路径操作装饰器依赖项

有时，我们并不需要在*路径操作函数*中使用依赖项的返回值。

或者说，有些依赖项不返回值。

但仍要执行或解析该依赖项。

对于这种情况，不必在声明*路径操作函数*的参数时使用 `Depends`，而是可以在*路径操作装饰器*中添加一个由 `dependencies` 组成的 `list`。

在*路径操作装饰器*中添加 `dependencies` 参数

*路径操作装饰器*支持可选参数 ~ `dependencies`。

该参数的值是由 `Depends()` 组成的 `list`：

```
{!../../../docs_src/dependencies/tutorial006.py!}
```

路径操作装饰器依赖项（以下简称为**"路径装饰器依赖项"**）的执行或解析方式和普通依赖项一样，但就算这些依赖项会返回值，它们的值也不会传递给*路径操作函数*。

!!! tip "提示"

有些编辑器会检查代码中没使用过的函数参数，并显示错误提示。

在*路径操作装饰器*中使用 `dependencies` 参数，可以确保在执行依赖项的同时，避免编辑器显示错误提示。

使用路径装饰器依赖项还可以避免开发新人误会代码中包含无用的未使用参数。

!!! info "说明"

本例中，使用的是自定义响应头 `X-Key` 和 `X-Token`。

但实际开发中，尤其是在实现安全措施时，最好使用 `FastAPI` 内置的[安全工具](../security/index.md){.internal-link target=_blank}（详见下一章）。

依赖项错误和返回值

路径装饰器依赖项也可以使用普通的依赖项*函数*。

依赖项的需求项

路径装饰器依赖项可以声明请求的需求项（比如响应头）或其他子依赖项：

```
{!../../../docs_src/dependencies/tutorial006.py!}
```

触发异常

路径装饰器依赖项与正常的依赖项一样，可以 `raise` 异常：

```
{!../../../docs_src/dependencies/tutorial006.py!}
```

返回值

无论路径装饰器依赖项是否返回值，路径操作都不会使用这些值。

因此，可以复用在其他位置使用过的、（能返回值的）普通依赖项，即使没有使用这个值，也会执行该依赖项：

```
{!../../../docs_src/dependencies/tutorial006.py!}
```

为一组路径操作定义依赖项

稍后，大型应用 - 多文件{.internal-link target=_blank}一章中会介绍如何使用多个文件创建大型应用程序，在这一章中，您将了解到如何为一组*路径操作*声明单个 `dependencies` 参数。

全局依赖项

接下来，我们将学习如何为 `FastAPI` 应用程序添加全局依赖项，创建应用于每个*路径操作*的依赖项。