Лабораторно упражнение 10

Разработване на уеб приложения с Flask

1. Въведение във Flask

Flask е micro web framework, написан на Python. Той е класифициран като micro framework, защото не изисква специални инструменти или библиотеки. Във Flask няма абстракционен слой за бази от данни, валидиране на формуляри или други компоненти, където вече съществуващите библиотеки на трети страни предоставят общи функции. Въпреки това Flask поддържа разширения, които могат да добавят функционалности на приложението, сякаш са вградени в самия Flask. Съществуват разширения за обектно-релационни картографи (Object-Relational Mappers - ORM), валидиране на формуляри и други.

Приложения, които използват Flask, включват например Pinterest и LinkedIn.

2. Структура на Flask приложение

2.1. Инсталиране на Flask

Инсталирането на Flask се изпълнява със следната команда:

```
pip install flask
```

2.2. Базова структура на Flask приложение

Минимално Flask приложение изглежда така:

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

В това приложение:

- Включваме Flask класа. Инстанция на този клас ще бъде нашият Web Server Gateway Interface (WSGI)
- Създаваме инстанция на Flask класа. Първият аргумент е името на приложението или модула. __name__ е удобен shortcut, подходящ за ситуацията. Това е нужно, за да знае Flask къде ще бъдат разположени ресурси на проекта като статични файлове и шаблони.

- Използваме route() декоратор, за да посочим на Flask кой URL трябва да задейства нашата функция.
- Функцията hello_world() връща съобщението, което искаме да покажем на браузъра на нашия потребител, като по подразбиране това съобщение е от тип HTML.

2.3. Стартиране на Flask приложение

За да стартираме приложение, трябва да посочим на терминала приложението, с което работим. Това се прави със задаване на подходяща стойност на променливата на средата (environment variable) FLASK_APP.

Установяването на FLASK_APP става по следния начин в различните командни интерпретатори:

CMD:

```
> set FLASK_APP=hello
> flask run
```

Powershell:

```
> $env:FLASK_APP = "hello"
> flask run
```

Bash:

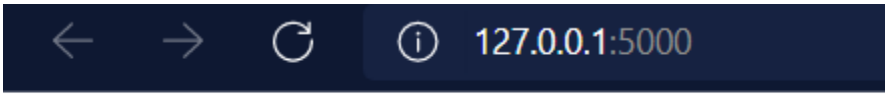
```
$ export FLASK_APP=hello
$ flask run
```

Fish:

```
$ set -x FLASK_APP hello
$ flask run
```

PyCharm включва вграден емулатор на терминал. Командите, които трябва да изпълним, са като тези в Powershell.

Готовото ни тестово приложение изглежда така:



Hello, World!

3. Добавяне на страници в приложението

```
from flask import Flask

app = Flask(__name__)
```

```
@app.route('/')
@app.route('/home')
def hello_world():
    return '<p> Hello World </p>'


@app.route('/about')
def about():
    return '<h1> This is an about page! </h1>'
```

4. HTML rendering.

Създаваме папка templates и в нея добавяме файл index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Index page!</title>
</head>
<body>
<h1> This is the index page! </h1>
</body>
</html>
```

След това можем да използваме функцията render_template(), която ни връща html страница с даден контекст.

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

Развитие на примера:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Index page!</title>
</head>
```

```
<body>
<h1> This is the index page! Hello, {{name}}! </h1>
</body>
</html>
```

```
@app.route('/')
@app.route('/home')
def home():
    name = 'Viktor'
    return render_template('index.html', name=name)
```

Функционалността на render_template() ни позволява да създадем layout страница, която да бъде “наследявана” от някои или всички други страници от уеб приложението ни. В тази layout страница обикновено се намират стилизиращите елементи на приложението, главно меню и т.н.

```
from flask import Flask
from flask import render_template

app = Flask(__name__)


@app.route('/')
@app.route('/home')
def home():
    name = 'Viktor'
    return render_template('index.html', name=name)


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

layout.html:

```
<!DOCTYPE html>
<html>
  <body>
    <header>
      <div class="container">
        <strong><nav>
```

