



Funded by the
Erasmus+ Programme
of the European Union



This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



KI204 - JAVA 7: JAVA ENTERPRISE EDITION

Java Server Pages (JSP)

Lekcija 02

PRIRUČNIK ZA STUDENTE

KI204 - JAVA 7: JAVA ENTERPRISE EDITION

Lekcija 02

JAVA SERVER PAGES (JSP)

- ✓ Java Server Pages (JSP)
- ✓ Poglavlje 1: JSP arhitektura
- ✓ Poglavlje 2: JSP sintaksa
- ✓ Poglavlje 3: Procesiranje forme
- ✓ Poglavlje 4: Domaći zadatak 2
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

▼ Uvod

UVOD

JSP je tehnologija razvoja veb stranica koja podržava umetanje Java koda u HTML.

Java Server Pages (JSP) predstavlja tehnologiju za razvoj veb stranica koja podržava primenu dinamičkog sadržaja na način da omogućava programerima umetanje Java koda u HTML dokument. Ova funkcionalnost je omogućena primenom JSP tagova koji počinju sa <% i završavaju se simbolima %>.

JSP komponenta je tip Java servleta koji je dizajniran da izvede neku ulogu korisničkog interfejsa Java veb aplikacije. Veb programeri pišu JSP datoteke kao tekstualne datoteke koje kombinuju HTML ili XHTML, XML elemente i ugrađene JSP akcije i komande. Primenom JSP stranica moguće pokupiti korisničke unose pomoću formi veb stranice, prezentovati zapise izvučene iz baza podataka ili drugih izvora i dinamički kreirati sadržaj veb stranica.

JSP tagovi koriste se u različite svrhe - vraćanje podataka iz baza podataka ili registrovanje korisničkih preferencija, pristup komponentama tipa JavaBeans, prosleđivanje kontrole između stranica, kao i deljenje informacija između stranica, zahteva i tako dalje. Sledeće teze pokazuju dobre strane korišćenja JSP stranica:

- Značajno poboljšanje u performansama, budući da JSP dozvoljavaju ugrađivanje dinamičkih elemenata u HTML stranice umesto primene posebnih fajlova;
- JSP su uvek kompajlirane pre nego što su procesirane;
- JSP imaju pristup svim Java EE API - jima, uključujući JDBC, JNDI, EJB, JAXP i tako dalje;
- JSP mogu da se koriste u kombinaciji sa servletima koji rukuju poslovnim logikom;
- JSP su integralni deo Java EE platforme za razvoj složenih Java aplikacija. To praktično znači da se JSP može naći od najjednostavnijih do veoma kompleksnih Java EE veb aplikacija.

Slede prednosti JSP - a u odnosu na neke druge tehnologije:

- JSP i Active Server Pages (ASP) - Prednosti JSP su dvostruke. Dinamički kod je pisan JAVA jezikom, a ne Visual Basic ili drugim specifičnim jezikom. JSP je prenosiv na različite operativne sisteme i veb servere;
- JSP i "čist" servlet - JSP se lakše pišu;
- JSP i Server-Side Includes (SSI) - SSI se koristi za jednostavne pozive, a ne realne programe koji koriste baze podataka;
- JSP i JavaScript - JavaScript je okrenuta generisanju HTML stranica na klijent strani i ima poteškoće u obraćanju veb serveru za izvođenje kompleksnih zadataka, kao što su: pristup bazi podataka, procesiranje slika i tako dalje;
- JSP i statičke HTML stranice - "regularne" HTML stranice ne mogu da sadrže dinamički sadržaj.

▼ Poglavlje 1

JSP arhitektura

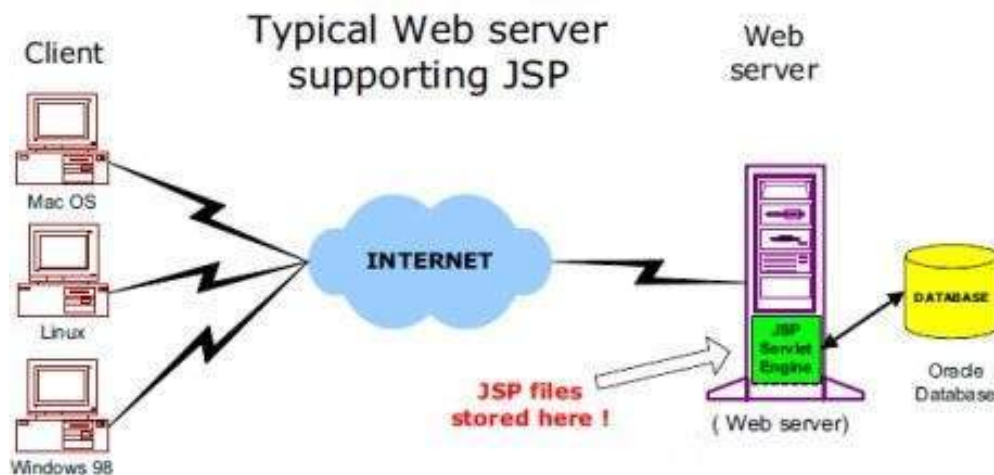
JSP MEHANIZAM

JSP kontejner je odgovoran za presretanje zahteva za JSP stranice.

Veb server zahteva JSP mehanizam, kao što je kontejner za obradu JSP stranica. JSP kontejner je odgovoran za presretanje zahteva za JSP stranice. U ovoj lekciji će biti prikazana primena GlassFish servera koji poseduje ugrađeni JSP kontejner za podršku razvoju JSP stranica.

JSP kontejner radi zajedno sa veb serverom za obezbeđivanje okruženja za izvršavanje JSP stranica i brojnih servisa koje JSP stranice koriste. JSP kontejner razume specijalne elemente koji su integrisani u JSP stranice.

Sledećom slikom je prikazan položaj JSP kontejnera i JSP datoteka u Java EE veb aplikaciji.



Slika 1.1.1 Položaj JSP kontejnera i JSP datoteka u Java EE veb aplikaciji.

JSP PROCESIRANJE

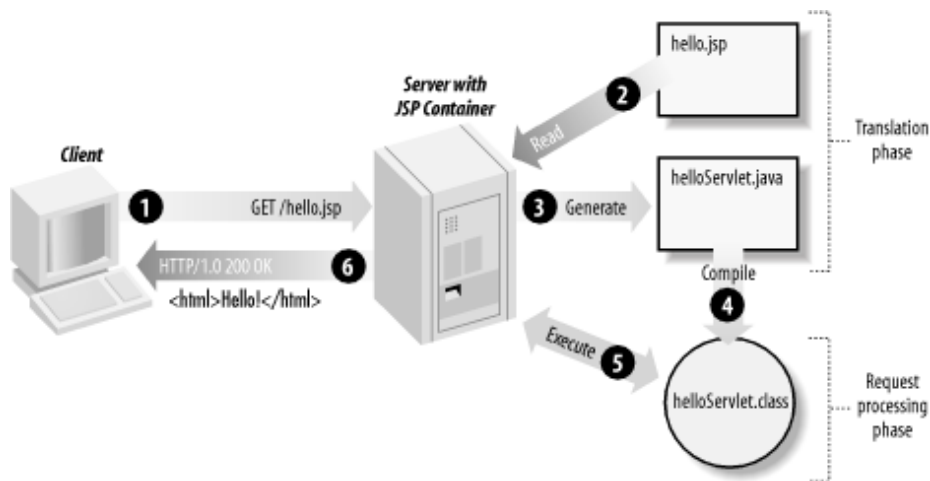
U narednom izlaganju neophodno je pokazati kako funkcioniše obrada JSP stranica.

U narednom izlaganju neophodno je pokazati kako funkcioniše obrada JSP stranica. Sledećim koracima je pokazano kako veb server kreira veb strancu koristeći JSP:

- Veb pregledač šalje HTTP zahtev serveru;

- Veb server prepoznaje da je HTTP zahtev za veb stranicu i prosleđuje ga JSP kontejneru. Ovo je urađeno pomoću URL ili veb stranice sa ekstenzijom .jsp (umesto .html);
- JSP mehanizam učitava JSP stranicu sa diska i konvertuje je u servlet sadržaj. Konverzija je veoma jednostavna i podrazumeva konverziju teksta u println() izraze i JSP elemente u Java kod koji implementira odgovarajuće dinamičko ponašanje veb stranice;
- JSP kontejner prevodi servlet u izvršivuklasu i prosleđuje originalni zahtev servlet kontejneru;
- Veb server, pomoću svoje komponente servlet mehanizam (kontejner), učitava servlet klasu i izvršava je. Tokom izvršavanja, servlet kreira izlaz u HTML formi, kojeg servlet kontejner prosleđuje veb serveru u okviru HTTP odgovora;
- Veb server prosleđuje HTTP odgovor ka veb pregledaču;
- Veb pregledač rukuje dinamički generisanim HTML sadržajem iz HTTP odgovora na isti način kao što radi sa statičkim stranicama.

Navedeni koraci su prikazani dijagramom koji je priložen sledećom slikom.



Slika 1.1.2 JSP procesiranje

JSP kontejner proverava da li JSP datoteka već postoji i da li je vreme modifikacije JSP stranice starije od servleta. Ukoliko je vreme modifikacije JSP stranice starije od servleta, JSP kontejner će smatrati da JSP stranica nije promenjena i da se generisani servlet još uvek podudara sa JSP sadržajem. Navedeno čini procesiranje efikasnijim, nego što je slučaj sa drugim skripting jezicima (kao što je PHP), a otuda i bržim.

JSP stranicom se rukuje na isti način kao "regularnim" servletom.

✓ 1.1 JSP životni ciklus

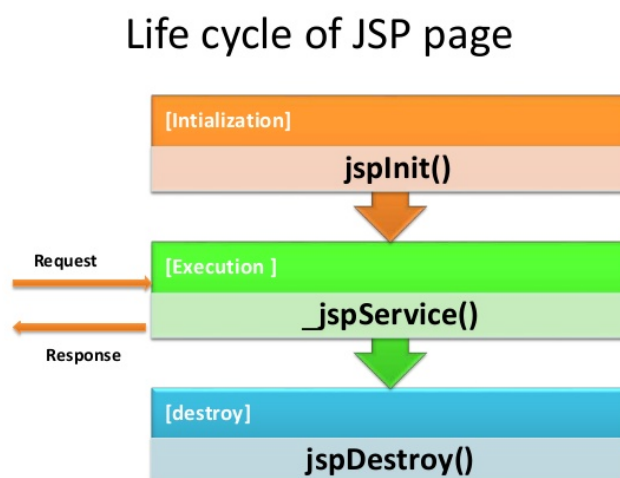
PRIKAZ ŽIVOTNOG CIKLUSA JSP

U ovom delu lekcije biće detaljno opisan JSP životni ciklus sa svim pripadajućim fazama.

U ovom delu lekcije biće detaljno opisan JSP životni ciklus sa svim pripadajućim fazama. JSP životni ciklus može biti shvaćen kao kompletan proces od JSP kreacije do destrukcije, slično kao kod servleta, uključujući poseban, dopunski, korak kojim se vrši prevođenje JSP stranice u servlet. U tom svetlu slede koraci koji karakterišu primenu svake JSP stranice:

- Kompajliranje;
- Inicijalizacija;
- Izvršavanje;
- Čišćenje.

Naveden faze životnog ciklusa JSP stranice, veoma podsećaju na životni ciklus servleta. Za lakše razumevanje, moguće ih je ilustrovati sledećom slikom.



Slika 1.2.1 JSP funkcionisanje nakon koraka "kompajliranje"

PRIMER 1 - FAZE ŽIVOTNOG CIKLUSA JSP

U narednom izlaganju predstoji osvrt na pojedinačne faze životnog ciklusa JSP stranice.

U narednom izlaganju predstoji osvrt na pojedinačne faze životnog ciklusa JSP stranice.

Kompajliranje - Kada veb pregledač zatraži JSP stranicu, JSP kontejner će prvo izvršiti proveru da li je neophodno prvo kompajlirati stranicu. Ukoliko stranica, do tada, nije bila prevedena, ili je modifikovana od vremena prethodnog prevođenja, JSP kontejner će izvršiti prevođenje konkretne JSP stranice. Ovaj proces se odigrava u tri koraka:

- Parsiranje JSP stranice;
- Prevođenje JSP stranice u servlet;
- Kompajliranje servleta.

JSP inicijalizacija - Kada kontejner učitava JSP stranicu on poziva `jspInit()` metodu pre servisiranja bilo kakvog zahteva. Za izvršavanje neke specifične JSP inicijalizacije, neophodno je redefinisati metodu `jspInit()` na način priložen sledećim kodom:

```
public void jspInit(){  
    // Kod inicijalizacije  
}
```

Inicijalizacija se izvodi samo jednom kao i `init()` servlet metoda.

