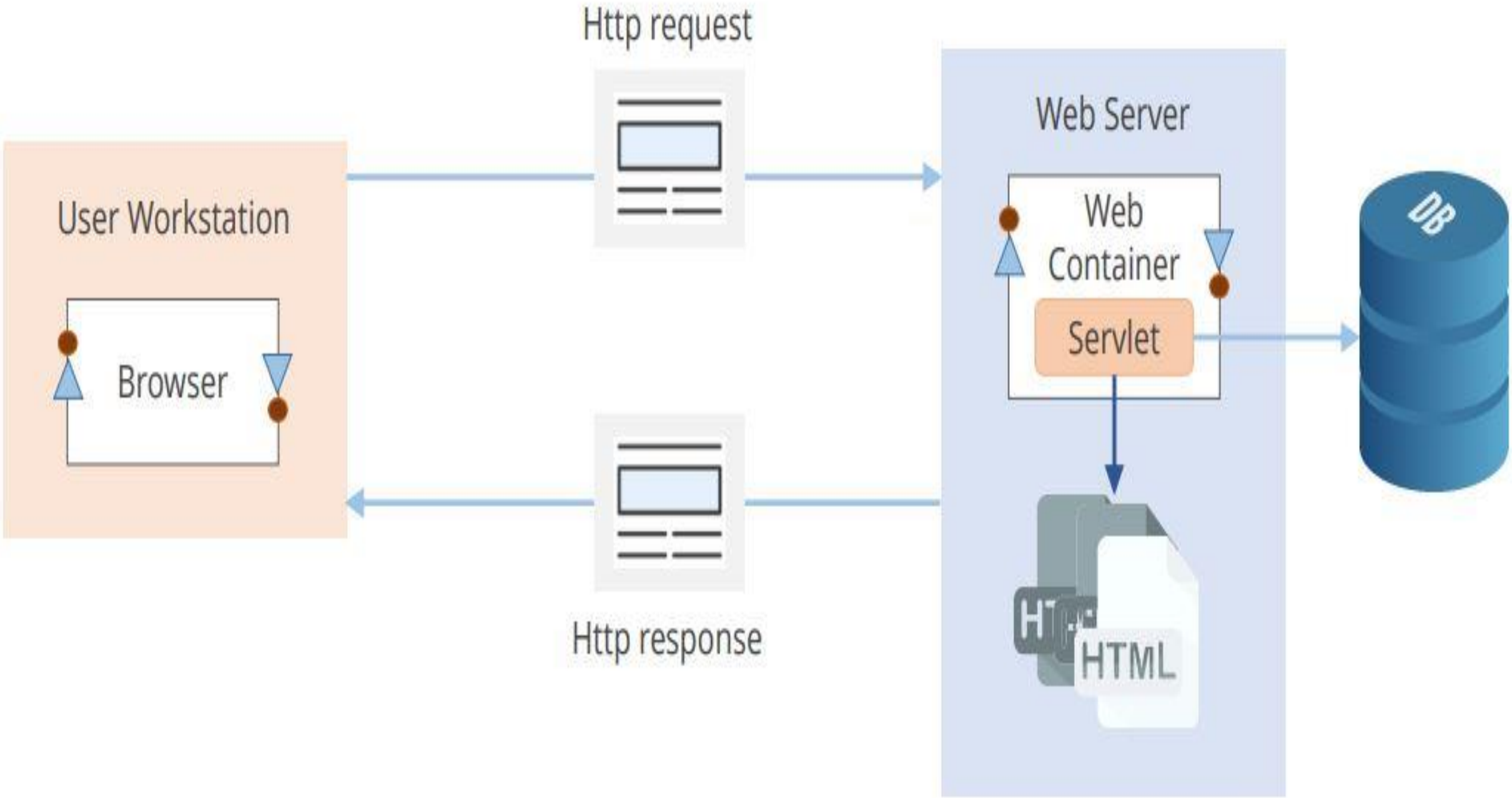
Servleti

Dražen Drašković
Sanja Delčev
Jelica Cincović

Oktobar 2022

Jakarta EE

- Pre: Java Enterprise Edition (Java EE)
- 2017. Oracle prebacuje Java EE Eclipse-u
- Proširenje Java SE 8 za rad sa distribuiranim programiranjem i veb servisima


