

第三章 MyBatis框架

-MyBatis持久层框架 2018/11/9 [泽林.王峰]

授课目标

- 1、 上章回顾
- 2、 MyBatis与Spring整合开发
- 3、 SM+Servlet整合开发
- 4、 MyBatis自动生成工具(逆向工程)的使用
- 5、 利用自动生成的工具类完成学生操作
- 6、 MyBatis分页插件-PageHelper使用

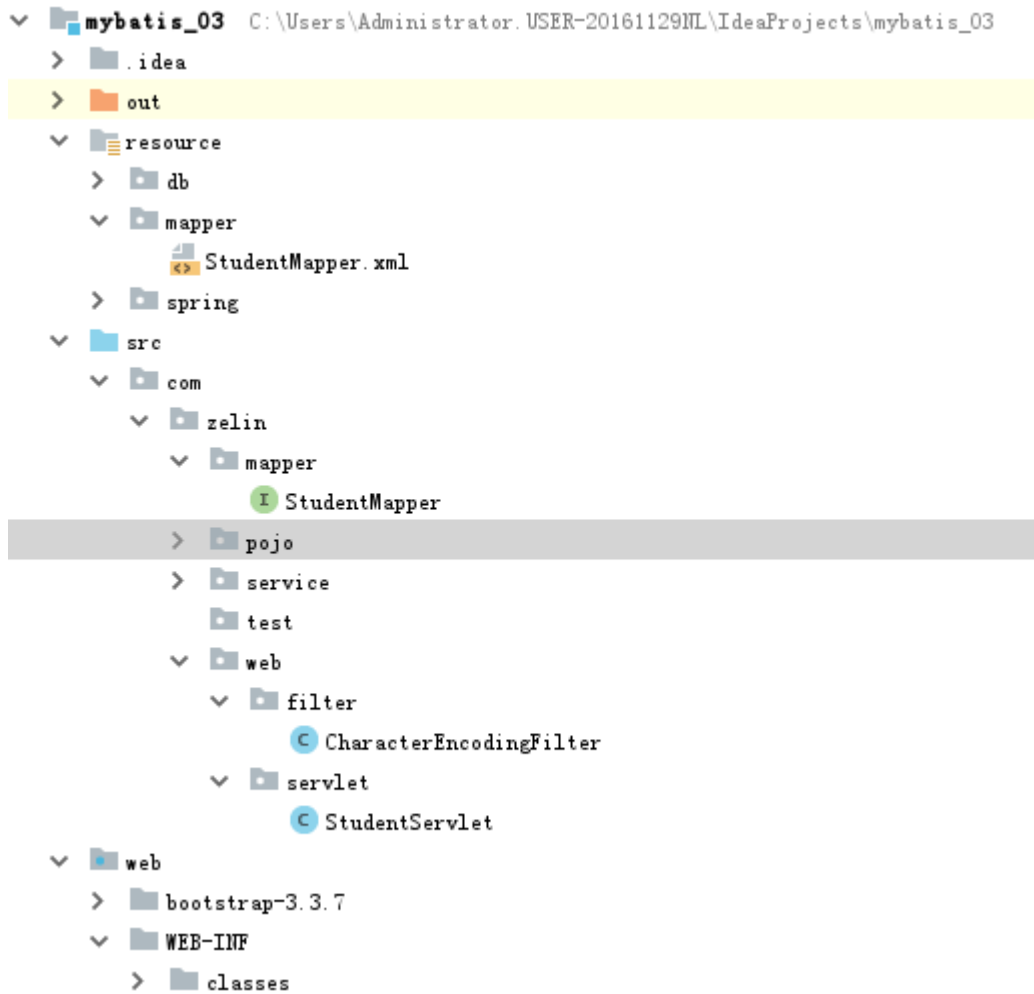
授课内容

1、 上章回顾

2、 MyBatis与Spring整合开发

3、 SM+Servlet整合开发

3.1) 定义工程的目录结构如下：



3.2) 定义resource/db/db.properties文件

```
1 jdbc.driver=com.mysql.jdbc.Driver
2 jdbc.url=jdbc:mysql:///java1301
3 jdbc.user=root
4 jdbc.password=123
```

3.3) 定义resource/spring/applicationContext.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       xsi:schemaLocation="http://www.springframework.org/schema/beans
6         http://www.springframework.org/schema/beans/spring-beans.xsd
7         http://www.springframework.org/schema/context
8         http://www.springframework.org/schema/context/spring-context.xsd">
9   <!--1. 读取属性文件-->
10  <context:property-placeholder location="classpath*:db/db.properties"/>
11  <!--2. 指定自动扫描包-->
12  <context:component-scan base-package="com.zelin"/>
13  <!--3. 配置数据源-->
14  <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
```

