Osnove web programiranja

Servleti, Tomcat

Termin 1

Sadržaj

1. Servleti uvod
2. Setup i Apache Tomcat
3. Kreiranje projekta
4. HTTP request i HTTP Response
5. Preuzimanje podataka iz formi
6. Memorija na nivou aplikacije

Dodatno:

1. web.xml dodatno
2. ServletConfig i ServletContext
3. Servlet classes
4. Servlet lifecycle
5. Konkurentni pristup servletu

Servleti uvod

Uvod

- Tehnologija za generisanje dinamičkih sadržaja
- WWW server se proširuje podrškom za servlete
- Rezultat izvršenja servleta je dinamički kreiran sadržaj
- Kreiramo klase koje koje nasleđuje klasu **`HttpServlet`**

Servleti uvod

```
public abstract class HttpServlet {

  protected void init(ServletConfig cnf) {}

  protected void doGet(HttpServletRequest request, HttpServletResponse response){}
  protected void doPost(HttpServletRequest request, HttpServletResponse response){}
  protected void doPut(HttpServletRequest request, HttpServletResponse response){}
  protected void doHead(HttpServletRequest request, HttpServletResponse response){}
  protected void doDelete(HttpServletRequest request, HttpServletResponse response{}
  protected void doOptions(HttpServletRequest request, HttpServletResponse response){}
  protected void doTrace(HttpServletRequest request, HttpServletResponse response){}

  protected void destroy(){}

  protected void service(HttpServletRequest request, HttpServletResponse response) {

    if (request.getMethod().equals("GET"))
      doGet(request, response);
    else if (request.getMethod().equals("POST"))
      doPost(request, response);
       else if ...
  }
}
```

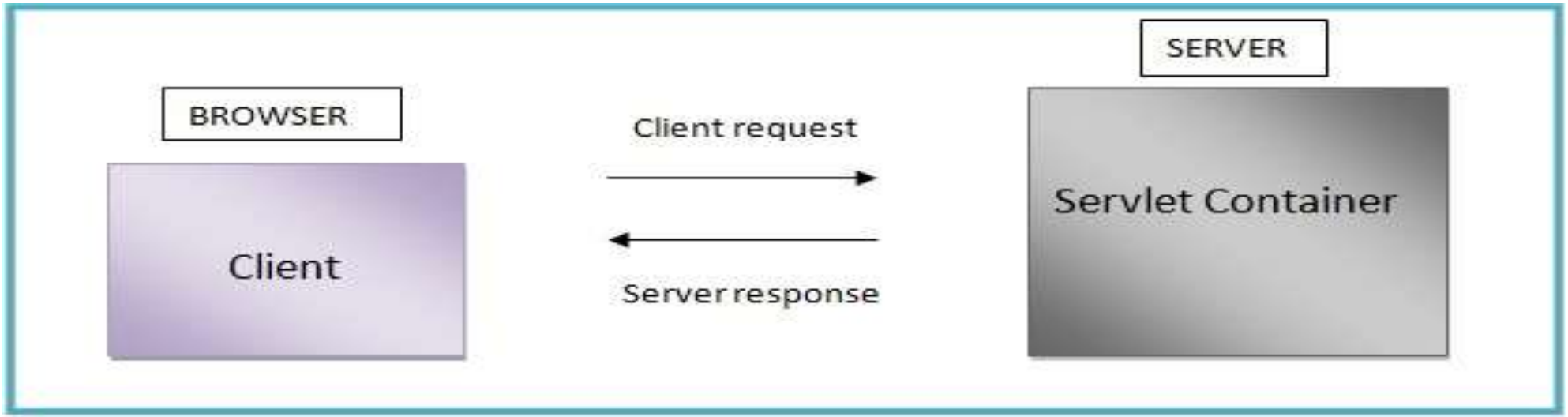
Redefinisati metode

...

