Curso de FastAPI

Diego Saavedra Miguel Amaya

Jul 24, 2024

Table of contents

1 Bienvenido

¡Bienvenido al Curso Completo de FastAPI!

En este curso, exploraremos todo, desde los fundamentos hasta las aplicaciones prácticas.

1.1 ¿De qué trata este curso?

Este curso completo me llevará desde los fundamentos básicos de la programación hasta la construcción de aplicaciones prácticas utilizando los frameworks Django y la biblioteca de React.

A través de una combinación de teoría y ejercicios prácticos, me sumergiré en los conceptos esenciales del desarrollo web y avanzaré hacia la creación de proyectos del mundo real.

Desde la configuración del entorno de desarrollo hasta la construcción de una aplicación web de pila completa, este curso me proporcionará una comprensión sólida y experiencia práctica con FastAPI.

1.2 ¿Para quién es este curso?

Este curso está diseñado para principiantes y aquellos con poca o ninguna experiencia en programación.

Ya sea que sea un estudiante curioso, un profesional que busca cambiar de carrera o simplemente alguien que quiere aprender desarrollo web, este curso es para usted. Desde adolescentes hasta adultos, todos son bienvenidos a participar y explorar el emocionante mundo del desarrollo web con FastAPI.

1.3 ¿Cómo contribuir?

Valoramos su contribución a este curso. Si encuentra algún error, desea sugerir mejoras o agregar contenido adicional, me encantaría saber de usted.

Puede contribuir a través del repositorio en linea, donde puede compartir sus comentarios y sugerencias.

Juntos, podemos mejorar continuamente este recurso educativo para beneficiar a la comunidad de estudiantes y entusiastas de la programación.

Este ebook ha sido creado con el objetivo de proporcionar acceso gratuito y universal al conocimiento.

Estará disponible en línea para cualquier persona, sin importar su ubicación o circunstancias, para acceder y aprender a su propio ritmo.

Puede descargarlo en formato PDF, Epub o verlo en línea en cualquier momento y lugar.

Esperamos que disfrute este emocionante viaje de aprendizaje y descubrimiento en el mundo del desarrollo web con FastAPI!

Part I

Unidad 1: Introducción a Python

2 Introducción a FastAPI

Figure 2.1: FastAPI

2.1 ¿Qué es FastAPI?

FastAPI es un framework web moderno y rápido para crear APIs con Python 3.6+ basado en estándares abiertos y estándares de tipo de datos Python. Es muy rápido (alto rendimiento) gracias a Starlette y Pydantic. Es fácil de aprender y usar, y también es rápido en la ejecución.

2.2 ¿Por qué usar FastAPI?

- **Rápido**: Muy alto rendimiento, en par con NodeJS y Go (gracias a Starlette y Pydantic). Uno de los frameworks web más rápidos disponibles.
- **Rápido para escribir**: Aumenta la velocidad para escribir APIs web en comparación con otros frameworks, como Flask y Django.
- **Menos errores**: Reduce los errores de código (gracias a la comprobación estática de tipos Python).
- **Menos tiempo de depuración**: Reduce el tiempo de depuración con un sistema de comprobación estática de tipos Python.
- **Integración con Swagger UI**: Todos los endpoints automáticamente documentados con Swagger UI y Redoc.
- **Fácil de aprender**: Diseñado para ser fácil de aprender y usar. Menos tiempo leyendo documentación.

2.3 ¿Cómo instalar FastAPI?

Para instalar FastAPI, simplemente ejecuta el siguiente comando:

```
pyton -m venv env
```

```
source env/bin/activate
```

```
pip install fastapi
```

```
pip install uvicorn
```

2.4 ¿Cómo ejecutar FastAPI?

Para ejecutar FastAPI, simplemente ejecuta el siguiente comando:

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

Donde **main** es el nombre del archivo principal de tu aplicación y **app** es la instancia de FastAPI.

2.5 ¿Cómo probar FastAPI?

