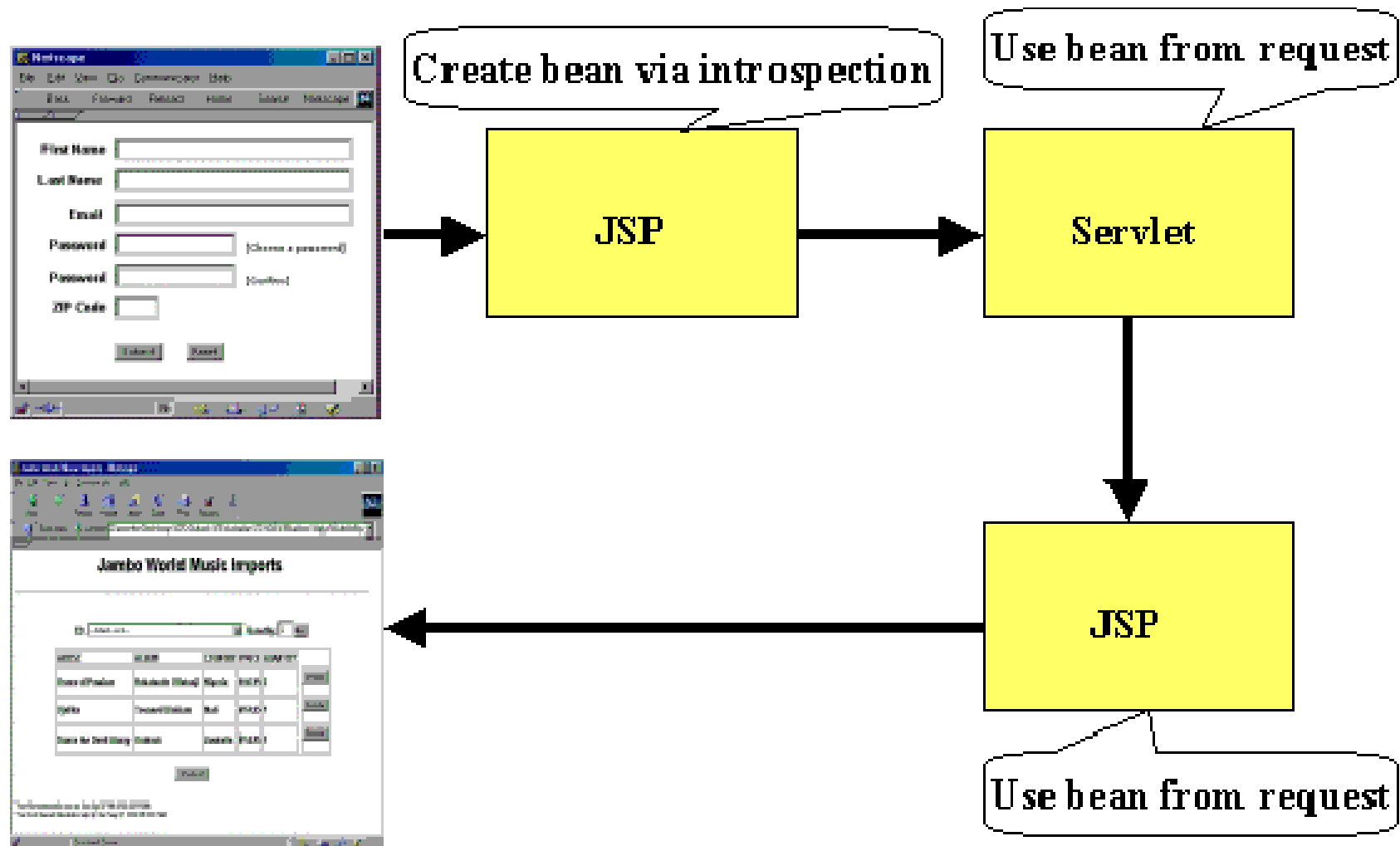


Java Server Pages

Servlet tehnologija

- **Pomoću servleta je jednostavno**
 - čitati podatke sa forme
 - čitati header-e HTTP zahteva
 - postavljati HTTP status kod i header odgovora
 - koristiti cookie-ije i rad sa sesijom
 - deliti podatke između servleta
 - zapamtiti podatke između različitih zahteva
- **Ali je naporno**
 - koristiti println naredbe da bi se generisao HTML kod
 - održavati i menjati generisani HTML

JSP realizacija



JSP realizacija

```
<%@page import="java.text *, java.util. *" %>
<html>
<body>
<%
Date d = new Date();
String today = DateFormat.getDateInstance().format(d);
%>
Today is:
<em> <%=today%> </em>
</body>
</html>
```

.jsp file



Page Compilation

Servlet

Servlet container

JSP realizacija

```
package jsp;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.PrintWriter;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.util.Vector;
import org.apache.jasper.runtime.*;
import java.beans.*;
import org.apache.jasper.JasperException;
import java.text.*;
import java.util.*;
public class _0005cjsp_0005cjsptest_0002ejspsptest_jsp_0
extends HttpJspBase {
    static {
    }
    public _0005cjsp_0005cjsptest_0002ejspsptest_jsp_0( ) {
    }
    private static boolean _jspx_inited = false;
    public final void _jspx_init() throws JasperException {
    }
    public void _jspService(HttpServletRequest request,
    HttpServletResponse response)
    throws IOException, ServletException {
        JspFactory _jspxFactory = null;
        PageContext pageContext = null;
        HttpSession session = null;
        ServletContext application = null;
        ServletConfig config = null;
        JspWriter out = null;
        Object page = this;
        String _value = null;
        try {
            if (_jspx_inited == false) {
                _jspx_init();
                _jspx_inited = true;
            }

```

```
_jspxFactory = JspFactory.getDefaultFactory();
response.setContentType("text/html");
pageContext = _jspxFactory.getPageContext(this,
request,response, "", true, 8192, true);
application = pageContext.getServletContext();
config = pageContext.getServletConfig();
session = pageContext.getSession();
out = pageContext.getOut();
// begin
out.write("\r\n<html>\r\n<body>\r\n");
// end
// begin [file="E:\\jsp\\jsptest.jsp";from=(3,2);to=(5,0)]
Date d = new Date();
String today = DateFormat.getDateInstance().format(d);
// end
// begin
out.write("\r\nToday is: \r\n<em> ");
// end
// begin [file="E:\\jsp\\jsptest.jsp";from=(7,8);to=(7,13)]
out.print(today);</b>
// end
// begin
out.write(" </em>\r\n</body>\r\n</html>\r\n");
// end
} catch (Exception ex) {
    if (out.getBufferSize() != 0)
        out.clear();
    pageContext.handlePageException(ex);
} finally {
    out.flush();
    _jspxFactory.releasePageContext(pageContext);
}
}
}
```

Prednosti JSP tehnologije

- **Iako pomoću JSP se ne može uraditi ništa novo što se ne može postići i pomoću servleta, korišćenje JSP tehnologije olakšava:**
 - pisanje samog HTML koda
 - čitanje i održavanje HTML koda
- **Pomoću JSP tehnologije moguće je:**
 - koristiti standardne HTML alate kao što je Macromedia DreamWeaver.
 - Podeliti posao između dizajne (koji koriste HTML)i Java programera
- **JSP ohrabruje**
 - odvajanje (Java) kod koji predstavlja sam sadržaj, od koda (HTML) pomoću koga se sadržaje prezentuje

Prednosti JSP tehnologije

- **U odnosu na ASP ili ColdFusion**
 - moćniji jezik za dinamički deo aplikacije
 - izvršavanje na većem broju servera i operativnih sistema
- **U odnosu na PHP**
 - moćniji jezik za dinamički deo aplikacije
 - bolja podrška alata
- **U odnosu na korišćenje samo servleta**
 - jednostavnije kreiranje HTML koda
 - korišćenje standardnih alata (n.p., DreamWeaver)
 - podeli i vladaj
 - JSP programeri i dalje moraju poznavati servlet programiranje

Podešavanje okruženja

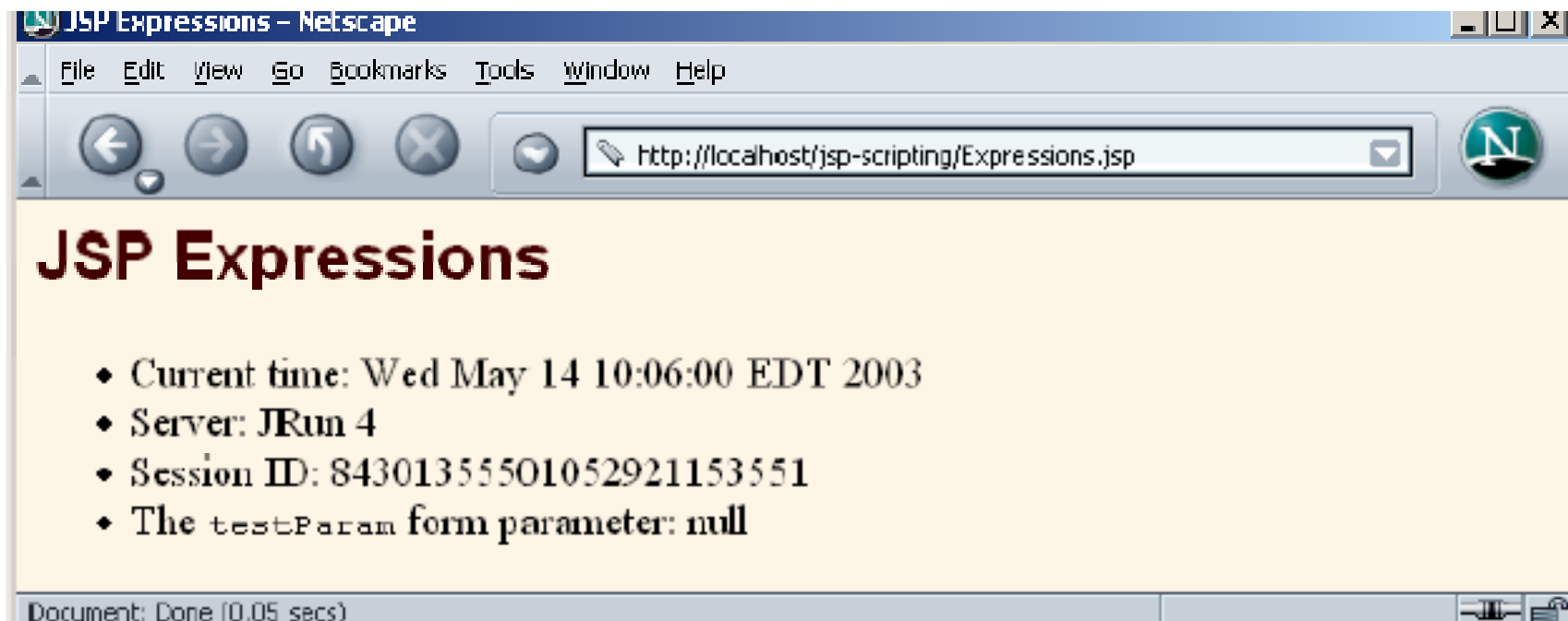
- **Postavljanje CLASSPATH. Ne.**
- **Kompajliranje koda. Not.**
- **Korišćenje paketa da bi se izbegao konflikt sa imenima. Ne.**
- **Postavljanje JSP stranica u specijalne direktorijume. Ne.**
 - *install_dir\webapps\ROOT* (HTML *and* JSP -- Tomcat)
- **Korišćenje specijalnih URLs za poziv JSP stranice. Ne.**
 - Koriste se ista imena kao za HTML stranice (izuzev ekstenzije fajla)
- **Izuzeci**
 - Ptavila oko CLASSPATH, install dirs, ..., i dalje se primenjuju na regularne Java klase koje se koriste u okviru JSP stranice

Primer

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>JSP Expressions</TITLE>
<META NAME="keywords"
CONTENT="JSP,expressions,JavaServer Pages">
<META NAME="description"
CONTENT="A quick example of JSP expressions.">
<LINK REL=STYLESHEET
HREF="JSP-Styles.css"
TYPE="text/css">
</HEAD>
<BODY>
<H2>JSP Expressions</H2>
<UL>
<LI>Current time: <%= new java.util.Date() %>
<LI>Server: <%= application.getServerInfo() %>
<LI>Session ID: <%= session.getId() %>
<LI>The <CODE>testParam</CODE> form parameter:
<%= request.getParameter("testParam") %>
</UL>
</BODY></HTML>
```

Primer

- **Ako je lokacija fajla**
 - C:\jakarta-tomcat-xx\webapps\ROOT\jsp-scripting\Expressions.jsp
- **URL je**
 - <http://localhost/jsp-scripting/Expressions.jsp>



Životni ciklus JSP stranice

		Request #1	Request #2		Request #3	Request #4		Request #5	Request #6
JSP page translated into servlet	Page first written	Yes	No	Server restarted	No	No	Page modified	Yes	No
Servlet compiled		Yes	No		No	No		Yes	No
Servlet instantiated and loaded into server's memory		Yes	No		Yes	No		Yes	No
init (or equivalent) called		Yes	No		Yes	No		Yes	No
doGet (or equivalent) called		Yes	Yes		Yes	Yes		Yes	Yes

Mogućnosti JSP tehnologije

- **Scripting elementi pozivaju kod servleta direktno**
- **Scripting elementi pozivaju kod servleta indirektno (korišćenjem pomoćnih klasa)**
- **JavaBeans**
- **Servlet/JSP aplikacija (MVC)**
- **MVC sa JSP expression jezikom**
- **Posebni tagovi**
- **MVC sa beans, pomoćnim tagovima, i frameworkcima kao što su Struts ili JSF**

**Simple
Application**



**Complex
Application**

Mogućnosti JSP tehnologije

- **Postoje dve opcije**

- Upisati 25 linija Java koda direktno u JSP stranice
- Upisati ovih 25 linija u odvojenu Java klasu i 1 liniju u JSP stranicu koja poziva novu klasu

- **Zašto je druga opcija mnogo bolja?**

- **Razvoj.** Odvojena klasa se piše u Java okruženju (editor ili IDE), ne u HTML okruženju
- **Debugovanje.** Ako postoje syntaksne greške, primetiće se odmah tokom procesa kompajliranja. Jednostavne print naredbe se mogu videti.
- **Testiranje.** Može se napisati test rutina sa petljom koja 10,000 puta testira i ponavlja se nakon svake promene.
- **Višestruko korišćenje.** Može se koristiti ista klasa za više stranica

Osnovna JSP sintaksa

- **HTML Tekst**

- `<H1>Blah</H1>`

- Šalje se dalje klijentu. Prevodi si u servlet kod koji je sličan sledećem

- ```
out.print("<H1>Blah</H1>");
```

- **HTML Komentari**

- `<!-- Komentar -->`

- Isto kao u HTMLu: šalje se dalje klijentu

- **JSP Komentari**

- `<%-- Komentar --%>`

- Ne šalje se klijentu

# Vrste dinamičkih elemenata

- **izrazi (expressions):**

`<%= java_izraz %>`

`<%= new java.util.Date() %>`

- **skriptleti (scriptlets):**

`<% java_kod %>`

`<% for (int i = 0; i < 10; i++) ... %>`

- **deklaracije (declarations):**

`<%! java_deklaracija %>`

`<%! int a; %>`

- **direktive (directives)**

`<%@ direktiva attr="..." %>`

`<%@ page contentType="text/plain" %>`

# JSP izrazi

```
<html>
```

```
...
```

```
<h4>Dobrodošli, <%= username %></h4>
```

```
Danas je <%= new java.util.Date() %>.
```

```
...
```

```
</html>
```

u pitanju je izraz, dakle  
ne završava se sa ;

za izraze koji nisu tipa String  
automatski se poziva toString()



# JSP skriptleti

```
<html>
```

```
...
```

```
<% if (Math.random() < 0.5) { %>
```

```
Dobar dan!
```

```
<% } else { %>
```

```
Dobro veče!
```

```
<% } %>
```

```
...
```

```
</html>
```

# JSP skriptleti

```
<html>
...

<table border=1>
<tr>
 <td>R.br.</td>
 <td>Ime</td>
</tr>
<%
String names[] = {"Marko", "Nikola", "Igor", "Vladimir", "Dejan"};
for (int i = 0; i < names.length; i++) {
%>
<tr>
 <td><%= i %></td>
 <td><%= names[i] %></td>
</tr>
<% } %>
</table>

...
</html>
```

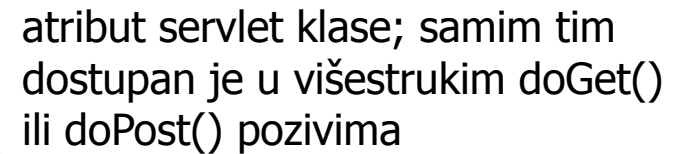
skriptlet se ugrađuje direktno u kod generisanog servleta; tako je brojač petlje **i** vidljiv i u okviru drugog skriptleta (on se nalazi "unutar" for petlje)

# JSP deklaracije

- **definisanje metoda ili atributa servlet klase – izvan metode za obradu zahteva**

```
<%! int hitCount = 0; %>
```

```
<%! private int getRandom() {
 return (int) (Math.random()*100);
}
%>
```



atribut servlet klase; samim tim dostupan je u višestrukim doGet() ili doPost() pozivima

# JSP direktive

- **Omogućavaju kontrolu strukture generisanog servleta.**
- **Dva osnovna tipa direktiva:**
  - page direktive
    - **import** – za import paketa
    - **contentType** – podešava **ContentType** odgovora
  - include direktive
    - **uključuje zadanu stranicu u postojeću**

# JSP direktive

text/html; charset=utf-8

- page direktive

```
<%@ page contentType="text/html" %>
<%@ page import="java.util.Vector" %>
```

- include direktive

```
<jsp:include page="asd.html" />
– uključuje stranicu u momentu zahtjevanja strane
<%@ include file="asd.jsp" %>
– uključuje stranicu u momentu kada se stranica prevodi u servlet
```

## **Predefinisane promjenljive**

- **HttpServletResponse povezan sa odgovorom klijentu**
- **Dozvoljeno je postavljanje HTTP statusnih kodova i zaglavlja odgovora (response headers)**
- **PrintWriter koji se koristi za slanje odgovora klijentu**
- **out je baferovana verzija PrintWriter pod nazivom JspWriter**
- **Moguće je podešavanje veličine bafera, kao i njegovo potpuno isključivanje pomoću buffer atributa page direktive**
- **out se koristi skoro isključivo u skriptletima, jer se JSP izrazi automatski smestaju u izlazni tok**
- **HttpSession objekt povezan sa zahtevom**
- **Sesije se kreiraju automatski, tako da je ova varijabla već povezana čak iako nema ulazne reference na sesiju.**
  - **izuzetak je ako se koristi session atribut page direktive kako bi se isključilo praćenje sesija. U tom slučaju pokušaj pristupanja session varijabli rezultuje generisanjem poruke o grešci od strane servera u momentu prevođenja JSP strane u servlet**

## **Predefinisane promjenljive**

- **application** - objekat klase **ServletContext** koji služi za komunikaciju servleta i aplikacionog servera
- Uobičajena upotreba je za smeštanje globalnih promenljivih uz pomoć metoda **setAttribute()/getAttribute()**
- **page** - sinonim za ključnu reč **this**

# Predefinisane promenljive

