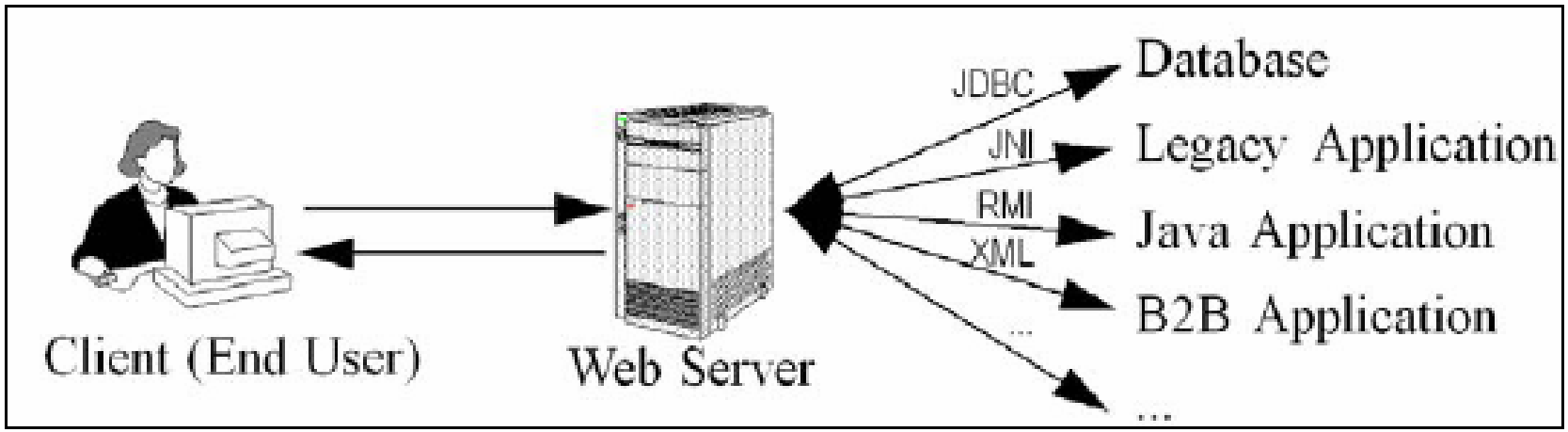
Programiranje Internet aplikacija
Servleti

Uvod

- Osnovna struktura servleta
- Primer jednostavnog servleta koji generiše tekst
- Primer servleta koji generiše HTML
- Servleti i paketi
- Neki alati koji pomažu pri geneisanju HTML koda
- Životni ciklus servleta
- Strategije debagovanja servleta

Posao servleta

- Primanje i čitanje eksplicitnih podataka poslatih od strane klijenta (podaci sa forme)
- Primanje i čitanje implicitnih podataka poslatih od strane klijenta (heder zahteva)
- Generisanje rezultata
- Slanje eksplicitnih podataka klijentu (HTML)
- Slanje implicitnih podataka klijentu (kodovi statusa i heder odgovora)



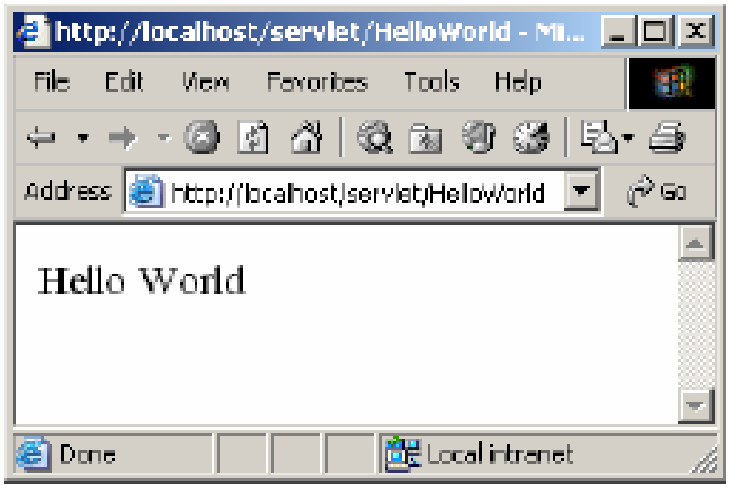
Primeri korišćenja servleta

- Servleti mogu da se izvršavaju na mnogim različitim serverima, jer servlet API, koji se koristi za pisanje servleta, ne koristi ništa iz serverskog okruženja ili protokola. Servleti su postali sastavni deo skoro svih HTTP servera koji na taj način podržavaju servlet tehnologiju.
- Komunikacija između korisnika. Servlet može da obradi više zahteva konkurentno i da ih ujedno sihronizuje. Na ovaj način servleti podržavaju sisteme kao što su on-line konferencije
- Prosleđivanje zahteva. Servleti mogu da proslede zahteve ostalim serverima i servletima. Na taj način se odražava bolji balans opterećenja nekoliko servera koji imaju istu funkciju.

Primeri jednostavnog servleta

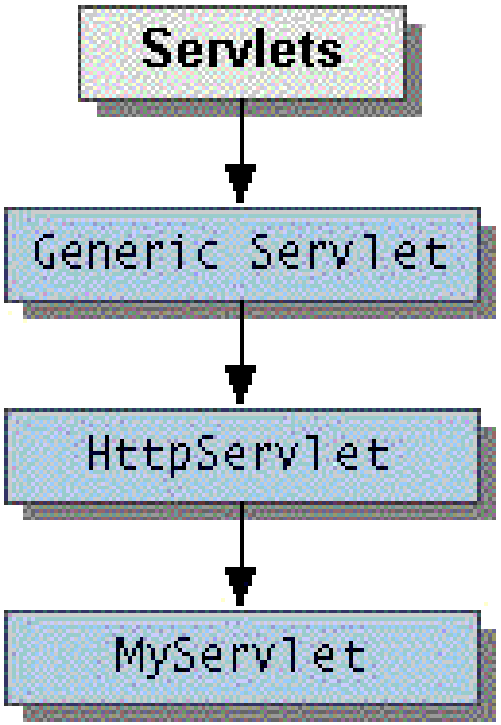
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    out.println("Hello World");
  }
}
```



Arhitektura Servlet Paketa

- Servleti koriste paket javax.servlet, koji sadrži interfejs i klase za njihovo pisanje
- Svi servleti implementuju interfejs Servlets, ili češće nasleđuju klasu koja je većimplementirala ovaj interfejs.
- Najkorišćenija klasa ove vrste je klasa HttpServlet.



Interakcija sa klijentom

- Kada servlet dobije poziv od strane klijenta, on prihvata dva objekta
- ServletRequest, koji održava komunikaciju od klijenta do servera.
- ServletResponse, koji održava komunikaciju od servera ponovo ka klijentu.
- ServletRequest and ServletResponse su interface definisani u javax.servlet paketu

ServletRequest

- ServletRequest interfejs dozvoljava servletu da:
- Pristupi informacijama (imena parametara) koje šalje klijent, protokol koji koristi klijent, i ime udaljenog računara koji je poslao zahtev koji je server primio.
- ServletInputStream - servleti dobijaju podatke od klijenta koji su poslati preko određenog protokola.
- Primer HTML koda

