链滴

spring security 权限控制

作者：tianyunperfect
原文链接：https://ld246.com/article/1545812805218
来源网站：链滴

前言

spring自带角色权限控制框架

用户-角色

数据库和domain

- 数据库

```
-- 用户表
CREATE TABLE sys_user(
id int auto_increment PRIMARY KEY ,
username VARCHAR(50),
email VARCHAR(50) ,
PASSWORD VARCHAR(80),
phoneNum VARCHAR(20),
STATUS int(1)
);

-- 角色表
CREATE TABLE sys_role(
    id int auto_increment PRIMARY KEY,
    roleName VARCHAR(50) ,
    roleDesc VARCHAR(50)
)
-- 用户和角色中间表
CREATE TABLE sys_user_role(
    userId int,
    roleId int,
```

```
    PRIMARY KEY(userId,roleId),
    FOREIGN KEY (userId) REFERENCES sys_USER(id),
    FOREIGN KEY (roleId) REFERENCES sys_role(id)
)
```

- domain

```
public class SysUser {
    private Long id;
    private String username;
    private String email;
    private String password;
    private String phoneNum;
    private int status;
    private List<Role> roles;
}
public class Role {
    private Long id;
    private String roleName;
    private String roleDesc;
}
```

静态页面

403.jsp

login.jsp

error.jsp

配置文件

springSecurity.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:security="http://www.springframework.org/schema/security"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/security
    http://www.springframework.org/schema/security/spring-security.xsd">

    <!--放行未登录访问的页面-->
    <security:http pattern="/login.jsp" security="none"></security:http>
    <security:http pattern="/error.jsp" security="none"></security:http>
    <security:http pattern="/css/**" security="none"></security:http>
    <security:http pattern="/img/**" security="none"></security:http>
    <security:http pattern="/pages/**" security="none"></security:http>
    <security:http pattern="/plugins/**" security="none"></security:http>
    <!--配置拦截器的路径规则
    auto-config="true" 表示使用权限框架默认的配置
    use-expressions="false" 关闭权限框架的表达式 spel
    intercept-url  拦截请求资源的路径
```

```
    access="ROLE_USER" 允许访问的条件 当前用户必须拥有ROLE_USER的角色才可以访问
    -->
    <security:http auto-config="true" use-expressions="true">
        <!--权限框架支持多种角色的登录 角色之间的关系为or 或者的关系-->
        <security:intercept-url pattern="/**" access="hasAnyRole('ROLE_USER','ROLE_ADMIN')"
</security:intercept-url>
        <!--自定义页面的配置节点-->
        <security:form-login login-page="/login.jsp"
                             login-processing-url="/login"
                             default-target-url="/index.jsp"
                             authentication-failure-url="/error.jsp"></security:form-login>

        <!--登录成功权限不足的处理-->
        <security:access-denied-handler error-page="/403.jsp"></security:access-denied-handl
r>
        <!--csrf关闭跨域请求的攻击-->
        <security:csrf disabled="true"></security:csrf>
        <!--
        logout 退出请求的url路径 实际是页面点击按钮请求的地址
        logout-success-url 成功注销后 跳转的页面
        invalidate-session 设置session失效
        -->
        <security:logout logout-url="/logOut" logout-success-url="/login.jsp" invalidate-sessio
="true"></security:logout>

    </security:http>

    <!--配置拦截后验证的节点-->
    <security:authentication-manager>
        <security:authentication-provider user-service-ref="userService">
            <!--自定义的加密工具类-->
            <security:password-encoder ref="pwdEncoder">></security:password-encoder>
        </security:authentication-provider>
    </security:authentication-manager>

    <!--配置自定义的加密工具类，这里使用自带的-->
    <bean id="pwdEncoder" class="org.springframework.security.crypto.bcrypt.BCryptPasswor
Encoder">></bean>

    <!--开启注解支持-->
    <security:global-method-security secured-annotations="enabled"/>

</beans>
```

web.xml引入

- 引入filter

```
<!--配置框架使用的filter过滤器-->
<filter>
    <!--过滤器执行链名臣固定springSecurityFilterChain-->
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
```

```
</filter>
<filter-mapping>
   <filter-name>springSecurityFilterChain</filter-name>
   <url-pattern>/*</url-pattern>
</filter-mapping>
```

- 引入配置文件

```
<!--spring的listener-->
<listener>
   <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<context-param>
   <param-name>contextConfigLocation</param-name>
   <param-value>classpath:applicationContext.xml,classpath:springSecurity.xml</param-valu
>
</context-param>
```

service校验方法

- 在service中继承spring接口

```
public interface SysUserService extends UserDetailsService {
   @Override
   UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;
}
```

- 实现接口

```
@Service("userService")
public class SysUserServiceImpl implements SysUserService {

   @Autowired
   private SysUserDao userDao;

   @Autowired
   BCryptPasswordEncoder pwdEncoder;

   @Override
   public UserDetails loadUserByUsername(String username) throws UsernameNotFoundExce
tion {
      //得到数据库的用户
      SysUser sysUser =  userDao.findUserByName(username);
      //框架的User对象用于验证返回 用户名 密码 用户的权限集合
      //查询得到用户真正的角色集合返回
      List<Role> roles =sysUser.getRoles();
      List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();
      ///如果当前用户确实拥有角色 循环添加到集合中
      if(roles!=null&&roles.size()>0){
         for (Role role : roles) {
            authorities.add(new SimpleGrantedAuthority(role.getRoleName()));
         }
```

```
        }

        User user = new User(sysUser.getUsername(),sysUser.getPassword(),authorities);
        return user;

    }
}
```

用户名的获取

- 前台获取

```
# 方式一
${ sessionScope.SPRING_SECURITY_CONTEXT.authentication.principal.username }

# 方式二
<security:authentication property="principal.username"/>
```

- 后台获取

```
// 先获取到SecurityContext对象
SecurityContext context = SecurityContextHolder.getContext();
// 获取到认证的对象
Authentication authentication = context.getAuthentication();
// 获取到登录的用户信息
User user = (User) authentication.getPrincipal();
System.out.println(user.getUsername());
```

不同角色访问控制权限

jsp页面

```
<security:authorize access="hasAnyRole('ROLE_USER','ROLE_ADMIN')">
  <li id="system-setting">
  <a href="${pageContext.request.contextPath}/product/findByPageHelper">
  <i class="fa fa-circle-o"></i> 产品管理
  </a>
  </li>
</security:authorize>
```

后台

- springmcv.xml

```
<!--手打才能自动引入-->
<security:global-method-security secured-annotations="enabled"></security:global-method
security>
```

- controller

```
@Secured("ROLE_ADMIN")
public class RoleController
```

原文链接：spring security 权限控制