# 一、Fastapi的简介

## 1) FastAPI是什么

FastAPI是一个现代的，快速（高性能）python web框架。基于标准的python类型提示，使用python3.6+构建API的Web框架。

## 2) FastAPI的主要特点

1) 快速：非常高的性能，与NodeJS和Go相当(这个要感谢Starlette和Pydantic），是最快的Python框架之一。

2) 快速编码：将开发速度提高约200%到300%。

3) 更少的bug：减少大约40%的开发人员人为引起的错误。

4) 直观：强大的编辑器支持，调试时间更短。

5) 简单：易于使用和学习。减少阅读文档的时间。

6) 代码简洁：尽量减少代码重复。每个参数可以声明多个功能，减少程序的bug。

7) 健壮：生产代码会自动生成交互式文档。

8) 基于标准：基于并完全兼容API的开放标准：OpenAPI和JSON模式。

# 二、Fastapi的优势

## 1) Django

设计理念：以MTV模式为核心设计理念，提供了丰富的内置功能和插件库，适合构建大型、高可用性的Web应用程序。

适用场景：适用于构建复杂的Web应用程序、电商平台等。

优点：完备的文档和教程，强大的ORM和自带的管理后台，拥有完整和稳定的生态系统。

缺点：配置繁琐，灵活性不够。

发展趋势：Django已成为Python Web框架中最受欢迎的框架之一，发展趋势稳步上升，社区活跃，生态圈完善。

## 2) Flask

设计理念：以轻量、灵活为设计理念，提供了基础的路由和模板引擎，适合快速开发小型Web应用程序或原型。

适用场景：适用于构建小型的Web应用程序、接口请求等。

优点：学习曲线低，易于扩展和定制，自由度高。

缺点：生态圈较小，对于大型应用缺乏一些必要的内置功能。

发展趋势：Flask是一个不断发展的Python Web框架，相对于Django更加轻量级，发展趋势稳定但不如Django快。

## 3) FastAPI

设计理念：采用异步、基于声明的方式设计API，支持OpenAPI规范和自动文档生成，可以大幅提升开发效率。

适用场景：适用于构建高性能、高并发的Web API服务。

优点：性能出色，文档完备，易于开发和维护，支持异步编程，自动完成API文档和测试。

缺点：相对较新的框架，生态圈还在发展中。

发展趋势：FastAPI是一个快速发展的Web框架，得到了越来越多开发者的青睐，其在性能、开发效率和文档化方面都有很大优势，未来有望成为Python Web框架中的重要一员。

# 三、Fastapi的安装

## 1) fastapi的安装

pip install fastapi -i <https://mirrors.aliyun.com/pypi/simple>

## 2) uvicorn的安装

pip install uvicorn -i <https://mirrors.aliyun.com/pypi/simple>

## 3）安装成功后的验证





参数介绍：

main: main.py 文件(也可理解为Python模块)；

app: main.py 中app = FastAPI()语句创建的app对象；

--reload: 在代码改变后重启服务器，只能在开发的时候使用；

# 四、Fastapi的用法介绍

## 1) 路径操作装饰器中的路径参数

### (1) 声明路径参数



### (2) 申明路径参数的类型



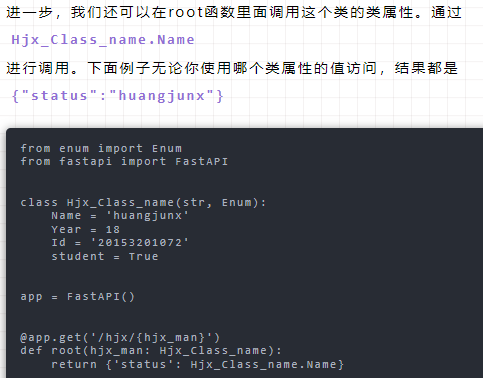
当我们声明了路径参数的类型，如果我们在访问链接的时候提供的参数类型不对，FastAPI还会自动为我们做数据校验的功能，在开发和调试与您的API交互的代码时，这非常有用。注意两点：

所有的数据验证都是由Pydantic实现的，你可以用同样的类型声明比如str,float,bool或者其他更复杂的类型.

