深入了解 Spring Security

2016年 10月 1日

0. 前言

本文将从宏观的架构的角度带你深入了解Spring Security。读者可以通过本文了解到Spring Security的基础构建，以及其实现安全机制的基本架构等。

本文不会讲解Spring Security的使用方式，不会讲解Web安全的概念原理，也不会解释设计模式等深奥的原理，而是通过从十万八千里的高空，从宏观角度带您领略Spring Security的微妙之处。

通过阅读本文，读者可以了解到Spring Security的基础构建，以及各基础构建之间是如何互相协作的。了解了这些基础知识之后，读者可以更好地理解Spring Security的运作原理，从而使得Spring Security的使用更为方便明了，遇到问题时解决也更容易，还可以轻松地针对其进行客制化的改造。

1. 什么是 Spring Security

Spring Security，人如其名，是一款基于Spring Framework的安全框架，原名Acegi Security，最早于2003年起源于Spring社区，由那时的社区成员们尝试着使用基于Spring的bean来实现安全机制，但并未正式发布，直到2004年3月才正式在SourceForge上成立项目，当时命名为”The Acegi Security System for Spring”。

经过多年发展，Acegi Security于2007年末正式归为Spring Framework的正式子项目，并改名为Spring Security。自那以后Spring Security有了长远发展，现已成为一款基于Spring Framework的各标准通吃的安全框架。Spring Security支持的标准非常多，大家可以在Spring Security的官方文档上看到，我就不再这里赘述了。

如果你熟悉Spring Framework，那么使用Spring Security将易如反掌；如果你不熟悉Spring Framework，虽然Spring Security也可以与其他非Spring框架一起使用，但还是非常推荐你去了解一下Spring Framework的。作为Java EE的龙头老大的Spring，还是很值得一位专业的Java EE开发人员去深入了解一下的。

Spring Security主要针对安全方面的两大难题——鉴权（Authentication）和授权（Authorization，又叫访问控制[access-control]；下文将详细讲述鉴权与授权的区别）提供了灵活强大的解决方案。说灵活是在于Spring Security并不局限于Spring MVC，虽然它是基于Spring Framework实现的，但它并不依赖于Spring MVC，可以独立于MVC应用在其他Java EE框架之上。说强大是在于Spring Security的安全管制并不只限制于Web请求，除此之外它还可以针对方法调用通过AOP的方式进行安全管制，甚至可以对域对象实例（Domain Object Instance）进行访问控制。

本文将着重针对于Web请求方面进行讲解，由于Spring Security的安全管制机制是基于一个统一的抽象过程所实现的，所以，只要理解了其中一个方面即可，其他两个方面的区别只是实现细节上的问题了。

2. 基本概念

2.1. Spring Security 的基建

在了解 Spring Security 之前，你必须先了解一些基础构件，我先列在下方，稍后针对每一个构件做详细的介绍。

一开始你可能对这些东西一头雾水，不要慌张，没有关系，等后面谈到整体全局观时，你可以再来回顾看看，就能明白我在说什么了。现在，你只需记住有这些构件，并能大致记住它们的作用即可。

- `SecurityContextHolder` ，为使用者提供全局的 `SecurityContext`
- `SecurityContext` ，为了候得住 (hold) `Authentication`
- `Authentication` ，主要负责候住两方面信息，一个是当前用户的详细信息(`Principal` 、`UserDetails`)，一个是用户鉴权时需要的信息。

- `GrantedAuthority` ，提供当前用户(`UserDetails`)所获得的系统范围内的授权。
- `UserDetails` ，提供了用户的详细信息，主要被用来构建 `Authentication` 。
- `UserDetailsService` ，这个接口的实现主要是负责通过用户名查找并提供用户的详细信息(`UserDetails`)。

我们来详细介绍一下各个构件，第一次阅读可以跳过2.1.1 - 2.1.6节。

2.1.1. `SecurityContextHolder`

`SecurityContextHolder` 为用户提供了统一的 `SecurityContext` 获取方法，通过这个类的 `getContext()` 静态方法即可获取当前线程上的 `SecurityContext` ，不需要将其作为参数到处传递。