```
<FORM name=”MojaPostForma” action=”ProbaServlet”
    method=POST>
 </FORM>
```

- ili

```
<FORM name=”MojaGetForma”
    action=”ProbaServlet?par1=88” method=GET>
 </FORM>
```

- HttpServletRequest interface sadrži metode koje pristupaju HTTP header informacijama

ServletResponse

- **ServletResponse interfejs omogućava servletima metode za generisanje odgovora:**
- **Dozvoljava servletu da definiše sadržaj odgovora i tip podataka (MIME).**
- **Obezbeđuje izlazni stream, ServletOutputStream i Writer preko kojih servlet šalje podatke.**

Dodatne mogućnosti HTTP Servleta

- **HTTP servleti imaju dodatne objekte koji omogućavaju rad sa sesijama.**
- **Može se sačuvati stanje promenljivih između više poziva sa klijentske i serverske strane.**
- **HTTP servleti imaju i objekte koji omogućavaju rad sa cookie.**

Generisanje HTML koda

- Potrebno je obavestiti Tell the browser da se šalju podaci koji su tipa HTML

```
response.setContentType("text/html");
```

- Menja se naredba println da ni se dobio korektna Web stranica
- Naredbe print koriste HTML tagove
- Provera dobijenog HTML koda sa formalnim sintaksnim validatorima
- http://validator.w3.org/
- http://www.htmlhelp.com/tools/validator/

Generisanje HTML koda

```
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</H1>\n" +
"</BODY></HTML>");
}
}
```

Generisanje HTML koda

```
public class PrviServlet extends HttpServlet
    {
public void doGet (HttpServletRequest zahtev,
                   HttpServletResponse odgovor)
        throws ServletException, IOException
        {
       PrintWriter              out;
      String title = "Jednostavan odgovor Servera";
      odgovor.setContentType("text/html");
      out = odgovor.getWriter();
          out.println("<HTML><HEAD><TITLE>");
       out.println(title);
       out.println("</TITLE></HEAD><BODY>");
       out.println("<H1>" + title + "</H1>");
       out.println("<P>Ova strana je generisana pomocu
  Servleta.");
       out.println("</BODY></HTML>");
       out.close();
        }
    }
```

Generisanje HTML koda

- PrviServlet nasleđuje klasu HttpServlet, koja je implementirala Servlet interface.
- PrviServlet definiše svoj doGet metod. Ovaj metod se poziva kada klijent generiše GET zahtev i kao rezultat se jednostavna HTML stranica vraća klijentu.
  Zahtevi od strane klijenta su sadržani u HttpServletRequest objektu.
  - Odgovor klijentu je u HttpServletResponse.
- Zato što se tekstualni podaci vraćaju klijentu, odgovor je poslat pomoću objekta Writer koji je sadržan u okviru HttpServletResponse objekta.

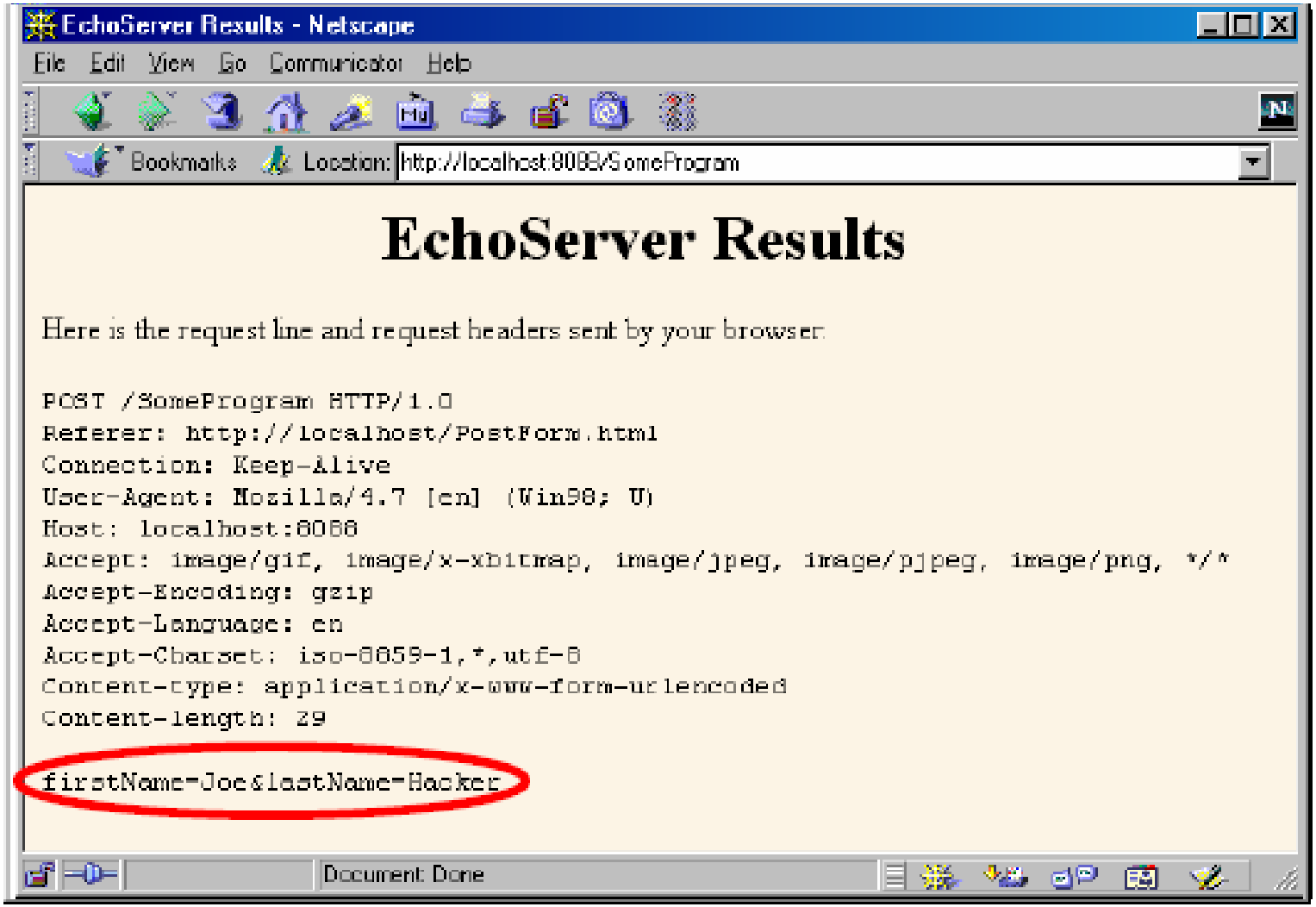
HttpServletRequest

- HttpServletRequest objekat omogućava pristup HTTP header podacima
- Omogućava prihvatanje argumenata koje klijent šalje kao deo svog zahteva.
- Podacima koje šalje klijent može se pristupiti pomoću getParameter metoda, koji vraća vrednost imenovanog parametra.
- Ako parametar ima više od jedne vrednosti, koristi se metod getParameterValues.
- Metod getParameterValues vraća niz vrednosti za imenovani parametar.
- Metod getParameterNames vraća imena svih parametera.

HttpServletResponse

- Objekat HttpServletResponse daje dve mogućnosti za vraćanje podataka klijentu
- Metod getWriter vraća Writer
- Metod getOutputStream vraća ServletOutputStream
- Metoda getWriter se koristi kada su podaci koji svraćaju klijentu tekstualnog tipa, a metod getOutputStream za binarne podatke.
- Pozivanje metoda close ovih objekata nakon slanja odgovora omogućava serveru da zna kada je odgovor komplentiran

GET i POST zahtev



GET zahtev

GET zahtev se obrađuje preklapanjem metoda doGet.

