

# 第五章 Shiro+SpringBoot整合开发

泽林.王峰 [2019.4.17 星期三]

## 授课目标

- 1、上章回顾
- 2、Shiro与SpringBoot整合工程搭建
- 3、配置文件的编写
- 4、自定义realm的编写
- 5、自定义异常处理
- 6、重写shiro过滤器完成验证码验证
- 7、"记住我"功能完成

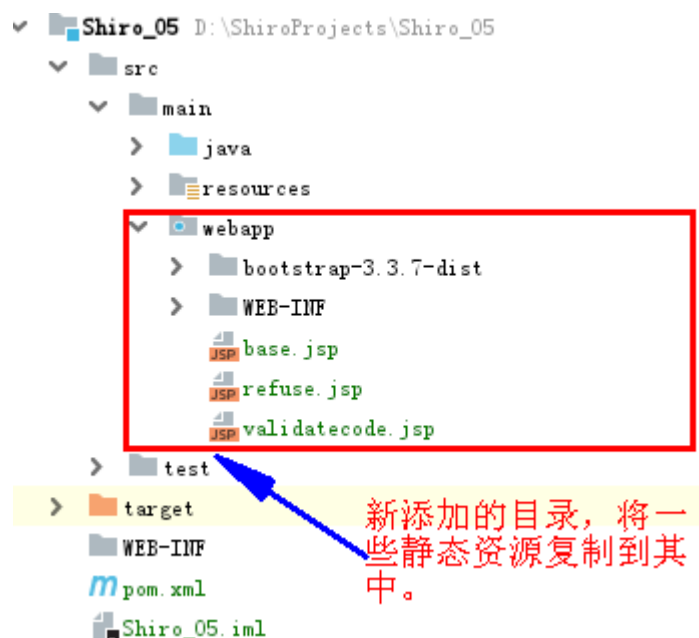
## 授课内容

### 1、上章回顾（略）

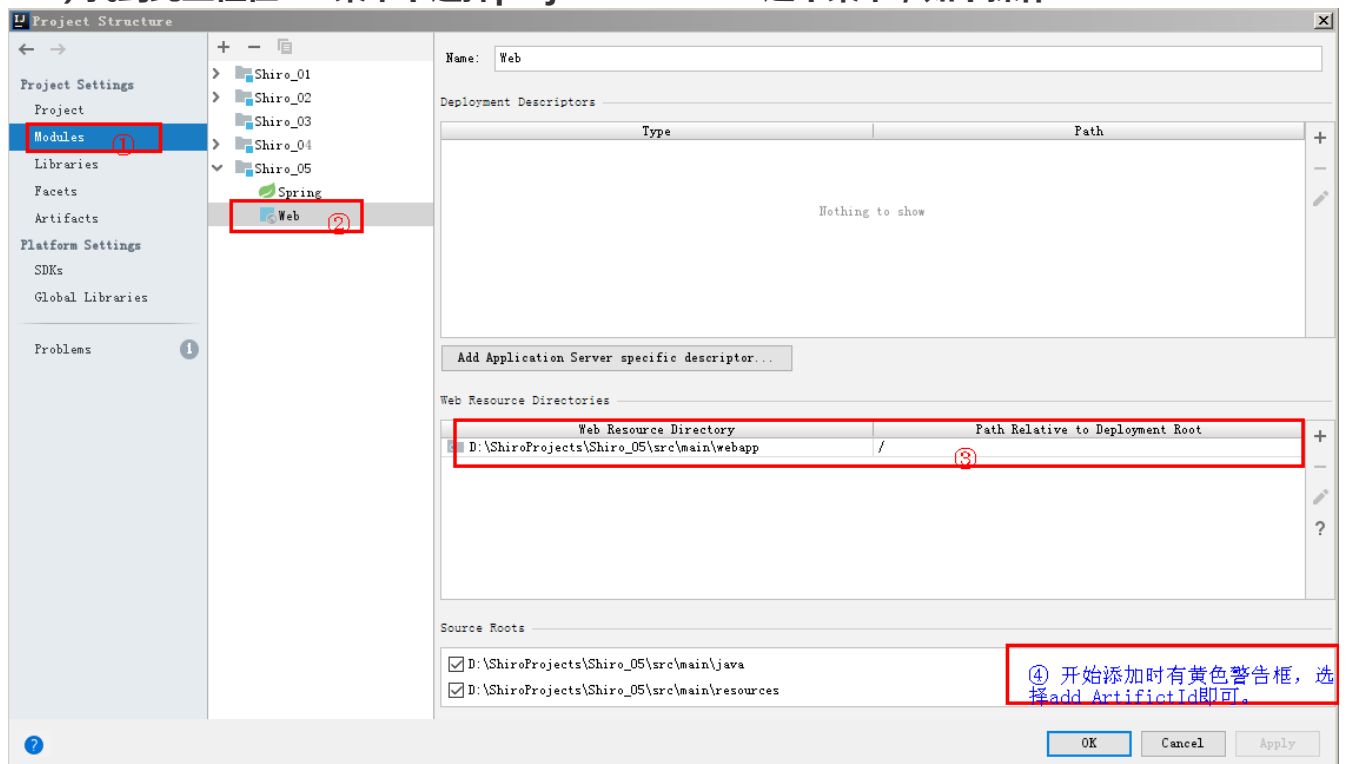
### 2、Shiro与SpringBoot整合工程搭建

2.1 ) 在当前的工程下新建一个普通的maven工程Shiro\_05。

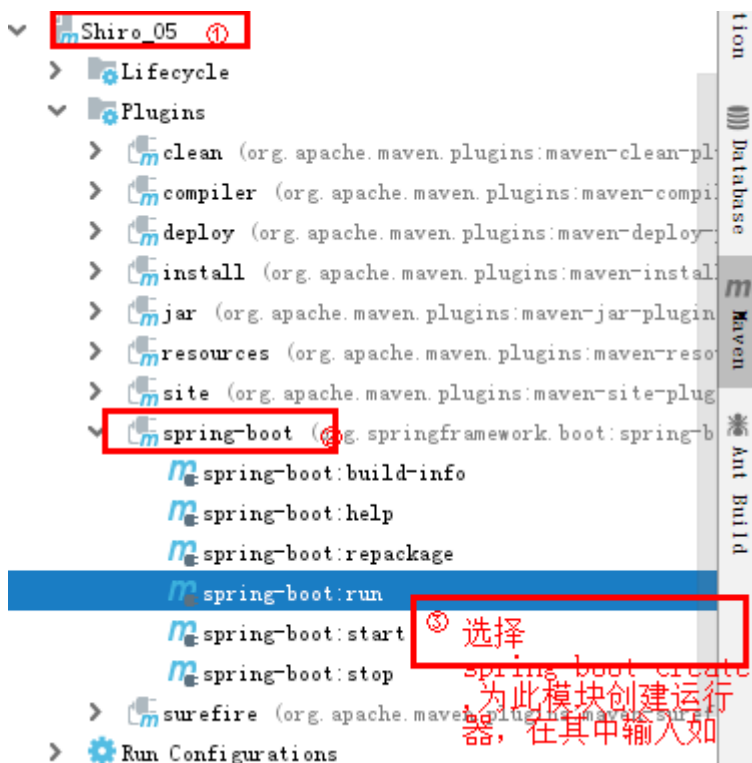
2.2 ) 在Shiro\_05工程下的src/main/添加webapp目录，如下图所示：



## 2.3 ) 找到此工程在File菜单下选择project structure这个菜单，如下操作：



## 2.4 ) 找到此工程在File菜单下选择project structure这个菜单，如下操作：



注意：上面的第③步输入的命令是: Springboot:run

## 2.5 ) 在pom.xml文件中引入相关的依赖

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
```

```
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
5         <modelVersion>4.0.0</modelVersion>
6         <groupId>com.zelin</groupId>
7         <artifactId>Shiro_05</artifactId>
8         <version>1.0-SNAPSHOT</version>
9         <!--1.配置下面的依赖所用到的父工程-->
10        <parent>
11            <groupId>org.springframework.boot</groupId>
12            <artifactId>spring-boot-starter-parent</artifactId>
13            <version>2.0.1.RELEASE</version>
14            <relativePath/> <!-- lookup parent from repository -->
15        </parent>
16        <properties>
17            <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
18            <java.version>1.8</java.version>
19        </properties>
20        <dependencies>
21            <dependency>
22                <groupId>org.springframework.boot</groupId>
23                <artifactId>spring-boot-starter</artifactId>
24            </dependency>
25            <!--2.与junit进行整合的starter-->
26            <dependency>
27                <groupId>org.springframework.boot</groupId>
28                <artifactId>spring-boot-starter-test</artifactId>
29                <scope>test</scope>
30            </dependency>
31            <!--3.代表springboot与web的整合的starter-->
32            <dependency>
33                <groupId>org.springframework.boot</groupId>
34                <artifactId>spring-boot-starter-web</artifactId>
35            </dependency>
36
37            <!--4.添加数据库-->
38            <dependency>
39                <groupId>mysql</groupId>
40                <artifactId>mysql-connector-java</artifactId>
41                <version>5.1.47</version>