### (3) 限定路径参数有效值





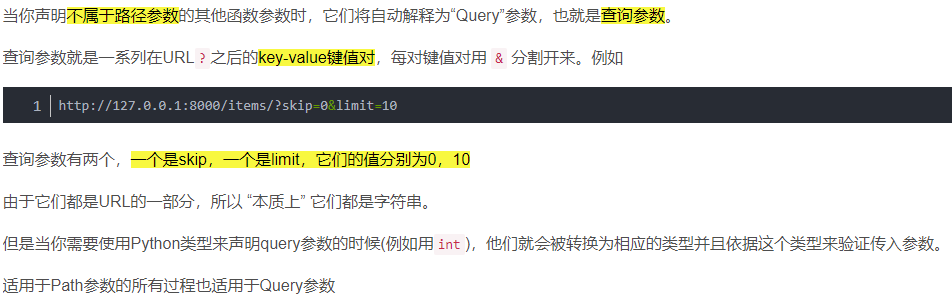


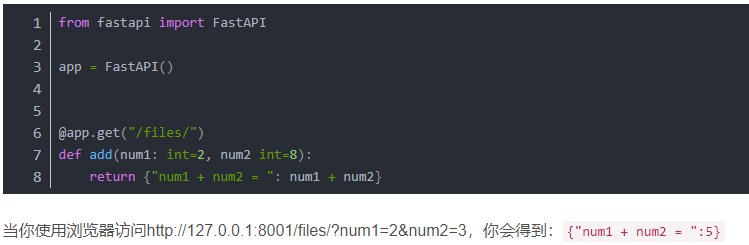
### (4) 路径参数的值是路径类型变量



## 2) 查询参数

### (1) 查询参数概念





### (2) 给查询参数设置默认值

query参数类不是path中固定的一部分，所以他们是可选的，并且可以有默认值。

例如上面的例子，当你使用浏览器访问http://127.0.0.1:8001/files/，你会得到：{"num1 + num2 = ":10}。

### (3) 设置可选的查询参数

声明可选的Query参数，只需要将他们的默认值设置为None即可。

关于查询参数还需要注意以下几点：

a.如果设置的查询参数没有默认值不是None，那么这个查询参数就是必需查询参数，必须要传入，否则会报错；

b.查询参数可以和路径参数结合使用。

## 3) 请求体

### (1) 什么是请求体

当您需要将数据从客户端（例如浏览器）发送到API时，可以将其作为 “请求体” 发送。

请求体是客户端发送到您的API的数据。 响应体是您的API发送给客户端的数据。API几乎总是必须发送一个响应体，但是客户端并不需要一直发送请求体。定义请求体，需要使用 Pydantic 模型。注意以下几点：

a.不能通过GET请求发送请求体；

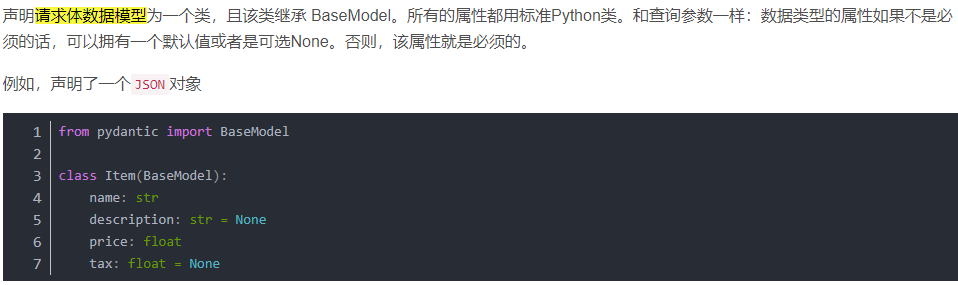
b.发送请求体数据，必须使用以下几种方法之一：POST（最常见）、PUT、DELETE、PATCH。

### (2) 什么是请求体

第一步，从pydantic中导入BaseModel；



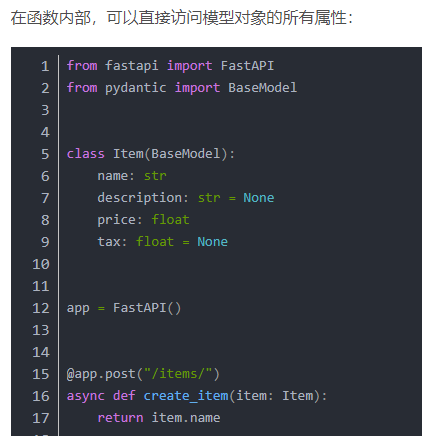
第二步，创建请求体数据模型；



第三步、将模型定义为参数；



### (3) 使用请求体模型



注意以下几点：

a.可以同时定义路径参数和请求体参数；

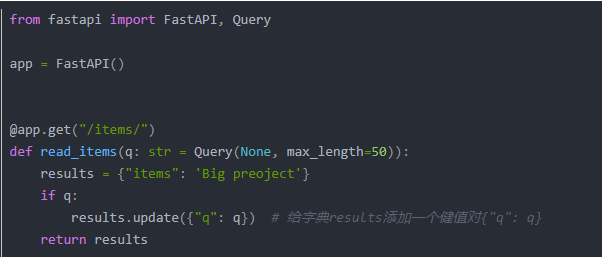
b.可以同时定义路径参数和查询参数和请求体参数，如果path中声明了某个参数，那么这个参数将作为路径参数是使用；如果参数是 单一类型（例如int，float，str，str，bool等），它将被解释为 query参数。

