

Razvoj programske podpore za web i pokretne uređaje

**- predavanja -
2021./2022.**

11. Dinamički web

3/4

Creative Commons



- slobodno smijete:
 - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
 - **prerađivati** djelo
- pod sljedećim uvjetima:
 - **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
 - **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
 - **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.

Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.

Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

Tekst licence preuzet je s <http://creativecommons.org/>

Nastavak prošlog predavanja...

- ALI:
 - Naše aplikacije temeljene na dinamičkom webu ne podržavaju više korisnika?!
- Rješenje:
 - Održavanje stanja aplikacije na više razina
 - Praćenje interakcija s više različitih korisnika istovremeno

Sustavi i pamćenje stanja

- Procesi bez pamćenja stanja (*stateless*)

$$o_i = f(i_i)$$

Izlaz operacije ovisi samo o ulazu

- Procesi s pamćenjem stanja (*stateful*)

$$s_i = f_s(i_i, s_{i-1})$$

Novo stanje sustava ovisi o ulazu i povijesti sustava (prethodnom stanju)

$$o_i = f_o(i_i, s_{i-1})$$

Izlaz operacije ovisi o ulazu i povijesti sustava (prethodnom stanju)

Web i stanja sustava

- Proces posluživanja stranica/statičkih resursa jednostavan proces bez pamćenja stanja

$\text{resurs}_1 = \text{GET}(\text{URI}_1)$

$\text{resurs}_2 = \text{GET}(\text{URI}_2)$

...

- Poslužitelj ne raspoznaje:
 - različite preglednike i korisnike od kojih zahtjevi za stranicama/resursima dolaze
 - niz zahtjeva od strane pojedinog preglednika ili korisnika
- HTTP
 - Aplikacijski protokol bez stanja (*stateless*)
 - Transakcije dohvata resursa su međusobno neovisne

Primjer 1 - stateless

express

routes/index.js

```
const express = require('express');
const router = express.Router();
```

```
//global state
```

```
let globalAccessCounter = 0
```

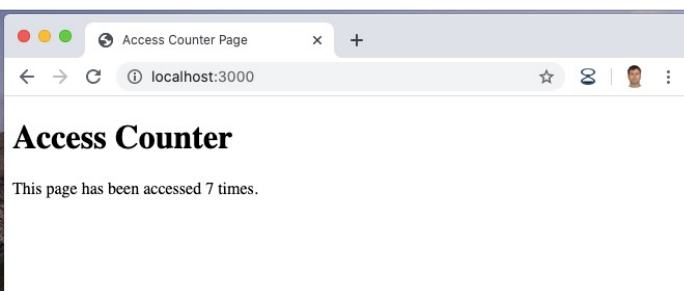
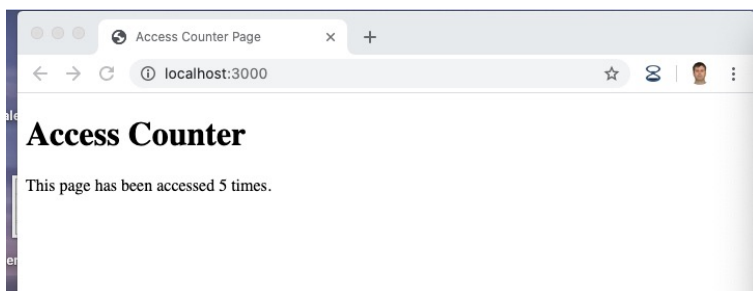
```
/* GET home page. */
```

```
router.get('/', function(req, res, next) {
  res.render('index', { counterValue: ++globalAccessCounter });
});
```

```
module.exports = router;
```

Varijabla kao globalno stanje (na razini instance poslužitelja), prestankom rada poslužitelja gubi se stanje

Promjena globalnog stanja na svaki zahtjev na *endpoint*



Web i ostvarenje složenijih sustava (I)

- Potrebno je ostvariti:

odgovor = metoda (resurs, korisnik, stanje*)

- Stanje* - jednostavno ili složeno

1. Na razini čitavog sustava
2. Na razini pojedinog korisnika sustava
3. Na razini niza transakcija (sjednice) između korisnika i sustava

- Primjeri stanja

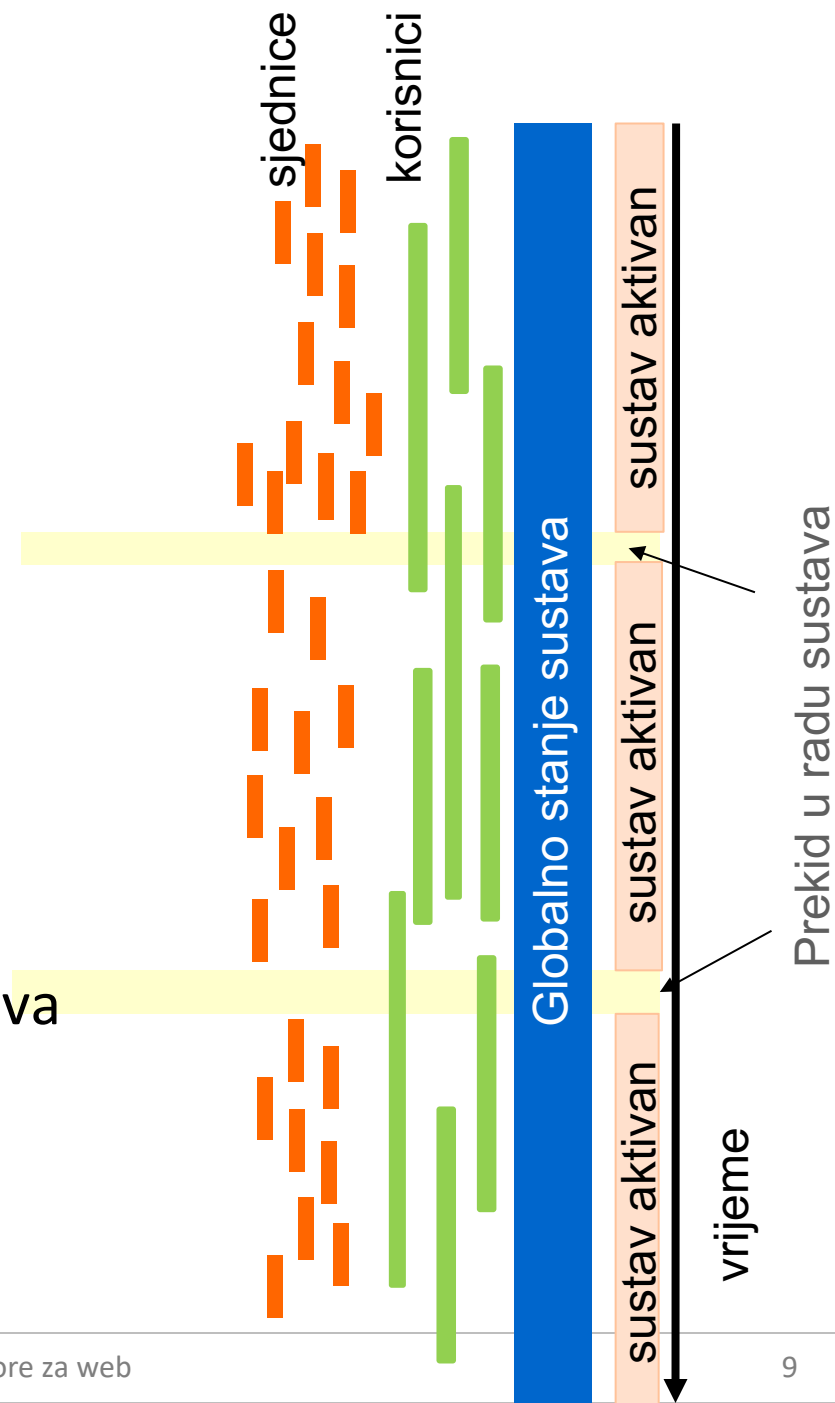
1. globalno – stanje artikala na zalihi web trgovine
2. korisničko – stanje košarice korisnika web trgovine
3. sjedničko – status autorizacije korisnika sustava

Web i ostvarenje složenijih sustava (II)

- Neka otvorena implementacijska i filozofska pitanja:
 - Tko pamti stanje s ?
 - Klijent, poslužitelj, oba pamte dio složenog stanja?
 - Tko vrši promjenu stanja $s_i = f_s(i_i, s_{i-1})$?
 - Klijent, poslužitelj?

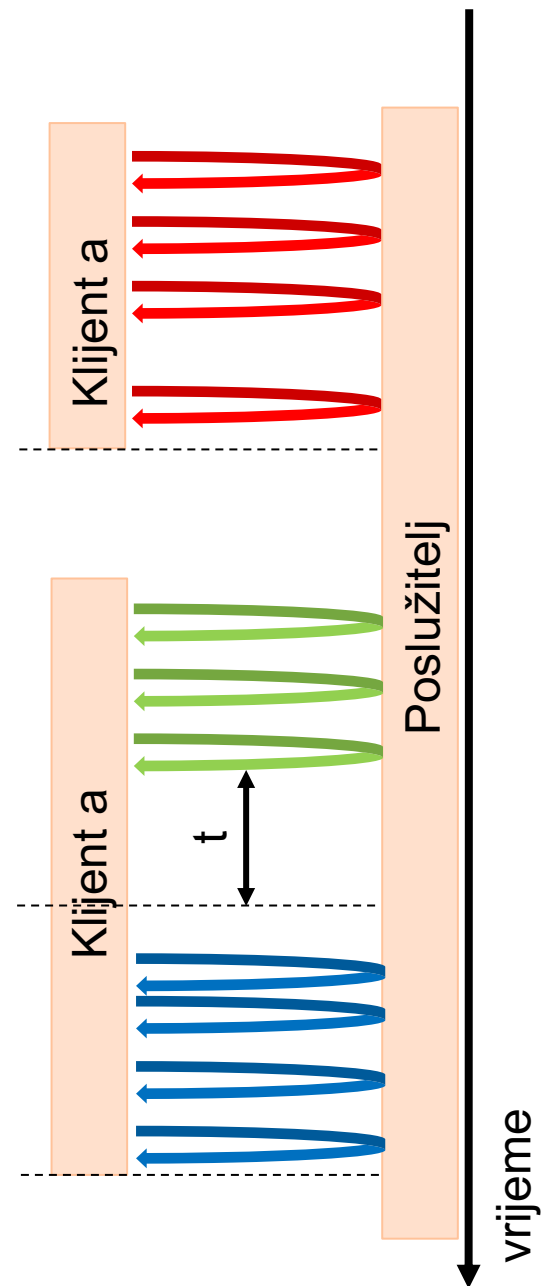
Trajnost podataka o stanju

- Pohrana podataka o stanju
 - **Tranzijentna** pohrana – nema trajnog čuvanja stanja (najčešće stanja **sjednice**)
 - **Perzistentna** pohrana – trajno čuvanje stanja (najčešće stanja **korisnika** i **sustava**)
 - Perzistentno pohranjena stanja moraju se očuvati bez obzira na povremene prekide u radu sustava



Sjednice (I)

