Springboot是如何启动的？

·SpringApplication 初始化

1. 审查ApplicationContext类型
2. 加载ApplicationContextInitializer
3. 加载ApplicationListener

小结：完成初始化后，可以知道如下信息

ApplicationContext是Web或是其他类型

SpringApplication中有一些ApplicationContextInitializer实现类

SpringApplication中有一些ApplicationListener实现类

·Environment初始化

1. 解析命令行参数
2. 创建Environment
3. 配置Environment
4. 配置SpringApplication

小结：环境准备阶段大致工作

根据WebApplicationType枚举创建环境

设置ConversionService用于属性转换

将命令行参数args添加到环境

将外部设置的Profiles添加到环境

绑定SpringApplication属性

发送环境Prepared事件

·ApplicationContext初始化

1. 创建ApplicationContext
2. 设置ApplicationContext
3. 刷新ApplicationContext

·程序运行入口

文件：spring.factories讲解

一个properties文件，在classPath:/META-INF/目录下，每个jar包都有此文件。

String的工具类SpringFactoriesLoader加载，解析此文件

内容：

Key是完整接口名

Value是以逗号间隔完整类型列表，且是key的实现类

**审查ApplicationContext类型**

通过枚举WebApplicationType的静态方法判断，WebApplicationType的三个枚举值：NONE（不是web程序）、SERVLET（基于Servlet的Web程序）、REACTIVE（基于Reactive的web程序）

WebApplicationType枚举中的成员变量存在着完整的class类型。

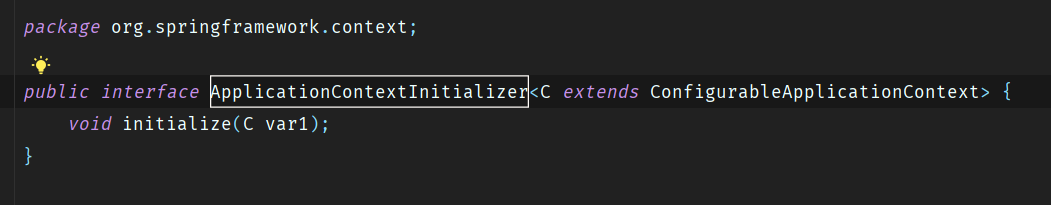
例如：通过pom.xml文件引入spring-boot-starter-web，那classPath下就会由ConfigurableWebApplicationContext和javax.servlet.Servlet类，这样就决定了 程序的ApplicationContext的类型是Servlet

**加载ApplicationContextInitializer**

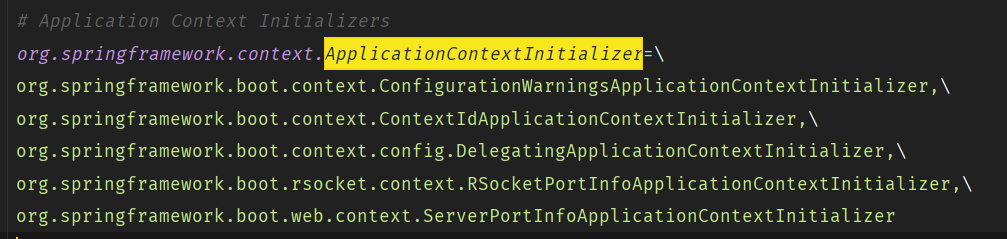
在刷新context之前，做一些额外的工作：添加PropertySource、设置ContextId等，只有一个方法。ApplicationContextInitializer是在spring容器刷新之前执行的一个回调函数。

