



原创 ITKaven 于 2022-01-04 22:55:06 发布 阅读量1.5k 收藏4 点赞数3

分类专栏：Spring Security 文章标签：spring java redis



Spring Security 专栏收录该内容

31 订阅 12 篇文章

在上一篇博主中，博主介绍了 Spring Security 的 UserDetails 接口及其实现，Spring Security 使用 UserDetails 实例（实现类的实例）表示用户，行验证时（提供用户名和密码），Spring Security 会通过用户服务（UserDetailsService 接口及其实现）来获取对应的 UserDetails 实例（相同的）。如果该 UserDetails 实例存在并且与客户端输入的信息匹配，则验证成功，否则验证失败，想了解 UserDetails 接口及其实现可以看下面这篇博客：

- Spring Security：用户UserDetails源码与Debug分析

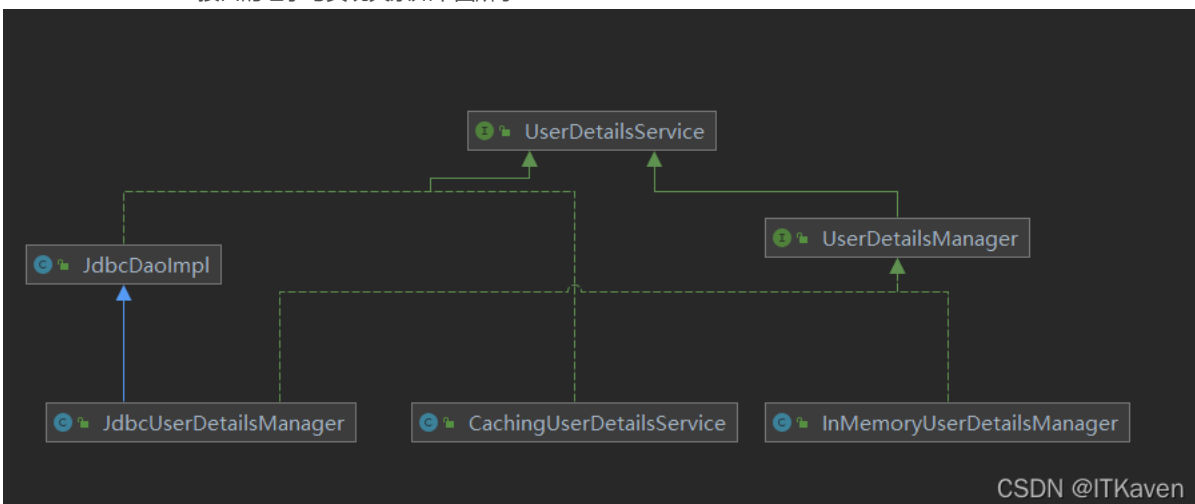
UserDetailsService

UserDetailsService 接口源码：

```
1 package org.springframework.security.core.userdetails;
2
3 /**
4  * 加载用户特定数据的核心接口
5  * 它在整个框架中作为用户的DAO层（用户数据访问层）
6  */
7 public interface UserDetailsService {
8     /**
9      * 根据用户名定位用户
10     */
11     UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;
12 }
```

UserDetailsService 接口只定义了一个方法，即通过用户名查找 UserDetails 实例的方法，因此子类可以有各种各样的实现，可以基于 JVM 的堆内存 ConcurrentHashMap 存储 UserDetails 实例，或者基于中间件（比如 Mysql、Redis），或者两者的混合模式，可以根据需求来自定义实现。

