



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 2

Lekcija 03

PRIRUČNIK ZA STUDENTE

# KI204 - JAVA 7: JAVA ENTERPRISE EDITION

## Lekcija 03

### *JAVA SERVER PAGES 2*

- ✓ Java Server Pages 2
- ✓ Poglavlje 1: Primena JSP filtera
- ✓ Poglavlje 2: Upravljanje kolačićima u JSP stranicama
- ✓ Poglavlje 3: Otpremanje datoteka u JSP
- ✓ Poglavlje 4: Rukovanje datumom u JSP stranicama
- ✓ Poglavlje 5: Preusmeravanje JSP stranica
- ✓ Poglavlje 6: JSTL - JavaServer Pages Standard Tag Library
- ✓ Poglavlje 7: JSP - JavaBean
- ✓ Poglavlje 8: JSP - jezik izraza
- ✓ Poglavlje 9: JSP - internacionalizacija
- ✓ Poglavlje 10: Domaći zadatak 2-1
- ✓ 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

*Ovovom lekcijom se obrađuje primena naprednijih koncepata u kreiranju JSP stranica.*

Ovom lekcijom biće obrađeni napredni JSP koncepti kao nastavak na izlaganje iz prethodne lekcije koje je postavilo temelje za korišćenje JSP stranica u Java veb aplikacijama.

Posebno, lekcija će se baviti sledećim temama:

- Demonstracija korišćenja filtera u JSP stranicama;
- Detaljna analiza i demonstracija upravljanja kolačićima u JSP stranicama;
- Savladavanje **upload** - ovanja datoteka iz JSP stranica;
- Rad sa vremenom i datumom u JSP stranicama;
- Preusmeravanje sa jedna na drugu JSP stranicu;
- Detaljan uvid u JSTL (JavaServer Pages Standard Tag Library) biblioteku;
- Rad sa bazama podataka iz JSP stranica;
- Upoznavanje i rad sa JavaBean objektima;
- Savladavanje elemenata JSP EL jezika izraza;
- Internacionalizacija u JSP stranicama.

Savladavanjem ove lekcije student u potpunosti će ovladati konceptima i principima primene JSP stranica u Java veb aplikacijama.

## ▼ Poglavlje 1

# Primena JSP filtera

## JSP FILTERI

*Postoje različiti tipovi JSP filtera.*

Servlet i JSP filter su JAVA klase koje se koriste u veb i JSP programiranju na brojne načine i u različite svrhe:

- Presretanje zahteva klijenata pre nego što pristupe resursima servera;
- Manipulisanje odgovorima servera pre nego što se proslede klijentu;

Postoje različiti tipovi JSP filtera:

- filteri autentifikacije;
- filteri kompresovanja podataka;
- filteri enkripcije;
- filteri okidači za pristupanje događajima;
- filteri konverzije slike;
- filteri logovanja i provere;
- filteri lanca MIME-TYPE;
- filteri tokenizacije;
- XSL/T filteri za transformisanje XML sadržaja.

Filteri su angažovani u `web.xml` datoteci, koja je poznata kao deskriptor veb razvoja, i nakon toga se povezuje sa servletom ili JSP stranicom ili URL šablonom u deskriptoru.

Kada JSP kontejner pokrene kreiranu veb aplikaciju, kreira se instanca za svaki filter koji je deklarisan u veb deskriptoru razvoja. Filteri se izvršavaju po redosledu kojim su deklarisanim u `web.xml` datoteci.

Važno je još napomenuti da je filter Java klasa koja implementira interfejs `javax.servlet.Filter` od kojeg dobija sledeće metode:

- `public void doFilter (ServletRequest, ServletResponse, FilterChain)` - metoda se poziva od strane JSP kontejnera svaki put kada se par zahtev / odgovor prosledi kroz lanac filtera;
- `public void init(FilterConfig filterConfig)` - metodu poziva veb kontejner da ukaže na filter koji će biti aktiviran;
- `public void destroy()` - metodu poziva veb kontejner da ukaže na filter koji će biti isključen.

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

## PRIMER 1 - JSP FILTERI

*Uvodi se jedan primer primene filtera u veb aplikacijama.*

@WebFilter je anotacija kojom se smanjuje značajno količina podešavanja u web.xml datoteci. Anotacija ne može da funkcioniše bez deskriptora budući da njom nije moguće definisati redosled filtera.

@WebFilter može da uzme u razmatranje neke od sledećih vrednosti:

- **asyncSupported** - podržava asinhronne operacije u filteru;
- **dispatcherTypes** - pridružuje dispečera filteru;
- **filterName** - definiše naziv filtera;
- **initParams** - inicira parametre filteru;
- **servletNames**: definiše servlet na kojeg se filter primenjuje;
- **urlPatterns** - definiše URL šablon.

U primeru će biti kreirana dva filtera i servlet. Redosled filtera će biti definisan u datoteci web.xml. Sledi kod datoteke **web** deskriptora.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/
j2ee/web-app_2_4.xsd"
  version="3.0">
  <display-name>Filter Demo</display-name>
  <filter-mapping>
    <filter-name>filterJedan</filter-name>
    <url-pattern>/msg/*</url-pattern>
  </filter-mapping>
  <filter-mapping>
    <filter-name>filterDva</filter-name>
    <url-pattern>/msg/*</url-pattern>
  </filter-mapping>
</web-app>
```

Kreira se klasa koja ukazuje na prvi filter. Klasa se naziva FilterJedan.java.

```
package com.filter;

/**
 *
 * @author Vladimir Milicevic
 */
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
```

```
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
@WebFilter(filterName="filterjedan")
public class FilterJedan implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
    }
    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {
        System.out.println("Unutar filtera broj 1.");
        chain.doFilter(request, response);
    }
    @Override
    public void destroy() {
    }
}
```

## JSP FILTERI - NASTAVAK PRIMERA

*Nastavlja se kreiranje druge klase filtera i servleta.*

U nastavku se kreira druga klasa filtera deklarisanog o veb deskriptoru **web.xml**. Klasa se naziva FilterDva.java.

```
package com.filter;

/**
 *
 * @author Vladimir Milicevic
 */
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
@WebFilter(filterName="filterDva")
public class FilterDva implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
    }
    @Override
    public void doFilter(ServletRequest request, ServletResponse response,
        FilterChain chain) throws IOException, ServletException {
        System.out.println("Unutar filtera broj dva.");
    }
}
```

```
        chain.doFilter(request, response);
    }
    @Override
    public void destroy() {
    }
}
```

Na kraju se kreira servlet klasa PozdravServlet.java.

```
package com.filter;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(urlPatterns = "/msg/PozdravServlet", loadOnStartup = 1)
public class PozdravServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException{
        doGet(request, response);
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException{
        response.setContentType("text/html");
        System.out.println("Unutar servleta.");
        PrintWriter out = response.getWriter();
        out.println("Veliki pozdrav!");
    }
}
```

Unutar filtera broj jedan.

Unutar filtera broj dva.

Unutar servleta.

Slika 1.1 Redosled filtriranja

## ZADATAK 1

### *Samostalno kreiranje JSP filtera*

Po uzoru na prethodni primer, samostalno kreirajte i implementirajte JSP filter.



## ▼ Poglavlje 2

# Upravljanje kolačićima u JSP stranicama

## KOLAČIĆI (COOKIES)

*JSP jasno podržava HTTP kolačiće primenom servlet tehnologije.*

Kolačići (Cookies) predstavljaju tekstualne datoteke koje su snimljene na klijent računaru i čuvaju različite informacije značajne za praćenje. JSP jasno podržava HTTP kolačiće primenom servlet tehnologije.

Tri koraka su od posebnog značaja kada se koriste kolačići za identifikovanje korisnika:

- Server skript šalje skup kolačića ka veb pregledaču. Na primer: ime, godine, identifikator i tako dalje;
- Veb pregledač snima kolačiće na lokalnom računaru za buduću upotrebu;
- Sledeći put kada pregledač pošalje bilo kakav zahtev ka veb serveru, on šalje i podatke kolačića serveru. Server može da iskoristi ove podatke za identifikovanje korisnika ili u neku drugu svrhu.

U narednom izlaganju je cilj da se pokaže kako se kolačići postavljaju i restartuju, kako im se pristupa i kako mogu da se obrišu u JSP programu.

Kolačići se obično postavljaju u HTTP zaglavlju. JSP stranica koja postavlja kolačić može da pošalje, na primer informaciju koja je sledećeg oblika:

```
HTTP/1.1 200 OK
Date: Fri, 23 Dec 2016 21:03:38 GMT
Server: Oracle GlassFish Server 4.1.1
Set-Cookie: name=xyz;
expires=Friday, 30-Dec-16 22:03:38 GMT;
path=/; domain=tutorialspoint.com
Connection: close Content-Type: text/html
```

Kao što je moguće primetiti zaglavlje **Set-Cookie** sadrži par name / value, vreme, putanju i domen. Nazivi vrednost će biti URL kodirani. Takođe, poljem **expires** kolačić je podešen da bude zaboravljen nakon isteka deklarisanog vremena. Ukoliko korisnik uputi veb pregledač na bilo koju stranicu koja se tiče putanje i domena kolačića, kolačić se upućuje nazad ka serveru. Zaglavlje pregledača može da ima sledeći oblik:

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/50.1.0
```

```
Host: neki.host.co.rs:1126
Accept: image/gif, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: name=xyz
```

Sada će JSP skript imati pristup kolačićima preko metode zahteva `request.getCookies()` koja vraća niz **Cookie** objekata.

## LISTA COOKIE METODA

*Sledi lista metoda koje su korisne u radu sa kolačićima.*

U narednom izlaganju će biti priložena lista korisnih metoda koje se koriste sa objektima kolačića i mogu biti upotrebljene u JSP stranicama za manipulisanje kolačićima:

- `public void setDomain(String pattern)` - Podešava domen na koji se kolačić primenjuje;
- `public String getDomain()` - Vraća domen na koji je kolačić primenjen;
- `public void setMaxAge(int expiry)` - Podešava vreme u sekundama pre do isteka kolačića. Ako se ne podesi, kolačić traje do isteka sesije;
- `public int getMaxAge()` - Vraća najveće vreme u sekundama do isteka kolačića. Vrednost -1 znači da će kolačić postojati do gašenja veb pregledača;
- `public String getName()` - Vraća naziv kolačića;
- `public void setValue(String newValue)` - Podešava vrednost pridruženu kolačiću;
- `public String getValue()` - Vraća vrednost pridruženu kolačiću;
- `public void setPath(String uri)` - Podešava putanju na kojoj se kolačić primenjuje. Ukoliko se putanja ne specificira, kolačić se vraća za sve URL - e iz istog direktorijuma kao tekuća stranica, računajući i poddirektorijume;
- `public String getPath()` - Vraća putanju na kojoj se kolačić primenjuje;
- `public void setSecure(boolean flag)` - Podešava logičku vrednost koja ukazuje da li bi kolačić trebalo da bude poslat isključivo preko kodiranih konekcija (na primer SSL);
- `public void setComment(String purpose)` - Podešava komentar koji govori o svrsi kolačića;
- `public String getComment()` - Vraća komentar koji govori o svrsi kolačića;

U daljem izlaganju će biti demonstrirana manipulacija kolačićima u programima koji koriste JSP stranice.