## 4) 给查询参数设置验证条件(字符串验证)

### (1) 为查询参数添加验证

第一步，导入Query；

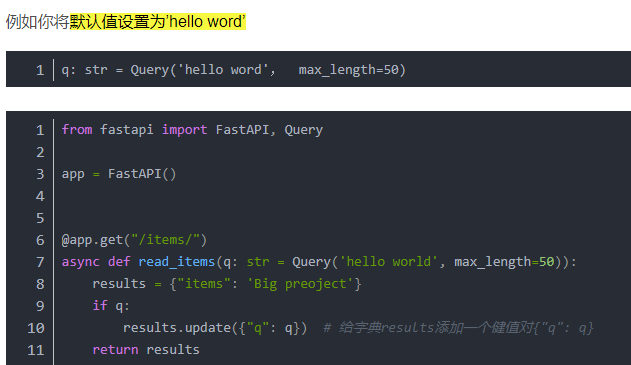
第二步、使用 Query 设置max\_length验证；



说明几点：

一、因为我们必须用 Query(None) 替换默认值None，所以Query的第一个参数是定义默认值。当默认值为None,也表示这个参数是可选的。当默认值为...，表示这个参数是必选的。

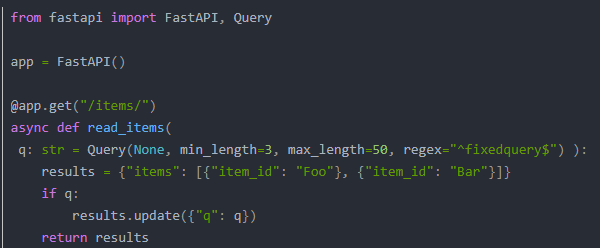




二、这样我们就可以向Query对象中传递更多验证参数来验证查询参数，例如本例的max\_length参数，这将验证数据，当数据无效时显示清楚的错误，并在OpenAPI Schema路径操作中记录参数。



**添加正则表达式验证：**



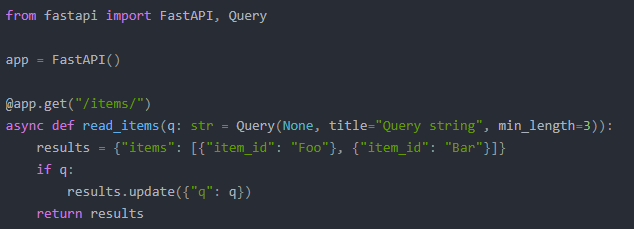
这个特定的正则表达式检查接收到的参数值：

^: 以…开头，表示字符串fixedquery前面没有字符。

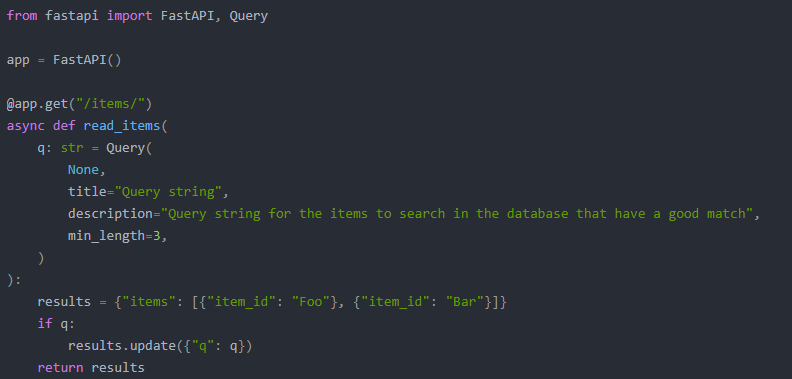
fixedquery: 匹配 fixedquery 字符串.

