# ERP项目笔记 Day-13

## 权限验证框架Shiro

1. Shiro简介
   1. 什么是Shiro

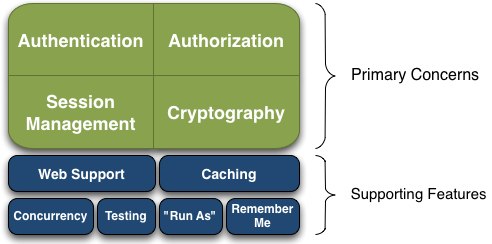
Apache Shiro是一个强大易用的Java安全框架，提供了认证、授权、加密和会话管理等功能：

认证（Authentication）：用户身份识别，常被称为用户“登录”，判断用户是否登陆，如果未登陆，则拦截其请求

授权（Authorization）：访问控制。当用户登陆后，判断其身份是否有权限访问相应的资源，如果没有权限则拦截

密码加密（Cryptography）：保护或隐藏数据防止被偷窃。将MD5进行二次封装，让其更加容易使用。注意MD5不可逆运算

会话管理（Session Management）



* 1. Shiro内置过滤器

|  |  |
| --- | --- |
| anon | org.apache.shiro.web.filter.authc.AnonymousFilter |
| authc | org.apache.shiro.web.filter.authc.FormAuthenticationFilter |
| authcBasic | org.apache.shiro.web.filter.authc.BasicHttpAuthenticationFilter |
| perms | org.apache.shiro.web.filter.authz.PermissionsAuthorizationFilter |
| port | org.apache.shiro.web.filter.authz.PortFilter |
| rest | org.apache.shiro.web.filter.authz.HttpMethodPermissionFilter |
| roles | org.apache.shiro.web.filter.authz.RolesAuthorizationFilter |
| ssl | org.apache.shiro.web.filter.authz.SslFilter |
| user | org.apache.shiro.web.filter.authc.UserFilter |

这些过滤器分为两组：

* 认证过滤器：**anon(不认证也可以访问)**，authcBasic, **authc(必须认证后才可**

**访问)**

* 授权过滤器：**perms（指定资源需要哪些权限才可以访问）**，Roles, ssl, rest, port

**ERP项目中所使用到过滤器**

anon: /admins/\*=anon 没有参数，表示可以匿名使用

authc: /admins/user/\*=authc

perms: /admins/user/\*=perms[“权限名称”]，参数可以写多个，多个之间用逗号

* 1. ERP整合Shiro

1. 添加依赖:

给erp\_parent父工程添加依赖。

定义版本号：





1. 配置web.xml, 添加过滤器代理DelegatingFilterProxy，要放在struts2的核心过滤器之前



Spring提供的一个简便的过滤器处理方案，它将具体的操作交给内部的Filter对象delegate去处理，而这个delegate对象通过Spring的IOC容器获取，这里采用的是Spring的FactorBean的方式获取这个对象。

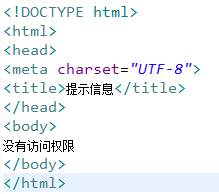
虽然中配置了这一个filter，但是它并没做任何实际的工作，而是把这个工作交由Spring容器中一个bean的id为shiroFilter的类，即ShiroFilterFactoryBean。

