

第二章 Shiro权限管理

主讲：泽林.王峰 (2019.4.13)

授课大纲

- 1、SpringBoot简介
- 2、在Idea中搭建SpringBoot工程
- 3、在pom.xml文件中引入相关依赖 (mybatis、通过Mapper、分页插件)
- 4、在application.yml中完成SpringBoot相关配置
- 5、如何切换数据源
- 6、完成上章的用户认证功能

授课内容

1、SpringBoot简介

- 1 Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。通过这种方式，Spring Boot致力于在蓬勃发展的快速应用开发领域(rapid application development)成为领导者。

2、在Idea中搭建SpringBoot工程

2.1) 以maven创建普通java工程 (在此略过) 。

2.2) 配置pom.xml文件

```
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
5         http://maven.apache.org/xsd/maven-4.0.0.xsd">
6     <modelVersion>4.0.0</modelVersion>
7
8     <groupId>com.zelin</groupId>
9     <artifactId>Shiro_02</artifactId>
10    <version>1.0-SNAPSHOT</version>
11    <!--1.配置下面的依赖所用到的父工程-->
12    <parent>
13        <groupId>org.springframework.boot</groupId>
14        <artifactId>spring-boot-starter-parent</artifactId>
15        <version>2.0.1.RELEASE</version>
```

```
15     <relativePath/> <!-- lookup parent from repository -->
16 </parent>
17 <properties>
18     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
19     <java.version>1.8</java.version>
20 </properties>
21 <dependencies>
22     <dependency>
23         <groupId>org.springframework.boot</groupId>
24         <artifactId>spring-boot-starter</artifactId>
25     </dependency>
26     <!--2.与junit进行整合的starter-->
27     <dependency>
28         <groupId>org.springframework.boot</groupId>
29         <artifactId>spring-boot-starter-test</artifactId>
30         <scope>test</scope>
31     </dependency>
32     <!--3.代表springboot与web的整合的starter-->
33     <dependency>
34         <groupId>org.springframework.boot</groupId>
35         <artifactId>spring-boot-starter-web</artifactId>
36     </dependency>
37
38     <!--4.添加数据库-->
39     <dependency>
40         <groupId>mysql</groupId>
41         <artifactId>mysql-connector-java</artifactId>
42         <version>5.1.47</version>
43     </dependency>
44     <!--5.添加mybatis与springboot整合的依赖-->
45     <dependency>
46         <groupId>org.mybatis.spring.boot</groupId>
47         <artifactId>mybatis-spring-boot-starter</artifactId>
48         <version>2.0.1</version>
49     </dependency>
50     <!-- 6.添加mybatis分页插件与springboot整合的依赖 -->
51     <dependency>
52         <groupId>com.github.pagehelper</groupId>
53         <artifactId>pagehelper-spring-boot-starter</artifactId>
54         <version>1.2.10</version>
55     </dependency>
56     <!-- 7.添加通过mapper与springboot整合的依赖 -->
57     <dependency>
58         <groupId>tk.mybatis</groupId>
59         <artifactId>mapper-spring-boot-starter</artifactId>
60         <version>2.1.5</version>
61     </dependency>
62
63     <dependency>
64         <groupId>org.apache.commons</groupId>
65         <artifactId>commons-lang3</artifactId>
66         <version>3.3.2</version>
67     </dependency>
```

```

68     </dependencies>
69     <build>
70         <plugins>
71             <plugin>
72                 <groupId>org.springframework.boot</groupId>
73                 <artifactId>spring-boot-maven-plugin</artifactId>
74             </plugin>
75         </plugins>
76     </build>
77
78 </project>

```

2.3) 配置启动文件

```

1  @SpringBootApplication
2  @MapperScan(basePackages = "com.zelin.mapper")    //此注解用于扫描mapper接口
3  public class ShiroApplication {
4      public static void main(String[] args) {
5          SpringApplication.run(ShiroApplication.class);
6      }
7  }
8  //注意：此启动文件定义在com.zelin这个父包下，目的是可以读取此包及其子包下的所有注解，可以将这些java
   类的实例放到spring容器中。

