****

**实 验 报 告**

|  |  |  |
| --- | --- | --- |
| **课程名称**： WEB应用开发技术 | **任课教师：** 雷 宇 | **实验日期**： 2021.5.28 |
| **班 级**： 信1901-4 | **姓 名**： 闫竞存 | **学 号**： 20194127 |

**实验项目名称：实验三 Web数据库程序设计**

一、实验目的

通过使用JSP技术设计一个简单的数据库管理系统，了解展示页面和编辑页面的区别，掌握Web服务器与MySQL数据库的连接和数据库操作的方法，掌握使用Java语言编写JSP文件的方法。

二、实验内容和基本要求

开发一个学生管理系统，完成学生信息的查看、增加、删除和修改等功能。对于客户端增加和修改信息页面，要使用javascript进行必要的数据的非空验证。自行添加一些CSS，使得页面和字体更加美观。将网页代码、程序源代码和浏览器截图写入实验报告。

三、实验步骤

**SpringBoot + LayUI 简易学生管理系统**

**项目：**[**http://39.103.184.190:8889/**](http://39.103.184.190:8889/)

**源码：**[**https://github.com/jancoyan/Curriculum/tree/master/weblabs/lab03**](https://github.com/jancoyan/Curriculum/tree/master/weblabs/lab03)

