*项目名称*

建设方案

2022年7 月

目 录

[第1章 项目综述 1](#_Toc25655)

[1.1 项目背景 1](#_Toc14196)

[1.2 现状分析 1](#_Toc5279)

[1.3 建设目标 1](#_Toc16811)

[1.4 建设原则 1](#_Toc20814)

[第2章 总体设计 2](#_Toc28869)

[2.1 业务架构 2](#_Toc17860)

[2.2 应用架构 2](#_Toc21514)

[2.3 技术架构 2](#_Toc7354)

[2.4 安全架构 2](#_Toc27407)

[第3章 其它 3](#_Toc15893)

第1章 项目综述

1.1 项目背景

问卷调查是指，通过制定详细周密的问卷，要求被调查者据此进行回答以收集资料的方法。电子或网络问卷调查是在计算机技术和互联网背景之下发展起来的，目前较为多用的一种问卷调查方式。网络问卷调查突破了地域的限制，可以实现跨地区，甚至跨国家的数据收集；另外，网络问卷调查相比纸介质问卷调查成本大大降低，既节省了纸张和印刷等实体资源，更节省了人力成本，在后期数据的整理和分析时，也大幅提高了处理效率。目前国内已有可以提供网络问卷调查的平台，如答题吧、问卷星和答题派等。

1.2 现状分析

截止项目迭代二结束，项目已经实现了迭代二点全部功能，本文档拟对迭代三中的shiro部分进行研究和实现。

1.3 建设目标

项目拟用Apache Shiro进行：

验证用户来核实他们的身份

对用户执行访问控制，如：

判断用户是否被分配了一个确定的安全角色；

判断用户是否被允许做某事；

在任何环境下使用Session API，即使没有Web或EJB容器。

在身份验证，访问控制期间或在会话的生命周期，对事件作出反应。

聚集一个或多个用户安全数据的数据源，并作为一个单一的复合用户“视图”。

启用单点登录（SSO）功能。

Shiro 视图在所有应用程序环境下实现这些目标——从最简单的命令行应用程序到最大的企业应用，不强制依赖其他第三方框架，容器，或应用服务器。当然，该项目的目标是尽可能地融入到这些环境，但它能够在任何环境下立即可用。

1.4 建设原则

（1）实用性原则

坚持实用性原则，在实用的基础上考虑先进性和前瞻性，选用符合标准的，先进成熟的产品和开发平台。问卷系统的首要目标就是实现需求中的功能，达到实用性。

（2）标准化原则

通过统一的应用功能规划，统一设计开发，实现计量现场全业务的标准化。遵循一系列标准化原则。

（3）可靠性原则

保障系统7×24小时不间断、可靠运行，配备完善的可靠性措施设计，保证系统运行的高度可靠运行，使用shiro后需要提高系统的可靠性。

（4）安全性原则

注重权限管理、安全隔离、信息加密等措施，构建通信层、网络层、系统层、应用层等多层面的立体安全防护体系，确保系统安全，尤其是控制功能的应用安全。使用shiro技术将对用户登录等一系列功能实现安全保护、身份核对、权限判断等功能，优化系统安全性。