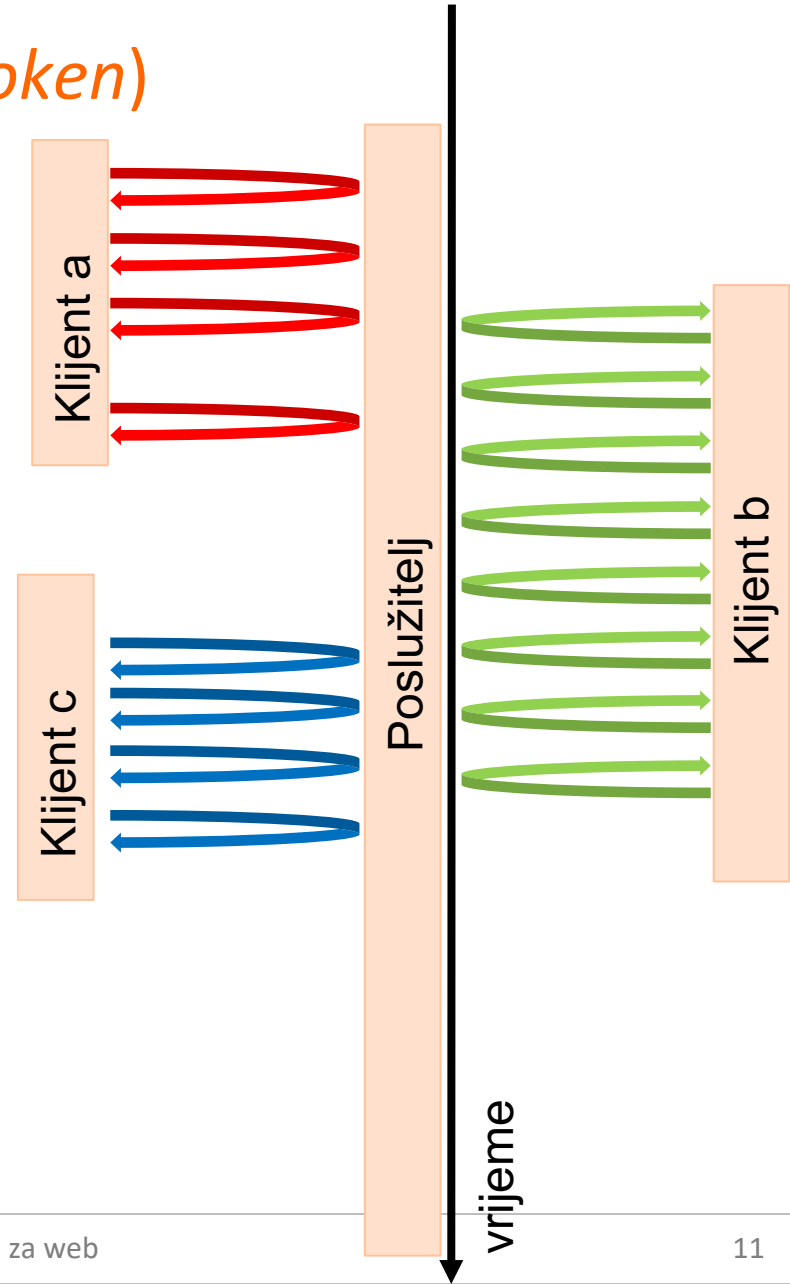
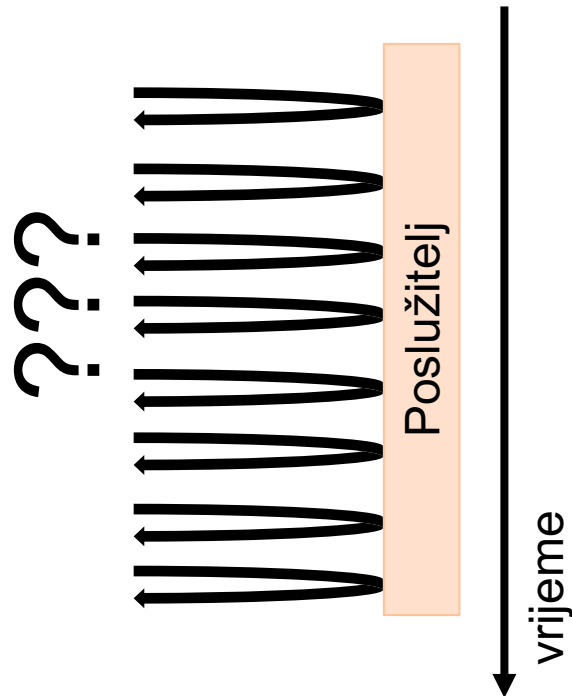
- **Sjednica (*session*)** – slijed vremenski omeđenih i logički povezanih transakcija između pojedinog klijenta i poslužitelja
 - **početak sjednice** određen npr. zahtjevom klijenta prema poslužitelju nakon duljeg vremenskog perioda neaktivnosti, prijavom korisnika itd.
 - **trajanje sjednice** (slijed logički povezanih transakcija između klijenta i poslužitelja)
 - **završetak sjednice** određen npr. prestankom rada klijenta, logičkim završetkom sjednice odjavom korisnika, izostanka transakcija itd.



Sjednice (II)

■ Identifikator sjednice (*session token*)

- Predstavlja konverzacijsko stanje
- jednoznačno određuje sjednicu
- pridijeljen svakoj transakciji koja pripada određenoj sjednici



Prenošenje stanja protokolom HTTP

- Gdje unutar HTTP zahtjeva i odgovora ima mjesta za prenošenje stanja (identifikatora sjednice)?
 1. Prvi redak zaglavlja (dio URI)
 2. Zaglavlje (parovi ime-vrijednost)
 3. Tijelo zahtjeva ili odgovora (resurs)
- Mehanizmi prenošenja podataka o stanju - obilježavanja sjednica:
 - **Skrivena polja** (*hidden fields*)
 - **Prepisivanje URLa** (*URL rewriting*)
 - **Kolačići** (*cookies*)

Skrivena polja

- Sadržaj stranice HTML uključuje i podatak o stanju/sjednici
- **Skriveno polje** (*hidden field*) unutar **obrasca**
- Dinamička stranica stvara HTML dokument
 - dokument sadrži HTML obrazac
 - stanje/identifikator sjednice smješten unutar skrivenog polja obrasca

```
<input name="naziv" type="hidden" value="SID=34cca95f43d">
```

- Slanjem sadržaja obrasca šalje se i sadržaj skrivenog polja
 - preporuča se metoda POST
 - preporuča se model **postback** (sadržaj obrasca se šalje URI-ju s kojeg je stranica s obrascem dobavljena)
 - vrijednosti skrivenog polja pristupa se istim mehanizmom kao i ostalim podacima iz obrasca

Skrivena polja – prednosti i nedostaci

■ Prednosti:

- neovisnost o pregledniku
- obrasci podržani na svim preglednicima
- ne mogu se onemogućiti na pregledniku, kao npr. kolačići
- jednostavnost korištenja, performanse

■ Nedostaci:

- lako dostupni pri pogledu na izvorni kôd stranice u pregledniku
- prenose se kod svake transakcije, u oba smjera
- zahtijevaju korištenje obrazaca (akcije *submit*)

Primjer 2 (I)

express

routes/index.js

```
...
//global state
let globalAccessCounter = 0

//session manager middleware
router.use(session.sessionManager);

/* GET home page. */
router.get('/', function(req, res, next) {

  if(req.session.access_counter === undefined)
    req.session.access_counter = 0;

  res.render('index', {
    counterValue: ++globalAccessCounter,
    userCounterValue: ++req.session.access_counter,
    sessionId: req.session.id
  });
});
...
```

Naš middleware
za implementaciju
sjednica

Objekt *session*
dodano u *req*
objekt od naše
middleware
funkcije

Varijabla *access_counter*
dodana u objekt *session*
naknadno (dio **sjedničkog**
konteksta)

Primjer 2 (II)

sessions/sessionFER.js

```
//session record store
let sessionStore = new Map();
...
//extract sessionID from GET or POST request
let sessionID = (req.query[sIDName] || req.body[sIDName]);
...
//fetch the session record
let sidRecord = sessionStore.get(sessionID);
...
if(!sidRecord) {
  sidRecord = {id: uuid.v4(), created: Date.now()};
  sessionStore.set(sidRecord.id, sidRecord)
...
//add the session record to the request object
req.session = sidRecord;
...
//pass the control to the next middleware layer
next();
```

Tablica aktivnih
sjednica

Dobavljanje
*identifikatora
sjednice* iz GET ili
POST zahtjeva

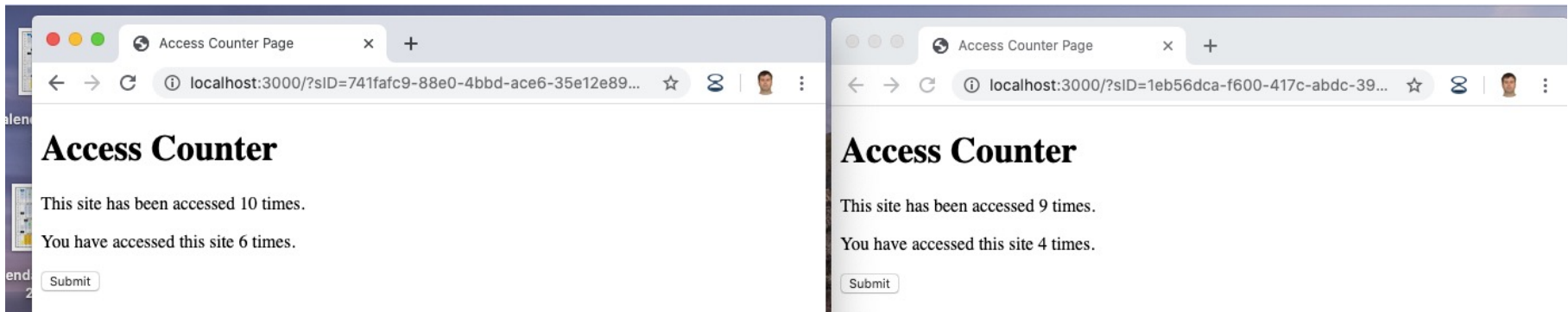
Dohvaćanje zapisa o
aktivnoj sjednici na
temelju zaprimljenog
identifikatora sjednice

Ako zapis o sjednici ne
postoji, stvori novi
identifikator sjednice i
ostale podatke,
pohrani zapis u tablicu

Dodavanje objekta
sjednice objektu *request*

Poziv sljedeće *middleware* funkcije u nizu

Primjer 2 (III)



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Access Counter Page</title>
5   </head>
6   <body>
7     <h1>Access Counter</h1>
8     <p>This site has been accessed 10 times.</p>
9
10    <!-- ispis broja posjeta pojedinog korisnika -->
11    <p>You have accessed this site 6 times.</p>
12
13    <form action="http://localhost:3000/" method="get">
14      <input type="text" hidden name="sID" value="741fafc9-88e0-4bbd-ace6-
15 35e12e891a12">
16      <input type="submit">
17    </form>
18  </body>
19 </html>
```

Prepisivanje URLa

- **Poveznice** na stranici **uključuju podatak o sjednici**
- Prepisivanje URL-a (*URL rewriting*) je:
 - mehanizam automatizirane promjene URL dolaznog zahtjeva na poslužitelj Weba, unutar ulaznog niza filtara poslužitelja
 - mehanizam dodavanja informacija unutar poveznica HTML stranice dostavljene pregledniku
 - podaci: parovi ime-vrijednost
 - alternativan mehanizam označavanja sjednica (kada mehanizam kolačića nije dostupan)
 - primjer prepisanog URL-a s podatkom o identifikatoru sjednice

`http://www.fer.unizg.hr/predmet/or?sid=234a3f0cc7`

Prepisivanje URL-a – prednosti i nedostaci

■ Prednosti:

- potpuna neovisnost o klijentu (pregledniku)
- ne može se onemogućiti na klijentu (pregledniku)
- jednostavna implementacija

■ Nedostaci:

- podaci se prenose unutar polja upita URI
- moguće koristiti vrlo ograničenu količinu podataka
- podaci dio svih poveznica koje vode na dinamičke stranice izvorišnog poslužitelja
- potrebna dodatna funkcionalnost kod implementacije
- ekstrakcija podataka o sjednici iz polja upita URI
- smanjena čitljivost poveznica

Primjer 3 (I)

express

routes/index.js

