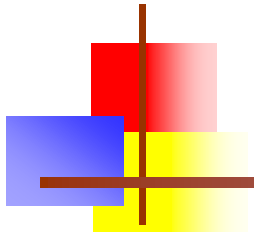
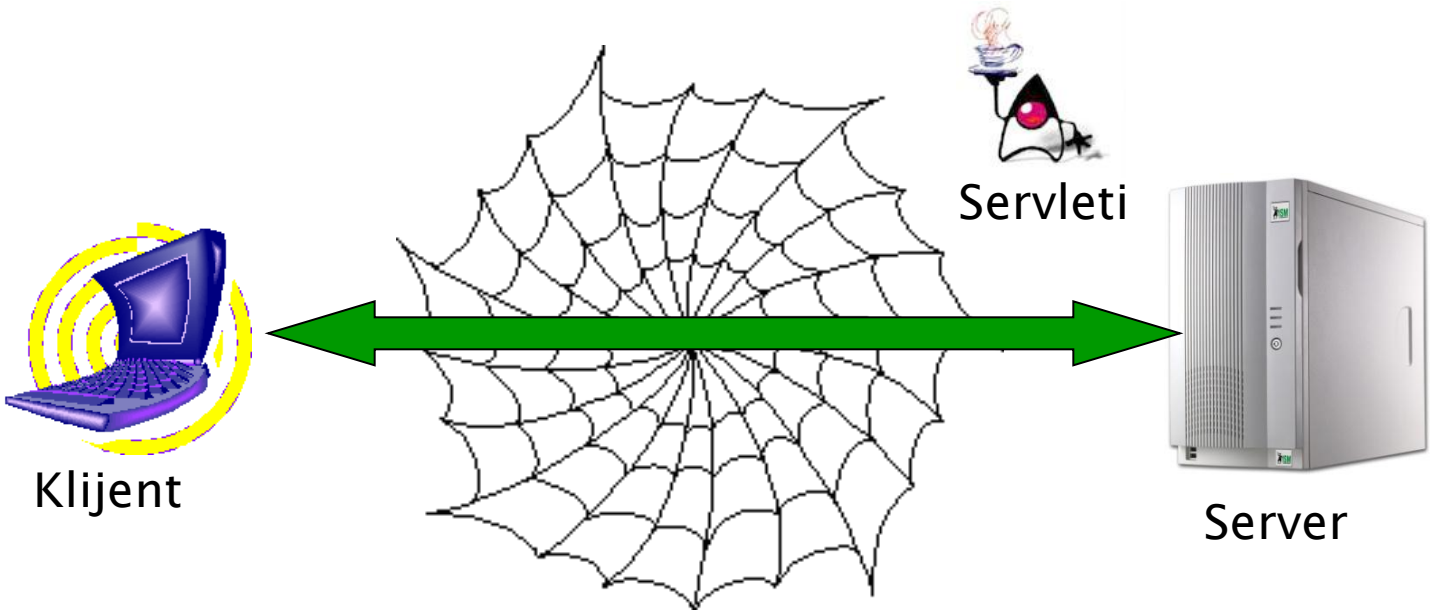


Servleti

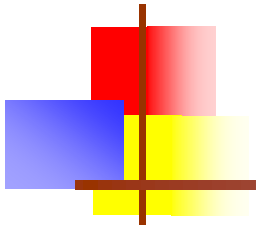
Dražen Drašković, ETF Beograd

Java na Webu: J2EE

- Tanki klijent
- Java “serverska strana”

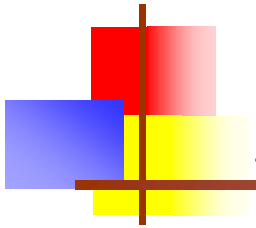


Servlet - delovi Java koda koji rade na serveru



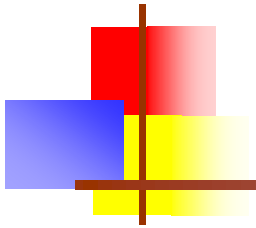
Server

- Server je računar koji prihvata zahteve od klijenta
  - Karakteristični zahtevi: uploadovanje ili downloadovanje fajlova, slanje e-mail poruke,...
- Server je takođe i softver koji prihvate te zahteve; klijent može da bude web pretraživač ili drugi softver koji šalje zahteve
- Obično je klijent naš računar, a server je tzv. veliki računar
  - Svaki računar može biti server
  - Nije neuobičajno da imamo na jednom računaru podignut i server i klijent



