



Osnove web programiranja

HTTP i mehanizam sesije

Termin 5

Sadržaj

1. HTTP i mehanizam sesije
2. Praćenje sesije klijenta
3. Čuvanje objekata u Springu
4. Case study – primena sesije u bioskop aplikaciji
 1. prikaz liste posećenih filmova za Klijenta
5. Spring sesija, alternativni načini implementacije sesijse
6. Inicijalizacija objekata sesije korišćenjem Listener
7. Inicijalizacija objekata ServletContex
8. Inicijalizacija bean objekta

Dodatno:

HTTP i mehanizam sesije

HTTP komunikacija

HTTP je stateless protokol koji ne zateva od servera čuvanje statusa klijenta ili korisničke sesije klijenta tj. stanja klijenta proizašlog iz niza zahteva upućenih od strane istog klijenta

- HTTP serveri prevazilaze prethodno tako što implementiraju različite metode za održavanje i upravljanje sesijom, tipično se oslanjajući na jedinstveni identifikator *cookie* ili neki drugi parametar koji omogućava praćenje zahteva koji originiraju od istog klijenta (npr. URL Rewriting mehanizam), kreirajući stateful protokol iznad HTTP protokola.

HTTP i mehanizam sesije

Praćenje sesije klijenta

HTTP protokol ne prati sesiju – komunikacija klijenta i servera se ne prati po isporuci resursa.

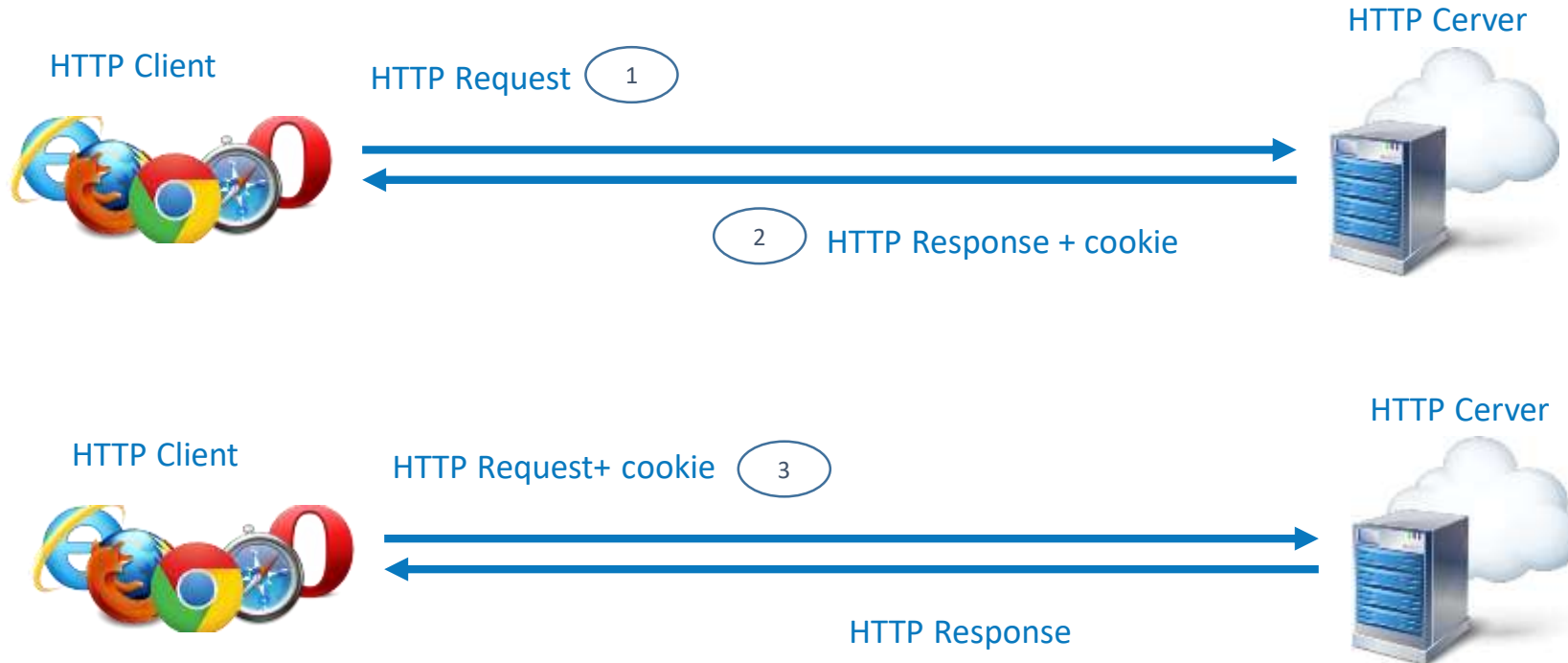
- Koristi se *cookie* mehanizam:
 - Server šalje *cookie* klijentu u okviru http response
 - klijent čuva primljeni *cookie* i šalje ga uz svaki http request.
- Ako navigator ne prihvata *cookie-je*, koristi se URL Rewriting mehanizam:

```
<a href="http://www.myserver.com/catalog/index.html;jsessionid=1234"> link </a>
```