UserDetailsService 接口的继承与实现关系如下图所示：



CSDN @ITKaven

UserDetailsManager

UserDetailsManager 接口源码（继承 UserDetailsService 接口）：

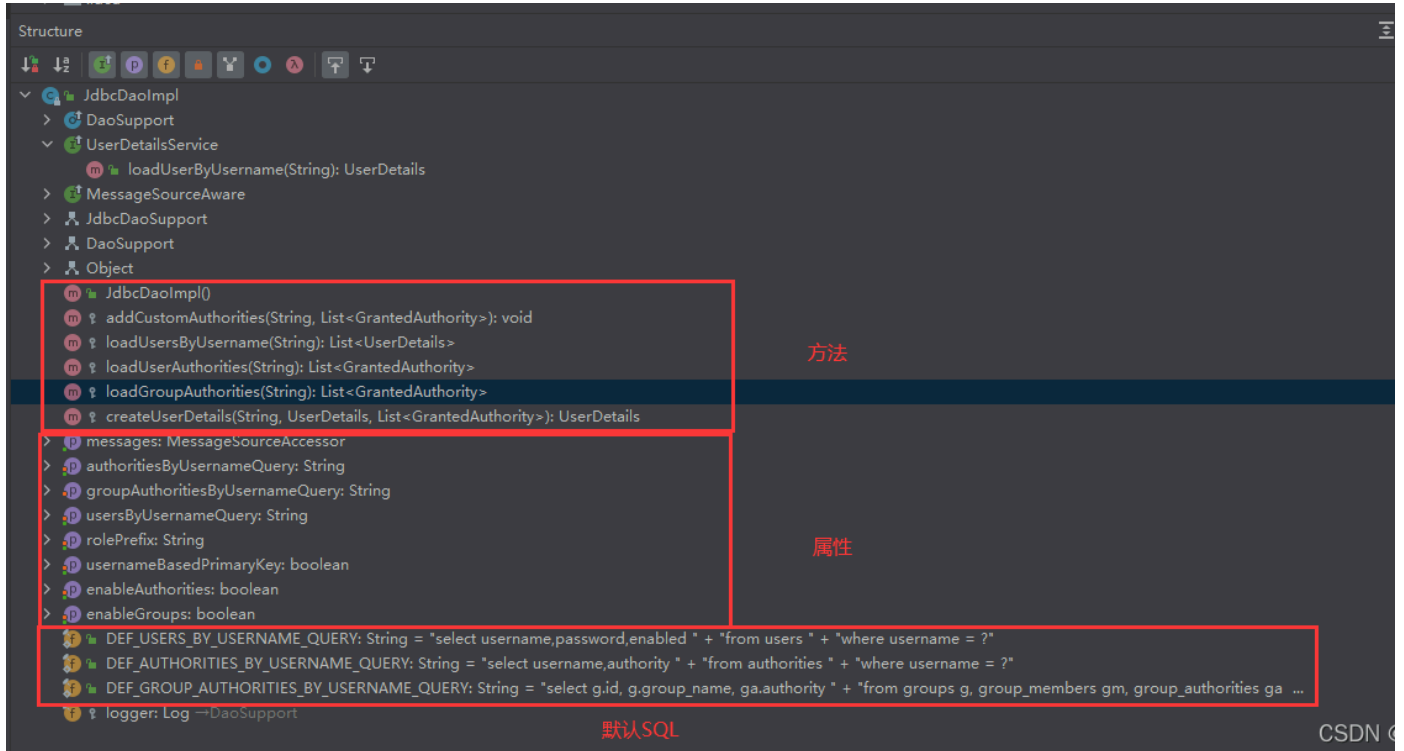
```
1 package org.springframework.security.provisioning;
2
3 import org.springframework.security.core.userdetails.UserDetails;
4 import org.springframework.security.core.userdetails.UserDetailsService;
5
6 /**
7  * UserDetailsService的扩展，提供创建新用户和更新现有用户的能力
8  */
9 public interface UserDetailsManager extends UserDetailsService {
10
11 }
```



很显然 `UserDetailsManager` 接口是 `UserDetailsService` 接口的扩展，提供了创建新用户和更新现有用户的能力。

JdbcDaoImpl

`JdbcDaoImpl` 类的结构如下图所示：



使用 `JDBC` 从数据库中检索用户详细信息（用户名、密码、启用标志和权限）。假设有一个默认的数据库模式（有 `users` 和 `authorities` 两个表）。

```
1 create table users(  
2     username varchar_ignorecase(50) not null primary key,  
3     password varchar_ignorecase(500) not null,  
4     enabled boolean not null  
5 );  
6  
7 create table authorities (  
8     username varchar_ignorecase(50) not null,  
9     authority varchar_ignorecase(50) not null,  
10    constraint fk_authorities_users foreign key(username) references users(username)  
11 );  
12 create unique index ix_auth_username on authorities (username,authority);
```

