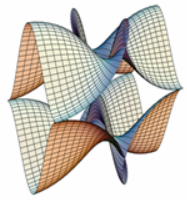


Prirodoslovno-matematički fakultet
Matematički odsjek
Sveučilište u Zagrebu

RAZVOJ WEB-APLIKACIJA

Predavanje 06 - Datoteke. Baze podataka.

31. listopada 2025.

Sastavio: Zvonimir Bujanović

Pohrana podataka: Datoteke i baza podataka

Podatke web-aplikacije obično pohranjujemo u datotekama ili u bazama podataka.

- Datoteke tipično koristimo za *binarne* podatke: slike, PDF-ove, zvučne zapise i slično.
- Ostale podatke, npr. korisnička imena, dinamički sadržaj (članci, komentari, podaci o kupovini i slično) čuvamo u bazi podataka.
- Često se koristi i kombinacija, npr. sliku čuvamo kao datoteku, a podatke o slici (putanja, veličina, vlasnik) u bazi podataka.

Ranije smo vidjeli kako napraviti upload datoteke preko forme.

- Sigurnosni aspekti: uvijek koristiti `secure_filename()` umjesto imena datoteke koje predloži korisnik (ne samo kod uploada).

Rad s datotekama

- User i/ili grupa kojem pripada web-server (za Apache/Nginx na Linuxu obično `www-data`) mora imati pravo čitanja/pisanja/pristupanja direktoriju i datoteci s kojom radimo.

- Najjednostavnije: "ostalima" damo pravo `r` ili `w` ili `x`.
  Ispravnije: prenijeti vlasništvo nad datotekama aplikacije na `www-data`.

- Ako želimo čitati iz datoteke, onda:
  - Folder u kojem se datoteka nalazi mora imati pravo `x`.
  - Sama datoteka mora imati pravo `r`.
- Ako želimo pisati u već postojeću datoteku, onda:
  - Folder u kojem se datoteka nalazi mora imati pravo `x`.
  - Sama datoteka mora imati pravo `w`.
- Ako želimo stvoriti novu datoteku, onda:
  - Folder u kojem će se datoteka nalaziti mora imati pravo `w`.

Rad s datotekama

Flask aplikacije mogu koristiti standardne Pythonove funkcije za rad s datotekama.

- Obično ne želimo manipulirati liniju po liniju ili znak po znak u datoteci, nego raditi na nivou cijele datoteke.
- Binarne datoteke tipično samo premještamo kod uploada, nudimo za download ili brišemo.
- Primjer: u HTML-u imamo link na `/download` koji vodi na dowload PDF datoteke.

```
1  from flask import Flask, send_file;
2
3  @app.route('/download')
4  def download_file():
5      # Putanja do datoteke na serveru.
6      # Paziti da datoteka nije dostupna direktno preko URL-a!
7      path_to_file = 'files/report.pdf';
8
9      return send_file(
10         path_to_file,
11         as_attachment=True,
12         download_name='document.pdf' # Korisnik vidi ovo ime PDF datoteke.
13     );
```

Rad s datotekama

Flask aplikacije mogu koristiti standardne Pythonove funkcije za rad s datotekama.

- Obično ne želimo manipulirati liniju po liniju ili znak po znak u datoteci, nego raditi na nivou cijele datoteke.
- Ponekad je korisno čuvati podatke u maloj tekstualnoj datoteci umjesto u bazi podataka.

```
1 from pathlib import Path;
2
3 # Učitaj cijelu datoteku u string text.
4 text = Path('data.txt').read_text(encoding='utf-8');
5
6 # Spremi sadržaj stringa text u data.txt.
7 Path('data.txt').write_text(text, encoding='utf-8');
```

Zadatak 1

Napišite Flask aplikaciju koja:

- Pruža mogućnost korisniku da upiše svoje ime u formu.
- U datoteci `users.txt` drži popis od najviše 5 zadnjih imena korisnika koji su posjetili stranicu (starija imena se "zaboravljaju", tj. brišu). Imena su odvojena zarezima i sva se nalaze u jedinom retku datoteke.
- Iza forme za upis imena, ispisuje taj popis.

