**Apache 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**功能，可为任何应用提供安全保障 - 从命令行应用、移动应用到大型网络及企业应用。

Shiro为解决下列问题（应用安全的四要素）提供保护API：

**·认证** - 用户身份识别，常被称为用户“登录”；

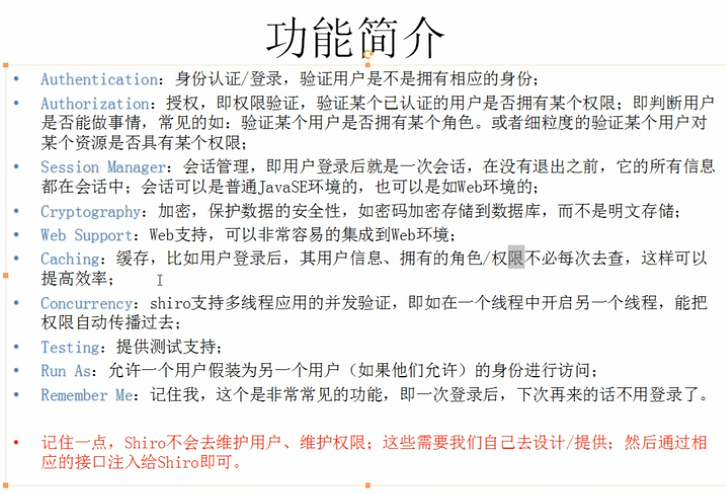
**·授权** - 访问控制；

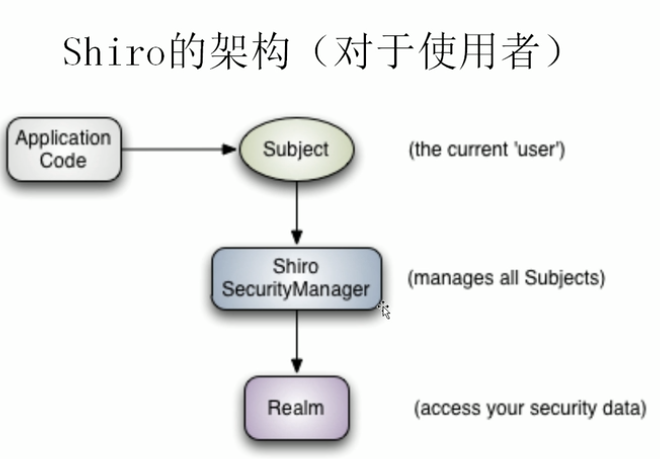
**·密码加密** - 保护或隐藏数据防止被偷窥；

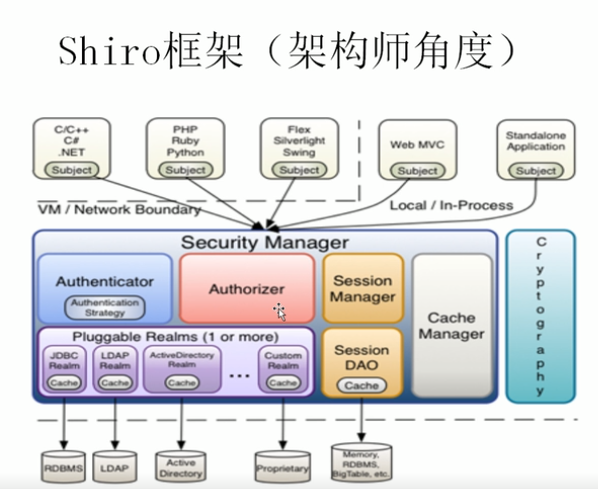
**·会话管理** - 每用户相关的时间敏感的状态。

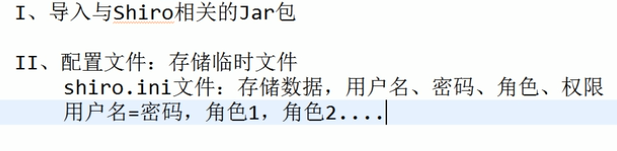
**·web集成**

**·缓存**

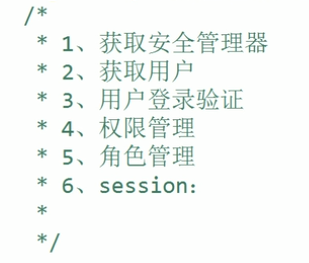














安全管理：

<!-- 安全管理器 -->

<bean id=*"securityManager"* class=*"org.apache.shiro.web.mgt.DefaultWebSecurityManager"*>