由于源码比较长，这里只粘贴部分源码（pojo、controller、mapper、html、js）

|  |
| --- |
| */\*\*  \* @Author: Yan Jingcun  \* @Date: 2021/5/26  \* @Description:  \* @Version: 1.0  \*/* package com.jancoyan.lab03.pojo;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  import java.util.Date;  @Data @AllArgsConstructor @NoArgsConstructor public class Student {   *// INT(10) PRIMARY\_KEY 学号* private Integer id;    *// VARCHAR(10) 学生姓名* private String name;    *// INT(3) 年龄* private Integer age;    *// INT(1) 性别* private Integer sex;    *// INT(1) 年级* private Integer grade;    *// DATE 生日* private Date birthday;    *// INT(2) 院系* private Integer dept;    *// VARCHAR(11) 电话号码* private String telphone;    *// VARCHAR(10) qq号码* private String qq;  }  */\*\*  \* @Author: Yan Jingcun  \* @Date: 2021/5/27  \* @Description:  \* @Version: 1.0  \*/* package com.jancoyan.lab03.pojo;  import lombok.AllArgsConstructor; import lombok.Data; import lombok.NoArgsConstructor;  @Data @AllArgsConstructor @NoArgsConstructor public class Department {    *// INT(2) PRIMARY\_KEY 院系号* private int deptid;    *// VARCHAR(20) 院系名称* private String deptname;    *// VARCHAR(50) 院系地址* private String location; }  */\*\*  \* @Author: Yan Jingcun  \* @Date: 2021/5/28  \* @Description:  \* @Version: 1.0  \*/* package com.jancoyan.lab03.controller;  import org.springframework.context.annotation.Configuration; import org.springframework.web.servlet.config.annotation.ViewControllerRegistry; import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  @Configuration public class MvcController implements WebMvcConfigurer {   @Override  public void addViewControllers(ViewControllerRegistry registry) {  registry.addViewController("/").setViewName("index");  registry.addViewController("/index.html").setViewName("index");  } }  */\*\*  \* @Author: Yan Jingcun  \* @Date: 2021/5/26  \* @Description:  \* @Version: 1.0  \*/* package com.jancoyan.lab03.controller;  import com.github.pagehelper.PageInfo; import com.jancoyan.lab03.pojo.Student; import com.jancoyan.lab03.service.StudentService; import com.jancoyan.lab03.utils.Msg; import com.jancoyan.lab03.utils.TimeUtils; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.ResponseBody;   @Controller public class StudentController {   @Autowired  private StudentService studentService;   */\*\*  \* RESTful  \* /students  \* - GET 获取所有学生信息  \* - DELETE 删除一条记录  \* - POST 修改一条记录  \* - PUT 插入一条记录  \*/   /\*\*  \* 获取所有学生的分页列表  \* @param page 第几页  \* @param limit 一页多少数据  \* @return json数据  \*/* @RequestMapping(value = "/students", method = RequestMethod.*GET*)  @ResponseBody  public Msg getStudentsByPage(Integer page, Integer limit){  PageInfo<Student> pageInfo = studentService.findAll(page, limit);  return Msg.*success*().add("count", pageInfo.getTotal()).add("data", pageInfo);  }   */\*\*  \* 删除一条记录  \* @param id 主键  \* @return 删除状态消息  \*/* @RequestMapping(value = "/students", method = RequestMethod.*DELETE*)  @ResponseBody  public Msg deleteStudent(Integer id){  studentService.deleteStudentById(id);  return Msg.*success*();  }   */\*\*  \* 更新学生信息  \* @param id 学号  \* @param name 姓名  \* @param age 年龄  \* @param sex 性别  \* @param grade 年级  \* @param birthday 生日  \* @param dept 学院  \* @param telphone 电话  \* @param qq qq  \* @return 消息  \*/* @RequestMapping(value = "/students", method = RequestMethod.*POST*)  @ResponseBody  public Msg updateStudent(Integer id,  String name,  Integer age,  Integer sex,  Integer grade,  String birthday,  Integer dept,  String telphone,  String qq){  Student student = new Student();  student.setId(id);  student.setAge(age);  student.setBirthday(TimeUtils.*castDateStringToDateTypeYMD*(birthday));  student.setDept(dept);  student.setGrade(grade);  student.setSex(sex);  student.setName(name);  student.setTelphone(telphone);  student.setQq(qq);  studentService.updateStudent(student);  return Msg.