如果数据库模式和默认不一样，可以设置 `usersByUsernameQuery` 和 `authorityByUsernameQuery` 属性以匹配数据库设置，不然它们的值都是默认 `SQL`。

```
1 public JdbcDaoImpl() {  
2     this.usersByUsernameQuery = DEF_USERS_BY_USERNAME_QUERY;  
3     this.authoritiesByUsernameQuery = DEF_AUTHORITIES_BY_USERNAME_QUERY;  
4     this.groupAuthoritiesByUsernameQuery = DEF_GROUP_AUTHORITIES_BY_USERNAME_QUERY;  
5 }
```

可以通过将 `enableGroups` 属性设置为 `true` 来启用对组权限的支持（还可以将 `enableAuthorities` 设置为 `false` 以直接禁用权限加载）。通过这种方法分配给组，并且用户的权限是根据他们所属的组来确定的。最终结果是相同的（加载了包含一组 `GrantedAuthority` 的 `UserDetails` 实例）。使用组时 `groups`、`group_members` 和 `group_authorities`。

```
1 create table groups (  
2     id bigint generated by default as identity(start with 0) primary key,  
3     group_name varchar_ignorecase(50) not null  
4 );  
5  
6
```



```
9      authority varchar(50) not null,  
10      constraint fk_group_authorities_group foreign key(group_id) references groups(id)  
11  );
```

关于加载组权限的默认查询，可以参考 `DEF_GROUP_AUTHORITIES_BY_USERNAME_QUERY` 。同样，可以通过设置 `groupAuthoritiesByUsernameQuery` 属性来

`JdbcDaoImpl` 类源码（实现了 `UserDetailsService` 接口，提供了使用 `JDBC` 获取用户数据的基本实现，下面代码删除了上面已经提到的内容）：

```
1 public class JdbcDaoImpl extends JdbcDaoSupport  
2     implements UserDetailsService, MessageSourceAware {  
3  
4     /**  
5      * 允许子类将他们自己授予的权限添加到UserDetail实例的权限列表中  
6      */  
7     protected void addCustomAuthorities(String username,  
8         List<GrantedAuthority> authorities) {  
9     }  
10  
11     // 通过用户名加载用户的核心逻辑 通过使用其他方法来完成
```

`JdbcDaoImpl` 类提供了使用 `JDBC` 获取用户数据的基本实现。

JdbcUserDetailsManager

`JdbcUserDetailsManager` 类继承了 `JdbcDaoImpl` 类（提供了使用 `JDBC` 获取用户数据的基本实现），并且实现了 `UserDetailsService` 和 `GroupManager` 接口。`UserDetailsService` 接口提供了创建新用户和更新现有用户的能力。`GroupManager` 接口允许管理组权限及其成员，通常用于在以下情况下补充 `UserDetailsService` 的功能：

- 将应用程序授予的权限组织到组中，而不是直接将用户与角色进行映射。
- 在这种情况下，用户被分配到组并获得分配组的权限列表，从而提供更灵活的管理选项。

新增的默认 SQL（`UserDetailsManager SQL` 和 `GroupManager SQL`）：

```
1 // UserDetailsManager SQL  
2 public static final String DEF_CREATE_USER_SQL = "insert into users (username, password, enabled) values (?, ?, ?)";  
3 public static final String DEF_DELETE_USER_SQL = "delete from users where username = ?";  
4 public static final String DEF_UPDATE_USER_SQL = "update users set password = ?, enabled = ? where username = ?";  
5 public static final String DEF_INSERT_AUTHORITY_SQL = "insert into authorities (username, authority) values (?, ?)";  
6 public static final String DEF_DELETE_USER_AUTHORITIES_SQL = "delete from authorities where username = ?";  
7 public static final String DEF_USER_EXISTS_SQL = "select username from users where username = ?";  
8 public static final String DEF_CHANGE_PASSWORD_SQL = "update users set password = ? where username = ?";  
9  
10 // GroupManager SQL  
11 public static final String DEF_FIND_GROUPS_SQL = "select group name from groups";
```

