Matematički fakultet Univerziteta u Beogradu

Septembar, 2012.

Master rad

Obrada korisničkih zahteva korišćenjem JavaServer Pages (JSP) okruženja

Mentor:
Prof. dr Dušan Tošić

Student:
Nevena Ćurčić 1151/2010

Apstrakt

Potreba za dinamickim web stranicama raste iz dana u dan. U korak sa tim, javlja se i potreba za njihovo jednostavno i brzo kreiranje. Ovaj rad ima za cilj da predstavi upravo takvu jednu tehnologiju, JavaServer Pages (JSP) tehnologiju koja omogucava efikasno kreiranje i jednostavno održavanje složenih web aplikacija. Takode, kroz primer web aplikacije bice predstavljeno kako se korišćenjem JSP okruženja kreira jedna takva aplkacija i kako se obraduju neki od zahteva korisnika te aplikacije.

Abstract

The need for dynamic web pages grows every day. Keeping pace with this need appears another need for simple and quick creating of those pages. The goal of this thesis is to present one of this technologies , JavaServer Pages(JSP) technology that enables efficient creating and simple maintaining of complex web applications. Also, through the example of one web application it will be resented how to create such application using JSP environment and how to process some demands from users of this application.

Sadržaj

1.0 Uvod

Od prve pojave Interneta pa do danas uradeno je mnogo na proširenju mogucnosti koje Internet pruža korisnicima. Ali, potrebe i ocekivanja Internet korisnika svakodnevno rastu. Jedna od tendencija današnjice je da se web aplikacije koriste za rešavanje najrazličitijih problema iz sfere poslovanja kako malih tako i velikih preduzeca. U skladu sa današnjim načinom života i sve vecom potrebom za distribuiranim, transakcionim i prenosivim web aplikacijama, javlja se i potreba web programera za alatom koji će na što jednostavniji i brži nacin omoguciti kreiranje upravo takvih aplikacija. Jedno od najefikasnijih rešenja, samo po sebi se namece - JavaServer Pages (JSP) kao jedna od najbitnijih komponenti Java Enterprise Edition tehnologije, Sun-ove visoko skalabilne arhitekture za poslovne aplikacije.

JavaServer Pages (u daljem tekst JSP) predstavlja Javinu tehnologiju za kreiranje dinamicki generisanih web stranica na siguran, prenosiv i dobro definisan nacin. JSP tehnologija razdvaja korisnicki interfejs od Java koda, omogućavajući dizajnerima da promene opšti izgled stranice bez menjanja dinamickog sadržaja. Takode, kao Javina tehnologija, JSP stranice su platformski nezavisne pa se mogu izvršavati na bilo kojoj platformi, odnosno operativnom sistemu. Zbog svoje jednostavnosti i velikih mogucnosti koje pruža web programerima, JSP tehnologija je danas našla široku primenu u kreiranju dinamickih web stranica.

2.0 Java Platforma

Java platforma je naziv za skup povezanih Sun-ovih tehnologija koji omogucavaju razvoj i pokretanje aplikacija pisanih u Java programskom jeziku. Jezgro Java platforme cini Java virtuelna mašina (JVM).

Sve Java platforme se sastoje iz dva osnovna dela:

- Java virtuelne mašine (JVM) - platformski nezavisno izvršno okruženje koje konvertuje Java bajtkod u mašinski kod.
- Java aplikacioni programski interfejs (Java API) – predstavlja okruženje za razvoj aplikacija.

Svaka Java platforma obezbeđuje vurtuelnu mašinu i API, i ovo omogucava da se aplikacije pisane za tu platformu izvršavaju na bilo kojem kompatibilnom sistemu.

