PROGRAMIRANJE U INTEGRISANIM TEHNOLOGIJAMA

Oznaka predmeta: PIT
Predavanje broj: 10

Nastavna jedinica: Flask,
Nastavne teme:

Flask: mail, WTForms, SQLite3, SQLAlchemy.

Predavač: prof. dr Perica S. Štrbac, dipl. ing.

Literatura:

Miguel Grinberg. "Flask Web Development", O'REILLY, 2014.

Flask-mail

- Za instalaciju Flask-mail-a uraditi:

```
pip install Flask-mail
```

instaliraće se sve potrebne komponente

Parametri Flask-mail-a:

- MAIL_SERVER, naziv/IP adresa email servera.
- MAIL_PORT, port servera.
- MAIL_USE_TLS, o(ne)mogućavanje Transport Security Layer enkripcije.
- MAIL_USE_SSL, o(ne)mogućavanje Secure Sockets Layer enkripcije.
- MAIL_DEBUG, debug podrška (podrazumevano kao i flask aplikacije).
- MAIL_USERNAME, ime pošiljaoca.
- MAIL_PASSWORD, password pošiljaoca.
- MAIL_DEFAULT_SENDER, podrazumevani pošiljalac.
- MAIL_MAX_EMAILS, maksimalni broj mejlova za slanje.
- MAIL_SUPPRESS_SEND, Zabrana slanja emaila ako je app.testing ==True.
- MAIL_ASCII_ATTACHMENTS, da li se prikačeni fajlovi konvertuju u ASCII.

flask-mail.Mail flask-mail.Message

- Klasa flask-mail.Mail:

```
flask-mail.Mail(app = None)
```

u konstruktoru prihvata Flask objekat.

Metode klase flask-mail.Mail:

  - `connect()` otvaranje konekcije.
  - `send_message()` slanje **objekta** klase `flask-mail.Message`
  - `send()` slanje **sadržaja objekta** klase `flask-mail.Message`

- Klasa `flask-mail.Message`:

```
flask-mail.Message(subject, recipients, body, html, sender, cc,
                   bcc, reply-to, date, charset, extra_headers,
                   mail_options, rcpt_options)
```

respektivno:

naslov mail-a, primaoci, telo, html-poruka, pošiljalac, carbon-copy, blind-carbon-copy, povratna adresa, datum slanja, karakter set poruke, recnik dodatnih zaglavlja poruke,
MAIL FROM komanda ESMTPa (SMTP serveru znači da počinje nova email transakcija tako isti resetuje sve tabele stanja i bafere),
RCPT TO komanda ESMTPa (isti MAIL FROM na nekoliko RCPT TO adresa).

flask-mail.Message

- Metode klase flask-mail.Mesage:
  - `attach()` dodaje attachment mail-u
    - parametri su:
      - `filename` naziv fajla
      - `content_type` MIME tip fajla
      - `data` podaci fajla
      - `disposition` header multipart body-a.
  - `add_recipient()` dodaje primaoca poruke
- Primer korišćenja SMTP servera Google-ovog gmail servisa:

```
from flask import Flask
from flask_mail import Mail, Message

app =Flask(__name__)

app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'perica@gmail.com'
app.config['MAIL_PASSWORD'] = '*****'
```

Flask-mail

```
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True
mail = Mail(app)

@app.route("/")
def index():
   msg = Message('Hello', sender = 'perica@gmail.com', recipients =
['joe@gmail.com'])
   msg.body = "Hello Flask message sent from Flask-Mail"
   mail.send(msg)
   return "Sent"

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

- Sada na stranici `localhost:5000` sledi slanje navedenog email-a.
  - Ako Gmail blokira login, ulogujte se iz browser-a na svoj gmail nalog i posetite stranicu:

    `https://www.google.com/settings/security/lesssecureapps`

    na kojoj umanjite security (Allow less secure apps: ON)

    a onda probajte kod.

Flask-mail

- Slanje više mail-ova sa jedne konekcije (paziti na MAIL_MAX_EMAILS):

```
with mail.connect() as conn:
    for user in users:
        message = '...'
        subject = "hello, %s" % user.name
        msg = Message(recipients=[user.email],
                      body=message,
                      subject=subject)
        conn.send(msg)
```

- Dodavanje attachment-a (filename, content_type, data):

```
with app.open_resource("image.png") as fp:
    msg.attach("image.png", "image/png", fp.read())
```

flask-WTF

- Za instalaciju flask-WTF-a uraditi:

  ```
  pip install flask-WTF
  ```

  instaliraće se sve potrebne komponente
- WTF forme se koriste za renderovanje i validaciju.
- WTForms se definišu u python skripti, a renderišu u template-u.
- Validacija se primenjuje na nivou WTForms polja (wtforms).
- Slede neka standardna polja:
  - `StringField` → <input type = 'text'> HTML element
  - `TextAreaField` → <textarea> HTML element
  - `RadioField` → <input type = 'radio'> HTML element
  - `BooleanField` → <input type = 'checkbox'> HTML element
  - `IntegerField` → TextField za prikaz celog broja
  - `DecimalField` → TextField za prikaz decimalnog broja
  - `SelectField` → Prikazuje select form element
  - `PasswordField` → <input type = 'password'> HTML element
  - `SubmitField` → <input type = 'submit'> HTML element

flask-WTF

- Primer WTF sa StringField-om (fajl `ContactForm.py`):

