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

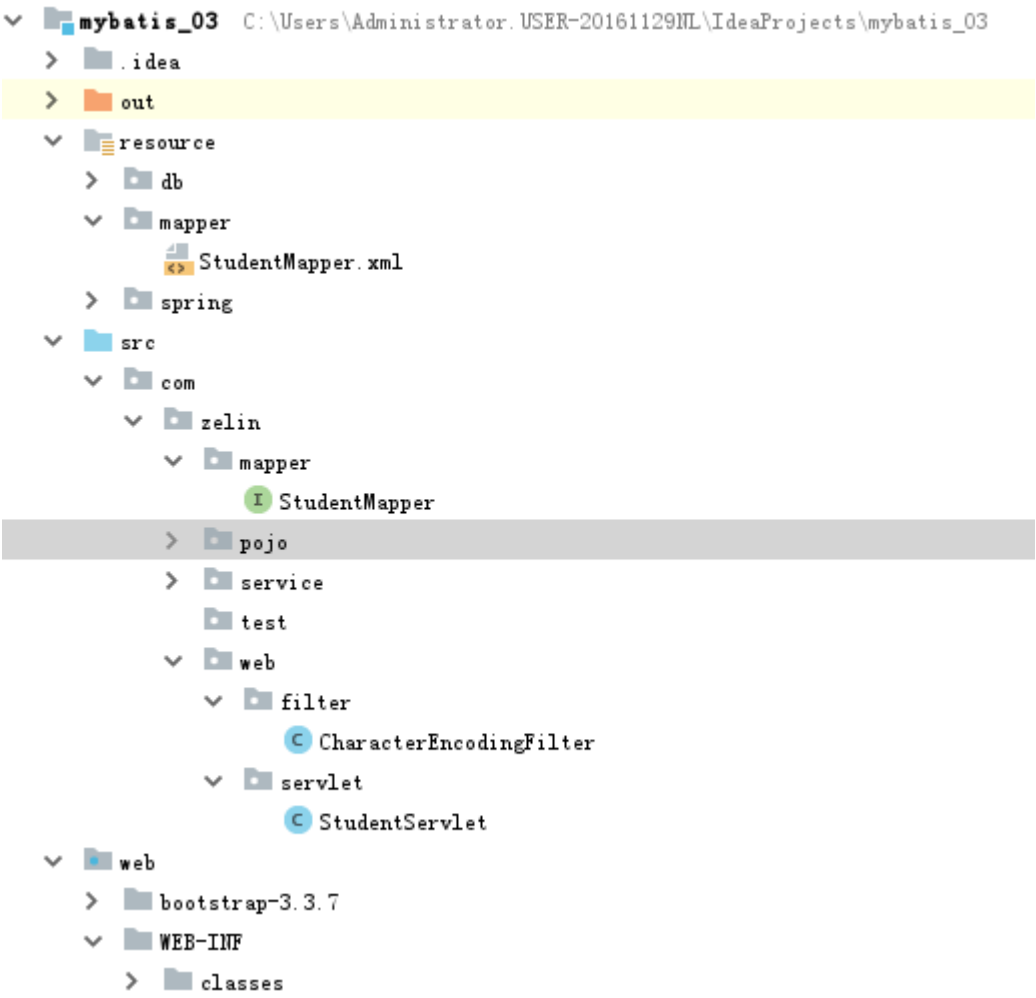
授课内容

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

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

3.3) 定义resource/spring/applicationContex.xml文件

```
<?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"
       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">
    <!--1.读取属性文件-->
    <context:property-placeholder location="classpath*:db/db.properties"/>
    <!--2.指定自动扫描包-->
    <context:component-scan base-package="com.zelin"/>
    <!--3.配置数据源-->

    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
```

```
        <property name="driverClassName" value="${jdbc.driver}"/>
        <property name="url" value="${jdbc.url}"/>
        <property name="username" value="${jdbc.user}"/>
        <property name="password" value="${jdbc.password}"/>
    </bean>
    <!--4.配置SqlSessionFactoryBean-->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="typeAliasesPackage" value="com.zelin.pojo"/>
        <!--配置mapper的映射文件的扫描位置-->
        <property name="mapperLocations" value="classpath*:mapper/*.xml"/>
    </bean>
    <!--5.配置MapperScannerConfigurer-->
    <bean id="mapperScannerConfigurer"
class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!--配置mapper接口的位置-->
        <property name="basePackage" value="com.zelin.mapper"/>
    </bean>
</beans>
```