Servleti uvod

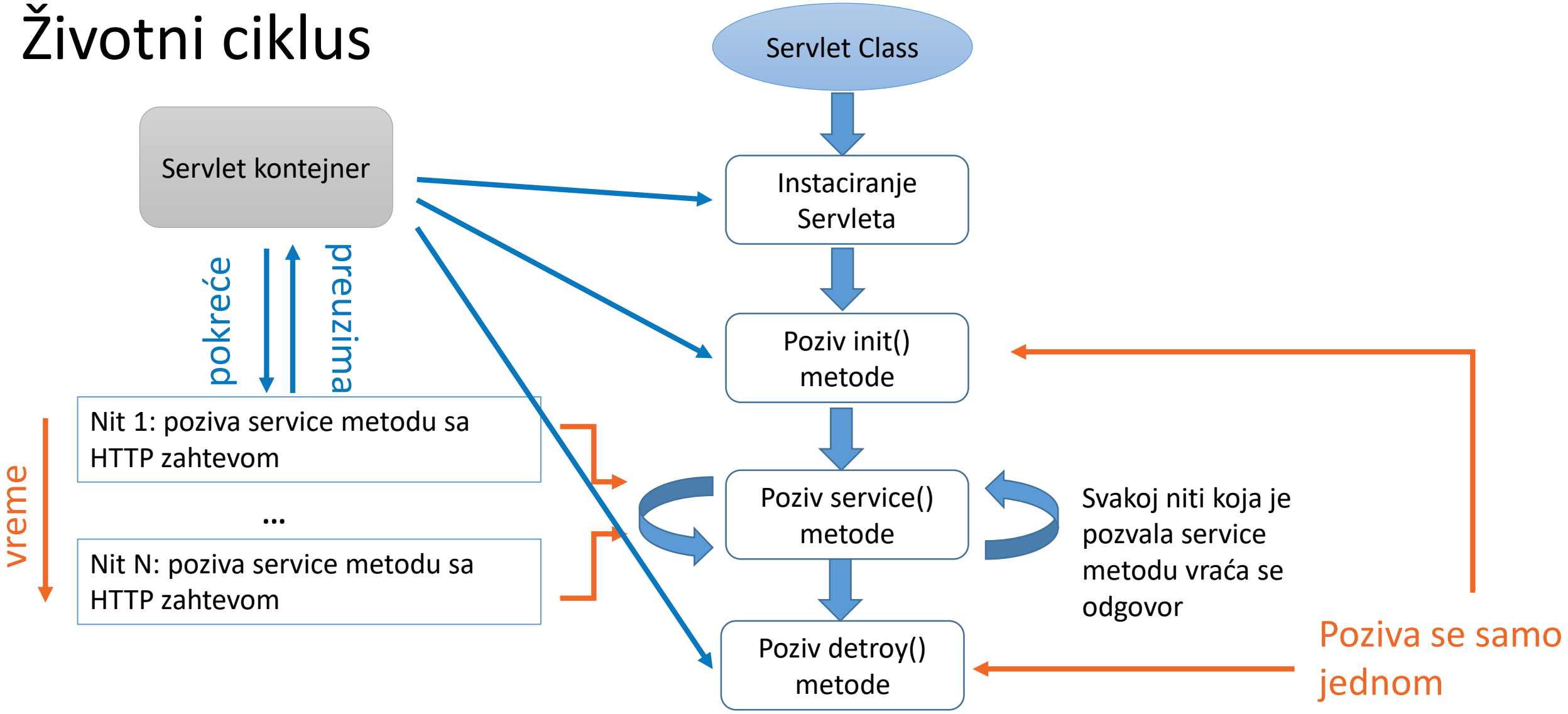
Životni ciklus

- Klijent šalje HTTP zahtev ka Servlet kontejneru. Kontejner se ponaša kao veb server.
- Veb server obrađuje URL i na osnovu njega pretražuje koji Servlet treba da se koristi i inicijalizuje ga.
- Podaci iz HTTP zahtev se posleđuju servletu, on ih procesira i vraća odgovor veb serveru.
- Veb server na osnovu odgovora servleta kreira HTTP odgovor koji se zatim prosleđuje Klijentu



Servleti uvod

Životni ciklus



Servleti

HttpServlet.init()

- namenjena za inicijalizaciju prilikom pokretanja servleta

```
public void init() {
   Connection conn = DriverManager.getConnection(...);
   ...
}
```

```
public void init(ServletConfig cnf) {
   super.init(cnf);
   Connection conn = DriverManager.getConnection(...);
   ...
}
```

Servleti

HttpServlet.destroy()

- namenjena za clean-up zadatke neposredno pre uništenja servleta

```
public void destroy() {
   conn.close();
}
```

Servleti

HttpServlet.doGet()

Namenjena za obradu GET zahteva

Tipičan scenario poziva:

- postavi Content-type HTTP odgovora
- uzmi PrintWriter ka klijentu
- kroz PrintWriter šalji dinamički kreiran sadržaj

```
public void doGet(HttpServletRequest req, HttpServletResponse res) {

  res.setContentType("text/html");

  PrintWriter out = res.getWriter();
  out.println("<HTML>");
  out.println("<HEAD><TITLE>Test</TITLE></HEAD>");
  out.println("<BODY>");
  ...
}
```

Setup i Apache Tomcat

Setup za pokretanje primera

Napravi novi workspace ImePrezime/Predmet2Web

Otpakujte materijale za ovo predavanje, u njima imate zip arhivu *apache-tomcat-9.0.31.zip*

Napravi folder ImePrezime/Servers i u njemu otpakujte tomket apache-tomcat-9.0.31.zip

Pokreni Eclipse EE i postavi putanju na novi workspace

- Eclipse New->Dynamic Web Project (ne može, nedostaje server)

Vraćamo se u Eclipse gde moramo u podešavanjima da postavimo putanju do otpakovanog servera

- Eclipse *Window->Preferences->Server->Runtime Environments->Add*
- Wizard za dodavanje *Apache->Apache Tomcat v9.0->Next*
- Dugme *Browse*… pa se pozicionirati na lokaciju gde je otpakovan Tomcat tj. na ImePrezime/Servers/apache-tomcat-9.0.31
- Dugme *Finish->Apply and Close*

primer01

Setup i Apache Tomcat

Apache Tomcat

Apache Tomcat je veb server otvorenog koda razvijen od strane *Apache Software Fondation*.

Tomcat predstavlja veb kontejner koji u potpunosti podržava java veb bazirane aplikacije.

Izgled *apache tomcat* web servera, struktura foldera:

- *bin* - sadrži skripte i exe fajlove koji omogućavaju upravljanje bazičnim radom samog servera (pokretanje i zaustavljanje).
  - Za Windows os iz cmd pokrećemo ***startup.bat***
  - Za Linux os iz terminala pokrećemo ***sh catalina.sh run***
- *conf* – sadrži skripte za podešavanje konfiguracije veb servera.
- *lib* – sadrži biblioteke koje koristi Tomcat prilikom rada. Bibloteke koje se nalaze u ovom folderu dostupne su za sve aplikacije koje su podignute pod Tomacat serverom
  - *servlet-api.jar* je ključna biblioteka koja se koristi u *Eclipse* za nasleđivanje *HttpServlet*
  - možemo ubaciti mysql connector jar

