Spring Security 4.0.3 入门

肖 宇

Spring Security 两大内容

- Authentication

  The process of establishing a principal is who they claim to be.

- Authorization(Access-Control)

  The process of deciding whether a principal is allowed to perform an action within your application.

增加Spring Security依赖

- Maven：pom.xml

```
<dependencies>
    <!-- ... other dependency elements ... -->
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-web</artifactId>
        <version>4.0.3.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-config</artifactId>
        <version>4.0.3.RELEASE</version>
    </dependency>
</dependencies>
```

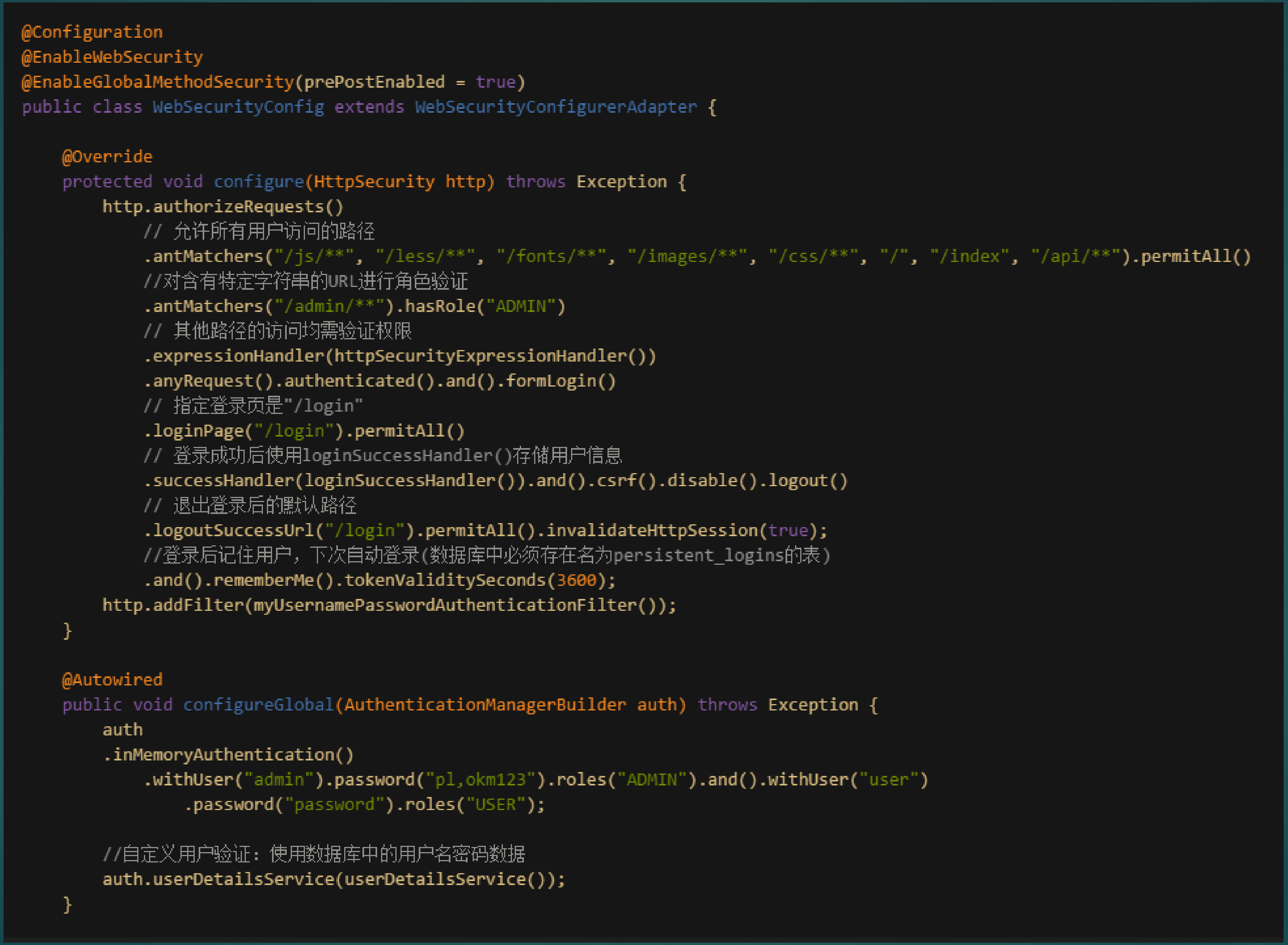
- 版本：
  - 4.0.3
  - 4.1.1

Web Security Java Configuration

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.*;
import org.springframework.security.config.annotation.authentication.builders.*;
import org.springframework.security.config.annotation.web.configuration.*;