*success*();  }   */\*\*  \* 添加学生信息  \* @param id 学号  \* @param name 姓名  \* @param age 年龄  \* @param sex 性别  \* @param grade 年级  \* @param birthday 生日  \* @param dept 学院  \* @param telphone 电话  \* @param qq qq  \* @return 消息  \*/* @RequestMapping(value = "/students", method = RequestMethod.*PUT*)  @ResponseBody  public Msg insertStudent(Integer id,  String name,  Integer age,  Integer sex,  Integer grade,  String birthday,  Integer dept,  String telphone,  String qq){  Student student = new Student();  student.setId(id);  student.setAge(age);  student.setBirthday(TimeUtils.*castDateStringToDateTypeYMD*(birthday));  student.setDept(dept);  student.setGrade(grade);  student.setSex(sex);  student.setName(name);  student.setTelphone(telphone);  student.setQq(qq);  System.*out*.println(student);  studentService.insertStudent(student);  return Msg.*success*();  }    */\*\*  \* 获取最高学生序号，用于添加学生时候的学号填充  \* @return 消息  \*/* @RequestMapping(value = "/maxid", method = RequestMethod.*GET*)  @ResponseBody  public Msg getStudentMaxId(){  Integer id = studentService.getStudentMaxId();  return Msg.*success*().add("id", id);  } }  */\*\*  \* @Author: Yan Jingcun  \* @Date: 2021/5/27  \* @Description:  \* @Version: 1.0  \*/* package com.jancoyan.lab03.controller;  import com.jancoyan.lab03.pojo.Department; import com.jancoyan.lab03.service.DepartmentService; import com.jancoyan.lab03.utils.Msg; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.ResponseBody;  import java.util.List;  @Controller public class DepartmentController {   @Autowired  DepartmentService departmentService;   */\*\*  \* 获取所有部门用于填充下拉列表  \* @return 消息  \*/* @RequestMapping(value = "/depts", method = RequestMethod.*GET*)  @ResponseBody  public Msg getDepts(){  List<Department> list = departmentService.getAll();  return Msg.*success*().add("list", list);  }   }  */\*\*  \* @Author: Yan Jingcun  \* @Date: 2021/5/27  \* @Description:  \* @Version: 1.0  \*/*  package com.jancoyan.lab03.mapper;  import com.jancoyan.lab03.pojo.Department; import org.apache.ibatis.annotations.Select; import org.springframework.stereotype.Repository;  import java.util.List;  @Repository public interface DepartmentMapper {   */\*\*  \* 获取所有院系  \* @return  \*/* @Select("select \* from tbl\_dept")  List<Department> getAll();  }  */\*\*  \* @Author: Yan Jingcun  \* @Date: 2021/5/27  \* @Description:  \* @Version: 1.0  \*/*  package com.jancoyan.lab03.mapper;  import com.jancoyan.lab03.pojo.Student; import org.apache.ibatis.annotations.\*; import org.springframework.stereotype.Repository;  import java.util.Date; import java.util.List;  @Repository public interface StudentMapper {   *// 查询所有学生* @Select("select \* from tbl\_student")  List<Student> findAll();   *// 查询最大id* @Select("select max(id) from tbl\_student")  Integer getStudentMaxId();   *// 删除学生* @Delete("delete from tbl\_student where id = #{id}")  boolean deleteStudentByPrimaryKey(@Param("id") Integer id);    *// 插入学生* @Insert("insert into tbl\_student values(#{id}, #{name}, #{age}, #{sex}, #{grade}, " +  "#{birthday}, #{dept}, #{telphone}, #{qq})")  void insertStudent(@Param("id") Integer id,  @Param("name") String name,  @Param("age") Integer age,  @Param("sex")Integer sex,  @Param("grade")Integer grade,  @Param("birthday")Date birthday,  @Param("dept")Integer dept,  @Param("telphone")String telphone,  @Param("qq")String qq);  *// 更新学生* @Update("update tbl\_student set name=#{name}, age=#{age}, sex=#{sex}, grade=#{grade}, " +  "birthday=#{birthday}, dept=#{dept}, telphone=#{telphone}, qq=#{qq} where id = #{id}")  void updateStudent(@Param("id") Integer id,  @Param("name") String name,  @Param("age") Integer age,  @Param("sex")Integer sex,  @Param("grade")Integer grade,  @Param("birthday")Date birthday,  @Param("dept")Integer dept,  @Param("telphone")String telphone,  @Param("qq")String qq);  }  <!DOCTYPE html> <html> <head>  <meta charset="UTF-8">  <title>lab03</title>  <script src="../layui/layui.js"></script>  <script src="../js/index.js"></script>  <link rel="stylesheet" href="../css/index.css">  <link rel="stylesheet" href="../layui/css/layui.css"> </head> <style>  body{  text-align: center;  }  h1{  margin: 20px 0;  }  .main{  margin: 30px auto;  width: 85%;  }  </style> <body>  <h1>SpringBoot + LayUI 简易学生信息管理系统</h1>  <div class="main">  <table id="student-list" class="layui-hide" lay-filter="student-list"></table> </div>  <div class="layui-btn-group demoTable">  <button class="layui-btn" data-type="addStudent">添加学生</button>  <button class="layui-btn layui-btn-danger" data-type="batchDelete">批量删除</button> </div>  *<!-- 添加、编辑、查看的模态框 -->* <div id="myModal" class="modal">  <div class="modal-content">  <fieldset class="layui-elem-field layui-field-title" style="margin-top: 20px;">  <legend>标题</legend>  </fieldset>  <form class="layui-form" action="" lay-filter="information" method="post">  <div class="layui-form-item">  <label class="layui-form-label">id</label>  <div class="layui-input-block">  <input id="modal-student-id" type="text" name="id" autocomplete="off"  placeholder="不可编辑" class="layui-input" disabled="disabled"  lay-verify="required">  </div>  </div>   <div class="layui-form-item">  <label class="layui-form-label">姓名</label>  <div class="layui-input-block">  <input id="modal-student-name" type="text" name="name" autocomplete="off"  placeholder="请输入姓名" class="layui-input" lay-verify="required">  </div>  </div>   <div class="layui-form-item">  <label class="layui-form-label">年级</label>  <div class="layui-input-block">  <select id="modal-student-grade" name="grade" lay-filter="grade"  lay-verify="required">  <option value="" selected></option>  <option value="1">1</option>  <option value="2">2</option>  <option value="3">3</option>  <option value="4">4</option>  <option value="5">5</option>  </select>  </div>  </div>   <div class="layui-form-item">  <label class="layui-form-label">学院</label>  <div class="layui-input-block">  <select id="modal-dept-list" name="dept" lay-filter="dept"  lay-verify="required">  <option value="" selected></option>  </select>  </div>  </div>   <div class="layui-form-item">  <label class="layui-form-label">性别</label>  <div class="layui-input-block">  <select id="modal-student-sex" name="sex" lay-filter="sex"  lay-verify="required">  <option value="" selected></option>  <option value="0">女</option>  <option value="1">男</option>  </select>  </div>  </div>   <div class="layui-form-item" style="float: left">  <div class="layui-inline">  <label class="layui-form-label">出生日期</label>  <div class="layui-input-block">  <input type="text" class="layui-input" id="modal-student-birthday"  name="birthday" placeholder="yyyy-MM-dd" style="width: 350px" lay-verify="required">  </div>  </div>  </div>   <div class="layui-form-item">  <label class="layui-form-label">QQ号</label>  <div class="layui-input-block">  <input id="modal-student-qq" type="text" name="qq" autocomplete="off"  placeholder="qq号" class="layui-input" lay-verify="qq">  </div>  </div>   <div class="layui-form-item">  <label class="layui-form-label">手机号</label>  <div class="layui-input-block">  <input id="modal-student-tel" type="text" name="telphone" lay-verify="phone"  autocomplete="off"  placeholder="手机号" class="layui-input">  </div>  </div>   <div class="layui-form-item">  <div id="modal-buttons" class="layui-input-block">  <button id="update-btn" type="submit" class="layui-btn" lay-submit  lay-filter="update">修改  </button>  <button id="insert-btn" type="submit" class="layui-btn" lay-submit  lay-filter="insert">添加  </button>  <button type="button" id="return-to-index"  class="layui-btn layui-btn-primary">返回</button>  </div>  </div>  </form>  </div> </div>  <script type="text/html" id="toolBar">  <a class="layui-btn layui-btn-primary layui-btn-xs" lay-event="detail">查看</a>  <a class="layui-btn layui-btn-xs" lay-event="edit">编辑</a>  <a class="layui-btn layui-btn-danger layui-btn-xs" lay-event="del">删除</a> </script>  </body> </html>  // index.js var $ = layui.