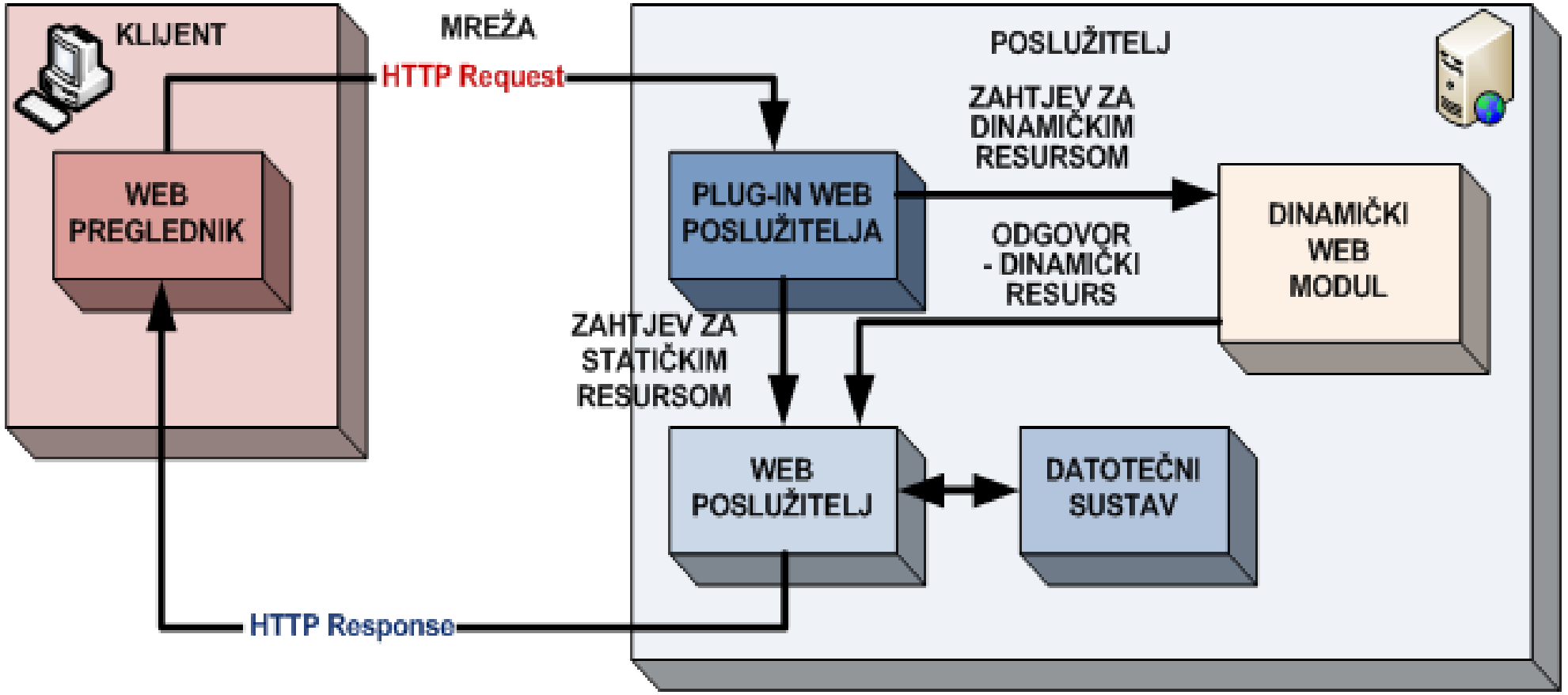
# Otvoreno računarstvo

## Pripremno predavanje – Servleti

Branko Mihaljević

# Servleti – ponavljanje

# Dinamičke stranice #2



# Što su Servleti?

- **Java razredi**, odnosno objekti koji se nalaze **na poslužitelju**
- Odgovaraju na **zahtjev** (Request) **odgovorom** (Response) korištenjem **protokola HTTP**
- Ovisno o ulaznim parametrima koje je poslao Web preglednik može se generirati različita stranica
- Povratna informacija je **dinamički generirana HTML stranica**
  - Stranica se generira unutar programskog koda
- Java Servlet API – sučelje za rad sa Servletima
- Sadrže poslovnu logiku koja odgovara na HTTP Request – **odgovaraju na GET i POST metode**
- **Pokreće i održava ih poslužitelj** – zapravo Web sadržnik (*container*) poslužitelja