Setup i Apache Tomcat

Apache Tomcat

Izgled *apache tomcat* web servera, struktura foldera:

- *logs* - Tomcat upisuje izveštaje tokom svog rada
- temp – se koristi kao pomoćni folder za smeštanje privremenih resursa(fajlova) za sam *Tomcat*, koji nastaju tokom rada *Tomcat*, može da se briše sadržaj kada je T*omcat* isključen
- *work* – se koristi kao pomoćni folder za smeštanje privremenih resursa(fajlova) za aplikacije postavljene na *Tomcat*, koji nastaju prilikom izvršavanja aplikacije, može da se briše sadržaj kada je *Tomcat* isključen
- webapps – najbitiniji folder, zadrži war arhive i web aplikacije
  - war arhiva je zip fajl koji sadrži *class* fajlove i *WebContent*
  - U njemu kopiramo war arhivu koja se raspakuje prilikom pokretanja tomcat servera u foldere koji predstavljaju postavljene web aplikacije na web server

Setup i Apache Tomcat

Apache Tomcat pokretanje

- Apache *Tomcat* se pokreće tako što se otvori konzola u direktorijumu bin
- Pokretanje naredbe
  - Linux: **sh catalina.sh run**
  - Windows: startup.bat
- Zaustavljanje - zatvorite konzolu u kojoj je on startovan ili istovremeno pritisnete tastere CTRL i C dok se nalazite u konzoli
- Provera da li je *Tomcat* pokrenut?
  - u adress baru web brauzera uneti localhost:8080
- Pokretanje konkretne aplikacije koja je postavljena pod *Tomcat* veb kontejnerom obavlja se tako što se u adres baru veb brauzera unese
  - localhost:8080/Ime_Aplikacije
- localhost – je simboličko ime vaše lokalne mašine. Umesto ovoga može da stoji i IP adresa.
- 8080 – predefinisani port na kome je pokrenut *Tomcat* veb server
- Ime_Aplikacije – Najčešće se poklapa sa imenom projekta(aplikacije) koju ste razvijali.

Kreiranje projekta

Dynamic Web Module je 2.5

- Eclipse *New->Dynamic Web Project*
- Odredimo ime projekta npr. ***PrviServletiProjekat*** i odabiremo Runtime Environment *Apache Tomcat v9.0* i odaberemo *Dynamic web module version 2.5,* pa *Next-> Next*
- *ContexRoot* je ključno jer je to ime preko kojeg se aplikacija poziva iz web browser-a
- U *Content directory* folderu čuvamo slike, fajlovi, HTML stranice, CSS stilovi...
- Generate *web.xml* treba da je selektovano, sadrži opis konfiguracije web aplikacije za potrebe servera
  - Note: projekat ima ikonicu Zemlje

Kreiranje projekta

Opis strukture projekta

- Stavke projekta ne moraju predstavljati fizičke direktorijume ili datoteke na disku
  - Deployment deskriptor – je u stvarnosti fajl *web.xml*
  - Jax WS Web Services – ne postoji folder na disku
  - Java resuorces – ne postoji folder na disku.
- Struktura foldera
  - Java resuorces – sadrži java klase i java referencirane biblioteke.
  - src folder – idu java klase (izvorne datoteke, binovi i servleti)
  - WebContent - nalaze veb resursi kao što su slike, fajlovi, HTML stranice, CSS stilovi .
  - META-INF – ne dirati jer je to za jar fajlove
  - WEB-INF – je folder rad web aplikacije. Sadrži relevantne veb resurse za funkcionisanje veb aplikacije kao što su prekompajlirane java klase (podfolder *classes*), dodatne biblioteke koje koristimo (folder *lib*), opis rada veb aplikacije (datoteka *web.xml*)

Kreiranje projekta

Kreiranje Servleta

- U projektu je potrebno kreirati stukturu paketa npr. *webt1.zad01*
- *New->Servlet* dati ime klase npr. *ZdravoSvete*, pa -> *NEXT*
  - Servlet je najobičnija java klasa koja mora da nasledi klasu HttpServlet (roditeljska klasa je servlet) i implementira neke metode
- Opciono se može postaviti opis npr. *moj prvi servlet*
- *URL mapping* definiše putanju u web brauzeru preko koje će se preistupati kreiranom servletu (NOTE: ne mora biti isti kao naziv *ZdravoSvete* servlet java klase). Pored postojećeg mapiranja dodajte mapiranje */Hello.*
- U seldećem dijalogu selektuju se metode koje se žele implementirati, *doGet*, *doPost*, pa *Finish*

Kreiranje projekta

Kreiranje Servleta

- Metodi *doPost* stavimo da poziva doGet metodu doGet(request, response);
- Metoda *doGet* prekopirati kod

```
PrintWriter pw = response.getWriter();
response.setContentType("text/html");
pw.println("<html>");
pw.println("<body>");
pw.println("<h1>Zdravo svete</h1>");
pw.println("<p>"+Math.random()+"</p>");
pw.println("</body>");
pw.println("</html>");
pw.flush();
pw.close();
```

Kreiranje projekta

Postavljanje aplikacije na Tomcat

Prilikom svakog postavljanja aplikacije obavezno uvek izvršavati sledeće korake

- Zaustaviti rad *Tomcat* veb konterjenra (pritisnete zajedno tastere CTRL i C dok se nalazite u konzoli)
- Iz foldera *webapps* obrisati *war* arhive i raspakovane aplikacije
- U Eclipse desni klik na projekat pa *Export->War File*
- Potrebno je specificirati odredište na kom se postavlja war arhiva. *Browse*, odabira se putanja do *webapps* foldera *Tomcat*
- Selektujte *override existing file*, to se selektuje obavezno svaki naredni put kada se eksportuje *war* fajl
  - Pogledati sadržaj *webapps* foldera *Tomcat*

