# Spring Security 框架简介

官网：[https://spring.io/projects/spring-security#overview](https://spring.io/projects/spring-security" \l "overview)

Spring Security是强大的，且容易定制的实现认证，与授权的基于Spring开发的框架。

Spring Security的功能：  
1）Authentication：认证，就是用户邓丽

2）Authorization：授权，判断用户拥有什么权限，可以访问什么资源

3）安全防护，防止跨站请求，session攻击等

4）非常容易结合SpringMVC进行使用

# 2. Spring Security和Shiro的相同与不同

**2.1.相同点**

1）认证功能

2）授权功能

3）加密功能

4）会话功能

5）缓存支持

6）rememberMe功能

……

**2.2.不同点**

优点：

1. Spring Security基于Spring开发，项目如果使用Spring作为基础，配合Spring Security做权限更加方便。而Shiro需要和Spring进行整合开发。
2. Spring Security功能比Shiro更加丰富些，例如安全防护方面
3. Spring Security社区资源相对于Shiro更加丰富

缺点：

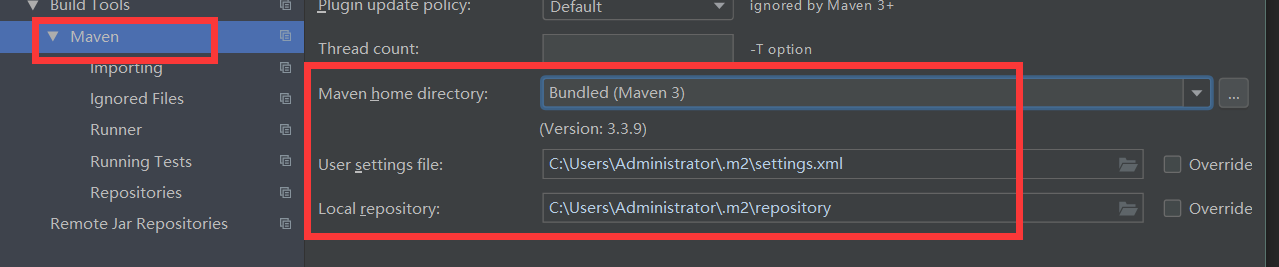
1. Shiro的配置和使用比较简单，Spring Security上手复杂些

2）Shiro依赖性低，不需要任何框架和容器，可以独立运行，Spring Security依旧依赖Spring容器

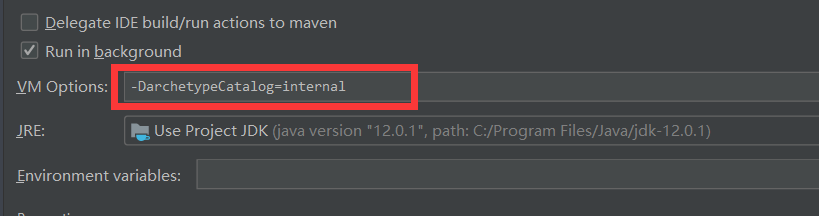
# 3. Spring Security，Spring和SpringMVC整合

## 3.1 Idea配置

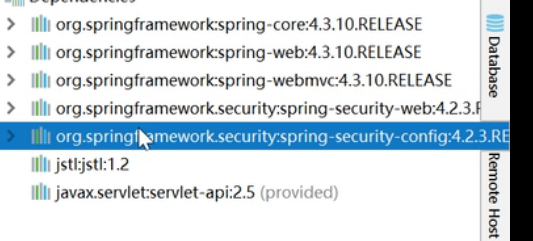
Maven版本配置



Maven参数配置 – 在runner中



## 3.2项目的pom文件中添加所需要的Jar



|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-core</**artifactId**>  <**version**>5.1.5.RELEASE</**version**> </**dependency**> <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-web</**artifactId**>  <**version**>5.1.5.RELEASE</**version**> </**dependency**> <**dependency**>  <**groupId**>org.springframework</**groupId**>  <**artifactId**>spring-webmvc</**artifactId**>  <**version**>5.1.5.RELEASE</**version**> </**dependency**> <**dependency**>  <**groupId**>javax.servlet</**groupId**>  <**artifactId**>servlet-api</**artifactId**>  <**version**>2.5</**version**>  <**scope**>provided</**scope**> </**dependency**> <**dependency**>  <**groupId**>org.springframework.security</**groupId**>  <**artifactId**>spring-security-web</**artifactId**>  <**version**>4.2.3.RELEASE</**version**> </**dependency**> <**dependency**>  <**groupId**>org.springframework.security</**groupId**>  <**artifactId**>spring-security-config</**artifactId**>  <**version**>4.2.3.RELEASE</**version**> </**dependency**> |

## 3.3 ApplicationContext.xml和SpringMVC.xml配置

|  |
| --- |
