# Spring

## Tomcat，Spring, SpringMVC几种容器的关系

从功能上： tomcat大体上有两部分组成，连接器和容器，连接提供端口并负责请求接入，把端口进来的传输层消息通过协议解析转换成指定的协议类型，如http, http2等, 然后把解析出的消息传递给后面的容器；Spring主要负责把相关的bean生成并管理起来； SpringMVC负责把http的url和对应的Controller映射起来。

以一个http请求为例来说明一下， http请求通过tomcat提供的端口接入，tomcat负责把从端口获取的TCP码流按照HTTP协议解析并最终封装成ServletRequest对象，ServletRequest对象最终都会经过Dispatcherservlet, DispatcherServlet会根据请求的url从缓存SpringMVC容器中缓存的url和Conreoller映射关系找到对应的Controller（在服务启动的过程中已经缓存了url和Controller的映射关系）。

从容器关系上： tomcat作为web容器，实现了Servlet规范， 本质上是Servlet容器。Servlet规范中ServletContext表示web应用的上下文环境，而web应用在tomcat中对应的的概念是Context，所以ServletContext自然会成为tomcat的Context具体实现的一个成员变量； Spring上下文使用ApplicationContext表示， 是在tomcat的一个监听tomcat容器启动的监听器中（ContextLoaderListener）初始化完成的， 初始化成功之后会成为ServletContext的一个属性， 所以可以认为Spring容器属于tomcat的子容器； SpringMVC容器是在DispatcherServlet的init方法中启动的，启动完成之后会设置自己的父容器为Spring容器，即SpringMVC容器和Spring容器是父子关系。

tomcat

引用

Parent属性

SpringMVC容器

引用

Spring容器

ServletContext属性

连接器

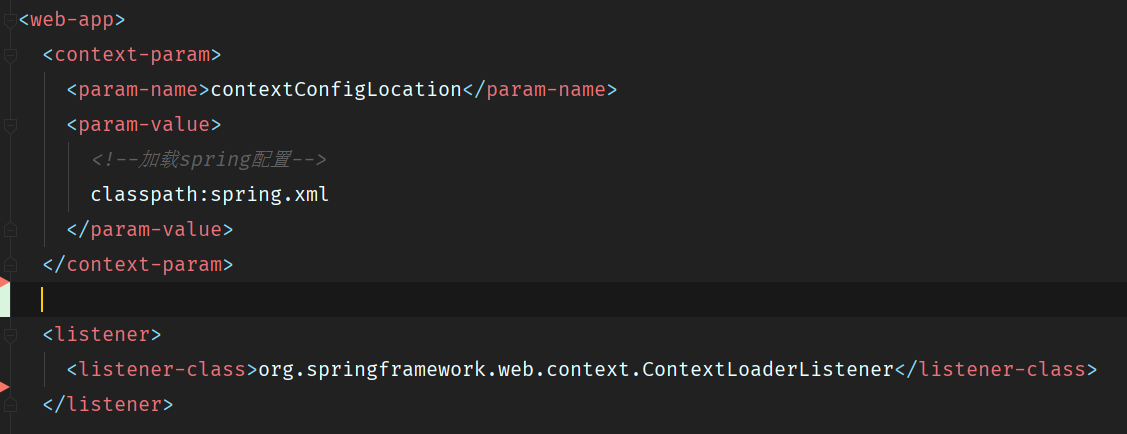
Context容器（省略了engine和host）

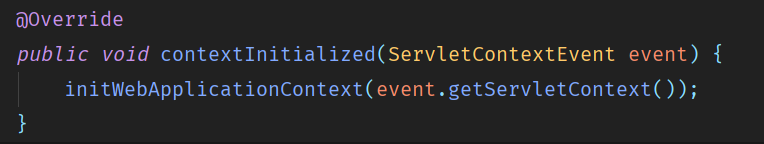
## Spring启动流程源码分析

不同的开发方式对应的启动流程也有差异，最终都会调用到Spring应用上下文ApplicationContext的refresh方法来完成初始化。注：以下代码只保留主流程

