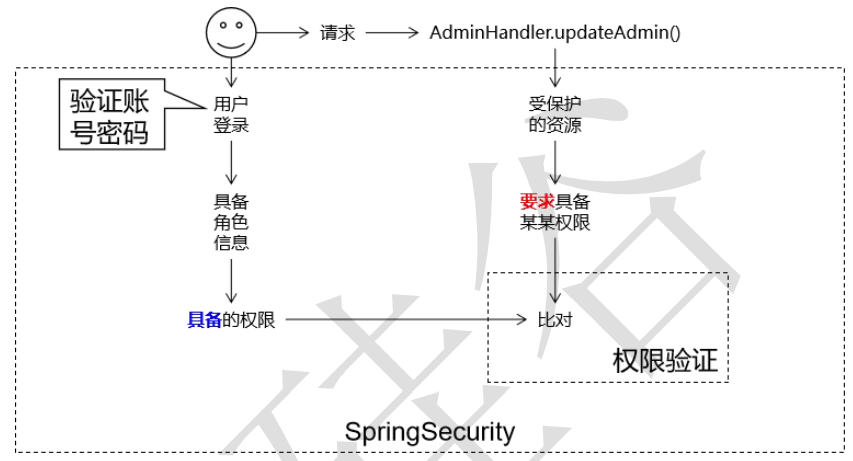
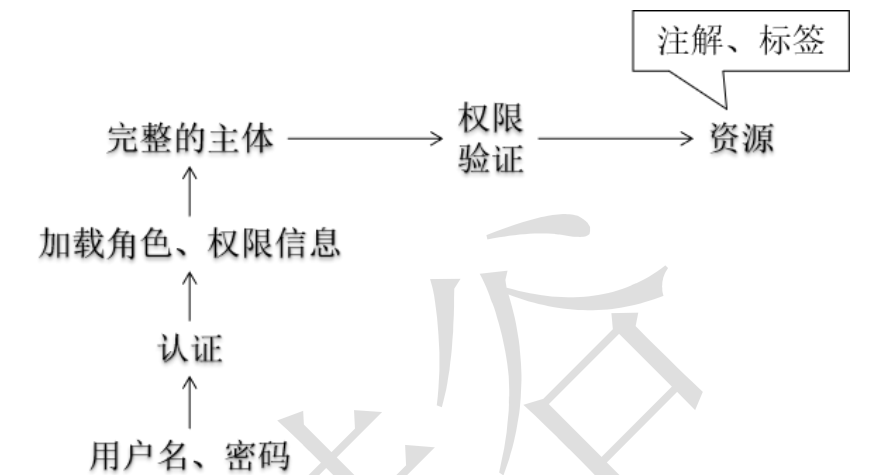
# SpringSecurity简介

用户登录系统时我们协助 SpringSecurity 把用户对应的角色、权限组装好，同时把各个资源所要求的权限信息设定好，剩下的“登录验证”、“权限验证”等等工作都交给SpringSecurity。