```
public class ObradaGetMetodaServlet extends HttpServlet
   {
public void doGet (HttpServletRequest zahtev,
   HttpServletResponse odgovor)throws ServletException,
   IOException
       {        ...
   odgovor.setContentType("text/html");
            PrintWriter out = odgovor.getWriter();
   out.println("<html>" +
      "<head><title>Primer citanja vrednosti
      parametra</title></head>" +          ...);
   String servletPar1 = zahtev.getParameter("par1");
            if (servletPar1 != null) {
   out.println("<body>" + servletPar1);
            }
            out.println("</body></html>");
            out.close(); }
        ...}
```

POST zahtev

```
public class ObradaPostMetodaServlet extends
   HttpServlet {
    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
   throws ServletException, IOException
    {
        ...
      response.setContentType("text/html");
        PrintWriter out = response.getWriter();
     out.println("<html>" + "<head><title>Primer
   citanja vrednosti parametra</title></head>" +
            ...);
String servletPar1 = request.getParameter("par1");
        if (servletPar1 != null) {
            // prikazivanje procitane vrednosti
            out.println("<body>" + servletPar1);
        }
        out.println("</body></html>");
        out.close();}
    ...}
```

Poziv servleta

- **Servleti se mogu pozivati direktno upisom njihove URL putanje unutar browsera.**
- **Format URL putanje generalno zavisi od servera koji se koristi. Primer je za Jakarta Tomcat web server kod koga je je u direktorijumu *TOMCAT_HOME*/webapps instalirana aplikacija:**

(Tomcat) http://ime_mas:port/Dir_Aplik/Ime_Serv

- **(Tomcat) http://localhost:8080/Proba/ObradaPostMetodaServlet**
- **Za Get metod poziva:**

(Tomcat)

http://localhost:8080/Proba/ObradaGetMetodaServlet?par1=88

Poziv servleta sa HTML stranice

```
public class PozivHTMLServlet extends HttpServlet {
    public void doGet (HttpServletRequest request,
                       HttpServletResponse response)
                       throws ServletException, IOException {
        ...
        out.println(... +
                    "<a href=\"" +

  response.encodeURL("/Proba/ObradaPostMetodaServlet") +
                    "\">Poziv servleta</a>     " +
                    ...);
        ...
    }
    ...
}
```

Poziv servleta sa HTML stranice

```
public class PozivHTMLServlet extends HttpServlet {
    public void doGet (HttpServletRequest request,
                       HttpServletResponse response)
                       throws ServletException, IOException {
        ...
        out.println(... +
                    "<form action=\"" +
response.encodeURL("/Proba/ObradaPostMetodaServlet") +
                    "\" method=\"post\">" +
                    ...
                    "<td><input type=\"text\" name=\"par1\"" +
                    "value=\"88\" size=\"19\"></td>" +
                    ...
                    "<td><input type=\"submit\"" +
                    "value=\"Posalji informacije\"></td>" +
                    ...
                    "</form>" +
                    ...);
        out.close();
    }
    ...
}
```

Čitanje podataka sa forme

- **request.getParameter("ime")**
  - Dobija se URL-dekodirana vrednost prvog elementa koji se zove ime
  - Ponaša se isto i za GET i za POST zahteve
  - Rezultat je null ako ne postoji takav parametar u elementima forme
- **request.getParameterValues("ime")**
  - Dobija se niz URL-dekodiranih vrednosti za sve elemente koji se zovu ime
  - Dobija se niz sa jednim elementom, ako se ime pojavljuje samo jednom
  - Rezultat je null ako ne postoji takav parametar u elementima forme
- **request.getParameterNames() ili request.getParameterMap()**
  - Dobija se Enumeration ili Map objekti od poslatih elemenata
  - Uobičajeno je da je rezervisano za debagovanje

Čitanje podataka sa forme