```
<html>
```

```
...
```

```
<% if (request.getParameter("username") != null)
```

```
{ %>
```

```
Vrednost parametra: <%=
```

```
request.getParameter("username") %>
```

```
<% } %>
```

```
...
```

```
</html>
```



## **jspInit and jspDestroy Metodi**

- **Kod JSP stranica, kao i kod regularnih servleta, ponekad postoji potreba za korišćenjem init i destroy metoda**
- **Problem: servlet koji napravljen od JSP stranice možda već koristi svoje init i destroy metode**
  - Njihovo preklapanje može da prouzrokuje probleme.
  - pa je nelegalno koristiti JSP deklaracije da bi se deklarirali init ili destroy metodi.
- **Rešenje: koristiti jspInit i jspDestroy.**
  - Auto-generisani servlet garantuje da će pozivati ove metode u okviru svojih init i destroy metoda, ali je standardna verzija jspInit i jspDestroy metoda prazna (sa mogućnošću preklapanja)

# JSP deklaracija

- **Problem**

- Predefinisane promenljive (request, response, out, session,...) su *lokalne* u okviru \_jspService metoda.
  - Tako da nisu dostupne u metodama koji su definisani u okviru JSP deklaracije ili u metodama u pomoćnim klasama.

- **Rešenje: proslediti ih kao argumente.**

```
<%! Private void nekiMetod(HttpSession s){
 uradiNestoSa(s);}

```

```
%>
```

```
<% nekiMetod(session); %>
```

- **Treba primetiti da println metod od JspWriter baca IOException**

- Koristiti "throws IOException" ya metode koji koriste println naredbu

# Korišćenje page direktive

- **Pruža informacije na visokom nivou o samom servletu koji se izvršava nakon JSP stranice**
- **Može kontrolisati**
  - Koje se klase importuju
  - Koju klasu servlet nasleđuje
  - Koji MIME tipovi se generišu
  - Kako se obrađuje multithread
  - Da li servlet pripada sesiji
  - Veličinu i ponašanje izlaznog bafera
  - Koja stranica obrađuje neočekivane greške

# import atribut

- **Sintaksa**

- `<%@ page import="package.class" %>`
- `<%@ page import="package.class1,...,package.classN" %>`

- **Upotreba**

- Generisanje import naredbu na vrhu definicije servleta

- **Napomene**

- JSP stranice se mogu nalaziti bilo gde na serveru, ali klase koje se koriste u okviru JSP stranica moraju biti u uobičajenim servlet direktorijumima

- `.../WEB-INF/classes` ili
- `.../WEB-INF/classes/directoryMatchingPackage`
- Preporuka je da se uvek koriste paketi za klase koje se koriste u okviru JSP stranice!

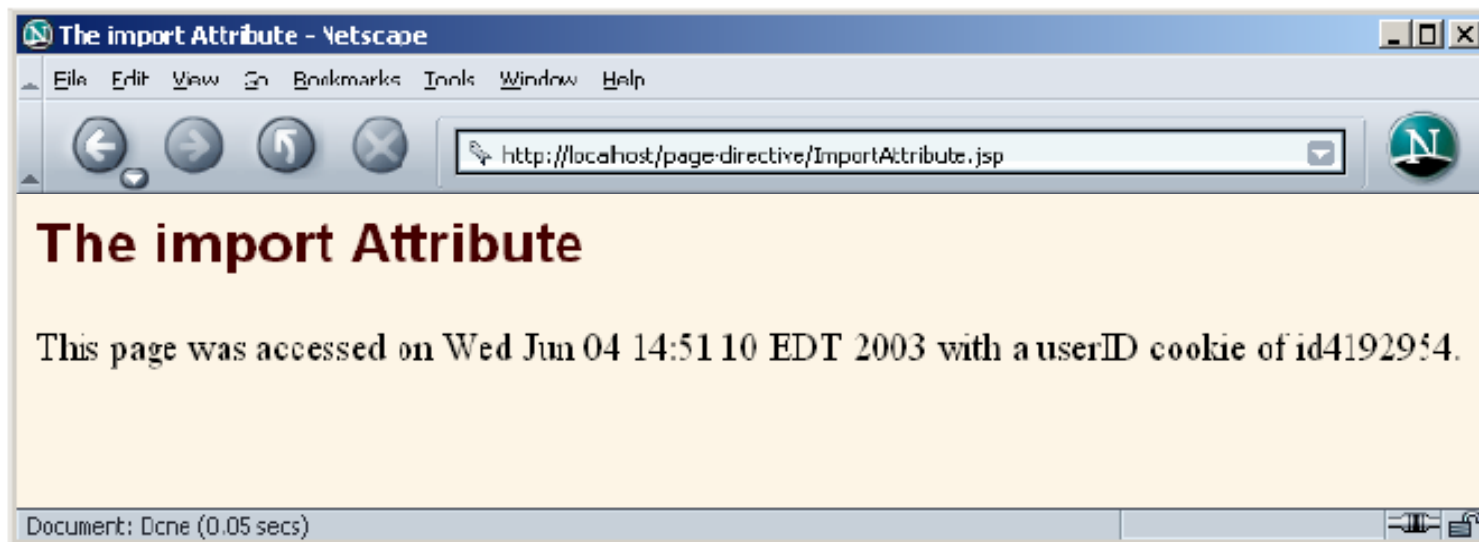
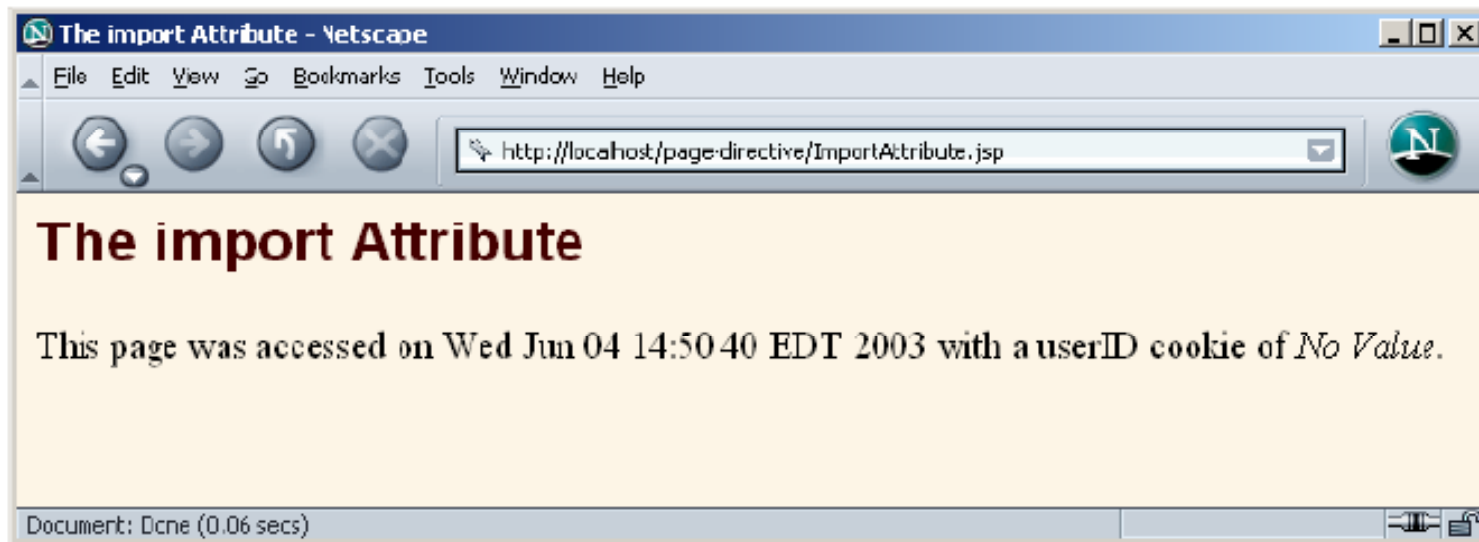
# Paketi

```
public class SomeClass {
 public String someMethod(...) {
 SomeHelperClass test = new SomeHelperClass(...);
 String someString =
 SomeUtilityClass.someStaticMethod(...);
 ...
 }
}
▪ I
<%
SomeHelperClass test = new SomeHelperClass(...);
String someString =
SomeUtilityClass.someStaticMethod(...);
%>
```

# Primer

```
<H2>The import Attribute</H2>
<%@ page import="java.util.*,coreservlets.*" %>
<%!
private String randomID() {
int num = (int)(Math.random()*10000000.0);
return("id" + num);
}
private final String NO_VALUE = "<I>No Value</I>";
%>
<%
String oldID =
CookieUtilities.getCookieValue(request, "userID",
NO_VALUE);
if (oldID.equals(NO_VALUE)) {
String newID = randomID();
Cookie cookie = new LongLivedCookie("userID", newID);
response.addCookie(cookie);
}
%>
This page was accessed on <%= new Date() %> with a userID
cookie of <%= oldID %>.
</BODY></HTML>
```

# Primer



# **contentType i pageEncoding atributi**

## ▪ **Sintaksa**

- `<%@ page contentType="MIME-Type" %>`
- `<%@ page contentType="MIME-Type; charset=Character-Set" %>`
- `<%@ page pageEncoding="Character-Set" %>`

## ▪ **Upotreba**

- Specificiraju MIME tip stranice generisane od strane servleta koji je nastao od JSP stranice

## ▪ **Napomene**

- Vrednost atributa se ne može izračunati u vreme zahteva



# Uslovno generisanje Excel stranice

- **Za ovaj problem ne može se koristiti atribut `contentType`, jer on ne može da bude uslovljen.**

- Sledećim kodom se **uvek** dobija Excel MIME tip

```
<% boolean usingExcel = checkUserRequest(request); %>
```

```
<% if (usingExcel) { %>
```

```
<%@ page contentType="application/vnd.ms-excel" %>
```

```
<% } %>
```

- **Rešenje: koristiti regularne JSP skriptlete sa `response.setContentType`**

# Uslovno generisanje Excel stranice

...

<BODY>

<CENTER>

<H2>Comparing Apples and Oranges</H2>

<%

String format = request.getParameter("format");

if ((format != null) && (format.equals("excel"))) {

**response.setContentType("application/vnd.ms-excel");**

}

%>

<TABLE BORDER=1>

<TR><TH></TH> <TH>Apples<TH>Oranges

<TR><TH>First Quarter <TD>2307 <TD>4706

<TR><TH>Second Quarter<TD>2982 <TD>5104

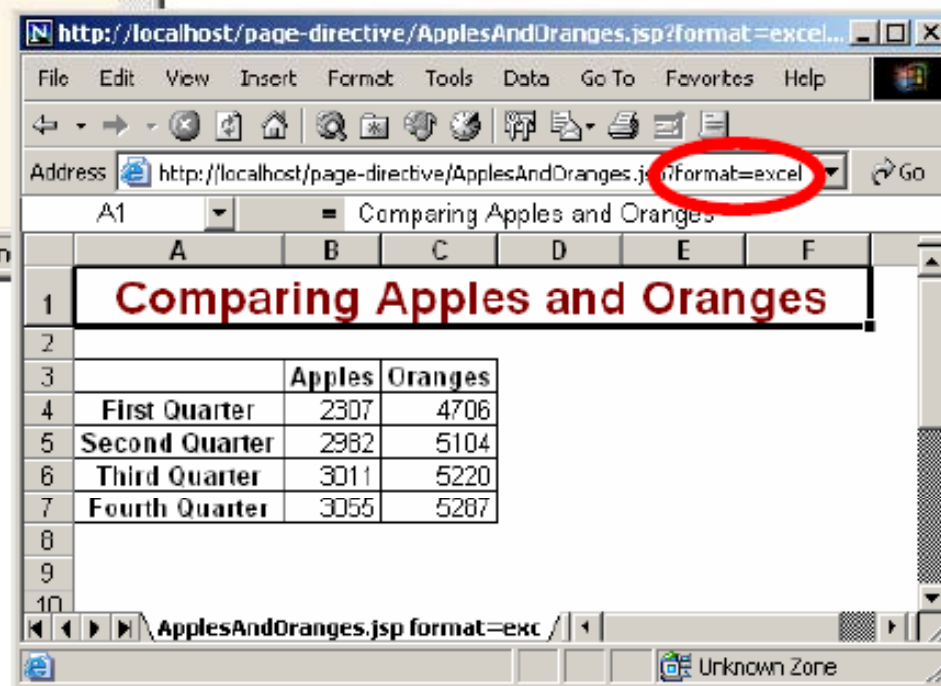
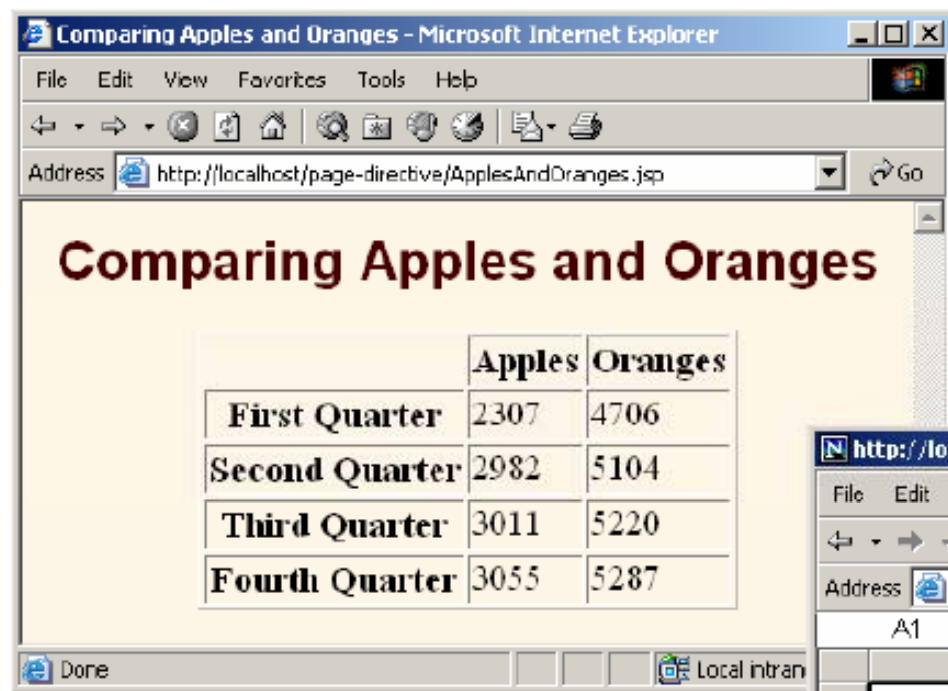
<TR><TH>Third Quarter <TD>3011 <TD>5220

<TR><TH>Fourth Quarter<TD>3055 <TD>5287

</TABLE>

</CENTER></BODY></HTML>

# Uslovno generisanje Excel stranice



# session atribut

- **Sintaksa**

- `<%@ page session="true" %>` `<%-- Default --%>`
- `<%@ page session="false" %>`

- **Upotreba**

- Da se označi da neka stranica ne pripada sesiji

- **Napomena**

- Ako se ništa ne navede, stranica je deo sesije
- Smanjuje se količina potrebne memorije na serveru, ako je sajt visokog inteziteta

# isELIgnored atribut

- **Sintaksa**

- `<%@ page isELIgnored="false" %>`
- `<%@ page isELIgnored="true" %>`

- **Upotreba**

- Kontrola da li se JSP 2.0 Expression Language (EL) ignoriše (true) ili normalno obrađuje (false).

- **Napomena**

- Ako web.xml specificira verziju servleta 2.3 (JSP 1.2) ili raniju, default je true
- I dalje je moguće menjati default vrednosti
- Ako web.xml specificira verziju servleta 2.4 (JSP 2.0) ili raniju, default je false

# **errorPage atribut**

- **Sintaksa**

- `<%@ page errorPage="Relative URL" %>`

- **Upotreba**

- Specificira JSP stranicu koja obrađuje bilo koji izuzetak koji se dogodio na tekućoj stranici

- **Napomene**

- Izuzetak koji se dogodio je automatski dostupan dizajniranoj error stranici u obliku "exception" promenljive

- web.xml fajl dozvoljava definisanje error stranica na nivou aplikacija

# **isErrorPage atribut**

- **Sintaksa**

- `<%@ page isErrorPage="true" %>`
- `<%@ page isErrorPage="false" %> <%-- Default -- %>`

- **Upotreba**

- Specificira da li se trenutna stranica može izvršavati kao error stranica za neku drugu JSP stranicu

- **Napomena**

- Nova predefinisana promenljiva exception se kreira i dostupna je u okviru error stranica
- Ovakav rad treba koristiti samo za hitan backup; trebalo bi eksplicitno obraditi što je moguće više izuzetaka
- Uvek treba proveriti unete podatke

# isErrorPage primer

```
...<BODY>
<%@ page errorPage="/WEB-INF/SpeedErrors.jsp" %>
<TABLE BORDER=5 ALIGN="CENTER">
<TR><TH CLASS="TITLE">Computing Speed</TABLE>
<%!
private double toDouble(String value) {
return(Double.parseDouble(value));
}
%>
<%
double furlongs = toDouble(request.getParameter("furlongs"));
double fortnights = toDouble(request.getParameter("fortnights"));
double speed = furlongs/fortnights;
%>

Distance: <%= furlongs %> furlongs.
Time: <%= fortnights %> fortnights.
Speed: <%= speed %> furlongs per fortnight.