42            </dependency>
43            <!--5.添加mybatis与springboot整合的依赖-->
44            <dependency>
45                <groupId>org.mybatis.spring.boot</groupId>
46                <artifactId>mybatis-spring-boot-starter</artifactId>
47                <version>2.0.1</version>
48            </dependency>
49            <!-- 6.添加mybatis分页插件与springboot整合的依赖 -->
50            <dependency>
51                <groupId>com.github.pagehelper</groupId>
52                <artifactId>pagehelper-spring-boot-starter</artifactId>
53                <version>1.2.10</version>
54            </dependency>
```

```
55 <!-- 7.添加通过mapper与springboot整合的依赖 -->
56 <dependency>
57     <groupId>tk.mybatis</groupId>
58     <artifactId>mapper-spring-boot-starter</artifactId>
59     <version>2.1.5</version>
60 </dependency>
61 <!--8.添加druid连接池-->
62 <dependency>
63     <groupId>com.alibaba</groupId>
64     <artifactId>druid</artifactId>
65     <version>1.0.14</version>
66 </dependency>
67 <dependency>
68     <groupId>org.apache.commons</groupId>
69     <artifactId>commons-lang3</artifactId>
70     <version>3.3.2</version>
71 </dependency>
72 <!-- 关于shiro相关 -->
73 <dependency>
74     <groupId>org.apache.shiro</groupId>
75     <artifactId>shiro-core</artifactId>
76     <version>1.3.2</version>
77 </dependency>
78 <dependency>
79     <groupId>org.apache.shiro</groupId>
80     <artifactId>shiro-web</artifactId>
81     <version>1.3.2</version>
82 </dependency>
83 <dependency>
84     <groupId>org.apache.shiro</groupId>
85     <artifactId>shiro-spring</artifactId>
86     <version>1.3.2</version>
87 </dependency>
88 <dependency>
89     <groupId>org.apache.shiro</groupId>
90     <artifactId>shiro-ehcache</artifactId>
91     <version>1.3.2</version>
92 </dependency>
93 <!--在springboot中添加热部署功能-->
94 <dependency>
95     <groupId>org.springframework.boot</groupId>
96     <artifactId>spring-boot-devtools</artifactId>
97 </dependency>
98 <!--添加对jsp的依赖-->
99 <dependency>
100     <groupId>jstl</groupId>
101     <artifactId>jstl</artifactId>
102     <version>1.2</version>
103 </dependency>
104 <!-- 使用jsp引擎，springboot内置tomcat没有此依赖 -->
105 <dependency>
106     <groupId>org.apache.tomcat.embed</groupId>
107     <artifactId>tomcat-embed-jasper</artifactId>
```