```
Enumeration paramNames = request.getParameterNames();
while(paramNames.hasMoreElements()) {
  String paramName = (String)paramNames.nextElement();
  out.print("<TR><TD>" + paramName + "\n<TD>");
  String[] paramValues =
    request.getParameterValues(paramName);
  if (paramValues.length == 1) {
    String paramValue = paramValues[0];
    if (paramValue.length() == 0)
      out.println("<I>No Value</I>");
    else
      out.println(paramValue);
  } else {
    out.println("<UL>");
    for(int i=0; i<paramValues.length; i++) {
      out.println("<LI>" + paramValues[i]);
    }
    out.println("</UL>");
  }
}
out.println("</TABLE>\n</BODY></HTML>");
```

Životni ciklus servleta

- **init**
- Izvršava se jednom kada se servlet prvi put učitava.
- *Ne* poziva se za svaki zahtev.
- **service**
- Poziva se kod novog thread-a od strane servera za svaki zahtev.
- Ne treba preklapati ovaj metod!
- • **doGet, doPost, doXxx**
- Obrađuju GET, POST, etc. zahteve.
- Preklapanje ovih metoda omogućava željeno ponašanje servleta.
- **destroy**
- Poziva se kada server briše instancu servleta.
- *Ne* poziva se posle svakog zahteva.

Servlet.service()

- **Svaki poziv servleta se svodi na poziv ove metode**
- **GET, HEAD, PUT, POST, DELETE, OPTIONS i TRACE**
- **Tipičan scenario poziva:**
  - postavi Content-type HTTP odgovora
  - uzimi PrintWriter ka klijentu
  - kroz PrintWriter šalji dinamički kreiran HTML

HttpServlet.init()

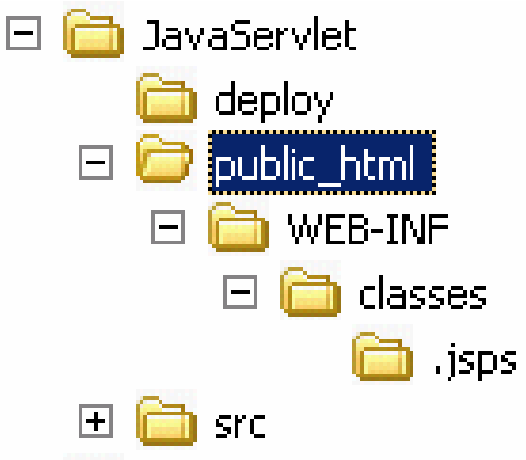
- namijenjena za inicijalizaciju servleta, prije njegove prve upotrebe
- poziva se tačno jednom
- nema prepreke za postojanje konstruktora u servlet klasi u kome će se odvijati dio inicijalizacije, ali je na raspolaganju i init metoda

HttpServlet.destroy()

- **poziva se prilikom uništavanja servleta**
- **namenjena za clean-up zadatke neposredno prije uništenja servleta – oslobađanje resursa koje je servlet zauzimao:**
  - otvorene datoteke, konekcija sa bazom podataka
- **obično prilikom zaustavljanja Web servera**

Servlet Deployment

- **root folder (public_html)**
  - "Polazna tačka" Web aplikacije
  - Sve datoteke i poddirektorijumi su unutar ovog foldera: html, slike...
- **/public_html/WEB-INF/**
  - Sadrži konfiguracione datoteke i kompajlirane klase
  - Nije direktno dostupan putem Web-a
- **/public_html/WEB-INF/classes/**
  - Sve kompajlirane klase (servlet klase i druge klase) se nalaze



Mapiranje servleta

- **Servlet klasa mora biti mapirana - URI**

- **Pristup servletu - general pattern (*invoker servlet*)**
  - http://[domain]/[context]/servlet/[ServletClassName]
  - http://localhost:8988/servletintro/servlet/SimpleServlet

- **Mapiranje korištenjem konfiguracione datoteke *web.xml***
  - Servlet se mapira u URL koji je definisao administrator

