请求体 - 多个参数

既然我们已经知道了如何使用 `Path` 和 `Query`，下面让我们来了解一下请求体声明的更高级用法。

混合使用 `Path`、`Query` 和请求体参数

首先，毫无疑问地，你可以随意地混合使用 `Path`、`Query` 和请求体参数声明，**FastAPI** 会知道该如何处理。

你还可以通过将默认值设置为 `None` 来将请求体参数声明为可选参数：

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

!!! note 请注意，在这种情况下，将从请求体获取的 `item` 是可选的。因为它的默认值为 `None`。

多个请求体参数

在上面的示例中，*路径操作*将期望一个具有 `Item` 的属性的 JSON 请求体，就像：

```
{
    "name": "Foo",
    "description": "The pretender",
    "price": 42.0,
    "tax": 3.2
}
```

但是你也可以声明多个请求体参数，例如 `item` 和 `user`：

```
{!../../../docs_src/body_multiple_params/tutorial002.py!}
```

在这种情况下，**FastAPI** 将注意到该函数中有多个请求体参数（两个 Pydantic 模型参数）。

因此，它将使用参数名称作为请求体中的键（字段名称），并期望一个类似于以下内容的请求体：

```
{
    "item": {
        "name": "Foo",
        "description": "The pretender",
        "price": 42.0,
        "tax": 3.2
    },
    "user": {
        "username": "dave",
        "full_name": "Dave Grohl"
    }
}
```

!!! note 请注意，即使 `item` 的声明方式与之前相同，但现在它被期望通过 `item` 键内嵌在请求体中。

FastAPI 将自动对请求中的数据进行转换，因此 `item` 参数将接收指定的内容，`user` 参数也是如此。

它将执行对复合数据的校验，并且像现在这样为 OpenAPI 模式和自动化文档对其进行记录。

请求体中的单一值

与使用 `Query` 和 `Path` 为查询参数和路径参数定义额外数据的方式相同，**FastAPI** 提供了一个同等的 `Body`。

例如，为了扩展先前的模型，你可能决定除了 `item` 和 `user` 之外，还想在同一请求体中具有另一个键 `importance`。

如果你就按原样声明它，因为它是一个单一值，**FastAPI** 将假定它是一个查询参数。

但是你可以使用 `Body` 指示 **FastAPI** 将其作为请求体的另一个键进行处理。

```
{!../../../docs_src/body_multiple_params/tutorial003.py!}
```

在这种情况下，**FastAPI** 将期望像这样的请求体：

```
{
    "item": {
        "name": "Foo",
        "description": "The pretender",
        "price": 42.0,
        "tax": 3.2
    },
    "user": {
        "username": "dave",
        "full_name": "Dave Grohl"
    },
    "importance": 5
}
```

同样的，它将转换数据类型，校验，生成文档等。

多个请求体参数和查询参数

当然，除了请求体参数外，你还可以在任何需要的时候声明额外的查询参数。

由于默认情况下单一值被解释为查询参数，因此你不必显式地添加 `Query`，你可以仅执行以下操作：

```
q: str = None
```

比如：

```
{!../../../docs_src/body_multiple_params/tutorial004.py!}
```

!!! info `Body` 同样具有与 `Query`、`Path` 以及其他后面将看到的类完全相同的额外校验和元数据参数。

嵌入单个请求体参数

假设你只有一个来自 Pydantic 模型 `Item` 的请求体参数 `item`。

默认情况下，**FastAPI** 将直接期望这样的请求体。

但是，如果你希望它期望一个拥有 `item` 键并在值中包含模型内容的 JSON，就像在声明额外的请求体参数时所做的那样，则可以使用一个特殊的 `Body` 参数 `embed`：

```
item: Item = Body(..., embed=True)
```

比如：

```
{!../../../docs_src/body_multiple_params/tutorial005.py!}
```

在这种情况下，**FastAPI** 将期望像这样的请求体：

```
{
    "item": {
        "name": "Foo",
        "description": "The pretender",
        "price": 42.0,
        "tax": 3.2
    }
}
```

而不是：

```
{
    "name": "Foo",
    "description": "The pretender",
    "price": 42.0,
    "tax": 3.2
}
```

总结

你可以添加多个请求体参数到*路径操作函数*中，即使一个请求只能有一个请求体。

但是 **FastAPI** 会处理它，在函数中为你提供正确的数据，并在*路径操作*中校验并记录正确的模式。

你还可以声明将作为请求体的一部分所接收的单一值。

你还可以指示 **FastAPI** 在仅声明了一个请求体参数的情况下，将原本的请求体嵌入到一个键中。