# 目录

[目录 1](#_Toc386464012)

[版本修订 3](#_Toc386464013)

[1 背景 4](#_Toc386464014)

[2 目的 4](#_Toc386464015)

[3 缩写 4](#_Toc386464016)

[4 内容 4](#_Toc386464017)

[5 认证授权相关术语扫盲 4](#_Toc386464018)

[5.1 身份验证(Authentication) 5](#_Toc386464019)

[5.2 授权(Authorization) 5](#_Toc386464020)

[5.3 主体(Subject) 6](#_Toc386464021)

[5.4 资源(Resource) 6](#_Toc386464022)

[5.5 权限(Permission) 6](#_Toc386464023)

[5.6 角色(Role) 6](#_Toc386464024)

[5.7 RBAC模型 6](#_Toc386464025)

[5.8 SSO 7](#_Toc386464026)

[5.9 用户权限模型 7](#_Toc386464027)

[6 安全框架选型 8](#_Toc386464028)

[7 Apache Shiro项目 9](#_Toc386464029)

[7.1 Apache Shiro基础 9](#_Toc386464030)

[7.2 Apache Shiro基础概念 10](#_Toc386464031)

[7.2.1 Shiro Subject 10](#_Toc386464032)

[7.2.2 Shiro SecurityManager 10](#_Toc386464033)

[7.2.3 Shiro Realm 11](#_Toc386464034)

[7.2.4 Shiro SecurityUtils 11](#_Toc386464035)

[7.2.5 用户令牌 11](#_Toc386464036)

[7.3 Shiro认证 11](#_Toc386464037)

[7.3.1 身份认证涉及的概念 11](#_Toc386464038)

[7.3.2 身份认证流程 12](#_Toc386464039)

[7.1 身份认证流程 12](#_Toc386464040)

[7.2 Shiro授权 13](#_Toc386464041)

[7.2.1 Shiro所涉及的概念 13](#_Toc386464042)

[7.2.2 授权方式 14](#_Toc386464043)

[7.3 缓存 15](#_Toc386464044)

[8 Apche Shiro实践 15](#_Toc386464045)

[8.1 Hello,Shiro 15](#_Toc386464046)

[8.2 在 Web 应用中使用 Shiro 15](#_Toc386464047)

[8.3 Spring 与 Shiro 整合 15](#_Toc386464048)

[9 实现思路与步骤 15](#_Toc386464049)

[9.1 密码重试次数限制 15](#_Toc386464050)

[9.2 SSO 16](#_Toc386464051)

[9.2.1 CAS 16](#_Toc386464052)

# 版本修订

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 日期 | 描述 | 修订人 |
| V1.0 | 2014-4-18 | 创建 | 陈绪绍 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 背景

随着3D项目越来越多及3D平台化的需求，有必要增加身份认证与权限管理等安全控制功能。

# 目的

1. 提供最实用的功能，忽略不常用的功能，但提供扩展机制。

2. 隐藏具体实现细节，底层实现(如权限模型来源)可以任意切换，从而不会影响到应用程序的改动。

# 缩写词

后期添加

# 内容

1. 认证授权相关术语扫盲
2. Apache Shiro基础
3. 单点登录(SSO)
4. 3D平台认证授权解决方案

就安全方面的内容，实在是一个较为宽广和复杂的内容。本文档尽量以浅显的语言，让读者理解这方面内容，并找出适合3D平台的权限管理框架。

本预研文档，是一个迭代完善的过程。在后续的技术预研和实践当中，逐步添加与完善相关章节的内容。

# 认证授权相关术语扫盲

在对系统进行安全保障时，有两个安全性元素非常重要：身份验证和授权。认证和授权为权限控制的核心，简单来说，“认证”就是证明你是谁？ Web 应用程序一般做法通过表单提交用户名及密码达到认证目的。“授权”即是否允许已认证用户访问受保护资源。虽然这两个术语代表的是不同的含义，但出于它们在应用程序安全性方面各自的角色考虑，它们有时会被交换使用。

在这进行这些技术进行阐述时，会涉及如下一些术语，如身份验证、授权、主体（Subject）、资源（Resource）、权限（Permission）、角色（Role）等。本章节将对这些术语进行基本的介绍。