```
//create middleware method handler functions
function makeRouteWithCounter(template) {

  return function(req, res, next) {

    if(req.session.access_counter === undefined)
      req.session.access_counter = 0;

    res.render(template, {
      counterValue: ++globalAccessCounter,
      userCounterValue: ++req.session.access_counter,
      sessionID: req.session.id,
      s_url: session.sessionURLBuilder(req.session.id)
    });
  }
}

router.get('/', makeRouteWithCounter('index'));
router.get('/first', makeRouteWithCounter('index'));
router.get('/second', makeRouteWithCounter('second'));
router.get('/third', makeRouteWithCounter('third'));
```

Funkcija za stvaranje *middleware* funkcija (za naše rute s predlošcima stranica)

Dodavanje varijable u sjednički kontekst

Prosljeđivanje podataka predlošku stvaranje stranice

Funkcija kao parametar predloška !

Definiranje ruta za tri stranice

Primjer 3 (II)

sessions/sessionFER.js

```
//create a function for adding sessionID to URL
function sessionURLBuilder(sessionID) {

  //add sessionID parameter to URL query segment
  return function(ur1) {
    let newURL = new URL(ur1)
    newURL.searchParams.append(sIDName, sessionID)

    return newURL.toString()
  }
}
```

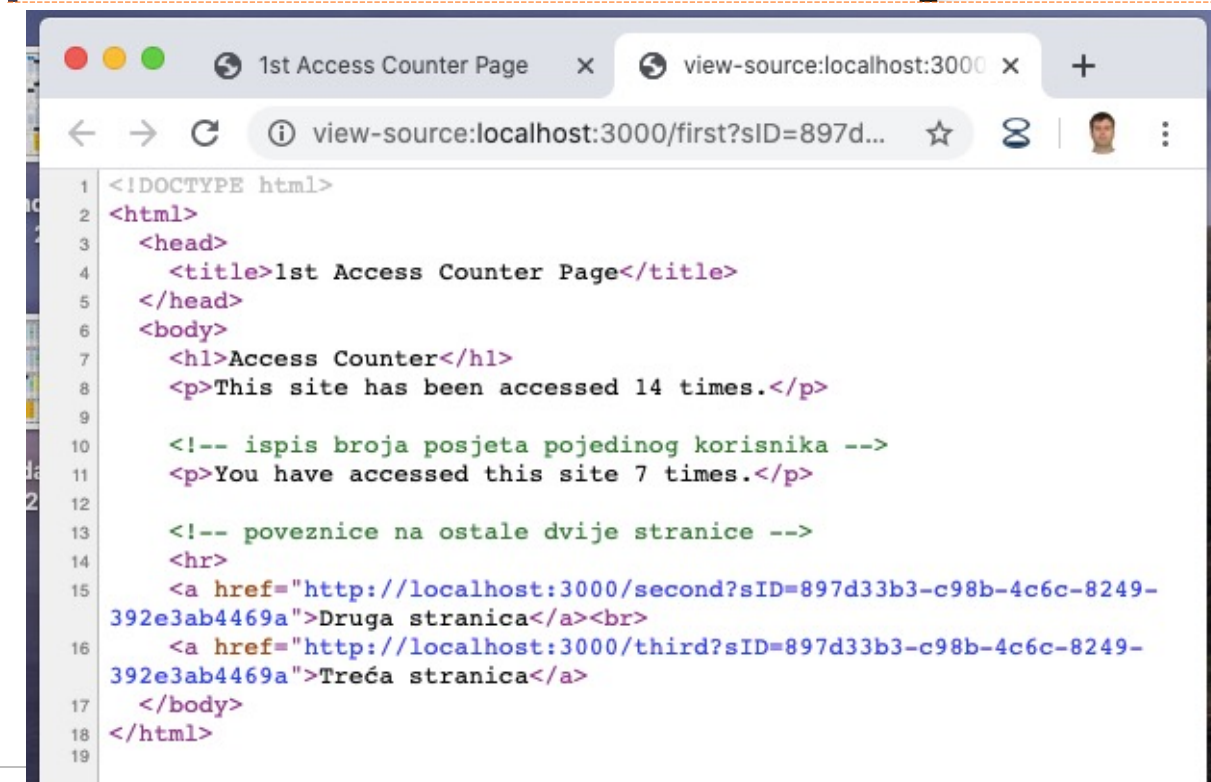
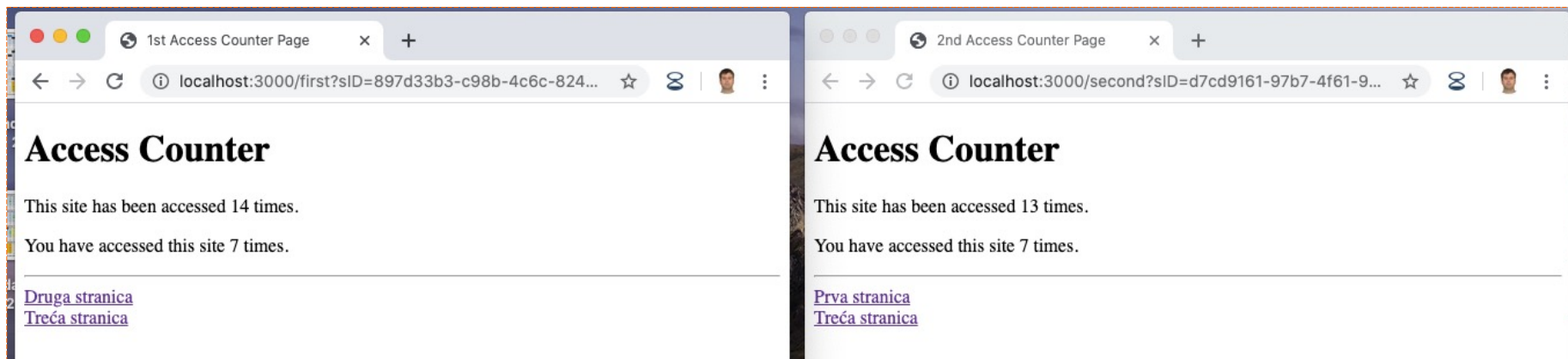
Pomoćna funkcija za
formatiranje URL-a s
dodatkom
identifikatora sjednice

Poziv pomoćne funkcije
unutar predloška

views/index.ejs