```

108         <scope>provided</scope>
109     </dependency>
110 </dependencies>
111 <build>
112     <plugins>
113         <!--如果使用springboot的热部署功能的话，需要添加如下的配置-->
114         <plugin>
115             <groupId>org.springframework.boot</groupId>
116             <artifactId>spring-boot-maven-plugin</artifactId>
117             <configuration>
118                 <fork>true</fork>
119             </configuration>
120         </plugin>
121     </plugins>
122 </build>
123 </project>

```

## 2.6 ) 定义resources/application.yml文件

```

1  server:
2      port: 9000
3      #配置spring相关
4  spring:
5      datasource:
6          driver-class-name: com.mysql.jdbc.Driver
7          url: jdbc:mysql:///shiro
8          username: root
9          password: 123
10     mvc:
11         view:
12             prefix: /WEB-INF/jsp/
13             suffix: .jsp
14     http:
15         encoding:
16             charset: UTF-8
17     #配置mybatis
18     mybatis:
19         mapper-locations: mapper/*.xml

```

## 2.7 ) 定义SpringBoot的启动器：

```

1  /**
2   * @Author: Feng.Wang
3   * @Company: Zelin.ShenZhen
4   * @Description:
5   * @Date: Create in 2019/4/17 09:29
6   */
7  @SpringBootApplication
8  @MapperScan("com.zelin.mapper")
9  public class ShiroApplication {
10     public static void main(String[] args) {
11         SpringApplication.run(ShiroApplication.class);
12     }
13 }

```

## 2.8 ) 定义异常与全局异常处理：

```

1  /**
2   * 自定义异常类
3   */
4  public class MyException extends RuntimeException {
5      private static final long serialVersionUID = 1L;
6      //代表异常的错误信息
7      private String message;
8      public String getMessage() {
9          return message;
10     }
11     public void setMessage(String message) {
12         this.message = message;
13     }
14     public MyException() {
15         super();
16     }
17     public MyException(String message) {
18         super();
19         this.message = message;
20     }
21 }
22

```

```

1  /**
2   * @Author: Feng.Wang
3   * @Company: Zelin.ShenZhen
4   * @Description:
5   * @Date: Create in 2019/4/16 09:56
6   */
7  @ControllerAdvice
8  public class MyExceptionHandler {
9      //定义异常处理器
10     @ExceptionHandler(MyException.class)
11     public String myException(MyException ex, Model model){
12         //1.得到异常出错信息

```

```

13     String message = ex.getMessage();
14     //2.如果信息为null,就指定一个message值
15     if(StringUtils.isEmpty(message)){
16         message = "未知异常!";
17     }
18     model.addAttribute("message",message);
19     //3.到错误页面进行处理(注意:这里的错误视图,不能为error,以免视图的错误视图冲突,报
timestamp找不到异常!)
20     return "errorpage";
21 }
22 }

```

## 2.9) 自定义realm :

```

1  package com.zelin.realm;
2
3  import com.zelin.pojo.SysPermission;
4  import com.zelin.pojo.SysUser;
5  import com.zelin.service.SysUserService;
6  import org.apache.shiro.authc.AuthenticationException;
7  import org.apache.shiro.authc.AuthenticationInfo;
8  import org.apache.shiro.authc.AuthenticationToken;
9  import org.apache.shiro.authc.SimpleAuthenticationInfo;
10 import org.apache.shiro.authz.AuthorizationInfo;
11 import org.apache.shiro.authz.SimpleAuthorizationInfo;
12 import org.apache.shiro.realm.AuthorizingRealm;
13 import org.apache.shiro.subject.PrincipalCollection;
14 import org.apache.shiro.util.ByteSource;
15 import org.springframework.beans.factory.annotation.Autowired;
16 import org.springframework.stereotype.Component;
17
18 import java.util.ArrayList;
19 import java.util.List;
20
21 /**
22  * @Author: Feng.Wang
23  * @Company: Zelin.ShenZhen
24  * @Description:
25  * @Date: Create in 2019/4/16 09:21
26  */
27
28 public class CustomRealm extends AuthorizingRealm {
29     @Autowired
30     private SysUserService userService;
31     //用户授权
32     @Override
33     protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection
principals) {
34         //1.取得主身份信息
35         SysUser user = (SysUser) principals.getPrimaryPrincipal();
36         List<SysPermission> permissions = null;
37         if(null != user){
38             System.out.println("开始查询数据库...");

```

```

39         //2.取得此用户的权限列表
40         permissions =
userService.findPermissionsByUserCode(user.getUsercode());
41         if(permissions != null && permissions.size() > 0){
42             user.setPermissions(permissions);
43         }
44     }
45     //3.定义授权对象
46     SimpleAuthorizationInfo authorizationInfo = new SimpleAuthorizationInfo();
47     //4.将上面的用户的权限列表字符串赋值给当前的授权对象
48     List<String> permissionList = new ArrayList<>();
49     //5.转换上面的List<SysPermission>对象为List<String>对象
50     for (SysPermission permission : permissions) {
51         permissionList.add(permission.getPercode());
52     }
53     //6.将权限码列表与当前授权对象关联
54     authorizationInfo.addStringPermissions(permissionList);
55     //7.返回授权对象
56     return authorizationInfo;
57 }
58
59 //用户认证
60 @Override
61 protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token)
throws AuthenticationException {
62     //1.得到主身份信息
63     String usercode = (String) token.getPrincipal();
64     //2.如果用户名存在
65     if(usercode != null){
66         SysUser user = userService.findUserByUserCode(usercode);
67         if(user == null) return null;
68         //2.1)如果不为null,就查询其对应的菜单列表
69         List<SysPermission> menus = userService.findMenusByUserCode(usercode);
70         user.setMenus(menus);
71         //2.2)取出数据库中的密码及加盐值
72         String password = user.getPassword();
73         String salt = user.getSalt();
74         return new SimpleAuthenticationInfo(user,password,
ByteSource.Util.bytes(salt),"aaa");
75     }
76     return null;
77 }
78 }

```

## 2.10) 重写表单过滤器，为了校验验证码：

```

1  /**
2   * @Author: Feng.Wang
3   * @Company: Zelin.ShenZhen
4   * @Description: 自定义表单过滤器完成验证码的验证
5   * @Date: Create in 2019/4/16 15:27
6   */
7  public class CustomFormFilter extends FormAuthenticationFilter {

```



