# Shiro笔记大全

### 1.0 分析shiro的核心API

Subject:用户主体(把操作交给SecurityManager)

SecurityManager: 安全管理器(关联Realm)

Realm:shiro连接数据的桥梁

### Springboot整合依赖 导入shiro与spring整合依赖

修改pom.xml

|  |
| --- |
| <!-- shiro与spring依赖 -->  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-spring</artifactId>  <version>1.4.0</version>  </dependency> |

### 自定义Realm类

|  |
| --- |
| /\*\*  \* 自定义Realm  \* **@author** Administrator  \*  \*/  **public** **class** UserRealm **extends** AuthorizingRealm{  /\*\*  \* 执行授权逻辑  \*/  @Override  **protected** AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals) {  System.***out***.println("执行授权逻辑");  **return** **null**;  }    /\*\*  \* 执行认证逻辑  \*/  @Override  **protected** AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) **throws** AuthenticationException {  System.***out***.println("执行认证逻辑");  **return** **null**;  }    } |

### 编写shiro配置类(\*)

|  |
| --- |
| /\*\*  \* Shiro的配置类  \* **@author** 林  \*  \*/  @Configuration  **public** **class** ShiroConfig {  //Subject:用户主体(把操作交给SecurityManager)  //SecurityManager: 安全管理器(关联Realm)  //Realm:shiro连接数据的桥梁  /\*\*  \* 创建ShiroFilterFactoryBean  \*/  @Bean  **public** ShiroFilterFactoryBean getShiroFilterFactoryBean(@Qualifier("securityManager")DefaultWebSecurityManager securityManager) {    ShiroFilterFactoryBean shiroFilterFactoryBean = **new** ShiroFilterFactoryBean();  //设置安全管理器  shiroFilterFactoryBean.setSecurityManager(securityManager);  //添加Shiro内置过滤器  /\*\*  \* Shiro内置过滤器，可以实现权限相关的拦截器  \* 常用的过滤器：  \* anon:无需认证(登录) 可以访问  \* authc：必须认证才可以访问  \* user：如果使用rememberMe的功能可以直接访问  \* perms:该资源必须得到资源权限才可以访问  \* role:该资源必须得到角色权限才能访问  \*/  Map<String,String> filterMap = **new** LinkedHashMap<String,String>();  filterMap.put("/user/add", "authc");  filterMap.put("/user/update","authc");  // filterMap.put("/user/\*","authc");  //修改调整的登陆页面  shiroFilterFactoryBean.setLoginUrl("/login");    shiroFilterFactoryBean.setFilterChainDefinitionMap(filterMap);    **return** shiroFilterFactoryBean;  }    /\*\*  \* 创建DefaultWebSecurityManager  \*/  @Bean(name="securityManager")  **public** DefaultWebSecurityManager getDefaultWebSecurityManager(@Qualifier("userRealm")UserRealm userRealm) {  DefaultWebSecurityManager securityManager = **new** DefaultWebSecurityManager();  //关联realm  securityManager.setRealm(userRealm);  **return** securityManager;  }  /\*\*  \* 创建Realm  \*/  @Bean(name="userRealm")  **public** UserRealm getRealm() {  **return** **new** UserRealm();  } } |

### 1.4使用shiro内置过滤器实现页面拦截

Shiro内置过滤器，可以实现权限相关的拦截器

常用的过滤器：

anon:无需认证(登录) 可以访问

authc：必须认证才可以访问

user：如果使用rememberMe的功能可以直接访问

perms:该资源必须得到资源权限才可以访问

role:该资源必须得到角色权限才能访问

|  |
| --- |
| /\*\*  \* 创建ShiroFilterFactoryBean  \*/  @Bean  **public** ShiroFilterFactoryBean getShiroFilterFactoryBean(@Qualifier("securityManager")DefaultWebSecurityManager securityManager) {    ShiroFilterFactoryBean shiroFilterFactoryBean = **new** ShiroFilterFactoryBean();  //设置安全管理器  shiroFilterFactoryBean.setSecurityManager(securityManager);  //添加Shiro内置过滤器  /\*\*  \* Shiro内置过滤器，可以实现权限相关的拦截器  \* 常用的过滤器：  \* anon:无需认证(登录) 可以访问  \* authc：必须认证才可以访问  \* user：如果使用rememberMe的功能可以直接访问  \* perms:该资源必须得到资源权限才可以访问  \* role:该资源必须得到角色权限才能访问  \*/  Map<String,String> filterMap = **new** LinkedHashMap<String,String>();  filterMap.put("/user/add", "authc"); //要拦截的页面  filterMap.put("/user/update","authc");    // filterMap.put("/user/\*","authc");  //修改调整的登陆页面  shiroFilterFactoryBean.setLoginUrl("/login");    shiroFilterFactoryBean.setFilterChainDefinitionMap(filterMap);    **return** shiroFilterFactoryBean;  } |

### 1.5 实现用户认证(登陆) 操作

#### 1.5.1设计登陆页面

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset=*"UTF-8"*>  <title>登陆页面</title>  </head>  <body>  <h3>登陆页面</h3>  <from method=*"post"* action=*"toLogin"*>  用户名：<input type=*"text"* name=*"name"*/>    密码：<input type=*"password"* name=*"password"*/>    <input type=*"submit"* value=*"登陆"*/>    </from>  </body>  </html> |

#### 1.5.2 编写Controller的登陆逻辑

|  |
| --- |
| /\*\*  \*登陆逻辑处理  \*/  @RequestMapping("/tologin")  **public** String tologin(String name,String password,Model model) {  System.***out***.println("name:"+name);  System.***out***.println("============================");  /\*\*  \* 使用Shiro编写认证操作  \*/  //1 获取Subject  Subject subject = SecurityUtils.*getSubject*();    //2 封装用户数据  UsernamePasswordToken token = **new** UsernamePasswordToken(name,password);  **try** {  //执行登陆方式  subject.login(token);  //登陆成功  //跳转到test.html  **return** "redirect:/testThymeleaf";  } **catch** (UnknownAccountException e) {  //登陆失败:用户名不存在  model.addAttribute("msg","用户名不存在");    **return** "/login";  } **catch** (IncorrectCredentialsException e) {  //登陆失败:用户名不存在  model.addAttribute("msg","密码错误");  **return** "/login";  }  } |

#### 1.5.3 编写Realm的判断逻辑

|  |
| --- |
|  |