```

2.4) 配置application.yml文件或application.properties文件

```

1  server:
2      port: 9000
3      #配置数据源设置
4  spring:
5      datasource:
6          driver-class-name: com.mysql.jdbc.Driver
7          url: jdbc:mysql://localhost:3306/shiro
8          username: root
9          password: 123
10     #配置mybatis的内容
11     mybatis:
12         mapper-locations: mapper/*.xml

```

2.5) 如何切换数据源

2.5.0) 在pom.xml文件中添加对druid的依赖

```

1  <!--8.添加druid连接池-->
2      <dependency>
3          <groupId>com.alibaba</groupId>
4          <artifactId>druid</artifactId>
5          <version>1.0.14</version>
6      </dependency>

```

2.5.1) 定义druid.properties文件

```

1 #配置数据源设置(改变数据源为druid)
2 spring.datasource.driver-class-name=com.mysql.jdbc.Driver
3 spring.datasource.url=jdbc:mysql://localhost:3306/shiro
4 spring.datasource.username=root
5 spring.datasource.password=123

```

2.5.2) 定义配置类: 在com.zelin.config包下DruidConfiguration.java

```

1 /**
2  * @Author: Feng.Wang
3  * @Company: Zelin.ShenZhen
4  * @Description:
5  * @Date: Create in 2019/4/13 09:55
6  */
7 @Configuration //此注解就是相当于原来的applicationContext.xml文件
8 @PropertySource("classpath:druid.properties")
9 public class DruidConfiguration {
10     @Bean //此注解：相当于spring配置文件中的<bean>标签
11     @ConfigurationProperties(prefix = "spring.datasource")
12     //工作原理：就是将druid.properties文件中的前缀后的字符串取出在DruidDataSource类中找以前面
13     //取出的字符串的属性，并为其
14     //赋值
15     public DruidDataSource druidDataSource(){
16         return new DruidDataSource();
17     }
18 }

```

效果如下所示：

```

has been autodetected for JMX exposure
'019-04-13 10:05:39.846 INFO 6844 --- [main] o.s.j.e.a.AnnotationMBeanExporter : Located MBean 'druidDataSource':
registering with JMX server as MBean [com.alibaba.druid.pool:name=druidDataSource,type=DruidDataSource]
'019-04-13 10:05:39.916 INFO 6844 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 9000

```

代表切换连接池成功

3、完成上章的用户认证功能

3.1) 使用angular+Bootstrap完成登录功能：

实现步骤：

1. 添加各种静态资源，如下图所示：



2. 定义login登录页面，内容如下：

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>用户登录</title>
6     <!--引入bootstrap样式-->
7     <link rel="stylesheet" href="bootstrap-3.3.7-dist/css/bootstrap.min.css">
8     <!--引入分页的样式-->
9     <link rel="stylesheet" href="plugins/angularjs/pagination.css">
10    <script src="plugins/jquery/jquery-2.2.3.min.js"></script>
11    <script src="bootstrap-3.3.7-dist/js/bootstrap.js"></script>
12    <!--1.引入angularjs的库-->
13    <script src="plugins/angularjs/angular.min.js"></script>
14    <!--引入angularjs的分页库-->
15    <script src="plugins/angularjs/pagination.js"></script>
16    <style>
17        .container{
18            width: 500px;
19            margin-top: 50px;
20        }
21        .form-signin{
22            padding: 5px;
23        }
24        .btn{
25            margin-top: 20px;
26        }
27        .error{
28            color: red;
29        }

```