```

8      @Override
9      protected boolean onAccessDenied(ServletRequest request, ServletResponse response)
      throws Exception {
10         //1.将原始的请求与响应对象转换为基于http请求的对象
11         HttpServletRequest req = (HttpServletRequest) request;
12         HttpServletResponse resp = (HttpServletResponse) response;
13         //2.从session中取出原始的验证码
14         HttpSession session = req.getSession();
15         String validateCode = (String) session.getAttribute("validateCode");
16         //3.得到用户输入的验证码
17         String validcode = req.getParameter("validcode");
18         //4.比较两次验证码是否一样,如果两次验证码不相等,就将错误信息放到request对象中
19         if(StringUtils.isEmpty(validcode) && !validateCode.equals(validcode)){
20             req.setAttribute("shiroLoginFailure","validCodeError");
21             return true;           //代表验证未通过
22         }
23         return super.onAccessDenied(request, response);
24     }
25 }

```

2.11 ) 将pojo中的类实现序列化 , 否则 , 报错。

2.12 ) 定义登录的控制器login

```

1  /**
2   * @Author: Feng.Wang
3   * @Company: Zelin.Shenzhen
4   * @Description:
5   * @Date: Create in 2019/4/17 09:39
6   */
7  @Controller
8  public class LogController {
9      @RequestMapping("/login")
10         //默认情况下, shiro的用户认证工作由FormAuthenticationFilter(authc)过滤器完成,
11         //当出现异常时, shiro会将异常对象的类名放到以shiroLoginFailure为key的request
12         //对象中, 我们只需要通过此key取出对应的异常类名就可以知道, 用户是何种异常
13         public String login(HttpServletRequest request, HttpServletResponse response)
      throws Exception {
14             //1.得到异常的名称
15             String exceptionName = (String) request.getAttribute("shiroLoginFailure");
16             //2.根据异常的名称来判断执行的是何种异常, 从而处理此异常
17             //2.1)判断是否为null
18             if(StringUtils.isEmpty(exceptionName)){
19                 if(UnknownAccountException.class.getName().equals(exceptionName)){
20                     throw new MyException("账户异常!");
21                 }else
22             if(IncorrectCredentialsException.class.getName().equals(exceptionName)){
23                 throw new MyException("用户名或密码输入有误!");
24             }else if(UnauthorizedException.class.getName().equals(exceptionName)){
25                 throw new MyException("无权限访问异常!");
26             }else if("validCodeError".equals(exceptionName)){
27                 throw new MyException("验证码错误!");
28             }else{

```

```

28         throw new Exception();
29     }
30 }
31 //此方法只有在FormAuthenticationFilter认证失败时工作，认证成功返回页面就是上一次的页面
32 return "login";
33 }
34 }
35

```

## 2.13 ) 定义用户控制器

```

1  /**
2   * @Author: Feng.Wang
3   * @Company: Zelin.ShenZhen
4   * @Description:
5   * @Date: Create in 2019/4/17 10:53
6   */
7  @Controller
8  @RequestMapping("/user")
9  public class UserController {
10     @Autowired
11     private SysUserService userService;
12
13     /**
14      * 根据登录用户动态显示菜单
15      * @param model
16      * @return
17      */
18     @RequestMapping("/listmenu")
19     public String listmenu(Model model){
20         try {
21             //1.得到主体对象
22             Subject subject = SecurityUtils.getSubject();
23             //2.得到身份对象（当前登录的用户）
24             SysUser user = (SysUser) subject.getPrincipal();
25             //2.从此用户身上得到其有的菜单
26             List<SysPermission> menus = user.getMenus();
27             //3.将上面的用户菜单放到model中
28             model.addAttribute("menus",menus);
29
30         } catch (Exception e) {
31             e.printStackTrace();
32         }
33         return "user/listmenu";
34     }
35
36     /**
37      * 查询所有的用户
38      * @return
39      */
40     @RequiresPermissions("user:userlistxxx")
41     @RequestMapping("/userlist")
42     public String findAll(Model model){

```

```

43         try {
44             model.addAttribute("users",userService.findUsers());
45         } catch (Exception e) {
46             System.out.println("对不起，你无权访问此页面!");
47         }
48         return "user/list";
49     }
50 }
51

```

## 2.14 ) 定义学生控制器

```

1  /**
2   * @Author: Feng.Wang
3   * @Company: Zelin.ShenZhen
4   * @Description:
5   * @Date: Create in 2019/4/8 10:50
6   */
7  @Controller
8  @RequestMapping("/student")
9  public class StudentController {
10     @Autowired
11     private StudentService studentService;
12     @Autowired
13     private ClassesService classesService;
14     int pageSize = 5;          //代表每页的大小
15     //查询所有的学生
16     @RequestMapping("/tolist")
17     @RequiresPermissions("student:tolist")    //说明具有student:tolist权限才能访问/tolist
资源
18     public String findAll(Model model){
19         try {
20             //1.查询所有的学生集合
21             List<Student> students = studentService.findStudents();
22             System.out.println("----->" + students);
23             //2.将上面的集合放到model中
24             model.addAttribute("students",students);
25             model.addAttribute("classes",classesService.findAll());
26         } catch (Exception e) {
27             e.printStackTrace();
28         }
29         //4.返回逻辑视图
30         return "student/list";
31     }
32     /**
33     * 查询所有学生带分页功能
34     * @param page
35     * @return
36     */
37     @RequiresPermissions("student:listpage")
38     @RequestMapping("/listpage")
39     public String listPage(@RequestParam(defaultValue = "1") int page,Model model){
40         try {

```