| *SpringMVC：*  *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:aop="http://www.springframework.org/schema/aop"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xmlns:p="http://www.springframework.org/schema/p"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/aop  http://www.springframework.org/schema/aop/spring-aop-4.0.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd  http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd"**>   <**context:component-scan base-package="org.lzq.Controller"**></**context:component-scan**>   <**mvc:annotation-driven**></**mvc:annotation-driven**>   <**bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  <**property name="prefix" value="/jsp/"** />  <**property name="suffix" value=".jsp"** />  </**bean**>  </**beans**> |

|  |
| --- |
| *applicationContext.xml*  *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:aop="http://www.springframework.org/schema/aop"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:beans="http://www.springframework.org/schema/beans"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/aop  http://www.springframework.org/schema/aop/spring-aop-4.0.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd  "**>  </**beans**> |

## 3. 4 Spring Sercurity配置

在web.xml配置过滤器还有Spring Sercurity的配置文件

|  |
| --- |
| *<!-- Spring Sercurity-->* <**filter**>  <**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**> |

Spring Sercurity的配置文件

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:security="http://www.springframework.org/schema/security"  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"**>   <**security:http**>  <**security:http-basic**/>  </**security:http**>   <**security:authentication-manager**>   </**security:authentication-manager**>  </**beans**> |

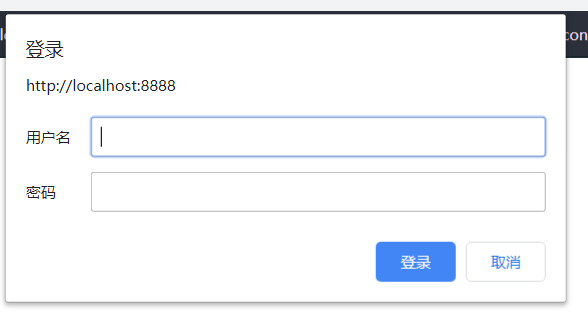
# 4. HttpBasic方式的权限实现

## 4.1 Controller层编写

|  |
| --- |
| **package** org.lzq.Controller;   **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RestController;  @RestController @RequestMapping(**"/"**) **public class** ProjectController {   *//增加商品* @RequestMapping(**"/addProject"**)  **private** String AddProject(){   **return "addProject"**;  }  *//删除商品* @RequestMapping(**"/delProject"**)  **private** String delProject(){   **return "delProject"**;  }  *//查询商品* @RequestMapping(**"/queryProject"**)  **private** String queryProject(){   **return "queryProject"**;  }  *//更新商品* @RequestMapping(**"/updataProject"**)  **private** String updataProject(){   **return "updataProject"**;  } } |