Uputa:

- Koristite `pathlib.Path.exists` da provjerite da li datoteka postoji.
- Koristite `split` i `join` za brzu konverziju pročitanog stringa u listu i obratno.

Rad s datotekama - Lokoti

Što ako više korisnika istovremeno treba čitati/pisati u istu datoteku?

- Rješenje su tzv. lokoti i poziv funkcije `flock`.
  - `LOCK_EX` – ekskluzivni lokot. Ograničava pristup datoteci na samo jedan proces. Koristi se prilikom pisanja u datoteku.
  - `LOCK_SH` – dijeljeni lokot. Više procesa ima pravo pristupa datoteci. Koristi se prilikom čitanja iz datoteke.
  - `LOCK_UN` – otpuštanje lokota.
- Po defaultu, pokušaj dobivanja lokota sa `flock()` je blokirajući, tj. izvršavanje stoji na toj naredbi dok ne dobije pristup datoteci.

- Tipično, umjesto datoteka ćemo koristiti baze podataka koje nemaju ovakve probleme.

Rad s datotekama - Lokoti

```
import fcntl;

with open('guestbook.txt', 'a+', encoding='utf-8') as f:
    # Pokušaj dobiti ekskluzivni lokot.
    fcntl.flock(f, fcntl.LOCK_EX);

    try:
        # Dobili smo lokot. Sada smijemo zapisati nešto u datoteku.
        f.write( request.form.get('poruka') );

        # Prije otključavanja, treba osigurati da OS stvarno zapiše file na disk.
        f.flush();
        os.fsync(f.fileno());
    finally:
        # Otključamo lokot.
        fcntl.flock(f, fcntl.LOCK_UN);
```

Baze podataka

- Ekstenzije u Pythonu daju podršku za praktički svaki tip baze podataka (SQL/NoSQL).
- Te ekstenzije možemo koristiti i u web-aplikacijama.
- Prvenstveno ćemo se orijentirati na MySQL (tj. MariaDB).
- Python nudi više sučelja za pristup MySQL bazama:
  - službeni MariaDB Connector/Python
  - službeni Oracle MySQL Connector/Python
  - PyMySQL $\rightsquigarrow$ koristit ćemo ga zbog jednostavnosti i podrške za SQLAlchemy.
- Python podržava i SQLite baze:
  - Baza podataka koja ne zahtijeva dedicirani server, nego koristi običnu datoteku.
  - `import sqlite3;`
  - Izvršavanje upita vrlo slično kao sa PyMySQL.

Zadatak 2

- Ulogirajte se u phpmyadmin na rwa-serveru.
- Stvorite novu bazu sa željenim imenom.
- Ime baze će dobiti prefiks po vašem username-u.
- U toj bazi stvorite tablicu `Studenti`:
  - Tablica neka ima stupce JMBAG, Ime, Prezime, Ocjena.
- Unesite nekoliko redaka u tu tablicu.

Baze podataka - Spajanje pomoću PyMySQL

- Konekcija na bazu: stvaranje objekta tipa `Connection`.

```
1 import pymysql.cursors;
2
3 db = pymysql.connect(
4     host='hostname',
5     user='username',
6     password='passwd',
7     database='db',
8     cursorclass=pymysql.cursors.DictCursor );
```

- Podatke za spajanje bi trebalo držati odvojeno od koda aplikacije (sigurnost, fleksibilnost) $\rightsquigarrow$ više o tome kasnije.