Para probar FastAPI, simplemente abre tu navegador y navega a la siguiente URL: http://127.0.0.1:8000

2.6 ¿Cómo probar la documentación de FastAPI?

Para probar la documentación de FastAPI, simplemente abre tu navegador y navega a la siguiente URL: http://127.0.0.1:8000/docs o http://127.0.0.1:8000/redoc

2.7 ¿Cómo crear una API con FastAPI?

Para crear una API con FastAPI, simplemente crea un archivo Python con el siguiente contenido:

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"Hello": "World"}
```

Guarda el archivo con el nombre **main.py** y ejecuta el siguiente comando:

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

2.8 ¿Cómo probar una API con FastAPI?

Para probar una API con FastAPI, simplemente abre tu navegador y navega a la siguiente URL: http://127.0.0.1:8000 y verás el siguiente resultado:

```
{"Hello": "World"}
```

2.9 ¿Cómo documentar una API con FastAPI?

Para documentar una API con FastAPI, simplemente abre tu navegador y navega a la siguiente URL: http://127.0.0.1:8000/docs o http://127.0.0.1:8000/redoc y verás la documentación de tu API.

3 ¿Qué sigue?

En la siguiente lección, aprenderemos cómo crear una API RESTful con FastAPI.

> **Tip**
>
> Para más información, visita la documentación oficial de FastAPI.

¡Hora de practicar!

Prueba tu conocimiento

4 Quiz FastAPI

4.1 Pregunta 1

¿Cuál es el propósito principal de FastAPI?

- ☐ Crear aplicaciones web rápidas y eficientes.
- ☐ Facilitar el desarrollo de APIs RESTful.
- ☐ Ambas opciones anteriores.

4.2 Pregunta 2

Cuál de las siguientes afirmaciones sobre FastAPI es correcta?

- ☐ FastAPI es un framework basado en Flask.
- ☐ FastAPI utiliza el lenguaje de programación Java.
- ☐ FastAPI es compatible con Python 3.7 y versiones posteriores.

4.3 Pregunta 3

Cuál de las siguientes características es una ventaja de FastAPI?

- ☐ Soporte para tipos de datos estáticos.
- ☐ Generación automática de documentación interactiva.
- ☐ Ambas opciones anteriores.

4.4 Pregunta 4

Cuál de las siguientes afirmaciones sobre FastAPI es incorrecta?

- ☐ FastAPI es un framework asincrónico.
- ☐ FastAPI es compatible con bases de datos SQL y NoSQL.
- ☐ FastAPI no es adecuado para aplicaciones de alto rendimiento.

4.5 Pregunta 5

¿Cuál es el comando correcto para instalar FastAPI?

- ☐ pip install fastapi
- ☐ pip install fastapi uvicorn
- ☐ pip install fastapi[all]

Respuestas

1. Ambas opciones anteriores.
2. FastAPI es compatible con Python 3.7 y versiones posteriores.
3. Ambas opciones anteriores.
4. FastAPI no es adecuado para aplicaciones de alto rendimiento.
5. **pip install fastapi uvicorn**

5 API Restful con FastAPI

Para crear un API Restful con FastAPI, empezamos creando el directorio **app** y dentro de este directorio creamos el archivo **main.py** con el siguiente contenido:

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"Hello": "World"}
```

En el directorio **app**, ejecutamos el siguiente comando para iniciar el servidor de FastAPI:

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

Para probar el API Restful, simplemente abrimos nuestro navegador y navegamos a la siguiente URL: http://127.0.0.1:8000 y veremos el siguiente resultado:

```
{"Hello": "World"}
```

Para probar la documentación interactiva de FastAPI, simplemente abrimos nuestro navegador y navegamos a la siguiente URL: http://127.0.0.1:8000/docs o http://127.0.0.1:8000/redoc y veremos la documentación de nuestro API.

Para más información sobre FastAPI, visita la documentación oficial de FastAPI.

6 ¿Qué sigue?

En la siguiente lección, aprenderemos cómo crear una API RESTful con FastAPI.

Prueba tu conocimiento

7 Quiz FastAPI

7.1 Pregunta 1

¿Qué devuelve el servidor de uvicorn?