## UPOTREBA JSP ZA PODEŠAVANJE KOLAČIĆA

*U nastavku se pokazuje kako se postavljaju kolačići primenom JSP.*

Pre nego što se ilustruje primena kolačića na konkretnom primeru, neophodno je, u ovom delu lekcije, pozabaviti se sa procesom podešavanja kolačića.

Podešavanje kolačića je jednostavan proces i neophodno je da ispoštuje sledeća tri koraka:

- **Kreiranje Cookie objekta** - Poziva se konstruktor sa nazivom i vrednošću kolačića. Oba argumenta su tipa **String**. Navedeno je moguće realizovati na sledeći način:

```
Cookie cookie = new Cookie("ključ","vrednost");
```

Trebalo bi imati na umu da ni ključ ni vrednost ne mogu da sadrže znake priložene sledećom listom: [ ] ( ) = , " / ? @ : ;

- **Podešavanje maksimalnog vremena validnosti kolačića** - U ovom koraku se podešava vrednost u sekundama koja ukazuje na maksimalni period validnosti kolačića. Na primer, na sledeći način se podešava da kolačić bude validan 24 sata:

```
cookie.setMaxAge(60*60*24);
```

- **Slanje kolačića u zaglavlje HTTP odgovora** - Kolačić se šalje u zaglavlje HTTP odgovora na način priložen sledećom instrukcijom:

```
response.addCookie(cookie);
```

Konačno, u nastavku lekcije moguće je uvesti odovarajući primer koji ilustruje primenu kolačića u veb aplikacijama koje koriste JSP stranice.

## PRIMER 2 - PODEŠAVANJE KOLAČIĆA U JSP

*Sledećim primerom je demonstrirano podešavanje kolačića u JSP stranicama.*

Sledećim primerom je demonstrirano podešavanje kolačića u JSP stranicama. Moguće je podesiti dve JSP stranice. Stranicom **index.jsp** data je forma u okviru koje se unose ime i prezime koje će dobiti vlastite kolačiće. Sledećim listingom je priložena navedena datoteka:

```
<%- -
    Document    : index
    Created on  : 27.12.2016., 14.06.08
    Author      : Vladimir Milicevic
- -%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<body>
    <%- -
<form action="forma.jsp" method="POST">
    - -%>
        <form action="forma.jsp" method="GET">
Ime: <input type="text" name="ime">
<br/>
```

```
Preime: <input type="text" name="prezime" />
<br/>
<input type="submit" value="POSALJI" />
</form>
</body>
</html>
```

Forma sadrži dva tekst polja i dugme "POSALJI". Nakon klika na dugme vrši se kreiranje kolačića koji će čuvati ime i prezime.

Nakon kreiranja objekata vrši se njihovo dodavanje u response objekat. Kreiranje kolačića i njihovo dodavanje u zaglavlje odgovora realizovano je JSP stranicom **forma.jsp** čiji listing sledi:

```
<%--
    Document    : forma
    Created on  : 27.12.2016., 13.26.59
    Author      : Vladimir Milicevic
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<% // Kreira kolacice za ime i prezime.
    Cookie ime = new Cookie("ime", request.getParameter("ime"));
    Cookie prezime = new Cookie("prezime", request.getParameter("prezime"));

// Podesava se isticanje kolacica za 24 sata.
ime.setMaxAge(60*60*24);
prezime.setMaxAge(60*60*24);

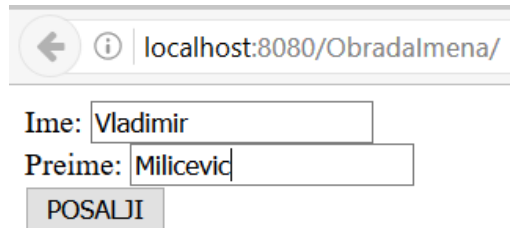
// Dodavanje oba kolacica u zaglavlje odgovora.
response.addCookie( ime );
response.addCookie( prezime ); %>

<html>
    <head>
        <title>Podesavanje kolacica</title>
    </head>
    <body> <center> <h1>Podesavanje kolacica</h1> </center>
    <ul>
        <li><p><b>Ime:</b>
            <%= request.getParameter("ime")%>
        </p></li>
        <li><p><b>Prezime:</b>
            <%= request.getParameter("prezime")%>
        </p></li>
    </ul>
</body>
</html>
```

## PRIMER 3 - ČITANJE KOLAČIĆA U JSP

*Pokazuje se kako se dobijaju podaci o sačuvanim kolačićima u JSP stranicama.*

U prethodnom izlaganju je pokazano kako se postavljaju i čuvaju **Cookie** objekti. Izvršavanjem prethodnog primera prvo se javlja sledeća forma:



A screenshot of a web browser showing a form at the URL localhost:8080/Obradalmena/. The form contains two input fields: 'Ime:' with the value 'Vladimir' and 'Preime:' with the value 'Milicevic'. Below the fields is a button labeled 'POSALJI'.

Slika 2.1 Forma za unos vrednosti za kolačiće

Klikom na "POSALJI" prosleđuju se vrednosti o objekte kolačića, koji će čuvati ove vrednosti. Sačuvani podaci mogu se videti na sledećoj slici:



Slika 2.2 Sačuvane vrednosti za kolačiće

U nastavku je neophodno pokazati kako je moguće dobiti podatke o sačuvanim kolačićima u JSP stranicama na malo drugačiji način.

U istom projektu moguće je kreirati novu datoteku, čiji je naziv **citajKolacic.jsp**, koja će pročitati kolačiće i prikazati odgovarajući izveštaj o rezultatima ovog procesiranja. Sledećim listingom je priložen kod ove JSP datoteke:

```
<%--
    Document    : citajKolacic
    Created on  : 27.12.2016., 14.08.10
    Author      : Vladimir Milicevic
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<title>Citanje kolacica</title>
</head>
<body>
<center><h1>Citanje kolacica</h1></center>
<%
Cookie cookie = null;
```

```
Cookie[] cookies = null;  
// Vraca niz kolacica pridruzenih domenu  
cookies = request.getCookies();  
if( cookies != null ){  
    out.println("<h2> Pronadjeni nazivi i vrednosti kolacica</h2>");  
    for (int i = 0; i < cookies.length; i++){  
        cookie = cookies[i]; out.print("Naziv : " + cookie.getName( ) + ", ");  
        out.print("Vrednost: " + cookie.getValue( )+" <br/>"); } }  
else{  
    out.println("<h2>Nisu pronadjeni kolacici!!!</h2>"); } %>  
</body>  
</html>
```

Rezultati, dostupni na linku <http://localhost:8080/Obradalmena/citajKolacic.jsp>, prikazani su sledećom slikom:



Slika 2.3 Podaci o pročitanim kolačićima

## PRIMER 4 - BRISANJE KOLAČIĆA U JSP

*Sledeći vid manipulacije kolačićima je brisanje.*

Brisanje kolačića u JSP - u je veoma jednostavno. Za realizovanje ovog procesa, neophodno je ispoštovati sledeća tri koraka:

- Čitanje postojećeg kolačića i njegovo čuvanje u Cookie objektu;
- Podešavanje "starosti" kolačića na nulu pozivom metode `setMaxAge()`;
- Vraćanje konkretnog kolačića u zaglavlje odgovora.

Da bi navedeno bilo realizovano, moguće je kreirati novu JSP stranicu `brisiKolacice.jsp` kojom će biti realizovano navedeno.

Pokretanjem ove stranice, dostupne na linku <http://localhost:8080/Obradalmena/brisiKolacic.jsp>, moguće je videti sledeći izveštaj:



Slika 2.4 Prikaz obrisanog kolačića

Sledi listing datoteke `brisiKolacice.jsp` koja je obezbedila navedenu funkcionalnost:

```
<%--
    Document    : brisiKolacic
    Created on   : 27.12.2016., 14.25.54
    Author      : Vladimir Milicevic
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <title>Citanje i brisanje kolacica</title>
    </head> <body> <center>
        <h1>Citanje i brisanje kolacica</h1> </center>
        <% Cookie cookie = null;
           Cookie[] cookies = null;
// Vracanje niza kolacica vezanih za domen
cookies = request.getCookies();
if( cookies != null ){
    out.println("<h2> Pronadjeni nazivi i vrednosti kolacica</h2>");
    for (int i = 0; i < cookies.length; i++){
        cookie = cookies[i];
        if((cookie.getName( )).compareTo("ime") == 0 ){
            cookie.setMaxAge(0); response.addCookie(cookie);
            out.print("Obrisan kolacic: " + cookie.getName( ) + "<br/>");
        }
        out.print("Naziv : " + cookie.getName( ) + ", ");
        out.print("Vrednost : " + cookie.getValue( )+" <br/>");
    }
}
else{ out.println( "<h2>Nisu pronadjeni kolacici!!!</h2>");
} %>
</body>
</html>
```

## ZADATAK 2

### *Samostalan rad sa kolačićima*

Samostalno implementirajte sledeće:

- kreirajte i podesite kolačić;
- preuzmite ga i pročitajte;
- obrišite ga.

## ▼ Poglavlje 3

# Otpremanje datoteka u JSP

## SLANJE DATOTEKA NA SERVER - UVOD I FORMA IZBORA DATOTEKE

*U JSP stranicama je moguće realizovati otpremanje datoteka ka serveru.*

Ubacivanjem Java koda u standardni HTML kod, moguće je postići brojne zadatke. Jedan od njih može da bude slanje datoteka ka serveru. Datoteke mogu biti tekstualne, binarne, slike ili bilo koje druge.

Ovu problematiku je najlakše i najefikasnije obraditi na primeru pa će ovo izlaganje i teći u tom pravcu. Za početak će biti kreirana forma čiji je zadatak da omogući korisniku izbor datoteke i njeno slanje na server. Sledeće teze su veoma važne za realizovanje JSP datoteke kojom će navedena forma biti kreirana i prikazana:

- Atribut metode forme mora biti podešen na POST, metoda GET ne može da bude upotrebljena;
- Atribut forme **enctype** trebalo bi da bude podešen na **multipart/form-data**;
- Atribut forme **action** bi trebalo da ukazuje na servlet ili JSP stranicu, u ovom konkretnom slučaju, koja će rukovati slanjem datoteke na server.

Forma će biti realizovana kreiranjem JSP datoteke **index.jsp**, a rukovanje slanjem datoteke ka serveru biće povereno JSP stranici sa nazivom **upload.jsp**.