```

41         model.addAttribute("pr", studentService.findAllStudents(page, pageSize));
42         model.addAttribute("classes", classesService.findAll());
43     } catch (Exception e) {
44         e.printStackTrace();
45     }
46     return "student/list";
47 }
48
49 /**
50  * 条件查询带分页
51  * @param page
52  * @param student
53  * @return
54  */
55 @RequestMapping("/search")
56 public String search(@RequestParam(defaultValue = "1") int page, Student student,
57 Model model){
58     try {
59         PageBean pageBean = studentService.search(page, pageSize, student);
60         model.addAttribute("pr", pageBean);
61         model.addAttribute("page", page);
62         model.addAttribute("classes", classesService.findAll());
63     } catch (Exception e) {
64         e.printStackTrace();
65     }
66     return "student/list";
67 }

```

## 2.15 ) 定义学生列表页面/WEB-INF/jsp/student/list.jsp

```

1  <%@ page contentType="text/html; charset=UTF-8" language="java" isELIgnored="false" %>
2  <%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
3  <%@taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
4  <%@taglib prefix="shiro" uri="http://shiro.apache.org/tags" %>
5  <html>
6  <jsp:include page="/base.jsp"/>
7  <head>
8      <meta charset="UTF-8">
9      <style>
10         .table {
11             text-align: center;
12         }
13         .container {
14             margin-top: 20px;
15         }
16         #form1 {
17             margin-top: 10px;
18             margin-left: 30px;
19         }
20     </style>
21 </head>
22 <body>

```

```
<div class="container">
<div class="panel panel-primary">
  <div class="panel-heading">
    <h3 class="panel-title">学生查询</h3>
  </div>

  <form class="form-inline" id="form1" method="post"
        action="${pageContext.request.contextPath}/student/search.do">
    <input type="hidden" name="page" id="page">
    学生姓名 : <input type="text" name="sname" value="${param.sname}"
placeholder="学生姓名关键字" class="form-control">
    学生住址 : <input type="text" name="addr" value="${param.addr}"
placeholder="学生住址关键字" class="form-control">
    所在班级 :
    <select name="cid" class="form-control">
      <option value="0">所有班级</option>
      <c:forEach items="${classes}" var="c">
        <option value="${c.cid}"
${c.cid==param.cid?"selected":""}>${c.cname}</option>
      </c:forEach>
    </select>
    <input type="submit" value="查询" class="btn btn-default btn-sm">
    <!--如果有student:create这个权限，则下面的超链接就会显示出来-->
    <shiro:hasPermission name="student:createxxx">
      <input type="button" value="添加学生" class="btn btn-success btn-sm" >
    </shiro:hasPermission>
    <a href="javascript:parent.location.href = '/logout.do'" class="btn btn-
success btn-sm">注销</a>
  </form>

  <table class="table table-bordered table-striped">
    <tr>
      <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>
    <c:forEach items="${students}" var="stud">
      <tr>
        <td>${stud.sname}</td>
        <td>${stud.sex}</td>
        <td>${stud.age}</td>
        <td>${stud.addr}</td>
        <td>
          <fmt:formatDate value="${stud.birth}" pattern="yyyy-MM-dd E"/>
        </td>
        <td>
          ${stud.classes.cname}
        </td>
        <td>
          <a class="btn btn-primary btn-sm"
```

```

72 href="${pageContext.request.contextPath}/student/toupdate.do?sid=${stud.sid}">修改</a>
73     <a class="btn btn-danger btn-sm"
74 href="${pageContext.request.contextPath}/student/deleteBySid.do?sid=${stud.sid}"
75     onclick="return confirm('你真的要删除吗?')">删除</a>
76     </td>
77 </tr>
78 </c:forEach>
79 </table>
80 </div>
81 </div>
82 <script>
83     //执行提交 表单
84     function skip(i) {
85         //1. 对表单中的隐藏域赋值
86         $("#page").val(i);
87         //2. 提交表单
88         $("#form1").submit();
89     }
90
91     //添加学生
92     function addStudent() {
93         location.href = "${pageContext.request.contextPath}/student/toadd.do";
94     }
95
96 </script>
97 </body>
98 </html>
99

```

## 2.16 ) 定义用户菜单显示/WEB-INF/jsp/user/listmenu.jsp

```

1  <%@ page contentType="text/html; charset=UTF-8" language="java" %>
2  <%@include file="/base.jsp"%>
3  <html>
4  <head>
5      <title>用户登录</title>
6      <style>
7          .container{
8              width: 500px;
9              margin-top: 50px;
10         }
11         .form-signin{
12             padding: 5px;
13         }
14         .btn{
15             margin-top: 20px;
16         }
17         .error{
18             color: red;
19         }
20     </style>

```

```

21 </head>
22 <body>
23 <div class="container">
24     <div class="panel panel-primary">
25         <div class="panel-heading">
26             <h3 class="panel-title">
27                 用户登录
28             </h3>
29         </div>
30         <div class="panel-body">
31             <form class="form-signin" action="${pageContext.request.contextPath}/login"
method="post">
32                 <label>用户名</label>
33                 <input type="text" name="username" class="form-control" placeholder="输入
用户名" required autofocus>
34                 <label>密码</label>
35                 <input type="password" name="password" class="form-control"
placeholder="输入密码" required>
36                 <label>输入验证码</label>
37                 <input type="text" name="validcode" style="width:360px" class="form-
control" placeholder="输入验证码">
38                 
39                 <div style="padding:0;margin-top: -20px;">
40                     <input type="checkbox" name="rememberMe">&nbsp;记住我
41                 </div>
42                 <button class="btn btn-lg btn-primary btn-block" type="submit">登录
</button>
43                 <span class="error">${message}</span>
44             </form>
45         </div>
46     </div>
47 </div> <!-- /container -->
48 </body>
49 </html>
50

```

## 2.17 ) 定义出现异常时处理页面/WEB-INF/jsp/user/errorpage.jsp

```

1 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
2 <html>
3 <head>
4     <title>异常处理页面</title>
5 </head>
6 <body>
7     <h1>异常信息 : <hr></h1>
8     <h2>${message}</h2>
9 </body>
10 </html>

```

## 2.18 ) 定义未授权时访问的页面/WEB-INF/jsp/refuse.jsp

