

НАСЛОВИНИЦА

Сервлет технологија
JSP идеја
JSP реализација
Предности JSP технологије
Могућности JSP технологије
JSP – основна синтакса
Врсте динамичких елемената
JSP – изрази
JSP – скриптлети
JSP – декларације
JSP – директиве
Поређење укључивања
Укључивања страница
Укључивање страница јсп тагом
Укључивање и прослеђивање
JSP page директива
Предефинисане промјене и
Response
out
session
application
page
JSP тагови
jsp:plugin таг
Java beans
jsp:setProperty
Java beans – са таговима
Опсег видљивости
Рад са сесијама

prednosti:

- prenosivost na različite računarske platforme i Web servere
- efikasno izvršavanje
- jednostavno čitanje podataka sa forme
- jednostavno čitanje header-a HTTP zahtjeva
- jednostavno postavljanje HTTP statusnog koda i header-a odgovora
- jednostavno korišćenje *cookie-ija* i rad sa sesijama
- jednostavno dijeljenje podataka između servleta
- jednostavno pamćenje podataka između različitih zahtjeva

nedostaci:

- dizajn stranica (HTML) i programska obrada (Java) su pomiješani u istim datotekama
- teško je razdvojiti funkcije dizajnera i programera – oba modifikuju iste datoteke – dizajneri nisu programeri
- svaka promjena u izgledu stranice zahtijeva kompajliranje servleta

JSP ideja

▶ HTML + dinamički elementi

```
<html>
...
<h4>Dobrodošli, <%= username %></h4>
Danas je <%= new java.util.Date() %>.
...
</html>
```

HTML tagovi

dinamički elementi

JSP realizacija

- ▶ JSP stranice se konvertuju u servlete koji generišu upravo onakav izlaz kakav je specificiran u JSP fajlu
- ▶ JSP

```
<html>
...
<h4>Dobrodošli, <%= username %></h4>
Danas je <%= new java.util.Date() %>.
...
</html>
```

org.apache.jasper.runtime

Class HttpJspBase

```
java.lang.Object
├─ javax.servlet.GenericServlet
│   └─ javax.servlet.http.HttpServlet
│       └─ org.apache.jasper.runtime.HttpJspBase
```

▶ servlet

```
public class SomeServlet extends ... {
    public void doGet(...) {
        ...
        out.println("<html>");
        ...
    }
}
```

- ▶ dobijeni servlet se kompajlira i poziva
- ▶ rezultat njegovog izvršavanja je tražena JSP stranica
- ▶ sekvenca događaja:
 - smještanje stranice u odgovarajući direktorijum Web servera
 - Web klijent prvi put traži datu stranicu
 - Web server na osnovu nje generiše servlet, prevodi ga i pokreće
 - naredni zahtjev za istom stranicom – pozivanje generisanog servleta
- ▶ generisanje servleta, njegovo kompajliranje i pozivanje je zadatak aplikativnog servera !!!

PREDNOSTI KORISTENJA JSP – TEHNOLOGIJE

- ▶ pomoću JSP-a se ne može uraditi ništa novo što se ne može postići i pomoću servleta, ali JSP tehnologija olakšava:
 - pisanje samog HTML koda
 - čitanje i održavanje HTML koda
- ▶ pomoću JSP tehnologije moguće je:
 - koristiti standardne alate za dizajniranje HTML stranica
 - podijeliti posao između dizajnera (koji koriste HTML) i Java programera
- ▶ JSP ohrabruje
 - odvajanje Java koda koji predstavlja sam sadržaj, od HTML koda pomoću koga se sadržaj prezentuje

dvije varijante:

- napisati npr. N linija Java koda direktno u JSP stranice
- napisati ovih N linija u odvojenu Java klasu i 1 liniju u JSP stranicu koja koristi objekat date klase ili samu klasu

druga opcija je mnogo bolja:

- razvoj – odvojena klasa se piše u Java okruženju (editor ili IDE), ne u HTML okruženju
- *debug*-ovanje – ako postoje sintaksne greške, primijetiće se odmah tokom procesa kompajliranja
- testiranje – jednostavnije testiranje Java koda
- višestruko korišćenje – ista klasa se može koristiti za više stranica

JSP OSNOVNA SINTAKSA

► HTML Tekst

```
<H1>Blah</H1>
```

- šalje se dalje klijentu – prevodi se u servlet kod koji je sličan sljedećem:

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

► HTML Komentari

```
<!-- Komentar -->
```

- isto kao u HTML-u: šalju se klijentu

► JSP Komentari

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

- ne šalju se klijentu

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

```
<%= java_izraz %>
```

```
<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()

```
<% java_kod %>
```

```
<html>
```

```
...
```

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

```
Dobar dan!
```

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

```
Dobro veče!
```

```
<% } %>
```

```
...
```

```
</html>
```

```

<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 СКРИПТЛЈЕТИ

- ▶ skriptlet sam po sebi ne generiše HTML
- ▶ ako je potrebno da generiše HTML, može se koristiti predefinisana promjenljiva "out"

Trenutno vrijeme:

```

<%
    out.println( String.valueOf( date ));
%>

```


<%! java_deklaracija %>

- ▶ definisanje metoda ili atributa servlet klase – izvan metode za obradu zahtjeva

<%! int hitCount = 0; %>

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

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

- ▶ korištenje ovako deklariranih varijabli, u opštem slučaju, nije dobra praksa
- ▶ "JSP stranica" će se obično izvršavati kao jedna instanca u višestrukim nitima
- ▶ mogućnost interferencije niti – sve niti će raditi nad istom (jednom) promjenljivom
- ▶ ako se moraju koristiti promjenljive – trebala bi se izvršiti sinhronizacija – ovo može bitno uticati na performanse
- ▶ u opštem slučaju, svi neophodni podaci bi trebali da se nađu u *session* ili *request* objektima
- ▶ promjenljive koje se deklariraju unutar skriptleta (npr. `<% int i = 45; %>`) ne predstavljaju problem jer se deklariraju kao lokalne i nisu dijeljene

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

- ▶ omogućavaju kontrolu strukture generisanog servleta
- ▶ tri osnovna tipa direktiva:
 - page direktive
 - definiše stranica-zavisne attribute
 - Language, Extends, Import, contentType, info, session, isThreadSafe, autoflush, buffer, isErrorPage, pageEncoding, errorPage, isELIgnored
 - include direktive
 - uključuje zadatu stranicu u postojeću
 - taglib directive
 - deklarise tag biblioteku koja sadrži korisnički definisane akcije koje se koriste na stranici

page direktive

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

```
<%@ page import="java.util.Vector" %>
```

include direktive

```
<jsp:include page="asd.html"/>
```

- uključuje stranicu u momentu zahtijevanja strane

```
<%@ include file="asd.jsp" %>
```

- uključuje stranicu u momentu kada se stranica prevodi u servlet

<jsp:include> i <%@ include>

Sintaksa	<jsp:include page="...">	<%@ include file="...">
Kada se stranica uključuje	U momentu zahtjevanja strane	U momentu prevođenja u servlet
Šta se uključuje	Izlaz stranice	Sadržaj datoteke
Broj rezultujućih servleta	2	1

<jsp:include>

▶ sintaksa

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

▶ upotreba

- više puta koristiti iste JSP, HTML ili obične tekst dokumente
- dozvoliti promjene uključenog sadržaja bez promjene osnovnih JSP strana

▶ napomene

- JSP sadržaj ne može da mijenja osnovne stranice – koristi se samo izlaz uključenih JSP stranica
- ne treba zaboraviti znak / 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

<%@ include >

- ▶ sintaksa
 - <%@ include file="Relative URL" %>
- ▶ upotreba
 - upotreba više puta istog JSP sadržaja
- ▶ napomene
 - serveri ne provjeravaju da li postoje promjene uključenih fajlova
 - zato je potrebno promijeniti i osnovne JSP fajlove svaki put kada su i uključeni fajlovi promijenjeni

JSP page direktiva

- ▶ pruža informacije na visokom nivou o samom servletu koji se izvršava nakon JSP stranice
- ▶ može kontrolisati:
 - koje se klase importuju
 - koju klasu servlet nasljeđuje
 - 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 naredbe 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/package

- ▶ Preporuka je da se uvijek koriste paketi za klase koje se koriste u okviru JSP stranice!

