Programski Sistemi

Java Server Pages (JSP)

Nedostaci servleta

- Da bi se razvili i održavali svi delovi aplikacije, neophodno je detaljno poznavanje Jave
- Bilo kakva izmena aplikacije zahteva ponovno kompajliranje i deploy
- Prilikom projektovanja interfejsa ne mogu se koristiti alati za HTML (npr. Adobeov Dreamweaver), tj. HTML mora biti ručno unošen u servletski kod

Java Server Pages (JSP)

- Omogućuju razdvajanje obrade zahteva i poslovne logike od prezentacije, tako da se posao podeli, što ima više prednosti
  - Svako radi svoj posao: Java programeri bave se kontrolom aplikacije, a Web dizajneri prave interfejs
  - Različiti aspekti aplikacije razvijaju se nezavisno, pa je moguće promeniti pravila poslovne logike bez promene korisničkog interfejsa

Poređenje sa drugim tehnologijama

- U odnosu na ASP ili PHP
  - moćniji jezik za dinamički deo aplikacije
  - izvršavanje na većem broju servera i operativnih sistema
  - bolja podrška alata
- Nedostaci
  - Slaba podrška kod hosting provajdera u Srbiji
  - Složenija tehnologija, zahteva više vremena za savladavanje

JSP

- JSP strana je HTML strana sa JSP elementima koji generišu delove strane različite za svaki request
- Servlet kontejner prevodi u pozadini svaku JSP stranu u servlet kada se ona prvi put pozove; nakon toga kompajlira se samo ako je promenjena
- Servleti i JSP se dopunjavaju: servlete ćemo koristiti kao kontrolere na serverskoj strani i za komunikaciju sa klasama modela, a JSP strane će biti sloj prezentacije

JSP API

- Svi osnovni JSP interfejsi i klase definisani su u paketu javax.servlet.jsp
- Expression Language API nalazi se u paketu javax.servlet.jsp.el
- JSP biblioteke tagova definisane su u paketu javax.servlet.jsp.tagext

Životni ciklus JSP stranice

Životnim ciklusom JSP strane upravlja servlet kontejner

	Page first written	Request #1	Request #2	Server restarted	Request #3	Request #4	Page modified	Request #5	Request #6
JSP page translated into servlet		Yes	No		No	No		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

Organizacija JSP koda

- Postoje dve opcije
  - Upisati mnogo Java koda direktno u JSP stranice
  - Upisati taj kod u odvojenu Java klasu i jedan red koda u JSP stranicu koja ga poziva
- Zašto je druga opcija mnogo bolja?
  - Razvoj: Odvojena klasa se piše u Java okruženju (editor ili IDE), ne u HTML okruženju
  - Debagovanje: ako postoje sintaksne greške, primetiće se odmah tokom procesa kompajliranja
  - Višestruko korišćenje: može se koristiti ista klasa za više stranica

JSP u strukturi Web aplikacije

- JSP fajlove možemo staviti bilo gde u WAR arhivu, ali ako ih stavimo u direktorijum WEB-INF, nećemo moći da im pristupimo direktno iz klijenta
- Obično se stavljaju direktno u korenski direktorijum WAR arhive

Osnovna sintaksa

- HTML kod
  - <H1>Naslov</H1>
  - Šalje se dalje klijentu. Prevodi se u sledeći servlet kod: out.print("<H1>Naslov</H1>");
- HTML Komentari
  - <!-- Komentar -->
  - Isto kao u HTML-u: šalje se dalje klijentu
- JSP komentar
  - <%-- 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 atribut="..."%>
  - <%@ page contentType="text/plain" %>

JSP izrazi

Prečica za naredbu out.print(), gde je out objekat klase JspWriter, slično PrintWriteru u servletima

```
<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

- Skriptleti se uopšte ne prevode već se samo umeću u izvorni kod servleta
- Promenljive uvedene u skriptletu mogu se koristiti u narednim skriptletima na istoj strani (u skladu sa pravilima o oblasti važenja)
- Kod napisan unutar skriptleta završava u metodi _jspService()

JSP deklaracije

- Promenljive definisane u deklaracijama postaju podaci članovi servletske klase u koju se JSP strana kompajlira, a metode postaju funkcije članice