```

1  <%@ page contentType="text/html;charset=UTF-8" language="java" %>
2  <!DOCTYPE html>
3  <html lang="en">
4  <head>
5      <meta charset="UTF-8">
6      <title>Access Control</title>
7  </head>
8  <body>
9      <h2>无访问权限！</h2>
10 </body>
11 </html>

```

## 2.19 ) 用户列表页面/WEB-INF/jsp/user/list.jsp

```

1  <%@ page contentType="text/html;charset=UTF-8" language="java" %>
2  <html>
3  <head>
4      <title>用户列表</title>
5  </head>
6  <body>
7      ${users}
8  </body>
9  </html>
10

```

## 2.20 ) 定义缓存文件resources/cache/shiro-cache.xml

```

1  <ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2      xsi:noNamespaceSchemaLocation="../config/ehcache.xsd">
3      <!--diskStore：缓存数据持久化的目录 地址 -->
4      <diskStore path="e:\develop\ehcache" />
5      <defaultCache
6          maxElementsInMemory="1000"
7          maxElementsOnDisk="10000000"
8          eternal="false"
9          overflowToDisk="false"
10         diskPersistent="false"
11         timeToIdleSeconds="120"
12         timeToLiveSeconds="120"
13         diskExpiryThreadIntervalSeconds="120"
14         memoryStoreEvictionPolicy="LRU">
15     </defaultCache>
16     <!--name：cache的名字，用来识别不同的cache，必须惟一。-->
17     <!--maxElementsInMemory：内存管理的缓存元素数量最大值。-->
18     <!--maxElementsOnDisk：硬盘管理的缓存元素数量最大值。默认值为0，就是没有限制。-->
19     <!--eternal：设定元素是否持久话。若设为true，则缓存元素不会过期。-->
20     <!--overflowToDisk：设定是否在内存填满的时候把数据转到磁盘上。-->
21 </ehcache>

```

## 2.21 ) 在webapp这个目录下定义validatecode.jsp这个生成验证码的jsp文件



```

1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      pageEncoding="UTF-8"%>
3  <%@ page import="java.util.Random"%>
4  <%@ page import="java.io.OutputStream"%>
5  <%@ page import="java.awt.Color"%>
6  <%@ page import="java.awt.Font"%>
7  <%@ page import="java.awt.Graphics"%>
8  <%@ page import="java.awt.image.BufferedImage"%>
9  <%@ page import="javax.imageio.ImageIO"%>
10 <%
11     int width = 60;
12     int height = 32;
13     //create the image
14     BufferedImage image = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
15     Graphics g = image.getGraphics();
16     // set the background color
17     g.setColor(new Color(0xDCDCDC));
18     g.fillRect(0, 0, width, height);
19     // draw the border
20
21     g.setColor(new Color(0x337ab7));
22     g.drawRect(0, 0, width - 1, height - 1);
23     // create a random instance to generate the codes
24     Random rdm = new Random();
25     String hash1 = Integer.toHexString(rdm.nextInt());
26     System.out.print(hash1);
27     // make some confusion
28     for (int i = 0; i < 50; i++) {
29         int x = rdm.nextInt(width);
30         int y = rdm.nextInt(height);
31         g.drawOval(x, y, 0, 0);
32     }
33     // generate a random code
34     String capstr = hash1.substring(0, 4);
35     //将生成的验证码存入session
36     session.setAttribute("validateCode", capstr);
37     g.setColor(new Color(0, 100, 0));
38     g.setFont(new Font("Candara", Font.BOLD, 24));
39     g.drawString(capstr, 8, 24);
40     g.dispose();
41     //输出图片
42     response.setContentType("image/jpeg");
43     out.clear();
44     out = pageContext.pushBody();
45     OutputStream strm = response.getOutputStream();
46     ImageIO.write(image, "jpeg", strm);
47     strm.close();
48 %>

```

### 3、本工程中关于Shiro的配置说明：

#### 3.1) 定义Shiro的相关配置信息，com.zelin.config包下

```

1  /**
2   * @Author: Feng.Wang
3   * @Company: Zelin.ShenZhen
4   * @Description: 配置ShiroFactoryFilterBean
5   * @Date: Create in 2019/4/17 09:28
6   */
7  @Configuration
8  public class ShiroConfig {
9      @Bean(name = "shiroFilter")
10     public ShiroFilterFactoryBean shiroFilterFactoryBean(SecurityManager
securityManager){
11         //1.定义ShiroFilterFactoryBean对象
12         ShiroFilterFactoryBean shiroFilter = new ShiroFilterFactoryBean();
13         //2.将securityManager与ShiroFilterFactoryBean对象绑定
14         shiroFilter.setSecurityManager(securityManager);
15         //3.设置一些其它的属性
16         shiroFilter.setLoginUrl("/login");
17         shiroFilter.setSuccessUrl("/user/listmenu");
18         shiroFilter.setUnauthorizedUrl("/refuse.jsp");
19         //4.设置自定义的过滤器
20         Map<String, Filter> filters = new LinkedHashMap<String, Filter>();
21         filters.put("authc", new CustomFormFilter());
22         shiroFilter.setFilters(filters);
23         //5.定义当前工程中使用的一系列的过滤器链(一定要使用LinkedHashMap, 因为这些过滤器必须保
证是有序的)
24         Map<String, String> filterChainDefinitionMap = new LinkedHashMap<String,
String>();
25
26         // filterChainDefinitionMap.put("/**/*.*.js", "anon");
27         // filterChainDefinitionMap.put("/**/*.*.png", "anon");
28         // filterChainDefinitionMap.put("/**/*.*.jpg", "anon");
29         // filterChainDefinitionMap.put("/**/*.*.css", "anon");
30         // filterChainDefinitionMap.put("/**/*.*.map", "anon");
31         //上面的配置, 在本工程的情况下, 可以使用下面的来代替
32         filterChainDefinitionMap.put("/bootstrap-3.3.7-dist/**/*.*", "anon");
33
34         filterChainDefinitionMap.put("/validatecode.jsp", "anon");
35         filterChainDefinitionMap.put("/refuse.jsp", "anon");
36         filterChainDefinitionMap.put("/logout", "logout");
37         filterChainDefinitionMap.put("/**", "authc");
38         shiroFilter.setFilterChainDefinitionMap(filterChainDefinitionMap);
39         return shiroFilter;
40     }
41     //配置安全管理器
42     @Bean
43     public SecurityManager securityManager(CustomRealm customRealm){
44         DefaultWebSecurityManager securityManager = new DefaultWebSecurityManager();
45         //添加自定义realm
46         securityManager.setRealm(customRealm);
47         //添加缓存管理器
48         securityManager.setCacheManager(cacheManager());
49
50         return securityManager;

```