JSP izvršavanje - U ovoj fazi životnog ciklusa JSP stranice dešavaju se sve interakcije sa zahtevima, do konačnog uništenja JSP stranice. Kada veb pregledač zahteva JSP stranicu i kada je stranica inicijalizovana i učitana, JSP kontejner poziva metodu `jspService()` u JSP. Ova metoda preuzima dva parametra, `HttpServletRequest` i `HttpServletResponse`, kao vlastite parametre na način prikazan sledećim kodom:

```
void jspService(HttpServletRequest request, HttpServletResponse response)  
{  
    // Kod za rukovanje servisom...  
}
```

Metoda `jspService()` se poziva jedanput po zahtevu i odgovorna je za kreiranje odgovora po tom zahtevu. Takođe, ova metoda je odgovorna za kreiranje odgovora po svim HTTP zahtevima GET, POST, DELETE i tako dalje.

JSP čišćenje - Faza destrukcije JSP stranice odnosi se na uklanjanje JSP stranice iz upotrebe od strane kontejnera. Metoda `jspDestroy()` je u potpunosti odgovarajuća servlet metodi `destroy()`. Redefinisanje metode `jspDestroy()` kada se zahteva izvođenje bilo kakvog čišćenja, na primer oslobađanje konekcije baze podataka ili zatvaranje otvorenog fajla, izvodi se veoma jednostavno i na sledeći, opšti, način:

```
public void jspDestroy() {  
    // Kod za čišćenje  
}
```

ZADATAK 1

JSP kod - analiza izvršenja

Šta će se dogoditi ako se izvrši sledeći kod? (možete da koristite Internet)


```
<html>
  <head>
    <title>Demo JSP</title>
  </head>
  <%
    int demvar=0;%>
  <body>
    Count is:
    <% Out.println(demovar++); %>
  </body>
</html>
```

▼ Poglavlje 2

JSP sintaksa

SKRIPTLET (SCRIPTLET)

U ovom delu lekcije biće predložena sintaksa uključen u JSP razvoj.

Osnovni element JSP sintakse se naziva skriptlet (eng. scriptlet). Skriptlet je određen sledećom osnovnom sintaksom:

```
<% deo koda %>
```

Skriptletom može biti obuhvaćen bilo koji broj Java iskaza, promenljivih ili deklaracija metoda, ili izraza koji su validni u nekom skripting jeziku stranica.

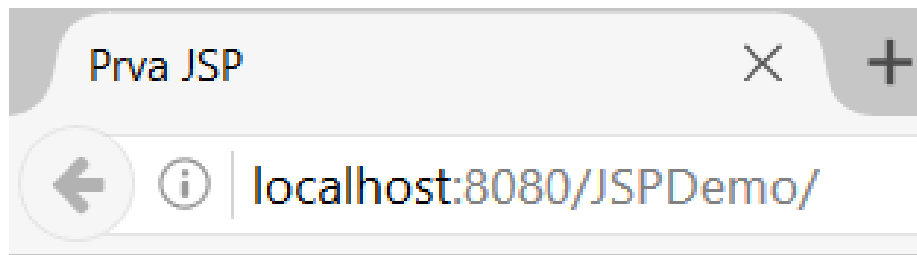
Svaki skriptlet ima ekvivalentan XML prikazan sledećom opštom sintaksom:

```
<jsp:scriptlet> deo koda </jsp:scriptlet>
```

Kada se kreira skriptlet važno je imati i sledeće na umu - svi HTML tagovi ili JSP elementi, moraju da se nalaze izvan skriptleta. Sledećim listingom je priložen jedan skriptlet u okviru jednostavne JSP stranice.

```
<html>
  <head>
    <title>Prva JSP</title>
  </head>
  <body><br> Prva JSP!
    <br/> <% out.println("Nas Univezitet je Metropolitan!!! "); %>
    <br/><% out.println("Zbir brojeva 2 i 3 je " + (2 + 3)); %>
  </body>
</html>
```

Izvršavanjem ove datoteke, pod nazivom **index.jsp**, korisnik će u veb pregledaču videti sledeći sadržaj.



Prva JSP!
Nas Univezitet je Metropolitan!!!
Zbir brojeva 2 i 3 je 5

Slika 2.1.1 Prva JSP stranica

PRIMER 2- JSP DEKLARACIJE

Ovaj deo lekcije pokazuje implementaciju Java deklaracija u JSP stranicama.

Pomoću deklaracija se uvode promenljive, konstante i metode u program. Java programski jezik ima jasno definisana pravila pomoću kojih se deklaracije uvode u programski kod. Kada su u pitanju Java deklaracije i njihova implementacija u JSP stranicama, obavezna je njihova upotreba da bi odgovarajući sadržaj (promenljive, konstante ili metode) mogao da bude korišćen u JSP stranicama. Sledećim iskazom je data opšta sintaksa JSP deklaracije:

```
<%! declaration; [ declaration; ]+ ... %>
```

Moguće je koristiti i XML ekvivalent na sledeći način:

```
<jsp:declaration>  
  code fragment  
</jsp:declaration>
```

Sledećim primerom priloženo je nekoliko deklaracija radi ilustrovanja načina njihovog pisanja prilikom kreiranja JSP stranica.

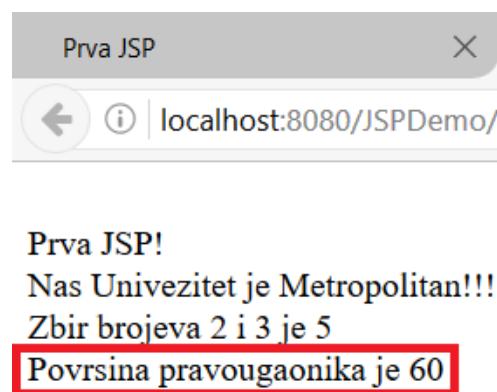
```
<%! int i = 0; %>  
<%! int a, b, c; %>  
<%! Circle a = new Circle(2.0); %>
```

Koristeći kreirani primer, biće deklarirano nekoliko promenljivih koje će, nakon toga, biti iskorišćene kao deo nekog Java izraza u JSP stranici index.jsp.

Navedeno je realizovano sledećim programskim kodom.

```
<html>
  <head>
    <title>Prva JSP</title>
  </head>
  <body><br> Prva JSP!
    <br/> <% out.println("Nas Univezitet je Metropolitan!!! "); %>
    <br/><% out.println("Zbir brojeva 2 i 3 je " + (2 + 3)); %>
    <% int a=5; %>
    <% int b=12; %>
    <% int p = a * b; %>
    <br/><% out.println("Povrsina pravougaonika je " + p); %>
    <br/><%= 3 + 2 %>
  </body>
</html>
```

Izvršavanjem ovog koda, korisnik u veb pregledaču vidi sledeći sadržaj.



Slika 2.1.2 Primena JSP deklaracija - rezultat

PRIMER 3- JSP IZRAZI

Element koji odgovara JSP izrazu sadrži programski izraz.

U prethodnom primeru, ako se dobro pogleda kod, moguće je primetiti da je već ugrađen jedan Java izraz u kreiranu JSP stranicu kojim se računa zbir dva cela broja. U narednom izlaganju je neophodno dati detaljniji opis pisanja i primene JSP izraza.

Element koji odgovara JSP izrazu sadrži programski izraz koji je izračunat, pretvoren u String i umetnut na odgovarajuće mesto u JSP fajlu. Budući da je vrednost izraza konvertovana u String, moguće je koristiti izraz između linija teksta, bez obzira da li je obeležen HTML elementima u JSP datoteci.

Element izraza može da sadrži bilo koji izraz koji je ispravan prema specifikacijama Java programskog jezika ali nije dozvoljeno korišćenje znaka ";" na kraju izraza. Sledi opšti oblik JSP izraza:

```
<%= izraz %>
```

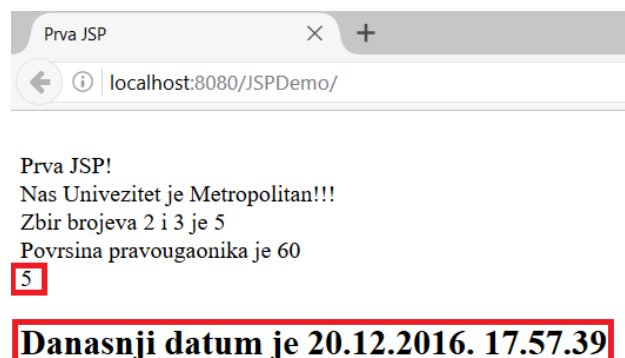
Prethodnom skriptletu je ekvivalentan sledeći XML kod:

```
<jsp:expression>  
izraz  
</jsp:expression>
```

Za demonstraciju u stranicu index.jsp biće ugrađene nove linije koje predstavljaju primenu JSP izraza. Sledi odgovarajući listing.

```
<html>  
  <head>  
    <title>Prva JSP</title>  
  </head>  
  <body><br> Prva JSP!  
    <br/> <% out.println("Nas Univezitet je Metropolitan!!! "); %>  
    <br/><% out.println("Zbir brojeva 2 i 3 je " + (2 + 3)); %>  
    <% int a=5; %>  
    <% int b=12; %>  
    <% int p = a * b; %>  
    <br/><% out.println("Povrsina pravougaonika je " + p); %>  
    <br/><%= 3 + 2 %>  
    <h2> Danasnji datum je <%= (new java.util.Date()).toLocaleString()%> </h2>  
  </body>  
</html>
```

Izvršavanjem ovog koda, korisnik će u pregledaču videti sledeće elemente.



Slika 2.1.3 Primen JSP izraza

PRIMER 4- PISANJE JSP KOMENTARA

Komentari su veoma značajni i nose važne informacije u napisanom kodu.

JSP komentar predstavlja tekst ili iskaz koji će JSP kontejner ignorisati. Komentari su veoma značajni i nose važne informacije u napisanom kodu koje je neophodno sakriti od prikazivanja u veb pregledaču. Sledećom linijom je prikazan opšti oblik JSP komentara:

```
<%-- Ovo je JSP komentar --%>
```

U prethodno kreirani JSP fajl, pod nazivom index.jsp, moguće je ugraditi nekoliko komentara sa ciljem isticanja pojedinih linija koda. Neka je navedeno urađeno na sledeći način:

```
<html>
  <head>
    <title>Prva JSP</title>
  </head>
  <body><br> Prva JSP!
    <%-- Podaci o nasem Univerzitetu --%>
    <br/> <% out.println("Nas Univezitet je Metropolitan!!! "); %>

    <br/><% out.println("Zbir brojeva 2 i 3 je " + (2 + 3)); %>

    <%-- Deklaracije u JSP --%>
    <% int a=5; %>
    <% int b=12; %>
    <% int p = a * b; %>

    <%-- Sledeca linija racuna povrsinu pravougaonika --%>

    <br/><% out.println("Povrsina pravougaonika je " + p); %>

    <%-- Koriscenje JSP izraza --%>
    <br/><%= 3 + 2 %>
    <h2> Danasnji datum je <%= (new java.util.Date()).toLocaleString()%> </h2>
  </body>
</html>
```

Postoji izvestan broj specijalnih konstrukcija koje je moguće koristiti u različitim scenarijima primene komentara u JSP datotekama za umetanje komentara ili specijalnih simbola koji bi u drugim slučajevima bili interpretirani drugačije. Sledećom tabelom je dat njihov pregled.

Sintaksa	Svrha
<%-- comment --%>	JSP komentar. Ignoriše ga JSP kontejner.
<!-- comment -->	HTML komentar. Ignoriše ga pregledač.
<\%	Reprezentuje statički literal <\%
%\>	Reprezentuje statički literal %\>
\'	Reprezentuje apostrof
\"	Reprezentuje navodnik

Slika 2.1.4 Specijalne konstrukcije u komentarima

PRIMER 5- GRANANJE U JSP KODU

Neophodno je pokazati kako se instrukcije granjanja realizuju u JSP stranicama.

U JSP kodu je moguće primeniti sve Java naredbe grananja: **if - else**, kao i **switch - case**.

Kod strukture **if - else**, blok počinje kao najobičniji skriptlet. Međutim, skriptlet je zatvoren u svakoj liniji u kojoj se pojavljuje HTML tekst uključen između skriptlet tagova.

