Header 参数

你可以使用定义 `Query`，`Path` 和 `Cookie` 参数一样的方法定义 Header 参数。

导入 `Header`

首先导入 `Header`：

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

声明 `Header` 参数

然后使用和 `Path`，`Query` and `Cookie` 一样的结构定义 header 参数

第一个值是默认值，你可以传递所有的额外验证或注释参数：

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

!!! note "技术细节" `Header` 是 `Path`，`Query` 和 `Cookie` 的兄弟类型。它也继承自通用的 `Param` 类.

但是请记得，当你从`fastapi`导入 `Query`, `Path`, `Header`, 或其他时，实际上导入的是返回特定类型的函数。

!!! info 为了声明headers， 你需要使用 `Header`，因为否则参数将被解释为查询参数。

自动转换

`Header` 在 `Path`，`Query` 和 `Cookie` 提供的功能之上有一点额外的功能。

大多数标准的headers用 "连字符" 分隔，也称为 "减号" (`-`)。

但是像 `user-agent` 这样的变量在Python中是无效的。

因此, 默认情况下, `Header` 将把参数名称的字符从下划线 (`_`) 转换为连字符 (`-`) 来提取并记录 headers.

同时，HTTP headers 是大小写不敏感的，因此，因此可以使用标准Python样式(也称为 "snake_case")声明它们。

因此，您可以像通常在Python代码中那样使用 `user_agent` ，而不需要将首字母大写为 `User_Agent` 或类似的东西。

如果出于某些原因，你需要禁用下划线到连字符的自动转换，设置 `Header` 的参数 `convert_underscores` 为 `False`：

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

!!! warning 在设置 `convert_underscores` 为 `False` 之前，请记住，一些HTTP代理和服务器不允许使用带有下划线的headers。

重复的 headers

有可能收到重复的headers。这意味着，相同的header具有多个值。

您可以在类型声明中使用一个list来定义这些情况。

你可以通过一个Python `list` 的形式获得重复header的所有值。

比如, 为了声明一个 `X-Token` header 可以出现多次，你可以这样写：

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

如果你与*路径操作*通信时发送两个HTTP headers，就像：

```
X-Token: foo
X-Token: bar
```

响应会是:

```
{
    "X-Token values": [
        "bar",
        "foo"
    ]
}
```

回顾

使用 `Header` 来声明 header , 使用和 `Query` , `Path` 与 `Cookie` 相同的模式。

不用担心变量中的下划线，**FastAPI** 会负责转换它们。