第2章 总体设计

1

2.1 业务架构

项目在用户登录和身份验证的业务上使用shiro架构。

2.2 应用架构

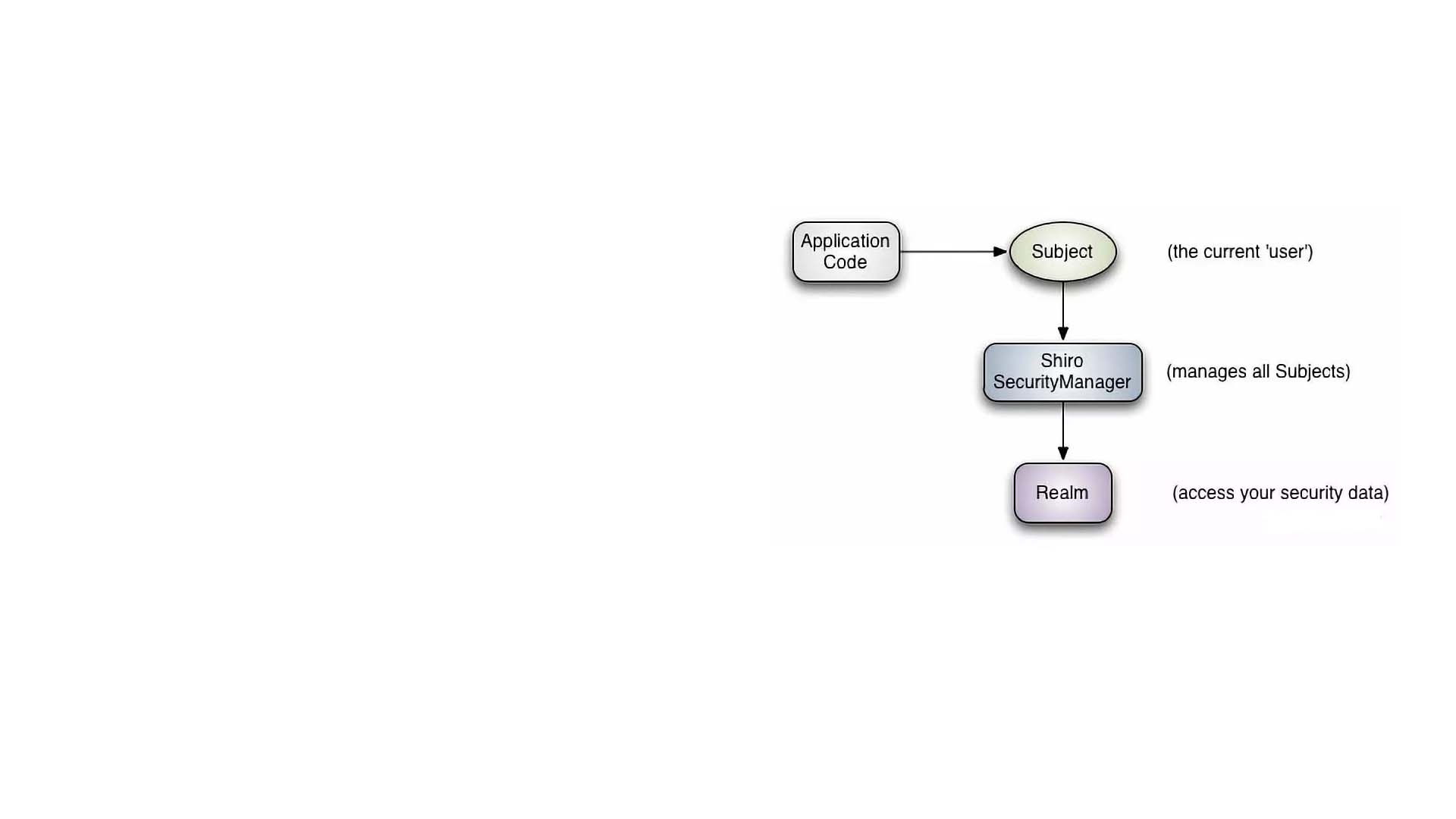
Apache Shiro的设计目标是通过直观和易于使用来简化应用程序安全。Shiro 的核心设计体现了大多数人们是如何考虑应用程序安全的——在某些人（或某些事）与应用程序交互的背景下。

应用软件通常是基于用户背景情况设计的。也就是说，你将经常设计用户接口或服务API，基于一个用户将要（或应该）如何与该软件交互。

应用程序很大程度上的编写是为了满足用户的要求和需要。即使该“用户”是另一个软件系统而不是一个人类，你仍然得编写代码来响应行为，基于当前与你的软件进行交互的人或物。

Shiro在它自己的设计中体现了这些概念。通过匹配那些对于软件开发人员来说已经很直观的东西，Apache Shiro几乎在任何应用程序保持了直观和易用性。

在最高的概念层次，Shiro的架构有3个主要的概念：Subject，SecurityManager 和 Realms。



Subject： Subject实质上是一个当前执行用户的特定的安全“视图”。鉴于"User"一词通常意味着一个人，而一个Subject可以是一个人，但它还可以代表第三方服务。所有Subject实例都被绑定到（且这是必须的）一个SecurityManager上。当你与一个Subject交互时，那些交互作用转化为与SecurityManager交互的特定subject的交互作用。

SecurityManager：SecurityManager是Shiro架构的心脏，并作为一种“保护伞”对象来协调内部的安全组件共同构成一个对象图。然而，一旦SecurityManager和它的内置对象图已经配置给一个应用程序，那么它单独留下来，且应用程序开发人员几乎使用他们所有的时间来处理Subject API。