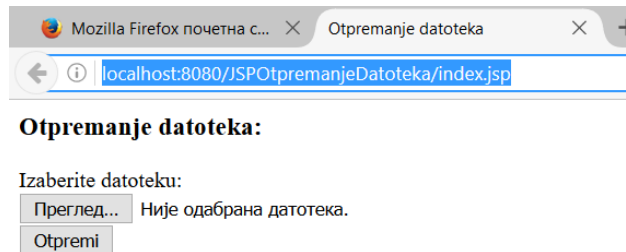
Sledi listing i slika koja reprezentuje ovu formu, respektivno.

```
<% -  
    Document    : index  
    Created on  : 28.12.2016., 11.30.21  
    Author      : Vladimir Milicevic  
- %>  
  
<html>  
    <head>  
        <title>Otpremanje datoteka</title>  
    </head>  
    <body> <h3>Otpremanje datoteka:</h3> Izaberite datoteku: <br/>  
        <form action="upload.jsp" method="post" enctype="multipart/form-data">  
            <input type="file" name="file" size="50" /> <br/>  
            <input type="submit" value="Otpremi" />  
        </form>
```



```
</body>  
</html>
```

Pozivom linka `http://localhost:8080/JSPotprenanjeDatoteka/index.jsp`, otvara se sledeća JSP stanica sa odgovarajućom formom:



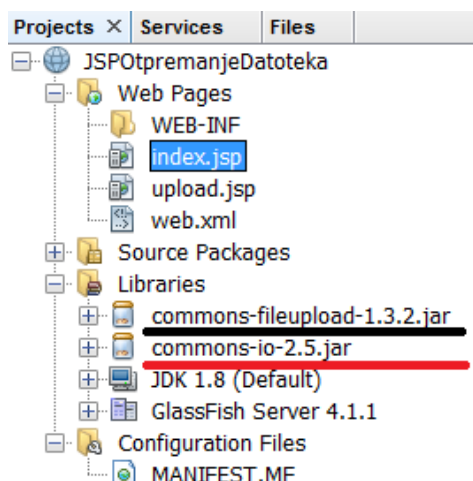
Slika 3.1 Forma za izbor datoteke

## JSP DATOTEKA ZA UPRAVLJANJE OTPREMANJEM DATOTEKA

*Posebna pažnja je na konstrukciji JSP datoteke kojom se rukuje slanjem datoteka ka serveru.*

U nastavku je moguće kreirati JSP datoteku, neka joj je naziv `upload.jsp` koja u jednom trenutku može da pošalje više datoteka ka serveru. Pre isticanja koda ove datoteke, odgovarajućim listingom, trebalo bi imati na umu nekoliko detalja:

- Biće korišćeno nekoliko klasa koje nisu direktno dostupne, otuda ih je neophodno dodati odgovarajućim bibliotekama u okviru foldera Libraries kreiranog veb projekta. Ove datoteke su prikazane sledećom slikom:



Slika 3.2 Apache commons biblioteke dodate u projekat

- Trebalo bi preuzeti najnovije verzije ovih biblioteka;
- Moguće je definisati lokacije gde će datoteke biti otpremljene, kao i ograničenje u veličini fajla koji može biti otpremljen;

- Ako se otpremanje vrši na lokalnom računaru, kreirati unapred foldere u koje će server proslediti otpremljene datoteke.

Sledi listing JSP stranice `upload.jsp` koja ima funkcionalnosti kreirane na osnovu prethodnih instrukcija. Instrukcije **import** omogućene su uvozom navedenih biblioteka.

```
<%--
    Document      : Upload
    Created on    : 28.12.2016., 10.08.47
    Author       : Vladimir Milicevic
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%@ page import="java.io.*,java.util.*, javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
<%@ page import="org.apache.commons.fileupload.*" %>
<%@ page import="org.apache.commons.fileupload.disk.*" %>
<%@ page import="org.apache.commons.fileupload.servlet.*" %>
<%@ page import="org.apache.commons.io.output.*" %>

<%
File file ;

int maxFileSize = 5000 * 1024;
int maxMemSize = 5000 * 1024;
// context = pageContext.getServletContext();
String filePath = "c:\\priv\\";
//String filePath = context.getInitParameter("file-upload");

// Proverava tip sadrzaja
String contentType = request.getContentType();

if ((contentType.indexOf("multipart/form-data") >= 0)) {

    DiskFileItemFactory factory = new DiskFileItemFactory();
    // Maksimalna velicina koja moze biti acuvana u memoriji
    factory.setSizeThreshold(maxMemSize);
    // Lokacija na kojoj se cuvaju podaci veci od maksimalno dozvoljene velicine
    factory.setRepository(new File("c:\\temp"));
    // Kreiranje novog rukovaoca za otpremanje datoteka
    ServletFileUpload upload = new ServletFileUpload(factory);
    // maksimalna velicina za otpremanje.
    upload.setSizeMax( maxFileSize );
    try{

        // Prosledjuje zahtev za uzimanje stavki datoteke
        List fileItems = upload.parseRequest(request);
        // Proces otpremanja
        Iterator i = fileItems.iterator();
        out.println("<html>");
    }
}
```

```

        out.println("<head>");
        out.println("<title>JSP otpremanje datoteka</title>");
        out.println("</head>");
        out.println("<body>");
        while ( i.hasNext () ) {
            FileItem fi = (FileItem)i.next();

            if ( !fi.isFormField () ) {

                // Preuzimanje parametara otpremljene datoteke
                String fieldName = fi.getFieldName();
                String fileName = fi.getName();
                boolean isInMemory = fi.isInMemory();
                long sizeInBytes = fi.getSize();

                // Pisanje datoteke

                if ( fileName.lastIndexOf("\\") >= 0 ){
                    file = new File( filePath + fileName.substring(
fileName.lastIndexOf("\\")) ) ;
                }
                else{ file = new File( filePath +
fileName.substring(fileName.lastIndexOf("\\")+1))
;
                }
                fi.write( file ) ;
                out.println("Otpremljena datoteka: " + filePath + fileName +
"<br>");

            }

        }

        out.println("</body>");
        out.println("</html>");
    }
    catch(Exception ex)
        { System.out.println(ex); }
    } else{
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet upload</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<p>Datoteka nije otpremljena</p>");
        out.println("</body>");
        out.println("</html>");
    }
}

```

%&gt;

## UPOTREBA DESKRIPTORA VEB ANGAŽOVANJA

*Moguće je definisati lokaciju za čuvanje datoteka i na alternativni način.*

U primeru je navedena lokacija gde će biti smeštena datoteka koja je otpremljena direktnim navođenjem putanje do foldera na C disku (videti kod datoteke `upload.jsp`). Međutim, ako se bolje pogleda kod navedene datoteke moguće je primetiti jednu instrukciju koja je obeležena kao Java komentar i koja će u slučaju prevođenja veb aplikacije biti ignorisana. Ta instrukcija je data sledećom linijom koda:

```
....
context = pageContext.getServletContext();
String filePath = context.getInitParameter("file-upload");
.....
```

U ovom slučaju je neophodno definisati lokaciju na kojoj će prosledjene datoteke biti sačuvane primenom datoteke `web.xml`. Na ovaj način kreiramo backend JSP skript koji će uzeti parametar `file-upload` i na osnovu njega u XML elementu `<param-value>` označiti lokaciju za čuvanje datoteka kojima se manipuliše. Sledi listing datoteke `web.xml`.

```
<web-app>

  <context-param>
    <description>Lokacija otpremljene datoteke</description>
    <param-name>file-upload</param-name>
    <param-value> c:\priv\ </param-value>
  </context-param>

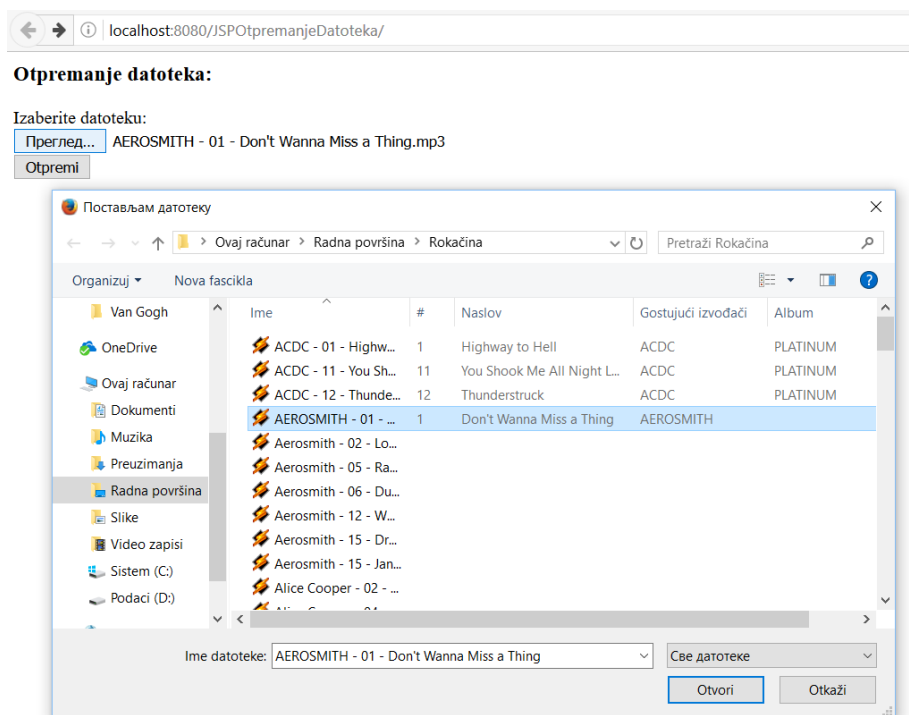
</web-app>
```

Svaki aplikativni server ima svoje specifičnosti. Datoteka `web.xml` je opšta konfiguraciona datoteka za veb aplikacije. Budući da se koristi Glassfish server, dominantno u primerima ovih materijala, neke specifične primene ovog servera na veb aplikacije moguće je realizovati datotekom `glassfish-web.xml`. Specifični tagovi i hijerarhija ovog konfiguracionog fajla dostupna je na linku [https://docs.oracle.com/cd/E18930\\_01/html/821-2417/beaq.html](https://docs.oracle.com/cd/E18930_01/html/821-2417/beaq.html).

## PRIMER 5 - OTPREMANJE DATOTEKA - DEMONSTRACIJA

*U narednom izlaganju je neophodno pokazati kako funkcioniše kreirana veb aplikacija.*

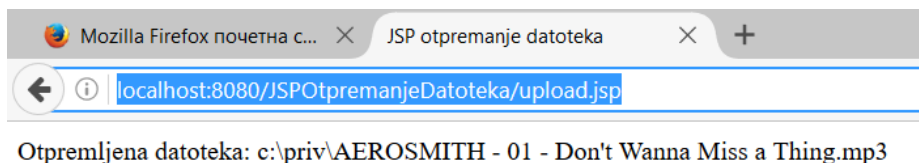
U narednom izlaganju je neophodno pokazati kako funkcioniše kreirana veb aplikacija. Na formi sa slike broj 1, bira se datoteka, ili više njih, koja će biti poslata ka serveru. Navedeno je pokazano sledećom slikom.



