链滴

spring-boot: 简述 springboot 启动流程

作者：Ethan
原文链接：https://ld246.com/article/1524796283527
来源网站：链滴

正文

说springboot的启动流程当然少不了springboot启动入口类

```
@SpringBootApplication
public class SpringBootWebApplication {
    public static void main(String[] args) {
        SpringApplication application = new SpringApplication(SpringBootWebApplication.class)

        application.run(args);
    }
}```
```

以上代码很容易看出哪些是关键,当然是@SpringBootApplication和application.run()分别是springboot加载配置和启动,下面详细说明这两块。

1.SpringBootApplication的背后

@Target(ElementType.TYPE)

@Retention(RetentionPolicy.RUNTIME)

@Documented

@Inherited

@SpringBootConfiguration

@EnableAutoConfiguration

@ComponentScan(excludeFilters = {

@Filter(type = FilterType.CUSTOM, classes = TypeExcludeFilter.class),

@Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExcludeFilter.class) })

public @interface SpringBootApplication {
}```
其中@Configuration(@SpringBootConfiguration中其实用的也是@Configuration)；@EnableAuoConfiguration；@ComponentScan三个是最重要的注解,@SpringBootApplication整合了三个注使用者写起来看起来都比较简洁。

1.1 @Configuration

它就是JavaConfig形式的Spring Ioc容器的配置类使用的那个@Configuration，这里的启动类标注@Configuration之后，本身其实也是一个IoC容器的配置类。如下案例说明xml和注解实现bean的定义

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd"
      default-lazy-init="true">
      <!--bean定义-->
</beans>```
#### 1.1 @EnableAutoConfiguration


@SuppressWarnings("deprecation")
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@AutoConfigurationPackage
@Import(EnableAutoConfigurationImportSelector.class)
public @interface EnableAutoConfiguration {
```

@EnableAutoConfiguration简单的说它的作用就是借助@Import的支持，收集和注册特定场景相的bean定义。其中，最关键的要属@Import(EnableAutoConfigurationImportSelector.class)，借EnableAutoConfigurationImportSelector，@EnableAutoConfiguration可以帮助SpringBoot应将所有符合条件的@Configuration配置都加载到当前SpringBoot创建并使用的IoC容器。借助于Sprig框架原有的一个工具类：SpringFactoriesLoader的支持很智能的自动配置:

SpringFactoriesLoader其主要功能就是从指定的配置文件META-INF/spring.factories加载配置。将中org.springframework.boot.autoconfigure.EnableutoConfiguration对应的配置项通过反射（Jav Refletion）实例化为对应的标注了@Configuration的JavaConfig形式的IoC容器配置类，然后汇总一个并加载到IoC容器。(如下页面模版的配置)

* * *

1.2 springboot启动简单流程

1.2.1 当我们运行SpringApplication的main方法时,调用静态方法run()首先是实例化,SpringApplicati

n初始化的时候主要做主要做三件事：

* 根据classpath下是否存在(ConfigurableWebApplicationContext)判断是否要启动一个web applicationContext。
* SpringFactoriesInstances加载classpath下所有可用的ApplicationContextInitializer
* SpringFactoriesInstances加载classpath下所有可用的ApplicationListener

1.2.2 SpringApplicatio实例化完成并且完成配置后调用run()方法,首先遍历初始化过程中加载的SprinApplicationRunListeners，然后调用starting(),开始监听springApplication的启动。

1.2.3 加载SpringBoot配置环境(ConfigurableEnvironment)，如果是通过web容器发布，会加载StadardEnvironment。将配置环境(Environment)加入到监听器对象中(SpringApplicationRunListeners。

1.2.4 banner属性的设置

1.2.5 ConfigurableApplicationContext(应用配置上下文)创建，根据webEnvironment是否是web境创建默认的contextClass
,AnnotationConfigEmbeddedWebApplicationContext(通过扫描所有注解类来加载bean)和ConfigrableWebApplicationContext),最后通过BeanUtils实例化上下文对象，并返回。

1.2.5 prepareContext()方法将listeners、environment、applicationArguments、banner等重要件与上下文对象关联。

1.2.6 refreshContext(context),bean的实例化完成IoC容器可用的最后一道工序。

1.2.7 最后springboot做一些收尾工作。自此springboot的简单流程到此结束。