

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。

```
> commons-dbcp-1.2.2.jar
> commons-fileupload-1.2.2.jar
> commons-io-2.4.jar
> commons-logging-1.1.1.jar
> commons-pool2-2.3.jar
> commons-pool-1.3.jar
> jackson-annotations-2.4.0.jar
> jackson-core-2.4.2.jar
> jackson-databind-2.4.2.jar
> javassist-3.17.1-GA.jar
> jsqlparser-0.9.1.jar
> jstl-1.2.jar
> junit-4.9.jar
> log4j-1.2.17.jar
```

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

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

```

2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:context="http://www.springframework.org/schema/context"
5     xmlns:mvc="http://www.springframework.org/schema/mvc"
6     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">
7     <!--1、配置springmvc的包扫描-->
8     <context:component-scan base-package="com.zelin.web.controller"/>
9     <mvc:annotation-driven conversion-service="conversionService"/>
10    <!--2.添加视图解析器-->
11    <!--3.配置转换器-->
12    <bean id="conversionService"
class="org.springframework.format.support.FormattingConversionServiceFactoryBean">
13        <property name="converters">
14            <list>
15                <bean class="com.zelin.web.converter.DateConverter"/>
16            </list>
17        </property>
18    </bean>
19 </beans>

```

2.1.2) 前端页面index.html编写：

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>学生列表</title>
6     <link rel="stylesheet" href="bootstrap-3.3.7/css/bootstrap.min.css">
7     <script src="bootstrap-3.3.7/js/jquery.min.js"></script>
8     <script src="bootstrap-3.3.7/js/bootstrap.min.js"></script>
9     <style>
10         .table {
11             text-align: center;
12         }
13
14         .container {
15             margin-top: 20px;
16         }
17     </style>
18 </head>
19 <body>
20 <div class="container">
21     <div class="panel panel-primary">
22         <div class="panel-heading">
23             <h4 class="glyphicon glyphicon-user">学生管理系统</h4>
24         </div>
25
26         <table class="table table-bordered table-hover">

```

```

26         <tr class="bg-info">
27             <td>学号</td>
28             <td>姓名</td>
29             <td>性别</td>
30             <td>年龄</td>
31             <td>住址</td>
32             <td>生日</td>
33             <td>所在班级</td>
34             <td>操作</td>
35         </tr>
36     </tbody id="tb"></tbody>
37
38 </table>
39 <div class="panel-footer clearfix" style="text-align:right;padding:10px;">
40     <a href="javascript:add()" class="btn btn-sm btn-warning pull-left
glyphicon glyphicon-plus-sign"
41         id="btn_add">添加学生</a>
42     泽林信息版权所有 2000-2018.
43 </div>
44 </div>
45
46 </div>
47
48 <!-- 模态框（添加或修改学生） -->
49 <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel">
50     <div class="modal-dialog" role="document">
51         <div class="modal-content">
52             <div class="modal-header">
53                 <button type="button" class="close" data-dismiss="modal" aria-
label="Close"><span aria-hidden="true">&times;</span></button>
54                 <h4 class="modal-title" id="myModalLabel">编辑学生信息</h4>
55             </div>
56             <div class="modal-body">
57                 <form class="form-horizontal" id="form1">
58                     <input type="hidden" name="sid" id="sid">
59                     <div class="form-group">
60                         <label class="col-sm-2 control-label">学生姓名:</label>
61                         <div class="col-sm-10">
62                             <input type="text" class="form-control" placeholder="输
入姓名" name="sname" id='sname'>
63                         </div>
64                     </div>
65                     <div class="form-group">
66                         <label class="col-sm-2 control-label">性别:</label>
67                         <div class="radio">
68                             <label>
69                                 <input type="radio" name="sex" value="男">男
70                             </label>
71                             <label>
72                                 <input type="radio" name="sex" value="女">女
73                             </label>
74                         </div>

```