Http request
User Workstation
Browser
Web Server
Web
Container
Servlet
DB
HTML
Http response

Arhitektura

- **Server** je računar (ili softver) koji prihvata zahteve od klijenta
  - Karakteristični zahtevi: slanje fajla na server (upload) ili preuzimanje fajla sa servera (download), slanje e-mail poruke, dohvatanje informacija iz baze podataka
- **Klijent** može biti veb pretraživač ili neki drugi softver koji šalje zahteve
- Obično je klijent naš računar, a server je tzv. veliki računar
  - Svaki računar može biti server
  - Na jednom računaru mogu biti podignuti i server i klijent
- Jakarta **Servlet** (Java Servlet) – Java komponente koje proširuju funkcionalnosti servera

Server

- Veb server + Servlet/Web Container
- Aplikativni server

- Primeri:
  - IIS
  - Nginx
  - Apache
  - Tomcat
  - Glassfish

Apache Tomcat vs Glassfish

- Apache
  - 70% veb sajtova na Internetu koristi Apache veb server
  - Softver otvorenog koda
  - Bezbedan
- Tomcat
  - Servlet container – obrađuje servlete, JSP stranice...
  - Softver otvorenog koda
- Mogu se instalirati i nezavisno

- Aplikacioni server
- Verzija 6.2.5 – Februar 2022
- U potpunosti podržava Jakarta EE i samim tim JSF, JSP, servlete...
- Softver otvorenog koda

Portovi (1)

- Port je konekcija između servera i klijenta
  - Portovi se identifikuju pozitivnim celim brojevima
- Neki servisi su vezani za određeni port
  - 21 – FTP, File Transfer Protocol
  - 22 – SSH, Secure Shell
  - 25 – SMTP, Simple Mail Transfer Protocol
  - 53 – DNS, Domain Name Service
  - 80 – HTTP, Hypertext Transfer Protocol
  - 443 – HTTPS, Hypertext Transfer Protocol Secure
  - 8080 – HTTP (za testiranje HTTP)

Portovi (2)

Da smo kroz URL prosledili drugi port, naš zahtev bi bio nepoznat

- http://rti.etf.rs/ == http://rti.etf.rs:80/

- Podrazumevano, Veb server osluškuje port 80 za HTTP, odnosno 443 za HTTPS
  - Veb server može osluškivati bilo koji port koji odabere, osim portova koji su već zauzeti
  - Za testiranje servleta, server obično osluškuje port 8080

CGI skripte

- Common Gateway Interface
- Mogu da se napišu u bilo kom programskom jeziku za pisanje skripti
- Definišu interfejs između Veb servera i programa
  - Klijent šalje zahtev serveru
  - Server pokreće CGI skriptu
  - Skripta izračunava rezultat na serveru i završava rad
  - Server vraća odgovor klijentu
  - Klijent šalje drugi zahtev
  - Server ponovo pokreće CGI skriptu…

Servleti (1)

- Java komponente koje proširuju funkcionalnosti servera
  - Klijent šalje zahtev
  - Server pokreće servlet
  - Servlet izračunava rezultat na serveru i NE zatvara se
  - Server vraća odgovor klijentu
  - Klijent šalje drugi zahtev
  - Server poziva servlet koji je već učitan…

Servleti (2)

- Prednosti servleta u odnosu na CGI skripte
  - Pokretanje servleta ne zahteva stvaranje zasebnog procesa svaki put
  - Servlet ostaje u memoriji, tako da ne mora biti stalno učitavan
  - Postoji samo jedna instanca za obradu različitih zahteva, a ne odvojene instance za svaki zahtev
  - Laka koordinacija između servleta pri pravljenju Veb aplikacije

