第三章 SpringMVC(三)

-Ajax异步、统一全局异常处理、RESTFul编程、拦截器2018/11/14 [泽林.王峰]

授课目标

1、 上章回顾

2、 SpringMVC-Ajax异步处理

3、 SpringMVC全局异常统一处理

4、 SpringMVC-RESTful编程实践

5、 SpringMVC拦截器实战

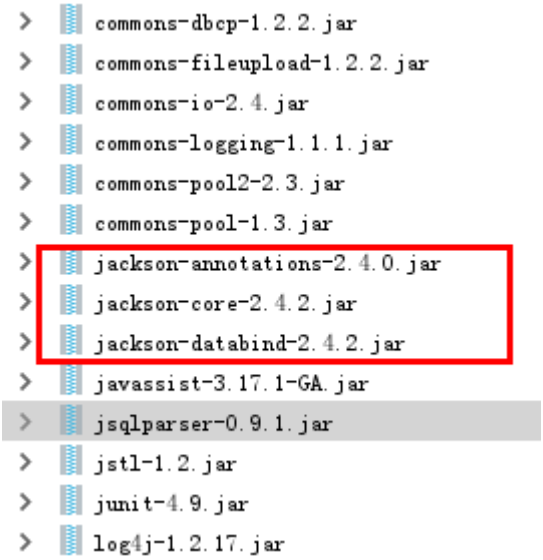
授课内容

1、 上章回顾

2、 Spring-Ajax异步处理

2.1) 方式二：

只需要添加如下三个jar包就可以了。



2.1.1) 修改resource/spring/applicationContext-mvc.xml文件

```
1  <?xml version="1.0" encoding="UTF-8"?>
```

