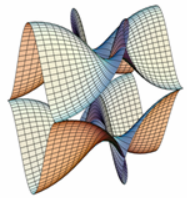

Prirodoslovno-matematički fakultet
Matematički odsjek
Sveučilište u Zagrebu

RAZVOJ WEB-APLIKACIJA

Predavanje 04 - Rute u Flasku. Jinja predlošci. Obrada formi.

10. listopada 2025.

Sastavio: Zvonimir Bujanović

Rute u Flasku

Statičke datoteke (slike, PDF-ovi, statički HTML, CSS, . . .)

- Smjestiti u poddirektorij `static`.
- Pristup preko *fizičke* adrese `/static/ime_datoteke`.

Sve ostale adrese (rute) su logičke.

- Aplikacija u Flasku treba definirati što se dogodi kada netko pristupi *logičkom* URL.

Donjim kodom definirana su dva logička URL-a: `/hello` i `/webpage`.

```
1  from flask import Flask;
2
3  app = Flask( __name__ );
4
5  @app.route( '/hello' )
6  def hello_world():
7      return 'Hello world!';
8
9  @app.route( '/webpage' )
10 def webpage():
11     return '<!DOCTYPE html><html><body>Hello world!</body></html>';
```

Rute u Flasku

Defaultna ruta aplikacije je `/`.

```
@app.route( '/' )
def index():
    return 'Glavna podstranica web-aplikacije.';
```

Rute mogu biti parametrizirane.

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

@app.route( '/article/<string:month>/<int:day>' )
def get_article( month, day ):
    return f'News for {month} {day}: ' + get_news( month, day );
```

Sada su definirane rute poput:

- `/hello/Ana`, `/hello/12345`;
- `/article/October/10`, ali ne i `/article/10/abc`.

Mogući tipovi parametara su još `float`, `path`, `uuid`.

Rute u Flasku

Možemo detektirati pristup nepostojećoj ruti.
Ispis sa `print` je vidljiv u konzoli gdje je pokrenut Flaskov server.

```
1 from flask import Flask, request;
2
3 @app.errorhandler( 404 )
4 def page_not_found(e):
5     url = request.path;
6     print( f'Pokušaj pristupa stranici {url}.' );
7
8     return f'Web-stranica {url} nije pronađena!';
```

Možemo preusmjeriti korisnika na drugi URL.

```
1 from flask import Flask, redirect;
2
3 @app.route( '/user/<string:username>' )
4 def user_homepage( username ):
5     if( not has_logged_in( username ) ):
6         return redirect( '/login' );
7     ...
```

HTML predlošci: Jinja2

Generiranje web-stranica unutar stringa je iznimno nepraktično.

Flask stoga koristi tzv. Jinja2 predloške.

```
1 from flask import Flask, render_template;
2
3 app = Flask( __name__ );
4
5 @app.route( '/' )
6 def index():
7     return render_template( 'index.html', name='Ana' );
```

Datoteku `index.html` sada treba staviti u poddirektorij `templates`.

- Uz obični HTML kod, ona može koristiti i dodatnu Jinja2 notaciju.
- Pomoću te notacije, moguće je ispisivati Python varijable u HTML.
- Glavna svrha: odvajanje logike aplikacije (Python) od HTML-a.

HTML predlošci: Jinja2

Jinja2 predlošci su HTML datoteke koje podržavaju dodatnu sintaksu.

- Pomoću `render_template` smo proslijedili varijablu `name`.
- Vrijednost te varijable možemo ispisati u HTML pomoću `{{name}}`.

Sadržaj datoteke `templates/index.html`:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/static/style.css">
    <title>Hello Flask!</title>
</head>
<body>
    Hello world! Hello, {{ name }}!
</body>
</html>
```

HTML predlošci: Jinja2

Jinja2 predlošci su HTML datoteke koje podržavaju dodatnu sintaksu.