### 非springboot且有web.xml配置场景

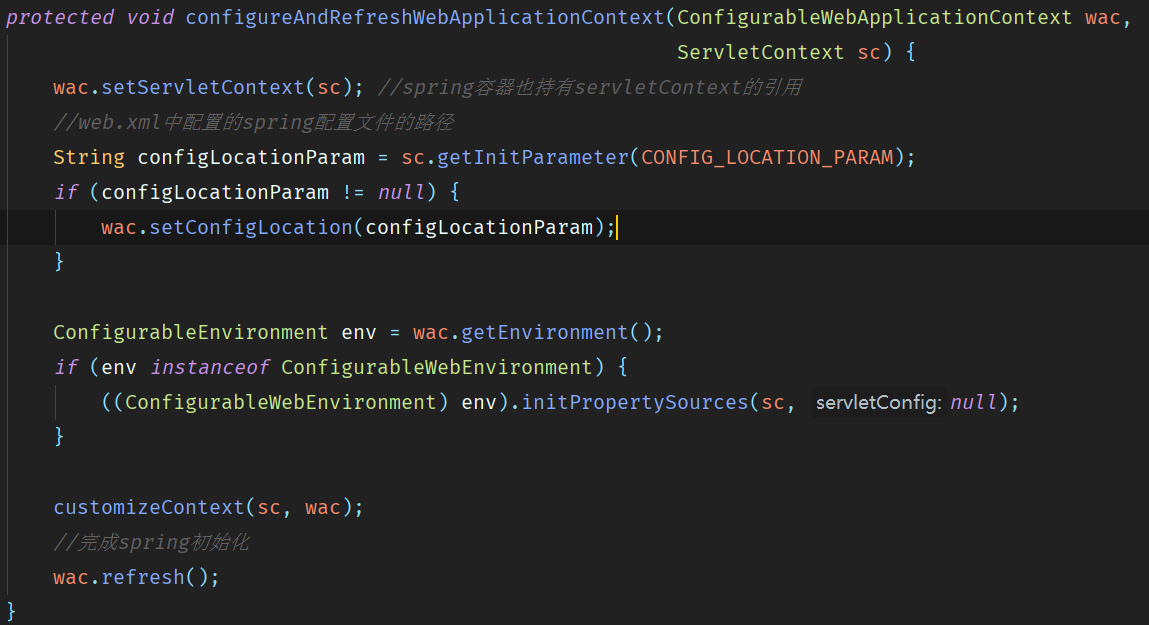
早期的web应用在webapp/WEB-INF/web.xml中可以配置web应用的Servlet, Listener, Filter等。如下为ContextLoaderListener的配置，容器在启动之后，该监听器感知到web容器启动，会调用该监听器的contextInitialized方法，spring容器正是在这个方法中完成初始化的。