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd">
    <!--1、配置springmvc的包扫描-->
    <context:component-scan base-package="com.zelin.web.controller"/>
    <mvc:annotation-driven conversion-service="conversionService"/>
    <!--2.添加视图解析器-->
    <!--3.配置转换器-->
    <bean id="conversionService"
class="org.springframework.format.support.FormattingConversionServiceFactoryBean">
        <property name="converters">
            <list>
                <bean class="com.zelin.web.converter.DateConverter"/>
            </list>
        </property>
    </bean>
</beans>
```

2.1.2) 前端页面index.html编写：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>学生列表</title>
    <link rel="stylesheet" href="bootstrap-3.3.7/css/bootstrap.min.css">
    <script src="bootstrap-3.3.7/js/jquery.min.js"></script>
    <script src="bootstrap-3.3.7/js/bootstrap.min.js"></script>
    <style>
        .table {
            text-align: center;
        }

        .container {
            margin-top: 20px;
        }
    </style>
</head>
<body>
<div class="container">
    <div class="panel panel-primary">
        <div class="panel-heading ">
            <h4 class="glyphicon glyphicon-user">学生管理系统</h4>
        </div>

        <table class="table table-bordered table-hover">
```

```
            <tr class="bg-info">
                <td>学号</td>
                <td>姓名</td>
                <td>性别</td>
                <td>年龄</td>
                <td>住址</td>
                <td>生日</td>
                <td>所在班级</td>
                <td>操作</td>
            </tr>
            <tbody id="tb"></tbody>

        </table>
        <div class="panel-footer clearfix" style="text-align:right;padding:10px;">
            <a href="javascript:add()" class="btn btn-sm btn-warning pull-left glyphicon glyphicon-plus-sign"
               id="btn_add">添加学生</a>
            泽林信息版权所有 2000-2018.
        </div>
    </div>

</div>

<!-- 模态框（添加或修改学生） -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span>
                </button>
                <h4 class="modal-title" id="myModalLabel">编辑学生信息</h4>
            </div>
            <div class="modal-body">
                <form class="form-horizontal" id="form1">
                    <input type="hidden" name="sid" id="sid">
                    <div class="form-group">
                        <label class="col-sm-2 control-label">学生姓名:</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" placeholder="输入姓名" name="sname" id='sname'>
                        </div>
                    </div>
                    <div class="form-group">
                        <label class="col-sm-2 control-label">性别：</label>
                        <div class="radio">
                            <label>
                                <input type="radio" name="sex" value="男" >男
                            </label>
                            <label>
                                <input type="radio" name="sex" value="女">女

                            </label>
```

```
                        </div>
                    </div>
                    <div class="form-group">
                        <label class="col-sm-2 control-label">学生年龄:</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" placeholder="输入年龄" name="age" id="age">
                        </div>
                    </div>
                    <div class="form-group">
                        <label class="col-sm-2 control-label">学生住址:</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" placeholder="输入住址" name="addr" id="addr">
                        </div>
                    </div>
                    <div class="form-group">
                        <label class="col-sm-2 control-label">生日:</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" placeholder="输入生日" name="birth" id="birth">
                        </div>
                    </div>
                    <div class="form-group">
                        <label class="col-sm-2 control-label">班级:</label>
                        <div class="col-sm-10">
                            <select class="form-control" name="cid" id="cid">

                            </select>
                        </div>
                    </div>

                </form>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">关闭</button>
                <button type="button" class="btn btn-primary" id="btn_save">保存</button>
            </div>
        </div>
    </div>
</div>
</body>
</html>
<script>
    $(function () {
        //刷新列表
        reloadList();
    })

    //刷新列表

    function reloadList() {
```

```
        //1、发出异步请求加载所有的学生列表
        $.post(
            'student/list.action',
            function (data) {
                var info = "";
                $.each(data, function (i, v) {
                    var dt = new Date(v.birth);
                    var dtStr = dt.getFullYear() + "-" + (dt.getMonth() + 1) + "-" + dt.getDate();
                    info += "<tr>";
                    info += "<td>" + v.sid + "</td>";
                    info += "<td>" + v.sname + "</td>";
                    info += "<td>" + v.sex + "</td>";
                    info += "<td>" + v.age + "</td>";
                    info += "<td>" + v.addr + "</td>";
                    info += "<td>" + dtStr + "</td>";
                    info += "<td>" + v.classes.cname + "</td>";
                    info += "<td>";
                    info += "<a href='javascript:;' onclick='updateStudent("+v.sid+")' class='btn btn-sm btn-info glyphicon glyphicon-pencil'>修改</a> "
                    info += "<a href='javascript:;' class='btn btn-sm btn-danger glyphicon glyphicon-trash' onclick='deleStudent("+v.sid+")'>删除</a>"
                    info += "</td>";
                    info += "</tr>";
                })
                //放上面的字符串放到tbody中
                $("#tb").html(info);
            }, "json"
        );
        //2.发出异步请求，得到所有的班级列表
        $.post(
            'classes/list.action',
            function (data) {
                var classesInfo = "";
                $(data).each(function (i, v) {
                    classesInfo += "<option value='" + v.cid + "'>" + v.cname + "</option>";
                })
                $("#cid").html(classesInfo);
            }, "json")
    }

    //点击“添加学生”按钮
    $("#btn_add").click(function () {
        //1.显示模态对话框
        $('#myModal').modal('show');
        //2.清空表单
        $('#form1')[0].reset();
    })
    //点击”保存“按钮
    $("#btn_save").click(function () {

        //判断是添加还是修改操作？（因为此二者共用一个对话框）
```

```
171         if($('#sid').val() ){         //做修改操作
172             operation('student/update.action');
173         }else{                                  //做添加操作
174             operation('student/add.action');
175         }
176 
177     })
178     function operation(url){
179         //1.取出表单中的值
180         var studentEntity = $("#form1").serialize();
181         //2.将数据异步提交到数据库中
182         $.post(
183             url,
184             studentEntity,
185             function (data) {
186                 if(data.success){
187                     //刷新列表
188                     reloadList();
189                 }else{
190                     alert(data.message);
191                 }
192                 //关闭对话框
193                 $('#myModal').modal('hide');
194             }, "json")
195     }
196     //删除学生
197     function deleStudent(v){
198         $.post(
199             "student/delete.action&sid="+v,
200             function(data){
201                 if(data.success){
202                     //刷新列表
203                     reloadList();
204                 }else{
205                     alert(data.message);
206                 }
207             },"json"
208         )
209     }
210     //修改学生
211     function  updateStudent(v) {
212         $('#form1')[0].reset();
213         //1.根据sid查询出这条记录（异步查询）
214         $.post(
215             "student/findOne.action&sid="+v,
216             function(data){
217                 //将得到的学生对象显示到表单中
218                 $("#sname").val(data.sname);
219                 $("#age").val(data.age);
220                 $("#addr").val(data.addr);
221                 $("#birth").val(data.birth);
222                 $("#sname").val(data.sname);
223                 $('#cid').val(data.cid);
```

```
                //设置默认选择性别
                $('input[name=sex]').each(function(i,v){
                    $(this).attr("checked",$(v).val() == data.sex);
                })
            },'json'
        )
        //2.显示对话框，并将当前这条记录显示到表单中
        $('#myModal').modal('show');
        //3.为表单中的隐藏域字段赋值
        $("#sid").val(v);
    }
</script>
```