## 相关概念



1.主体 英文单词：principal

使用系统的用户或设备或从其他系统远程登录的用户等等。简单说就是谁使用系统谁就是主体。

2. 认证 英文单词：authentication

权限管理系统确认一个主体的身份，允许主体进入系统。简单说就是“主体”证明自己是谁。 笼统的认为就是以前所做的登录操作。

2.授权 英文单词：authorization

将操作系统的“权力”“授予”“主体”，这样主体就具备了操作系统中特定功能的能力。 所以简单来说，授权就是给用户分配权限。

## 优缺点

Spring 技术栈的组成部分。

通过提供完整可扩展的认证和授权支持保护你的应用程序

https://spring.io/projects/spring-security

SpringSecurity 特点： 

1.和 Spring 无缝整合。 

2.全面的权限控制。 

3.专门为 Web 开发而设计。 

旧版本不能脱离 Web 环境使用。 

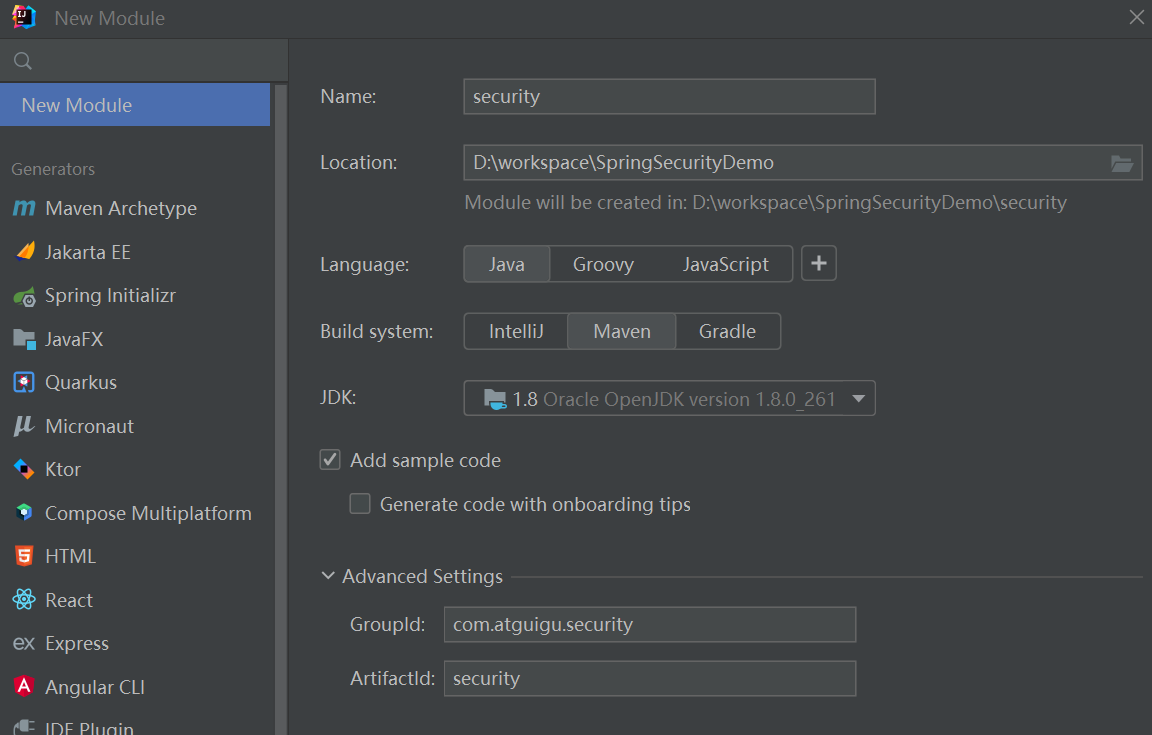
新版本对整个框架进行了分层抽取，分成了核心模块和Web 模块。单独引入核心模块就可以脱离 Web 环境。 