`SecurityContextHolder` 是通过 `ThreadLocal` 将 `SecurityContext` 与线程绑定的。不用担心线程间串 `SecurityContext` ，Spring Security会负责在当前线程完成时将 `SecurityContext` 移除。

针对某些不适合将 `SecurityContext` 与线程绑定的程序，譬如Swing App等， `SecurityContextHolder` 提供了其他的绑定策略，如 `SecurityContextHolder.MODE_GLOBAL` 、 `SecurityContextHolder.MODE_INHERITABLETHREADLOCAL` 等。

哦对了，顺便说一句， `SecurityContextHolder.getContext()` 保证不会返回 `null` 值，所以，不用担心 `NullPointException` 。

2.1.2. `SecurityContext`

从字面理解 `SecurityContext` 的话，应该是（当前环境下的）安全上下文。简单猜测一下就可以知道当前环境下的安全上下文应该要包含点啥了，譬如（包括但不限于）当前用户的详细信息等。但当前用户的详细信息这种这么细节的东西，被Spring Security包在了 `Authentication` 对象中了，所以，这个对象的主要作用就是被用来获取当前环境下的 `Authentication` 对象。

注意，如果当前用户未鉴权（这里不是指匿名用户，未鉴权和匿名用户是两个概念，下文详细讨论），你将会拿到 `null` 。

2.1.3. `Authentication`

鉴权对象，该对象主要包含了用户的详细信息（ `UserDetails` ）和用户鉴权时所需要的信息，如用户提交的用户名密码、Remember-me Token，或者digest hash值等，按不同鉴权方式使用不同的 `Authentication` 实现。

该对象可能会包含敏感信息（如明文密码），但不用担心，如果你使用Spring Security提供的默认实现，它已经考虑到了这方面问题，在用户鉴权完成后敏感信息会被即刻删除。

除了上面提到的两方面信息， `Authentication` 还提供了一个列表的 `GrantedAuthority` 来表示用户所拥有的权限。

2.1.4. `GrantedAuthority`

该对象表示了当前用户所拥有的权限（或者角色）信息。这些信息由授权负责对象 `AccessDecisionManager` 来使用，并决定最终用户是否可以访问某资源（URL或方法调用或域对象）。鉴权时并不会使用到该对象。

`GrantedAuthority` 默认权限（或角色）是基于字符串的（String-based），但你也可以实现非字符串的复杂权限（或角色）（”complex” `GrantedAuthority` ），但这需要被 `AccessDecisionManager` 的实现所支持。

一般 `GrantedAuthority` 由用户拥有，我们一般会说一个用户有什么什么样的权限或角色。在Spring Security里面，用户的详细信息由 `UserDetails` 接口来规范。

2.1.5. `UserDetails`

这个接口规范了用户详细信息所拥有的字段，譬如用户名、密码、账号是否过期、是否锁定等。在Spring Security中，获取当前登录的用户的信息，通常是通过 `SecurityContext` -> `Authentication` -> `UserDetails` 这样一个过程过来的。

这个接口还规定了实现类必须提供用户的密码（明文或哈希值都行），是因为在用户名/密码鉴权过程中需要比较

登录用户提交的密码和数据库中的密码是否一致。然而Spring Security并没有规定用户在数据库中保存的密码必须是明文还是通过什么方式哈希的，所以两者皆可。如果你选择将用户密码哈希后保存在数据库中，则针对密码的哈希方式是由 `PasswordEncoder` 的实现来决定的。

Spring Security没有规定 `UserDetails` 应该以什么方式保存在持久层，也没有规定持久层应该是RMDBS还是NoSQL，所以 `UserDetails` 的持久层可以是MySql，也可以是MongoDB、Redis，甚至直接放Properties文件或内存里也行（In-Memory），只要你能够通过实现 `UserDetailsService` 来提供 `UserDetails` 对象即可。