```
<a href="<%= s_url("http://localhost:3000/second") %>">Druga stranica</a><br>
<a href="<%= s_url("http://localhost:3000/third") %>">Treća stranica</a>
```

Primjer 3 (III)



Kolačići (cookies)

- Mehanizam razmjene **male količine** slobodno definiranih podataka između klijenta i poslužitelja unutar svake transakcije protokola HTTP
 - Opisani u RFC 6265
- Ukupna količina informacija unutar kolačića do 4kB
- Sadržaj kolačića: jedan par **ime=vrijednost**
- Meta-podaci (svojstva kolačića)
 - **domena**
 - **put**
 - **rok valjanosti**
 - **ograničenje pristupa**
 - **ograničenje na sigurnost prosljeđivanja**
 - **ograničenje na prosljeđivanje iz drugih domena**

} domena + put = **doseg kolačića (scope)**



Stvaranje/promjena kolačića (I)

- **Kolačiće stvara poslužitelj**
 - definira *sadržaj* i *svojstva* kolačića
 - uključuje ih u zaglavlje odgovora protokola HTTP
 - kolačić *može* stvoriti/promijeniti i klijent pomoću JavaScript kôda, ali nije uobičajeno
- **Kolačiće pohranjuje klijent**
 - prihvata kolačić iz zaglavlja odgovora protokola HTTP i pohranjuje ga u lokalnom spremištu
 - ako je kolačić istog imena s istog poslužitelja već bio definiran, prethodna definicija se zamjenjuje novom

Stvaranje/promjena kolačića (II)

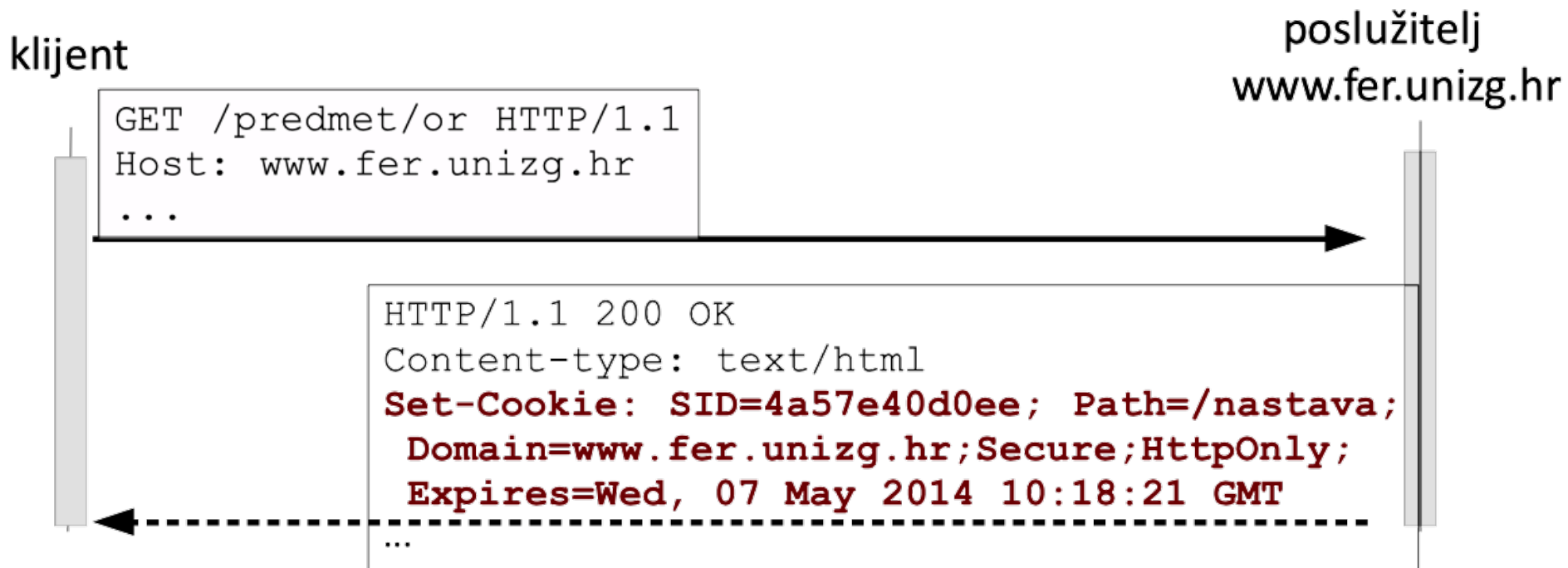
- Kod definiranja kolačića poslužitelj određuje:
 - **sadržaj** - par ime=vrijednost – **obavezan** dio kolačića
 - **domenu** - ako nije definirana, podrazumijeva se kvalificirano ime poslužitelja koji definira kolačić, npr. `www.fer.unizg.hr`
 - **put** - ako nije definiran, podrazumijeva se put dio URLja resursa dohvaćanog tekućom transakcijom, npr. ako klijent dohvaća resurs:

`http://www.fer.unizg.hr/nastava/or/labosi.html`

za svojstvo *put* se podrazumijeva vrijednost `/nastava/or`

- **rok valjanosti** (opcionalno)
- **ograničenje pristupa** (opcionalno)
- **ograničenje na sigurnost prosljeđivanja** (opcionalno)
- **ograničenje na prosljeđivanje iz drugih domena** (opcionalno, novo)

Postavljanje kolačića na klijentu - HTTP



Sadržaj kolačića: **SID=4a57e40d0ee**

Put za koji kolačić vrijedi: **Path=/nastava**

Domena za koju kolačić vrijedi: **Domain=www.fer.unizg.hr**

Kolačić se može prosljeđivati samo sigurnim kanalima: **Secure**

Kolačiću se ne može pristupiti lokalno: **HttpOnly**

Kolačić istječe: **Expires=Wed, 07 May 2014 10:18:21 GMT**

Alternativni istek vremena (sekundi od trenutka primitka kolačića na klijentu): **Max-Age=3600**

Prosljeđivanje kolačića (I)

- Klijent (preglednik) kod slanja svakog zahtjeva za resursom nekom poslužitelju pretražuje **lokalno spremište kolačića**
- Unutar zahtjeva se prosljeđuju svi kolačići koji zadovoljavaju **uvjete** (slijedi ...)
- Prosljeđivani kolačići sadrže **samo par ime=vrijednost**, meta podaci o kolačiću se ne prosljeđuju poslužitelju



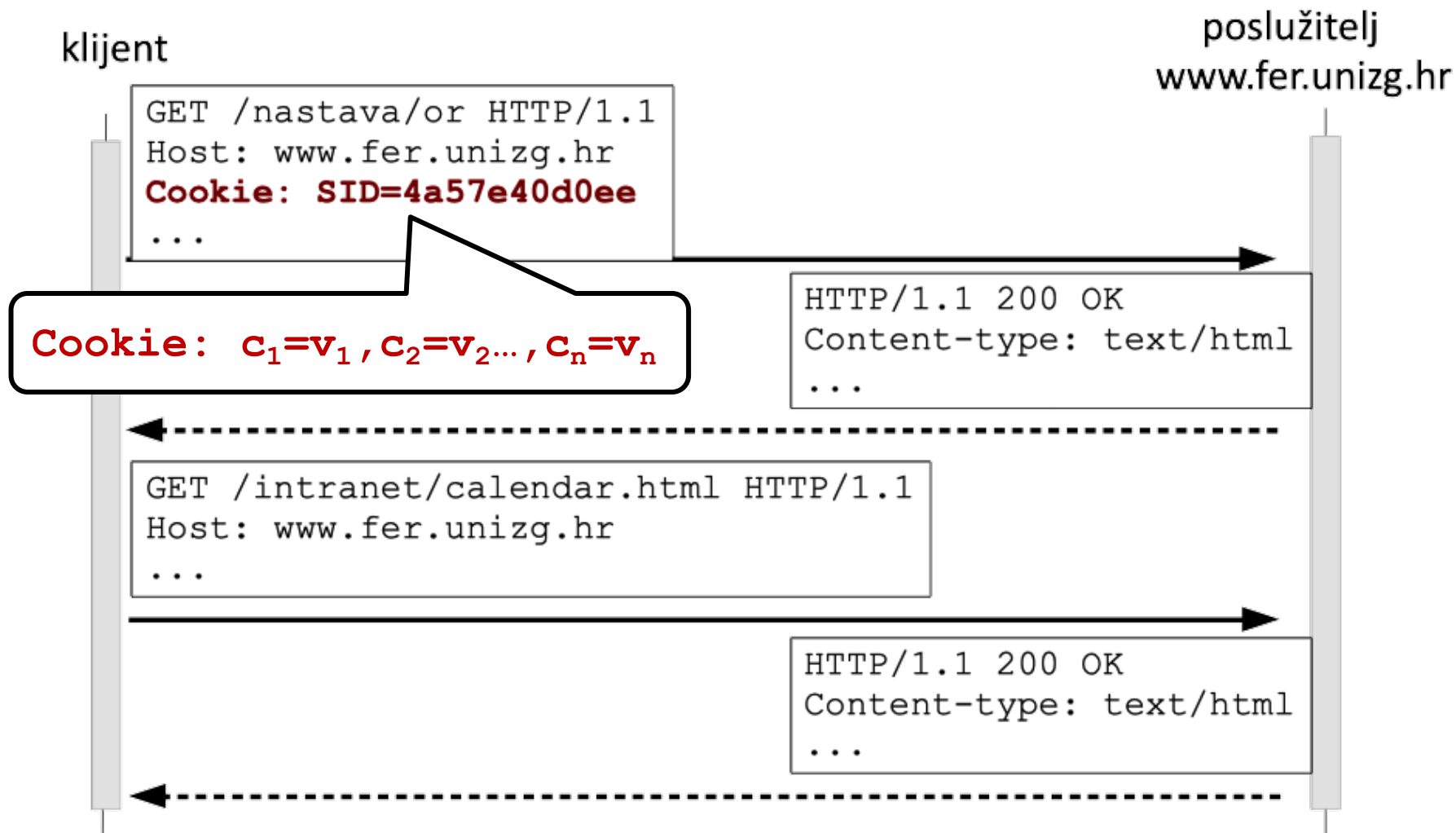
Prosljeđivanje kolačića (II)

- Unutar zahtjeva za resursom prema nekom poslužitelju, preglednik prosljeđuje sve kolačiće koji zadovoljavaju **svih pet navedenih uvjeta**:
 1. **Poslužitelj pripada domeni za koju je kolačić definiran**
 - npr. ako je cilj zahtjeva poslužitelj `www.fer.unizg.hr`, tada ovaj uvjet zadovoljavaju kolačići koji imaju vrijednost svojstva
 - `Domain=www.fer.unizg.hr` (*host-only)
 - `Domain=fer.unizg.hr`
 - `Domain=unizg.hr`
 - `Domain=.hr`
- ali ne i kolačići kojima je vrijednost svojstva
- `Domain=carnet.hr`
 - `Domain=google.com`

Prosljeđivanje kolačića (III)

2. Put definiran za kolačić u cjelini je sadržan unutar puta dohvaćanog resursa
 - npr. ako je za kolačić definirano svojstvo Path=/nastava/or, tada je uvjet zadovoljen za resurse čiji je segment puta URLja: /nastava/or/labosi, /nastava/or, ali ne i za resurse unutar puteva /nastava/opp, /intranet ...
3. Kolačiću nije istekao rok trajanja, ili kolačić nema definiran rok trajanja
4. Kolačići koji imaju definirano svojstvo *secure* mogu biti prosljeđivani samo sigurnim kanalima (https, ne http)
5. Samo ako je to izravno zabranjeno, kolačić neće biti prosljeđen iz druge domene

Prosljeđivanje kolačića (IV)



* pretpostavlja se da se u prvoj transakciji koristi siguran komunikacijski kanal (HTTPS), kolačić ne bi bio prenošen preko običnog, nesigurnog kanala (HTTP)

Trajnost kolačića (I)