```
<%! 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

- Koriste se za zadavanje specijalnih instrukcija servlet kontejneru dok se JSP strana prevodi u servlet
  - **page**
    - import – za import paketa
    - contentType – podešava ContentType odgovora
  - **include**
    - uključuje zadatu stranicu u postojeću
  - **taglib**
    - deklariše biblioteku tagova (custom tag library) koja se koristi na strani

Direktiva page

- <%@ page *attribute1="value1" attribute2="value2" attribute3=... %>

atribut	**vrednost**	**default**	**primer**
info	string	nema	info="Registration form“
language	Ime skript jezika	java	language="java"
contentType	MIME tip, skup znakova	vidi primer	contentType="text/html; charset=ISO-8859-1"
pageEncoding	skup znakova	"ISO-8859-1"	pageEncoding=“UTF-8"
extends	ime klase	nema	extends="com.taglib.wdjsp.MyJsp Page"
import	imena klasa i(li) paketa	nema	import="java.net.URL" import="java.util.*, java.text.*"
session	Logička vrednost	true	session=“false"
errorPage	Lokalni URL	nema	errorPage="results/failed.jsp"

Direktiva include

- Omogućuje autorima strana da uključuju sadržaj jednog fajla (JSP, HTML) u drugi
- <%@ include file="*localURL*" %>
- Nema ograničenja u broju strana koje se mogu uključivati; dozvoljeno je i ugnježđivanje
- U atributu *file* mogu se koristiti i relativne i apsolutne putanje; obično se zadaje relativna putanja u odnosu na stranu gde se koristi
- Preporučuje se da se strane (fragmenti) koje nisu samostalne, već se uključuju u druge strane, čuvaju sa ekstenzijom .jspf ili .jsf

Direktiva include

- Kada se koristi direktiva include, sadržaj uključene strane se direktno umeće (kopira) na mestu direktive
- Međutim, ako se strane uključene pomoću direktive include izmene, to neće prouzrokovati ponovno kompajliranje strane u koju su uključeni, jer servlet kontejner ne prati zavisnosti između fajlova

Direktiva taglib

- Obaveštava servlet kontejner da se strana oslanja na posebnu biblioteku tagova
- <%@ taglib uri="tagLibraryURI" prefix="tagPrefix" %>
- Konstruisanje tag biblioteka je posebna tema kojom se ovde nećemo baviti

Implicitni JSP objekti

- JSP kontejner nudi niz implicitnih objekata autorima JSP strana; ovi objekti su automatski dostupni
- Devet implicitnih objekata mogu se podeliti u četiri kategorije:
  - objekti povezani sa servletom JSP strane
  - objekti povezani sa ulazom i izlazom strane
  - objekti koji nude informacije o kontekstu u kome se izvršava JSP strana
  - objekti koji su rezultat grešaka

Implicitni JSP objekti

Ime	Klasa	Opis
page	javax.servlet.jsp.HttpJspPage	Instanca servleta strane
config	javax.servlet.ServletConfig	Informacije o konfiguraciji servleta za stranu
request	javax.servlet.ServletRequest	Informacije o trenutnom requestu
response	javax.servlet.http.HttpServletResponse	Informacije u vezi sa response
session	javax.servlet.http.HttpSession	Objekat sesije servleta za stranu
out	javax.servlet.jsp.JspWriter	Klasa izvedena iz java.io.Writer koja se koristi za ispis teksta koji će biti uključen na Web stranu
pageContext	javax.servlet.jsp.PageContext	Kontekst JSP strane (uglavnom se koristi kada se implementiraju namenski tagovi)
application	javax.servlet.ServletContext	Kontekst za sve Web komponente u istoj aplikaciji; veza do RequestDispatchera
exception	java.lang.Throwable	Greška ili izuzetak koji nije uhvaćen; dostupan samo na error page

Upravljanje greškama na JSP stranama

- Da bismo upravljali greškama koje baca JSP strana, treba da napravimo error page i postavimo njen atribut **isErrorPage** na true u direktivi page.
- Pošto obično koristimo istu error page za sve JSP strane, umesto da konfigurišemo sve JSP strane pojedinačno, bolje je da definišemo error page u fajlu web.xml pomoću elementa **error-page**.