HTTP i mehanizam sesije

Praćenje sesije klijenta

- Koristi se *cookie* mehanizam



HTTP i mehanizam sesije

Praćenje sesije klijenta

1

▼ Request Headers view parsed

GET /WebProgOrg/LogIn?username=pera&password=peric HTTP/1.1

Host: localhost:8080

Connection: keep-alive

Upgrade-Insecure-Requests: 1

DNT: 1

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9

Sec-Fetch-Site: same-origin

Sec-Fetch-Mode: navigate

Sec-Fetch-User: ?1

Sec-Fetch-Dest: document

Referer: http://localhost:8080/WebProgOrg/login.html

Accept-Encoding: gzip, deflate, br

Accept-Language: en-US,en;q=0.9

HTTP i mehanizam sesije

Praćenje sesije klijenta

2

▼ Response Headers [view parsed](#)

HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Set-Cookie: JSESSIONID=568231692A3653D3FC23C2E1DA4C7C11; Path=/WebProgOrg

Content-Type: text/html

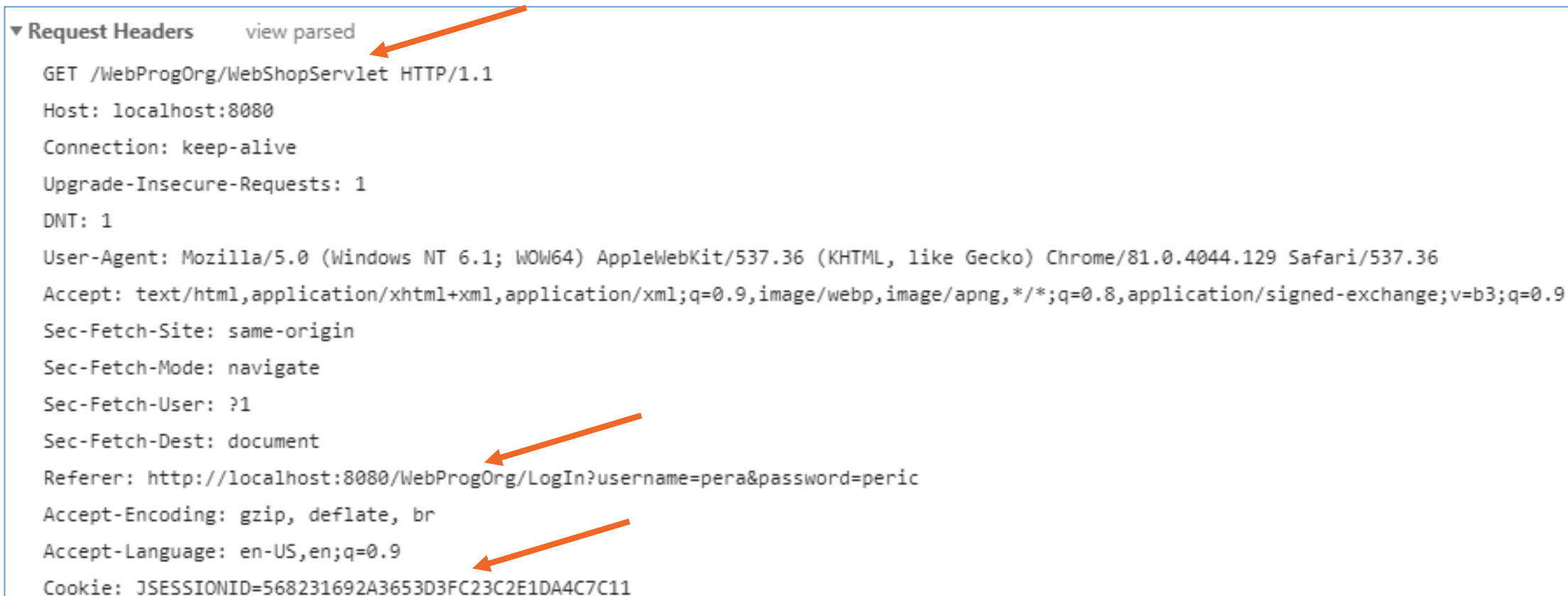
Content-Length: 140

Date: Thu, 21 May 2020 12:46:34 GMT

HTTP i mehanizam sesije

Praćenje sesije klijenta

3



```
▼ Request Headers  view parsed
GET /WebProgOrg/WebShopServlet HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Upgrade-Insecure-Requests: 1
DNT: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.129 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://localhost:8080/WebProgOrg/LogIn?username=pera&password=peric
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=568231692A3653D3FC23C2E1DA4C7C11
```

The screenshot displays the 'Request Headers' section of a browser's developer tools. Three orange arrows highlight specific headers: the first points to 'view parsed', the second points to the 'Referer' header, and the third points to the 'Cookie' header.

HTTP i mehanizam sesije

HTTP Response + cookie

- Vraća se u atributu Set-Cookie
- Opcioni atributi
 - domain – domen u kome važi cookie
 - path – za koje URL-ove na sajtu važi
 - expires – datum isticanja
- Kada ističe cookie?
 - eksplicitan momenat (expires)
 - nedefinisano – kada se ugasi browser

HTTP i mehanizam sesije

HTTP Response + cookie

2

▼ Response Headers

```
alt-svc: h3-27=":443"; ma=2592000,h3-25=":443"; ma=2592000,h3-T050=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q049=":443"; ma=2592000,h3-Q048=":443"; ma=2592000,h3-Q046=":443"; ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
cache-control: private, max-age=0
content-encoding: br
content-length: 61103
content-type: text/html; charset=UTF-8
date: Thu, 21 May 2020 13:07:20 GMT
expires: -1
p3p: CP="This is not a P3P policy! See g.co/p3phelp for more info."
server: gws
set-cookie: 1P_JAR=2020-05-21-13; expires=Sat, 20-Jun-2020 13:07:20 GMT; path=/; domain=.google.com; Secure; SameSite=None
set-cookie: NID=204=yAKOgUfmf46dB_m_aOxpquiMYIIN-rVQivAag7VUs1-c5j33iL6zTWbALz72nd2piVDLPgn8se0JFducxp4Q7kgN7ZF7Q2Igr5e7EqAq236fA5xa1ROPdGmcRmpyTkpcFfqop7t43eGmpOm-GoKYCYSRgmoTQCYvK3Q_b7GVMY; expires=Fri, 20-Nov-2020 13:07:20 GMT; path=/; domain=.google.com; Secure; HttpOnly; SameSite=None
status: 200
```