$: 以…结尾，表示字符串fixedquery后面不匹配任何字符

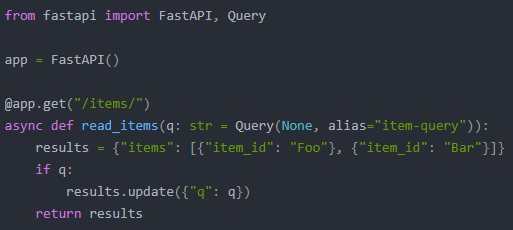
**添加参数标题title：**



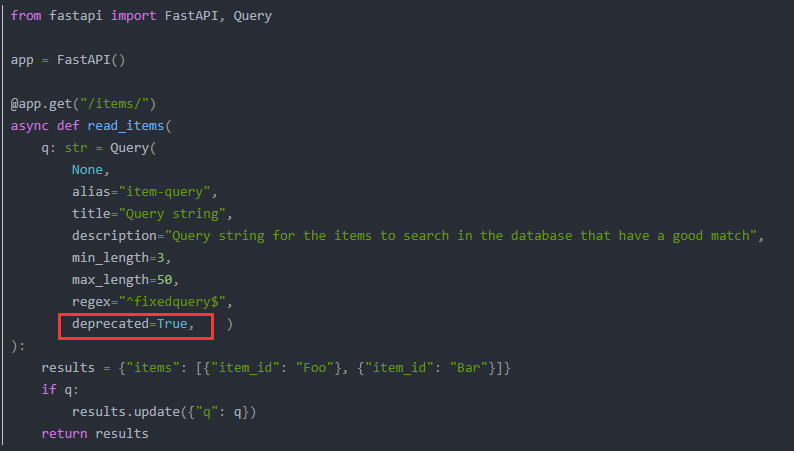
**添加一个参数描述description：**



**给参数起一个别名：**



**给参数做个标记，说明这个参数已被弃用：**

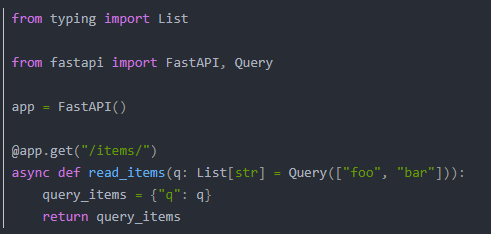


常用的验证参数有alias、title、description、deprecated，常见的对字符串的验证有min\_length、max\_length、regex。

