查询参数和字符串校验

FastAPI 允许你为参数声明额外的信息和校验。

让我们以下面的应用程序为例：

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

查询参数 `q` 的类型为 `str`，默认值为 `None`，因此它是可选的。

额外的校验

我们打算添加约束条件：即使 `q` 是可选的，但只要提供了该参数，则该参数值**不能超过50个字符的长度。**

导入 `Query`

为此，首先从 `fastapi` 导入 `Query`：

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

使用 `Query` 作为默认值

现在，将 `Query` 用作查询参数的默认值，并将它的 `max_length` 参数设置为 50：

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

由于我们必须用 `Query(None)` 替换默认值 `None`，`Query` 的第一个参数同样也是用于定义默认值。

所以：

```
q: str = Query(None)
```

...使得参数可选，等同于：

```
q: str = None
```

但是 `Query` 显式地将其声明为查询参数。

然后，我们可以将更多的参数传递给 `Query`。在本例中，适用于字符串的 `max_length` 参数：

```
q: str = Query(None, max_length=50)
```

将会校验数据，在数据无效时展示清晰的错误信息，并在 OpenAPI 模式的*路径操作*中记录该参数。

添加更多校验

你还可以添加 `min_length` 参数：

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

添加正则表达式

你可以定义一个参数值必须匹配的正则表达式：

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

这个指定的正则表达式通过以下规则检查接收到的参数值：

- `^`：以该符号之后的字符开头，符号之前没有字符。
- `fixedquery`：值精确地等于 `fixedquery`。
- `$`：到此结束，在 `fixedquery` 之后没有更多字符。

如果你对所有的这些**「正则表达式」**概念感到迷茫，请不要担心。对于许多人来说这都是一个困难的主题。你仍然可以在无需正则表达式的情况下做很多事情。

但是，一旦你需要用到并去学习它们时，请了解你已经可以在 **FastAPI** 中直接使用它们。

默认值

你可以向 `Query` 的第一个参数传入 `None` 用作查询参数的默认值，以同样的方式你也可以传递其他默认值。

假设你想要声明查询参数 `q`，使其 `min_length` 为 `3`，并且默认值为 `fixedquery`：

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

!!! note 具有默认值还会使该参数成为可选参数。

声明为必需参数

当我们不需要声明额外的校验或元数据时，只需不声明默认值就可以使 `q` 参数成为必需参数，例如：

```
q: str
```

代替：

```
q: str = None
```

但是现在我们正在用 `Query` 声明它，例如：

```
q: str = Query(None, min_length=3)
```

因此，当你在使用 `Query` 且需要声明一个值是必需的时，可以将 `...` 用作第一个参数值：

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

!!! info 如果你之前没见过 `...` 这种用法：它是一个特殊的单独值，它是 Python 的一部分并且被称为「省略号」。

这将使 **FastAPI** 知道此查询参数是必需的。

查询参数列表 / 多个值

当你使用 `Query` 显式地定义查询参数时，你还可以声明它去接收一组值，或换句话来说，接收多个值。

例如，要声明一个可在 URL 中出现多次的查询参数 `q`，你可以这样写：

```
{!../../../docs_src/query_params_str_validations/tutorial011.py!}
```

