Langserve

```
pip install fastapi
```

```
pip install uvicorn
```

```
pip install langserve
```

```
pip install sse_starlette
```

🧠 What is LangServe?

LangServe is a tool that helps you **turn your LangChain-based app into a website backend (an API)** that other apps (like your frontend, chatbot, or browser) can talk to.

> **LangChain chain (LLM app)** → served as a **web API endpoint**
> → that you (or others) can call from anywhere (frontend, JS, curl, etc.)

🌐 What does "Serve as a Web API" mean?

- Imagine your chatbot is ready. But you want others to use it from their browser or frontend code.
- LangServe helps you **expose your chain** as a **web URL** (like `http://localhost:8000/chat/invoke`) that others can send data to.

This is called an **API endpoint**.

So the full idea is:

> **Your LangChain logic → exposed to the web using LangServe → now anyone can send questions and get answers using URLs or frontend code.**

🧱 Why Use LangServe?

Feature	Explanation
🚀 Rapid Deployment	Serve chains as web APIs instantly with minimal code
🧪 Auto-generated Docs	FastAPI generates Swagger/OpenAPI docs automatically
⚙️ Flexible	Accepts input/output schemas, handles batches, streaming, etc.
🔐 Auth, CORS, etc.	Inherits FastAPI's full power for security, rate limits, etc.

FastAPI

Imports:

```
from fastapi import FastAPI
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_groq import ChatGroq
import os
from dotenv import load_dotenv
from langserve import add_routes
```

```
load_dotenv()
groq_api_key = os.getenv("GROQ_API_KEY")
```

Model:

```
model=ChatGroq(model="Gemma2-9b-It",groq_api_key=groq_api_key)
```

Langserve:

```
pip install "langserve[all]"
```

This installs:

- LangServe
- FastAPI
- Uvicorn (production server)
- Pydantic (input/output schemas)

`from langserve import add_routes` : Helps to create APIs

Write the previous translation code :

```
# 1. Create prompt template
system_template = "Translate the following into {language}:"
prompt_template = ChatPromptTemplate.from_messages([
    ('system', system_template),
    ('user', '{text}')
])

parser=StrOutputParser()

##create chain
chain=prompt_template|model|parser
```

App Definition

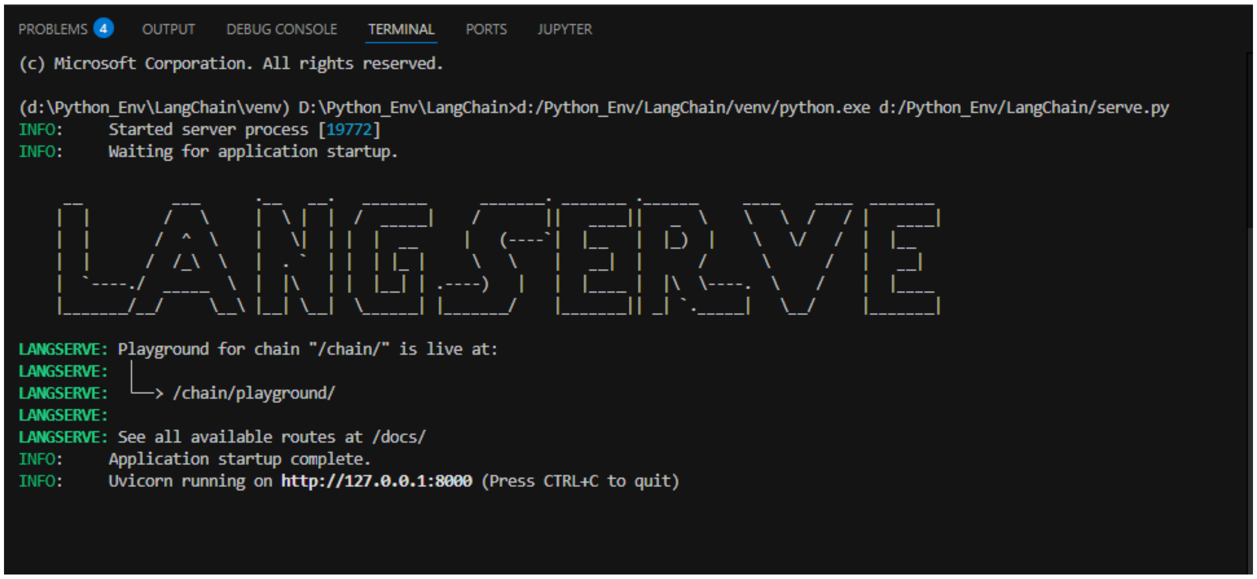
```
app=FastAPI(title="Langchain Server",
          version="1.0",
          description="A simple API server using Langchain runnable interface
s")
```

```
add_routes(
    app,
    chain,
    path = "/chain"
)
```