</BODY></HTML>
```



## isErrorPage primer

```
...<BODY>
```

```
<%@ page isErrorPage="true" %>
```

```
<TABLE BORDER=5 ALIGN="CENTER">
```

```
<TR><TH CLASS="TITLE">
```

```
Error Computing Speed</TABLE>
```

```
<P>
```

ComputeSpeed.jsp reported the following error:

<I><%= exception %></I>. This problem occurred in the following place:

```
<PRE>
```

```
<%@ page import="java.io.*" %>
```

```
<% exception.printStackTrace(new PrintWriter(out)); %>
```

```
</PRE>
```

```
</BODY></HTML>
```

# isThreadSafe atribut

## ▪ Sintaksa

- `<%@ page isThreadSafe="true" %>` `<%-- Default --%>`
- `<%@ page isThreadSafe="false" %>`

## ▪ Upotreba

- Da se naglasi sistemu da kod nije threadsafe, tako da sistem može da spreči konkurentne pristupe kodu
  - U stvari, servletu se prenosi da implementira `SingleThreadModel`

## ▪ Napomene

- Može prouzrokovati lošije performanse u nekim situacijama
- Može prouzrokovati nekorektan rezultat u drugim

## **isThreadSafe primer**

```
<%! private int idNum = 0; %>
<%
String userID = "userID" + idNum;
out.println("Your ID is " + userID + ".");
idNum = idNum + 1;
%>
```

- **ID mora biti jedinstven**

# Da li je `isThreadSafe` potreban

- **Nije potreban!. Moguća je sinhronizacija:**

```
<%! private int idNum = 0; %>
<%
synchronized(this) {
String userID = "userID" + idNum;
out.println("Your ID is " + userID + ".");
idNum = idNum + 1;
}
%>
```

- **Dobijaju se bolje performanse u okruženju sa velikim saobraćaje**
- **`isThreadSafe="false"` se neće korektno izvršiti, ako server koristi pristup sa pool-of-instances**

# **extends atribut**

- **Sintaksa**

- `<%@ page extends="package.class" %>`

- **Upotreba**

- Da specificira klasu roditelja servleta koji se dobija od JSP stranice

- **Napomene**

- Koristiti sa ekstremnom pažnjom
  - Uobičajena upotreba je da bi se nasledile klase koje prezentuje proizvođač servera (različite vrste podrške), a ne da bi se nasleđivale sopstvene klase.

# Korišćenje fajlova

- Pomoću `jsp:include` mogu se uključiti spoljne stranice u trenutku prihvatanja zahteva
- Pomoću `<%@ include ... %>` (include direktive) se mogu uključiti spoljne stranice u trenutku prevođenja stranice
- Postoje razlozi zašto je u većem broju slučajeva bolje koristiti `jsp:include`
- Pomoću `jsp:plugin` mogu se uključiti apleti za izvršavanje pomoću Java Plug-in

# **jsp:include**

- **Sintaksa**

- `<jsp:include page="Relative URL" />`

- **Upotreba**

- Više puta koristiti iste JSP, HTML, ili obične tekst dokumente
  - Dozvoliti promene uključenog sadržaja bez promene osnovnih JSP strana

- **Napomene**

- JSP sadržaj ne može da menja osnovne stranice: koristi se samo izlaz uključenih JSP stranica
  - Ne treba zaboraviti simbol / na kraju taga
  - Relativne URL adrese koje počinju sa znakom / se interpretiraju relativno u odnosu na Web aplikaciju, a ne relativno u odnosu na root servera.
  - Moguće je uključiti fajlove iz WEB-INF direktorijuma

# **jsp:include primer**

...

**<BODY>**

**<TABLE BORDER=5 ALIGN="CENTER">**

**<TR><TH CLASS="TITLE">**

**What's New at JspNews.com</TABLE>**

**<P>**

**Here is a summary of our three  
most recent news stories:**

**<OL>**

**<LI><jsp:include page="/WEB-INF/Item1.html" />**

**<LI><jsp:include page="/WEB-INF/Item2.html" />**

**<LI><jsp:include page="/WEB-INF/Item3.html" />**

**</OL>**

**</BODY></HTML>**



## **jsp:include primer Item1.html**

**<B>Bill Gates acts humble.</B>** In a startling and unexpected development, Microsoft big wig Bill Gates put on an open act of humility yesterday.

**<A HREF="http://www.microsoft.com/Never.html">**  
More details...**</A>**

- Može se primetiti da stranice nije celokupan HTML dokument;
- Postoje samo tagovi koji treba da se pojave na osnovnoj strani

# jsp:include primer rezultat



# **jsp:param element**

- **Primer**

```
<jsp:include
 page="/fragments/StandardHeading.jsp">
<jsp:param name="bgColor" value="YELLOW" />
</jsp:include>
```

- **Dobijeni URL**

- http://host/path/MainPage.jsp?fgColor=RED

- **Osnovna stranica**

- fgColor: RED

- bgColor: null

- **Uključena stranica**

- fgColor: RED

- bgColor: YELLOW

# include

- **Sintaksa**

- `<%@ include file="Relative URL" %>`

- **Upotreba**

- Upotreba više puta istog JSP sadržaja

- **Napomene**

- Serveri ne proveravaju da li postoje promene uključenih fajlova.

- Zato je potrebno promeniti i osnovne JSP fajlove svaki put kada su i uključeni fajlovi promenjeni.

- Mogu se koristiti i specifični mehanizmi operativnih sistema, na primer na Unix-u

- `<%-- Navbar.jsp modified 12/1/03 --%>`

- `<%@ include file="Navbar.jsp" %>`

## **include i jsp:include**

	<b>jsp:include</b>	<b>&lt;%@ include ...%&gt;</b>
<b>Basic syntax</b>	<code>&lt;jsp:include page="..." /&gt;</code>	<code>&lt;%@ include file="..." %&gt;</code>
<b>When inclusion occurs</b>	Request time	Page translation time
<b>What is included</b>	Output of page	Contents of file
<b>Number of resulting servlets</b>	Two	One
<b>Can included page set response headers that affect the main page?</b>	No	Yes
<b>Can included page define fields or methods that main page uses?</b>	No	Yes
<b>Does main page need to be updated when included page changes?</b>	No	Yes

# Kada šta upotrebiti

- **Koristiti jsp:include Uvek kada je moguće**
  - Promene uključenih stranica ne zahtevaju nikakvu ručnu obradu
  - Postoji razlika u brzini primene ovih naredbi
- **Ipak i include naredba (<%@ include ...%>) ima i dodatna svojstva kao što su**
  - Osnovna stranica  
`<%! int accessCount = 0; %>`
  - Uključena stranica  
`<%@ include file="snippet.jsp" %>`  
`<%= accessCount++ %>`

# Primer

```
<%@ page import="java.util.Date" %>
<%-- The following become fields in each servlet that
results from a JSP page that includes this file. --%>
<%!
private int accessCount = 0;
private Date accessDate = new Date();
private String accessHost = "<I>No previous access</I>";
%>
<P>
<HR>
This page © 2006
my-company.com.
This page has been accessed <%= ++accessCount %>
times since server reboot. It was most recently
accessed from
<%= accessHost %> at <%= accessDate %>.
<% accessHost = request.getRemoteHost(); %>
<% accessDate = new Date(); %>
```

# Primer

**<BODY>**

**<TABLE BORDER=5 ALIGN="CENTER">**

**<TR><TH CLASS="TITLE">**

**Some Random Page</TABLE>**

**<P>**

**Information about our products and services.**

**<P>**

**Blah, blah, blah.**

**<P>**

**Yadda, yadda, yadda.**

**<%@ include file="/WEB-INF/ContactSection.jsp" %>**

**</BODY></HTML>**

**15 J2EE training: <http://courses.coreservlets.com>**

**Reusing Footers: Result**



# Primer



# Kada šta upotrebiti

- **Footer definiše polje accessCount kao promenljivu instance**
- **Ako osnovna stranica koristi accessCount, trebalo bi da se koristi naredba @include**
  - U drugim slučajevima accessCount će biti undefined
- **U ovom primeru osnovna stranica ne koristi accessCount**
  - Zašto se onda koristi naredba @include?

# Poziv apleta

- **Ako je aplet realizovan sa JDK 1.1 ili 1.02 (da bi ga izvršavali i veoma stari čitači).**
  - Izvršava se u skoro svakom čitaču
  - Koristi se APPLET tag
- **Ako je aplet realizovan sa JDK 1.5.**
  - Zahteva IE 5 ili kasnije verzije ili Netscape 6 kasnije verzije
  - Koristi se APPLET tag
- **Ako je aplet realizovan sa Java 2 okruženjem.**
  - Izvršava se u skoro svakom čitaču
  - Koriste se OBJECT i EMBED tagovi
  - Ova opcija je pojednostavljena pomoću jsp:plugin taga

# Poziv apleta

- **Koristi se sintaksa slična APPLET tagu**

- Dobijaju se OBJECT i EMBED tagovi

- **APPLET Tag**

**<APPLET CODE="MyApplet.class" WIDTH=475  
HEIGHT=350>**

**</APPLET>**

- **jsp:plugin**

**<jsp:plugin type="applet" width="475"  
height="350">**

**</jsp:plugin>**

- **Napomena**

- JSP element i imena atributa su case sensitive
  - Sve vrednosti atributa moraju biti između ' ' ili ""
  - Slično XMLu, a ne HTMLu

# Dobijeni HTML kod

```
<object classid=
"clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
width="300" height="200"
codebase="http://java.sun.com/products/plugin/1.2.
2/jinst all-1_2_2-win.cab#Version=1,2,2,0">
<param name="java_code" value="SomeApplet.class">
<param name="type" value="application/x-java-applet;">
<COMMENT>
<embed type="application/x-java-applet;" width="300"
height="200"
pluginspage="http://java.sun.com/products/plugin/"
java_code="SomeApplet.class">
<noembed>
</COMMENT>
</noembed></embed>
</object>
```

# Primer

...

<BODY>

<CENTER>

<TABLE BORDER=5>

<TR><TH CLASS="TITLE">

Using jsp:plugin</TABLE>

<P>

<jsp:plugin type="applet"

code="PluginApplet.class"

width="370" height="420">

</jsp:plugin>

</CENTER></BODY></HTML>

## **Dobijeni Java kod**

```
import javax.swing.*;
/** An applet that uses Swing and Java 2D
* and thus requires the Java Plug-in.
*/
public class PluginApplet extends JApplet {
 public void init() {
 WindowUtilities.setNativeLookAndFeel();
 setContentPane(new TextPanel());
 }
}
```

# Atributi jsp:plugin elementa

- **type**
  - Za aplete mora imati vrednost "applet".
  - Koristi se "bean" da bi se ubacili JavaBeans elementi u okviru Web stranice.
- **code**
  - Upotreba je ista kao CODE atributa u okviru APPLET taga, specificira klasu apleta na najvišem nivou
- **width, height**
  - Upotreba je ista kao WIDTH, HEIGHT u okviru APPLET taga
- **codebase**
  - Upotreba je ista kao CODEBASE u okviru APPLET taga
- **align**
  - Upotreba je ista kao ALIGN u okviru APPLET i IMG taga



# Atributi `jsp:plugin` elementa

- **hspace, vspace**
  - Upotreba je ista kao HSPACE, VSPACE u okviru APPLET taga
- **archive**
  - Upotreba je ista kao ARCHIVE u okviru APPLET taga, specificira JAR fajl iz koga se klase i slike učitavaju
- **name**
  - Upotreba je ista kao NAME u okviru APPLET taga, specificira ime koje se koristi u okviru komunikacije između apleta ili za pristup apletu iz skript jezika kao što je JavaScript.
- **title**
  - Upotreba je ista kao TITLE atributa

# Atributi `jsp:plugin` elementa

- **jreversion**

- Identifikuje verziju Java Runtime Environment (JRE) koje se zahteva. Default je 1.2.

- **iepluginurl**

- Definiše URL sa koga se plug-in za Internet Explorer može dobiti. Korisnici koji nemaju instaliran ovaj plug-in, biće upitani da izvrše njegov download sa navedene lokacije. Podrazumevana vrednost je Sun-ov, ali za intranet korisnike je bolje rešenje lokalna kopija

- **nspluginurl**

- Definiše URL sa koga se plug-in za Netscape može dobiti. Podrazumevana vrednost je Sun-ov, ali za intranet korisnike je bolje rešenje lokalna kopija

## Atributi jsp:plugin elementa

- PARAM Tagovi

- **<APPLET CODE="MyApplet.class" WIDTH=475  
HEIGHT=350>**

**<PARAM NAME="PARAM1" VALUE="VALUE1">**

**<PARAM NAME="PARAM2" VALUE="VALUE2">**

**</APPLET>**

- Ekvivalentan jsp:param kod

**<jsp:plugin type="applet" code="MyApplet.class"  
width="475" height="350">**

**<jsp:params>**

**<jsp:param name="PARAM1" value="VALUE1" />**

**<jsp:param name="PARAM2" value="VALUE2" />**

**</jsp:params>**

**</jsp:plugin>**

## **jsp:fallback elementa**

- **APPLET Tag**

**<APPLET CODE="MyApplet.class"**

**WIDTH=475 HEIGHT=350>**

**<B>Error: this example requires Java.</B>**

**</APPLET>**

- **Ekvivalent jsp:plugin kod sa jsp:fallback**

**<jsp:plugin type="applet" code="MyApplet.class"**

**width="475" height="350">**

**<jsp:fallback>**

**<B>Error: this example requires Java.</B>**

**</jsp:fallback>**

**</jsp:plugin>**

# **Rad sa bean-ovima**

- **Potrebno je razumeti prednosti korišćenja bean-ova**
- **Kreiranje bean-ova**
- **Instaliranje bean klasa na serveru**
- **Pristup bean property-ijima**
- **Eksplicitno postavljanje bean property-ija**
- **Automatsko postavljanje bean property-ija iz parametara**
- **Deljenje bean-ova između više servleta i JSP stranica**

# Rad sa bean-ovima

- **To su Java klase koje moraju ispunjavati nekoliko pravila**
  - Moraju imati zero-argument (prazan) konstruktor
  - Ovaj zahtev se može ispuniti eksplicitnim definisanjem takvog konstruktora ili izostavljanjem svih konstruktora
  - Ne bi trebalo da ima public promenljive instanci (polja)
  - Neposredne vrednosti bi trebale da se dobijaju pomoću metoda nazvanih *getXxx / setXxx*
  - Ako klasa ima metod *getTitle* koji kao rezultat vraća *String*, kaže se da klasa poseduje *String property* nazvanu *title*
  - Boolean properties koriste *Xxx* umesto *getXxx*
  - Više informacije na <http://java.sun.com/beans/docs/>

# Rad sa bean-ovima

- Da bi se napisao bean, ne smeju postojati public polja
- Znači sledeći kod  
public double speed;
- treba zameniti sa  
private double speed;  
public double getSpeed() {  
return(speed);  
}  
public void setSpeed(double newSpeed) {  
speed = newSpeed;  
}
- I ovo treba uraditi u celokupnom Java kodu na svakom mestu

# Osnovna sintaksa

- **jsp:useBean**

- U najjednostavnijem slučaju, ovaj element pravi novibean.
- Način korišćenja:

```
<jsp:useBean id="beanName" class="package.Class" />
```

- **jsp:getProperty**

- Ovaj element čita i prikazuje vrednosti definisanih bean property-ija.
- Način korišćenja:

```
<jsp:getProperty name="beanName" property="propertyName" />
```

- **jsp:setProperty**

- Ovaj element menja vrednost bean property-ija
- Način korišćenja:

```
<jsp:setProperty name="beanName" property="propertyName"
value="propertyValue" />
```



# jsp:useBean

- **Sintaksa**

- `<jsp:useBean id="name" class="package.Class" />`

- **Upotreba**

- Dozvoljava instanciranje Java klasa bez eksplicitnog Java programiranja (XML-kompatibilna sintaksa)

- **Napomene**

- Interpretacija:

- `<jsp:useBean id="book1" class="coreservlets.Book" />`

- Može se dobiti i pomoću skriptleta

- `<% coreservlets.Book book1 = new coreservlets.Book();  
%>`

- ali jsp:useBean ima dve prednosti:

- Jednostavnije je dobijati vrednosti objekata iz zahtevanih parametera

- Jednostavnije je deliti objekte između stranica ili servleta

# **jsp:getProperty**

- **Sintaksa**

`<jsp:getProperty name="name" property="property" />`

- **Upotreba**

- Dozvoljava pristup bean property-ijima bez eksplicitnog Java programiranja

- **Napomene**

- `<jsp:getProperty name="book1" property="title" />`

- Je ekvivalentno sa sledećim JSP izrazom

- `<%= book1.getTitle() %>`

# jsp:setProperty

- **Sintaksa**

```
<jsp:setProperty name="name" property="property"
 value="value" />
```

- **Upotreba**

- Dozvoljava postavljanje bean property-ija bez eksplicitnog Java programiranja

- **Napomene**

```
<jsp:setProperty name="book1" property="title"
 value="Core Servlets and JavaServer Pages" />
```

- Je ekvivalentno sa sledećem skriptletu

```
<% book1.setTitle("Core Servlets and JavaServer Pages");
 %>
```

# Primer

```
package coreservlets;

public class StringBean {
 private String message = "No message specified";

 public String getMessage () {
 return(message);
 }

 public void setMessage(String message) {
 this.message = message;
 }
}
```

- **Beanovi se postavljaju u uobičajeni Java direktorijum**

.../WEB-INF/classes/ *directoryMatchingPackageName*

- **Beanovi (i pomoćne klase) moraju uvek biti u okviru paketa!**

# Primer

```
<jsp:useBean id="stringBean" class="coreservlets.StringBean" />
```

```

```

```
Initial value (from jsp:getProperty):
```

```
<I><jsp:getProperty name="stringBean" property="message" /></I>
```

```
Initial value (from JSP expression):
```

```
<I><%= stringBean.getMessage() %></I>
```

```
<jsp:setProperty name="stringBean" property="message"
value="Best string bean: Fortex" />
```

```
Value after setting property with jsp:setProperty: <I> <jsp:getProperty
name="stringBean" property="message" /></I>
```

```
<%= stringBean.setMessage ("My favorite: Kentucky Wonder");
%>
```

```
Value after setting property with scriptlet:
```

```
<I><%= stringBean.getMessage() %></I>
```

```

```

# Primer



## Postavljanje property-ija

```
<!DOCTYPE ...>
```

```
...
```

```
<jsp:useBean id="entry"
```

```
class="coreservlets.SaleEntry" />
```

```
<%-- setItemID expects a String --%>
```

```
<jsp:setProperty
```

```
name="entry"
```

```
property="itemID"
```

```
value='<%= request.getParameter("itemID") %>' />
```

## Postavljanje property-ija

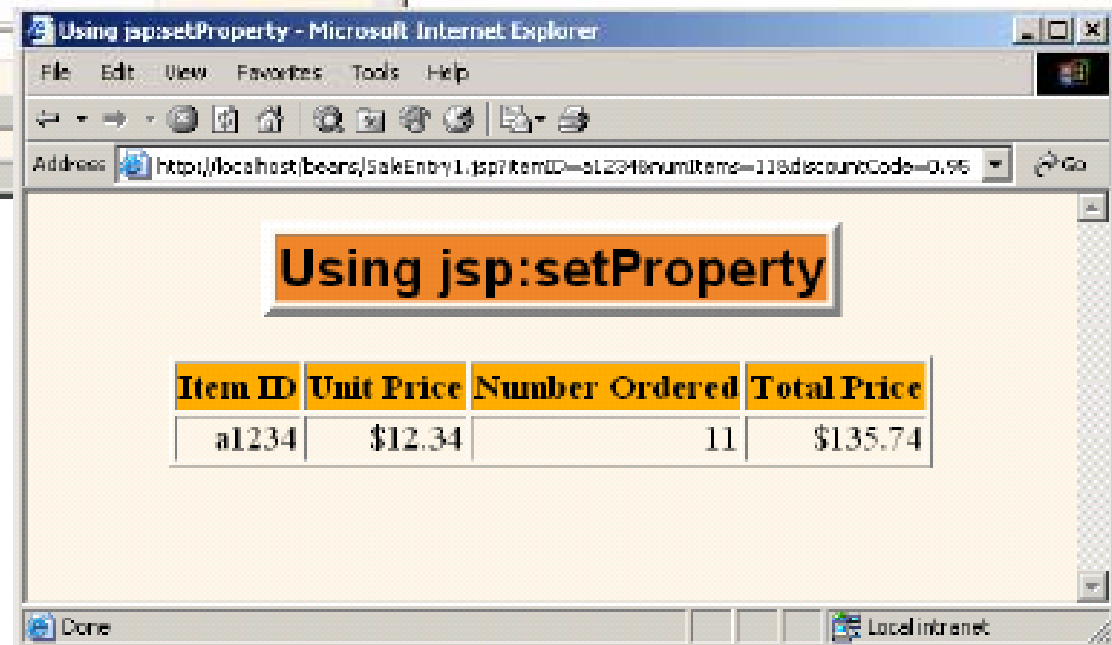
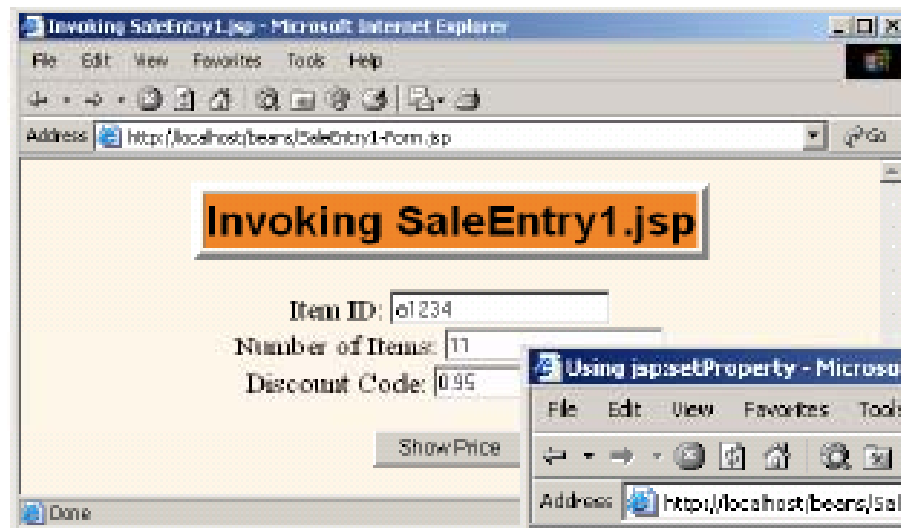
```
<%
int numItemsOrdered = 1;
try {
numItemsOrdered =
Integer.parseInt(request.getParameter("numItems"));
} catch(NumberFormatException nfe) {}
%>
<%-- setNumItems expects an int --%>
<jsp:setProperty
name="entry"
property="numItems"
value="<%= numItemsOrdered %>" />
```



## Postavljanje property-ija