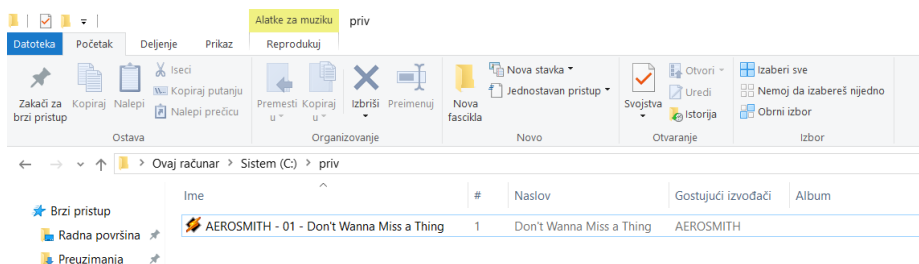
Slika 3.3 Izbor datoteke za otpremanje

Moguće je primetiti da je angažovan Windows Explorer, klikom na dugme "Pregled", koji omogućava izbor datoteke. Zatim, vrši se izbor željene datoteke i proces se, nakon izbora, završava klikom na dugme "Otvori". Sada je kontrola vraćena na formu i klikom na dugme "Otpremi" datoteka se šalje ka serveru i skladišti na naznačenoj lokaciji.

Klikom na poslednje dugme realizuje se URL string `http://localhost:8080/JSPotpremanjeDatoteka/upload.jsp` i stranica `upload.jsp` je prikazana u veb pregledaču (sledeća slika).



Slika 3.4 JSP stanica za rukovanje otpremanjem



Slika 3.5 Datoteka je otpremljena na pravu lokaciju

## ZADATAK 3

### *Implementirajte otpremanje datoteka*

Koristeći prethodni primer, implementirajte u vlastitoj aplikaciji otpremanje datoteka.

## ✓ Poglavlje 4

# Rukovanje datumom u JSP stranicama

## KLASA DATE U JSP STRANICAMA

*Klasu Date je moguće koristiti na brojne načine u JSP stranicama.*

Jedna od najznačajnijih prednosti primene JSP stranica jeste mogućnost potpunog korišćenja metoda iz Java jezgra. Na taj način je dostupna i Java klasa Date čija će široka primena u veb aplikacijama, a u okviru kreiranih JSP stranica, biti analizirana i demonstrirana u ovom delu lekcije.

Klasa Date se nalazi u paketu java.util i enkapsulira tekući datum i vreme.

Važno je napomenuti da klasa **Date** poseduje dva konstruktora. Prvi konstruktor je tzv. podrazumevani konstruktor i priložen je sledećom linijom koda:

```
Date( )
```

Drugi konstruktor uzima jedan argument koji odgovara broju milisekundi koje su protekle od ponoći 1. januara 1970. Upravo ovako računar računa tekuće vreme. Ovaj konstruktor je priložen sledećom linijom:

```
Date(long millisec)
```

Kreiranjem objekta klase **Date**, korisniku se stavlja na raspolaganje veliki broj korisnih klasa kojima ovaj objekat može da manipuliše. U sledećem izlaganju će biti navedene i opisane najznačajnije od njih:

- **boolean after(Date date)** - Vraća vrednost **true** ukoliko je objekat Date koji poziva metodu "mlađi" od objekta argumenta metode. U suprotnom vraća **false**;
- **boolean before(Date date)** - Vraća vrednost **true** ukoliko je objekat Date koji poziva metodu "stariji" od objekta argumenta metode. U suprotnom vraća **false**;
- **Object clone( )** - Duplira objekat **Date** koji ga poziva;
- **int compareTo(Date date)** - Poredi da li su objekti dva datuma jednaka. Vraća 0 ako su jednaki, broj veći od nule ako je objekat pozivač metode "stariji" od objekta argumenta metode i, na kraju, broj manji od nule ako je objekat pozivač metode "mlađi", u smislu vrednosti datuma, od objekta argumenta metode
- **boolean equals(Object date)** - Poredi da li su objekti dva datuma jednaka. Vraća **true** ako su jednaki, a u suprotnom vraća **false**;

- `long getTime( )` - Vraća broj milisekundi koje su protekle od ponoći 1. januara 1970 do trenutka poziva metode;
- `int hashCode( )` - Vraća **hash code** objekta pozivača;
- `void setTime(long time)` - Podešava broj milisekundi koje su protekle od ponoći 1. januara 1970 do trenutka poziva metode;
- `String toString( )` - Konvertuje objekat datuma u **String** i vraća rezultat.

## PRIMER 6 - PRIHVATANJE TEKUĆEG DATUMA I VREMENA

*Cilj da se uradi detaljna analiza i demonstracija problematike korišćenja tekućeg vremena.*

U jednom od polaznih primera je pokazano kako se tekući datum može koristiti u veb aplikacijama. Sada je cilj da se uradi detaljna analiza i demonstracija ove problematike. Za ovu svrhu će biti kreirana konkretna veb aplikacija kao primer korišćenja objekata i metoda java klase **Date** u JSP stranicama.

Moguće je koristiti veoma jednostavan primer. U JSP stranici se kreira **Date** objekat koji će pozvati metodu `toString()`, nakon čega će JSP stranica u veb pregledaču pokazati rešenje, kao na sledećoj slici:

### Prikaz trenutnog datuma i vremena

Wed Dec 28 19:51:25 CET 2016

Slika 4.1 Rezultat izvršavanja kreirane JSP stranice

Prikazana JSP stranica je realizovana sledećim JSP kodom:

```
<%--
    Document    : index
    Created on  : 28.12.2016., 19.50.13
    Author      : Vladimir Milicevic
--%>

<%@ page import="java.io.*,java.util.*, javax.servlet.*" %>
<html>
  <head>
    <title>Prikaz trenutnog datuma i vremena</title>
  </head>
  <body>
    <center> <h1>Prikaz trenutnog datuma i vremena</h1> </center>
    <% Date date = new Date();
    out.print( "<h2 align=\"center\">" +date.toString()+"</h2>"); %>
  </body>
</html>
```

Slika 1 je rezultat izvršavanja sledećeg URL Stringa u veb pregledaču:



<http://localhost:8080/JSPDatumDemo/>

## PRIMER 7 - FORMATIRANJE DATUMA U JSP STRANICAMA

*U veb aplikacijama je često potrebno prikazati datum i vreme u konkretnom formatu.*

U nastavku izlaganja će biti govora o načinu formatiranja datuma i vremena u nekom konkretnom formatu. **SimpleDateFormat** klasa za formatiranje i prosleđivanje datuma u zavisnosti od standarda regiona u kojem će softver biti primenjivan. Postoje brojni šabloni za prikazivanje datuma i vremena i ova klasa ih sve podržava.

Na primer, u prethodnoj JSP datoteci moguće je uvesti izvesne modifikacije pomoću kojih će datum koji se prikazuje biti prikazan u nekom od formata koji je karakterističan za region u kojem živimo i radimo.

Modifikacije bi trebalo uraditi na takav način da se kao rezultat izvršavanja JSP stranice pojavi sledeći izlaz u veb pregledaču:

**Prikaz trenutnog datuma i vremena**  
**sre 28.12.2016 u 23:24:52 CET**

Slika 4.2 Formatiran datum i vreme

Modifikacije su priložene sledećim listingom:

```
<%--
    Document    : index
    Created on   : 28.12.2016., 19.50.13
    Author      : Vladimir Milicevic
--%>

<%@page import="java.text.SimpleDateFormat"%>
<%@ page import="java.io.*,java.util.*, javax.servlet.*" %>
<html>
    <head>
        <title>Prikaz trenutnog datuma i vremena</title>
    </head>
    <body>
        <center> <h1>Prikaz trenutnog datuma i vremena</h1> </center>
        <% /*Date date = new Date();
        out.print( "<h2 align=\"center\">" +date.toString() + "</h2>"); */
        Date dNow = new Date( );
        SimpleDateFormat ft = new SimpleDateFormat ("E dd.MM.yyyy 'u' HH:mm:ss zzz");
```

```
out.print( "<h2 align=\"center\">" + ft.format(dNow) + "</h2>");%>  
</body>  
</html>
```

Iz listinga je moguće primetiti da se priloženim šablonom formatiranja prikazuje naziv dana u nedelji, datum po lokalnom formatu, 24 h format vremena i vremenska zona.

## KODOVI KLASSE SIMPLEDATEFORMAT

*Određeni karakteri imaju posebno značenje u šablonima formatiranja datuma i vremena.*

Kao što je moguće primetiti u prethodnom primeru se javljaju neki specijalni karakteri koji imaju posebno značenje u šablonima formatiranja datuma i vremena. U ovom delu lekcije je od velikog značaja njihovo navođenje i objašnjenje. U **String** šablonima datuma i vremena javljaju se sledeći specijalni karakteri:

- **G** - označava eru (n. e ili p. n. e - nova era ili pre nove ere; engleski standard je AD ili BC);
- **y** - označava godinu (može biti u dvocifrenom ili četvorocifrenom formatu);
- **M** - mesec u godini;
- **d** - dan u mesecu;
- **h** - sat u formatu 12h;
- **H** - sat u formatu 24h;
- **m** - minuti u satu;
- **s** - sekunde u minutu;
- **S** - milisekunde;
- **E** - dan u nedelji;
- **F** - dan u nedelji u mesecu (na primer 2 (druga nedelja u julu));
- **w** - nedelja u godini;
- **W** - nedelja u mesecu;
- **a** - AM / PM marker;
- **k** - sat u danu (12h format - ceo broj npr. 2 označava drugi sat u danu);
- **K** - sat u danu (24h format);
- **Z** - vremenska zona;
- **'** - prostor za malo teksta - jednostruki navodnik;

Za kompletnu listu ovi specijalnih karaktera, moguće je konsultovati Java dokumentaciju.

## ZADATAK 4

*Rad sa datumom i vremenom u JSP*

Kreirajte vlastitu JSP stranicu koja prikazuje formatirano trenutno vreme i datum.

## ▼ Poglavlje 5

# Preusmeravanje JSP stranica

## PREUSMERAVANJE STRANICA

*Preusmeravanje se koristi da bi klijent bio upućen na novu lokaciju.*

Preusmeravanje stranica (redirect) je veoma važna funkcionalnost u veb aplikacijama koja se koristi kada se izvesni dokument premešta na novu stranicu. Preusmeravanje se koristi da bi klijent bio upućen na novu lokaciju, kada je neophodno izvršiti balansiranje učitavanja podataka ili neko drugo prilagođavanje veb aplikacije.