1. 添加shiro核心控制器的spring配置文件

复制资料中的applicationContext\_shiro.xml文件到erp\_web的资源目录下



1. 创建error.html



1. 认证
   1. 需求分析

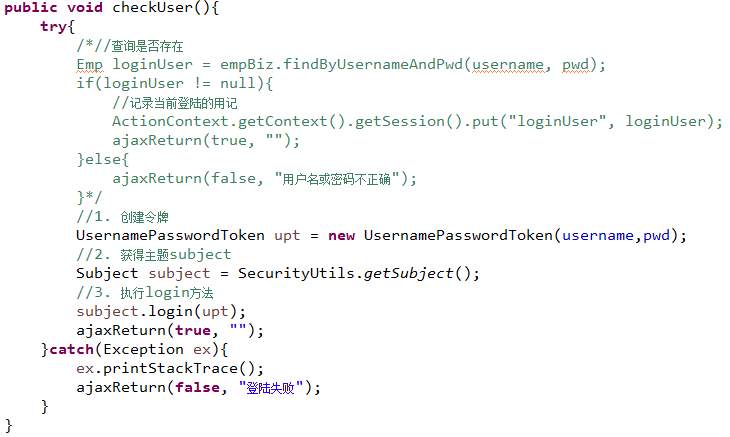
判断当前用户是否登陆，如果没有登陆则跳转到登陆页面

* 1. 认证实现（重点）
     1. Subject的login方法

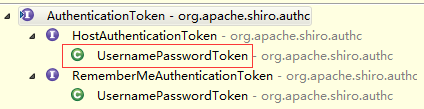
修改LoginAction的checkUser方法。

步骤：

* 创建令牌：UsernamePassowrdToken
* 获取subject
* 执行subject.login()方法



UsernamePasswordTokey是AuthenticationToken的子接口的实现类，它是对用户名和密码的封装。



Subject是对当前用户执行操作的封装，此处我们用它来执行登陆的操作。如果用户名和密码错误，login方法会抛出AuthenticationException异常

* + 1. 自定义Realm

我们改用subject.login方法后，并不会调用登陆的业务层进行登陆的验证查询，即不会从数据库查找登陆的用户名和密码是否正确，而是将这项工作交给shiro去完成。那shiro是怎么知道登陆的用户名和密码是否正确的呢？其实它也需要用到我的登陆验证业务，这时它就得向“别人”打听一下，那就是Realm了。

真正实现登陆验证的是Realm，而shiro只是去调Realm

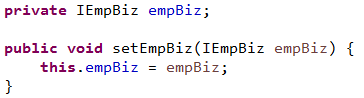
Realm： Realm充当了Shiro与应用安全数据间的“桥梁”或者“连接器”。也就是说，当对用户执行认证（登录）和授权（访问控制）验证时，Shiro会从应用配置的Realm中查找用户及其权限信息。   
 从这个意义上讲，Realm实质上是一个安全相关的DAO：它封装了数据源的连接细节，并在需要时将相关数据提供给Shiro。当配置Shiro时，你必须至少指定一个Realm，用于认证和（或）授权。配置多个Realm是可以的，但是至少需要一个。

1. 在erp\_web子工程下创建包cn.itcast.erp.realm。
2. 创建ErpRealm类继承自AuthorizingRealm



1. 实现登陆验证

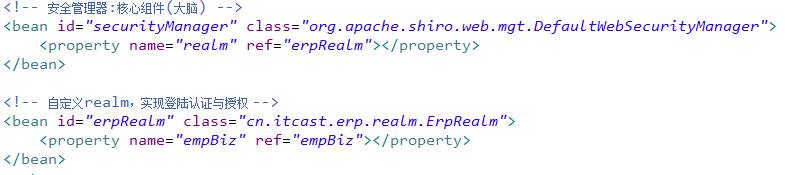
注入IEmpBiz



修改认证方法



配置ErpRealm, 在applicationContext\_shiro.xml中配置

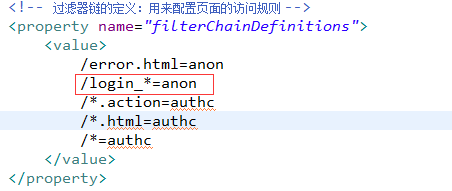


删除LoginAction中注入的IEmpBiz，配置文件也要同步删除。

* + 1. Url访问规则

运行项目，发现无法登陆。通过查看浏览器的network项看到：发送login\_checkUser请求的返回码为302，302的意思是页面跳转。由于我们配置了/\*=authc，这个Url规则包含login\_checkUser，所以无法执行并强行跳转到登陆页面。