```

75         </div>
76     </div>
77     <div class="form-group">
78         <label class="col-sm-2 control-label">学生年龄:</label>
79         <div class="col-sm-10">
80             <input type="text" class="form-control" placeholder="输入年龄" name="age" id="age">
81         </div>
82     </div>
83     <div class="form-group">
84         <label class="col-sm-2 control-label">学生住址:</label>
85         <div class="col-sm-10">
86             <input type="text" class="form-control" placeholder="输入住址" name="addr" id="addr">
87         </div>
88     </div>
89     <div class="form-group">
90         <label class="col-sm-2 control-label">生日:</label>
91         <div class="col-sm-10">
92             <input type="text" class="form-control" placeholder="输入生日" name="birth" id="birth">
93         </div>
94     </div>
95     <div class="form-group">
96         <label class="col-sm-2 control-label">班级:</label>
97         <div class="col-sm-10">
98             <select class="form-control" name="cid" id="cid">
99
100             </select>
101         </div>
102     </div>
103
104     </form>
105 </div>
106 <div class="modal-footer">
107     <button type="button" class="btn btn-default" data-dismiss="modal">关闭</button>
108     <button type="button" class="btn btn-primary" id="btn_save">保存</button>
109 </div>
110 </div>
111 </div>
112 </div>
113 </body>
114 </html>
115 <script>
116     $(function () {
117         //刷新列表
118         reloadList();
119     })
120
121     //刷新列表
122     function reloadList() {

```

```

123 //1、发出异步请求加载所有的学生列表
124 $.post(
125     'student/list.action',
126     function (data) {
127         var info = "";
128         $.each(data, function (i, v) {
129             var dt = new Date(v.birth);
130             var dtStr = dt.getFullYear() + "-" + (dt.getMonth() + 1) + "-" +
dt.getDate();
131             info += "<tr>";
132             info += "<td>" + v.sid + "</td>";
133             info += "<td>" + v.sname + "</td>";
134             info += "<td>" + v.sex + "</td>";
135             info += "<td>" + v.age + "</td>";
136             info += "<td>" + v.addr + "</td>";
137             info += "<td>" + dtStr + "</td>";
138             info += "<td>" + v.classes.cname + "</td>";
139             info += "<td>";
140             info += "<a href='javascript:;'
onclick='updateStudent(\"+v.sid+\")' class='btn btn-sm btn-info glyphicon glyphicon-
pencil'>修改</a>&nbsp;";
141             info += "<a href='javascript:;' class='btn btn-sm btn-danger
glyphicon glyphicon-trash' onclick='deleteStudent(\"+v.sid+\")'>删除</a>"
142             info += "</td>";
143             info += "</tr>";
144         })
145         //放上面的字符串放到tbody中
146         $("#tb").html(info);
147     }, "json"
148 );
149 //2.发出异步请求，得到所有的班级列表
150 $.post(
151     'classes/list.action',
152     function (data) {
153         var classesInfo = "";
154         $(data).each(function (i, v) {
155             classesInfo += "<option value='" + v.cid + "'>" + v.cname + "
</option>";
156         })
157         $("#cid").html(classesInfo);
158     }, "json")
159 }
160
161 //点击“添加学生”按钮
162 $("#btn_add").click(function () {
163     //1.显示模态对话框
164     $('#myModal').modal('show');
165     //2.清空表单
166     $('#form1')[0].reset();
167 })
168 //点击“保存”按钮
169 $("#btn_save").click(function () {
170     //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
224                 $('#cid').val(data.cid);

```

```

224         //设置默认选择性别
225         $('input[name=sex]').each(function(i,v){
226             $(this).attr("checked",$(v).val() == data.sex);
227         })
228     }, 'json'
229 )
230 //2. 显示对话框，并将当前这条记录显示到表单中
231 $('#myModal').modal('show');
232 //3. 为表单中的隐藏域字段赋值
233 $("#sid").val(v);
234 }
235 </script>

```

2.1.3) 后台页面StudentController.java

```

1  @RestController
2  @RequestMapping("/student")
3  public class StudentController {
4      @Autowired
5      private StudentService studentService;
6      //查询所有的学生
7      @RequestMapping("/list")
8      public List<Student> findAll(){
9          return studentService.findAll();
10     }
11     //添加学生
12     @RequestMapping("/add")
13     public Result add(Student student){
14         try {
15             studentService.add(student);
16             return new Result(true, "添加成功");
17         } catch (Exception e){
18             e.printStackTrace();
19             return new Result(false, "添加失败");
20         }
21     }
22 }
23