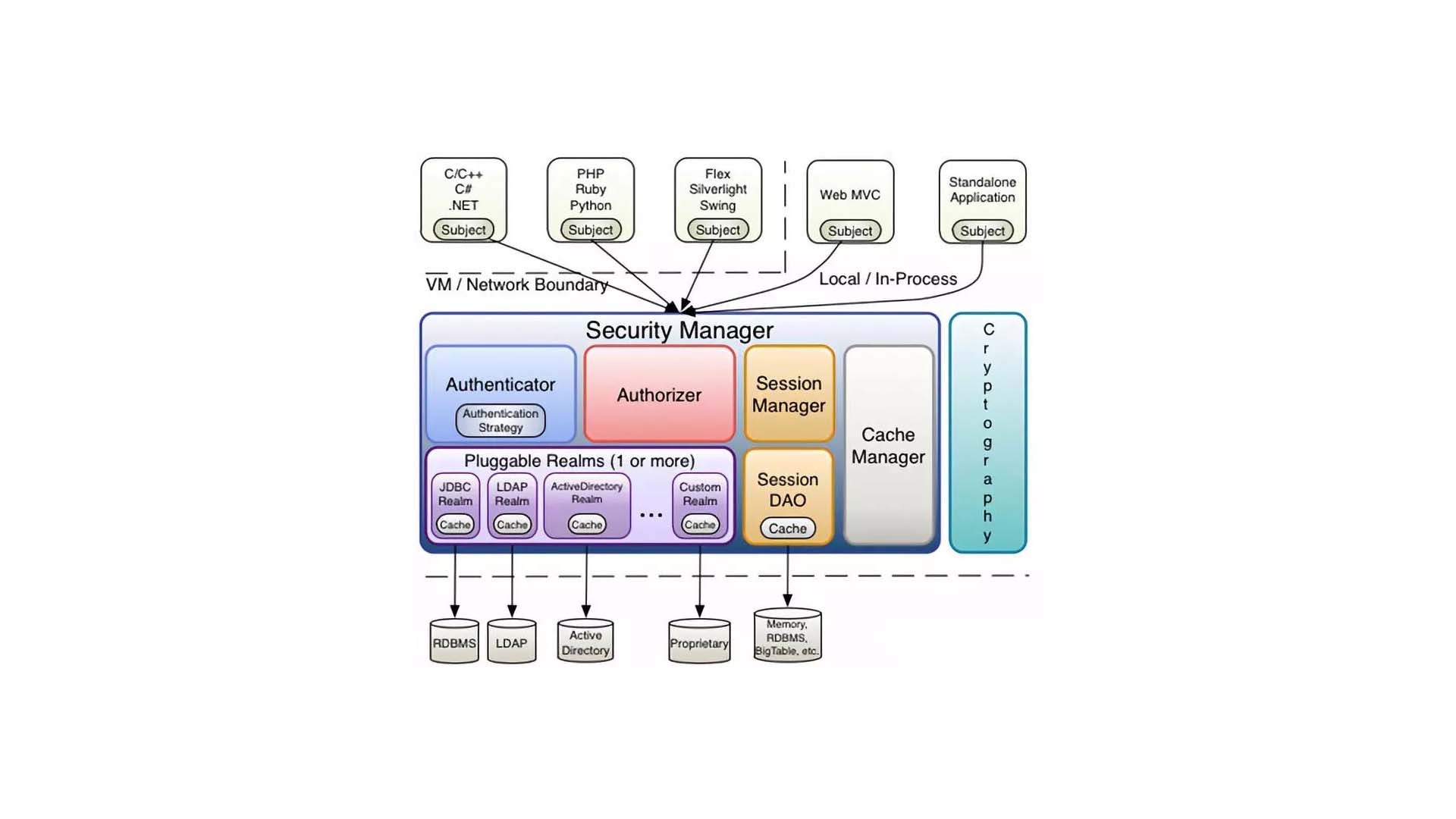
Realms：Realms担当Shiro和你的应用程序的安全数据之间的“桥梁”或“连接器”。当它实际上与安全相关的数据如用来执行身份验证（登录）及授权（访问控制）的用户帐户交互时，Shiro 从一个或多个为应用程序配置的Realm中寻找许多这样的东西。

Shiro提供了立即可用的Realms来连接一些安全数据源（即目录），如LDAP，关系数据库（JDBC），文本配置源，像 INI 及属性文件，以及更多。你可以插入你自己的Realm 实现来代表自定义的数据源，如果默认地Realm不符合你的需求。

像其他内置组件一样，Shiro SecurityManager控制 Realms是如何被用来获取安全和身份数据来代表 Subject 实例的。

2.3 技术架构

下图展示了Shiro的核心架构概念，紧跟其后的是每个的简短总结：



Subject(org.apache.shiro.subject.Subject)

当前与软件进行交互的实体（用户，第三方服务，cron job，等等）的安全特定“视图”。

SecurityManager(org.apache.shiro.mgt.SecurityManager)

如上所述，SecurityManager是Shiro架构的心脏。它基本上是一个“保护伞”对象，协调其管理的组件以确保它们能够一起顺利的工作。它还管理每个应用程序用户的Shiro 的视图，因此它知道如何执行每个用户的安全操作。

Authenticator(org.apache.shiro.authc.Authenticator)