4.重量级。

# 环境搭建

## web工程

先创建一个maven工程，再手动添加webapp模块

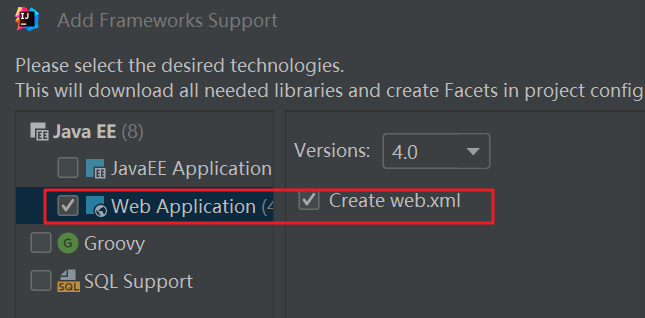


手动添加webapp模块

1.右键模块，找到add framework support

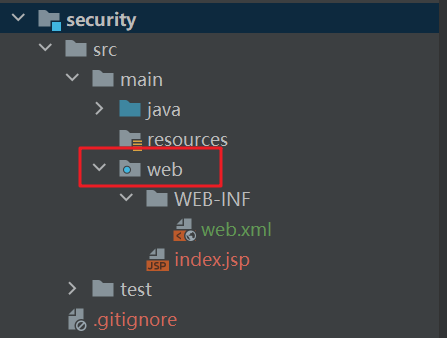


2.勾选上web application->create web.xml

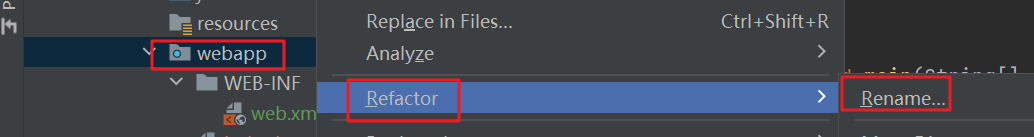


3.手动移到main目录下

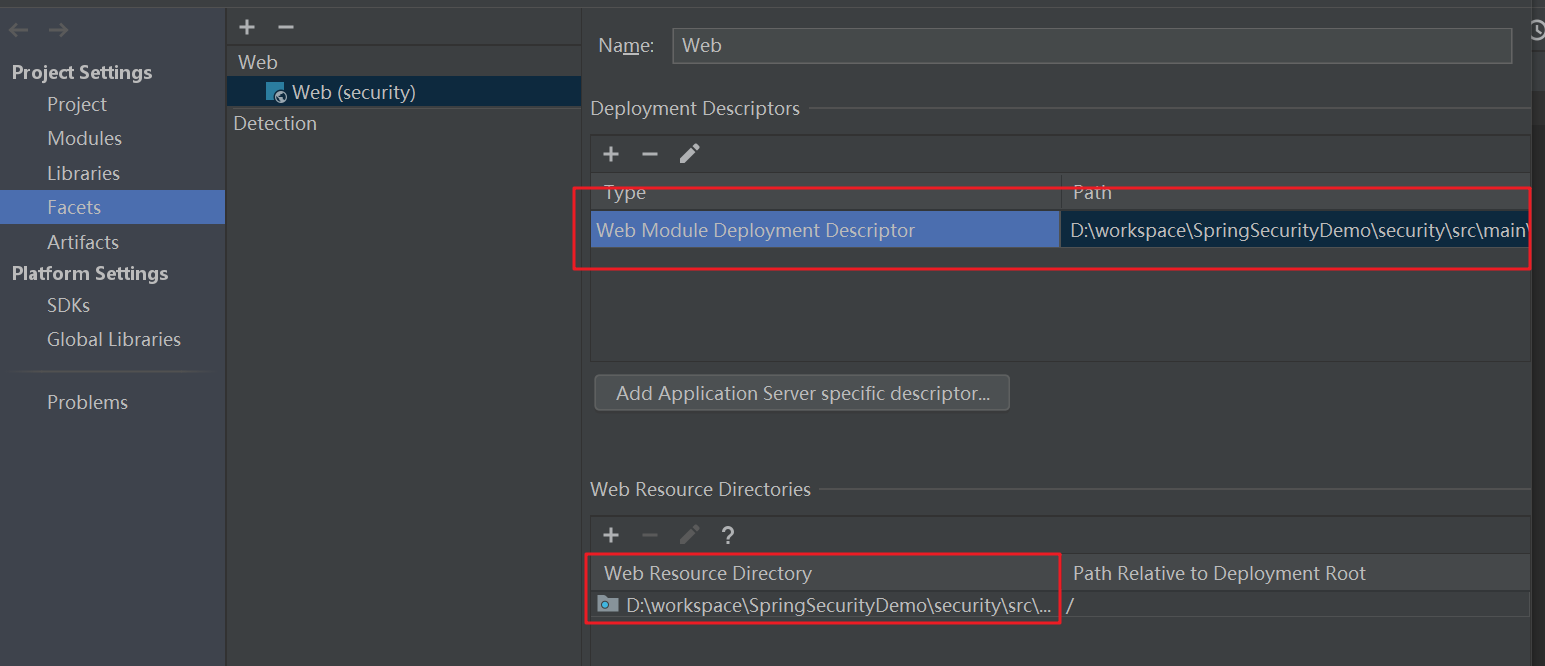
add framework support创建的web目录是与src并列的目录，要放到main目录下，与java和resource并列