2.1.3 ）后台页面StudentController.java

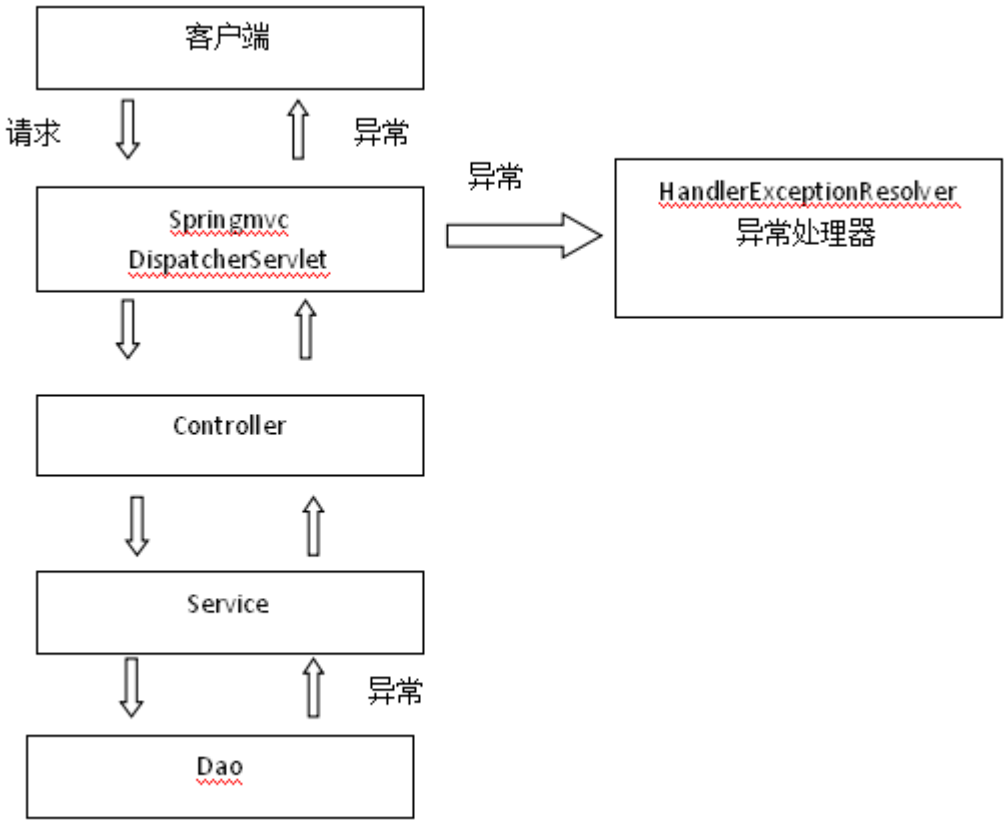
```
@RestController
@RequestMapping("/student")
public class StudentController {
    @Autowired
    private StudentService studentService;
    //查询所有的学生
    @RequestMapping("/list")
    public List<Student> findAll(){
        return studentService.findAll();
    }
    //添加学生
    @RequestMapping("/add")
    public Result add(Student student){
        try {
            studentService.add(student);
            return new Result(true,"添加成功");
        }catch (Exception e){
            e.printStackTrace();
            return new Result(false,"添加失败");
        }
    }
}