# Servlet – koraci komunikacije

- Preglednik poziva Servlet na poslužitelju – URL
  - Ime servleta je dio URL-a
- Web poslužitelj prima zahtjev i zaključuje da se radi o Servletu
- Web poslužitelj prosljeđuje poziv dinamičkom Web modulu (Web sadržniku)
- Pronalazi se postojeća ili se kreira nova instanca (objekt) Servlet razreda istog imena
- Poziva se metoda Servlet objekta
- Servlet generira HTML kôd i vraća ga Web poslužitelju
- Web poslužitelj prosljeđuje HTML kod pregledniku

# Izgradnja HTTP Servleta

- Izgradnja razreda koji nasljeđuje **HttpServlet**
  - javax.servlet.http.HttpServlet
- Izgradnja (nadjačanih) **doGet**() i/ili **doPost**() metoda
  - Procesuiranje ulaznih parametara iz dobivenog **HttpServletRequest** objekta
  - Izgradnja odgovarajućeg poslovnog procesa
    - Izvršavanje metoda poslovne logike
    - Izgradnja HTML kôda
  - Postavljanje odgovarajućih vrijednosti na **HttpServletResponse** objekt
  - Prosljeđivanje (generiranog) HTML kôda **PrintWriter** objektu
  - Hvatanje iznimaka (IOException, ServletException)

# HttpServlet

- Razred HttpServlet je **apstraktni** razred koji se mora naslijediti kako bi se stvorio HTTP servlet
- Podklasa razreda HttpServlet mora nadjačati (*override*) barem jednu od sljedećih metoda:
  - **doGet**() - ako Servlet treba podržati GET metodu
  - **doPost**() - ako Servlet treba podržati POST metodu
  - **doPut**() - ako Servlet treba podržati PUT metodu
  - **doDelete**() - ako Servlet treba podržati DELETE metodu
  - **init**() i **destroy**()- ako Servlet upravlja resursima prije/nakon njegovo životnog vijeka
  - **getServletInfo**() - ako Servlet mora isporučiti informaciju o sebi
- 2 parametra:
  - **HttpServletRequest** – dohvat parametara, HttpSession, itd.
  - **HttpServletResponse** – povratni odgovor

# Servlet – neka objašnjenja

- Zašto se ne nadjačava metoda **service()**?
  - Nema potrebe za nadjačavanjem service() metode, jer ona prosljeđuje HTTP zahtjeve odgovarajućim doXXX() metodama ovisno o HTTP metodi u zahtjevu (npr. doPost() za POST metodu)
- Zašto postoji metoda **init()**, a ne konstruktor?
  - Povijesno, konstruktori za dinamički stvorene objekte (kakvi su Servleti) ne mogu primati argumente, a Servletu se prosljeđuje objekt koji implementra ServletConfig sučelje i sadrži informacije o okruženju
  - javax.servlet.Servlet je sučelje, pa ne može deklarirati konstruktor sa ServletConfig argumentom, već deklarira metodu init()
- Zašto postoji metoda **destroy()**?
  - Kako bi Servlet objekt mogao osloboditi sve resurse koji se ne mogu/znaju automatski očistiti (*garbage collection*), a mogu se i zapisati sve informacije koje je potrebno sačuvati.