## 身份验证(Authentication)

身份验证指的是验证用户的身份。在验证用户身份时，需要确认用户的身份的确如他们所声称的那样。在大多数应用程序中，身份验证是通过用户名和密码的组合完成的。只要用户选择了他人很难猜到的密码，那么用户名和密码的组合通常就足以确立身份。但是，还有其他的身份验证方式可用，比如指纹、证书等。

在shiro中，用户需要提供principals （身份）和credentials（证明）给shiro，从而应用能验证用户身份：

principals：身份，即主体的标识属性，可以是任何东西，如用户名、邮箱等，唯一即可。一个主体可以有多个principals，但只有一个Primary principals，一般是用户名/密码/手机号。

credentials：证明/凭证，即只有主体知道的安全值，如密码/数字证书等。

最常见的principals和credentials组合就是用户名/密码了。接下来先进行一个基本的身份认证。

## 授权(Authorization)

一旦身份验证过程成功地建立起身份，授权就会接管以便进行访问的限制或允许。 所以，有这样的可能性：用户虽然通过了身份验证可以登录到一个系统，但是未经过授权，不准做任何事情。还有一种可能是用户虽然具有了某种程度的授权，却并未经过身份验证。

简单的讲，授权也叫访问控制，即在应用中控制谁能访问哪些资源（如访问页面/编辑数据/页面操作等）。

## 主体(Subject)

主体，即访问应用的用户，在Shiro中使用Subject代表该用户。用户只有授权后才允许访问相应的资源。

## 资源(Resource)

在应用中用户可以访问的任何东西，比如访问JSP页面、查看/编辑某些数据、访问某个业务方法、打印文本等等都是资源。用户只要授权后才能访问。

## 权限(Permission)

安全策略中的原子授权单位，通过权限我们可以表示在应用中用户有没有操作某个资源的权力。即权限表示在应用中用户能不能访问某个资源，如：

* 访问用户列表页面
* 查看/新增/修改/删除用户数据（即很多时候都是CRUD式权限控制）
* 打印文档

如上可以看出，权限代表了用户有没有操作某个资源的权利，即反映在某个资源上的操作允不允许，不反映谁去执行这个操作。所以后续还需要把权限赋予给用。

## 角色(Role)

角色代表了操作集合，可以理解为权限的集合，一般情况下我们会赋予用户角色而不是权限，即这样用户可以拥有一组权限，赋予权限时比较方便。典型的如：项目经理、技术总监、CTO、开发工程师等都是角色，不同的角色拥有一组不同的权限。

## RBAC模型

既然角色代表了可执行的操作这一概念，一个合乎逻辑的做法是在软件开发中使用角色来控制对软件功能和数据的访问。这种权限控制方法就叫基于角色的访问控制(Role-Based Access Control)，简称为RBAC。

## SSO

单点登录SSO（Single Sign-On）是身份管理中的一部分。SSO的一种较为通俗的定义是：SSO是指访问同一服务器不同应用中的受保护资源的同一用户，只需要登录一次，即通过一个应用中的安全验证后，再访问其他应用中的受保护资源时，不再需要重新登录验证。

目前的企业应用环境中，往往有很多的应用系统，如办公自动化（OA）系统，财务管理系统，档案管理系统，信息查询系统等等。这些应用系统服务于企业的信息化建设，为企业带来了很好的效益。但是，用户在使用这些应用系统时，并不方便。用户每次使用系统，都必须输入用户名称和用户密码，进行身份验证；而且应用系统不同，用户账号就不同，用户必须同时牢记多套用户名称和用户密码。特别是对于应用系统数目较多，用户数目也很多的企业，这个问题尤为突出。问题的原因并不是系统开发出现失误，而是缺少整体规划，缺乏统一的用户登录平台，使用SSO技术可以解决以上这些问题。