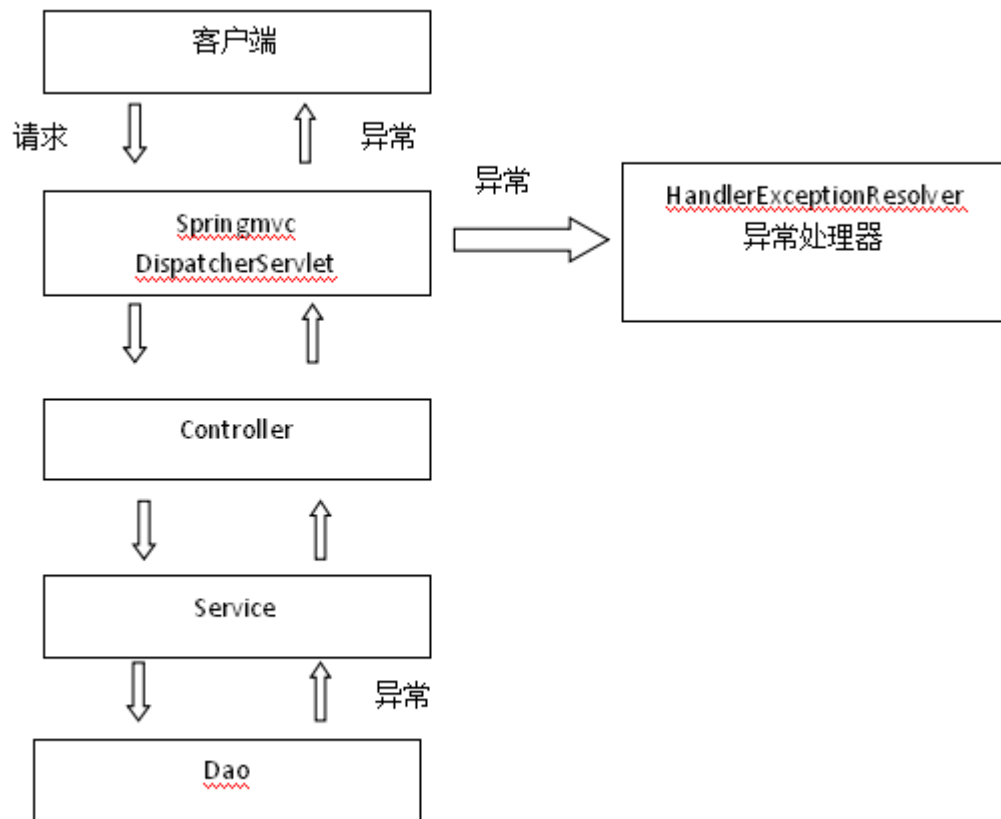
```

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：

```
1  /**
2   * 自定义异常
3   */
4  public class MyException extends RuntimeException {
5      private String message;
6
7      @Override
8      public String getMessage() {
9          return message;
10     }
11
12     public void setMessage(String message) {
13         this.message = message;
14     }
15
16     public MyException(String message) {
17         this.message = message;
18     }
19 }
```

3.2.2)自定义统一异常处理器：


```

1  @Component
2  public class CustomHandlerException implements HandlerExceptionResolver{
3      @Override
4      public ModelAndView resolveException(HttpServletRequest request,
5      HttpServletResponse response, Object o, Exception e) {
6          //1.如果当前获取到的异常对象e就是MyException, 则可以进行
7          MyException exception = null;
8          if(e instanceof MyException){
9              exception = (MyException) e;
10         }
11         //2.取得异常信息
12         String message = "";
13         if(exception != null){
14             message = exception.getMessage();
15         }else{
16             message = "对不起, 出现了未知异常!";
17         }
18         return new ModelAndView("error", "message", message);
19     }
20 }

```

3.2.3)在/WEB-INF/jsp/下添加error.jsp文件：