<property name=*"sessionManager"* ref=*"sessionManager"* />

<property name=*"rememberMeManager"* ref=*"rememberMeManager"* />

<property name=*"cacheManager"* ref=*"shiroCacheManager"* />

<property name=*"realm"* ref=*"sampleRealm"* />

</bean>

缓存管理：

<!-- shiro权限缓存管理 -->

<bean id=*"shiroCacheManager"* class=*"org.apache.shiro.cache.ehcache.EhCacheManager"*>

<property name=*"cacheManager"* ref=*"ehCacheManager"* />

</bean>

<!-- MyBatis集成ehcache缓存 -->

<bean id=*"ehCacheManager"* class=*"org.springframework.cache.ehcache.EhCacheManagerFactoryBean"*>

<property name=*"configLocation"* value=*"classpath:ehcache-shiro.xml"* />

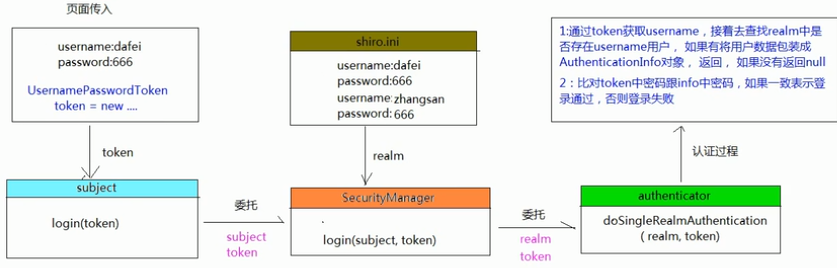
<property name=*"shared"* value=*"true"*></property>

</bean>

【Subject于securityManager两者区别】

Subject实现的是登录 登出 访问会话 执行授权检查

securityManager实现subject的委托的各种功能 具体是执行与其他各个组件之间的相互调用



**·**主体登录->securityManager->委托Authentication认证(对比realm和输入的账号密码是否一致进行判断）

**·动态设置realm的值：**

**shiro-realm.ini:**

myRealm=com.lingxuanyu.relam.MyRelam

securityManager.realms=$myRealm

**public** **class** MyRelam **extends** AuthorizingRealm{

//认证entication

@Override

protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {

String username = (String) token.getPrincipal();

System.out.println(username);

if(!"user1".equals(username)) {

return null;

}

String password = "12345";

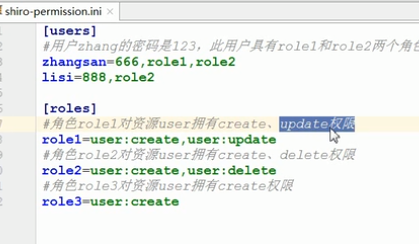
SimpleAuthenticationInfo simpleAuthenticationInfo = new SimpleAuthenticationInfo(username,password,getName());

return simpleAuthenticationInfo;

}

}

权限授权方式：





【权限判断和角色判断都需要**subject.login()**登录认证之后方可使用】

**角色判断:**

System.out.println("user isLogin: " + subject.isAuthenticated());

System.out.println(subject.hasRole("role1"));

System.out.println(subject.hasAllRoles(Arrays.asList("role1","role2")));

System.out.println(Arrays.toString(subject.hasRoles(Arrays.asList("role1","role2"))));

**权限判断：**

System.***out***.println("user permission: " + subject.isPermitted("user:select"));

System.***out***.println("users all persmission: " + subject.isPermittedAll("user:delete","user:updat"));

System.***out***.println("user permission's list:" + Arrays.*toString*(subject.isPermitted("user:delete","user:create")));