# Životni ciklus Servleta

- Servlet se po potrebi učitava, instancira, ali i uništava
  - Pri pokretanju poslužitelja ili
  - Kad poslužitelj zaključi da je to potrebno
- Servleti se izvršavaju u JVM Web sadržnika (*container*)
  - Web sadržnik (a ne programer) **brine** kad će **stvoriti** novi Servlet objekt ili **uništiti** postojeći
- Zašto uopće imamo više instanci Servleta?
  - Zato jer istovremeno može **više korisnika** pristupiti aplikaciji i svaki želi da se njegov **zahtjev obradi** (trenutno, odnosno čim prije može)

# Servlet – višekorisnički rad

- Smisao svake Web aplikacije je da joj može istovremeno pristupiti veći broj korisnika
  - Aplikacije i sustavi se “grade” kako bi zadovoljile određeni (predviđeni) broj konkurentnih korisnika
  - Svaki korisnik pristupa svom skupu resursa koji rješavaju određenu funkcionalnost
- Ako više korisnika pristupi istom Servletu, na poslužitelju, odnosno Web sadržniku je da svakom korisniku pruži jednu instancu Servleta te može:
  - Iskoristiti postojeću, ako je objekt stvoren i nitko ga ne koristi
  - Stvoriti novi objekt, ako nema dovoljno već stvorenih
    - Princip višenitnog poslužitelja i bazena (*pool*) resursa

# Servlet – konfiguracija

- Aplikacijski poslužitelj se konfigurira kako bi znao da Servlet **postoji** i kako bi ga mogao **pokrenuti**
  - Konfiguracija zapisana kao XML datoteka **web.xml** (tzv. Web Application Deployment Descriptor)
    - Bit će detaljno kasnije objašnjena ... zasad samo ukratko

```
<web-app>                          - korijenski element
  …
  <servlet>                        - deklaracija Servleta (za svaki Servlet)
    …
  </servlet>
  …
  <servlet-mapping>                - definicija mapiranja Servleta na URL
    …
  </servlet-mapping>
  …
</web-app>
```

# Servlet – konfiguracija #2

- Deklaracija Servleta sadrži:
  - **Naziv** samog Servleta
  - **Naziv razreda** koji implementira Servlet
  - Uputa za poredak učitavanja pri pokretanju (opcionalno)
  - Naziv Servleta za prikaz (opcionalno)
  - Niz drugih stvari (npr. inicijalizacijski parametri) (opcionalno)

```
<servlet>
   <servlet-name>MySearchServlet</servlet-name>
   <servlet-class>hr.fer.or.SearchServlet</servlet-class>
   <load-on-startup>1</load-on-startup>
   <display-name>Moj Servlet</display-name>
</servlet>
```

- Napomena: ovo postoji za svaki Servlet koji se nalazi u Web sadržniku

# Servlet – konfiguracija #3

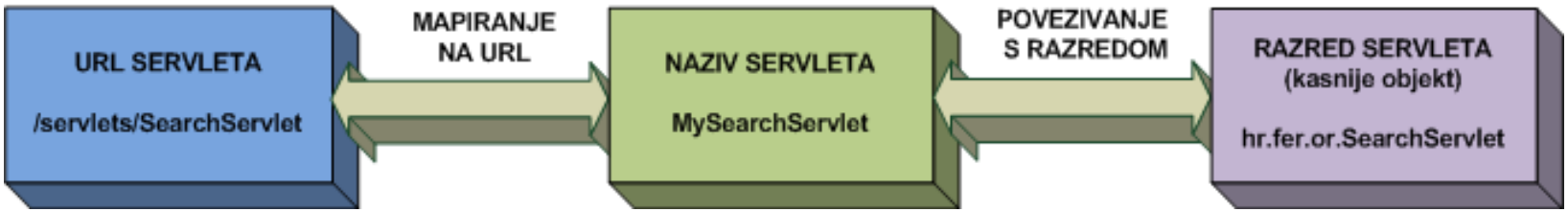
- Definicija mapiranja Servleta sadrži:
  - **Naziv** samog Servleta
    - **Mora biti isti naziv kao u deklaraciji!!!**
  - **URL** koji odgovara pozivom Servleta
    - Može biti proizvoljan
    - Kada dođe zahtjev za resursom koji je označen ovim URL-om Web sadržnik prosljeđuje zahtjev tom Servletu
  - ...

```
</servlet-mapping>
   <servlet-name>MySearchServlet</servlet-name>
   <url-pattern>/servlets/SearchServlet</url-pattern>
</servlet-mapping>
```