- ☐ Un mensaje de error.
- ☐ Un mensaje de advertencia.
- ☐ Un mensaje de éxito.
- ☐ Un mensaje de información.

7.2 Pregunta 2

¿Para qué me sirve un json en FASTAPI?

- ☐ Para crear una API RESTful.
- ☐ Para crear una API SOAP.
- ☐ Para crear una API GraphQL.
- ☐ Para crear una API RESTful y una API SOAP.

7.3 Pregunta 3

¿Como ejecuto un servidor de FastAPI?

- ☐ uvicorn main:app –reload
- ☐ uvicorn main:app –no-reload
- ☐ uvicorn main:app –reload –port 8000
- ☐ uvicorn main:app –reload –port 8000 –host

7.4 Pregunta 4

¿Por qué debo instalar uvicorn?

- ☐ Para ejecutar FastAPI.
- ☐ Para ejecutar Flask.
- ☐ Para ejecutar Django.
- ☐ Para ejecutar FastAPI y Flask.

7.5 Pregunta 5

¿Qué comando debo ejecutar para instalar FastAPI y uvicorn?

- ☐ pip install fastapi
- ☐ pip install uvicorn
- ☐ pip install fastapi uvicorn
- ☐ pip install fastapi[all]

Respuestas

1. Un mensaje de éxito.
2. Para crear una API RESTful.
3. uvicorn main:app –reload
4. Para ejecutar FastAPI.
5. pip install fastapi uvicorn

8 Crear un API RESTful con FastAPI

En este capítulo aprenderemos a crear un API RESTful con FastAPI que nos permita crear un CMS (Content Management System) para gestionar artículos de un blog.

8.1 Crear un proyecto FastAPI

> Tip
>
> No olvides crear un entorno virtual para instalar las dependencias de tu proyecto.
>
> ```
> python -m venv venv
> source venv/bin/activate
> ```

Para crear un proyecto FastAPI, primero debemos instalar FastAPI y Uvicorn. Para ello, ejecuta los siguientes comandos:

```
pip install fastapi
pip install uvicorn
```

Crear un directorio para tu proyecto con el nombre **app** y dentro de este directorio crea un archivo **main.py** con el siguiente contenido:

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"Hello": "World"}
```

Para ejecutar el servidor de FastAPI, ejecuta el siguiente comando:

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

Abre tu navegador y navega a la siguiente URL: http://127.0.0.1:8000 y verás el siguiente resultado:

```
{"Hello": "World"}
```

Para probar la documentación interactiva de FastAPI, simplemente abre tu navegador y navega a la siguiente URL: http://127.0.0.1:8000/docs o http://127.0.0.1:8000/redoc y verás la documentación de tu API.

Para más información sobre FastAPI, visita la documentación oficial de FastAPI.

Ahora vamos a crear el CRUD (Create, Read, Update, Delete) para gestionar artículos de un blog.

8.2 Crear un modelo de datos

Crea un directorio **models** y dentro de este directorio crea un archivo **article.py** con el siguiente contenido:

```
from pydantic import BaseModel

class Article(BaseModel):
    id: int
    title: str
    content: str
```

8.3 Crear un controlador

Crea un directorio **controllers** y dentro de este directorio crea un archivo **article_controller.py** con el siguiente contenido:

```
from fastapi import APIRouter
from models.article import Article

router = APIRouter()

articles = [
    Article(id=1, title="First Article", content="This is the content of the first articl
    Article(id=2, title="Second Article", content="This is the content of the second arti
]

@router.get("/articles")
def read_articles():
    return articles

@router.get("/articles/{article_id}")
def read_article(article_id: int):
    for article in articles:
        if article.id == article_id:
```

```
            return article
    return {"message": "Article not found"}

@router.post("/articles")
def create_article(article: Article):
    articles.append(article)
    return article

@router.put("/articles/{article_id}")
def update_article(article_id: int, article: Article):
    for i, a in enumerate(articles):
        if a.id == article_id:
            articles[i] = article
            return article
    return {"message": "Article not found"}