- S obzirom na rok valjanosti kolačiće dijelimo na **trajne** i **privremene**
- **Trajni** (*persistent*) kolačići
 - imaju definirano vrijeme isteka valjanosti
 - brišu se nakon roka isteka valjanosti
- **Privremeni** (*transient*) kolačići
 - tzv. sjednički kolačići (*session cookies*)
 - nemaju definirano vrijeme isteka valjanosti
 - brišu se nakon prestanka rada klijenta



Trajnost kolačića (II)

- Brisanje kolačića
 - "ručno" brisanje od strane korisnika preglednika odabirom opcije brisanja (kolačića, povijesti ...)
 - Automatsko brisanje kolačića s isteklim vremenom trajanja od strane preglednika
 - Promjenom sadržaja kolačića (od strane poslužitelja ili JavaScript kôda) u cilju trenutnog isteka valjanosti
 - postavljanjem roka valjanosti koji je u trenutku postavljanja već istekao ($\text{MaxAge}=0$ ili $\text{MaxAge}=\text{now}() - 3600$)
 - Postavljanjem vrijednosti kolačića na „” (prazno)
 - Ne postoji mehanizam brisanja kolačića u protokolu HTTP (npr. zaglavlje Delete-cookie:)



Ograničenja lokalnog pristupa kolačiću

- Pristup kolačiću može biti
 - **udaljen** - poslužitelj(i), mehanizmom prosljeđivanja kolačića u sklopu HTTP odgovora
 - **lokalni** - klijent, programski – korištenjem JavaScript koda
- ako je postavljeno svojstvo ograničenja pristupa (**HttpOnly**), lokalni pristup na strani klijenta nije moguć
- Otežava XSS napade (pristup kolačićima korištenjem injektiranog JavaScript koda)
- Kolačići stvoreni od JavaScript koda ne mogu imati postavljeno svojstvo HttpOnly

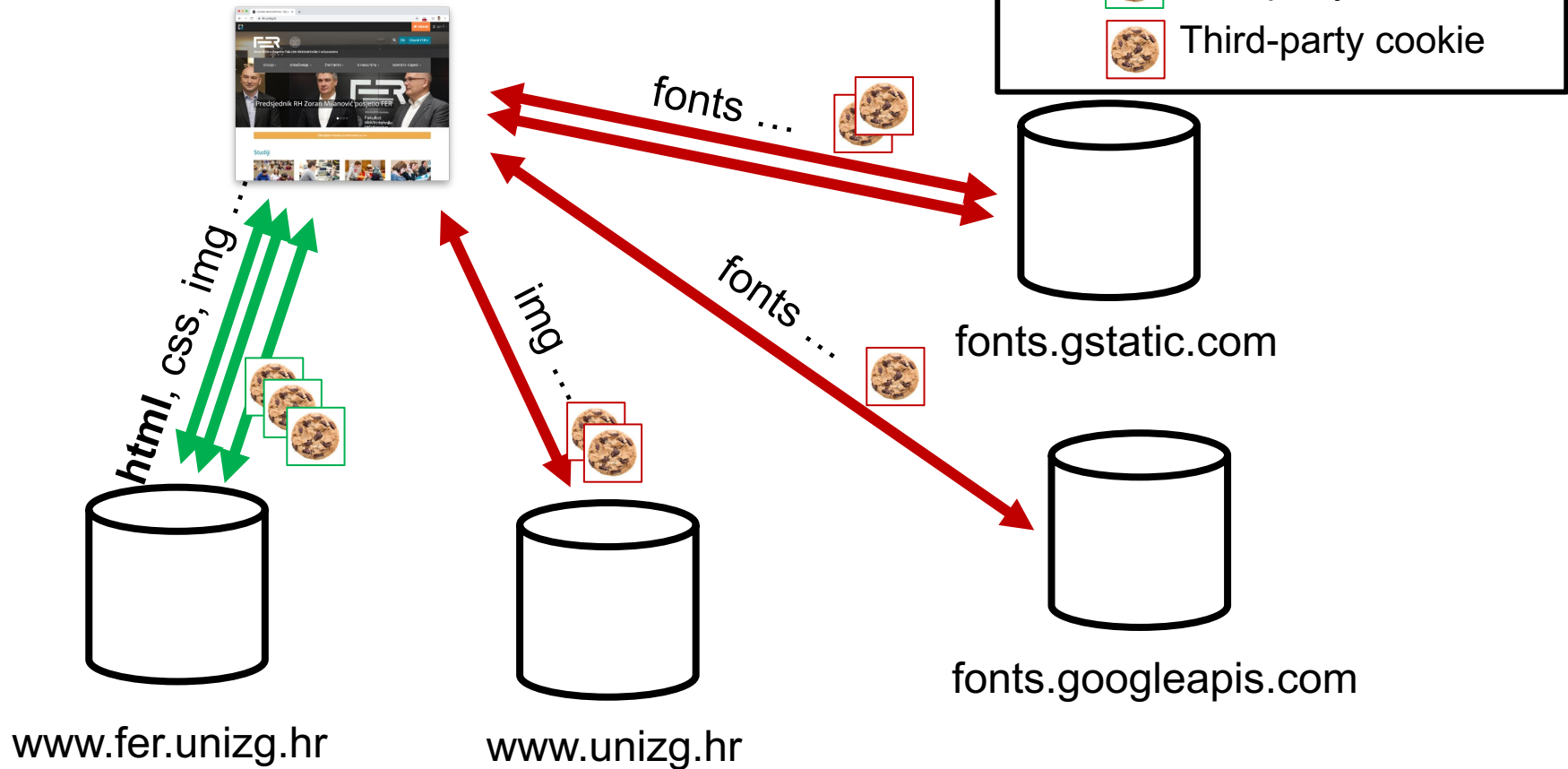
Ograničenje prosljeđivanja sigurnim kanalima

- Ograničenje prosljeđivanja kolačića na sigurne kanale
 - mehanizmi stvaranja i prosljeđivanja kolačića mogu koristiti bilo koji komunikacijski kanal između klijenta i poslužitelja (HTTP, HTTPS ...)
 - ako je postavljeno ograničavanje prosljeđivanja samo na sigurne komunikacijske kanale (*Secure*), kolačić neće biti prosljeđivan poslužitelju ako se koristi nesiguran kanal (npr. HTTP)
 - Kolačić s definiranim svojstvom *Secure* ne može biti postavljen preko nesigurne veze (vrijedi za novije preglednike)

Ograničenje na prosljeđivanje iz drugih domena (I)

domena stranice:

www.fer.unizg.hr



Ograničenje na prosljeđivanje iz drugih domena (II)

- Ograničenje na prosljeđivanje od preglednika do poslužitelja definirano svojstvom kolačića *SameSite*
 - *SameSite = (None | Strict | Lax)*
 - **None** – kolačić se prosljeđuje u sklopu i *same-site* i *cross-site* zahtjeva
 - **Strict** – kolačić se prosljeđuje samo u sklopu *same-site* zahtjeva
 - **Lax** – kolačić se prosljeđuje i u sklopu *cross-site* navigacijskih zahtjeva, ne i u sklopu zahtjeva za dohvaćanjem resursa, ali bez posljedične promjene domene
- Strict i lax postavke otežavaju CSFR (*cross-site request forgery*) tipove napada (korištenje aktivnih sjedničkih kolačića)

Kolačići – ograničenja

- Vrlo ograničena količina podataka unutar kolačića
 - do 4kB
- Povećanje količine prenošenih podataka
 - dodaju se svakom zahtjevu za resursom
- Trajnost podataka unutar kolačića
 - privremeni kolačići, istek roka, ručno obrisani ...
- Vezanost kolačića na klijenta, ne korisnika
 - različiti preglednici ne dijele kolačiće
 - različiti korisnici istog preglednika dijele kolačiće
- Zabrana prihvata i slanja kolačića na pregledniku
 - Detekcija zabrane, korištenje mehanizma URL Rewrite

Sigurnost i kolačići

- Čitljivost informacija unutar kolačića
 - vidljivi kod prijenosa nesigurnim komunikacijskim kanalom
 - zapisani na strani klijenta
- Presretanje kolačića
 - *krađa podataka iz kolačića prisluškivanjem*
- Slanje kolačića poslužitelju van definiranog dosega kolačića korištenjem JavaScript kôda
 - *Cross-site scripting, Session cookie theft, Session hijacking*
- Korištenje sjedničkih kolačića za pristup štićenim stranicama
 - *Cross-site request forgery*
- Maliciozna promjena sadržaja kolačića na pregledniku
 - *Cookie poisoning*

Privatnost i kolačići

- Praćenje navika i ponašanja korisnika
 - kolačići + sadržaj drugih poslužitelja ("third-party cookies")
- EU direktive glede korištenja kolačića
 - korisnici moraju imati mogućnost odbijanja pohrane kolačića
 - više na poveznici:
http://ec.europa.eu/ipg/basics/legal/cookies/index_en.htm#section_2

Primjer 4 (I)

routes/cookies.js

Postavljanje odlaznog kolačića u *response* objekt