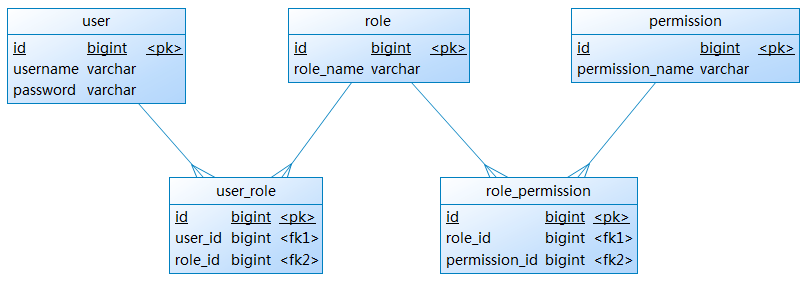
SSO 就是为了增强用户体验的，比如有 10 个系统，当用户登录了其中 1 个系统，其余的 9 个系统无需再次登录就可以自由访问，好比这 10 个系统就是一个系统，而不需要用户不断地去登录这 10 个系统。所以我们可以将这 10 个系统做成一个 Portal（门户），让这个 Portal 来展示这些系统之间的页面或数据。

## 用户权限模型

所谓用户权限模型，指的是用来表达用户信息及用户权限信息的数据模型。即能证明“你是谁？”、“你能访问多少受保护资源？”。为实现一个较为灵活的用户权限数据模型，通常把用户信息单独用一个实体表示，用户权限信息用两个实体表示。如下：

1. 用户信息用 User 表示，最简单的用户信息可能只包含用户名 (username) 及密码 (password) 两个属性。实际应用中可能会包含用户是否被禁用，用户信息是否过期等信息。
2. 用户权限信息用 Role 与 Permission 表示，Role 与 Permission 之间构成多对多关系。Permission 可以理解为对一个资源的操作，Role 可以简单理解为 Permission 的集合。
3. 用户信息与 Role 之间构成多对多关系。表示同一个用户可以拥有多个 Role，一个 Role 可以被多个用户所拥有。
4. 另外还涉及两个关联表user\_role和role\_permission。

如下图：



**图1-2** 用户权限模型

# 安全框架选型

近期主要了解的安全框架有如下三款：

* Apache Shiro
* Spring Security
* Ralasafe

Ralasafe由于是个人维护产品，学习成本也并不低，在此并没有列入比较。

1. 上手难易程度

Spring Security：功能太过强大以至于功能比较分散，使用起来也比较复杂，跟Spring结合的比较好。对于初学Spring Security者来说，曲线还是较大，需要深入学习其源码和框架，配置起来也需要费比较大的力气，扩展性也不是特别强。

Shiro:对于新秀Shiro来说，好评还是比较多的，使用起来比较简单，功能也足够强大，扩展性也较好。

1. API完善性

Shiro与SpringSecurity的不同在于支持权限判断与角色判断两套API，代码里既可以简单的判断hasRole("administrator")，也可以更准确的判断hasPermission("Order:Approve")。

实际上，大部分的应用都采用后者，角色只用来组织管理一堆权限的集合，而不会在最终代码(JSP, URL Filter, Java Code)里用作直接的权限判断。

综合比较，选择Apache Shiro。

# Apache Shiro项目

Apache Shiro(以下简称Shiro)是功能强大并且容易集成的Java开源权限框架，旨在简化身份验证和授权。其通过简单易用的 API 提供身份验证和授权。

使用 Shiro，能够为您的应用程序提供安全性而又无需从头编写所有代码。它能够完成认证、授权、加密、会话管理等功能。

主页：<http://shiro.apache.org/>

## Apache Shiro基础

从官网上，我们基本上可以了解到，Apache Shiro提供的服务非常明确，主要包括：

1. Authentication（认证）
2. Authorization（授权）
3. Cryptography（加密）
4. Session Management（会话管理）
5. Web支持
6. 集成

首先，她提供了 Authentication（认证）服务，也就是说，通过她可以完成身份认证，让她去判断您是否为真实的会员。

其次，她还提供了 Authorization（授权）服务，其实说白了就是“访问控制”服务，也就是让她来识别您是否可以做某件事情，毕竟不同的会员是拥有不同的权限的。

