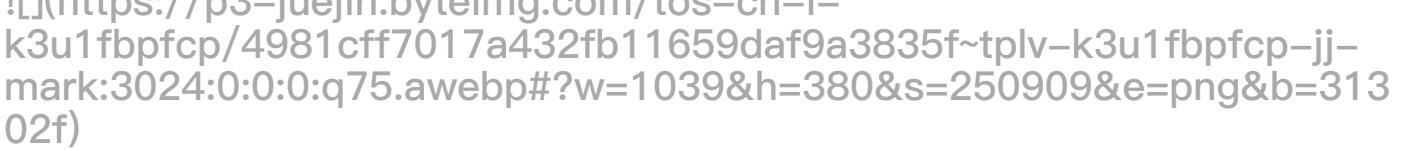
什么时候需要用到 @EnableWebSecurity 注解?

有小伙伴在学习 Spring Security 的遇到一个问题：



箭头所指的位置报红，也就是 Spring 容器中没有找到一个类型为 HttpSecurity 的 Bean。

小伙伴说如果他在配置类上加 @EnableWebSecurity 注解，就不报错；不加该注解则会报错。那么到底该不该加 @EnableWebSecurity 注解呢？今天就来和大伙聊一聊这个话题。

一 @EnableWebSecurity

首先我们来说下 @EnableWebSecurity 这个注解。从名字上就能看出来，这个注解就是启动 Spring Security 的，我们来看下这个注解的定义：

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
@Documented
@Import({ WebSecurityConfiguration.class,
SpringWebMvcImportSelector.class, OAuth2ImportSelector.class,
HttpSecurityConfiguration.class })
@EnableGlobalAuthentication
public @interface EnableWebSecurity {

/**
 * Controls debugging support for Spring Security. Default is false.
 * @return if true, enables debug support with Spring Security
 */
boolean debug() default false;
```

}

```

小伙伴们看到，这个注解的核心能力有两方面：

1. 导入了四个配置类。
2. 启用了 @EnableGlobalAuthentication 注解。

我们先来看这四个配置类。

### 1.1 WebSecurityConfiguration

`WebSecurityConfiguration` 是 Spring Security 框架中的一个核心配置类，它的主要作用是自动配置 Web 安全相关的组件和服务，以确保应用程序的 Web 层安全。

`WebSecurityConfiguration` 主要有如下一些作用：

1. **初始化 WebSecurity**：它创建并配置了 `WebSecurity` 对象，这是一个高级抽象，负责配置安全过滤器链（`FilterChainProxy`），该过滤器链处理所有进入的 HTTP 请求，执行认证、授权、安全头设置、CSRF 保护等安全操作。
2. **自动配置 SecurityFilterChain**：通过 `WebSecurity`，`WebSecurityConfiguration` 会设置一系列的安全过滤器，这些过滤器形成了 SecurityFilterChain，负责处理所有 Web 请求的安全性。
3. **配置默认安全行为**：它提供了默认的安全配置，如允许未认证访问静态资源、设置默认的登录表单和登出逻辑、异常处理等，这些都是大多数 Web 应用的基础安全需求。
4. **支持自定义配置**：尽管提供了默认配置，`WebSecurityConfiguration` 也允许开发者通过扩展 `WebSecurityConfigurerAdapter`（Spring Security5.7及以后版本已被弃用）或直接实现相关接口来自定义安全策略，以满足特定应用的需求。开发者可以覆盖默认的认证、授权规则、安全过滤器等配置。

从这里大家可以看到，`WebSecurityConfiguration` 是 Spring Security 自动配置的核心，它简化了安全配置过程，同时保持了高度的可定制性，是构建安全 Web 应用不可或缺的一部分。
```

1.2 SpringWebMvcImportSelector

SpringWebMvcImportSelector 主要是判断当前是不是 Web 环境，如果是，则返回 WebMvcSecurityConfiguration 对象进行配置，WebMvcSecurityConfiguration 在最新版 Spring Security 中已经废弃了。

1.3 OAuth2ImportSelector

OAuth2ImportSelector 主要是检查当前项目有没有使用 OAuth2 Client，如果使用了，则返回和 OAuth2 Client 相关的配置类完成自动配置。

1.4 HttpSecurityConfiguration

HttpSecurityConfiguration 是提供了默认的 HttpSecurity 对象，关于 HttpSecurity 对象松哥在前面的文章中已经详细介绍过了。[HttpSecurity 是如何组装过滤器链的](http://cxyroad.com/”https://mp.weixin.qq.com/s/hGOhqla5xhqJWaIhlBwcYQ”)。

1.5 @EnableGlobalAuthentication

这个注解主要是为了导入 AuthenticationConfiguration 配置，AuthenticationConfiguration 配置中主要是配置了一些全局的认证器，因为 Spring Security 中的认证器有全局和局部之分（感兴趣的小伙伴可以了解下松哥的[Spring Security6+OAuth2 视频精讲](http://cxyroad.com/”https://mp.weixin.qq.com/s/a7CvhHcnRuJQ-gEDFQNhsQ”)）。

从上面的代码中我们可以看到，@EnableWebSecurity 注解存在的意义，主要是就是开启了 Spring Security 的一些默认配置，相当于是一件启用 Spring Security。

二 加还是不加?

一键启用 Spring Security，在没有 Spring Boot 的年代，这确实是个好玩意！

但是！！！

现在我们都是使用 Spring Boot，相信各位小伙伴使用 Spring Security 的场景基本上也都是在 Spring Boot 项目中，而 Spring Boot 中关于 Spring Security 提供了自动化配置类 SecurityAutoConfiguration，当我们分析 SecurityAutoConfiguration 的源码时候，发现其实它里边最终也是调用了 @EnableWebSecurity。

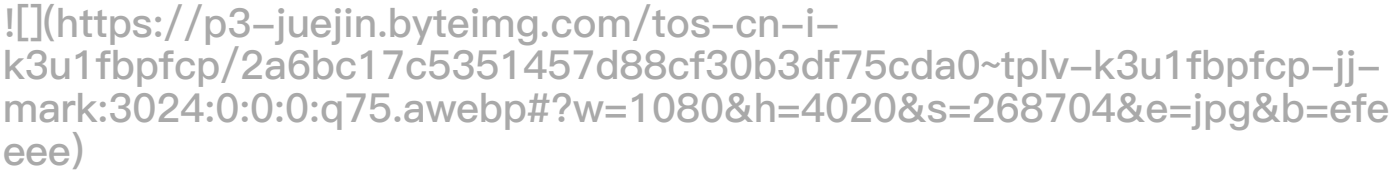
所以，在 Spring Boot 项目中使用 Spring Security 的话，一般是不需要自己手动添加 @EnableWebSecurity 注解的，默认的自动化配置类已经帮我们启用了该注解。

三 问题答案
======

回到一开始的问题。小伙伴遇到这种问题，一般是两种可能：

1. IDEA 检测问题，IDEA 提示不能全信，对于这样的报错问题，如果确认自己代码没问题，那就大胆去运行，可能是 IDEA 误报。
2. 当前配置类没有被 Spring 容器扫描到，即当前配置类不在 Spring 容器中，但是却尝试注入 Spring 容器中的 Bean，于是就有了错误提示。当添加了 @EnableWebSecurity 注解之后，IDEA 检测到有 HttpSecurity 对象导入了，就不报错了，但是问题在根源其实在于当前类没有被注入到 Spring 容器中。

后来证明是第二种原因。



欢迎各位小伙伴一起来精进 Spring Security～

原文链接: https://juejin.cn/post/7376617794424930345