@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth
        .inMemoryAuthentication()
            .withUser("user").password("password").roles("USER");
    }
}
```

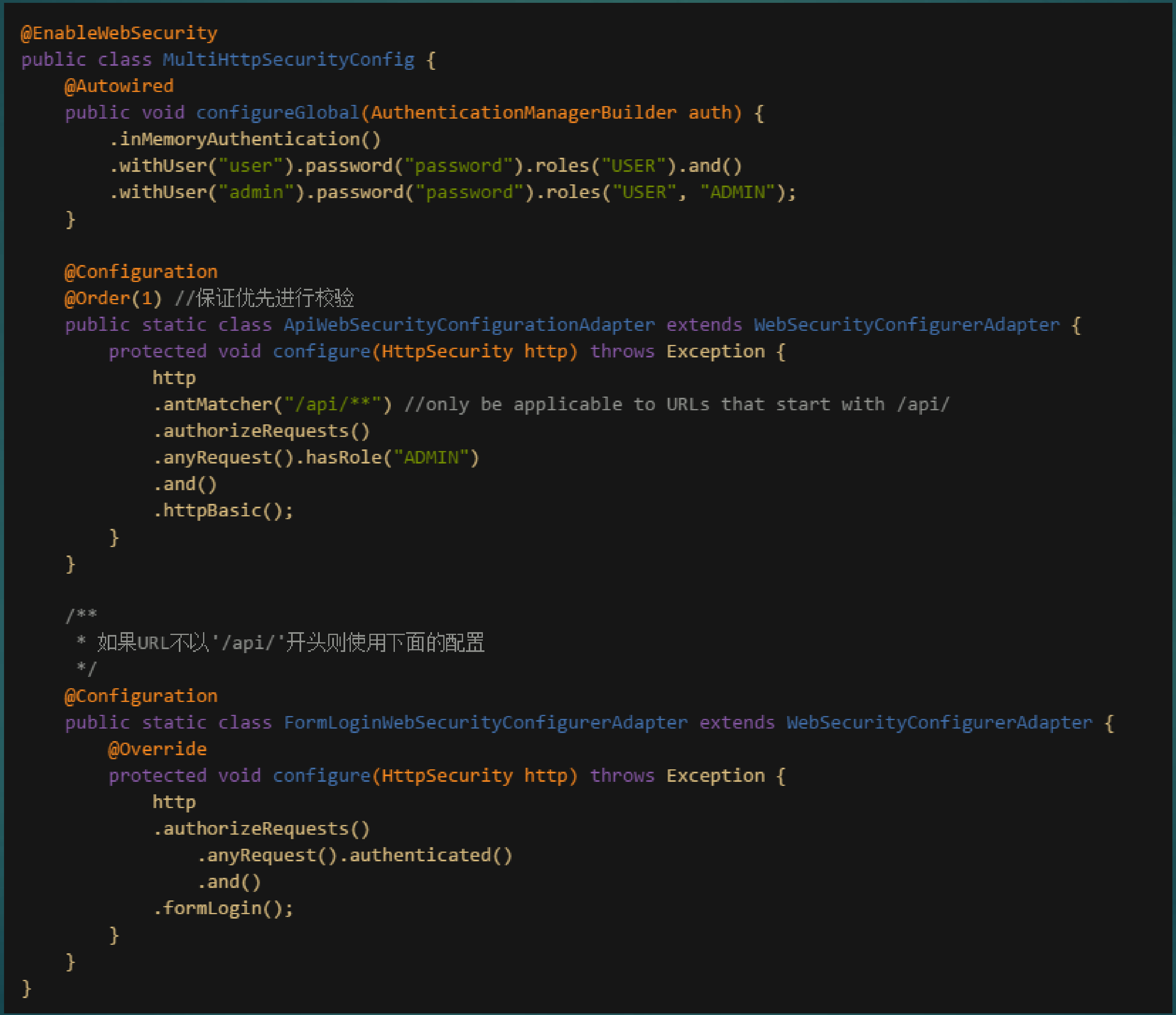
- 所有URL都要进行认证
- 提供了一个默认的登录页面
- 提供了 in-memory 的用户名/密码
- ……

DIY 1