```
<%
double discountCode = 1.0;
try {
String discountString =
request.getParameter("discountCode");
discountCode =
Double.parseDouble(discountString);
} catch(NumberFormatException nfe) {}
%>
<%-- setDiscountCode expects a double --%>
<jsp:setProperty
name="entry"
property="discountCode"
value="<%= discountCode %>" />
```

# Postavljanje property-ija



# Postavljanje property-ija

- **Pomoću atributa `jsp:setProperty` može se postići da**
  - se vrednosti postavljaju pomoću specificiranog parametra zahteva
  - Potrebno je izvršiti jednostavnu automatsku konverziju tipova za property-ije koji očekuju vrednosti standardnih vrednosti `boolean`, `Boolean`, `byte`, `Byte`, `char`, `Character`, `double`, `Double`, `int`, `Integer`, `float`, `Float`, `long`, ili `Long`.

## Ulazni parametri

```
<jsp:useBean id="entry"
class="coreservlets.SaleEntry" />
<jsp:setProperty
name="entry"
property="itemID"
param="itemID" />
<jsp:setProperty
name="entry"
property="numItems"
param="numItems" />
<jsp:setProperty
name="entry"
property="discountCode"
param="discountCode" />
```

# Ulazni parametri

- **Simbol "\*" se može upotrebiti za vrednost property-ija atributa jsp:setProperty da bi se postiglo da**
  - se vrednost dobija preko parametara zahteva koji ima isto ime kao i ime property-ija
  - Jednostavna automatska konverzija tipova se izvršava

## Ulazni parametri

```
<jsp:useBean id="entry"
class="coreservlets.SaleEntry" />
<jsp:setProperty name="entry" property="*" />
```

- **Ovo je ekstremni primer za realizaciju "form beans" – objekata čiji se property-iji popunjavaju prilikom slanja forme.**
  - Može se dati proces podeliti između više formi, tako da svaka forma popunjava određeni deo objekta.

# Deljenje bean-ova

- **Može se koristiti atribut scope da bi se definisale dodatne lokacije gde se može smestiti bean**
  - I dalje ograničenje je lokalne pormenljive u okviru \_jspService
  - `<jsp:useBean id="..." class="..." scope="..." />`
- **Može se definisati da više servleta ili JSP stranica dele podatke**
- **Može se dozvoliti i uslovno kreiranje bean-a**
  - Kreirati novi objekat *samo* ako se ne može pronaći postojeći

# Vrednosti atributa scope

- **page**

**(`<jsp:useBean ... scope="page"/>` ili `<jsp:useBean...>`)**

– Default vrednost. Bean objekat treba da bude smešten u okviru PageContext objekta za vreme trajanja zahteva. Metodi u okviru istog servleta pristupaju bean-u

- **application**

**(`<jsp:useBean ... scope="application"/>`)**

– Bean će biti smešten u ServletContext (dostupan preko promenljivih aplikacije ili pozvan pomoću `getServletContext()`).

- ServletContext je deljen između svih servleta iste Web aplikacije (ili svih servleta na serveru, ako eksplicitno nijedna Web aplikacija nije definisana).



# Vrednosti atributa scope

- **session**

**(<jsp:useBean ... scope="session"/>)**

– Bean će biti smešten u HttpSession objektu povezanom sa trenutnim zahtevom, gde mu se može pristupiti iz regularnog servlet koda sa metodama getAttribute i setAttribute, kao bilo kom drugom objektu sesije

- **request**

**(<jsp:useBean ... scope="request"/>)**

– Bean objekat treba da bude smešten u ServletRequest objektu za vreme trajanja trenutnog zahteva, kada mu se može pristupiti pomoću getAttribute

# Uslovne operacije sa bean-ovima

- **Bean se može uslovno kreirati**

- Rezultat `jsp:useBean` je novi bean čija će se instanca formirati samo ako se ne može pronaći bean sa istim id i oblasti važenja.

- Ako bean sa istim id i oblasti važenja se pronađe, postojeći bean se jednostavno poveže sa promenljivom pomoću id.

- **Uslovno postavljanje Bean property-ija**

- `<jsp:useBean ... />`

- Se zamenjuje sa

`<jsp:useBean ...>statements</jsp:useBean>`

- statements (`jsp:setProperty` elementi) se izvršavaju samo ako se kreira novi bean, a ne izvršavaju se kada se pronađe postojeći.

# AccessCountBean

```
public class AccessCountBean {
 private String firstPage;
 private int accessCount = 1;
 public String getFirstPage() {
 return(firstPage);
 }
 public void setFirstPage(String firstPage) {
 this.firstPage = firstPage;
 }
 public int getAccessCount() {
 return(accessCount);
 }
 public void setAccessCountIncrement(int increment) {
 accessCount = accessCount + increment;
 }
}
```

# SharedCounts1.jsp

```
<jsp:useBean id="counter"
class="coreservlets.AccessCountBean"
scope="application">
<jsp:setProperty name="counter"
property="firstPage"
value="SharedCounts1.jsp" />
</jsp:useBean>
```

Of SharedCounts1.jsp (this page),

```
SharedCounts2.jsp, and
```

```
SharedCounts3.jsp,
```

```
<jsp:getProperty name="counter" property="firstPage" />
```

was the first page accessed.

```
<P>
```

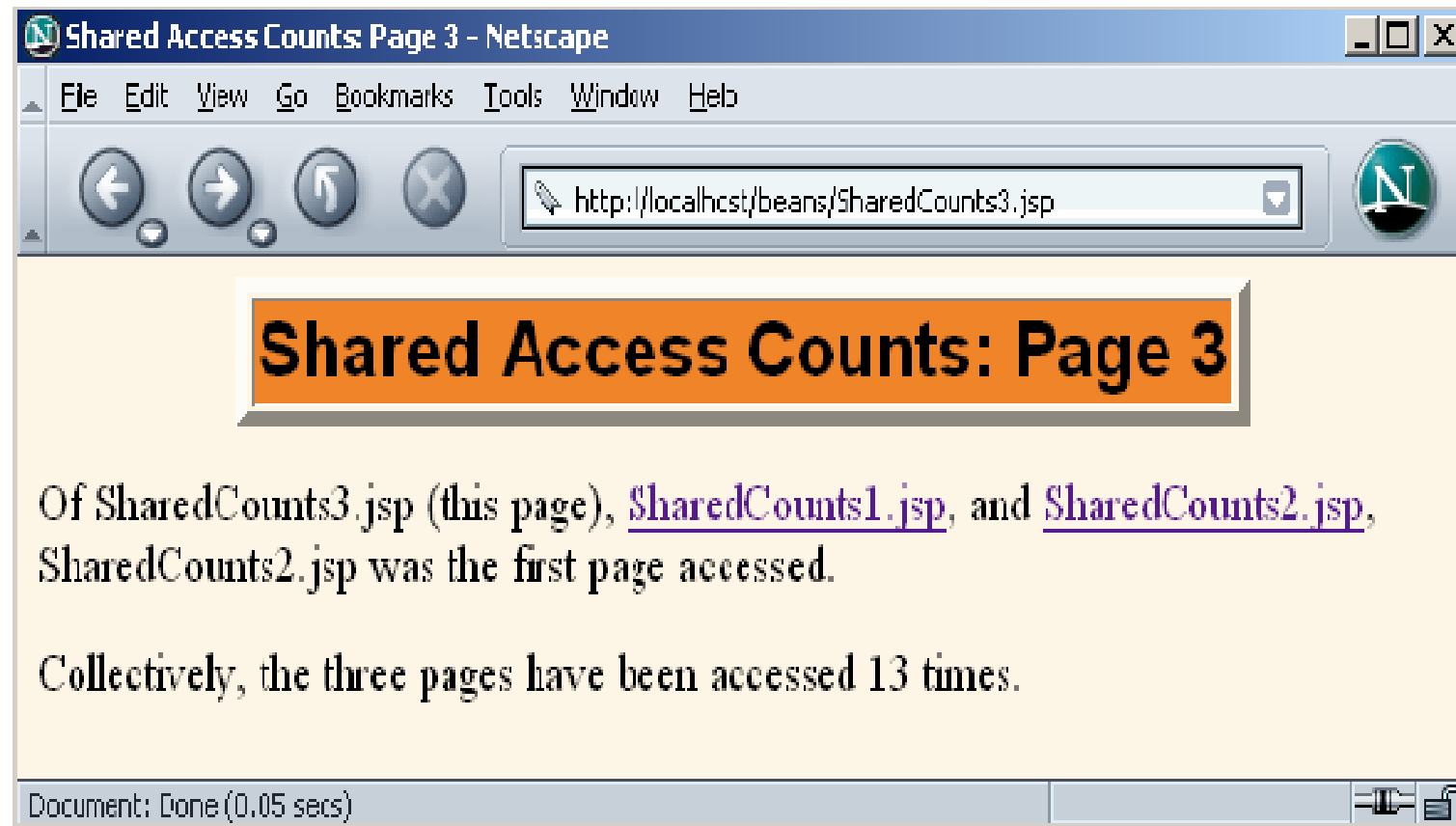
Collectively, the three pages have been accessed

```
<jsp:getProperty name="counter" property="accessCount" />
times.
```

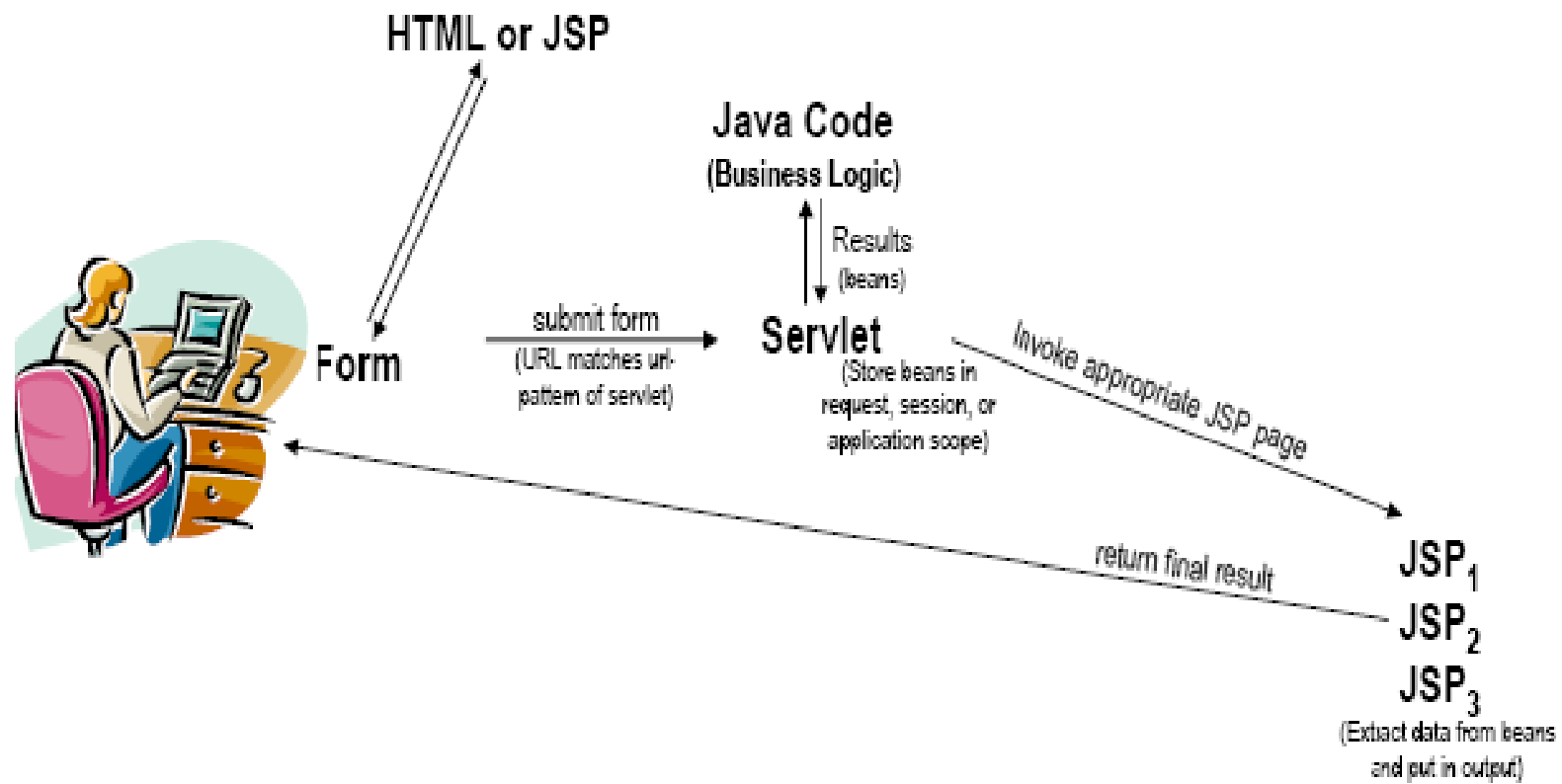
```
<jsp:setProperty name="counter"
property="accessCountIncrement"
value="1" />
```

# Primer

- **SharedCounts2.jsp** će se prvo posetiti.



# MVC



# MVC

- **Određeni framework se može koristiti**
  - Ponekada je upotreba frameworka korisna
    - Struts
    - JavaServer Faces (JSF)
  - Ne zahteva se njihovo korišćenje!
  - Moguće je implementirati MVC pomoću RequestDispatcher i dobiti sasvim zadovoljavajuće rezultate za jednostavne i srednje kompleksne aplikacije
- **MVC u potpunosti menja dizajn sistema**
  - Moguće je koristiti MVC za pojedinačne zahteve
  - Razmišljati kao o MVC pristupu, a ne MVC *arhitektura*
  - Još se naziva i *Model 2* pristup

# **MVC i RequestDispatcher**

**1. Definirati beanove da bi prezentovali podatke**

**2. Koristiti servlet da bi se prihvatio podataka**

– Servlet čita parametre zahteva, proverava unete podatke, itd.

**3. Postaviti beanove**

– Servlet realizuje poslovnu logiku (kod specifičan za aplikaciju) ili kod za pristup podacima da bi došao do rezultata. Resultati se smeštaju u beanove koji su definisani u okviru koraka 1.

**4. Postaviti bean u okviru zahteva, sesije, ili sadržaja servleta**

– Servlet poziva setAttribute u okviru objekata zahteva, sesije, ili sadržaja servleta da bi smestio referencu beanova koji predstavljaju rezultat zahteva.



# **MVC i RequestDispatcher**

## **5. Proslediti zahtev JSP stranici.**

- Servlet prepozna je koja JSP stranica odgovara danoj situaciji i koristi metod forward od RequestDispatcher objekta da bi prebacio kontrolu danoj stranici.

## **6. Preuzeti podatke iz beana.**

- JSP stranica pristupa beanu pomoću jsp:useBean i odgovarajuće oblasti važenja iz koraka 4. Stranica tada koristi jsp:getProperty da bi pristupila beanovim property-ijima.
- JSP stranica ne kreira ili modifikuje bean; ona samo prihvata i pokazuje podatke koje servlet kreira.

## MVC i RequestDispatcher

```
public void doGet(HttpServletRequest request,
 HttpServletResponse response)
 throws ServletException, IOException {
 String operation = request.getParameter("operation");
 if (operation == null) {
 operation = "unknown";
 }
 String address;
 if (operation.equals("order")) {
 address = "/WEB-INF/Order.jsp";
 } else if (operation.equals("cancel")) {
 address = "/WEB-INF/Cancel.jsp";
 } else {
 address = "/WEB-INF/UnknownOperation.jsp";
 }
 RequestDispatcher dispatcher =
 request.getRequestDispatcher(address);
 dispatcher.forward(request, response);
}
```

## **jsp:useBean i MVC nasuprot samo JSP**

- **JSP stranice ne bi trebale da kreiraju objekte**

- Servlet, a ne JSP stranica, trebalo bi da kreira sve objekte koji predstavljaju podatke. Da bi se garantovalo da se u okviru JSP stranice neće kreirati objekti, trebalo bi koristiti

- ```
<jsp:useBean ... type="package.Class" />
```

- umesto

- ```
<jsp:useBean ... class="package.Class" />
```

- **JSP stranica ne bi trebalo da modifikuje objekte**

- Moguće je koristiti jsp:getProperty ali ne jsp:setProperty

# **jsp:useBean i oblasti važenja**

- **request**

- `<jsp:useBean id="..." type="..." scope="request" />`

- **session**

- `<jsp:useBean id="..." type="..." scope="session" />`

- **application**

- `<jsp:useBean id="..." type="..." scope="application" />`

- **page**

- `<jsp:useBean id="..." type="..." scope="page" />`

- ili samo

- `<jsp:useBean id="..." type="..." />`

- Ovakav pristup se ne koristi u okviru MVC (Model 2) arhitekture

# Request-Based deljenje podataka

- **Servlet**

