WORLDLY KNOWLEDGE
INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCHERS
ISSN : 3030-332X IMPACT FACTOR (Research bib) – 7,293

Hasan Rustamovich Rasulov
Asia International University, teacher of the "General Technical Sciences" department

FASTAPI: A MODERN WEB FRAMEWORK FOR PYTHON

Abstract: This article analyzes FastAPI, a modern web framework for building APIs with Python. Detailed information is provided on FastAPI's main features, its asynchronous capabilities, data validation, and the use of Python type hints. The article covers how FastAPI compares with other frameworks like Flask and Django, its advantages in terms of speed and ease of use, and its applications for building modern web services. Recommendations are made on where FastAPI is best utilized, including real-time applications, microservices, and data-heavy APIs.

Keywords: FastAPI, Python, web framework, asynchronous programming, REST API, data validation, dependency injection, Pydantic, OpenAPI, performance, scalability

Introduction

FastAPI is a modern web framework for Python that enables developers to build high-performance APIs quickly and efficiently. It takes full advantage of Python type hints and asynchronous programming, providing automatic data validation, request parsing, and documentation generation through OpenAPI standards. FastAPI has gained significant popularity in recent years due to its speed, ease of use, and comprehensive feature set.

Asynchronous Programming in FastAPI

FastAPI is built on asynchronous Python frameworks like Starlette and Uvicorn, which allow non-blocking code execution. Asynchronous programming is essential for applications that handle many concurrent tasks, such as chat applications, live data processing, and any system requiring real-time performance.

In synchronous frameworks, tasks are executed one at a time, and requests can only be handled after previous requests finish. This creates bottlenecks for applications dealing with many simultaneous requests. FastAPI allows developers to use async and await syntax, taking full advantage of Python's asynchronous capabilities, which makes it faster than many traditional Python web frameworks like Flask and Django.

Below is an example of how asynchronous routes are implemented in FastAPI:

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/items/{item_id}")
async def read_item(item_id: int):
    return {"item_id": item_id}
```

In this example, the read_item function is asynchronous, meaning it can handle multiple requests concurrently, which improves performance in high-traffic environments.

Data Validation with Pydantic

One of the key features of FastAPI is its seamless integration with Pydantic, a data validation library that uses Python type hints to validate incoming requests. By defining Pydantic models, developers can ensure that their APIs receive and return structured, validated data, reducing the risk of errors.

Here is an example of how Pydantic is used for request validation:

```
from pydantic import BaseModel

class Item(BaseModel):
    name: str
    price: float
    is_offer: bool = None

@app.post("/items/")
async def create_item(item: Item):
    return item
```

In this example, the Item model ensures that any data sent to the /items/ endpoint follows the structure defined by the model, including correct types and required fields.

Dependency Injection in FastAPI

FastAPI supports dependency injection, allowing developers to define dependencies that are automatically resolved when required. This can be useful for injecting database connections, configurations, or other resources into the routes. Dependency injection in FastAPI improves code modularity and reusability.

An example of using dependency injection:

```
from fastapi import Depends

def get_db():
    db = DBSession()
    try:
        yield db
    finally:
        db.close()

@app.get("/users/")
async def get_users(db: Session = Depends(get_db)):
    users = db.query(User).all()
    return users
```

In this example, the get_db function is injected into the route as a dependency, ensuring that a database session is available to query data. FastAPI will automatically call get_db when needed and manage its lifecycle.

Security and Authentication

FastAPI offers robust tools for implementing security features, including OAuth2, JWT tokens, and API key-based authentication. The framework provides decorators to protect routes, ensuring that only authenticated users have access.

Here is a basic example of implementing OAuth2 authentication:

```
from fastapi.security import OAuth2PasswordBearer

oauth2_scheme = OAuth2PasswordBearer(tokenUrl="token")