```

12     <property name="driverClassName" value="${jdbc.driver}"/>
13     <property name="url" value="${jdbc.url}"/>
14     <property name="username" value="${jdbc.user}"/>
15     <property name="password" value="${jdbc.password}"/>
16 </bean>
17 <!--4.配置SqlSessionFactoryBean-->
18 <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
19     <property name="dataSource" ref="dataSource"/>
20     <property name="typeAliasesPackage" value="com.zelin.pojo"/>
21     <!--配置mapper的映射文件的扫描位置-->
22     <property name="mapperLocations" value="classpath*:mapper/*.xml"/>
23 </bean>
24 <!--5.配置MapperScannerConfigurer-->
25 <bean id="mapperScannerConfigurer"
26 class="org.mybatis.spring.mapper.MapperScannerConfigurer">
27     <!--配置mapper接口的位置-->
28     <property name="basePackage" value="com.zelin.mapper"/>
29 </bean>
30 </beans>

```

3.4) 定义resource/mapper/StudentMapper.xml文件

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper
3     PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6 <mapper namespace="com.zelin.mapper.StudentMapper">
7     <!--1.查询所有学生-->
8     <select id="findAll" resultType="studentCustom">
9         select st.*,cname
10        from student st,classes c
11        where st.cid=c.cid
12    </select>
13 </mapper>

```

3.5) 定义com.zelin.mapper下的StudentMapper.java接口

```

1 public interface StudentMapper {
2     public List<StudentCustom> findAll() throws Exception;
3 }

```

3.6) 定义com.zelin.service下的StudentService.java接口

```

1 public interface StudentService {
2     public List<StudentCustom> findAll() throws Exception;
3 }

```

3.7) 定义com.zelin.service.impl下的StudentServiceImpl.java实现类

```

1  @Service
2  public class StudentServiceImpl implements StudentService {
3      @Autowired
4      private StudentMapper studentMapper;
5      @Override
6      public List<StudentCustom> findAll() throws Exception {
7          return studentMapper.findAll();
8      }
9  }
10

```

3.8) 定义web.xml文件

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4           xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
5                               http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
6           version="3.1">
7      <!--1. 配置spring容器的xml文件的加载位置-->
8      <context-param>
9          <param-name>contextConfigLocation</param-name>
10         <param-value>classpath*:spring/applicationContext*.xml</param-value>
11     </context-param>
12     <!--2. 配置spring的监听器-->
13     <listener>
14         <listener-
15             class>org.springframework.web.context.ContextLoaderListener</listener-class>
16     </listener>
17 </web-app>

```

3.9) 定义com.zelin.web.filter下的CharacterEncodingFilter过滤器

```

1  @WebFilter(urlPatterns = "/*")
2  public class CharacterEncodingFilter implements Filter {
3      public void destroy() {
4      }
5
6      public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)
7      throws ServletException, IOException {
8          req.setCharacterEncoding("UTF-8");
9          chain.doFilter(req, resp);
10     }
11
12     public void init(FilterConfig config) throws ServletException {
13     }
14 }

```

3.10) 定义com.zelin.web.servlet下的StudentServlet.java

```

1  @WebServlet("/student")
2  public class StudentServlet extends HttpServlet {
3      private StudentService studentService;
4      @Override
5      public void init() throws ServletException {
6          WebApplicationContext webApplicationContext =
WebApplicationContextUtils.getWebApplicationContext(getServletContext());
7          studentService = webApplicationContext.getBean(StudentService.class);
8      }
9      protected void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
10         response.setContentType("text/html;charset=utf-8");
11         //1.得到请求参数
12         String method = request.getParameter("method");
13         //2.根据请求参数执行不同的方法
14         if(!StringUtils.isEmpty(method)){
15             if("list".equals(method)){           //列表学生
16                 list(request,response);
17             }
18         }
19     }
20     //列表学生
21     private void list(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
22         try {
23             //1.得到所有的学生对象
24             List<StudentCustom> students = studentService.findAll();
25             //2.将学生对象转换为json串并输出
26             response.getWriter().print(JSON.toJSONString(students));
27             response.getWriter().close();
28         } catch (Exception e) {
29             e.printStackTrace();
30         }
31     }
32 }

```

3.11) 定义/web/index.html

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Title</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:add()" class="btn btn-sm btn-warning pull-left
41             glyphicon glyphicon-plus-sign"
42                 id="btn_add">添加学生</a>
43             泽林信息版权所有 2000-2018.
44         </div>
45     </div>
46 </div>
47
48 </body>
49 </html>
50 <script>
51     $(function () {
52         //刷新列表
53         reloadList();
54     })
55
56     //刷新列表
57     function reloadList() {
58         //1、发出异步请求加载所有的学生列表
59         $.post(
60             'student?method=list',
61             function (data) {
62                 var info = "";
63                 $.each(data, function (i, v) {
64                     info += "<tr>";
65                     info += "<td>" + v.sid + "</td>";
66
67                     info += "<td>" + v.sname + "</td>";

