

Springboot 优雅停止服务的几种方法

ImportNew 1月12日

(给ImportNew加星标, 提高Java技能)

转自: 博客园, 作者: 黄青石

www.cnblogs.com/huangqingshi/p/11370291.html

在使用 **SpringBoot** 的时候, 都要涉及到服务的停止和启动, 当我们停止服务的时候, 很多时候大家都是kill -9 直接把程序进程杀掉, 这样程序不会执行优雅的关闭。而且一些没有执行完的程序就会直接退出。

我们很多时候都需要安全的将服务停止, 也就是把没有处理完的工作继续处理完成。比如停止一些依赖的服务, 输出一些日志, 发一些信号给其他的应用系统, 这个在保证系统的高可用是非常有必要的。那么咱们就来看一下几种停止 **SpringBoot** 的方法。

第一种就是Springboot提供的actuator的功能, 它可以执行shutdown, health, info等, 默认情况下, actuator的shutdown是disable的, 我们需要打开它。首先引入actuator的maven依赖。

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

然后将shutdown节点打开, 也将/actuator/shutdown暴露web访问也设置上, 除了shutdown之外还有health, info的web访问都打开的话将management.endpoints.web.exposure.include=*就可以。将如下配置设置到application.properties里边。设置一下服务的端口号为3333。

```
server.port=3333
management.endpoint.shutdown.enabled=true
management.endpoints.web.exposure.include=shutdown
```

接下来, 咱们创建一个springboot工程, 然后设置一个bean对象, 配置上PreDestroy方法。这样在停止的时候会打印语句。bean的整个生命周期分为创建、初始化、销毁, 当最后关闭的时候会执行销毁操作。在销毁的方法中执行一条输出日志。

```
package com.hqs.springboot.shutdowndemo.bean;

import javax.annotation.PreDestroy;

/**
 * @author huangqingshi
 * @Date 2019-08-17
 */
public class TerminateBean {

    @PreDestroy
    public void preDestroy() {
        System.out.println("TerminalBean is destroyed");
    }

}
```

做一个configuration，然后提供一个获取bean的方法，这样该bean对象会被初始化。

```
package com.hqs.springboot.shutdowndemo.config;

import com.hqs.springboot.shutdowndemo.bean.TerminateBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * @author huangqingshi
 * @Date 2019-08-17
 */
@Configuration
public class ShutDownConfig {

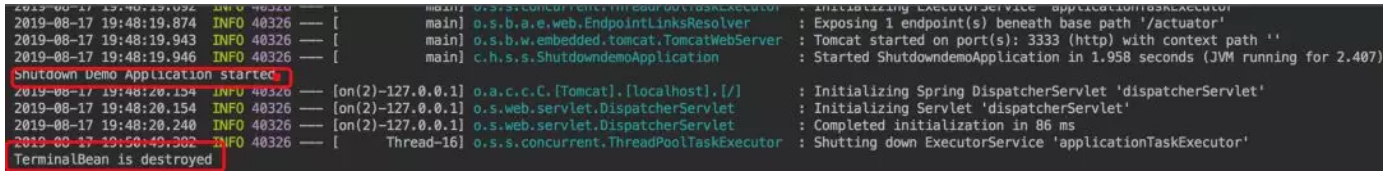
    @Bean
    public TerminateBean getTerminateBean() {
        return new TerminateBean();
    }

}
```

在启动类里边输出一个启动日志，当工程启动的时候，会看到启动的输出，接下来咱们执行停止命令。

```
curl -X POST http://localhost:3333/actuator/shutdown
```

以下日志可以输出启动时的日志打印和停止时的日志打印，同时程序已经停止。是不是比较神奇。



```

2019-08-17 19:48:19.874 INFO 40326 --- [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 1 endpoint(s) beneath base path '/actuator'
2019-08-17 19:48:19.943 INFO 40326 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 3333 (http) with context path ''
2019-08-17 19:48:19.946 INFO 40326 --- [main] c.h.s.s.ShutdownDemoApplication : Started ShutdownDemoApplication in 1.958 seconds (JVM running for 2.407)
Shutdown Demo Application started.
2019-08-17 19:48:20.154 INFO 40326 --- [on(2)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2019-08-17 19:48:20.154 INFO 40326 --- [on(2)-127.0.0.1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2019-08-17 19:48:20.240 INFO 40326 --- [on(2)-127.0.0.1] o.s.web.servlet.DispatcherServlet : Completed initialization in 86 ms
2019-08-17 19:50:49.302 INFO 40326 --- [Thread-16] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService 'applicationTaskExecutor'
TerminalBean is destroyed
  
```

第二种方法也比较简单，获取程序启动时候的context，然后关闭主程序启动时的context。这样程序在关闭的时候也会调用PreDestroy注解。如下方法在程序启动十秒后进行关闭。

```

/* method 2: use ctx.close to shutdown all application context */
ConfigurableApplicationContext ctx = SpringApplication.run(ShutdownDemoApplication.class, args);

try {
    TimeUnit.SECONDS.sleep(10);
} catch (InterruptedException e) {
    e.printStackTrace();
}

ctx.close();
  