```java
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            // 允许所有用户访问的路径
            .antMatchers("/js/**", "/less/**", "/fonts/**", "/images/**", "/css/**", "/", "/index", "/api/**").permitAll()
            //对含有特定字符串的URL进行角色验证
            .antMatchers("/admin/**").hasRole("ADMIN")
            // 其他路径的访问均需验证权限
            .expressionHandler(httpSecurityExpressionHandler())
            .anyRequest().authenticated().and().formLogin()
            // 指定登录页是"/login"
            .loginPage("/login").permitAll()
            // 登录成功后使用loginSuccessHandler()存储用户信息
            .successHandler(loginSuccessHandler()).and().csrf().disable().logout()
            // 退出登录后的默认路径
            .logoutSuccessUrl("/login").permitAll().invalidateHttpSession(true);
            //登录后记住用户，下次自动登录(数据库中必须存在名为persistent_logins的表)
            .and().rememberMe().tokenValiditySeconds(3600);
        http.addFilter(myUsernamePasswordAuthenticationFilter());
    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth
        .inMemoryAuthentication()
            .withUser("admin").password("pl,okm123").roles("ADMIN").and().withUser("user")
                .password("password").roles("USER");

        //自定义用户验证：使用数据库中的用户名密码数据
        auth.userDetailsService(userDetailsService());
    }
```

DIY 2



```
@EnableWebSecurity
public class MultiHttpSecurityConfig {
    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) {
        .inMemoryAuthentication()
        .withUser("user").password("password").roles("USER").and()
        .withUser("admin").password("password").roles("USER", "ADMIN");
    }

    @Configuration
    @Order(1) //保证优先进行校验
    public static class ApiWebSecurityConfigurationAdapter extends WebSecurityConfigurerAdapter {
        protected void configure(HttpSecurity http) throws Exception {
            http
            .antMatcher("/api/**") //only be applicable to URLs that start with /api/
            .authorizeRequests()
            .anyRequest().hasRole("ADMIN")
            .and()
            .httpBasic();
        }
    }