```
from flask_wtf import FlaskForm
from wtforms import StringField
from wtforms.validators import DataRequired

class ContactForm(FlaskForm):
   name = StringField("Ime studenta", validators=[DataRequired()])
```

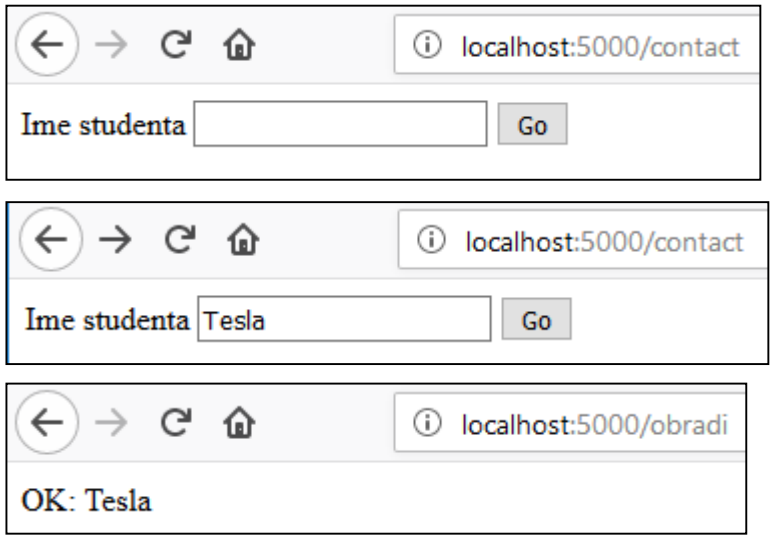
- Template koji će biti renderovan sa ulaznim parametrom tipa `ContactForm` je (`contact.html`):

```
<!DOCTYPE html> <html lang="en">
<head> <meta charset="UTF-8"> <title>Title</title> </head> <body>
    <form method="POST" action="/obradi">
        {{ wtforma.csrf_token }}
        {{ wtforma.name.label }} {{ wtforma.name(size=20) }}
    <input type="submit" value="Go">
    <!- moze da se koristiti i hidden_tag() za sva hidden polja
    <form method="POST" action="/obradi">
        {{ form.hidden_tag() }}
        {{ form.name.label }} {{ form.name(size=20) }}
    <input type="submit" value="Go">
    </form>    -->
</form> </body> </html>
```

flask-WTF

- Kod koji koristi prethodno:

```
from flask import Flask, render_template, request
from ContactForm import ContactForm
app = Flask(__name__)
app.secret_key = 'development key'

@app.route("/obradi", methods=['POST'])
def obradi():
    if request.method=='POST':
        return "OK: "+request.form['name']

@app.route('/contact')
def contact():
   form = ContactForm()
   return render_template('contact.html', wtforma = form)

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



- Date su slike unosa imena studenta (renderovani template `contact.html` sa ulaznim parametrom WTF `ContactForm`) na stranici `/contact` i obrada unosenog imena na stranici `/obrada`

flask-WTF

- Slede neke klase WTF validatora:
  - `DataRequired(message)` → provera da li je polje prazno
  - `Email(message)` → provera upisanog emaila prema email ID konvenciji
  - `IPAddress(message)` → provera polja IP adrese
  - `Length(min,max,message)` → provera dužine stringa prema datom opsegu
  - `NumberRange(min,max,message)` → provera polja broja prema datom opsegu
  - `URL(require_tld,message)` → provera unesenog URL-a
- Sledeći kod omogućuje unos i validaciju podataka o studentu.
  - Na stranici `/contact` poziva se rendering template-a `contact2.html` za unos podataka o studentu.
  - Pri ovom `GET` pristupu prvi put nema flash-ovanih poruka tako da će se prikazati samo forma unosa.
  - Pri pozivu stranice `/contact` iz template-a `contact2.html` prenos podataka je metodom `POST` tako da se sada proverava i validacija unesenih podataka.
  - Ako je unos validan renderuje se template `success2.html`, a u suprotnom se ponovo ide na obradu template-a `contact2.html`.