```

1  <!--
2      Created by IntelliJ IDEA.
3      User: Administrator
4      Date: 2018/11/14
5      Time: 15:28
6      To change this template use File | Settings | File Templates.
7  -->
8  <%@ page contentType="text/html; charset=UTF-8" language="java" %>
9  <html>
10 <head>
11     <title>错误处理</title>
12     <link rel="stylesheet" href="${pageContext.request.contextPath}/bootstrap-
13 3.3.7/css/bootstrap.min.css">
14     <script src="${pageContext.request.contextPath}/bootstrap-
15 3.3.7/js/jquery.min.js"></script>
16     <script src="${pageContext.request.contextPath}/bootstrap-
17 3.3.7/js/bootstrap.min.js"></script>
18 </head>
19 <body>
20 <div class="container">
21     <div class="panel panel-primary">
22         <div class="panel-heading">
23             <h4 class="glyphicon glyphicon-user">学生管理系统</h4>
24         </div>
25         <div class="panel-body">
26             ${message}
27         </div>
28         <div class="panel-footer" style="text-align: right;">

```

```

26         泽林信息版权所有 2000-2018.
27     </div>
28 </div>
29 </div>
30 </body>
31 </html>

```

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

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns:context="http://www.springframework.org/schema/context"
5      xmlns:mvc="http://www.springframework.org/schema/mvc"
6      xsi:schemaLocation="http://www.springframework.org/schema/beans
7          http://www.springframework.org/schema/beans/spring-beans.xsd
8          http://www.springframework.org/schema/context
9          http://www.springframework.org/schema/context/spring-context.xsd
10         http://www.springframework.org/schema/mvc
11         http://www.springframework.org/schema/mvc/spring-mvc.xsd">
12      <!--1、配置springmvc的包扫描-->
13      <context:component-scan base-
14          package="com.zelin.web.controller,com.zelin.web.exception"/>
15      <mvc:annotation-driven conversion-service="conversionService"/>
16      <!--2.添加视图解析器-->
17      <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
18          <property name="prefix" value="/WEB-INF/jsp/" />
19          <property name="suffix" value=".jsp" />
20      </bean>
21      <!--3.配置转换器-->
22      <bean id="conversionService"
23          class="org.springframework.format.support.FormattingConversionServiceFactoryBean">
24          <property name="converters">
25              <list>
26                  <bean class="com.zelin.web.converter.DateConverter"/>
27              </list>
28          </property>
29      </bean>
30  </beans>

```

3.2.5)修改index.html页面

```

1 <div class="panel-footer clearfix" style="text-align:right;padding:10px;">
2     <a href="javascript:;" class="btn btn-sm btn-warning pull-left glyphicon
glyphicon-plus-sign"
3         id="btn_add">添加学生</a>
4     <a href="javascript:;" class="btn btn-sm btn-success pull-left glyphicon
glyphicon-plus-sign"
5         id="btn_exception">测试统一异常处理</a>
6     泽林信息版权所有 2000-2018.
7 </div>
8
9 //测试异常处理
10 $("#btn_exception").click(function(){
11     location.href="student/exceHandler.action"
12 })

```

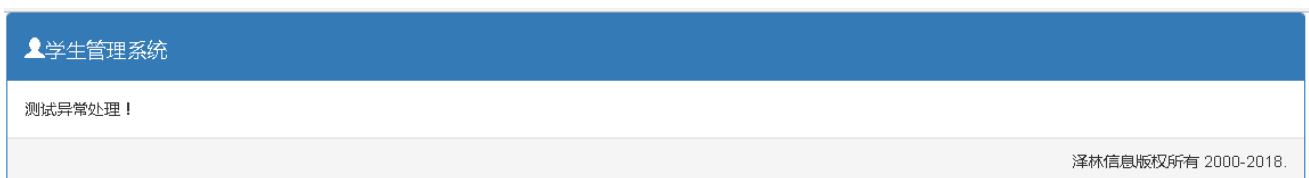
3.2.6)后台

```

1 @RequestMapping("/exceHandler")
2 public void exceHandler(){
3     throw new MyException("测试异常处理！");
4 }