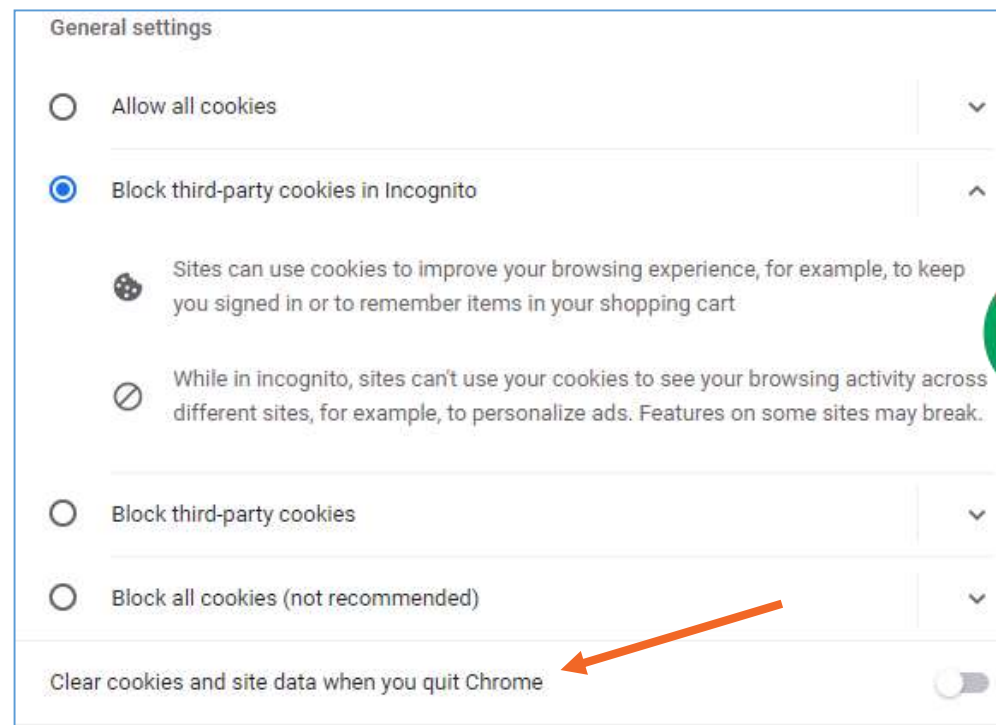
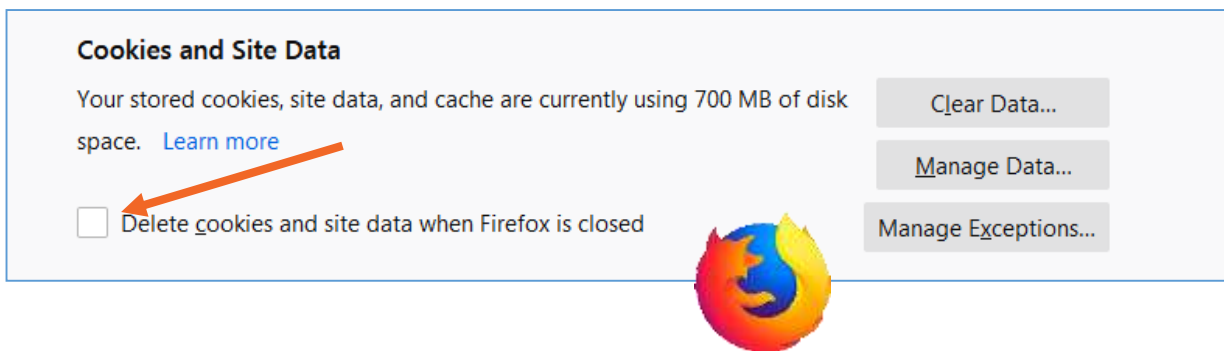
Response Cookies

Name	Value	Domain	Path	Expires / Ma...	Size	HttpOnly	Secure	SameSite	Priority
1P_JAR	2020-05-21-13	.google.com	/	2020-06-20...	111		✓	none	Medium
NID	204=yAKOgUfmf46dB_m_aOxpquiMYIIN-rVQivAag7VUs1-c5j33iL6zTWb...	.google.com	/	2020-11-20...	279	✓	✓	none	Medium

HTTP i mehanizam sesije

Brauzeri

- Svaki brauzer ima svoje kukije i ne deli ih sa drugim brauzerima
 - za svaki sajt postoji poseban kuki
- Tabovi i stranice u brauzerima dele kukije



HTTP i mehanizam sesije

Kako očistiti određene cookie iz brauzera

Delete specific cookies

1. On your computer, open **Chrome**.
2. At the top right, click More. Settings.
3. Under 'Privacy and security', click **Cookies** and other site data.
4. Click See all **cookies** and site data.
5. At the top right, search for the website's name.
6. To the right of the site, click Remove .