flask-WTF

- Kod template-a `contact2.html`:

```
<!DOCTYPE html>
<html lang="en"><head><meta charset="UTF-8"><title>Title</title></head>
<body>
      <h2 style = "text-align: center;">Contact Form</h2>

      {% for message in form.name.errors %}
         <div>{{ message }}</div>
      {% endfor %}

      {% for message in form.email.errors %}
         <div>{{ message }}</div>
      {% endfor %}

      <form action = "http://localhost:5000/contact" method = post>
         <fieldset>
            <legend>CF</legend>
            {{ form.hidden_tag() }}

            <div style = "font-size:20px; font-weight:bold; margin-
left:150px;">
               {{ form.name.label }}<br>
               {{ form.name }}
               <br>
```

flask-WTF

```
                {{ form.Gender.label }} {{ form.Gender }}
                {{ form.Address.label }}<br>
                {{ form.Address }}
                <br>

                {{ form.email.label }}<br>
                {{ form.email }}
                <br>

                {{ form.Age.label }}<br>
                {{ form.Age }}
                <br>

                {{ form.language.label }}<br>
                {{ form.language }}
                <br>
                {{ form.submit }}
            </div>

        </fieldset>
    </form>
</body>
</html>
```

flask-WTF

- Kod WTF `ContactForm2.py` omogućuje kreiranje objekta `ContactForm2()` koji se prosleđuje kao parametar u template `contact2.html`

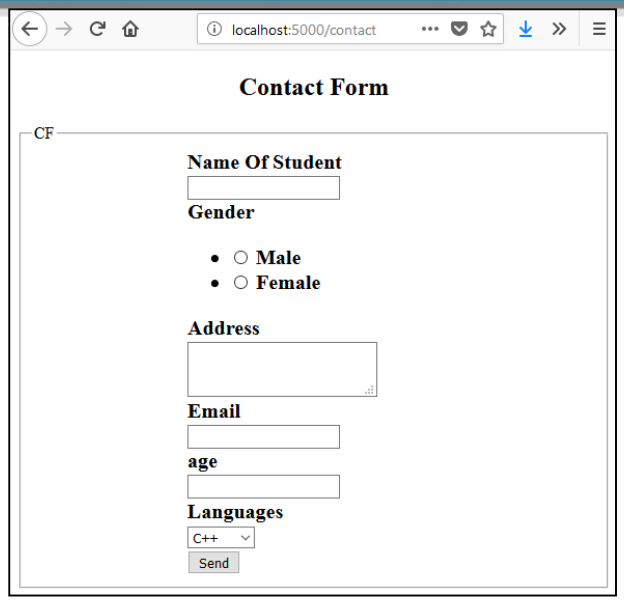
```
from flask_wtf import FlaskForm
from wtforms import StringField, IntegerField, TextAreaField, \
                    SubmitField, RadioField, SelectField
from wtforms.validators import DataRequired, Email

class ContactForm2(FlaskForm):
    name = StringField("Name Of Student",
                       validators=[DataRequired("Enter your name.")])
    Gender = RadioField('Gender',
                        choices=[('M', 'Male'), ('F', 'Female')])
    Address = TextAreaField("Address")
    email = StringField("Email",
                        validators=[DataRequired("Enter your email
address."),
                                    Email("Enter your email address.")])
    Age = IntegerField("age")
    language = SelectField('Languages',
                           choices=[('cpp', 'C++'), ('py', 'Python')])
    submit = SubmitField("Send")
```