```
res.cookie(req.query.name, req.query.value, { path: req.query.path })  
res.clearCookie(req.query.name, {path: req.query.path})
```

Brisanje odlaznog kolačića u *response* objektu

routes/page.js

```
router.get('/', function(req, res, next) {  
  
  res.render('page', {  
    title: 'Cookie spotter',  
    path: req.path,  
    cookies: req.cookies  
  });  
});
```

Dolazni kolačići u *request* objektu

Modul i cookie-parser middleware

app.js

```
const cookieParser = require('cookie-parser')  
  
//cookie parser middleware  
app.use(cookieParser());
```


Primjer 4 (II)

Kolačići za dani put

Prikaz kolačića u
Chrome Devtools

The screenshot shows a web browser window with the address bar at `localhost:3000`. The page title is "Cookie spotter" and it says "Welcome to /". Below this, it states "Cookies arrived with the page request:" and displays a table of cookies.

name	value
appuserID	igorcavrak@fer
sID	2c35e111-eff9-4616-b10a-e06239546d47

Overlaid on the right is the Chrome Devtools "Application" tab. The left sidebar shows the "Storage" section with "Cookies" selected. The main pane shows a table of cookies:

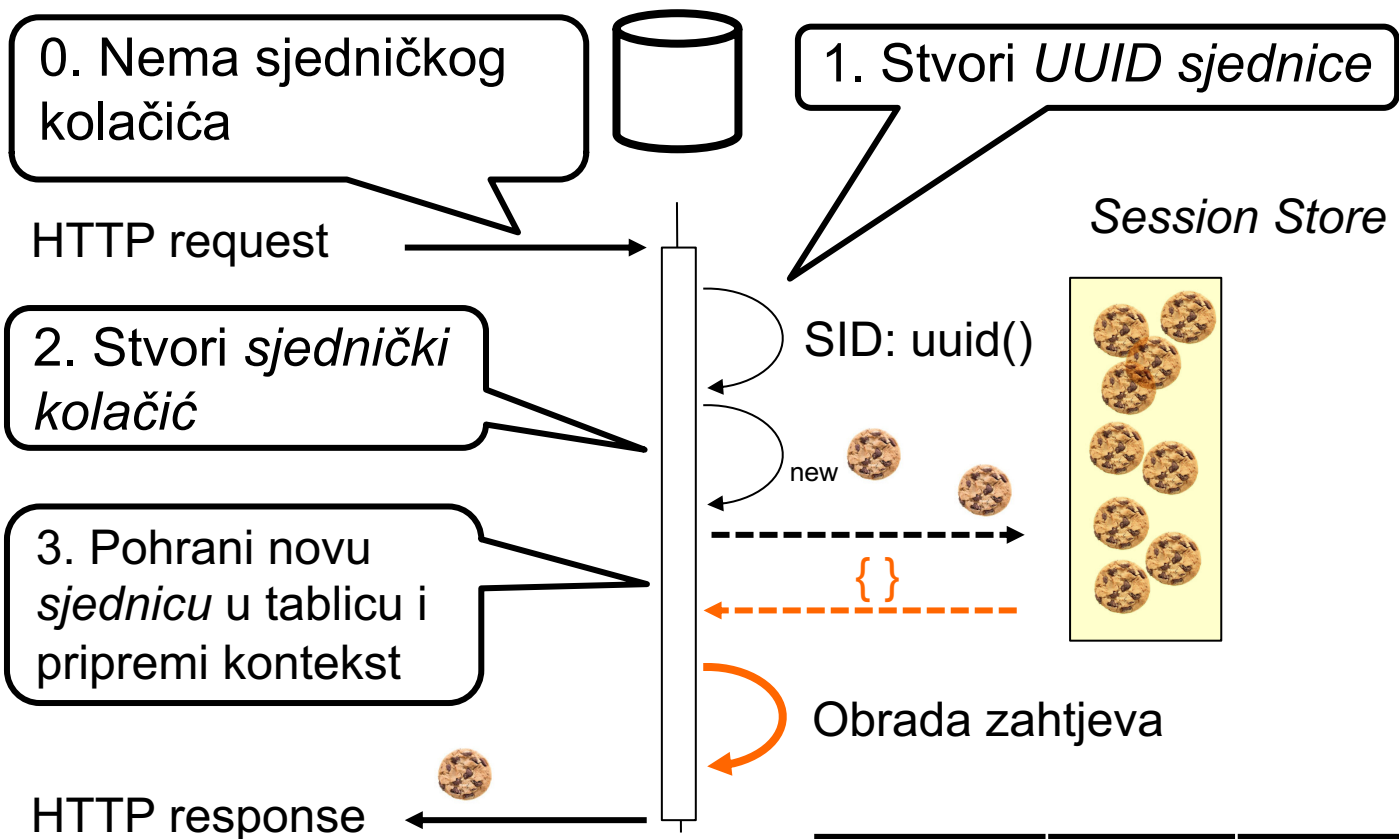
Name	Value	Domain	Path	Expires / ...	Size	Http...	S..	S..	Priority
sID	2c35e111-eff...	localhost	/	Session	39				Medi...
appuserID	igorcavrak%...	localhost	/	2020-05-...	25	✓			Medi...

Vrste i izvedba sjednica

(napokon)

Sjednice korištenjem kolačića (I)

■ Stvaranje (anonimne) sjednice



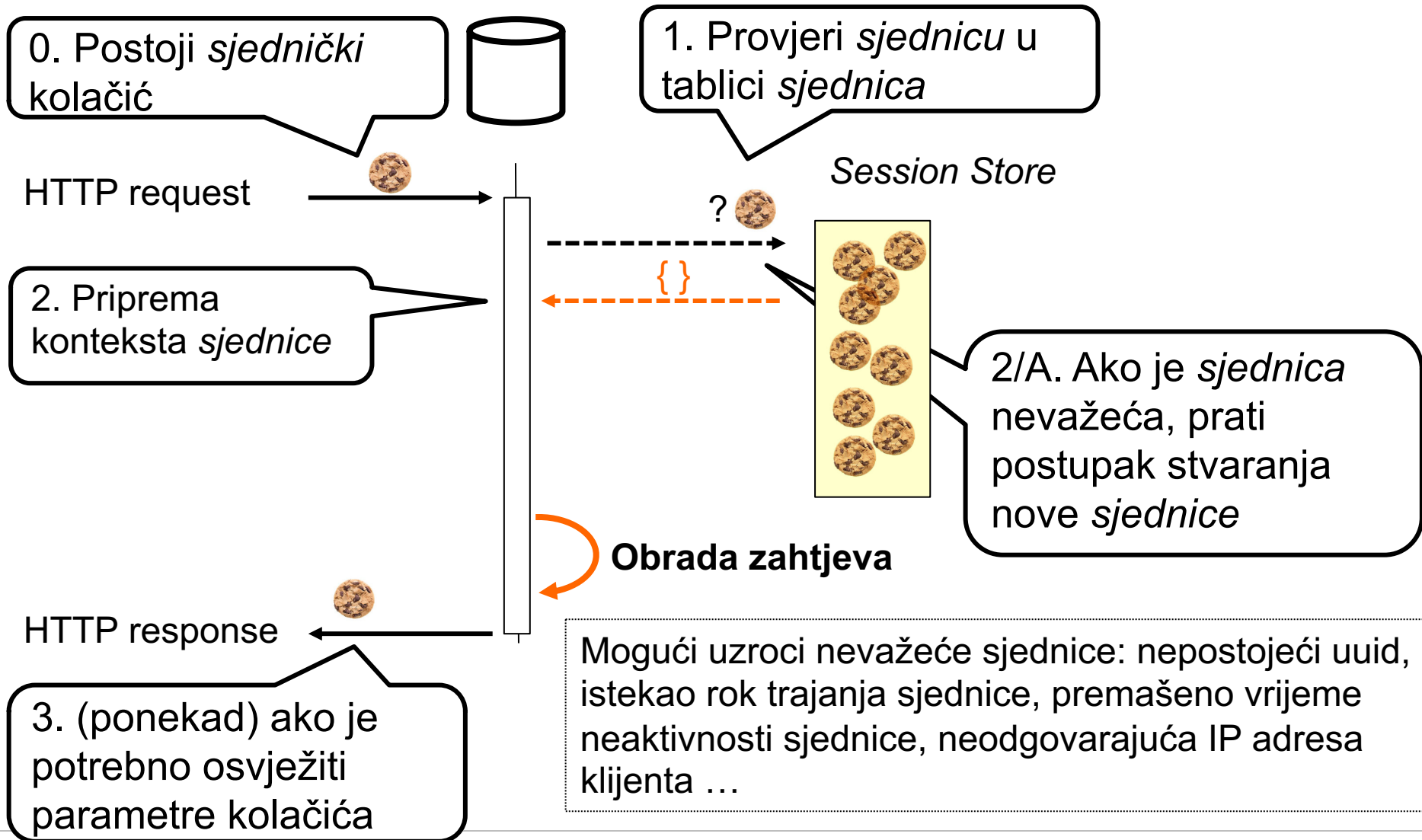
{ }
Kontekst sjednice:
stanje (variable)
vezano uz sjednicu



sessionID	created	accessed	IP	Context	...
2gy4-54g63yf...	23525343	23527345	205..	a=4, count=7, ...	

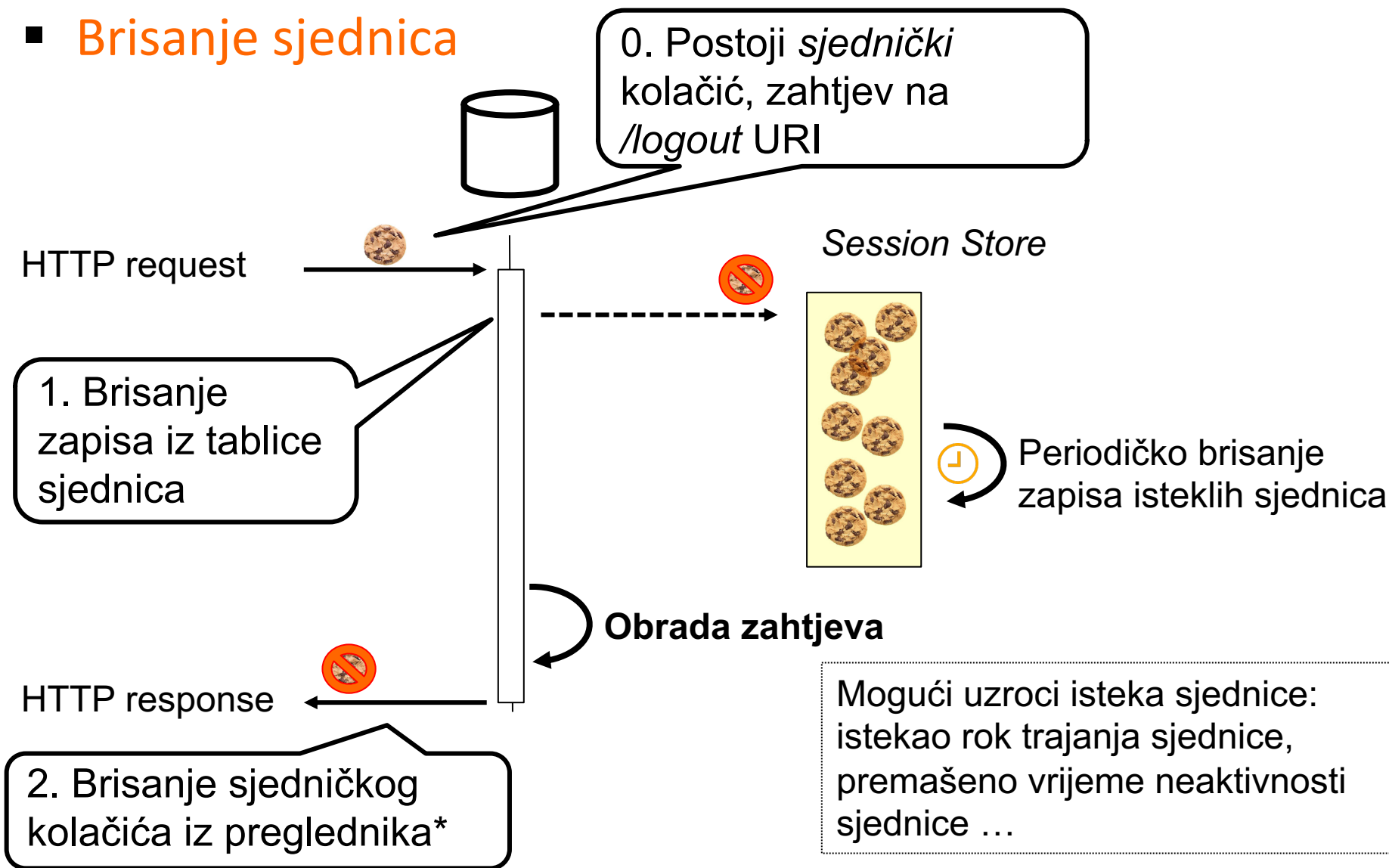
Sjednice korištenjem kolačića (II)

■ Provjera valjanosti sjednice



Sjednice korištenjem kolačića (III)

■ Brisanje sjednica



Primjer 5 (I)

express

sessions/sessionFER.js – Primjer 5