```

3.2.7)页面展示



4、 SpringMVC-RESTful编程实践

4.1)修改web.xml文件：

```

1 <!--配置springmvc-->
2 <servlet>
3     <servlet-name>springmvc</servlet-name>
4     <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
5     <init-param>
6         <param-name>contextConfigLocation</param-name>
7         <param-value>classpath*:spring/applicationContext-mvc.xml</param-value>
8     </init-param>
9 </servlet>
10 <servlet-mapping>
11     <servlet-name>springmvc</servlet-name>
12     <url-pattern>/</url-pattern>
13 </servlet-mapping>

```

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

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

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

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>学生列表</title>
6      <link rel="stylesheet" href="../../bootstrap-3.3.7/css/bootstrap.min.css">
7      <script src="../../bootstrap-3.3.7/js/jquery.min.js"></script>
8      <script src="../../bootstrap-3.3.7/js/bootstrap.min.js"></script>
9      <style>
10         .table {
11             text-align: center;
12         }
13
14         .container {
15             margin-top: 20px;
16         }
17     </style>
18 </head>
19 <body>
20 <div class="container">
21     <div class="panel panel-primary">
22         <div class="panel-heading">
23             <h4 class="glyphicon glyphicon-user">学生管理系统</h4>
24         </div>
25         <table class="table table-bordered table-hover">
26             <tr class="bg-info">
27                 <td>学号</td>
28                 <td>姓名</td>
29                 <td>性别</td>
30                 <td>年龄</td>
31                 <td>住址</td>
32                 <td>生日</td>
33                 <td>所在班级</td>
34                 <td>操作</td>
35             </tr>
36             <tbody id="tb"></tbody>
37
38         </table>
39         <div class="panel-footer clearfix" style="text-align:right;padding:10px;">
40             <a href="javascript:;" class="btn btn-sm btn-warning pull-left glyphicon glyphicon-plus-sign"
41                 id="btn_add">添加学生</a>
```

```

42     <a href="javascript:;" class="btn btn-sm btn-success pull-left glyphicon
glyphicon-plus-sign"
43     id="btn_exception">测试统一异常处理</a>
44     泽林信息版权所有 2000-2018.
45     </div>
46     </div>
47
48 </div>
49
50 <!-- 模态框（添加或修改学生）-->
51 <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-
labelledby="myModalLabel">
52     <div class="modal-dialog" role="document">
53         <div class="modal-content">
54             <div class="modal-header">
55                 <button type="button" class="close" data-dismiss="modal" aria-
label="Close"><span aria-hidden="true">&times;</span>
56                 </button>
57                 <h4 class="modal-title" id="myModalLabel">编辑学生信息</h4>
58             </div>
59             <div class="modal-body">
60                 <form class="form-horizontal" id="form1">
61                     <input type="hidden" name="sid" id="sid">
62                     <div class="form-group">
63                         <label class="col-sm-2 control-label">学生姓名:</label>
64                         <div class="col-sm-10">
65                             <input type="text" class="form-control" placeholder="输
入姓名" name="sname" id='sname'>
66                         </div>
67                     </div>
68                     <div class="form-group">
69                         <label class="col-sm-2 control-label">性别:</label>
70                         <div class="radio">
71                             <label>
72                                 <input type="radio" name="sex" value="男">男
73                             </label>
74                             <label>
75                                 <input type="radio" name="sex" value="女">女
76                             </label>
77                         </div>
78                     </div>
79                     <div class="form-group">
80                         <label class="col-sm-2 control-label">学生年龄:</label>
81                         <div class="col-sm-10">
82                             <input type="text" class="form-control" placeholder="输
入年龄" name="age" id="age">
83                         </div>
84                     </div>
85                     <div class="form-group">
86                         <label class="col-sm-2 control-label">学生住址:</label>
87                         <div class="col-sm-10">
88                             <input type="text" class="form-control" placeholder="输

```