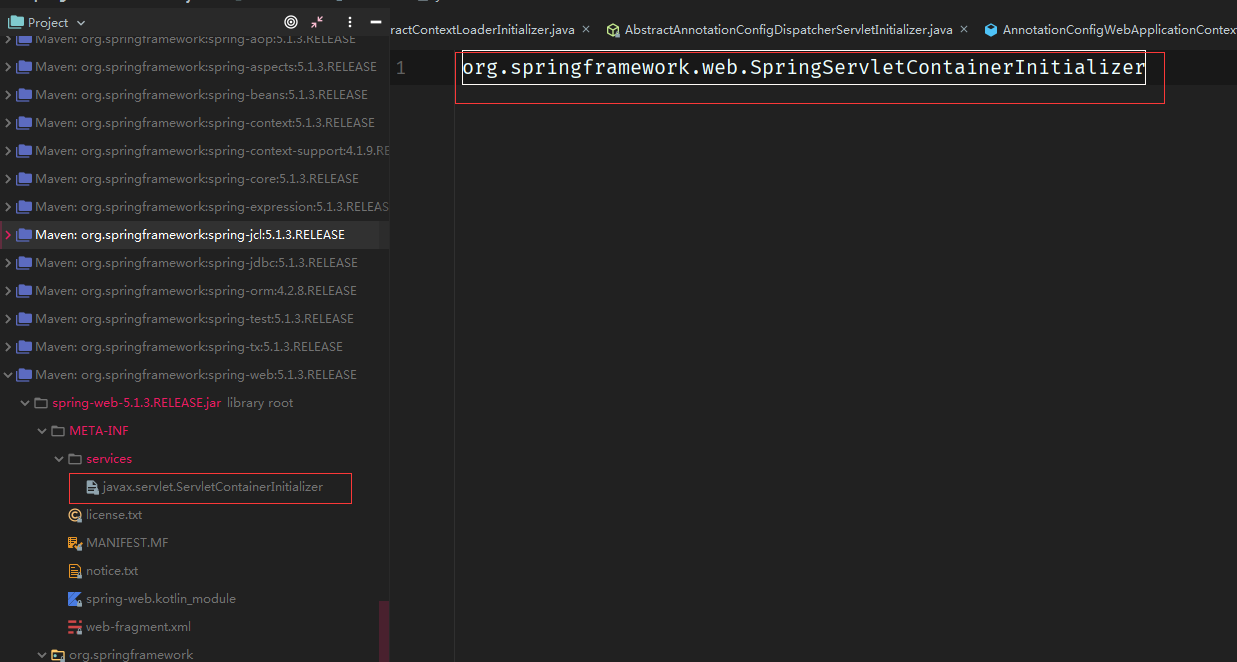
Refresh方法的调用流程



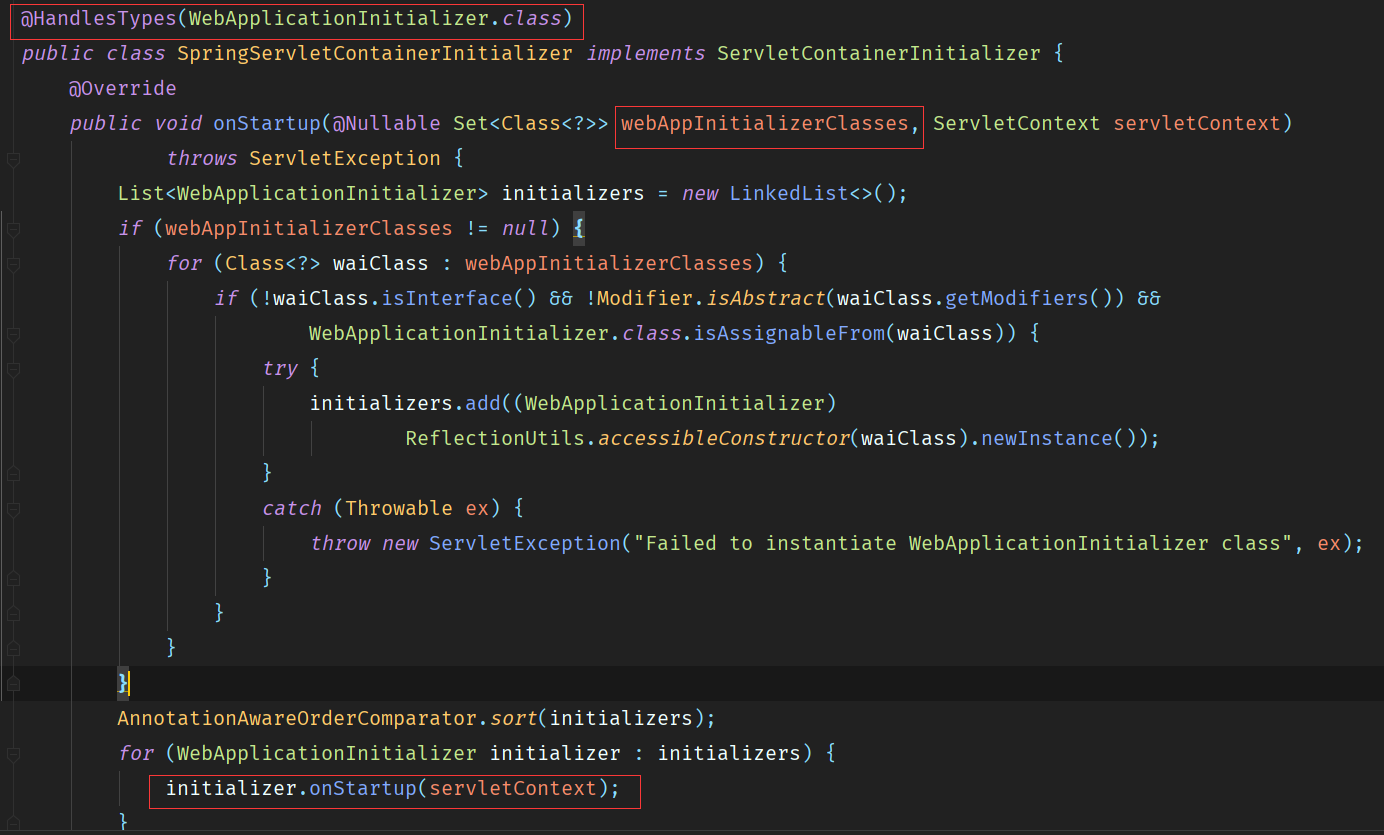
### 非springboot无web.xml配置场景

当前已经很少使用web.xml了，对于spring容器的初始化来说，是否有web.xml配置的差别是ContextLoaderListener是谁来初始化的， 有web.xml的时候是tomcat通过解析web.xml配置来初始化， 无web.xml的时候spring体系自身要完成ContextLoaderListener的初始化。ContextLoaderListener初始化spring的流程是一样的，此处仅分析ContextLoaderListener本身的初始化。

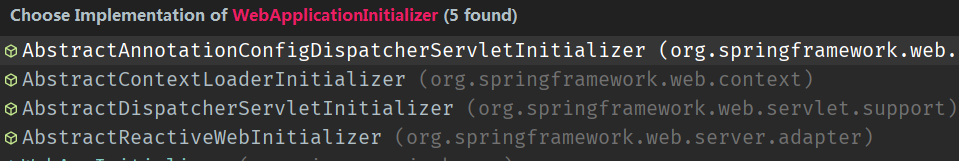
整体初始化流程： spring在spring-web这个依赖的META-INF/services目录下有个配置文件，文件里面有初始化类， tomcat通过SPI机制读取配置文件并加载对应的配置类SpringServletContainerInitializer（tomcat会自动读取所有jar包的META-INF/services目录并自动加载对应的类），调用配置类SpringServletContainerInitializer的onStartup方法进行初始化。