3.4) 定义resource/mapper/StudentMapper.xml文件

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.zelin.mapper.StudentMapper">
    <!--1.查询所有学生-->
    <select id="findAll" resultType="studentCustom">
        select st.*,cname
        from student st,classes c
        where st.cid=c.cid
    </select>
</mapper>
```

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

```
public interface StudentMapper {
    public List<StudentCustom> findAll() throws  Exception;
}
```

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

```
public interface StudentService {
    public List<StudentCustom> findAll() throws  Exception;
}
```

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

```java
@Service
public class StudentServiceImpl implements StudentService {
    @Autowired
    private StudentMapper studentMapper;
    @Override
    public List<StudentCustom> findAll() throws Exception {
        return studentMapper.findAll();
    }
}

```

3.8) 定义web.xml文件

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

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

```java
@WebFilter(urlPatterns = "/*")
public class CharacterEncodingFilter implements Filter {
    public void destroy() {
    }

    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain)
throws ServletException, IOException {
        req.setCharacterEncoding("UTF-8");
        chain.doFilter(req, resp);
    }

    public void init(FilterConfig config) throws ServletException {

    }
}
```

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

```java
@WebServlet("/student")
public class StudentServlet extends HttpServlet {
    private StudentService studentService;
    @Override
    public void init() throws ServletException {
        WebApplicationContext webApplicationContext =
WebApplicationContextUtils.getWebApplicationContext(getServletContext());
        studentService = webApplicationContext.getBean(StudentService.class);
    }
    protected void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        //1.得到请求参数
        String method = request.getParameter("method");
        //2.根据请求参数执行不同的方法
        if(!StringUtils.isEmpty(method)){
            if("list".equals(method)){          //列表学生
                list(request,response);
            }
        }
    }
    //列表学生
    private void list(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        try {
            //1.得到所有的学生对象
            List<StudentCustom> students = studentService.findAll();
            //2.将学生对象转换为json串并输出
            response.getWriter().print(JSON.toJSONString(students));
            response.getWriter().close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

3.11) 定义/web/index.html

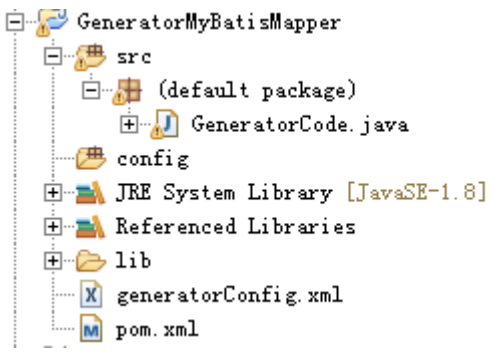
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="bootstrap-3.3.7/css/bootstrap.min.css">
    <script src="bootstrap-3.3.7/js/jquery.min.js"></script>
    <script src="bootstrap-3.3.7/js/bootstrap.min.js"></script>
    <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>

</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?method=list',
            function (data) {
                var info = "";
                $.each(data, function (i, v) {
                    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>" + v.birth + "</td>";
                    info += "<td>" + v.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?method=list',
            function (data) {
                var classesInfo = "";
                $(data).each(function (i, v) {
                    classesInfo += "<option value='" + v.cid + "'>" + v.cname + "</option>";
                })
                $("#cid").html(classesInfo);
            }, "json")
    }

