FastAPI 基础到实战

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌


黑马程序员
www.itheima.com
传智教育旗下
高端IT教育品牌

FastAPI 基础入门

基础程序、路由、请求与响应

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

FastAPI 是一个基于 Python 的高性能 Web 框架，专门用于快速构建 API 接口服务

FastAPI - 原生异步支持，释放真正性

同步与异步

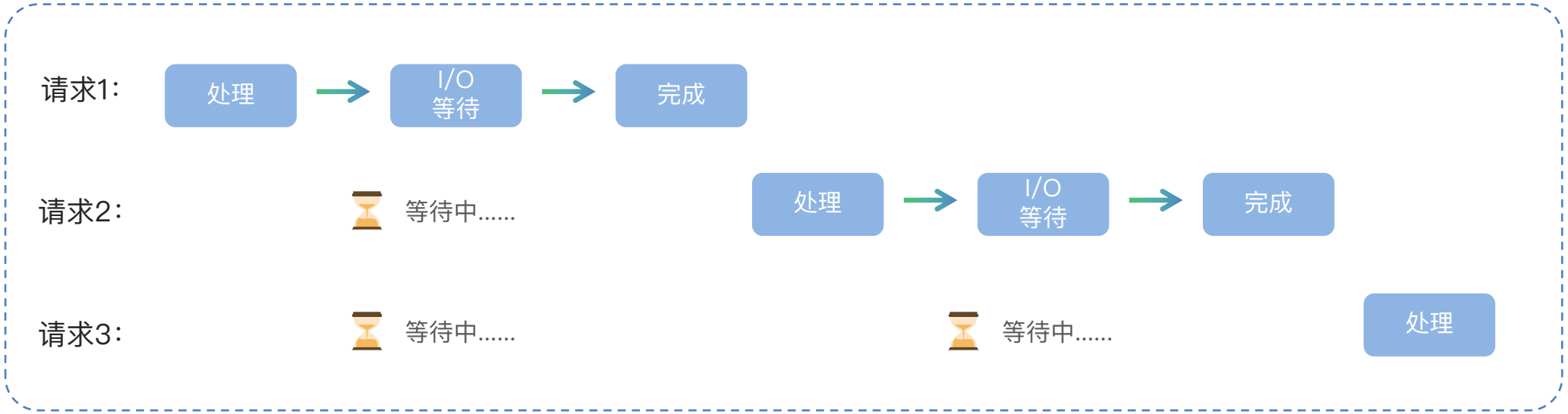
```python
@app.get("/sync")
def func_sync():
  start = time.time()
  for i in range(10):
    time.sleep(1)
  end = time.time()
  return {"time": f'{end-start:.2f}s'}
```

10秒+

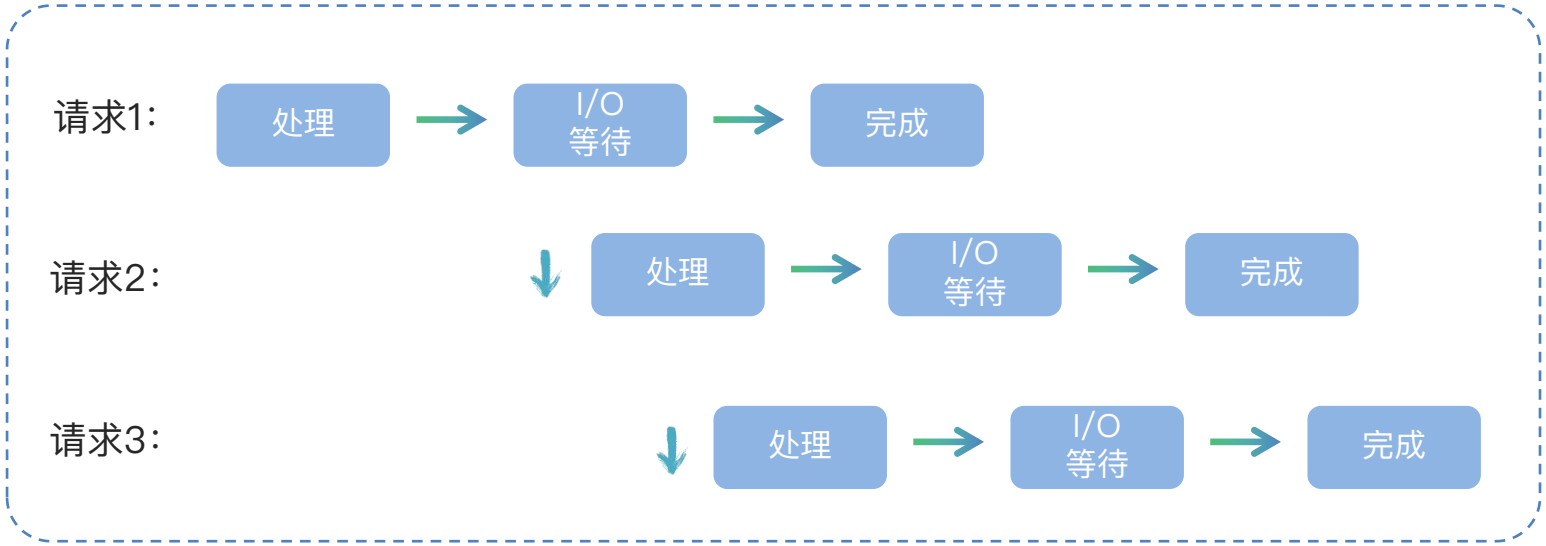
```python
@app.get("/async")
async def func_async():
  start = time.time()
  tasks = [asyncio.sleep(1) for i in range(10)]
  await asyncio.gather(*tasks)
  end = time.time()
  return {"time": f'{end-start:.2f}s'}
```