Servleti (3)

- Servlet je bilo koja klasa koja implementira javax.servlet.Servlet interfejs
  - U praksi, većina servleta proširuje javax.servlet.http.HttpServlet klasu
  - Neki servleti proširuju javax.servlet.GenericServlet
- Servletima obično nedostaje main metoda, ali moraju implementirati ili preklopiti neke druge metode

Značajne metode kod servleta (1)

- Kada se servlet prvi put pokrene, poziva se njegova metoda init(ServletConfig config)
  - init treba da uradi potrebnu inicijalizaciju
  - init se poziva samo jednom
- Svaki servlet dobija rezultate pozivajući service(ServletRequest request, ServletResponse response)
  - service poziva drugu metodu, u zavisnosti od tipa usluge koja se zahteva
  - Obično se preklapaju metode od interesa
  - service obrađuje više istovremenih zahteva
- Kada se servlet isključi poziva se metoda destroy()

HTTP zahtevi

- Kada se zahtev podnese sa Veb stranice, on je najčešće **GET** ili **POST**
- HTTP <form> tag ima atribut action, čija vrednost može biti "get" ili "post"

GET	POST
• GET zahtev dodaje parametre u obliku URL ? name1=value1 & name2=value2 & name3=value3 • Imena parametara se mogu pojaviti više od jednom, sa različitim vrednostima • Znak space kod vrednosti parametra se pretvara u znak + • Drugi specijalni karakteri se pretvaraju u hex;	• POST zahtev dodaje parametre u istoj sintaksi, samo je to u “body” delu i korisniku je mnogo teže da vidi • Ako želimo potpunu sigurnost podataka moramo koristiti enkripciju

Značajne metode kod servleta (2)

- Metoda service izvršava sledeće vrste zahteva: DELETE, GET, HEAD, OPTIONS, POST, PUT, i TRACE
  - GET zahtev se izvršava metodom doGet(HttpServletRequest request, HttpServletResponse response)
  - POST zahtev se izvršava preko metode doPost(HttpServletRequest request, HttpServletResponse response)
  - To su dve metode koje se najčešće preklapaju
  - doGet i doPost obično rade isto, tako da možemo da radimo u jednoj pomoćnoj metodi, koju pozivamo iz doGet i doPost

Hello world primer

```
public class HelloServlet extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html><html><head>title>Servlet HelloServlet</title></head>");
            out.println("<body><h1>Hello World</h1></body>");
            out.println("</html>");
        }
    }
    @Override
     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        processRequest(request, response);
     }
}
```

Nadklasa

- public class HelloServlet extends HttpServlet
- Svaka klasa mora proširivati GenericServlet ili potklasu klase GenericServlet
  - Gotovo uvek želimo da odgovorimo na HTTP zahtev, tako da proširujemo klasu HttpServlet
- Potklasa klase HttpServlet mora preklopiti najmanje jednu metodu, doGet, doPost, doPut, doDelete, init i destroy, ili getServletInfo

doGet

- protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
- Ova metoda opslužuje GET zahtev
- Metoda koristi request da prihvati informacije koje su joj poslate
- Metoda nema povratni tip; umesto toga, ona koristi response i daje na standardnom izlazu odgovor
- Metoda može baciti izuzetke tipa IOException
- Bilo koji drugi tip izuzetka treba biti enkapsuliran kao ServletException

Parametri doGet

- Ulaz je HttpServletRequest parametar
  - Naš prvi primer nema nikakav ulaz, pa ćemo ovo videti malo kasnije
- Izlaz je predstavljen HttpServletResponse objektom, koji smo nazvali response
  - I/O u Javi je vrlo fleksibilan, ali i složen, pa ovaj objekat deluje kao “asistent”