### (2) 声明在URL中出现多次的查询参数

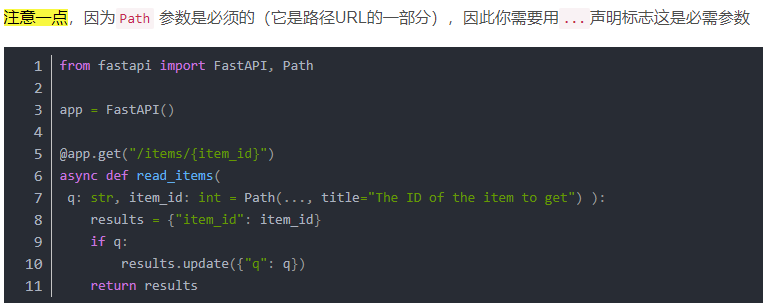


除了可以给它设置可选默认值，我们也可以给它设置一个默认值。

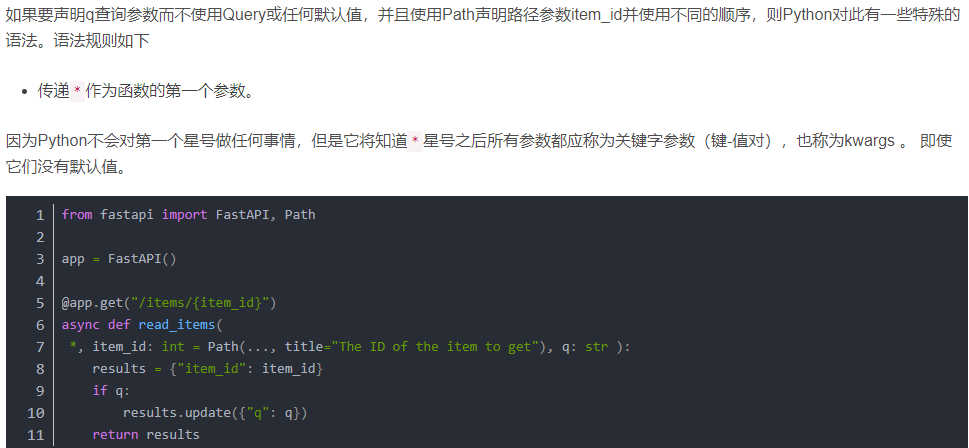


## 5) 给路径参数设置验证条件(数值验证)

### (1) 给路径参数添加验证

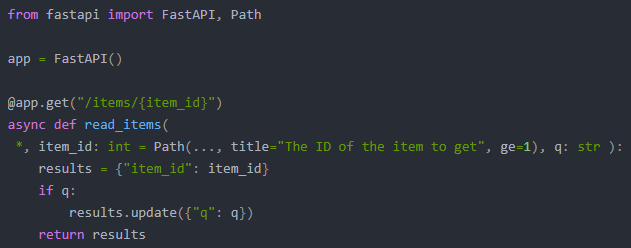


### (2) 参数排序技巧

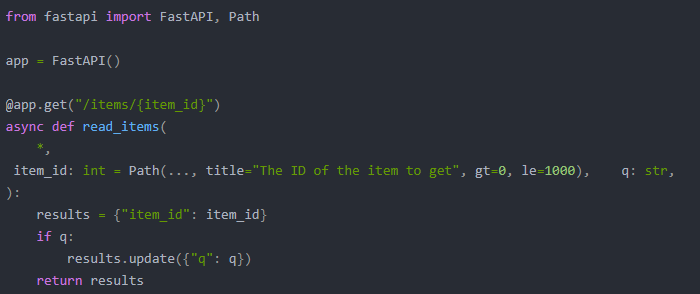


### (3) 给路径参数添加数字验证：对数字大小

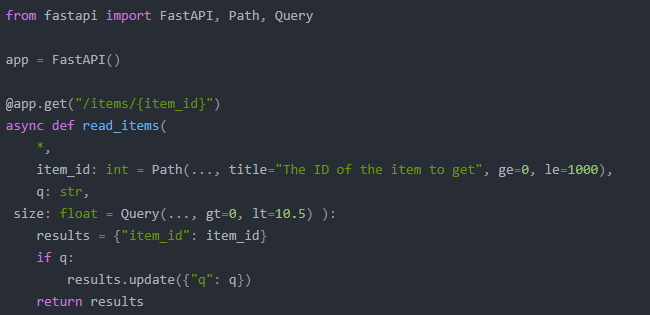
第一个，大于或者等于ge，只能比较整数。例如大于等于1；



第二个，小于或者等于le，只能比较整数。例如小于等于1000；



第三个，大于gt, 小于lt，可以比较浮点数和整数；



## 6) 请求体多种参数

### (1) 混合使用Path, Query 和 请求体参数



这里混合使用了路径参数item\_id，查询参数q，请求体参数item。第一个参数星号表示星号之后所有参数都应称为关键字参数（键-值对），即使它们没有默认值。这是一种有效的写法。

### (2) 多个模型的请求体参数



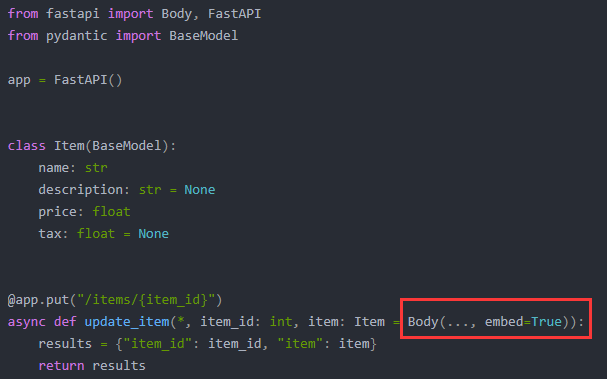
### (3) 使用Body方法定义单值的请求体参数

扩展之前的模型，增加一个参数：除item和user这两个字段外，添加字段importance，如果直接定义它，因为他是一个单值，FastAPI会默认将其定义为query参数。可以使用Body方法，让FastAPI将其视为请求体的key：



### (4) 嵌入单个请求体参数

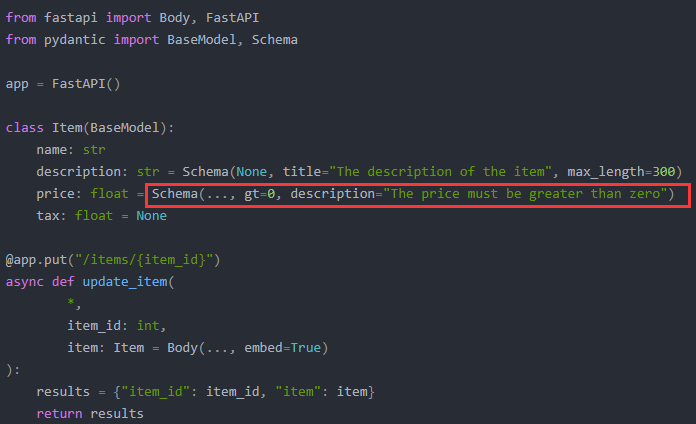
得到一个带有key的JSON，并且在key对应的包含模型内容，就像声明额外的主体参数时那样，则可以使用嵌入的特殊Body参数：



## 7) Body-Schema模型

### (1) 创建Schema模型

可以使用Schema模型，在定义请求体模型的类属性的时候，给类属性添加默认值或者验证。



注意一点：Schema与 Query，Path和Body的工作方式相同，具有相同的参数。

### (2) 额外参数给请求体添加示例字段



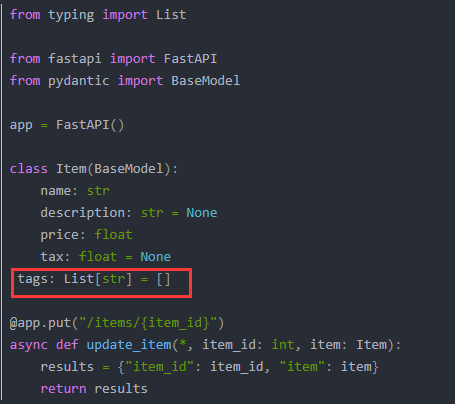
这样的话，当我们访问/docs，我们将看到如下：Example Value | Schema 数据示例，告诉用户，这个api想要什么样格式的请求体格式数据。