修改applicationContext\_shiro.xml中的配置



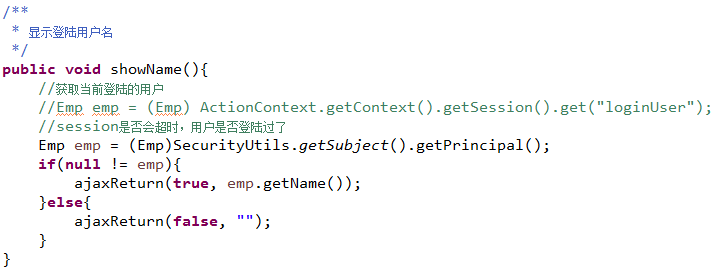
anon为内置的匿名认证过滤器，也就是不需要认证就可以访问。

authc需要认证后才能访问

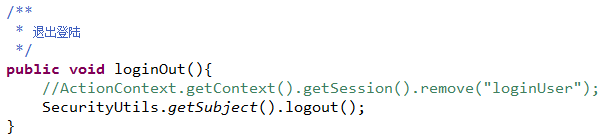
* + 1. 主角对象的提取

Shiro提供了会话管理机制，实际上我们自定义的realm的认证方法返回值对象中的主角对象就是登陆的用户，它可以代替我们之前存入session中的emp对象。我们可以通过subject的getPrincipal方法将其提取出来。

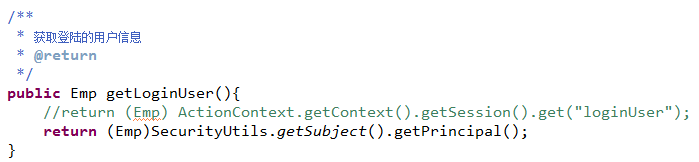
1. 修改LoginAction中的showName方法



1. 修改LoginAction中的退出登陆方法



1. 修改BaseAction中的getLoginUser方法

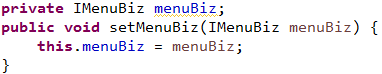


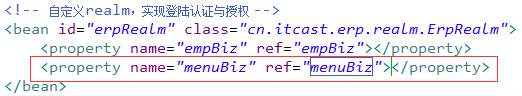
1. 授权
   1. 需求分析

授权就是通过设置规则，指定哪些URL需要哪些权限才可以访问

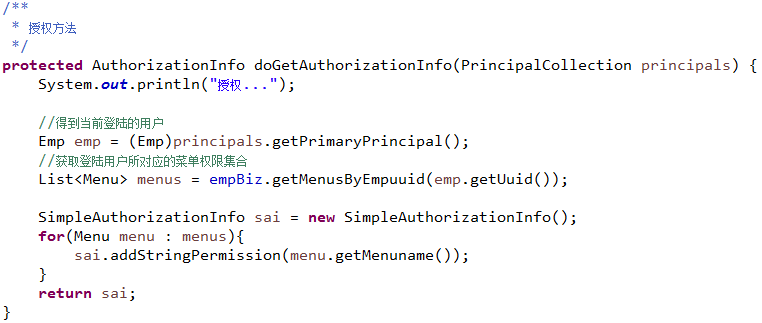
* 1. 授权的实现
     1. 授权方法与配置（重点）

1. 编写授权方法
   1. 注入IMenuBiz，用于查询当前登陆用户所具的菜单权限信息

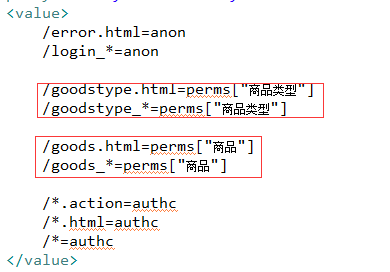




* 1. 修改ErpRealm中的授权方法：



1. 配置授权控制规则



1. 小结：

* 授权方法的作用：告诉shiro当前用户有什么权限
* 配置信息的作用：告诉shiro什么资源有什么权限才可以访问
  + 1. 自定义授权过滤器

当一个URL有多个权限需要访问的时候，我们应该怎么配置呢？

如果按下面的方法来配置

/orders.html=perms[“采购订单查询”]

/orders.html=perms[“采购订单审核”]