```

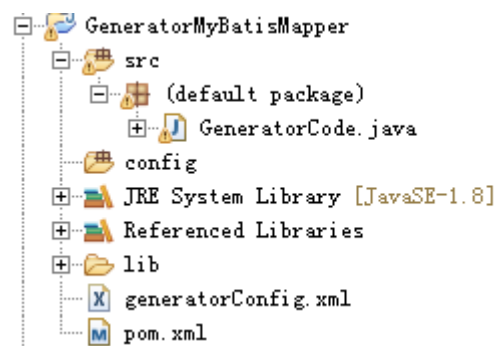
```

67         info += "<td>" + v.sex + "</td>";
68         info += "<td>" + v.age + "</td>";
69         info += "<td>" + v.addr + "</td>";
70         info += "<td>" + v.birth + "</td>";
71         info += "<td>" + v.cname + "</td>";
72         info += "<td>";
73         info += "<a href='javascript:;' "
onclick='updateStudent("+v.sid+")' class='btn btn-sm btn-info glyphicon glyphicon-
pencil'>修改</a>&nbsp;"
74         info += "<a href='javascript:;' class='btn btn-sm btn-danger
glyphicon glyphicon-trash' onclick='deleStudent("+v.sid+")'>删除</a>"
75         info += "</td>";
76         info += "</tr>";
77     })
78     //放上面的字符串放到tbody中
79     $("#tb").html(info);
80     }, "json"
81 );
82 //2.发出异步请求，得到所有的班级列表
83 $.post(
84     'classes?method=list',
85     function (data) {
86         var classesInfo = "";
87         $(data).each(function (i, v) {
88             classesInfo += "<option value='" + v.cid + "'>" + v.cname + "
</option>";
89         })
90         $("#cid").html(classesInfo);
91     }, "json")
92 }
93
94 </script>

```

4、 MyBatis自动生成工具(逆向工程)的使用

4.1)定义自动生成的java工程（普通的java工程）



4.2)定义自动生成的xml文件

```

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

```

```

2 <!DOCTYPE generatorConfiguration
3 PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
4 "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">
5 <generatorConfiguration>
6   <context id="testTables" targetRuntime="MyBatis3">
7     <!-- 添加此插件可以让生成的实体类自动实现Serializable接口 -->
8     <plugin type="org.mybatis.generator.plugins.SerializablePlugin" />
9     <commentGenerator>
10       <!-- 是否去除自动生成的注释 true:是 : false:否 -->
11       <property name="suppressAllComments" value="true" />
12     </commentGenerator>
13     <!-- 数据库连接的信息：驱动类、连接地址、用户名、密码 -->
14     <jdbcConnection driverClass="com.mysql.jdbc.Driver"
15       connectionURL="jdbc:mysql:///java1301"
16       userId="root" password="123">
17     </jdbcConnection>
18     <!-- Oracle数据库配置 -->
19     <!-- <jdbcConnection driverClass="oracle.jdbc.OracleDriver"
20       connectionURL="jdbc:oracle:thin:@127.0.0.1:1521:orcl"
21       userId="scott"
22       password="tiger">
23     </jdbcConnection> -->
24
25     <!-- 默认false，把JDBC DECIMAL 和 NUMERIC 类型解析为 Integer，为 true时把JDBC
DECIMAL 和
26       NUMERIC 类型解析为java.math.BigDecimal -->
27     <javaTypeResolver>
28       <property name="forceBigDecimals" value="false" />
29     </javaTypeResolver>
30
31     <!-- targetProject:生成POJO类的位置 -->
32     <javaModelGenerator targetPackage="com.zelin.pojo"
33       targetProject=".\\src">
34       <!-- enableSubPackages:是否让schema作为包的后缀 -->
35       <property name="enableSubPackages" value="false" />
36       <!-- 从数据库返回的值被清理前后的空格 -->
37       <property name="trimStrings" value="true" />
38     </javaModelGenerator>
39     <!-- targetProject:mapper映射XML文件生成的位置 -->
40     <sqlMapGenerator targetPackage="mapper"
41       targetProject=".\\config">
42       <!-- enableSubPackages:是否让schema作为包的后缀 -->
43       <property name="enableSubPackages" value="false" />
44     </sqlMapGenerator>
45     <!-- targetPackage: mapper映射的接口生成的位置 -->
46     <javaClientGenerator type="XMLMAPPER"
47       targetPackage="com.zelin.mapper"
48       targetProject=".\\src">
49       <!-- enableSubPackages:是否让schema作为包的后缀 -->
50       <property name="enableSubPackages" value="false" />
51     </javaClientGenerator>
52     <!-- 指定数据库表 -->
53
54     <table schema="" tableName="book"></table>

```



```

54     <table schema="" tableName="category"></table>
55     </context>
56 </generatorConfiguration>