- Možemo koristiti if-elif-else blokove.
- Možemo koristiti for petlje.
- Puni opis sintakse možete vidjeti ovdje.

```
...
<body>
    Hello world! Hello, {{ name }}!

    {% if( name == 'Ana' ) %}
        <h1>Ana!</h1>
    {% else %}
        <h6>Nije Ana...</h6>
    {% endif %}

    <p>
        {% for k in range(1, 6) %}
            {{ k }}
        {% endfor %}
    </p>
</body>
```

Zadatak 1

Napišite Flask aplikaciju koja:

- Na URL-u `/mnozi` generira HTML tablicu množenja brojeva od 1 do 10.
- Na URL-u `/mnozi/N` za cijeli broj N generira HTML tablicu množenja brojeva od 1 do N.

*	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

HTML forme: metode GET i POST

Kada definiramo HTML formu možemo odabrati metodu GET ili POST.

- To određuje tip HTTP zahtjeva kojim će podaci iz forme biti poslani serveru.

GET zahtjevi:

- Parametri su vidljivi iz URL, kroz tzv. *query string*: https://www.example.com?ime=Pero&starost=20
- Mogu biti spremljeni u cache, ostaju u povijesti browsanja, mogu se *bookmarkirati*.
- Imaju ograničenje na duljinu parametara.
- Nikad ih se ne koristi za osjetljive podatke (passwordi i slično).
- Trebaju se koristiti se samo za dohvaćenje podataka sa servera.

HTML forme: metode GET i POST

GET zahtjevi, npr. https://www.example.com?ime=Pero&starost=20

Pristup iz Flaska:

- `from flask import request;`
- `request.args` je `MultiDict` koji sadrži podatke poslane pomoću GET.

```
# Sljedeći izrazi su True.
request.args.get('ime') == 'Pero'
request.args['ime'] == 'Pero'

request.args.get('starost') == '20'          # Pazi, string!
request.args.get('starost', type=int) == 20 # Sada je int!

request.args.get('ovognema') == None
request.args.get('ovognema', type=int, default=0) == 0
# Pokušaj pristupa request.args['ovognema'] sruši aplikaciju.
```

HTML forme: metode GET i POST

POST zahtjevi:

- Parametri nisu vidljivi iz URL.
- Parametre možemo vidjeti u Firefoxu ovako:
  - Web developer tools (F12) -> Network -> klik na zahtjev -> Request.
- Tipično se koristi za slanje podataka unesenih u formu.
- Osjetljivi podaci su i dalje bez enkripcije ⇝ HTTPS.
- Nikad se ne spremaju u cache, ne ostaju u povijesti browsanja, ne mogu se *bookmarkirati*.
- Nema ograničenja na dužinu podataka koji se šalju.
- Mogu se koristiti i za dohvaćanje i za spremanje podataka na server.

HTML forme: metode GET i POST

POST zahtjevi, npr. forma koja sadrži textboxeve s atributima `name` jednakim "ime" i "starost" u koje je korisnik unio redom 'Pero' i '20'.

Pristup iz Flaska:

- `from flask import request;`
- `@app.route('/skripta', methods=['POST'])`
- `request.form` je `MultiDict` koji sadrži podatke poslane pomoću POST.

```
1  # Sljedeći izrazi su True.
2  request.form.get('ime') == 'Pero'
3  request.form['ime'] == 'Pero'
4
5  request.form.get('starost') == '20'          # Pazi, string!
6  request.form.get('starost', type=int) == 20 # Sada je int!
7
8  request.form.get('ovognema') == None
9  request.form.get('ovognema', type=int, default=0) == 0
10 # Pokušaj pristupa request.form['ovognema'] sruši aplikaciju.
```

HTML forme: metode GET i POST

Pojedine rute moguće je omogućiti za točno određene tipove HTTP zahtjeva (GET, POST, PUT, DELETE, PATCH, itd.).

