# SpringBoot多线程继承线程本地变量，传递MDC

# SpringBoot多线程继承线程本地变量，传递MDC

## 传递MDC方案

通过实现`org.springframework.core.task.TaskDecorator`实现传递MDC。给springboot默认线程池添加上下文，作用于`@Async`，和`@Autowired`注入的默认线程池。ThreadPoolTaskExecutor。
springboot异步和默认线程池自动配置类`org.springframework.boot.autoconfigure.task.TaskExecutionAutoConfiguration`。

```
import org.slf4j.MDC;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.beans.factory.config.BeanDefinition;
import org.springframework.boot.task.ThreadPoolTaskExecutorBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Lazy;
import org.springframework.core.task.TaskDecorator;
import
org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor;

import java.util.concurrent.ThreadPoolExecutor;


@Configuration
public class ContextCopyingToExecutorTaskDecorator implements
TaskDecorator {

    @Bean
    public ThreadPoolTaskExecutor
taskExecutor(ThreadPoolTaskExecutorBuilder
threadPoolTaskExecutorBuilder,
@Value("${spring.threads.virtual.enabled}") Boolean
virtualEnabled,ContextCopyingToExecutorTaskDecorator
```

```
contextCopyingToExecutorTaskDecorator){
        var r = threadPoolTaskExecutorBuilder.build();
        r.setRejectedExecutionHandler(new
ThreadPoolExecutor.CallerRunsPolicy());
        r.setMaxPoolSize(Math.min(r.getMaxPoolSize(),1000));
        r.setTaskDecorator(contextCopyingToExecutorTaskDecorator);
        if(virtualEnabled)r.setThreadFactory(Thread.ofVirtual().factory());
        return r;
    }

    @Override
    public Runnable decorate(Runnable runnable) {
        var mdc = MDC.getCopyOfContextMap();
        if ((mdc==null||mdc.isEmpty()))
            return runnable;
        return ()->{
            try {
                MDC.setContextMap(mdc);
                runnable.run();
            }finally{
                MDC.clear();
            }
        };
    }
}

```

传递线程本地变量
--------

可以直接使用阿里开源的[alibaba/transmittable-thread-local (github.com)](http://cxyroad.com/ "https://github.com/alibaba/transmittable-thread-local")

其是对`java.lang.InheritableThreadLocal`的扩展，原理是在Thread类保存名为inheritableThreadLocals的成员属性，并在初始化创建子线程时，将父线程的inheritableThreadLocals赋给子线程。

```
TransmittableThreadLocal<String> context = new
TransmittableThreadLocal<>();

// ==================================================
```

```
// 在父线程中设置
context.set("value-set-in-parent");

// ============================================================

// 在子线程中可以读取，值是"value-set-in-parent"
String value = context.get();

```
原文链接: https://juejin.cn/post/7366062210932326454