التعامل مع الفريم ورك لبناء مواقع الإنترنتFlask

Flaskإطار عمل ويب خفيف لبايثون يُصنَّف كـ"Micro-framework" ، لأنه يوفّر لبنات أساسية مرنة دون فرض طبقات ثقيلة، ما يجعله مناسبًا للبداية السريعة والتعلّم التدريجي مقارنةً بأطر شاملة مثل .Django إن أردت مسارًا تعليميًا منظّمًا بالفيديو من الصفر إلى الاحتراف فهناك دورات عربية شاملة تغطي التنصيب، القوالب، التعامل مع قواعد البيانات، وبناء موقع كامل خطوة بخطوة.YouTube+1

لماذاFlask ؟

- **بساطة البداية :**إعداد مشروع وتجربة أول مسار Routing لا تتطلب بنية معقدة، مما يجعله خيارًا ممتازًا للمبتدئين في تطوير الويب باستخدام بايثون.
- **مرونة عالية :**تختار بنفسك مكتبات القوالب، قواعد البيانات، وإدارة النماذج، بدلًا من حلول مفروضة مسبقًا.
- **نظام قوالب :Jinja2** يتيح فصل منطق السيرفر عن العرض، مع توريث القوالب والعناصر الجزئية.
- **قابلية التوسعة :**تضيف مكتبات مثل SQLAlchemy و WTFormsو Flask-Loginبحسب حاجتك، لبناء تطبيقات كاملة دون إثقال المشروع منذ البداية.
- **موارد تعلّم عربية :**تتوفر سلاسل دروس عربية تبدأ بالمفاهيم الأساسية وتنتقل عمليًا لبناء تطبيق ويب متكامل

البدء والتثبيت

- **متطلبات أساسية :**إلمام بسيط ببايثون، و HTML/CSSيكفي للانطلاق، ثم توسّع تدريجيًا في مفاهيم الويب ومكتبات Flask الشائعة.
- **إنشاء بيئة عمل :**استخدم virtualenv أو venv لعزل الحزم، ثم ثبّت Flask عبر.pip
- **هيكلة مبدئية :**ملف تطبيق واحد ثم الانتقال إلى هيكلة وحدات/حزم مع نمو المشروع.

bash

```
# إنشاء بيئة افتراضية وتثبيتFlask
python -m venv .venv
source .venv/bin/activate  # على ويندوز: .venv\Scripts\activate
pip install flask
```

python

```
# app.py – أول تطبيق
from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "مرحباً بك في Flask!"

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

- **نصيحة تعلم :**إن أردت مسارًا تعليميًا بالفيديو مع مشروع عملي خطوة بخطوة، راجع الدورة العربية المتسلسلة من الصفر إلى الاحتراف.

التوجيه Routing والقوالبTemplates

- **تعريف المسارات :**تربط عنوان URL بدالة تعرض استجابة. تدعم المتغيرات داخل المسار والطرائق HTTPمثل.GET/POST
- **مُحَوِّلات المسار :**مثل int و stringو pathلضبط نوع المتغير.

python

```
from flask import Flask

app = Flask(__name__)

@app.route("/hello/<name>")
def hello(name):
    return f"أهلاً، {name}!"

@app.route("/post/<int:post_id>")
def show_post(post_id):
    return f"عرض المقال رقم {post_id}"
```

القوالب معJinja2

- **فصل العرض عن المنطق :**ضع HTML داخل مجلدtemplates ، واستخدم render_template لتمرير البيانات.
- **توريث القوالب :**أنشئ base.htmlوورّثه في الصفحات الفرعية لتوحيد الرأس والتذييل.
- **عناصر جزئية :**استخدم include للأجزاء المتكررة مثل شريط التنقل.

python

```
# app.py
from flask import Flask, render_template
app = Flask(__name__)

@app.route("/about")
def about():
    team = ["زيد", "سارة", "علي"]
    return render_template("about.html", team=team)
```

html

```
<!-- templates/base.html -->
<!doctype html>
<html lang="ar">
<head>
  <meta charset="utf-8">
  <title>{% block title %}موقعي{% endblock %}</title>
</head>
<body dir="rtl">
  {% include "partials/navbar.html" %}
  <main>{% block content %}{% endblock %}</main>
</body>
</html>
```

html

