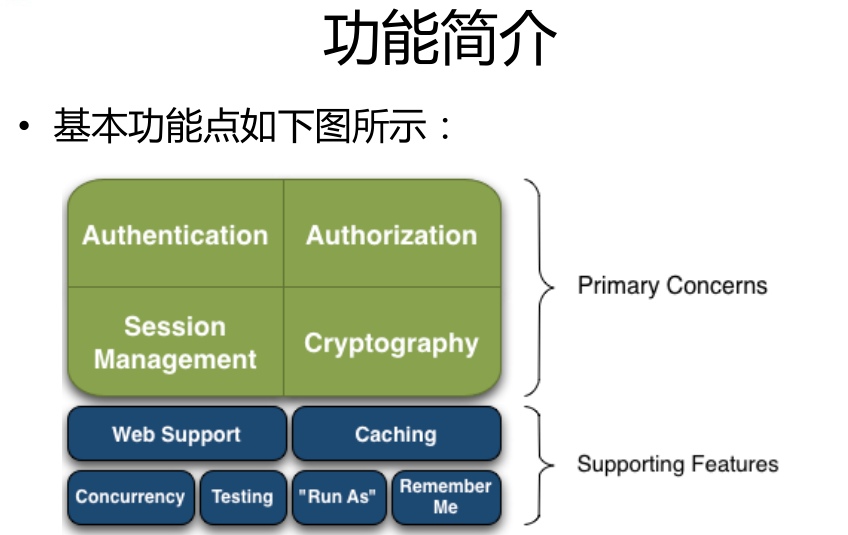
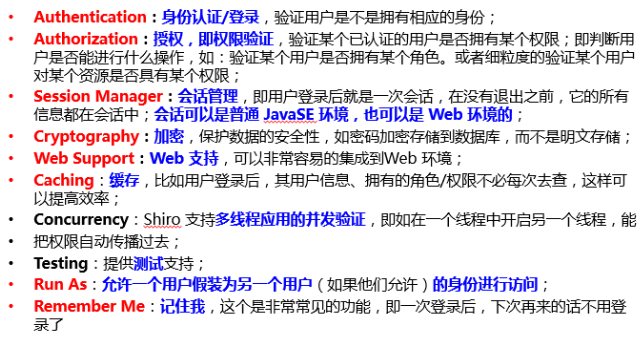
Shiro教程