Apache

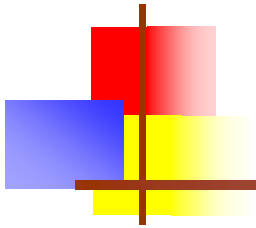
- Apache se često koristi kao server
  - 66% web sajtova na Internetu koristi Apache
- Apache je:
  - Potpuno opremljen i proširiv
  - Uspešan
  - Jasan
  - Bezbedan (mnogo više od drugih)
  - Do danas prati standarde
  - Open source
  - Besplatan
- Zašto koristiti nešto drugo?



Portovi

- Port je konekcija između servera i klijenta
  - Portovi se identifikuju pozitivnim celim brojevima
  - Port je pojam koji se koristi kod softvera, ne kod hardvera, i ima ih mnogo
- Neki servisi su vezani za određeni port
  - Brojevi portova:
    - 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**
    - 8080—**HTTP (used for testing HTTP)**
    - 7648, 7649—CU-SeeMe
    - 27960—Quake III

} Ovo su portovi koji su nama trenutno u fokusu



Portovi II

- Moja Web stranica je:
  http://rti.etf.rs/
- Ali takođe mogu pisati i sa brojem porta:
  http://rti.etf.rs :80/
- Oznaka http: na početku označava određeni protokol (tzv. komunikacioni jezik), Hypertext Transfer Protocol
- Oznaka :80 označava port
- Podrazumevano, Web server osluškuje port 80
  - Web server može osluškivati bilo koji port koji odabere
  - To može dovesti do problema, ako je port bio u upotrebi od strane nekog drugog servera
  - Za testiranje servleta, obično server oslušuje port 8080
- Kod druge URL adrese sam eksplicitno poslao moj zahtev za port 80
  - Da sam poslao po nekom drugom portu, recimo 99, moj zahtev bi bio nepoznat

CGI Scripts

- CGI označava “Common Gateway Interface”
- Definiše interfejs između Web servera i programa

Klijent šalje zahtev serveru

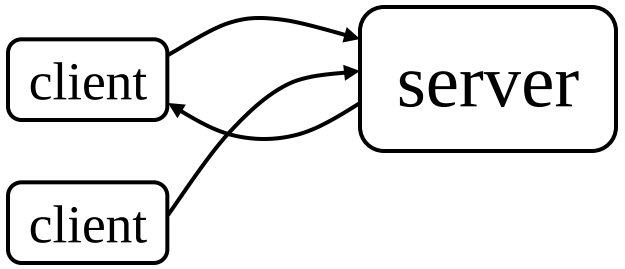
Server pokreće CGI skript

Skript izračunava rezultat na serveru i završava rad

Server vraća odgovor klijentu

Drugi klijent šalje zahtev

Server pokreće CGI skript ponovo
(Novi proces za svaki zahtev!!!)
itd.



Servleti

- Servlet je sličan apletu, ali na serverskoj strani

Klijent šalje zahtev serveru

Server pokreće servlet

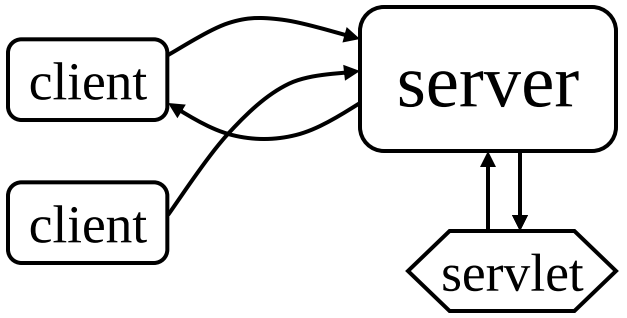
Servlet izračunava rezultat na serveru i *ne zatvara se*

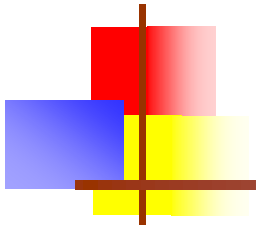
Server vraća odgovor klijentu

Drugi klijent šalje zahtev serveru

Server poziva servlet ponovo. Servlet se učitava jednom.