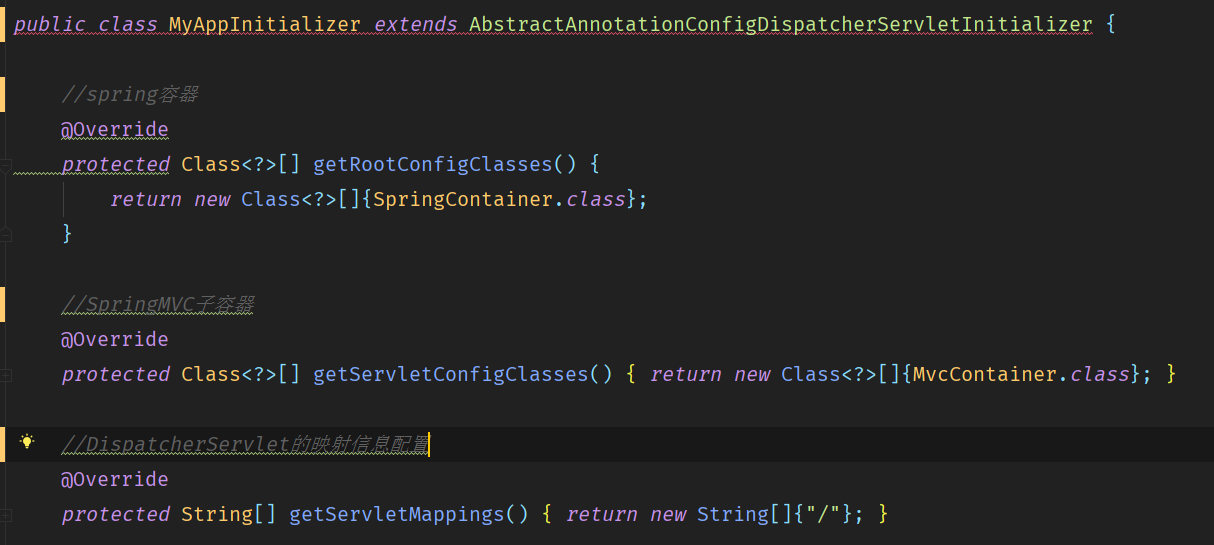
打开SpringServletContainerInitializer类如下，类上有个注解@HandlesTypes, 里面指定了WebApplicationInitinalizer接口类，tomcat会收集所有实现了WebApplicationInitinalizer的类，并作为SpringServletContainerInitializer.onStartup方法的入参. 再看该方法最后会调用WebApplicationInitinalizer实现类的onStartup方法。因此接下来只需要看WebApplicationInitinalizer的实现类的onStartUp方法中干了什么就可以了。



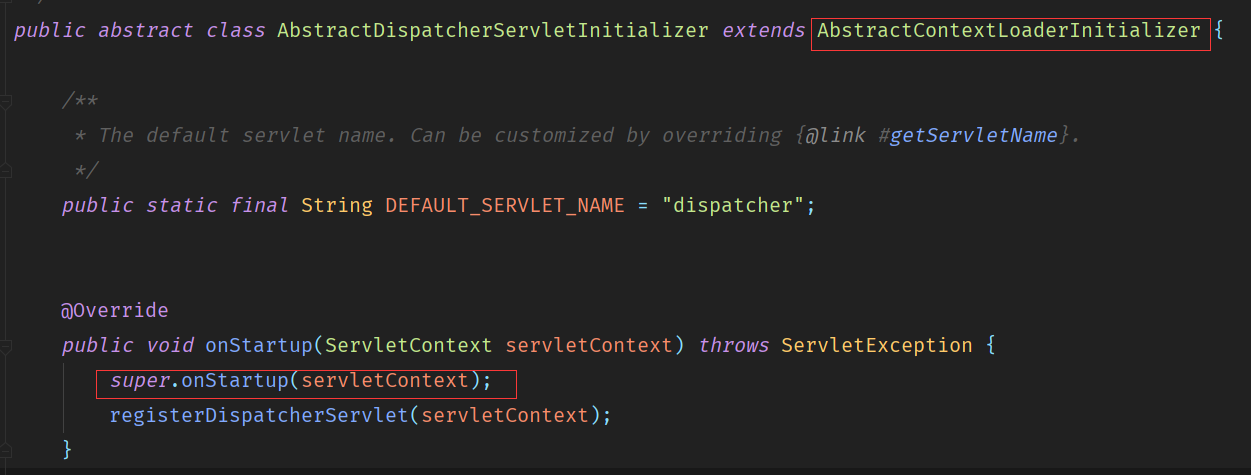
WebApplicationInitinalizer的子类如下，可以看到都是抽象类， 因此需要业务自身根据需要去实现。 其中AbstractDispatcherServletInitializer会初始化spring以及springmvc容器，实际使用中基本使用的是这个，我们把重点放到这个类上面，AbstractDispatcherServletInitializer还有一个子抽象类AbstractAnnotationConfigDispatcherServletInitializer，继承他可以自定义spring容器以及springMVC容器的扫描范围。