Preusmeravanje zahteva na novu stranicu može veoma jednostavno da bude obavljeno pozivom metode `sendRedirect()` objektom odgovora. Sledećim kodom je priložen potpis ove metode:

```
public void response.sendRedirect(String location)
    throws IOException
```

Ova metoda šalje nazad odgovor veb pregledaču, zajedno sa statusnim kodom i lokacijom nove stranice, na koju se vrši preusmeravanje. Takođe je moguće koristiti metode `setStatus()` i `setHeader()`, u kombinaciji sa objektom odgovora, za postizanje identičnih rezultata preusmeravanja. Navedeno je, u opštem obliku, priloženo sledećim linijama koda:

```
....
String site = "http://www.novastranica.com" ;
response.setStatus(response.SC_MOVED_TEMPORARILY);
response.setHeader("Location", site);
....
```

U narednom izlaganju je neophodno kroz primer pokazati kako se u JSP stranicama primenjuje preusmeravanje-

## PRIMER 8 - PREUSMERAVANJE STRANICA - DEMONSTRACIJA

*Kreiranje primera za demonstraciju preusmeravanja.*

Sada je moguće prikazati na konkretnom primeru kako se u JSP stranicama implementira funkcionalnost preusmeravanja. Neka je kreirana JSP stranica sa sledećim kodom:

```
<%@page import="java.text.SimpleDateFormat"%>
<%@ page import="java.io.*,java.util.*, javax.servlet.*" %>
<html>
  <head>
    <title>Prikaz trenutnog datuma i vremena</title>
  </head>
  <body>
    <center> <h1>Prikaz trenutnog datuma i vremena</h1> </center>
    <% /*Date date = new Date();
    out.print( "<h2 align=\"center\">" +date.toString() + "</h2>"); */
    Date dNow = new Date( );
    SimpleDateFormat ft = new SimpleDateFormat ("G E dd.MM.yyyy 'u' HH:mm:ss zzz");
    out.print( "<h2 align=\"center\">" + ft.format(dNow) + "</h2>");
    %>

    <center> <h1>Cekamo na preusmeravanje na novu stranicu
      <% for (int i=0; i<5; i++)
        out.print('.');
      %>
    </h1> </center>
    <% // Nova lokacija za preusmeravanje
      String site = new String("http://www.metropolitan.ac.rs");

      response.setStatus(response.SC_MOVED_TEMPORARILY);
      response.setHeader("Refresh", "5; URL=" + site);

    %>
  </body>
</html>
```

Izvršavanjem navedene stranice, u veb pregledaču se dobija sledeći izlaz:

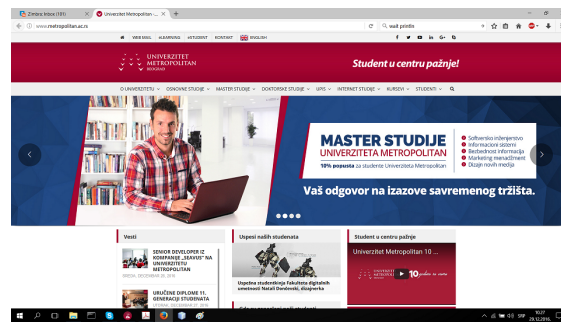
## Prikaz trenutnog datuma i vremena

n. e čet 29.12.2016 u 10:17:24 CET

## Cekamo na preusmeravanje na novu stranicu .....

Slika 5.1 JSP stranica koja čeka na preusmeravanje

Ako se obrati pažnja na instrukciju: `response.setHeader("Refresh", "5; URL="+site)`, može se uočiti da se njenim izvršavanjem omogućava čekanje od 5 sekundi pre nego što se izvrši preusmeravanje na URL koji joj je predat **String** objektom `site`. Konstantom **SC\_MOVED\_TEMPORARILY** preusmeravanje će biti privremeno (može biti i permanentno za **SC\_MOVED\_PERMANENTLY**, videti link <https://tomcat.apache.org/tomcat-5.5-doc/servletapi/javax/servlet/http/HttpServletResponse.html>).



Slika 5.2 Stranica na koju je kreirana JSP preusmerena

## ZADATAK 5

*Implemetirajte preusmeravanje JSP stranica.*

Kreirajte JSP stranicu koja vrši preusmeravanje na vašu Facebook stranicu.

## ▼ Poglavlje 6

# JSTL - JavaServer Pages Standard Tag Library

## JAVASERVER PAGES STANDARD TAG LIBRARY

*JSTL nudi brojne JSP tagove koji su od velike koristi u radu sa JSP stranicama.*

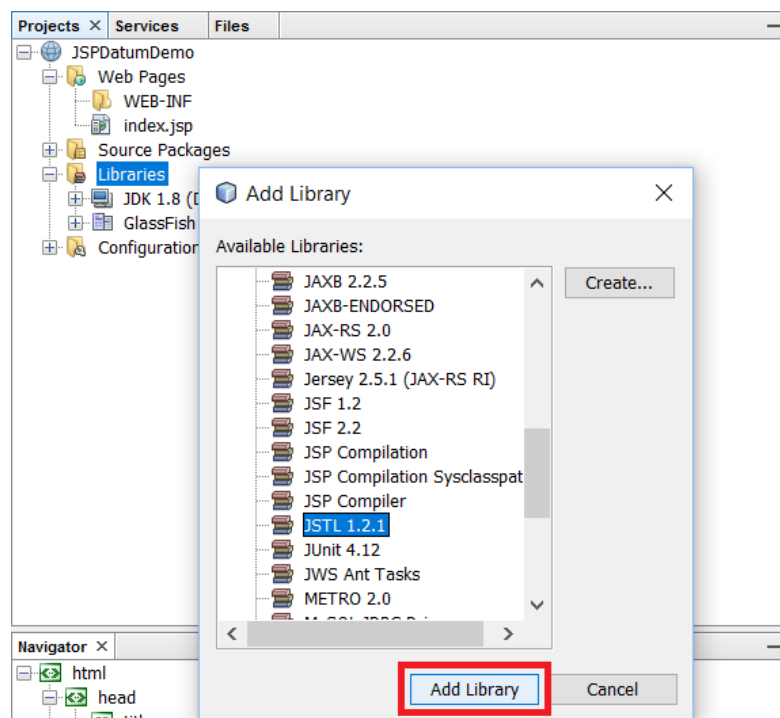
JavaServer Pages Standard Tag Library je kolekcija korisnih JSPTagova koji obuhvataju osnovne funkcionalnosti brojnih JSP stranica. JSTL daje podršku za opšte i strukturane zadatke poput iteracija i uslova, za manipulisanje XML dokumentima, za internacionalizaciju, za podršku realizovanju SQL upita i tako dalje.

JSTL tagovi mogu biti klasifikovani po nameni u odgovarajuće grupe JSTL tagova koje se mogu koristiti prilikom kreiranja JSP stranica. Postoje sledeće grupe JSTL tagova:

- Osnovni tagovi;
- Tagovi za formatiranje;
- SQL tagovi;
- XML tagovi;
- JSTL funkcije.

Ukoliko razvojno okruženje, u kojem se razvija veb aplikacija, ne poseduje podršku za JSTL, neophodno je preuzeti odgovarajuće biblioteke i uključiti ih u projekat. Preuzimanje je moguće obaviti pozivom linka: <https://jstl.java.net/download.html>.

Kada u razvojnom okruženju, kao što je na primer NetBeans IDE, postoji podrška za JSTL tagove, veoma je jednostavno omogućiti njihovu primenu u projektu koji se kreira. Desnim klikom na folder **Libraries**, u hijerarhiji projekta, bira se opcija **Add Library** koja je prikazana prozorom sa sledeće slike. Klikom na **Add Library** dugme, sve je spremno za dalji rad.



Slika 6.1.1 Dodavanje JSTL biblioteke u projekat

## JSTL OSNOVNI TAGOVI

### *Sledi pregled osnovnih JSTL tagova.*

U sledećem izlaganju će biti akcenat na pisanju i primeni osnovnih JSTL tagova. Ovi tagovi su u najfrekventnijoj primeni i zahtevaju uključivanje Core JSTL biblioteke da bi mogli da budu korišćeni. Navedena biblioteka se uključuje na sledeći način:

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

Sledi lista osnovnih JSTL tagova zajedno sa odgovarajućim objašnjenjima:

- **<c:out >** - Koristi se kao `<%= ... >`, ali za izraze;
- **<c:set >** - Postavlja rezultat provere izraza u određeni kontekst;
- **<c:remove >** - Briše konkretnu varijablu iz konteksta;
- **<c:catch>** - Hvata bilo koji **Throwable** objekat i po potrebi ga prikazuje;
- **<c:if>** - Jednostavan uslov koji proverava vlastito telo ukoliko je uslov tačan;
- **<c:choose>** - Jednostavni uslov koji obezbeđuje kontekst za primenu međusobno isključivih operacija **<when>** i **<otherwise>**;
- **<c:when>** - Podtag taga **<choose>**, ekvivalentan uslovu **if**;
- **<c:otherwise >** - Podtag taga **<choose>**, ekvivalentan uslovu **else**;
- **<c:import>** - Vraća apsolutni ili relativni URL i dostavlja njegov sadržaj stranici, Stringu u 'var' ili Reader - u u 'varReader';
- **<c:forEach >** - Osnovni tag iteracije koji prihvata različite kolekcije tipova;
- **<c:forTokens>** - Iterira kroz tokene;
- **<c:param>** - Dodaje parametar u URL taga import;

- `<c:redirect>` - Vršiti preusmeravanje na novi URL;
- `<c:url>` - Kreira URL sa opcionim parametrima upita.

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

## JSTL TAGOVI FORMATIRANJA

*Sledi prikaz JSTL elemenata sintakse kojima se obavljaju formatiranja u JSP stranicama.*

JSTL tagovi za formatiranje se koriste za formatiranje i prikazivanje teksta, vremena, datuma i brojeva za internacionalizovane veb sajtove. Sledeća sintaksa uključuje biblioteku za formatiranje u JSP stranicu:

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

Kao i u prethodnom slučaju, biće data detaljna lista tagova formatiranja koja uključuje i odgovarajuće opise:

- `<fmt:formatNumber>` - Kreira numeričku vrednost sa određenom specifikacijom i preciznošću;
- `<fmt:parseNumber>` - Parsira String reprezentaciju broja, valute ili procenta;
- `<fmt:formatDate>` - Formatira datum i može da koristi dostupne stilove i šablone;
- `<fmt:parseDate>` - Parsira String reprezentaciju datuma;
- `<fmt:bundle>` - Učitava paket resursa koji će biti korišćen u okviru svog tela;
- `<fmt:setLocale>` - Podešava lokalizaciju u lokalnoj konfiguracionoj varijabli;
- `<fmt:setBundle>` - Učitava paket resursa i čuva ga u odgovarajućoj varijabli;
- `<fmt:timeZone>` - Specificira vremensku zonu;
- `<fmt:setTimeZone>` - Podešava vremensku zonu za konfiguracionu varijablu vremenske zone;
- `<fmt:message>` - Prikazuje internacionalizovanu poruku;
- `<fmt:requestEncoding>` - Podešava zahtevano kodiranje karaktera.

