Inspiration and Used Tools

What inspired to build **MSA SDK**, how it came to life and why i use an SDK in Micro Service Architectures.

Intro

MSA SDK wouldn't exist if not for the previous work of others, esp. **FastAPI** and **Dapr**.

There have been many tools created before that have helped inspire its creation.

I have been avoiding the creation of a SDK for several years. I always started with all the features covered by **FastAPI** and adding many different frameworks, plug-ins, and tools.

But at some point, the hustle to always start from scratch doing the same things slightly different all the times, each PoC or MVP was a new adventure, esp. since we now build systems with 10s or hundred's of microservices.

So i fell in love with FastAPI, Pydantic, Tyype hints and then SQLModel a while ago when i learned about Dapr. Dapr and FastAPI combined is simply developers heaven. So i build the SDK to ease management of all the versions and which tool to use for what, get rid of repeated implementations and ease to build many microservices which are part of one system and can run on any Cloud deployment option.

As i work in AI and mainly only build AI based automation solutions, i also added the Document Models which i typically need to process document content through AI systems.

The MSA SDK basically covers my toolbox which i use for any microservice and API, the bottom line layer. As i am not a 'real' programmer and tend to believe i just play with building blocks and maybe develop some glue between them, i decided to also make it Open Source.

Hope this helps or even inspire some people on how to use those awsome Open Source packages in real world applications.

Used by **MSA SDK** beside **FastAPI** and **Dapr**

Pydantic

Pydantic is a library to define data validation, serialization and documentation (using JSON Schema) based on Python type hints.

That makes it extremely intuitive.

It is comparable to Marshmallow. Although it's faster than Marshmallow in benchmarks. And as it is based on the same Python type hints, the editor support is great.

> ✔ **FastAPI uses it to**
>
> Handle all the data validation, data serialization and automatic model documentation (based on JSON Schema).
>
> **FastAPI** then takes that JSON Schema data and puts it in OpenAPI, apart from all the other things it does.

FastAPI and Starlette

FastAPI is build on top of Starlette, which is a lightweight ASGI framework/toolkit, which is ideal for building high-performance asyncio services.

It is a high performance, easy to learn, fast to code, ready for production framework.

It has:

- Seriously impressive performance.
- WebSocket support.
- In-process background tasks.
- Startup and shutdown events.
- Test client built on requests.
- CORS, GZip, Static Files, Streaming responses.
- Session and Cookie support.
- 100% test coverage.
- 100% type annotated codebase.
- Few hard dependencies.

Starlette is currently the fastest Python framework tested. Only surpassed by Uvicorn, which is not a framework, but a server.

Starlette provides all the basic web microframework functionality.

But it doesn't provide automatic data validation, serialization or documentation.

That's one of the main things that **FastAPI** adds on top, all based on Python type hints (using Pydantic). That, plus the dependency injection system, security utilities, OpenAPI schema generation, etc.

Technical Details

ASGI is a new "standard" being developed by Django core team members. It is still not a "Python standard" (a PEP), although they are in the process of doing that.

Nevertheless, it is already being used as a "standard" by several tools. This greatly improves interoperability, as you could switch Uvicorn for any other ASGI server (like Daphne or Hypercorn), or you could add ASGI compatible tools, like `python-socketio`.

FastAPI uses it to

Handle all the core web parts. Adding features on top.

The class `FastAPI` itself inherits directly from the class `Starlette`.

So, anything that you can do with Starlette, you can do it directly with **FastAPI**, as it is basically Starlette on steroids.

Uvicorn

Uvicorn is a lightning-fast ASGI server, built on uvloop and httptools.

It is not a web framework, but a server. For example, it doesn't provide tools for routing by paths. That's something that a framework (like Starlette or FastAPI) would provide on top.

It is the recommended server for Starlette, FastAPI and you guessed it **MSA SDK**.

FastAPI recommends it as

The main web server to run **FastAPI** applications.

You can combine it with Gunicorn, to have an asynchronous multi-process server.

Check more details in the Deployment section.

Last update: September 11, 2022

Created: September 11, 2022