```

第三种方法，在springboot启动的时候将进程号写入一个app.pid文件，生成的路径是可以指定的，可以通过命令 `cat /Users/huangqingshi/app.id | xargs kill` 命令直接停止服务，这个时候bean对象的PreDestroy方法也会调用的。这种方法大家使用的比较普遍。写一个start.sh用于启动springboot程序，然后写一个停止程序将服务停止。

```

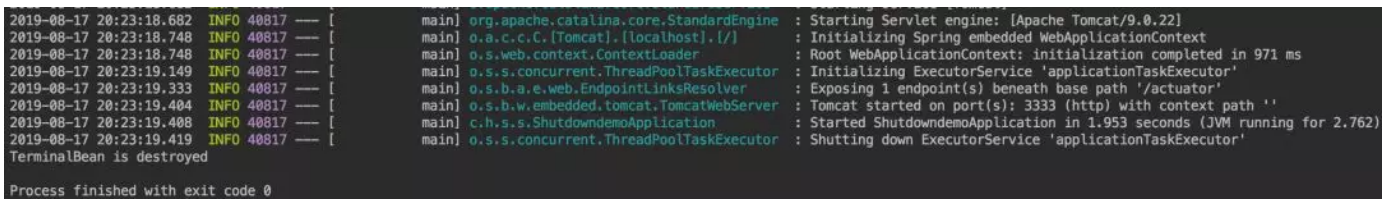
/* method 3 : generate a pid in a specified path, while use command to shutdown */
'cat /Users/huangqingshi/app.pid | xargs kill' */
SpringApplication application = new SpringApplication(ShutdownDemoApplication.class);
application.addListeners(new ApplicationPidFileWriter("/Users/huangqingshi/app.pid"));
application.run();
  
```

第四种方法，通过调用一个SpringApplication.exit() 方法也可以退出程序，同时将生成一个退出码，这个退出码可以传递给所有的context。这个就是一个JVM的钩子，通过调用这个方法的话会把所有PreDestroy的方法执行并停止，并且传递给具体的退出码给所有Context。通过调用

System.exit(exitCode)可以将这个错误码也传给JVM。程序执行完后最后会输出：Process finished with exit code 0，给JVM一个SIGNAL。

```
/* method 4: exit this application using static method */
ConfigurableApplicationContext ctx = SpringApplication.run(ShutdownDemoApplication.class, args);
exitApplication(ctx);

public static void exitApplication(ConfigurableApplicationContext context,
    int exitCode = SpringApplication.EXIT_CODE) {
    System.exit(exitCode);
}
```



```
2019-08-17 20:23:18.682 INFO 40817 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.22]
2019-08-17 20:23:18.748 INFO 40817 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2019-08-17 20:23:18.748 INFO 40817 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 971 ms
2019-08-17 20:23:19.149 INFO 40817 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2019-08-17 20:23:19.333 INFO 40817 --- [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 1 endpoint(s) beneath base path '/actuator'
2019-08-17 20:23:19.404 INFO 40817 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 3333 (http) with context path ''
2019-08-17 20:23:19.408 INFO 40817 --- [main] c.h.s.s.ShutdownDemoApplication : Started ShutdownDemoApplication in 1.953 seconds (JVM running for 2.762)
2019-08-17 20:23:19.419 INFO 40817 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService 'applicationTaskExecutor'
TerminalBean is destroyed
Process finished with exit code 0
```

第五种方法，自己写一个Controller，然后将自己写好的Controller获取到程序的context，然后调用自己配置的Controller方法退出程序。通过调用自己写的/shutDownContext方法关闭程序：curl -X POST http://localhost:3333/shutDownContext。

```
package com.hqs.springboot.shutdowndemo.controller;

import org.springframework.beans.BeansException;
import org.springframework.context.ApplicationContext;
import org.springframework.context.ApplicationContextAware;
import org.springframework.context.ConfigurableApplicationContext;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * @author huangqingshi
 * @Date 2019-08-17
 */
@RestController
public class ShutDownController implements ApplicationContextAware {

    private ApplicationContext context;
```

```
@PostMapping("/shutDownContext")
public String shutDownContext() {
    ConfigurableApplicationContext ctx = (ConfigurableApplicationContext)
    ctx.close();
    return "context is shutdown";
}

@GetMapping("/")
public String getIndex() {
    return "OK";
}

@Override
public void setApplicationContext(ApplicationContext applicationContext) {
    context = applicationContext;
}
}
```

好了，springboot的优雅关闭方法也都实现好了，也有同学问，如何暴力停止呢，简单，直接kill -9 相应的PID即可。

总结一下：

以上这几种方法实现的话比较简单，但是真实工作中还需要考虑的点还很多，比如需要保护暴露的点不被别人利用，一般要加一些防火墙，或者只在内网使用，保证程序安全。

在真实的工作中的时候第三种比较常用，程序中一般使用内存队列或线程池的时候最好要优雅的关机，将内存队列没有处理的保存起来或线程池中没处理完的程序处理完。但是因为停机的时候比较快，所以停服务的时候最好不要处理大量的数据操作，这样会影响程序停止。

好了，大家觉得还没看全的话，可以访问我的GIT代码：

<https://github.com/stonehqs/shutdowndemo.git> 。

推荐阅读 — 点击标题可跳转

[Spring Boot面试问题集锦](#)

[SpringBoot 并发登录人数控制](#)