- Pokrenuti *Tomcat (*otvoriti konzolu i uneti za Linux *sh catalina.sh run* ili za Windows *startup.bat*
  - Pogledati sadržaj *webapps* foldera *Tomcat*
- Pristupiti aplikaciji preko brauzera. Kucate sledeću adresu
  - *http://localhost:8080/Ime_Aplikacije/Resurs*
  - *http://localhost:8080/PrviServletiProjekat/ZdravoSvete*
  - *http://localhost:8080/PrviServletiProjekat/Hello*

Kreiranje projekta

Web Deployment Descriptor - web.xml

Opis veb aplikacija za potrebe servera se nalazi u *web.xml*

Opisuju se mapiranja za Servlete, resursi i konfiguracije, kao i način kako će ih web server koristiti kada odgovara na zahteve klijenata

Otvorite web.xml projekta iz Eclipse, nalazi se u direktorijumu *WebContent->WEB-INF*

- <servlet> definiše java web servlet klasu
  - <servlet-name> predstavlja ime servleta,
  - <servlet-class> predstavlja putanju (hijerarhiju paketa-direktorijuma) do konkretne klase Servlet u projeku
- <servlet-mapping> mapiranje koje koristi sam web server prilikom odgovra na zahteve korisnika, ide u paru sa elementom <servlet>
  - <servlet-name> mora da se poklapa sa istim imenom <servlet-name> iz <servlet> elementa
  - <url-pattern> je naziv URL putanje preko koji se može pristupiti resursu servletu

Kreiranje projekta

Dynamic Web Module je 4.0

- *Dynamic web module version 4.0* predstavlja ternutno aktuelnu verziju modula
- Za razliku od verzije 2.5 u verziji 3.0 ka na dalje nije potrebno ekspicitno definisati mapiranja u *web.xml* fajlu već se mapiranja mogu odraditi direktno iz java koda anotacijama u okviru Servlet klase. U *web.xml* se razlikuje *namespace*.
- Kreiranje novog projekta je isto kao što je ranije već bilo pokazano, samo se sada postavi *Dynamic web module version 4.0*
- Import postojećeg projekta
  - U okviru workspace ImePrezime/Predmet2Web raspakovati zip arhivu DrugiServletiProjekat.zip
  - Izvršite uvlačenje projekta DrugiServletiProjekat u Eclipse
  - Izvršite postavljanje aplikacije na Tomcat
  - Pristupiti aplikaciji preko brauzera. Kucate adresu
    - http://localhost:8080/DrugiServletiProjekat/HelloHello

Import

Primer02 - ZdravoSvete

Kreiranje projekta

Dynamic Web Module je 4.0

- Uvođenje anotacija u source kodu preko kojih se konfiguriše rad web aplikacije nezaobilazni je deo svakog novog radnog okvira (framework), bez obzira na programski jezik na koji se taj radni okvir oslanja.

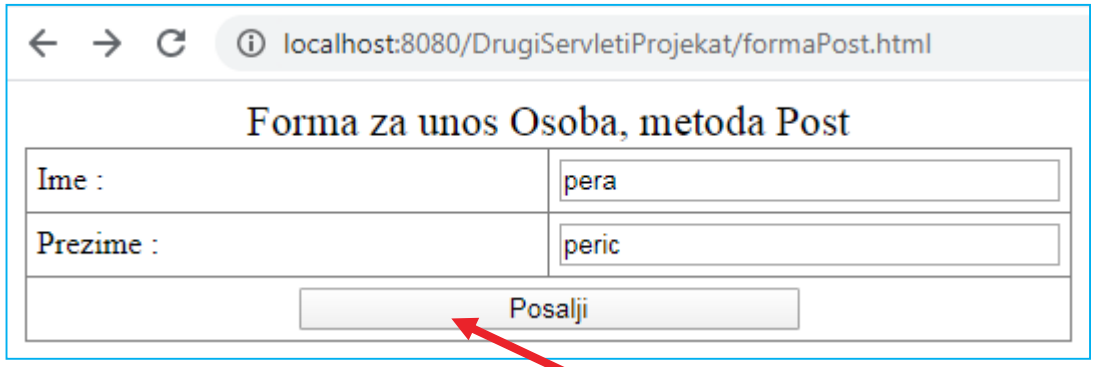
HTTP request i HTTP Response

HTTP zahtev (klasa HttpServletRequest)

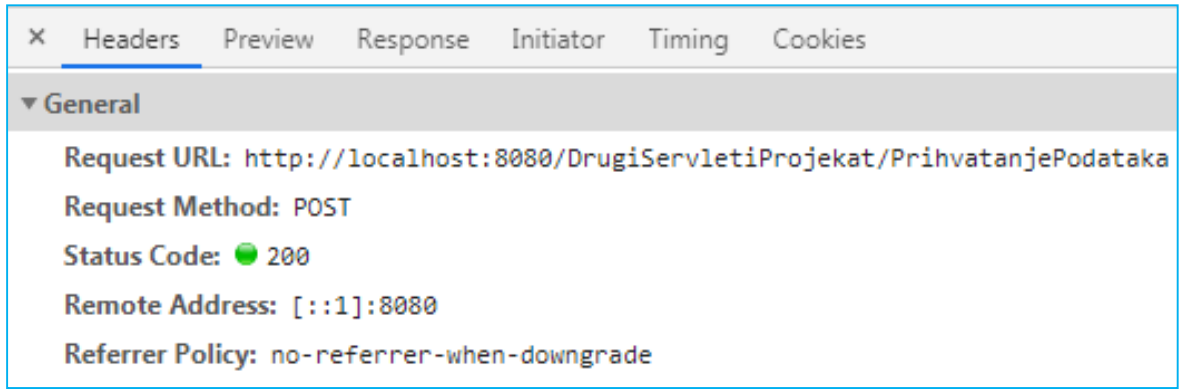
- Reprezentuje HTTP zahtev
- Izdvaja parametre forme prenete GET ili POST metodom i smešta ih u asocijativnu listu (naziv_polja_iz_forme, vrednost)
  - metode getParameter(ime), getParameterNames(), getParameterMap()
- Prikuplja sve parametre zaglavlja HTTP zahteva i smešta ih u asocijativnu listu (naziv, vrednost)
  - metode getHeader(ime), getHeaderNames() i getHeaders(ime)
- Metodom setCharacterEncoding se podešava *character encoding*
  - request.setCharacterEncoding("UTF-8");