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

- vrijednost atributa se ne može izračunati u vrijeme zahtjeva

<%@ include >

- ▶ sintaksa

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

- ▶ upotreba

- upotreba više puta istog JSP sadržaja

- ▶ napomene

- serveri ne provjeravaju da li postoje promjene uključenih fajlova
 - zato je potrebno promijeniti i osnovne JSP fajlove svaki put kada su i uključeni fajlovi promijenjeni

isELIgnored atribut

sintaksa

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

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

upotreba

- kontrola da li se JSP 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
- ako web.xml specificira verziju servleta 2.4 (JSP 2.0), default je false

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

- vrijednost atributa se ne može izračunati u vrijeme zahtjeva

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 dio sesije

import atribut

Sintaksa

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

upotreba

- generisanje import naredbe 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/package

Preporuka je da se uvijek koriste paketi za klase koje se koriste u okviru JSP stranice!

▶ **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" promjenljive
- 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 promjenljiva exception se kreira i dostupna je u okviru error stranica
- ovakav rad treba ne treba praktikovati
- trebalo bi eksplicitno obraditi što je moguće više izuzetaka
- uvijek treba provjeriti unesene podatke

▶ **isThreadSafe atribut**

▶ sintaksa

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

▶ upotreba

- da se naglasi sistemu da kod nije threadsafe, tako da sistem može da spriječi konkurentne pristupe kodu
- rezultat – servletu se prenosi da implementira SingleThreadModel

▶ napomene

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

▶ ovaj atribut nije potreban – moguća je sinhronizacija

- synchronized(this) {}
- dobijaju se bolje performanse u okruženju sa velikim saobraćajem

▶ **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 naslijedile klase koje daje proizvođač servera, a ne da bi se nasljeđivale sopstvene klase

ime	tip
request	HttpServletRequest
response	HttpServletResponse
out	JspWriter
session	HttpSession
application	ServletContext
page	(this)

response

- ▶ HttpServletResponse povezan sa odgovorom klijentu
- ▶ dozvoljeno je postavljanje HTTP statusnih kodova i zaglavlja odgovora (*response headers*)

out

- ▶ out je baferovana verzija `PrintWriter`-a (pod nazivom `JspWriter`) koji se koristi za slanje odgovora klijentu
- ▶ 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 smještaju u izlazni tok

session

- ▶ `HttpSession` objekt povezan sa sesijom
- ▶ 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 rezultira generisanjem poruke o grešci od strane servera u momentu prevođenja JSP strane u servlet

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`

- ▶ JSP tagovi posjeduju:
 - start tag
 - tijelo
 - end tag
- ▶ start i end tagovi se sastoje iz imena taga koje se nalazi unutar znakova `< i >`, s tim da end tag ima i znak `/` iza `<`
- ▶ imena tagova imaju znak `:`, pri čemu dio prije ovog znaka opisuje tip taga
`<some:tag> body </some:tag>`
- ▶ ako tag nema tijela, start i end tag se mogu spojiti u jedan
`<some:tag/>`

- ▶ dva tipa tagova:
 - predefinisani tagovi
 - tagovi iz eksterne tag biblioteke
- ▶ predefinisani tagovi počinju sa **jsp:** nizom karaktera

jsp:plugin tag

- ▶ 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 Java 2 okruženjem.
 - izvršava se u skoro svakom čitaču
 - koriste se OBJECT i EMBED tagovi
 - ova opcija je pojednostavljena pomoću jsp:plugin taga
- ```
jsp:plugin
<jsp:plugin type="applet" width="475" height="350">
</jsp:plugin>
```
- ▶ sintaksa slična APPLET tagu – dobijaju se OBJECT i EMBED tagovi
  - ▶ Napomena
    - JSP element i imena atributa su case sensitive
    - Sve vrijednosti atributa moraju biti između ' ' ili ""

## Primjer

- ▶ podaci se unose preko HTML formi
- ▶ koristi se ista klasa/metoda kao kod servleta – `request.getParameter()`
- ▶ mogu se automatski smještati u zadati objekat – JavaBeans

```
<html>
```

```
...
```

```
<% if (request.getParameter("username") != null) { %>
Vrijednost parametra: <%= request.getParameter("username")
%>
<% } %>
```

```
...
```

```
</html>
```



