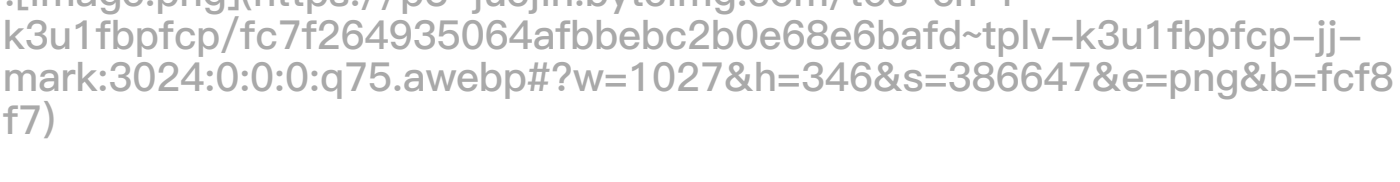
网易面试：SpringBoot如何开启虚拟线程?

虚拟线程（Virtual Thread）也称协程或纤程，是一种轻量级的线程实现，与传统的线程以及操作系统级别的线程（也称为平台线程）相比，它的创建开销更小、资源利用率更高，是 Java 并发编程领域的一项重要创新。

> PS：虚拟线程正式发布于 Java 长期支持版（Long Term Suort，LTS）Java 21（也就是 JDK 21）。

虚拟线程是一种在 Java 虚拟机（JVM）层面实现的逻辑线程，不直接和操作系统的物理线程一一对应，因此它可以减少上下文切换所带来的性能开销。

操作系统线程、普通线程（Java 线程）和虚拟线程的关系如下：


1.虚拟线程使用

虚拟线程的创建有以下 4 种方式：

1. Thread.startVirtualThread(Runnable task)
2. Thread.ofVirtual().unstarted(Runnable task)
3. Thread.ofVirtual().factory()
4. Executors.newVirtualThreadPerTaskExecutor()

具体使用如下。

1.1 startVirtualThread

创建虚拟线程，并直接启动执行任务：

```
// 创建并启动虚拟线程
Thread.startVirtualThread(() -> {
    System.out.println("Do virtual thread.");
});

```

1.2 unstarted

只创建虚拟线程，但不直接启动（创建之后通过 start 启动）：

```
// 创建虚拟线程
Thread vt = Thread.ofVirtual().unstarted(()->{
    System.out.println("Do virtual thread.");
});
// 运行虚拟线程
vt.start();

```

1.3 factory

先创建虚拟线程工厂，然后再使用工厂创建虚拟线程，之后再调用 start() 方法进行执行：

```
// 创建虚拟线程工厂
ThreadFactory tf = Thread.ofVirtual().factory();
// 创建虚拟线程
Thread vt = tf.newThread(()->{
    System.out.println("Do virtual thread.");
});
// 运行虚拟线程
vt.start();

```

1.4 newVirtualThreadPerTaskExecutor

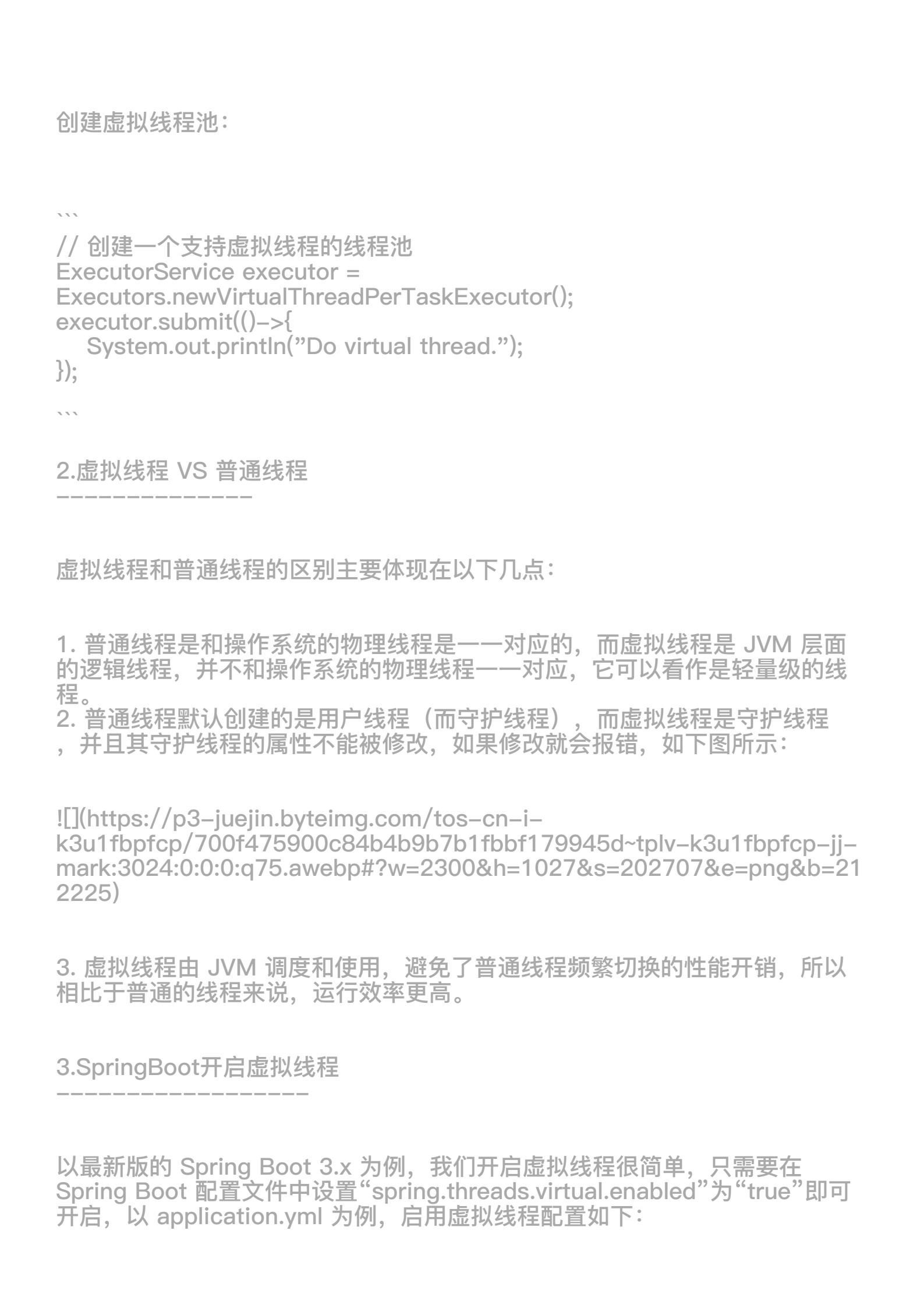
创建虚拟线程池：

```
// 创建一个支持虚拟线程的线程池
ExecutorService executor =
Executors.newVirtualThreadPerTaskExecutor();
executor.submit(()->{
    System.out.println("Do virtual thread.");
});
```