- Napomena: ovo postoji za svaki Servlet koji se nalazi u Web sadržniku

# Servlet – konfiguracija – ponavljanje

```
<web-app>                                   - korijenski element
    ...
    <servlet>                               - deklaracija Servleta
       <servlet-name>MySearchServlet</servlet-name>
       <servlet-class>hr.fer.or.SearchServlet</servlet-class>
       <load-on-startup>1</load-on-startup>
       <display-name>Moj Servlet</display-name>
    </servlet>
    ...
    </servlet-mapping>                      - definicija mapiranja Servleta na URL
       <servlet-name>MySearchServlet</servlet-name>
       <url-pattern>/servlets/SearchServlet</url-pattern>
    </servlet-mapping>
    ...
</web-app>
```



# Pripremno predavanje

# 5. lab. vježba – Java Servlet i PHP

- Cilj vježbe:
  - Izrada dinamičke Web stranice koja dohvaća XML podatke isporučene od strane izrađenog Servlet poslužiteljskog programa
- Proučiti:
  - **Osnove izrade Java Servleta**
    - Java Servlet Programming (O'Reilly) – poglavlja 1 – 5
      - http://www.unix.com.ua/orelly/java-ent/servlet/index.htm
    - The Java EE 5 Tutorial - Java Servlet Technology, Chapter 4 – dijelovi koji se odnose na vježbu (stvaranje, inicijalizacija, odgovor)
  - **Konfiguriranje web.xml za Servlete**
  - **Pokretanje Servleta na poslužitelju**

# Zadatak

- Potrebno je **izraditi Java Servlet** koji isporučuje podatke u obliku XML
  - Java Servlet je potrebno izraditi na temelju gotove aplikacije iz 4. laboratorijske vježbe koju je potrebno preraditi prema Servlet specifikaciji kako bi odgovarala na potrebne HTTP metode.
- Dodatno je potrebno doraditi **PHP skriptu** (koristeći rješenje 3. laboratorijske vježbe) za pretraživanje strukturiranog zapisa podataka u XML obliku koje sad dobiva od Java Servleta, a ne iz XML datoteke.

# HttpServlet ulaz, izlaz, životni ciklus

# HttpServletRequest

- Učahuruje (enkapsulira) HttpRequest tok
- Sadrži:
  - Zaglavlje, tip sadržaja, duljinu, metodu, kolačiće
  - Pozivni URL ili putanju do Servleta
  - Ostale parametre
- 2 verzije dohvata parametara:
  - getParameter() ili getParameterValues()
  - getReader()

# Servlet – čitanje parametara – primjer

```
public class SearchServlet extends HttpServlet {
        public void doPost(HttpServletRequest req, HttpServletResponse
            res) throws ServletException, IOException {
            ...
            Enumeration enum = req.getParameterNames();
            while (enum.hasMoreElements()) {
                String name = (String) enum.nextElement();
                String value = req.getParameter(name);
                //... ovdje ide neki korisni posao sa svakim parom ...
            }
            ...
        }
}
```

ili za pojedini: **String title = req.getParameter("ime_parametra");**

# HttpServletResponse

- Postavlja **tip sadržaja** (MIME)
  - Metoda **setContentType**()
- postavlja **statusni kod** (HTTP 1.1)
  - Metoda **setStatus**()
- Postavlja **zaglavlje**
  - Kolačići, caching
- Sadrži referencu na **PrintWriter**
  - Koristi se **samo za tekstualne dokumente** (HTML, XML)
  - Metoda **getWriter**()
  - uvijek nakon postavljanja tipa sadržaja i statusnog koda
- Sadrži referencu na **ServletOutputStream**
  - Koristi se **za ostali sadržaj** (npr. PDF, Word, ...)
  - Metoda **getOutputStream**()
- Moguće **redirekcije**:
  - Na druge URL-ove i na stranice greške