Postoje tri platforme Java programskog jezika:

- **Java Platform,Standard Edition (Java SE)** - obezbeđuje izvršno okruženje za razvoj desktop aplikacija i predstavlja jezgro funkcionalnosti za naredna izdanja Java platforme. ([11])
- **Java Platform, Enterprise Edition (Java EE)** - predstavlja nadskup JSE koji podržava razvoj složenih poslovnih (enterprise) aplikacija.
- **Java Platform, Micro Edition (Java ME)** - definiše skup izvršnih okruženja za razvoj aplikacija za uređaje kao što su mobilni telefoni, PDA uređaji, TV uredaji i ostali uredaji koji imaju manjak resursa da podrže JSE ili JEE. ([11])

2.1 Java Enterprise Edition Platforma (Java EE)

Java EE platforma obezbeduje API i izvršno okruženje za razvoj i pokretanje višeslojnih (multitier) poslovnih aplikacija. Java EE platforma može se podeliti u cetiri dela:

1. Web tehnologije koje se koriste u razvoju prezentacionog nivoa JEE ili samostalne (stand-alone) Web aplikacije:
   - Java Servleti.
   - Java server strane (JavaServer Pages - JSP).

- Java server strane sa standardnom bibliotekom tagova (JavaServer Pages Standard Tag Library - JSTL).
- Java server oblici (JavaServer Faces – JSF).
- Internacionalizacija i lokalizacija web aplikacija.

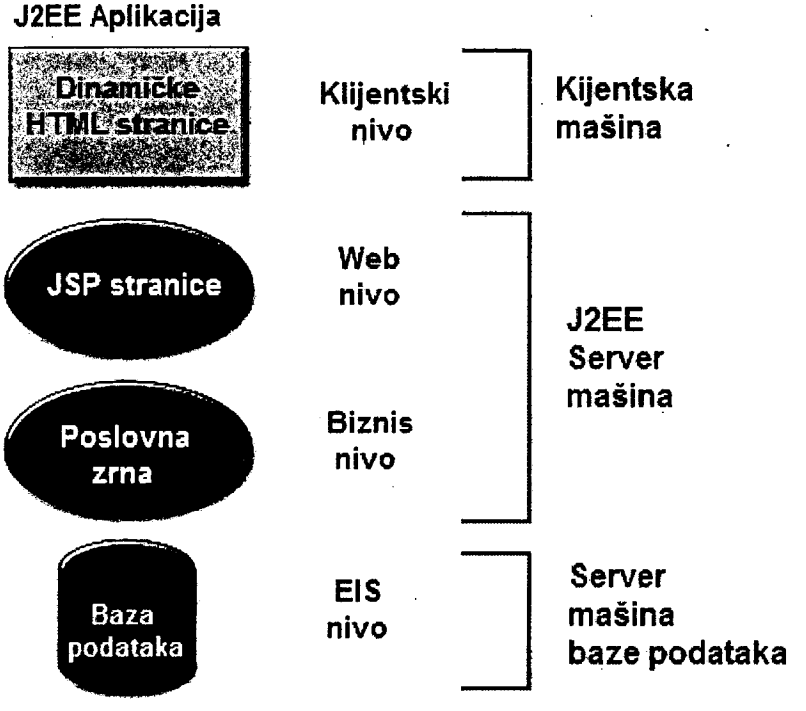
2. Enterprise JavaBeans (EJB) tehnologije koje se koriste u razvoju poslovne logike JEE aplikacije:
   - Session bean-ovi.
   - Enterprise bean-ovi.
   - Message-driven bean-ovi.
   - Enterprise JavaBeans upitni jezik.
3. Java XML tehnologije za razvoj aplikacija koje procesiraju XML dokumente i implementiraju Web servise:
   - Java API za XML procesiranje (JAXP).
   - Java API za XML-RPC (JAX-RPC).
   - SOAP attachments API za Javu (SAAJ).
   - Java API za XML registrovanje (JAXR).
4. Servisi JEE platforme koje koriste sve navedene tehnologije:
   - Transakcije (Transactions).
   - Povezivanje resursa (Resource Connections).
   - Zaštita (Security).
   - Java servis poruke (Java Message Service).

Svi Java SE API mogu biti korišćeni prilikom kreiranja Java EE aplikacija. Java EE razvojno okruženje za složene aplikacije se sastoji od JEE komponenti, web servisa (usluga), API - ja i protokola. Sa Java Enterprise Edition platformom, razvoj Java poslovnih (enterprise) aplikacija nikada nije bio jednostavniji i brži. Glavni cilj Java EE platforme je da obezbedi programerima mocan aplikacioni programski interfejs (API), dok sa druge strane smanjuje vreme razvoja i složenost aplikacije.