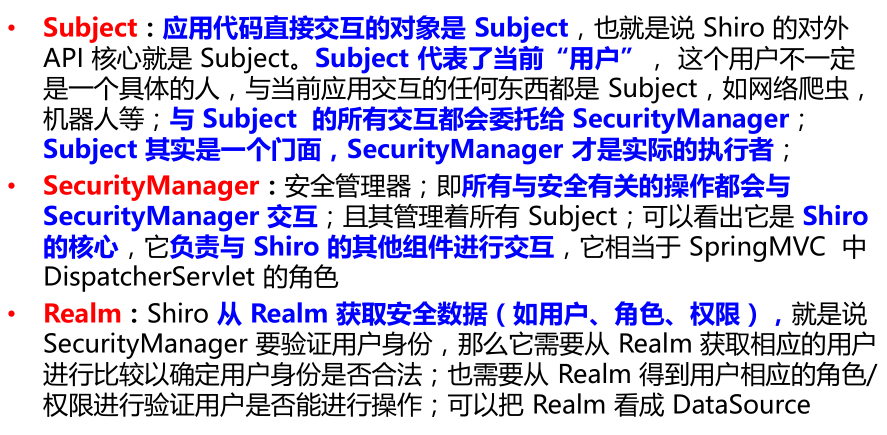
1. **概述**

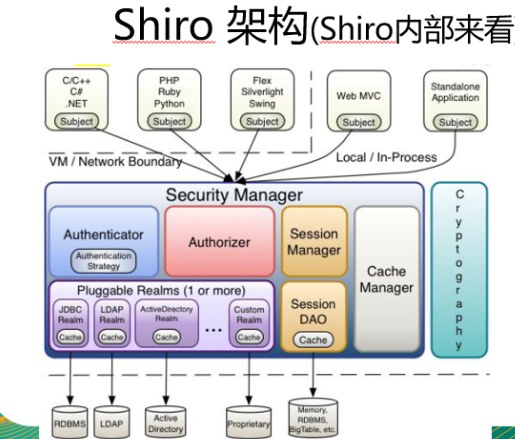
Apache Shiro是一个强大且易用的Java安全框架,执行身份验证、授权、密码学和会话管理。使用Shiro的易于理解的API,您可以快速、轻松地获得任何应用程序,从最小的移动应用程序到最大的网络和企业应用程序。

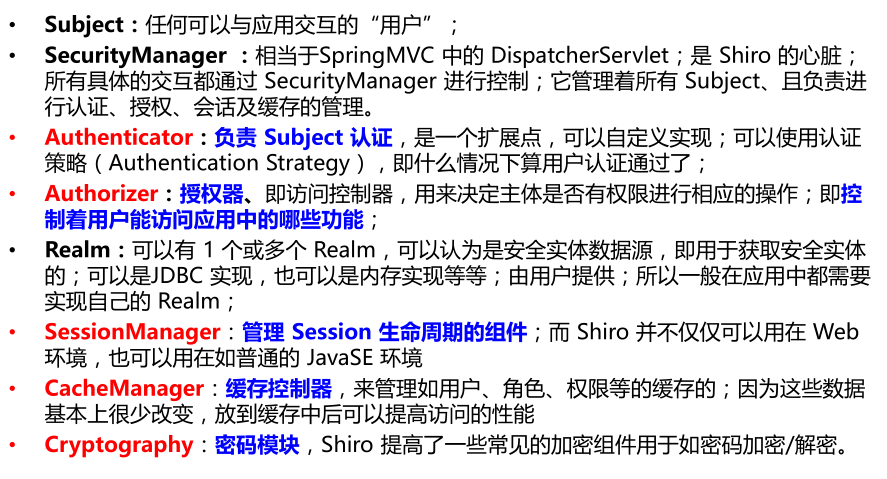








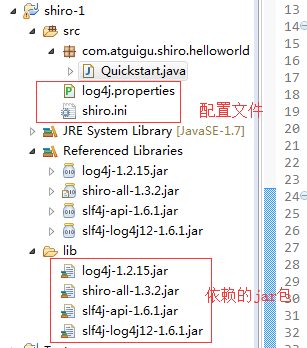




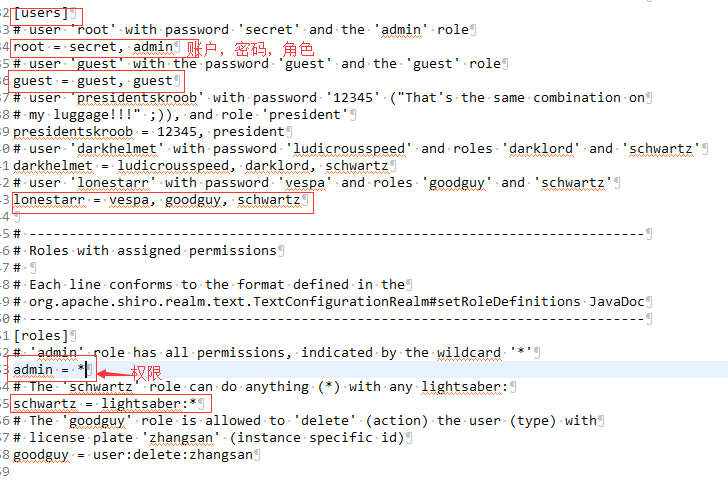
注:shiro的session和我们web项目的http里面的session是不一样的，这里的session是由shiro自己提供的，即便项目不是web项目，也可以使用shiro的session

1. **Shiro的一个demo**

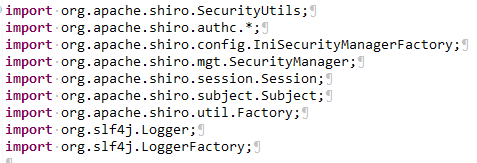
结构：



shiro.ini文件：



Quickstart文件：



**public** **class** Quickstart {

**private** **static** **final** **transient** Logger ***log*** = LoggerFactory.*getLogger*(Quickstart.**class**);

**public** **static** **void** main(String[] args) {

// The easiest way to create a Shiro SecurityManager with configured

// realms, users, roles and permissions is to use the simple INI config.