Sledećim listingom je priložena primena **if - else** naredbe u JSP stranicama:

```
<%! int dan = 3; %>
<html>
<head>
<title>IF...ELSE Primer</title>
</head>
<body> <% if (dan == 6 | dan == 7) { %>
<p> Danas je vikend</p>
<% } else { %>
<p> Danas nije vikend</p>
<% } %>
</body>
</html>
```

Kada se radi o **switch - case** bloku, pisanje je poprilično pojednostavljeno. Korišćenjem naredbe `out.println()` ceo blok može da se smesti u okviru jednog skriptleta. Navedeno je ilustrovano sledećim listingom:

```
<%! int dan = 3; %>
<html>
<head>
<title>SWITCH...CASE - primer</title>
</head>
<body>
<% switch(dan) {
case 0: out.println("Dan je ponedeljak."); break;
case 1: out.println("Dan je utorak."); break;
case 2: out.println("Dan je sreda."); break;
case 3: out.println("Dan je cetvrtak."); break;
case 4: out.println("Dan je petak."); break;
case 5: out.println("Dan je subota."); break;
default: out.println("Dan je nedelja."); } %>
</body>
</html>
```

PRIMER 6- PETLJE U JSP KODU

Neophodno je pokazati kako se instrukcije iteracije realizuju u JSP stranicama.

U narednom izlaganju je neophodno pokazati kako se Java petlje pišu i koriste unutar JSP stranica. Takođe, trebalo bi napomenuti da su je u ovom slučaju podržana primena sva tri bloka petlji: **for**, **while** i **do - while**.

Primer upotrebe petlje **for** prikazan je sledećim listingom koji je izolovan iz neke JSP stranice:

```
<%! int fontSize; %>
<html>
<head>
<title>FOR LOOP Example</title>
</head>
<body>
<%for ( fontSize = 1; fontSize <= 3; fontSize++){ %>
    <font color="green" size="<%= fontSize %>">
        JSP Tutorial
    </font><br/>
<%}%>
</body>
</html>
```

Sledećom sliko je pokazan rezultat izvršavanja ovog *for* bloka.



Slika 2.1.5 Izvršavanje JSP strancie sa for blokom

Prethodni primer može da bude urađen na način da umesto *for* bloka, koristimo *while*. Navedeno može biti implementirano primenom sledećeg listinga:

```
<%! int fontSize=1; %>
<html>
<head>
<title>WHILE petlja - primer</title>
</head>
<body> <%while ( fontSize <= 3){ %>
    <font color="green" size="<%= fontSize %>">
        JSP Tutorial
    </font>
    <br/> <%fontSize++;%>
    <%}%>
</body>
</html>
```

Rezultat izvršavanja je identičan slici broj 5. Imajući u vidu prikazane listinge, veoma je lako implementirati *do - while* blok u JSP stranici.

```
<%! int fontSize=1; %>
<html>
<head>
<title>Do - WHILE petlja - primer</title>
</head>
```



```
<body> <%do){ %>
    <font color="green" size="<%= fontSize %>">
    JSP Tutorial
    </font>
    <br/> <%fontSize++;%>
    <%} while ( fontSize > 3)%>
</body>
</html>
```

JSP OPERATORI I LITERALI

Cilj ovog dela lekcije je davanje liste operatora koje je moguće koristiti u JSP stranicama.

Svi logički i aritmetički operatori Java jezika podržani su u JSP stranicama. Sledećom tabelom je priložena lista operatora poređanih, od vrha ka dnu, po prioritetu izvršavanja.

Precedence	Operator	Operand type	Description
1	++, --	Arithmetic	Increment and decrement
1	+, -	Arithmetic	Unary plus and minus
1	~	Integral	Bitwise complement
1	!	Boolean	Logical complement
1	(type)	Any	Cast
2	*, /, %	Arithmetic	Multiplication, division, remainder
3	+, -	Arithmetic	Addition and subtraction
3	+	String	String concatenation
4	<<	Integral	Left shift
4	>>	Integral	Right shift with sign extension
4	>>>	Integral	Right shift with no extension
5	<, <=, >, >=	Arithmetic	Numeric comparison
5	instanceof	Object	Type comparison
6	==, !=	Primitive	Equality and inequality of value
6	==, !=	Object	Equality and inequality of reference
7	&	Integral	Bitwise AND
7	&	Boolean	Boolean AND
8	^	Integral	Bitwise XOR
8	^	Boolean	Boolean XOR
9		Integral	Bitwise OR
9		Boolean	Boolean OR
10	&&	Boolean	Conditional AND
11		Boolean	Conditional OR
12	?:	N/A	Conditional ternary operator
13	=	Any	Assignment

Slika 2.1.6 Java operatori podržani u JSP

U JSP Izrazima dozvoljena je upotreba svih literala koji se koriste u programskom jeziku Java. JSP koristi sledeće literalne:

- **Boolean:** *true* i *false*;
- Celobrojni (*int*): isto kao i u Java programskom jeziku;
- pokretna tačka (*float* i *double*) point: as in Java
- **String:** sa jednostrukim i dvostrukim navodnicima i specijalnim karakterima;
- Null literal: *null*

Pokazni primer koji će pratiti izlaganje u ovom delu lekcije, dostupan Vam je za preuzimanje odmah nakon ovog objekta učenja u aktivnosti Shared Resources.

▼ 2.1 JSP direktive

JSP DIREKTIVE - PREGLED

Direktive utiču na celokupnu strukturu servlet klase.

JSP direktive predstavljaju instrukcije koje imaju implikacije na celokupnu strukturu servlet klase. Upravo iz navedenog razloga, u ovom delu lekcije, neophodno ime je posvetiti odgovarajuću pažnju.

Direktiva, u JSP notaciji, određena je sledećim opštim kodom:

```
<%@ direktiva atribut="vrednost" %>
```

Postoje tri tipa JSP direktiva za koje je potrebno dati detaljan opis u sledećem izlaganju. Navedeno je realizovano korišćenjem sledeće tabele.

Direktiva	Opis
<%@ page ... %>	Definiše attribute zavisne od stranice, kao što su skripting jezici, stranice sa greškama, zahtevi bafera...
<%@ include ... %>	Uključuje datoteku tokom faze prevođenja (translacije).
<%@ taglib ... %>	Deklariše tag biblioteku, sadržeći odgovarajuću akciju koja se koristi u stranici.

Slika 2.2.1 JSP direktive

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 7 - PAGE DIREKTIVA

Page direktiva obezbeđuje informacije kontejneru koje se odnose na tekuću JSP stranicu.

Page direktiva se koristi za obezbeđivanje informacije kontejneru koje se odnose na tekuću JSP stranicu. Ovu direktivu je moguće kodirati bilo gde u JSP stranici. Međutim, poštujući konvenciju, direktiva tipa page se kodira na vrhu JSP stranice. Sledećim kodom je data opšta sintaksa ove direktive:

```
<%@ page atribut="vrednost" %>
```

Sledi XML ekvivalent za opštu sintaksu page direktive:

```
<jsp:directive.page atribut="vrednost" />
```

Od posebnog je značaja isticanje liste atributa koji su u direktnoj vezi sa ovom direktivom. Uporedo sa navođenjem liste pomenutih atributa, neophodno je dati i opis njihove namene.

Navedeno je prikazano sledećom tabelom.

Atribut	Namena
buffer	Specificira buffering model za izlazni tok
autoFlush	Kontroliše ponašanje servletovog izlaznog bafera
contentType	Definiše šemu kodiranja karaktera.
errorPage	Definiše URL druge JSP za izveštaje o Java izuzecima (runtime).
isErrorPage	Ukazuje da li je neka JSP stranica označena kao stranica za izveštavanje o greškama.
extends	Specificira super klasu koju servlet mora da proširi.
import	Specificira listu paketa ili klasa koje Java kod ugrađen u JSP zahteva za izvršavanje.
info	Definiše String kojem je moguće pristupiti servletovoms <code>getServletInfo()</code> metodom.
isThreadSafe	Definiše threading model za generisani servlet.
language	Definiše programski jezik koje se koristi u JSP stranici.
session	Specificira da li, ili ne, JSP stranica učestvuje u HTTP sesijama.
isELIgnored	Specificira da li će, ili ne, EL izraz u JSP stranici biti ignorisan.
isScriptingEnabled	Određuje da li su scripting elementi dozvoljeni za upotrebu.

Slika 2.2.2 Lista i opis atributa direktive page

PRIMER 8 - DIREKTIVA INCLUDE

Direktiva include se koristi da uključi fajl tokom faze translacije

Direktiva include se koristi da uključi fajl tokom faze translacije. Ova direktiva govori JSP kontejneru da spoji sadržaj iz eksternog fajla sa postojećim JSP sadržajem tokom faze translacije. Ovu direktivu je moguće kodirati bilo gde u JSP stranci.

Ova direktiva određena je sledećim opštim kodom:

```
<%@ include file="relative url" >
```

Navedenom kodu odgovara sledeći XML ekvivalent:

```
<jsp:directive.include file="relative url" />
```

Za lakše razumevanje biće proširen aktuelni primer datotekom **brojac.jsp**, čiji je zadatak prebrojavanje broja pristupa stranici **index.jsp**. Direktivom include, u datoteci **index.jsp**. kodovi ovih JSP datoteka biće povezani. Sledi listing datoteke **brojac.jsp**.

```
<%!  
    int pageCount = 0;  
    void addCount() {  
        pageCount++;  
    }  
%>  
<% addCount(); %>  
<html>  
<head>  
<title>Primer direktive include</title>  
</head>  
<body>  
<center>  
<h2>Primer direktive include</h2>  
<p>Stranica je posećena <%= pageCount %> puta.</p>  
</center>  
<br/><br/>
```

Sledi još malo ulepšavanja koda novom datotekom footer.jsp

```
<br/><br/>  
<center>  
<p>Copyright © 2016 by KI204</p>  
</center>  
</body>  
</html>
```

Direktive include se ugrađuju u index.jsp stranicu zbog realizovanja navedenih funkcionalnosti.

```
<%@ include file="WEB-INF/brojac.jsp" %>  
<center>  
    <p>Hvala sto ste posetili nasu stranicu.</p>  
      
</center>  
<%@ include file="WEB-INF/footer.jsp" %>
```



Slika 2.2.3 Stranica za demonstraciju direktive include,.

PRIMER 9 - DIREKTIVA TAGLIB

Java Server Pages API dozvoljava kreiranje prilagođenih JSP tagova.

Java Server Pages API dozvoljava kreiranje prilagođenih JSP tagova koji podsećaju na HTML ili XML tagove. Tag biblioteka predstavlja skup tagova koje je definisao korisnik da bi implementirali određeno ponašanje.

Direktiva `taglib` deklarise da JSP stranica koristi skup prilagođenih tagova, identifikuje lokaciju biblioteke i obezbeđuje sredstva za identifikaciju prilagođenih JSP tagova u konkretnoj JSP stranici.

Direktiva `taglib` je određena sledećom opštom sintaksom:

```
<%@ taglib uri="uri" prefix="prefiksTaga" >

<!-- ili kao XML ekvivalent -->

<jsp:directive.taglib uri="uri" prefix="prefixOfTag" />
```

Vrednost atributa `uri` odnosi se na lokaciju koju kontejner razume, a atribut `prefiks` informiše kontejner o radnjama koje slede i obeležene su prefiksom. Navedeno je moguće ilustrovati sledećim listingom:

```
<%@ taglib uri="http://www.example.com/custlib" prefix="mytag" %>
<html>
<body>
<mytag:hello/>
</body>
</html>
```

▼ 2.2 JSP akcije

UVOD U JSP AKCIJE

U ovom delu lekcije će biti prezentovane JSP akcije koje se često primenjuju u veb aplikacijama.

JSP akcije su elementi XML sintakse za kontrolu ponašanja servlet kontejnera. Ovde je moguće dinamički umetnuti datoteku, ponovo koristiti JavaBeans komponente, proslediti korisnika na drugu stranicu ili generisati HTML za Java dodatak (plugin). Postoji samo jedan način korišćenja sintakse vezane za akciju i on odgovara XML standardu. Opšti prikaz JSP akcije dat je sledećim kodom:

```
<jsp:naziv_akcije atribut = "vrednost" />
```

Elementi akcije su uglavnom predefinisane funkcije. Sledećom tabelom je priložena lista dostupnih JSP akcija zajedno sa njihovim objašnjenjima.

Sintaksa	Namena
jsp:include	Uključuje fajl u trenutku kada je stranica zahtevana
jsp:useBean	Pronalazi instance za JavaBean
jsp:setProperty	Podešava osobine za JavaBean
jsp:getProperty	Dodaje osobine JavaBean u izlaz
jsp:forward	Prosleđuje zahtev novoj stranici
jsp:plugin	Generiše specifični kod pregledača za kreiranje OBJECT ili EMBED tagova za Java plugin - ove
jsp:element	Dinamički definiše XML elemente
jsp:attribute	Definiše attribute dinamički definisanih XML elemenata
jsp:body	Definiše telo dinamički definisanih XML elemenata.
jsp:text	Piše šablonski tekst u JSP stranicama i dokumentima.