## JSTL SQL I XML TAGOVI

*Predstoje još neki korisni JSTL tagovi u narednom izlaganju.*

JSTL SQL tagovima je omogućeno interagovanje sa relacionim bazama podataka u JSP stranicama. Sledeća sintaksa uključuje JSTL SQL biblioteku u JSP stranicu:

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

Sledi lista JSTL SQL tagova zajedno sa odgovarajućim objašnjenjima:



- `<sql:setDataSource>` - Kreira jednostavan izvor podataka primenjiv samo za kreiranje prototipova;
- `<sql:query>` - Izvršava SQL upit definisan u telu elementa ili kroz sql atribut;
- `<sql:update>` - Izvršava SQL ažuriranje u telu elementa ili kroz sql atribut;
- `<sql:param>` - Podešava parametar u SQL iskazu za specificiranu vrednost;
- `<sql:dateParam>` - Podešava parametar u SQL iskazu za specificiranu vrednost `java.util.Date`;
- `<sql:transaction>` - Podešava izvršavanje svih SQL iskaza kao jednu transakciju.

JSTL XML tagovima je omogućeno kreiranje i manipulisanje XML dokumentima iz JSP stranica. Sledeća sintaksa uključuje JSTL XML biblioteku u JSP stranicu:

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

Sledi lista JSTL XML tagova zajedno sa odgovarajućim objašnjenjima:

- `<x:out>` - Kao i `<%= ... >`, ali za XPath izraze;
- `<x:parse>` - Parsira XML podatke specificirane preko atributa ili u telu elementa;
- `<x:set>` - Podešava varijablu na vrednost XPath izraza;
- `<x:if>` - Proverava vrednost XPath izraza. Ako je tačan izvršava se telo **if** bloka, u suprotnom se ignoriše;
- `<x:forEach>` - Petlja nad čvorovima XML dokumenta;
- `<x:choose>` - Još jedno jednostavno granjanje;
- `<x:when>` - Podelement za `<x:choose>` koji je ekvivalentan **if** uslovu;
- `<x:otherwise>` - Podelement za `<x:choose>` koji je ekvivalentan **else** delu grananja;
- `<x:transform>` - Primenjuje XSL transformaciju na XML dokument;
- `<x:param>` - Koristi se zajedno sa prethodnim tagom za podešavanje parametara u XSLT stilovima.

## JSTL FUNKCIJE

*Sledi pregled korisnih JSTL funkcija koje je moguće koristiti u JSP stranicama.*

JSTL uključuje brojne korisne funkcije od kojih je većina zadužena za manipulisanje sa stringovima. Sledeća sintaksa uključuje biblioteku za JSTL funkcije u JSP stranicu:

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

Sledi lista JSTL funkcija zajedno sa odgovarajućim objašnjenjima:

- `fn:contains()` - Proverava da li string sadrži zadati podstring;
- `fn:containsIgnoreCase()` - Proverava da li string sadrži zadati podstring ignorišući velika i mala slova;
- `fn:endsWith()` - Testira da li se string završava određenim sufiksom;
- `fn:escapeXml()` - Znaci za izlaz se mogu tumačiti kao XM oznake;

- **fn:indexOf()** - Vraća indeks prvog pojavljivanja zadatog podstringa;
- **fn:join()** - Spaja sve elemente niza u String objekat;
- **fn:length()** - Vraća dužinu stringa;
- **fn:replace()** - Vraća string koji je nastao zamenom svih pojavljivanja nekog njegovog podstringa zadatim stringom;
- **fn:split()** - Deli string u niz podstringova;
- **fn:startsWith()** - Proverava da li string počinje odgovarajućim prefiksom;
- **fn:substring()** - Vraća podskup stringa;
- **fn:substringAfter()** - Vraća podskup stringa koji sledi nakon nekog drugog podstringa;
- **fn:substringBefore()** - Vraća podskup stringa koji prethodi nekom drugom podstringu;
- **fn:toLowerCase()** - Konvertuje sve karaktere stringa u mala slova;
- **fn:toUpperCase()** - Konvertuje sve karaktere stringa u velika slova;
- **fn:trim()** - Eliminise praznine iz stringa.

Mnogo jednostavnih JSTL primera je moguće pronaći na linku: [https://www.tutorialspoint.com/jsp/jsp\\_standard\\_tag\\_library.htm](https://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm).

## ▼ 6.1 Primer 9 - Rad sa bazama podataka

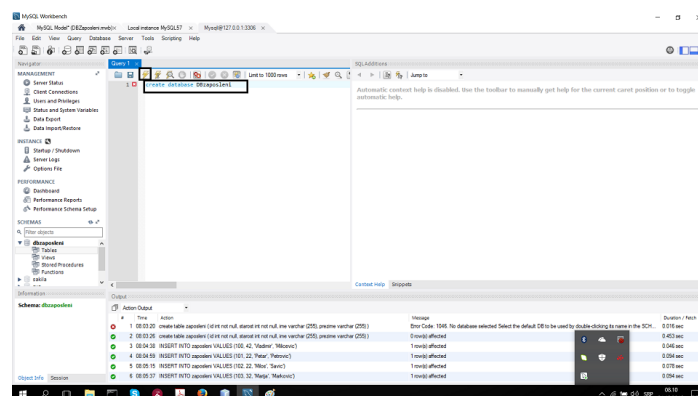
### KREIRANJE BAZE PODATAKA I TABELA

#### *Demonstracija kreiranja tabele baze podataka iz JSP stranice.*

Budući da će u posebnoj lekciji akcenat biti na pristupima i manipulaciji podacima, ova lekcija će imati jednostavan i kratak zadatak koji se odnosi na primenu i izvršavanje osnovnih **CRUD** operacija u JSP stranicama.

Za početak je neophodno kreirati jednu jednostavnu bazu podataka koja će poslužiti kao osnova za izvođenje primera i demonstraciju problematike korišćenja **SQL** upita u JSP stranicama. Baza će biti kreirana koristeći i upravljanja primenom **MySQL servera**.

Za prvih nekoliko upita, kreiranje baze podataka, kreiranje i popunjavanje tabele koja će biti korišćena u primeru, upotrebljen je **MySQL Workbench** (sledeća slika):



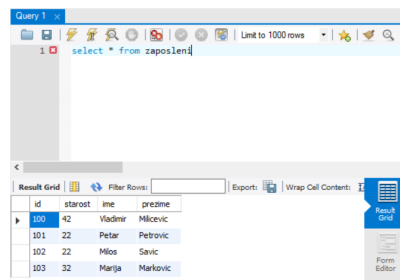
Slika 6.2.1 Kreiranje baze podataka

Nakon toga neophodno je pokrenuti kreiranu bazu podataka i u njoj kreirati tabelu koja će biti osnov za testiranje izvršavanja SQL upita u JSP stranicama. U polju za upite, alata MySQL Workbench, unosi se i pokreće sledeći SQL upit:

```
create table zaposleni ( id int not null,  
starost int not null,  
ime varchar (255),  
prezime varchar (255) );
```

Nakon toga je neophodno izvršiti popunjavanje kreirane tabele konkretnim unosima. Izvršavanjem sledećih SQL instrukcija, kreirana tabela **zaposleni**, baze podataka **DBZaposleni**, dobiće željeni sadržaj:

```
INSERT INTO zaposleni VALUES (100, 42, 'Vladimir', 'Milicevic')  
INSERT INTO zaposleni VALUES (101, 22, 'Petar', 'Petrovic')  
INSERT INTO zaposleni VALUES (102, 22, 'Milos', 'Savic')  
INSERT INTO zaposleni VALUES (103, 32, 'Marija', 'Markovic')
```



Slika 6.2.2 Testirane unosa u MySQL Workbench

## IZVRŠAVANJE OPERACIJE SELECT U JSP STRANICAMA

*Operacija **SELECT** je najčešće korišćenja u SQL upitima.*

Prethodnom slikom je pokazano kako se koristi opšti SQL **select** upit za proveru korektnosti unosa vrsta u kreiranoj bazi podataka. U nastavku, koristeći JSP stranice sa JSTL tagovima, objašnjenim u prethodnom objektu učenja, biće demonstrirana implementacija naredbe **select** za prikazivanje vrednosti konkretnih kolona, iz konkretnih tabela baze podataka.

Očekivani izlaz, vidljiv korisniku u veb pregledaču, prikazan je sledećom slikom:

ID	Ime	Prezime	Starost
100	Vladimir	Milicevic	42
101	Petar	Petrovic	22
102	Milos	Savic	22
103	Marija	Markovic	32

Slika 6.2.3 Izvršavanje instrukcije SELECT u JSP stranici

Da bi program mogao da koristi MySQL bazu podataka, neophodno je bilo u folder Libraries, projekta JSPDBDemo, uključiti JDBC drajver za MySQL, koji se naziva (naziv uključuje njegovu najnoviju verziju): mysql-connector-java-5.1.40-bin.jar.

U narednom izlaganju je neophodno pokazati programski kod kojim je realizovana funkcionalnost prikazana prethodnom slikom. Sledi listing:

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
  <head>
    <title>SELECT Operacija</title>
  </head>
  <body> <sql:setDataSource var="snapshot"
                                driver="com.mysql.jdbc.Driver"
                                url="jdbc:mysql://localhost/DBZaposleni"
                                user="root" password="vlada"/>
    <sql:query dataSource="${snapshot}" var="result"> SELECT * from zaposleni;
  </sql:query> <table border="1" width="100%">
    <tr> <th>ID</th>
    <th>Ime</th>
    <th>Prezime</th>
    <th>Starost</th> </tr>
    <c:forEach var="row" items="${result.rows}">
      <tr> <td><c:out value="${row.id}"/></td>
        <td> <c:out value="${row.ime}"/></td>
        <td> <c:out value="${row.prezime}"/></td>
        <td><c:out value="${row.starost}"/></td> </tr>
    </c:forEach>
  </table>
</body>
</html>
```

## IZVRŠAVANJE OPERACIJE INSERT U JSP STRANICAMA

*Neophodno je pokazati kako se u JSP stranicama mogu dodavati novi zapisi u tabele baze podataka.*

U prethodnom izlaganju je pokazano kako se iz JSP stranica može izvršiti prikazivanje željenog sadržaja baze podataka implementacijom SQL instrukcije **SELECT**. Sada je cilj da se pokaže funkcionalnost kojom je omogućeno dodavanje novih zapisa u bazu podataka, implementacijom SQL naredbe **INSERT**, ukoliko za tim postoji potreba. Za demonstraciju ove funkcionalnosti biće kreirana JSP stranica **insert.jsp** koja je realizovana kodom priloženim u sledećem listingu:


```
<%--
    Document    : insert
    Created on   : 31.12.2016., 08.44.27
    Author      : Vladimir Milicevic
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
    <head>
        <title>INSERT Operacija</title>
    </head>
    <body>
        <sql:setDataSource var="snapshot"
                        driver="com.mysql.jdbc.Driver"
                        url="jdbc:mysql://localhost/DBZaposleni"
                        user="root" password="vlada"/>
        <sql:update dataSource="${snapshot}" var="result">
            INSERT INTO zaposleni VALUES (104, 33, 'Jovana', 'Jovic');
        </sql:update>
        <% response.sendRedirect("index.jsp"); %>

    </body>
</html>
```

Nakon što je obavljeno umetanje novog reda u bazu podataka, pomoću SQL upita **insert** izvršavanjem ažuriranja tabele elementom `<sql:update>`, vrši se preusmeravanje na JSP stranicu `index.jsp` koja će prikazati rezultate nastale dodavanjem novog sloga u tabelu baze podataka.

Navedeno je pokazano sledećom slikom.



ID	Ime	Prezime	Starost
100	Vladimir	Milicevic	42
101	Petar	Petrovic	22
102	Milos	Savic	22
103	Marija	Markovic	32
104	Jovana	Jovic	33

Slika 6.2.4 Dodavanje novog reda u tabelu baze podataka

## IZVRŠAVANJE OPERACIJE DELETE U JSP STRANICAMA

*Neophodno je pokazati kako se u JSP stranicama mogu izbrisati zapisi iz tabele baze podataka.*

Prethodno je pokazano kako je u JSP stranicama moguće dodavanje novih zapisa u tabelama baze podataka. Ovde će se ići obrnutom logikom. Implementacijom SQL instrukcije **DELETE** cilj je omogućavanje brisanja nekog konkretnog reda iz navedene tabele baze podataka.

Od posebnog značaja je dalja demonstracija i usavršavanje primene JSTL instrukcija u JSP programiranju.

Za potrebe demonstracije biće kreirana nova JSP stranica, pod nazivom **delete.jsp**. Njen zadatak će biti brisanje konkretnog reda iz tabele **zaposleni**, na primer reda određenog sa id = 102. Nakon brisanja i ažuriranja, vrši se preusmeravanje na početnu JSP stranicu **index.jsp**, koja bi trebalo da korisniku prikaže sledeće rezultate:



ID	Ime	Prezime	Starost
100	Vladimir	Milicevic	42
101	Petar	Petrovic	22
103	Marija	Markovic	32
104	Jovana	Jovic	33

Slika 6.2.5 Brisanje reda iz tabele baze podataka

Kreirana JSP stranica **delete.jsp** realizovana je programskim kodom koji je priložen listingom koji sledi u narednom izlaganju:

```
<% -
    Document    : delete
    Created on   : 31.12.2016., 09.20.46
    Author      : Vladimir Milicevic
- %>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
    <head>
        <title>DELETE Operacija</title>
    </head>
    <body>
        <sql:setDataSource var="snapshot"
                           driver="com.mysql.jdbc.Driver"
                           url="jdbc:mysql://localhost/DBZaposleni"
                           user="root" password="vlada"/>
        <sql:update dataSource="${snapshot}" var="result">
            DELETE FROM zaposleni WHERE Id = 102
        </sql:update>
        <% response.sendRedirect("index.jsp"); %>

    </body>
</html>
```

## IZVRŠAVANJE OPERACIJE UPDATE U JSP STRANICAMA


*Neophodno je pokazati kako se vrši ažuriranje postojećih podataka u tabeli baze podataka.*

Promena podataka u tabeli baze podataka, u JSP stranicama, vrši se implementacijom SQL naredbe **UPDATE** u okviru elementa `<sql:update>`, baš kao što je bio slučaj i sa prethodnim SQL klauzulama.

U konkretnom slučaju, biće formirana nova JSP datoteka, `update.jsp`, čiji je zadatak da promeni broj godina za zaposlenog čiji je id = 103. Neka je navedeno realizovano sledećom SQL instrukcijom:

```
UPDATE zaposleni SET starost = 40
WHERE id = 103
```

Izvršavanjem ove datoteke i ažuriranjem tabele `zaposleni`, menjaju se konkretne vrednosti u tabeli, a zatim se vrši preusmeravanje na stranicu `index.jsp` koja bi trebalo da prikaže sledeći izlaz, dostupan korisniku putem veb pregledača:



ID	Ime	Prezime	Starost
100	Vladimir	Milicevic	42
101	Petar	Petrovic	22
103	Marija	Markovic	40
104	Jovana	Jovic	33

Slika 6.2.6 Ažuriranje podataka vrste baze podataka u JSP stranici

Posebno će u listingu JSP koda stranice `update.jsp` biti pokazano kako se ovaj upit povezuje sa konkretnim identifikatorom. Sledi listing koda kojim je realizovano navedeno:

```
<% -
    Document    : update
    Created on  : 31.12.2016., 09.34.47
    Author      : Vladimir Milicevic
- %>

<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
    <head>
        <title>DELETE Operacija</title>
    </head>
    <body>
        <sql:setDataSource var="snapshot"
                           driver="com.mysql.jdbc.Driver"
                           url="jdbc:mysql://localhost/DBZaposleni"
                           user="root" password="vlada"/>

        <sql:update dataSource="${snapshot}" var="result">
            UPDATE zaposleni SET starost = 40
            WHERE id = 103
        </sql:update>
        <% response.sendRedirect("index.jsp"); %>
```

```
</body>  
</html>
```

## ZADATAK 6

### *Samostalno vežbanje rada sa bazom podataka iz JSP*

Kreirajte bazu podataka FAKULTET koja, za sada, ima jednu tabelu STUDENTI (brojIndeksa, ime, prezime, godinaUpisa);

Iskoristite prethodni primer i izvršite CRUD operacije iz JSP stranica nad ovom bazom podataka.



## ▼ Poglavlje 7

# JSP - JavaBean

## KLASA JAVABEAN

*JavaBeans klase su specifične i postoji nekoliko osobina po kojima se razlikuju od ostalih Java klasa*

JavaBean je specijalno konstruisana Java klasa kreirana u skladu sa JavaBeans API specifikacijama. JavaBeans klase su specifične i postoji nekoliko osobina po kojima se razlikuju od ostalih Java klasa. Sledi lista tih osobina:

- Obezbeđuju podrazumevane argumente bez konstruktora;
- Trebalo bi da implementiraju Serializable interfejs;
- Sadrže brojne osobine koje je moguće učitavati i upisivati;
- Za osobine sadrže brojne **setter** i **getter** metode.

U narednom izlaganju je neophodno priložiti listu metoda kojima se manipuliše JavaBean objektima, odnosno njihovim osobinama:

- **getProperty** - preuzima vrednost osobine na osnovu njenog naziva. Ako se, na primer, osobina naziva "Ime" odgovarajuća metoda za preuzimanje vrednosti ove osobine će nositi naziv **getIme()**;
- **setProperty** - menja vrednost osobine na osnovu njenog naziva. Ako se, na primer, osobina naziva "Ime" odgovarajuća metoda za preuzimanje vrednosti ove osobine će nositi naziv **setIme()**;

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

## PRIMER 10 - KONSTRUISANJE JAVABEAN KLASE I PRISTUP ZRNIMA

*U narednom izlaganju je neophodno pokazati kako se kreira klasa iz koje se izvode Java zrna.*

U narednom izlaganju je neophodno pokazati kako se kreira klasa iz koje se izvode Java zrna. Klasa će da obrađuje studente i njihove osobine, a kao što je napomenuto u prethodnom izlaganju, implementiraće interfejs Serializable.

```
package com.metropolitan.primer;

/**
 *
 * @author Vladimir Milicevic
 */
public class StudentiBean implements java.io.Serializable{
    private String ime = null;
    private String prezime = null;
    private int starost = 0;

    public StudentiBean() { }

    public String getIme(){
        return ime;
    }

    public String getPrezime(){
        return prezime;
    }

    public int getStarost(){
        return starost;
    }

    public void setIme(String ime){
        this.ime = ime;
    }

    public void setPrezime(String prezime){
        this.prezime = prezime;
    }

    public void setStarost(int starost){
        this.starost = starost;
    }
}
```

U nastavku je potrebno kreirati JSP stanicu kojom je moguće pristupati kreiranom zrnju. Akcija [useBean](#) omogućava primenu zrna u JSP stranici. Kada je jednom zrno deklarirano, ono postaje skripting promenljiva kojoj je moguće pristupati odgovarajućim elementima u JSP stranici. Navedeno je moguće prikazati sledećim opštim kodom:

```
<jsp:useBean id="id" class="bean's class" scope="bean's scope">
    <jsp:setProperty name="bean's id" property="property name" value="value"/>
    <jsp:getProperty name="bean's id" property="property name"/> .....
</jsp:useBean>
```

U narednom izlaganju će upravo biti akcenat na kreiranju JSP datoteke koja će sadržati elemente za manipulisanje osobinama kreiranog Java zrna.

## JSP DATOTEKA ZA PRISTUP JAVA ZRNU

*Sledećom datotekom se implementiraju set i get metode za osobine zrna.*

U prethodnom izlaganju je kreirana klasa zrna iz koje bi trebalo da se kreiraju konkretne instance ove klase koje će odgovarati pojedinačnim studentima. Takođe je pokazano i koji akciju bi trebalo da implementira JSP datoteka ako želi da pristupa i manipuliše osobinama kreiranih Java zrna.

Sledećim listingom može biti definisana JSP stranica sa traženim funkcionalnostima:

```
<html>
  <head>
    <title>Primer primene get i set metoda</title>
  </head>
  <body>
    <jsp:useBean id="student" class="com.metropolitan.primer.StudentiBean">
      <jsp:setProperty name="student" property="ime" value="Petar"/>
      <jsp:setProperty name="student" property="prezime" value="Petrovic"/>
      <jsp:setProperty name="student" property="starost" value="22"/>

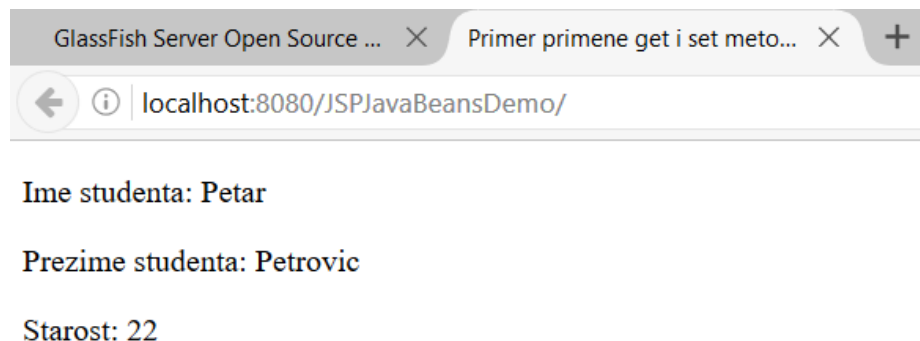
    </jsp:useBean>

    <p>Ime studenta: <jsp:getProperty name="student" property="ime"/> </p>
    <p>Prezime studenta: <jsp:getProperty name="student" property="prezime"/>
  </p>