```
ValueObject value = new ValueObject(...);
request.setAttribute("key", value);
RequestDispatcher dispatcher =
 request.getRequestDispatcher ("/WEB-
INF/SomePage.jsp");
dispatcher.forward(request, response);
```

- **JSP 1.2**

```
<jsp:useBean id="key" type="somePackage.ValueObject"
scope="request" />
<jsp:getProperty name="key" property="someProperty" />
```

- **JSP 2.0**

```
${key.someProperty}
```

# Session-Based deljenje podataka

- **Servlet**

```
ValueObject value = new ValueObject(...);
HttpSession session = request.getSession();
session.setAttribute("key", value);
RequestDispatcher dispatcher =
 request.getRequestDispatcher
("/WEB-INF/SomePage.jsp");
dispatcher.forward(request, response);
```

- **JSP 1.2**

```
<jsp:useBean id="key" type="somePackage.ValueObject"
 scope="session" />
<jsp:getProperty name="key" property="someProperty" />
```

- **JSP 2.0**

```
${key.someProperty}
```

# Session-Based deljenje podataka

- **Treba koristiti `response.sendRedirect` umesto `RequestDispatcher.forward`**
- **Situacija sa `sendRedirect`:**
  - Korisnik vidi URL JSP stranice (ako se koristi `RequestDispatcher.forward` korisnik vidi URL servleta)
- **Prednosti `sendRedirect`**
  - Korisnik može pristupiti JSP stranici odvojeno
  - Korisnik može da zapamti adresu JSP stranice
- **Nedostaci `sendRedirect`**
  - Korisnik može da poseti JSP stranici bez pristupanja servletu, pa JSP podaci mogu da budu nedostupni
  - U okviru JSP stranice trebalo bi detektovati ovu situaciju

# Application-Based deljenje podataka

- **Servlet**

```
synchronized(this) {
 ValueObject value = new ValueObject(...);
 getServletContext().setAttribute("key", value);
 RequestDispatcher dispatcher
 =request.getRequestDispatcher
 ("/WEB-INF/SomePage.jsp");
 dispatcher.forward(request, response); }
```

- **JSP 1.2**

```
<jsp:useBean id="key" type="somePackage.ValueObject"
scope="application" />
<jsp:getProperty name="key" property="someProperty" />
```

- **JSP 2.0**

```
${key.someProperty}
```



# Relativni URL u okviru JSP stranice

- **Korist:**

- Prosleđivanje pomoću request dispatcher-a je transparentno klijentu. *Originalni* URL je jedini URL za koji čitač zna.

- **Kakve su posledice?**

- Šta će čitač uraditi sa sledećim tagovima:

- ```
<IMG SRC="foo.gif" ...>
```

- ```
<LINK REL=STYLESHEET
```

- ```
HREF="JSP-Styles.css"
```

- ```
TYPE="text/css">
```

- ```
<A HREF="bar.jsp">...</A>
```

- Odgovor: čitač ih shvata kao relativne u odnosu na URL *servleta*

- **Najjednostavnije rešenje:**

- Koristiti URL koji počinju sa /

Primer Bank Account Balance

- **Bean**
 - BankCustomer
- **Servlet koji popunjava bean i prosleđuje ga odgovarajućoj JSP stranici**
 - Pročitati customer ID, pozvati kod za pristup podacima da bi se popunio BankCustomer
 - U zavisnosti od trenutne vrednosti stanja (balance) računa pozvati odgovarajuću rezultujuću stranu
- **JSP stranice koje prikazuju rezultate**
 - Negativno stanje: stranica sa upozorenjem
 - Regularno stanje: standardna stranica
 - Veoma pozitivno stanje: stranica koja sadrži dodatne reklame
 - Nepoznat customer ID: stranica sa porukom o grešci

Primer Bank Account Balance

```
public class ShowBalance extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        BankCustomer customer
            =BankCustomer.getCustomer(request.getParameter("id"));
        String address;
        if (customer == null) {
            address ="/WEB-INF/bank-account/UnknownCustomer.jsp";
        } else if (customer.getBalance() < 0) {
            address ="/WEB-INF/bank-account/NegativeBalance.jsp";
            request.setAttribute("badCustomer", customer);
        }
        ...
        RequestDispatcher dispatcher =
            request.getRequestDispatcher(address);
        dispatcher.forward(request, response);
    }
}
```

Primer Bank Account Balance negativen

JSP 1.2

```
<BODY>
```

```
<TABLE BORDER=5 ALIGN="CENTER">
```

```
<TR><TH CLASS="TITLE">
```

```
We Know Where You Live!</TABLE>
```

```
<P>
```

```
<IMG SRC="/bank-support/Club.gif" ALIGN="LEFT">
```

```
<jsp:useBean id="badCustomer"
```

```
type="coreservlets.BankCustomer"
```

```
scope="request" />
```

```
Watch out,
```

```
<jsp:getProperty name="badCustomer"
```

```
property="firstName" />,
```

```
we know where you live.
```

```
<P>
```

```
Pay us the $<jsp:getProperty name="badCustomer"
```

```
property="balanceNoSign" />
```

```
you owe us before it is too late!
```

```
</BODY></HTML>
```

Primer Bank Account Balance negativen JSP 2.0

```
<BODY>
```

```
<TABLE BORDER=5 ALIGN="CENTER">
```

```
<TR><TH CLASS="TITLE">
```

```
We Know Where You Live!</TABLE>
```

```
<P>
```

```
<IMG SRC="/bank-support/Club.gif" ALIGN="LEFT">
```

```
Watch out,
```

```
${badCustomer.firstName},
```

```
we know where you live.
```

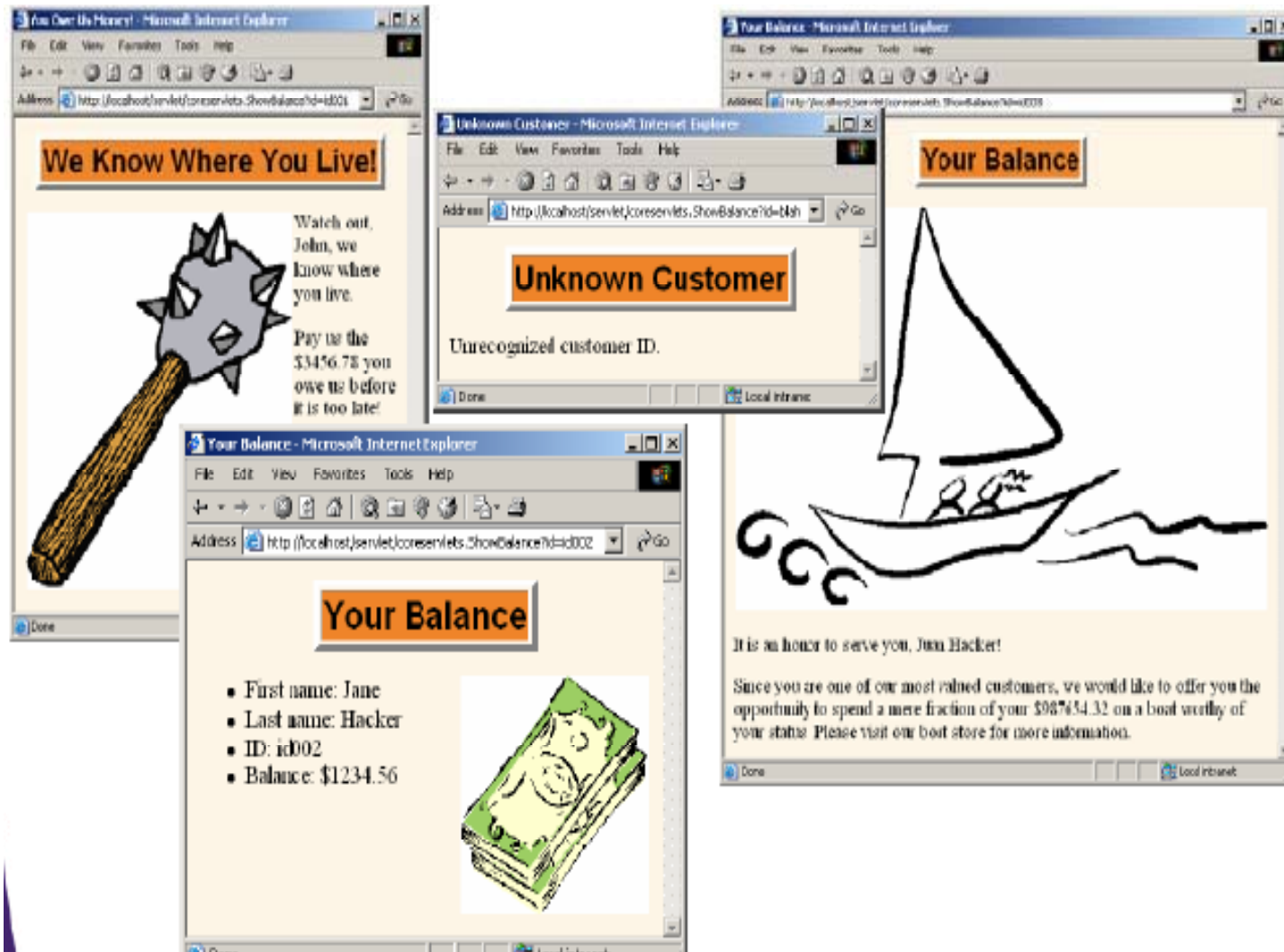
```
<P>
```

```
Pay us the $$${badCustomer.balanceNoSign}
```

```
you owe us before it is too late!
```

```
</BODY></HTML>
```

Primer Bank Account Balance



Deljenje podataka: Request

- **Cilj**
 - Prikazati slučajan broj korisniku
- **Tip deljenja podataka**
 - Svaki zahtev dobija novi broj, tako da je odgovarajuće deljenje podataka na nivou zahteva (request based).

Deljenje podataka: Request Bean

```
package coreservlets;  
  
public class NumberBean {  
    private double num = 0;  
    public NumberBean(double number) {  
        setNumber(number);  
    }  
    public double getNumber() {  
        return(num);  
    }  
    public void setNumber(double number) {  
        num = number;  
    }  
}
```


Deljenje podataka: Request JSP 1.2

```
<BODY>
```

```
<jsp:useBean id="randomNum"  
type="coreservlets.NumberBean"  
scope="request" />
```

```
<H2>Random Number:
```

```
<jsp:getProperty name="randomNum"  
property="number" />
```

```
</H2>
```

```
</BODY></HTML>
```

Deljenje podataka: Request JSP 2.0

...

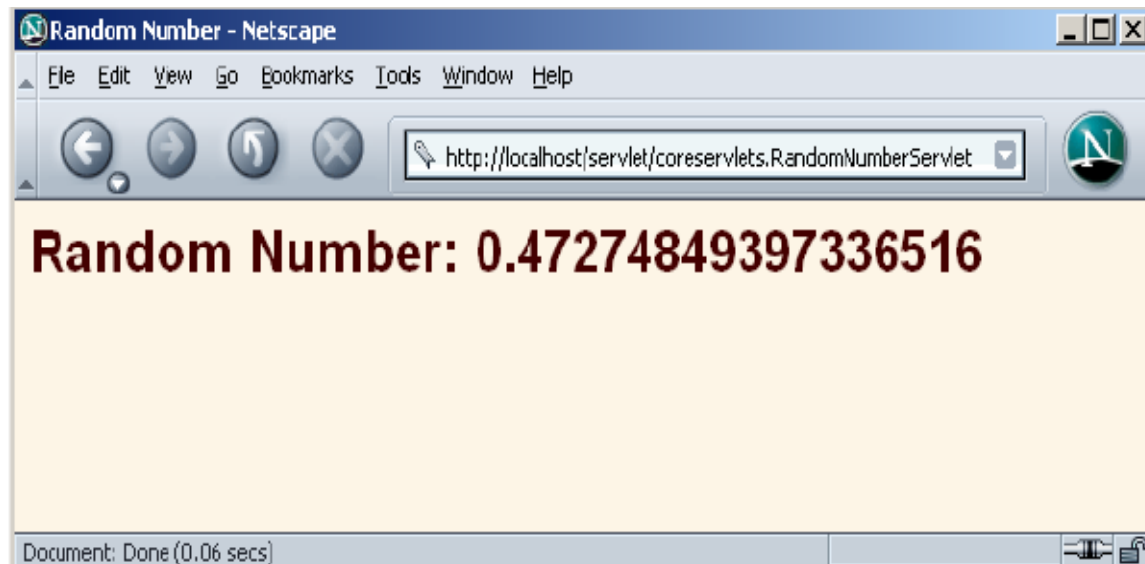
<BODY>

<H2>Random Number:

`${randomNum.number}`

</H2>

</BODY> </HTML>



Deljenje podataka: Session

- **Cilj**

- Prikazati korisnikovo ime i prezime.
- Ako nam korisnik ne definiše svoje ime, želimo da mu prikažemo ime koje nam je prethodno definisao.
- Ako nam korisnik ne definiše svoje ime i ne možemo da pronađemo nijedno prethodno definisano ime, potrebno je prikazati upozorenje.

- **Tip deljenja podataka**

- Podatak se smešta za svakog klijenta, tako da je deljenje podataka na nivou sesije (session-based) odgovarajuće.

Deljenje podataka: Session Bean

```
package coreservlets;
public class NameBean {
    private String firstName = "Missing first name";
    private String lastName = "Missing last name";
    public NameBean() {}
    public NameBean(String firstName, String lastName) {
        setFirstName(firstName);
        setLastName(lastName);
    }
    public String getFirstName() {
        return(firstName);
    }
    ...
}
```

Deljenje podataka: Session – servlet

```
public class RegistrationServlet extends HttpServlet {  
    public void doGet(HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
        HttpSession session = request.getSession();  
        NameBean nameBean =  
            (NameBean)session.getAttribute("nameBean");  
        if (nameBean == null) {  
            nameBean = new NameBean();  
            session.setAttribute("nameBean", nameBean);  
        }  
    }  
}
```

Deljenje podataka: Session – servlet

```
String firstName =  
request.getParameter("firstName");  
if ((firstName != null) &&  
(!firstName.trim().equals(""))) {  
    nameBean.setFirstName(firstName);  
}  
String lastName =  
request.getParameter("lastName");  
if ((lastName != null) &&  
(!lastName.trim().equals(""))) {  
    nameBean.setLastName(lastName);  
}  
String address =  
"/WEB-INF/mvc-sharing/ShowName.jsp";  
RequestDispatcher dispatcher =  
request.getRequestDispatcher(address);  
dispatcher.forward(request, response);  
}  
}
```

Deljenje podataka: Session – JSP 1.2

```
<BODY>
```

```
<H1>Thanks for Registering</H1>
```

```
<jsp:useBean id="nameBean"  
type="coreservlets.NameBean"  
scope="session" />
```

```
<H2>First Name:
```

```
<jsp:getProperty name="nameBean"  
property="firstName" /></H2>
```

```
<H2>Last Name:
```

```
<jsp:getProperty name="nameBean"  
property="lastName" /></H2>
```

```
</BODY></HTML>
```

Deljenje podataka: Session – JSP 2.0

<BODY>

<H1>Thanks for Registering</H1>

<H2>First Name:

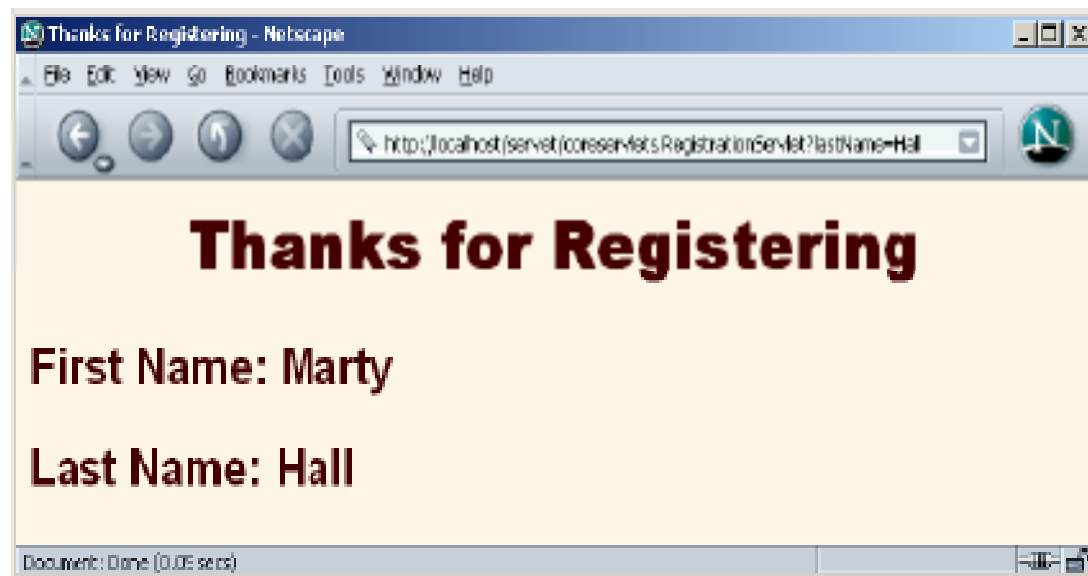
`${nameBean.firstName}`</H2>

<H2>Last Name:

`${nameBean.lastName}`</H2>

</BODY></HTML>

Deljenje podataka: Session – JSP 2.0



Deljenje podataka: Application

- **Cilj**

- Prikazati prost broj određene dužine.
- Ako korisnik ne defniše željenu dužinu, trebalo bi prikazati bilo koji prost broj koji se koristio za bilo kog korisnika.

- **Tip deljenja podataka**

- Podatak se deli između više klijenata, tako da je deljenje podataka na nivou aplikacije (application-based) odgovarajuće.

Deljenje podataka: Application - Bean

```
import java.math.BigInteger;
public class PrimeBean {
    private BigInteger prime;
    public PrimeBean(String lengthString) {
        int length = 150;
        try {
            length = Integer.parseInt(lengthString);
        } catch (NumberFormatException nfe) {}
        setPrime(Primes.nextPrime(Primes.random(length)));
    }
    public BigInteger getPrime() {
        return(prime);
    }...
}
```

Deljenje podataka: Application - Servlet

```
public class PrimeServlet extends HttpServlet {  
    public void doGet(HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
        String length = request.getParameter("primeLength");  
        ServletContext context = getServletContext();  
        synchronized(this) {  
            if ((context.getAttribute("primeBean") == null) ||  
                (length != null)) {PrimeBean primeBean = new PrimeBean(length);  
                context.setAttribute("primeBean", primeBean);  
            }  
            String address =  
                "/WEB-INF/mvc-sharing/ShowPrime.jsp";  
            RequestDispatcher dispatcher =  
                request.getRequestDispatcher(address);  
            dispatcher.forward(request, response);  
        }  
    }  
}
```

Deljenje podataka: Application – JSP 1.2

```
<BODY>
```

```
<H1>A Prime Number</H1>
```

```
<jsp:useBean id="primeBean"  
    type="coreservlets.PrimeBean" scope="application" />
```