Upravljanje greškama na JSP stranama

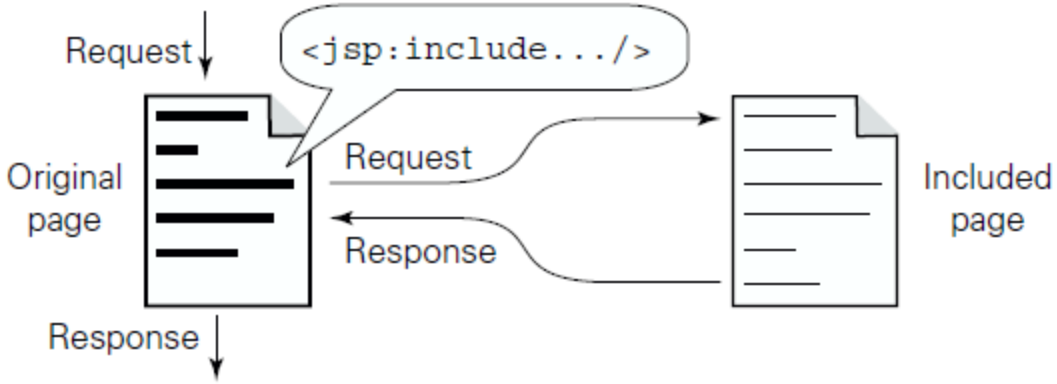
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <display-name>JSPExceptionHandling</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
   </welcome-file-list>

   <error-page>
   <error-code>404</error-code>
   <location>/error.jsp</location>
   </error-page>

   <error-page>
   <exception-type>java.lang.Throwable</exception-type>
   <location>/error.jsp</location>
   </error-page>

</web-app>
```

JSP akcije

- Omogućuju prenos kontrole između strana
- Podržavaju specifikaciju Java apleta na način nezavisan od čitača Weba
- Omogućuju JSP stranama da komuniciraju sa Java Beans komponentama koje se nalaze na serveru

Akcija <jsp:include>

- Omogućuje ugrađivanje sadržaja koji je generisan u drugom dokumentu u izlaz tekuće strane
- <jsp:include page="*localURL*" *flush="flushFlag"* />
- Atribut page može biti statički dokument, servlet ili neka druga JSP strana

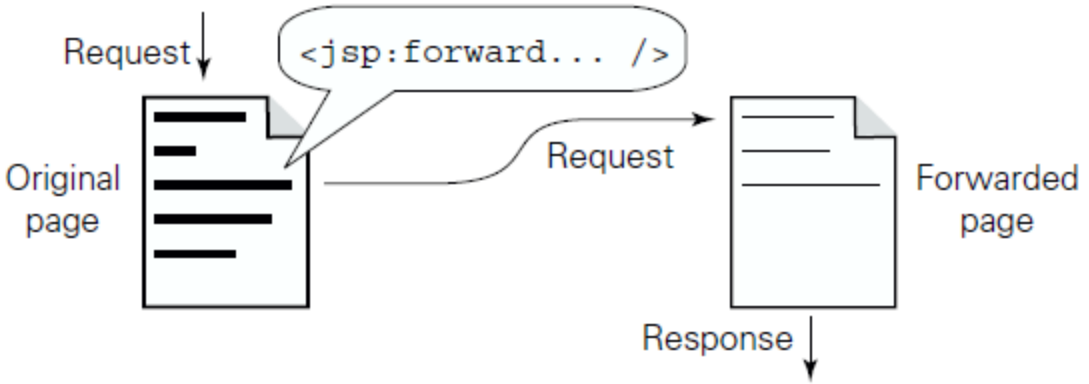


Akcija <jsp:include>

- Request se prosleđuje strani koja je uključena; JSP kontejner ugrađuje izlaz te strane u originalnu stranu, i zatim nastavlja sa obradom originalne strane
- Sve se to dešava tokom obrade zahteva
- Za razliku od direktive include koja deluje u trenutku kada se strana prevodi u servlet, akcija <jsp:include> umeće izlaz uključene strane, a ne njen sadržaj

Akcija <jsp:forward>

- Trajno prenosi kontrolu iz JSP strane na drugu lokaciju unutar iste Web aplikacije
- <jsp:forward page="*localURL*" />
- Sav sadržaj koji je generisan na tekućoj strani se odbacuje, a obrada zahteva počinje od početka na novoj lokaciji zadatoj u atributu page



Akcija <jsp:forward>

- Pošto je objekat request zajednički za originalnu stranu i onu na koju se prosleđuje, svi request parametri biće dostupni i na strani na koju se prosleđuje
- Mogu se zadati i dodatni parametri za request, sintaksom