然后，输入如下网址：

```
http://localhost:8000/items/?q=foo&q=bar
```

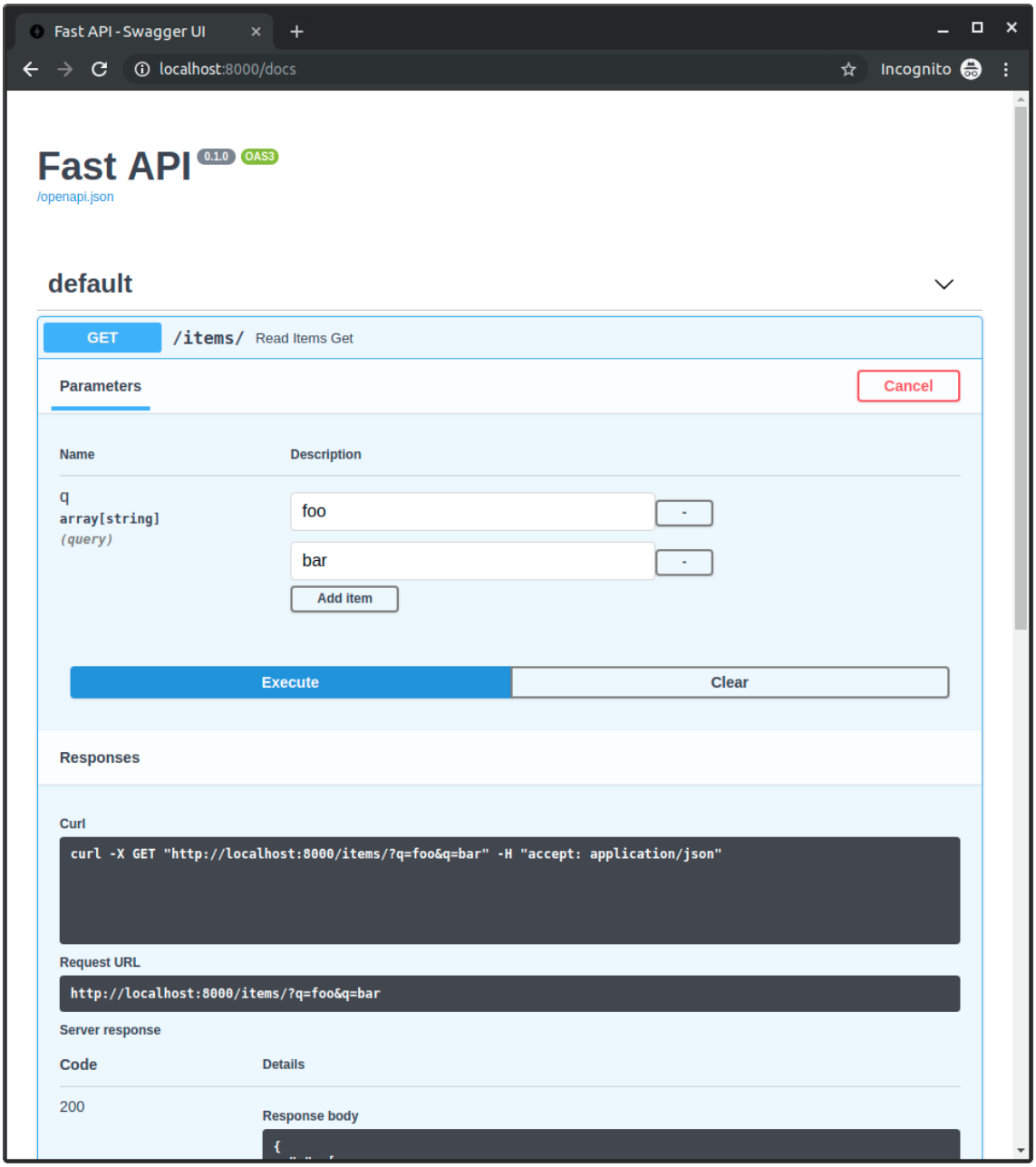
你会在*路径操作函数*的*函数参数* `q` 中以一个 Python `list` 的形式接收到*查询参数* `q` 的多个值（`foo` 和 `bar`）。

因此，该 URL 的响应将会是：

```
{
  "q": [
    "foo",
    "bar"
  ]
}
```

!!! tip 要声明类型为 `list` 的查询参数，如上例所示，你需要显式地使用 `Query`，否则该参数将被解释为请求体。

交互式 API 文档将会相应地进行更新，以允许使用多个值：



具有默认值的查询参数列表 / 多个值

你还可以定义在没有任何给定值时的默认 `list` 值：

```
{!../../../docs_src/query_params_str_validations/tutorial012.py!}
```

如果你访问：

```
http://localhost:8000/items/
```

`q` 的默认值将为：`["foo", "bar"]`，你的响应会是：

```
{
  "q": [
    "foo",
    "bar"
  ]
}
```

使用 `list`

你也可以直接使用 `list` 代替 `List [str]`：

```
{!../../../docs_src/query_params_str_validations/tutorial013.py!}
```

!!! note 请记住，在这种情况下 FastAPI 将不会检查列表的内容。

例如，`List[int]` 将检查（并记录到文档）列表的内容必须是整数。但是单独的 `list` 不会。

声明更多元数据

你可以添加更多有关该参数的信息。

这些信息将包含在生成的 OpenAPI 模式中，并由文档用户界面和外部工具所使用。

!!! note 请记住，不同的工具对 OpenAPI 的支持程度可能不同。

其中一些可能不会展示所有已声明的额外信息，尽管在大多数情况下，缺少的这部分功能已经计划进行开发。

你可以添加 `title`：

```
{!../../../docs_src/query_params_str_validations/tutorial007.py!}
```

以及 `description`：

```
{!../../../docs_src/query_params_str_validations/tutorial008.py!}
```

别名参数

假设你想要查询参数为 `item-query`。

像下面这样：

```
http://127.0.0.1:8000/items/?item-query=foobaritems
```

但是 `item-query` 不是一个有效的 Python 变量名称。

最接近的有效名称是 `item_query`。

但是你仍然要求它在 URL 中必须是 `item-query`...