    <p>Starost: <jsp:getProperty name="student" property="starost" /> </p>

  </body>
</html>
```

Ako se testira kreirana JSP datoteka dobija se izlaz kojeg korisnik može da uoči u veb pregledaču u sledećem obliku:



Slika 7.1 Manipulisanje Java zrnom iz JSP

U JSP stranici, koja je zadržala naziv `index.jsp`, prvo je kreiran objekat klase `StudentiBean` pod nazivom `student`. Zatim su u JSP stranici angažovane `setter` metode klase da bi kreirano zrno dobilo osobine sa konkretnim vrednostima.

Na kraju je u JSP stranci izvršeno angažovanje `getter` metoda klase `StudentiBean` da bi informacije o objektu konkretnog studenta bile prikazane kao rezultat u veb pregledaču.

## ZADATAK 7

### *Kreirajte vlastito zrno*

Kreirajte zrno klase Automobil (marka, tip, godište) i upravljajte kreiranim zrnom na način demonstriran u prethodnom primeru.

## ▼ Poglavlje 8

# JSP - jezik izraza

## EL - EXPRESSION LANGUAGE (JEZIK IZRAZA)

*Prikaz elemenata jezika za manipulisanje podacima iz Java zrna u JSP stranicama.*

EL - Expression Language (jezik izraza) omogućava jednostavan pristup aplikativnim podacima koji se čuvaju u komponentama koje odgovaraju Java zrnima. EL dozvoljava kreiranje izraza koji mogu biti aritmetički i logički. U okviru JSP EL izraza moguće je koristiti podatke različitih tipova: numeričke (celobrojne i realne), stringove, ugrađene konstante, logičke konstante, null vrednosti i tako dalje.

JSP EL ima beoma jednostavnu sintaksu. Opšti oblik izraza glasi:

```
${izraz}
```

Ovakav izraz može da se ugradi i testira na pogodnom mestu u JSP stranici, ili što je ovde od posebnog značaja, u string koji odgovara atributu **values** u okviru elementa `<jsp:setProperty>`, na primer na sledeći opšti način:

```
<jsp:setProperty name="nazivZrna" property="nazivOsobine" value="${izraz}"/>
```

Na primer, prethodni slučaj može biti modifikovan i podešavanje osobine za starost studenta može biti obavljeno na sledeći način:

```
<jsp:setProperty name="student" property="starost" value="${student.getStarost()+22}"/>
```

Rezultati izvršavanja modifikovanog primera u potpunosti su identični prethodnom slučaju prikazanom slikom1 u prethodnom objektu učenja.

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

## JSP EL - ELEMENTI SINTAKSE

*U ovom delu lekcije cilje je kratak pregled elemenata sintakse JSP EL jezika.*

JSP EL podržava većinu aritmetički i logičkih Java operatora. Sledi lista najčešće korišćenih operatora u JSP stranicama primenom EL jezika.

Operator	Opis
.	Pristup osobini zrna
[]	Pristup nizu ili listi elemenata
( )	Grupisanje podizraza za promenu redosleda ocenjivanja
+	Sabiranje
-	Oduzimanje ili negacija
*	Množenje
/ or div	Deljenje
% or mod	Moduo
== or eq	Jednakost
!= or ne	Nejednakost
< or lt	Manje od
> or gt	Veće od
<= or le	Manje ili jednako
>= or ge	Veće ili jednako
&& or and	Logičko "i"
or or	Logičko "ili"
! or not	Logičko "ne"

Slika 8.1 Operatori u EL

Takođe u JSP EL jeziku je moguće pozivanje i funkcija na sledeći opšti način:

```
${ns:func(param1, param2, ...)}
```

Primer poziva funkcije može biti sledeći:

```
${fn:length("Vrati duzinu stringa")}
```

Za korišćenje funkcije iz neke biblioteke tagova, neophodno je prvo instalirati tu biblioteku na serveru ili je uključiti u JSP stranicu <taglib> elementom.

JSP EL jezik podržava i izvesne implicitne objekte koji su priloženi i objašnjeni sledećom tabelom.

Implicitni objekat	Opis
pageScope	Varijable iz oblasti stranice
requestScope	Varijable iz oblasti zahteva
sessionScope	Varijable iz oblasti sesije
applicationScope	Varijable iz oblasti aplikacije
param	Parametri zahteva kao string
paramValues	Parametri zahteva kao kolekcija stringova
header	Zaglavlja HTTP zahteva kao string
headerValues	Zaglavlja HTTP zahteva kao kolekcija stringova
initParam	Kontekstna inicijalizacija parametra
cookie	Vrednost kolačića
pageContext	JSP PageContext objekat za tekuću stranicu

Slika 8.2 JSP EL implicitni objekti

## PRIMER 11- JSP EL - ELEMENTI SINTAKSE

*Neophodno je dati još neka razmatranja u vezi sa JSP EL.*

Na jednostavnom primeru, neophodno je, za kraj, pokazati još jednom kako je moguće elemente JST EL jezika uključiti i koristiti u JSP stranicama. Na primer, biće iskorišćen **header** implicitni objekat koji je priložen i opisan u prethodnoj tabeli. Neka je to urađeno na način priložen sledećim JSP listingom:

```
<%@ page import="java.io.*,java.util.*" %>
<% String title = "Korisnicki agent primer"; %>
<html>
  <head>
    <title><% out.print(title); %></title>
  </head> <body>
    <center> <h1><% out.print(title); %></h1> </center>
    <div align="center"> <p>${header["user-agent"]}</p> </div>
  </body>
</html>
```

JSP stranica pristupa parametrima zaglavlja koja su označena kao **user-agent** i u veb pregledaču daje sledeći izlaz.



Slika 8.3 Korišćenje implicitnog objekta "header"

## ZADATAK 8

### *Analiza JSP EL koda*

Šta će se desiti kada se izvrši sledeća JSP naredba?

```
<div align="center"> <p>${header["user-agent"]}</p> </div>
```



## ▼ Poglavlje 9

# JSP - internacionalizacija

## DETEKCIJA LOKALIZACIJE

*JSP često koristi klasu `Locale` i njena polja i metode.*

JSP često koristi klasu `Locale` i njena polja i metode za obezbeđivanje upotrebe odgovarajućeg jezika i lokalnih podešavanja. Objekat ove klase moguće je dobiti sledećim pozivom:

```
java.util.Locale request.getLocale()
```

Postoje brojne značajne metode ove klase koje mogu biti upotrebljene za detekciju lokacije klijenta koji je uputio zahtev, jezika i brojnih lokalnih podešavanja i standarda. Sledi lista najčešće korišćenih metoda:

- `String getCountry()` - vraća kod zemlje / regiona u ISO 3166 - 2 formatu;
- `String getDisplayCountry()` - vraća naziv zemlje;
- `String getLanguage()` - vraća kod jezika u ISO 639 formatu;
- `String getDisplayLanguage()` - vraća naziv jezika;
- `String getISO3Country()` - vraća skraćenicu za odgovarajuću zemlju (tri slova);
- `String getISO3Language()` - vraća skraćenicu za odgovarajući jezik (tri slova).

## PRIMER 12- LOKALIZACIJA

### *Demonstracija lokalizacije u JSP*

U nastavku je neophodno pokazati kako je moguće prikazati lokalne podatke u JSP stranicama. Neka je kreirana, u tu svrhu, JSP stranica sa sledećim kodom:

```
<%@ page import="java.io.*,java.util.Locale" %>
<%@ page import="javax.servlet.*,javax.servlet.http.*" %>
<% //Get the client's Locale
    Locale locale = request.getLocale();
    String language = locale.getLanguage();
    String country = locale.getCountry(); %>
<html>
    <head>
        <title>Detekcija lokalizacije</title>
    </head> <body>
        <center> <h1>Detekcija lokalizacije</h1> </center>
        <p align="center">
            <% out.println("Jezik : " + language + "<br/>");
```

```
out.println("Drzava : " + country + "<br/>"); %> </p>  
</body>  
</html>
```

Pokretanjem ove JSP stranice kreira se lokalni objekat koji obezbeđuje informacije o jeziku i zemlji, koje se prikazuju korisniku u JSP stranici na pregledaču, kao što je prikazano sledećom slikom.



Slika 9.1 Lokalne vrednosti za jezik i zemlju

Više o podešavanju jezika, datuma, valuta i procenata na linku [https://www.tutorialspoint.com/jsp/jsp\\_internationalization.htm](https://www.tutorialspoint.com/jsp/jsp_internationalization.htm).

## ZADATAK 9

### *Kreiranje vlastite stranice sa lokalnim podacima*

Kreirajte vlastitu JSP stranicu koja prikazuje lokalne podatke.

## ▼ Poglavlje 10

### Domaći zadatak 2-1

#### ZADATAK 2

*Kreiranje veb aplikacije sa naprednim JSP konceptima.*

U zadatku urađenom na vežbama uraditi sledeće modifikacije:

1. U početnoj formi dodati tekst polje za unos količine akcija;
2. Kao rešenje vratiti vrednost pojedinačne akcije i tražene količine akcija;
3. Ukoliko nije unet korektan simbol za akciju, posle određenog vremena izvršiti automatsko preusmeravanje na početnu stranicu;
4. Kreirati stranicu dobrodošlice (ona će u ovom slučaju nositi naziv index.html) koja će naglasiti da se radi o "Domaćem zadatku broj 4" i nakon klika na odgovarajući link izvršiti preusmeravanje na stranicu sa formom;

## ▼ Poglavlje 11

# Zaključak

## ZAKLJUČAK

*Lekcija je obrađivala napredne JSP elemente.*

Ovom lekcijom su obrađeni napredni JSP koncepti i kao takva ona predstavlja nastavak izlaganja iz Lekcije 3. Posebno, lekcija se bavila sledećim temama:

- Demonstracija korišćenja filtera u JSP stranicama;
- Detaljna analiza i demonstracija upravljanja kolačićima u JSP stranicama;
- Savladavanje upload - ovanja datoteka iz JSP stranica;
- Rad sa vremenom i datumom u JSP stranicama;
- Preusmeravanje sa jedna na drugu JSP stranicu;
- Detaljan uvid u JSTL (JavaServer Pages Standard Tag Library) biblioteku;
- Rad sa bazama podataka iz JSP stranica;
- Upoznavanje i rad sa JavaBean objektima;
- Savladavanje elemenata JSP EL jezika izraza;
- Internacionalizacija u JSP stranicama.

Savladavanjem ove lekcije, kao i prethodne lekcije, student u potpunosti ovladao konceptima i principima primene JSP stranica u Java veb aplikacijama i sposoban je da kreira veliki broj veb aplikacija koristeći HTML, JSP, Servlet i JavaBean tehnologije.

## LITERATURA

*U pripremanju lekcije korišćena je najnovija pisana i elektronska literatura.*

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