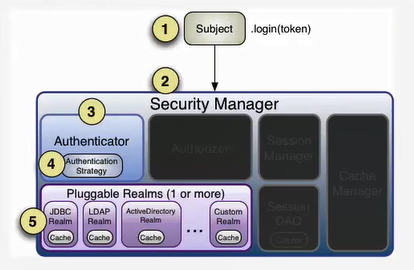
**Subject**主体包含两个信息

**Principals**：身份，可以是用户名、邮件、手机号码等，用来标识一个登录主体身份；

**Credentials**：凭证，常见有密码、数字证书等

**认证流程**



认证时，将token发送给SecurityManager，通过认证策略转发给配置的Realm，完成具体的认证。

**权限认证，三要素**

**权限** 操作资源的权力

**角色** 权限的集合

**用户** 访问系统的用户，即Subject

**授权**

**基于角色授权**，在ini文件中配置role1,rol2,…

通过Subject对象的hasRole()方法判断是否包含该角色。hasRoles()返回一个数组，hasAllRoles()判断是否包含所有角色。

**基于权限授权**，在ini文件中配置

[roles]

role1=user:select

role2=user:insert,user:update,user:delete

通过Subject对象的isPermitted()方法判断是否包含该权限。

**注解式授权**

@RequiresAuthentication 当前Subject在当前的Session中被验证通过才能被访问或者调用。

@RequiresGuest当前Subject是一个guest

@RequiresPermission(“account:create”) 权限要求

@RequiresRoles(“administrator”) 角色要求

@RequiresUser 当前Subject是一个应用程序用户

**JSP标签授权**

在JSP中导入

<%@ taglib prefix=”shiro” uri=”http://shiro.apache.org/tags” %>

后即可使用shiro标签做权限控制

**Permissions权限扩展**

单个权限 query

单个资源多个权限 user:query,user:add 多值 user:query,add

单个资源所有权限 user:query,add,update,delete user:\*

所有资源某个权限 \*:view

实例级别的权限控制

单个实例的单个权限 printer:query:lp7200 printer:print:epsoncolor

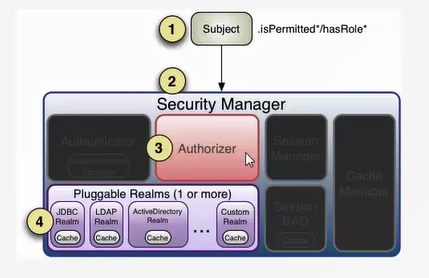
所有实例的单个权限 printer:print:\*

所有实例的所有权限 printer:\*:\*

单个实例的所有权限 printer:\*:lp7200

单个实例的多个权限 printer:query,print:lp7200

**授权流程**



**Web开发中配置认证url**

[main]

authc.loginUrl=/login

**配置url需要做认证，或者哪些权限判断**

[urls]

/login=anno

/admin=authc

/student=roles[teacher]

/teacher=perms[“user:create”]

**url的匹配方式**

精确匹配 /admin

?匹配一个字符 /admin?可以匹配/admin1 /admin2，但是不能匹配/admin12

\*匹配任意多个字符，但是不能多路径 /admin\*可以匹配/admin1 /admin123，但是不能匹配/admin/abc

\*\*匹配多路径 /admin/\*\*可以匹配/admin /admin/abc

/admin\*/\*\*可以匹配任意/admin开头的路径

**自定义Realm**

首先，要在数据库中维护用户、角色、权限表

新建一个类，继承AuthorizingRealm，实现两个方法1）验证当前登录的用户doGetAuthenticationInfo；2）为当前登录的用户授予角色和权限doGetAuthorizationInfo。

配置Realm

myRealm=com.xxx.MyRealm

securityManager.realms=$myRealm

**加密**

对称加解密

Base64.encodeToString(str.getBytes());

Base64.decodeToString(str)

Md5加密、不可逆

new Md5Hash(str, salt).toString()

md5不可逆的原因，采用hash算法，部分原文丢失