Itd.

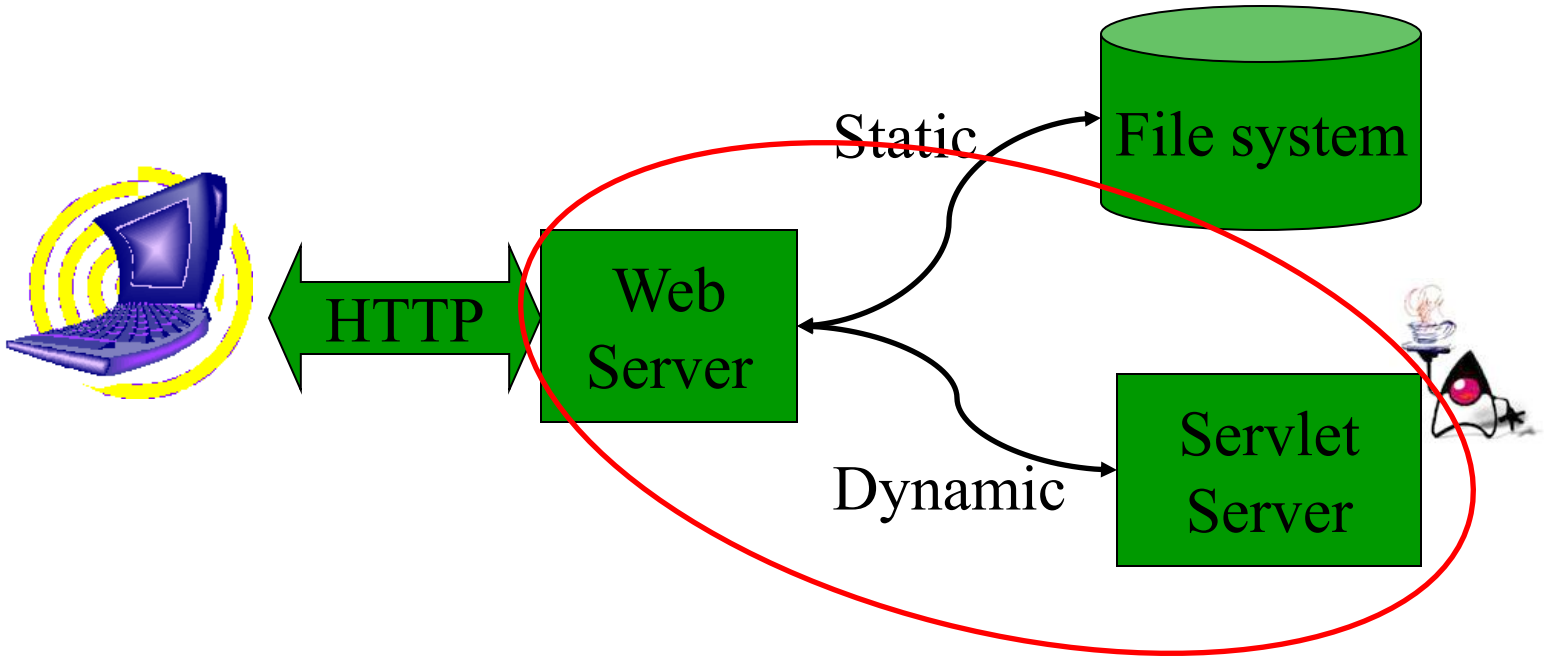




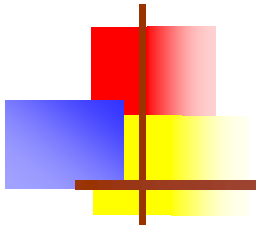
Servlets vs. CGI scripts

- Prednosti:
  - 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 Web aplikacije
- Nedostaci:
  - Smanjen izbor jezika (CGI skripte mogu biti u bilo kom programskom jeziku)

Where are Servlets?