Execute the program:

```
if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app)
```

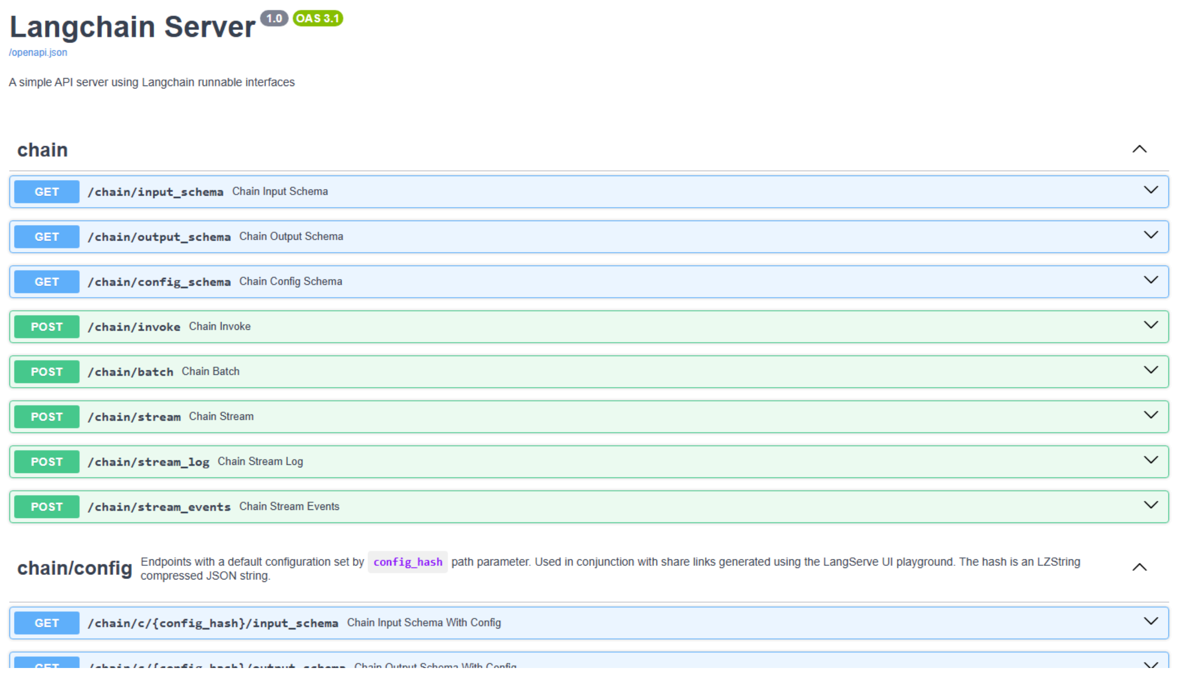
💡 **!!Do not run the above file in Jupyter Notebook**



```
(c) Microsoft Corporation. All rights reserved.

(d:\Python_Env\LangChain\venv) D:\Python_Env\LangChain>d:/Python_Env/LangChain/venv/python.exe d:/Python_Env/LangChain/serve.py
INFO:     Started server process [19772]
INFO:     Waiting for application startup.

 __          ___      .__   __.   _______      _______. _______ .______     ____    ____  _______
|  |        /   \     |  \ |  |  /  _____|    /       ||   ____||   _  \    \   \  /   / |   ____|
|  |       /  ^  \    |   \|  | |  |  __     |   (----`|  |__   |  |_)  |    \   \/   /  |  |__
|  |      /  /_\  \   |  . `  | |  | |_ |     \   \    |   __|  |      /      \      /   |   __|
|  `----./  _____  \  |  |\   | |  |__| | .----)   |   |  |____ |  |\  \----.  \    /    |  |____
|_______/__/     \__\ |__| \__|  \______| |_______/    |_______|| _| `._____|   \__/     |_______|

LANGSERVE: Playground for chain "/chain/" is live at:
LANGSERVE:  │
LANGSERVE:  └──> /chain/playground/
LANGSERVE:
LANGSERVE: See all available routes at /docs/
INFO:     Application startup complete.
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
```

