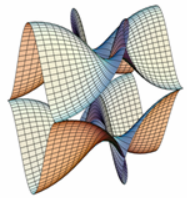

Prirodoslovno-matematički fakultet
Matematički odsjek
Sveučilište u Zagrebu

RAZVOJ WEB-APLIKACIJA

Predavanje 05 - Cookie. Session.

17. listopada 2025.

Sastavio: Zvonimir Bujanović

Cookie

HTTP poruke osim tijela sadržavaju i zaglavlje (header).

- Zaglavlje sadrži opis tipa podataka koje sadrži tijelo poruke (npr. `text/html`), duljinu poruke u byteovima, hostname servera i slično.
- Zaglavlje može sadržavati tzv. cookie (množina).
- Svaki cookie je par stringova: ključ-vrijednost.
- Server može u svom odgovoru (*response*) postaviti nove cookie.
- Klijent potom pohranjuje taj cookie i u svakom idućem kontaktu s tim serverom šalje sve pohranjene cookie.
- Tako server može dobiti informacije o ranijoj interakciji s klijentom.
  - Zapamtiti da li se klijent ulogirao i njegov username.
  - Zapamtiti npr. sadržaj klijentove košarice.
  - Zapamtiti postavke (odabrani jezik, svijetlu/tamnu temu).
- Cookie ima rok trajanja, nakon čega se automatski briše.
- Server može obrisati cookie.
- Firefox: Web developer tools -> Storage -> Cookies.

Cookie

Postavljanje novog, čitanje i brisanje postojećeg cookie.

```
from flask import Flask, render_template, make_response, request;

@app.route('/')
def index():
    resp = make_response( render_template(...) );

    # Cookie će isteći kad klijent zatvori browser.
    resp.set_cookie( 'kljuc', 'vrijednost' );

    # Cookie će isteći nakon 1 sata = 3600 sekundi.
    resp.set_cookie( 'drugi-kljuc', 'druga-vrijednost', max_age=3600 );

    # Čitanje postojećeg cookie kojeg smo postavili ranije.
    # request.cookies je ImmutableMultiDict.
    vrijednost = request.cookies.get( 'stari-kljuc' );

    # Brisanje ranije postavljenog cookie.
    resp.delete_cookie( 'stari-kljuc' );

    # Response sad uz HTML šalje i 2 nova cookie te briše jedan stari.
    return resp;
```

Zadatak 1

Napravite Flask aplikaciju koja generira web-stranicu ovako:

- Web-stranica ima inicijalno bijelu boju pozadine.
- Ako je korisnik već ranije bio posjetio stranicu i promijenio joj boju pozadine, onda se prikazuje ta ranije odabrana boja pozadine.
- Na web-stranici se nalazi padajući izbornik (select) na kojem korisnik može odabrati nekoliko unaprijed zadanih boja. (Postavite neki odabir kao defaultan.)
- Također, nalazi se textbox u kojeg korisnik može unijeti HTML kod boje.
- Klikom su gumb submit, ponovno se poziva ista skripta koja:
  - Ako je išta upisano u textbox, postavlja tu boju pozadine na web-stranicu.
  - U protivnom, postavlja boju pozadine koja je odabrana u padajućem izborniku.

Možete li svojim unosom u formu ostvariti neplanirani efekt?

Neki sigurnosni aspekti

Vrlo je lako slučajno napraviti grešku koja će ugroziti sigurnost servera!

- Cross-site scripting (XSS)
  - Ubacivanje maliciozne skripte koja se izvršava na klijentu.
- Primjer: neka u `index.html` piše:

```
<p>Hello, {{ name }}!</p>
```

  Što će se dogoditi ako netko pristupi adresi
  `skripta?name=Pero<script>alert('XSS');</script>`
- Primjer: neka u `index.html` piše:

```
<input type="text" value={{iznos}}>
```

  Što će se dogoditi ako netko pristupi
  `skripta?iznos=10000 onmouseover=<script>...</script>`

Neki sigurnosni aspekti

Treba provesti *sanitizaciju* svih podataka koje nam korisnik šalje!

Jinja2 to radi za nas automatski kod ispisa sa `{{ username }}`.

- Takav ispis zamjenjuje specijalne znakove poput `<>&"` HTML kodovima `<>&&quot;`.
- To se zove HTML escaping.
- Provjerite generirani HTML kod u browseru!
- Vrijednost HTML atributa uvijek treba stavljati u navodnike!

```
1 <input type="text" value="{{ iznos }}">
```

- Ako smo 100% sigurni i provjerili unos korisnika, možemo ga ispisati i bez HTML escaping-a ovako:

```
1 <input type="text" value="{{ iznos|safe }}">
```

Neki sigurnosni aspekti

Ponekad svejedno treba dodatna provjera - na primjer, ispis unutar CSS-a je imao neželjen efekt.