```
<jsp:getProperty name="primeBean" property="prime" />
```

```
</BODY></HTML>
```

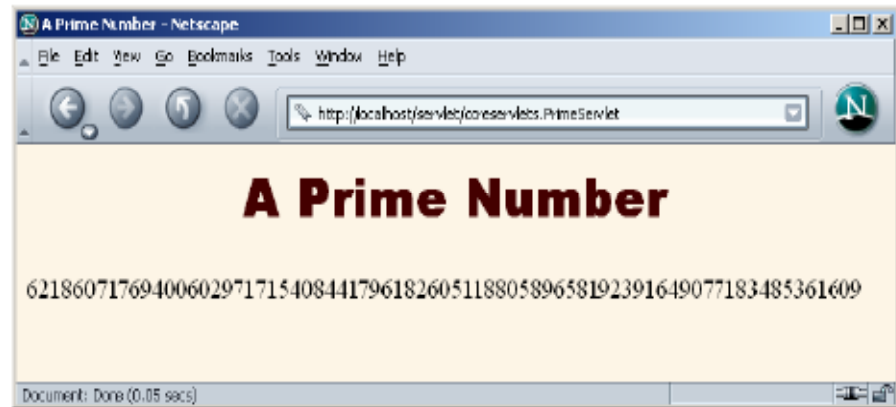
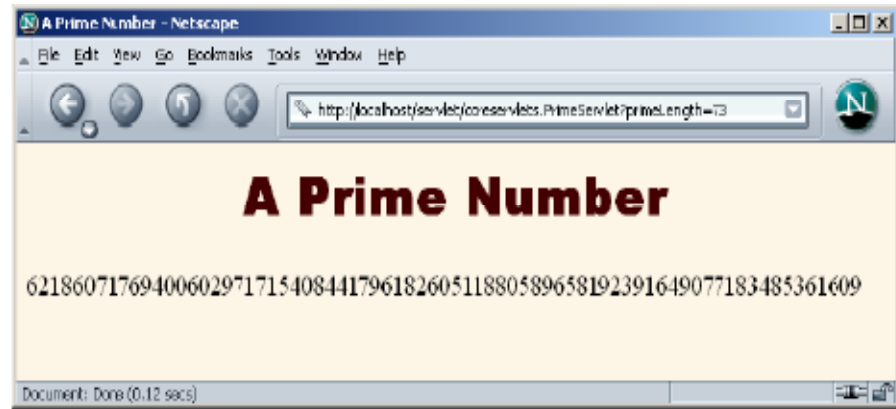
Deljenje podataka: Application – JSP 1.2

<BODY>

<H1>A Prime Number</H1>

`${primeBean.prime}`

</BODY> </HTML>



Prosleđivanje sa JSP stranice

```
<% String destination;  
if (Math.random() > 0.5) {  
destination = "/examples/page1.jsp";  
} else {  
destination = "/examples/page2.jsp";  
}  
%>  
<jsp:forward page="<%= destination %>" />
```

- **Dozvoljeno, ali loša ideja**

- Poslovna i kontrolna logika pripada servletima
- Zadatak JSP stranice je prezentacija korisniku

Korišćenje include

- **Pomoću metoda forward objekta RequestDispatcher:**
 - Kontrola se nepovratno predaje novoj stranici
 - Originalna stranica ne može generisati bilo kakav izlaz
- **Pomoću metoda include objekta RequestDispatcher:**
 - Kontrola se trenutno predaje novoj stranici
 - Originalna stranica može generisati izlaz pre i posle uključene stranice
 - Originalni servlet ne vidi izlaz uključene stranice
 - Korisno za portale: JSP prikazuje delove, ali se delovi uređuju različito za različite korisnike

Korišćenje include

```
response.setContentType("text/html");
String firstTable, secondTable, thirdTable;
if (someCondition) {
    firstTable = "/WEB-INF/Sports-Scores.jsp";
    secondTable = "/WEB-INF/Stock-Prices.jsp";
    thirdTable = "/WEB-INF/Weather.jsp";
} else if (...) { ... }
RequestDispatcher dispatcher =
    request.getRequestDispatcher("/WEB-INF/Header.jsp");
dispatcher.include(request, response);
dispatcher =
    request.getRequestDispatcher(firstTable);
dispatcher.include(request, response);
dispatcher =
    request.getRequestDispatcher(secondTable);
dispatcher.include(request, response);
dispatcher =
    request.getRequestDispatcher(thirdTable);
dispatcher.include(request, response);
dispatcher =
    request.getRequestDispatcher("/WEB-INF/Footer.jsp");
dispatcher.include(request, response);
```

Expression jezik (EL)

- **Zašto ga koristiti**
- **Osnovna sintaksa**
- **Relacija između expression jezik i MVC arhitekture**
- **Referenciranje smeštenih promenljivih**
- **Pristup bean property-ijima, elementima niza, elementima liste i Map ulazima**
- **Upotreba operatora expression jezika**
- **Uslovno izvršavanje izraza**

Nedostaci MVC

- **Glavni nedostatak je u završnom koraku: prezentovanje rezultata u okviru JSP stranice.**
 - Koriste se `jsp:useBean` i `jsp:getProperty`
 - Nespretno i preopširno
 - Ne može se pristupiti bean podproperty-ijima
 - JSP scripting elementi
 - Rezultat je kod koji se teško otežava
 - Umanjuje se uticaj MVC arhitekture.
- **Cilj**
 - Upotrebiti koncizniji pristup
 - Omogućiti pristup bean podproperty-ijima
 - Jednostavnija sintaksa pristupa za Web developere

Upotreba EL

- **Ako se koristi MVC pristup u okviru servera koji podržavaju JSP 2.0 sa trenutnom web.xml verzijom, potrebno je promeniti samo:**

```
<jsp:useBean id="someName"  
type="somePackage.someClass"  
scope="request, session, or application" />  
<jsp:getProperty name="someName"  
property="someProperty" />
```

- **U sledeći kod:**

```
${someName.someProperty}
```

Prednosti upotrebe EL

- **Koncizniji pristup smeštenim objektima.**
 - Da bi se pristupilo “smeštenim promenljivama” (objekti smešteni metodom `setAttribute` u okviru `PageContext`, `HttpServletRequest`, `HttpSession`, ili `ServletContext`), na primer nekoj promenljivoj `saleItem`, koristi se `${saleItem}`.
- **Skraćena notacija za bean property-ije.**
 - Da bi se pristupilo `companyName` property-iju (n.p., rezultat dobijen od metoda `getCompanyName`) smeštene promenljive pod imenom `company`, koristi se `${company.companyName}`. Da bi se pristupilo property-iju `firstName` od property-ija `president` smeštene promenljive `company`, koristi se `${company.president.firstName}`.
- **Jednostavniji pristup collection elementima.**
 - Pristup elementima niza (`array`), `List`, ili `Map`, koristi se `${variable[indexOrKey]}`. Treba primetiti da je potrebno realizovati prikaz indeksa ili ključa u formi legalnoj za imena Java promenljivih.

Prednosti upotrebe EL

- **Jednostavan pristup parametrima zahteva, cookies, i drugih podataka zahteva.**
 - Da bise pristupilo standardnim tipovima podataka zahteva, moguće je koristiti jedan od nekoliko predefinisanih implicitnih objekata.
- **Mali, ali koristan skup jednostavnih operatora.**
 - Za rad sa objektima u okviru EL izraza, može se koristiti bilo koji od nekoliko aritetičkih, relacionih, logičkih, ili empty-testing operatora.
- **Uslovni rezultati**
 - Da bi se izabrao prikaz između nekoliko opcija, nije potrebno koristiti Java skripte. Moguće je koristiti `${test ? option1 : option2}`.
- **Automatska konverzija tipova.**
 - EL uklanja potrebu za korišćenjem konverzije tipova i omogućava ovaj proces automatskim.
- **Bez vrednosti, umesto greške.**
 - U većini slučajeva, nedefinisana promenljiva ili `NullPointerException` prikazaće se kao prazni stringovi, a neće se dogoditi izuzetak.

Aktiviranje EL

- **Moguće je upotrebljavati samo u okviru servera koji imaju podršku za JSP 2.0 (servlets 2.4)**

- Na primer Tomcat 5, a ne Tomcat 4
 - Za celokupnu listu kompatibilnih servera, pogledati <http://theserverside.com/reviews/matrix.tss>

- **Potrebno je koristiti JSP 2.0 web.xml file**

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation=  
"http://java.sun.com/xml/ns/j2ee web-app_2_4.xsd"  
version="2.4">
```

...

```
</web-app>
```

Osnove EL

- **Osnovna forma: `${expression}`**

- Ovi EL elementi se mogu pojaviti kako u običnom HTML tekstu tako i u okviru JSP tag atributa, i omogućiti izvršavanje regularnih JSP izraza. Na primer:

- ```

```

- ```
<LI>Name: ${expression1}
```

- ```
Address: ${expression2}
```

- ```
</UL>
```

- ```
<jsp:include page="${expression3}" />
```

- **EL i tag atributi**

- Može se koristiti više izraza u okviru jednog atributa (moguće ih je kombinovati i sa statičkim tekstom), tako da se na kraju konvertuju u stringove i izvrši njihova konkatencija. Na primer:

- ```
<jsp:include page="${expr1}blah${expr2}" />
```


EL problem

▪ Scenario

- Koristi se `${something}` u okviru JSP stranice
- Postoji `"${something}"` u generisanom izlazu
- Nije promenjen web.xml file da referiše na verziju servleta 2.4
- Prenese se Web aplikacija na server i restartuje se
- I dalje postoji `"${something}"` u generisanom izlazu

▪ Zašto?

- JSP stranica je prevedena u servlet
- I to verziju servleta koji ignorišeEL

▪ Rešenje

- Presnimiti JSP stranicu da izvrši update datuma promene

Sprečavanje izvršavanja EL izraza

- **Šta se dešava ako JSP stranica sadrži \${ ?**
- **Potrebno je deaktivirati EL u trenutnoj Web aplikaciji.**
 - U okviru web.xml fajla referisati na servlet verziju 2.3 (JSP 1.2) ili raniju.
- **Deaktivirati EL u više JSP stranica.**
 - Koristiti jsp-property-group element u okviru web.xml
- **Deaktivirati EL u pojedinačnim JSP stranicama.**
 - Koristiti `<%@ page isELIgnored="true" %>`
- **Deaktivirati pojedinačne EL elemente.**
 - U okviru JSP 1.2 stranica (bez promena web.xml), moguće je zameniti \$ sa `$`, HTML karakterom za \$.
 - U okviru JSP 2.0 stranica koje sadrže i EL izraze i \${ moguće je koristiti `\${` kada se želi generisati \${

Sprečavanje izvršavanja skripleta

- **Da bi se sprečilo pisanje skripleta moguće je koristiti scripting-invalid u okviru web.xml**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation=
"http://java.sun.com/xml/ns/j2ee web-app_2_4.xsd"
version="2.4">
<jsp-property-group>
<url-pattern>*.jsp</url-pattern>
<scripting-invalid>true</scripting-invalid>
</jsp-property-group>
</web-app>
```

Pristup smeštenim promenljivama

- **`${varName}`**

- Rezultat je pretraga objekata tipa `PageContext`, `HttpServletRequest`, `HttpSession`, `ServletContext`, u **navedenom redosledu** i prikaz objekta koji sadrži navedeno ime
- Upotreba `PageContext` ne odgovara MVC pristupu.

- **Ekvivalentne forme**

- `${name}`
- `<%= pageContext.findAttribute("name") %>`
- `<jsp:useBean id="name"`
 `type="somePackage.SomeClass"`
 `scope="...">`
 `<%= name %>`

Primer

```
public class ScopedVars extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        request.setAttribute("attribute1", "First Value");
        HttpSession session = request.getSession();
        session.setAttribute("attribute2", "Second Value");
        ServletContext application = getServletContext();
        application.setAttribute("attribute3",
            new java.util.Date());
        request.setAttribute("repeated", "Request");
        session.setAttribute("repeated", "Session");
        application.setAttribute("repeated", "ServletContext");
        RequestDispatcher dispatcher =
            request.getRequestDispatcher("/el/scoped-vars.jsp");
        dispatcher.forward(request, response);
    }
}
```

Primer

```
<!DOCTYPE ...>
```

```
...
```

```
<TABLE BORDER=5 ALIGN="CENTER">
```

```
<TR><TH CLASS="TITLE">
```

Accessing Scoped Variables

```
</TABLE>
```

```
<P>
```

```
<UL>
```

```
<LI><B>attribute1:</B> ${attribute1}
```

```
<LI><B>attribute2:</B> ${attribute2}
```

```
<LI><B>attribute3:</B> ${attribute3}
```

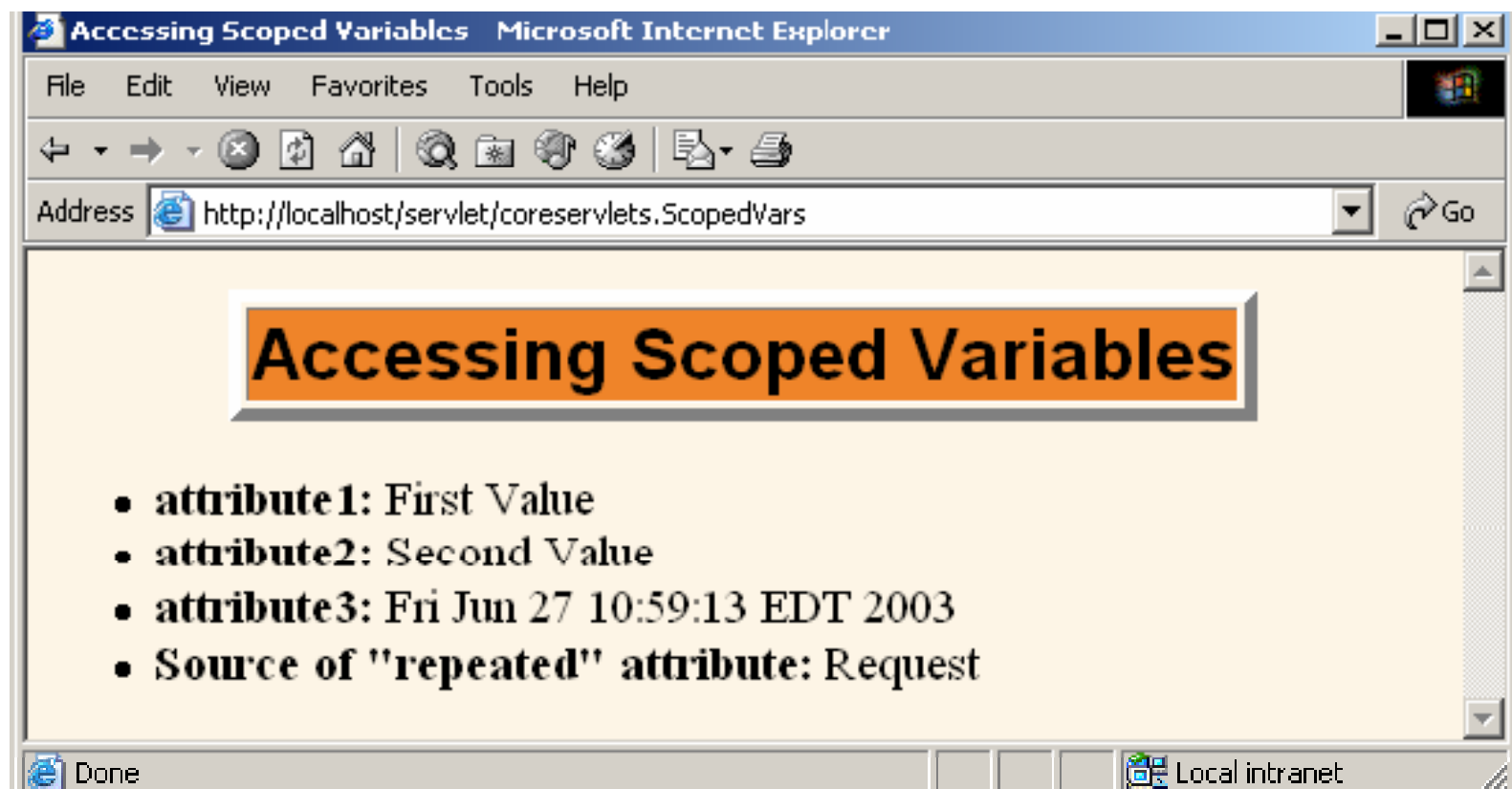
```
<LI><B>Source of "repeated" attribute:</B>
```

```
${repeated}
```

```
</UL>
```

```
</BODY></HTML>
```

Primer



Pristup bean property-ijima

- **`${varName.propertyName}`**
 - Značenje je pronaći smeštenu promenljivu sa datim imenom i prikazati vrednost navedenog bean property-ija
- **Ekvivalentne forme**
 - `${customer.firstName}`
 - `<%@ page import="coreservlets.NameBean" %>`
`<% NameBean person =`
`(NameBean)pageContext.findAttribute("customer");`
`%>`
`<%= person.getFirstName() %>`

Pristup bean property-ijima

- **Ekvivalentne forme**

- `${customer.firstName}`
- `<jsp:useBean id="customer" type="coreservlets.NameBean" scope="request, session, or application" />`
`<jsp:getProperty name="customer" property="firstName" />`

- **Ovaj pristup je bolji nego skript na prethodnom slajdu.**

- Ali zahteva da se poznaje oblast važenja
- Ne radi za podproperty-ije.
- Nije ne-Java ekvivalent
`${customer.address.zipCode}`

Primer

```
public class BeanProperties extends HttpServlet {  
    public void doGet(HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
        NameBean name = new NameBean("Marty", "Hall");  
        CompanyBean company =  
            new CompanyBean("coreservlets.com",  
                "J2EE Training and Consulting");  
        EmployeeBean employee =  
            new EmployeeBean(name, company);  
        request.setAttribute("employee", employee);  
        RequestDispatcher dispatcher =  
            request.getRequestDispatcher  
                ("/el/bean-properties.jsp");  
        dispatcher.forward(request, response);  
    }  
}
```

Primer

```
public class EmployeeBean {  
    private NameBean name;  
    private CompanyBean company;  
    public EmployeeBean(NameBean name, CompanyBean company) {  
        setName(name);  
        setCompany(company);  
    }  
    public NameBean getName() { return(name); }  
    public void setName(NameBean newName) {  
        name = newName;  
    }  
    public CompanyBean getCompany() { return(company); }  
    public void setCompany(CompanyBean newCompany) {  
        company = newCompany;  
    }  
}
```

Primer

```
public class NameBean {  
    private String firstName = "Missing first name";  
    private String lastName = "Missing last name";  
    public NameBean() {}  
    public NameBean(String firstName, String lastName) {  
        setFirstName(firstName);  
        setLastName(lastName);  
    }  
    public String getFirstName() {  
        return(firstName);  
    }  
    public void setFirstName(String newFirstName) {  
        firstName = newFirstName;  
    }...  
}
```

Primer