Go to → http://127.0.0.1:8000/docs

💡 If the url doesn't work → `pip install pydantic==2.7.0 --force-reinstall`



Langchain Server 1.0 OAS 3.1
/openapi.json

A simple API server using Langchain runnable interfaces

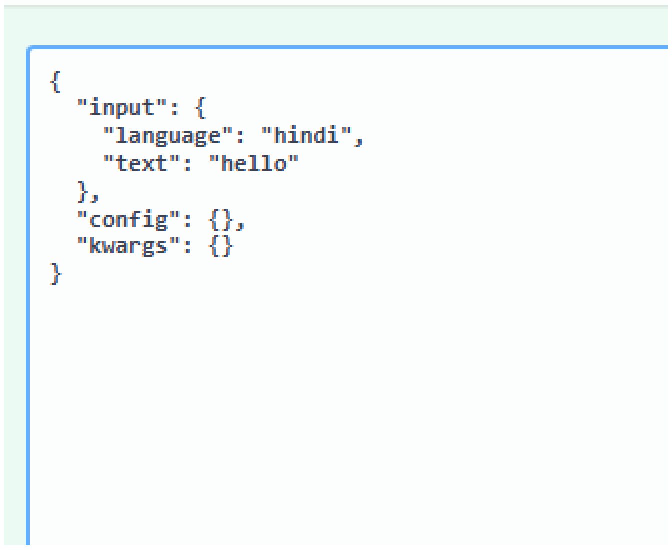
chain

GET	/chain/input_schema	Chain Input Schema
GET	/chain/output_schema	Chain Output Schema
GET	/chain/config_schema	Chain Config Schema
POST	/chain/invoke	Chain Invoke
POST	/chain/batch	Chain Batch
POST	/chain/stream	Chain Stream
POST	/chain/stream_log	Chain Stream Log
POST	/chain/stream_events	Chain Stream Events

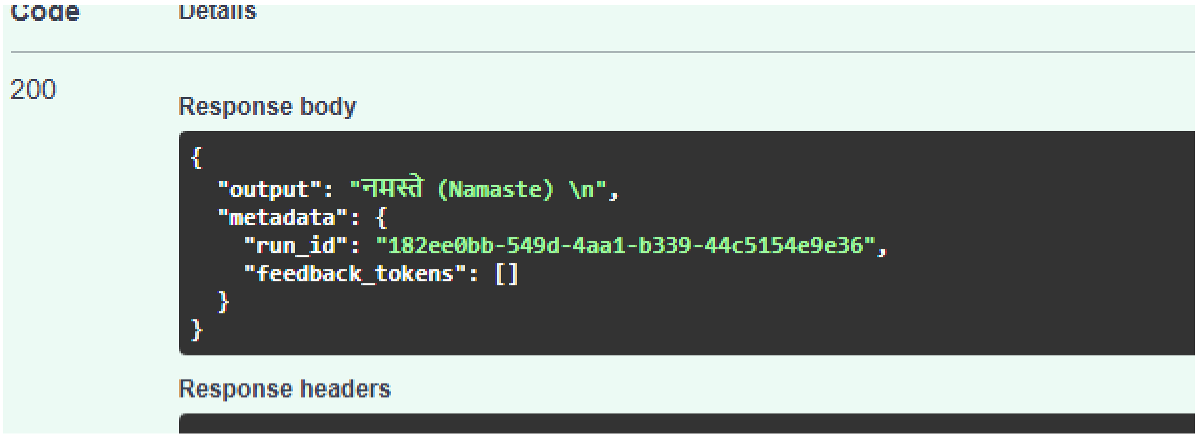
chain/config Endpoints with a default configuration set by `config_hash` path parameter. Used in conjunction with share links generated using the LangServe UI playground. The hash is an LZString compressed JSON string.

GET	/chain/c/{config_hash}/input_schema	Chain Input Schema With Config
GET	/chain/c/{config_hash}/output_schema	Chain Output Schema With Config

Try it out:



```
{
  "input": {
    "language": "hindi",
    "text": "hello"
  },
  "config": {},
  "kwargs": {}
}
```



Code Details

200 Response body

```
{
  "output": "नमस्ते (Namaste) \n",
  "metadata": {
    "run_id": "182ee0bb-549d-4aa1-b339-44c5154e9e36",
    "feedback_tokens": []
  }
}
```

Response headers