$;  $(function () {  *// 关闭注册模态框* $("#return-to-index").click(function(){  $("#myModal").css("display", "none");  *// 清空模态框  toggleModal*(0);  });  *// 渲染日期组件* layui.use('laydate', function(){  var laydate = layui.laydate;  *//执行一个laydate实例* laydate.render({  elem: '#modal-student-birthday' *//指定元素* });  });  layui.form.render('select');  })  layui.use('table', function(){  var table = layui.table;   *// 监听表格复选框选择* table.on('checkbox(student-list)', function(obj){  console.log(obj)  });   *//查看、修改、删除  //这一部分是点击查看、修改、删除之后要进行的操作  //而不是点击模态框上的按钮之后进行的操作  //关于表单的提交会在下面用 form.on 来写* table.on('tool(student-list)', function(obj){  var data = obj.data;  if(obj.event === 'detail'){  *// 查看* $("#myModal legend").text("查看");  *// 筛选按钮* $("#update-btn").css("display", "none");  $("#insert-btn").css("display", "none");  *// 展示* $("#myModal").css("display", "block");  form = layui.form;  *// 填充信息* form.val("information", {  "id": data.id  ,"name": data.name  , "age": data.age  , "dept": *get\_dept\_id\_by\_name*(data.dept)  , "grade": data.grade  ,"sex": data.sex === "男" ? 1 : 0  ,"qq": data.qq  , "telphone": data.telphone  });  layui.laydate.render({  elem: '#modal-student-birthday'  ,value: *dateFormat*(data.birthday)  ,isInitValue: true *//是否允许填充初始值，默认为 true* });  *// 锁住信息  toggleModal*(1);  } else if(obj.event === 'del'){  layer.confirm('真的要删除' + obj.data.id + " " + obj.data.name + '么', function(index){  *// console.log(obj);* obj.del();  layer.close(index);  *// 向服务端发送删除指令* var $ = layui.$;  $.ajax({  url: "students",  type: "DELETE",  data: "id=" + obj.data.id,  success: function (){  layer.msg('删除成功');  }  });  });  } else if(obj.event === 'edit'){  *//修改* $("#myModal legend").text("修改");  *// 筛选按钮* $("#update-btn").css("display", "inline");  $("#insert-btn").css("display", "none");  $("#myModal").css("display", "block");  *toggleModal*(2);  $("#modal-student-id").attr("disabled", "disabled");  *//给表单赋值* form = layui.form;  form.val(`information`, {  "id": data.id  ,"name": data.name  , "age": data.age  , "dept": *get\_dept\_id\_by\_name*(data.dept)  , "grade": data.grade  ,"sex": data.sex === "男" ? 1 : 0  ,"qq": data.qq  , "telphone": data.telphone  });  layui.laydate.render({  elem: '#modal-student-birthday'  ,value: *dateFormat*(data.birthday)  ,isInitValue: true *//是否允许填充初始值，默认为 true* });  }  });   *// 数据的批量删除* var $ = layui.$, active = {  batchDelete: function(){ *//获取选中数据* var checkStatus = table.checkStatus('student-table')  ,data = checkStatus.data;  if (data.length === 0){  layer.msg("当前没有选中数据");  return;  }  layer.confirm('真的要删除选中的行么', function() {  *// 已经获得到了data，点击了确认，下一步是执行删除* var $ = layui.$;  *// console.log(data);* $.each(data, function (index, item) {  $.ajax({  url: "students",  type: "DELETE",  data: "id=" + item.id,  success: function () {  *// 删除完毕之后暂时不执行任何操作* }  })  })  layer.msg("删除成功");  table.reload('student-table', {  url: 'students'  });  })  }  ,addStudent:function () {  *// 添加学生  toggleModal*(0);  *toggleModal*(2);  *// 获取最大id* $.ajax({  url: "maxid",  type:"GET",  success: function (result){  *// 当前id为最大 id + 1* var id = result.extend.id + 1;  *// id 填充* $("#modal-student-id").val(id).attr("disabled", "disabled");  }  });  $("#myModal legend").text("添加学生");  *// 筛选按钮* $("#update-btn").css("display", "none");  $("#insert-btn").css("display", "inline");  $("#myModal").css("display", "block");  *// 发送ajax请求添加学生  // 这部分的代码写在下面了* }  };   $('.demoTable .layui-btn').on('click', function(){  var type = $(this).data('type');  active[type] ? active[type].call(this) : '';  });   *// 表格初始化渲染* table.render({  elem: '#student-list' *// 挂载点* ,id: 'student-table' *// 表格id* ,height: 580 *// 表格高度* ,url: '/students' *//数据接口* ,toolbar: '#toolbarDemo' *//开启头部工具栏，并为其绑定左侧模板* ,skin: 'line' *//行边框风格* ,even: true *//开启隔行背景* ,height: 'full-200' *// 页面的最大高度 - 100 （自适应）* ,defaultToolbar: ['filter', 'exports', 'print', { *//自定义头部工具栏右侧图标。