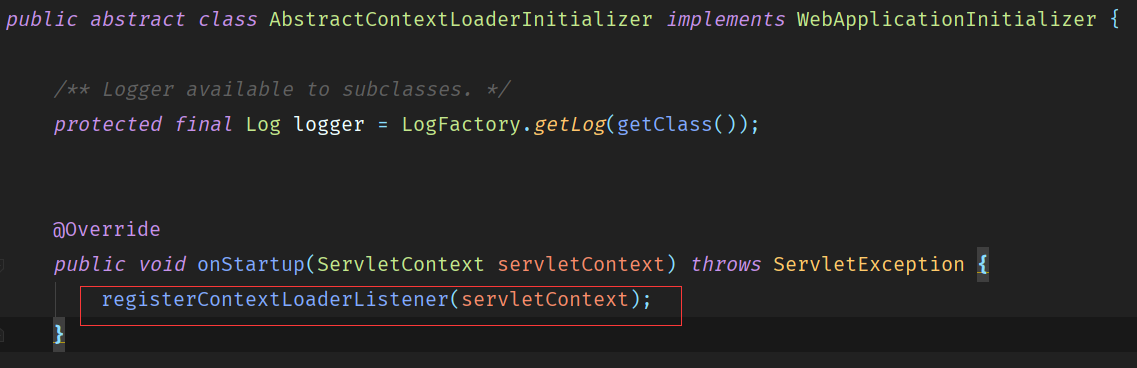


自定义业务类实现如下，自定义的方法会被父类钩子方法调用，此处不做过多深入分析。

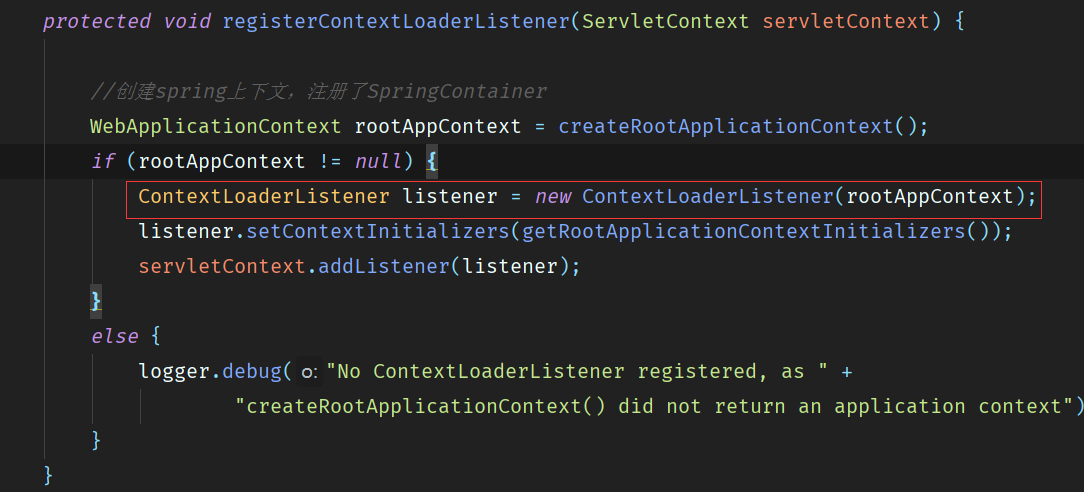


接下来回到正题，ContextLoadListener如何实例化，AbstractDispatcherServletInitializer.onStartup方法如下，会调用AbstractContextLoaderInitializer.onStartup：