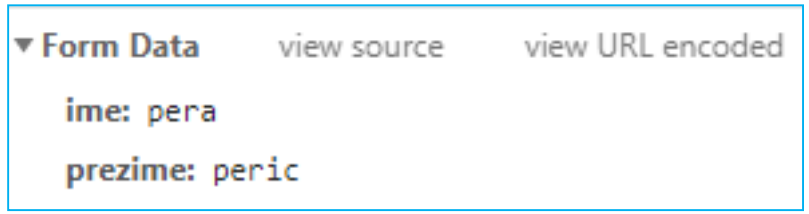
HTTP request i HTTP Response

Post metoda i preuzimanje podataka



Pošalji





formPost.html

HTTP request i HTTP Response

Preuzimanje podataka iz forme

```
request.setCharacterEncoding("UTF-8");
response.setContentType("text/html; charset=UTF-8");
PrintWriter pw = response.getWriter();
pw.println("<html>");
pw.println("<body>");
pw.println("<h1>Prihvatanje podataka</h1>");
pw.println("<br>Klijent koji je pozvao ovaj servlet je: "
                            + request.getHeader("User-Agent"));
pw.println("<p>Poslali ste:"+request.getParameter("ime")
                            +" "+request.getParameter("prezime")+ "</p>");
pw.println("</body>");
pw.println("</html>");
pw.flush();
pw.close();
```

Primer03 - Prihvatanje parametara

HTTP request i HTTP Response

HTTP odgovor (klasa HttpServletResponse)

- Reprezentuje HTTP odgovor
- Čuva tip odgovora (atribut Content-Type)
  - metoda setContentType(vrednost)
- Čuva cookie (atribut SetCookie)
  - metoda addCookie(cookie)
- **Omogućuje redirekciju (Location) na drugu stanicu**
  - **metoda sendRedirect(nova_lokacija)**
- Podešava proizvoljan atribut zaglavlja
  - metoda setHeader(naziv, vrednost)
- Ugrađuje ID sesije ako cookies nisu uključeni
  - metode encodeURL(url) i encodeRedirectURL(url)
- Čuva izlazni tok podataka

HTTP request i HTTP Response

HTTP odgovor (klasa HttpServletResponse)

- Metodom setContentType se podešava i *character encoding*

```
response.setContentType("text/html; charset=UTF-8");
```

- Parametar *charset* definiše kodnu stranu kojom će biti kodirani svi stringovi ka klijentu.

Primer

```
request.setCharacterEncoding("UTF-8");
response.setContentType("text/html; charset=UTF-8");
PrintWriter pout = response.getWriter();
pout.println("<html>");
pout.println("<head>");
pout.println("<meta http-equiv=\"Content-Type\" content=\"text/html;
  charset=UTF-8\">");
pout.println("</head>");
pout.println("<body>");
try {
  pout.println("Ovo je stranica sa UTF-8 karakterima: \u0428 \u0429<br>");
} catch(Exception ex) {
  pout.println(ex.getMessage());
}
pout.println("</body>");
pout.println("</html>");
pout.flush();
pout.close();
```

Memorija na nivou aplikacije

Servlet Context

- Reprezentuje kompletnu veb aplikaciju unutar JVM, sadrži sve Servlete
- Deljena memorija za sve Servlete
- Podaci koji se dodaju u ovu memoriju vidlivi su u celoj aplikaciji
  - u svim klasama koje predstavlju kontrolere
  - U svim dinamički generisanim pogledima
- Ponaše se kao asocijativna mapa
- Moramo upisati podatak u tu memoriju:

```
Objekat objekat;
getServletContext().setAttribute("ime_atributa", objekat);
```

- Ime atributa mora biti jedinstveno
- Čitanje atributa iz Servlet Context:

```
Objekat obj
 =(Objekat)getServletContext().getAttribute("ime_atributa")
```

Memorija na nivou aplikacije

Servlet Context

- Metoda getRealPath(java.lang.String path)
  - omogućuje vraćanje realne putanje resursa koju on zauzima na čvrstom disku u odnosu na virtualnu putanju resursa unutar veb aplikacije
  - korisno da se utvrdi relna putanja fajla koji se nalazi i kosriti unutar veb aplikacije