@app.get("/users/me")
async def read_users_me(token: str = Depends(oauth2_scheme)):
    return {"token": token}
```

Performance Benchmarks

Numerous benchmarks demonstrate FastAPI’s superior performance compared to traditional Python frameworks. Its asynchronous capabilities allow it to handle tens of thousands of requests per second. For instance, FastAPI is capable of reaching speeds comparable to Node.js, while still maintaining the flexibility of Python’s ecosystem.

Deployment Strategies

FastAPI applications can be deployed using several strategies, including Docker containers, cloud services (AWS Lambda, Google Cloud), and traditional server setups. Uvicorn, the ASGI server on which FastAPI runs, is highly optimized for handling asynchronous web traffic.

Summary

FastAPI is a versatile and high-performance framework that has changed the landscape of Python web development. Its built-in support for asynchronous programming, automatic data validation, dependency injection, and OpenAPI-based documentation make it a go-to choice for building modern APIs. FastAPI is particularly well-suited for applications that require high throughput, low latency, and scalability, such as real-time services, microservices architectures, and data-driven APIs.

Used Literature:

1. Muxtaram Boboqulova Xamroyevna. (2024). THERMODYNAMICS OF LIVING SYSTEMS. Multidisciplinary Journal of Science and Technology, 4(3), 303–308.
2. Muxtaram Boboqulova Xamroyevna. (2024). QUYOSH ENERGIYASIDAN FOYDALANISH . TADQIQOTLAR.UZ, 34(2), 213–220.
3. Xamroyevna, M. B. (2024). Klassik fizika rivojlanishida kvant fizikasining orni. Ta'limning zamonaviy transformatsiyasi, 6(1), 9-19.
4. Xamroyevna, M. B. (2024). ELEKTRON MIKROSKOPIYA USULLARINI TIBBIYOTDA AHAMIYATI. PEDAGOG, 7(4), 273-280.
5. Boboqulova, M. X. (2024). FIZIKANING ISTIQBOLLI TADQIQOTLARI. PEDAGOG, 7(5), 277-283.23.Xamroyevna, M. B. (2024). RADIATSION NURLARNING INSON ORGANIZMIGA TASIRI. PEDAGOG, 7(6), 114-125.

6. Бобокулова Мухтарам. (2024). Альтернативные источники энергии и их использование. Междисциплинарный журнал науки и техники, 2 (9), 282-291.
7. Usmonov Firdavs. (2024). MINERAL ENRICHMENT PROCESSES. МЕДИЦИНА, ПЕДАГОГИКА И ТЕХНОЛОГИЯ: ТЕОРИЯ И ПРАКТИКА, 2(9), 250–260
8. 8. Jalilov, R., Latipov, S., Aslonov, Q., Choriyev, A., & Maxbuba, C. (2021, January). To the question of the development of servers of real-time management systems of electrical engineering complexes on the basis of modern automation systems. In CEUR Workshop Proceedings (Vol. 2843).
9. 9. Otajonova Sitorabonu. (2024). ПРИМЕНЕНИЕ ЭЛЕМЕНТОВ ТРИГОНОМЕТРИИ При РЕШЕНИИ ТРЕУГОЛЬНИКОВ. МЕДИЦИНА, ПЕДАГОГИКА И ТЕХНОЛОГИЯ: ТЕОРИЯ И ПРАКТИКА, 2(9), 292–304.
10. To'raqulovich, M. O. (2024). OLIY TA'LIM MUASSASALARIDA AXBOROT KOMMUNIKASIYA TEXNOLOGIYALARI DARSLARINI TASHKIL ETISHDA ZAMONAVIY USULLARDAN FOYDALANISH. PEDAGOG, 7(6), 63-74.
11. Muradov, O. (2024, January). IN TEACHING INFORMATICS AND INFORMATION TECHNOLOGIES REQUIREMENTS. In Международная конференция академических наук (Vol. 3, No. 1, pp. 97-102).
12. To'raqulovich, M. O. (2024). OLIY TA'LIM MUASSASALARIDA TA'LIMNING INNOVASION TEXNOLOGIYALARDAN FOYDALANISH. PEDAGOG, 7(5), 627-635.
13. To'raqulovich, M. O. (2024). IMPROVING THE TEACHING PROCESS OF IT AND INFORMATION TECHNOLOGIES BASED ON AN INNOVATIVE APPROACH. Multidisciplinary Journal of Science and Technology, 4(3), 851-859.
14. Murodov, O. (2024). DEVELOPMENT AND INSTALLATION OF AN AUTOMATIC TEMPERATURE CONTROL SYSTEM IN ROOMS. Solution of social problems in management and economy, 3(2), 91-94.
15. Вакаева Мехринисо. (2024). ИСПОЛЬЗОВАНИЕ ВИРТУАЛЬНЫХ ЛАБОРАТОРНЫХ РАБОТ В ОБРАЗОВАТЕЛЬНОМ ПРОЦЕССЕ И ИХ ПРЕИМУЩЕСТВА. Многопрофильный журнал науки и технологий, 2(9), 174–183.
16. Djuraevich, A. J. (2021). Zamonaviy ta'lim muhitida raqamli pedagogikaning o'rni va ahamiyati. Евразийский журнал академических исследований, 1(9), 103-107.
17. Ashurov, J. D. (2024). TA'LIM JARAYONIDA SUN'IY INTELEKTNI QO'LLASHNING AHAMIYATI. PEDAGOG, 7(5), 698-704.
18. Djo'rayevich, A. J. (2024). THE IMPORTANCE OF USING THE PEDAGOGICAL METHOD OF THE" INSERT" STRATEGY IN INFORMATION TECHNOLOGY PRACTICAL EXERCISES. Multidisciplinary Journal of Science and Technology, 4(3), 425-432.
19. Ashurov, J. D. (2024). AXBOROT TEXNOLOGIYALARI VA JARAYONLARNI MATEMATIK MODELLASHTIRISH FANINI O 'QITISHDA INNOVATSION YONDASHUVGA ASOSLANGAN METODLARNING AHAMIYATI. Zamonaviy fan va ta'lim yangiliklari xalqaro ilmiy jurnal, 2(1), 72-78.
20. Ashurov, J. (2023). OLIY TA'LIM MUASSASALARIDA "RADIOFARMATSEVTIK PREPARATLARNING GAMMA TERAPIYADA QO 'LLANILISHI" MAVZUSINI "FIKR, SABAB, MISOL, UMUMLASHTIRISH (FSMU)" METODI YORDAMIDA YORITISH. Центральноазиатский журнал образования и инноваций, 2(6 Part 4), 175-181.