```

89         入住址" name="addr" id="addr">
90             </div>
91         </div>
92         <div class="form-group">
93             <label class="col-sm-2 control-label">生日:</label>
94             <div class="col-sm-10">
95                 入生日" name="birth" id="birth">
96                     <input type="text" class="form-control" placeholder="输
97             </div>
98         </div>
99         <div class="form-group">
100             <label class="col-sm-2 control-label">班级:</label>
101             <div class="col-sm-10">
102                 <select class="form-control" name="cid" id="cid">
103             </select>
104             </div>
105         </div>
106     </form>
107 </div>
108 <div class="modal-footer">
109     关闭<button type="button" class="btn btn-default" data-dismiss="modal">
110     <button type="button" class="btn btn-primary" id="btn_save">保存
111 </button>
112 </div>
113 </div>
114 </div>
115 </body>
116 </html>
117 <script>
118     $(function () {
119         //刷新列表
120         reloadList();
121     })
122
123     //刷新列表
124     function reloadList() {
125         //1、发出异步请求加载所有的学生列表
126         $.post(
127             'list',
128             function (data) {
129                 var info = "";
130                 $.each(data, function (i, v) {
131                     var dt = new Date(v.birth);
132                     var dtStr = dt.getFullYear() + "-" + (dt.getMonth() + 1) + "-" +
133                     dt.getDate();
134                     info += "<tr>";
135                     info += "<td>" + v.sid + "</td>";
136                     info += "<td>" + v.sname + "</td>";
137                     info += "<td>" + v.sex + "</td>";

```

```

137         info += "<td>" + v.age + "</td>";
138         info += "<td>" + v.addr + "</td>";
139         info += "<td>" + dtStr + "</td>";
140         info += "<td>" + v.classes.cname + "</td>";
141         info += "<td>";
142         info += "<a href='javascript:;' "
onclick='updateStudent("+v.sid+")' class='btn btn-sm btn-info glyphicon glyphicon-
pencil'>修改</a>&nbsp;"
143         info += "<a href='javascript:;' class='btn btn-sm btn-danger
glyphicon glyphicon-trash' onclick='deleStudent("+v.sid+")'>删除</a>"
144         info += "</td>";
145         info += "</tr>";
146     })
147     //放上面的字符串放到tbody中
148     $("#tb").html(info);
149     }, "json"
150 );
151 //2.发出异步请求,得到所有的班级列表
152 $.post(
153     '../classes/list',
154     function (data) {
155         var classesInfo = "";
156         $(data).each(function (i, v) {
157             classesInfo += "<option value='" + v.cid + "'>" + v.cname + "
</option>";
158         })
159         $("#cid").html(classesInfo);
160     }, "json")
161 }
162
163 //点击“添加学生”按钮
164 $("#btn_add").click(function () {
165     //1.显示模态对话框
166     $('#myModal').modal('show');
167     //2.清空表单
168     $('#form1')[0].reset();
169 })
170 //测试异常处理
171 $("#btn_exception").click(function(){
172     location.href="student/exceHandler.action"
173 })
174 //点击“保存”按钮
175 $("#btn_save").click(function () {
176     //判断是添加还是修改操作? (因为此二者共用一个对话框)
177     if($('#sid').val() ){           //做修改操作
178         operation('student/update');
179     }else{                          //做添加操作
180         operation('student/add');
181     }
182 }
183 })
184 function operation(url){
185     //1.取出表单中的值

```