1秒

同步



异步



开发体验好

类型提示与验证

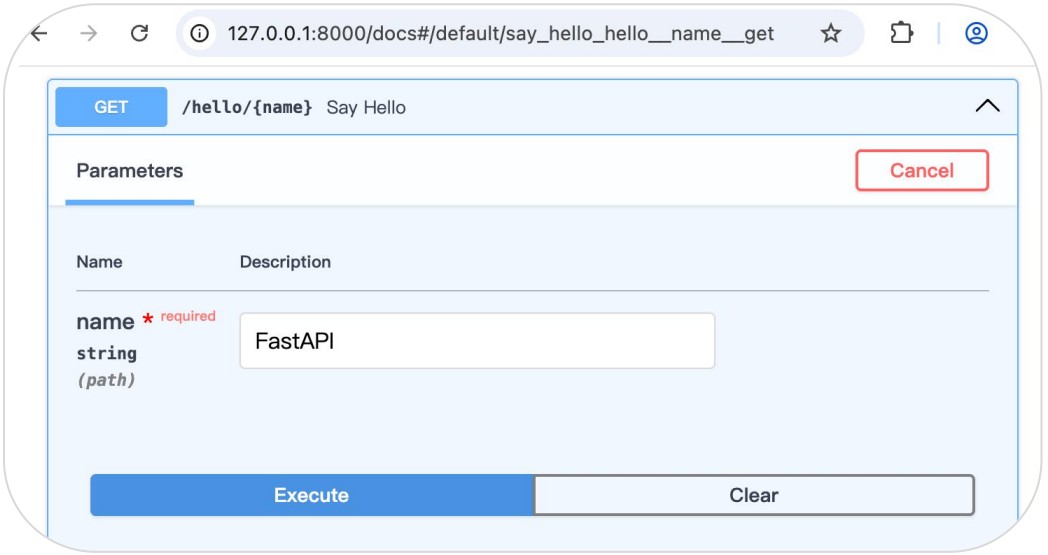
Pydantic 类型提示与验证，减少手动校验代码

```python
from pydantic import BaseModel
class User(BaseModel):
  username: str
  password: str

@app.post("/register")
async def register(user: User):
  return user
```