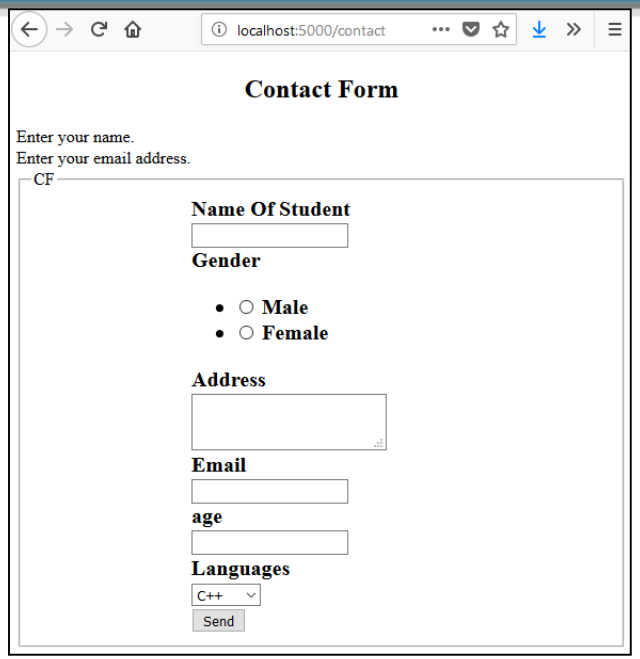
flask-WTF

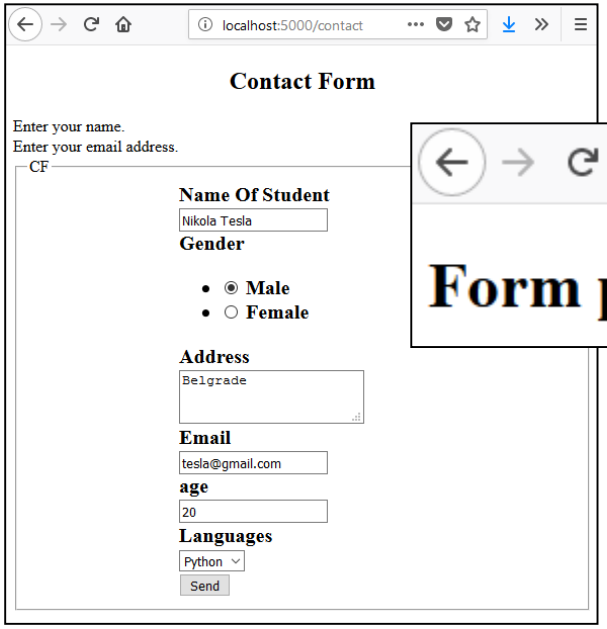
- Glavni kod `WTFExample.py` omogućuje pristup stranici `/contact` gde se kreira objekat tipa `ContactForm2`, te renderuje templat-e `contact2.html`, a pri pozivu stranice `/contact` iz ovog template-a proverava validaciju podataka te flash-uje poruku o nevalidnom unosu, a onda renderuje templat-e `contact2.html` gde se prikazuje i flashovana poruka. Ako je unos bio validan renderuje se template `success2.html`

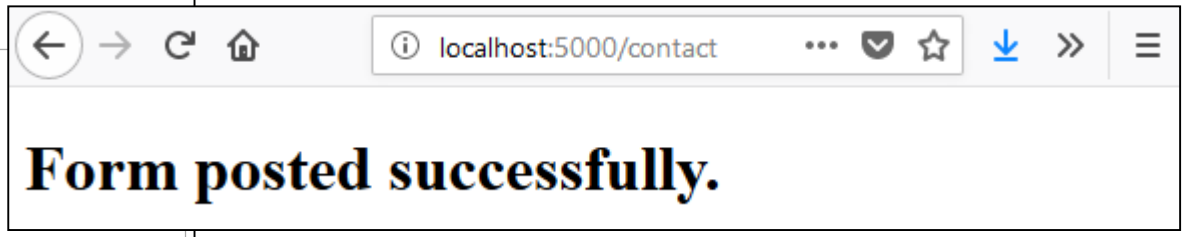
```
from flask import Flask, render_template, request, flash
from ContactForm2 import ContactForm2
app = Flask(__name__)
app.secret_key = 'development key'

@app.route('/contact', methods=['GET', 'POST'])
def contact():
    form = ContactForm2()
    if request.method == 'POST':
        if form.validate() == False:
            flash('All fields are required.')
            return render_template('contact2.html', form=form)
        else:
            return render_template('success2.html')
    return render_template('contact2.html', form=form)
if __name__ == '__main__':    app.run(debug=True)
```

flask-WTF


localhost:5000/contact
Contact Form
CF
Name Of Student
Gender
Male
Female
Address
Email
age
Languages
C++
Send


localhost:5000/contact
Contact Form
Enter your name.
Enter your email address.
CF
Name Of Student
Gender
Male
Female
Address
Email
age
Languages
C++
Send


localhost:5000/contact
Contact Form
Enter your name.
Enter your email address.
CF
Name Of Student
Nikola Tesla
Gender
Male
Female
Address
Belgrade
Email
tesla@gmail.com
age
20
Languages
Python
Send


localhost:5000/contact
Form posted successfully.

flask-WTF, flask-sqlite3

- Trivijalni kod template-a `success2.html`

```
<!DOCTYPE html>
<html lang="en"><head><meta charset="UTF-8"><title>Title</title></head>
<body>
    <h1> Form posted successfully. </h1>
</body>
</html>
```

- Ugrađena podrška python-a za sql (sqlite3) može se koristiti i u flask aplikacijama.
- Ideja je da se na stranici / omogući dodavanje podataka o novom studentu u sqlite3 bazu (`db.db`) ili da se prikažu tabelarni podaci o studentima (renderuje se template `home.html`) koji su u bazi.
- Kod `home.html`

```
<!DOCTYPE html>
<html lang="en"><head><meta charset="UTF-8"><title>Title</title></head>
<body>
    <a href="{{ url_for('new_student') }}" style="font-size: 30px;">New
student</a>
    <br>
    <a href="{{ url_for('list') }}" style="font-size: 30px;">Show list</a>
    <br></body></html>
```

flask-sqlite3

- View funkcija `new_student` renderuje template `student.html` (forma za unos podataka o novom studentu) i forma se šalje view funkciji `addrec()`:

```
<!DOCTYPE html>
<html lang="en"><head><meta charset="UTF-8"><title>Title</title>
</head>
<body>
      <form action = "{{ url_for('addrec') }}" method = "POST">
         <h3>Student Information</h3>
         Name<br>
         <input type = "text" name = "nm" /></br>

         Address<br>
         <textarea name = "add" ></textarea><br>

         City<br>
         <input type = "text" name = "city" /><br>

         PINCODE<br>
         <input type = "text" name = "pin" /><br>
         <input type = "submit" value = "submit" /><br>
      </form>
</body>
</html>
```

flask-sqlite3

- View funkcija `addrec` pokušava da upiše podatke o studentu u sqlite3 bazu.
  - Ako uspe kreira poruku uspeha, u suprotnom poruku neuspeha.
  - Na kraju renderuje template `result.html` koji prikazuje poruku o uspešnosti upisa zapisa o novom studentu u sqlite3 bazu i prikazuje link ka stranici / (view funkcija `home()`).
- Kod template-a `addstudentresult.html`

```
<!doctype html>
<html>
   <body>
      result of addition : {{ msg }}
      <h2><a href = "{{ url_for('home') }}">go back to home page</a></h2>
   </body>
</html>
```