// We'll do that by using a factory that can ingest a .ini file and

// return a SecurityManager instance:

// Use the shiro.ini file at the root of the classpath

// (file: and url: prefixes load from files and urls respectively):

Factory<SecurityManager> factory = **new** IniSecurityManagerFactory("classpath:shiro.ini");

SecurityManager securityManager = factory.getInstance();

// for this simple example quickstart, make the SecurityManager

// accessible as a JVM singleton. Most applications wouldn't do this

// and instead rely on their container configuration or web.xml for

// webapps. That is outside the scope of this simple quickstart, so

// we'll just do the bare minimum so you can continue to get a feel

// for things.

SecurityUtils.*setSecurityManager*(securityManager);

// Now that a simple Shiro environment is set up, let's see what you can

// do:

// get the currently executing user:

// 获取当前的 Subject. 调用 SecurityUtils.getSubject();

Subject currentUser = SecurityUtils.*getSubject*();

// Do some stuff with a Session (no need for a web or EJB container!!!)

// 测试使用 Session

// 获取 Session: Subject#getSession()

Session session = currentUser.getSession();

session.setAttribute("someKey", "aValue");

String value = (String) session.getAttribute("someKey");

**if** (value.equals("aValue")) {

***log***.info("---> Retrieved the correct value! [" + value + "]");

}

// let's login the current user so we can check against roles and

// permissions:

// 测试当前的用户是否已经被认证. 即是否已经登录.

// 调动 Subject 的 isAuthenticated()

**if** (!currentUser.isAuthenticated()) {

// 把用户名和密码封装为 UsernamePasswordToken 对象

UsernamePasswordToken token = **new** UsernamePasswordToken("lonestarr", "vespa");

// rememberme

token.setRememberMe(**true**);

**try** {

// 执行登录.

currentUser.login(token);

}

// 若没有指定的账户, 则 shiro 将会抛出 UnknownAccountException 异常.

**catch** (UnknownAccountException uae) {

***log***.info("----> There is no user with username of " + token.getPrincipal());

**return**;

}

// 若账户存在, 但密码不匹配, 则 shiro 会抛出 IncorrectCredentialsException 异常。

**catch** (IncorrectCredentialsException ice) {

***log***.info("----> Password for account " + token.getPrincipal() + " was incorrect!");

**return**;

}

// 用户被锁定的异常 LockedAccountException

**catch** (LockedAccountException lae) {

***log***.info("The account for username " + token.getPrincipal() + " is locked. "

+ "Please contact your administrator to unlock it.");

}

// catch more exceptions here (maybe custom ones specific to

// your application?

// 所有认证时异常的父类.

**catch** (AuthenticationException ae) {

// unexpected condition? error?

}

}

// say who they are:

// print their identifying principal (in this case, a username):

***log***.info("----> User [" + currentUser.getPrincipal() + "] logged in successfully.");

// test a role:

// 测试是否有某一个角色. 调用 Subject 的 hasRole 方法.

**if** (currentUser.hasRole("schwartz")) {

***log***.info("----> May the Schwartz be with you!");

} **else** {

***log***.info("----> Hello, mere mortal.");

**return**;

}

// test a typed permission (not instance-level)

// 测试用户是否具备某一个行为. 调用 Subject 的 isPermitted() 方法。

**if** (currentUser.isPermitted("lightsaber:weild")) {

***log***.info("----> You may use a lightsaber ring. Use it wisely.");

} **else** {

***log***.info("Sorry, lightsaber rings are for schwartz masters only.");

}

// a (very powerful) Instance Level permission:

// 测试用户是否具备某一个行为.