```

3、 SpringMVC全局异常统一处理

3.1 ）SpringMVC中异常处理原理：

系统中异常包括两类：预期异常和运行时异常RuntimeException，前者通过捕获异常从而获取异常信息，后者主要通过规范代码开发、测试通过手段减少运行时异常的发生。

系统的dao、service、controller出现都通过throws Exception向上抛出，最后由springmvc前端控制器交由异常处理器进行异常处理，如下图：



3.2)统一异常的处理步骤：

3.2.1)自定义异常：

```
/**
 * 自定义异常
 */
public class MyException extends  RuntimeException {
    private String message;

    @Override
    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public MyException(String message) {
        this.message = message;
    }
}
```

3.2.2)自定义统一异常处理器：

```java
@Component
public class CustomHandlerException  implements HandlerExceptionResolver{
    @Override
    public ModelAndView resolveException(HttpServletRequest request, HttpServletResponse response, Object o, Exception e) {
        //1.如果当前获取到的异常对象e就是MyException，则可以进行
        MyException exception = null;
        if(e instanceof  MyException){
            exception = (MyException) e;
        }
        //2.取得异常信息
        String message = "";
        if(exception != null){
            message = exception.getMessage();
        }else{
            message = "对不起，出现了未知异常！";
        }
        return new ModelAndView("error","message",message);
    }
}

```

3.2.3)在/WEB-INF/jsp/下添加error.jsp文件：

```jsp
<%--
  Created by IntelliJ IDEA.
  User: Administrator
  Date: 2018/11/14
  Time: 15:28
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>错误处理</title>
    <link rel="stylesheet" href="${pageContext.request.contextPath}/bootstrap-3.3.7/css/bootstrap.min.css">
    <script src="${pageContext.request.contextPath}/bootstrap-3.3.7/js/jquery.min.js"></script>
    <script src="${pageContext.request.contextPath}/bootstrap-3.3.7/js/bootstrap.min.js"></script>
</head>
<body>
<div class="container">
    <div class="panel panel-primary">
        <div class="panel-heading ">
            <h4 class="glyphicon glyphicon-user">学生管理系统</h4>
        </div>
        <div class="panel-body">
            ${message}
        </div>
        <div class="panel-footer" style="text-align: right;">
```

```
            泽林信息版权所有 2000-2018.
        </div>
    </div>
</div>
</body>
</html>
```

3.2.4)在resource/spring/applicationContext-mvc.xml中添加视图解析器及扫描包：

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd">
    <!--1、配置springmvc的包扫描-->
    <context:component-scan base-
package="com.zelin.web.controller,com.zelin.web.exception"/>
    <mvc:annotation-driven conversion-service="conversionService"/>
    <!--2.添加视图解析器-->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/"/>
        <property name="suffix" value=".jsp"/>
    </bean>
    <!--3.配置转换器-->
    <bean id="conversionService"
class="org.springframework.format.support.FormattingConversionServiceFactoryBean">
        <property name="converters">
            <list>
                <bean class="com.zelin.web.converter.DateConverter"/>
            </list>
        </property>
    </bean>
</beans>
```

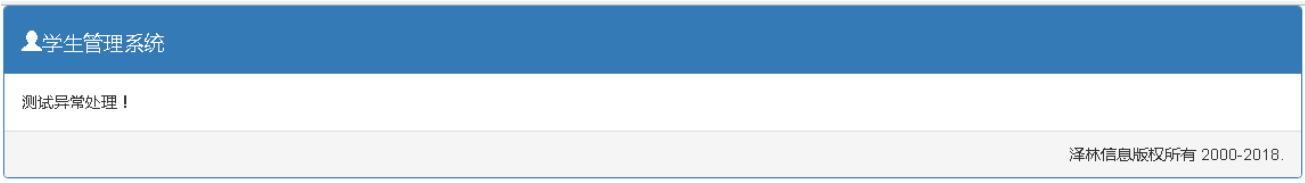
3.2.5)修改index.html页面