```

30     </style>
31
32     <script>
33         //1.定义angularjs的模块：
34         var app = angular.module("myApp", []);
35         //2.定义控制器
36         app.controller("loginController", function($scope, $http){
37             //2.1)查询所有的学生
38             $scope.login = ()=>{
39                 $http.get("login?
username="+$scope.username+"&password="+$scope.password).success(response=>{
40                     if (response.status){ //代表登录成功
41                         location.href = "./user/listmenu.html"
42                     } else{
43                         $scope.message = response.message;
44                     }
45                 })
46             }
47         })
48     </script>
49 </head>
50 <body ng-app="myApp" ng-controller="loginController">
51 <div class="container">
52     <div class="panel panel-primary">
53         <div class="panel-heading">
54             <h3 class="panel-title">
55                 用户登录
56             </h3>
57         </div>
58         <div class="panel-body">
59             <label>用户名</label>
60             <input type="text" ng-model="username" class="form-control"
placeholder="输入用户名" required autofocus>
61             <label>密码</label>
62             <input type="password" ng-model="password" class="form-control"
placeholder="输入密码" required>
63             <button class="btn btn-lg btn-primary btn-block" type="button" ng-
click="login()">登录</button>
64             <span class="error">{{message}}</span>
65         </div>
66     </div>
67 </div> <!-- /container -->
68 </body>
69 </html>
70

```

3) 定义登录的控制器login

```

1  @RestController
2  public class LoginController {
3      @Autowired
4      private SysUserService userService;
5      @RequestMapping("/login")

```

```

6      public AjaxResult login(String username, String password, HttpServletRequest
request, HttpServletResponse response){
7          try {
8              //1. 根据用户名在数据库中查询是否存在此用户
9              SysUser user = userService.findUserByUserCodeAndPassword(username,
password);
10             //2. 判断用户是否存在
11             //2.1) 如果此用户存在, 就将其放到session中
12             if(user != null){
13                 request.getSession().setAttribute("user",user);
14                 return new AjaxResult("登录成功",true);
15             }
16         } catch (Exception e) {
17             e.printStackTrace();
18         }
19         return new AjaxResult("登录失败",false);
20     }
21 }

```

4. 登录界面如下：

用户登录

用户名

密码

登录

3.2) 完成拦截器定义：

```

1  @Component
2  public class AuthenticationInterceptor extends HandlerInterceptorAdapter {
3      @Override
4      public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
Object handler) throws Exception {
5          //1. 情况一：如果是匿名用户，就放行
6          //1.1) 得到当前登录的用户的url地址
7          String uri = request.getRequestURI();
8          //1.2) 对uri地址作处理
9          uri = CommUtils.getPath(uri);
10         //1.3) 得到匿名用户的地址列表
11         List<String> anonymousURL = ResourcesUtil.gekeyList("anonymousURL");
12         //1.4) 判断当前登录的url地址是否在上面的地址集合中,如果在就放行

```

```

13         if (anonymousURL.contains(uri)) return true;
14
15         //情况二：看用户是否登录，如果登录就看其访问地址是否在公用的url地址列表中，
16         //2.1) 得到session，并且取得当前登录的用户对象user
17         SysUser user = (SysUser) request.getSession().getAttribute("user");
18         //2.2) 判断是否存在此用户
19         if (null != user) { //代表用户登录过
20             //2.3) 读取公用的用户列表(只有登录成功的用户才能访问此列表)
21             List<String> commonURL = ResourcesUtil.gekeyList("commonURL");
22             //2.4) 判断当前登录用户的访问url地址是否在commonURL中，是就放行
23             if (commonURL.contains(uri)) return true;
24
25             //情况三：根据当前用户，得到其访问资源的权限列表
26             //3.1) 得到当前登录成功的用户的所有权限列表
27             List<SysPermission> permissions = user.getPermissions();
28             //3.2) 根据此权限列表遍历出每个权限，再看当前用户的访问url地址是否在其中
29             for (SysPermission permission : permissions) {
30                 if (permission.getUrl().contains(uri)) {
31                     return true;
32                 }
33             }
34         }
35         //其它情况：如果上面的情况都不成立，就跳转到无权访问页面。
36         request.getRequestDispatcher("/resure.html").forward(request, response);
37         return false;
38     }
39 }
40

```

3.3) 配置拦截器：