```

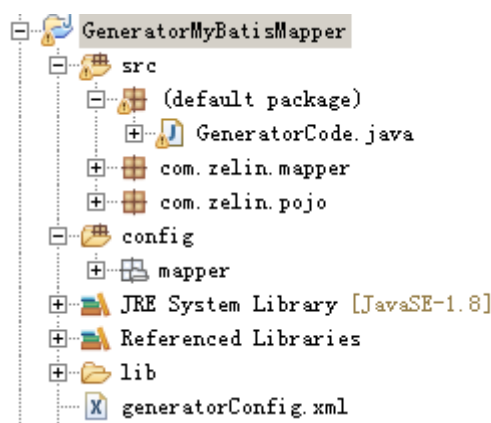
4.3)定义自动生成的java类：

```

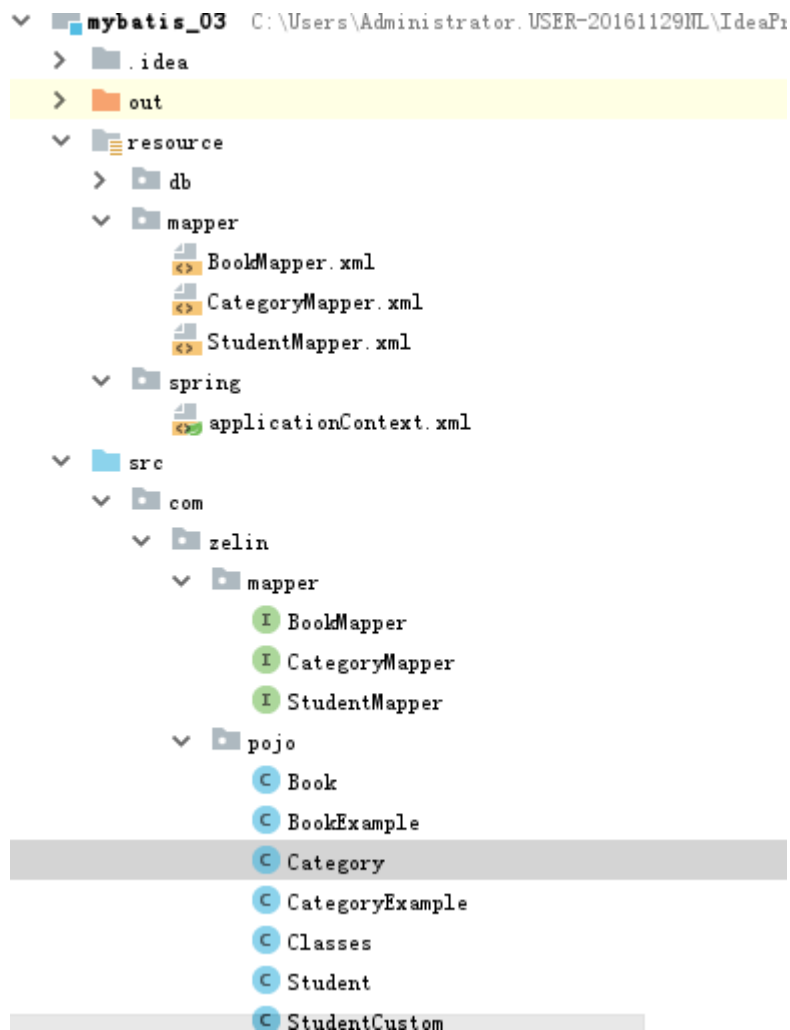
1  public class GeneratorCode {
2
3      public static void main(String[] args) throws Exception {
4          generator();
5          System.out.println("生成成功!");
6      }
7      private static void generator() throws Exception {
8          List<String> warnings = new ArrayList<String>();
9          boolean overwrite = true;
10         //这里的配置文件一定要放到项目的根目录下
11         File configFile = new File("generatorConfig.xml");
12         ConfigurationParser cp = new ConfigurationParser(warnings);
13         Configuration config = cp.parseConfiguration(configFile);
14         DefaultShellCallback callback = new DefaultShellCallback(overwrite);
15         MyBatisGenerator myBatisGenerator = new MyBatisGenerator(config, callback,
warnings);
16         myBatisGenerator.generate(null);
17     }
18 }

```

4.4)运行自动生成后的工程效果如下图：



4.5) 将自动生成的相关文件拷贝到你自己的项目中



4.6) 简单测试生成的内容：

```
1 @RunWith(SpringJUnit4ClassRunner.class)
2 @ContextConfiguration("classpath*:spring/applicationContext*.xml")
3 public class TestBookMapper {
4     @Autowired
5     private BookMapper bookMapper;
6     @Test
7     public void testFindAll() throws Exception{
8         List<Book> books = bookMapper.selectByExample(null);
9         for(Book book : books){
10             System.out.println(book);
11         }
12     }
13 }
```

5、 利用自动生成的工具类完成学生操作(CRU)

5.1) 定义StudentService.java接口及StudentServiceImpl.java实现类：

```
1 //接口
2 public interface StudentService {
```

```

3     public List<StudentCustom> findAll() throws Exception;
4
5     void insert(Student student);
6
7     Student findOne(String sid);
8
9     void update(Student student);
10 }
11 //实现类
12 @Service
13 public class StudentServiceImpl implements StudentService {
14     @Autowired
15     private StudentMapper studentMapper;
16     @Override
17     public List<StudentCustom> findAll() throws Exception {
18         return studentMapper.findAll();
19     }
20
21     @Override
22     public void insert(Student student) {
23         studentMapper.insert(student);
24     }
25
26     @Override
27     public Student findOne(String sid) {
28         return studentMapper.selectByPrimaryKey(Integer.valueOf(sid));
29     }
30
31     @Override
32     public void update(Student student) {
33         studentMapper.updateByPrimaryKey(student);
34     }
35 }