Korišćenje HttpServletResponse

- Drugi parametar doGet (ili doPost) je HttpServletResponse response
- Sve poslato putem veba ima “MIME tip”
- Prva stvar koju moramo uraditi sa response je da postavimo MIME tip našeg odgovora:

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

  Ovo govori klijentu da interpretira odgovor kao HTML stranicu
- Na izlazu štampati karaktere i koristimo PrintWriter. Potrebno je pozvati getWriter metodu response: PrintWriter out = response.getWriter();
- Sada smo spremni da kreiramo stranu koja će vratiti povratnu vrednost

Korišćenje PrintWriter

- Pomoću objekta klase PrintWriter, nazvanog out, formiramo Veb stranicu
- Koristimo println metodu objekta out, da kreiramo sadržaj html stranice
- out.println("<!DOCTYPE html><html><head>title>Servlet HelloServlet</title></head>") ...

Deployment Descriptor

web.xml u datom primeru mora imati sledeći sadržaj:

```
<web-app version="4.0" xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd">
   <servlet>
      <servlet-name>HelloServlet</servlet-name>
      <servlet-class>HelloServlet</servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name>HelloServlet</servlet-name>
      <url-pattern>/HelloServlet</url-pattern>
   </servlet-mapping>
</web-app>
```

Ulaz kod servleta – primer GET zahteva (1)

```
public class GetServlet extends HttpServlet {
    processRequest(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet HelloServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Podatak ime:"+request.getParameter("ime")+"<br/></h1>");
```

Ulaz kod servleta – primer GET zahteva (2)

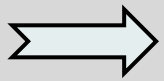
```
            out.println("<form method='get'>");
            out.println("Ime:<input type='text' name='ime'>");
            out.println("<input type='submit' value='Posalji'>");
            out.println("</form>");
            out.println("</body>");
            out.println("</html>");
        }
    }
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

Ulaz kod servleta – primer GET zahteva (3)

localhost:8080/maveproject/
GetServlet?ime=Pera

Prihvatanje parametara (1)

- Ulazni podaci se skupljaju u porukama objekta HttpServletRequest
  - Većina značajnih metoda su nasleđene od nad-interfejsa ServletRequest
- public String getParameter(String name)
  - Vraća vrednost parametra name kao String
  - Ako parametar ne postoji, vraća null
  - Ako više elemenata ima istu name vrednost u formi, metoda vraća vrednost prvog elementa
- public String[] getParameterValues(name)
  - Vraća niz vrednosti parametra name
  - Ako parametar ne postoji, vraća null



```
String names[] =
request.getParameterValues("name");
if (names != null) {
        for (int i = 0; i < names.length; i++)
                out.println(" " + names[i]);
}
```

Prihavtanje parametara (2)

- public Enumeration getParameterNames()
  - Vraća Enumeration imena parametara
  - Ako nema parametara, vraća prazan Enumeration
  - Enumeration je vrlo sličan iteratoru

```
Enumeration e<String> = request.getParameterNames();
while (e.hasMoreElements()) {
        System.out.println(e.nextElement());
}
```

Podaci u String formatu

- Sve vrednosti parametara se preuzimaju kao String
- Često ove Stringove predstavljaju brojevi, ili mi želimo numeričku vrednost
  - int n = Integer.parseInt(param);
  - double d = Double.parseDouble(param);
  - NumberFormatException
- Za char: char c = param.charAt(0);

Primeri

- HelloWorld
- GetServlet
- Sabiranje brojeva
- Prosledjivanje zahteva
- Rad sa sesijom

Pristupanje bazama podataka

- Mnoge veb aplikacije koriste baze podataka. Pristup bazama podataka i programiranje igraju značajnu ulogu u veb razvoju.
- U Javi, tehnologija koja omogućava pristup bazama podataka zove se JDBC (Java Database Connectivity).
- JDBC je aplikativni programski interfejs (API) za programski jezik Java, koji definiše način povezivanja klijenta na relacionu bazu podataka (postoje podrške za različite baze - MySQL, PostgreSQL,...)