作用：向springboot容器中注册属性；比如注册一些属性资源额激活文件配置到Environment的环境变量中。

支持Ordered接口和@Order注解

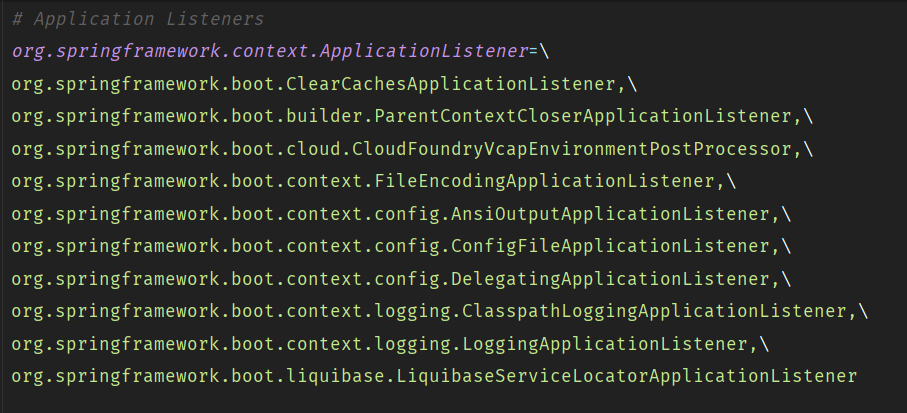


Spring通过SpringFactoriesLoader加载spring.factories的配置，读取key为ApplicationContextInitializer的value（实现类），一系列需要初始化的类



**加载ApplicationListener**

ApplicationListener 用于监听ApplicationEvent事件，它的初始加载流程与ApplicationContextInitializer类似，在spring.factories中找到相应的key



**解析命令行参数**

由main方法的args参数传递

**创建Environment**

在SpringApplication类中



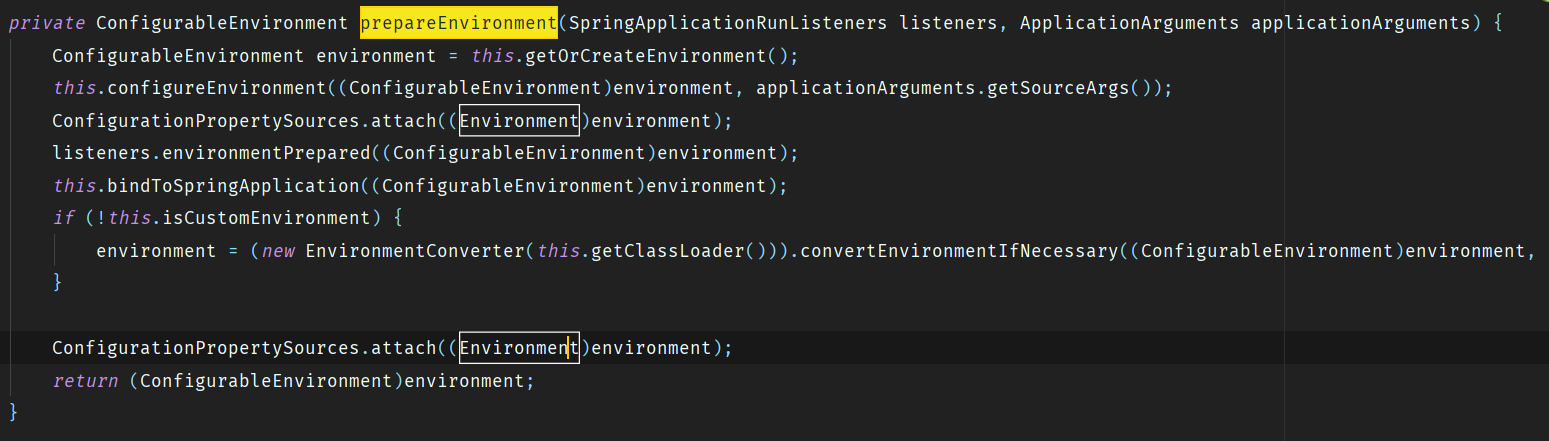
**准备Environment**

环境大致由Profile和PropertyResolver组成：

Profile：是BeanDefinition的逻辑分组，定义Bean时可以指定Profile使SpringBoot在运行时会根据Bean的Profile决定会否注册Bean

