Python 2 Flask

Installatie

Met Python kan je ook web applicaties maken. Daarvoor zijn twee bekende frameworks beschikbaar, Django en Flask.

Django is een beetje de grote broer van Flask. Flask is eenvoudiger en beter geschikt voor wat eenvoudigere web applicaties.

Flask heeft niet echt een MVC structuur, maar heeft wel routing, een mooie template engine en database integratie.

Wij gaan kennismaken met Flask en zullen kijken naar routing en de (Jinja) template engine van Flask.

Maar eerst installeren.

Installatie

We hebben Python al geïnstalleerd (heb je dat niet meer vraag dan Chat GPT hoe je Python moet installeren of kijk in Python L1).

Met pip installeer je Flask:

```
pip install flask
```

Maak een map waarin je jouw Flask project gaat maken, en noem die map bijvoorbeeld mijFirstFlask.

Open de nieuwe folder in VCS.

Het eerste begin

1. Maak een nieuw Flask-project en maak een `templates` -map aan in de projectdirectory. Hierin plaatsen we de HTML-templates.

2. Maak een nieuw Python-bestand, bijvoorbeeld `app.py` , en plaats het in de projectdirectory.
3. Open het Python-bestand (`app.py`) met een teksteditor en voeg de volgende code toe:

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

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

4. Maak een nieuw HTML-bestand in de templates-map, index.html, en open het met een teksteditor.
   Voeg deze HTML-code toe aan het index.html-bestand en zet achter Welkom jouw naam.

```
<!DOCTYPE html>
<html>
<head>
    <title>Mijn Flask-project</title>
</head>
<body>
    <h1>Welkom <vul hier jouw naam in> bij mijn Flask-project!</h1>
    <p>Dit is een basisstructuur voor een Flask-project.</p>
</body>
</html>
```

5. Sla zowel het Python-bestand (`app.py`) als het HTML-bestand (`index.html`) op.
6. Ga naar de opdrachtprompt of terminalvenster en navigeer naar de projectdirectory waar het Python-bestand zich bevindt.
7. Voer het volgende commando in: `python app.py` (of `python3 app.py` als je meerdere Python-versies hebt geïnstalleerd).
8. Flask start de ontwikkelingsserver en geeft een URL weer, bijvoorbeeld `http://127.0.0.1:5000/` .
9. Open de weergegeven URL in je webbrowser en je zou de inhoud van het `index.html` -bestand moeten zien, inclusief de welkomstboodschap.

Inleveren

Een schermafdruk van je gehele browser waarin je laat zien dat de template de welkomstboodschap wordt getoond.

Flask Form

In deze opdracht ga je in Flask je een eenvoudige webpagina maken met een formulier. In het formulier vraag je de gebruiker om zijn naam en vervolgens gebruik je die naam voor een begroeting.

1. Maak een nieuw Python-bestand, bijvoorbeeld `app.py` , en open het met een teksteditor.
2. Voeg de volgende code toe aan het Python-bestand:

app.py

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/greet', methods=['POST'])
def greet():
    name = request.form.get('name')
    return render_template('greet.html', name=name)

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

Uitleg

Met **@app** wordt een route bepaald. Zo wordt er bijvoorbeeld bepaald dat als er wat naar */greet* wordt gepost, de functie *def greet():* wordt uitgevoerd.
render_template opent een *html tempate* en in het voorbeeld bij *greet()* wordt de variabele name meegegeven en die krijgt de waarde van name. (de eerste *name* is de naam die je in de template kan gebruiken en de tweede *name* is de variabele).
Heb je meer vragen over de code, zoek het dan op (internet, ChatGPT) of vraag het aan de docent.

3. Maak een `templates` -map in dezelfde directory als het Python-bestand.
4. Maak een nieuw HTML-bestand genaamd `index.html` in de `templates` -map en open het met een teksteditor.

5. Voeg de volgende code toe aan het `index.html` -bestand:

```
<!DOCTYPE html>
<html>
<head>
    <title>Flask Opdracht</title>
</head>
<body>
    <h1>Vul je naam in:</h1>
    <form action="/greet" method="POST">
        <input type="text" name="name" required>
        <input type="submit" value="Verzenden">
    </form>
</body>
</html>
```

6. Maak een nieuw HTML-bestand genaamd `greet.html` in de `templates` -map en open het met een teksteditor.
7. Voeg de volgende code toe aan het `greet.html` -bestand:

```
<!DOCTYPE html>
<html>
<head>
    <title>Flask Opdracht</title>
</head>
<body>
    <h1>Hallo, {{ name }}!</h1>
</body>
</html>
```

Uitleg

De templates in Flask, zijn JINJA tempaltes. Dit zijn HTML templates waarin variabelen kunnen worden afgedrukt. In dit voorbeeld {{ name }}.
Een JINJA template kan ook code bevatten dat staat tussen {% en %}, daarover later meer.