```

5.2) 定义StudentServlet.java

```

1  @WebServlet("/student")
2  public class StudentServlet extends HttpServlet {
3      private StudentService studentService;
4      @Override
5      public void init() throws ServletException {
6          WebApplicationContext webApplicationContext =
7      WebApplicationContextUtils.getWebApplicationContext(getServletContext());
8          studentService = webApplicationContext.getBean(StudentService.class);
9      }
10     protected void service(HttpServletRequest request, HttpServletResponse response)
11     throws ServletException, IOException {
12         response.setContentType("text/html;charset=utf-8");
13         //1.得到请求参数
14         String method = request.getParameter("method");
15         //2.根据请求参数执行不同的方法
16
17         if(!StringUtils.isEmpty(method)){

```

```

15         if("list".equals(method)){           //列表学生
16             list(request,response);
17         }else if("add".equals(method)){       //添加学生
18             add(request,response);
19         }else if("findOne".equals(method)){ //到后台查询学生
20             findOne(request,response);
21         }else if("update".equals(method)){ //修改学生
22             update(request,response);
23         }
24     }
25 }
26
27     private void update(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
28         Result result = null;
29         try {
30             //1.得到表单数据
31             Student student = new Student();
32             DateConverter dateConverter = new DateConverter();
33             dateConverter.setPattern("yyyy-MM-dd");
34             ConvertUtils.register(dateConverter,Date.class);
35             //2.封装得到的表单数据
36             BeanUtils.populate(student,request.getParameterMap());
37             studentService.update(student);
38             result = new Result(true,"修改成功");
39
40         } catch (Exception e) {
41             e.printStackTrace();
42             result = new Result(false,"修改失败");
43         }
44         response.getWriter().print(JSON.toJSONString(result));
45         response.getWriter().close();
46     }
47
48     //根据学生编号查询学生
49     private void findOne(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
50         //1.得到学生编号
51         String sid = request.getParameter("sid");
52         //2.根据学生编号查询学生对象
53         Student student = studentService.findOne(sid);
54         //2.将学生对象转换为json串并输出
55         response.getWriter().print(JSON.toJSONString(student));
56         response.getWriter().close();
57     }
58
59
60     //添加学生
61     private void add(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
62         Result result = null;
63         try {
64             //1.得到表单数据

```

```

65         Student student = new Student();
66         DateConverter dateConverter = new DateConverter();
67         dateConverter.setPattern("yyyy-MM-dd");
68         ConvertUtils.register(dateConverter, Date.class);
69         //2. 封装得到的表单数据
70         BeanUtils.populate(student, request.getParameterMap());
71         //3. 提交数据到数据库中
72         studentService.insert(student);
73         result = new Result(true, "添加成功");
74     } catch (Exception e) {
75         e.printStackTrace();
76         result = new Result(false, "添加失败");
77     }
78     response.getWriter().print(JSON.toJSONString(result));
79     response.getWriter().close();
80 }
81
82 //列表学生
83 private void list(HttpServletRequest request, HttpServletResponse response)
84 throws ServletException, IOException {
85     try {
86         //1. 得到所有的学生对象
87         List<StudentCustom> students = studentService.findAll();
88         //2. 将学生对象转换为json串并输出
89         response.getWriter().print(JSON.toJSONString(students));
90         response.getWriter().close();
91     } catch (Exception e) {
92         e.printStackTrace();
93     }
94 }
95

```

5.3) 定义index.html页面：

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Title</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:add()" class="btn btn-sm btn-warning pull-left
41             glyphicon glyphicon-plus-sign
42             id="btn_add">添加学生</a>
43             泽林信息版权所有 2000-2018.
44         </div>
45     </div>
46 </div>
47 <!-- 模态框（添加或修改学生） -->
48 <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-
49     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-
54                 label="Close"><span aria-hidden="true">&times;</span></button>
55                 <h4 class="modal-title" id="myModalLabel">编辑学生信息</h4>
56             </div>
57             <div class="modal-body">
58                 <form class="form-horizontal" id="form1">
59                     <input type="hidden" name="sid" id="sid">
60                     <div class="form-group">
61                         <label class="col-sm-2 control-label">学生姓名:</label>
62                         <div class="col-sm-10">
63                             <input type="text" class="form-control" placeholder="输
64                             入姓名" name="sname" id='sname'>
65                         </div>
66                     </div>
67                     <div class="form-group">
68                         <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 class="form-group">
77         <label class="col-sm-2 control-label">学生年龄:</label>
78         <div class="col-sm-10">
79             <input type="text" class="form-control" placeholder="输
入年龄" name="age" id="age">
80         </div>
81     </div>
82     <div class="form-group">
83         <label class="col-sm-2 control-label">学生住址:</label>
84         <div class="col-sm-10">
85             <input type="text" class="form-control" placeholder="输
入住址" name="addr" id="addr">
86         </div>
87     </div>
88     <div class="form-group">
89         <label class="col-sm-2 control-label">生日:</label>
90         <div class="col-sm-10">
91             <input type="text" class="form-control" placeholder="输
入生日" name="birth" id="birth">
92         </div>
93     </div>
94     <div class="form-group">
95         <label class="col-sm-2 control-label">班级:</label>
96         <div class="col-sm-10">
97             <select class="form-control" name="cid" id="cid">
98
99             </select>
100         </div>
101     </div>
102
103     </form>
104 </div>
105 <div class="modal-footer">
106     <button type="button" class="btn btn-default" data-dismiss="modal">
关闭</button>
107     <button type="button" class="btn btn-primary" id="btn_save"
onclick="save()">保存</button>
108 </div>
109 </div>
110 </div>
111 </div>
112 </body>
113 </html>
114 <script>

```