这时你可以用 `alias` 参数声明一个别名，该别名将用于在 URL 中查找查询参数值：

```
{!../../../docs_src/query_params_str_validations/tutorial009.py!}
```

弃用参数

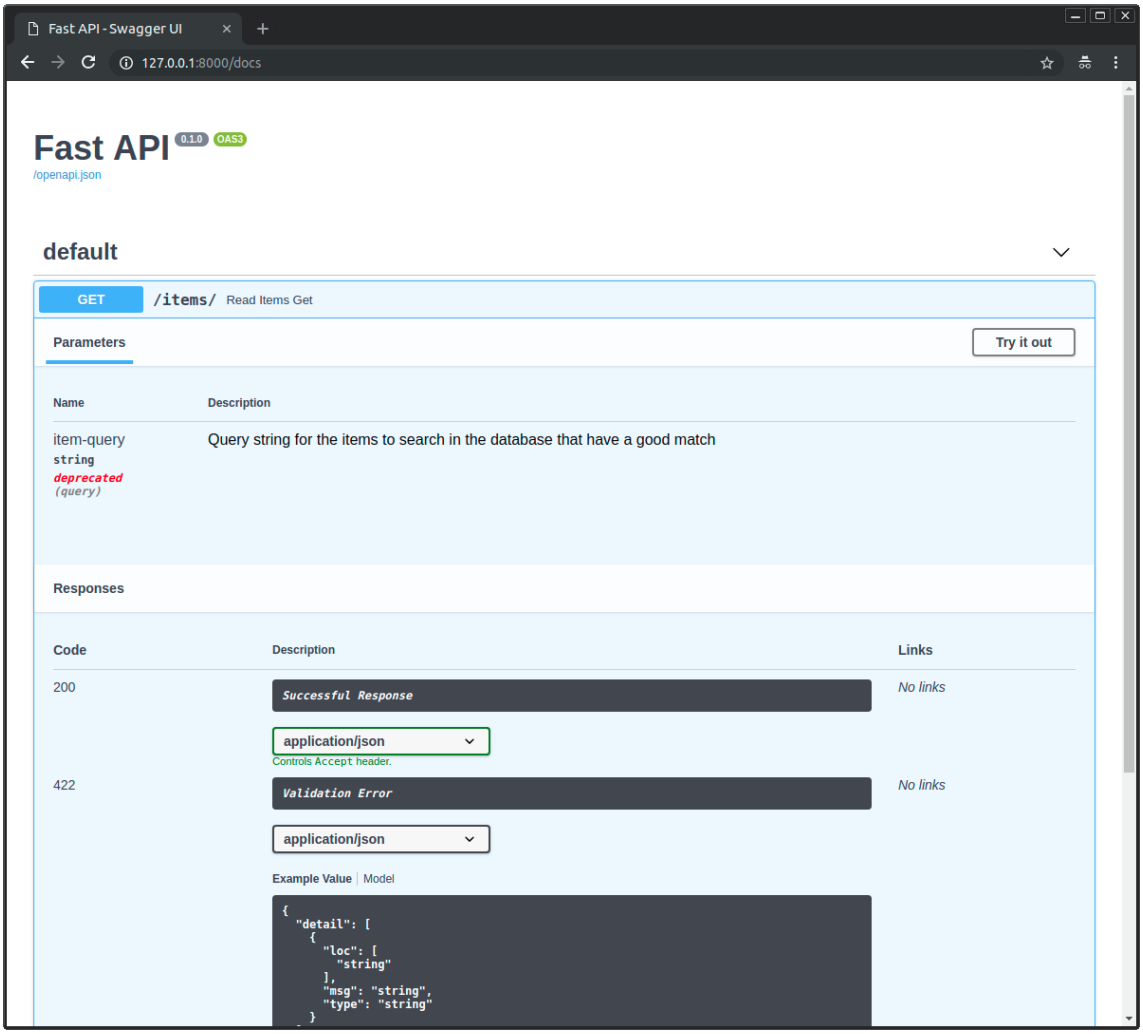
现在假设你不再喜欢此参数。

你不得不将其保留一段时间，因为有些客户端正在使用它，但你希望文档清楚地将其展示为已弃用。

那么将参数 `deprecated=True` 传入 `Query`：

```
{!../../../docs_src/query_params_str_validations/tutorial010.py!}
```

文档将会像下面这样展示它：



总结

你可以为查询参数声明额外的校验和元数据。

通用的校验和元数据：

- `alias`
- `title`
- `description`
- `deprecated`

特定于字符串的校验：

- `min_length`
- `max_length`
- `regex`

在这些示例中，你了解了如何声明对 `str` 值的校验。

请参阅下一章节，以了解如何声明对其他类型例如数值的校验。