    /**
     * 如果URL不以'/api/'开头则使用下面的配置
     */
    @Configuration
    public static class FormLoginWebSecurityConfigurerAdapter extends WebSecurityConfigurerAdapter {
        @Override
        protected void configure(HttpSecurity http) throws Exception {
            http
            .authorizeRequests()
                .anyRequest().authenticated()
                .and()
            .formLogin();
        }
    }
}
```

Security Namespace Configuration

- Web/HTTP Security
- Business Object (Method) Security
- AuthenticationManager
- AccessDecisionManager
- AuthenticationProviders
- UserDetailsService

A namespace element can be used simply to allow a more concise way of configuring an individual bean or, more powerfully, to define an alternative configuration syntax which more closely matches the problem domain and hides the underlying complexity from the user.

Core Components

- SecurityContextHolder

  This is where we store details of the present security context of the application, which includes details of the principal currently using the application.

- SecurityContext

  Details of the present security context of the application

- Authentication

  Details of the principal currently interacting with the application.

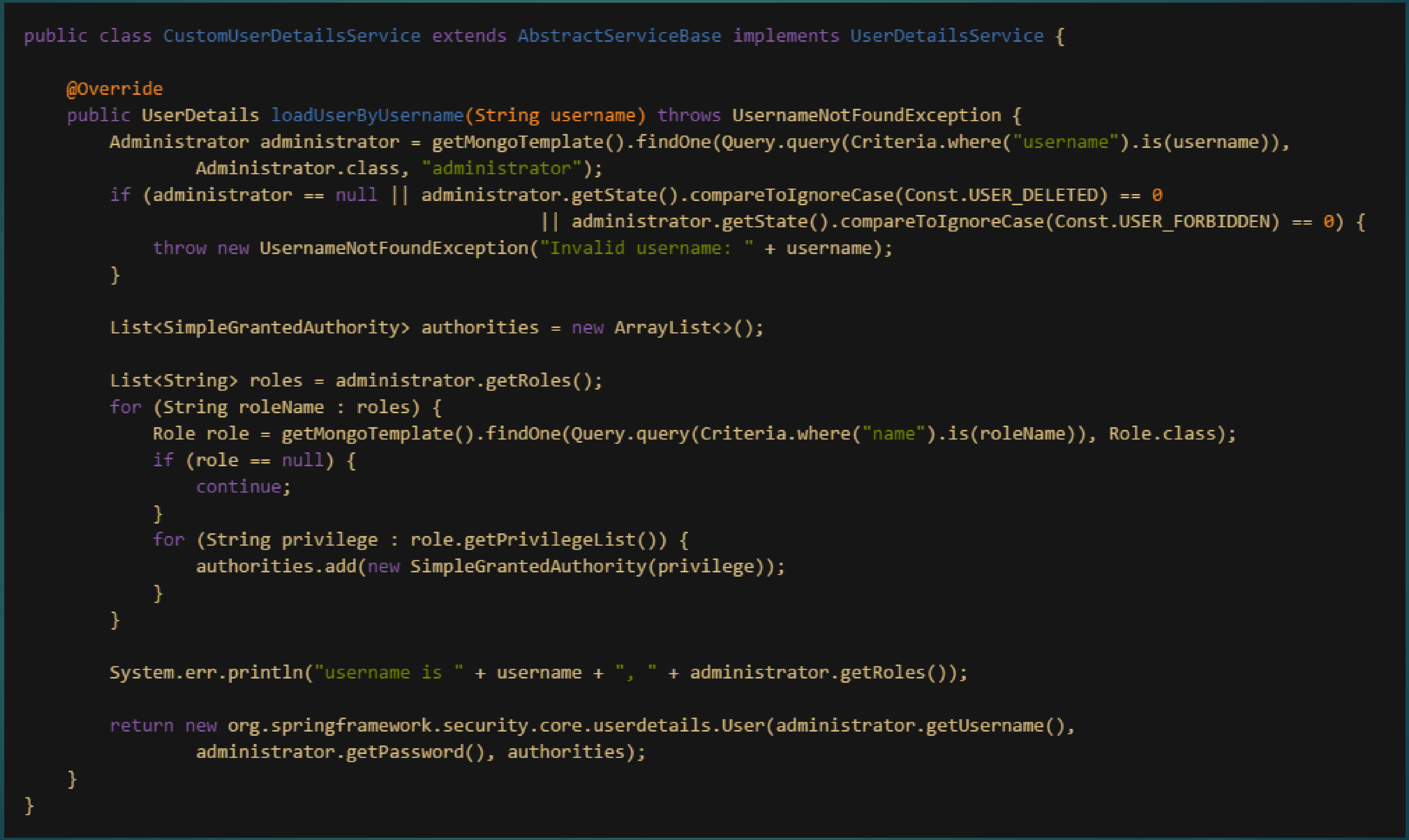
- UserDetailsService

  A interface which privodes a only method to accept a String-based username argument and returns a UserDetails.

- GrantedAuthority

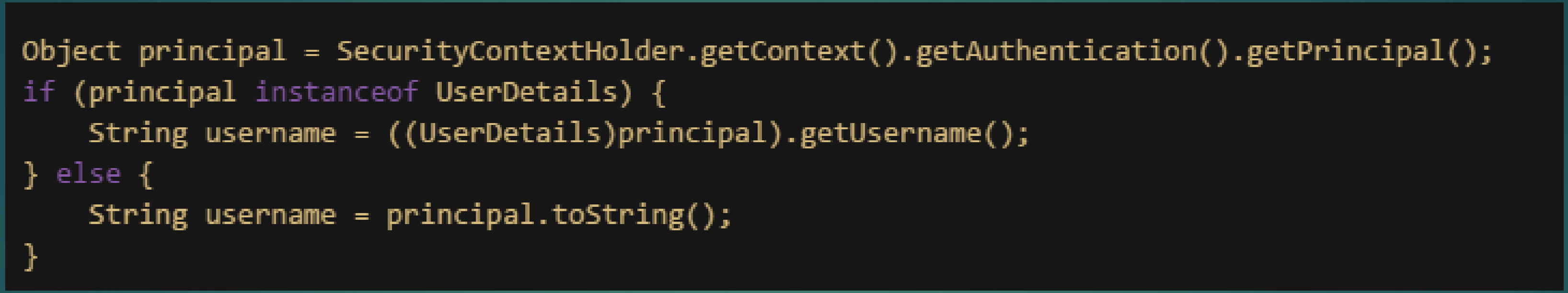
  An authority that is granted to the principal.

UserDetailService



