2、 使用自定义方法

虽然这里面已经包含了很多的表达式(方法)但是在实际项目中很有可能出现需要自己自定义逻辑的情况。

判断登录用户是否具有访问当前 URL 权限。

新建接口及实现类

MyService.java

```java
package com.xxxx.springsecuritydemo.service;

import org.springframework.security.core.Authentication;

import javax.servlet.http.HttpServletRequest;

public interface MyService {
   boolean hasPermission(HttpServletRequest request, Authentication authentication);
}
```

MyServiceImpl.java

```java
package com.xxxx.springsecuritydemo.service.impl;

import com.xxxx.springsecuritydemo.service.MyService;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;

import javax.servlet.http.HttpServletRequest;
import java.util.Collection;

/**
 * @author zhoubin
 * @since 1.0.0
 */
@Component
public class MyServiceImpl implements MyService {

   @Override
   public boolean hasPermission(HttpServletRequest request, Authentication authentication)
{
      Object obj = authentication.getPrincipal();
      if (obj instanceof UserDetails){
         UserDetails userDetails = (UserDetails) obj;
         Collection<? extends GrantedAuthority> authorities = userDetails.getAuthorities();
         return authorities.contains(new SimpleGrantedAuthority(request.getRequestURI()));
      }
      return false;
```

```
    }
}
```

修改配置类

在 access 中通过@bean的id名.方法(参数)的形式进行调用配置类中修改如下:

```
//url拦截
http.authorizeRequests()
      //login.html不需要被认证
      // .antMatchers("/login.html").permitAll()
      .antMatchers("/login.html").access("permitAll")
      // .antMatchers("/main.html").hasRole("abc")
      .antMatchers("/main.html").access("hasRole('abc')")
      .anyRequest().access("@myServiceImpl.hasPermission(request,authentication)")
```