源码就不贴了，太多了，实现方式和 `JdbcDaoImpl` 类差不多（通过使用默认 SQL，也可以使用满足要求的自定义 SQL，查询用户相关数据），需要使用读源码（还是要多看源码）。

CachingUserDetailsService

`CachingUserDetailsService` 类源码（实现了 `UserDetailsService` 接口）：

```
1 public class CachingUserDetailsService implements UserDetailsService {  
2     // 用户缓存，NullUserCache不执行任何缓存
```



```
5     private final UserDetailsService delegate;
6
7     public CachingUserDetailsService(UserDetailsService delegate) {
8         this.delegate = delegate;
9     }
10
11     public UserCache getUserCache() {
```



`NullUserCache` 类，不执行任何缓存。

```
1 public class NullUserCache implements UserCache {
2
3     public UserDetails getUserFromCache(String username) {
4         return null;
5     }
6
7     public void putUserInCache(UserDetails user) {
8     }
9
10    public void removeUserFromCache(String username) {
11    }
12 }
```

`CachingUserDetailsService` 实例可以通过设置不同的用户缓存（以后介绍）实例来达到不同的缓存效果。

```
1 // 设置用户缓存实例
2 public void setUserCache(UserCache userCache) {
3     this.userCache = userCache;
4 }
```

InMemoryUserDetailsService

`InMemoryUserDetailsService` 通过 `HashMap` 存储用户数据，是 `UserDetailsService` 的一种非持久化实现。主要用于测试和演示目的，不需要完整的持

`InMemoryUserDetailsService` 类源码（实现了 `UserDetailsService` 和 `UserDetailsService` 接口，`UserDetailsService` 接口定义了 `UserDetails` 密码的方法）

```
1 public class InMemoryUserDetailsService implements UserDetailsService,
2     UserDetailsPasswordService {
3     protected final Log logger = LoggerFactory.getLog(getClass());
4
5     // 存储用户数据的容器
6     private final Map<String, MutableUserDetails> users = new HashMap<>();
7     // 用于处理验证请求，以后会详细介绍
8     private AuthenticationManager authenticationManager;
9
10    // 无参构造器
11    public InMemoryUserDetailsService() {
```



自定义用户服务

自定义用户服务：

```
1 @EnableWebSecurity
2 public class SecurityConfig extends WebSecurityConfigurerAdapter {
3
4     @Override
5     protected void configure(AuthenticationManagerBuilder auth) throws Exception {
6         // 配置自定义的用户服务
7     }
```



```
10 |  
11 }
```



Spring Security 的用户服务 UserDetailsService 源码分析就到这里，如果博主有说错的地方或者大家有不同的见解，欢迎大家评论补充。

文章知识点与官方知识档案匹配，可进一步学习相关知识

Java技能树 首页 概览 149725 人正在系统学习中

AI无限·码力全开，立即参与通义灵码开源大赛，赢灵码限量周边！

>>立即参与

3 条评论



何壹时 热评 好文三连了！欢迎回访！回应互评五星！ [网页链接](#)

...UserDetailsService接口和UserDetailsService接口(原理、流程)

文章浏览阅读3.3k次,点赞7次,收藏10次。接下来自己定义一个用户类SecurityUser类,也去实现UserDetailsService接口,重写UserDetailsService接口方法时,直接写死一个用户信息,一会儿

SpringSecurity-5-UserDetailsService与UserDetailsService接口(用户来源)-CS...