- Template `list.html` renderuje view funkcija `list()` koja selektuje sve zapise o studentima u sqlite3 bazi. Sledi kod tremplate-a `list.html`

```
<!DOCTYPE html>
<html lang="en"><head><meta charset="UTF-8"><title>Title</title></head>
<body>
      <table border = 1>
         <thead>
```

flask-sqlite3

```
            <td>Name</td>
            <td>Address></td>
            <td>city</td>
            <td>Pincode</td>
        </thead>

        {% for row in rows %}
            <tr>
                <td>{{row["name"]}}</td>
                <td>{{row["addr"]}}</td>
                <td>{{row["city"]}}</td>
                <td>{{row["pin"]}}</td>
            </tr>
        {% endfor %}
    </table>
    <a href = "/">Go back to home page</a>
</body>
</html>
```

- Trivijalni kod koji kreira bazu db.db i u njoj tabelu students.

```
import sqlite3
conn = sqlite3.connect('db.db')
print("Opened database successfully")
```

flask-sqlite3

```
conn.execute('CREATE TABLE students (name TEXT, addr TEXT, city TEXT,
pin TEXT)')
print("Table created successfully")
conn.close()
```

- Sledi kod flask aplikacije `StudentiSQLite3.py`

```
from flask import Flask, render_template, request
import sqlite3 as sql

app = Flask(__name__)

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

@app.route('/enternew')
def new_student():
    return render_template('student.html')

@app.route('/addrec', methods=['POST', 'GET'])
def addrec():
    if request.method == 'POST':
        try:
```

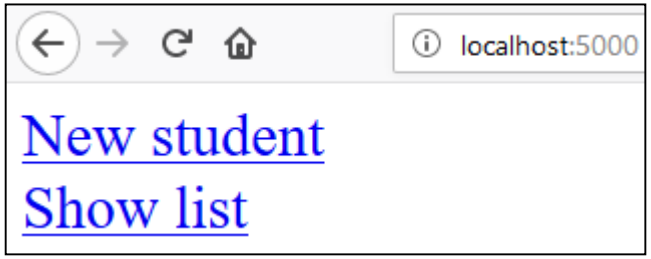
flask-sqlite3

```
            nm   = request.form['nm']
            addr = request.form['add']
            city = request.form['city']
            pin  = request.form['pin']

            with sql.connect("db.db") as con:
                cur = con.cursor()
                cur.execute("INSERT INTO students (name,addr,city,pin)\
                VALUES(?, ?, ?, ?)",(nm,addr,city,pin) )
                con.commit()
                msg = "Record successfully added"
        except:
            con.rollback()
            msg = "error in insert operation"
        finally:
            return render_template("addstudentresult.html", msg=msg)
            con.close()

@app.route('/list')
def list():
    con = sql.connect("db.db")
    con.row_factory = sql.Row
    cur = con.cursor()
```