Slika 2.3.1 JSP akcije sa opisom

Za sve element, koji odgovaraju JSP akcijama, karakteristična su dva atributa:

- **Id atribut** - Id atribut jedinstveno određuje element akcije i dozvoljava da akcija bude referencirana unutar JSP stranice. Ukoliko akcija kreira objekat, Id vrednost može biti upotrebljena da ga referencira kroz implicitni **PageContext** objekat;
- **Scope atribut** - Ovaj atribut određuje životni ciklus elementa akcije. Id i Scope atribut su direktno povezani budući da Scope atribut određuje životni vek objekta obeleženog Id atributom. Scope atribut može da ima jednu od sledeće četiri vrednosti: **page**, **request**, **session** i **application**.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PRIMER 10 - AKCIJA TIP A JSP:INCLUDE

Sledi obrada pojedinačnih tipova JSP akcija.

Ovom akcijom je omogućeno umetanje fajlova u stranicu koja se kreira. Opšta sintaksa akcije `include` data je sledećim kodom:

```
<jsp:include page="relative URL" flush="true" />
```

Za razliku od direktive tipa `include`, koja vrši umetanje datoteke tokom vremena prevođenja JSP stranice u servlet, akcija vrši umetanje datoteke u trenutku kada je JSP stranica zahtevana.

U sledećem izlaganju će biti priloženi atributi koji su u vezi sa `include` akcijom:

- `page` - Relativni URL stranice koja se uključuje;
- `flush` - Logički atribut koji određuje da li uključeni resurs poseduje osvežen bafer pre uključivanja u JSP stranicu.

Ova akcija može biti demonstrirana odgovarajućim primerom. Biće definisane dve stranice. Prvom se učitava današnji datum i naziva se `date.jsp`, a drugom stranicom se preuzima kreirana `date.jsp` stranica, primenom `include` akcije, i vrši se prikazivanje odgovarajućeg JSP sadržaja korisniku. Ova stranica može da odgovara `index.jsp` stranici.

Sledi kod za `date.jsp`.

```
<p>
  Danasnji datum je: <%= (new java.util.Date()).toLocaleString()%>
</p>
```

U nastavku, kao što je već i istaknuto, definiše se JSP datoteka kojom je implementirana `include` akcija. Sledećim listingom je priložen kod ove datoteke:

```
<html>
<head>
<title>Primer akcije include</title>
</head>
<body>

<center>
<h2>Primer akcije include</h2>
<jsp:include page="WEB-INF/date.jsp" flush="true" />
</center>

</body>
</html>
```

Pokretanjem kreirane veb aplikacije, stranicom `index.jsp` će biti prikazan sledeći sadržaj korisniku.

Primer akcije include

Danasnji datum je: 21.12.2016. 13.39.59

Slika 2.3.2 Demonstracija akcije include

PRIMER 11 - AKCIJA TIPA JSP:USEBEAN

Akcija tipa `jsp:useBean` je sledeća JSP akcija koja će biti obrađena.

Akcija pod nazivom `useBean` je veoma svestrana. Ona prvo traži postojeće objekte angažujući varijable `Id` i `Scope`. Ukoliko objekat nije pronađen, ona će nastojati da kreira specificirani objekat.

Najjednostavniji način da se učitava zrno jeste primenom sledećeg opšteg koda:

```
<jsp:useBean id="naziv" class="paket.klasa" />
```

Jednom kada je zrno učitano, na scenu stupaju nove mogućnosti. One se odnose na primenu akcija `jsp:setProperty` i `jsp:getProperty` za modifikovanje i vraćanje osobina zrna.

Sledećom tabelom je data lista atributa koji su pridruženi akciji `useBean`.

Atribut	Opis
class	Označava pun naziv paketa zrna.
type	Označava tip promenljive koja će da ukazuje na objekat.
beanName	Daje naziv zrna na način specificiran <code>instantiate()</code> metodom klase <code>java.beans.Beans</code> class.

Slika 2.3.3 Atributi akcije `useBean` i njihov opis

U nastavku izlaganja neophodno je diskutovati o `jsp:setProperty` i `jsp:getProperty` akcijama i nakon obavljene analize i diskusije izvršiti odgovarajuću demonstraciju u formi kreiranog primera.

AKCIJA TIPA JSP:SETPROPERTY

Podešavanje osobina zrna obavlja se primenom JSP akcije `jsp:setProperty`.

Kao nastavak izlaganja vezan za primenu akcije `jsp:useBean`, ističe se analiza i diskusija u vezi sa primenom akcije `jsp:setProperty`. **Ovom akcijom je omogućeno podešavanje osobina zrna.** Postoje dva načina pomoću kojih je moguće koristiti ovu akciju.

Prvo, moguće je koristiti `jsp:setProperty` akciju nakon i izvan elementa `jsp:useBean`, na način prikazan sledećim opštim kodom:

```
<jsp:useBean id="myName" ... />
...
<jsp:setProperty name="myName" property="nekaOsobina" .../>
```

U ovom slučaju, akcija `jsp:setProperty` se izvršava u zavisnosti od toga da li je novo zrno instancirani ili je postojeće pronađeno.

U drugom slučaju, akcija `jsp:setProperty` se može naći unutar tela elementa `jsp:useBean` element, na način prikazan sledećim opštim kodom:

```
<jsp:useBean id="myName" ... >
...
    <jsp:setProperty name="myName" property="nekaOsobina" .../>
</jsp:useBean>
```

U ovom slučaju, akcija `jsp:setProperty` se izvršava isključivo u slučaju kada je instancirano novo zrno.

Sledećom tabelom je priložena lista atributa, sa odgovarajućim opisima, koji odgovaraju akciji `jsp:setProperty`.

Atribut	Opis
name	Označava zrno čija će osobina biti podešena. Zrno mora biti prethodno definisano.
property	Označava promenljivu koja će biti podešena. Vrednost "*" ukazuje da će svi parametri čiji se nazivi podudaraju sa nazivima osobina zrna biti prosleđeni do odgovarajućih seter metoda.
value	Označava vrednost koja će biti pridružena odgovarajućoj osobini. Ukoliko parametar ne postoji, ili je njegova vrednost null, akcija <code>setProperty</code> će biti ignosrisana.
param	Označava parametar zahteva čiju vrednost osobina prima.

Slika 2.3.4 Atributi akcije `jsp:setProperty`

AKCIJA TIPA JSP:GETPROPERTY

Preuzimanje osobina zrna obavlja se primenom JSP akcije `jsp:getProperty`.

Sledeća akcija koju je neophodno obraditi je inverzna prethodnoj akciji. **Preuzimanje osobina zrna obavlja se primenom JSP akcije `jsp:getProperty`**. Preciznije rečeno, primenom akcije `jsp:getProperty` vraća se vrednost posmatrane osobine, konvertuje se u String i, nakon toga, se prosleđuje kao izlaz.

Akcija `jsp:getProperty` poseduje samo dva atributa, od kojih su oba obavezna i primenjuju se na način prikazan sledećom opštom sintaksom:

```
<jsp:useBean id="myName" ... />
...
<jsp:getProperty name="myName" property="nekaOsobina" .../>
```

Sledećom tabelom je priložena lista atributa metode `jsp:getProperty` zajedno sa odgovarajućim objašnjenjima.

Atribut	Opis
name	Označava zrno čija će osobina biti vraćena. Zrno mora biti prethodno definisano.
property	Označava naziv osobine zrna koja će biti vraćena.

Slika 2.3.5 Lista atributa metode `jsp:getProperty` zajedno sa odgovarajućim objašnjenjima

Sada je moguće uvesti odgovarajući primer za demonstraciju funkcionisanja obrađenih JSP akcija.

PRIMER 12 - KORIŠĆENJE JSP AKCIJA - ZRNA U JSP

Uvodi se odgovarajući primer za demonstraciju funkcionisanja obrađenih JSP akcija.

Za demonstraciju obrađenih JSP akcija uvodi se primer koji će obuhvatiti sledeće aktivnosti:

- Kreiranje klase Java zrna za rad sa porukama;
- Kreiranje JSP stranice za implementiranje obrađenih JSP akcija.

Ove datoteke će biti deo web aplikacije pod nazivom **JSPAkcijeDemo**. U NetBeans IDE razvojnom okruženju će biti kreiran projekat sa navedenim nazivom i u folderu Source Packages biće kreirana klasa zrna za rad sa predefinisanim porukama. Sledećim listingom je priložen kod klase `ZrnoTest`.

```
**
 *
 * @author Vladimir Milicevic
 */
package jsp.demo;
public class ZrnoTest {
    private String message = "Poruka nije specificirana!!!";
```

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

U nastavku, neophodno je modifikovati JSP stranicu index.jsp, koja se nalazi u folderu /WEB-INF, na načina kako je to urađeno sledećim kodom:

```
<html>  
<head>  
<title>Primena JavaBeans u JSP</title>  
</head>  
<body>  
<center>  
<h2>Primena JavaBeans u JSP</h2>  
  
<jsp:useBean id="test" class="jsp.demo.ZrnoTest" />  
  
<jsp:setProperty name="test"  
                property="message"  
                value="Pozdrav JSP..." />  
  
<p>Primio poruku....</p>  
  
<jsp:getProperty name="test" property="message" />  
  
</center>  
</body>  
</html>
```

Nakon što je kreirana instanca zrna, pod nazivom test, njoj se menja osobina message odgovarajućom vrednošću. Konačno, ta vrednost se preuzima i prikazuje korisniku na stranici (sledeća slika).

Primena JavaBeans u JSP

Primio poruku....

Pozdrav JSP...

Slika 2.3.6 Primer primene zrna u JSP stranicama

PRIMER 13 - AKCIJA TIP A JSP:FORWARD

Ovom akcijom se prosleđuje aktivnost sa tekuće stranice na drugi resurs.

Akcija jsp: [forward](#) gasiaktivnost na tekućoj stranici i prosleđuje zahtev nekom drugom resursu, kao što je statička stranica, druga JSP stranica ili Java servlet.

Akcija tipa jsp:forward određena je sledećim opštim kodom:

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

Akcija poseduje samo jedan atribut koji je priložen sledećim opisom:

- **page** - Čini ga relativni URL drugog resursa kao što je statička stranica, druga JSP stranica ili Java servlet.

Za demonstraciju ove akcije moguće je ponovo upotrebiti datoteke date.jsp i index.jsp iz primera sa akcijom tipa jsp:include. Ove datoteke će biti prilagođene u skladu sa novim zahtevima.

Sledi listing datoteke date.jsp:

```
<% -
    Document    : date
    Created on  : 21.12.2016., 22.46.29
    Author      : Vladimir Milicevic
- %>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<p>
    Danasnji datum je <%= (new java.util.Date()).toLocaleString()%>
</p>
```

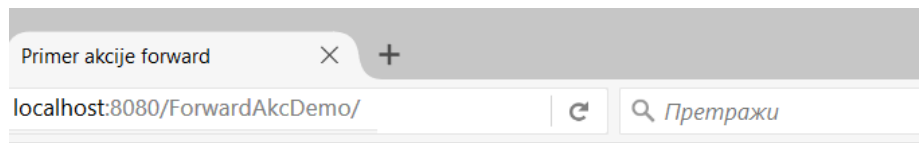
Sada je neophodno redefinisati JSP datoteku index.jsp. Navedeno je realizovano primenom sledećeg koda:

```
<html>
<head>
<title>Primer akcije forward</title>
</head>
<body>

<center>
<h2>Primer akcije forward</h2>
<jsp:forward page="WEB-INF/date.jsp" />
</center>

</body>
</html>
```

Sada je moguće testirati datoteku index.jsp. Izvršavanjem linije koda [<jsp:forward page="WEB-INF/date.jsp" />](#) desiće se preusmeravanje sa JSP stranice index na stranicu JSP date. Rezultat primene navedene akcije korisnik vidi na način prikazan sledećom slikom.



Primer akcije forward

Slika 2.3.7 Izvršavanje akcije jsp:forward

PRIMER 14 - AKCIJA TIPA JSP:PLUGIN

Ovom akcijom se dodaju Java komponente u JSP stranicu.

Akcija tipa jsp:plugin je veoma značajna za ovaj nivo analize i diskusije budući da se pomoću nje u JSP datoteci ugrađuju Java komponente. Ova akcija određuje tip veb pregledača i po potrebi ugrađuje tagove <object> ili <embed>.

Ako traženi dodatak nije dostupan, ovom akcijom će biti preuzet i izvršen kao Java komponenta ugrađena u posmatranu JSP stranicu. Java komponenta može biti applet (napuštena tehnologija zbog bezbednosnih razmatranja) ili java zrnno.

Na sledeći način može biti prikazana tipična primena ove akcije priložena opštim kodom:

```
<jsp:plugin
    type="bean|applet"
    code="className"
    codebase="classFileDirectoryName"
</jsp:plugin>
```

Za primer i lakše razumevanje ove akcije biće priložen konkretan kod. Neka je sledeći kod do neke konkretne JSP stranice.

```
<jsp:plugin type="applet" codebase="dirname" code="MyApplet.class"
            width="60" height="80">
  <jsp:param name="fontcolor" value="red" />
  <jsp:param name="background" value="black" />

  <jsp:fallback>
    Nije moguće inicijalizovati Java Plugin
  </jsp:fallback>
</jsp:plugin>
```

U priloženom kodu moguće je uočiti još jednu JSP akciju pod nazivom jsp:fallback. Ovom akcijom je specificiran String greške u slučaju neuspešnog učitavanja željenog Java dodatka.