**if** (currentUser.isPermitted("user:delete:zhangsan")) {

***log***.info("----> You are permitted to 'drive' the winnebago with license plate (id) 'eagle5'. "

+ "Here are the keys - have fun!");

} **else** {

***log***.info("Sorry, you aren't allowed to drive the 'eagle5' winnebago!");

}

// all done - log out!

// 执行登出. 调用 Subject 的 Logout() 方法.

System.***out***.println("---->" + currentUser.isAuthenticated());

currentUser.logout();

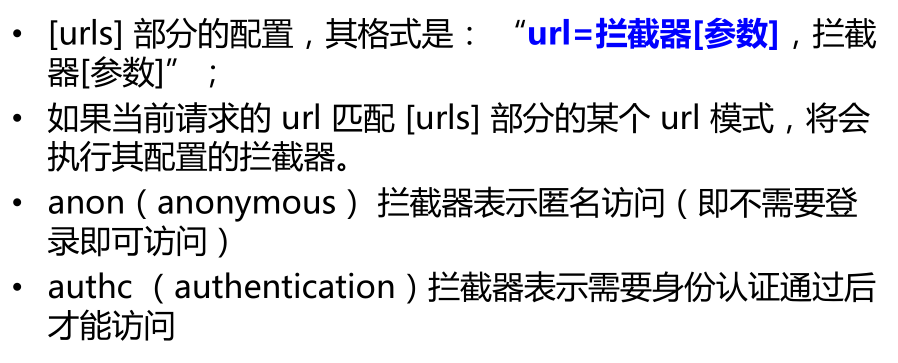
System.***out***.println("---->" + currentUser.isAuthenticated());

System.*exit*(0);

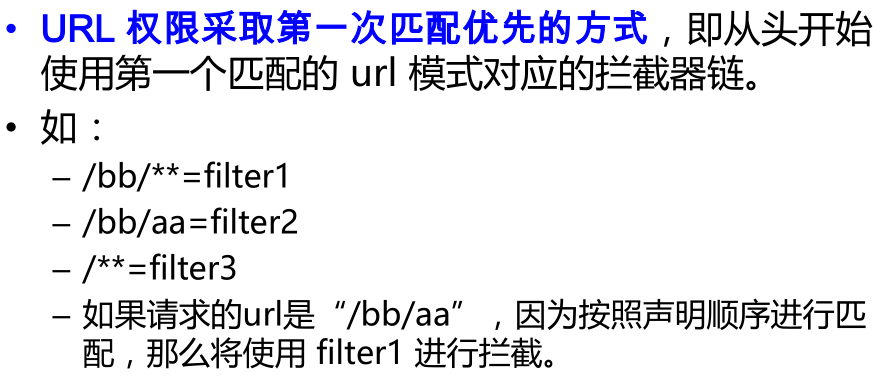
}

}

1. **配置URL权限拦截器**









1. **Spring集成shiro的demo**

**源码地址**：百度网盘->技术资料->code->shiro-2

1. **Shiro认证**

**步骤**：

1. 获取当前的 Subject. 调用 SecurityUtils.getSubject();

2. 测试当前的用户是否已经被认证. 即是否已经登录. 调用 Subject 的 isAuthenticated()

3. 若没有被认证, 则把用户名和密码封装为 UsernamePasswordToken 对象

1). 创建一个表单页面

2). 把请求提交到 SpringMVC 的 Handler

3). 获取用户名和密码.

4. 执行登录: 调用 Subject 的 login(AuthenticationToken) 方法.

5. 自定义 Realm 的方法, 从数据库中获取对应的记录, 返回给 Shiro.

1). 实际上需要继承 org.apache.shiro.realm.AuthenticatingRealm 类

2). 实现 doGetAuthenticationInfo(AuthenticationToken) 方法.

6. 由 shiro 完成对密码的比对.

