what will happen if we didn't define @Bean under @Configuration class in spring boot

Asked 3 years, 5 months ago Modified 3 years, 5 months ago Viewed 994 times

0

so usually when we write a class and add @Configuration to the class, we will define bean in that class for example:

```
@Configuration
public class AppConfig {
    @Bean
    public DemoClass service()
    {

    }
}
```

but we I review some codes, I saw some class didn't define @bean method in inside these class,like:

```
@Configuration
public class AutoRefreshConfig {
    @Scheduled(fixedRate = 60000)
    public void update(){
      // update something with a fix rate

    }
}
```

so is this correct? actually it works well. but I am wondering what will happen when I start running the project. what kind of behavior of will spring boot act? Is it just like a normal java class?

java spring spring-boot

Share Improve this question Follow

asked Jul 1, 2020 at 0:05
Hongli Bu
481 13 38

1 When JavaConfig encounters such a method with `@Bean` annotation, it will execute that method and register the return value as a bean within a BeanFactory which is managed by Spring context and that reference can be used anywhere in your application using `@Autowired` configuration. Now to your main question, if you have a configuration class without any method having `@Bean` annotation, then it will simply not register any bean. And it is absolutely fine. – Arindam Jul 1, 2020 at 0:20

2 Answers

Sorted by: Highest score (default)

1

`@Configuration` is a special type of `@Component` where the annotated class can contain bean definitions (using `@Bean`). But if it doesn't contain any bean definition, spring does not throw any exception. In fact, the configuration class can still be used as a bean similar to `@Component` annotated class and can be autowired in dependent classes.

The code referenced above should really be annotated with `@Component` as it does not have bean definition, but since `@Configuration` in itself meta-annotated with `@Component` , it still works. The code is syntactically correct, but it doesn't follow spring convention.

A `@Configuration` is also a `@Component` , but vice versa is not true.

Share Improve this answer Follow

edited Jul 1, 2020 at 6:06
M. Deinum
117k 22 223 226

answered Jul 1, 2020 at 3:16
Kumar V
1,590 1 12 19

1

The `@Scheduled` annotation results in Spring creating a `TaskScheduler` implementation to execute your provided `Runnable` (in this case, the `void update()` method). According to the Spring docs:

> The 'default' implementation is ThreadPoolTaskScheduler, wrapping a native ScheduledExecutorService and adding extended trigger capabilities.

So to answer your question, Spring ultimately uses your annotation to create a `ScheduledExecutorService` , a native executor service in the `java.util.concurrent` package, to execute your task at the desired frequency you provided

Share Improve this answer Follow

answered Jul 1, 2020 at 0:50
bluesky33
69 4