可交互式文档

自动生成可交互式文档，浏览器中直接调用和测试 API

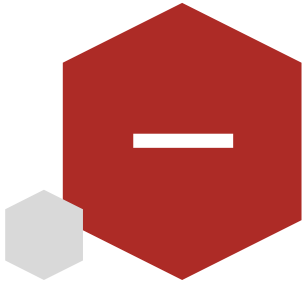


异步性能高

开发效率高

自动生成文档

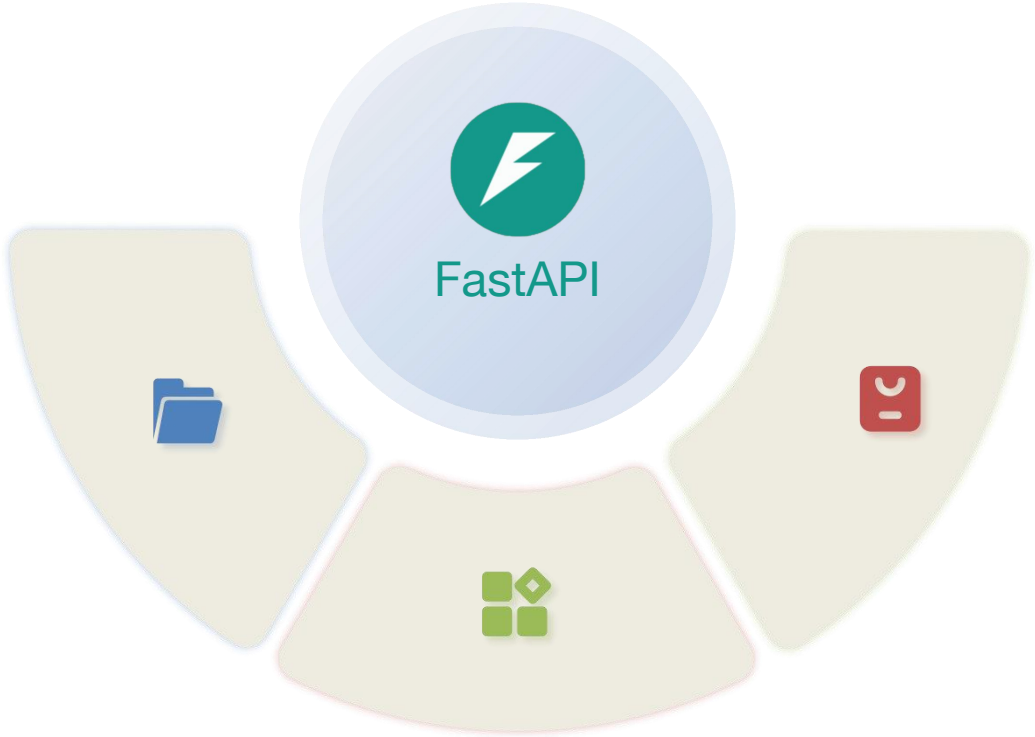

一起
出发
YIQI
CHUFA



FastAPI 框架基础

- 第一个 FastAPI 程序

使用 FastAPI 框架搭建 Web 服务



1. 创建项目

- FastAPI框架
- 虚拟环境

2. 运行项目

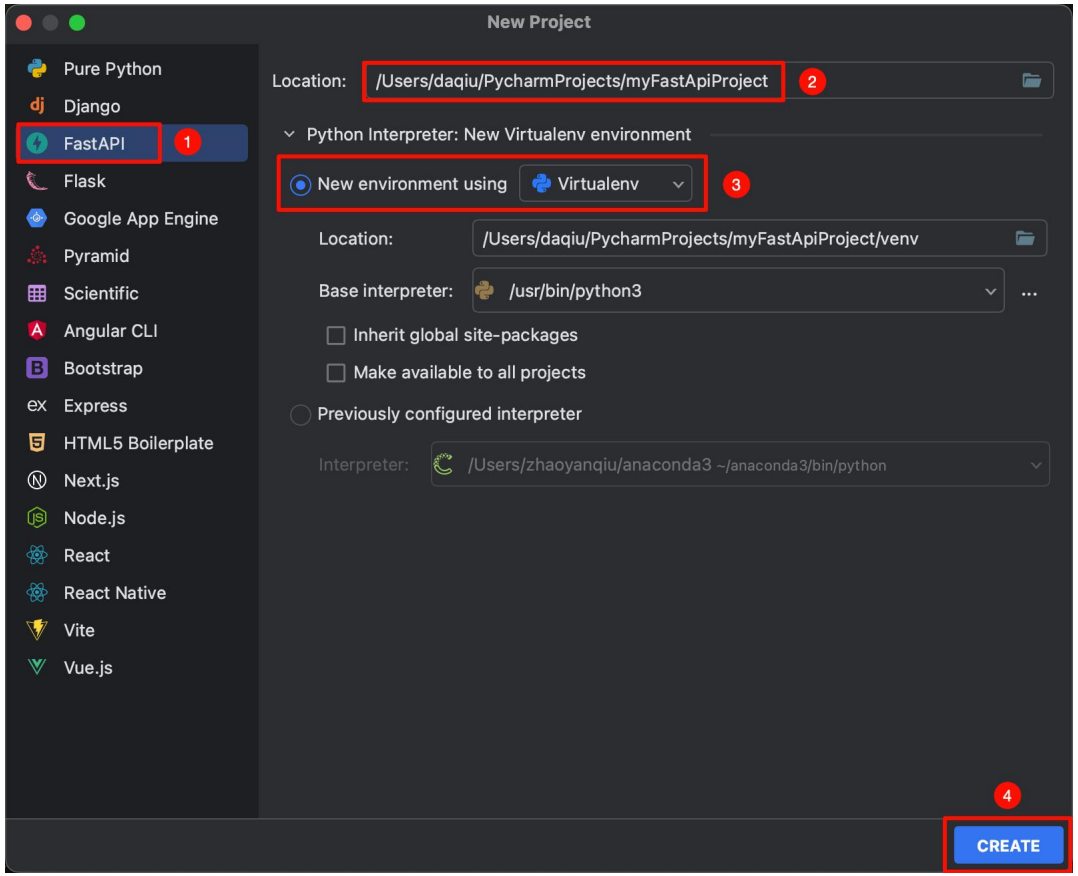
- run 项目
- uvicorn main:app --reload

3. 访问项目

- 访问路由
- 访问交互式文档

1. 创建项目

FastAPI → 存储位置及项目名称 → 创建虚拟环境 → Create



3. 访问项目


127.0.0.1:8000
美观输出
{
"message": "Hello World"
}
01
访问路由
前端请求数据的接口


127.0.0.1:8000/docs
default
GET
/ Root
Parameters
Try it out
02
访问交互文档
开发调试使用

- 为什么要创建虚拟环境?

  隔离项目运行环境，避免依赖冲突，保持全局环境的干净和稳定

- 怎么运行 FastAPI 项目?

  run 项目