## 8) Body-Schema模型

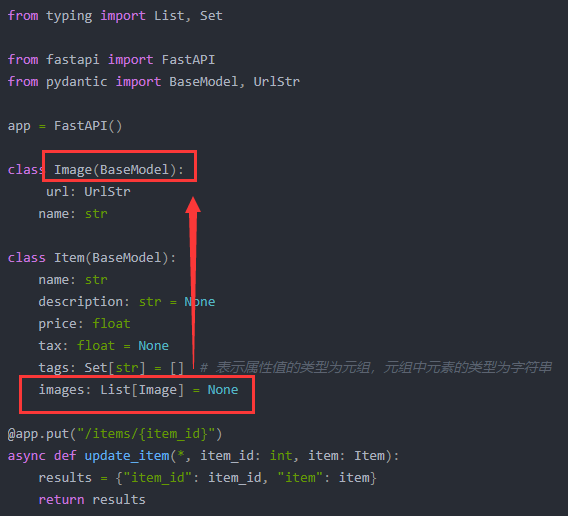
可以将请求体的属性值类型设置为列表、元组，列表里面的元素可以是正常的数据类型，也可以是请求体模型。当然，请求体的属性值也可以设置为请求体模型。这种现象我们称之为模型的嵌套。

### (1) 请求体模型的属性值类型为列表

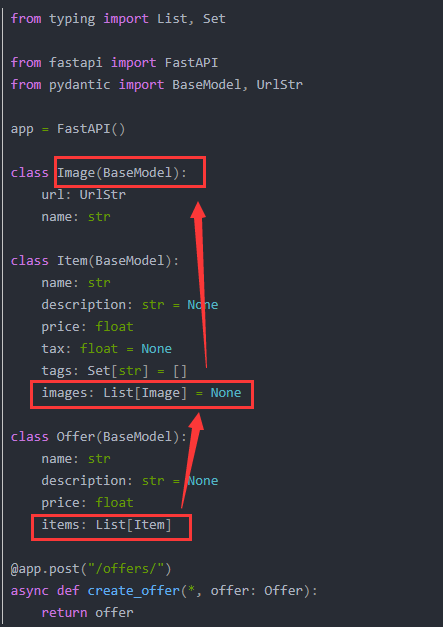
定义列表里面元素的类型为字符串：



另外一个请求体模型：



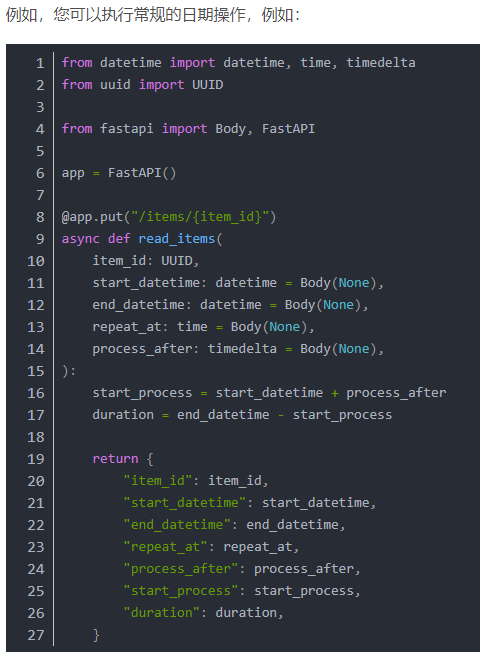
### (2) 深层嵌套模型



## 9) Extra data types(额外数据类型)

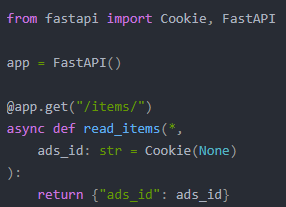
除了下面常见的几种数据类型之外，int，float，str，bool；fastAPI还提供了其它更复杂的数据类型：





## 10) Cookie Parameters(Cookie参数)

定义Cookie参数，就和你定义Query和Path参数一样的方式。

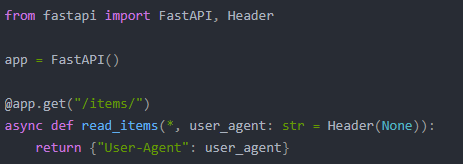


注意两点：

Cookie是Path和Query的姐妹类，它也同样继承自相同的Param类，所以适用路径参数和查询参数的验证，Cookie参数也适用；申明cookies，你需要使用Cookie方法，否则参数会被解释为查询参数。