2.2 Distribuirane višeslojne aplikacije

Java EE platforma koristi distribuirani višeslojni aplikacioni model za poslovne aplikacije. Aplikaciona logika je podeljena na komponente prema funkciji. Komponente su instalirane na različitim mašinama u zavisnosti od nivoa, odnosno sloja kome komponenta pripada. Iako se Java EE višeslojne aplikacije mogu sastojati od tri ili više nivoa, generalno su to troslojne aplikacije. Na slici 1. je prikazana višeslojna Java EE aplikacija podeljena na tri nivoa:

- Klijentski nivo – komponente se izvršavaju na klijentskim mašinama.
- Nivo aplikacione logike – komponente se izvršavaju na JEE serveru. Ovaj nivo je podeljen na dva podnivoa: Web nivo i poslovni nivo.
- Enterprise information system (EIS) nivo – komponente se izvršavaju na EIS serveru. EIS server obicno predstavlja neki sistem za upravljanje bazama podataka – Database Management System.



Slika 1: *Primer troslojne aplikacije*

2.3 JEE komponente

Java EE komponente su sastavni deo svake Java EE aplikacije i one predstavljaju relativno nezavisan deo koda koji se može dodati ili oduzeti iz aplikacije. Java EE komponente su pisane u Java programskom jeziku i kompajliraju se na isti nacin kao i bilo koji drugi program napisan u Javi.

JEE specifikacija definiše sledeće komponente:

- Komponente koje se izvršavaju na klijentu - aplikacioni klijenti i apleti.
- Web komponente koje se izvršavaju na serveru – Java Servlet, JavaServar Pages (JSP) i JavaServer Faces (JSF).
- Poslovne komponente koje se izvršavaju na serveru - Enterprise JavaBeans (EJB) komponente.

2.3.1 Web komponente

Java EE Web komponente:

- Servleti - Servlet je komponenta koja predstavlja Java klasu koja implementira proces prihvatanja, obrade i vracanja odgovora internet citacu koji je zahtev poslao.
- JSP (JavaServer Pages) - JavaServer Pages tehnologija omogucava lako kreiranje web sadržaja koji se sastoji i od statickih i od dinamickih elemenata.
- JSF (Java Server Faces) t - JavaServer Faces tehnologija se nadograduje na Servlete i JSP tehnologiju i obezbeduje okvir komponenti korisnickog interfejsa (user interface component framework) za Web aplikacije.

(Više informacija o JEE komponentama kao i o ostalim delovima Java EE razvojnog okruženja može se naci u [8] i [11]).

3.0 Model - View - Controller (MVC) arhitektura

Model-View-Controller (MVC) je softverska arhitektura koja razdvaja aplikacioni model podataka, korisnicki interfejs i kontrolu logike u tri odvojene komponente. MVC predstavlja okvir za razvoj složenih Web aplikacija. Glavni razlog korišćenja MVC modela jeste da se odvoje komponente u tri različite jedinice, odnosno sloja: Model, View i Controller.

- Model je deo aplikacije koji je odgovoran za upravljanje podacima. Ovaj sloj predstavlja podatke i pravila koja regulišu pristup i ažuriranje tih podataka. Model skladišti i preuzima entitete koje koristi aplikacija, najčešće iz baze podataka.
- View prikazuje sadržaj Model-a. Ovaj sloj precizira kako podaci iz Modela treba da budu predstavljeni krajnjem korisniku. View je odgovoran i za prosledjivanje ulaznih podataka do Controller-a.
- Controller je deo aplikacije koji prevodi interakciju korisnika sa slojem View u akcije koje ce Model da obavlja. Ovaj sloj povezuje Model i View i omogucava njihov uzajamni rad.