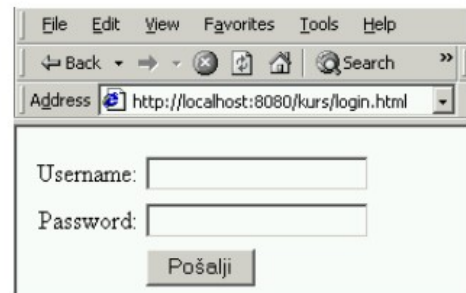
```

public class User {
 public void setUsername(String x) {
 username = x;
 }
 public void setPassword(String x) {
 password = x;
 }

 public String getUsername() {
 return username;
 }
 public String getPassword() {
 return password;
 }

 private String username;
 private String password;
}

```



## JAVA BEANS

### ► Java bean klasa:

- moraju imati podrazumijevani (zero-argument, prazan) konstruktor – ovaj zahtjev se može ispuniti eksplicitnim definisanjem takvog konstruktora ili izostavljanjem svih konstruktora s parametrima
- ne bi trebalo da ima public promjenljive instanci (polja)
- vrijednosti atributa treba da se dobijaju/postavljaju pomoću geter i seter metoda – getXxx i setXxx
- ako klasa ima metod getTitle koji kao rezultat vraća String, kaže se da klasa posjeduje String property nazvanu title
- boolean properties koriste isXxx umjesto getXxx

### ► beanovi se postavljaju u uobičajeni Java direktorijum

.../WEB-INF/classes/package



### ▶ sintaksa

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

### ▶ upotreba

- dozvoljava instanciranje Java klase bez eksplicitnog Java programiranja

### ▶ napomene

- interpretacija:

```
<jsp:useBean id="book1" class="org.unibl.etf.Book" />
```

- može se dobiti i pomoću skriptleta

```
<% org.unibl.etf.Book book1 = new org.unibl.etf.Book();
%>
```

### ▶ jsp:useBean prednosti:

- jednostavnije je dobijati vrijednosti objekata iz zahtijevanih parametara
- jednostavnije je dijeliti objekte između stranica ili servleta

### ▶ sintaksa

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

### ▶ upotreba

- dozvoljava pristup bean property-ijima bez eksplicitnog Java programiranja

### ▶ napomene

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

- je ekvivalentno sa sljedećim JSP izrazom

```
<%= book.getTitle() %>
```

## ▶ sintaksa

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

## ▶ upotreba

- dozvoljava postavljanje bean property-ija bez eksplicitnog Java programiranja

## ▶ napomene

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

- je ekvivalentno sljedećem skriptletu

```
<% book.setTitle("Java"); %>
```



login.jsp

```
<form action="result.jsp">
Username:
 <input type="text" name="username">
Password:
 <input type="password" name="password">
 <input type="submit" value="Prijavi me">
</form>
```

GET /result.jsp?username=marko&password=\*\*\*

result.jsp

```
<jsp:useBean id="user" class="org.unibl.etf.User"/>
<jsp:setProperty name="user" property="username" param="username"/>
<jsp:setProperty name="user" property="password" param="password"/>
<html>
...
</html>
```



```
<jsp:useBean id="user" class="org.unibl.etf.User"/>
<jsp:setProperty name="user" property="username" param="username"/>
<jsp:setProperty name="user" property="password" param="password"/>
<html>
...
<% if (user.login()) { %>
 Uspješno ste se prijavili!
<% } else { %>
 Niste se uspješno prijavili!
<% } %>
...
</html>
```

koristimo user kao da je u  
pitanju JSP deklaracija

- ▶ kod u JavaBeans
- ▶ ograničiti Java kod u jsp stranicama
  - razdvajanje vizuelnog dijela koda od programskog dijela
- ▶ efikasniji Java kod u klasi nego u jsp stranici:
  - sintaksne greške se vide prilikom kompajliranja klase, a ne prilikom ponovnog učitavanja stranice
  - testiranje se lakše izvodi iz komandne linije/integrisanog okruženja, nego iz jsp stranice
  - ponovna upotreba u drugim stranicama/projektima



# Opseg vidljivosti

- ▶ application  
istu instancu beana dijele svi korisnici aplikacije (sajta)
- ▶ session  
svaki korisnik aplikacije (sajta) ima svoju instancu
- ▶ request  
svaki zahtjev za stranicom ima svoju instancu
- ▶ page  
svaka stranica ima svoju instancu
- ▶ specificira se u `<jsp:useBean>` elementu

```
<jsp:useBean id="user" class="somepackage.User"
scope="session"/>
```

## Rad sa sesijama

- ▶ promjenljiva session
- ▶ HttpSession klasa