Authenticator是一个对执行及对用户的身份验证（登录）尝试负责的组件。当一个用户尝试登录时，该逻辑被 Authenticator执行。Authenticator知道如何与一个或多个Realm协调来存储相关的用户/帐户信息。从这些Realm中获得的数据被用来验证用户的身份来保证用户确实是他们所说的他们是谁。

Authenticatio：Strateg 如果不止一个Realm被配置，则AuthenticationStrategy将会协调这些Realm来决定身份认证尝试成功或失败下的条件（例如，如果一个Realm成功，而其他的均失败，是否该尝试成功？是否所有的Realm必须成功？或只有第一个成功即可？）。

Authorizer(org.apache.shiro.authz.Authorizer)

Authorizer是负责在应用程序中决定用户的访问控制的组件。它是一种最终判定用户是否被允许做某事的机制。与 Authenticator相似，Authorizer也知道如何协调多个后台数据源来访问角色恶化权限信息。Authorizer使用该信息来准确地决定用户是否被允许执行给定的动作。

SessionManager(org.apache.shiro.session.SessionManager)

SessionManager知道如何去创建及管理用户Session生命周期来为所有环境下的用户提供一个强健的Session体验。这在安全框架界是一个独有的特色——Shiro拥有能够在任何环境下本地化管理用户Session的能力，即使没有可用的Web/Servlet或EJB容器，它将会使用它内置的企业级会话管理来提供同样的编程体验。SessionDAO的存在允许任何数据源能够在持久会话中使用。

SessionDAO(org.apache.shiro.session.mgt.eis.SessionDAO)

SesssionDAO代表SessionManager执行Session持久化（CRUD）操作。这允许任何数据存储被插入到会话管理的基础之中。

CacheManager(org.apahce.shiro.cache.CacheManager)

CacheManager创建并管理其他Shiro组件使用的Cache实例生命周期。因为Shiro能够访问许多后台数据源，由于身份验证，授权和会话管理，缓存在框架中一直是一流的架构功能，用来在同时使用这些数据源时提高性能。任何现代开源和/或企业的缓存产品能够被插入到Shiro来提供一个快速及高效的用户体验。

Cryptography(org.apache.shiro.crypto.\*)

Cryptography是对企业安全框架的一个很自然的补充。Shiro的crypto包包含量易于使用和理解的cryptographic Ciphers，Hasher（又名digests）以及不同的编码器实现的代表。所有在这个包中的类都被精心地设计以易于使用和易于理解。任何使用Java的本地密码支持的人都知道它可以是一个难以驯服的具有挑战性的动物。Shiro的cryptoAPI 简化了复杂的Java机制，并使加密对于普通人也易于使用。

Realms(org.apache.shiro.realm.Realm)

如上所述，Realms在Shiro和你的应用程序的安全数据之间担当“桥梁”或“连接器”。当它实际上与安全相关的数据如用来执行身份验证（登录）及授权（访问控制）的用户帐户交互时，Shiro从一个或多个为应用程序配置的Realm中寻找许多这样的东西。你可以按你的需要配置多个Realm（通常一个数据源一个Realm），且Shiro将为身份验证和授权对它们进行必要的协调。

The SecurityManager

因为Shiro的API鼓励一个以Subject为中心的编程方式，大多数应用程序开发人员很少，如果真有，与SecurityManager直接进行交互（框架开发人员有时候会觉得它很有用）。即便如此，了解如何SecurityManager是如何工作的仍然是很重要的，尤其是在为应用程序配置一个SecurityManager的时候。

Design

如前所述，应用程序的SecurityManager执行安全操作并管理所有应用程序用户的状态。

但这是许多功能来尝试管理一个单一的组件。而且，使这些东西灵活而又可定制将会是非常困难的，如果一切都集中到一个单一的实现类。

为了简化配置并启用灵活配置/可插性，Shiro的实现都是高度模块化设计——由于如此的模块化，SecurityManager实现（以及它的类层次结构）并没有做很多事情。相反，SecurityManager 实现主要是作为一个轻量级的“容器”组件，委托计划所有的行为到嵌套/包裹的组件。这种“包装”的设计体现在上面的详细构架图。

虽然组件实际上执行逻辑，但SecurityManager实现知道何时以及如何协调组件来完成正确的行为。SecurityManager 实现和组件都是兼容JavaBean的，它允许你（或某个配置机制）通过标准的JavaBean的accessor/mutator 方法（get/set）轻松地自定义可拔插组件。这意味着 Shiro 的架构的组件性能够把自定义行为转化为非常容易的配置文件。

2.4 安全架构

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；

SecurityManager 负责真正的身份验证逻辑；它会委托给 Authenticator 进行身份验证；

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；

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；

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。

第3章 其它