U serverskim aplikacijama, najčešće se vrši klasifikacija delova aplikacije na: biznis logiku, prezentaciju i obradu zahteva. U MVC terminologiji, Model odgovara biznis logici, View prezentaciji a Contorller obradi zahteva. Korišcenjem ovakvog dizajna, omogucava se odvajanje obrade zahteva i biznis logike od prezentacije, što je jedna od glavnih prednosti korišćenja JSP-a u odnosu na Servlete. Neke od prednosti MVC arhitekture su:

- Jasan nacin projektovanja
- Efikasna modularnost
- Jednostavan nacin distribucije
- Mocni korisnicki interfejs
- Jednostavan nacin rasta aplikacije

4.0 Evolucija JSP tehnologije

Prve web stranice bile su staticke. Nakon toga, programski jezici Perl i C su korišćeni na web serveru u cilju omogućavanja dinamičkog sadržaja. Ubrzo, vecina jezika ukljucujuci VisualBasic, Delphi, C i Java mogli su da se koriste za pisanje aplikacija koje su omogućavale dinamički sadržaj korišćenjem podataka iz tekstualnih fajlova ili baza podataka. Te aplikacije su bile poznate kao Common Gateway Interface (CGI) serverske aplikacije. Kada web server primi zahtev koji pristupa CGI-u, server mora da napravi novi proces kako bi pokrenuo CGI. Ovo rešenje nije bilo dovoljno dobro jer je server za svaki zahtev koji primi morao da pravi nov proces , ucita i interpretira script, izvrši script, ponovo sve sastavi i pošalje natrag klijentu. Ovo je bilo veoma zahtevno za server. CGI script nije mogao da bude u direktnoj interakciji sa web serverom, a to je zapravo znacilo da nije bilo moguce napisati web server log fajl, nisu mogle da se dobiju informacije o serverskom okruženju, itd.

Veliki broj CGI alternativa i nadogradnji, kao što su FastCGI, mod_perl za Apache, NSAPI za Netscape, ISAPI za Microsoft bilo je napravljeno tokom godina. Ove nadogradnje su se pokretale u serverskom procesu, nisu bile ogranicene resursima, bile su brže, komunicirale su direktno sa web serverom. Medutim, njihov pad mogao je da dovede do rušenja servera jer izolacija aplikacije ovim rešenjima nije bila omogućena.

Posle CGI skripta pojavljuju se Java Servlet-i i postaju osnova svih web tehnologija na Java EE platformi. Java Servlet je Javina klasa koja sadrži kôd za obradu HTTP zahteva. Java Servlet se pokrece u okviru Java virtuelne mašine (JVM) na serverskoj strani. JVM omogućava bolju zaštitu od pada servera u odnosu na druge aplikacije dajucii izolaciju aplikacija. Svaka Javina aplikacija ukljucujuci Java Servlete je prenosiva kroz operativne sisteme i web servere.

Jedan od nedostataka Servleta je to što oni generišu web stranice ugradjivanjem HTML-a direktno u kod programskog jezika. Ovo je podstaklo razvoj dinamickih web stranica u oblasti programiranja. JSP predstavlja sledeci nivo u razvoju web tehnologija. JSP je razvijen od strane Sun Microsystems kompanije u cilju omogucavanja HTML programerima jednostavan nacin za pravljenje dinamickog sadržaja. Prvi put se JSP pojavljuje 1998. godine a

zvanicne verzije 1.0 i 1.1 su objavljene 1999. godine. Trenutno, odnosno najnovija objavljena verzija je JSP 2.2 u okviru Java EE 6 platforme.([8]) JSP tehnologija predstavlja specifikaciju koja se nadograduje na Java Servlet API, pri čemu je JSP tehnologija mnogo lakša za programirnje. Zajedno, JSP i Servleti pružaju atraktivnu alternativu za konkurentne tehnologije kao što su PHP i ASP.NET pružajuci nezavisnost od platforme, poboljšane performanse, razdvajanje logike od prikaza podataka, lakocu u administraciji, i najvažnije, jednostavnost za korišćenje.

5.0 Java Servlet tehnologija

Kako JSP specifikacija predstavlja Java Servlet API nadogradnju, i što je još važnije , JSP se ne razlikuje od Servleta u vreme izvršavanja (JSP je kompajliran i ucitan potpuno isto kao Servlet), u ovom delu teksta ce ukratko biti predstavljena i opisana Java Servlet tehnologija.