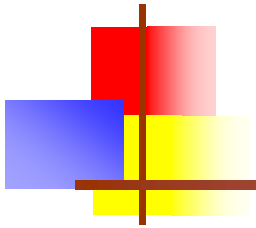


Tomcat = Web Server + Servlet Server



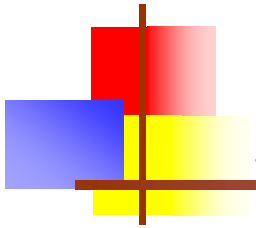
Tomcat

- Tomcat je Servlet Engine koji obrađuje zahteve servleta kod Apache
  - Tomcat je “pomoćna aplikacija” za Apache
- Apache može obrađivati razne vrste Internet usluga
  - Apache se može instalirati bez Tomcat
  - Tomcat se može instalirati bez Apache
- Lakše je instalirati Tomcat samostalno, nego kao deo Apache
  - Tomcat može obrađivati Web stranice, servlete, JSP stranice
- Apache i Tomcat su open source (dakle besplatni)



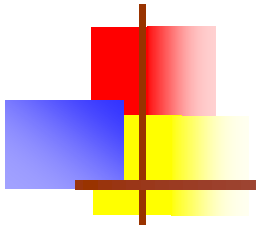
Servleti

- 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, kao i apletima, obično nedostaje main metoda, ali moraju implementirati ili preklopiti neke druge metode



Značajne metode kod servleta, I

- 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 ćete preklopiti metode od interesa
  - service obrađuje više istovremenih zahteva
- Kada se servlet isključi, zove se metoda destroy()
  - destroy se poziva samo jednom



HTTP requests

- Kada se zahtev podnese sa Web stranice, on je gotovo uvek GET ili POST zahtev
- HTTP <form> tag ima atribut action, čija vrednost može biti "get" ili "post"
- Akcija "get" daje rezultate u okviru podataka koji se stavljanju nakon znaka ? u URL adresu
  - Primer: http://www.google.com/search?hl=en&ie=UTF-8&oe=UTF-8&q=servlets
  - & odvaja različite parametre
  - Samo ograničena količina informacija može biti poslata na taj način
- "put" može poslati velike količine podataka

Značajne metode kod servleta, II

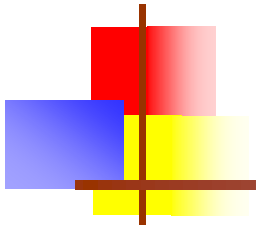
- Metoda service izvršava sledeće vrste zahteva: DELETE, GET, HEAD, OPTIONS, POST, PUT, i TRACE
  - GET zahtev se izvršava metodom doGet(HttpServletRequest request, HttpServletResponse response)
  - POST zahtev se izvršava preko metode doPost(HttpServletRequest request, HttpServletResponse response)
  - To su dve metode koje ćete najčešće preklapati
  - doGet i doPost obično rade isto, tako da možemo da radimo u jednoj metodi, a da pozovemo drugu metodu
  - public void doGet(HttpServletRequest request,
    HttpServletResponse response) {
    doPost(request, response);
    }

“Hello World” servlet

(i kako radi servlet korišćenjem NetBeans)

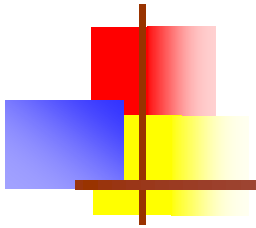
```
public class HelloServlet extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String docType =
      "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
      "Transitional//EN\">\n";
    out.println(docType +
                "<HTML>\n" +
                "<HEAD><TITLE>Hello</TITLE></HEAD>\n" +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1>Hello World</H1>\n" +
                "</BODY></HTML>");
  }
}
```

Ne brinite, ovo je tek početak ☺



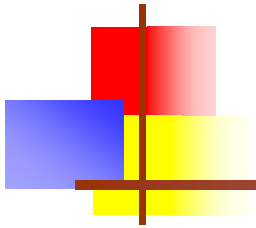
Nadklasa

- public class HelloServlet extends HttpServlet {
- Svaka klasa mora proširivati GenericServlet ili potklasu klase GenericServlet
  - GenericServlet je “protokolski nezavisan”, tako da se može napisati servlet za proces bilo kog protokola
  - Gotovo uvek želimo odgovoriti na HTTP zahtev, tako da proširujemo klasu HttpServlet
- Podklasa klase HttpServlet mora preklopiti najmanje jednu metodu, doGet, doPost, doPut, doDelete, init i destroy, ili getServletInfo