## 4.2在Spring Security中的配置

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:security="http://www.springframework.org/schema/security"  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"**>   *<!-- security:http :Spring过滤器链配置：  1）需要拦截什么资源  2）什么资源什么角色权限  3）定义认证方式：httpBasic，FormLogin(\*)  4）定义登陆页面，定义登陆请求地址，定义错误处理方式  -->* <**security:http**>  *<!--  pattern:需要拦截的资源  access:拦截方式  isFullyAuthenticated():资源需要认证才可以访问  permitAll():允许所有人(匿名和登陆用户)访问  isAnonymous():只有匿名用户才可以访问(如果登陆用户就无法访问)  -->* <**security:intercept-url access="permitAll()" pattern="/index.jsp"**></**security:intercept-url**>  <**security:intercept-url access="isFullyAuthenticated()" pattern="/\*\*"**></**security:intercept-url**>   *<security:http-basic/>* </**security:http**>  *<!--  security:authentication-manager: 认证管理器  1）认证信息提供方式（账户名，密码，当前用户权限）  -->* <**security:authentication-manager**>  <**security:authentication-provider**>  <**security:user-service**>  <**security:user name="Sakura" password="123456" authorities="ROLE\_USER"** />  </**security:user-service**>  </**security:authentication-provider**>  </**security:authentication-manager**>  </**beans**> |