Takođe, primećuje se i akcija jsp:param čiji je zadatak slanje parametara zrnno ili appletu.

PRIMER 15 - AKCIJE TIPRA JSP:ELEMENT, JSP:ATTRIBUTE I JSP:BODY

Određenim akcijama je moguće dinamički generisati XML elemente.

Akcije `<jsp:element>`, `<jsp:attribute>` i `<jsp:body>` se koriste za dinamičko generisanje XML elemenata u JSP stranicama. Ovo je veoma značajno i praktično znači da je XML elemente moguće generisati tokom zahteva (dinamički) pre nego za vreme kompajliranja (statički).

Dinamičko generisanje XML elemenata je moguće pokazati sledećim jednostavnim primerom.

```
<%@page language="java" contentType="text/html"%>
<html xmlns="http://www.w3c.org/1999/xhtml"
      xmlns:jsp="http://java.sun.com/JSP/Page">

<head><title><%@page language="java" contentType="text/html"%>
<html xmlns="http://www.w3c.org/1999/xhtml"
      xmlns:jsp="http://java.sun.com/JSP/Page">

<head><title>Generisi XML element</title></head>
<body>
<jsp:element name="xmlElement">
<jsp:attribute name="xmlElementAttr">
    Vrednost atributa
</jsp:attribute>
<jsp:body>
    Telo XML elementa
</jsp:body>
</jsp:element>
</body>
</html>
```

Tokom vremena izvršavanja, prethodno pokazani kod će proizvoditi sledeći HTML kod:

```
<html xmlns="http://www.w3c.org/1999/xhtml"
      xmlns:jsp="http://java.sun.com/JSP/Page">

<head><title>Generisi XML elementt</title></head>
<body>
<xmlElement xmlElementAttr="Vrednost atributa">
    Telo XML elementa
</xmlElement>
</body>
</html>
```

PRIMER 16 - AKCIJA TIPRA JSP:TEXT

Pomoć u definisanju šablonskog teksta koji će biti korišćen u JSP stranicama.

Akcijom tipa `<jsp:text>` moguće je definisati neki šablonski tekst koji će biti korišćen u JSP stranicama i dokumentima. Ova akcija može biti prikazana u sledećem opštem obliku:

```
<jsp:text>Šablonski podaci</jsp:text>
```

Šablonski podaci ne mogu da sadrže ostale elemente. Mogu da sadrže samo tekst i iskaze EL (**Expression Language**) jezika izraza.

Ukoliko je neophodno uključiti u JSP stranicu **DOCTYPE** deklaraciju, na primer za XHTML, takođe mora da se koristi element `jsp:text` akcije. To je moguće uraditi na sledeći način:

```
<jsp:text><![CDATA[<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"DTD/xhtml11-strict.dtd">]]>
</jsp:text>
<head><title>jsp:text action</title></head>
<body>

<books><book><jsp:text>
    Dobrodošli u JSP programiranje
</jsp:text></book></books>

</body>
</html>
```

▼ 2.3 JSP - Implicitni objekti

PREGLED IMPLICITNIH OBJEKATA

JSP podržava devet implicitnih objekata.

JSP implicitni objekti predstavljaju Java objekte koje JSP kontejner čini dostupnim na svakoj stranici programerima. Programeri mogu direktno da ih koriste bez potrebe za njihovim prethodnim eksplicitnim deklarisanjem. JSP implicitni objekti se još nazivaju i predefinisanim varijablama. JSP podržava devet implicitnih objekata koji su navedeni i objašnjeni u sledećem izlaganju:

- **request** - Predstavlja **HttpServletRequest** objekat pridružen zahtevu;
- **response** - Predstavlja **HttpServletResponse** objekat pridružen odgovoru klijentu;
- **out** - Predstavlja **PrintWriter** objekat zadužen da pošalje izlaz ka klijentu;
- **session** - Predstavlja **HttpSession** objekat pridružen zahtevu;
- **application** - Predstavlja **ServletContext** objekat pridružen kontekstu aplikacije;
- **config** - Predstavlja **ServletConfig** objekat pridružen stranici;
- **pageContext** - Enkapsulira primenu specifičnih elemenata servera;
- **page** - Predstavlja sinonim za **this** i koristi se za pozivanje metoda definisanih prevedenom servlet klasom;

- **Exception** - Ovaj objekat dozvoljava pristup podacima izuzetaka odgovarajućom JSP stranicom.

U narednom izlaganju biće posebno analizirani i demonstrirani navedeni JSP implicitni objekti pojedinačno.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

OBJEKAT REQUEST

Objekat request predstavlja instancu klase `HttpServletRequest`.

Objekat `request` predstavlja instancu klase `javax.servlet.http.HttpServletRequest`. Svaki put kada klijent zahteva stranicu JSP kontejner kreira novi objekat kojim se zahtev reprezentuje. Objekat zahteva obezbeđuje veliki broj metoda za dobijanje informacija HTTP zaglavlja, uključujući podatke sa forme, kolačiće, HTTP metode i tako dalje. Kada veb pregledač zahteva stranicu, on šalje izvesne informacije veb serveru koje ne mogu da se vide direktno jer predstavljaju deo zaglavlja HTTP zahteva. Upravo, u narednom izlaganju je neophodno istaći brojne informacije iz HTTP zaglavlja, koje dolaze sa strane pregledača i veoma često se koriste u veb programiranju:

- **Accept** - Ovo zaglavlje specificira MIME tipove kojima pregledač i drugi klijenti mogu da rukuju. Vrednosti `image/png` ili `image/jpeg` predstavljaju dva najčešće korišćena slučaja.
- **Accept-Charset** - Ovo zaglavlje specificira set karaktera koji pregledač može da koristi za prikazivanje informacije. Na primer, `ISO-8859-1`.
- **Accept-Encoding** - Ovo zaglavlje specificira tipove kodiranja kojima pregledač može da rukuje. Vrednosti `gzip` ili `compress` se često koriste.
- **Accept-Language** - Ovo zaglavlje specificira klijentom preferirani jezik u slučaju da servlet može da proizvede rezultate na više jezika. Na primer, `en`, `en-us`, `ru`, `sr` i tako dalje.
- **Authorization** - Ovo zaglavlje koristi klijent da bi se identifikovao kada pristupa lozinkom zaštićenim veb stranicama.
- **Connection** - Ovo zaglavlje ukazuje da li klijent može da rukuje HTTP konekcijom.
- **Content-Length** - Ovo zaglavlje se primenjuje isključivo na POST zahtev i daje veličinu POST podatak u bajtovima.
- **Cookie** - Ovo zaglavlje vraća kolačić serveru koji je prethodno poslat veb pregledaču.
- **If-Modified-Since** - Ovo zaglavlje ukazuje da klijent zahteva stranicu samo u slučaju ako je modifikovana nakon navedenog datuma. Server šalje kod 304, a to znači **Not Modified** zaglavlje, ukoliko noviji rezultat ne postoji.
- **If-Unmodified-Since** - Ovo zaglavlje je obrnuta verzija `If-Modified-Since`.
- **Referer** - Ovo zaglavlje ukazuje URL ukazane veb stranice.
- **User-Agent** - Ovo zaglavlje ukazuje na pregledač ili drugog klijenta koji kreira zahtev i može da se koristi za vraćanje različitog sadržaja, različitim tipovima pregledača.

OBJEKAT RESPONSE

Objekat response predstavlja instancu klase `HttpServletResponse`.

Objekat **response** predstavlja instancu klase `javax.servlet.http.HttpServletResponse`. Upravo kada servlet kreira objekat zahteva, on takođe kreira i objekat kojim je reprezentovan odgovor klijentu. Objekat odgovora, takođe, definiše interfejs koji se bavi kreiranjem novih HTTP zaglavlja. Pomoću ovog objekta programer ima mogućnost dodavanja novih kolačića, oznaka datuma, kodova HTTP statusa i tako dalje.

Kada veb server odgovori na zahtev klijenta (veb pregledača), odgovor se obično sastoji od statusne linije, nekoliko zaglavlja odgovora, prazne linije i dokumenta. Na primer, klasičan odgovor može da izgleda na sledeći način:

```
HTTP/1.1 200 OK
Content-Type: text/html
Header2: ...
...
HeaderN: ...
(Blank Line)
<!doctype ...>
<html>
<head>...</head>
<body>
...
</body>
</html>
```

Statusna linija se sastoji od HTTP verzije (1.1 za potrebe primera), koda statusa (200 za potrebe primera) i kratke poruke (OK za potrebe primera). Nakon toga sledi lista veoma korisnih zaglavlja odgovora koji se prosleđuju nazad do veb pregledača i vide se veoma često tokom razvoja.

Zaglavlje	Opis
Allow	Specificira metodu zahteva - GET, POST ...
Cache-Control	Specificira okolnosti u kojima je dokument odgovora moguće bezbedno keširati. Može imati vrednosti public , private or no-cache . Public znači slobodno keširanje dokumenta, Private ukazuje da je dokument za jednog korisnika i može biti sačuvan u privatnom i nedeljenom kešu i no-cache znači da dokument ne može da bude keširan.
Connection	Daje pregledaču instrukciju da li da koristi perzistenciju u HTTP konekciji, ili ne. Vrednost close nalaže da pregledač ne koristi perzistentne HTTP konekcije, a keep-alive podrazumeva korišćenje preristentne konekcije.
Content-Disposition	Omogućava zahtev pregledaču na osnovu kojeg će on tražiti od korisnika dozvolu da snimi odgovor kao fajl na disku.
Content-Encoding	Specificira način na koji će stranica biti kodirana tokom prenosa.
Content-Language	Označava jezik na kojem je dokument napisan. Na primer en, en-us, ru, sr itd.
Content-Length	Označava broj bajtova odgovora
Content-Type	Daje MIME (Multipurpose Internet Mail Extension) tip dokumenta odgovora.
Expires	Vreme važenja keširanog sadržaja.
Last-Modified	Vreme poslednje izmene dokumenta
Location	Obaveštava pregledač o adresi dokumenta
Refresh	Određuje period ažuriranja stranice.
Retry-After	Koristi se sa kodom 503 (Service Unavailable) i ukazuje klijentu kada da ponovi zahtev.
Set-Cookie	Specificira kolačić pridružen stranici.

Slika 2.4.1 Zaglavlja response objekta

PRIMER 17 - PRIMER HTTP ZAGLAVLJA ODGOVORA

Primenu zaglavlja najbolje je pokazati na konkretnom primeru.

Primenu zaglavlja najbolje je pokazati na konkretnom primeru. Za početak je neophodno kreirati JSP stranicu koja će pokazivati trenutno vreme. Ideja je da se stranica automatski ažurira nakon nekog vremenskog intervala, na primer 5 sekundi. Za omogućavanje navedene funkcionalnosti, neophodno je implementirati metodu `setIntHeader()` koju će pozvati objekat **response**. Metoda uzima dva argumenta, **Refresh** zaglavlje (prikazano u prethodnoj sekciji) i celobrojnu konstantu koja ukazuje na interval (u sekundama) nakon čijeg isteka će biti ažurirana posmatrana stranica.

Sledećim slikama su prikazane početna i "osvežena" stranica.

Primer zaglavlja sa opcijom auto - refresh

Trenutno vreme je : 10:59:49 AM

Slika 2.4.2 Početna stranica

Primer zaglavlja sa opcijom auto - refresh

Trenutno vreme je : 10:59:54 AM

Slika 2.4.3 Osvežena stranica

Kod koji se nalazi u odgovarajućoj JSP datoteci priložen je sledećim listingom:

```
<%@ page import="java.io.*,java.util.*" %>
<html>
<head>
<title>Primer zaglavlja sa opcijom auto - refresh</title>
</head>
<body>
<center>
<h2>Primer zaglavlja sa opcijom auto - refresh</h2>
<%
    // Podesavanje osvezavnja, automatski učitaj za 5s
    response.setIntHeader("Refresh", 5);
    // Vрати текуće време
    Calendar calendar = new GregorianCalendar();
    String am_pm;
    int hour = calendar.get(Calendar.HOUR);
    int minute = calendar.get(Calendar.MINUTE);
    int second = calendar.get(Calendar.SECOND);
```

```
if(calendar.get(Calendar.AM_PM) == 0)
    am_pm = "AM";
else
    am_pm = "PM";
String CT = hour+":"+ minute +":"+ second + " "+ am_pm;
out.println("Trenutno vreme je : " + CT + "\n");
%>
</center>
</body>
</html>
```

OBJEKAT OUT

Objekat out je instanca klase javax.servlet.jsp.JspWriter.