这个UserDetailsService接口是用户数据的来源,就是登录用户的数据从哪里来,是从数据库来?还是从你电脑的TXT文本上过来的,还是从其他地方,该接口会返回UserDetailsService

spring security 项目配置源码

- 通过源码分析，了解Spring Security的拦截器如何工作，以及如何自定义安全规则。 - 查看`SecurityConfig`类，这是Spring Security的核心配置，它扩展了`WebSecurity

详解Spring Security进阶身份认证之UserDetailsService(附源码)

weixin_33738555的时

在上一篇Spring Security身份认证博文中，我们采用了配置文件的方式从数据库中读取用户进行登录。虽然该方式的灵活性相较于静态账号密码的方式灵活了许多，但是

Spring Security(五)--管理用户_spring security userdetails...

UserDetailsService则在其基础上添加了对用户添加、修改、删除的行为,这是大多数应用程序中必需的功能,虽然我们完全也可以自己定义,UserDetailsService接口扩展

...显式声明用户名和使用UserDetailsService(二)_userdetailsservice注 ...

SpringSecurity-SpringBoot-显式声明用户名和使用UserDetailsService(二) 在上一节中,我们实现了SpringSecurity和SpringBoot的初步整合,没有进行任何配置就实现了一个

【Spring】Spring Security 核心类介绍及Spring Security 的验证机制 最新发布

No8g攻城狮的时

Authentication 用来表示用户认证信息，在用户登录认证之前，Spring Security 会将相关信息封装为一个 Authentication 具体实现类的对象，在登录认证成功之后又会生成

SpringSecurity之UserDetailsService详解

风非

当什么也没有配置的时候，账号和密码是由 定义生成的。而在实际项目中账号和密码都是从数据 库中查询出来的。所以我们要通过自定义逻辑控制认证逻辑。如果需要自

Spring Security_auth.userdetailsservice(userdetailsservice...

当自定义逻辑时,需要实现UserDetailsService接口。实现过程: 第一步:创建类继承UsernamePasswordAuthenticationFilter,重写三个方法。 第二步:创建类实现UserDetailS

【Spring Security】UserDetailsService 接口介绍_springsecurity 实现...

UserDetailsService 在Spring Security 中主要承担查询系统内用户、验证密码、封装用户信息和角色权限。大白话就是你写一个实现类实现 UserDetailsService 接口,在这

Spring Security(二) UserDetailsService 和 PasswordEncoder 密码解析器 详解

奥迪的时

一、 UserDetailsService 详解 当什么也没有配置的时候，账号和密码是由 Spring Security 定义生成的。而在实际项目中账号和密码都是从数据库中查询出来的。所以我们

Spring Security 实战干货：用户信息UserDetails相关入门

码农小

1. 前言 前一篇介绍了 Spring Security 入门的基础准备。从今天开始我们来一步步窥探它是如何工作的。我们又该如何驾驭它。请多多关注公众号： Felordcn 。本篇将通

Spring Security——03,认证_登录_引入userdetails接口后怎么获取用户...

2、自定义UserDetailsService 在这个实现类中去查询数据库 校验: 1、定义Jwt认证过滤器 获取token 解析token获取其中的userid 从redis中获取用户信息 存入SecurityC

...学习(五)——账号密码的存取(UserDetailService、PasswordEncoder、Dao...

终于来到重头戏了。现在的Java Web项目通常都是用自定义UserDetailsService的方式。这种方式需要做两件事:1、编写一个实现UserDetailsService接口的类;2、编写一个UserC

SpringBoot集成Spring Security实现角色继承【完整源码+数据库】

7. **源码分析**: `src`目录下的代码应包含关键组件,如`WebSecurityConfig`配置类、`UserDetailsService`实现、以及可能的自定义过滤器或访问决策器。通过阅读和理

spring security 3.1.3 源码含sample源码

Spring Security是Spring框架的一个核心组件,专注于提供身份验证和授权服务,用于构建安全的Java Web应用程序。3.1.3版本是它的一个较早版本,但依然包含了许多