2.1.6. `UserDetailsService`

`UserDetailsService` 只规定了一个 `loadUserByUsername` 方法，这个方法在用户鉴权时非常重要，譬如当登录用户提交用户名密码过来时，Spring Security正是通过这个方法取得登录用户提交的用户名所对应的用户详细信息包括密码的。但必须注意的是，用户密码的比较并不是在 `UserDetailsService` 中进行的，而是由 `AuthenticationManager` 以及 `AuthenticationProvider` 负责的。

特别需要注意的是，当无法找到用户名对应的用户时，必须抛出 `UsernameNotFoundException` 而不是返回 `null` 。

由这样一个抽象的接口来规范用户详细信息的读取过程的好处是，Spring Security并不需要知道你保存 `UserDetails` 的持久层是什么样的，Spring Security与持久层完全解耦，因此你保存 `UserDetails` 的持久层可以是任何东西，RMDBS也行NoSQL也行，甚至Properties文件或直接放内存里也可以（In-Memory）。

Spring Security提供了两个比较常用的实现，一个是In-Memory的实现，主要负责从Properties文件中读取用户信息并保存在内存中；另外一个是 `JdbcDaoImpl` ，负责从数据库中读取用户信息。In-Memory的方式通常适合在测试时使用，而 `JdbcDaoImpl` 因为规定了数据库的schema，所以一般不会直接使用，而是当做实现参考来使用。

如果你使用JPA，则需要自己实现 `UserDetailsService` 。

2.2. 区别 `Authentication` （鉴权） 和 `Authorization` （授权）

在继续了解Spring Security之前，我觉得很有必要先来了解一下鉴权（ `Authentication` ）和授权（ `Authorization` ）的区别。

简而言之，鉴权就是鉴定用户“是谁”，而授权则是鉴定用户“是否可以”做这件事情或访问某个URL。当然授权的前提是明确知道当前的用户“是谁”，所以一般情况下鉴权发生在授权之前。

在Spring Security中，匿名用户会被鉴定为 `anonymousUser` ，并且拥有 `ROLE_ANONYMOUS` 角色，所以，当未登录用户访问一个在Spring Security管辖范围内，但被配置成 `permitAll` 的URL时，如果你通过 `SecurityContextHolder.getContext().getAuthentication()` 方法获取当前用户，你将会得到一个匿名用户的对象。这个时候访问 `Authentication.getName()` 将会得到 `anonymousUser` ，而 `Authentication.getAuthorities()` 将会返回一个只有 `ROLE_ANONYMOUS` 单个元素的列表。这是匿名用户的场景。

但是，如果用户访问一个不在Spring Security管辖范围内的URL时，如果你同样通过 `SecurityContextHolder.getContext().getAuthentication()` 方法获取 `Authentication` 对象时，你将会得到一个 `null` ，这是前文所说的未鉴权用户的场景。

那么什么叫Spring Security管辖范围？

因为Spring Security是通过一系列 `javax.servlet.Filter` 来实现的，所以在配置Spring Security时，如果将URL mapping到Spring Security的 `Filter` 的 `filter-mapping` 之外的的话，就属于Spring Security的管辖范围之外了。Spring Security对这些URL鞭长莫及。通常，一些常用的配置如 `static` 文件夹（通常包括css、js、html等静态文件）、 `swagger-ui.html` 、 `webjars` 、 `h2-console` 等都会配置到Spring Security的管辖之外。

在Spring Security中，鉴权是通过 `Authentication` 对象和 `AuthenticationManager` 以及 `AuthenticationProvider` 互相合作来实现的。而授权则是通过 `GrantedAuthor`

ity 、 AccessDecisionManager 和 AccessDecisionVoter 等互相合作来实现的。

在接下来的文章中，我们将通过参考一个抽象鉴权过程和一个现实场景来介绍Spring Security的鉴权过程，而授权以及针对Spring Security各 Filter 的介绍将会在以后的文章中详细介绍。

微信打赏

若本文对您有帮助，欢迎打赏，感谢您的鼓励。

相关文章

再谈相等性（Equality）和同一性（Identity） 2019年 1月 19日

BAT lock-in 2019年 1月 4日

泛型的协变、逆变、不变 2018年 12月 3日