Web Services Python

Het Flask micro web framework (1)

Kristof Michiels

Flask?

Flask?

- Python web framework
- Micro-framework: zeer eenvoudig starten met 1 bestand en enkele regels code
- Web sites (deze week)
- API's (volgende week)
- url-routing
- templating systeem
- statische files tonen

Flask?

- Enorm flexibel
- WSGI-compatibel (Web Server Gateway Interface)
- Je moet alles zelf bouwen
- Geen standaard database-ondersteuning
- Django geeft daar meer mogelijkheden

Een eerste web-toepassing

Flask installeren

- Pycharm: creëren als Flask-toepassing (File -> New Project -> Flask)
- VSCode: pipenv!

Flask runnen

```
export FLASK_APP = main
export FLASK_ENV = development
flask run
```

- Enkel als je in de terminal werkt
- In Pycharm, gewoon met de Run-knop
- "app.py" als naam is de default app (dan hoeft de eerste regel niet)

Een eerste web-toepassing

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def start():
    return "<h1>Webfav.es - je favorieten, beschikbaar via het web</h1>"

@app.route("/over")
def over():
    return "<h1>Webfav.es - Over ons</h1><p>Hier komt informatie ...</p>"

@app.route("/contact")
def contact():
    return "<h1>Webfav.es - Contact</h1><p>Hier komt contactinformatie ...</p>"
```

Pagina templates met Jinja

- https://jinja.palletsprojects.com/en/3.0.x/
- Templates worden aan een folder templates toegevoegd

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def start():
    return render_template("start.html", naam="Kristof")
```

```
<h2>{{naam}}</h2>
```

Data in Flask krijgen via gebruikersinput met forms

```
<form action="jouw-webfave" method="get">
    <label for="url">Website URL</label>
      <input type="url" name="url" value="" required="">
      <br>
      <label for="naam">Naam</label>
      <input type="text" name="naam" value="" required="">
      <br>
      <input type="submit" value="Toevoegen">
  </form>
```

Let op de action en de method!

Form-variabelen in andere routes krijgen

```
from flask import Flask, render_template, request
...
@app.route('/jouw-webfave')
def jouw_webfave():
    return render_template('jouw-webfave.html', url=request.args['url'],
    naam=request.args['naam'])
```

```
<h2>Jouw Webfave: <a href="{{ url }}">{{naam}}</a></h2>
```

GET en POST requests gebruiken

```
<form action="jouw-webfave" method="post">
    ...
</form>
```

```
@app.route('/jouw-webfave', methods=['GET','POST'])
def jouw_webfave():
    if request.method == 'POST':
        return render_template('jouw-webfave.html', url=request.form['url'],
        naam=request.form['naam'])
    else:
        return 'Dit is niet geldig'
```

Flask data-flow

redirect en url_for

```
from flask import Flask, render_template, request, redirect, url_for

@app.route('/jouw-webfave', methods=['GET','POST'])
def jouw_webfave():
    if request.method == 'POST':
        return render_template('jouw-webfave.html', url=request.form['url'],
        naam=request.form['naam'])
    else:
        # wijziging!
        return redirect(url_for('start'))
```

Data opslaan in een JSON-bestand

```
{"ap": {"url": "http://www.ap.be"}}
```

```
import json
@app.route('/jouw-webfave', methods=['GET','POST'])
def jouw_webfave():
    if request.method == 'POST':
        webfaves = {}
        webfaves[request.form['naam']] = {'url': request.form['url']}
        with open('webfaves.json', 'w') as webfaves_file:
            json.dump(webfaves, webfaves_file)
        return render_template('jouw-webfave.html', url=request.form['url'],
        naam=request.form['naam'])
    else:
        return redirect(url_for('start'))
```

JSON-bestand parsen voor conflicterende entries

```
import os.path
if os.path.exists('webfaves.json'):
with open('webfaves.json') as webfaves_file:
    webfaves = json.load(webfaves_file)

if request.form['naam'] in webfaves.keys():
    return redirect(url_for('start'))
```

Feedback aan de gebruiker

```
from flask import Flask, render_template, request, redirect, url_for, flash

app.secret_key = 'qlslsmz342fki432ebee2ytee'

flash('Deze naam is reeds in gebruik. Selecteer een nieuwe naam aub')
```

```
{% for boodschap in get_flashed_messages() %}
<h2>{{ boodschap }}</h2>
{% endfor %}
```