web.xml

- **web.xml se nalazi u "WEB-INF" folder**

- **Primer**
  - Servlet klasa
    - **HelloWorld.class**
  - Kontekst aplikacije:
    - **http://localhost:8988/servletintro/**
  - Invoker class mapiranje
    - **http://localhost:8988/servletintro/servlet/HelloWorld**
  - Mapiranje putem web.xml datoteke
    - **http://localhost:8988/servletintro/hello**

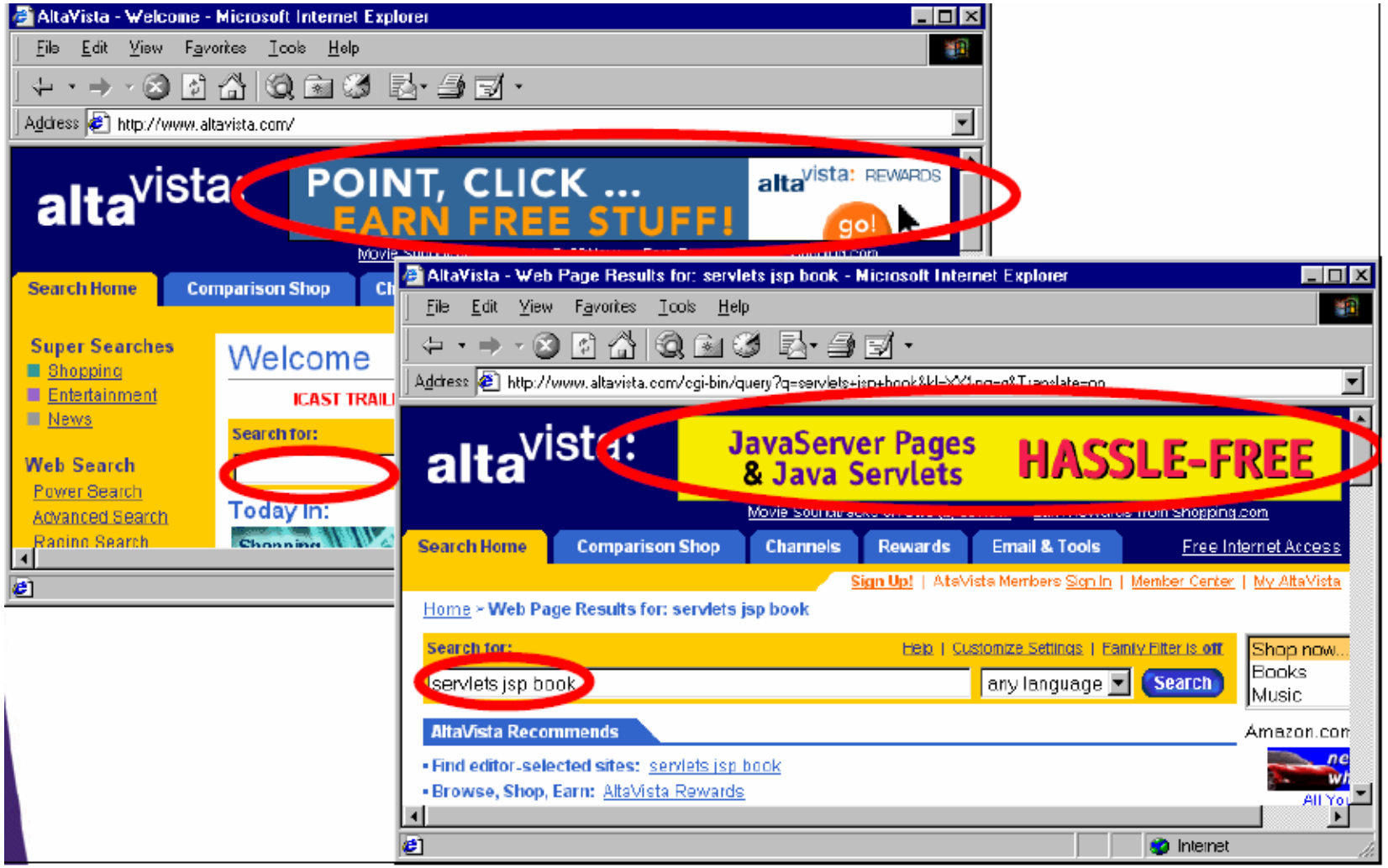
```
<servlet>
    <servlet-name>HelloW</servlet-name>
    <servlet-class>HelloWorld</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>HelloW</servlet-name>
    <url-pattern>hello</url-pattern>
</servlet-mapping>
```

Mogućnosti rada sa cookie-ijima

- **Ideja**
  - – Servlet generiše jedinstveno ime i vrednost klijentu.
  - – Klijent vraća isto ime i vrednost kada se ponovo konektuje na isti sajt (ili isti domen u zavisnosti od podešavanja vrednosti cookie-ija).
- **Tipična upotreba cookie-ija**
  - – Identifikacija korisnika tokom e-commerce sesije
  - • Servlet imaju API na višem nivou za ovaj zadatak
  - – Izbegavanje korisničkog imena i šifre
  - – Prilagođavanje sajta korisniku
  - – Ciljane reklame

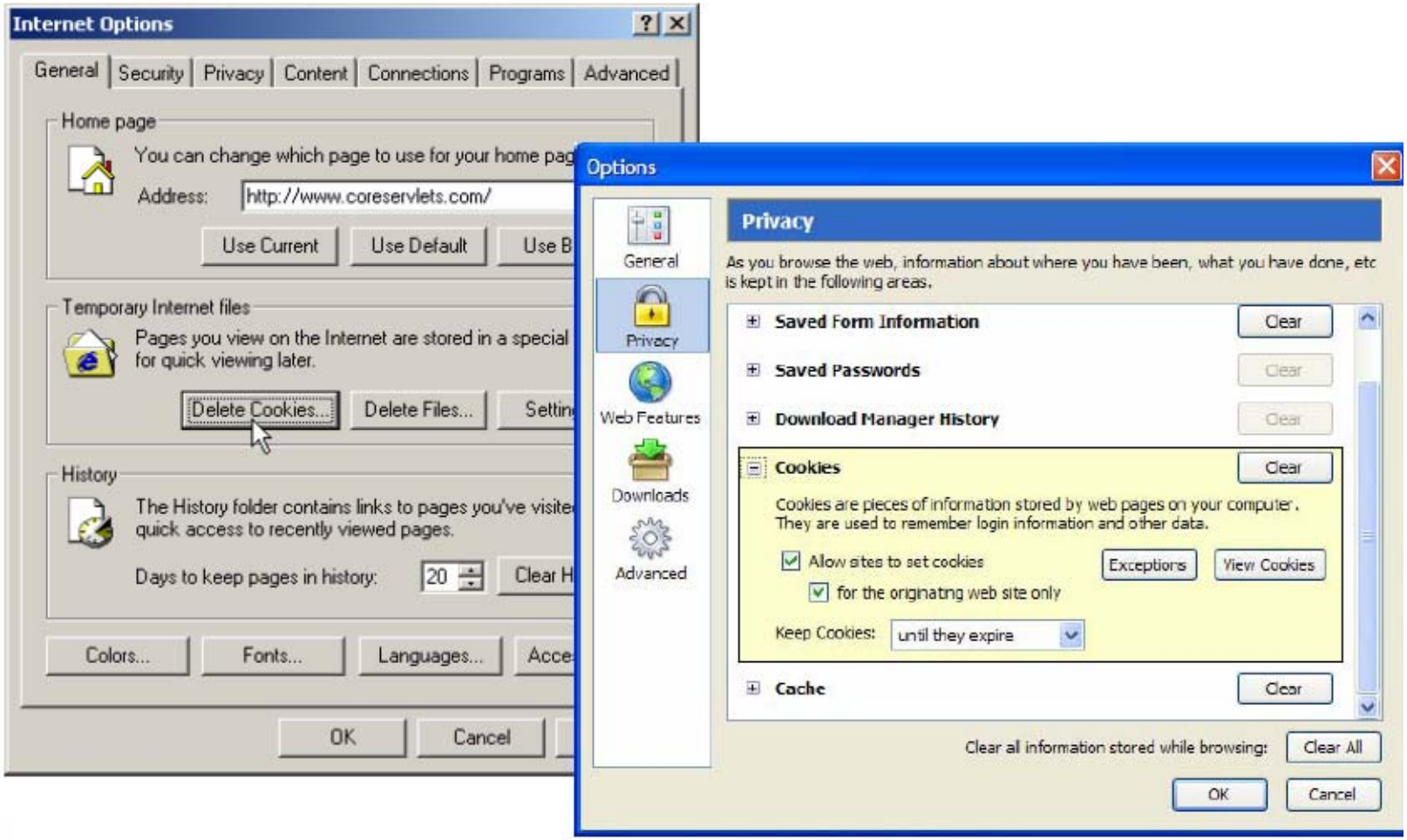
Mogućnosti rada sa cookie-ijima



Problemi rada sa cookie-ijima

- **Problem je privatnost, a ne bezbednost.**
  – Serveri mogu da pamte korisnikove ranije akcije
  – Ako korisnik daje personalne informacije, serveri mogu da povezuju te informacije sa prethodnim akcijama
  – Serveri mogu deliti cookie informacije sa trećom stranom
  – Loše dizajnirani sajtovi smeštaju poverljive informacije, kao što su brojevi kreditnih kartica, direktno u cookie
- **Preporuka**
  – Ako cookie-iji nisu kritični za izvršavanje zadatka, treba izbeći servlete koji u potpunosti prestaju sa radom ako se cookie-iji zabrane
  – Ne smeštati poverljive informacije u cookie-ije

Brisanje cookie-ija



Slanje cookie-ija klijentu

- **Kreira se Cookie objekat.**

  – Poziva se Cookie konstruktor sa imenom cookie-ija i željenom vrednosti, oba argumenta su tipa Strings

  Cookie c = new Cookie("userID", "a1234");
- **Postavlja se maksimalno vreme života cookie-ija.**

  – Da bi se naglasilo čitaču da smesti cookie na disk umesto samo u memorijukoristi se setMaxAge (argument je broj secondi)

  c.setMaxAge(60*60*24*7); // Jedna nedelja
- **Smesti se Cookie u objekat HTTP response**

  – Koristi se response.addCookie.

  – Bez ove naredbe, cookie se neće poslati klijentu!

  response.addCookie(c);

Čitanje cookie-ija

- **Treba pozvati request.getCookies**
  – Dobija se niz objekata tipa Cookie.
- **Treba se kretati po nizu, pozvati metod getName za svaki element niza sve dok se ne pronađe željeni cookie**
  – Nakon toga koristiti upisanu vrednost (pomoću getValue)