```
18 function sessionManager(req, res, next) {
19
20   //extract sessionID from cookie, GET or POST request
21   let sessionId = (req.cookies[sIDName] || req.query[sIDName] || req
22   console.log("Received sessionID: " + sessionId)
23
24   //fetch the session record
25   let sidRecord = sessionStore.get(sessionID);
26
27   if( sidRecord !== undefined )
28     console.log("Fetched session record: " + JSON.stringify(sidRec
29   else
30     console.log("Session record not found for sessionID: " + sessi
31
32   //check for session timeout
33   if( (sidRecord) && ((sidRecord.lastUsed + sessionTimeout) < Date.n
34     sessionStore.delete(sidRecord.id);
35     sidRecord = undefined;
36     console.log("Session expired: " + sessionID)
37   }
38
39   //if the session record does not exist, create one
40   if(!sidRecord) {
41     sidRecord = {id: uuid.v4(), created: Date.now()};
42     sessionStore.set(sidRecord.id, sidRecord)
43
44     //add session cookie to the response object
45     res.cookie(sIDName, sidRecord.id, { httpOnly: true })
46
47     console.log("Created new session record " + sIDName + ": " + s
48   }
49
50   //update the last used timestamp
51   sidRecord.lastUsed = Date.now();
52
53   //add the session record to the request object
54   req.session = sidRecord;
55
56   console.log("Session data attached to the request object: " + JSON
57
58   //pass the control to the next middleware layer
59   next();
60 }
```

Dohvat **sjedničkog identifikatora** iz dolaznih kolačića

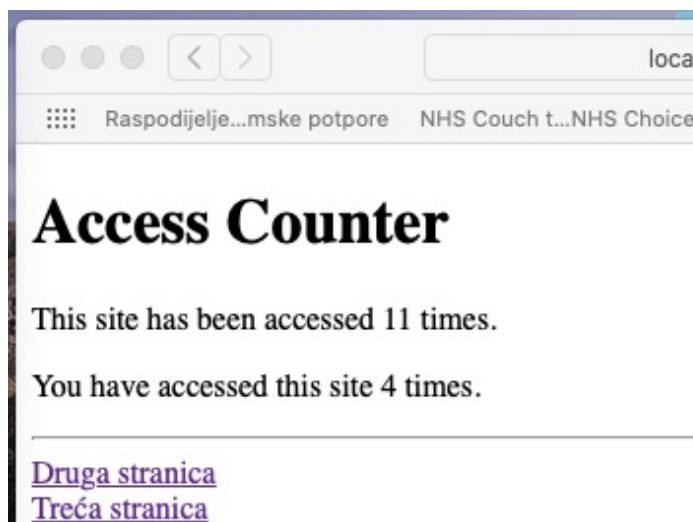
Postavljanje **sjedničkog identifikatora** u sjednički kolačić ako je stvorena nove sjednica

HttpOnly svojstvo – nema promjene kolačića na klijentu

sessions/sessionFER.js – Primjer 3

Primjer 5 (II)

Različiti identifikatori sjednica



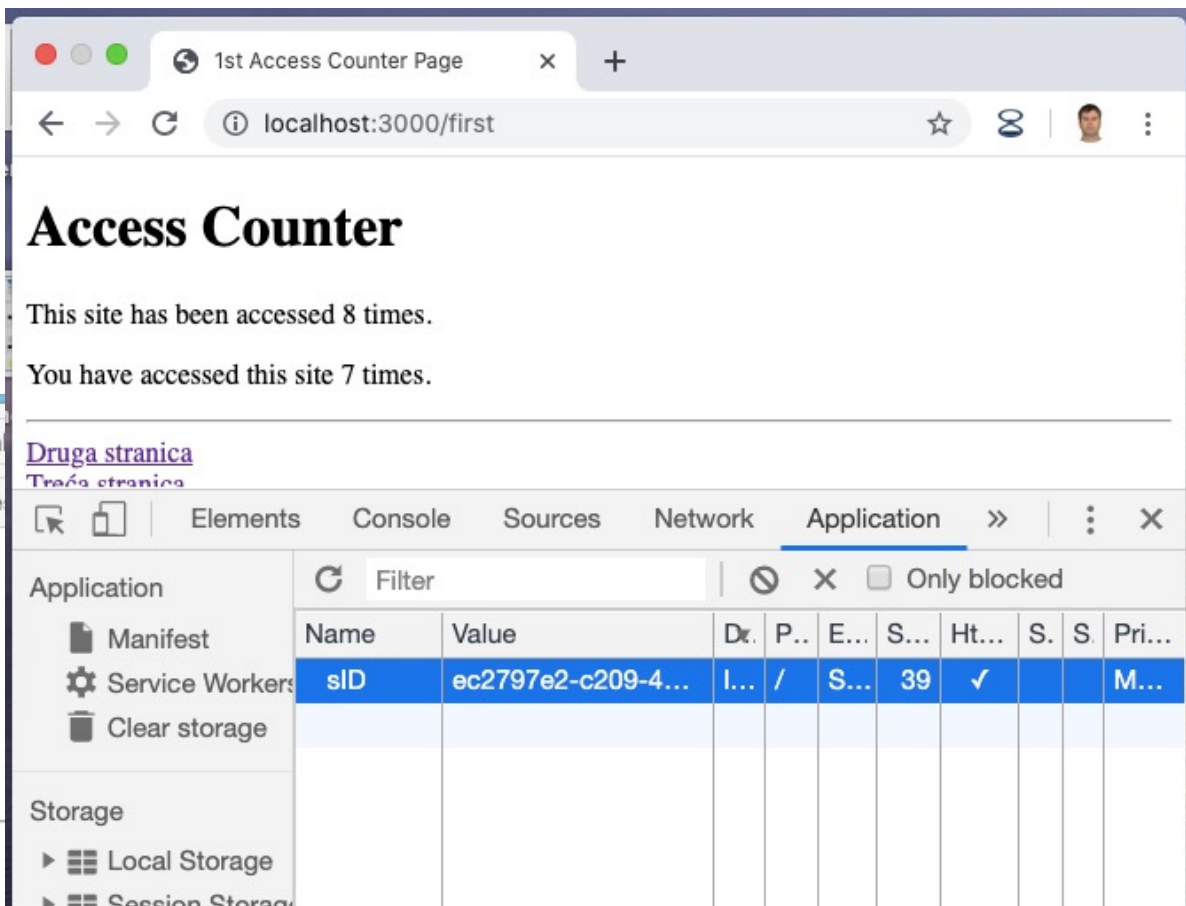
Access Counter

This site has been accessed 11 times.

You have accessed this site 4 times.

[Druga stranica](#)

[Treća stranica](#)



Access Counter

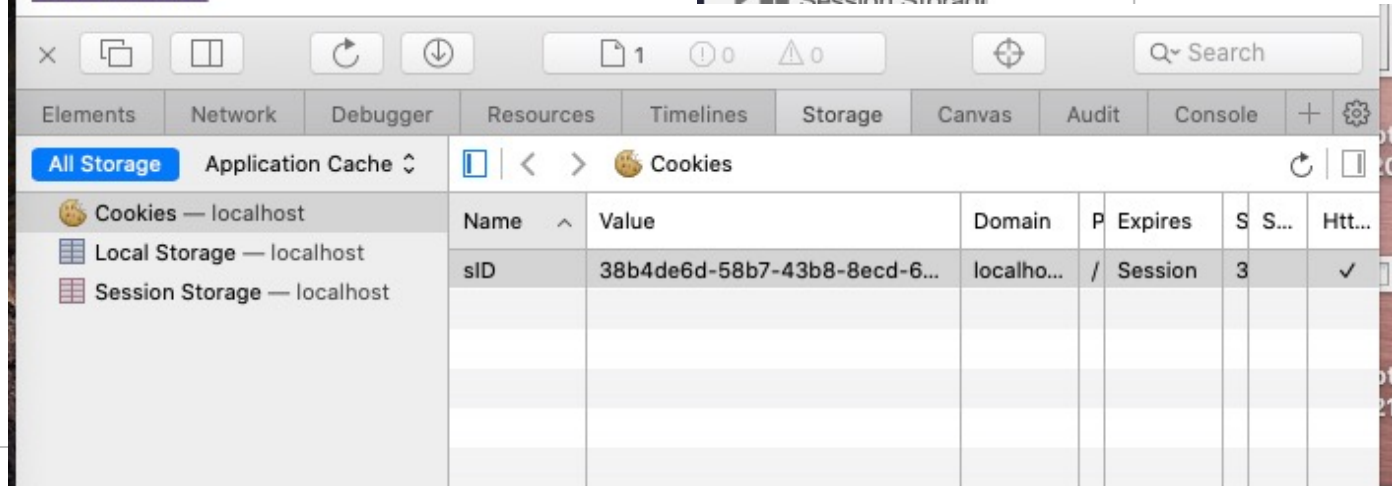
This site has been accessed 8 times.

You have accessed this site 7 times.

[Druga stranica](#)

[Treća stranica](#)

Name	Value	Dr.	P..	E...	S...	Ht...	S.	S.	Pri...
sID	ec2797e2-c209-4...	I...	/	S...	39	✓			M...



Access Counter

This site has been accessed 11 times.

You have accessed this site 4 times.

[Druga stranica](#)

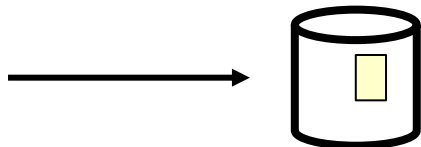
[Treća stranica](#)

Name	Value	Domain	P	Expires	S	S...	Htt...
sID	38b4de6d-58b7-43b8-8ecd-6...	localho...	/	Session	3		✓

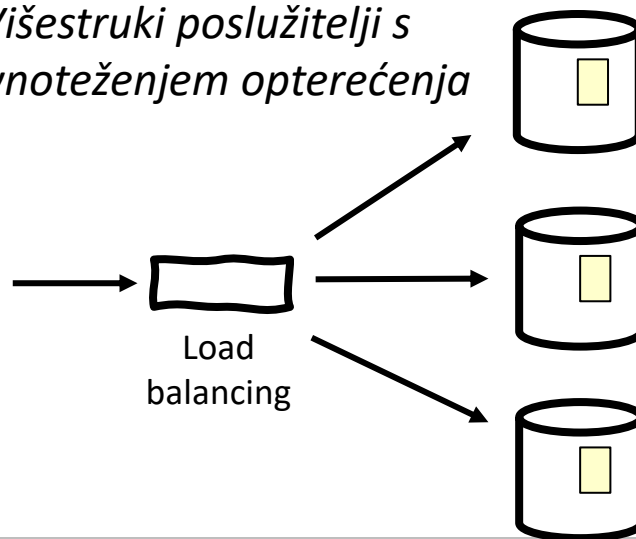
Smještaj tablice sjednica (I)

- **Brzina pristupa** podacima o sjednicama
 - podaci se često čitaju (brzo pretraživanje na osnovu *sessionID* podatka)
 - Podaci se rjeđe upisuju i brišu
- **Trajnost** sjedničkih podataka s obzirom na vijek poslužitelja
 - Privremeni podaci o sjednici, ne moraju nadživjeti „inkarnaciju” poslužitelja
 - Trajni podaci, trebaju biti dostupni kroz više „inkarnacija” poslužitelja
- **Dostupnost** podataka o sjednicama na:
 - Više instanci poslužitelja na istom računalu
 - Više instanci poslužitelja na više računala

Jedan poslužitelj



Višestruki poslužitelji s uravnoteženjem opterećenja



Smještaj tablice sjednica (II)

1. Struktura podataka u radnoj memoriji poslužitelja

- Velika brzina pristupa podacima
- Podaci dostupni samo unutar te instance poslužitelja
- Prestankom rada poslužitelja gube se podaci o aktivnim sjednicama

2. Perzistencija strukture podataka u datotečnom sustavu

- Sporost/opterećenje zbog česte pohrane podataka u datotečni sustav
- Moguće dijeljenje između instanci poslužitelja na istom računalu/lokalnoj mreži (problem konzistencije dijeljenog resursa!)
- Otpornost na prestanke rada poslužitelja

Smještaj tablice sjednica (III)

3. Priručna memorija, usluge

- Brzina (naročito ako je usluga na istom računalu kao i poslužitelj)
- Dijeljenje podataka o sjednicama između više instanci poslužitelja na više računala
- Opcionalna perzistencija podataka
- Primjeri usluga:
 - Redis (<https://redis.io/>)
 - Memcached (<https://memcached.org/>)

4. Baza podataka

- Sporiji pristup podacima o sjednicama (upiti prema bazi podataka)
- Dijeljenje podataka o sjednicama između više instanci poslužitelja na više računala
- Robusnost podataka

Primjer 6 (I)

express

app.js

*express-session
middleware modul*

*Dodatna funkcionalnost pohrane
sjedničkih podataka u datoteku*

*Naš model podataka
(globalni podaci
poslužitelja)*

*Inicijalizira
express-session
middleware*

*Inicijalizira naš model
globalnih podataka
poslužitelja i učitava
podatke iz datoteke
(perzistencija podataka
između pokretanja
poslužitelja)*

```
...  
const session = require('express-session')  
const FileStore = require('session-file-store')(session);  
...  
const globalData = require('./model/GlobalData')  
...  
//session middleware  
app.use(session({  
  secret: 'FER WiM',  
  resave: false,  
  store: new FileStore(),  
  saveUninitialized: true  
}))  
...  
//read the global settings  
app.global = new globalData('./data/global.json');  
app.global.initialize(true);
```

Primjer 6 (II)

express

routes/index.js

```
...  
if(req.session.access_counter === undefined)  
    req.session.access_counter = 0;  
...  
  
//global context  
if(req.app.global.data.access_counter === undefined)  
    req.app.global.data.access_counter = 0  
...  
  