Baze podataka - Upiti

- Upiti se rade preko objekta tipa Cursor dobivenog iz konekcije na bazu.
- Upit izvršavamo pozivom članske funkcije `execute()` kojoj proslijedimo SQL naredbu.

```
cursor = db.cursor();
cursor.execute( 'SELECT JMBAG,Ime,Prezime FROM Studenti' );
```

- Po izvršenom upitu, možemo iterirati po "recima" koji odgovaraju rezultatu upita.
  - Možemo iterirati po kursoru for-petljom.

```
for row in cursor:
    print( f'JMBAG = {row.JMBAG}, Ime = {row.Ime}' );
```

  - Možemo dohvatiti odjednom sve retke pomoću `fetchall()`.

```
studenti = cursor.fetchall();
for row in studenti:
    print( f'JMBAG = {row.JMBAG}, Ime = {row.Ime} );
```

Baze podataka - Upiti

- `fetchall` sve rezultate odjednom dohvati i spremi u varijablu
  ⇝ nekad nepotrebno i memorijski rastrošno; bolje for-petlja.
- Dohvaćanje samo dijela podataka za tzv. paginaciju:
  `SELECT * FROM users ORDER BY id LIMIT 20 OFFSET 40;`

- Pomoću `cursor.rowcount` možemo doznati broj redaka na koje je djelovao upit.
  - Ako je upit tipa `SELECT`, prije toga treba pozvati `cursor.fetchall()`
  - Ako je upita tipa `INSERT`, `UPDATE`, `DELETE` onda ne treba.

- Upit tipa `INSERT`, `UPDATE`, `DELETE` funkcija `execute` samo *pripremi*. Treba ih nakon toga treba ga *izvršiti* pozivom `db.commit()`.

- Kada smo gotovi s upitima, treba zatvoriti i kursor i konekciju prema bazi.

```
1 cursor.close();
2 db.close();
```

Zadatak 3

- Spojite se na bazu koju ste napravili na rwa-serveru.
- Dohvatite sve podatke o studentima iz tablice `Studenti`.
- Prikažite te podatke u HTML tablici.

SQL Injection

- Napadač kreira ili mijenja postojeće SQL naredbe da bi otkrio skrivene podatke ili izvršio opasne naredbe na serveru gdje se nalazi baza.
- Ovo postiže tako da ubacuje neočekivane parametre u SQL upite.

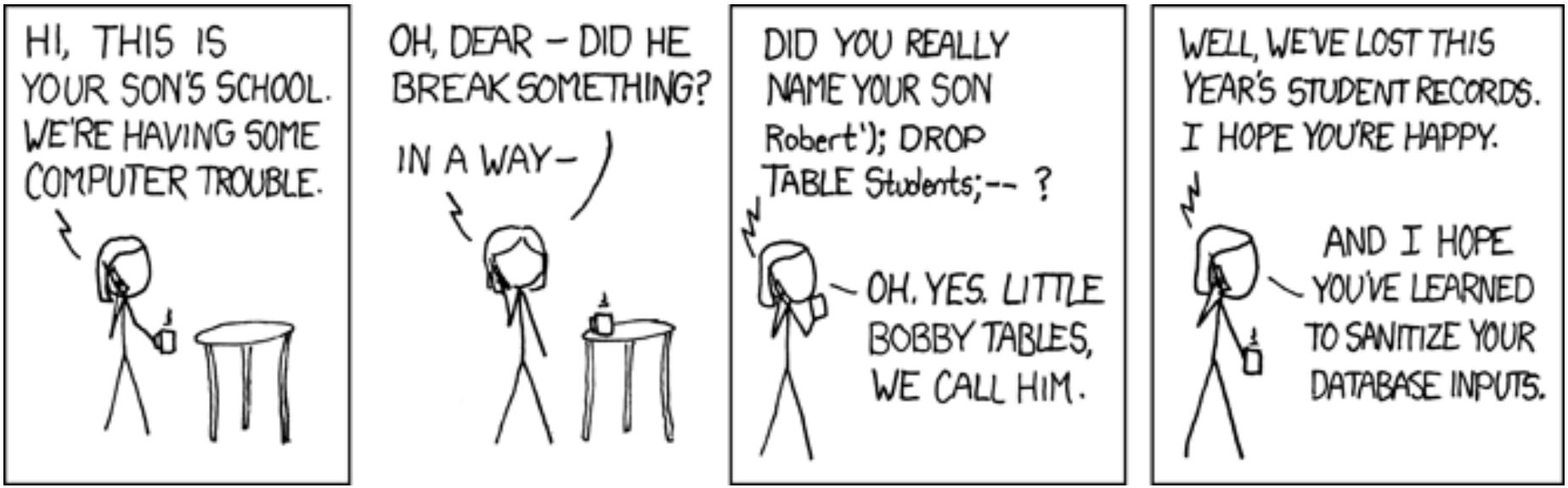
```
1 cursor.execute( f"SELECT * FROM users WHERE name='{username}'" );
```