如无需自定义，去除该参数即可* title: '提示'  ,layEvent: 'LAYTABLE\_TIPS'  ,icon: 'layui-icon-tips'  }]  ,parseData: function(res){ *// 数据源* var list = res.extend.data.list;  *//进行数据清洗* $.each(list, function (index, item) {  item.birthday = *dateFormat*(item.birthday);  item.dept = *get\_dept\_name\_by\_id*(item.dept);  item.sex = item.sex === 1 ? "男" : "女";  })  *//res 即为原始返回的数据* return {  "code": res.code, *//解析接口状态* "msg": res.msg, *//解析提示文本* "count": res.extend.count, *//解析数据长度* "data": res.extend.data.list *//解析数据列表* };  }  ,page: { *// 分页* groups: 5 *//显示 10 个连续页码* }  ,cols: [[ *//表头* {type:'checkbox'}  ,{field: 'id', title: 'ID', width:80, sort: true}  ,{field: 'name', title: '姓名', width:80, sort: true}  ,{field: 'age', title: '年龄', width:80, sort: true}  ,{field: 'sex', title: '性别', width:80}  ,{field: 'grade', title: '年级', width: 100, sort: true}  ,{field: 'birthday', title: '生日', width: 150}  ,{field: 'dept', title: '院系', width: 180}  ,{field: 'telphone', title: '电话', width: 160}  ,{field: 'qq', title: 'QQ', width: 135}  ,{fixed: 'right', title:'操作', toolbar: '#toolBar', width:170}  ]]  }); });   var form = layui.form; *// 提交修改* form.on('submit(update)', function(data){  *// 获取数据* var post = data.field;  post.birthday = $("#modal-student-birthday").val();  post.age = 2021 - Number(post.birthday.substr(0, 4));  *// 发送请求  // console.log(post) //当前容器的全部表单字段，名值对形式：{name: value}* $.ajax({  url: "students",  type: "POST",  data: post,  success: function () {  *// 关闭模态框* $("#myModal").css("display", "none");  *// 清空模态框信息  toggleModal*(0);  *// 提示用户修改成功* layer.msg("修改成功");  }  });  *// 重载表格* layui.table.reload('student-table', {  url: 'students'  });  return false; });  *// 提交插入数据* form.on('submit(insert)', function(data){  *// 获取数据* var post = data.field;  post.birthday = $("#modal-student-birthday").val();  post.age = 2021 - Number(post.birthday.substr(0, 4));  console.log(post) *//当前容器的全部表单字段，名值对形式：{name: value}* $.ajax({  url: "students",  type: "PUT",  data: post,  success: function () {  *// 关闭模态框* $("#myModal").css("display", "none");  *// 清空模态框信息  toggleModal*(0);  *// 提示用户修改成功* layer.msg("插入成功");  }  })  *// 重载表格* layui.table.reload('student-table', {  url: 'students'  });  return false; });  form.verify({  *// qq* qq: function(value, item){ *//value：表单的值、item：表单的DOM对象* if(value.length < 5 || value.length > 12){  return '请输入正确的qq号格式';  }  }  *//学院* ,dept: function (value, item) {  if (value){   }  }  *// 性别* ,sex: function (value, item){  if (value === -1){  return false;  }  }   });  */\*\*  \* 约定  \* 0 - 清空  \* 1 - 置为 disable  \* 2 - 置为 可用  \*/* function *toggleModal*(nextStatus) {  *// 转换模态框的状态* if (0 === nextStatus){  *// 清空模态框* $("#myModal input").val("");  $(".default-selected").attr("selected", "selected");  } else if (1 === nextStatus) {  *// 置为disable* $("#myModal input").attr("disabled", "disabled");  } else if (2 === nextStatus){  *// 置为可用的* $("#myModal input").removeAttr("disabled");  }  }   */\*\*  \* 将时间戳转化为标准时间  \* @param date 时间戳  \* @returns {string} 标准时间字符串  \*/* function *dateFormat*(date) {  var s = new Date(date)  var y = s.getFullYear()  var m = (s.getMonth() + 1) < 10 ? '0' + (s.getMonth() + 1) : (s.getMonth() + 1)  var dd = s.getDate() < 10 ? '0' + s.getDate() : s.getDate()  var enddate = y + '-' + m + '-' + dd  return enddate } |