```
String cookieName = "userID";
Cookie[] cookies = request.getCookies();
if (cookies != null) {
for(int i=0; i<cookies.length; i++) {
Cookie cookie = cookies[i];
if (cookieName.equals(cookie.getName())) {
doSomethingWith(cookie.getValue());
}
}
}
```

Cookie i pristupanje sajtu prvi put

```
public class RepeatVisitor extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse
    response)
throws ServletException, IOException {
boolean newbie = true;
Cookie[] cookies = request.getCookies();
if (cookies != null) {
for(int i=0; i<cookies.length; i++) {
Cookie c = cookies[i];
if ((c.getName().equals("repeatVisitor"))&& (c.getValue().equals("yes"))) {
newbie = false;
break;
}
}
}
```

Cookie i pristupanje sajtu prvi put

```
String title;
if (newbie) {
Cookie returnVisitorCookie =
new Cookie("repeatVisitor", "yes");
returnVisitorCookie.setMaxAge(60*60*24*365);
response.addCookie(returnVisitorCookie);
title = "Welcome Aboard";
} else {
title = "Welcome Back";
}
response.setContentType("text/html");
PrintWriter out = response.getWriter();
… // (Izlaz sa razlicitim naslovom)
```

Cookie i pristupanje sajtu prvi put

Atributi Cookie-ija

- **getDomain/setDomain**

  – Specificira se domen koji obrađuje cookie. Trenutni host mora biti deo specificiranog domena.
- **getMaxAge/setMaxAge**

  – Čita/postavlja vreme života cookie-ija (u sekundama). Ako se ova vrednost ne postavi, podrayumeva se da je vreme života cookie-ija samo trenutna sesija.
- **getName**

  – Dobija se ime cookie-ija. Ne postoji setName metod, jer se ime definiše u okviru konstruktora. U okviru niza cookie-ija koji se dobija od strane klijenta, metod getName se koristi da bi se pronašao željeni cookie.

Atributi Cookie-ija

- **getPath/setPath**

  – Čita/postavlja putanju koja obrađuje cookie. Ako se ne navede, cookie pripada URLu koji je u okviru ili direktorijum iznad trenutne stranice.
- **getSecure/setSecure**

  – Čita/postavlja flag koji definiše da li se cookie izvršava samo pomoću SSL konkecije ili bilo koje konekcije.
- **getValue/setValue**

  – Čita/postavlja vrednost koja se želi pamtiti u okviru cookie-ija. Za nove cookie-ije, ova vrednost se postavlja u konstruktoru, a ne pomoću setValue. getValue se koristi da bi se dobila upisana vrednost. Ako se postojećem cookie-iju menja vrednost,potrebno je poslati tu novu vrednost sa response.addCookie.

Primer 1

```
public class CookieTest extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
for(int i=0; i<3; i++) {
Cookie cookie =
new Cookie("Session-Cookie-" + i,
"Cookie-Value-S" + i);
// No maxAge (ie maxAge = -1)
response.addCookie(cookie);
cookie = new Cookie("Persistent-Cookie-" + i,
"Cookie-Value-P" + i);
cookie.setMaxAge(3600);
response.addCookie(cookie);
}
```

Primer 1

```
... // Start an HTML table
Cookie[] cookies = request.getCookies();
if (cookies == null) {
out.println("<TR><TH COLSPAN=2>No cookies");
} else {
for(Cookie cookie: cookies) {
out.println
("<TR>\n" +
" <TD>" + cookie.getName() + "\n" +
" <TD>" + cookie.getValue());
}
}
out.println("</TABLE></BODY></HTML>");
}
}
```

Primer 2

```
public class CookieUtilities {
public static String getCookieValue
(HttpServletRequest request,
String cookieName,
String defaultValue) {
Cookie[] cookies = request.getCookies();
if (cookies != null) {
for(Cookie cookie: cookies) {
if (cookieName.equals(cookie.getName())) {
return(cookie.getValue());
}
}
}
return(defaultValue);
}
}
```

Primer 3

```
public class LongLivedCookie extends Cookie {
public static final int SECONDS_PER_YEAR =
60*60*24*365;
public LongLivedCookie(String name, String value) {
super(name, value);
setMaxAge(SECONDS_PER_YEAR);
}
}
```

Primer 4

```
public class RepeatVisitor2 extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
boolean newbie = true;
String value =
CookieUtilities.getCookieValue(request,
"repeatVisitor2",
"no");
if (value.equals("yes")) {
newbie = false;
}
String title;
if (newbie) {
LongLivedCookie returnVisitorCookie =
new LongLivedCookie("repeatVisitor2", "yes");
response.addCookie(returnVisitorCookie);
title = "Welcome Aboard";
} else {
title = "Welcome Back";
}
```

Primer 5

```
public class ClientAccessCounts extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
String countString =
CookieUtilities.getCookieValue(request,
"accessCount",
"1");
int count = 1;
try {
count = Integer.parseInt(countString);
} catch(NumberFormatException nfe) { }
LongLivedCookie c =
new LongLivedCookie("accessCount",
String.valueOf(count+1));
response.addCookie(c);
```

Primer 5

```
out.println(docType +
"<HTML>\n" +
"<HEAD><TITLE>" + title +
"</TITLE></HEAD>\n" +
"<BODY BGCOLOR=\"#FDF5E6\">\n" +
"<CENTER>\n" +
"<H1>" + title + "</H1>\n" +
"<H2>This is visit number " +
count + " by this browser.</H2>\n"+
"</CENTER></BODY></HTML>");
}
}
```

Primer 6

```
public class RegistrationForm extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
String actionURL =
"/servlet/coreservlets.RegistrationServlet";
String firstName =
CookieUtilities.getCookieValue(request,
"firstName", "");
String lastName =
CookieUtilities.getCookieValue(request,
"lastName", "");
String emailAddress =
CookieUtilities.getCookieValue(request,
"emailAddress",
"");
```

Primer 6