```

186     var studentEntity = $("#form1").serialize();
187     //2.将数据异步提交到数据库中
188     $.post(
189         url,
190         studentEntity,
191         function (data) {
192             if(data.success){
193                 //刷新列表
194                 reloadList();
195             }else{
196                 alert(data.message);
197             }
198             //关闭对话框
199             $('#myModal').modal('hide');
200         }, "json")
201     }
202     //删除学生
203     function deleStudent(v){
204         $.post(
205             "student/delete/"+v,
206             function(data){
207                 if(data.success){
208                     //刷新列表
209                     reloadList();
210                 }else{
211                     alert(data.message);
212                 }
213             }, "json"
214         )
215     }
216     //修改学生
217     function updateStudent(v) {
218         $('#form1')[0].reset();
219
220         //1.根据sid查询出这条记录（异步查询）
221         $.post(
222             "findOne/"+v,
223             function(data){
224                 var dt = new Date(data.birth);
225                 var dtStr = dt.getFullYear() + "-" + (dt.getMonth() + 1) + "-" +
dt.getDate();
226                 //将得到的学生对象显示到表单中
227                 $("#sname").val(data.sname);
228                 $("#age").val(data.age);
229                 $("#addr").val(data.addr);
230                 $("#birth").val(dtStr);
231                 $("#sname").val(data.sname);
232                 $('#cid').val(data.cid);
233                 //设置默认选择性别
234                 $('input[name=sex]').each(function(i,v){
235                     $(this).attr("checked",$(v).val() == data.sex);
236                 })
237             }, 'json'

```



```

238     )
239     //2.显示对话框，并将当前这条记录显示到表单中
240     $('#myModal').modal('show');
241     //3.为表单中的隐藏域字段赋值
242     $("#sid").val(v);
243 }
244 </script>

```

4.4)在StudentController.java中添加处理器

```

1  @Controller
2  @RequestMapping("/student")
3  public class StudentController {
4      @Autowired
5      private StudentService studentService;
6
7      //进行学生列表的首页
8      @RequestMapping("/tolist")
9      public String welcome(){
10         return "student/list";
11     }
12     //根据学生编号查询学生
13     @RequestMapping("/findOne/{sid}")
14     @ResponseBody
15     public Student findOne(@PathVariable() String sid){
16         return studentService.findOne(sid);
17     }
18     //查询所有的学生
19     @RequestMapping("/list")
20     @ResponseBody
21     public List<Student> findAll(){
22         return studentService.findAll();
23     }
24     //根据学号删除学生
25     @RequestMapping("/delete/{sid}")
26     @ResponseBody
27     public Result delete(@PathVariable String sid){
28         try{
29             studentService.delete(sid);
30             return new Result(true,"删除成功!");
31         }catch (Exception e){
32             e.printStackTrace();
33             return new Result(false,"删除失败!");
34         }
35     }
36 }
37 //添加学生
38 @RequestMapping("/add")
39 @ResponseBody
40 public Result add(Student student){
41     try {
42         studentService.add(student);

```

```

43         return new Result(true, "添加成功");
44     } catch (Exception e) {
45         e.printStackTrace();
46         return new Result(false, "添加失败");
47     }
48 }
49 @RequestMapping("/exceHandler")
50 public void exceHandler() {
51     throw new MyException("测试异常处理!");
52 }
53 }
54

```

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：

```
2018-06-13 16:04:50,077 [http-bio-8080-exec-3] DEBUG [org.springframework]
FirstInterceptor→preHandle(),执行时间是：1528877090079
2018-06-13 16:04:50,192 [http-bio-8080-exec-3] DEBUG [org.springframework]
FirstInterceptor→postHandle(),执行时间是：1528877091584
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.entity.Classes@74ab8e68, birth=Thu
2018-06-13 16:04:51,596 [http-bio-8080-exec-3] DEBUG [org.springframework.beans.factory.support.DefaultListableBe
2018-06-13 16:04:51,597 [http-bio-8080-exec-3] DEBUG [org.springframework.web.servlet.DispatcherServlet] - Render
2018-06-13 16:04:51,597 [http-bio-8080-exec-3] DEBUG [org.springframework.web.servlet.view.JstlView] - Added mode
2018-06-13 16:04:51,604 [http-bio-8080-exec-3] DEBUG [org.springframework.web.servlet.view.JstlView] - Forwarding
FirstInterceptor→afterCompletion(),执行时间是：1528877091805
2018-06-13 16:04:51,806 [http-bio-8080-exec-3] DEBUG [org.springframework.web.servlet.DispatcherServlet] - Succes
2018-06-13 16:04:51,813 [http-bio-8080-exec-3] DEBUG [org.springframework.beans.factory.support.DefaultListableBe
```

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：

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：

用户登录

输入用户名：

admin,登录成功！