额外数据类型

到目前为止，您一直在使用常见的数据类型，如:

- `int`
- `float`
- `str`
- `bool`

但是您也可以使用更复杂的数据类型。

您仍然会拥有现在已经看到的相同的特性:

- 很棒的编辑器支持。
- 传入请求的数据转换。
- 响应数据转换。
- 数据验证。
- 自动补全和文档。

其他数据类型

下面是一些你可以使用的其他数据类型:

- `UUID`：
  - 一种标准的 "通用唯一标识符"，在许多数据库和系统中用作ID。
  - 在请求和响应中将以 `str` 表示。
- `datetime.datetime`：
  - 一个 Python `datetime.datetime`.
  - 在请求和响应中将表示为 ISO 8601 格式的 `str`，比如: `2008-09-15T15:53:00+05:00`.
- `datetime.date`：
  - Python `datetime.date`.
  - 在请求和响应中将表示为 ISO 8601 格式的 `str`，比如: `2008-09-15`.
- `datetime.time`：
  - 一个 Python `datetime.time`.
  - 在请求和响应中将表示为 ISO 8601 格式的 `str`，比如: `14:23:55.003`.
- `datetime.timedelta`：
  - 一个 Python `datetime.timedelta`.
  - 在请求和响应中将表示为 `float` 代表总秒数。
  - Pydantic 也允许将其表示为 "ISO 8601 时间差异编码", 查看文档了解更多信息。
- `frozenset`：
  - 在请求和响应中，作为 `set` 对待:
    - 在请求中，列表将被读取，消除重复，并将其转换为一个 `set`。
    - 在响应中 `set` 将被转换为 `list`。
    - 产生的模式将指定那些 `set` 的值是唯一的 (使用 JSON 模式的 `uniqueItems`)。
- `bytes`：
  - 标准的 Python `bytes`。
  - 在请求和相应中被当作 `str` 处理。
  - 生成的模式将指定这个 `str` 是 `binary` "格式"。

- `Decimal`：
  - 标准的 Python `Decimal`。
  - 在请求和相应中被当做 `float` 一样处理。
- 您可以在这里检查所有有效的pydantic数据类型: Pydantic data types.

例子

下面是一个*路径操作*的示例，其中的参数使用了上面的一些类型。

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

注意，函数内的参数有原生的数据类型，你可以，例如，执行正常的日期操作，如:

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