```
 <div class="panel-footer clearfix" style="text-align:right;padding:10px;">
            <a href="javascript:;" class="btn btn-sm btn-warning pull-left glyphicon glyphicon-plus-sign"
               id="btn_add">添加学生</a>
            <a href="javascript:;" class="btn btn-sm btn-success pull-left glyphicon glyphicon-plus-sign"
               id="btn_exception">测试统一异常处理</a>
            泽林信息版权所有 2000-2018.
        </div>

//测试异常处理
    $("#btn_exception").click(function(){
        location.href="student/exceHandler.action"
    })
```

3.2.6)后台

```
@RequestMapping("/exceHandler")
    public void exceHandler(){
       throw new MyException("测试异常处理！");
    }
```

3.2.7)页面展示



4、 SpringMVC-RESTful编程实践

4.1)修改web.xml文件：

```
<!--配置springmvc-->
    <servlet>
        <servlet-name>springmvc</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath*:spring/applicationContext-mvc.xml</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
```

4.2)排除静态资源,修改resource/spring/applictionContext-mvc.xml

```
<!--因为要使用restful模式，要添加对静态资源的映射 -->
 <mvc:resources mapping="/bootstrap-3.3.7/**" location="/bootstrap-3.3.7/"/>
 <mvc:resources mapping="/WEB-INF/html/**" location="/WEB-INF/html/"/>
 <mvc:annotation-driven conversion-service="conversionService"/>
```

4.3)修改页面/WEB-INF/html/student/list.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>学生列表</title>
    <link rel="stylesheet" href="../../../bootstrap-3.3.7/css/bootstrap.min.css">
    <script src="../../../bootstrap-3.3.7/js/jquery.min.js"></script>
    <script src="../../../bootstrap-3.3.7/js/bootstrap.min.js"></script>
    <style>
        .table {
            text-align: center;
        }

        .container {
            margin-top: 20px;
        }
    </style>
</head>
<body>
<div class="container">
    <div class="panel panel-primary">
        <div class="panel-heading ">
            <h4 class="glyphicon glyphicon-user">学生管理系统</h4>
        </div>
        <table class="table table-bordered table-hover">
            <tr class="bg-info">
                <td>学号</td>
                <td>姓名</td>
                <td>性别</td>
                <td>年龄</td>
                <td>住址</td>
                <td>生日</td>
                <td>所在班级</td>
                <td>操作</td>
            </tr>
            <tbody id="tb"></tbody>

        </table>
        <div class="panel-footer clearfix" style="text-align:right;padding:10px;">
        <a href="javascript:;" class="btn btn-sm btn-warning pull-left glyphicon glyphicon-plus-sign"

           id="btn_add">添加学生</a>
```

```
        <a href="javascript:;" class="btn btn-sm btn-success pull-left glyphicon glyphicon-plus-sign"
           id="btn_exception">测试统一异常处理</a>
        泽林信息版权所有 2000-2018.
    </div>
    </div>

</div>

<!-- 模态框（添加或修改学生） -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
    <div class="modal-dialog" role="document">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="true">&times;</span>
                </button>
                <h4 class="modal-title" id="myModalLabel">编辑学生信息</h4>
            </div>
            <div class="modal-body">
                <form class="form-horizontal" id="form1">
                    <input type="hidden" name="sid" id="sid">
                    <div class="form-group">
                        <label class="col-sm-2 control-label">学生姓名:</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" placeholder="输入姓名" name="sname" id='sname'>
                        </div>
                    </div>
                    <div class="form-group">
                        <label class="col-sm-2 control-label">性别：</label>
                        <div class="radio">
                            <label>
                                <input type="radio" name="sex" value="男" >男
                            </label>
                            <label>
                                <input type="radio" name="sex" value="女">女
                            </label>
                        </div>
                    </div>
                    <div class="form-group">
                        <label class="col-sm-2 control-label">学生年龄:</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" placeholder="输入年龄" name="age" id="age">
                        </div>
                    </div>
                    <div class="form-group">
                        <label class="col-sm-2 control-label">学生住址:</label>
                        <div class="col-sm-10">


                            <input type="text" class="form-control" placeholder="输
```

```
入住址" name="addr" id="addr">
                        </div>
                    </div>
                    <div class="form-group">
                        <label class="col-sm-2 control-label">生日:</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" placeholder="输入生日" name="birth" id="birth">
