中间件

你可以向 **FastAPI** 应用添加中间件.

"中间件"是一个函数,它在每个**请求**被特定的*路径操作*处理之前,以及在每个**响应**返回之前工作.

- 它接收你的应用程序的每一个**请求**.
- 然后它可以对这个**请求**做一些事情或者执行任何需要的代码.
- 然后它将**请求**传递给应用程序的其他部分 (通过某种*路径操作*).
- 然后它获取应用程序生产的**响应** (通过某种*路径操作*).
- 它可以对该**响应**做些什么或者执行任何需要的代码.
- 然后它返回这个 **响应**.

!!! note "技术细节" 如果你使用了 `yield` 关键字依赖, 依赖中的退出代码将在执行中间件*后*执行.

如果有任何后台任务（稍后记录）， 它们将在执行中间件*后*运行.

创建中间件

要创建中间件你可以在函数的顶部使用装饰器 `@app.middleware("http")`.

中间件参数接收如下参数:

- `request`.
- 一个函数 `call_next` 它将接收 `request` 作为参数.
  - 这个函数将 `request` 传递给相应的 *路径操作*.
  - 然后它将返回由相应的*路径操作*生成的 `response`.
- 然后你可以在返回 `response` 前进一步修改它.

```
{!../../../docs_src/middleware/tutorial001.py!}
```

!!! tip 请记住可以 用'X-' 前缀添加专有自定义请求头.

但是如果你想让浏览器中的客户端看到你的自定义请求头， 你需要把它们加到 `CORS` 配置 ([CORS (Cross-Origin Resource Sharing)](cors.md){.internal-link target=_blank}) 的 `expose_headers` 参数中,在 `<a href="https://www.starlette.io/middleware/#corsmiddleware" class="external-link" target="_blank">Starlette's CORS docs</a>`文档中.

!!! note "技术细节" 你也可以使用 `from starlette.requests import Request`.

FastAPI 为了开发者方便提供了该对象. 但其实它直接来自于 `Starlette`.

在 `response` 的前和后

在任何*路径操作*收到 `request` 前,可以添加要和请求一起运行的代码.

也可以在*响应*生成但是返回之前添加代码.

例如你可以添加自定义请求头 `X-Process-Time` 包含以秒为单位的接收请求和生成响应的时间:

```
{!../../../docs_src/middleware/tutorial001.py!}
```

其他中间件

你可以稍后在 Advanced User Guide: Advanced Middleware{.internal-link target=_blank}阅读更多关于中间件的教程.

你将在下一节中学习如何使用中间件处理 CORS .