Primer04 - Rad se ServletContext

PrviServlet
DrugiServlet

Dodatno

web.xml dodatno

web.xml

- Pogledati web.xml u DrugiServletiProjekat

ServletConfig i ServletContext

ServletConfig

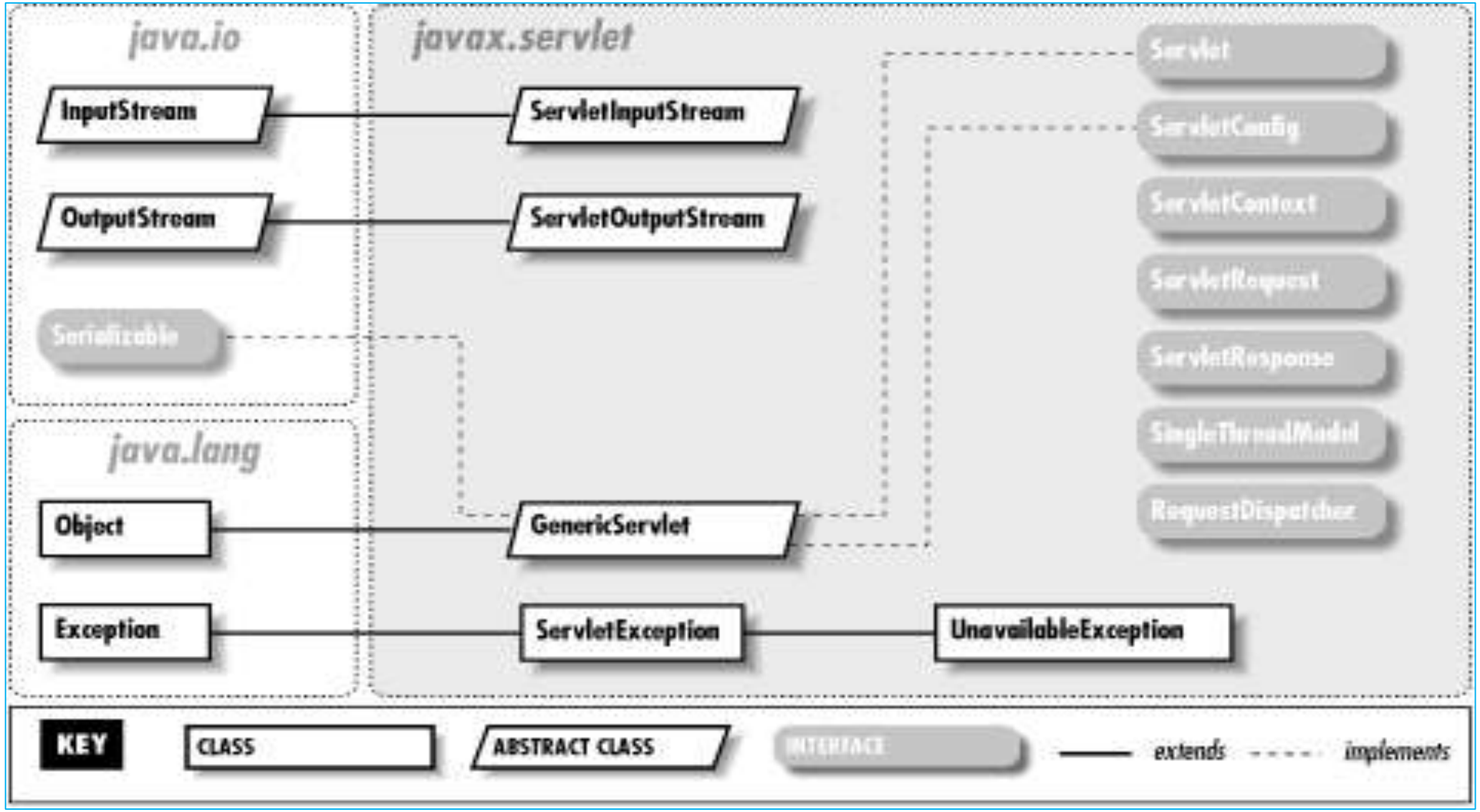
- A ServletConfig object is used by a Servlet Container to pass information to a servlet during initialization.
- Information like init parameters and their values are available in this object for a init parameters set in web.xml for a specific Servlet.
- Info like JDBC driver name, path to database and stuffs like this can be obtained from servlet *config*.
- Usefull methods
  - getInitParameter(java.lang.String name),
  - getInitParameterNames(),
  - getServletContext()