```java
public class CustomUserDetailsService extends AbstractServiceBase implements UserDetailsService {

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        Administrator administrator = getMongoTemplate().findOne(Query.query(Criteria.where("username").is(username)),
                Administrator.class, "administrator");
        if (administrator == null || administrator.getState().compareToIgnoreCase(Const.USER_DELETED) == 0
                                                || administrator.getState().compareToIgnoreCase(Const.USER_FORBIDDEN) == 0) {
            throw new UsernameNotFoundException("Invalid username: " + username);
        }

        List<SimpleGrantedAuthority> authorities = new ArrayList<>();

        List<String> roles = administrator.getRoles();
        for (String roleName : roles) {
            Role role = getMongoTemplate().findOne(Query.query(Criteria.where("name").is(roleName)), Role.class);
            if (role == null) {
                continue;
            }
            for (String privilege : role.getPrivilegeList()) {
                authorities.add(new SimpleGrantedAuthority(privilege));
            }
        }

        System.err.println("username is " + username + ", " + administrator.getRoles());

        return new org.springframework.security.core.userdetails.User(administrator.getUsername(),
                administrator.getPassword(), authorities);
    }
}
```

UserDetails



```java
Object principal = SecurityContextHolder.getContext().getAuthentication().getPrincipal();
if (principal instanceof UserDetails) {
    String username = ((UserDetails)principal).getUsername();
} else {
    String username = principal.toString();
}
```

- UserDetails is the adapter between your own user database and what Spring Security needs inside the SecurityContextHolder;
- On successful authentication, UserDetails is used to build the Authentication object that is stored in the SecurityContextHolder;
- Whatever your UserDetailsService returns can always be obtained from the SecurityContextHolder using the above code fragment.

Expression-Based Access Control

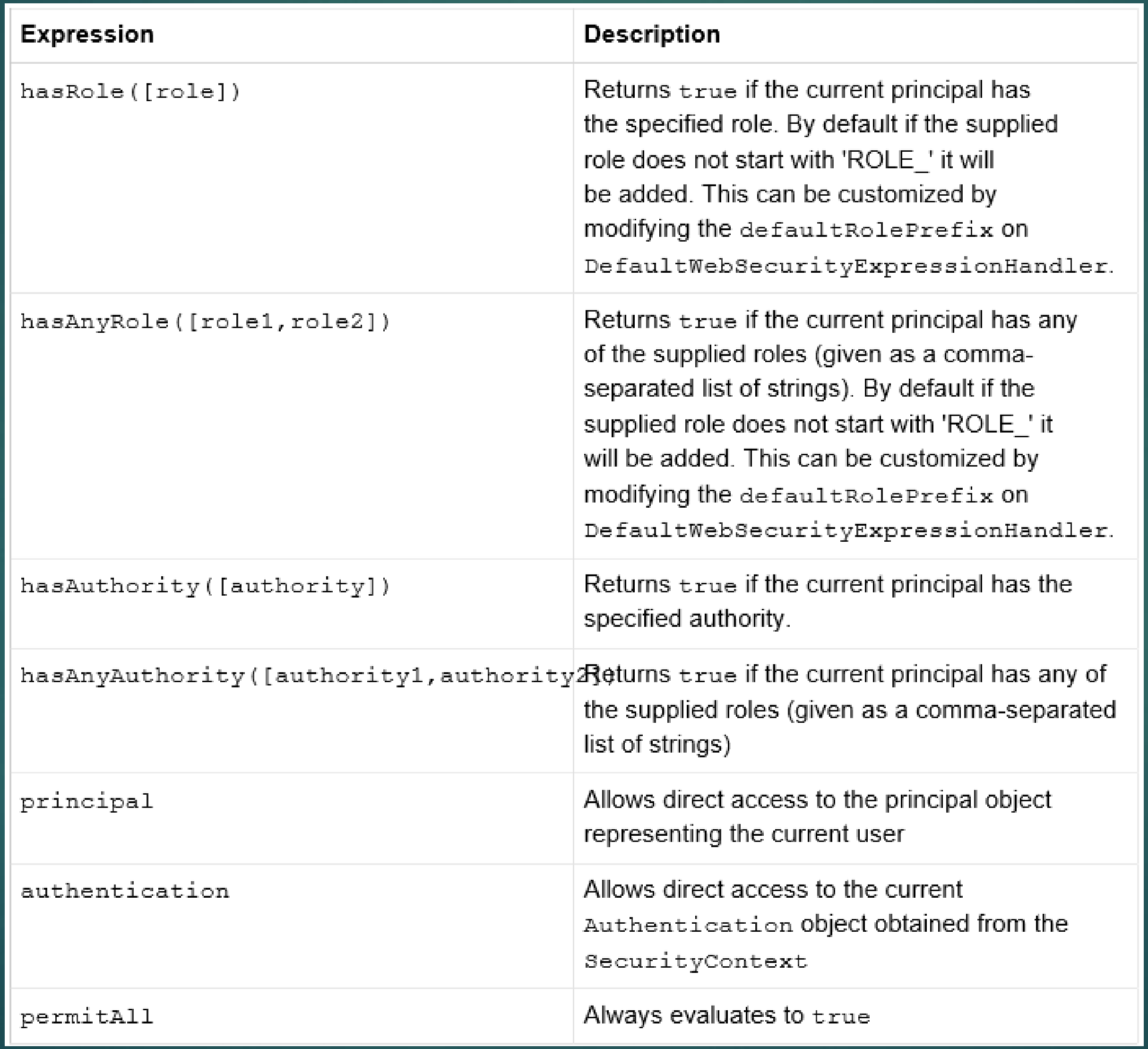
- Spring Expression Language(Spring EL)
- Web Security Expressions

  Use expressions to secure individual URLs
- Method Security Expressions

  Allow pre and post-invocation authorization checks and also to support filtering of submitted collection arguments or return values.
- The PermissionEvaluator interface

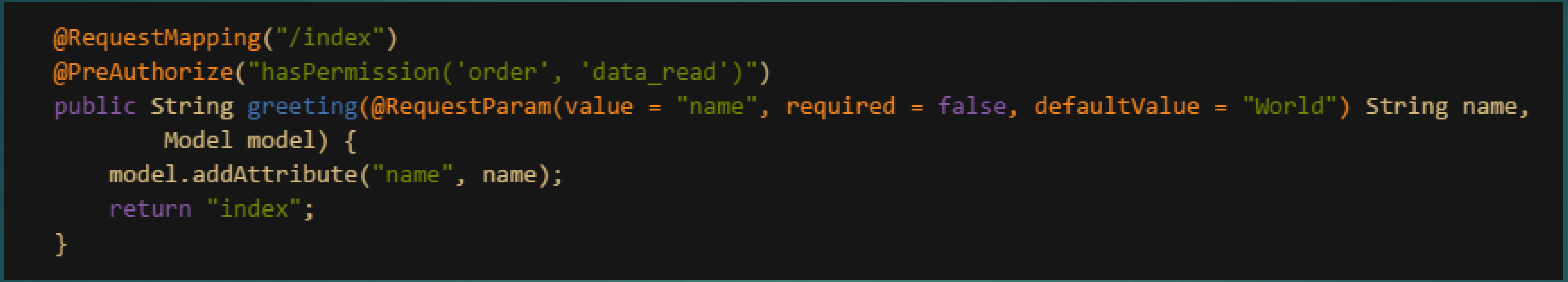