**Handler代码：**

@RequestMapping("/login")

**public** String login(@RequestParam("username") String username,

@RequestParam("password") String password){

Subject currentUser = SecurityUtils.*getSubject*();

**if** (!currentUser.isAuthenticated()) {

// 把用户名和密码封装为 UsernamePasswordToken 对象

UsernamePasswordToken token = **new** UsernamePasswordToken(username, password);

// rememberme

token.setRememberMe(**true**);

**try** {

System.***out***.println("1. " + token.hashCode());

// 执行登录.

currentUser.login(token);

}

// ... catch more exceptions here (maybe custom ones specific to your application?

// 所有认证时异常的父类.

**catch** (AuthenticationException ae) {

//unexpected condition? error?

System.***out***.println("登录失败: " + ae.getMessage());

}

}

**return** "redirect:/list.jsp";

}

**Realm代码：**

**public** **class** ShiroRealm **extends** AuthorizingRealm {

@Override

**protected** AuthenticationInfo doGetAuthenticationInfo(

AuthenticationToken token) **throws** AuthenticationException {

System.***out***.println("[FirstRealm] doGetAuthenticationInfo");

//1. 把 AuthenticationToken 转换为 UsernamePasswordToken

UsernamePasswordToken upToken = (UsernamePasswordToken) token;

//2. 从 UsernamePasswordToken 中来获取 username

String username = upToken.getUsername();

//3. 调用数据库的方法, 从数据库中查询 username 对应的用户记录

System.***out***.println("从数据库中获取 username: " + username + " 所对应的用户信息.");

//4. 若用户不存在, 则可以抛出 UnknownAccountException 异常

**if**("unknown".equals(username)){

**throw** **new** UnknownAccountException("用户不存在!");

}

//5. 根据用户信息的情况, 决定是否需要抛出其他的 AuthenticationException 异常.

**if**("monster".equals(username)){

**throw** **new** LockedAccountException("用户被锁定");

}

//6. 根据用户的情况, 来构建 AuthenticationInfo 对象并返回. 通常使用的实现类为: SimpleAuthenticationInfo

//以下信息是从数据库中获取的.

//1). principal: 认证的实体信息. 可以是 username, 也可以是数据表对应的用户的实体类对象.

Object principal = username;

//2). credentials: 密码.

Object credentials = **null**; //"fc1709d0a95a6be30bc5926fdb7f22f4";

**if**("admin".equals(username)){

credentials = "038bdaf98f2037b31f1e75b5b4c9b26e";

}**else** **if**("user".equals(username)){

credentials = "098d2c478e9c11555ce2823231e02ec1";

}

//3). realmName: 当前 realm 对象的 name. 调用父类的 getName() 方法即可

String realmName = getName();

//4). 盐值.

ByteSource credentialsSalt = ByteSource.Util.*bytes*(username);

SimpleAuthenticationInfo info = **null**; //new SimpleAuthenticationInfo(principal, credentials, realmName);

info = **new** SimpleAuthenticationInfo(principal, credentials, credentialsSalt, realmName);

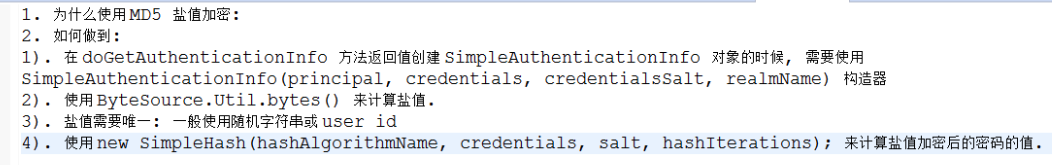
**return** info;

}

}

1. **MD5盐值加密**

有的时候，不同用户的密码可能相同，经过md5加密后的值也是一样的，这样是不安全的，我们希望不同的用户，即便是密码相同，md5加密后的值也不同，这样系统才更安全，因此引入了盐值加密