PropertyResolver：专门解析属性，SpringBoot会在启动时加载配置文件、系统变量等属性

调用SpringApplication中的prepareEnvironment方法



大致完成：

创建环境、配置环境、设置SpringApplication的属性

**配置Environment**

通过方法configureEnvironment 和 configurePropertySources

设置ConversionService：用于属性转换

将命令行参数添加到环境中

添加额外的ActiveProfiles

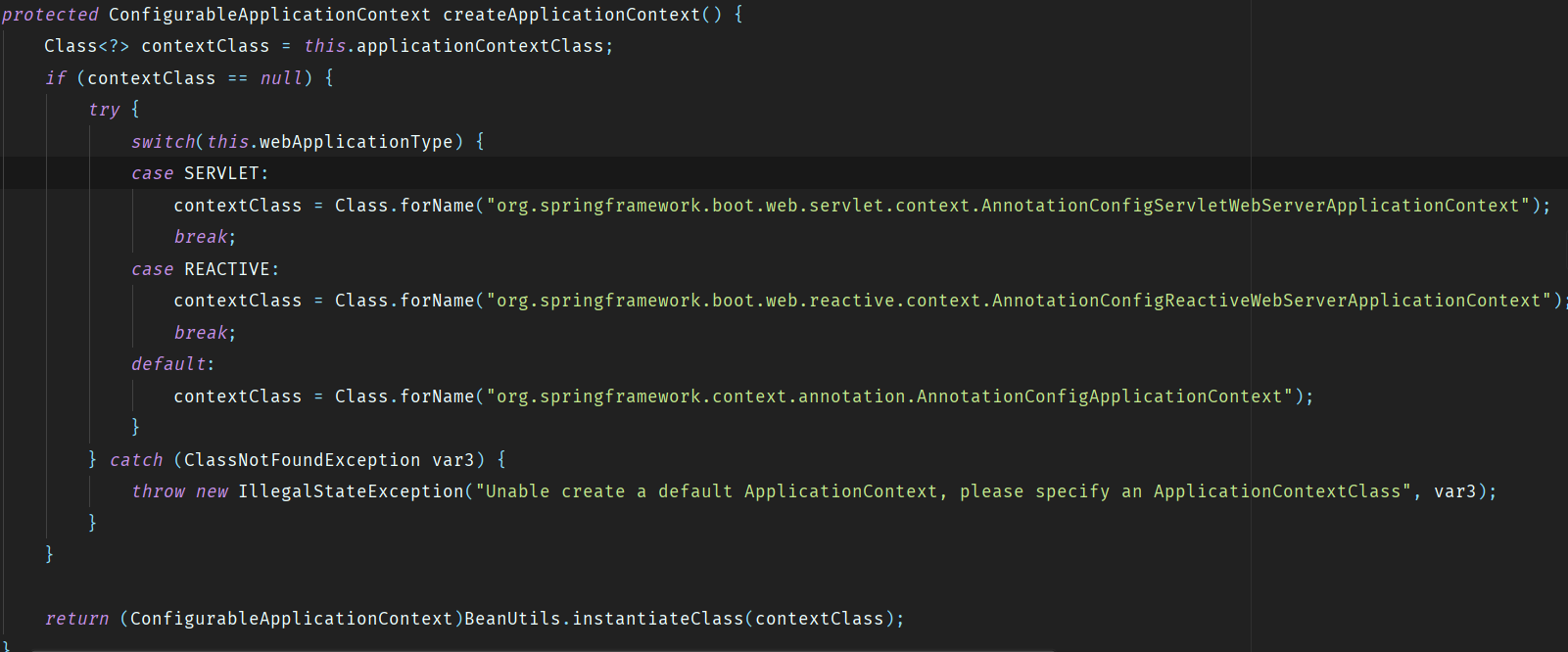
**SpringApplication设置属性**

将已有的属性连接到SpringApplication实例，Springboot会将前缀为spring.main的属性绑定到SpringApplication实例中

**创建ApplicationContext**

前面的工作基本是为了准备**ApplicationContext**而做的，**ApplicationContext**提供了加载Bean、加载资源、发送事件等功能；

根据WebApplicationType来创建不同的ApplicationContext



**准备ApplicationContext**

创建完成后需要初始化，设置环境、应用ApplicationContextInitailizer、注册Source类等信息，归纳如下：

·为ApplicationContext设置环境（之前创建的环境）

·基础设置、操作设置 BeanNameGenerator、ResourceLoader、ConversionService

·执行ApplicationContextInitializer的initialize方法

·注册命令行参数（SpringApplicationArguments）

·注册banner（SpringBootBanner）

·注册sources（由@Configuration注解的类），原理是通过解析注解元数据，创建BeanDefination然后将它注册到ApplicationContext中

**刷新ApplicationContext**

准备刷新

初始化BeanFactory

执行BeanFactoryPostProcessor

注册BeanPostProcessor

初始化MessageSource

初始化事件广播

注册ApplicationListeners