8. Sla zowel het Python-bestand (`app.py`), het `index.html` -bestand als het `greet.html` -bestand op.
9. Ga naar de opdrachtprompt of terminalvenster en navigeer naar de projectdirectory waar het Python-bestand zich bevindt.
10. Voer het volgende commando in: `python app.py.`
11. Flask start de ontwikkelingsserver en geeft een URL weer, `http://127.0.0.1:5000/` .

12. Open de weergegeven URL in je webbrowser en je zou een formulier moeten zien waar je je naam kunt invullen en verzenden.
13. Nadat je op "Verzenden" hebt geklikt, zou je begroet moeten worden met de boodschap "Hallo, [naam]!" waarbij `[naam]` de ingevoerde naam is.

Met deze eenvoudige opdracht kun je een formulier maken in Flask en de ingevoerde naam gebruiken om een begroeting weer te geven. Je kunt deze opdracht verder aanpassen en uitbreiden met meer functionaliteit en stijling naar wens.

Zorg ervoor dat je begrijpt hoe dit werkt, want in een volgende opdracht moet je zelf een formulier maken.

JINJA

De HTML-template is een zogenaamde JINJA template. In een JINJA template kan je python variabelen kan je afdrukken door de variabelen tussen {{ }} te zetten.

Als je (via ChatGPT) meer informatie wilt geef dan aan dat je in Flask met een JINJA template werkt.

Inleveren

1. Schermafdruk van de gehele browser van de pagina waarin je de naam moet invullen, en
2. schermafdruk van de gehele browser van de pagina waarin de boodschap wordt getoond.

Aanpassen app en form

Neem de code die je bij de vorige opdracht hebt gemaakt als uitgangspunt.

Pas de code die aan de route /greet is verbonden aan:

```
@app.route('/greet', methods=['POST'])
def greet():
```

```
current_time = datetime.now().strftime('%H:%M:%S')
    current_date = datetime.now().strftime('%Y-%m-%d')
name = request.form.get('name') return render_template('greet.html', name=name)
```

Je ziet dat er twee variabelen zijn bijgekomen, *current_time* en *current_date.*

In de HTML template moet je deze waarden afdrukken. Je moet daarvoor twee dingen doen:

1. Zorg ervoor dat je de waarden current_time en current_date doorgeeft aan de HTML-template.
   Pas daarvoor de laatste regel van de def greet(): aan.
   Gebruik Internet of ChatGPT om uit te vinden hoe dat moet.
2. Pas de Jinja template aat en zorg ervoor dat de datum en tijd wordt afgedrukt.

Inleveren

1. Aangepaste code in app.py (waar de def greet() in staat, en
2. aangepaste Jinja template, waar de datum en tijd wordt afgedrukt.

Welke dag?

Inleiding

Weet jij op welke dag van de week jij bent geboren?

We gaan een formulier waarin we de gebruiker om een datum vragen. Python bepaald dan op welke dag van de week deze datum valt en toont dat aan de gebruiker.

Jij voert dus (bijvoorbeeld) jouw geboortedag in en jouw programma berekend dan dat jij op een maandag bent geboren.

