# Shiro[pom.xml]

|  |
| --- |
| <project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>  <modelVersion>4.0.0</modelVersion>  <groupId>com.4sir</groupId>  <artifactId>Miya</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>  <name>Miya</name>  <url>http://maven.apache.org</url>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.6.RELEASE</version>  <relativePath /> <!-- lookup parent from repository -->  </parent>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-test</artifactId>  <scope>test</scope>  </dependency>  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.6</version>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-devtools</artifactId>  </dependency>  <dependency>  <groupId>org.mybatis.spring.boot</groupId>  <artifactId>mybatis-spring-boot-starter</artifactId>  <version>1.1.1</version>  </dependency>  <dependency>  <groupId>com.alibaba</groupId>  <artifactId>druid</artifactId>  <version>1.0.28</version>  </dependency>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>3.8.1</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.apache.shiro</groupId>  <artifactId>shiro-spring</artifactId>  <version>1.4.0</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>3.0.1</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>javax.servlet.jsp</groupId>  <artifactId>jsp-api</artifactId>  <version>2.1</version>  <scope>provided</scope>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  <plugin>  <groupId>org.jacoco</groupId>  <artifactId>jacoco-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

# Shiro[ShiroConfig]

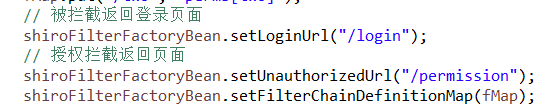
|  |
| --- |
| @Configuration  **public** **class** ShiroConfig {  // 创建ShiroFilterFactoryBean  @Bean(name = "shiroFilterFactoryBean")  **public** ShiroFilterFactoryBean getShiroFilterFactoryBean(  @Qualifier("defaultWebSecurityManager") DefaultWebSecurityManager defaultWebSecurityManager) {  ShiroFilterFactoryBean shiroFilterFactoryBean = **new** ShiroFilterFactoryBean();  // 设置安全管理器  shiroFilterFactoryBean.setSecurityManager(defaultWebSecurityManager);  // 添加shiro内置过滤器  /\*  \* anon:表示可以匿名使用。 authc:表示需要认证(登录)才能使用，没有参数  \* roles：参数可以写多个，多个时必须加上引号，并且参数之间用逗号分割，当有多个参数时，例如admins/user/\*\*=roles[  \* "admin,guest"],每个参数通过才算通过，相当于hasAllRoles()方法。  \* perms：参数可以写多个，多个时必须加上引号，并且参数之间用逗号分割，例如/admins/user/\*\*=perms[  \* "user:add:\*,user:modify:\*"]，当有多个参数时必须每个参数都通过才通过，想当于isPermitedAll()方法。  \* rest：根据请求的方法，相当于/admins/user/\*\*=perms[user:method]  \* ,其中method为post，get，delete等。  \* port：当请求的url的端口不是8081是跳转到schemal://serverName:8081?queryString,  \* 其中schmal是协议http或https等，serverName是你访问的host,8081是url配置里port的端口，  \* queryString是你访问的url里的？后面的参数。 authcBasic：没有参数表示httpBasic认证  \* ssl:表示安全的url请求，协议为https user:当登入操作时不做检查  \*/  Map<String, String> fMap = **new** HashMap<String, String>();  // 拦截页面  fMap.put("/one", "authc");  fMap.put("/two", "authc");  // 拦截未授权  fMap.put("/one", "perms[one]");  fMap.put("/two", "perms[two]");  // 被拦截返回登录页面  shiroFilterFactoryBean.setLoginUrl("/login");  // 授权拦截返回页面  shiroFilterFactoryBean.setUnauthorizedUrl("/permission");  shiroFilterFactoryBean.setFilterChainDefinitionMap(fMap);  **return** shiroFilterFactoryBean;  }  @Bean(name = "defaultWebSecurityManager")  // 创建DefaultWebSecurityManager  **public** DefaultWebSecurityManager getDefaultWebSecurityManager(@Qualifier("userRealm") UserRealm userRealm) {    System.***err***.println("defaultWebSecurityManager");  DefaultWebSecurityManager defaultWebSecurityManager = **new** DefaultWebSecurityManager();  defaultWebSecurityManager.setRealm(userRealm);  **return** defaultWebSecurityManager;  }  // 创建Realm  @Bean(name = "userRealm")  **public** UserRealm getUserRealm() {  System.***err***.println("userRealm");  **return** **new** UserRealm();  } |