# Servlet – izlazni proces – primjer

```
public class MyServlet extends HttpServlet {
        public void doGet(HttpServletRequest req,   HttpServletResponse res)
            throws ServletException, IOException {
            ...
            // dohvat izlaznog toka
            res.setStatus(HttpServletResponse.SC_OK);
            res.setContentType("text/html");

            PrintWriter out = res.getWriter();
            // jednostavni dinamički sadržaj
            out.println("<HTML><BODY><H1>");
            out.println("Danas je "+(new Date()));
            out.println("</H1></BODY></HTML>");
            out.close();
        }
}
```

# Servlet – primjer

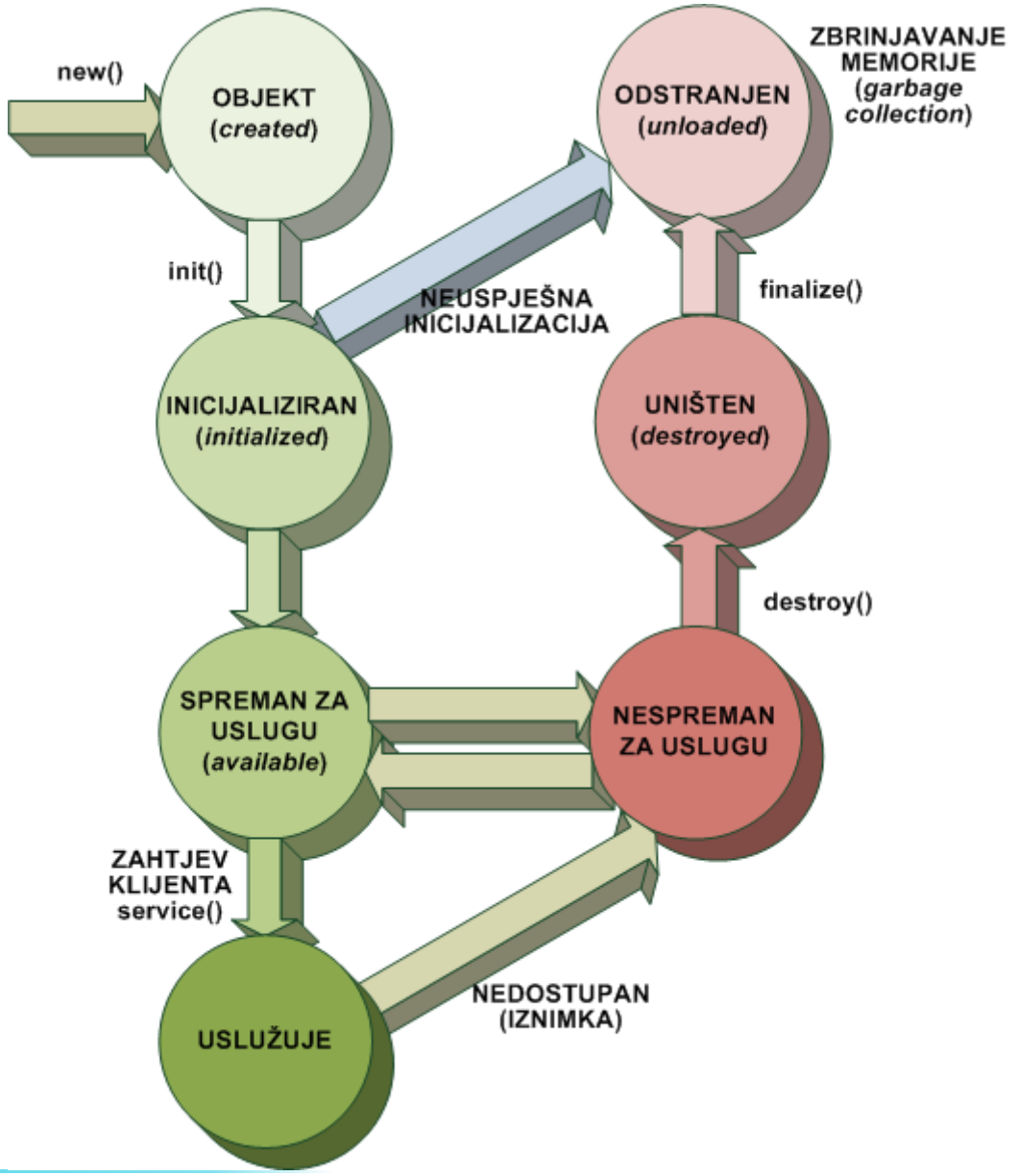
```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import java.io.IOException;
import java.io.PrintWriter;

public class SimpleServlet extends HttpServlet {
        public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

            String browser = request.getHeader("User-Agent");
            response.setStatus(HttpServletResponse.SC_OK);
            response.setContentType("text/html");

            PrintWriter out = response.getWriter();
            out.println("<HTML><BODY>");
            out.println("Koristite preglednik: " + browser);
            out.println("</BODY></HTML>");
            out.close();
        }
}
```