  Bridge between the expression system and Spring Security's ACL system, allowing you to specify authorization constraints on domain objects, based on abstract permissions.

Common Built-in Expressions



Expression	Description
`hasRole([role])`	Returns `true` if the current principal has the specified role. By default if the supplied role does not start with 'ROLE_' it will be added. This can be customized by modifying the `defaultRolePrefix` on `DefaultWebSecurityExpressionHandler`.
`hasAnyRole([role1,role2])`	Returns `true` if the current principal has any of the supplied roles (given as a comma-separated list of strings). By default if the supplied role does not start with 'ROLE_' it will be added. This can be customized by modifying the `defaultRolePrefix` on `DefaultWebSecurityExpressionHandler`.
`hasAuthority([authority])`	Returns `true` if the current principal has the specified authority.
`hasAnyAuthority([authority1,authority2])`	Returns `true` if the current principal has any of the supplied roles (given as a comma-separated list of strings)
`principal`	Allows direct access to the principal object representing the current user
`authentication`	Allows direct access to the current `Authentication` object obtained from the `SecurityContext`
`permitAll`	Always evaluates to `true`

Method Security Expressions

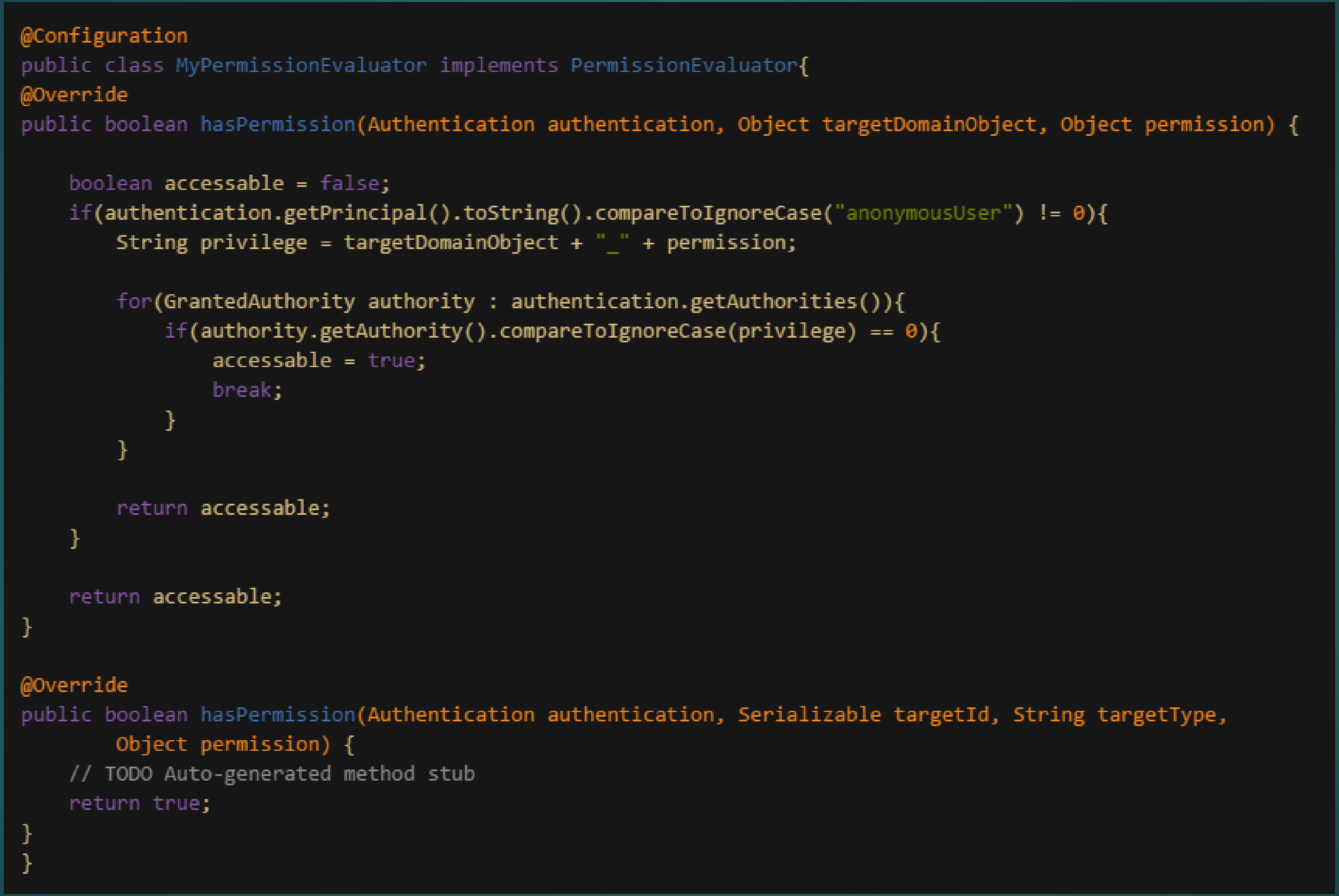