- Ako je `username` jednak `' OR '1'='1`, onda:
  `SELECT * FROM users WHERE name='' OR '1'='1'`
  ⇝ dohvaćena je cijela tablica sa svim korisnicima!

- Ako je `username` jednak `a';DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't`, onda:
  ⇝ obrisana tablica `users`, dohvaćeni svi podaci iz druge tablice!

- Rješenje: parametrizirani upiti.

Baze podataka - Sigurnosni aspekti



https://xkcd.com/327/

Baze podataka - Parametrizirani upiti

- **Nikada** ne smijemo graditi SQL-upit pomoću f-stringa ili spajajući korisnikov unos manualno u string!
- Umjesto toga, koristimo parametrizirane upite.

```
1 cursor.execute(
2     'SELECT JMBAG FROM Studenti ' +
3     'WHERE Ime LIKE %(ime)s OR Prezime LIKE %(prezime)s',
4     {'ime': form.name, 'prezime': form.surname } );
```

- Parametrizirani upiti su sigurni od SQL Injection napada. SQL tada tretira parametre kao podatke, a ne kao dio SQL sintakse/naredbe.

Baze podataka - Detekcija grešaka kod spajanja i upita

- Greške je moguće detektirati okruživanjem naredbe u try-catch blok.

```
1 from pymysql.err import MySQLError;
2
3 try:
4     cursor = db.cursor();
5     cursor.execute( ... );
6 except MySQLError as err:
7     print( f'MySQL error: {err}' );
```

Zadatak 4

Modificirajte rješenje Zadatka 3 tako da:

- Dodate formu u kojoj pitate korisnika za ocjenu studenta.
- Nakon slanja forme, prikazuju se samo studenti koji imaju tu ocjenu.
- Koristite *parametrizirani upit*.

Problem: Ponavljanje uspostavljanja konekcije

- Problem: puno pomoćnih funkcija dohvaća podatke iz baze podataka.
  - ili bi se svaka funkcija trebala ponovno spajati na bazu,
  - ili bismo svakoj trebali prosljeđivati objekt `db` kao parametar.

- Rješenje (jedno moguće): Flaskov globalni objekt `g`.
  - Sve funkcije u Flasku dijele podatke spremljene u `g`.
  - Prvi poziv `get_db_connection` uspostavi konekciju i spremi ju u `g`.
  - Svaki idući poziv `get_db_connection` vraća tu istu konekciju.
  - Tako sada postoji samo jedan `db` objekt u cijelom programu.
  - Tu jednu, spremljenu konekciju onda koristimo svuda.

Rješenje: Spremi konekciju u globalni objekt

```
from flask import g; # Flaskov globalni objekt, dostupan iz svih funkcija!

def get_db_connection():
    if( 'db_connection' not in g ):
        # Konekciju uspostavljamo samo prilikom prvog poziva ove fje.
        g.db_connection = pymysql.connect(
            host=app.config['DATABASE_HOST'],
            user=app.config['DATABASE_USER'],
            password=app.config['DATABASE_PASS'],
            database=app.config['DATABASE_DB'],
            cursorclass=pymysql.cursors.DictCursor );

    return g.db_connection;

# Dekorator -> funkcija se automatski pozove jednom kada generiramo response.
@app.teardown_appcontext
def close_db( error ):
    # Dohvatimo i obrišemo db_connection iz g.
    db = g.pop( 'db_connection', None );
    if( db is not None ):
        # Zatvorimo konekciju na bazu - više to ne treba raditi manualno!
        db.close();
```

Problem: Podaci za spajanje na bazu su dio koda

Podaci za spajanje na bazu su trenutno spremljeni unutar `app.py`.