```
<!-- templates/about.html -->
```

```
{% extends "base.html" %}
{% block title %}عن الموقع{% endblock %}
{% block content %}
  <h1>فريق العمل</h1>
  <ul>
    {% for member in team %}
      <li>{{ member }}</li>
    {% endfor %}
  </ul>
{% endblock %}
```

- **لماذا هذا النهج؟** لأنه يسهّل تنظيم المشروع وتوسّعه، وهو ما يشجّع عليه مسار Flask التعليمي للمبتدئين.

مصادر :

النماذجForms ، قواعد البيانات، والمصادقة

- **POST وCSRF:** استخدم WTForms أو Flask-WTF لتوليد حقول وإدارة التحقق و CSRFبسهولة.
- **رفع الملفات:** اعمل عبر request.filesواضبط الحد الأقصى للحجم والمسارات الآمنة.

python

```
from flask import Flask, request, render_template, redirect, url_for
app = Flask(__name__)
app.config["SECRET_KEY"] = "change-me"

@app.route("/contact", methods=["GET", "POST"])
def contact():
    if request.method == "POST":
        name = request.form.get("name")
        message = request.form.get("message")
        # TODO: التحقق والحفظ
        return redirect(url_for("thanks"))
    return render_template("contact.html")

@app.route("/thanks")
def thanks():
    return "شكراً لتواصلك!"
```

قواعد البيانات

- **ORM شائع:** SQLAlchemy مع Flask-SQLAlchemy لإدارة النماذج والاستعلامات.
- **هجرة المخطط:** استخدم Flask-Migrate لمزامنة تغييرات الجداول عبر Alembic.

python

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config["SQLALCHEMY_DATABASE_URI"] = "sqlite:///site.db"
db = SQLAlchemy(app)

class User(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
email = db.Column(db.String(120), unique=True, nullable=False)
```

- **Flask-Login:** يدير تسجيل الدخول والخروج والجلسات، مع دوال مثل login_user و current_user.
- **تخزين كلمات المرور :** استخدم تجزئة قوية مثل werkzeug.security أو passlib.
- **تعزيز المفاهيم عمليًا :** تتناول الدورات العربية بناء موقع تعليمي شامل يشمل القوالب، قواعد البيانات، وإدارة المستخدمين خطوة بخطوة.

مصادر :

تنظيم المشروع، الإعدادات، والاختبارات

- **Blueprints:** تقسيم التطبيق لوحدات (auth ، blog، admin) لتسهيل التطوير والعمل الجماعي.
- **Factory pattern:** دالة create_app لضبط الإعدادات وربط الملحقات.

python

```
# myapp/__init__.py
from flask import Flask

def create_app(config_object="config.DevConfig"):
    app = Flask(__name__)
    app.config.from_object(config_object)

    from .auth import bp as auth_bp
    app.register_blueprint(auth_bp, url_prefix="/auth")

    return app
```

python

```
# myapp/auth.py
from flask import Blueprint

bp = Blueprint("auth", __name__)

@bp.route("/login", methods=["GET", "POST"])
def login():
    return "نموذج تسجيل الدخول"
```

إدارة الإعدادات

- **فصل البيئات** Dev/Testing/Prod **:** مع مفاتيح سرية عبر متغيرات البيئة.
- **أمان الإعدادات :** لا تُدخل الأسرار في المستودع؛ استخدم env. أو مدير أسرار.

اختبارات

- **pytest** و **Flask testing client:** اكتب اختبارات للمسارات، النماذج، وطبقة البيانات لضمان الاستقرار.
- **نهج تدريجي :** يبدأ التعلم بمشروع بسيط ثم تنظيم متقدّم مع نمو التطبيق، كما يظهر في المسارات التعليمية العربية.

مصادر :

النشر والأمان

- **سيرفر WSGI:** استخدم gunicorn أو uWSGI خلف عكس عكسي مثل Nginx.
- **منصات الاستضافة:** خيارات مثل Docker و Linux VPS وتمكين HTTPS عبر Let’s Encrypt.

- **CSRF/XSS:** فعّل CSRF في النماذج، واسمح فقط بالحقن الآمن في Jinja2.
- **الرؤوس الأمنية:** Content-Security-Policy، X-Frame-Options، و HSTS.
- **إدارة الجلسات:** ضبط مفاتيح سرية، تأمين الكوكيز بـ HttpOnly وSecure