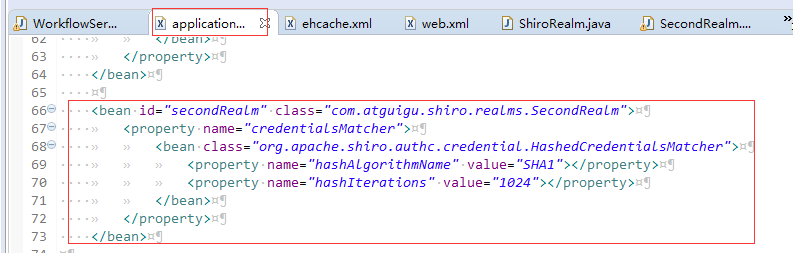
1. **Shiro多realm验证**

比如我们有两个数据库，一个是mysql，一个是oracle，两个数据库的用户表的加密方式不一样，这个时候我们需要两个realm进行认证

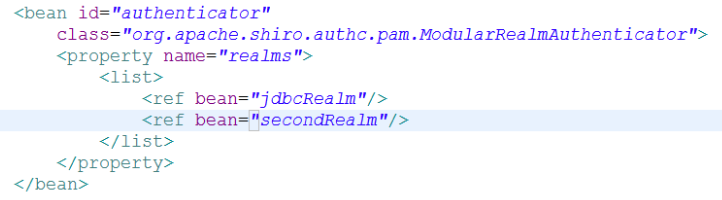
新增一个realm



将新增的realm配置到spring配置文件里

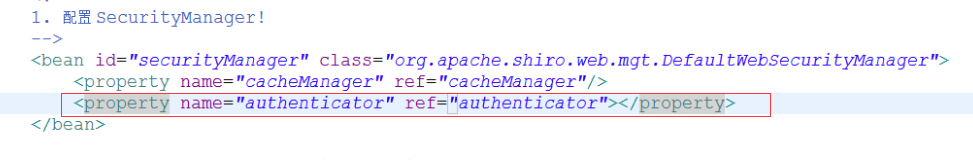


在spring配置文件里配置认证器



也可以配置在securityManager里面，推荐用这种方式配置

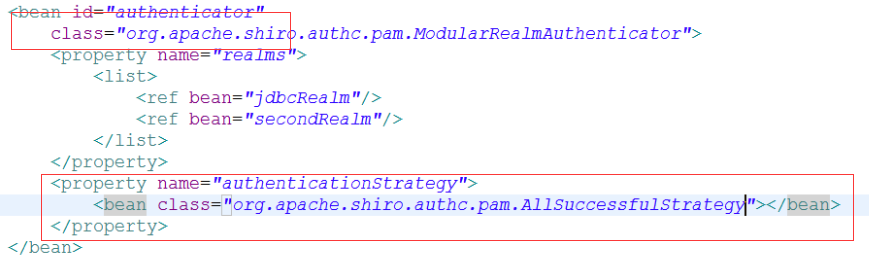




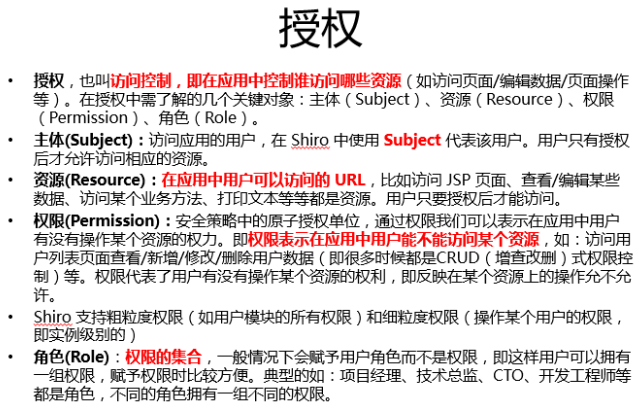
**验证策略：**

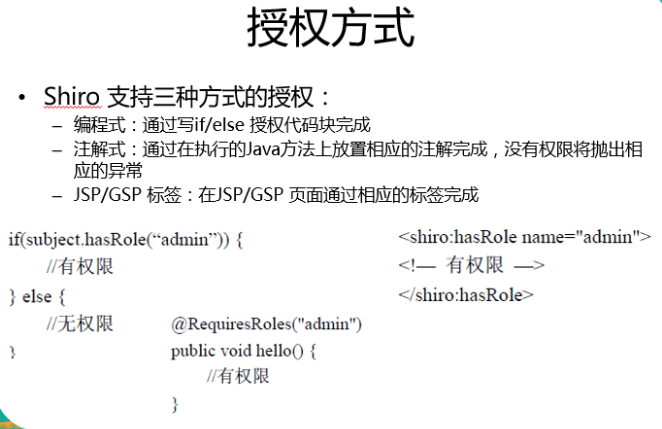


配置认证策略：



1. **授权**

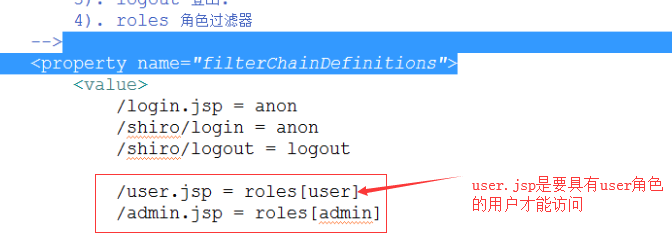


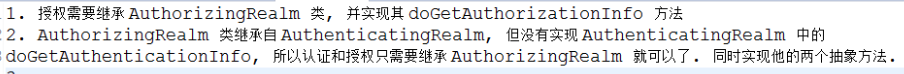


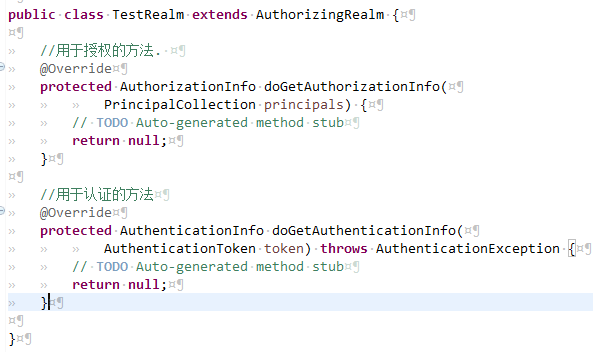
注：多用注解的方式



配置文件里面配置角色：







具体实现：