1. Click **Firefox** button or Tools menu, then click Options.
2. Select Privacy panel.
3. Set **Firefox** will: to "Use custom settings for history"
4. Click Show **Cookies**.
5. In the search field, type the name of the site whose **cookies** you want to remove.
6. Select the **cookie** in the list to remove and click "Remove **Cookie**"
7. Click "Close"



HTTP i mehanizam sesije

Alternativan način čuvanje statusa klijenta

- Ako navigator ne prihvata cookie-je, koristi se URL Rewriting mehanizam
- u hiperlink () koji "gađa" naš server ugradimo id sesije:

- Da li uvek možemo da koristimo ovu tehniku?
 - Exposing session information in the URL is a growing [security risk](#) (from place 7 in 2007 to place 2 in 2013 on the OWASP Top 10 List)
- Gde se stavlja id sesije u formi?
 - u hidden polje

Praćenje sesije klijenta

Praćenje sesije klijenta

- Za sesiju se definiše period neaktivnosti
 - Ako klijent više od navedenog perioda nije komunicirao sa serverom sesija za klijenta se automatski poništava
- Session inactivity – obično 30 minuta, ali može da se podesi

Praćenje sesije klijenta

Praćenje sesije klijenta

- Session inactivity – obično 30 minuta, ali može da se podese na različitim nivoima
 - nivo veb servera - u /conf/web.xml fajlu za Apache Tomcat i odnosi se na minute

```
<session-config>  
  <session-timeout>30</session-timeout>  
</session-config>
```
 - nivo aplikacije - u /WEB-INF/web.xml za aplikaciju i odnosi se na minute

```
<session-config>  
  <session-timeout>30</session-timeout>  
</session-config>
```
 - nivo aplikacije - u application.properties fajlu za aplikaciju, na dva načina, odnosi se na minute ili sekunde

```
server.servlet.session.timeout=30  
spring.session.timeout.seconds=180
```
 - nivo koda - u kodu kontrolera kada se kreira sesija za određenog klijenta i odnosi se na sekunde

```
session.setMaxInactiveInterval(180);
```

Praćenje sesije klijenta

Klasa HttpSession

- Reprezentuje sesiju
- Čuva cookie ili ID sesije za URL redirection
 - metoda getId()
- Čuva objekte vezane za sesiju
 - metode getAttribute(ime), setAttribute(ime, objekat), removeAttribute(ime)
- Invalidira sesiju i razvezuje sve objekte vezane za nju
 - metoda invalidate()
- podešava period neaktivnosti
 - metoda setMaxInactiveInterval(sekunde)
- **NAPOMENA:** ograničenje je da se HTTP sesija čuva u memoriji servera
 - Posledice?

Praćenje sesije klijenta

URL Rewriting mehanizam

- Ako navigator ne prihvata cookie-je, koristi se URL Rewriting mehanizam
- u hiperlink () koji "gađa" naš server ugradimo id sesije:

- **HttpServletResponse response.encodeURL()** metoda
 - response.encodeURL("http://www.myserver.com/catalog/index.html")
- **HttpServletResponse response.encodeRedirectURL()** metoda

Praćenje sesije klijenta

Praćenje sesije klijenta iz kontrolera

- Praćenje sesije se svodi na pribavljanje HttpSession objekta i kreiranje objekata koji se vezuju na sesiju
 - objekat bi trebalo da je serijalizabilan
- Kada korisnik prozove neki kontroler, u okviru njega se koristi objekat vezan za sesiju:
 - ako do tada nije postojao objekat, on se kreira i veže za sesiju;
 - ako postoji, koristi se.
- **NAPOMENA:** Voditi računa da se podaci o sesijama za sve klijenate nalaze u memoriji i da će ti podaci nestati po prekidu rada aplikacije

```
List<Film> poseceniFilmovi = (List<Film>) request.getSession().getAttribute("PoseceniFilmoviZaKorisnika");
if (poseceniFilmovi == null) {
    poseceniFilmovi = new ArrayList<>();
    request.getSession().setAttribute("PoseceniFilmoviZaKorisnika", poseceniFilmovi);
}
```

Praćenje sesije klijenta

Praćenje sesije klijenta iz kontrolera

```
retVal+= "<table class=\"horizontalni-meni\">\r\n" +
        "  <caption>Posećeni filmovi</caption>\r\n" +
        "  <tr>\r\n" +
        "    <td>\r\n" +
        "      <ul>\r\n";
if(poseceniFilmovi.isEmpty()) {
    retVal+="      <li>Nema posećenih filmova</li>\r\n";
}
for (int i=0; i < poseceniFilmovi.size(); i++) {
    retVal+="      <li><a href=\"/Filmovi/Details?id="+poseceniFilmovi.get(i).getId()
        +"\">"+poseceniFilmovi.get(i).getNaziv()+ "</a></li>\r\n";
}
retVal+= "    </ul>\r\n" +
        "  </td>\r\n" +
        " </tr>\r\n" +
        "</table>\r\n" +
```

Praćenje sesije klijenta

Pribavljanje HttpSession objekta

- Može na više načina oslanjajući se na
 - parametar metode tipa HttpSession, pri pozivanju metode Spring automatski kreira sesiju za klijenta ili preuzima postojeću

```
@GetMapping(value="/Details")
@ResponseBody
public String Details(@RequestParam Long id, HttpSession session) {
    List<Film> poseceniFilmovi = session.getAttribute("PoseceniFilmoviZaKorisnika");
    if (poseceniFilmovi == null) {
        poseceniFilmovi = new ArrayList<>();
        request.getSession().setAttribute("PoseceniFilmoviZaKorisnika", poseceniFilmovi);
    }
}
```