App.py

De app.py ziet er als volgt uit:

```
from flask import Flask, render_template, request
import datetime

app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        date_string = request.form['date']
        date_object = datetime.datetime.strptime(date_string, "%Y-%m-%d")
        day_as_number = date_object.weekday()
        days_of_week_array = ['Maandag', 'Dinsdag', 'Woensdag', 'Vrijdag', 'Zaterdag', 'Zondag']
        day_of_week = days_of_week_array[day_as_number]
        return render_template('template.html', day_of_week=day_of_week)
    return render_template('template.html')

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

De functie index() kijkt eerst of er een post is gedaan of dat er een pagina wordt opgevraagd. Als er een post is gedaan dan moet het resultaat worden berekend en worden getoond en anders moet de gebruiker eerst om input worden gevraagd (index.html).

Er zijn dus twee manieren om de functie te returnen: via de result.html template of via de index.html template.

Opdracht

Maak een werkende (Flask) app en plaats alle bestanden op de juiste plaats.

De templates die je moet gebruiken staan hieronder.Neem de code van de *app.py* over en vul de juiste template in. Vervang de 'template.html' voor de juiste template naam!

De templates zelf staan hieronder.

Je moet dus de volgende stappen uitvoeren:

1. Zet alle bestanden in de juiste folders en laat de Flask app werken.
2. Zorg dat de juiste templates worden aangeroepen; vervang hiervoor de 'template.html' in de app.py

3. Probeer te begrijpen hoe de app werkt en voorzie de app.py van commentaar.
   Plaats in ieder geval commentaar bij regels 9, 10 en 11 (de eerste drie regels na de if).
   Leg in eigen woorden uit wat deze regels doen en plaats dat in het commentaar.
4. Test of alles werkt; er zit één foutje in de code. Corrigeer deze fout en lever de gecorrigeerde code van app.py in.

Templates

We hebben twee templates, die moeten beiden in de template directory staan.

De index.html template ziet er als volgt uit:

```
<!DOCTYPE html>
<html>
<head>
    <title>Dag van de week</title>
</head>
<body>
    <h1>Vind de dag van de week</h1>
    <form method="POST" action="/">
        <label for="date">Voer een datum in:</label>
        <input type="date" id="date" name="date" required>
        <br><br>
        <input type="submit" value="Vind dag van de week">
    </form>
</body>
</html>
```

En de result.html ziet er als volgt uit:

```
<!DOCTYPE html>
<html>
<head>
    <title>Dag van de week</title>
</head>
<body>
    <h1>Dag van de week</h1>
    <p>De opgegeven datum valt op een: {{ day_of_week }}</p>
    <a href="/">Terug naar startpagina</a>
</body>
</html>
```

Inleveren

1. Zorg dat de app werkt en dat er commentaar wordt toegevoegd en lever alleen de app.py in.

Weersvoorspelling 1

We gaan een eenvoudige webapplicatie maken die een weersvoorspelling laat zien.

API

We halen de informatie van https://www.weatherapi.com registreer je daar voor een gratis account. Je krijgt dan een API key, zeg maar een sleutel die toegang geeft tot de weersvoorspelling.

Als je de key hebt dan gaan we verder.

Flask app

Maak een nieuwe Flask app. Maak een nieuwe folder en noem die bijvoorbeeld *weather* .

In *weather* maak je een bestand app.py en je maakt twee folders *templates* en *static* .

Screenshot 2023-07-11 210134.png

In het bestand app.py zetten we de volgende code.

```
from flask import Flask, render_template, request
import requests

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/forecast', methods=['POST'])
def forecast():
    country = request.form.get('country')
    city = request.form.get('city')
```

```
    api_key = 'your_weather_api_key'  # Vervang dit met je eigen WeatherAPI-sleutel

    url =
f'http://api.weatherapi.com/v1/forecast.json?key={api_key}&q={city},{country}&days=3'

    response = requests.get(url)
    data = response.json()

    forecast_days = data['forecast']['forecastday']

    return render_template('forecast.html', country=country, city=city,
forecast_days=forecast_days)

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

Plaats de persoonlijke API key die je hebt gekregen in de code.

De applicatie werkt als volgt:

1. Je gaat naar localhost:5000 en via de route functie in de app wordt de pagina geopend die in de template index.html staat.
2. Deze *index.html* is een form waarin de gebruiker wordt gevraagd om input; de naam van het land en de naam van de stad.
3. Het formulier wordt gepost naar de route /forecast en daar wordt het ingevoerd land en stad gebruikt om via de API een weersvoorspelling op te vragen.
4. De voorspelling wordt getoond met de template *forecast.html.*

Zorg dat je dit goed begrijpt want je moet zo een paar onderdelen zelf aanvullen.