更有特色的是，她还提供了 Session Management（会话管理）服务，这个就厉害了，这并不是您熟知的 HTTP Session，而是一个独立的 Session 管理框架，不管是否为 Web 应用，都可以用这套框架。

最后（但并不是最不重要的），她还提供了 Cryptography（加密）服务，封装了许多密码学算法，有您知道的，也有您不知道的，总之琳琅满目，应有尽有。

除了以上 4 个基本服务以外，她也提供很好的系统集成方案，您可以轻松将其运用到 Web 应用中，可能这也是您最关心的。此外，还可以集成第三方框架，例如：Spring、Guice、CAS 等。

官方功能介绍如下图：



**图1-3** 用户权限模型

## Apache Shiro基础概念

### ****Shiro Subject****

**主体，代表了当前“用户”，这个用户不一定是一个具体的人，与当前应用交互的任何东西都是Subject，如网络爬虫，机器人等；即一个抽象概念。**

### ****Shiro SecurityManager****

安全管理器，即所有与安全有关的操作都会与SecurityManager交互；且它管理着所有Subject；可以看出它是Shiro的核心，它负责与后边介绍的其他组件进行交互，如果学习过SpringMVC，你可以把它看成DispatcherServlet前端控制器。

### Shiro Realm

域，Shiro从从Realm获取安全数据（如用户、角色、权限），就是说SecurityManager要验证用户身份，那么它需要从Realm获取相应的用户进行比较以确定用户身份是否合法；也需要从Realm得到用户相应的角色/权限进行验证用户是否能进行操作；可以把Realm看成DataSource，即安全数据源。通俗地讲，Realm可以理解为读取用户信息、角色及权限的DAO。

Shiro中常用的Realm有JDBCRealm、IniRealm、CASRealm等。

### Shiro SecurityUtils

SecurityUtils 对象是一个 singleton，这意味着不同的对象可以使用它来获得对当前用户的访问。一旦成功地设置了这个 SecurityManager，就可以在应用程序不同部分调用 SecurityUtils.getSubject() 来获得当前用户的信息。

### 用户令牌

用户令牌提供了一种获得凭证和用户的主体（帐户身份）的方式最基本和常用的令牌是 UsernamePasswordToken，用以指定用户的用户名和密码。

## Shiro认证

### 身份认证涉及的概念

**身份验证**，即在应用中谁能证明他就是他本人。一般提供如他们的身份ID一些标识信息来表明他就是他本人，如提供身份证，用户名/密码来证明。

在shiro中，用户需要提供principals （身份）和credentials（证明）给shiro，从而应用能验证用户身份：

principals：身份，即主体的标识属性，可以是任何东西，如用户名、邮箱等，唯一即可。一个主体可以有多个principals，但只有一个Primary principals，一般是用户名/密码/手机号。

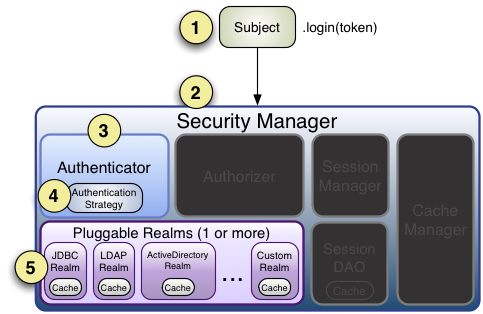
credentials：证明/凭证，即只有主体知道的安全值，如密码/数字证书等。

最常见的principals和credentials组合就是用户名/密码了。接下来先进行一个基本的身份认证。

 另外两个相关的概念是之前提到的Subject及Realm，分别是主体及验证主体的数据源。

### 身份认证流程

## 身份认证流程



流程如下：

1. 首先调用Subject.login(token)进行登录，其会自动委托给Security Manager，调用之前必须通过SecurityUtils. setSecurityManager()设置；
2. SecurityManager负责真正的身份验证逻辑；它会委托给Authenticator进行身份验证；
3. Authenticator才是真正的身份验证者，Shiro API中核心的身份认证入口点，此处可以自定义插入自己的实现；
4. Authenticator可能会委托给相应的AuthenticationStrategy进行多Realm身份验证，默认ModularRealmAuthenticator会调用AuthenticationStrategy进行多Realm身份验证；
5. Authenticator会把相应的token传入Realm，从Realm获取身份验证信息，如果没有返回/抛出异常表示身份验证失败了。此处可以配置多个Realm，将按照相应的顺序及策略进行访问。

