# 1.通过配置文件获得用户并判断角色和角色数组

## 1.配置文件 shiro-role.ini

[users]  
zhang=123,role1,role2  
wang=123,role1

## 2.编写一个基础的登录验证

**public abstract class** BaseTest {  
  
  
 @After  
 **public void** tearDown() **throws** Exception {  
 ThreadContext.*unbindSubject*();*//退出时请解除绑定Subject到线程 否则对下次测试造成影响*

System.***out***.println(**"\*\*\*\*\*\* ThreadContext.unbindSubject();\*\*\*\*\*\*"**);

}  
  
 **protected void** login(String configFile, String username, String password) {  
Factory< SecurityManager> factory =**new** IniSecurityManagerFactory(configFile);  
  
 *//2、得到SecurityManager实例 并绑定给SecurityUtils* org.apache.shiro.mgt.SecurityManager securityManager = factory.getInstance();  
 SecurityUtils.*setSecurityManager*(securityManager);  
  
 *//3、得到Subject及创建用户名/密码身份验证Token（即用户身份/凭证）* Subject subject = SecurityUtils.*getSubject*();  
 UsernamePasswordToken token = **new** UsernamePasswordToken(username, password);  
  
 subject.login(token);  
 }  
  
 **public** Subject subject() {  
 **return** SecurityUtils.*getSubject*();  
 }  
}

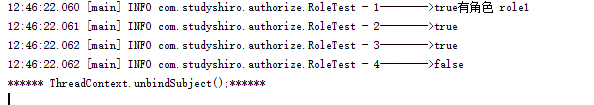
## 3.角色，和角色数组 验证

**public class** RoleTest **extends** BaseTest {  
 **private** Logger **logger** = LoggerFactory.*getLogger*(RoleTest.**class**);  
  
 @Test  
 **public void** testHasRole() {  
 login(**"classpath:shiro-role.ini"**, **"zhang"**, **"123"**);  
 *//判断拥有角色：role1* **logger**.info(**"1-------->"**+subject().hasRole(**"role1"**)+**"有角色 role1"**);  
 *//判断拥有角色：role1 and role2 and !role3* **boolean**[] result = subject().hasRoles(Arrays.*asList*(**"role1"**, **"role2"**, **"role3"**));  
 **logger**.info(**"2-------->"**+result[0]);  
 **logger**.info(**"3-------->"**+result[1]);  
 **logger**.info(**"4-------->"**+result[2]);  
  
 }

## 4.subject().checkRole断言有角色（返回的是空方法）

@Test(expected = UnauthorizedException.**class**)  
**public void** testCheckRole() {  
 login(**"classpath:shiro-role.ini"**, **"zhang"**, **"123"**);  
 *//断言拥有角色：role1* subject().checkRole(**"role1"**);   
 *//断言拥有角色：role1 and role3 失败抛出异常* subject().checkRoles(**"role1"**, **"role3"**);

## 5.验证成功



# 2.通过配置文件获得用户判断权限和权限数组

## 1.编写配置文件shiro-permission.ini

[users]  
zhang=123,role1,role2  
wang=123,role1  
li=123,role41,role42,role51,role52,role53,role61,role62,role71,role72,role73,role74,role75,role81,role82  
  
  
[roles]  
#对资源user拥有create、update权限  
role1=user:create,user:update  
#对资源user拥有create、delete权限  
role2=user:create,user:delete  
#对资源user拥有create权限  
role3=system:user:create

## 2. 权限和权限数组验证

@Test  
**public void** testIsPermitted() {  
 login(**"classpath:shiro-permission.ini"**, **"zhang"**, **"123"**);  
 *//判断拥有权限：user:create* Assert.*assertTrue*(subject().isPermitted(**"user:create"**));  
 *//判断拥有权限：user:update and user:delete* Assert.*assertTrue*(subject().isPermittedAll(**"user:update"**, **"user:delete"**));  
 *//判断没有权限：user:view* Assert.*assertFalse*(subject().isPermitted(**"user:view"**));  
}  
  
@Test(expected = UnauthorizedException.**class**)  
**public void** testCheckPermission() {  
 login(**"classpath:shiro-permission.ini"**, **"zhang"**, **"123"**);  
 *//断言拥有权限：user:create* subject().checkPermission(**"user:create"**);  
 *//断言拥有权限：user:delete and user:update* subject().checkPermissions(**"user:delete"**, **"user:update"**);  
 *//断言拥有权限：user:view 失败抛出异常* subject().checkPermissions(**"user:view"**);  
}

## 3.验证成功