FastApi

Machine Learning Architects Basel

Bassem Ben Hamed

December 2022

ML ARCHITECTS
BASEL

Agenda

- What is FastApi
- FastApi Hello World
- Path & Query Params
- Request Body

What is FastApi

High-performance: As the name suggests, FastAPI is fast. It's considered to be one of the fastest Python frameworks currently available.
Robust: You can create production-ready code using automatic interactive documentation.
Intuitive: FastAPI was designed to be easy to use and learn. It offers great editor support and documentation.
Quick to code: FastAPI increases your developing speed by 200%-300%.
Fewer bugs: It reduces around 40% of induced bugs.
Compatible: It works well with the open standards for APIS, OpenAPI (previously known as Swagger), and JSON schema.
Plugins: You can easily create plugins using dependency injection.
Type hints: You can use type hinting for data validation and conversion

FastApi Hello world

Line 1: We import FastAPI, which is a Python class that provides all the functionality for the API.
Line 3: We create an instance of the class FastAPI and name it app. This is the app referred to by uvicorn in the above command.
Line 5: We create a GET path.
Line 6: We define the function that will execute whenever someone visits the above path.
Line 7: We return a response to the client whenever the route is accessed.

```python
1  from fastapi import FastAPI
2
3  app = FastAPI()
4
5  @app.get("/")
6  def root ():
7      return {"message": "Hello World!"}
```

```
1  uvicorn main:app --reload
```

main: refers to the file name
app: refers to the object of FastAPI created inside the hello.py file
--reload: parameter that makes the server restart after the code changes

ML ARCHITECTS BASEL

The *query* is the set of key-value pairs that comes after the question mark ? in a URL, separated by an ampersand &.

Path & Query Params

Path parameters:

The value of the path parameter course_name will be passed to the function read_course() as the argument course_name.

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/courses/{course_name}")
def read_course(course_name):
    return {"course_name": course_name}
```

Query parameters:

The query is the set of key-value pairs that comes after the question mark ? in a URL, separated by an ampersand &.

```python
from fastapi import FastAPI

app = FastAPI()

course_items = [{"course_name": "Python"}, {"course_name": "SQLAlchemy"}, {"course_name":

@app.get("/courses/")
def read_courses(start: int, end: int):
    return course_items[start : start + end]
```

The *query* is the set of key-value pairs that comes after the question mark ? in a URL, separated by an ampersand &.

Request Body

Lines 1-3: We import the required packages.
Line 5: We declare the request data model.
Line 11: We create an instance of the FastAPI class.
Line 13: We create a POST path.
Line 14: We add the request data model to the path.

```
1  from typing import Optional
2  from fastapi import FastAPI
3  from pydantic import BaseModel
4
5  class Course(BaseModel):
6      name: str
7      description: Optional[str] = None
8      price: int
9      author: Optional[str] = None
10
11 app = FastAPI()
12
13 @app.post("/courses/")
14 def create_course(course: Course):
15     return course
```

Implementing reliable
machine learning solutions

ML ARCHITECTS
BASEL

Operating Models – Technologies – Culture & Skills

Consulting – Engineering - Training