@router.delete("/articles/{article_id}")
def delete_article(article_id: int):
    for i, article in enumerate(articles):
        if article.id == article_id:
            del articles[i]
            return {"message": "Article deleted"}
    return {"message": "Article not found"}
```

8.4 Crear un enrutador

Crea un directorio **routers** y dentro de este directorio crea un archivo **article_router.py** con el siguiente contenido:

```
from fastapi import APIRouter
from controllers.article_controller import router as article_router

router = APIRouter()

router.include_router(article_router)
```

8.5 Crear un servicio

Crea un directorio **services** y dentro de este directorio crea un archivo **article_service.py** con el siguiente contenido:

```
from models.article import Article

class ArticleService:
    def __init__(self):
```

```
        self.articles = [
            Article(id=1, title="First Article", content="This is the content of the firs
            Article(id=2, title="Second Article", content="This is the content of the sec
        ]

    def get_articles(self):
        return self.articles

    def get_article(self, article_id: int):
        for article in self.articles:
            if article.id == article_id:
                return article
        return None

    def create_article(self, article: Article):
        self.articles.append(article)
        return article

    def update_article(self, article_id: int, article: Article):
        for i, a in enumerate(self.articles):
            if a.id == article_id:
                self.articles[i] = article
                return article
        return None

    def delete_article(self, article_id: int):
        for i, article in enumerate(self.articles):
            if article.id == article_id:
                del self.articles[i]
                return True
        return False
```

8.6 Crear un controlador con servicios

Crea un directorio **controllers** y dentro de este directorio crea un archivo **article_controller.py** con el siguiente contenido:

```
from fastapi import APIRouter
from services.article_service import ArticleService
from models.article import Article

router = APIRouter()
article_service = ArticleService()

@router.get("/articles")
def read_articles():
```

```
    return article_service.get_articles()

@router.get("/articles/{article_id}")
def read_article(article_id: int):
    article = article_service.get_article(article_id)
    if article:
        return article
    return {"message": "Article not found"}

@router.post("/articles")
def create_article(article: Article):
    return article_service.create_article(article)

@router.put("/articles/{article_id}")
def update_article(article_id: int, article: Article):
    updated_article = article_service.update_article(article_id, article)
    if updated_article:
        return updated_article
    return {"message": "Article not found"}

@router.delete("/articles/{article_id}")
def delete_article(article_id: int):
    if article_service.delete_article(article_id):
        return {"message": "Article deleted"}
    return {"message": "Article not found"}
```

8.7 Crear un enrutador con controladores

Crea un directorio **routers** y dentro de este directorio crea un archivo **article_router.py** con el siguiente contenido:

```
from fastapi import APIRouter
from controllers.article_controller import router as article_router

router = APIRouter()

router.include_router(article_router)
```

8.8 Crear un archivo principal

Modifica un archivo **main.py** en el directorio **app** con el siguiente contenido:

```
from fastapi import FastAPI
from routers.article_router import router as article_router
```

```
app = FastAPI()