```

51     }
52     //配置自定义realm
53     @Bean
54     public CustomRealm customRealm(HashedCredentialsMatcher matcher){
55         CustomRealm customRealm = new CustomRealm();
56         customRealm.setCredentialsMatcher(matcher);
57         return customRealm;
58     }
59     //配置凭证匹配器
60     @Bean
61     public HashedCredentialsMatcher hashedCredentialsMatcher(){
62         HashedCredentialsMatcher matcher = new HashedCredentialsMatcher();
63         matcher.setHashAlgorithmName("md5");
64         matcher.setHashIterations(1);
65         return matcher;
66     }
67     //配置缓存管理器
68     @Bean
69     public CacheManager cacheManager(){
70         EhCacheManager cacheManager = new EhCacheManager();
71         cacheManager.setCacheManagerConfigFile("classpath:cache/shiro-cache.xml");
72         return cacheManager;
73     }
74     //自定义过滤器完成验证码验证
75     @Bean
76     public CustomFormFilter customFormFilter(){
77         return new CustomFormFilter();
78     }
79     /**
80      * 配置shiro跟spring的关联
81      * 以下AuthorizationAttributeSourceAdvisor,DefaultAdvisorAutoProxyCreator两个类是为
    了支持shiro注解
82      * @param securityManager
83      * @return
84      */
85     @Bean
86     public AuthorizationAttributeSourceAdvisor
    authorizationAttributeSourceAdvisor(SecurityManager securityManager) {
87         AuthorizationAttributeSourceAdvisor advisor = new
    AuthorizationAttributeSourceAdvisor();
88         advisor.setSecurityManager(securityManager);
89         return advisor;
90     }
91
92     /**
93      * Spring的一个bean ， 由Advisor决定对哪些类的方法进行AOP代理
94      * @return
95      */
96     @Bean
97     public DefaultAdvisorAutoProxyCreator defaultAdvisorAutoProxyCreator() {
98         DefaultAdvisorAutoProxyCreator creator = new
    DefaultAdvisorAutoProxyCreator();
99         creator.setProxyTargetClass(true);

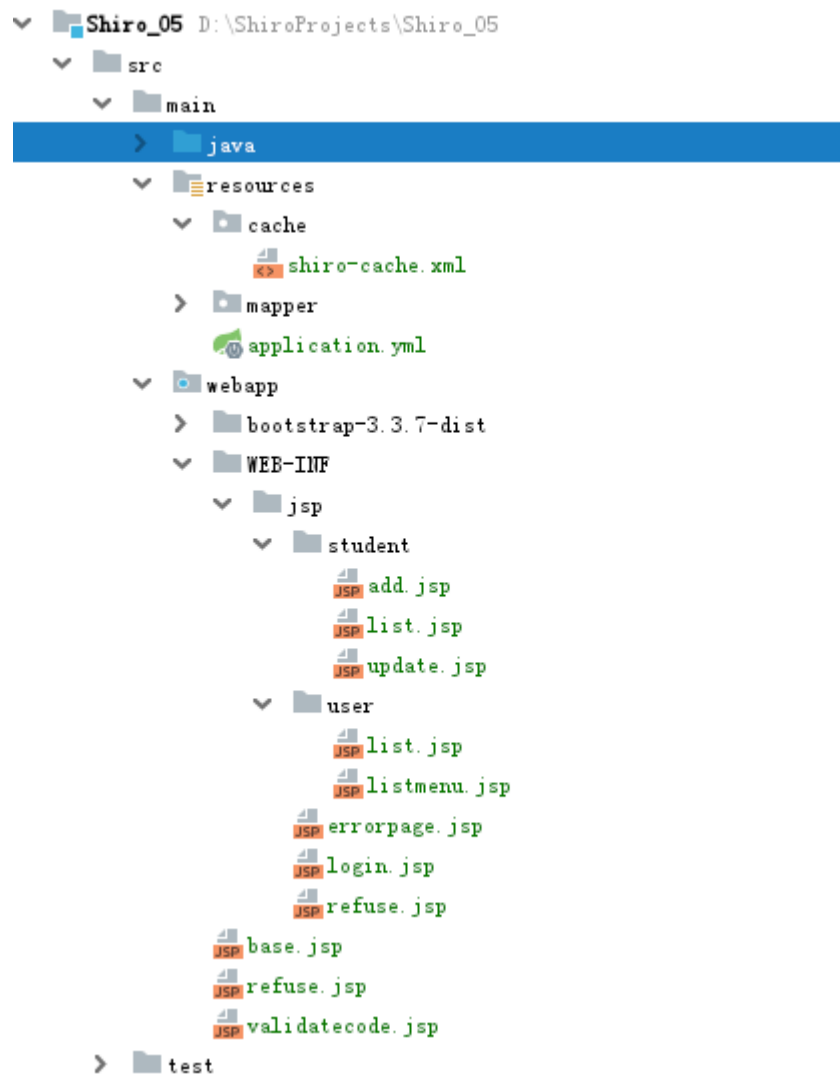
```

```

100         return creator;
101     }
102
103     /**
104      * lifecycleBeanPostProcessor是负责生命周期的，初始化和销毁的类
105      * (可选)
106      * @return
107      */
108     @Bean
109     public LifecycleBeanPostProcessor lifecycleBeanPostProcessor() {
110         return new LifecycleBeanPostProcessor();
111     }
112     @Bean
113     public SimpleMappingExceptionResolver createSimpleMappingExceptionResolver() {
114         SimpleMappingExceptionResolver exceptionResolver = new
SimpleMappingExceptionResolver();
115         //定义Properties对象，用于存放处理简单异常的属性
116         Properties mappings = new Properties();
117         mappings.setProperty("UnauthorizedException","refuse");    //参数1：要处理的异常
参数2：跳转的视图
118         mappings.setProperty("UnauthenticatedException","refuse");
119         exceptionResolver.setExceptionMappings(mappings);    // None by default
120         exceptionResolver.setDefaultErrorView("refuse");    // No default
121         exceptionResolver.setExceptionHandler("ex");    // Default is "exception"
122         return exceptionResolver;
123     }
124 }

```

3.2 ) 本工程的截图如下：



## 4、“记住我”功能的实现：

### 4.1 ) /WEB-INF/jsp/login.jsp页面