Sledeći JSP implicitni objekat koji je neophodno obraditi naziva se **out**. Objekat out je instanca klase javax.servlet.jsp.JspWriter i koristi se da pošalje neki sadržaj u okviru odgovora.

Po osnovnim podešavanjima **JspWriter** objekat se instancira različito u zavisnosti od toga da li je stranica baferizovana ili ne. Baferizovanje može veoma lako da se isključi navođenjem **buffered = 'false'** za direktivu **page**.

Klasa **JspWriter** deli većinu metoda sa klasom **java.io.PrintWriter class**, međutim ima i neke vlastite metode, specijalno dizajnirane da rukuju sa baferizacijom. Za razliku od **PrintWriter** objekta, **JspWriter** izbacuje **IOExceptions** izuzetke.

Sledećom tabelom su priložene važne metode za pisanje podataka tipa: **boolean**, **char**, **int**, **double**, **object**, **String** i tako dalje.

Metoda	Opis
out.print(dataType dt)	Štampa vrednost tipa data
out.println(dataType dt)	Štampa vrednost tipa data i prelazi u nov red
out.flush()	Osvežava tok.

Slika 2.4.4 JspWriter metode sa opisom

OBJEKAT SESSION

Objekat session je instanca javax.servlet.http.HttpSession klase.

Objekat **session** predstavlja instancu klase **javax.servlet.http.HttpSession** i ponaša se na potpuno isti način kao objekti sesija u Java Servlet klasama. Ovaj objekat se koristi za praćenje sesije klijenta između njegovih zahteva. Praćenje sesije može biti obavljeno na neki od poznatih načina:

- **kolačići** - veb server može da pridruži jedinstveni id sesiji, kao kolačić, za svakog od klijenata i svaki naredni zahtev, konkretnog klijenta, može biti prepoznat na osnovu

sačuvanog kolačića. Nedostatak može da predstavlja činjenica da svi veb pregledači ne podržavaju primenu kolačića.

- **skrivena polja forme** - veb server može da pošalje skriveno polje HTML forme sa jedinstvenim indikatorom sesije, kao na primer:

```
<input type="hidden" name="sessionid" value="12345">
```

- **URL modifikacija** - moguće je dodati dopunske podatke na kraj svake URL adrese koja identifikuje sesiju i server će pridružiti taj identifikator sesije podacima koji su sačuvani u okviru posmatrane sesije. .
- **objekat sesije** - pored navedena tri načina, JSP omogućava upotrebu **HttpSession** interfejsa koji obezbeđuje način identifikovanja korisnika nakon više zahteva za stranicom, ili poseta veb sajtu, čuvanjem podataka koji su u vezi sa korisnikom.

Po osnovnim podešavanjima, praćene sesije je omogućeno i objekat sesije se kreira za svakog novog korisnika, automatski. Sledećom instrukcijom je moguće isključiti navedenu funkcionalnost:

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

Budući da je objekat sesije automatski dostupan programeru, on može istog momenta da počne sa čuvanjem i preuzimanjem podataka od objekta sesije bez potrebe za prethodnom inicijalizacijom ili pozivom metode getSession().

Metoda	Opis
<code>public Object getAttribute(String name)</code>	Vraća objekat sesije sa specificiranim imenom ako postoji, u suprotnom vraća null.
<code>public Enumeration getAttributeNames()</code>	Vraća nabranje svih String objekata koji sadrže imena svih objekata vezanih za sesiju.
<code>public long getCreationTime()</code>	Vraća vreme kreiranja sesije, mereno u milisekundama od ponoći 1. 1. 1970.
<code>public String getId()</code>	Vraća Sting sa jedinstvenim Id pridruženim sesiji.
<code>public long getLastAccessedTime()</code>	Vraća vreme poslednjeg pristupa sesiji, mereno u milisekundama od ponoći 1. 1. 1970.
<code>public int getMaxInactiveInterval()</code>	Vraća najveći interval u kojem je kontejner držao otvorenom sesiju bez kljijentskih akcija.
<code>public void invalidate()</code>	Metoda poništava sesiju i sve objekte koji su u vezi sa njom.
<code>public boolean isNew()</code>	Vraća vrednost true ukoliko je klijent nepoznat za sesiju ili je upravo izabrao da se priključi sesiji.
<code>public void removeAttribute(String name)</code>	Uklanja objekat iz sesije sa specificiranim nazivom.
<code>public void setAttribute(String name, Object value)</code>	Povezuje objekat u sesiji sa specificiranim nazivom.
<code>public void setMaxInactiveInterval(int interval)</code>	Podšava najveće neaktivno vreme u sekundama, nakon kojeg će kontejner ugasiti sesiju.

Slika 2.4.5 Metode objekta sesije

PRIMER 18 - PRAĆENJE SESIJE U JSP STRANICI - PRIMER

Kreiranje JSP stranice za demonstriranje praćenja sesija.

Ovaj primer pokazuje kako se koristi objekat HTTP sesije za pronalaženje vremena kreiranja i poslednjeg pristupa sesiji (pogledati tabelu na prethodnoj sekciji). Ukoliko sesija ne postoji, biće kreirana nova sa odgovarajućim zahtevom.

Sledećom sliko je prikazan izlaz koji na veb pregledaču može da uoči korisnik.

Pracenje sesije

Info o sesiji	Vrednost
id	b1ccbd0d7716a037b7a30233f7d4
Vreme kreiranja	Fri Dec 23 10:57:04 CET 2016
Poslednji pristup	Fri Dec 23 10:58:54 CET 2016
ID korisnika	ABCD
Broj poseta	4

Slika 2.4.6 Praćenje sesije

Sledeći zadatak je sa se metode iz tabele sa prethodne sekcije integrišu u JSP stranicu, primenom objekta sesije, i da se pokaže njihova funkcionalnost.

Za realizovanje traženih funkcionalnosti, moguće je iskoristiti podrazumevanu JSP stranicu `index.jsp`, u okviru novog projekta `JSPPracenjeSesija`, i modifikovati je sledećim kodom:

```
<%@ page import="java.io.*,java.util.*" %>
<%
    // Vraca vreme kreiranja sesije.
    Date createTime = new Date(session.getCreationTime());
    // Vrava poslednje vreme pristupa veb stranici.
    Date lastAccessTime = new Date(session.getLastAccessedTime());

    String title = "Dobrodosli na nasu veb stranicu";
    Integer visitCount = new Integer(0);
    String visitCountKey = new String("visitCount");
    String userIDKey = new String("userID");
    String userID = new String("ABCD");

    // Proverava da li je posetilac nov.
    if (session.isNew()){
        title = "Dobrodosli na nasu veb stranicu";
        session.setAttribute(userIDKey, userID);
        session.setAttribute(visitCountKey, visitCount);
    }
    visitCount = (Integer)session.getAttribute(visitCountKey);
    visitCount = visitCount + 1;
    userID = (String)session.getAttribute(userIDKey);
    session.setAttribute(visitCountKey, visitCount);
%>
<html>
<head>
<title>Session Tracking</title>
</head>
```

```
<body>
<center>
<h1>Pracenje sesije</h1>
</center>
<table border="1" align="center">
<tr bgcolor="#949494">
    <th>Info o sesiji</th>
    <th>Vrednost</th>
</tr>
<tr>
    <td>id</td>
    <td><% out.print( session.getId()); %></td>
</tr>
<tr>
    <td>Vreme kreiranja</td>
    <td><% out.print(createTime); %></td>
</tr>
<tr>
    <td>Poslednji pristup</td>
    <td><% out.print(lastAccessTime); %></td>
</tr>
<tr>
    <td>ID korisnika</td>
    <td><% out.print(userID); %></td>
</tr>
<tr>
    <td>Broj poseta</td>
    <td><% out.print(visitCount); %></td>
</tr>
</table>
</body>
</html>
```

OBJEKAT APPLICATION

Objekat application je direktni omotač ServletContext objekta .

Objekat application je direktni omotač ServletContext objekta generisanog servleta i instanca klase `javax.servlet.ServletContext`. Ovaj objekat predstavlja reprezentaciju JSP stranice kroz sve faze njenog životnog ciklusa. Objekat je kreiran kada je JSP stranica inicijalizovana i biće uklonjen kada JSP stranica bude ugašena pozivom metode `jspDestroy()`.

Sledećom sintaksom se podešava varijabla na nivou aplikacije:

```
application.setAttribute(String Key, Object Value);
```

Preuzimanje podešene varijable, na gore opisani način, moguće je izvesti pozivom sledeće metode:

```
application.getAttribute(String Key);
```

Primena ovog objekta može biti demonstrirana na sledećem primeru - JSP stranica pokazuje ukupan broj poseta veb stranici realizovan primenom **application** objekta (sledeća slika).

Dobrodosli nazad na nasu veb stranicu!!

Ukupan broj poseta: 3

Slika 2.4.7 Praćenje posećenosti

Implementacija objekta **application** i realizovanje traženih funkcionalnosti omogućeno je sledećim kodom:

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

<html>
<head>
<title>Objekat application u JSP</title>
</head>
<body>
<center>
<%
    Integer hitsCount =
        (Integer)application.getAttribute("hitCounter");
    if( hitsCount ==null || hitsCount == 0 ){
        /* Prva poseta stranici */
        out.println("Dobrodosli na nasu veb stranicu!");
        hitsCount = 1;
    }else{
        /* Ponovljena poseta */
        out.println("Dobrodosli nazad na nasu veb stranicu!!");
        hitsCount += 1;
    }
    application.setAttribute("hitCounter", hitsCount);
%>

<p>Ukupan broj poseta: <%= hitsCount%></p>
</center>
</body>
</html>
```

OBJEKTI CONFIG I PAGECONTEXT

U ovoj sekciji će biti posebno obrađeni implicitni objekti config i pageContext.

Implicitni objekat **config** je instanca klase **javax.servlet.ServletConfig** i predstavlja direktan omotač objekta **ServletConfig** generisanog servleta. Ovaj objekat dozvoljava JSP

programerina da pristupaju inicijalizujućim parametrima servleta ili JSP kontejnera kao što su putanje, lokacije datoteka i slično.

Postoji samo jedna metoda **config** objekta i njena primena je krajnje trivijalna i prikazana je sledećim kodom:

```
config.getServletName();
```

Metoda vraća naziv servleta.

Implicitni objekat **pageContext** je instanca klase `javax.servlet.jsp.PageContext`. Ovaj objekat se koristi da reprezentuje celokupnu JSP stranicu. Ovaj objekat je zamišljen kao sredstvo za pristup informacijama o stranici izbegavajući većinu detalja implementacije. Ovaj objekat čuva reference na objekte zahteva i odgovora za svaki kreirani zahtev. Objekti **application**, **config**, **session** i **out** su izvedeni pristupom atributima objekta **pageContext**.

Objekat **pageContext** takođe sadrži informacije o direktivama izdatim na JSP stranici, uključujući bafer informacije, `errorPageURL` i informacije u vezi sa oblašću stranice. Objekat uključuje sledeća polja: **PAGE_SCOPE**, **REQUEST_SCOPE**, **SESSION_SCOPE** i **APPLICATION_SCOPE**, kojima su identifikovane četiri oblasti.

Važna metoda **pageContext** objekta je `removeAttribute()` čiji je zadatak brisanje atributa iz svih ili neke konkretne oblasti, kao što je prikazano sledećim linijama koda respektivno:

```
// brisanje iz svih oblasti
pageContext.removeAttribute("attrName");

// brisanje iz konkretne oblasti
pageContext.removeAttribute("attrName", PAGE_SCOPE);
```

OBJEKTI PAGE I EXCEPTION

Objekti page i exception su poslednja dva implicitna objekta koji će biti obrađeni.

Objekat **page** predstavlja aktuelnu referencu na objekat stranice. Ovaj objekat je direktni sinonim za objekat **this**.

Objekat **Exception** je omotač koji sadrži sve izuzetke koje je stranica izbacila. Ovaj objekat se koristi za generisanje odgovarajućeg odgovora u uslovima pojavljivanja grešaka. U JSP kodu je moguće sresti jedan od sledećih tipova grešaka:

- **Checked exceptions**: Predstavlja grešku korisnika ili grešku koju programer nije mogao da predvidi. Na primer, neophodno je otvoriti fajl, a on ne postoji. Ovi izuzeci ne mogu jednostavno biti ignorisani tokom prevođenja;
- **Runtime exceptions**: Dešavaju se tokom izvršavanja i ovi izuzeci mogu jednostavno biti ignorisani tokom prevođenja;
- **Errors**: Nisu izuzeci i predstavljaju greške izvan kontrole programera.

Objekat **Exception** je potklasa super klase **Throwable** i dostupna je samo na stranicama grešaka. Na osnovu navedenog, jednostavno je zaključiti da od roditeljske klase dobija brojne metode koje se koriste za upravljanje greškama i izuzecima. Više o ovim metodama može se pročitati na linku: http://www.tutorialspoint.com/jsp/jsp_exception_handling.htm.

Za simuliranje rada sa izuzecima biće kreirane stranice **index.jsp** i **PrikaziGresku.jsp** koje su respektivno priložene sledećim listinzima ().

