

[🏠 / 开发指南 / 后端手册](#)[👤 芋道源码](#) [📅 2022-04-03](#)

异步任务

[yudao-spring-boot-starter-job](#) [🔗](#) 技术组件，除了提供定时任务的功能，还提供了 Async 异步任务的能力。系统使用异步任务，提升执行效率。例如说：

- [操作日志模块](#) [🔗](#)，异步记录【操作日志】
- [访问日志模块](#) [🔗](#)，异步记录【访问日志】

友情提示：

如果你未学习过 Spring 异步任务，可以后续阅读 [《芋道 Spring Boot 异步任务入门》](#) [🔗](#) 文章。

1. Async 配置

在 [YudaoAsyncAutoConfiguration](#) [🔗](#) 配置类，设置使用 [TransmittableThreadLocal](#) [🔗](#)，解决异步执行时上下文传递的问题。如下图所示：

```
@Configuration
@EnableAsync 开启 Spring Async 功能
public class YudaoAsyncAutoConfiguration {

    @Bean
    public BeanPostProcessor threadPoolTaskExecutorBeanPostProcessor() {
        return new BeanPostProcessor() {

            @Override
            public Object postProcessBeforeInitialization(Object bean, String beanName) throws BeansException {
                if (!(bean instanceof ThreadPoolTaskExecutor)) {
                    return bean;
                }
                // 修改提交的任务，接入 TransmittableThreadLocal
                ThreadPoolTaskExecutor executor = (ThreadPoolTaskExecutor) bean;
                executor.setTaskDecorator(TtlRunnable::get); ② 设置 TransmittableThreadLocal
                return executor;
            }
        };
    }
}
```

友情提示：

项目使用到 ThreadLocal 的地方，建议都使用 TransmittableThreadLocal 进行替换。

2. 引入依赖

以访问日志模块为例，讲解它如何使用异步任务，实现异步记录【访问日志】的功能。

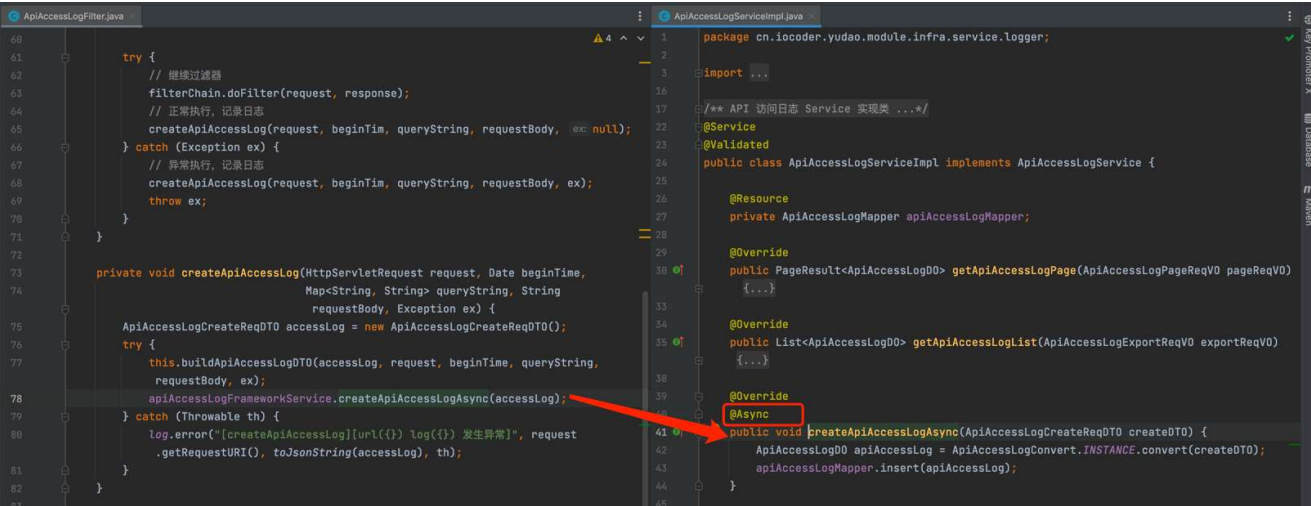
2.1 引入依赖

在 `yudao-module-system-infra` 模块中，引入 `yudao-spring-boot-starter-job` 技术组件。如下所示：

```
<dependency>
  <groupId>cn.iocoder.boot</groupId>
  <artifactId>yudao-spring-boot-starter-job</artifactId>
</dependency>
```

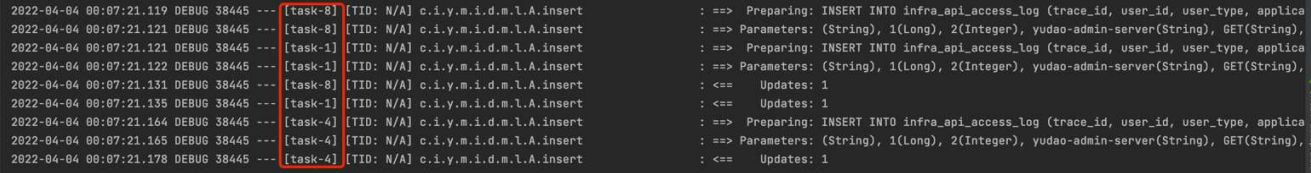
2.2 添加 @Async 注解

在 `ApiAccessLogServiceImpl` 的 `#createApiAccessLogAsync(...)` 方法上，添加 `@Async` 注解，声明它要异步执行。如下图所示：



2.3 测试调用

随便请求一个 RESTful API 接口，可以看到在异步任务的线程池中，进行了访问日志的记录。如下图所示：





Theme by **Vdoing** | Copyright © 2019-2024 芋道源码 | MIT License