```
public class CompanyBean {  
    private String companyName;  
    private String business;  
    public CompanyBean(String companyName, String business) {  
        setCompanyName(companyName);  
        setBusiness(business);  
    }  
    public String getCompanyName() { return(companyName); }  
    public void setCompanyName(String newCompanyName) {  
        companyName = newCompanyName;  
    }  
    public String getBusiness() { return(business); }  
    public void setBusiness(String newBusiness) {  
        business = newBusiness;  
    }  
}
```

Primer

```
<!DOCTYPE ...>
```

```
...
```

```
<UL>
```

```
<LI><B>First Name:</B>
```

```
${employee.name.firstName}
```

```
<LI><B>Last Name:</B>
```

```
${employee.name.lastName}
```

```
<LI><B>Company Name:</B>
```

```
${employee.company.companyName}
```

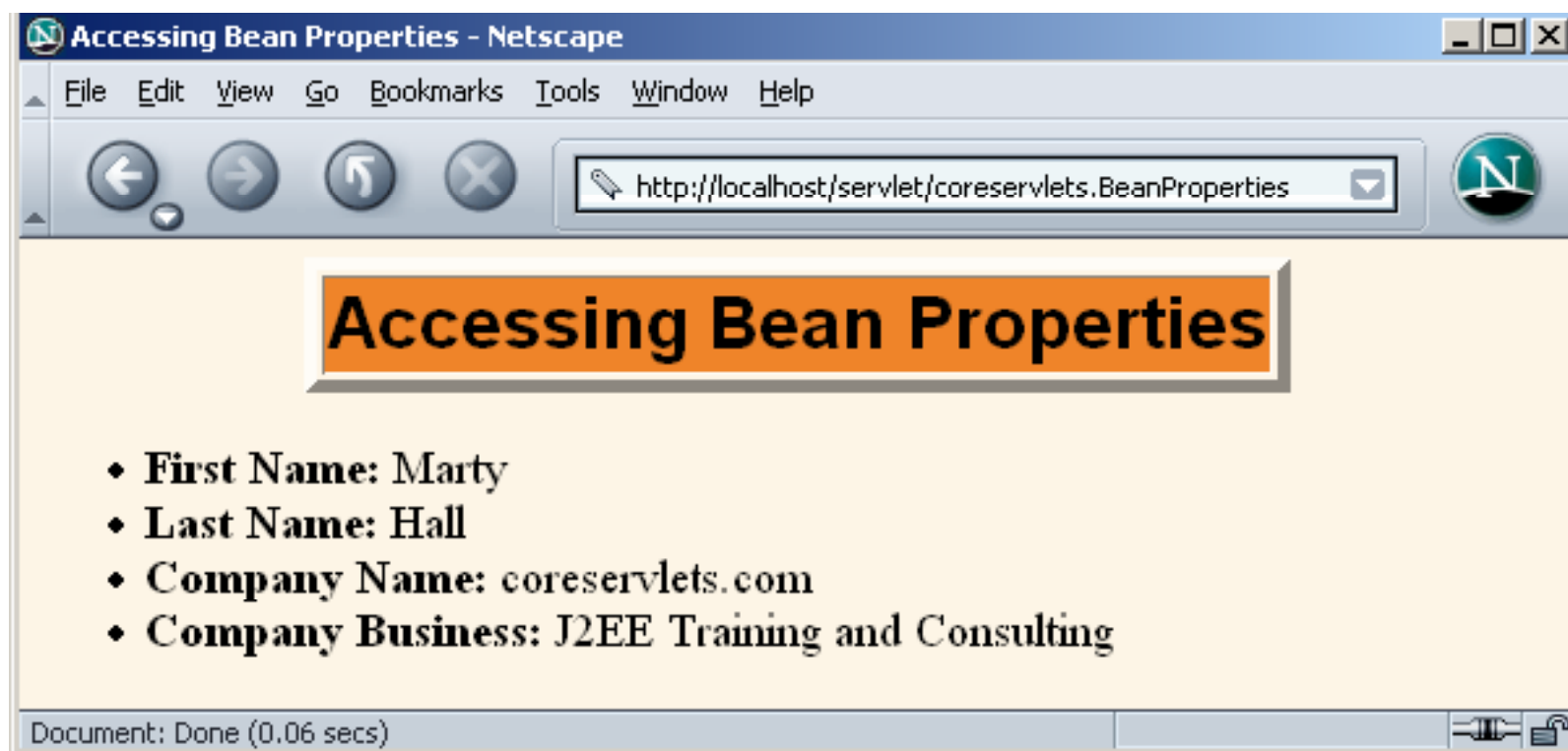
```
<LI><B>Company Business:</B>
```

```
${employee.company.business}
```

```
</UL>
```

```
</BODY></HTML>
```

Primer



Notacija

- **Ekvivalentne forme**

- `${name.property}`
- `${name["property"]}`

- **Razlozi za korišćenje notacije nizova**

- Zbog pristupa nizovima, listama, i drugim klasama kolekcija
- Da bi se dobilo ime property-ija u trenutku zahteva.
 - `{name1[name2]}` (nema navodnika oko name2)
- Da bi se koristila imena koja su zabranjena kao imena Java promenljivih
 - `{foo["bar-baz"]}`
 - `{foo["bar.baz"]}`

Pristup kolekcijama

- `${attributeName[entryName]}`
- **Može se upotrebiti kod**
 - Nizova. Ekvivalentno sa
 - `theArray[index]`
 - Lista. Ekvivalentno sa
 - `theList.get(index)`
 - Mapa. Ekvivalentno sa
 - `theMap.get(keyName)`
- **Ekvivalentne forme (za HashMap)**
 - `${stateCapitals["maryland"]}`
 - `${stateCapitals.maryland}`
 - Sledeći element je nelegalan, jer 2 nije legalno ime promenljive
 - `${listVar.2}`

Pristup kolekcijama

```
public class Collections extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        String[] firstNames = { "Bill", "Scott", "Larry" };
        ArrayList lastNames = new ArrayList();
        lastNames.add("Ellison");
        lastNames.add("Gates");
        lastNames.add("McNealy");
        HashMap companyNames = new HashMap();
        companyNames.put("Ellison", "Sun");
        companyNames.put("Gates", "Oracle");
        companyNames.put("McNealy", "Microsoft");
        request.setAttribute("first", firstNames);
        request.setAttribute("last", lastNames);
        request.setAttribute("company", companyNames);
        RequestDispatcher dispatcher =
            request.getRequestDispatcher("/el/collections.jsp");
        dispatcher.forward(request, response);
    }
}
```

Primer

```
<!DOCTYPE ...>
```

```
...
```

```
<BODY>
```

```
<TABLE BORDER=5 ALIGN="CENTER">
```

```
<TR><TH CLASS="TITLE">
```

Accessing Collections

```
</TABLE>
```

```
<P>
```

```
<UL>
```

```
<LI>${first[0]} ${last[0]} (${company["Ellison"]})
```

```
<LI>${first[1]} ${last[1]} (${company["Gates"]})
```

```
<LI>${first[2]} ${last[2]} (${company["McNealy"]})
```

```
</UL>
```

```
</BODY></HTML>
```

Primer



Predefinisane promenljive

- **pageContext. PageContext objekat.**
 - `${pageContext.session.id}`
- **param i paramValues. Parametri zahteva.**
 - `${param.custID}`
- **header i headerValues. Hederi zahteva.**
 - `${header.Accept}` ili `${header["Accept"]}`
 - `${header["Accept-Encoding"]}`
- **cookie. Cookie objekat (a ne cookie vrednost).**
 - `${cookie.userCookie.value}` ili `${cookie["userCookie"].value}`
- **initParam. Sadrži parametre inicijalizacije sadržaja.**
- **pageScope, requestScope, sessionScope, applicationScope.**
 - Umesto pretrage oblasti važenja.
- **Problem**
 - Upotreba implicitnih objekata ne odgovara MVC modelu

Predefinisane promenljive primer

```
<!DOCTYPE ...>
```

```
...
```

```
<P>
```

```
<UL>
```

```
<LI><B>test Request Parameter:</B>
```

```
  ${param.test}
```

```
<LI><B>User-Agent Header:</B>
```

```
  ${header["User-Agent"]}
```

```
<LI><B>JSESSIONID Cookie Value:</B>
```

```
  ${cookie.JSESSIONID.value}
```

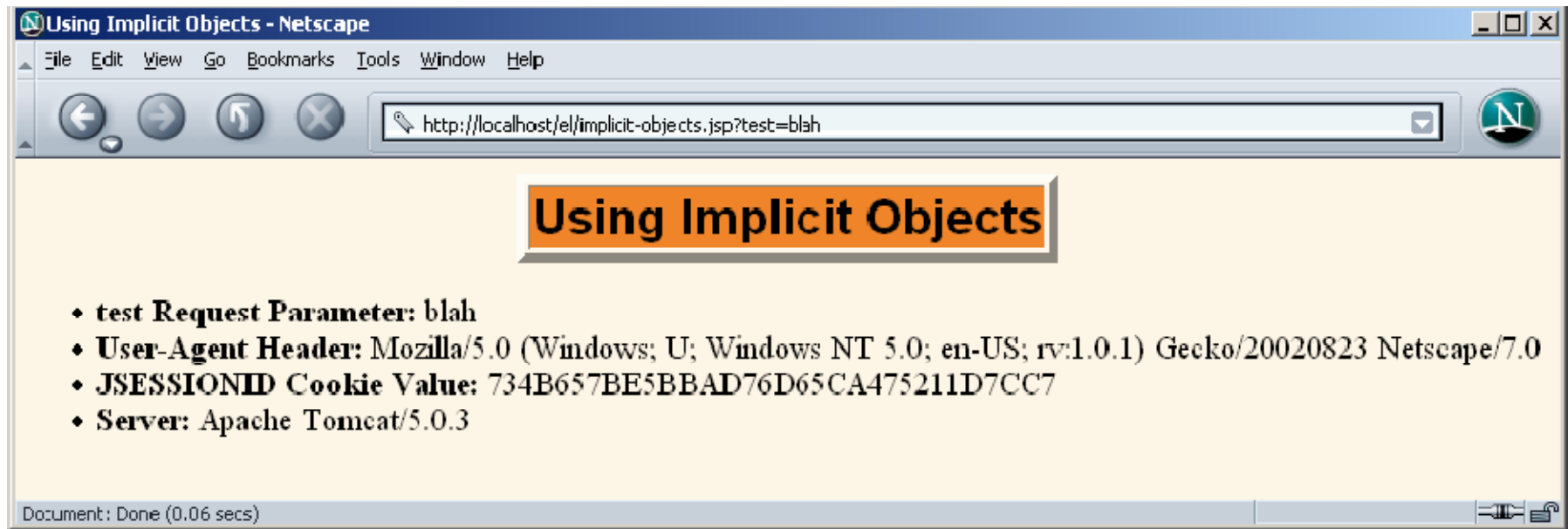
```
<LI><B>Server:</B>
```

```
  ${pageContext.servletContext.serverInfo}
```

```
</UL>
```

```
</BODY></HTML>
```

Predefinisane promenljive primer



EL operatori

- **Aritmetički**

- + - * / div % mod

- **Relacioni**

- == eq != ne < lt > gt <= le >= ge

- **Logički**

- && and || or ! Not

- **Bez vrednosti**

- Bez vrednosti

- True za null, prazan string, prazan niz, prazna lista, prazna map.

- **Upozorenje**

- Koristiti veoma retko da bi se zaštitio MVC model

Primer

```
<TABLE BORDER=1 ALIGN="CENTER">
<TR><TH CLASS="COLORED" COLSPAN=2>Arithmetic Operators
<TH CLASS="COLORED" COLSPAN=2>Relational Operators
<TR><TH>Expression<TH>Result<TH>Expression<TH>Result
<TR ALIGN="CENTER">
<TD>\${3+2-1}<TD>\${3+2-1}
<TD>\${1<2}<TD>\${1<2}
<TR ALIGN="CENTER">
<TD>\${"1"+2}<TD>\${"1"+2}
<TD>\${"a"<"b"}<TD>\${"a"<"b"}
<TR ALIGN="CENTER">
<TD>\${1 + 2*3 + 3/4}<TD>\${1 + 2*3 + 3/4}
<TD>\${2/3 >= 3/2}<TD>\${2/3 >= 3/2}
<TR ALIGN="CENTER">
<TD>\${3%2}<TD>\${3%2}
<TD>\${3/4 == 0.75}<TD>\${3/4 == 0.75}
```

Primer

EL Operators - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address <http://localhost/el/operators.jsp> Go

EL Operators

Arithmetic Operators		Relational Operators	
Expression	Result	Expression	Result
<code>\${3+2-1}</code>	4	<code>\${1<2}</code>	true
<code>\${"1"+2}</code>	3	<code>\${"a"<"b"}</code>	true
<code>\${1 + 2*3 + 3/4}</code>	7.75	<code>\${2/3 >= 3/2}</code>	false
<code>\${3%2}</code>	1	<code>\${3/4 == 0.75}</code>	true
<code>\${(8 div 2) mod 3}</code>	1.0	<code>\${null == "test"}</code>	false

Logical Operators		empty Operator	
Expression	Result	Expression	Result
<code>\${(1<2) && (4<3)}</code>	false	<code>\${empty ""}</code>	true
<code>\${(1<2) (4<3)}</code>	true	<code>\${empty null}</code>	true
<code>\${!(1<2)}</code>	false	<code>\${empty param.blah}</code>	true

Done Local intranet

Uslovno izvršavanje

- **`${ test ? expression1 : expression2 }`**
 - Izvršava se Evaluates test i prikazuje se ili expression1 ili expression2
- **Problems**
 - Relativno slabo
 - Upotreba `c:if` i `c:choose` pomoću JSTL (The JavaServer Pages Standard Tag Library)
 - Dozvoljava rad sa poslovnom/procesnom logikom u okviru JSP strana.
 - Trebalo bi da se koristi samo za prezentacionu logiku.
 - I u tim slučajevima postoje alternative

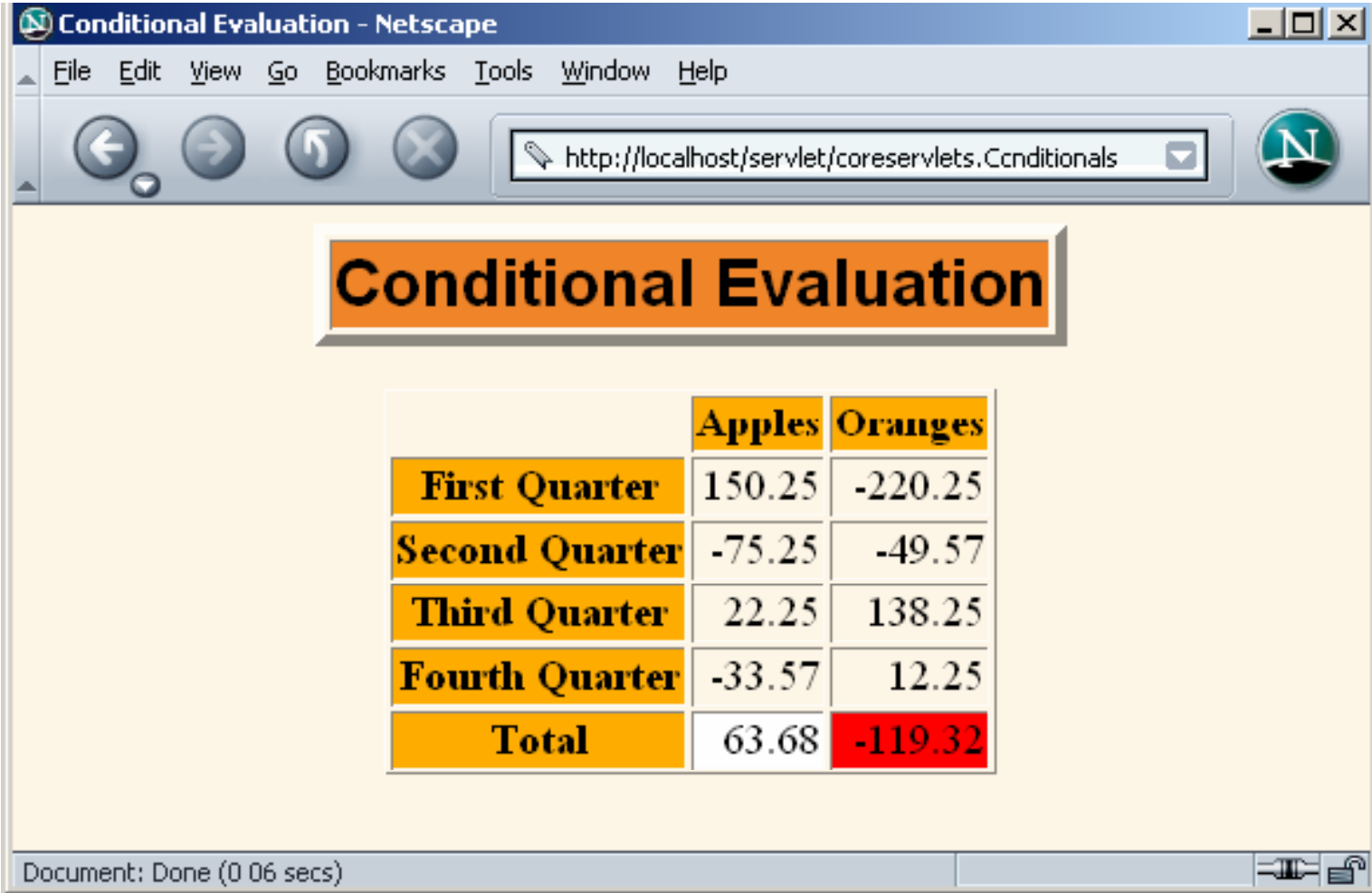
Primer

```
public class Conditionals extends HttpServlet {  
    public void doGet(HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
        SalesBean apples =  
            new SalesBean(150.25, -75.25, 22.25, -33.57);  
        SalesBean oranges =  
            new SalesBean(-220.25, -49.57, 138.25, 12.25);  
        request.setAttribute("apples", apples);  
        request.setAttribute("oranges", oranges);  
        RequestDispatcher dispatcher =  
            request.getRequestDispatcher  
            ("/el/conditionals.jsp");  
        dispatcher.forward(request, response);  
    }  
}
```

Primer

```
public class SalesBean {  
    private double q1, q2, q3, q4;  
    public SalesBean(double q1Sales,  
        double q2Sales,  
        double q3Sales,  
        double q4Sales) {  
        q1 = q1Sales; q2 = q2Sales;  
        q3 = q3Sales; q4 = q4Sales;  
    }  
    public double getQ1() { return(q1); }  
    public double getQ2() { return(q2); }  
    public double getQ3() { return(q3); }  
    public double getQ4() { return(q4); }  
    public double getTotal() {  
        return(q1 + q2 + q3 + q4); }  
}
```

Primer



The screenshot shows a Netscape browser window with the title "Conditional Evaluation - Netscape". The address bar displays the URL "http://localhost/servlet/coreservlets.Cconditionals". The main content area features a large orange button labeled "Conditional Evaluation". Below this button is a table with the following data:

	Apples	Oranges
First Quarter	150.25	-220.25
Second Quarter	-75.25	-49.57
Third Quarter	22.25	138.25
Fourth Quarter	-33.57	12.25
Total	63.68	-119.32

The status bar at the bottom indicates "Document: Done (0 06 secs)".

Primer