```

115     $(function () {
116         //刷新列表
117         reloadList();
118     })
119     //添加学生
120     function add() {
121         $('#form1')[0].reset();
122         //1.弹出对话框
123         $("#myModal").modal("show");
124     }
125     //保存学生(因为添加/修改使用的是同一对话框,所以,要判断当前是添加还是修改)
126     function save(){
127         //0.判断表单隐藏域中的sid是否存在值
128         var url = "student?method=update"
129         if(!$("#sid").val()){           //主键没有值就是执行添加操作
130             url = "student?method=add";
131         }
132         //1.得到表单数据并提交
133         $.post(
134             url,
135             $("#form1").serialize(),
136             function(data){
137                 if(data.success){
138                     reloadList();
139                 }else{
140                     alert(data.message);
141                 }
142                 //关闭对话框
143                 $("#myModal").modal("hide");
144             }, "json"
145         )
146     }
147     //修改学生
148     function updateStudent(v){
149         $('#form1')[0].reset();
150         //1.显示对话框
151         $("#myModal").modal("show");
152         //2.到后台查询学生
153         $.post(
154             "student?method=findOne&sid="+v,
155             function(data){
156                 var dt = new Date(data.birth);
157                 var y = dt.getFullYear();
158                 var m = dt.getMonth() + 1;
159                 var day = dt.getDate();
160                 var birth = y + "-" + m + "-" + day;
161                 //将得到的学生对象显示到表单中
162                 $("#sid").val(data.sid);
163                 $("#sname").val(data.sname);
164                 $("#age").val(data.age);
165                 $("#addr").val(data.addr);
166                 $("#birth").val(birth);
167                 $("#sname").val(data.sname);

```



```

168         $('#cid').val(data.cid);
169         //设置默认选择性别
170         $('#input[name=sex]').each(function(i,v){
171             $(this).attr("checked",$(v).val() == data.sex);
172         })
173
174     }, "json"
175 )
176 }
177 //刷新列表
178 function reloadList() {
179     //1、发出异步请求加载所有的学生列表
180     $.post(
181         'student?method=list',
182         function (data) {
183             var info = "";
184             $.each(data, function (i, v) {
185                 var dt = new Date(v.birth);
186                 var y = dt.getFullYear();
187                 var m = dt.getMonth() + 1;
188                 var day = dt.getDate();
189                 var birth = y + "-" + m + "-" + day;
190                 info += "<tr>";
191                 info += "<td>" + v.sid + "</td>";
192                 info += "<td>" + v.sname + "</td>";
193                 info += "<td>" + v.sex + "</td>";
194                 info += "<td>" + v.age + "</td>";
195                 info += "<td>" + v.addr + "</td>";
196                 info += "<td>" + birth + "</td>";
197                 info += "<td>" + v.cname + "</td>";
198                 info += "<td>";
199                 info += "<a href='javascript:;' "
onclick='updateStudent("+v.sid+")' class='btn btn-sm btn-info glyphicon glyphicon-
pencil'>修改</a>&nbsp;"
200                 info += "<a href='javascript:;' class='btn btn-sm btn-danger
glyphicon glyphicon-trash' onclick='deleStudent("+v.sid+")'>删除</a>"
201                 info += "</td>";
202                 info += "</tr>";
203             })
204             //放上面的字符串放到tbody中
205             $("#tb").html(info);
206         }, "json"
207     );
208     //2.发出异步请求，得到所有的班级列表
209     $.post(
210         'classes?method=list',
211         function (data) {
212             var classesInfo = "";
213             $(data).each(function (i, v) {
214                 classesInfo += "<option value='" + v.cid + "'>" + v.cname + "
</option>";
215             })
216             $("#cid").html(classesInfo);

```

```
217         }, "json")
218     }
219
220 </script>
```

6、 MyBatis分页插件-PageHelper使用

6.0) 添加分页的jar包：

```
> jsqlparser-0.9.1.jar
> junit-4.9.jar
> log4j-1.2.17.jar
> mybatis-3.4.6.jar
> mybatis-spring-1.3.2.jar
> mysql-connector-java-5.1.7-bin.jar
> pagehelper-3.4.2-fix.jar
```

6.1) 在resource/spring/applicationContext.xml文件中定义分页插件：