</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

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

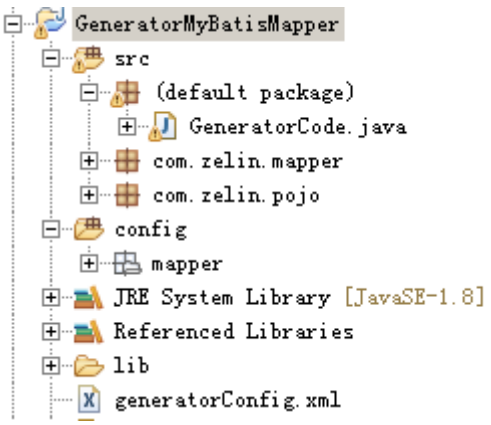
```
<!DOCTYPE generatorConfiguration
  PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
  "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">
<generatorConfiguration>
    <context id="testTables" targetRuntime="MyBatis3">
        <!-- 添加此插件可以让生成的实体类自动实现Serializable接口 -->
        <plugin type="org.mybatis.generator.plugins.SerializablePlugin" />
        <commentGenerator>
            <!-- 是否去除自动生成的注释 true：是 ： false:否 -->
            <property name="suppressAllComments" value="true" />
        </commentGenerator>
        <!--数据库连接的信息：驱动类、连接地址、用户名、密码 -->
        <jdbcConnection driverClass="com.mysql.jdbc.Driver"
            connectionURL="jdbc:mysql:///java1301"
            userId="root" password="123">
        </jdbcConnection>
        <!-- Oracle数据库配置 -->
        <!-- <jdbcConnection driverClass="oracle.jdbc.OracleDriver"
            connectionURL="jdbc:oracle:thin:@127.0.0.1:1521:orcl"
            userId="scott"
            password="tiger">
        </jdbcConnection> -->

        <!-- 默认false，把JDBC DECIMAL 和 NUMERIC 类型解析为 Integer，为 true时把JDBC DECIMAL 和
            NUMERIC 类型解析为java.math.BigDecimal -->
        <javaTypeResolver>
            <property name="forceBigDecimals" value="false" />
        </javaTypeResolver>

        <!-- targetProject:生成POJO类的位置 -->
        <javaModelGenerator targetPackage="com.zelin.pojo"
            targetProject=".\src">
            <!-- enableSubPackages:是否让schema作为包的后缀 -->
            <property name="enableSubPackages" value="false" />
            <!-- 从数据库返回的值被清理前后的空格 -->
            <property name="trimStrings" value="true" />
        </javaModelGenerator>
        <!-- targetProject:mapper映射XML文件生成的位置 -->
        <sqlMapGenerator targetPackage="mapper"
            targetProject=".\config">
            <!-- enableSubPackages:是否让schema作为包的后缀 -->
            <property name="enableSubPackages" value="false" />
        </sqlMapGenerator>
        <!-- targetPackage：mapper映射的接口生成的位置 -->
        <javaClientGenerator type="XMLMAPPER"
            targetPackage="com.zelin.mapper"
            targetProject=".\src">
            <!-- enableSubPackages:是否让schema作为包的后缀 -->
            <property name="enableSubPackages" value="false" />
        </javaClientGenerator>
        <!-- 指定数据库表 -->

        <table schema="" tableName="book"></table>
```

```
        <table schema="" tableName="category"></table>
    </context>
</generatorConfiguration>
```

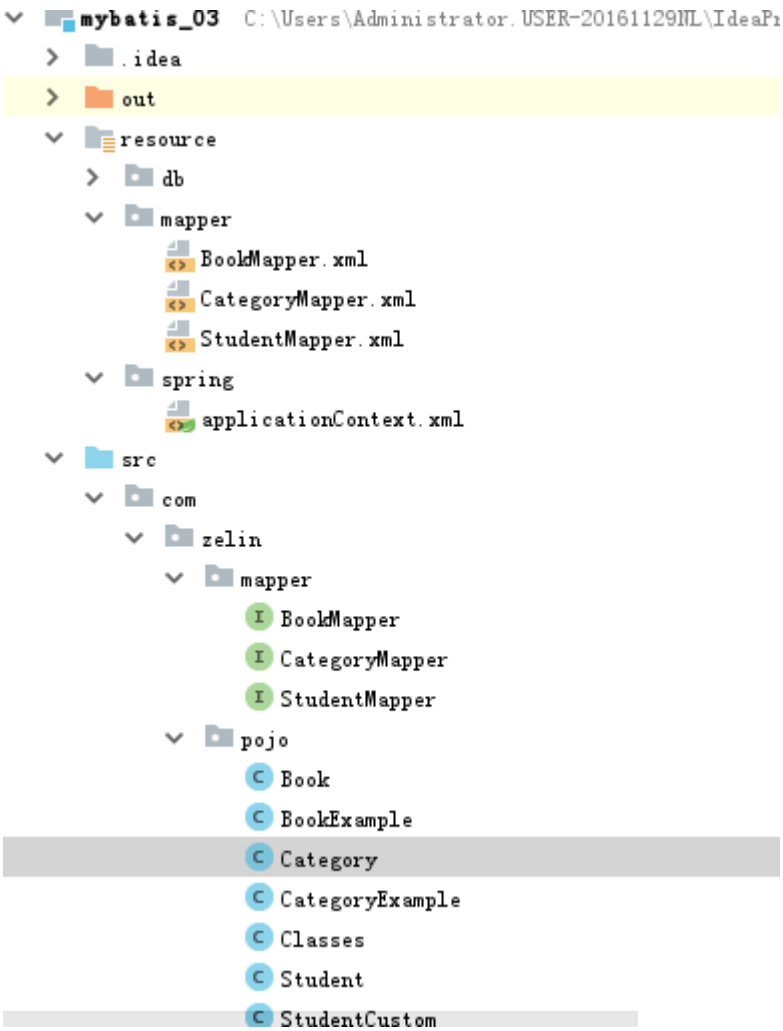
4.3)定义自动生成的java类：

```
public class GeneratorCode {

    public static void main(String[] args) throws Exception {
        generator();
        System.out.println("生成成功！");
    }
    private static void generator() throws Exception {
        List<String> warnings = new ArrayList<String>();
        boolean overwrite = true;
        //这里的配置文件一定要放到项目的根目录下
        File configFile = new File("generatorConfig.xml");
        ConfigurationParser cp = new ConfigurationParser(warnings);
        Configuration config = cp.parseConfiguration(configFile);
        DefaultShellCallback callback = new DefaultShellCallback(overwrite);
        MyBatisGenerator myBatisGenerator = new MyBatisGenerator(config, callback,
warnings);
        myBatisGenerator.generate(null);
    }
}
```