```

1  @Configuration
2  public class MyConfiguration implements WebMvcConfigurer {
3      @Autowired
4      private AuthenticationInterceptor authenticationInterceptor;
5      //注册拦截器 (springboot拦截器与springmvc不一样，还会拦截静态资源 )
6      @Override
7      public void addInterceptors(InterceptorRegistry registry) {
8          registry.addInterceptor(authenticationInterceptor).addPathPatterns("/**")
9              .excludePathPatterns("/static/**");
10
11      }
12  }

```

3.4) 定义学生列表功能

```

1  public class StudentController {
2      @Autowired
3      private StudentService studentService;
4      @Autowired
5      private ClassesService classesService;
6      private int pageSize = 5;

```



```

7   @RequestMapping("/list")
8   public List<Student> findAll(){
9       try {
10          return studentService.findStudents();
11      } catch (Exception e) {
12          e.printStackTrace();
13          return null;
14      }
15  }
16 }

```

```

1  /student/list.html---》页面
2  <!DOCTYPE html>
3  <html lang="en">
4  <head>
5      <meta charset="UTF-8">
6      <title>列表学生</title>
7      <style>
8          .table{
9              text-align: center;
10         }
11         .container{
12             margin-top: 20px;
13         }
14     </style>
15     <!--引入bootstrap样式-->
16     <link rel="stylesheet" href="../bootstrap-3.3.7-dist/css/bootstrap.min.css">
17     <!--引入分页的样式-->
18     <link rel="stylesheet" href="../plugins/angularjs/pagination.css">
19     <script src="../plugins/jquery/jquery-2.2.3.min.js"></script>
20     <script src="../bootstrap-3.3.7-dist/js/bootstrap.js"></script>
21     <!--1.引入angularjs的库-->
22     <script src="../plugins/angularjs/angular.min.js"></script>
23     <!--引入angularjs的分页库-->
24     <script src="../plugins/angularjs/pagination.js"></script>
25     <script>
26         //1.定义angularjs的模块：
27         var app = angular.module("myApp", []);
28         //2.定义控制器
29         app.controller("studentController",function($scope,$http){
30             //2.1)查询所有的学生
31             $scope.findAll=()=>{
32                 $http.get("../student/list").success(response=>{
33                     //1.为list变量分配值
34                     $scope.list = response;
35                 } )
36             }
37         })
38     </script>
39 </head>
40 <body ng-app="myApp" ng-controller="studentController" ng-init="findAll()">
41 <div class="container">
42     <div class="panel panel-primary">

```