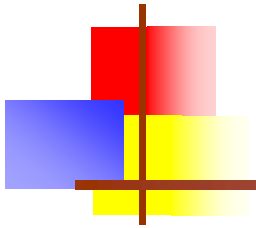
doGet metod

- public 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 izuzetaka treba biti enkapsuliran kao ServletException
- doPost metoda radi zapravo na isti način



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 Weba ima "MIME tip"
- Prva stvar koju moramo uraditi sa response je da postavimo *MIME tip* našeg odgovora:
  response.setContentType("text/html");
  - Ovo govori klijentu da interpretira kao HTML stranicu
- Zato što ćemo na izlazu štampati karaktere, we need a PrintWriter, potrebno je pozvati getWriter metodu response:
  PrintWriter out = response.getWriter();
- Sada smo spremni da kreiramo stranu koja će vratiti povratnu vrednost

Korišćenje PrintWriter

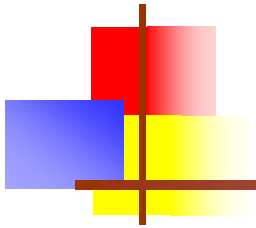
- Pomoću objekta klase PrintWriter, nazvanog out, formiramo Web stranicu
- Na početku kreiramo string za zaglavlje:

```
String docType =
    "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
    "Transitional//EN\">\n";
```

  - Ova linija je tehnički potrebna za HTML
  - Browseri nisu osetljivi, ali HTML validatori *jesu*
- Posle koristiti println metod objekta out jednom ili više puta:

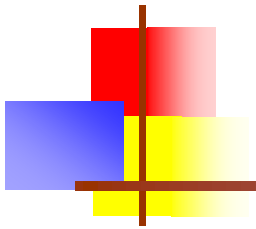
```
out.println(docType +
            "<HTML>\n" +
            "<HEAD> ... </BODY></HTML>");
```

- I gotovi smo!



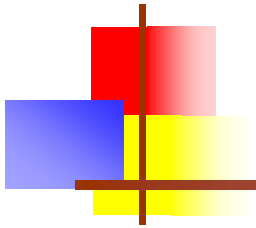
Ulaz kod servleta

- GET zahtev dodaje parametre u obliku
  URL ? name1=value1 & name2=value2 & name3=value3
  - (Space znakovi dodati su zbog čitljivosti!)
  - Znak space kod vrednosti parametra se pretvara u znak **+** (primer: probajte da tražite 2-3 reči na pretraživaču Google)
  - Drugi specijalni karakterise pretvaraju u hex; na primer, znak ampersant se prikazuje kao %26
- Imena parametara se mogu pojaviti više od jednom, sa različitim vrednostima
- POST zahtev dodaje parametre u istoj sintaksi, samo je to u "body" delu i korisniku je mnogo teže da vidi



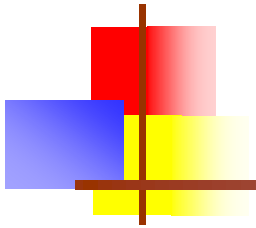
Primer GET zahteva (1)

```
public class DemoServlet extends HttpServlet {


    protected void doGet(HttpServletRequest request,
  HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>DemoServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("Ispis: " + request.getQueryString()
  + "<BR>");
```



Primer GET zahteva (2)

```
    out.println("<form method=get>");
    out.println("<br>Ime: <input type=text name=Ime>");
     out.println("<br>Prezime:
                  <input type=text   name=Prezime>");
    out.println("<br> <input type=submit value=Potvrdi>");
    out.println("</form>");
    out.println("</body>");
    out.println("</html>");
    out.close();
}
```



Ispis: null

Ime:

Prezime:

