

Deleteaction

package *action*;  
import *bean.Student*;  
import *com.opensymphony.xwork2.ActionSupport*;  
import *util.DBUtil*;  
public class *DeleteAction* extends *ActionSupport*{  
 private *Student* student;  
 public *Student* getStudent() {  
 return student;  
 }  
 public void setStudent(*Student student*) {  
 this.student=*student*;  
 }  
 public *String* execute() throws *Exception* {  
 *Student* student1 = new Student();  
 *DBUtil* db = new DBUtil();  
 db.deleteStudent(student.getNum());  
 *System*.***out***.println("学号为"+student.getNum() + "的学生信息已经被删除！");  
 return ***SUCCESS***;  
 }  
}

Saveaction

package *action*;  
import *bean.Student*;  
import *util.DBUtil*;  
import *com.opensymphony.xwork2.ActionSupport*;  
public class *SaveAction* extends *ActionSupport* {  
 private *Student* student;  
 public *Student* getStudent() {  
 return student;  
 }  
 public void setStudent(*Student student*) {  
 this.student=*student*;  
 }  
 public *String* execute() throws *Exception*{  
 *DBUtil* db=new DBUtil();  
 *Student* stu=new Student();  
 stu.setNum(student.getNum());  
 stu.setName(student.getName());  
 if(db.save(stu)){  
 return ***SUCCESS***;  
 }else{  
 return ***ERROR***;  
 }  
 }  
  
}

Updateaction

package *action*;  
import *bean.Student*;  
import *com.opensymphony.xwork2.ActionContext*;  
import *com.opensymphony.xwork2.ActionSupport*;  
import *util.DBUtil*;  
import *java.sql.SQLException*;  
import *java.util.ArrayList*;  
import *java.util.List*;  
import *java.util.Map*;  
public class *UpdateAction* extends *ActionSupport* {  
 private *Student* student;  
  
 public *Student* getStudent() {  
 return student;  
 }  
  
 public void setStudent(*Student student*) {  
 this.student=*student*;  
 }  
 public *String* execute() throws *Exception* {  
 *DBUtil* db = new DBUtil();  
 student.setNum(student.getNum());  
 student.setName(student.getName());  
 db.updateSaveStudent(student);  
 return ***SUCCESS***;  
 }  
}

Structs.xml

*<?xml version*="1.0" *encoding*="UTF-8"*?>*<!DOCTYPE *struts* PUBLIC  
 "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"  
 "http://struts.apache.org/dtds/struts-2.0.dtd"*>*<struts>  
 <constant *name*="struts.devMode" *value*="true"/>  
 <package *name*="basicstruts2" *extends*="struts-default">  
  
 <action *name*="save" *class*="action.SaveAction">  
 <result *name*="success">/success.jsp</result>  
 <result *name*="error">/insert.jsp</result>  
 </action>  
 <!--查询所有学生 -->  
 <action *name*="showAllStudent" *class*="action.ShowAction" *method*="execute">  
 <result *name*="success">/list.jsp</result>  
 <result *name*="error">/insert.jsp</result>  
 <result *name*="input">/insert.jsp</result>  
 </action>  
 <!-- 删除学生 -->  
 <action *name*="deleteStudent" *class*="action.DeleteAction" *method*="execute">  
 <result *name*="success">/showAllStudent.jsp</result>  
 <result *name*="error">/showAllStudent.jsp</result>  
 <result *name*="input">/showAllStudent.jsp</result>  
 </action>  
 <!-- 查询要更新的学生信息 -->  
 <action *name*="updateStudent" *class*="action.ShowOneAction" *method*="execute">  
 <result *name*="success">/update.jsp</result>  
 <result *name*="error">/showAllStudent.jsp</result>  
 <result *name*="input">/showAllStudent.jsp</result>  
 </action>  
 <!-- 更新学生信息 -->  
 <action *name*="updateSaveStudent" *class*="action.UpdateAction" *method*="execute">  
 <result *name*="success">/showAllStudent.jsp</result>  
 <result *name*="error">/showAllStudent.jsp</result>  
 <result *name*="input">/showAllStudent.jsp</result>  
 </action>  
 </package>  
</struts>

Insert.jsp

<%@ page *contentType*="text/html;charset=UTF-8" *language*="java" %>  
<%@ taglib *uri*="/struts-tags" *prefix*="s"%>  
<html>  
<head>  
 <title>添加学生</title>  
</head>  
<body>  
<center>  
<h2>添加学生</h2>  
<s:form *action*="save" *method*="post" *theme*="simple">  
 学号：<s:textfield *name*="student.num"></s:textfield><br>  
姓名：<s:textfield *name*="student.name"></s:textfield><br>  
 <s:submit *value*="添加"></s:submit>  
</s:form>  
 <a *href*="showAllStudent.jsp">返回</a>  
</center>  
</body>  
</html>

五、心得体会：

1、在软件设计上，strut2没有像strut1那样跟ServletAPI和strutAPI有着紧密的耦合，strut2的应用可以不依赖于ServletAPI和strutAPI。strut2的这种设计属于无侵入设计，而strut1属于侵入设计。

2、strut2提供了拦截器，利用拦截器可以进行AOP编程，实现如权限拦截等功能。

3、strut2提供了类型转换器，我们可以把特殊的请求参数转换成所需要的类型。在strut1中，如果要实现同样的功能，就必须向strut1的底层实现BeanUnit注册类型转换器才行。

4、strut2提供支持多种表现层技术，如果JSP、freeMarker等。

5、strut2的输入校验可以对指定方法进行校验，解决了strut1长久之痛。

6、提供了全局范围、包范围和Action范围的国际化资源文件管理实现。