- Za to možemo koristiti regularne izraze i paket **`re`**.

```
import re;

@app.route( '/skripta' )
def handle_post():
    iznos = request.form.get( 'iznos' );
    iznos_ok = False;
    if( iznos ):
        # Nije None i nije prazan string.
        if( re.match( r'^[1-9][0-9]{0,4}$', iznos ) ):
            # Iznos je pozitivan broj sa max 5 znamenki.
            iznos = int(iznos);
            iznos_ok = True;

    if( not iznos_ok ):
        ...
```

Zadatak 2

Korigirajte kod Zadatka 1 tako da napravite sanitizaciju i validaciju podataka koje skripta dobiva od korisnika:

- Kao unos u textbox, dozvolite samo znak # iza kojeg slijedi tro- ili šesteroznamenkasti hex kod boje.
- Kao unos u select, dozvolite samo riječi sastavljene od max 20 slova engleske abecede.

Nikad ne vjerujte unosu korisnika!

Cookie $\mapsto$ Session

Pomoću cookie možemo zapamtiti podatke o korisniku koji ponovno posjećuje web-stranicu.

- Na primjer, ako klijent ispravno pošalje username i password, postavimo cookie (npr. `username`).
- Kada idući put posjeti stranicu, provjerimo postoji li taj cookie $\rightsquigarrow$ tako znamo je li korisnik već ulogiran.
- Kad se odlogira, obrišemo taj cookie.
- Možemo dodati i druge cookie za tog korisnika (sadržaj košarice i sl.)

Skup svih podataka koje čuvamo o takvom korisniku zovemo session.

Cookie ↦ Session

Klijent bi mogao varati i sam izmijeniti/generirati svoj session.

- Flask može zapakirati sve podatke koje želimo čuvati u jedan cookie.
- Taj cookie može kriptografski potpisati tako da ga korisnik ne vidi u otvorenom tekstu.

Ovakav pristup zovemo cookie-based ili client-side session.
U Flasku je implementiran pomoću objekta session.

Potencijalni problemi:

- Postoji ograničenje na max količinu tih podataka (4kB).
- Podaci su u nekom obliku ipak dostupni korisniku.
- Podaci nisu zaštićeni od dekripcije, nego samo od izmjene.

Stoga ćemo koristiti server-side session.

Session

Server-side session

- Prilikom prve konekcije server generira cookie (session-id) koji jednoznačno određuje tog klijenta.
- Tom klijentu (session-id-u) mogu se pridružiti podaci koji se spremaju na serveru (u bazi, datoteci, ...).
- Svaki idući kontakt tog klijenta server može detektirati preko njegovog session-id-a poslanog kao cookie.

Ovakav session u Flasku omogućava ekstenzija flask-session.

```
1  from flask import Flask, request, session;
2  from flask_session import Session;
3
4  # ------------------------------------ Konfiguracija
5  app = Flask( __name__ );
6
7  # Klijentovi podaci se spremaju u datoteku, po defaultu u folder ./flask_session.
8  app.config['SESSION_TYPE'] = 'cachelib';
9  app.config['SESSION_PERMANENT'] = False; # Podaci se brišu zatvaranjem browsera.
10
11 Session(app); # Incijaliziramo session za aplikaciju.
```

Session

Nakon ove konfiguracije dostupan je dict `session`.

- Podatke dohvaćamo sa `session.get('podatak')` ili `session['podatak']`.
- Podatke dodajemo ili mijenjamo sa `session['podatak']=val;`
- Sve podatke iz sessiona možemo obrisati sa `session.clear()`.
- Možemo i posve "zaboraviti" na korisnika brisanjem cookie-a `session`. To bi trebalo raditi npr. prilikom odlogiravanja.

Zadatak 3

Napravite Flask aplikaciju koja:

- Inicijalizira sumu na 0.
- Preko forme omogućava korisniku unos novog cijelog broja.
- Klikom na gumb prizbraja uneseni broj u sumu, ispisuje sumu, te omogućava unos idućeg broja.
- Omogućava resetiranje sume na 0 (tj. povratak na početak).

Pratite što se događa sa session cookiem u browseru.
Pratite što se događa u podfolderu `flask_session`.

Primjer 1: Mehanizam ulogiravanja u aplikaciju

Ulogiravanje korisnika se tipično radi pomoću session.

- Korisnik je ulogiran akko `session.get('username') is not None`.
- Ako korisnik nije ulogiran, a ruta to zahtijeva
  ⇝ preusmjerimo korisnika na rutu `/login`.
- Ruta `/login` ispisuje formu za ulogiravanje.
  - Ista ruta provjerava jesu li uneseni username+password ispravni.
  - Ako jesu, postavlja `session.get('username')`.
- Vidi `primjer1`.

Rute koje zatijevaju da je korisnik ulogiran se zovu zaštićene.

- Zaštitu rute možemo elegantnije ostvariti putem dekoratora.
- Vidi `primjer1-dekorator`.

Operacije s autentifikacijom možemo odvojiti u zasebni modul.

- Vidi `primjer1-usermodule`.

Zadatak 4

Napravite Flask aplikaciju koja omogućava igranje igre "Pogađanje brojeva".