                        </div>
                    </div>
                    <div class="form-group">
                        <label class="col-sm-2 control-label">班级:</label>
                        <div class="col-sm-10">
                            <select class="form-control" name="cid" id="cid">

                            </select>
                        </div>
                    </div>

                </form>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss="modal">关闭</button>
                <button type="button" class="btn btn-primary" id="btn_save">保存</button>
            </div>
        </div>
    </div>
</div>
</body>
</html>
<script>
    $(function () {
        //刷新列表
        reloadList();
    })

    //刷新列表
    function reloadList() {
        //1、发出异步请求加载所有的学生列表
        $.post(
            'list',
            function (data) {
                var info = "";
                $.each(data, function (i, v) {
                    var dt = new Date(v.birth);
                    var dtStr = dt.getFullYear() + "-" + (dt.getMonth() + 1) + "-" + dt.getDate();
                    info += "<tr>";
                    info += "<td>" + v.sid + "</td>";
                    info += "<td>" + v.sname + "</td>";

                    info += "<td>" + v.sex + "</td>";
```

```
                    info += "<td>" + v.age + "</td>";
                    info += "<td>" + v.addr + "</td>";
                    info += "<td>" + dtStr + "</td>";
                    info += "<td>" + v.classes.cname + "</td>";
                    info += "<td>";
                    info += "<a href='javascript:;' onclick='updateStudent("+v.sid+")' class='btn btn-sm btn-info glyphicon glyphicon-pencil'>修改</a> "
                    info += "<a href='javascript:;' class='btn btn-sm btn-danger glyphicon glyphicon-trash' onclick='deleStudent("+v.sid+")'>删除</a>"
                    info += "</td>";
                    info += "</tr>";
                })
                //放上面的字符串放到tbody中
                $("#tb").html(info);
            }, "json"
        );
        //2.发出异步请求，得到所有的班级列表
        $.post(
            '../classes/list',
            function (data) {
                var classesInfo = "";
                $(data).each(function (i, v) {
                    classesInfo += "<option value='" + v.cid + "'>" + v.cname + "</option>";
                })
                $("#cid").html(classesInfo);
            }, "json")
    }

    //点击“添加学生”按钮
    $("#btn_add").click(function () {
        //1.显示模态对话框
        $('#myModal').modal('show');
        //2.清空表单
        $('#form1')[0].reset();
    })
    //测试异常处理
    $("#btn_exception").click(function(){
        location.href="student/exceHandler.action"
    })
    //点击”保存“按钮
    $("#btn_save").click(function () {
        //判断是添加还是修改操作？（因为此二者共用一个对话框）
        if($('#sid').val() ){        //做修改操作
            operation('student/update');
        }else{                              //做添加操作
            operation('student/add');
        }

    })
    function operation(url){

        //1.取出表单中的值
```

```
        var studentEntity = $("#form1").serialize();
        //2.将数据异步提交到数据库中
        $.post(
            url,
            studentEntity,
            function (data) {
                if(data.success){
                    //刷新列表
                    reloadList();
                }else{
                    alert(data.message);
                }
                //关闭对话框
                $('#myModal').modal('hide');
            }, "json")
    }
    //删除学生
    function deleStudent(v){
        $.post(
            "student/delete/"+v,
            function(data){
                if(data.success){
                    //刷新列表
                    reloadList();
                }else{
                    alert(data.message);
                }
            },"json"
        )
    }
    //修改学生
    function  updateStudent(v) {
        $('#form1')[0].reset();

        //1.根据sid查询出这条记录（异步查询）
        $.post(
            "findOne/"+v,
            function(data){
                var dt = new Date(data.birth);
                var dtStr = dt.getFullYear() + "-" + (dt.getMonth() + 1) + "-" +
dt.getDate();
                //将得到的学生对象显示到表单中
                $("#sname").val(data.sname);
                $("#age").val(data.age);
                $("#addr").val(data.addr);
                $("#birth").val(dtStr);
                $("#sname").val(data.sname);
                $('#cid').val(data.cid);
                //设置默认选择性别
                $('input[name=sex]').each(function(i,v){
                    $(this).attr("checked",$(v).val() == data.sex);
                })

            },'json'
```

```
        )
        //2.显示对话框，并将当前这条记录显示到表单中
        $('#myModal').modal('show');
        //3.为表单中的隐藏域字段赋值
        $("#sid").val(v);
    }
</script>
```