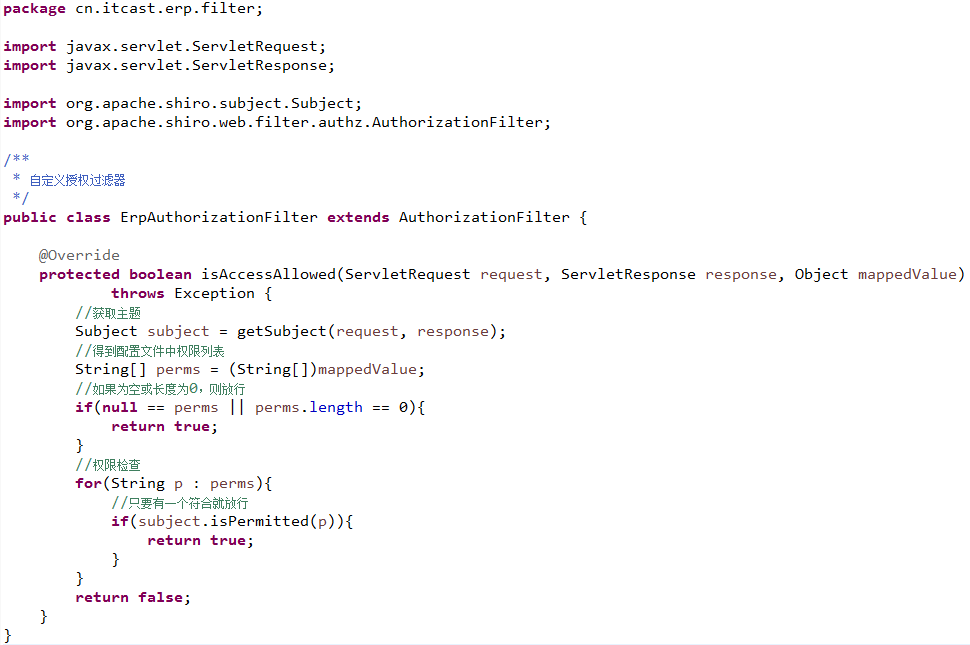
那么只有最后一条生效，前面的会被后面的覆盖掉

如果按下面的方法来配置

/orders.html=perms[“采购订单查询”,“采购订单审核”]

系统默认是同时具备这两个权限才可以访问此URL，而我们的需求是，只要有具备一种就可以访问此URL。系统使用的是and关系，而不是or关系，那我们如何让它实现or关系呢？这就需要我们去自定义授权过滤器啦

1. 创建自定义过滤器，继承自AuthorizationFilter

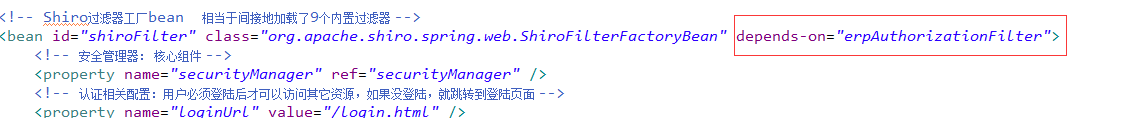


1. 配置过滤器

在applicationContext\_shiro.xml中添加





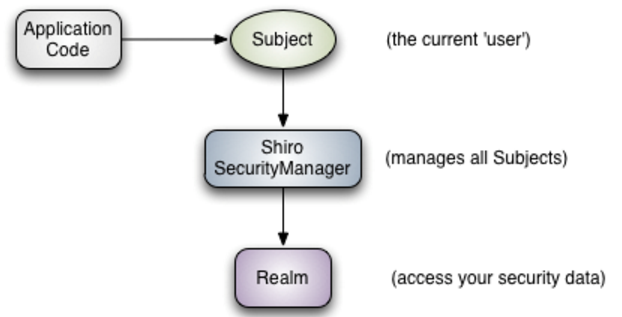


1. 配置或的关系



1. 知识点总结

Shiro的三大核心组件:



Subject：正与系统进行交互的人，或某一个第三方服务。所有Subject实例都被绑定到(且这是必须的)一个SecurityManager上。

SecurityManager：Shiro架构的心脏，典型的Facade模式。用来协调内部各安全组件，管理内部组件实例，并通过它来提供安全管理的各种服务。当Shiro与一个Subject进行交互时，实质上是幕后的SecurityManager处理所有繁重的Subject安全操作。

Realms：本质上是一个特定安全的DAO。当配置Shiro时，必须指定至少一个Realm用来进行身份验证和/或授权。Shiro提供了多种可用的Realms来获取安全相关的数据。如关系数据库(JDBC)，INI及属性文件等。可以定义自己Realm实现来代表自定义的数据源。