```
1 <%@ page contentType="text/html; charset=UTF-8" language="java" %>
2 <%@include file="/base.jsp"%>
3 <html>
4 <head>
5     <title>用户登录</title>
6     <style>
7         .container{
8             width: 500px;
9             margin-top: 50px;
10        }
11        .form-signin{
12            padding:5px;
13        }
14        .btn{
15            margin-top: 20px;
16        }
17        .error{
18            color:red;
```

```

19     }
20     </style>
21 </head>
22 <body>
23 <div class="container">
24     <div class="panel panel-primary">
25         <div class="panel-heading">
26             <h3 class="panel-title">
27                 用户登录
28             </h3>
29         </div>
30         <div class="panel-body">
31             <form class="form-signin" action="${pageContext.request.contextPath}/login"
method="post">
32                 <label>用户名</label>
33                 <input type="text" name="username" class="form-control" placeholder="输入
用户名" required autofocus>
34                 <label>密码</label>
35                 <input type="password" name="password" class="form-control"
placeholder="输入密码" required>
36                 <label>输入验证码</label>
37                 <input type="text" name="validcode" style="width:360px" class="form-
control" placeholder="输入验证码">
38                 
39                 <div style="padding:0;margin-top: -20px;">
40                     <input type="checkbox" name="rememberMe">&nbsp;记住我
41                 </div>
42                 <button class="btn btn-lg btn-primary btn-block" type="submit">登录
43             </button>
44             <span class="error">${message}</span>
45         </div>
46     </div>
47 </div> <!-- /container -->
48 </body>
49 </html>
50

```

#### 4.2 ) 在com.zelin.config/ShiroConfig.java配置文件中添加如下内容：

```

1  @Configuration
2  public class ShiroConfig {
3      @Bean(name = "shiroFilter")
4      public ShiroFilterFactoryBean shiroFilterFactoryBean(SecurityManager
securityManager){
5          ShiroFilterFactoryBean shiroFilter = new ShiroFilterFactoryBean();
6          //2.将securityManager与ShiroFilterFactoryBean对象绑定
7          shiroFilter.setSecurityManager(securityManager);
8          //3.设置一些其它的属性
9          shiroFilter.setLoginUrl("/login");
10         shiroFilter.setSuccessUrl("/user/listmenu");
11         shiroFilter.setUnauthorizedUrl("/refuse.jsp");

```

```

12     ...
13     filterChainDefinitionMap.put("/logout", "logout");
14     //添加“记住我”要放行的资源
15     filterChainDefinitionMap.put("/user/**", "user");
16     filterChainDefinitionMap.put("/student/**", "user");
17     //对所有的资源进行拦截(需要认证才能访问)
18     filterChainDefinitionMap.put("/**", "authc");
19 }
20 ...
21 //配置安全管理器
22 @Bean
23 public SecurityManager securityManager(CustomRealm customRealm, RememberMeManager
rememberMeManager){
24     DefaultWebSecurityManager securityManager = new DefaultWebSecurityManager();
25     //添加自定义realm
26     securityManager.setRealm(customRealm);
27     //添加缓存管理器
28     securityManager.setCacheManager(cacheManager());
29     //添加rememberMeManager
30     securityManager.setRememberMeManager(rememberMeManager);
31     return securityManager;
32 }
33 ...
34 //定义SimpleCookie用于定义rememberMeManager
35 @Bean
36 public SimpleCookie simpleCookie(){
37     SimpleCookie simpleCookie = new SimpleCookie();
38     simpleCookie.setMaxAge(2592000);
39     simpleCookie.setName("rememberMe");
40     return simpleCookie;
41 }
42 //定义rememberMeManager
43 @Bean
44 public RememberMeManager rememberMeManager(SimpleCookie simpleCookie){
45     CookieRememberMeManager rememberMeManager = new CookieRememberMeManager();
46     rememberMeManager.setCookie(simpleCookie);
47     return rememberMeManager;
48 }
49 }

```

#### 4.3 ) 测试，正常登录，再访问/user/listmenu资源，出现如下结果：

Name	Value	Domain	Path	Expires / Max...	...	...	Secure	SameSite
JSESSIONID	C774006E33861F21AF55631F06EE11E6	localhost	/	N/A	42	✓		
rememberMe	tOKNUM/hGHyVEu25SolgP+3xtt30zDu4aspbkRXfGQRcMaogHm2DfWGj...	localhost	/	2019-05-17T0...	...	✓		

用户菜单

学生管理

用户管理

分配角色

分配权限

学生查询

学生姓名： 学生住址： 所在班级：

姓名	性别	年龄	住址	生日	所在班级	操作
张三	男	20	上海	1995-07-13 星期四	1301班	<input type="button" value="修改"/> <input type="button" value="删除"/>
五二	男	28	广州	2000-07-20 星期四	1301班	<input type="button" value="修改"/> <input type="button" value="删除"/>
小红	女	19	杭州	1995-07-13 星期四	1302班	<input type="button" value="修改"/> <input type="button" value="删除"/>
王二小	男	12	岳阳	1986-06-11 星期三	1302班	<input type="button" value="修改"/> <input type="button" value="删除"/>
小2添	女	18	襄阳	1989-03-23 星期四	1302班	<input type="button" value="修改"/> <input type="button" value="删除"/>

用户菜单

学生管理

用户管理

分配角色

分配权限

无访问权限！