## 11) Header Parameters(Header参数)

定义Header参数，就和你定义Query和Path参数一样的方式。



注意两点：

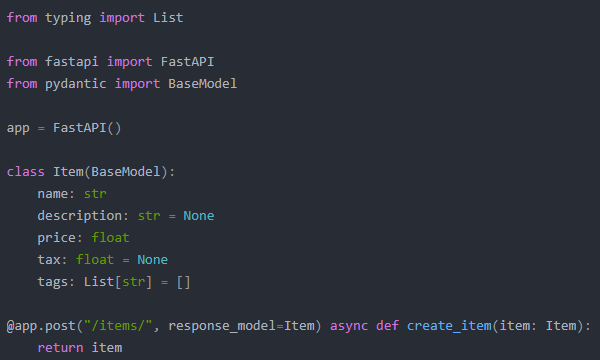
Header是Path、Query和Cookie的姐妹类，它也同样继承自相同的Param类.，所以适用路径参数和查询参数的验证，Cookie参数也适用；申明Header，你需要使用Header方法，否则参数会被解释为查询参数。

## 12) Response Model(响应模型)

可以在任何路径操作中使用参数 response\_model 声明用于响应的模型：

@app.get()，@app.post()，@app.put()，@app.delete()。

### (1) 响应模型的定义



从上面的代码不难看出：

response\_model是“ decorator”方法（get，post等）的参数，该参数接受一个模型类。它接收的类型与您为Pydantic模型属性声明的类型相同，因此它可以是Pydantic模型，但也可以是例如 一个Pydantic模型的清单，例如List [Item]。

response\_model功能：将输出数据限制为模型的数据。从而可以将输出数据转换为其类型声明，并且可以验证数据最后将由自动文档系统进行使用。

### (2) 响应模型的使用





# 五、Fastapi的开发流程

# 六、Fastapi的案例

# 七、Fastapi的注意事项

# 八、其他web框架简介

## 1) Django

Django是用于构建网站的免费开源 Web 开发框架。于 2003 年创建，使用 MTV 模式（Model-Template-View），如果不使用 Django 的模版语法，也可以认为是 MVC 模式（Model-View-Controler），对应三个文件：models.py, views.py, urls.py。

Django以其健壮和简单的特性而广受欢迎，地位相当于 Java 领域的 Spring，并被 Instagram，Youtube 等技术巨头使用，找工作面试，Django 也是重头戏。Django 非常流行，极客时间已经有专门的付费教程。其次它的设计哲学与 Python 非常搭配，鼓励代码复用。

### (1) 优点

Django框架的代码结构统一且清晰易懂，开发人员可以像做填空题一样添加更多功能，也可以设计自定义的类或接口，即简单也不失灵活性、扩展性。

Django生态还有一个利器就是 Django Rest Framework（DRF），DRF 可以让你轻松搭建具有 REST 风格的 API，由于它具有模块化和可自定义的架构，在前后端分离的趋势下，使用 Django 构建 Web API 也非常容易。

自带安全性的中间件。Django 通过提供针对联合 SQL 注入和跨站点请求伪造攻击的防御来提供安全性的访问。

Django 生态非常丰富，第三方类适配的类库也非常多，这意味着遇到问题时你并不孤单。

### (2) 缺点

Django 可能会因为许多可重用的模块而限制了开发速度。它还需要确保以前的版本仍与新版本兼容，从而导致其运行稍慢。

Django Models 没有 mixins，只有简单的继承。

Django 使用的 ORM 是在 SQLAlchemy 之前创建的，因此 Django ORM 的灵活性较差，因此它不如 SQLAlchemy。

### (3)用例

Django 可以处理任何规模和容量的项目。我们可以将其用于简单站点或高性能站点。一些用例如下：

高负荷预订引擎。

购物平台。

学校管理系统。

内置用于内部数据的自定义 CRM 系统。

支持基于 Web 的应用程序的 IOS 和 Android 应用程序。

管理仪表板，基于照片的验证，用于发送用户通知的电子邮件系统等。

简而言之，如果要做一个上规模，不断扩展的，大规模，商业化的 Web 项目，Django 是最好的选择。

## 2) Flask

Flask是一个用 Python 编写的微型 Web 框架，可用于开发简约的 Web 应用程序。Flask 带有诸如模板引擎，缓存和身份验证之类的选项。

Flask的创建是为了使用 Python 编程语言构建 Web 应用程序。它被设计为易用，快速并且可以扩展成为复杂的应用程序或微服务。

　Flask更像是创作者 Armin Ronacher 在 4 月 1 日对愚人节的恶作剧，在其中依赖了Werkzeug 和 Jinja。令人惊讶的是，它受到了社区的赞赏，这使他们在此方面做得更多。