Praćenje sesije klijenta

Pribavljanje HttpSession objekta

- Može na više načina oslanjajući se na
 - parametar metode tipa HttpServletRequest, iz request objekta se sessija kreira ili preuzima postojeća pozivanjem metode getSession()

```
@GetMapping(value="/Details")
@ResponseBody
public String Details(@RequestParam Long id, HttpServletRequest request) {
    List<Film> poseceniFilmovi = (List<Film>) request.getSession().getAttribute("PoseceniFilmoviZaKorisnika");
    if (poseceniFilmovi == null) {
        poseceniFilmovi = new ArrayList<>();
        request.getSession().setAttribute("PoseceniFilmoviZaKorisnika", poseceniFilmovi);
    }
}
```

Praćenje sesije klijenta

Pribavljanje HttpSession objekta

- Postavlja se pitanje može li se do sesije doći oslanjajući se na Dependency Injection i `@Autowired` anotaciju?
- Podsećane:
 - Spring kontejner je zadužen za kreiranje, inicijalizaciju, konfigurisanje i obezbeđivanje objekata dostupnim koji su anotirani sa `@Autowired`. Prethodno je iskorišćeno u ranijim primerima za pribavljanje ServletContext objekta, kada trebalo pribaviti vrednosti za sve filmove koji se čuvaju u memoriji na nivou aplikacije.
 - Objekti anotirani sa `@Autowired` injektuju po kreiranju bean objekta za kontroler, pre poziva bilo koje od Handler metoda za kontroler.

```
@Controller
@RequestMapping(value="/Filmovi")
public class FilmoviController {
```

```
    @Autowired
    HttpSession session;
```

Praćenje sesije klijenta

Pribavljanje HttpSession objekta

- Postavlja se pitanje da je razumno da se objekat sesije skladišti kao atribut klase kontrolera? Da bi bio dostupan za klijenta ako poziva ostale metode

```
@Controller
@RequestMapping(value="/Filmovi")
public class FilmoviController {
    HttpSession session;

    @GetMapping(value="/Details")
    @ResponseBody
    public String Details(@RequestParam Long id, HttpServletRequest request) {
        if(session == null)
            session=request.getSession();
        List<Film> poseceniFilmovi = session.getAttribute("PoseceniFilmoviZaKorisnika");
        ...
    }
    @GetMapping(value="/Create")
    @ResponseBody
    public String Create(HttpServletRequest request) {
        if(session == null)
            session=request.getSession();
        List<Film> poseceniFilmovi = session.getAttribute("PoseceniFilmoviZaKorisnika");
        ...
    }
    ...
}
```

Čuvanje objekata u Springu

Na šta se objekat može vezati?

- Na request:

```
request.setAttribute("ime", referenca);
```

- Na sesiju:

```
request.getSession().setAttribute("ime", referenca);
```

- Na aplikaciju kada se objekti ubacuju u ServletContext:

```
servletContext.setAttribute("ime", referenca);
```

- Na aplikaciju putem bean objekta koji se zadaje kao deo konfiguracije u okviru ApplicationContext:

```
@Configuration
public class SecondConfiguration {

    @Bean(name= {"memorijaAplikacije"}, initMethod="init", destroyMethod="destroy")
    public MemorijaAplikacije getMemorijaAplikacije() {
        return new MemorijaAplikacije();
    }

    public class MemorijaAplikacije extends HashMap {... }

}
```


Case study – primena sesije u bioskop aplikaciji

Import projekta

- Aplikacija ima za cilj da omogući rad online bioskopa
- Trenutna implementacija omogućuje CRUD operacije sa entitetom film i korisnik
- Aplikacija teba da omogući prikaz statistike za posećenost filmova na nivou aplikacije
- **Aplikacija teba da omogući personalizovani prikaz posećenih filmova za različite klijente**
 - Na svakoj HTML stanici omogućiti prikaz posećenih filmova
 - Prilikom pregleda detalja za određeni film dodati film u spisak posećenih filmova za klijenta

FilmController i rad sa sesijom

Case study – primena sesije u bioskop aplikaciji

Inicijalni prikaz filmova



The screenshot shows a web browser at the URL `localhost:8080/BioskopVebAplikacija/Filmovi`. The page has a navigation menu with links for `filmovi` and `projekcije`. The main content area is titled "Filmovi" and contains a table with three columns: "r. br.", "naziv", and "trajanje". The table lists three movies: "The Shining" (146 minutes), "One Flew Over the Cuckoo's Nest" (133 minutes), and "The Little Shop of Horrors" (72 minutes). Each row has a "projekcije" link and a "Projekcije" button. Below the table, there is a link for "dodavanje filma" and a section titled "Posećeni filmovi" which displays "Nema posećenih filmova".

r. br.	naziv	trajanje	projekcije
1	The Shining	146	projekcije Projekcije
2	One Flew Over the Cuckoo's Nest	133	projekcije Projekcije
3	The Little Shop of Horrors	72	projekcije Projekcije

- Pogledajte kod kontrolera `FilmoviController`
- U `init()` metodi se ne radi sa sesijom već samo u handler metodama
- Pogledajte kod handler metode `index()`, deo koji radi sa sesijom sesije
- Otvoriti stranicu za pregled filmova u dva različita brauzera
- <http://localhost:8080/BioskopVebAplikacija/Filmovi>

Case study – primena sesije u bioskop aplikaciji

Prikaz detalja za film

The screenshot shows a web browser at the URL `localhost:8080/BioskopVebAplikacija/Filmovi/Details`. The page has a navigation menu with links for [filmovi](#) and [projekcije](#). The main content area is titled "Film" and contains a form with the following fields:

naziv:	One Flew Over the Cuckoo's Nest
trajanje:	133
	<input type="button" value="Izmeni"/>

Below the form is a blue button and an button.