# Životni ciklus Servleta – kreiranje

- Kada će Web sadržnik kreirati instancu Servleta?
  - Kada mu se eksplicitno naredi da unaprijed učita Servlet
    - Konfiguracija u web.xml
      - U oznaci <load-on-startup> upisati vrijednost veću od 0
    - Servlet je učitan i spreman za uporabu kada dođe zahtjev
  - Ako ne postoji učitani Servlet, onda će se kreirati instanca kada dođe novi zahtjev za Servletom
- Prilikom učitavanja Servleta u memoriju, Web sadržnik pokreće inicijalizaciju

# Životni ciklus Servleta – inicijalizacija

- Inicijalizacija – **init**() metoda
  - Poziva se samo jednom
  - Služi za:
    - Učitavanje parametara
      - Npr. opće postavke
    - Inicijalno konfiguriranje
      - Podešavanje kako će Servlet raditi
      - Npr. kodna stranica i sl.
    - Veze prema resursima
  - Postoje 2 init() metode:
    - Bez parametara
    - S ulaznim parametrom ServletConfig
      - Dostup do okruženja u kojem se Servlet izvršava



# Životni ciklus Servleta – zahtjev

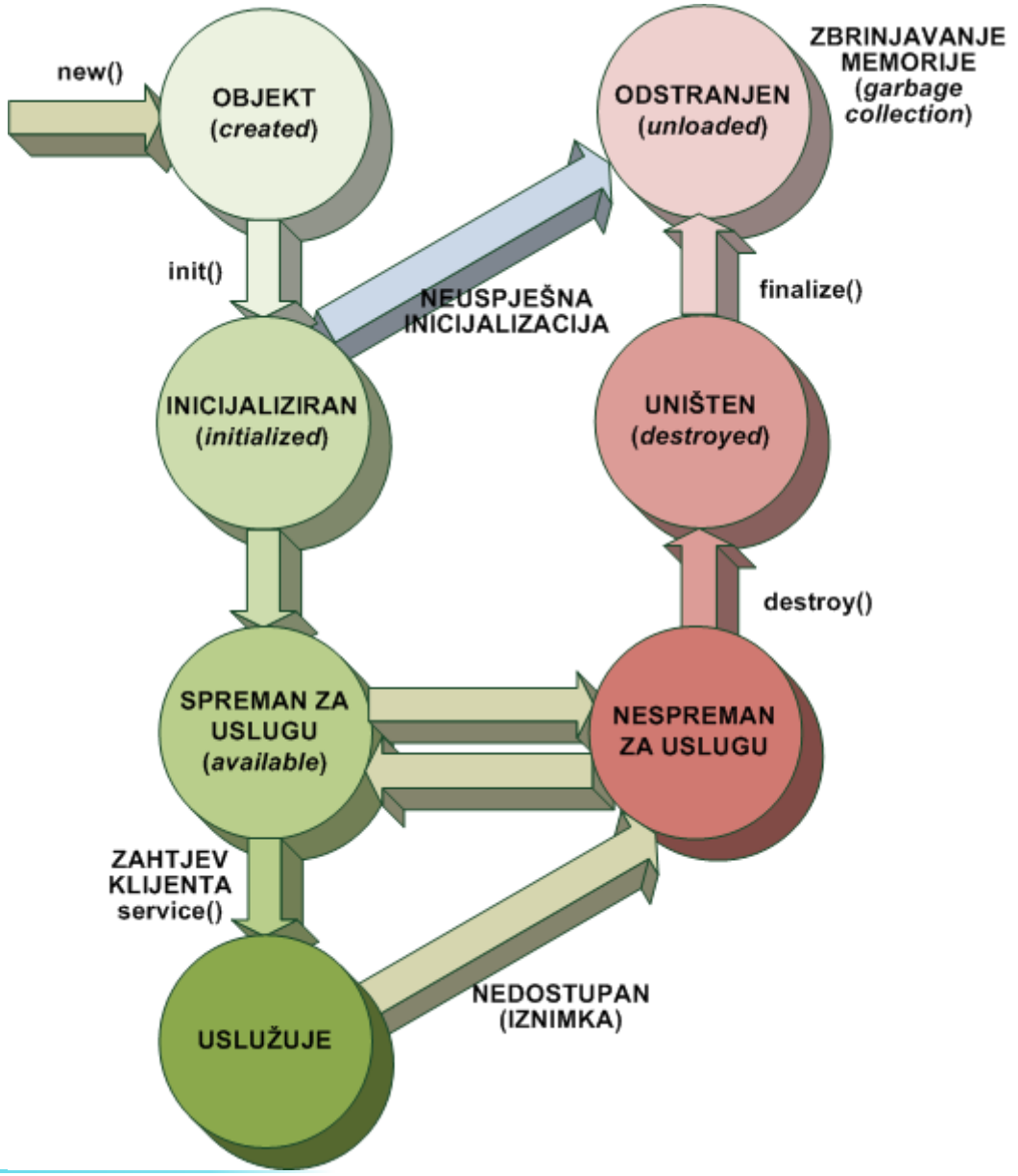
- Ako je inicijalizacija uspješna Servlet je **spreman za uslugu** (*available for service*)
  - Ako nije Servlet se **odstranjuje iz memorije** (*unload*)
- **Upravljanje zahtjevom**
  - Kada Web sadržnik dobije zahtjev za Servletom, **prosljeđuje** ga **service**() metodi Servleta
  - **Kod HTTP zahtjeva nadogradnja service() metode** s metodama **doXXX() ovisno o HTTP metodi zahtjeva**:
    - **doGet**() - odgovara na zahtjev poslan metodom **GET**
    - **doPost**() - odgovara na zahtjev poslan metodom **POST**
    - **doOptions**() - odgovara na zahtjev poslan metodom OPTIONS
    - **doPut**() - odgovara na zahtjev poslan metodom PUT
    - **doDelete**() - odgovara na zahtjev poslan metodom DELETE

# Servlet – obrada zahtjeva

- Servlet prihvaća zahtjev od klijenta u metodi **doXXX()** (npr. doPost())
- **Svaki zahtjev – nova nit**
- Metoda doXXX() obrađuje zahtjev i vraća odgovor
  - Preuzima ulazne parametre
  - Odrađuje “posao”
  - Vraća rezultat u HTML obliku