Servlet je Javina klasa koja koristi Servlet Aplication Programing Interfaca (API). Servlet API se sastoji iz velikog broja klasa i interfejsa koji definišu metode koje cine mogucim da se obradi HTTP zahtev na web-serveru, i to na nezavisan nacin. Servleti su jedan od nacina da se izgrade interaktivne web aplikacije. Servlet je komponenta na serverskoj strani koja se koristi da proširi sposobnosti servera koji izvršavaju aplikacije kojima se pristupa po principu modela zahtev - odgovor. Iako Servleti mogu da odgovore na bilo koji tip zahteva, obicno se koriste da bi proširili mogućnosti aplikacija koje se nalaze na web serveru. Svaki Servlet mora da implementira javax.servlet.Servlet interfejs koji definiše metode životnog ciklusa Servleta. Vecina Servleta implementira interfejs nasledivanjem jedne od klasa koja vec implementira neophodni javax.servlet.Servlet interfejs. Za web aplikacije najčešće se koristi javax.servlet.http.HttpServlet klasa kao roditeljska klasa. HttpServlet implementira metode za obradu HTTP zahteva kao što su: doPost(), doGet(), doPut(), doDelete(), doTrace() metod.

Servleti se u potpunosti pokreću u okviru Java virtuelne mašine i imaju pristup citavoj familiji Java API-ja, ukljucujuci JBDC API za pristup bazi podataka. Servletska klasa nije deo JDK (Java Development Kit). Servlet je objekat koji se ucitava i pokrece sa Servletske mašine ili Servletskog kontejnera, kao što je Apache Tomcat.

5.1 Životni ciklus Servlet-a

Životni ciklus Servleta je kontrolisan od strane kontejnera u kojem je Servlet rasporeden. Životlni ciklus Servleta se može opisati kroz iteraciju Servleta i web servera koja je definisana metodama u javax.servlet.Servlet interfejsu.

Životni ciklus Servleta se može opisati kroz sledece tri metode:

1. public void init(ServletConfig config) – init() metoda se poziva kada se Servlet ucitava, tj. kada kontejner kreira instance Servlet klase.
2. public void service(ServletRequest req, ServletResponse res) – metod service() se poziva nula ili više puta tokom života Servleta i prosleduje Servletu objekte koji predstavljaju zahtev i odgovor. Service() metod se poziva za svaki HTTP zahtev. Logika koda, tj. srž Servleta je smeštena u ovom metodu.
3. public void destroy( ) – poziva se neposredno pre nego što Servlet završi sa implementacijom.

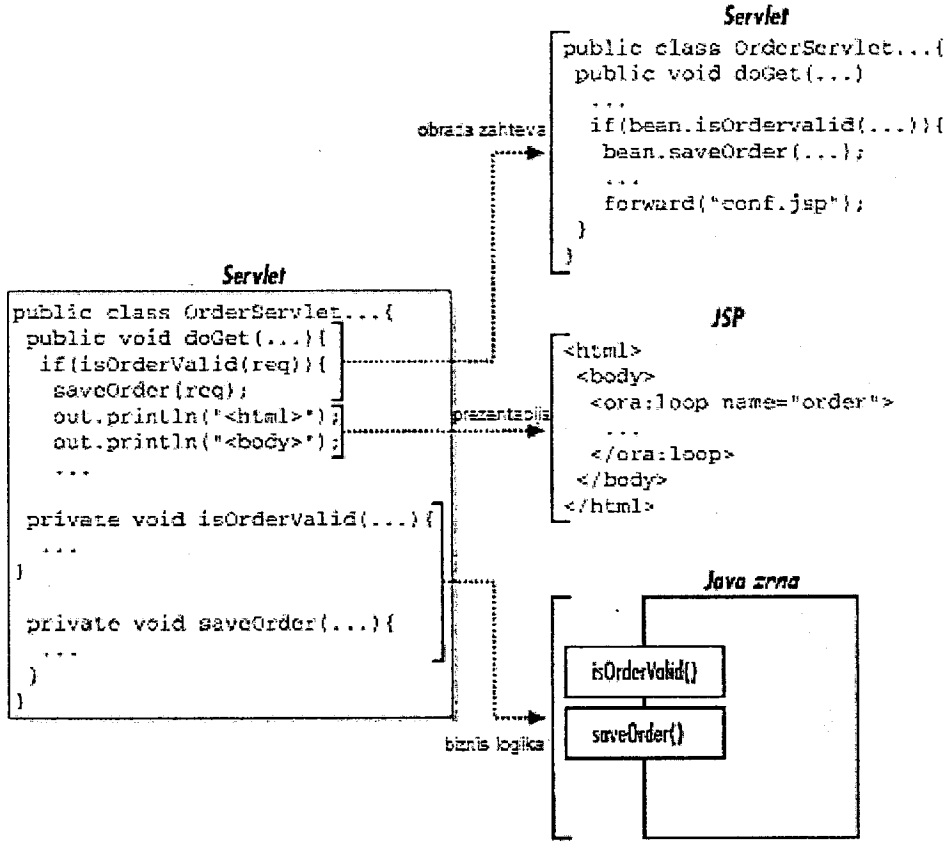
6.0 JavaServer Pages(JSP)