app.include_router(article_router)
```

Para ejecutar el servidor de FastAPI, ejecuta el siguiente comando:

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

Abre tu navegador y navega a la siguiente URL: http://127.0.0.1:8000/docs o http://127.0.0.1:8000/redoc y verás la documentación de tu API.

En la siguiente lección, aprenderemos cómo crear una API RESTful con FastAPI.

Prueba tu conocimiento‘;

9 Quiz FastAPI

9.1 Pregunta 1

¿Qué comando se utiliza para crear un entorno virtual en Python?

- ☐ python -m venv venv
- ☐ python -m venv env
- ☐ python -m venv virtualenv
- ☐ python -m venv env

9.2 Pregunta 2

¿Qué comando se utiliza para activar un entorno virtual en Python?

- ☐ source env/bin/activate
- ☐ source venv/bin/activate
- ☐ source env/activate
- ☐ source venv/activate

9.3 Pregunta 3

¿Qué código se utiliza para crear un método get en FASTAPI?

- ☐ @app.get(“/”)
- ☐ @app.post(“/”)
- ☐ @app.put(“/”)
- ☐ @app.delete(“/”)

9.4 Pregunta 4

¿Qué código se utiliza para crear un método post en FASTAPI?

- ☐ @app.get(“/”)
- ☐ @app.post(“/”)
- ☐ @app.put(“/”)
- ☐ @app.delete(“/”)

9.5 Pregunta 5

¿Qué código se utiliza para crear un método put en FASTAPI?

- ☐ @app.get(“/”)
- ☐ @app.post(“/”)
- ☐ @app.put(“/”)
- ☐ @app.delete(“/”)

Respuestas

1. python -m venv env
2. source env/bin/activate
3. @app.get(“/”)
4. @app.post(“/”)
5. @app.put(“/”)

Part II

Laboratorios

10 Introducción a FastAPI

Figure 10.1: FastAPI

10.1 ¿Qué es FastAPI?

FastAPI es un framework web moderno y rápido para crear APIs con Python 3.6+ basado en estándares abiertos y estándares de tipo de datos Python. Es muy rápido (alto rendimiento) gracias a Starlette y Pydantic. Es fácil de aprender y usar, y también es rápido en la ejecución.

10.2 ¿Por qué usar FastAPI?

- **Rápido**: Muy alto rendimiento, en par con NodeJS y Go (gracias a Starlette y Pydantic). Uno de los frameworks web más rápidos disponibles.
- **Rápido para escribir**: Aumenta la velocidad para escribir APIs web en comparación con otros frameworks, como Flask y Django.
- **Menos errores**: Reduce los errores de código (gracias a la comprobación estática de tipos Python).
- **Menos tiempo de depuración**: Reduce el tiempo de depuración con un sistema de comprobación estática de tipos Python.
- **Integración con Swagger UI**: Todos los endpoints automáticamente documentados con Swagger UI y Redoc.
- **Fácil de aprender**: Diseñado para ser fácil de aprender y usar. Menos tiempo leyendo documentación.

10.3 ¿Cómo instalar FastAPI?

Para instalar FastAPI, simplemente ejecuta el siguiente comando:

```
pyton -m venv env
```

```
source env/bin/activate
```

```
pip install fastapi
```

```
pip install uvicorn
```

10.4 ¿Cómo ejecutar FastAPI?

Para ejecutar FastAPI, simplemente ejecuta el siguiente comando:

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

Donde **main** es el nombre del archivo principal de tu aplicación y **app** es la instancia de FastAPI.

10.5 ¿Cómo probar FastAPI?

Para probar FastAPI, simplemente abre tu navegador y navega a la siguiente URL: http://127.0.0.1:8000

10.6 ¿Cómo probar la documentación de FastAPI?

Para probar la documentación de FastAPI, simplemente abre tu navegador y navega a la siguiente URL: http://127.0.0.1:8000/docs o http://127.0.0.1:8000/redoc

10.7 ¿Cómo crear una API con FastAPI?

Para crear una API con FastAPI, simplemente crea un archivo Python con el siguiente contenido:

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"Hello": "World"}
```

Guarda el archivo con el nombre **main.py** y ejecuta el siguiente comando:

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

10.8 ¿Cómo probar una API con FastAPI?

Para probar una API con FastAPI, simplemente abre tu navegador y navega a la siguiente URL: http://127.0.0.1:8000 y verás el siguiente resultado:

```
{"Hello": "World"}
```

10.9 ¿Cómo documentar una API con FastAPI?

Para documentar una API con FastAPI, simplemente abre tu navegador y navega a la siguiente URL: http://127.0.0.1:8000/docs o http://127.0.0.1:8000/redoc y verás la documentación de tu API.

11 ¿Qué sigue?

En la siguiente lección, aprenderemos cómo crear una API RESTful con FastAPI.

> **Tip**
>
> Para más información, visita la documentación oficial de FastAPI.

¡Hora de practicar!

Prueba tu conocimiento