```
out.println
(docType +
"<HTML>\n" +
"<HEAD><TITLE>" + title + "</TITLE></HEAD>\n" +
"<BODY BGCOLOR=\"#FDF5E6\">\n" +
"<CENTER>\n" +
"<H1>" + title + "</H1>\n" +
"<FORM ACTION=\"" + actionURL + "\">\n" +
"First Name:\n" +
" <INPUT TYPE=\"TEXT\" NAME=\"firstName\" " +
"VALUE=\"" + firstName + "\"><BR>\n" +
"Last Name:\n" +
" <INPUT TYPE=\"TEXT\" NAME=\"lastName\" " +
"VALUE=\"" + lastName + "\"><BR>\n"+
"Email Address: \n" +
" <INPUT TYPE=\"TEXT\" NAME=\"emailAddress\" " +
"VALUE=\"" + emailAddress + "\"><P>\n" +
"<INPUT TYPE=\"SUBMIT\" VALUE=\"Register\">\n" +
"</FORM></CENTER></BODY></HTML>");
}
}
```

Primer 6

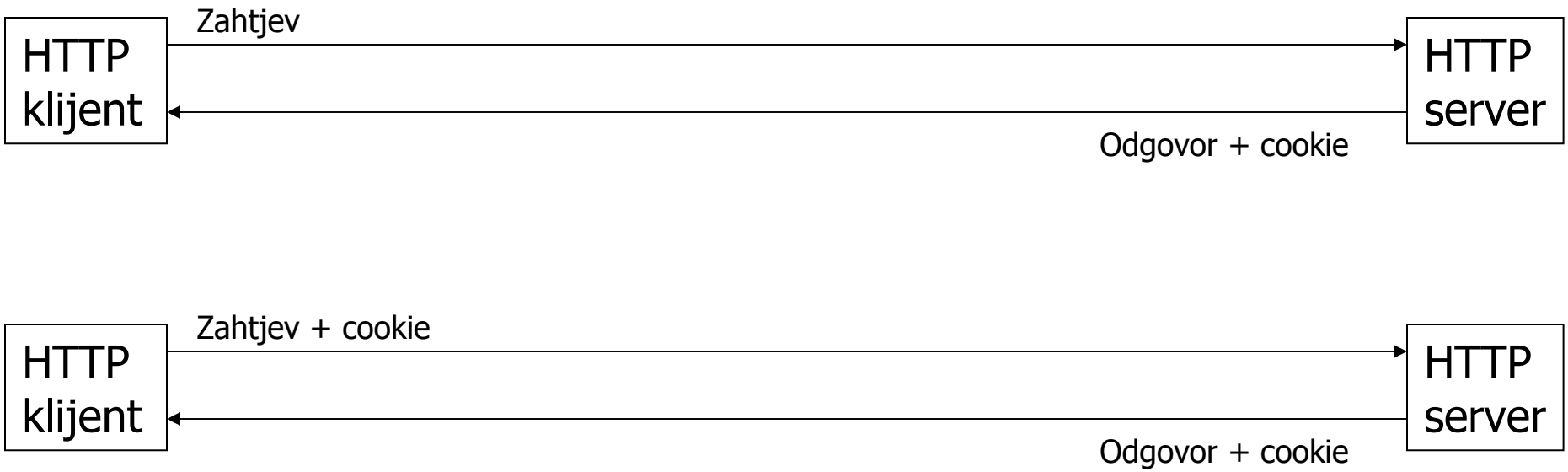
```
public class RegistrationServlet extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
boolean isMissingValue = false;
String firstName =
request.getParameter("firstName");
if (isMissing(firstName)) {
firstName = "Missing first name";
isMissingValue = true;
}
String lastName =
request.getParameter("lastName");
if (isMissing(lastName)) {
lastName = "Missing last name";
isMissingValue = true;
```

Primer 6

```
Cookie c1 =
new LongLivedCookie("firstName", firstName);
response.addCookie(c1);
Cookie c2 =
new LongLivedCookie("lastName", lastName);
response.addCookie(c2);
Cookie c3 = new LongLivedCookie("emailAddress",
emailAddress);
response.addCookie(c3);
String formAddress =
"/servlet/coreservlets.RegistrationForm";
if (isMissingValue) {
response.sendRedirect(formAddress);
} else { … }
```

Praćenje sesije korisnika

- **cookie mehanizam**



Praćenje sesije korisnika - cookie

- **Ideja: povezati određeni cookie sa podacima na serveru**

```
String sessionID = napravitiJedinstveniString();
HashMap sessionInfo = new HashMap();
HashMap globalTable = pronaciUTabeliSesiju();
globalTable.put(sessionID, sessionInfo);
Cookie sessionCookie =
new Cookie("JSESSIONID", sessionID);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```

- **Potrebno je još uraditi:**
  - izvršiti cookie koji smešta identifikator sesije
  - postaviti vreme zadržavanja za cookie
  - povezati hash tabelu sa svakim zahtevom
  - generisati jedinstveni identifikator sesije

Praćenje sesije korisnika – promena URLa

- **Ideja**
  – Na klijentskoj strani dodati novi sadržaj koji identifikuje datu sesiju na kraj svakog URLa
  – Server povezuje poslati podatak sa sesijom koja se izvršava
  – Na primer, http://host/path/file.html;jsessionid=1234
- **Prednosti**
  – Korektno se izvršava i u slučajevima kada su cookie-iji zabranjeni ili nisu podržani
- **Mane**
  – Moraju se menjati svi URL koji se pozivaju stranice sa sajta
  – Sve stranice se moraju dinamički generisati
  – Ne radi korektno za linkove sa drugih sajtova

Praćenje sesije korisnika – elementi forme

- **Ideja:**
- **<INPUT TYPE="HIDDEN" NAME="session" VALUE="...">**
- **Prednosti**

  – Korektno se izvršava i u slučajevima kada su cookie-iji zabranjeni ili nisu podržani
- **Mane**

  – Dosta napornog procesiranja

  – Sve stranice moraju biti rezultat slanja forme

Praćenje sesije korisnika pomoću Jave

- **Objekti sesije se čuvaju na serveru**
- **Sesije su automatski povezane sa klijentom pomoću cookie-ija ili promene URLa**

  – Koristi se request.getSession da bi se dobio objekat sesije

  - U pozadini, sistem pregleda cookie ili URL dodatne informacije i pretražuje da li se ključ poklapa sa nekim od prethodno smeštenih objekata sesije.

  - Ako pronađe poklapanje, kao rezultat vraća pronađeni objekat.

  - Ako ne pronađe, kreira novi, povezuje cookie ili URL informacije sa njegovim ključem i vraća kao rezultat novi objekat sesije.
- **Mehanizmi poput Hashtable dozvoljavaju da se smeštaju neophodni objekti unutar sesije**

  – setAttribute smešta vrednost

  – getAttribute prikazuje vrednost

Praćenje sesije korisnika pomoću Jave

- **Pristup objektu sesije**

  – Pozivom request.getSession dobija se HttpSession objekat

  - To je objekat tipa hashtable povezan sa korisnikom
- **Pretraga informacija povezanih sa sesijom**

  – Poziv getAttribute u okviru HttpSession objekta, prebacuje vrednost u odgovarajući tip i proverava da li je rezultat null.
- **Smeštanje informacije u okviru sesije.**

  – Koristi se setAttribute sa parom ključ, vrednost kao argumentom.
- **Uklanjanje podataka sesije**

  – Poziva se removeAttribute da bi se uklonila specifična vrednost.