2.虚拟线程 VS 普通线程

虚拟线程和普通线程的区别主要体现在以下几点：

1. 普通线程是和操作系统的物理线程是一一对应的，而虚拟线程是 JVM 层面的逻辑线程，并不和操作系统的物理线程一一对应，它可以看作是轻量级的线程。
2. 普通线程默认创建的是用户线程（而守护线程），而虚拟线程是守护线程，并且其守护线程的属性不能被修改，如果修改就会报错，如下图所示：



3. 虚拟线程由 JVM 调度和使用，避免了普通线程频繁切换的性能开销，所以相比于普通的线程来说，运行效率更高。

3.SpringBoot开启虚拟线程

以最新版的 Spring Boot 3.x 为例，我们开启虚拟线程很简单，只需要在 Spring Boot 配置文件中设置“spring.threads.virtual.enabled”为“true”即可开启，以 application.yml 为例，启用虚拟线程配置如下：

```
spring:
  threads:
    virtual:
      enabled: true # 启用虚拟线程
```

这样 Spinrg Boot 在启动 Tomcat 容器时，会使用一个虚拟线程执行器来代表原有的平台线程池。

> PS：这里是虚拟线程执行器，不是虚拟线程池。

如果以上配置未生效的话，还可以通过修改 Tomcat 配置类，让其使用虚拟线程来处理每一个请求，配置代码如下：

```
import java.util.concurrent.Executors;

import
org.springframework.boot.web.embedded.tomcat.TomcatProtocolHandler
Customizer;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class TomcatConfiguration {
    @Bean
    public TomcatProtocolHandlerCustomizer<?>
protocolHandlerVirtualThreadExecutorCustomizer() {
        return protocolHandler -> {
            // 使用虚拟线程来处理每一个请求
protocolHandler.setExecutor(Executors.newVirtualThreadPerTaskExecuto
r());
        };
    }
}
```

4.异步任务开启虚拟线程

如果你想为 Spring Boot 中的异步任务 @Async 也配置虚拟线程的话，可以在 AsyncConfigurer 配置类中设置，配置代码如下：

```
import java.util.concurrent.Executor;
import java.util.concurrent.Executors;

import org.springframework.context.annotation.Configuration;
import org.springframework.core.task.support.TaskExecutorAdapter;
import org.springframework.scheduling.annotation.AsyncConfigurer;
import org.springframework.scheduling.annotation.EnableAsync;

@Configuration
@EnableAsync  // 开启异步任务
public class AsyncTaskConfiguration implements AsyncConfigurer {
    @Override
    public Executor getAsyncExecutor() {
        return new
TaskExecutorAdapter(Executors.newThreadPerTaskExecutor(Thread.ofVirtual().name("virtual-async#", 1).factory()));
    }
}
```

课后思考

说说虚拟线程的底层实现？有了虚拟线程后还需要虚拟线程池吗？为什么？

> 本文已收录到我的面试小站 [www.javacn.site](http://cxyroad.com/"https://www.javacn.site")，其中包含的内容有：Redis、JVM、并发、并发、MySQL、Spring、Spring MVC、Spring Boot、Spring Cloud、MyBatis、设计模式、消息队列等模块。

原文链接: https://juejin.cn/post/7379147394755100712