```
...
<TABLE BORDER=1 ALIGN="CENTER">
<TR><TH>
<TH CLASS="COLORED">Apples
<TH CLASS="COLORED">Oranges
<TR><TH CLASS="COLORED">First Quarter
<TD ALIGN="RIGHT">${apples.q1}
<TD ALIGN="RIGHT">${oranges.q1}
<TR><TH CLASS="COLORED">Second Quarter
<TD ALIGN="RIGHT">${apples.q2}
<TD ALIGN="RIGHT">${oranges.q2}
...
<TR><TH CLASS="COLORED">Total
<TD ALIGN="RIGHT"
BGCOLOR="${(apples.total < 0) ? "RED" : "WHITE" }">
${apples.total}
<TD ALIGN="RIGHT"
BGCOLOR="${(oranges.total < 0) ? "RED" : "WHITE" }">
${oranges.total}
</TABLE>...
```

The JSP Standard Tag Library (JSTL)

- **Primeri JSTL dokumentacije i koda**
- **JSTL Expression Language**
- **Tagovi za realizaciju petlji**
 - Petlje sa tačno određenim brojem iteracija
 - Petlje na osnovu strukture podataka
- **Tagovi za uslovno izvršavanje**
 - Jedan izbor
 - Više izbora
- **Tagovi za pristup nazi podataka**
- **Drugi tagovi**

The JSP Standard Tag Library (JSTL)

- **JSTL je baziran na Struts tagovima za kontrolu petlji i logičke operacije**
- **JSTL nije bio deo JSP 1.2 ili 2.0 specifikacije**
 - Zahtevao je posebnu specifikaciju i podešavanje
 - Moguće samo na serverima koji podržavaju servlets 2.3 i JSP 1.2 ili više verzije
- **JSTL expression language je sada deo JSP 2.0**
- **JSTL specifikacija se nalazi na**
- <http://jcp.org/aboutJava/communityprocess/final/jsr052/>

Tagovi za realizaciju petlji

- **Petlje sa eksplicitnim numeričkim vrednostima**

```
<c:forEach var="name" begin="x" end="y" step="z">  
  Blah, blah <c:out value="\${name}" />  
</c:forEach>
```

- **Petlje koje koriste strukture podataka**

- Moguće je kretati se po nizovima, stringovima, kolekcijama, mapama

```
<c:forEach var="name" items="niz-kolekcija"> Blah, blah  
  <c:out value="\${name}" />  
</c:forEach>
```

- **Petlje koje koriste stringove sa delimiterima**

- forTokens

Tagovi za realizaciju petlji

- **JSP bez JSTL**

```
<UL>
```

```
<%
```

```
for(int i=0; i<messages.length; i++) { String message = messages[i];
```

```
%>
```

```
<LI><%= message %>
```

```
<% } %>
```

```
</UL>
```

- **JSP sa JSTL**

```
<UL>
```

```
<c:forEach var="message" items="${messages}">
```

```
<LI><c:out value="${message}"/>
```

```
</c:forEach>
```

```
</UL>
```

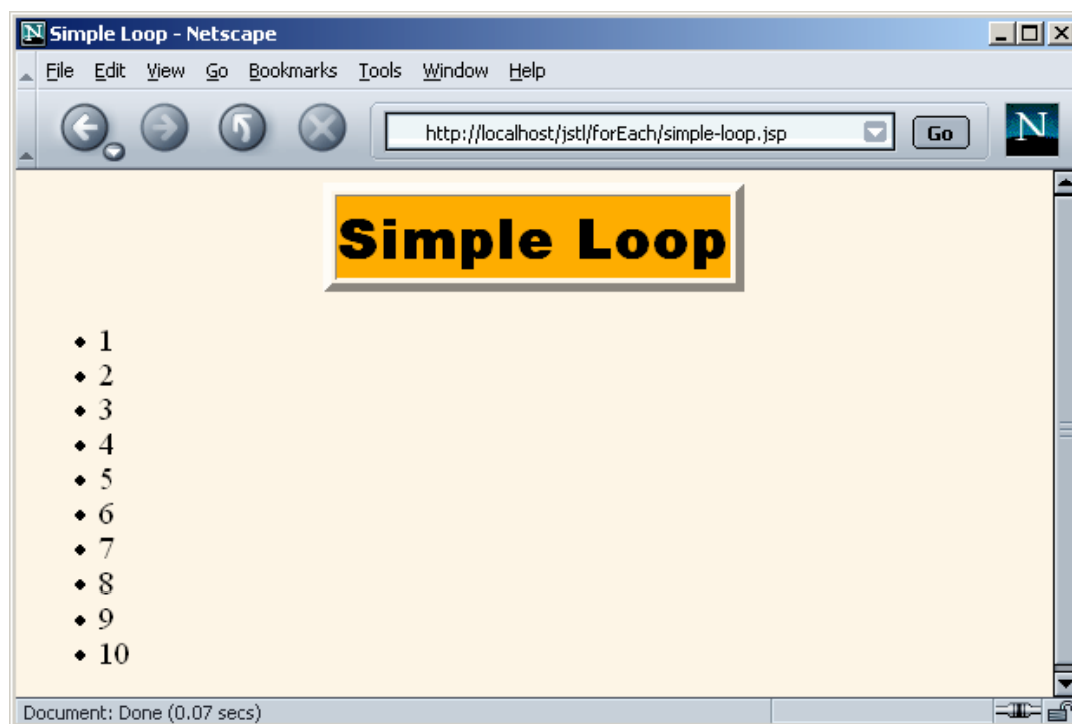
Tagovi za realizaciju petlji

<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core"%>

<c:forEach var="i" begin="1" end="10">

<c:out value="\${i}"/>

</c:forEach>



Tagovi za realizaciju petlji

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core"
    %>
```

```
<UL>
```

```
<c:forEach var="seconds" begin="0"
```

```
end="${pageContext.session.maxInactiveInterval}"
```

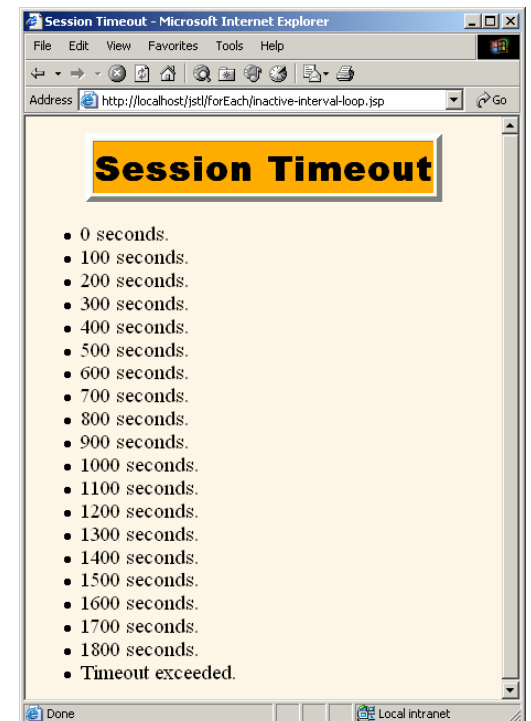
```
step="100">
```

```
<LI><c:out value="${seconds}"/> seconds.
```

```
</c:forEach>
```

```
<LI>Timeout exceeded.
```

```
</UL>
```



Tagovi za realizaciju petlji

```
<% String[] words = { "foo", "bar", "baz"};  
pageContext.setAttribute("words", words); %>
```

```
<%@ taglib prefix="c"  
uri="http://java.sun.com/jstl/core" %>
```

```
<H2>Key Words:</H2>
```

```
<UL>
```

```
<c:forEach var="word"
```

```
items="${words}">
```

```
<LI><c:out value="${word}"/>
```

```
</c:forEach>
```

```
</UL>
```

```
<H2>Values of the test Parameter:</H2>
```

```
<UL>
```

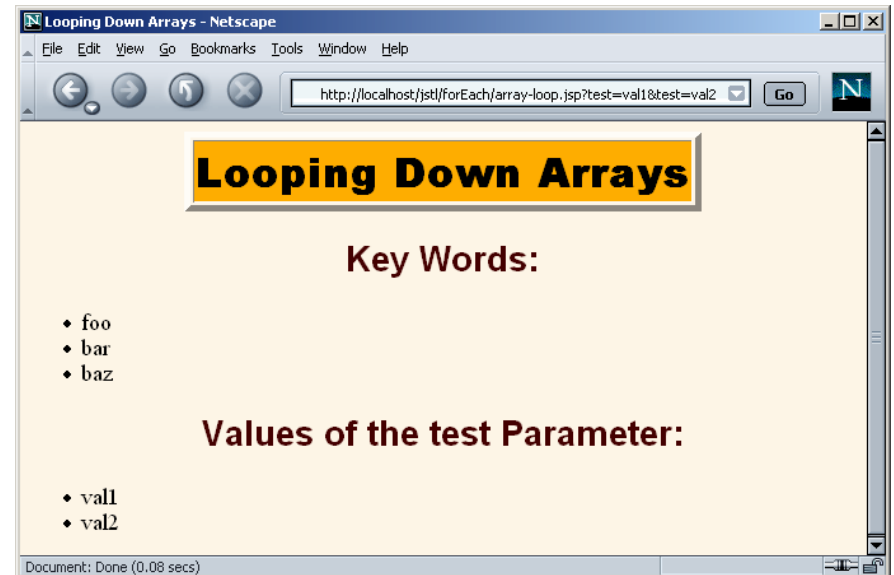
```
<c:forEach var="val"
```

```
items="${paramValues.test}">
```

```
<LI><c:out value="${val}"/>
```

```
</c:forEach>
```

```
</UL>
```



Tagovi za realizaciju petlji

```
<%@ taglib prefix="c"
uri="http://java.sun.com/jstl/core" %>
<UL>
<c:forEach var="country"
items="Australia,Canada,Japan,Philippines,USA">
<LI><c:out value="\${country}"/>
</c:forEach>
</UL>
```

Tagovi za realizaciju petlji

```
<%@ taglib prefix="c"
uri="http://java.sun.com/jstl/core" %>
<UL>
<c:forTokens var="color"
items="(red (orange) yellow)(green)((blue) violet)"
delims="(">
<LI><c:out value="${color}"/>
</c:forTokens>
</UL>
```



Tagovi za realizaciju uslova

- **Jedan izbor: if**

<c:if test="\$ {someTest}">

Sadržaj

</c:if>

- **Više izbora: choose**

<c:choose>

<c:when test="test1"> *Sadržaj1*</c:when>

<c:when test="test2"> *Sadržaj2*</c:when>

...

<c:when test="testN"> *SadržajN*</c:when>

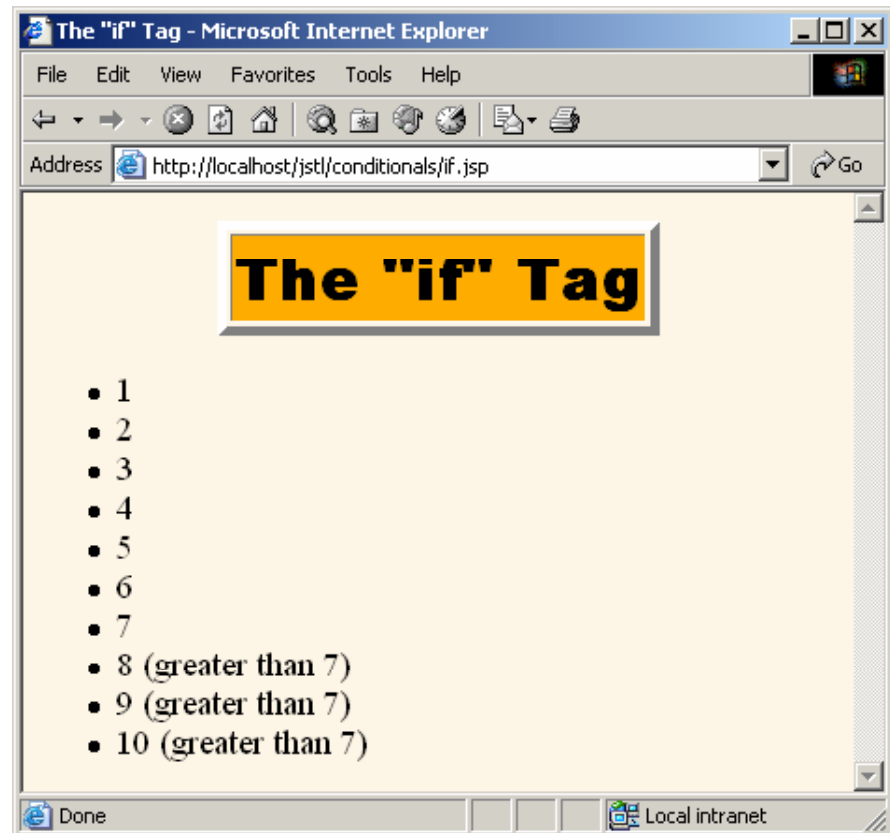
<c:otherwise> *Default Sadržaj* </c:otherwise>

</c:choose>

- **Napomena: realizacija poslovne logike!**

Tagovi za realizaciju uslova

```
<%@ taglib prefix="c"
uri="http://java.sun.com/jstl/core" %>
<UL>
<c:forEach var="i" begin="1" end="10">
<LI><c:out value="\${i}"/>
<c:if test="\${i > 7}">
(greater than 7)
</c:if>
</c:forEach>
</UL>
```



Tagovi za realizaciju uslova

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
```

```
<UL>
```

```
<c:forEach var="i" begin="1" end="10">
```

```
<LI><c:out value="{i}"/>
```

```
<c:choose>
```

```
<c:when test="{i < 4}">
```

```
(small)
```

```
</c:when>
```

```
<c:when test="{i < 8}">
```

```
(medium)
```

```
</c:when>
```

```
<c:otherwise>
```

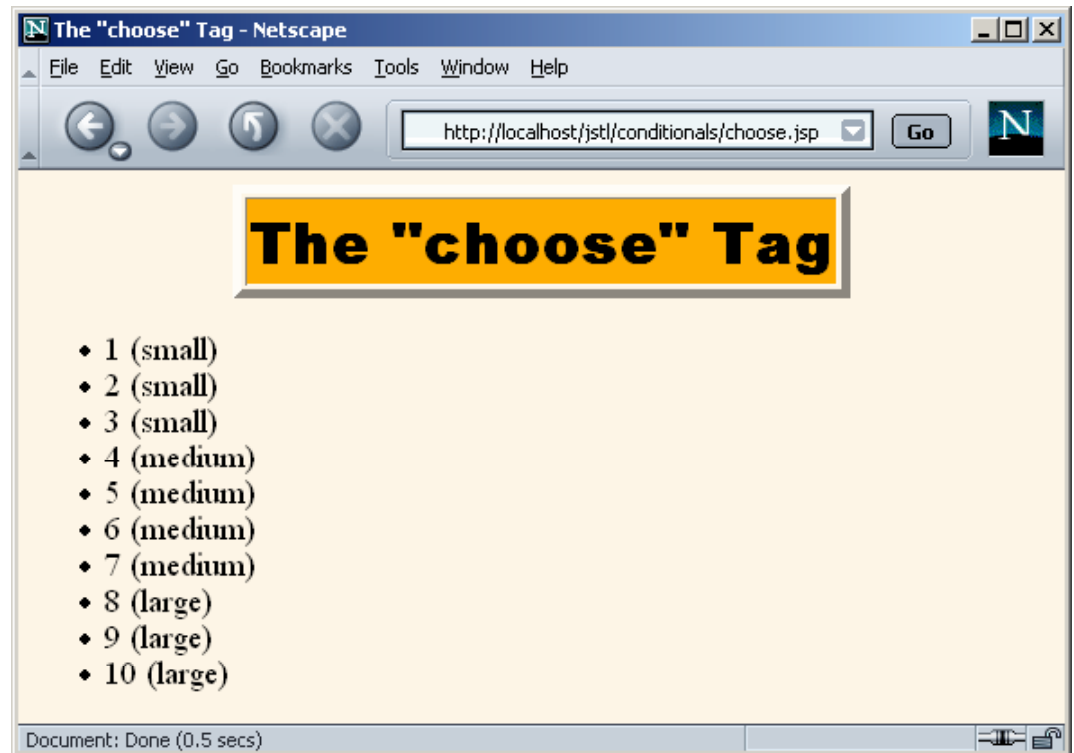
```
(large)
```

```
</c:otherwise>
```

```
</c:choose>
```

```
</c:forEach>
```

```
</UL>
```



Tagovi za pristup bazama podataka

- **<sql:setDataSource>**
 - Specificira se izvor podataka (može se podesiti i pomoću konfiguracionih podešavanja)
- **<sql:query>**
 - Upiti nad bazom podataka i smeštanje ResultSet u okviru promenljive
 - Upozorenje: ovo je suprotno ideji da se poslovna logika odvoji od prezentacione. Bolje je pristupati bazama u okviru servleta i proslediti rezultate u JSP.
- **<sql:update>**
- **<sql:param>**
- **<sql:dateParam>**
- **<sql:transaction>**
 - Izvršava odgovarajuće <sql:query> i <sql:update> akcije kao pojedinačnu transakciju

Tagovi za pristup bazama podataka

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core"
    %>
```

```
<%@ taglib prefix="sql" uri="http://java.sun.com/jstl/sql"
    %>
```

```
<sql:setDataSource driver="sun.jdbc.odbc.JdbcOdbcDriver"
    url="jdbc:odbc:Northwind" user="" password=""/>
```

```
<sql:query var="employees"> SELECT * FROM employees
</sql:query>
```

```
<UL>
```

```
<c:forEach var="row" items="${employees.rows}">
```

```
<LI><c:out value="${row.firstname}"/>
```

```
<c:out value="${row.lastname}"/>
```

```
</c:forEach>
```

```
</UL>
```

Preostali tagovi

- **<c:import>**

- Čita se sadržaj sa odgovarajućeg URLa
 - Ubacuje se u stranicu
 - Smešta u promenljivu
 - ili se omogućava pristup preko čitača
- Za razliku od <jsp:include>, ne važi samo za sopstveni sistem

- **<c:redirect>**

- Redirekcija ka specificiranom URLu

- **<c:param>**

- Dodaju se parametri zahteva u URL

Preostali tagovi

- **<fmt:formatNumber>**
 - Formatiraju se numeričke vrednosti kao broj, iznos u valuti, ili procenat, u lokalnom formatu
- **<fmt:parseNumber>**
 - Čitaju se string prezentacije broja, iznosa u valuti, ili procenta
- **<fmt:formatDate>**
- **<fmt:parseDate>**
- **<fmt:timeZone>**
- **<fmt:setTimeZone>**

Preostali tagovi – rad sa XMLom

- **Osnovni**

- <x:parse>

- **XPath**

- <x:if>

- <x:choose>

- <x:when>

- <x:otherwise>

- <x:forEach>

- **XSLT**

- <x:transform>

- <x:param>