### (1) 优点

与其他 Web 应用程序框架不同，Flask 可以完全控制 Web 开发，从而完全控制应用程序和 Web 开发。

Flask 允许进行单元测试，并且由于其内置的开发服务器，集成的支持等，因此可以通过对一些扩展进行调整来过渡到 Web 框架。

Flask 简单易用，非常适合初学者使用，为开发人员提供了更好地学习和理解它的空间。它还使开发人员可以毫不费力地快速创建应用程序。

### (2) 缺点

Flask 的很多模块由第三方开发，容易引起安全漏洞。

Flask 具有一个单一的来源，表示它将依次处理每个请求，因此，无论有多少个请求，它仍然会轮流处理它们，这会耗费更多时间。

如果开发人员水平不高，更容易使用低质量的代码创建一个不良的 Web 应用程序。

### (3) 用例

可以将 Flask 用于商业项目。它可以帮助你快速入门，但是网站高负荷情况下效果不佳。可以快速实施 Flask 项目，例如：

电子商务系统。

Facebook / Twitter机器人。

在线社交网络。

静态网站。

如果要做一些小型个人项目，比如聊天机器人，或者想实现产品的快速原型，或者喜欢自由的编写代码控制程序的流程，那么可以选择 Flask，如果说 Django 是具备半自动驾驶的大型车，那么 Flask 就是可自由操纵的手动挡小汽车。

## 3) FastAPI

FastAPI 是用于构建 Web API 的现代、开源、快速、高性能的 Web 框架，它基于Python 3.6+ 标准类型提示，支持异步，正如它的名字，FastAPI 就是为构建快速的 API 而生。

### (1) 优点

自动类型检查。这意味着更少的 Bug，即使在深度嵌套的 JSON 请求中，Fast API 也会验证开发人员的数据类型。

集众所长，站在巨人的肩膀上。FastAPI 建立在 JSON Schema（用于验证JSON数据结构的工具），OAuth 2.0（用于授权的行业标准协议）和OpenAPI（这是可公开获得的应用程序编程接口）之类的标准之上。

现代化。FastAPI使使用称为graphene-python的 Python 库轻松构建GraphQL API 。

快速、高性能。可以和 NodeJS 和 Go 相提并论。

### (2) 缺点

由于 FastAPI 相对较新，因此与其他框架相比，社区较小，第三方的教程相对较少。

### (3) 用例

FastAPI 适用于构建高性能的 API，本身支持异步，如果要构建异步 API，可以优先选择 FastAPI。Netflix 将其用于内部危机管理。它还可以在部署准备就绪的机器学习模型时完美缩放，因为当 ML 模型封装在 REST API 并部署在微服务中时，它在生产中会发挥最佳作用。

## 4) Django、Flask、FastAPI间的比较

### (1) 软件包丰富程度

Django 具有使代码可重用的大多数软件包，是一个完整的 Web 开发框架，而 Flask 和 FastAPI 是用于构建网站的简约框架，很多功能比如用户系统，后台管理要自己实现。

### (2) 社区活跃程度

Django 社区是最活跃的社区，这是因为它使用广泛，很多大厂使用，另一方面，Flask 的社区也很繁荣，仅次于 Django。FastAPI 的社区目前还比较小，因为它相对较新。

### (3) 性能

FastAPI 是领跑者，因为它是面向速度的，其次是Flask，最后是 Django。

### (4) 灵活性

灵活性是开发人员非常重视的东西，并且 Flask 比 Django 更灵活。另一方面，FastAPI 在代码方面是灵活的，并且不限制代码布局。因此，我们可以说 Flask 在这三者中是最灵活的。

### (5) 职位空缺

毫无疑问，Python 网络生态系统中有 Django 要求的职位空缺更多，其次是 Flask，最后是 FastAPI，其数量要少得多，因此，如果你打算快速找到工作，那么 Django 应该是首选。

### (6) 学习成本

FastAPI < Flask < Django。Django 虽然学习起来比较费劲，但是有完善的官方文档和很多在线资料和资源。Flask 既简单又直接，也有丰富的在线资料和资源。而 FastAPI 学习起来更简单直接，不过资源相对较少，因为还需要时间。

# 九、参考网址

第一个：

<https://www.cnblogs.com/zhuminghui/p/14741536.html>

第二个：

<https://blog.csdn.net/my_name_is_learn/article/details/109819127?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522167696171916800182755785%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=167696171916800182755785&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-109819127-null-null.142%5Ev73%5Ewechat,201%5Ev4%5Eadd_ask,239%5Ev2%5Einsert_chatgpt&utm_term=fastapi&spm=1018.2226.3001.4187>

第三个：