  – Poziva se invalidate da bi se uklonila trenutna sesija.

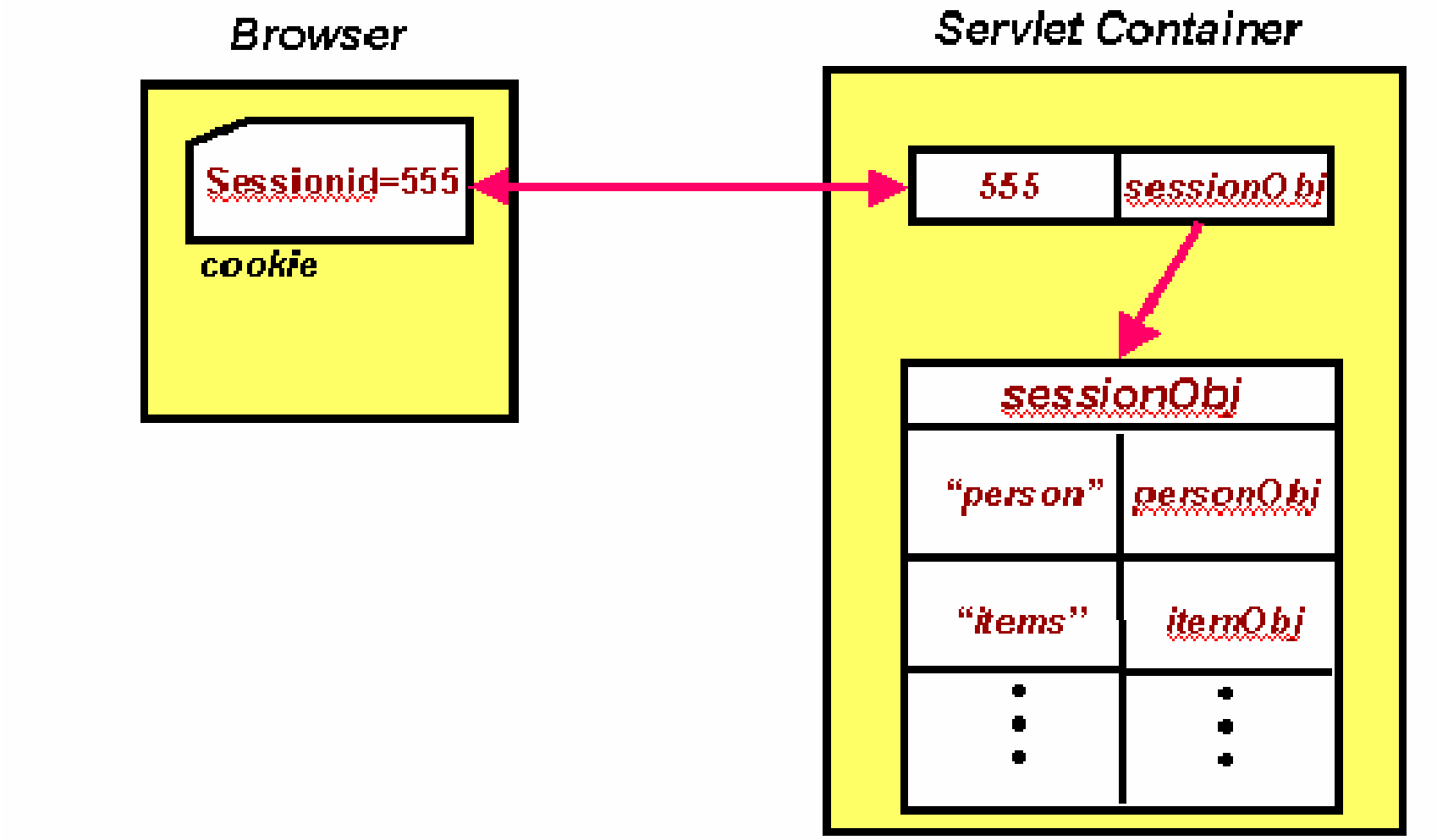
Praćenje sesije korisnika pomoću Jave

```
HttpSession session = request.getSession();
SomeClass value =
(SomeClass)session.getAttribute("someID");
if (value == null) {
value = new SomeClass(...);
session.setAttribute("someID", value);
}
doSomethingWith(value);
```

– Nie potrebno pozivati ponovo setAttribute (nakon promene vrednosti) ako je promenjena vrednost isti objekat.

- Ali ako je vrednost nepromenljiva, tada će promenjena vrednost biti novi objekta i mora se ponovo pozvati setAttribute.

Praćenje sesije korisnika pomoću Jave



HttpSession metode

- **getAttribute**
  – Vraća prethodno smeštenu vrednost iz objekta sesije.
  - Rezultat je null ako nijedna vrednost nije povezana sa datim imenom.
- **setAttribute**
  – povezuje vrednost sa navedenim imenom.
  - vrši se monitoring promena: vrednosti implementiraju HttpSessionBindingListener.
- **removeAttribute**
  – uklanja vrednosti koje su povezane sa navedenim imenom.
- **getAttributeNames**
  – vraća imena svih atributa u okviru sesije.
- **getId**
  – vraća jedinstveni identifikator.

HttpSession metode

- **isNew**
  – Prepoznaje da li je sesija nova za klijenta (ne za *stranicu*)
- **getCreationTime**
  – Vraća vreme kada je sesija prvi put kreirana
- **getLastAccessedTime**
  – Vraća kao rezultat vreme kada je sesija poslednji put poslata klijentu
- **getMaxInactiveInterval, setMaxInactiveInterval**
  – Pročitati ili postaviti vremenski interval kada je sesije bez pristupa i postavlja se kao nevalidna
- **invalidate**
  – Postaviti trenutnu sesiju kao nevalidnu

Praćenje sesije korisnika

```
public void doGet(HttpServletRequest req,
HttpServletResponse res) {
  HttpSession session = request.getSession(true);
  userInfo = new UserInfo();
  session.putValue(session.getId(), userInfo);
  res.setContentType("text/html");
  PrintWriter out = res.getWriter();
  out.println("<HTML>");
  out.println("<HEAD><TITLE>Test</TITLE></HEAD>");
  out.println("<BODY>");
  ...
  userInfo = (UserInfo)session.getValue(session.getId());
  out.close();
}
```

- **Samo session ID se prenosi do korisnika**