4.将web目录修改为webapp目录



5.指定工程的web.xml位置和web目录



## 依赖引入

暂时先引入SpringMVC的依赖

web工程打war包

<groupId>com.atguigu.security</groupId>  
<artifactId>security</artifactId>  
<version>1.0-SNAPSHOT</version>  
<packaging>war</packaging>

<dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>4.3.20.RELEASE</version>  
 </dependency>  
 <!-- 引入 Servlet 容器中相关依赖 -->  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>servlet-api</artifactId>  
 <version>2.5</version>  
 <scope>provided</scope>  
 </dependency>  
 <!-- JSP 页面使用的依赖 -->  
 <dependency>  
 <groupId>javax.servlet.jsp</groupId>  
 <artifactId>jsp-api</artifactId>  
 <version>2.1.3-b06</version>  
 <scope>provided</scope>  
 </dependency>  
</dependencies>

## 创建配置

1.创建SpringMVC的配置文件 spring-mvc.xml

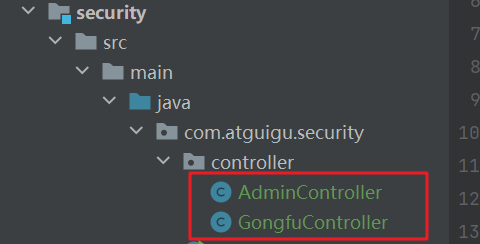
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mvc="http://www.springframework.org/schema/mvc"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">  
 <context:component-scan  
 base-package="com.atguigu.security"></context:component-scan>  
 <bean  
 class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
 <property name="prefix" value="/WEB-INF/views/"></property>  
 <property name="suffix" value=".jsp"></property>  
 </bean>  
 <mvc:annotation-driven></mvc:annotation-driven>  
 <mvc:default-servlet-handler/>  
</beans>

在web.xml中配置DispatcherServelt

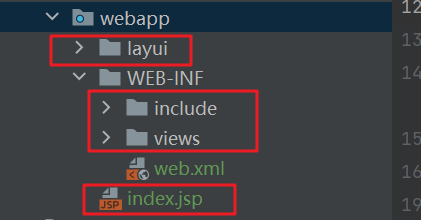
<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"  
 version="4.0">  
  
 <servlet>  
 <servlet-name>springDispatcherServlet</servlet-name>  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 <init-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:spring-mvc.xml</param-value>  
 </init-param>  
 <load-on-startup>1</load-on-startup>  
 </servlet>  
 <!-- Map all requests to the DispatcherServlet for handling -->  
 <servlet-mapping>  
 <servlet-name>springDispatcherServlet</servlet-name>  
 <url-pattern>/</url-pattern>  
 </servlet-mapping>  
  
</web-app>

## 基础类

从例子工程中复制



webapp资源



## 引入SpringSecurity

<!-- SpringSecurity 对 Web 应用进行权限管理 -->  
<dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-web</artifactId>  
 <version>4.2.10.RELEASE</version>  
</dependency>  
<!-- SpringSecurity 配置 -->  
<dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-config</artifactId>  
 <version>4.2.10.RELEASE</version>  
</dependency>  
<!-- SpringSecurity 标签库 -->  
<dependency>  
 <groupId>org.springframework.security</groupId>  
 <artifactId>spring-security-taglibs</artifactId>  
 <version>4.2.10.RELEASE</version>  
</dependency>

## 权限控制的filter

SpringSecurity使用的是过滤器Filter而不是拦截器Interceptor，意味着SpringSecurity能够管理的不仅仅是 SpringMVC 中的 handler 请求，还包含 Web 应用中所有请求。比如：项目中的静态资源也会被拦截，从而进行权限控制。

**特 别 注 意** ： springSecurityFilterChain 标签中必须是springSecurityFilterChain。因为 springSecurityFilterChain 在 IOC 容器中对应真正执行权限控制的二十几个 Filter，只有叫这个名字才能够加载到这些 Filter。