//授权会被 shiro 回调的方法

@Override

**protected** AuthorizationInfo doGetAuthorizationInfo(

PrincipalCollection principals) {

//1. 从 PrincipalCollection 中来获取登录用户的信息

Object principal = principals.getPrimaryPrincipal();

//2. 利用登录的用户的信息来用户当前用户的角色或权限(可能需要查询数据库)

Set<String> roles = **new** HashSet<>();

roles.add("user");

**if**("admin".equals(principal)){

roles.add("admin");

}

//3. 创建 SimpleAuthorizationInfo, 并设置其 roles 属性.

SimpleAuthorizationInfo info = **new** SimpleAuthorizationInfo(roles);

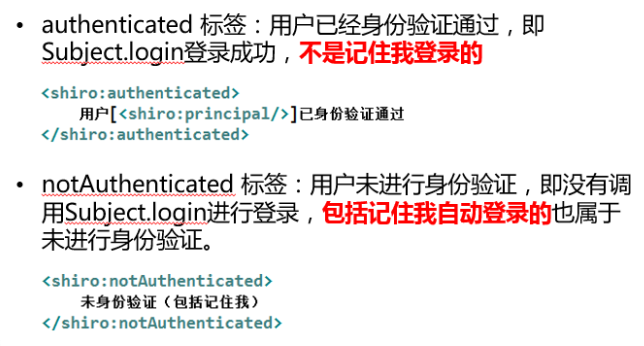
//4. 返回 SimpleAuthorizationInfo 对象.

**return** info;

}

1. **Shiro标签**





1. **Shiro注解**



1. **A**