4.4)在StudentController.java中添加处理器

```
@Controller
@RequestMapping("/student")
public class StudentController {
    @Autowired
    private StudentService studentService;

    //进行学生列表的首页
    @RequestMapping("/tolist")
    public String welcome(){
        return "student/list";
    }
    //根据学生编号查询学生
    @RequestMapping("/findOne/{sid}")
    @ResponseBody
    public Student findOne(@PathVariable()  String sid){
        return studentService.findOne(sid);
    }
    //查询所有的学生
    @RequestMapping("/list")
    @ResponseBody
    public List<Student> findAll(){
        return studentService.findAll();
    }
    //根据学号删除学生
    @RequestMapping("/delete/{sid}")
    @ResponseBody
    public Result delete(@PathVariable String sid){
        try{
            studentService.delete(sid);
            return new Result(true,"删除成功！");
        }catch (Exception e){
            e.printStackTrace();
            return new Result(false,"删除失败！");
        }

    }
    //添加学生
    @RequestMapping("/add")
    @ResponseBody
    public Result add(Student student){
        try {

            studentService.add(student);
```

```
            return new Result(true,"添加成功");
        }catch (Exception e){
            e.printStackTrace();
            return new Result(false,"添加失败");
        }
    }
    @RequestMapping("/exceHandler")
    public void exceHandler(){
       throw new MyException("测试异常处理！");
    }
}

```

4.5)演示略。

5、 SpringMVC拦截器实战

5.1）定义一个拦截器：

第一步：**实现HandlerInterceptor接口，实现其中的三个抽象方法：**

public class FirstInterceptor **implements** HandlerInterceptor {

//在执行处理器前调用

//返回值说明：true:代表可以调用下一个拦截器 false:代表不能再向下执行

@Override

public boolean preHandle(HttpServletRequest request, HttpServletResponse response,

Object arg2) **throws** Exception {

System.**out**.println("FirstInterceptor→preHandle(),执行时间是：" + **new** Date().getTime());

return true;

}

//在执行完处理器后调用，一般在此做收尾工作，比如，统一异常处理，也可以在此进行

@Override

public void afterCompletion(HttpServletRequest request, HttpServletResponse response,

Object arg2, Exception exception)

throws Exception {

System.**out**.println("FirstInterceptor→afterCompletion(),执行时间是：" + **new** Date().getTime());

}

//在执行处理器之后，在视图渲染前调用

@Override

public void postHandle(HttpServletRequest request, HttpServletResponse response,

Object arg2, ModelAndView mv)

throws Exception {

System.**out**.println("FirstInterceptor→postHandle(),执行时间是："+ **new** Date().getTime());

//取得模型和视图

String viewName = mv.getViewName(); //取得视图名称

List students = (List) mv.getModel().get("students");

System.**out**.println("视图名:" + viewName);

System.**out**.println("模型内容：" + students);