```java
@RequestMapping("/index")
@PreAuthorize("hasPermission('order', 'data_read')")
public String greeting(@RequestParam(value = "name", required = false, defaultValue = "World") String name,
        Model model) {
    model.addAttribute("name", name);
    return "index";
}
```

Web Security Expressions

```html
<div class="btn-group" >
    <button sec:authorize="${hasPermission('institution', 'read')}" type="button" class="btn btn-success btn-xs"
        th:attr="data-id=${institution.id}" data-toggle="modal"
        data-target="#modal_detail">详情</button>
        <button sec:authorize="${hasPermission('institution', 'modify')}" type="button" class="btn btn-info btn-xs"
        th:attr="data-id=${institution.id}" data-toggle="modal"
        data-target="#modal_check">审核</button>
    <button sec:authorize="${hasPermission('institution', 'modify')}" type="button" class="btn btn-default btn-xs"
        th:attr="data-id=${institution.id}" data-toggle="modal"
        data-target="#modal_modify">修改</button>
    <button sec:authorize="${hasPermission('institution', 'delete')}" type="button" class="btn btn-danger btn-xs"
        th:attr="data-id=${institution.id}" data-toggle="modal"
        data-target="#modal_delete">停用</button>
</div>
```

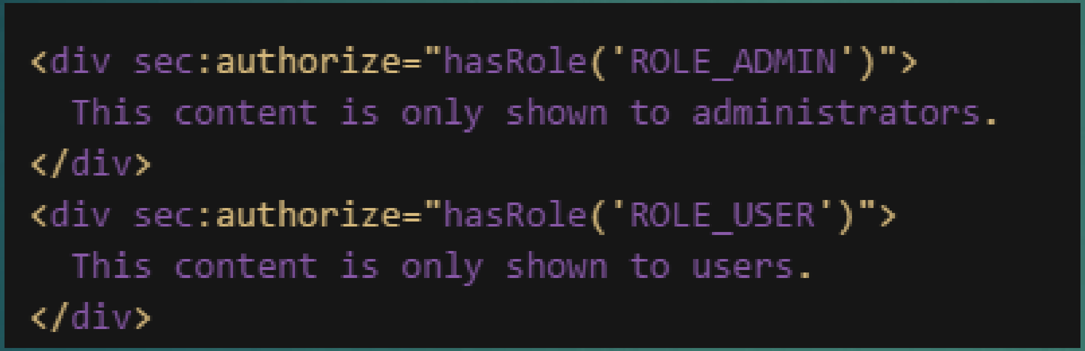
The PermissionEvaluator interface



```
@Configuration
public class MyPermissionEvaluator implements PermissionEvaluator{
@Override
public boolean hasPermission(Authentication authentication, Object targetDomainObject, Object permission) {

    boolean accessable = false;
    if(authentication.getPrincipal().toString().compareToIgnoreCase("anonymousUser") != 0){
        String privilege = targetDomainObject + "_" + permission;

        for(GrantedAuthority authority : authentication.getAuthorities()){
            if(authority.getAuthority().compareToIgnoreCase(privilege) == 0){
                accessable = true;
                break;
            }
        }

        return accessable;
    }

    return accessable;
}

@Override
public boolean hasPermission(Authentication authentication, Serializable targetId, String targetType,
        Object permission) {
    // TODO Auto-generated method stub
    return true;
}
}
```

Thymeleaf + Spring Security

- Use Thymeleaf dialect to show different content to different roles.



```html
<div sec:authorize="hasRole('ROLE_ADMIN')">
  This content is only shown to administrators.
</div>
<div sec:authorize="hasRole('ROLE_USER')">
  This content is only shown to users.
</div>
```

TODO

- CSRF Attack？
- Session？
- Access-decision voters？
- AOP？
- Hierarchical Roles？
- Spring Projects: http://spring.io/projects
- Spring Security Reference
- github clone https://github.com/spring-projects/spring-security.git
- Taglibs: Spring Security Taglib