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：

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("classpath*:spring/applicationContext*.xml")
public class TestBookMapper {
    @Autowired
    private BookMapper bookMapper;
    @Test
    public void testFindAll() throws  Exception{
        List<Book> books = bookMapper.selectByExample(null);
        for(Book book : books){
            System.out.println(book);
        }
    }
}
```

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：

```
//接口
public interface StudentService {
```

```
    public List<StudentCustom> findAll() throws  Exception;

    void insert(Student student);

    Student findOne(String sid);

    void update(Student student);
}
//实现类
@Service
public class StudentServiceImpl implements StudentService {
    @Autowired
    private StudentMapper studentMapper;
    @Override
    public List<StudentCustom> findAll() throws Exception {
        return studentMapper.findAll();
    }

    @Override
    public void insert(Student student) {
        studentMapper.insert(student);
    }

    @Override
    public Student findOne(String sid) {
        return studentMapper.selectByPrimaryKey(Integer.valueOf(sid));
    }

    @Override
    public void update(Student student) {
        studentMapper.updateByPrimaryKey(student);
    }
}
```

5.2) 定义StudentServlet.java

```
@WebServlet("/student")
public class StudentServlet extends HttpServlet {
    private StudentService studentService;
    @Override
    public void init() throws ServletException {
        WebApplicationContext webApplicationContext =
WebApplicationContextUtils.getWebApplicationContext(getServletContext());
        studentService = webApplicationContext.getBean(StudentService.class);
    }
    protected void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        response.setContentType("text/html;charset=utf-8");
        //1.得到请求参数
        String method = request.getParameter("method");
        //2.根据请求参数执行不同的方法

        if(!StringUtils.isEmpty(method)){
```

```
            if("list".equals(method)){          //列表学生
                list(request,response);
            }else if("add".equals(method)){     //添加学生
                add(request,response);
            }else if("findOne".equals(method)){//到后台查询学生
                findOne(request,response);
            }else if("update".equals(method)){ //修改学生
                update(request,response);
            }
        }
    }

    private void update(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        Result result = null;
        try {
            //1.得到表单数据
            Student student = new Student();
            DateConverter dateConverter = new DateConverter();
            dateConverter.setPattern("yyyy-MM-dd");
            ConvertUtils.register(dateConverter,Date.class);
            //2.封装得到的表单数据
            BeanUtils.populate(student,request.getParameterMap());
            studentService.update(student);
            result = new Result(true,"修改成功");

        } catch (Exception e) {
            e.printStackTrace();
            result = new Result(false,"修改失败");
        }
        response.getWriter().print(JSON.toJSONString(result));
        response.getWriter().close();
    }

    //根据学生编号查询学生
    private void findOne(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        //1.得到学生编号
        String sid = request.getParameter("sid");
        //2.根据学生编号查询学生对象
        Student student = studentService.findOne(sid);
        //2.将学生对象转换为json串并输出
        response.getWriter().print(JSON.toJSONString(student));
        response.getWriter().close();
    }