The page also features two sections:

- Popularni filmovi**: A list containing "One Flew Over the Cuckoo's Nest" with a green progress bar and the number "1".
- Posećeni filmovi**: A list containing "One Flew Over the Cuckoo's Nest".

- Koristi aplikaciju, pokreni više puta pregled detalja za određeni film da bi se podaci dodali u sesiju
- U dva brauzera otvori developer tools, network kartica,
 - Da li je cookie vredost ista?
- Svaki browser ima svoju memoriju za sesiju

Case study – primena sesije u bioskop aplikaciji

Prikaz filmova posle dužeg korišćenja

The screenshot displays a web browser window at localhost:8080/BioskopVebAplikacija/Filmovi. It features three main sections:

- Filmovi**: A table listing movies with columns for ID, Title, Duration, and Projections.
- dodavanje filma**: A section for adding new movies.
- Popularni filmovi**: A section for popular movies with progress bars indicating their status.
- Posećeni filmovi**: A section for visited movies with progress bars.

r. br.	naziv	trajanje	projekcije
1	The Shining	146	projekcije Projekcije
2	One Flew Over the Cuckoo's Nest	133	projekcije Projekcije
3	The Little Shop of Horrors	72	projekcije Projekcije

dodavanje filma

Popularni filmovi

The Little Shop of Horrors	One Flew Over the Cuckoo's Nest
<div style="width: 50%;"></div> 5	<div style="width: 20%;"></div> 2

Posećeni filmovi

One Flew Over the Cuckoo's Nest	The Little Shop of Horrors
<div style="width: 100%;"></div>	<div style="width: 100%;"></div>

Case study – primena sesije u bioskop aplikaciji

Prikaz filmova za drugog klijenta

← → ↺ 🏠 localhost:8080/BioskopVebAplikacija/Filmovi

- [filmovi](#)
- [projekcije](#)

Filmovi

r. br.	naziv	trajanje	projekcije
1	The Shining	146	projekcije Projekcije
2	One Flew Over the Cuckoo's Nest	133	projekcije Projekcije
3	The Little Shop of Horrors	72	projekcije Projekcije

- [dodavanje filma](#)

Popularni filmovi

The Little Shop of Horrors <div><div></div>5</div>	One Flew Over the Cuckoo's Nest <div><div></div>2</div>
---	--

Posećeni filmovi

Nema posećenih filmova

- Primetite da svaka metoda koja radi sa sesijom ima isti kod na početku, a to je da poverava da li postoji lista posećenih filmova u sesiji, ako se lista ne nalazi u sesiji tada se ona kreira (ponavlja se identičan kod)
 - Da li je to dobro?
 - Kako bi to rešili?

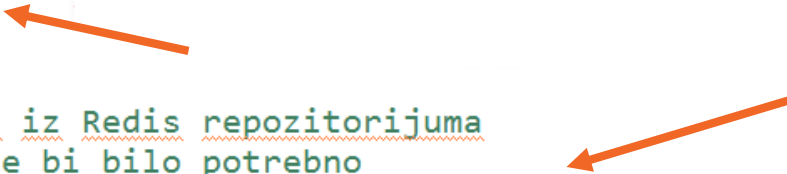
Case study – primena sesije u bioskop aplikaciji

Spring session

BioskopVebAplikacija, FilmoviController i details

@ResponseBody

```
public String details(@RequestParam Long id, HttpServletRequest request) {  
    //preuzimanje vrednosti iz konteksta  
    Filmovi filmovi = (Filmovi)servletContext.getAttribute(FilmoviController.FILMOVI_KEY);  
    Filmovi filmovi = (Filmovi)memorijaAplikacije.get(FilmoviController.FILMOVI_KEY);  
  
    FilmStatistika statistikaFilмова = (FilmStatistika)servletContext.getAttribute(FilmoviController.STATISTIKA_FILMOVI);  
    FilmStatistika statistikaFilмова = (FilmStatistika)memorijaAplikacije.get(FilmoviController.STATISTIKA_FILMOVI);  
  
    //preuzimanje vrednosti iz sesije za klijenta  
    @SuppressWarnings("unchecked")  
    List<Film> poseceniFilmovi = (List<Film>) request.getSession().getAttribute(FilmoviController.POSECENI_FILMOVI);  
    if (poseceniFilmovi == null) {  
        poseceniFilmovi = new ArrayList<>();  
        request.getSession().setAttribute(FilmoviController.POSECENI_FILMOVI_ZA_KORISNIKA_KEY, poseceniFilmovi);  
    }  
  
    Film film = filmovi.findOne(id);  
    if(film!=null) {  
        statistikaFilмова.incrementBrojac(film);  
        if(!poseceniFilmovi.contains(film))  
            poseceniFilmovi.add(film);  
    }  
  
    //mora jer se sesija svaki put cita iz Redis repozitorijuma  
    //da je samo in memory sesija ovo ne bi bilo potrebno  
    request.getSession().setAttribute(FilmoviController.POSECENI_FILMOVI_ZA_KORISNIKA_KEY, poseceniFilmovi);  
}
```



Case study – primena sesije u bioskop aplikaciji

Spring session

- Korišćenje @SessionAttributes anotacije za handler metode
- Više o tome na <https://www.baeldung.com/spring-mvc-session-attributes>