# Shiro[UserRealm]

|  |
| --- |
| package com.sir.Miya.config;  import java.util.Iterator;  import java.util.Set;  import org.apache.shiro.SecurityUtils;  import org.apache.shiro.authc.AuthenticationException;  import org.apache.shiro.authc.AuthenticationInfo;  import org.apache.shiro.authc.AuthenticationToken;  import org.apache.shiro.authc.SimpleAuthenticationInfo;  import org.apache.shiro.authc.UsernamePasswordToken;  import org.apache.shiro.authz.AuthorizationInfo;  import org.apache.shiro.authz.SimpleAuthorizationInfo;  import org.apache.shiro.realm.AuthorizingRealm;  import org.apache.shiro.subject.PrincipalCollection;  import org.apache.shiro.subject.Subject;  import org.springframework.beans.factory.annotation.Autowired;  /\*\*  \* 自定义realm  \* @author jianping.lu  \*  \*/  public class UserRealm extends AuthorizingRealm{    /\*\*  \* //执行授权  \*/  @Override  protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection arg0) {  // TODO Auto-generated method stub  System.out.println("授权");  //获取当前登录用户  Subject subject = SecurityUtils.getSubject();  Object principal = subject.getPrincipal();  System.out.println(principal);  //给资源授权  SimpleAuthorizationInfo simpleAuthorizationInfo = new SimpleAuthorizationInfo();    simpleAuthorizationInfo.addStringPermission("one");    Set<String> stringPermissions = simpleAuthorizationInfo.getStringPermissions();  Iterator<String> iterator = stringPermissions.iterator();  while(iterator.hasNext()){  String next = iterator.next();  System.err.println(next);  }  return simpleAuthorizationInfo;  }  /\*\*  \* 执行认证逻辑  \*/  @Override  protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken arg0) throws AuthenticationException {  // TODO Auto-generated method stub  System.out.println("认证");    //shiro判断逻辑  UsernamePasswordToken user = (UsernamePasswordToken) arg0;  System.err.println(user);    return new SimpleAuthenticationInfo("jean","123","");  }      } |

# Shiro[流程]

1. 当容器启动的时候,会将shiroFilterFactoryBean,defaultWebSecurityManager,userRealm注入进来
2. 当用户访问需要权限的页面的时候,



这个配置就会进行拦截,如果没有授权就进入登录权限,如果没有权限则进入没有权限的提示页面。

1. 当用户登录的时候,

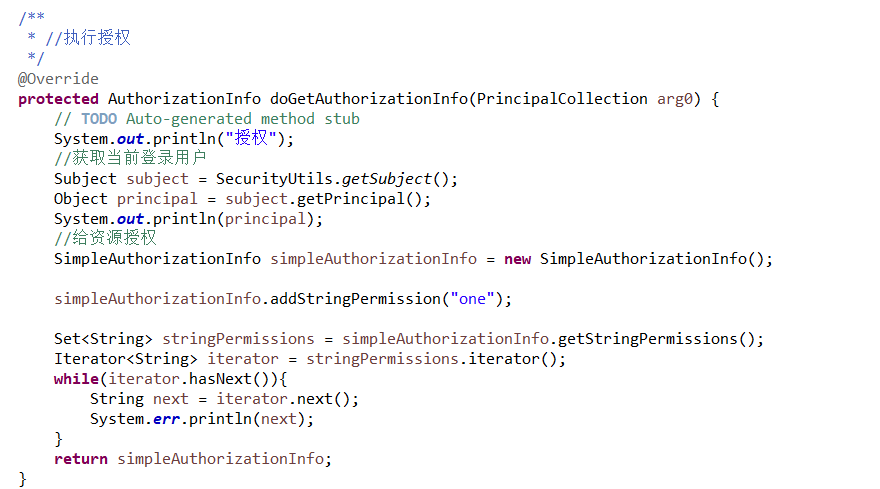


首先获取到username,passord实例化token,在将token进行登录.

此时,线程会走向认证逻辑



然后在进入授权逻辑代码



将当前这个用户的权限和角色都进行添加.然后返回权限。