File uploads

```
<form action="jouw-webfave" method="post" enctype="multipart/form-data">
    <label for="url">Website URL</label>
      <input type="url" name="url" value="" required="">
      <br>
      <label for="naam">Naam</label>
      <input type="text" name="naam" value="" required="">
      <br>
      <label for="file">Bestand</label>
      <input type="file" name="file" value="" required="">
      <br>
      <input type="submit" value="Toevoegen">
  </form>
```

File uploads

```
from werkzeug.utils import secure_filename

f = request.files['file']
volledige_naam = request.form['naam'] + secure_filename(f.filename)
f.save('/Users/kristofmichiels/PycharmProjects/webfaves-1/static/opgeladen/'
+ volledige_naam)
webfaves[request.form['naam']] = {'url': request.form['url'], 'file': volledige_naam}
```

Variabelen in URLs

```
@app.route('/<string:naam>')
def redirect_to_url(naam):
    if os.path.exists('webfaves.json'):
        with open('webfaves.json') as webfaves_file:
            webfaves = json.load(webfaves_file)
            if naam in webfaves.keys():
                if 'file' in webfaves[naam].keys():
                    return redirect(url_for('static', filename='opgeladen/' +
                    webfaves[naam]['file']))
                else:
                    return redirect(webfaves[naam]['url'])
```

Custom error messages

```
from flask import abort

return abort(404)

@app.errorhandler(404)
def pagina_niet_gevonden(error):
    return render_template('pagina_niet_gevonden.html'), 404
```

```
<h1>404 - Page Not Found</h1>
<p>We hebben deze naam niet kunnen vinden. Snel naar de startpagina ;-)</p>
<p><a href="{{ url_for('start') }}">Terug naar start</a></p>
```

Templates en styling

Sessies en cookies

```
from flask import session

def start():
    return render_template('start-v5.html', naam="Webfav.es", boodschap="Hallo",
    faves=session.keys())

with open('webfaves.json', 'w') as webfaves_file:
    json.dump(webfaves, webfaves_file)
    session[request.form['naam']] = True
return render_template('jouw-webfave.html', url=request.form['url'],
naam=request.form['naam'])
```

Sessies en cookies

Onderaan start.html

```
{% if faves %}
<h2>Webfaves die je hebt aangemaakt</h2>
<ul>
{% for fave in faves %}
<a href="{{ url_for('redirect_to_url', naam=fave) }}">
<li>{{ fave }}</li>
</a>
{% endfor %}
</ul>
{% endif %}
```

JSON API's: een voorproefje

- Eén van de beste aspecten aan Flask: zeer eenvoudig om API's te bouwen
- Flask heeft een goeie tool, jsonify: neemt een list of dictionary en zet ze om naar json

```
import jsonify

@app.route('/api')
def session_api():
    return jsonify(list(session.keys()))
```

Template blocks en base templates

templates/base.html

```
<head>
    <title>{% block titel %}{% endblock %}</title>
</head>

{% for boodschap in get_flashed_messages() %}
<h2>{{ boodschap }}</h2>
{% endfor %}

{% block main %}
{% endblock %}
```

Template blocks en base templates

```
{% extends 'base.html' %}

{% block titel %}URL Shortener{% endblock %}

{% block main %}

...

{% endblock %}
```

Schitterend voor bvb navigatie-elementen!

Labo

Labo 2

- Je exploreert de mogelijkheden van Flask zoals ze hier zijn meegegeven
- Je maakt zo economisch mogelijk een webtoepassing die achterliggend gebruik maakt van Storyblok
- Je zorgt voor navigatie en voor een aantal pagina's
- Maak je in Storyblok een nieuwe pagina aan, dan wordt dit automatisch aangepast in de navigatie
- De gebruiker kan doorklikken op een navigatie-element en komt op een nieuwe pagina terecht die content opvraagt uit Storyblok
- Je zorgt voor minstens één form die data verzamelt en toont op de manier die we hier hebben gezien
- Vrijblijvend: je deployt je toepassing online via eu.pythonanywhere.com