Spring sesija, alternativni načini implementacije sesije

Spring session

- Pri najjednostavijem rukovanju sesije kada se koristi HttpSession i prepušta se veb kontejneru rukovanje sesijom, tada postoji ograničenje je da se HTTP sesija čuva u memoriji servera.
- Posledice su da se sa prestankom rada servera svi podaci o sesijama izgube
- Postavlja se pitanje kako je moguće čuvati podatke o sesiji klijenta tako da oni budu dostupni i ako server prestane sa radom
- Za tu svrhu koristi se Spring session mehanizam rukovanja sesijom
- Spring Session ima jednostavan cilj da se oslobodi ograničenja da se HTTP sesije čuvaju na serveru.
 - Takođe omogućava deljenje podataka o sesiji između različitih servisa koji se nalaze u oblaku bez vezanosti za jedan kontejner (tj. Tomcat)
 - Podržava višestruke sesije u istom brauzeru i slanje sesije u HTTP zaglavlju

Spring sesija, alternativni načini implementacije sesijse

Spring session

Spring Session uključuje sledeće module:

- Spring Session Core - Spring Session core APIs - obavezan
- Spring Session Data Redis – pruža repozitorijum za skladištenje sesije i upravljanje sesijom koja se oslanja na Redis bazu podataka
- Spring Session JDBC - pruža repozitorijum za skladištenje sesije i upravljanje sesijom koje se oslanja relacionu bazu podataka kao što je MySQL
- Spring Session Hazelcast - pruža repozitorijum za skladištenje sesije i Hazelcast upravljanje sesijom
- Više informacija na <https://spring.io/projects/spring-session>
- Tutorijal kako se Spring Session koristi na
<https://www.techgeeknext.com/spring-boot/spring-boot-session-management>
<https://www.baeldung.com/spring-session>
- https://www.javainuse.com/spring/springboot_session_redis

Spring sesija, alternativni načini implementacije sesijse

Spring session

Spring Session uključuje sledeće module:

- Spring Session Core - Spring Session core APIs - obavezan

```
<!-- Add dependencies for using Spring Session -->
<dependency>
    <groupId>org.springframework.session</groupId>
    <artifactId>spring-session-core</artifactId>
</dependency>
```

pom.xml

Spring sesija, alternativni načini implementacije sesijse

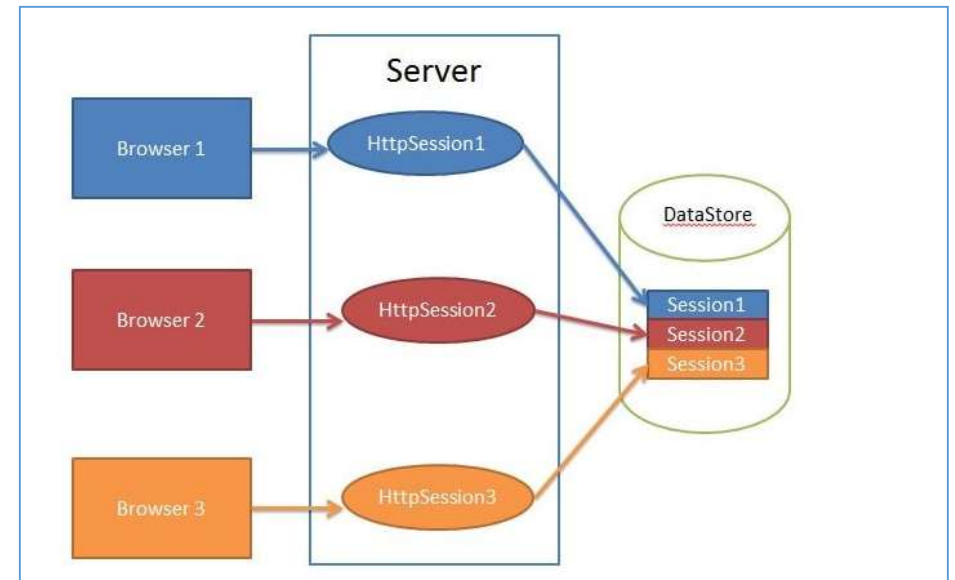
Spring session

Spring Session sa JDBC uključuje sledeće module:

- Spring Session Core - Spring Session core APIs - obavezan
- Spring Session JDBC - pruža repozitorijum za skladištenje sesije i upravljanje sesijom koje se oslanja relacionu bazu podataka kao što je MySQL

```
<!-- Add dependencies for using Spring Session -->  
<dependency>  
    <groupId>org.springframework.session</groupId>  
    <artifactId>spring-session-jdbc</artifactId>  
</dependency>
```

pom.xml



Spring sesija, alternativni načini implementacije sesijse

Spring session

Spring Session sa Redis uključuje sledeće module:

- Spring Session Core - Spring Session core APIs - obavezan
- Spring Session Data Redis – pruža repozitorijum za skladištenje sesije i upravljanje sesijom koja se oslanja na Redis bazu podataka
- Neophodno da je podignut Redis server baze i opciono Redis klijent za praćenje
- Svaka izmena nad objektima u sesiji iziskuje ponovno ubacivanje istih u sesiju

```
<!-- Provides session repository for Redis database session management -->
```

```
<dependency>
```

```
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-data-redis</artifactId>
```

```
</dependency>
```

```
<dependency>
```

```
    <groupId>org.springframework.session</groupId>
```

```
    <artifactId>spring-session-data-redis</artifactId>
```

```
</dependency>
```

pom.xml

PrimerSpringSesijeSaRedisRepozitorijumom

application.properties

spring.session.store-type=redis

spring.redis.host=localhost

spring.redis.port=6379

Inicijalizacija objekata sesije korišćenjem Listener

Osluškiivač za događaje kreiranja i brisanja sesije

- Moguće je prilikom svakog kreiranja objekta sesije uraditi i njenu inicijalizaciju sa određenim vrednostima
- Implementacijom interfejsa HttpSessionListener i redefinisanjem metode
 - sessionCreated moguće je definisati kod koji će se izvršiti nakon kreiranja sesije,
 - a redefinisanje metode sessionDestroyed moguće je definisati kod koji će se izvršiti nakon uništavanja sesije

```
@Component
public class InitHttpSessionListener implements HttpSessionListener {

    /** kod koji se izvrsava po kreiranju sesije */
    public void sessionCreated(HttpSessionEvent arg0) {}

    /** kod koji se izvrsava po brisanju sesije */
    public void sessionDestroyed(HttpSessionEvent arg0) {}

}
```

Inicijalizacija objekata sesije korišćenjem Listener

Registrovanje osluškivača

- Neohodno je da klasa koja implementira Interfejs bude označena sa anotacijom `@Component`, a Spring će prepoznati osluškivač prilikom skeniranje svih bean klasa.
- Ukoliko se ne koristi anotacija `@Component`, osluškivač se može registrovati na nivou aplikacije - u `/WEB-INF/web.xml`

```
<listener>  
  <listener-class>com.ftn.PrviMavenVebProjekat.listeners.InitHttpSessionListener</listener-class>  
</listener>
```

Inicijalizacija objekata sesije korišćenjem Listener

Implementacija koda

InitHttpSessionListener

@Component

```
public class InitHttpSessionListener implements HttpSessionListener {

    /** kod koji se izvrsava po kreiranju sesije */
    public void sessionCreated(HttpSessionEvent arg0) {
        System.out.println("Inicijalizacija sesisje HttpSessionListener...");

        //pri kreiranju sesije inicijalizujemo je ili radimo neke dodatne aktivnosti
        List<Film> poseceniFilmovi = new ArrayList<Film>();
        HttpSession session = arg0.getSession();
        System.out.println("session id korisnika je "+session.getId());
        session.setAttribute(FilmoviController.POSECENI_FILMOVI_ZA_KORISNIKA_KEY,
poseceniFilmovi);

        System.out.println("Uspeh!");
    }
}
```

- Da li je potreban kod za proveru i kreiranja liste posećenih filmova u svakoj od handler metoda za kontroler FilmoviController?

Inicijalizacija objekata ServletContext

Korišćenjem Initializer

InitServletContextInitializer

- Moguće je po prokretanju aplikacije izvršiti inicijalizaciju ServletContext objekta
- Implementacijom interfejsa ServletContextInitializer i redefinisanjem metode onStartup moguće je definisati kod koji će se izvršiti nakon pokretanja aplikacije kada je ServletContext kreiran

@Component

```
public final class InitServletContextInitializer implements ServletContextInitializer {
```

@Override

```
public void onStartup(ServletContext servletContext) throws ServletException {  
    System.out.println("Inicijalizacija konteksta pri ServletContextInitializer...");  
    servletContext.setAttribute(FilmoviController.FILMOVI_KEY, new Filmovi());  
    servletContext.setAttribute(FilmoviController.STATISTIKA_FILMOVA_KEY, new FilmStatistika());  
    System.out.println("Uspeh!");  
}
```

```
}
```


Inicijalizacija objekata ServletContext

Korišćenjem Listener

InitServletContextListener

- Moguće je nakon kreiranju ServletContext objekta i njegove inicijalizacije sa Initializer izvršiti kod Listener u kome se dodatno može upravljati ServletContext
- Implementacijom interfejsa ServletContextListener i redefinisanjem metode
 - contextInitialized moguće je definisati kod koji će se izvršiti nakon inicijalizacije ServletContext objekta,
 - a redefinisanje metode contextDestroyed moguće je definisati kod koji će se izvršiti nakon uništavanja ServletContext

```
@Component
public class InitServletContextListener implements ServletContextListener {

    /** kod koji se izvrsava po nakon inicijalizacije ServletContext objekta */
    public void contextInitialized(ServletContextEvent event) {

    }

    /** kod koji se izvrsava po nakon uništavanja ServletContext objekta */
    public void contextDestroyed(ServletContextEvent event) {

    }
}
```

Inicijalizacija bean objekata

Korišćenjem init metode

MemorijaAplikacije init

- Moguće je nakon kreiranju bean objekta izvršiti njegovu inicijalizaciju oslanjajući se na metodu koja je anotirana kao metoda za inicijalizaciju bean

```
@Bean(name= {"memorijaAplikacije"}, initMethod="init", destroyMethod="destroy")
public MemorijaAplikacije getMemorijaAplikacije() {
    return new MemorijaAplikacije();
}

public class MemorijaAplikacije extends HashMap {
    public void init() {
        //inicijalizacija
        System.out.println("init method called");
        Filmovi filmovi = new Filmovi();
        FilmStatistika statistikaFilmova = new FilmStatistika();
        this.put(FilmoviController.FILMOVI_KEY, filmovi);
        this.put(FilmoviController.STATISTIKA_FILMOVA_KEY, statistikaFilmova);
    }
    ...
}
```

Case study – CRUD bioskop veb aplikacija

Vežbanje posle predavanja

- Pokušajte da uradite ProjekcijeController

Dodatno

Servletske klase koje omogućavaju rad sa sesijom