- Kad korisnik prvi put dođe na stranicu `/`:
  - Aplikacija ga pita za ime.
  - Generira se slučajan broj X između 1 i 100. (Naravno, ne ispišemo ga korisniku.)
- Kad ima ime, aplikacija preusmjeri korisnika na stranicu `/pogodi`.
- Na ruti `/pogodi` korisnik može pogađati broj unosom u formu.
  - Klikom na gumb "Pogodi", broj se šalje na istu rutu te se ispiše je li korisnikov broj veći, manji ili jednak X.
  - Nakon što korisnik pogodi broj, ispisuje se prigodna čestitka.
  - Korisnik u bilo kojem trenu može resetirati igru na početak.

DZ:

- Da igraču bude lakše, ispisujte i njegov najveći pokušaj koji je manji od X, kao i najmanji pokušaj koji je veći od X.

Debugiranje: `print(request.form)`, `print(session)` u Pythonu ili `{{request.form}}`, `{{session}}` u HTML-u.

Zadatak 5

Napravite Flask aplikaciju koja omogućava igranje igre križić-kružić.

- Pretpostavite da oba igrača igraju naizmjenično za istim računalom. (Zasad ne znamo implementirati da sjede svaki za svojim!)
- Uputa: ako u formi postoje buttoni tipa `submit` koji imaju nameove A i B, onda će biti ili `request.form.get('A')==None` ili `request.form.get('B')==None`, ovisno o tome koji je gumb kliknut.
- Alternativno, svaka od 9 "kućica" za igru može sadržavati svoju formu, a u svakoj formi se uz vidljivi gumb nalazi i skriveno polje (`input type="hidden"`) koje sadrži identifikator kliknutog gumba.

Neki sigurnosni aspekti

Form spoofing / Cross-site request forgery (CSRF)

- Korisnik je ulogiran na naš site preko session/cookie, te nam (tj. našoj skripti) je poslao neke podatke u formi.
- Bez prekida sesije, korisnik ode na maliciozni site koji mu također ponudi naizgled "normalnu" formu.
- No "normalna" forma zapravo šalje podatke našoj skripti na našem site-u.
- Kako korisnik i dalje ima sesiju s našim siteom, skripta na našem site-u ne može razlikovati podatke koji joj stižu od forme na našem site-u i od forme na malicioznom.
- Ovaj napad je stvarna opasnost i sve forme bi morale imati zaštitu!

Neki sigurnosni aspekti

Form spoofing / Cross-site request forgery (CSRF)

- Zaštita: skriveno CSRF polje u formi.
- CSRF token je slučajan niz od npr. 64 slova koji spremimo u session.

```
1 session['csrf_token'] = generate_random_string(64);
```

- Isti taj niz spremimo u skriveno polje u formi.

```
1 <input type="hidden" name="csrf_token" value="{{session['csrf_token']}}">
```

- Kada obrađujemo podatke iz forme moramo provjeriti da li vrijedi:

```
1 if( request.form.get('csrf_token') == session['csrf_token'] ):
```

- Time smo provjerili je li zaista naša aplikacija napravila tu formu iz koje nam dolaze podaci. Maliciozni site/forma ne može nikako znati token iz sessiona koji je generirala naša aplikacija.

Zadatak 6

Napravite Flask aplikaciju koja:

- Validira sve podatke unešene u formu iz Zadatka 1 iz Predavanja 02.
- Ugradite zaštitu od CSRF napada u tu formu i aplikaciju.

Primjer 2: flask-wtf

Paket flask-wtf olakšava manipulaciju formama.

- Omogućava Flasku da koristi tzv. WTForms.
- Omogućava laku validaciju ispravnosti unesenih podataka u formu.
  - Je li neki podatak obavezno unijeti?
  - Je li uneseni podatak ispravne duljine?
  - Je li uneseni podatak ispravni email?
  - Odgovara li uneseni podatak nekom regularnom izrazu?
- Omogućava vrlo lagano generiranje i provjeru CSRF tokena.

Pogledajte Primjer 2 uz ova predavanja.

Primjer 3: flask-login

Paket flask-login olakšava ulogiravanje korisnika.

- Radi iznimno slično našem ulogiravanju putem modula `user`.
- Dolazi s gotovim dekoratorom `@login_required` i funkcijama `current_user()`, `login_user()`, `logout_user()`.

Pogledajte Primjer 3 uz ova predavanja.

url_for

Radi jednostavnosti na ovim slajdovima koristimo (na primjer):

```
<link rel="stylesheet" href="/static/style.css">
<a href="/static/slika.jpg">link na sliku</a>
<form action="/obradi" method="post">...</form>
return redirect( '/' );
```

To je u redu ako se naša aplikacija nalazi na korijenskoj putanji, na primjer https://example.com/ ili http://127.0.0.1:5000/.

⚠ Oprez

Međutim, ako se naša aplikacija nalazi na pod-putanji, na primjer https://example.com/aplikacija/, onda je nužno koristiti `url_for()`:

```
<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
<a href="{{ url_for('static', filename='slika.jpg') }}">link na sliku</a>
<form action="{{ url_for('obradi') }}" method="post">...</form>
return redirect( url_for('index') );
```

Prvi parametar od `url_for` je ime funkcije (ili `'static'`), a ne ime rute!