Allereerst moet je de twee templates maken in de templates folder:

1. index.html
2. forecast.html

In index.html zet je de volgende code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Weersvoorspelling</title>
```

```
</head>
<body>
    <h1>Weersvoorspelling</h1>
    <form action="!!zet hier de juiste action in!!" method="POST">
        <label for="country">Selecteer een land:</label>
        !! Zet hier de html-code in om een land in te kunnen voeren.!!
        !! Zorg ervoor dat je juiste naam gebruikt, want die heb je nodig in de code die je aanr
        <label for="city">Selecteer een stad:</label>
        <input type="text" id="city" name="city" required><br><br>
        <input type="submit" value="Zoek weersvoorspelling">
    </form>
</body>
</html>
```

Let op in de index.html is het formulier niet helemaal af:

1. je moet nog invullen waarnaar toe het formulier moet worden gepost en;
2. je moet nog een <input> maken.

In de code staat met !! aangegeven waar je iets moet aanpassen. Dus alles tussen !! en !! moet je vervangen door eigen code!

In forecast.html zet je de volgende code.

```
<!DOCTYPE html>
<html>
<head>
    <title>Weersvoorspelling</title>
</head>
<body>
    <h1>Weersvoorspelling voor {{ city }}, {{ country }}</h1>
    <table>
        <tr>
            <th>Datum</th>
            <th>Min Temperatuur</th>
            <th>Max Temperatuur</th>
            <th>Weersomstandigheden</th>
        </tr>
        {% for day in forecast_days %}
            <tr>
                <td>{{ day['date'] }}</td>
                <td>{{ day['day']['mintemp_c'] }}°C</td>
                <td>{{ day['day']['maxtemp_c'] }}°C</td>
                <td>{{ day['day']['condition']['text'] }}</td>
            </tr>
        {% endfor %}
    </table>
</body>
</html>
```

De forecast template is helemaal compleet.

Pas de template index.html aan en maak de applicatie werkend.

Inleveren

1. De aangepast index.html.
2. Een screenshot waarin je de weers voorspelling laat zien van Amsterdam. Vul hiervoor bij land *Netherlands* en bij stad *Amsterdam.*

Zorg dat alles goed werkt want in de volgende opdracht gaan we onze app uitbreiden.

Voorbeeld

Screenshot 2023-07-11 211742.png

Weersvoorspelling 2

We nemen de code van de vorige opdracht als uitgangspunt.

API output bestuderen

Zet in de app.py na de regel waar de url variabele wordt gemaakt (waarschijnlijk regel 18), de vogelende regel code:

```
print(f"{url}")
```

Draai het programma nog een keer en zoek in het cmd window de url die je net hebt afgerdukt op.

Screenshot 2023-07-11 212742.png

De API key is hier onleesbaar gemaakt.

Kopieer de URL in de browser en bestudeer de JSON die je terug krijgt.

Als je goed kijkt dan zie je ook dat de tijd dat de zon opkomt en ondergaat ook in JSON staat (*sunrise* en *sunset*).

Pas de code aan zodat je de volgende output krijgt:

Screenshot 2023-07-11 213749.png

1. Bedenk eerst welk bestand (template) je moet aanpassen.
2. Zoek in de JSON hoe de structuur in elkaar zit en hoe je de tijden van *sunrise* en *sunset* kan ophalen.
3. Bereid de tabel uit met twee kolommen en vergeet de kolom headers <th> niet.
4. Maak met CSS of Bootsptrap (of een ander CSS framework) jouw tabel netjes zoals in het voorbeeld.

Succes!

Inleveren

1. forecast.html
2. een schermafdruk van de tabel waarin de twee extra kommen *Zon op* en *Zon onder* worden afgedrukt.

Weersvoorspelling 3

We gaan een laatste stukje toevoegen aan onze code.

De opdracht is om de voorspelling er als volgt uit te laten zien.

Screenshot 2023-07-11 215151.png

Wat er is bijgekomen zijn de drie icoontje boven aan de tabel.

1. In de JSON die de API terug geeft staat een link naar het juiste icoontje (zoek naar 'icon').
2. Gebruik die link om een <img> af te drukken in de template.

3. Zet dan de datum (die je al hebt want die wordt in de eerste kolom van de tabel afgedrukt), voor het icoontje.

Nu wordt het lastig.

Probeer de datum om te zetten in een dag van de week (eerst in het Engels) en als het je lukt daarna in het Nederlands. Gebruik internet en ChatGPT.

Inleveren

1. app.py
2. een schermafdruk van je geheel browser met daarin de de dag van de week en de icoontjes boven de tabel zoals in het voorbeeld is aangegeven.

Succes!

Revision #3
Created 2024-05-30 17:58:42 UTC by Max
Updated 2024-05-30 18:00:15 UTC by Max