```
<jsp:forward page="localURL">
  <jsp:param name="parameterName1"
             value="parameterValue1"/>
      …
  <jsp:param name="parameterNameN"
             value="parameterValueN"/>
</jsp:forward>
```

Java Beans

- Java Bean je objekat koji izlaže podatke preko svojstava (properties)
- JSP kontejner može da pregleda bean i da otkrije njegova svojstva preko mehanizma koji se zove introspekcija
- Da bi bila bean, klasa mora da ispuni nekoliko uslova:
  - Mora imati default konstruktor (bez argumenata)
  - Ne sme da ima public properties (podatke članove), već se njima pristupa preko getera i setera
  - Više informacija na http://docs.oracle.com/javase/tutorial/javabeans/

Bean tagovi

- Tag <**jsp:useBean**> pronalazi i instancira Java Bean, tj. kreira objekat bean ako već ne postoji
- Bean se kasnije može pozivati na strani pomoću vrednosti dodeljene atributu
- Akcija <**jsp:setProperty**> postavlja vrednost za bean property pomoću njegovog metoda set
- Akcija <**jsp:getProperty**> čita vrednost za bean property pomoću njegovog metoda get
- Osim getera i setera, bean može da sadrži bilo koji drugi metod; bitno je da on poštuje sintaksu getera da bi mu se pristupilo iz strane kao da je property

Expression Language (EL)

- JSP od verzije 2.0 nudi sintaksu za ugrađivanje rezultata jednostavnih izraza u stranu: Expression Language (EL)
- Koristi sledeću sintaksu: ${izraz}
- Primer: Pre verzije JSP 2.0, da bi se očitala vrednost iz mape na JSP strani, pisalo se:
  <%=((Map) pageContext.findAttribute("mojaMapa")).get(“nekiKljuc") %>
- EL ekvivalent prethodnog primera je
  ${mojaMapa.nekiKljuc}

Implicitni EL objekti

Ovi implicitni objekti su različiti od implicitnih JSP objekata i mogu se koristiti samo u EL!

Ime	Opis
pageScope	Mapa atributa sadržanih u page scope
requestScope	Mapa atributa sadržanih u request scope
sessionScope	Mapa atributa u session scope
applicationScope	Mapa atributa u application scope
param	Dobijanje vrednosti jednog request parametra
paramValues	Dobijanje vrednosti request parametara u obliku niza
header	Dobijanje parametara iz zaglavlja u obliku niza
headerValues	Dobijanje parametara iz zaglavlja u obliku niza
cookie	Dobijanje vrednosti kolačića
initParam	Dobijanje context init parametara
pageContext	Isto kao JSP implicitni objekat pageContext

EL sintaksa

- Za pristup bilo kom objektu koristi se sintaksa ${objekat}
- Operator tačka se koristi za pristup vrednostima atributa: ${prviObjekat.drugiObjekat}
  - ${requestScope.zaposleni.adresa}
- Osim poslednjeg dela EL, svi objekti moraju da budu ili Map ili Java Bean; u prethodnom primeru requestScope je Map, a zaposleni je Java Bean ili Map
- Ako scope nije naveden, JSP EL će tražiti navedeni atribut u page, request, session i application scope

EL sintaksa

- Operator [] je snažniji od tačke i može se koristiti za pristup listama i nizovima
- ${nizIliLista[0]}
- ${mojaMapa[izraz]} – ako izraz u [] nije String, tumači se kao EL
- Podržani su aritmetički operatori +, -, *, / ili div, % ili mod za jednostavna izračunavanja
- Logički operatori su && (and), || (or) i ! (not)
- Relacioni operatori su ==, !=, <, >, <= i >=

EL: važno je znati

- EL izrazi u JSP strani se mogu isključiti postavljanjem atributa isELIgnored na true u direktivi page
- EL se može koristiti za čitanje atributa, parametara, kolačića i sl., ali ne i za njihovo postavljanje
- Osim objekta pageContext, implicitni EL objekti nisu isti kao JSP implicitni objekti
- JSP EL je NULL friendly: ako traženi atribut nije nađen ili izraz vraća null, ne baca se izuzetak. U aritmetičkim operacijama, EL tretira null kao 0, a u logičkim kao false.

JSTL

- EL i akcije su ograničeni; pomoću njih se ne može npr. iterirati kroz kolekciju
- **JSP Standard Tag Library** (**JSTL**) je standardna biblioteka tagova za kontrolu ponašanja JSP strane, iteraciju i kontrolu, XML i SQL
- **JSTL** je deo JavaEE API-ja i uključen je u većinu servlet kontejnera, ali da bismo koristili JSTL u stranama, moramo da preuzmemo jar specifičan za servlet kontejner koji koristimo
- Jar se smešta u direktorijum projekta **WEB-INF/lib**

JSTL

Tag	Opis
<c:out>	Ispis na strani; koristi se sa EL
<c:import>	Isto kao <jsp:include> ili direktiva include
<c:redirect>	Preusmeravanje requesta na drugi resurs
<c:set>	Postavljanje vrednosti promenljive u određeni scope
<c:remove>	Uklanjanje promenljive iz određenog scopea
<c:catch>	Hvatanje izuzetka u objekat
<c:if>	Jednostavna uslovna logika, koristi se sa EL
<c:choose>, <c:when>, <c:otherwise>	Međusobno isključiva logika (if then else)
<c:forEach>	Iteriranje kroz kolekciju
<c:param>	Koristi se sa <c:import> za prosleđivanje parametara

JSTL: primer