```
          <ul class="menu">
            <li><a href="{{ url_for('home') }}">Home</a></li>
            <li><a href="{{ url_for('about') }}">About</a></li>
          </ul>
        </nav></strong>
      </div>
    </header>
    <div class="container">
      {% block content %}
      {% endblock %}
    </div>
  </body>
</html>
```

about.html:

```
{% extends 'layout.html' %}
{% block content %}
<h1> This is an about page! </h1>
{% endblock %}
```

index.html:

```
{% extends 'layout.html' %}
{% block content %}
<h1> This is the index page! Hello, {{name}}! </h1>
{% endblock %}
```

5. Plotly.

Plotly е компания, която разработва онлайн инструменти за анализ и визуализация на данни. Plotly предоставя инструменти за онлайн графики, анализи и статистика както и библиотеки за анализи и ефективни графични визуализации за Python, R, MATLAB, Perl, Julia, Arduino и REST.

5.1. Инсталиране на Plotly

pip install plotly

5.2. Добавяне на графики в приложение

За визуализацията на Plotly графики е нужно включването на съответната javascript библиотека. Можем да променим дотук направеното за демонстрация на работата с plotly.

Контролер на приложението

```
from flask import Flask
from flask import render_template
import plotly
import plotly.express as px
import pandas as pd
import json

app = Flask(__name__)



@app.route('/')
@app.route('/fruits')
def fruits():
    df = pd.DataFrame({
        "Fruit": ["Apples", "Oranges", "Bananas", "Apples", "Oranges", "Bananas"],
        "Amount": [4, 1, 2, 2, 4, 5],
        "City": ["SF", "SF", "SF", "Montreal", "Montreal", "Montreal"]
    })

    fig = px.bar(df, x="Fruit", y="Amount", color="City", barmode="group")

    graph = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
    return render_template('fruit_page.html', graph=graph, title='Fruits in North America')



@app.route('/vegetables')
def vegetables():
    df = pd.DataFrame({
        "Vegetables": ["Lettuce", "Cauliflower", "Carrots", "Lettuce", "Cauliflower", "Carrots"],
        "Amount": [10, 15, 8, 5, 14, 25],
        "City": ["London", "London", "London", "Madrid", "Madrid", "Madrid"]
    })

    fig = px.bar(df, x="Vegetables", y="Amount", color="City", barmode="stack")

    graph = json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)

    return render_template('vegetable_page.html', graph=graph, title='Vegetables in Europe')
```

layout.html:

```
<!DOCTYPE html>
<html>
  <head>
<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
  </head>
  <body>
  <title>{{title}}</title>
  <header>
   <div class="container">
     <strong><nav>
       <ul class="menu">
       <li><a href="{{ url_for('fruits') }}">Fruits</a></li>
   <li><a href="{{ url_for('vegetables') }}">Vegetables</a></li>
        </ul>
      </nav></strong>
    </div>
  </header>
  <div class="container">
    {% block content %}
    {% endblock %}
  </div>
  </body>
</html>
```

vegetable_page.html

```
{% extends 'layout.html' %}
{% block content %}
<h1>Information about vegetables in Europe</h1>
<div class="chart" id="bargraph">
    <script>
        var graphs = {{ graph | safe }};
        var config = {responsive: true};
        Plotly.newPlot('bargraph',  graphs.data,  graphs.layout,
config );
    </script>
</div>
{% endblock %}
```

fruit_page.html

```
 {% extends 'layout.html' %}
{% block content %}
<h1>Information about fruits in North America</h1>
<div class="chart" id="bargraph">
    <script>
        var graphs = {{ graph | safe }};
        var config = {responsive: true};
        Plotly.newPlot('bargraph', graphs.data, graphs.layout,
config );
    </script>
</div>
{% endblock %}
```

Самостоятелна задача

Добавете допълнителна страница с heatmap за земетресения по света.

Може да създадете този heatmap по следния начин:

```
df=pd.read_csv('https://raw.githubusercontent.com/plotly/dataset
s/master/earthquakes-23k.csv')

fig = px.density_mapbox(df, lat='Latitude', lon='Longitude',
z='Magnitude', radius=10,
                            center=dict(lat=0, lon=180), zoom=0,
                            mapbox_style="stamen-terrain")
```