```

43     <div class="panel-heading">
44         <h3 class="panel-title">学生列表</h3>
45     </div>
46     <table class="table table-bordered table-striped">
47         <tr>
48             <td>姓名</td>
49             <td>性别</td>
50             <td>年龄</td>
51             <td>住址</td>
52             <td>生日</td>
53             <td>所在班级</td>
54             <td>操作</td>
55         </tr>
56         <tr ng-repeat="stud in list">
57             <td>{{stud.sname}}</td>
58             <td>{{stud.sex}}</td>
59             <td>{{stud.age}}</td>
60             <td>{{stud.addr}}</td>
61             <td>
62                 {{stud.birth}}
63             </td>
64             <td>
65                 {{stud.classes.cname}}
66             </td>
67             <td>
68                 <a class="btn btn-primary btn-sm"
69                     href="../student/toupdate?sid={{stud.sid}}">修改</a>
70                 <a class="btn btn-danger btn-sm"
71                     href="../student/deleteBySid?sid={{stud.sid}}"
72                     onclick="return confirm('你真的要删除吗?')">删除</a>
73             </td>
74         </tr>
75     </table>
76 </div>
77 </div>
78 <script>
79     //执行提交 表单
80     function skip(i) {
81         //1.对表单中的隐藏域赋值
82         $("#page").val(i);
83         //2.提交表单
84         $("#form1").submit();
85     }
86
87     //添加学生
88     function addStudent() {
89         location.href = "${pageContext.request.contextPath}/student/toadd.do";
90     }
91 </script>
92 </body>
93 </html>

```

3.5) 效果图如下：

用户菜单	学生列表						
学生管理	姓名	性别	年龄	住址	生日	所在班级	操作
用户管理	张三	男	20	上海	1995-07-12T16:00:00.000+0000	1301班	<button>修改</button> <button>删除</button>
分配角色	五二	男	28	广州	2000-07-19T16:00:00.000+0000	1301班	<button>修改</button> <button>删除</button>
分配权限	小红	女	19	杭州	1995-07-12T16:00:00.000+0000	1302班	<button>修改</button> <button>删除</button>
	王二小	男	12	岳阳	1986-06-10T15:00:00.000+0000	1302班	<button>修改</button> <button>删除</button>
	小2添	女	18	襄阳	1989-03-22T16:00:00.000+0000	1302班	<button>修改</button> <button>删除</button>
	小黄	男	21	南阳	1987-05-16T15:00:00.000+0000	1301班	<button>修改</button> <button>删除</button>
	罗成	男	22	邵阳	1998-04-11T16:00:00.000+0000	1301班	<button>修改</button> <button>删除</button>
	魏征	男	28	洛阳	1967-07-17T16:00:00.000+0000	1301班	<button>修改</button> <button>删除</button>
	徐达	男	29	江苏无锡	1985-05-10T16:00:00.000+0000	1302班	<button>修改</button> <button>删除</button>

4、完成为当前是用户修改角色的功能：

4.1) 在RoleController控制器得到所有的角色列表：(/role/list)

```
1  @RestController
2  @RequestMapping("/role")
3  public class RoleController {
4      @Autowired
5      private RoleService roleService;
6      @Autowired
7      private UserRoleService userRoleService;
8      /**
9       * 查询所有的角色
10      * @return
11      */
12      @RequestMapping("/list")
13      public List<SysRole> findAll(){
14          try {
15              return roleService.findRoles();
16          } catch (Exception e) {
17              e.printStackTrace();
18          }
19          return null;
20      }
21  }
```

4.2) 根据当前选择用户查询出其关联的角色(/role/findRolesByUser)

```

1  /**
2   * 根据当前选择用户查询出其关联的角色
3   * @return
4   */
5  @RequestMapping("/findRolesByUser")
6  public List<SysUserRole> findRolesByUser(String sysUserId){
7      List<SysUserRole> userRoles = userRoleService.findUserRole(sysUserId);
8      return userRoles;
9  }

```

4.3) 修改当前选择用户的角色(/role/changeRole)

```

1  /**
2   * 修改角色
3   * @param sysUserRolevo
4   * @return
5   */
6  @RequestMapping("/changeRole")
7  public AjaxResult changeRole(SysUserRolevo sysUserRolevo){
8      try {
9          userRoleService.update(sysUserRolevo);
10         return new AjaxResult("修改权限成功",true);
11     } catch (Exception e) {
12         e.printStackTrace();
13         return new AjaxResult("修改权限失败",false);
14     }
15 }

```

4.4) 查询所有用户列表(/user/userlist)

```

1  /**
2   * 查询所有用户
3   * @return
4   */
5  @RequestMapping("/userlist")
6  public List<SysUser> findUsers(){
7      try {
8          return userService.findUsers();
9      } catch (Exception e) {
10         e.printStackTrace();
11     }
12     return null;
13 }

```

4.5) user/changeRole.html页面的内容

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>修改用户权限</title>