}

}

第二步：在**applicationContext-springmvc.xml中配置拦截器：**

mvc:interceptors

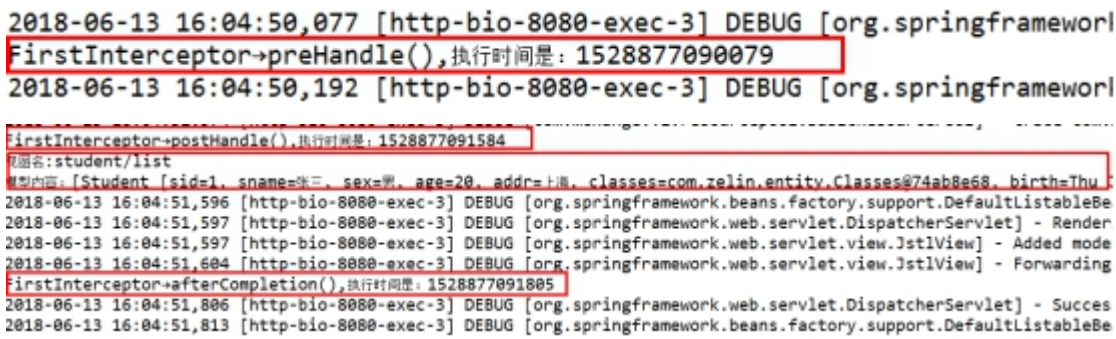
<!-- mvc:interceptor

<mvc:mapping path="/**"/>

/mvc:interceptor -->

/mvc:interceptors

第三步：测试拦截器：



5.2) 定义多个拦截器：

第一步：定义第二个拦截器

//第二种定义拦截器的方法

public class SecondInterceptor **extends HandlerInterceptorAdapter** {

@Override

```
public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler,
Exception ex)
throws Exception {
System.out.println("SecondInterceptor→afterCompletion()");
}
@Override
public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler,
ModelAndView modelAndView) throws Exception {
System.out.println("SecondInterceptor→postHandle()");
}
@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
throws Exception {
System.out.println("SecondInterceptor→preHandle()");
return true;
}
}
```

第二步：配置拦截器:(下面配置拦截器适用于所有控制器)

```
<!-- 配置拦截器 -->
<mvc:interceptors>
    <!-- 第一种配置方法：代表拦截所有的控制器 -->
    <!-- <mvc:interceptor>
        <mvc:mapping path="/**"/>
        <bean class="com.zelin.interceptor.FirstInterceptor"/>
    </mvc:interceptor> -->
    <!-- 第二种配置方法：可以直接定义在这里，会对所有的控制器起作用 -->
    <bean class="com.zelin.interceptor.FirstInterceptor"/>
    <bean class="com.zelin.interceptor.SecondInterceptor"/>
</mvc:interceptors>
```

第三步：测试多个拦截器执行顺序：

```
FirstInterceptor→preHandle(),执行时间是：1528877419874
SecondInterceptor→preHandle()

SecondInterceptor→postHandle()
FirstInterceptor→postHandle(),执行时间是：1528877420917
视图名:student/list
模型内容：[Student [sid=1, sname=张三, sex=男, age=20, addr=上海, classes=com.zelin.ent

SecondInterceptor→afterCompletion()
FirstInterceptor→afterCompletion(),执行时间是：1528877421067
```

小结：

1）在上面实验可以看出，拦截器的preHandler()方法根据定义的先后顺序执行。

2）拦截器的postHandler()方法及afterCompletion()都会根据定义的逆序执行。

5.3）用户登录-拦截实现：

第一步：定义login.jsp页面：

用户登录

输入用户名：

"提交"

第二步：定义一个登录拦截器：

```
//登录拦截器
public class LoginInterceptor extends HandlerInterceptorAdapter {
@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)
throws Exception {
HttpSession session = request.getSession();
//得到用户名
Object obj = session.getAttribute("username");
String username = null;
if(obj != null){
username = (String) obj;
}else{
response.sendRedirect(request.getContextPath() + "/login.jsp" );
}
return true;
}
}
```

第三步：配置拦截器，只拦截指定的控制器，并且可以排除指定的控制器方法

```
<!-- 配置拦截器 -->
<mvc:interceptors>
    <!-- 第一种配置方法：代表拦截所有的控制器 -->
    <!-- <mvc:interceptor>
        <mvc:mapping path="/**"/>
        <bean class="com.zelin.interceptor.FirstInterceptor"/>
    </mvc:interceptor> -->
    <!-- 第二种配置方法：可以直接定义在这里，会对所有的控制器起作用 -->
    <!-- <bean class="com.zelin.interceptor.FirstInterceptor"/>
    <bean class="com.zelin.interceptor.SecondInterceptor"/> -->
    <!-- 第三种配置方式：只拦截指定的控制器 -->
    <mvc:interceptor>
        <mvc:mapping path="/user/**"/>
        <mvc:exclude-mapping path="/user/login" />
        <bean class="com.zelin.interceptor.LoginInterceptor"/>
    </mvc:interceptor>
</mvc:interceptors>
```

代表只拦截user控制器，排除user下的login方法

第四步：演示功能：