## Shiro授权

### Shiro所涉及的概念

授权，也叫访问控制，即在应用中控制谁能访问哪些资源（如访问页面/编辑数据/页面操作等）。在授权中需了解的几个关键对象：主体（Subject）、资源（Resource）、权限（Permission）、角色（Role）。

**主体**

主体，即访问应用的用户，在Shiro中使用Subject代表该用户。用户只有授权后才允许访问相应的资源。

**资源**

在应用中用户可以访问的任何东西，比如访问JSP页面、查看/编辑某些数据、访问某个业务方法、打印文本等等都是资源。用户只要授权后才能访问。

**权限**

安全策略中的原子授权单位，通过权限我们可以表示在应用中用户有没有操作某个资源的权力。即权限表示在应用中用户能不能访问某个资源，如：

* 访问用户列表页面
* 查看/新增/修改/删除用户数据（即很多时候都是CRUD（增查改删）式权限控制）

如上可以看出，权限代表了用户有没有操作某个资源的权利，即反映在某个资源上的操作允不允许，不反映谁去执行这个操作。所以后续还需要把权限赋予给用户，即定义哪个用户允许在某个资源上做什么操作（权限），Shiro不会去做这件事情，而是由实现人员提供。

 Shiro支持粗粒度权限（如用户模块的所有权限）和细粒度权限（操作某个用户的权限，即实例级别的），后续部分介绍。

**角色**

角色代表了操作集合，可以理解为权限的集合，一般情况下我们会赋予用户角色而不是权限，即这样用户可以拥有一组权限，赋予权限时比较方便。典型的如：项目经理、技术总监、CTO、开发工程师等都是角色，不同的角色拥有一组不同的权限。

* **隐式角色**：即直接通过角色来验证用户有没有操作权限，如在应用中CTO、技术总监、开发工程师可以使用打印机，假设某天不允许开发工程师使用打印机，此时需要从应用中删除相应代码；再如在应用中CTO、技术总监可以查看用户、查看权限；突然有一天不允许技术总监查看用户、查看权限了，需要在相关代码中把技术总监角色从判断逻辑中删除掉；即粒度是以角色为单位进行访问控制的，粒度较粗；如果进行修改可能造成多处代码修改。
* **显示角色**：在程序中通过权限控制谁能访问某个资源，角色聚合一组权限集合；这样假设哪个角色不能访问某个资源，只需要从角色代表的权限集合中移除即可；无须修改多处代码；即粒度是以资源/实例为单位的；粒度较细。

### 授权方式

除了够简单，Shiro支持权限判断与角色判断两套API。

代码里既可以简单的判断hasRole("administrator")，也可以更准确的判断hasPermission("Order:Approve")。

1. 编程式
2. 注解式
3. JSPTag Library

## 缓存

Apache Shiro支持使用缓存以提高效率。

# Apche Shiro实践

## Hello,Shiro

## 在 Web 应用中使用 Shiro

## Spring 与 Shiro 整合

# 实现思路与步骤

1. 权限表结构设计
2. 权限模型设计
3. 相关的Dao、Service实现
4. Shiro相关的认证及权限接口实现
5. 提供外部调用接口
6. 整合加密与缓存
7. 利用Spring等IOC进行配置，对类之间的的强耦合进行解耦
8. 密码重试次数限制

如在1个小时内密码最多重试5次，如果尝试次数超过5次就锁定1小时，1小时后可再次重试，如果还是重试失败，可以锁定如1天，以此类推，防止密码被暴力破解。我们通过继承HashedCredentialsMatcher，且使用Ehcache记录重试次数和超时时间。

1. SSO

# 结论

通过最近2天对Shiro的学习及Demo验证。Shiro能满足当前3D平台的认证与授权需求，同时对今后的安全需求具有一定的扩展性。由于研究时间有限，相关的章节有待后续添加。