  uvicorn main:app --reload　　--reload：更改代码后自动重启服务器

- 怎么访问 FastAPI 交互式文档?

  http://127.0.0.1:8000/docs

路由

http://127.0.0.1:8000/ → {"message":"Hello World"}

路由

路由就是 URL 地址和处理函数之间的映射关系，它决定了当用户访问某个特定网址时，服务器应该执行哪段代码来返回结果。

```
@app.get("/")
async def root():
  return {"message": "hello world"}
```

路由

FastAPI 的路由定义基于 Python 的装饰器模式

- 什么是路由?

  路由是 URL 地址和处理函数之间的映射关系

- 说出下方关键代码的含义?

FastAPI实例　请求方法　请求路径

装饰器

响应结果

```python
@app.get("/")
async def root():
    return {"message": "hello world"}
```

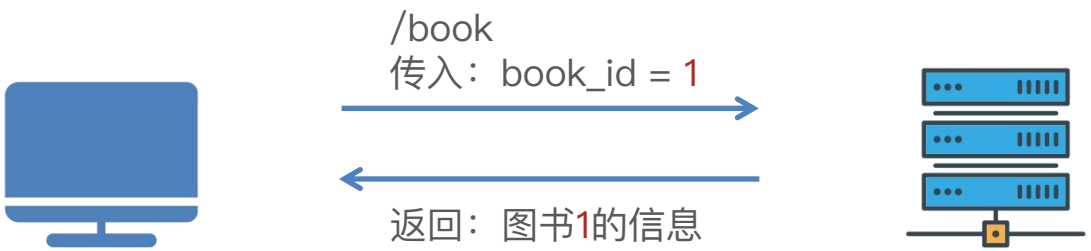
练习

- 路由

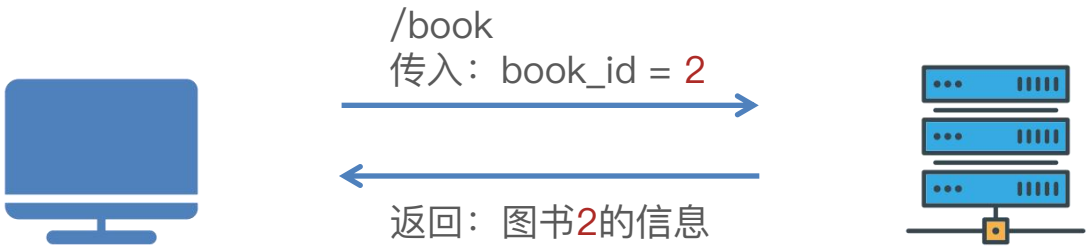
需求：访问路径 /user/hello，响应结果是 { "msg": "我正在学习 FastAPI ....." }