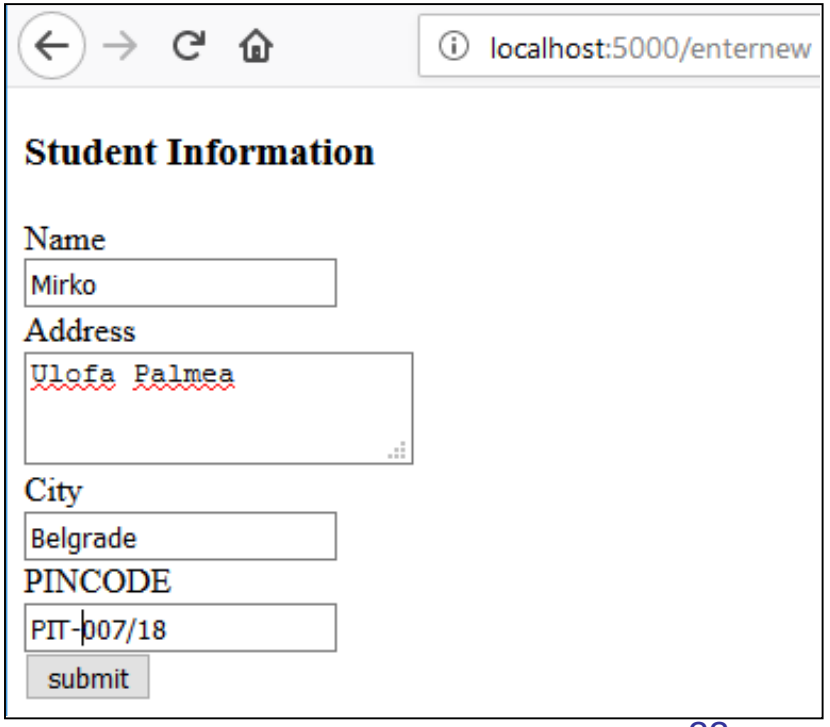
flask-sqlite3

```
    cur.execute("select * from students")
    rows = cur.fetchall();
    return render_template("list.html", rows=rows)

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

- Prethodni kod koristi kreiranu bazu db.db tako da na stranici / nudi linkove ka unosu novog studenta ili ka pregledu svih unesenih studenata.

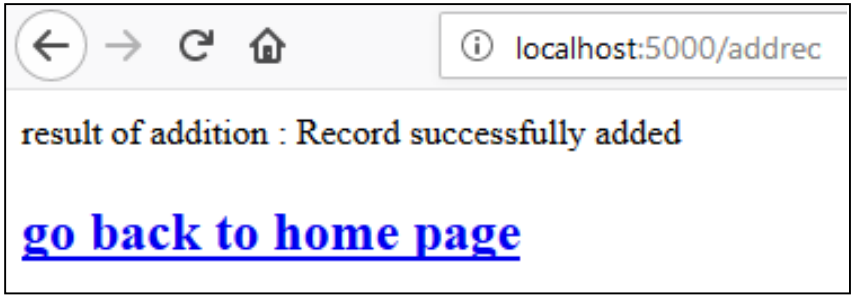


- Link New student vodi ka stranici `/enternew` gde se popunjava forma koju čine polja podataka o studentu.
  - Popunjena forma se šalje na stranicu `/addrec` gde se:
    - uneseni podaci ekstrahuju iz forme

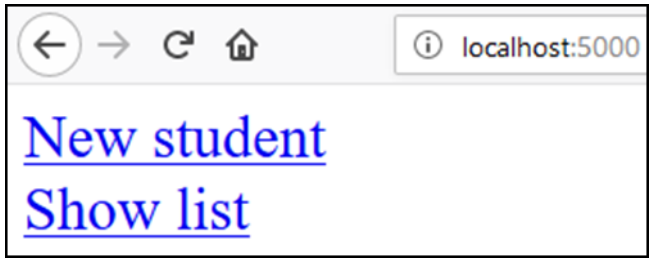


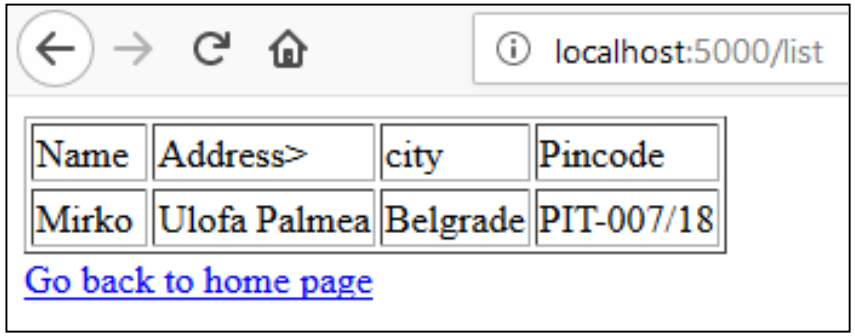
flask-sqlite3

- pokušava upis u tabelu studenti (baza db.db)
- priprema string msg koji opisuje uspešnost operacije upisa podataka
- na kraju renderuje template `addstudentresult.html` koji prikazuje poruku uspešnosti operacije upisa i link ka stranici /



- Odlaskom na stranicu `/` ,te klikom na link `Show list` odlazi se na stranicu `/list` gde se prikazuje tabela unesenih studenta i link za povratak na osnovnu stranicu.



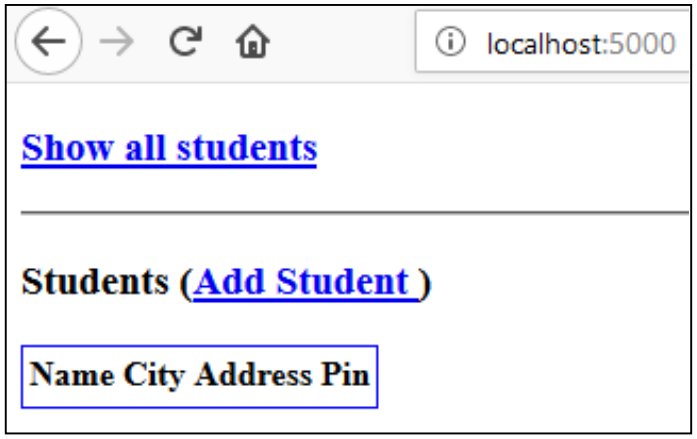


Flask – SQLAlchemy

- SQLAlchemy predstavlja ORM (*Object Relation Mapping*) maper.
- ORM API mapira objekte u RDBMS tabelarne strukture obezbeđujući CRUD operacije.
- Instalacija SQLAlchemy:

```
pip install flask-sqlalchemy
```

- Template `show_all_students.html` prikazuje:
  - link za osvežavanje prikaza svih studenata (npr. više korsnika unose studente),
  - flash-ovane poruke (uspešnosti unosa novog studenta u bazu, ako je bio upis od strane ovog korisnika),
  - link ka stranici za dodavanje novog studenta,
  - tabelarne podatke studenata.



Flask – SQLAlchemy

- Kod template-a `show_all_students.html`

```
<!DOCTYPE html>
<html lang = "en">
   <head></head>
   <body>
      <h3>
         <a href = "{{ url_for('show_all') }}">Show all students</a>
      </h3>
      <hr/>
      {%- for message in get_flashed_messages() %}
         {{ message }}
      {%- endfor %}
      <h3>Students (<a href = "{{ url_for('new') }}">Add Student
         </a>)</h3>
      <table style="border:1px solid blue;">
         <thead>
            <tr>
               <th>Name</th>
               <th>City</th>
               <th>Address</th>
               <th>Pin</th>
            </tr>
         </thead>
```

Flask – SQLAlchemy

```
            <tbody>
               {% for student in students %}
                  <tr>
                     <td>{{ student.name }}</td>
                     <td>{{ student.city }}</td>
                     <td>{{ student.addr }}</td>
                     <td>{{ student.pin }}</td>
                  </tr>
               {% endfor %}
            </tbody>
         </table>
      </body>
</html>
```

- Template `new_student.html` prikazuje:
  - flash-ovane poruke sa pripadnom kategorijom (ako ih ima):
  - formu za unos podataka o novom studentu koju prosleđuje stranici /new metodom POST
  - ovaj template se prvi put renderuje nakon poziva view metode new() metodom GET, a onda se prikazana forma šalje istoj stranici (view metodi new()) ali metodom POST.

Flask – SQLAlchemy

- Pripadni kod template-a `new_student.html`

```
<!DOCTYPE html>
<html>
   <body>
      <h3>New student</h3>
      <hr/>
      {%- for category, message in get_flashed_messages(with_categories =
true) %}
          <br> {{ message }} <br>
      {%- endfor %}
      <form action = "{{ request.path }}" method = "post">
         <label for = "name">Name</label><br>
         <input type = "text" name = "name" placeholder = "Name" /><br>
         <label for = "email">City</label><br>
         <input type = "text" name = "city" placeholder = "city" /><br>
         <label for = "addr">addr</label><br>
         <textarea name = "addr" placeholder = "addr"></textarea><br>
         <label for = "PIN">City</label><br>
         <input type = "text" name = "pin" placeholder = "pin" /><br>
         <input type = "submit" value = "Submit" />
      </form>
   </body>
</html>
```

Flask – SQLAlchemy

- Program `SQLAstudents.py`
  - flask objektu konfiguriše `SQLALCHEMY_DATABASE_URI` stazu do baze podataka koja se koristi za studente
  - flask objektu konfiguriše `SECRET_KEY`
  - kreira objekat `SQLAlchemy` za `ORM` mapiranje
  - kreira klasu za mapiranje objekta tipa `students` u pripadnu tabelarnu strukturu u bazi podataka
  - na stranici / renderuje template `show_all_students.html` kome prosleđuje sve studente
  - na stranici /new obrađuje:
    - `GET` poziv: renderuje se template-a `new_student.html`
    - `POST` poziv: proveravaju se polja pristigle forme
      - ako je neko polje nepopunjeno onda se postavlja flash poruka o nekorektnosti te renderuje template `new_student.html`
      - ako su polja popunjena, onda se kreira objekat tipa `students` sa vrednostima parametara konstruktora koji odgovaraju vrednostima polja forme, snimi se objekat tabelarno u bazu podataka, flash-uje poruka o uspešnosti i vrši redirekcija na stranicu `/ (show_all())`.

Flask – SQLAlchemy

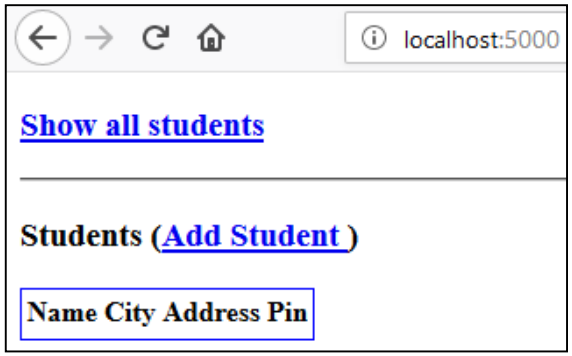
- Kod programa SQLAstudents.py

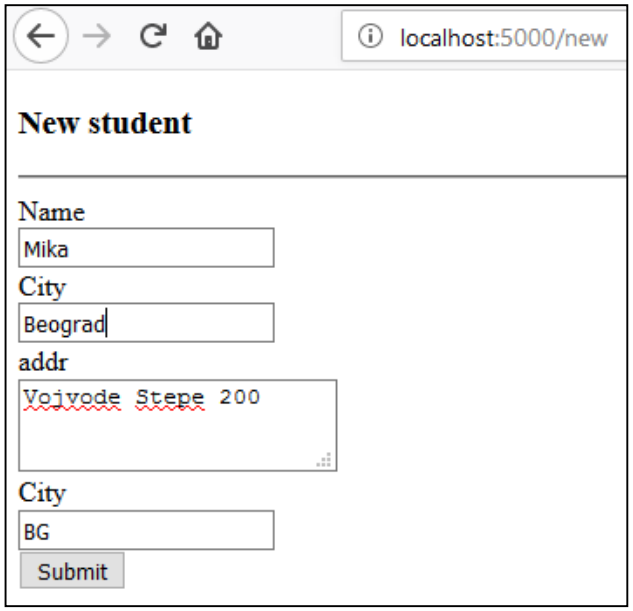
```
from flask import Flask, request, flash, url_for, redirect,
render_template
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///students.sqlite3'
app.config['SECRET_KEY'] = "random string"