```
<table>
<tbody>
<tr><th>ID</th><th>Name</th><th>Role</th></tr>
<c:forEach items="${requestScope.empList}" var="emp">
<tr><td><c:out value="${emp.id}"></c:out></td>
<td><c:out value="${emp.name}"></c:out></td>
<td><c:out value="${emp.role}"></c:out></td></tr>
</c:forEach>
</tbody>
</table>
<br><br>
<%-- simple c:if and c:out example with HTML escaping --%>
<c:if test="${requestScope.htmlTagData ne null }">
<c:out value="${requestScope.htmlTagData}" escapeXml="true"></c:out>
</c:if>
<br><br>
<%-- c:set example to set variable value --%>
<c:set var="id" value="5" scope="request"></c:set>
<c:out value="${requestScope.id }" ></c:out>
<br><br>
<%-- c:catch example --%>
<c:catch var ="exception">
   <% int x = 5/0;%>
</c:catch>

<c:if test = "${exception ne null}">
   <p>Exception is : ${exception} <br />
   Exception Message: ${exception.message}</p>
</c:if>
```

JSP: preporuke

- Izbegavati skriptlete u JSP stranama i umesto njih koristiti EL, akcije i JSTL
- U JSP stranama izbegavati poslovnu logiku; one treba da služe samo za generisanje odgovora klijentu
- Zbog boljih performansi isključiti sesiju u JSP stranama gde nije potrebna
- Direktive page i taglib koristiti na početku strane zbog čitljivosti

JSP: preporuke

- Koristiti direktivu jsp include ili akciju include u skladu sa potrebama: direktiva include je dobra za statički sadržaj, a akcija include za dinamički i sadržaj koji se umeće za vreme izvršavanja
- Upravljati greškama pomoću error page
- Ako strane sadrže CSS i JavaScript, staviti ih u posebne fajlove i uključivati u JSP strane
- Ako JSTL nije dovoljan, proveriti dizajn aplikacije: bolje je premestiti logiku u servlet koji će zatim postaviti atribute za prikaz na JSP strani