

## 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。所以我们要通过自定义逻辑控制认证逻辑。如果需要自定义逻辑时，只需要实现 UserDetailsService 接口即可。接口定义如下：

```
public interface UserDetailsService {  
    UserDetails loadUserByUsername(String var1) throws UsernameNotFoundException;  
}
```

### 返回值

返回值 UserDetails 是一个接口，定义如下

```
public interface UserDetails extends Serializable {  
    Collection<? extends GrantedAuthority> getAuthorities(); 获取所有权限  
  
    String getPassword(); 获取密码  
  
    String getUsername(); 获取用户名  
  
    boolean isAccountNonExpired(); 是否账号过期  
  
    boolean isAccountNonLocked(); 是否账号被锁定  
  
    boolean isCredentialsNonExpired(); 凭证（密码）是否过期  
  
    boolean isEnabled(); 是否可用  
}
```

要想返回 UserDetails 的实例就只能返回接口的实现类。SpringSecurity 中提供了如下的实例。对于我们只需要使用里面的 User 类即可。注意 User 的全限定路径是：

org.springframework.security.core.userdetails.User 此处经常和系统中自己开发的 User 类弄混。



在 User 类中提供了很多方法和属性。



```
▼ User
  > UserBuilder
  > AuthorityComparator
    m User(String, String, Collection<? extends GrantedAuthority>)
    m User(String, String, boolean, boolean, boolean, boolean, Collection<? extends GrantedAuthority>)
    m getAuthorities(): Collection<GrantedAuthority> ↑UserDetails
    m getPassword(): String ↑UserDetails
    m getUsername(): String ↑UserDetails
    m isEnabled(): boolean ↑UserDetails
    m isAccountNonExpired(): boolean ↑UserDetails
    m isAccountNonLocked(): boolean ↑UserDetails
    m isCredentialsNonExpired(): boolean ↑UserDetails
    m eraseCredentials(): void
    m sortAuthorities(Collection<? extends GrantedAuthority>): SortedSet<GrantedAuthority>
    m equals(Object): boolean ↑Object
    m hashCode(): int ↑Object
    m toString(): String ↑Object
    m withUsername(String): UserBuilder
    m builder(): UserBuilder
    m withDefaultPasswordEncoder(): UserBuilder
    m withUserDetails(UserDetails): UserBuilder
    f serialVersionUID: long = 510L
    f logger: Log = LoggerFactory.getLog(...)
    f password: String
    f username: String
    f authorities: Set<GrantedAuthority>
    f accountNonExpired: boolean
    f accountNonLocked: boolean
    f credentialsNonExpired: boolean
    f enabled: boolean
```

其中构造方法有两个，调用其中任何一个都可以实例化

`UserDetails` 实现类 `User` 类的实例。而三个参数的构造方法实际上也是调用 7 个参数的构造方法。

- `username`: 用户名
- `password`: 密码
- `authorities`: 用户具有的权限。此处不允许为 `null`

```
public User(String username, String password,
            Collection<? extends GrantedAuthority> authorities) {
    this(username, password, enabled: true, accountNonExpired: true, credentialsNonExpired: true, accountNonLocked: true,
    authorities);
}
```

此处的用户名应该是客户端传递过来的用户名。而密码应该是从数据库中查询出来的密码。Spring Security 会根据 `User` 中的 `password` 和客户端传递过来的 `password` 进行比较。如果相同则表示认证通过，如果不相同表示认证失败。

`authorities` 里面的权限对于后面学习授权是很有必要的，包含的所有内容为此用户具有的权限，如有里面没有包含某个权限，而在做某个事情时必须包含某个权限则会出现 403。通常都是通过 `AuthorityUtils.commaSeparatedStringToAuthorityList("")` 来创建 `authorities` 集合对象的。参数是一个字符串，多个权限使用逗号分隔。

## 方法参数

方法参数表示用户名。此值是客户端表单传递过来的数据。默认情况下必须叫 `username`，否则无法接收。

## 异常

`UsernameNotFoundException` 用户名没有发现异常。在 `loadUserByUsername` 中是需要通过自己的逻辑从数据库中取值的。如果通过用户名没有查询到对应的数据，应该抛出 `UsernameNotFoundException`，系统就知道用户名没有查询到。