```
1 <!--4.配置SqlSessionFactoryBean-->
2 <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
3     <property name="dataSource" ref="dataSource"/>
4     <property name="typeAliasesPackage" value="com.zelin.pojo"/>
5     <!--配置mapper的映射文件的扫描位置-->
6     <property name="mapperLocations" value="classpath*:mapper/*.xml"/>
7     <!--配置分页插件-->
8     <property name="plugins">
9         <array>
10             <bean class="com.github.pagehelper.PageHelper">
11                 <!--配置数据库的方言-->
12                 <property name="properties">
13                     <value>
14                         dialect=mysql
15                     </value>
16                 </property>
17             </bean>
18         </array>
19     </property>
20 </bean>
```

6.2) 在StudentService接口及实现类中完成分页功能：

```
1 public PageResult<StudentCustom> findByPage(int page);
2
3 @Override
4 public PageResult<StudentCustom> findByPage(int page) {
5     try {
6         //1.开始分页
7         PageHelper.startPage(page, pageSize); //参数1：当前页 参数2：每页大小
8         //2.分页
```

```

9         Page<StudentCustom> pagelist = (Page<StudentCustom>) findAll();
10        //pagelist.getTotal():得到总记录数
11        //pagelist.getPages():得到总页数
12        //pagelist.getResult():得到每页的结果集
13        return new PageResult<>(pagelist.getTotal(), pagelist.getPages(),
pagelist.getResult());
14    } catch (Exception e) {
15        e.printStackTrace();
16        return null;
17    }
18 }

```

6.3) 在StudentServlet.java中完成分页功能：

```

1    //分页查询
2    private void findByPage(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
3        try {
4            //0.得到某一页的页码
5            String pageStr = request.getParameter("page");
6            int page = 1;
7            if(!StringUtils.isEmpty(pageStr)){
8                page = Integer.parseInt(pageStr);
9            }
10           //1.得到所有的学生对象
11           PageResult<StudentCustom> result = studentService.findByPage(page);
12           //2.将学生对象转换为json串并输出
13           response.getWriter().print(JSON.toJSONString(result));
14           response.getWriter().close();
15       } catch (Exception e) {
16           e.printStackTrace();
17       }
18 }

```

6.4) 在index.html中完成分页：

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</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         </table>
38         <div class="panel-footer clearfix" style="text-align:right;padding:10px;">
39             <a href="javascript:add()" class="btn btn-sm btn-warning pull-left
40 glyphicon glyphicon-plus-sign"
41             id="btn_add">添加学生</a>
42             <!--分页导航-->
43             <nav aria-label="Page navigation">
44                 <ul class="pagination">
45                     <li class="first">
46                         <a href="#" aria-label="Previous">
47                             <span aria-hidden="true">上一页</span>
48                         </a>
49                     </li>
50
51                     <li>
52                         <a href="#" aria-label="Next">
53                             <span aria-hidden="true">下一页</span>
54                         </a>
55                     </li>
56                 </ul>
57             </nav>
58             泽林信息版权所有 2000-2018.
59         </div>
60     </div>
61
62 </div>
63 <!-- 模态框（添加或修改学生） -->
64 <div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-
65 labelledby="myModalLabel">
66     <div class="modal-dialog" role="document">
67         <div class="modal-content">

```

```

67         <div class="modal-header">
68             <button type="button" class="close" data-dismiss="modal" aria-
label="Close"><span aria-hidden="true">&times;</span>
69         </button>
70         <h4 class="modal-title" id="myModalLabel">编辑学生信息</h4>
71     </div>
72     <div class="modal-body">
73         <form class="form-horizontal" id="form1">
74             <input type="hidden" name="sid" id="sid">
75             <div class="form-group">
76                 <label class="col-sm-2 control-label">学生姓名:</label>
77                 <div class="col-sm-10">
78                     <input type="text" class="form-control" placeholder="输
入姓名" name="sname" id='sname'>
79                 </div>
80             </div>
81             <div class="form-group">
82                 <label class="col-sm-2 control-label">性别:</label>
83                 <div class="radio">
84                     <label>
85                         <input type="radio" name="sex" value="男">男
86                     </label>
87                     <label>
88                         <input type="radio" name="sex" value="女">女
89                     </label>
90                 </div>
91             </div>
92             <div class="form-group">
93                 <label class="col-sm-2 control-label">学生年龄:</label>
94                 <div class="col-sm-10">
95                     <input type="text" class="form-control" placeholder="输
入年龄" name="age" id="age">
96                 </div>
97             </div>
98             <div class="form-group">
99                 <label class="col-sm-2 control-label">学生住址:</label>
100                 <div class="col-sm-10">
101                     <input type="text" class="form-control" placeholder="输
入住址" name="addr" id="addr">
102                 </div>
103             </div>
104             <div class="form-group">
105                 <label class="col-sm-2 control-label">生日:</label>
106                 <div class="col-sm-10">
107                     <input type="text" class="form-control" placeholder="输
入生日" name="birth" id="birth">
108                 </div>
109             </div>
110             <div class="form-group">
111                 <label class="col-sm-2 control-label">班级:</label>
112                 <div class="col-sm-10">
113                     <select class="form-control" name="cid" id="cid">