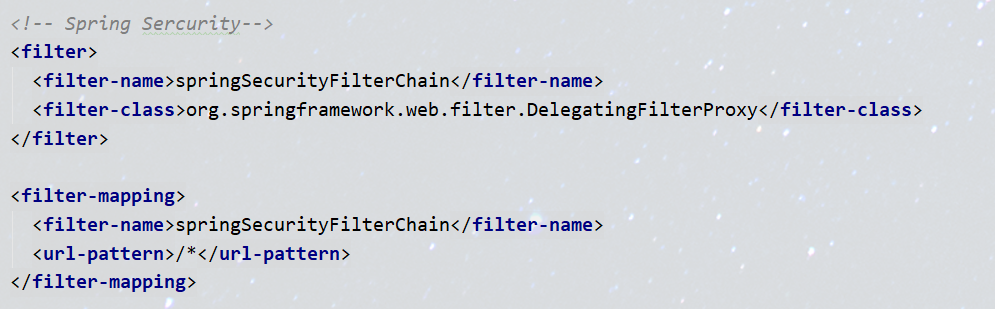


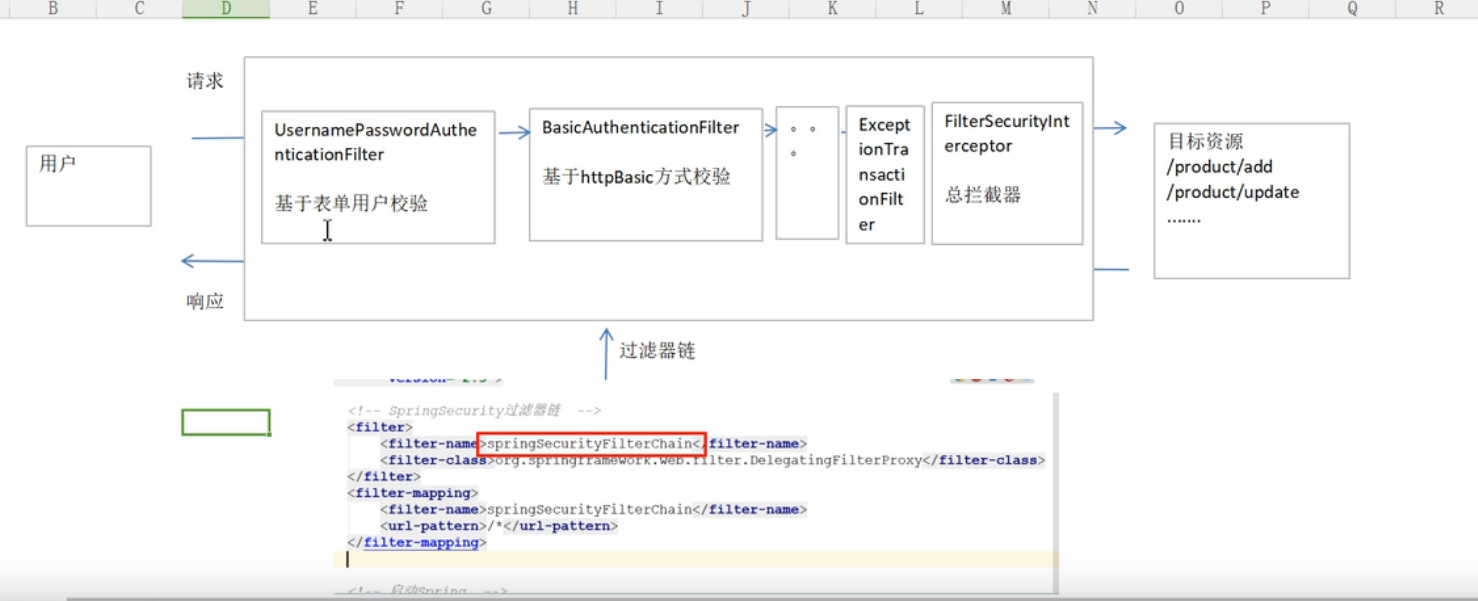
# 5. FormLogin方式的权限实现

|  |
| --- |
| <**security:http**>  *<!--  pattern:需要拦截的资源  access:拦截方式  isFullyAuthenticated():资源需要认证才可以访问  permitAll():允许所有人(匿名和登陆用户)访问  isAnonymous():只有匿名用户才可以访问(如果登陆用户就无法访问)*  **hasAuthority():*只有相对于拥有权限的用户才可以访问***  *-->* <**security:intercept-url access="permitAll()" pattern="/index.jsp"**></**security:intercept-url**>  <**security:intercept-url access="isFullyAuthenticated()" pattern="/\*\*"**></**security:intercept-url**>  *<!-- <security:http-basic/> -->* <**security:form-login**/> </**security:http**> |

# 6. Spring Security执行原理

核心：过滤器链



执行原理：  


# 7. 自定义FormLogin验证页面

## 7.1 在Spring Security配置文件中

Ps.如果更改完login页面后，记得将login页面添加到所有人可以访问

|  |
| --- |
| *<!--  security:form-login:  login-page=""自定义页面  login-processing-url=""自定义action跳转页面 -->* <**security:form-login login-page="xxx"**  **login-processing-url="xxx"**/> *<!--  security:csrf开启或者关闭csrf防护  true : 关闭  false : 开启 -->* <**security:csrf disabled="true"** /> |

## 7.2 编写登陆页面

必须是Web.xml为3.0才可以解析EL表达式

|  |
| --- |
| <**web-app xmlns="http://java.sun.com/xml/ns/javaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  version="3.0"** > |

根据UsernamePasswordAuthenicationFilter类可知：  
1.表单提交的action必须为login（如果需要修改则修改Spring Security中的login-processing-url=””）

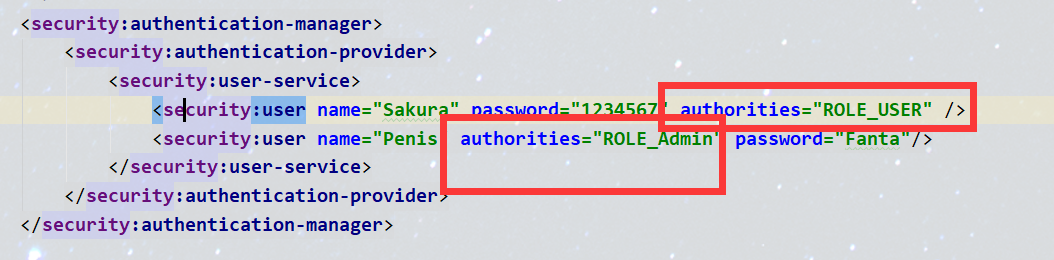
2.提交的方法必须为post方法

3．Input两个name参数也必须为username和password

|  |
| --- |
| <%@ **page contentType**="**text/html;charset=UTF-8**" **language**="**java**" %> <**html**> <**head**>  <**title**>Title</**title**> </**head**> <**body**>  <**h3**>登陆页面</**h3**>  <**form action="${**pageContext.request.contextPath**}/login" method="post"**>  用户名:<**input type="text" name="username"**/>  密码:<**input type="password" name="password"**/>  <**input type="submit" value="登陆"**>  </**form**> </**body**> </**html**> |

