

## Spring Boot的缓存预热如何实现

=====

### 前言

--

在日常开发过程中，我们需要在启动的时候加载一些项目运行中使用比较频繁的数据，为了避免在项目中频繁的查询数据库，给DB造成压力，需要将这些字典数据提前加载到缓存，这样在业务逻辑调用的时候，可以直接从缓存中获取数据；但是我需要在项目一启动的时候，就进行加载，就是缓存预热，让我们一起来研究下Spring boot是如何进行缓存预热的；

### 缓存预热几种方式

-----

#### ### 事件监听

第一种方式就是事件的监听，通过ApplicationReadyEvent，在应用程序启动的时候可以通过监听应用启动事件，将需要加载的缓存数据加载到缓存中；

让我们先来看下这个类的实现，ApplicationReadyEvent继承SpringApplicationEvent是Spring应用启动的时候一个监听器，看它构造方法有个SpringApplication，这个类是Java主方法和Spring应用程序启动的核心类；

![504F45A2-DEE2-4F06-A1FD-CF545A607D22.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/1c7c77f747494dc691d5d171928956fd~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1918&h=730&s=158518&e=png&b=2c2c2c)

在我们Spring boot项目的application启动类里面有个SpringApplication.run方法就是进行应用启动，还是回到ApplicationReadyEvent是Spring Boot框架中的一个事件类，基于事件我们可以在程序初始化的时候，通过监听这个类完成跟随着主程序的初始化，一起进行缓存预热；

通过EventListener注解监听ApplicationReadyEvent类；

![4A4B2C4E-5AD9-48D5-A649-BB50FBC8D853.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/a19e2533372c4923b654e9d7fac945c6~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=930&h=248&s=36532&e=png&b=2c2c2c)

通过启动项目可以看到，在进行上下文刷新的时候，监听配置文件的时候，断点就进来了，这时候可以在这个configuration配置类里面写缓存预热逻辑；

![D4E576FE-D5EC-42D8-99C0-E119C1922FBA.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/9a503d08d0a442928ff0f3dbead4829d~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=2194&h=1198&s=262513&e=png&b=2b2b2b)

### ### CommandLineRunner

还有一个方法是实现CommandLineRunner接口，在项目启动的时候，通过@Component注解，在run方法里写缓存预热的业务逻辑；

可以通过写内部类，实现多个CommandLineRunner接口，定义order先后加载顺序完成缓存预热；

![0E4A2877-CA6F-47DC-A6B2-2B0E16A951E9.png](https://p9-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/70d89078e4b64901b0f485052ceaf6b2~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1868&h=1112&s=287673&e=png&b=2c2c2c)

### ### ApplicationRunner

和CommandLineRunner也是一样的逻辑，实现ApplicationRunner接口，重写run方法，两者肯定是有区别；

\* \*CommandLineRunner\*

入参：先看CommandLineRunner的入参数String... args

类型：字符串，项目启动执行的命令参数

执行：ApplicationRunner之前执行

\* \*ApplicationRunner\*

入参：ApplicationArguments

类型：项目启动更高级别的参数

执行：CommandLineRunner之后执行

\* 执行场景区别

CommandLineRunner更加适合一些简单的场景数据初始化、缓存预热，而ApplicationRunner可以用来初始化一些应用配置

![16F05F21-8B4D-4A57-98D6-50793197DBEA.png](https://p1-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/74d80ee1a12d44b88d77a2c46ac6f5f9~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1682&h=1154&s=295620&e=png&b=2c2c2c)

### ### Spring的InitializingBean

了解Spring的伙伴都知道，在bean的初始化的时候通过InitializingBean的afterPropertiesSet方法在\*Init-method方法\*之前执行初始化；

![CFED7D51-6970-4642-BB3D-C992415FE300.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/353b3309e8b949cb981ecfa867a65e6c~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1722&h=1070&s=153624&e=png&b=2b2b2b)

### 总结

--

这边介绍了几种在项目启动的时候，需要对数据进行预热的几种方式，可以根据预热数据的类型，使用不同的接口进行实现，当然也需要合理使用，如果加载的数据过多，会造成项目启动很慢，可以将一部分优先使用到的数据进行加载，等待项目启动完成运行中在进行另外一部分的加载；

原文链接: <https://juejin.cn/post/7349829638887325734>