Po defaultu, rute dozvoljavaju samo GET.

```
1  from flask import Flask, request;
2
3  app = Flask( __name__ );
4
5  @app.route('/handle_post', methods=['POST'])
6  def handle_post():
7      ... # Izvrši se samo ako pristupamo pomoću POST
8
9  @app.route('/handle_both', methods=['GET', 'POST'])
10 def handle_both():
11     if( request.method == 'POST' ):
12         ime = request.form.get('ime');
13         starost = request.form.get('starost');
14         return render_template( 'ispis.html', ime=ime, starost=starost );
15     else:
16         return render_template( 'forma.html' );
```

Zadatak 2

Napišite Flask aplikaciju koja:

- Prikazuje formu za unos 2 broja pomoću textbox-eva.
- Klikom na gumb "Zbroji" ispisuje zbroj ta dva broja.

Napravite varijante koje podatke iz forme šalju pomoću GET-a i pomoću POST-a.

HTML forme: metode GET i POST

Moguće je lako sagraditi *query-string* iz Pythona:

```
from urllib.parse import urlencode;
params = {'q': 'flask', 'page': 2};
query_string = urlencode(params);
url = f'/handle_get?{query_string}';
```

Sigurnosni aspekti:

- Uvijek provjeriti je li varijabla definirana:

```
name = request.args.get('name');
if( name ):
    return f'Hello, {name}!'; # name nije None i nije ''
else:
    return 'Please enter your name.'; # name je None ili ''
```

- Za podatke dobivene od korisnika uvijek treba provesti:
  - sanitizaciju - ukloniti "zabranjene" znakove iz podataka.
  - validaciju - provjeriti jesu li podaci u ispravnom obliku.

HTML forme: metode GET i POST

Pristup podacima iz forme:

- Ako formi imamo: `<input name="nesto">`, onda u Flasku postoji `request.args.get('nesto')` ili `request.form.get('nesto')`.
- Specijalno za `checkbox`, `name` treba završavati sa `[]`.

```
<form method="post" action="/handle_post">
    Meals:<br>
    <input type="checkbox" name="meals[]" value="breakfast">Breakfast<br>
    <input type="checkbox" name="meals[]" value="lunch">Lunch<br>
    <input type="checkbox" name="meals[]" value="dinner">Dinner<br>
    <input type="submit" name="btn1" value="Submit Button 1">
    <input type="submit" name="btn2" value="Submit Button 2">
</form>
```

- Tada je `request.form.getlist('meals[]')` lista koje sadrži odabrane vrijednosti.
- Ovisno o tome na koji je gumb korisnik kliknuo, postojat će `request.form.get('btn1')` ili `request.form.get('btn2')`.

HTML forme: Upload datoteka (Primjer 1)

Upload datoteka pomoću HTML formi:

- HTML-forma mora imati atribut `enctype="multipart/form-data"`. Defaultni enctype je `application/x-www-form-urlencoded`.
- Metoda mora biti POST.

```
<form method="post" action="/handle_upload" enctype="multipart/form-data">
    <input type="file" name="document"><br>
    <input type="submit" value="Pošalji datoteku">
</form>
```

Flask:

- Ako je forma sadržavala input tipa `file` imena `filename`, onda će postojati `request.files.get('filename')`.
- Treba osigurati da Flask ima pravo pisanja u direktorij u kojeg ćemo spremiti file. Po defaultu Apache/Nginx imaju username i grupu `www-data`.
- Nikada ne spremati datoteku u direktorij dostupan preko URL-a!

HTML forme: Upload datoteka (Primjer 1)

```
import os;
from flask import Flask, request, render_template;
from werkzeug.utils import secure_filename;

UPLOAD_FOLDER = 'uploads/';

@app.route( '/handle_upload', methods=['POST'] )
def handle_upload():
    if( 'document' not in request.files ):
        return 'U zahtjevu uopće ne postoji file s name-om document.';

    file = request.files['document'];
    if( file.filename == '' ):
        return 'Korisnik nije odabrao datoteku za upload.';

    # Ukloni nedozvoljene znakove iz imena/putanje datoteke.
    filename = secure_filename( file.filename );

    # Spremi datoteku u UPLOAD_FOLDER.
    file.save( os.path.join( UPLOAD_FOLDER, filename ) );

    return f'Datoteka {filename} je uspješno uploadana!';
```