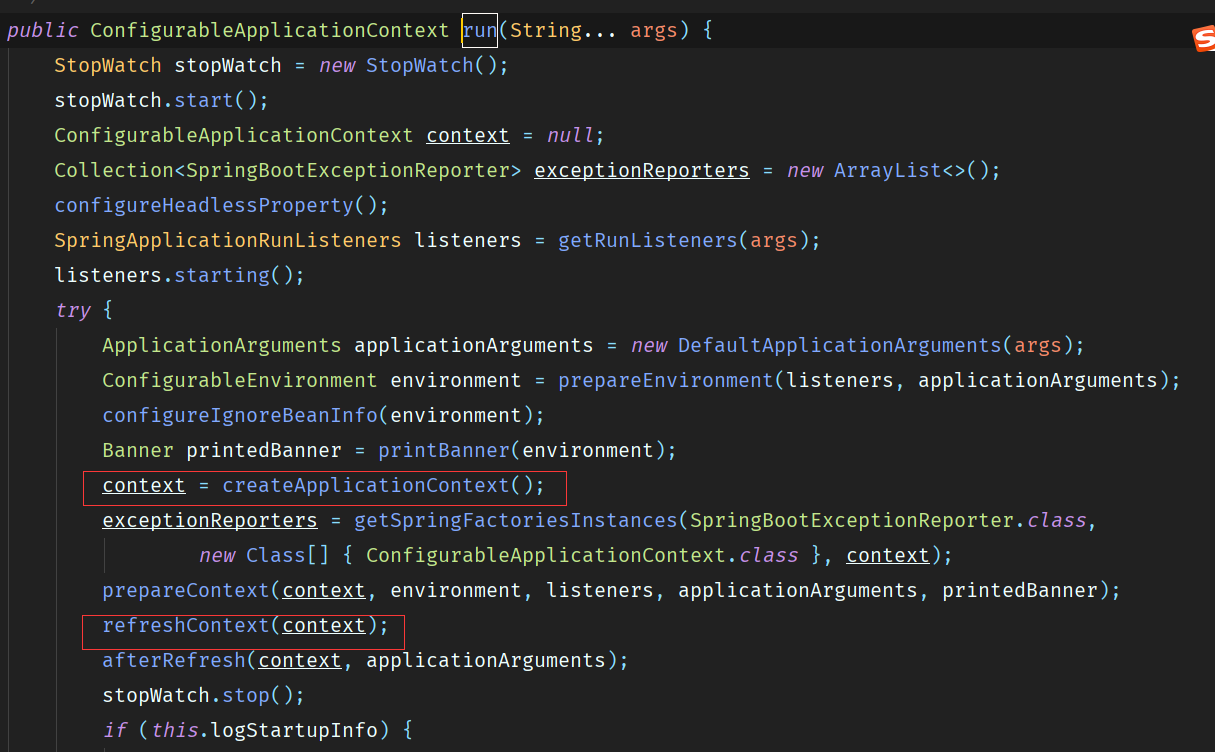




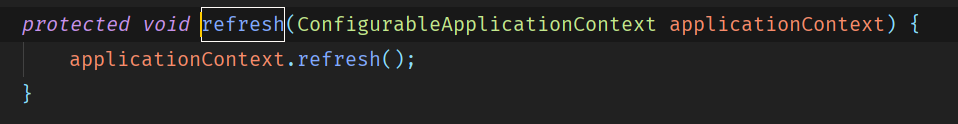
监听器创建如下,至此，监听器创建完成，接下来监听器的初始化和web.xml一致，不再赘述。



### Springboot启动场景

当前我们基本使用的都是这种场景， 这种场景spring容器的初始化是在应用的main方法中启动的，SpringApplication.run方法一直往下点击，会进入如下方法，

refreshContext方法最终会调用到ApplicationContext的refresh方法，从而完成spring的初始化



## Spring初始化流程源码分析