- Svaki developer ih treba prilagoditi za svoju okolinu.
- Ne želimo te podatke (password za bazu) staviti na git!
- Prilikom postavljanja aplikacije na server (*deploy*), treba mijenjati kod.

Rješenje (jedno moguće):

- Konfiguraciju spremiti u zasebnu datoteku.
  Na primjer: `app.config` ili `.config`
- Tu datoteku ne stavljamo na git.
- Umjesto toga, na git stavimo primjer konfiguracijske datoteke.
  Na primjer: `example.config`
- Svaki developer prilagodi `example.config` i spremi u svoj `app.config`. Slično prilikom deploy-a.

Rješenje: Konfiguracijska datoteka

Nakon svake promjene konfiguracijske datoteke treba ponovno pokrenuti server.

```
# Datoteka example.config
# Primjer konfiguracije, popis svih varijabli.
# Prava konfiguracija je u datoteci app.config koju NE STAVLJAMO na git.
DATABASE_HOST = 'db_host';
DATABASE_USER = 'user';
DATABASE_PASS = 'password';
DATABASE_DB   = 'db_name';

SESSION_TYPE = 'cachelib';
SESSION_PERMANENT = False;
```

```
# ------------------------------------ Konfiguracija
app = Flask( __name__ );
app.config.from_pyfile( 'app.config' );
```

Baze podataka - Sigurnosni aspekti

Kako čuvati korisničke lozinke u bazi podataka?

- **Nikad** ih ne čuvati doslovno onako kako ih je unio korisnik!
- Lozinke je potrebno hashirati i spremiti samo hash u tablicu.
- Ako netko i dobije neovlašteni pristup bazi, neće vidjeti lozinku nego samo hash.
- U donjem primjeru, u bazi za hash treba polje od 255 znakova.

```
1 from werkzeug.security import generate_password_hash;
2
3 # U bazu spremamo hash, a ne request.form.password!
4 hash = generate_password_hash( request.form.password );
```

```
1 from werkzeug.security import check_password_hash;
2
3 # Kasnije, da provjerimo je li uneseni request.form.password ispravan,
4 # dohvatimo hash iz baze i onda:
5 if( check_password_hash(hash, request.form.password) ):
6     ...
```

Zadatak 5

Napravite Flask aplikaciju koja prikazuje formu za unos korisničkog imena i lozinke.

- Forma ima 2 gumba: "Ulogiraj se", "Stvori novog korisnika".
- Klikom na prvi gumb, provjerava se u bazi postoji li taj korisnik i je li to njegova lozinka. Ako da, ispisuje se "Dobro došli, uspješno ste se ulogirali". Ako ne, ponovno se prikazuje forma.
- Klikom na drugi gumb, provjerava se u bazi postoji li taj korisnik, i ako ne postoji, dodaje se u bazu zajedno s pripadnom lozinkom. Treba ispisati odgovarajuću poruku ("Dodani ste u bazu" ili "Taj korisnik već postoji").

Funkcije za spajanje/odspajanje na bazu stavite u modul `db.py`.

- U modulu treba pristup objektu `app`:

```
1 from flask import g, current_app;
2
3 def get_db_connection():
4     if( 'db_connection' not in g ):
5         g.db_connection = pymysql.connect(
6             host=current_app.config['DATABASE_HOST'], # nije app.config[]!
7             ...
```

Primjer 1 - SQLite

SQLite

- Baza podataka koja je jednostavno spremljena u jednoj datoteci.
- Zbog toga ju je vrlo lako prenijeti s jednog servera na drugi.
- Spajanje na bazu:

```
1 import sqlite3;
2
3 # Baza je u datoteci 'database.sqlite' u podfolderu 'db'.
4 db = sqlite3.connect( 'db/database.sqlite' );
```

- Rad s bazom je dalje isti kao u PyMySQL!
- Postoji više programa za rad sa SQLite bazama, na primjer (besplatni i open-source) DB Browser for SQLite.

Primjer pokazuje rješenje Zadatka 3 pomoću SQLite.