ServletConfig i ServletContext

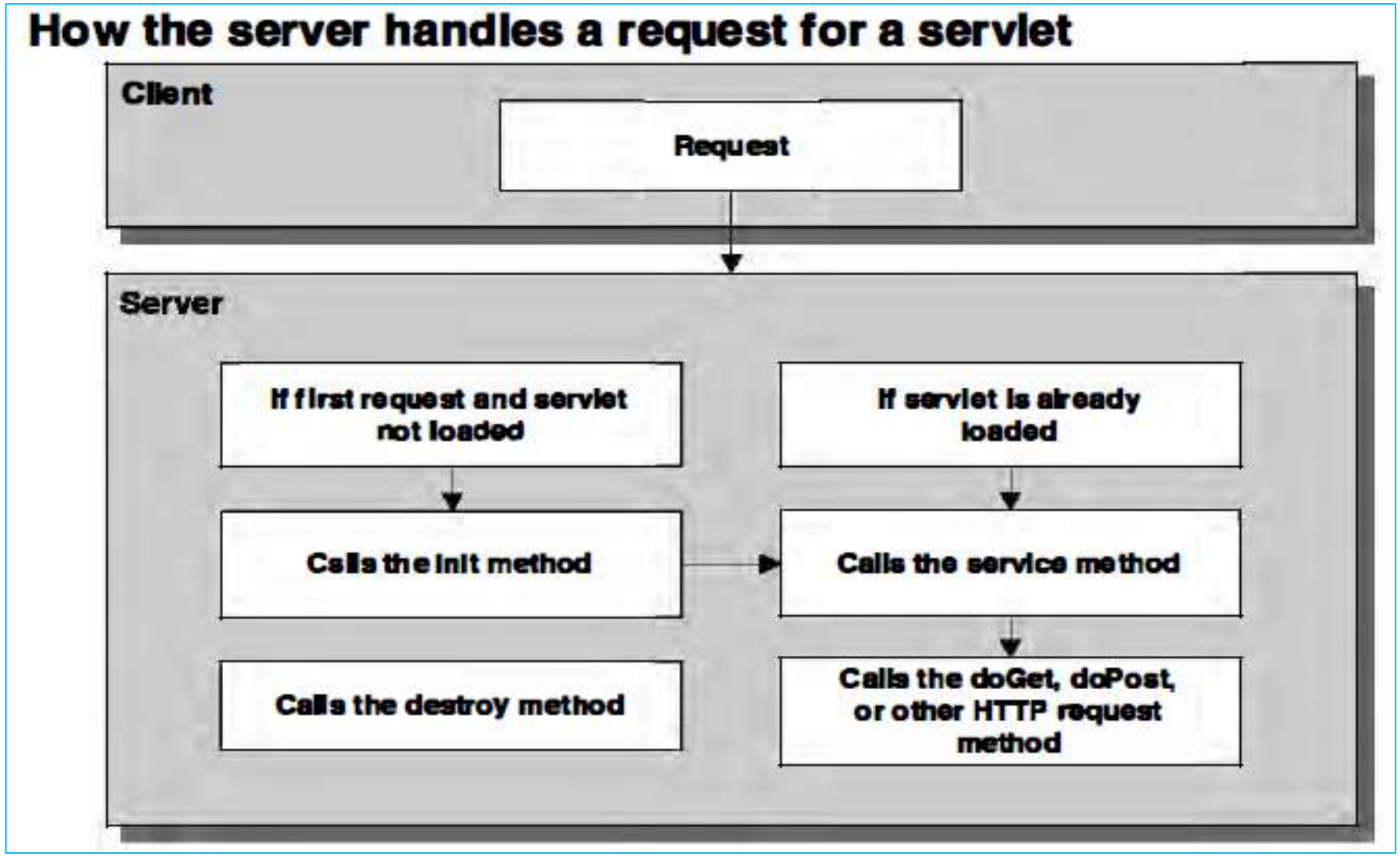
ServletContext

- Represent the compete web application within a JVM
- Typical web app containing servlets, JSPs, beans etc., run inside a single context called servlet context - a virtual wrapper.
  - ServletContext is an object which contains information about all Servlets in a JVM
- Defines a set of methods that a servlet uses to communicate with its Servlet Container
- There is one context per "web application" per Java Virtual Machine.
- Usefull methods
  - getInitParameter(java.lang.String name)
  - getInitParameterNames()
  - void setAttribute(java.lang.String name, java.lang.Object object)
  - Object getAtribute(java.lang.String name)
  - String getRealPath(java.lang.String path)
    - Returns a String containing the real path for a given virtual path.
    - Useful for knowing the real path of the file located within project