    //添加学生
    private void add(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        Result result = null;
        try {

            //1.得到表单数据
```

```
            Student student = new Student();
            DateConverter dateConverter = new DateConverter();
            dateConverter.setPattern("yyyy-MM-dd");
            ConvertUtils.register(dateConverter,Date.class);
            //2.封装得到的表单数据
            BeanUtils.populate(student,request.getParameterMap());
            //3.提交数据到数据库中
            studentService.insert(student);
            result = new Result(true,"添加成功");
        } catch (Exception e) {
            e.printStackTrace();
            result = new Result(false,"添加失败");
        }
        response.getWriter().print(JSON.toJSONString(result));
        response.getWriter().close();
    }

    //列表学生
    private void list(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        try {
            //1.得到所有的学生对象
            List<StudentCustom> students = studentService.findAll();
            //2.将学生对象转换为json串并输出
            response.getWriter().print(JSON.toJSONString(students));
            response.getWriter().close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

5.3) 定义index.html页面：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</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" onclick="save()">保存</button>
            </div>
        </div>
    </div>
</div>
</body>
</html>

<script>
```

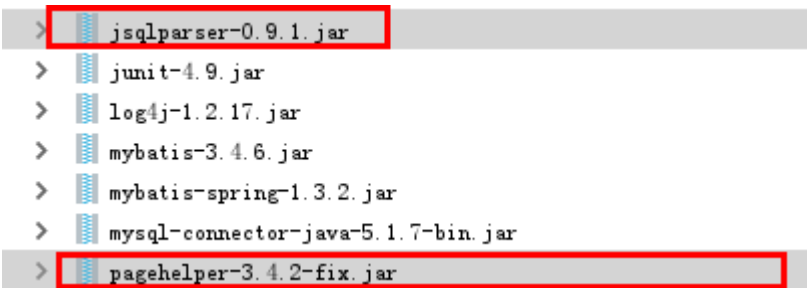
```
    $(function () {
        //刷新列表
        reloadList();
    })
    //添加学生
    function add() {
        $('#form1')[0].reset();
        //1.弹出对话框
        $("#myModal").modal("show");
    }
    //保存学生(因为添加/修改使用的是同一对话框，所以，要判断当前是添加还是修改)
    function save(){
        //0.判断表单隐藏域中的sid是否存在值
        var url = "student?method=update"
        if(!$("#sid").val()){       //主键没有值就是执行添加操作
            url = "student?method=add";
        }
        //1.得到表单数据并提交
        $.post(
            url,
            $("#form1").serialize(),
            function(data){
                if(data.success){
                    reloadList();
                }else{
                    alert(data.message);
                }
                //关闭对话框
                $("#myModal").modal("hide");
            },"json"
        )
    }
    //修改学生
    function updateStudent(v){
        $('#form1')[0].reset();
        //1.显示对话框
        $("#myModal").modal("show");
        //2.到后台查询学生
        $.post(
            "student?method=findOne&sid="+v,
            function(data){
                var dt = new Date(data.birth);
                var y = dt.getFullYear();
                var m = dt.getMonth() + 1;
                var day = dt.getDate();
                var birth = y + "-" + m + "-" + day;
                //将得到的学生对象显示到表单中
                $("#sid").val(data.sid);
                $("#sname").val(data.sname);
                $("#age").val(data.age);
                $("#addr").val(data.addr);
                $("#birth").val(birth);

                $("#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"
        )
    }
    //刷新列表
    function reloadList() {
        //1、发出异步请求加载所有的学生列表
        $.post(
            'student?method=list',
            function (data) {
                var info = "";
                $.each(data, function (i, v) {
                    var dt = new Date(v.birth);
                    var y = dt.getFullYear();
                    var m = dt.getMonth() + 1;
                    var day = dt.getDate();
                    var birth = y + "-" + m + "-" + day;
                    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>" + birth + "</td>";
                    info += "<td>" + v.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?method=list',
            function (data) {
                var classesInfo = "";
                $(data).each(function (i, v) {
                    classesInfo += "<option value='" + v.cid + "'>" + v.cname + "</option>";
                })

                $("#cid").html(classesInfo);
```

```
            }, "json")
    }

</script>
```

6、 MyBatis分页插件-PageHelper使用

6.0) 添加分页的jar包：



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