同一段接口逻辑，根据参数不同返回不同的数据

同一段接口逻辑，根据参数不同返回不同的数据



同一段接口逻辑，根据参数不同返回不同的数据



参数就是客户端发送请求时附带的额外信息和指令

参数的作用是让同一个接口能根据不同的输入，返回不同的输出，实现动态交互

参数分类

路径参数

位置：URL 路径的一部分
/book/{id}

作用：指向唯一的、特定的资源

方法：GET

查询参数

位置：URL? 之后
k1=v1&k2=v2

作用：对资源集合进行过滤、排序、分页等操作

方法：GET

请求体

位置：HTTP 请求的消息体（body）中

作用：创建、更新资源
携带大量数据，如：JSON

方法：POST、PUT 等

路径参数

路径参数

位置：URL 路径的一部分
/book/{id}

作用：指向唯一的、特定的资源

方法：GET

```python
@app.get("/book/{id}")
async def get_book(id: int):
  return {"id": id, "title": f"这是第{id}本书"}
```

参数类型注解

练习

- 路径参数

需求：以用户 id 为路径参数设计 URL，要求响应结果包含用户 id 和 名称（普通用户 id）

参考 URL：/user/{id}

响应结果：id: 123, name: 普通用户123

路径参数 – 类型注解 Path

FastAPI 允许为参数声明额外的信息和校验


```python
@app.get("/book/{id}")
async def get_book(id: int):
  return {"id": id, "title": f"这是第{id}本书"}
```

路径参数 – 类型注解 Path

导入 FastAPI 的 Path 函数

```python
@app.get("/book/{id}")
async def get_book(id: int = Path()):
    return {"id": id, "title": f"这是第{id}本书"}
```

Path 参数	说明
...	必填
gt/ge lt/le	大于/大于等于 小于/小于等于
description	描述
min_length max_length	长度限制

- 路径参数出现在什么位置?

  URL 路径的一部分　　/book/{id}

- 如何为路径参数添加类型注解?

  Python 原生注解 和 Path 注解

```python
@app.get("/book/{id}")
async def get_book(id: int):
  return {"id": id, "title": f"这是第{id}本书"}
```

```python
@app.get("/book/{id}")
async def get_book(id: int = Path()):
  return {"id": id, "title": f"这是第{id}本书"}
```

练习

- 路径参数 – 类型注解

需求：定义两个接口，携带路径参数，并使用 Path 来实现类型注解

具体如下：

- 接口1：以 新闻分类 id 为参数设计 URL，id 范围为 1 ~ 100
- 接口2：以 新闻分类名称为参数设计 URL，分类名称长度为 2 ~ 10

查询参数

声明的参数不是路径参数时，路径操作函数会把该参数自动解释为查询参数

查询参数

位置：URL? 之后
k1=v1&k2=v2

作用：对资源集合进行过滤、排序、分页等操作

方法：GET

```python
@app.get("/news/news_list")
async def get_news_list(skip: int, limit: int=10):
  return {"skip": skip, "limit": limit}
```

Request URL

```
http://127.0.0.1:8000/news/news_list?skip=0&limit=10
```

查询参数 – 类型注解 Query

导入 FastAPI 的 Query 函数

```python
@app.get("/user")
async def get_book(user_id: int = Query()):
  return {"id": id, "title": f"这是第{id}本书"}
```

Query 参数	说明
...	必填
gt/ge lt/le	大于/大于等于 小于/小于等于
description	描述
min_length max_length	长度限制

- 查询参数出现在什么位置?

  URL? 之后， k1=v1&k2=v2

Request URL

```
http://127.0.0.1:8000/news/news_list?skip=0&limit=10
```

- 如何为查询参数添加类型注解?

