路径参数和数值校验

与使用 `Query` 为查询参数声明更多的校验和元数据的方式相同，你也可以使用 `Path` 为路径参数声明相同类型的校验和元数据。

导入 Path

首先，从 `fastapi` 导入 `Path`：

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

声明元数据

你可以声明与 `Query` 相同的所有参数。

例如，要声明路径参数 `item_id` 的 `title` 元数据值，你可以输入：

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

!!! note 路径参数总是必需的，因为它必须是路径的一部分。

所以，你应该在声明时使用 `...` 将其标记为必需参数。

然而，即使你使用 `None` 声明路径参数或设置一个其他默认值也不会有任何影响，它依然会是必需参数。

按需对参数排序

假设你想要声明一个必需的 `str` 类型查询参数 `q`。

而且你不需要为该参数声明任何其他内容，所以实际上你并不需要使用 `Query`。

但是你仍然需要使用 `Path` 来声明路径参数 `item_id`。

如果你将带有「默认值」的参数放在没有「默认值」的参数之前，Python 将会报错。

但是你可以对其重新排序，并将不带默认值的值（查询参数 `q`）放到最前面。

对 **FastAPI** 来说这无关紧要。它将通过参数的名称、类型和默认值声明（`Query`、`Path` 等）来检测参数，而不在乎参数的顺序。

因此，你可以将函数声明为：

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

按需对参数排序的技巧

如果你想不使用 `Query` 声明没有默认值的查询参数 `q`，同时使用 `Path` 声明路径参数 `item_id`，并使它们的顺序与上面不同，Python 对此有一些特殊的语法。

传递 `*` 作为函数的第一个参数。

Python 不会对该 `*` 做任何事情，但是它将知道之后的所有参数都应作为关键字参数（键值对），也被称为 `kwargs`，来调用。即使它们没有默认值。

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

数值校验：大于等于

使用 `Query` 和 `Path`（以及你将在后面看到的其他类）可以声明字符串约束，但也可以声明数值约束。

像下面这样，添加 `ge=1` 后，`item_id` 将必须是一个大于（`g`reater than）或等于（`e`qual）`1` 的整数。

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

数值校验：大于和小于等于

同样的规则适用于：

- `gt`：大于（`g`reater `t`han）
- `le`：小于等于（`l`ess than or `e`qual）

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

数值校验：浮点数、大于和小于

数值校验同样适用于 `float` 值。

能够声明 `gt` 而不仅仅是 `ge` 在这个前提下变得重要起来。例如，你可以要求一个值必须大于 `0`，即使它小于 `1`。

因此，`0.5` 将是有效值。但是 `0.0` 或 `0` 不是。

对于 `lt` 也是一样的。

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

总结

你能够以与 查询参数和字符串校验{.internal-link target=_blank} 相同的方式使用 `Query`、`Path`（以及其他你还没见过的类）声明元数据和字符串校验。

而且你还可以声明数值校验：

- `gt`：大于（`g`reater `t`han）
- `ge`：大于等于（`g`reater than or `e`qual）
- `lt`：小于（`l`ess `t`han）
- `le`：小于等于（`l`ess than or `e`qual）

!!! info `Query`、`Path` 以及你后面会看到的其他类继承自一个共同的 `Param` 类（不需要直接使用它）。

而且它们都共享相同的所有你已看到并用于添加额外校验和元数据的参数。

!!! note "技术细节" 当你从 `fastapi` 导入 `Query`、`Path` 和其他同类对象时，它们实际上是函数。

当被调用时，它们返回同名类的实例。

如此，你导入 `Query` 这个函数。当你调用它时，它将返回一个同样命名为 `Query` 的类的实例。

因为使用了这些函数（而不是直接使用类），所以你的编辑器不会标记有关其类型的错误。

这样，你可以使用常规的编辑器和编码工具，而不必添加自定义配置来忽略这些错误。