```
<%@ page errorPage="WEB-INF/PrikaziGresku.jsp" %>

<html>
<head>
  <title>Primer upravljanja greskama</title>
</head>
<body>
<%
  // Izbacivanje izuzekta poziva stranicu greske
  int x = 1;
  if (x == 1)
  {
    throw new RuntimeException("Greska se pojavila!!!");
  }
%>
</body>
</html>
```

```
<%- -
  Document    : PrikaziGresku.jsp
  Created on  : 23.12.2016., 12.55.59
  Author      : Vladimir Milicevic
- -%>

<%@ page isErrorPage="true" %>
<html>
<head>
<title>Stranica greske</title>
</head>
<body>
<h1>Uuuuups...</h1>
<p>Izvinite, imamo gresku.</p>
<p>Stek izuzetka: </p>
<pre>
<% exception.printStackTrace(response.getWriter()); %>

</body>
</html>
```

PRIMER 19 - UPRAVLJANJE IZUZECIMA IZ JEDNE DATOTEKE

Izuzecima je moguće upravljati iz jedne datoteke.

Ukoliko se pokrene prethodni primer, on će simulirati grešku i pojaviće se informacija o greški, definisana na JSP stranici **PrikaziGresku.jsp**. Ako se pogleda listing ove datoteke moguće je primetiti instrukciju: `<%@ page isErrorPage="true" %>` kojom je ova stranica označena kao stranica grešaka.

Sledećom slikom je prikazana poruka o greški, angažovanjem JSP stranice **PrikaziGresku.jsp**.

```
java.lang.RuntimeException: Greska se pojavila!!! at org.apache.jsp.index_jsp._jspService(index_jsp.java:63) at org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:111) at
javax.servlet.http.HttpServlet.service(HttpServlet.java:790) at org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:411) at org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:473) at
org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:377) at javax.servlet.http.HttpServlet.service(HttpServlet.java:790) at org.apache.catalina.core.StandardWrapper.service(StandardWrapper.java:1682) at
org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:318) at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:160) at
org.apache.catalina.core.StandardPipeline.doInvoke(StandardPipeline.java:734) at org.apache.catalina.core.StandardPipeline.invoke(StandardPipeline.java:673) at com.sun.enterprise.web.WebPipeline.invoke(WebPipeline.java:99) at
org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:174) at org.apache.catalina.connector.CoyoteAdapter.doService(CoyoteAdapter.java:416) at
org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:283) at com.sun.enterprise.v3.services.impl.ContainerMapper$HttpHandlerCallable.call(ContainerMapper.java:459) at
com.sun.enterprise.v3.services.impl.ContainerMapper.service(ContainerMapper.java:167) at org.glassfish.grizzly.http.server.HttpHandler.runService(HttpHandler.java:206) at
org.glassfish.grizzly.http.server.HttpHandler.doHandle(HttpHandler.java:180) at org.glassfish.grizzly.http.server.HttpServerFilter.handleRead(HttpServerFilter.java:235) at
org.glassfish.grizzly.filterchain.ExecutorResolver$9.execute(ExecutorResolver.java:119) at org.glassfish.grizzly.filterchain.DefaultFilterChain.execute(DefaultFilterChain.java:283) at
org.glassfish.grizzly.filterchain.DefaultFilterChain.executeChainPart(DefaultFilterChain.java:200) at org.glassfish.grizzly.filterchain.DefaultFilterChain.execute(DefaultFilterChain.java:132) at
org.glassfish.grizzly.filterchain.DefaultFilterChain.process(DefaultFilterChain.java:111) at org.glassfish.grizzly.ProcessorExecutor.execute(ProcessorExecutor.java:77) at
org.glassfish.grizzly.nio.transport.TCPNIOTransport.fireIOEvent(TCPNIOTransport.java:536) at org.glassfish.grizzly.strategies.AbstractIOStrategy.fireIOEvent(AbstractIOStrategy.java:112) at
org.glassfish.grizzly.strategies.WorkerThreadIOStrategy.run(WorkerThreadIOStrategy.java:117) at org.glassfish.grizzly.strategies.WorkerThreadIOStrategy.access$100(WorkerThreadIOStrategy.java:56) at
org.glassfish.grizzly.strategies.WorkerThreadIOStrategy$WorkerThreadRunnable.run(WorkerThreadIOStrategy.java:137) at org.glassfish.grizzly.threadpool.AbstractThreadPool$Worker.doWork(AbstractThreadPool.java:591) at
org.glassfish.grizzly.threadpool.AbstractThreadPool$Worker.run(AbstractThreadPool.java:571) at java.lang.Thread.run(Thread.java:745)
```

Uuuuups...

Izvinite, imamo grešku.

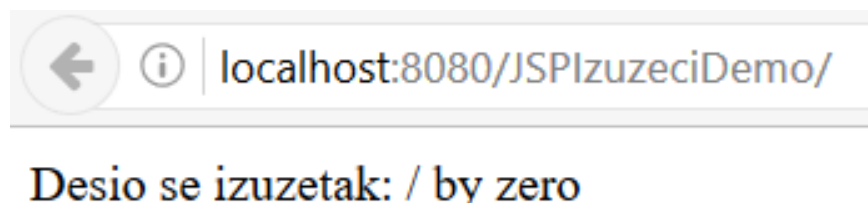
Stek izuzetka:

Slika 2.4.8 Stranica grešaka

Moguće je koristiti try - catch blok za upravljanje izuzecima u okviru jedne JSP stranice. Ako kod JSP stranice **indeks.jsp** bude zamenjen sledećim kodom:

```
<html>
<head>
  <title>Try...Catch Primer</title>
</head>
<body>
<%
  try{
    int i = 1;
    i = i / 0;
    out.println("Odgovor je " + i);
  }
  catch (Exception e){
    out.println("Desio se izuzetak: " + e.getMessage());
  }
%>
</body>
</html>
```

Pokretanjem i testiranjem programa dobija se izlaz prikazan sledećom slikom.



Slika 2.4.9 Testiranje izuzetka iz jedne JSP stranice.

ZADATAK 2

Kreirajte vlastitu JSP stranicu

Koristeći:

- obrađenu sintaksu;
- direktive;
- akcije;
- implicitne objekte;

samostalno kreirajte JSP stranicu sa sadržajem kojeg ćete sami izabrati.

▼ Poglavlje 3

Procesiranje forme

GET I POST METODA

Prezentovanje značaja i primene GET metode.

GET metoda šalje kodiranu korisničku informaciju zakačenu za zahtev stranice. Stanica i kodirana informacija razdvojene su specijalnim karakterom "?" na način koji može biti priložen u sledećem opštem obliku:

```
http://www.test.com/  
hello?key1=value1&key2=value2
```

Metoda GET je podrazumevana metoda za prosleđivanje informacija od veb pregledača ka veb serveru i kreira dugačak string koji se pojavljuje u pregledačevom tekst boksu lokacije. GET metodu nikada ne treba koristiti kada se prosleđuju informacije poput korisničkog imena i lozinke. Ova metoda poseduje jedno ograničenje - nije moguće primeniti više od 1024 karaktera u odgovarajućem GET stringu. Ova metoda je prosleđena pomoću **QUERY_STRING** zaglavlja i biće dostupne pomoću **QUERY_STRING** varijable okruženja kojom je moguće rukovati metodama **getQueryString()** i **getParameter()** objekta zahteva.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

U osnovi pouzdaniji pristup slanju podataka **backend** programu predstavlja metoda **POST**. Metoda pakuje informaciju na potpuno isti način kao i **GET** metoda, ali umesto upotrebe stringa nakon specijalnog znaka "?" u URL - u, informacija se šalje kao odvojena poruka. Poruka stiže do **backend** programa u obliku standardnog ulaza kojeg je moguće proslediti i koristiti za procesiranje.

JSP rukuje ovim tipom zahteva primenom metoda **getParameter()**, za čitanje prostih parametara, i **getInputStream()**, za čitanje binarnog toka podataka koji dolazi od klijenta.

PRIMER 20 - GET METODE PRIMENOM URL

Prezentovanje načina korišćenja i obrade podataka iz URL na formi..

JSP rukuje formom podataka automatskim parsiranjem, primenom neke od sledećih metoda, u zavisnosti od konkretne situacije:

- **getParameter()**: Vraća vrednost parametra forme;

- **getParameterValues()**: Ukoliko se parametar forme pojavljuje više puta, vraća listu njegovih vrednosti, na primer kao checkbox;
- **getParameterNames()**: Vraća kompletnu listu svih parametara za tekući zahtev;
- **getInputStream()**: Čita binarni tok podataka koji dolazi sa klijent strane.

Neka navedeno bude ilustrovano odgovarajućim primerom. U primeru će biti prosleđene dve informacije u formu **ObradaImena** primenom **GET** metode pomoću odgovarajućeg URL - a.

```
http://localhost:8080/ObradaImena/  
index.jsp?first_name=VLADIMIR&&&&&&&&&&last_name=MILI  
CEVIC
```

Neophodno je priložiti JSP stranicu, u ovom slučaju index.jsp za rukovanje ulazom prosleđenim od strane veb pregledača. Biće korišćena metoda `getParameter()` za pristup prosleđenoj informaciji.

Sljedećim listingom je prikazan kod stranice `index.jsp` koja implementira prethodno navedene funkcionalnosti:

```
<html>
  <head>
    <title>Primena GET metode za citanje podataka forme</title>
  </head>
  <body>
    <center> <h1>Primena GET metode za citanje podataka forme</h1> </center>
    <ul>
      <li><p><b>Ime:</b>
          <%= request.getParameter("first_name")%>
        </p></li>
      <li><p><b>Prezime:</b>
          <%= request.getParameter("last_name")%>
        </p></li>
    </ul>
  </body>
</html>
```

Ukoliko se izvrši testiranje kreiranog koda, primenom priloženog URL stringa, proširenog traženom informacijom, dobija se sledeći rezultat:



PRIMER 21 - GET METODE PRIMENOM FORME

Prezentovanje načina korišćenja i obrade podataka sa forme primenom JSP.

Sada će biti demonstriran obrnuti slučaj. Podaci se unose na formu i prosleđuju veb pregledačem kao **GET** zahtev. Forma se realizuje na sledeći način: i

- ime i prezime se unose u posebnim poljima za unos teksta,
- klikom na dugme "POSALJI" vrši se slanje podataka kao deo GET stringa.

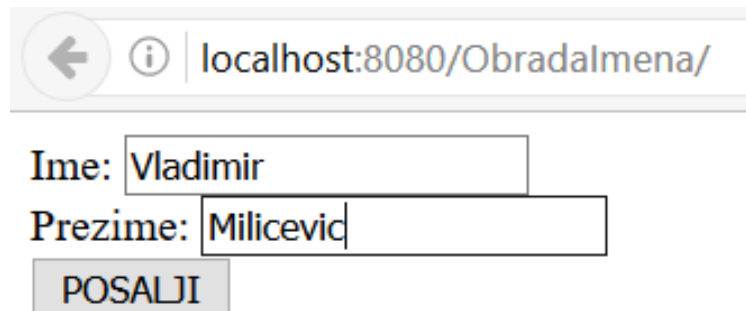
Sledećim listingom moguće je modifikovati kod prethodno definisani index.jsp stranice:

```
<html>
  <body>
    <form action="index.jsp" method="GET">
      Ime: <input type="text" name="first_name"> <br/>
      Prezime: <input type="text" name="last_name" /> <br/>
      <input type="submit" value="POSALJI" />
    </form>
  </body>
</html>
```

Prevođenjem i pokretanjem navedene stranice URL stringom:

```
http://localhost:8080/ObradaImena/
```

dobija se sledeći izlaz u pregledaču:



Slika 3.1.2 Forma za unos imena i prezimena

Klikom na dugme "POSALJI" kreira se zahtev prikazan sledećom slikom:



Slika 3.1.3 Zahtev kreiran iz forme

PRIMER 22 - POST METODE PRIMENOM FORME

Prezentovanje načina korišćenja i obrade podataka sa forme primenom JSP i POST metode.

U nastavku je neophodno pokazati kako se ponaša i koristi **POST** metoda za slanje podataka od pregledača ka serveru i kako se šalju i prikazuju odgovarajuće informacije primenom formi.

Sledećim listingom je prikazan kod index.jsp stranice:

```
<html>
<head>
<title>Primena GET i POST metoda za citanje podataka forme</title>
</head>
<body>
<center>
<h1>Primena GET metode za citanje podataka forme</h1>
</center>
<ul>
<li><p><b>Ime:</b>
<%= request.getParameter("first_name")%>
</p></li>
<li><p><b>Prezime:</b>
<%= request.getParameter("last_name")%>
</p></li>
</ul>
</body>
</html>
```

Implementacija POST metode biće realizovana forma.jsp stranicom koja kreira formu koja je identična prethodno prikazanoj. Sledi kod ove datoteke:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<body>
<form action="index.jsp" method="POST">
Ime: <input type="text" name="first_name">
<br/>
Preime: <input type="text" name="last_name" />
<br/>
<input type="submit" value="POSALJI" />
</form>
</body>
</html>
```

Unošenjem podataka u polja za ime i prezime, i klikom na "POSALJI" šalju se podaci i prikazuju na način koji je već prikazan prethodnim slikama.