  Python 原生注解 和 Query 注解

```python
@app.get("/user")
async def get_book(user_id: int = Query()):
  return {"id": id, "title": f"这是第{id}本书"}
```

练习

- 查询参数

需求：设计接口查询图书，要求携带两个查询参数：图书分类和价格

参数具体要求：

- 图书分类：默认值为 Python开发，长度限制5 ~ 255
- 价格：限制大小范围 50 ~ 100

请求体参数

请求体

位置：HTTP 请求的消息体（body）中

作用：创建、更新资源

携带大量数据，如：JSON

方法：POST、PUT 等

在HTTP协议中，一个完整的请求由三部分组成：

① 请求行：包含方法、URL、协议版本

② 请求头：元数据信息（Content-Type、Authorization等）

③ 请求体：实际要发送的数据内容

请求体参数

1. 定义类型

```python
from pydantic import BaseModel
class User(BaseModel):
  username: str
  password: str
```

2. 类型注解

```python
@app.post("/register")
async def register(user: User):
  return user
```

```
curl -X 'POST' \
  'http://127.0.0.1:8000/register' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "username": "daqiu",
  "password": "123456"
}'
```

练习

- 请求体参数

需求：设计接口新增图书，图书信息包含：书名、作者、出版社、售价

请求体参数 – 类型注解 Field

导入 pydantic 的 Field 函数

```python
from pydantic import BaseModel, Field

class User(BaseModel):
  username: str = Field(...)
  password: str = Field(...)
```

Field 参数	说明
...	必填
gt/ge lt/le	大于/大于等于 小于/小于等于
default	默认值
description	描述
min_length max_length	长度限制

- 请求体参数的作用是什么?

  创建、更新资源

- 如何定义、使用请求体参数?

```python
from pydantic import BaseModel
class User(BaseModel):
  username: str
  password: str
```

```python
@app.post("/register")
async def register(user: User):
  return user
```

- 如何为请求体参数添加类型注解?

  Python 原生注解 和 Field 注解

练习

- 请求体参数

需求：设计接口新增图书，图书信息包含：书名、作者、出版社、售价

具体要求如下：

- 书名：不能为空；长度 2 ~ 20
- 作者：长度 2 ~ 10
- 出版社：默认值“黑马出版社”
- 售价：不能为空；价格大于0元

请求与响应

响应类型

默认情况下，FastAPI 会自动将路径操作函数返回的 Python 对象（字典、列表、Pydantic 模型等），经由 jsonable_encoder 转换为 JSON 兼容格式，并包装为 JSONResponse 返回。这省去了手动序列化的步骤，让开发者能更专注于业务逻辑。

如果需要返回非 JSON 数据（如 HTML、文件流），FastAPI 提供了丰富的响应类型来返回不同数据

```python
@app.get("/")
async def root():
    return {"message": "hello world"}
```

响应类型

默认情况下，FastAPI 会自动将路径操作函数返回的 Python 对象（字典、列表、Pydantic 模型等），经由 jsonable_encoder 转换为 JSON 兼容格式，并包装为 JSONResponse 返回。这省去了手动序列化的步骤，让开发者能更专注于业务逻辑。

如果需要返回非 JSON 数据（如 HTML、文件流），FastAPI 提供了丰富的响应类型来返回不同数据

响应类型	用途	示例
JSONResponse	默认响应，返回JSON数据	return {"key": "value"}
HTMLResponse	返回HTML内容	return HTMLResponse(html_content)
PlainTextResponse	返回纯文本	return PlainTextResponse("text")
FileResponse	返回文件下载	return FileResponse(path)
StreamingResponse	流式响应	生成器函数返回数据
RedirectResponse	重定向	return RedirectResponse(url)

响应类型

默认情况下，FastAPI 会自动将路径操作函数返回的 Python 对象（字典、列表、Pydantic 模型等），经由 jsonable_encoder 转换为 JSON 兼容格式，并包装为 JSONResponse 返回。这省去了手动序列化的步骤，让开发者能更专注于业务逻辑。

如果需要返回非 JSON 数据（如 HTML、文件流），FastAPI 提供了丰富的响应类型来返回不同数据

响应类型	用途	示例
JSONResponse	默认响应，返回JSON数据	return {"key": "value"}
HTMLResponse	返回HTML内容	return HTMLResponse(html_content)
PlainTextResponse	返回纯文本	return PlainTextResponse("text")
FileResponse	返回文件下载	return FileResponse(path)
StreamingResponse	流式响应	生成器函数返回数据
RedirectResponse	重定向	return RedirectResponse(url)

响应类型_JSON 格式

默认情况下，FastAPI 会自动将路径操作函数返回的 Python 对象（字典、列表、Pydantic 模型等），经由 jsonable_encoder转换为 JSON 兼容格式，并包装为 JSONResponse 返回。

```python
@app.get("/")
async def root():
    return {"message": "hello world"}
```