**CheckRole()与checkPermission()两个方法判断角色与权限不存在时直接报错！！！**

**授权：**

**Shiro.ini:**

myRealm=com.lingxuanyu.relam.PermissionRealm

securityManager.realms=$myRealm

**PermissionRealm:**

**//授权orization**

**@Override**

**protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals) {**

**String username = (String) principals.getPrimaryPrincipal();**

**ArrayList<String> roles = new ArrayList<String>();**

**ArrayList<String> permissions = new ArrayList<String>();**

**//从数据库查询相应的信息**

**roles.add("role1");**

**permissions.add("user:update");**

**SimpleAuthorizationInfo simpleAuthorizationInfo = new SimpleAuthorizationInfo();**

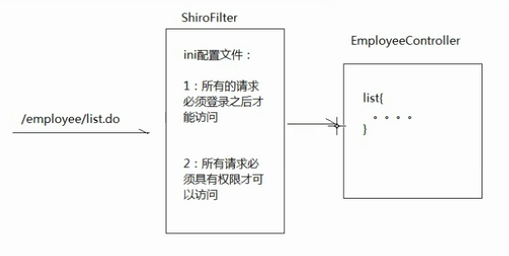
**simpleAuthorizationInfo.addRoles(roles);**

**simpleAuthorizationInfo.addStringPermissions(permissions);**

**return simpleAuthorizationInfo;**

**}**

Web集成shiro原理



**jar包依赖： Shiro-web**

**web.xml**

**·配置ShiroFilter过滤器 拦截所有的请求：**

**Org.apache.shiro.web.servlet.ShiroFilter -----> /\***

**· <listener-class>org.apache.shiro.web.env.EnviromentLoaderListener</listener-class>**

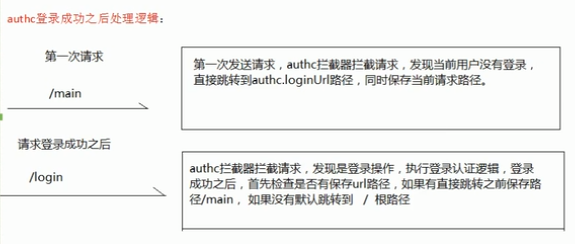




Shiro拦截器：



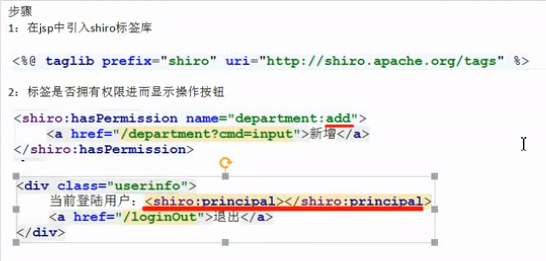




Shiro的标签的权限设置：



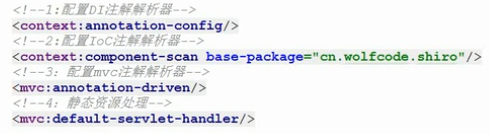
标签权限设置：



退出登录（logout过滤器）：