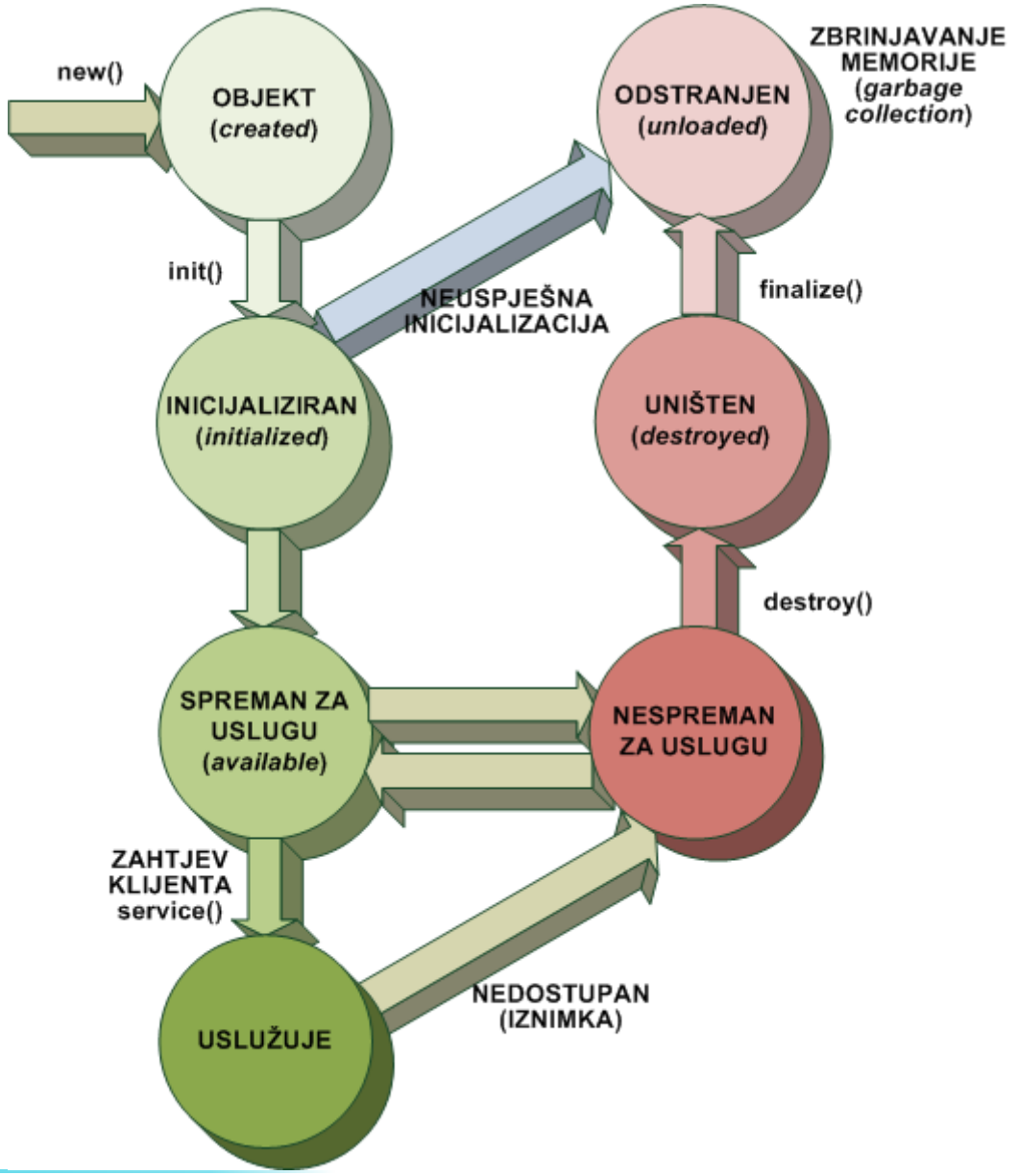
# Životni ciklus Servleta – dostupnost

- Servlet može biti nedostupan
  - Kada uslužuje zahtjev
  - Kada je nespreman za uslugu iz nekog drugog razloga
    - Npr. greška
- Ako je Servlet privremeno nedostupan, on može postati ponovno dostupan kada:
  - Usluži zahtjev
  - Razriješi problem (grešku)



# Životni ciklus Servleta – završetak

- Servlet se uništava kod:
  - Čuvanja memorijskih resursa
  - Gašenja poslužitelja
  - Greške koja onemogućava njegovu dostupnost
- Poziv metode **destroy**()
  - Provjera da li su sve niti (procesi) završili “posao”
  - Poništavanje svih promjena koje se nisu automatski riješile
- Uništavanje Servleta (*unload*)
  - Zbrinjavanje memorije (*garbage collection*)



# Pitanja?