```

```

6      <!--引入bootstrap样式-->
7      <link rel="stylesheet" href="../bootstrap-3.3.7-dist/css/bootstrap.min.css">
8      <!--引入分页的样式-->
9      <link rel="stylesheet" href="../plugins/angularjs/pagination.css">
10     <script src="../plugins/jquery/jquery-2.2.3.min.js"></script>
11     <script src="../bootstrap-3.3.7-dist/js/bootstrap.js"></script>
12     <!--1.引入angularjs的库-->
13     <script src="../plugins/angularjs/angular.min.js"></script>
14     <!--引入angularjs的分页库-->
15     <script src="../plugins/angularjs/pagination.js"></script>
16     <style>
17         .list-group{
18             width: 300px;
19             margin-top: 20px;
20             margin-left: 40px;
21         }
22     </style>
23     <script>
24         //1.定义angularjs的模块：
25         var app = angular.module("myApp", []);
26         //2.定义控制器
27         app.controller("roleController",function($scope,$http){
28             //2.1)查询所有的权限
29             $scope.findAll=()=>{
30                 $http.get("../role/list").success(response=>{
31                     $scope.list = response;
32                 })
33             }
34             //2.2)查询所有用户
35             $scope.findUsers=()=>{
36                 $http.get("../user/userlist").success(response=>{
37                     $scope.users=response;
38                 })
39             }
40             //2.2)根据当前用户取出其角色列表(当选择某个用户时就可以选择其对应的角色)
41             $scope.findRolesByUser = ()=>{
42                 $http.get("../role/findRolesByUser?
sysUserId="+$scope.sysUserId).success(response =>{
43                     $scope.roleList = response;
44                     //定义所选择的角色id数组
45                     $scope.sysRoleIds = [];
46                     //遍历$scope.roleList与$scope.list两个集合，选中复选框
47                     if ($scope.list.length > 0 && $scope.roleList.length > 0)
48                     for (var i = 0;i < $scope.list.length;i++){
49                         $scope.sysRoleIds[i]=0
50                         for (var j = 0; j < response.length;j++){
51                             if ($scope.list[i].id ==response[j].sysRoleId){
52                                 //如果两个id相等先放到角色id数组中
53                                 $scope.sysRoleIds[i]=$scope.list[i].id;
54                             }
55                         }
56                     }
57                 } )

```

```

58         }
59         //3.修改角色
60         $scope.changeRole = ()=>{
61
62             //将角色id数组提交到后台
63             $http.get("../role/changeRole?
sysRoleIds="+$scope.sysRoleIds+"&sysUserId="+$scope.sysUserId).success(response=>{
64                 if (response.status){ //修改成功就跳转到登录页面
65                     parent.location.href = "/login.html"
66                 }else{ //修改失败
67                     alert(response.message);
68                 }
69             })
70         }
71     })
72 </script>
73
74 </head>
75 <body ng-app="myApp" ng-controller="roleController" ng-init="findAll();findUsers()">
76 {{sysRoleIds}}--{{sysUserId}}
77     <ul class="list-group">
78         <li class="list-group-item" >
79             请选择用户 :
80             <select ng-model="sysUserId" class="form-control" ng-
change="findRolesByUser()"
81                 ng-options="u.id as u.username for u in users"></select>
82         </li>
83         <li class="list-group-item" ng-repeat="role in list ">
84
85             <!--<input type="checkbox"    ng-checked="role.result" ng-
model="sysRoleIds[$index]"-->
86             <!--ng-true-value="{{role.id}}" ng-false-value="">{{role.name}}-
->
87             <input type="checkbox"    ng-model="sysRoleIds[$index]"
88                 ng-true-value="{{role.id}}" ng-false-value="0">{{role.name}}
89         <br>
90         {{role.id}}
91         </li>
92         <li class="list-group-item" >
93             <input type="button" value="更改权限" class="btn btn-primary btn-sm"
ng-click="changeRole()">
94         </li>
95     </ul>
96 </body>
</html>

```

4.5) 运行结果如下 :

用户菜单
学生管理
分配角色
分配权限

请选择用户：

李四

☒商品管理员

ebc8a441-c6f9-11e4-b137-0adc305c3f28

☒超级管理员

ebc8a441-c6f9-11e4-b137-0adc305c3f29

☐用户管理员

ebc9d647-c6f9-11e4-b137-0adc305c3f28

更改权限