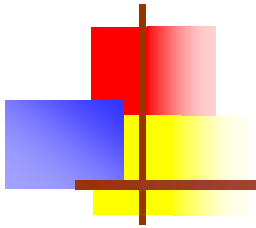
Potvrdi

Ispis: Ime=Drazen&Prezime=Draskovic

Ime: Drazen

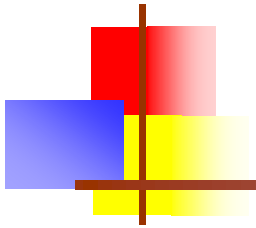
Prezime: Draskovic

Potvrdi



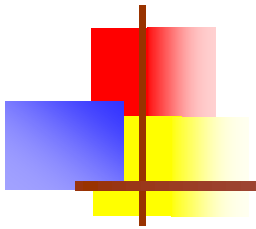
Prihvatanje parametara

- Ulazni podaci se skupljaju u porukama objekta HttpServletRequest
  - Većina značajnih metoda su nasleđene od nad-interfejsa ServletRequest
- public Enumeration getParameterNames()
  - Vraća Enumeration imena parametara
  - Ako nema parametara, vraća prazan Enumeration
- public String getParameter(String name)
  - Vraća vrednost parametra name kao String
  - Ako parametar ne postoji, vraća null
  - Ako name ima više vrednosti, vraća prvu
- public String[] getParameterValues(name)
  - Vraća niz vrednosti parametra name
  - Ako parametar ne postoji, vraća null



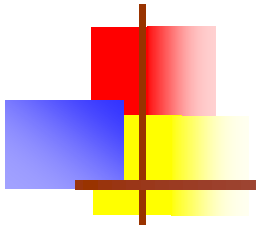
Enumeration pregled

- Enumeration je vrlo sličan Iterator-u
- Primer korišćenja:
  - Enumeration e = myVector.elements();
    while (e.hasMoreElements()) {
        System.out.println(e.nextElement());
    }



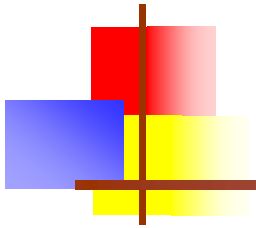
Primer ulaznih parametara

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response) {
 ... izostavljene stvari ...

    out.println("<H1>Hello");
      String names[] =
            request.getParameterValues("name");
      if (names != null)
        for (int i = 0; i < names.length; i++)
          out.println(" " + names[i]);
      out.println("!");
}
```



Java review: Data from Strings

- Sve vrednosti parametra se preuzimaju kao String
- Često ove Stringove predstavljaju brojevi, ili mi želimo numeričku vrednost
  - int n = new Integer(***param***).intValue();
  - double d = new Double(***param***).doubleValue();
  - byte b = new Byte(***param***).byteValue();
    - Slično je za short, float, i long
    - Oni svi mogu baciti izuzetak NumberFormatException, koji je podklasa RuntimeException
  - boolean p = new Boolean(***param***).booleanValue();
- Ali:
  - char c = ***param***.charAt(0);



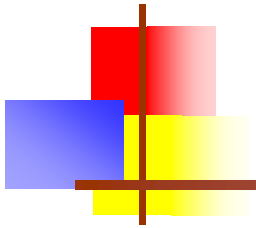
Primer - Prosledjivanje parametara iz HTML forme

- Napraviti HTML stranicu koja u tekstualnom polju prihvata ime korisnika i ima jedno dugme za potvrdu. Kada se dugme pritisne, potrebno je da se pozove servlet koji će ispisati:

  Dobro jutro zelimo korisniku Ime_Korisnika

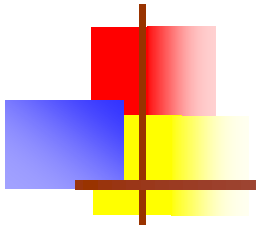
  i

  folder u kome se nalazi dati servlet.



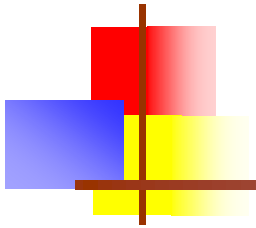
Rešenje - index.html

```
<html>
  <head>
    <title></title>
  </head>
  <body>
  <form name="nemaveze" action="Prvi" method="post">
    <input type="text" name="param" value="Drazen" />
    <input type="submit" value="Potvrdi">
  </form>
  </body>
</html>
```



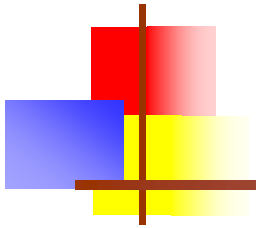
Rešenje - servlet (1)

```
import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class Prvi extends HttpServlet {

