请求体 - 字段

与使用 `Query`、`Path` 和 `Body` 在*路径操作函数*中声明额外的校验和元数据的方式相同，你可以使用 Pydantic 的 `Field` 在 Pydantic 模型内部声明校验和元数据。

导入 `Field`

首先，你必须导入它：

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

!!! warning 注意，`Field` 是直接从 `pydantic` 导入的，而不是像其他的（`Query`，`Path`，`Body` 等）都从 `fastapi` 导入。

声明模型属性

然后，你可以对模型属性使用 `Field`：

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

`Field` 的工作方式和 `Query`、`Path` 和 `Body` 相同，包括它们的参数等等也完全相同。

!!! note "技术细节" 实际上，`Query`、`Path` 和其他你将在之后看到的类，创建的是由一个共同的 `Params` 类派生的子类的对象，该共同类本身又是 Pydantic 的 `FieldInfo` 类的子类。

```
Pydantic 的 `Field` 也会返回一个 `FieldInfo` 的实例。

`Body` 也直接返回 `FieldInfo` 的一个子类的对象。还有其他一些你之后会看到的类是 `Body` 类的子类。

请记住当你从 `fastapi` 导入 `Query`、`Path` 等对象时，他们实际上是返回特殊类的函数。
```

!!! tip 注意每个模型属性如何使用类型、默认值和 `Field` 在代码结构上和*路径操作函数*的参数是相同的，区别是用 `Field` 替换 `Path`、`Query` 和 `Body`。

添加额外信息

你可以在 `Field`、`Query`、`Body` 中声明额外的信息。这些信息将包含在生成的 JSON Schema 中。

你将在文档的后面部分学习声明示例时，了解到更多有关添加额外信息的知识。

总结

你可以使用 Pydantic 的 `Field` 为模型属性声明额外的校验和元数据。

你还可以使用额外的关键字参数来传递额外的 JSON Schema 元数据。