//save global context  
req.app.global.store()  
...
```

Pristup kontekstu
sjednice isti kao i kod
naše implementacije!

*Globalni kontekst
pridružen objektu
express aplikacije –
app.global*

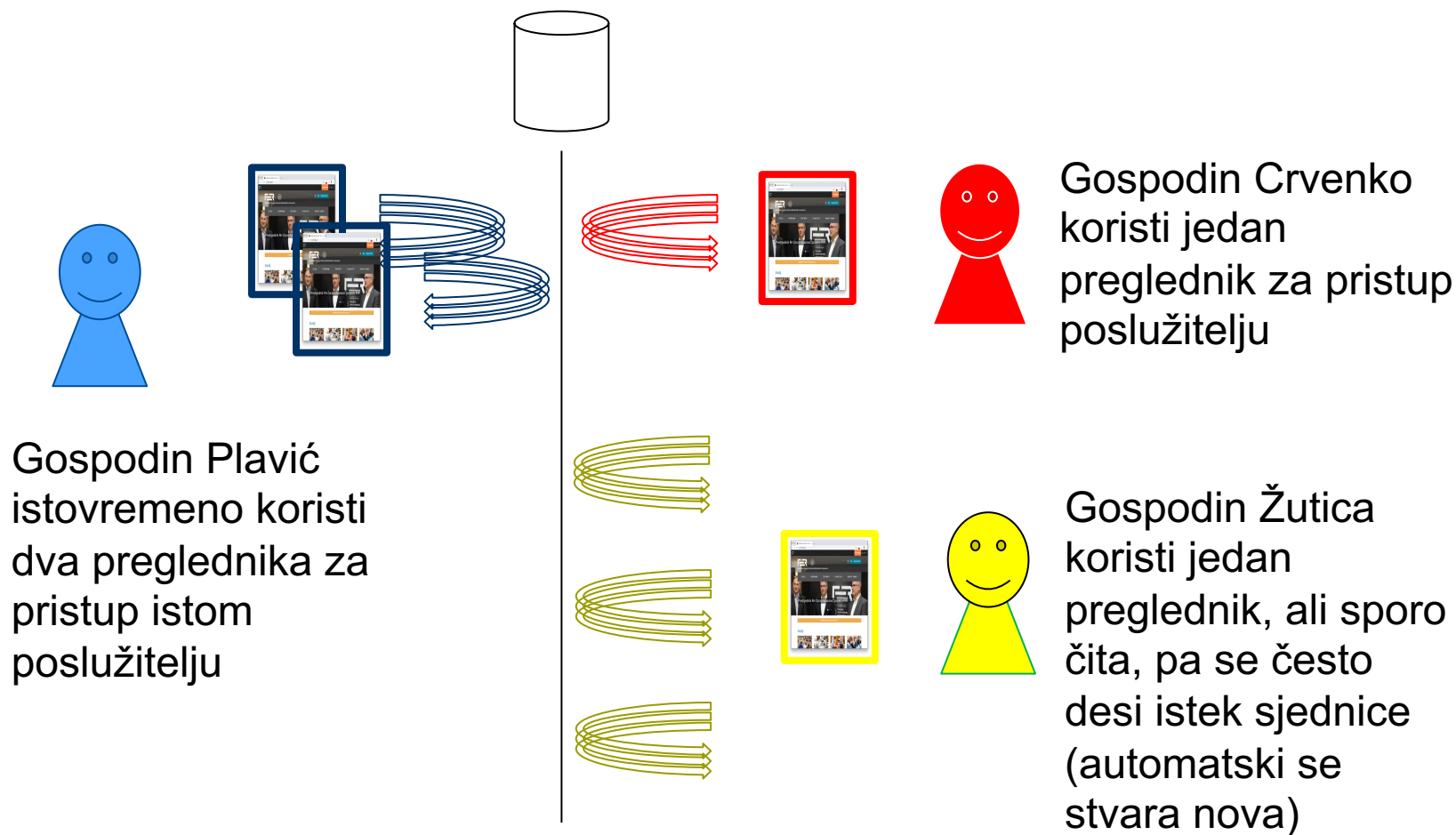
*Nema više globalne
varijable, brojač
posjeta kao dio
modela globalnog
stanja*

*Pohrana promjene
globalnog stanja u
datoteku*

Primjer 6 (III)

- Razlike u odnosu na Primjer 5:
 - (<https://www.npmjs.com/package/express-session>) standardni npm modul *express-session* za podršku sjednicama (temeljen na kolačićima), s podrškom za različite mehanizme pohranu stanja sjednica (<https://www.npmjs.com/package/express-session#compatible-session-stores>)
 - Model globalnog stanja i mehanizam (rudimentarne) pohrane i čitanja stanja iz datoteke u formatu JSON.
- ⇒ I stanje sjednice i globalno stanje su perzistentni – čuvaju se između dvaju slijednih izvođenja poslužitelja
- ⇒ Koja je korist od čuvanja stanja sjednice?
 - ⇒ Koja je korist od čuvanja globalnog stanja?
 - ⇒ I gdje je tu korisnik?

Korisnici i sjednice



- Vidjeli smo kako ostvariti da poslužitelj prepozna različite sjednice
- Ne vrijedi pravilo jedan korisnik = jedna sjednica
- **Kako poslužitelj nedvosmisleno prepozna korisnika?**

Korisnička sjednica (login) (I)

Prijava korisnika u sustav

0: Zahtjev sadrži podatke za autentikaciju korisnika*

HTTP request

2. Stvori asocijaciju između sjednice i korisnika

sessionID	userID
2gy4-54g63yf...	63yf-r65hf-3

HTTP response

1. Provjera autorizacijskih podataka

Session Store

User Store

Obrada zahtjeva

Kontekst korisnika:
stanje (varijable)
vezano uz korisnika

userID	Login	password	email	roles	Context	...
2gy4-54g63yf...	23525343	23527345	205..	r1, r3	account=4, basket={...}	

Korisnička sjednica (login) (II)

■ Provjera prava pristupa korisnika resursu

0: Zahtjev sadrži sjednički kolačić

HTTP request

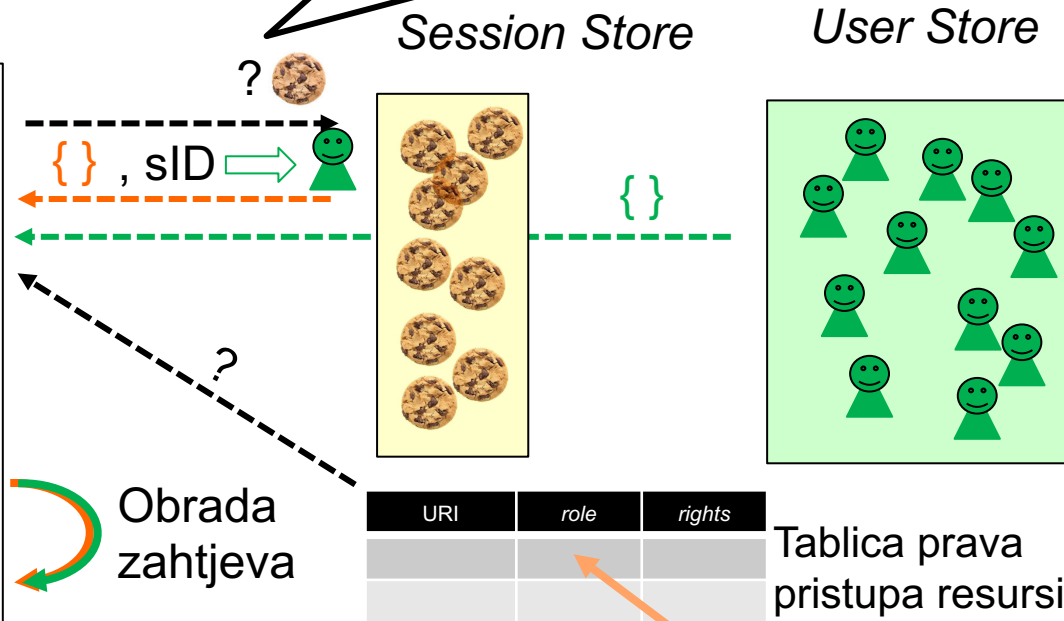
2. Provjera prava pristupa resursu

HTTP response

1. Dohvaćanje sjedničkog i korisničkog konteksta

Session Store

User Store



userID	Login	password	email	roles	Context	...
2gy4-54g63yf...	23525343	23527345	205..		account=4, basket={...}	

Korisnička sjednica (login) (III)

- **Odjava korisnika iz sustava**
 - Postupak istovjetan brisanju sjednice
 - U ovisnosti o implementaciji moguće:
 - Brisanje asocijacije sjednica-korisnik
 - Pohrana podataka korisnika u trajnu memoriju
 - ...

Smještaj podataka o korisniku

- Bitna razlika između podataka o sjednici i korisniku:
 - **Životni vijek** podataka o korisniku daleko dulji
 - **Važnost** podataka o korisniku daleko veća
 - **Količina** podataka o korisniku daleko veća
- Korisnik može istovremeno imati više (neovisnih) interakcija s istim poslužiteljem (više aktivnih sjednica)
- Podaci o korisniku u pravilu pohranjeni u trajnoj memoriji (baza podataka)
- Podaci o trenutno aktivnim korisnicima mogu biti privremeno pohranjeni u bržoj memoriji (radna memorija, usluge priručne memorije), ali se mora paziti na koherenciju između različitih kopija tih podataka

Primjer 7 (I)

express

app.js

```
...  
const globalData = require('./model/GlobalData')  
const userData = require('./model/UserData')  
...  
//read the global settings  
app.global = new globalData('./data/global.json');  
app.global.initialize(true);  
  
//read the user settings  
app.users = new userData('./data/users.json');  
app.users.initialize(true);  
...
```

Naši „modeli”
podataka (globalni i
korisnički)

Inicijalizira naš
„model” globalnih
podataka poslužitelja i
učitava podatke iz
datoteke

Inicijalizira naš
„model” korisničkih
podataka i učitava
podatke iz datoteke

Primjer 7 (II)

express

login.js

```
router.get('/', function(req, res, next) {  
  let loginName = req.cookies.appuserID || ""  
  ...  
});  
...  
//check for credentials  
if( req.app.users.userExists(req.body.email) &&  
req.app.users.getUser(req.body.email).password == req.body.password )  
  
  //if successful, set persistent cookie with username (timeout=1 week)  
  let expiryDate = new Date(Number(new Date()) + 604800000);  
  res.cookie('appuserID', req.body.email, { expires: expiryDate, httpOnly: true});  
  
  //if successful, redirect to the main page  
  req.session.user = req.body.email  
  res.redirect("/")  
}
```

Podatak o korisničkom imenu
dobavlja se iz trajnog kolačića
„appuserID”

Stvaranje
trajnog
kolačića

Povezivanje sjednice
i korisnika

Primjer 7 (III)

logout.js

```
...  
router.get('/', function(req, res, next) {  
  //clear session-user mapping  
  req.session.user == undefined  
  
  //destroy session object  
  req.session.destroy((err) => {  
    ...
```

Brisanje poveznice
sjednice i korisnika

Brisanje sjednice iz
tablice sjednica

Primjer 7 (IV)

express

profile.js

```
...
router.get('/', function(req, res, next) {

  //check if a registered user is trying to access the admin page
  if( req.session.user === undefined) {
    res.redirect("/")
    return
  }

  //render the page with the profile data for the registered user
  res.render('profile', { user: req.app.users.getUser(req.session.user)});
});
```

Registrirani korisnik - mora postojati poveznica sjednice s korisnikom

Dohvat podataka o korisniku (kontekst korisnika) pomoću poveznice sjednice i korisnika

Bitni elementi primjera (I)

- Rute u aplikaciji:
 - /signup – registracija novog korisnika
 - /login – prijava korisnika
 - /logout – odjava korisnika
 - /profile – podaci o trenutnom korisniku
 - / - javna stranica
 - /users – pristup ograničen na prijavljene korisnike (sadrži globalni i brojač posjeta prijavljenog korisnika)
 - /admin – pristup ograničen na prijavljene korisnike s pridjeljenom ulogom „admin”

Bitni elementi primjera (II)

- Perzistentni i tranzijentni podaci
 - Koristimo perzistenciju globalnog stanja i stanja korisnika
 - Ne čuvamo stanja sjednica (ako poslužitelj prestane s radom, svi korisnici će se morati ponovno prijaviti)
- Perzistentni i tranzijentni kolačići
 - Sjednički kolačići tranzijentni (session cookies)
 - Podaci o korisniku (korisničko ime) u trajnom kolačiću, ali s ograničenjem na put (/login)
- Ograničene pristupa resursima na osnovu uloga

Prijedlozi zadataka – nadogradnje primjera 7

- Štićenje zapisa lozinki

```
{"peroperic@fer":{"username":"peroperic@fer","password":"qqqqqqqqq","role":"user",
```

- Middleware za uspostavljanje konteksta korisnika

- `req.app.users.getUser(req.session.user).userspace_count = 2`

- + `req.user.userspace_count = 2`

- Uloge korisnika sustava

- Izbor uloge (user, admin, ...) kod registracije novog korisnika
- Jedan korisnik može imati više uloga
- Upravljanje dozvolama po pojedinom resursu temeljeno na ulogama (samo admin uloge smiju mijenjati dozvole)

- Pohrana stanja korisnika i globalnog stanja u bazu podataka (postgresql) ili priručnu memoriju (Redis)

Pitanja?

