Flask: Esecuzione di Python lato Server

Prof. Fedeli Massimo

18 gennaio 2026

Indice

1 Introduzione

Flask è un *micro-framework* web per Python, leggero, flessibile e ideale per creare applicazioni web o API RESTful. Permette di eseguire codice Python lato server, rispondendo a richieste HTTP da parte di client (browser, app, altri servizi).

2 Caratteristiche Principali

- **Leggero**: solo il necessario, niente overhead.
- **Modulare**: si espande con estensioni (es. Flask-CORS, Flask-SQLAlchemy).
- **Python puro**: niente sintassi aggiuntiva o generatori di codice.
- **Werkzeug-based**: usa un server WSGI di sviluppo robusto.

3 Installazione

```
pip install flask
```

4 Struttura Minima di un'App Flask

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
return "Ciao da Flask!"

if __name__ == '__main__':
app.run(debug=True)
```

5 Routing e Decoratori

5.1 Route Statiche

```
@app.route('/about')
def about():
return "Pagina About"
```

5.2 Route con Parametri

```
@app.route('/user/<nome>')
def user(nome):
return f"Ciao, {nome}!"
```

5.3 Metodi HTTP

```
@app.route('/api/data', methods=['GET', 'POST'])
def data():
if request.method == 'POST':
return jsonify({"ricevuto": request.json})
return jsonify({"messaggio": "Dati disponibili"})
```

6 Gestione delle Richieste

6.1 JSON (API)

```
from flask import request, jsonify

@app.route('/predict', methods=['POST'])
def predict():
data = request.json
risultato = {"predizione": data["valore"] * 2}
return jsonify(risultato)
```

6.2 Form HTML

```
@app.route('/form', methods=['GET', 'POST'])
def form():
if request.method == 'POST':
nome = request.form['nome']
return f"Ciao, {nome}!"
return '''
<form method="post">
Nome: <input name="nome">
<input type="submit">
</form>
'''
```

7 Template HTML (Jinja2)

```
from flask import render_template

@app.route('/hello/<nome>')
def hello(nome):
return render_template('hello.html', nome=nome)
```

File: `templates/hello.html`

```
<!doctype html>
<html>
<body>
<h1>Ciao, {{ nome }}!</h1>
</body>
</html>
```

8 CORS: Accesso da Frontend

```
pip install flask-cors
```

```
from flask_cors import CORS

app = Flask(__name__)
CORS(app)  # Abilita CORS su tutte le route
```

9 Esempio Completo: API RESTful

```
from flask import Flask, request, jsonify
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

@app.route('/api/somma', methods=['POST'])
def somma():
data = request.json
a = data.get('a', 0)
b = data.get('b', 0)
return jsonify({"risultato": a + b})

if __name__ == '__main__':
app.run(debug=True)
```

Richiesta curl:

```
curl -X POST http://localhost:5000/api/somma \
-H "Content-Type: application/json" \
-d '{"a": 3, "b": 4}'
```

10 Deploy di Base

10.1 Ambiente di Produzione

- **Non usare** il server di sviluppo in produzione.
- Usa **Gunicorn** o **uWSGI** dietro **Nginx**.

```
pip install gunicorn
gunicorn app:app -b 0.0.0.0:8000
```

11 Conclusione

Flask è lo strumento ideale per:

- Creare API RESTful in Python

- Integrare modelli di machine learning in applicazioni web
- Prototipare rapidamente servizi backend
- Apprendere i concetti fondamentali del web development