## 加配置类

Enable 理解为启用。

@EnableWebSecurity 注解表示启用 Web 安全功能。

以后会接触到很多@EnableXxx 注解，用来启用对应的功能。

效果 

所有请求都被 SpringSecurity 拦截，要求登录才可以访问。

静态资源也都被拦截，要求登录。 

登录失败有错误提示。

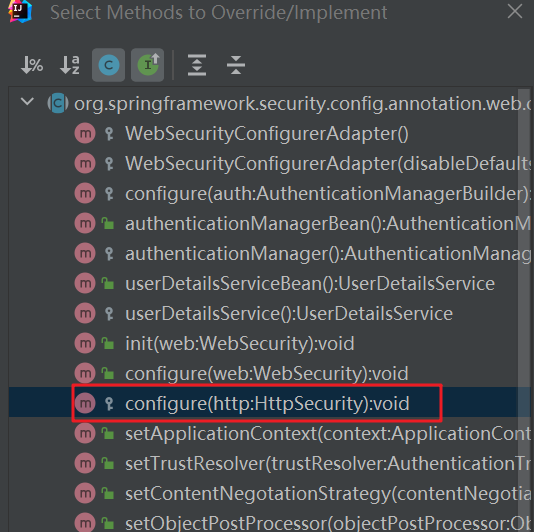
package com.atguigu.security.config;

@Configuration  
@EnableWebSecurity  
public class WebAppSecurityConfig extends WebSecurityConfigurerAdapter {  
 //@EnableWebSecurity 注解表示启用 Web 安全功能。  
   
}

# 使用

## 放行首页和静态资源

需要重写配置类的configure(HttpSecurity http)方法



重写后的配置类

package com.atguigu.security.config;  
  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
  
*/\*\*  
\** ***@author*** *shuyun  
\** ***@date*** *2024-09-08 20:26:42  
\*/*@Configuration  
//@EnableWebSecurity 注解表示启用 Web 安全功能。  
@EnableWebSecurity  
public class WebAppSecurityConfig extends WebSecurityConfigurerAdapter {  
 @Override  
 protected void configure(HttpSecurity security) throws Exception {  
  
 security  
 //对请求进行授权  
 .authorizeRequests()  
 //ant风格设置授权的url地址  
 .antMatchers("/index.jsp","/layui/\*\*")  
 //针对上述路径，可以无条件进行访问  
 .permitAll()  
 .and()  
 //对请求进行授权  
 .authorizeRequests()  
 //任意请求  
 .anyRequest()  
 //需要登录后才可以访问  
 .authenticated()  
 .and()  
 //未授权登录到SpringSecurity的默认登录页，使用表单登录  
 .formLogin()  
 //指定登录页  
 .loginPage("/index.jsp")  
 //loginProcessingUrl 指定提交登录表单的地址  
 //loginProcessingUrl()方法指定了登录地址，就会覆盖 loginPage()方法中设置的默认值 /index.jsp POST  
 .loginProcessingUrl("/login/login.html")  
 //所有人都可以访问  
 .permitAll();  
 }  
}

使用loginPage("index.jsp")后，会影响到：提交登录表单的地址、退出登录地址、登录失败地址

|  |  |  |
| --- | --- | --- |
| 指定前 | /login GET - the login form | 去登录页面 |
| /login POST - process the credentials and if valid authenticate the user | 提交登录表单 |
| /login?error GET - redirect here for failed authentication attempts | 登录失败 |
| /login?logout GET - redirect here after successfully logging out | 退出登录 |
| 指定后 | /index.jsp GET - the login form | 去登录页面 |
| /index.jsp POST - process the credentials and if valid authenticate the user | 提交登录表单 |
| /index.jsp?error GET - redirect here for failed authentication attempts | 登录失败 |
| /index.jsp?logout GET - redirect here after successfully logging out | 退出登录 |

可以使用loginProcessingUrl()。

loginProcessingUrl()方法指定了登录地址，就会覆盖 loginPage()方法中设置的默认值/index.jsp POST

# 最后