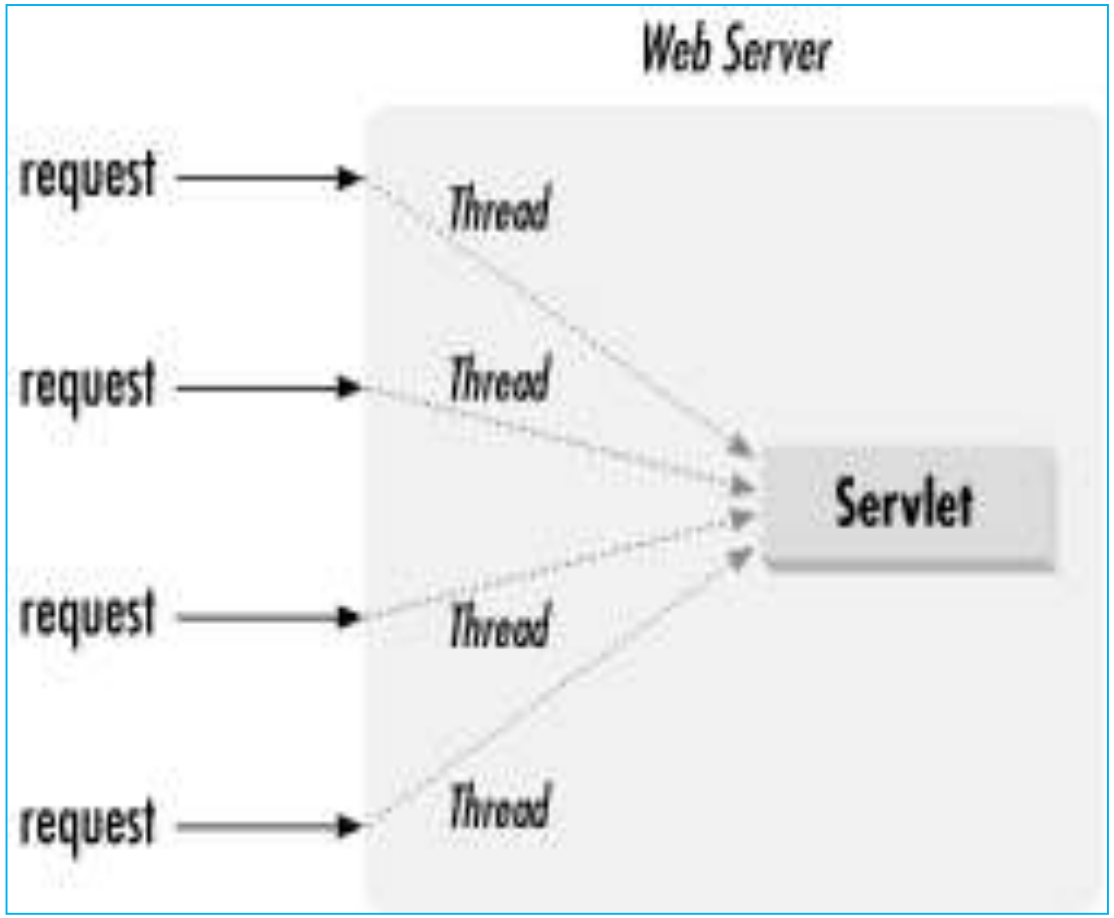
Servlet classes

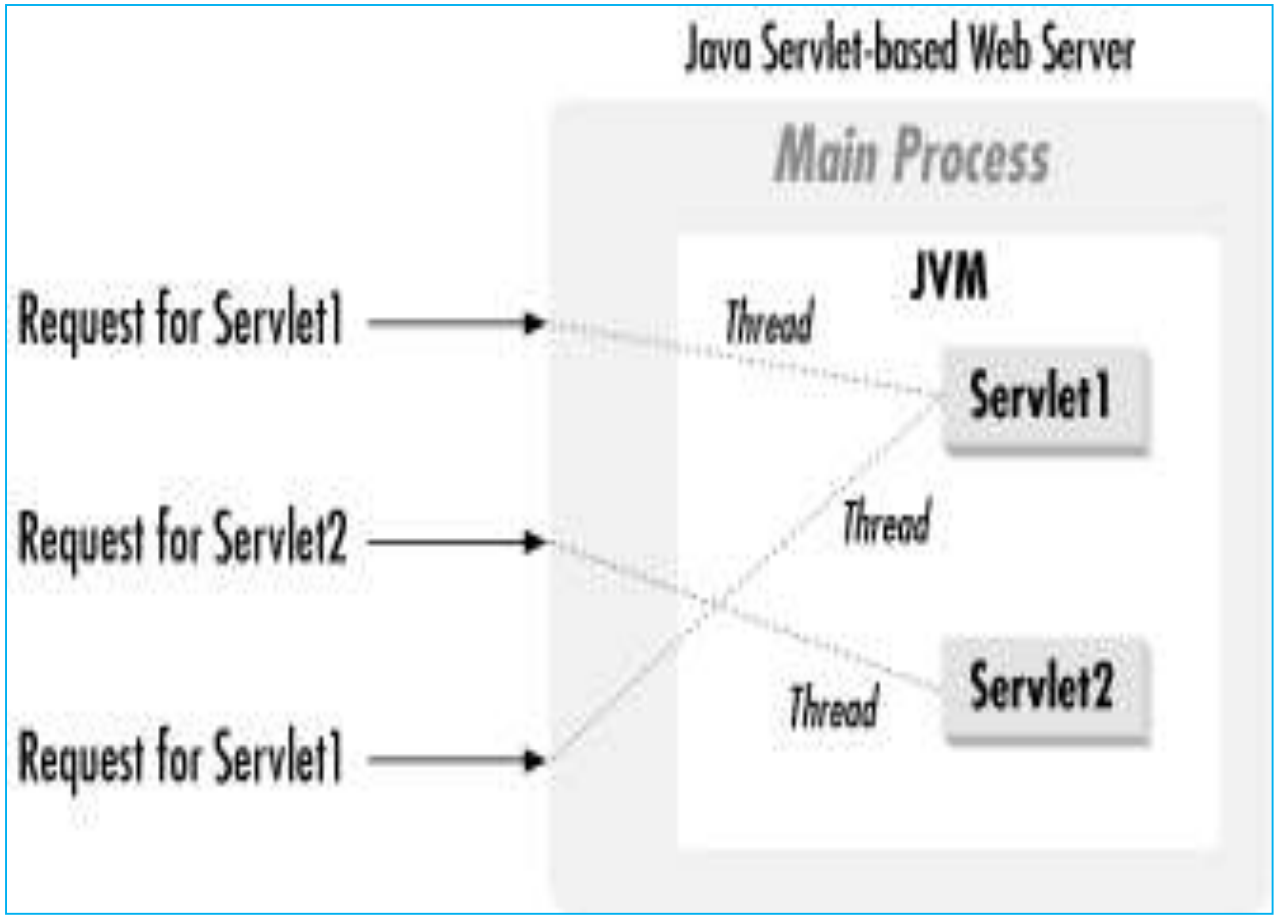


Servlet lifecycle



Servlet lifecycle





Konkurentni pristup servletu

Objekti Servlet klasa

- za svaku servlet klasu instancira se tačno jedan objekat koji opslužuje sve klijente
- njegove doGet() i doPost() metode mogu biti istovremeno pozvane iz više programskih niti Web servera
- atributi predstavljaju potencijalni problem, pošti ih dele niti
  - postoje "sigurni" repozitorijumi u koje će se smeštati deljene stvari: aplikacija, sesija, strana i zahtev

HTTP request i HTTP Response

Preuzimanje podataka iz forme ukoliko se šalje fajl

- Obrada parametara forme je drugačija ukoliko se kroz fomu šalje i datoteka
- Tada je neophodno pristupiti telu HTTP zahteva i preuzeti delove koji su označeni sa boundary vrednošću definisanoj u okviru Content-Type atributa HTTP zaglavlja.

Primer05 - Prihvatanje parametara datoteka