Spring项目启动后的加载流程

ContextLoaderListener通过调用继承自ContextLoader的initWebApplicationContext方法实例化SpringIoc容器。

在实例化Spring IoC容器的过程中，最主要的两个方法是createWebApplicationContext和configureAndRefreshWebApplicationContext方法。

createWebApplicationContext方法用于返回XmlWebApplicationContext实例，即Web环境下的SpringIoC容器。

configureAndRefreshWebApplicationContext用于配置XmlWebApplicationContext，读取web.xml中通过contextConfigLocation标签指定的XML文件，通过调用refresh来调用AbstractApplicationContext中的refresh初始化：

1. BeanFactory并实例化XML文件中配置的bean，Spring将配置文件的bean的信息解析成为一个个的BeanDefinition对象并装入到容器的Bean定义注册表，但此时Bean还未初始化；obtainFreshBeanFactory()会调用自身的refreshBeanFactory(),而refreshBeanFactory()方法由子类AbstractRefreshableApplicationContext实现，该方法返回了一个创建的DefaultListableBeanFactory对象，这个对象就是由ApplicationContext管理的BeanFactory容器对象。

2. 调用工厂后处理器：根据反射机制从BeanDefinitionRegistry中找出所有BeanFactoryPostProcessor类型的Bean，并调用其postProcessBeanFactory()接口方法

经过第一步加载配置文件，已经把配置文件中定义的所有bean装载到BeanDefinitionRegistry这个Beanfactory中，对于ApplicationContext应用来说这个BeanDefinitionRegistry类型的BeanFactory就是Spring默认的DefaultListableBeanFactory。容器扫描BeanDefinitionRegistry中的BeanDefinition，使用java反射自动识别出Bean工厂后处理器的bean，然后调用这些bean工厂后处理器对BeanDefinitionRegistry中的BeanDefinition进行加工处理，具体步骤包括：Resource定位、BeanDefinition的载入、向IOC容器注册这些BeanDefinition：

1. 对使用到占位符的<bean>元素标签进行解析，得到最终的配置值，这意味着对一些半成品式的BeanDefinition对象进行加工处理并取得成品的BeanDefinition对象。、

2. 对BeanDefinitionRegistry中的BeanDefinition进行扫描，通过Java反射机制找出所有属性编辑器的Bean，并自动将它们注册到Spring容器的属性编辑器注册表中

3. 注册Bean后处理器：根据反射机制从BeanDefinitionRegistry中找出所有BeanPostProcessor类型的Bean，并将它们注册到容器Bean后处理器的注册表中；

4. 初始化消息源：初始化容器的国际化信息资源；

5. 初始化应用上下文事件广播器

6. 初始化其他特殊的Bean

7. 注册事件监听器

8. 初始化singleton的Bean：实例化所有singleton的Bean，并将它们放入Spring容器的缓存中

9. 发布上下文刷新事件：在此处时容器已经启动完成，发布容器refresh事件创建上下文刷新事件，事件广播器负责将些事件广播到每个注册的事件监听器中。

依赖注入的原理

依赖注入的过程是用户第一次向IoC容器索要Bean时触发的。当用户向IoC容器索要Bean时会触发getBean方法，之后使用CGLIB通过反射生成Bean对象，之后将这些对象的依赖关系通过根据BeanDefinition的信息完成两个递归从而完成依赖注入。一个递归是在上下文体系中查找需要的Bean和创建Bean的递归调用；另一个递归是在依赖注入时通过递归调用容器的getBean方法得到当前Bean的依赖Bean，同时也触发对依赖Bean的创建和注入