PRIMER 23 - PROSLEĐIVANJE CHECK - BOX PODATAKA U JSP PROGRAM

Prezentovanje slanja izabranih stavki u JSP program.

U nastavku izlaganja moguće je pokazati kako se često korišćena Java kontrola **CheckBox** koristi za prosleđivanje informacija u JSP stranicama. Dobro je poznato da se ova kontrola koristi kada je neophodno izabrati jednu ili više opcija iz ponuđene liste opcija.

Sledećim listingom je priložen kod koji implementira formu sa **CheckBox** kontrolama. Kod odgovara datoteci: index.jsp.

```
<%--
    Document    : forma
    Created on   : 25.12.2016., 14.31.56
    Author       : Vladimir Milicevic
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<body>
<form action="index.jsp" method="POST" target="_blank">
<input type="checkbox" name="fit" checked="checked" /> FIT</br>
<input type="checkbox" name="fdu" /> FDU </br>
<input type="checkbox" name="fam" checked="checked" />
FAM</br>
<input type="submit" value="IZABERITE FAKULTET" />
</form>
</body>
</html>
```

U nastavku, za isti projekat, biće kreirana forma.jsp koja sa index.jsp stranice preuzima čekirane podatke i daje odgovarajući izveštaj.

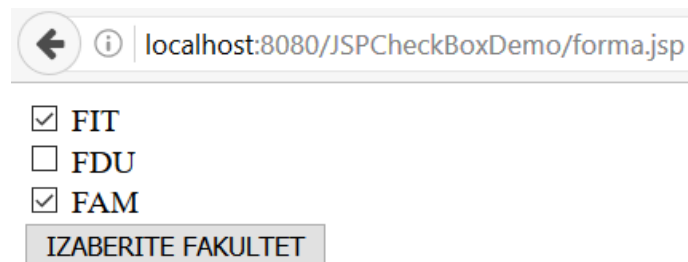
Sledećim listingom je dat kod navedene datoteke:

```
<%--
    Document    : forma
    Created on   : 25.12.2016., 14.31.56
    Author       : Vladimir Milicevic
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Citanje Checkbox podataka</title>
</head>
<body>
<center>
```



```
<h1>Citanje Checkbox podataka</h1>
</center>
<ul>
<li><p><b>FIT obalezen:</b>
<%= request.getParameter("fit")%>
</p></li>
<li><p><b>FDU obalezen:</b>
<%= request.getParameter("fdu")%>
</p></li>
<li><p><b>FAM obalezen:</b>
<%= request.getParameter("fam")%>
</p></li>
</ul>
</body>
</html>
```



localhost:8080/JSPCheckBoxDemo/forma.jsp

☒ FIT
☐ FDU
☒ FAM

IZABERITE FAKULTET

Slika 3.1.4 Izbor opcija



localhost:8080/JSPCheckBoxDemo/forma.jsp

Citanje Checkbox podataka

- FIT obalezen: on
- FDU obalezen: null
- FAM obalezen: on

Slika 3.1.5 Prosleđene informacije

PRIMER 24 - ČITANJE SVIH PARAMETARA FORME

Prethodno izlaganje se dodatno proširuje prmenom novih funkcionalnosti u jsp stranicama.

kao kraj ovog dela izlaganja, u vezi sa JSP stranicama, biće pokazana primena metode `getParameterNames()` objekta `HttpServletRequest` za iščitavanje svih dostupnih parametara forme. Metoda je povratnog tipa **Enumeration** i vraća nazive parametara po neodređenom rasporedu.

Kada se dobiju ovi parametri, moguće je koristiti petlje za prolazak i pretragu ovog niza primenjujući metode `hasMoreElements()`, za određivanje kada će se petlja zaustaviti, i `nextElement()` za uzimanje svakog pojedinačnog elementa iz niza **Enumeration**.

Neka je nepromenjena JSP stranica `index.jsp` iz prethodnog primera, a stranicu `forma.jsp`, koja čita sve obeležene podatke sa forme definisane `index.jsp`, stranicom moguće je modifikovati na sledeći način:

```
<%@ page import="java.io.*,java.util.*" %> <html>
<head>
    <title>HTTP zaglavlje zahteva - primer</title>
</head>
<body>
<center> <h2>HTTP zaglavlje zahteva - primer</h2>
    <table width="100%" border="1" align="center">
        <tr bgcolor="#949494">
            <th>Naziv parametra</th>
            <th>Vrednosti</th>
        </tr>
        <% Enumeration paramNames = request.getParameterNames();
            while (paramNames.hasMoreElements()) {
                String paramName = (String) paramNames.nextElement();
                out.print("<tr><td>" + paramName + "</td>\n");
                String paramValue = request.getParameter(paramName);
                out.println("<td> " + paramValue + "</td></tr>\n");
            }%>
    </table>
</center>
</body>
</html>
```

Primer se ponovo prevodi i učitava se forma sa stranice `index.jsp` (pogledati sliku 4). Neka su čekirana dva polja na potpuno isti način kao što je prikazan na slici. To znači da su izabrani fakulteti: FIT i FAM. Klikom na dugme "IZABERITE FAKULTET" izvršava se POST zahtev i izabrani podaci se prosleđuju i kroz iteracije JSP stranice `forma.jsp` obrađuju i prikazuju korisniku. Navedeno je realizovano na način prikazan sledećom slikom:



Naziv parametra	Vrednosti
fit	on
fam	on

Slika 3.1.6 Čitanje svih izabranih parametara forme

Pokazni primer koji je pratio izlaganje u ovom delu lekcije, dostupan Vam je za preuzimanje odmah nakon ovog objekta učenja u aktivnosti Shared Resources.

▼ 3.1 Procesiranje forme - kompletan primer

ZADATAK 3 - ZAHTEVI ZADATKA

Kreiranje jednostavne veb aplikacije sa JSP stranicama.

1. Otvoriti NetBeans razvojno okruženje i kreirati projekat CheckBoxDemo.
2. Projekat sadrži dve JSP stranice;
3. Na početnoj stranici se nalazi forma pomoću koje, koristeći CheckBox kontrole birate jedan od tri fakulteta Univerziteta Metropolitan. Stranica se naziva index.jsp;
4. Druga stranica, forma.jsp, preuzima rezultat izbora sa prve stranice, prilaže odgovarajući izveštaj o izvršenom izboru zajedno a linkom za posetu prezentaciji odgovarajućeg fakulteta;

Sledećim listingom je priložen JSP kod koji odgovara stranici index.jsp.

JSP STRANICA INDEX.JSP

Kreira se stranica koja sadrži formu koju je neophodno popuniti.

Za početak će biti kreirana početna stranica ove jednostavne Java veb aplikacije.

Stranica, pod nazivom *index.jsp*, koristi **POST** metodu za slanje podataka sa vlastite forme na stranicu *forma.jsp* na kojoj će biti obrađeni (linija koda 3). Na formi, primenom **CheckBox** kontrola, je omogućen izbor opcija (linije 4-7). Forma sadrži, konačno, i dugme "*submit*" kojim se potvrđuje izbor vrednosti sa forme (linija koda 8) i podaci se prosleđuju stranici *forma.jsp* gde će konačno biti obrađeni na odgovarajući način.

```
<html>
<body>
<form action="forma.jsp" method="POST" target="_blank">
<input type="checkbox" name="fit" checked="checked" /> FIT</br>
<input type="checkbox" name="fdu" /> FDU </br>
<input type="checkbox" name="fam" checked="checked" />
FAM</br>
<input type="submit" value="IZABERITE FAKULTET" />
</form>
</body>
</html>
```

JSP STRANICA FORMA.JSP

Keira se stranica za obradu forme.

Sledećim listingom je priložen kod koji odgovara JSP stranici forma.jsp.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Citanje Checkbox podataka</title>
</head>
<body>
<center>
<h1>Citanje Checkbox podataka</h1>
</center>
<ul>
<li><p><b>FIT oblelezen:</b>
<%= request.getParameter("fit")%>
<a href="http://www.metropolitan.ac.rs/osnovne-studije/
fakultet-informacionih-tehnologija/">Posetite FIT</a>
</p></li>
<li><p><b>FDU obelezen:</b>
<%= request.getParameter("fdu")%>
<a href="http://www.metropolitan.ac.rs/fakultet-digitalnih-umetnosti-2/">Posetite
FDU</a>
</p></li>
<li><p><b>FAM oblezen:</b>
<%= request.getParameter("fam")%>
<a href="http://www.metropolitan.ac.rs/osnovne-studije/
fakultet-za-menadzment/">Posetite FAM</a>
</p></li>
</ul>
</body>
</html>
```

Sledi objašnjenje funkcionalnosti ugrađenog koda u stranicu *forma.jsp*. Stranica, na prethodno opisani način, preuzima podatke koji su izabrani na formi stranice *index.jsp*. Svi podaci prosleđeni sa prethodne forme, prikazuju se sa linkom na veb stranicu, u konkretnom slučaju na stranice izabranih fakulteta Univerziteta Metropolitan (linije koda 12-22).

Klikom na link, aplikacija vrši preusmeravanje sa stranice *forma.jsp* na veb stranicu definisanu izabranim linkom.

Kompletno urađen i proveren zadatak je priložen u dodatnim materijalima kao aktivnost Shared Resources.

ZADATAK 4

Probajte sami

Pokušajte da u prethodni zadatak ugradite sledeću modifikaciju:

1. Kreiraćete novu stranicu koja će vršiti prikazivanje informacije o kliku na link iz tačke 4;

2. Kada se klikne na odgovarajući link, na ovoj stranici se prikazuje logo Univerziteta sa porukom "Vi ste student (zavisi od izbora linka) fakuleta";
3. Nakon par sekundi vrši se preusmeravanje na odgovarajuću stranicu za prezentaciju fakulteta.

▼ Poglavlje 4

Domaći zadatak 2

ZADATAK 1

Zadatak za rad kod kuće

Zadatak urađen na vežbama modifikovati na sledeći način:

1. Prikazuju se podaci samo za one fakultete koji su izabrani pomoću odgovarajuće CheckBox kontrole;
2. Na formu dodati polje za unos teksta. U polje se unosi ime koje će posle služiti kao dodatna informacija na JSP stranici forma.jsp;
3. Imenu se, na JSP stranici forma.jsp želi dobrodošlica sa informacijom "izabrali ste da se upoznate sa sledećim fakultetima: ".

▼ Poglavlje 5

Zaključak

ZAKLJUČAK

Lekcija je obradila osnovne JSP koncepte.

Lekcijom su dati analiza i demonstracija osnovnih JSP koncepata.

Izlaganje započinje pregledom JSP arhitekture sa akcentom na demonstriranje JSP kontejnera, kao mehanizma za upravljanje JSP stranicama, i principe i načine procesiranja JSP stranica.

Nakon toga, obrađivan je životni ciklus JSP stranica sa posebnom pažnjom na sve faze kroz koje prolazi JSP stranica tokom vlastitog životnog veka.

Najveći deo lekcije posvećuje pažnji sintaksi JSP stranica gde je do veoma sitnih detalja, uz navođenje adekvatnih primera, približen način primene naredbi i elemenata programskog jezika Java u JSP stranicama.

Lekcija završava izlaganje predstavljanjem GET i POST metoda u klijentskim zahtevima tokom procesiranja formi.

Savladavanjem ove lekcije student je upoznat sa JSP osnovama.

LITERATURA

Za pripremu Lekcije 3 korišćena je najsavremenija literatura.

1. <http://www.tutorialspoint.com/jsp/>
2. Eric Jendrock, Ricardo Cervera-Navarro, Ian Evans, Kim Haase, William Markito. 2014. Java Platform, Enterprise Edition The Java EE Tutorial, Release 7, ORACLE
3. David R. Heffelfinger. 2015. Java EE7 Development With NetBeans 8, PACK Publishing
4. Yakov Fain. 2015. Java 8 programiranje, Kombib (Wiley)
5. Josh JUNEau. 2015. Java EE7 Recipes, Apress