db = SQLAlchemy(app)

class students(db.Model):
    id   = db.Column('student_id', db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    city = db.Column(db.String(50))
    addr = db.Column(db.String(200))
    pin  = db.Column(db.String(10))

    def __init__(self, name, city, addr, pin):
        self.name = name
        self.city = city
        self.addr = addr
        self.pin = pin
```

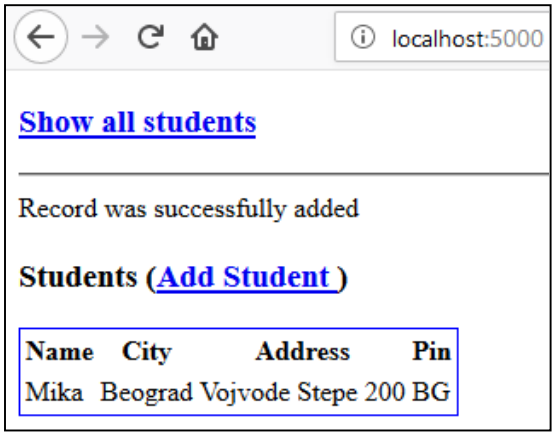
Flask – SQLAlchemy

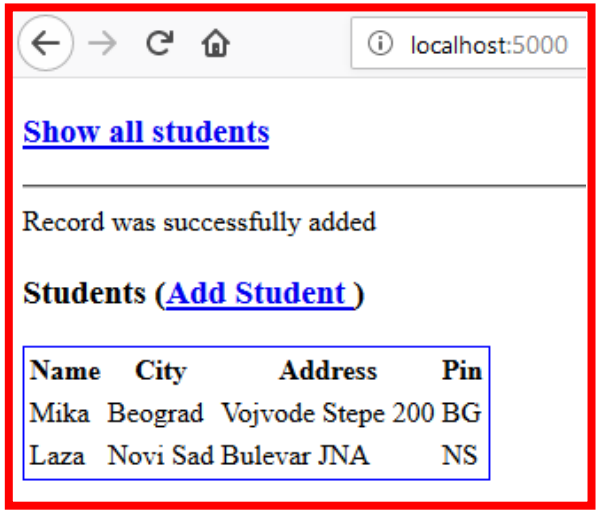
```
@app.route('/')
def show_all():
    return render_template('show_all_students.html',
                           students=students.query.all())

@app.route('/new', methods=['GET', 'POST'])
def new():
    if request.method == 'POST':
        if    not request.form['name'] or not request.form['city'] \
           or not request.form['addr'] or not request.form['pin']:
            flash('Please enter all the fields', 'error')
            return render_template('new_student.html')
        else:
            student = students(request.form['name'], request.form['city'],
                               request.form['addr'], request.form['pin'])
            db.session.add(student)
            db.session.commit()
            flash('Record was successfully added')
            return redirect(url_for('show_all'))
    return render_template('new_student.html')
if __name__ == '__main__':
    db.create_all(); app.run(debug=True)
```

Flask – SQLAlchemy

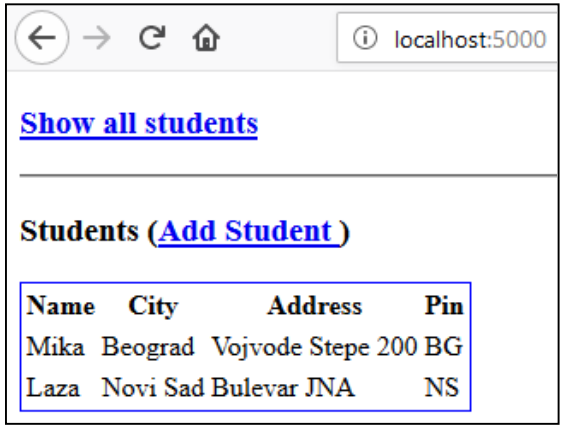








Lazu je uneo drugi korisnik



Prvi korisnik - Show all students