SpringSecurity学习汇总_userdetailsservice 配置多用户

- [spring-security-oauth2源码](#)
Spring Security OAuth2允许开发者自定义各种组件，如TokenEnhancer（增强令牌）、UserDetailsService（用户详情服务）等，以满足特定需求。 6. **spring-security-o
- [Spring_Security3_源码分析](#)
总结，Spring Security 3的源码分析是一个深度学习的过程，涵盖了安全领域的多个方面。通过理解其内部工作机制，开发者可以更好地利用这一强大的框架，为应用程序
- [详解Spring Security进阶身份认证之UserDetailsService\(附源码\)-附件资源](#)
详解Spring Security进阶身份认证之UserDetailsService(附源码)-附件资源
- [【SpringSecurity】五、UserDetails接口和UserDetailsService接口（原理、流程）](#)
接下来自己定义一个用户类SecurityUser类，也去实现UserDetails接口，重写UserDetails接口方法时，直接写死一个用户信息，一会儿new这个自定义的SecurityUser类，
- [Spring Security - UserDetails](#)
UserDetails 中几个字段的解释：//返回验证用户密码,无法返回则NULLString getPassword();String getUsername();//账户是否过期,过期无法验证boolean isAccountNonEx
- [Springsecurity之UserDetails](#)
2019独角兽企业重金招聘Python工程师标准>>> ...
- [SpringSecurity \(2\) UserDetailsService 热门推荐](#)
主要介绍如何利用 UserDetailsService 接口从数据库中获取用户信息, 并通过实现 AuthenticationProvider 接口编写自己的校验逻辑, 从而完成 SpringSecurity 身份校验.
- [SpringSecurity框架原理浅谈之UserDetailsService](#)
UserDetailsService的作用就是从特定的地方（通常是数据库）加载用户信息.
- [Spring Security：用户UserDetails源码与Debug分析](#)
Spring Security身份验证与授权的对象是用户，这里说的用户可以是配置文件中定义的用户，也可以是数据库中存储的用户，还可以是Spring Security自动创建的用户（
- [Spring Security多个UserDetailsService](#)
Spring Security支持多个UserDetailsService。你可以通过在配置文件中定义多个UserDetailsService的实现类来实现这一点。在配置文件中，你可以使用<bean>标签为每

 ITKaven
码龄7年  暂无认证

542
原创

2万+
周排名

105万+
总排名

227万+
访问


等级

2万+
积分

1368
粉丝

1992
获赞

937
评论

3844
收藏



私信

关注

AI圈早知道，每日最新动态
了解全球AI新鲜事！




大额流量券免费送
发布一篇就可获得！






- MyBatis-Plus 之分页查询 167673
- 同步和异步的区别 89904
- 怎么保存退出 vim 编辑 70673
- MySQL的driverClassName、url 50048
- Spring 中 ClassPathXmlApplicationContext 类的简单使用 40177

分类专栏

- JAVA

30篇
- Spring

9篇
- Spring Boot

31篇
- Spring Security

12篇
- Spring Cloud

12篇
- Spring Cloud Alibaba

12篇

最新评论

- ElasticJob-Lite：作业监听器
一只烤鸭朝北走啊: 好久没看了，有点忘记了，我那会儿做的是定时任务在分布式 ...
- ElasticJob-Lite：作业监听器
AKKO111: 那感觉没必要用这个Distribute的listener了，直接用ElasticJobServiceLo: ...
- Nginx：Nginx添加SSL实现HTTPS访问
风度翩翩609: 我这边配完 http+域名可以，但是https+域名就不行了
- 使用Vue和Spring Boot实现文件下载
一入程序无退路: 下载按钮点第二次就不太好使怎么回事
- Docker容器中使用PING命令报错：bash:...
桂哥317: 干脆利落、实测有效

最新文章

- MySQL: 备份 & 导入备份
- RabbitMQ： Docker Compose部署RabbitMQ 集群
- VirtualBox：系统安装重启后又要重新安装

2022年 80篇	2021年 55篇
2020年 206篇	2019年 10篇
2018年 220篇	2017年 31篇



目录

- UserDetailsService
- UserDetailsManager
- JdbcDaoImpl
- JdbcUserDetailsManager
- CachingUserDetailsService
- InMemoryUserDetailsManager
- 自定义用户服务