     /**Procesi zahtevaju Http get i post metode*/
 protected void processRequest(HttpServletRequest request,
                               HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Prvi</title>");
        out.println("</head>");
```



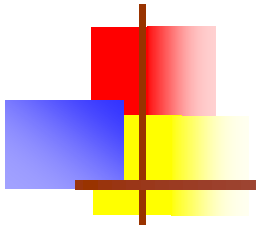
Rešenje - servlet (2)

```
out.println("<body>");
        out.println("<h1>Servlet Prvi se nalazi u " +
                    request.getContextPath () + "</h1>");
        out.println("<h1>Dobro jutro zelimo korisniku " +
                    request.getParameter("param") + "</h1>");
        out.println("</body>");
        out.println("</html>");

        out.close();
    }
```



Rešenje - servlet (3)

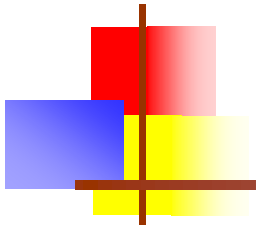
```
protected void doGet(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

protected void doPost(HttpServletRequest request,
HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    /** Metoda koja vraca kraci opis servleta*/
    public String getServletInfo() {
        return "Servlet koji prihvata ime korisnika";
    }
}
```



Deployment Descriptor

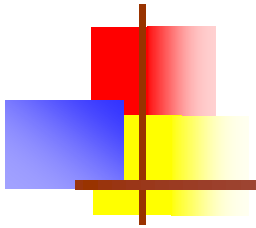
web.xml u datom primeru mora imati sledeći sadržaj:

```
<web-app>
    <servlet>
        <servlet-name>Prvi</servlet-name>
        <servlet-class>Prvi</servlet-class>
    </servlet>
</web-app>
```



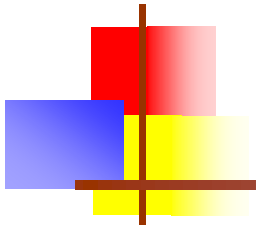
Primer - Prosledjivanje request drugom servletu

```
public class LoginServlet extends HttpServlet {
    private void sendLoginForm(HttpServletResponse response,
boolean withErrorMessage)
    throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Logovanje</title>");
        out.println("</head>");
        out.println("<body>");

        if(withErrorMessage)
            out.println("Greska pri logovanju. Pokusajte ponovo!
<br>");
        out.println("<br>");
        out.println("<br>Unesite vase korisnicko ime i
lozinku.");
```



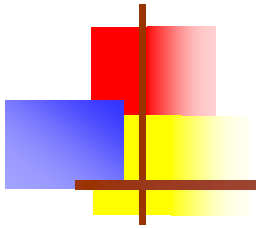
Primer (2)

```
out.println("<br><form method=post");
        out.println("<br>Username: <input type=text
name=userName>");
        out.println("<br>Lozinka: <input type=password
name=password>");
        out.println("<br><input type=submit value=Uloguj se");
        out.println("</form>");
        out.println("</body>");
        out.println("</html>");
    }

 protected void doGet(HttpServletRequest request,
                      HttpServletResponse response)
    throws ServletException, IOException {
        sendLoginForm(response, false);
    }
```



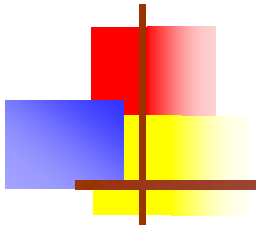
Primer (3)

```
protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
    throws ServletException, IOException {
        String ime = request.getParameter("userName");
        String lozinka = request.getParameter("password");
        if(ime != null && lozinka != null &&
           ime.equals("jamesbond") && lozinka.equals("007")){

       RequestDispatcher rd =
       request.getRequestDispatcher("WelcomeServlet");

       rd.forward(request, response);
        }
         else {
             sendLoginForm(response, true);
         }
    }
}
```



Primer (4)

```
public class WelcomeServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Welcome</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Dobrodosli na sajt ETF!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```



Pristupanje bazama podataka

Mnoge Web aplikacije koriste baze podataka. Pristup bazama podataka i programiranje igraju značajnu ulogu u Web razvoju.

U Javi, tehnologija koja omogućava pristup bazama podataka zove se JDBC (Java Database Connectivity).