```

```

115         </select>
116     </div>
117 </div>
118
119     </form>
120 </div>
121 <div class="modal-footer">
122     <button type="button" class="btn btn-default" data-dismiss="modal">
关闭</button>
123     <button type="button" class="btn btn-primary" id="btn_save"
onclick="save()">保存</button>
124 </div>
125 </div>
126 </div>
127 </div>
128 </body>
129 </html>
130 <script>
131     var flag = true;
132     $(function () {
133         //刷新列表
134         reloadList(1);
135         //显示导航栏
136         //显示分页栏
137
138     })
139     //添加学生
140     function add() {
141         $('#form1')[0].reset();
142         //1.弹出对话框
143         $("#myModal").modal("show");
144     }
145     //保存学生(因为添加/修改使用的是同一对话框，所以，要判断当前是添加还是修改)
146     function save(){
147         //0.判断表单隐藏域中的sid是否存在值
148         var url = "student?method=update"
149         if(!$("#sid").val()){ //主键没有值就是执行添加操作
150             url = "student?method=add";
151         }
152         //1.得到表单数据并提交
153         $.post(
154             url,
155             $("#form1").serialize(),
156             function(data){
157                 if(data.success){
158                     reloadList(1);
159                 }else{
160                     alert(data.message);
161                 }
162                 //关闭对话框
163                 $("#myModal").modal("hide");
164             }, "json"
165         )

```

```

166     }
167     //修改学生
168     function updateStudent(v){
169         $('#form1')[0].reset();
170         //1.显示对话框
171         $("#myModal").modal("show");
172         //2.到后台查询学生
173         $.post(
174             "student?method=findOne&sid="+v,
175             function(data){
176                 var dt = new Date(data.birth);
177                 var y = dt.getFullYear();
178                 var m = dt.getMonth() + 1;
179                 var day = dt.getDate();
180                 var birth = y + "-" + m + "-" + day;
181                 //将得到的学生对象显示到表单中
182                 $("#sid").val(data.sid);
183                 $("#sname").val(data.sname);
184                 $("#age").val(data.age);
185                 $("#addr").val(data.addr);
186                 $("#birth").val(birth);
187                 $("#sname").val(data.sname);
188                 $("#cid").val(data.cid);
189                 //设置默认选择性别
190                 $('#input[name=sex]').each(function(i,v){
191                     $(this).attr("checked",$(v).val() == data.sex);
192                 })
193             }, "json"
194         )
195     }
196 }
197 //刷新列表
198 function reloadList(i) {
199     //1、发出异步请求加载所有的学生列表
200     $.post(
201         'student?method=findByPage&page='+i,
202         function (data) {
203             var info = "";
204             $.each(data.list, function (i, v) {
205                 var dt = new Date(v.birth);
206                 var y = dt.getFullYear();
207                 var m = dt.getMonth() + 1;
208                 var day = dt.getDate();
209                 var birth = y + "-" + m + "-" + day;
210                 info += "<tr>";
211                 info += "<td>" + v.sid + "</td>";
212                 info += "<td>" + v.sname + "</td>";
213                 info += "<td>" + v.sex + "</td>";
214                 info += "<td>" + v.age + "</td>";
215                 info += "<td>" + v.addr + "</td>";
216                 info += "<td>" + birth + "</td>";
217                 info += "<td>" + v.cname + "</td>";
218
219                 info += "<td>";

```

```

219         info += "<a href='javascript:;' "
onclick='updateStudent("+v.sid+")' class='btn btn-sm btn-info glyphicon glyphicon-
pencil'>修改</a>&nbsp;"
220         info += "<a href='javascript:;' class='btn btn-sm btn-danger
glyphicon glyphicon-trash' onclick='deleStudent("+v.sid+")'>删除</a>"
221         info += "</td>";
222         info += "</tr>";
223     })
224     //放上面的字符串放到tbody中
225     $("#tb").html(info);
226     if(flag){ //导航条输出的代码只执行一次
227         var pageInfo = "";
228         for(var i = 1;i <= data.totalPage;i++){
229             pageInfo += "<li><a href='#'
onclick='reloadList("+i+")'>" + i + "</a></li>";
230         }
231         $(".first").after(pageInfo);
232     }
233     flag = false;
234 }, "json"
235 );
236 //2.发出异步请求,得到所有的班级列表
237 $.post(
238     'classes?method=list',
239     function (data) {
240         var classesInfo = "";
241         $(data).each(function (i, v) {
242             classesInfo += "<option value='" + v.cid + "'">" + v.cname + "
</option>";
243         })
244         $("#cid").html(classesInfo);
245     }, "json")
246 }
247 function toPage(v){
248
249 }
250 </script>

```