

WMNBE-2203

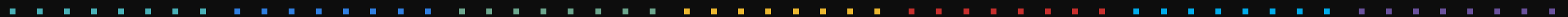
BACK-END DEVELOPMENT

Flask #1

Intro | Setup | Routing



ΠΕΡΙΕΧΟΜΕΝΑ



Περιεχόμενα

- Εισαγωγή
- Εγκατάσταση
 - **venv**
 - **pip**
 - **Flask**
- Πρώτα βήματα
 - **Routing**

FLASK

Flask

Γενικά

Το **Flask** είναι ένα *micro WSGI web application framework*, σχεδιασμένο για το γρήγορο και εύκολο "στήσιμο" του κομματιού *back-end* μιας web εφαρμογής.

Βασίζεται πάνω στα εργαλεία **Werkzeug** και **Jinja** και είναι ένα από τα πιο δημοφιλή *frameworks* ανάπτυξης web εφαρμογών.

Εγκατάσταση

Python

Για την εγκατάσταση, θεωρούμε δεδομένο ότι υπάρχει ήδη εγκατεστημένη η έκδοση **3.x** της **Python**.

```
> python --version
Python 3.10.4
```

Δημιουργούμε ένα φάκελο και "σετάρουμε" το **venv**.

```
$ mkdir flask_app
$ cd flask_app
$ python3 -m venv .venv
```

```
> python -m venv .venv
```

Εγκατάσταση

venv

Το εργαλείο **venv** είναι για τη δημιουργία ενός εικονικού περιβάλλοντος για τη φάση της ανάπτυξης μιας εφαρμογής.

Σε αυτό το περιβάλλον έχουμε πλήρη έλεγχο στα εργαλεία που είναι εγκατεστημένα και, πιο σημαντικό, στις εκδόσεις τους.

Πριν ξεκινήσουμε τη δουλειά σε ένα project, φροντίζουμε να ενεργοποιήσουμε το περιβάλλον αυτό.

```
$ . .venv/bin/activate
(.venv) $ _
```

```
> .venv\Scripts\activate
(.venv) > _
```

Εγκατάσταση

pip

Άλλο ένα εργαλείο που θα μας χρειαστεί στην εγκατάσταση είναι ο *package installer* της **Python**.

Μπορούμε να δούμε την έκδοση που υπάρχει ήδη (εγκαθίσταται μαζί με την **Python**) και προαιρετικά να αναβαθμίσουμε στην τελευταία.

```
$ python3 --version
$ python3 -m pip install --upgrade pip
```

```
> python --version
> python -m pip install --upgrade pip
```


Εγκατάσταση

Flask module

Ενώ βρισκόμαστε στο *virtual environment* που δημιουργήσαμε, εκτελούμε το παρακάτω:

```
(.venv) $ pip install Flask
```

ΠΡΩΤΑ ΒΗΜΑΤΑ

Πρώτα βήματα

Hello world!

hello.py

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

Πρώτα βήματα

Εκτέλεση

Για να "τρέξει" η εφαρμογή θα πρέπει να ορίσουμε το αρχείο το οποίο είναι το *entry point*.

Μιας και το **Flask** είναι ένα *micro framework*, μπορεί όλη η εφαρμογή να βρίσκεται σε ένα και μοναδικό αρχείο.

```
(.venv) $ export FLASK_APP=hello.py
(.venv) $ flask run
```

```
(.venv) > $env:FLASK_APP = "hello.py"
(.venv) > flask run
```

Πρώτα βήματα

Debug mode

Με την εντολή **flask run** ξεκινά ένας τοπικός *development server* αλλά κάθε φορά που γίνεται μία αλλαγή στον κώδικα, θα πρέπει ο *server* να επανεκκινείται.

Για το λόγο αυτό, όσο είμαστε στο στάδιο της ανάπτυξης, να ορίζουμε την παρακάτω ρύθμιση:

```
(.venv) $ export FLASK_ENV=development
(.venv) $ flask run
```

```
(.venv) > $env:FLASK_ENV = "development"
(.venv) > flask run
```

Πρώτα βήματα

Debug mode

Τι κάνει η ρύθμιση αυτή, σύμφωνα με το [documentation](#);

- *it activates the debugger*
- *it activates the automatic reloader*
- *it enables the debug mode on the Flask application.*

Routing

Γενικά

Σε μία σύγχρονη *web* εφαρμογή, συχνά αναφερόμαστε στα **URLs** ως **pretty / clean, meaningful** και **hackable**.

Για να επιτύχουμε όλα αυτά τα χαρακτηριστικά, χρειαζόμαστε τη βοήθεια των **routes**.

Η δήλωση ενός **route** γίνεται με τη χρήση ενός *decorator* (που μας παρέχει το **Flask**) πάνω σε συναρτήσεις.

```
@app.route('/url-segment')
def function_to_be_called_when_above_route_is_hit():
    statements
```

Routing

Παράδειγμα

```
@app.route('/')
def index():
    return '<h1>Home page</h1>'

@app.route('/hello')
def hello():
    return 'Hello, World'
```


Routing

Variables

Ένα **route** μπορεί να περιέχει και μεταβλητά τμήματα. Κάθε ένα από αυτά τα τμήματα θα πρέπει να αντιστοιχεί σε μία παράμετρο της *decorated* συνάρτησης.

```
@app.route('/hello/<name>')
def hello(name):
    return f'Hello, {name}!'
```

Routing

Default values

Η δήλωση περισσότερων του ενός **routes** πάνω σε μία συνάρτηση, είναι αποδεκτή. Σε αυτή την περίπτωση βοηθά και η δήλωση **default** τιμών για τις παραμέτρους.

```
@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name='World'):
    return f'Hello, {name}!'
```

Routing

Variable rules

Στη δήλωση των μεταβλητών αυτών μπορεί να προστεθεί και ένας **converter**, όπου ορίζεται ο τύπος της μεταβλητής.

Ένας **converter** λειτουργεί παράλληλα και ως περιορισμός.

Converter

string	(default) accepts any text without a slash
int	accepts positive integers
float	accepts positive floating point values
path	like string but also accepts slashes
uuid	accepts UUID strings

Routing

Παράδειγμα

```
@app.route('/product/<int:id>')
def product_details(id):
    # Get product details by it's id
    # id is already an int, no need to use int(id)
    return f'Show product details for product {id}: ...'
```

Χρήσιμα links

- Flask | The Pallets Projects

<https://palletsprojects.com/p/flask/>

- Welcome to Flask — Flask Documentation (2.1.x)

<https://flask.palletsprojects.com/en/2.1.x/>

- Installation — Flask Documentation (2.1.x)


<https://flask.palletsprojects.com/en/2.1.x/installation/>

- Quickstart — Flask Documentation (2.1.x)

<https://flask.palletsprojects.com/en/2.1.x/quickstart/>

Extra info

 Python and Flask Tutorial in Visual Studio Code
<https://code.visualstudio.com/docs/python/tutorial-fl...>

 The Art of Routing in Flask
<https://hackersandslackers.com/flask-routes/>

THANK YOU!