运行截图：

|  |
| --- |
| 添加学生：  添加学生要求每一项都要填写，并且id会自动填充，不可更改  其中年级、学院、性别为下拉框，出生日期为选择框，以下是演示        **查看学生信息**    **修改**      **单个删除**      **批量删除**    **信息筛选** |
| **按照年龄排序** |

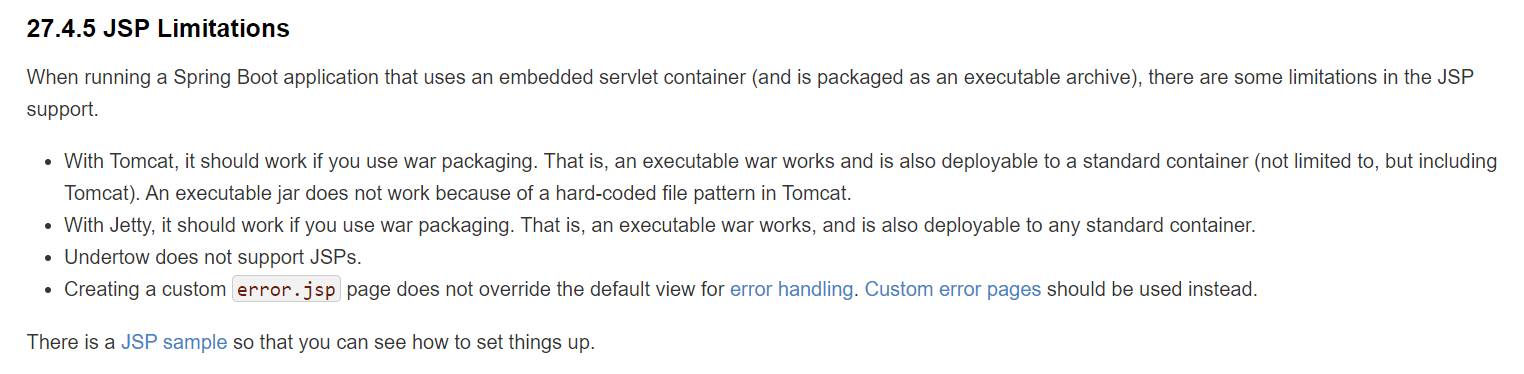
四、心得体会

最近刚好在学SpringBoot，又刚好了解了Layui前端框架，想着做个小项目练练手，正好遇上了这个实验，就借此机会写了个小系统。

整个数据库只有两张表，因此数据库设计不是难点，写CRUD接口也不是难点，写业务逻辑也不是难点，前后端连接调试也不难，从开始写到能在本地正常运行只用了大概10个小时。

但是把项目部署到服务器用了大概5个小时。

我最初的想法是SpringBoot 整合Jsp，SpringBoot默认打jar包，而不是war包，我用jsp整合（加载依赖、配置视图解析器、写controller）之后在本地能跑，但是放在服务器上之后一直跑不动，在查阅资料的过程中我看到SpringBoot官方文档这样说



SpringBoot项目可以打jar包或者war包，官方建议打jar包，但是如果整合了jsp，就只能打war包。于是将maven打包方式改为war包（屏蔽内置tomcat、重写SpringApplicationBuilder），部署到服务器之后可以访问静态资源，但是不能发起request请求，不能调用Controller，数据库连接没有问题，应该是因为Controller无法接受请求，我猜想应该是服务器相关的问题。

但是临近期末考试，没有很多时间查进行更多尝试了。

于是将jsp文件调整为html文件，放在资源路径下，删除jsp相关依赖，恢复内置tomcat，删除jsp相关的视图解析器，不再整合jsp，本地可以运行之后，mvn clear ， package生成jar包，上传到服务器，启动项目，访问<http://39.103.184.190:8889/>，完美运行。

（其实还想做查找功能来着，时间有限，就没做。）

五、自我评价

**分数自评**

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **评分标准** | **总分** | **自评分** |
| 1 | 完成学生系统的查看、增加、删除和修改功能 | 6 | 6 |
| 2 | 使用javascript进行了数据验证 | 2 | 2 |
| 3 | 代码格式有必要的缩进和空格，可读性强 | 1 | 1 |
| 4 | 使用CSS，页面美观 | 1 | 1 |
| 合计 |  | 10 | 10 |