响应类型

默认情况下，FastAPI 会自动将路径操作函数返回的 Python 对象（字典、列表、Pydantic 模型等），经由 jsonable_encoder 转换为 JSON 兼容格式，并包装为 JSONResponse 返回。这省去了手动序列化的步骤，让开发者能更专注于业务逻辑。

如果需要返回非 JSON 数据（如 HTML、文件流），FastAPI 提供了丰富的响应类型来返回不同数据

响应类型	用途	示例
JSONResponse	默认响应，返回JSON数据	return {"key": "value"}
HTMLResponse	返回HTML内容	return HTMLResponse(html_content)
PlainTextResponse	返回纯文本	return PlainTextResponse("text")
FileResponse	返回文件下载	return FileResponse(path)
StreamingResponse	流式响应	生成器函数返回数据
RedirectResponse	重定向	return RedirectResponse(url)

响应类型设置方式

装饰器中指定响应类

场景：固定返回类型（HTML、纯文本等）

```
@app.get("/html", response_class=HTMLResponse)
async def get_html():
  return "<h1>这是标题</h1>"
```

返回响应对象

场景：文件下载、图片、流式响应

```
@app.get("/file")
async def get_file():
  file_path = "./files/1.jpeg"
  return FileResponse(file_path)
```

响应类型

默认情况下，FastAPI 会自动将路径操作函数返回的 Python 对象（字典、列表、Pydantic 模型等），经由 jsonable_encoder 转换为 JSON 兼容格式，并包装为 JSONResponse 返回。这省去了手动序列化的步骤，让开发者能更专注于业务逻辑。

如果需要返回非 JSON 数据（如 HTML、文件流），FastAPI 提供了丰富的响应类型来返回不同数据

响应类型	用途	示例
JSONResponse	默认响应，返回JSON数据	return {"key": "value"}
HTMLResponse	返回HTML内容	return HTMLResponse(html_content)
PlainTextResponse	返回纯文本	return PlainTextResponse("text")
FileResponse	返回文件下载	return FileResponse(path)
StreamingResponse	流式响应	生成器函数返回数据
RedirectResponse	重定向	return RedirectResponse(url)

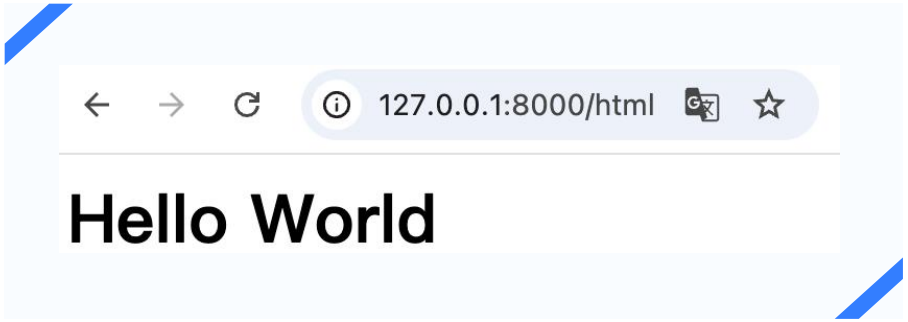
装饰器中指定响应类

响应 HTML 格式

设置响应类为 HTMLResponse，当前接口即可返回 HTML 内容

```python
from fastapi.responses import HTMLResponse

@app.get("/html", response_class=HTMLResponse)
async def get_html():
  return "<h1>Hello World</h1>"
```



响应类型

默认情况下，FastAPI 会自动将路径操作函数返回的 Python 对象（字典、列表、Pydantic 模型等），经由 jsonable_encoder 转换为 JSON 兼容格式，并包装为 JSONResponse 返回。这省去了手动序列化的步骤，让开发者能更专注于业务逻辑。

如果需要返回非 JSON 数据（如 HTML、文件流），FastAPI 提供了丰富的响应类型来返回不同数据

响应类型	用途	示例
JSONResponse	默认响应，返回JSON数据	return {"key": "value"}
HTMLResponse	返回HTML内容	return HTMLResponse(html_content)
PlainTextResponse	返回纯文本	return PlainTextResponse("text")
FileResponse	返回文件下载	return FileResponse(path)
StreamingResponse	流式响应	生成器函数返回数据
RedirectResponse	重定向	return RedirectResponse(url)