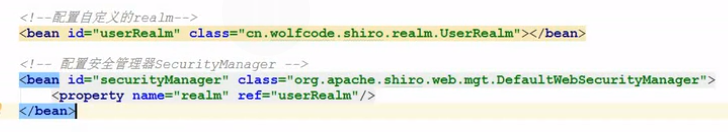
Spring-shiro集成

配置文件： ·自定义Realm文件(集成授权realm）

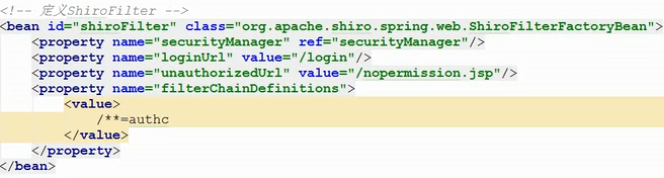
·安全管理器SecurityManager

·Shiro的Web过滤器

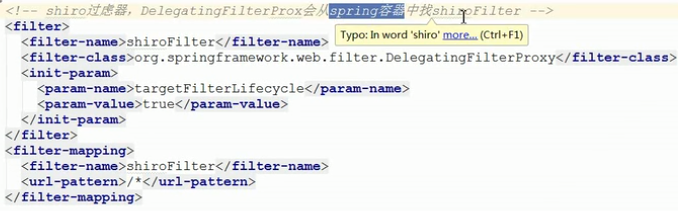
**Spring-shiro.xml:**



**配置代理对象：(id必须和web.xml)文件中的shiroFilter的名字是一样的**



**web.xml:**





静态注解授权：

spring-shiro配置：



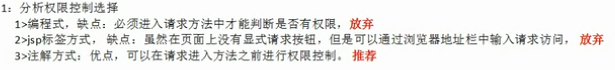
Springmvc对于没有权限会作为异常进行处理，所以需要配置实现没有权限跳转：



数据库未参与：（写死的静态授权）

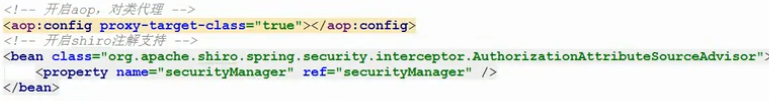


方式：





spring-shiro:

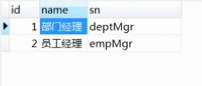


**使用数据库方式进行授权：**

说明： 用户<--->角色 角色<--->权限两两之间均是多对多的关系（five tables）



Role:



**Permission:**



**User:**



**加载权限表达式：（获取到方法头上的@requirePermission(xxx)）的注解，存储到数据库中，存储的时候，对于权限数据库ID NAME RESOURCE的NAME，可以自定义一个ANNOTATION注解进行配置；**



依据数据的权限查询对应的角色和权限进行分配：



## 缓存管理

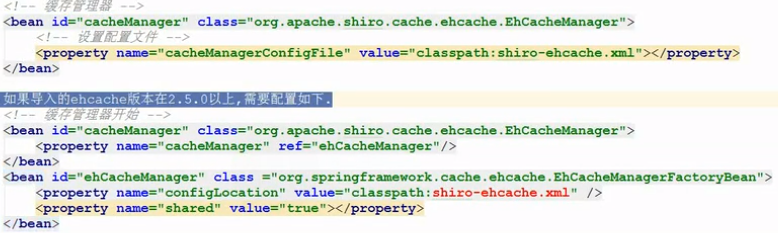
**配置缓存**

JAR包： ehcache-core

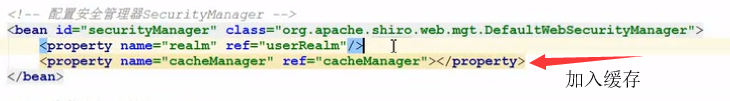
Shiro-ehcache

配置缓存管理器

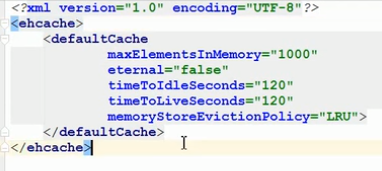
Spring-shiro.xml:



【说明】:shiro-ehcache.xml文件时shiro缓存管理文件



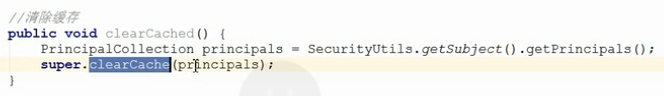
Shiro-ehcache.xml文件：



## 清除缓存

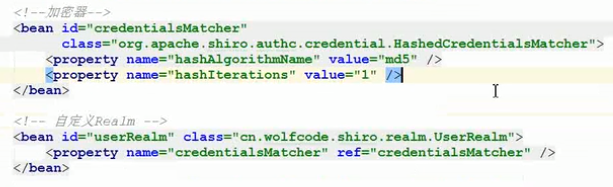
修改数据库用户角色之后，使其权限立即生效！

修改REALM.java中，添加如下clearCached()方法；



密码加密

Spring-shiro.xml:



加入自定义的加密器：



修改REALM.java中的认证操作： 