```
<!--4.配置SqlSessionFactoryBean-->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="typeAliasesPackage" value="com.zelin.pojo"/>
        <!--配置mapper的映射文件的扫描位置-->
        <property name="mapperLocations" value="classpath*:mapper/*.xml"/>
        <!--配置分页插件-->
        <property name="plugins">
            <array>
                <bean class="com.github.pagehelper.PageHelper">
                    <!--配置数据库的方言-->
                    <property name="properties">
                        <value>
                            dialect=mysql
                        </value>
                    </property>
                </bean>
            </array>
        </property>
    </bean>
```

6.2) 在StudentService接口及实现类中完成分页功能：

```
public PageResult<StudentCustom> findByPage(int page);

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

```
            Page<StudentCustom> pageList = (Page<StudentCustom>) findAll();
            //pageList.getTotal():得到总记录数
            //pageList.getPages():得到总页数
            //pageList.getResult():得到每页的结果集
            return new PageResult<>(pageList.getTotal(), pageList.getPages(),
pageList.getResult());
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
```

6.3) 在StudentServlet.java中完成分页功能：

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

6.4) 在index.html中完成分页：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</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>
            <!--分页导航-->
            <nav aria-label="Page navigation">
                <ul class="pagination">
                    <li class="first">
                        <a href="#" aria-label="Previous">
                            <span aria-hidden="true">上一页</span>
                        </a>
                    </li>

                    <li>
                        <a href="#" aria-label="Next">
                            <span aria-hidden="true">下一页</span>
                        </a>
                    </li>
                </ul>
            </nav>
            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" onclick="save()">保存</button>
            </div>
        </div>
    </div>
</div>
</body>
</html>
<script>
    var flag = true;
    $(function () {
        //刷新列表
        reloadList(1);
        //显示导航栏
        //显示分页栏

    })
    //添加学生
    function add() {
        $('#form1')[0].reset();
        //1.弹出对话框
        $("#myModal").modal("show");
    }
    //保存学生(因为添加/修改使用的是同一对话框，所以，要判断当前是添加还是修改)
    function save(){
        //0.判断表单隐藏域中的sid是否存在值
        var url = "student?method=update"
        if(!$("#sid").val()){       //主键没有值就是执行添加操作
            url = "student?method=add";
        }
        //1.得到表单数据并提交
        $.post(
            url,
            $("#form1").serialize(),
            function(data){
                if(data.success){
                    reloadList(1);
                }else{
                    alert(data.message);
                }
                //关闭对话框
                $("#myModal").modal("hide");
            },"json"

        )
```

```
    }
    //修改学生
    function updateStudent(v){
        $('#form1')[0].reset();
        //1.显示对话框
        $("#myModal").modal("show");
        //2.到后台查询学生
        $.post(
            "student?method=findOne&sid="+v,
            function(data){
                var dt = new Date(data.birth);
                var y = dt.getFullYear();
                var m = dt.getMonth() + 1;
                var day = dt.getDate();
                var birth = y + "-" + m + "-" + day;
                //将得到的学生对象显示到表单中
                $("#sid").val(data.sid);
                $("#sname").val(data.sname);
                $("#age").val(data.age);
                $("#addr").val(data.addr);
                $("#birth").val(birth);
                $("#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"
        )
    }
    //刷新列表
    function reloadList(i) {
        //1、发出异步请求加载所有的学生列表
        $.post(
            'student?method=findByPage&page='+i,
            function (data) {
                var info = "";
                $.each(data.list, function (i, v) {
                    var dt = new Date(v.birth);
                    var y = dt.getFullYear();
                    var m = dt.getMonth() + 1;
                    var day = dt.getDate();
                    var birth = y + "-" + m + "-" + day;
                    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>" + birth + "</td>";
                    info += "<td>" + v.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);
                if(flag){   //导航条输出的代码只执行一次
                    var pageInfo = "";
                    for(var i = 1;i <= data.totalPage;i++){
                        pageInfo += "<li><a href='#' onclick='reloadList("+i+")'>"+i+"</a></li>";
                    }
                    $(".first").after(pageInfo);
                }
                flag = false;
            }, "json"
        );
        //2.发出异步请求，得到所有的班级列表
        $.post(
            'classes?method=list',
            function (data) {
                var classesInfo = "";
                $(data).each(function (i, v) {
                    classesInfo += "<option value='" + v.cid + "'>" + v.cname + "</option>";
                })
                $("#cid").html(classesInfo);
            }, "json")
    }
    function toPage(v){

    }
</script>
```