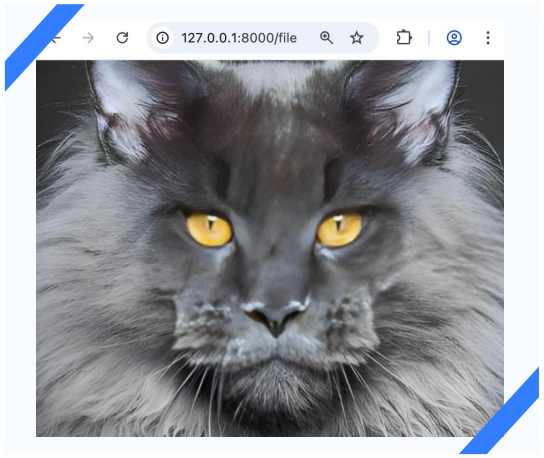
返回响应对象

响应文件格式

FileResponse 是 FastAPI 提供的专门用于高效返回文件内容（如图片、PDF、Excel、音视频等）的响应类。它能够智能处理文件路径、媒体类型推断、范围请求和缓存头部，是服务静态文件的推荐方式。

```python
from fastapi.responses import FileResponse

@app.get("/file")
async def get_file():
  file_path = "./files/1.jpeg"
  return FileResponse(file_path)
```



自定义响应数据格式

```python
@app.get("/news/{id}")
async def get_news(id: int):
  return {
    "id": id,
  }
```

自定义响应数据格式

```python
@app.get("/news/{id}")
async def get_news(id: int):
  return {
    "id": id,
    "title": f"这是第{id}本书"
  }
```

自定义响应数据格式

```python
@app.get("/news/{id}")
async def get_news(id: int):
  return {
    "id": id,
    "title": f"这是第{id}本书",
    "content": "这是一本好书"
  }
```


感觉哪里怪怪的

自定义响应数据格式

response_model 是路径操作装饰器（如 @app.get或 @app.post）的关键参数，它通过一个 Pydantic 模型来严格定义和约束 API 端点的输出格式。这一机制在提供自动数据验证和序列化的同时，更是保障数据安全性的第一道防线。

```python
from pydantic import BaseModel

class News(BaseModel):
  id: int
  title: str
  content: str

@app.get("/news/{id}", response_model=News)
async def get_news(id: int):
  return {
    "id": id,
    "title": f"这是第{id}本书",
    "content": "这是一本好书"
  }
```

FastAPI 内置响应类型

响应类型	用途	示例
JSONResponse	默认响应，返回JSON数据	return {"key": "value"}
HTMLResponse	返回HTML内容	return HTMLResponse(html_content)
PlainTextResponse	返回纯文本	return PlainTextResponse("text")
FileResponse	返回文件下载	return FileResponse(path)
StreamingResponse	流式响应	生成器函数返回数据
RedirectResponse	重定向	return RedirectResponse(url)

- FastAPI中，怎么自定义响应数据的格式?

  response_model

```python
from pydantic import BaseModel

class News(BaseModel):
  id: int
  title: str
  content: str

@app.get("/news/{id}", response_model=News)
async def get_news(id: int):
  return {
    "id": id,
    "title": f"这是第{id}本书",
    "content": "这是一本好书"
  }
```

FastAPI 内置响应类型

响应类型	用途	示例
JSONResponse	默认响应，返回JSON数据	return {"key": "value"}
HTMLResponse	返回HTML内容	return HTMLResponse(html_content)
PlainTextResponse	返回纯文本	return PlainTextResponse("text")
FileResponse	返回文件下载	return FileResponse(path)
StreamingResponse	流式响应	生成器函数返回数据
RedirectResponse	重定向	return RedirectResponse(url)

- FastAPI中，怎么设置响应类型?

  装饰器中设置响应类 和 返回响应对象

```python
@app.get("/html", response_class=HTMLResponse)
async def get_html():
  return "<h1>这是标题</h1>"
```

```python
@app.get("/file")
async def get_file():
  file_path = "./files/1.jpeg"
  return FileResponse(file_path)
```

异常处理

对于客户端引发的错误（4xx，如资源未找到、认证失败），应使用 fastapi.HTTPException 来中断正常处理流程，并返回标准错误响应 。

```python
from fastapi import FastAPI, HTTPException

@app.get('/news/{id}')
async def get_news(id: int):
  id_list = [1, 2, 3, 4, 5, 6]
  if id not in id_list:
    raise HTTPException(status_code=404, detail="当前id不存在")
  return {"id": id}
```

传智教育旗下高端IT教育品牌