# 8. user-service配置实现用户权限访问控制

## 8.1 关于用户管理器



在User-Service中编写好了用户权限后，可以在<security:http>中具体分配权限

实现的是**hasAnyRole**

|  |
| --- |
| <**security:intercept-url access="hasAnyRole('ROLE\_Admin')" pattern="/jsp/updataProject.jsp" /**> |

## 8.2 关于错误页面的编写

|  |
| --- |
| <**security:access-denied-handler error-page="/error.jsp"** /> |

# 9. 自定义UserDetailService类实现用户权限访问控制

关键：使用UserDetailService接口

## 9.1 创建UserDEtailService接口实现类

|  |
| --- |
| **public class** UserDetailService **implements** UserDetailsService {  @Override  **public** UserDetails loadUserByUsername(String s) **throws** UsernameNotFoundException {  User user = **new** User(**"Sakura"**,**"123456"**, AuthorityUtils.*commaSeparatedStringToAuthorityList*(**"ROLE\_USER,ROLE\_Admin"**));  **return** user;  } } |

其中User类就是UserDetail实现类，用于封装数据库账户信息

## 9.2 将实现的类注入到Spring Security中

|  |
| --- |
| *<!--  security:authentication-manager: 认证管理器  1）认证信息提供方式（账户名，密码，当前用户权限） -->* <**security:authentication-manager**>  <**security:authentication-provider user-service-ref="userDetailService"**>  </**security:authentication-provider**> </**security:authentication-manager**>  <**bean id="userDetailService" class="org.lzq.security.UserDetailService"** /> |

# 10. 自定义登陆成功与失败处理逻辑

关键：  
1）登陆成功处理：AuthenticationSuccessHandler

2) 登陆失败处理：AuthenticationFailureHandler

步骤；

1. 创建AuthenticationSuccess/FailureHandler的接口实现类
2. 在Spring Security配置文件中配置

## 10.1 创建Success/Failure接口实现类

|  |
| --- |
| Success：  **public class** MySucessAuthencation **implements** AuthenticationSuccessHandler {  **private** ObjectMapper **objectMapper** = **new** ObjectMapper();   @Override  **public void** onAuthenticationSuccess(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Authentication authentication) **throws** IOException, ServletException {  Map map = **new** HashMap<>();  map.put(**"sucess"**,**true**);  String json = **objectMapper**.writeValueAsString(map);  httpServletResponse.getWriter().write(json);  } } |
| Failure：  **public class** MyNoSucessfulAuthencation **implements** AuthenticationFailureHandler {   **private** ObjectMapper **objectMapper** = **new** ObjectMapper();   @Override  **public void** onAuthenticationFailure(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, AuthenticationException e) **throws** IOException, ServletException {  Map map = **new** HashMap<>();  map.put(**"Failure"**,**false**);  String json = **objectMapper**.writeValueAsString(map);  httpServletResponse.getWriter().write(json);  } } |

## 10.2在Spring Security配置文件中配置

通过

**authentication-success-handler-ref 配置成功登陆逻辑**

**authentication-failure-handler-ref 配置失败登陆逻辑**

|  |
| --- |
| <**security:form-login default-target-url="/login.jsp" authentication-success-handler**  **-ref="mySucessAuthencation"**  **authentication-failure-handler-ref="myNoSucessfulAuthencation"**/> |
| <**bean id="mySucessAuthencation" class="org.lzq.security.MySucessAuthencation"** /> |
| <**bean id="myNoSucessfulAuthencation" class="org.lzq.security.MyNoSucessfulAuthencation"** /> |

# 11. 源码分析以及认证流程