JSP je kljucna komponenta u Java 2 Enterprise Edition platformi. JSP specifikacija predstavlja Java Servlet API nadogradnju koja nam pruža cvrst okvir za kreiranje dinamičkih web aplikacija korišćenjem HTML, XML šablona i Java koda. JSP pruža odličnu script podršku za kreiranje web aplikacija na serverskoj strani.

Servleti mogu biti jako „glomazni" pri generisanju složenog HTML-a. Vecina Servleta sadrži malo koda koji obraduje aplikacionu logiku i puno koda koji obraduje izlazno formatiranje. Iz ovih razloga, programeri web aplikacija se okrecu prema JSP-u. JSP omogucava programerima da Java kôd ubace direktno u JSP fajl, što čini proces kreiranja veoma jednostavnim i lakim za održavanje. Za razliku od obicne HTML stranice ciji je sadržaj staticki i uvek ostaje isti, JSP stranica može da menja svoj sadržaj zasnovan na neodredjenom broju stavki kao što su npr. identifikacija korisnika, tip korisničkog internet čitača (browser), informacije dostavljene od strane korisnika, selekcije od starne korisnika, itd.

Vecina aplikacija koja je zasnovana na Java Servletima, obradu zahteva i generisanje odgovora obradje zajedno u okviru jedne Servlet klase. JSP za razliku od Servleta razdvaja obradu zahteva i biznis logiku od prezentacije, što je ilustrovano na Slici 2.

Upravo ovo razdvajanje obrade zahteva i biznis logike od prezentacije omogućava dizajnerima da promene opšti izgled stranice bez menjanja dinamickog sadržaja, odnosno programerima da promene neka pravila u biznis logici bez menjanja korisnickog interfejsa.



Slika 2: *Razdvajanje obrade zahteva, biznis logike i prezentacije*

JSP stranice su dokumenti zasnovani na tekstu koji sadrži staticki HTML i dinemicke akcije kojima se odreduje kako se obraduju odgovori klijentu na efikasan nacin. JSP stranice su najčešće HTML/DHTML dokumenti sa specijalnim etiketama (tagovima) ugradenim u HTML koji pozivaju JSP server, odnosno Java aplikacioni server. Glavna razlika izmedu JSP-a i Servleta jeste ta što su Servleti napisani u cisto Java kodu. JSP se ne razlikuje od Servleta u vreme izvršavanja. Ono u cemu se JSP razlikuje jeste u tome kako je kreiran. Okruženje u kojem se kreira JSP je jako slicno okruženju koje koriste tradicionalni web programeri (HTML, XML, CSS,.. programeri). Tacnije, jezik je više prepoznatljiv web programerima nego što je to slučaj sa Servletima.

Na jednostavnom primeru aplikacije koja ispisuje "Zdravo svete!", kroz Primere 1 i 2, može se videti suštinska razlika u kreiranju aplikacije korišćenjem Servlet tehnoloije, odnosno JSP tehnologije.

Primer 1: Servlet tehnologija

```
public class HelloWorldServlet implements Servlet
{
   public void service(ServletRequest request,
      ServletResponse response)
      throws ServletException, IOException
   {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter( );
      out.println("<html>");
      out.println("  <head>");
      out.println("    <title>Zdravo svete</title>");
      out.println("  </head>");
      out.println("  <body>");
      out.println("    <h1>Zdravo svete!</h1>");
      out.println("  </body>");
      out.println("</html>") }
  }
```

Primer 2: JSP tehnologija

```
<html>
  <head>
    <title>Zdravo svete</title>
  </head>
  <body>
    <h1>Zdravo svete!</h1>
  </body>
</html>
```

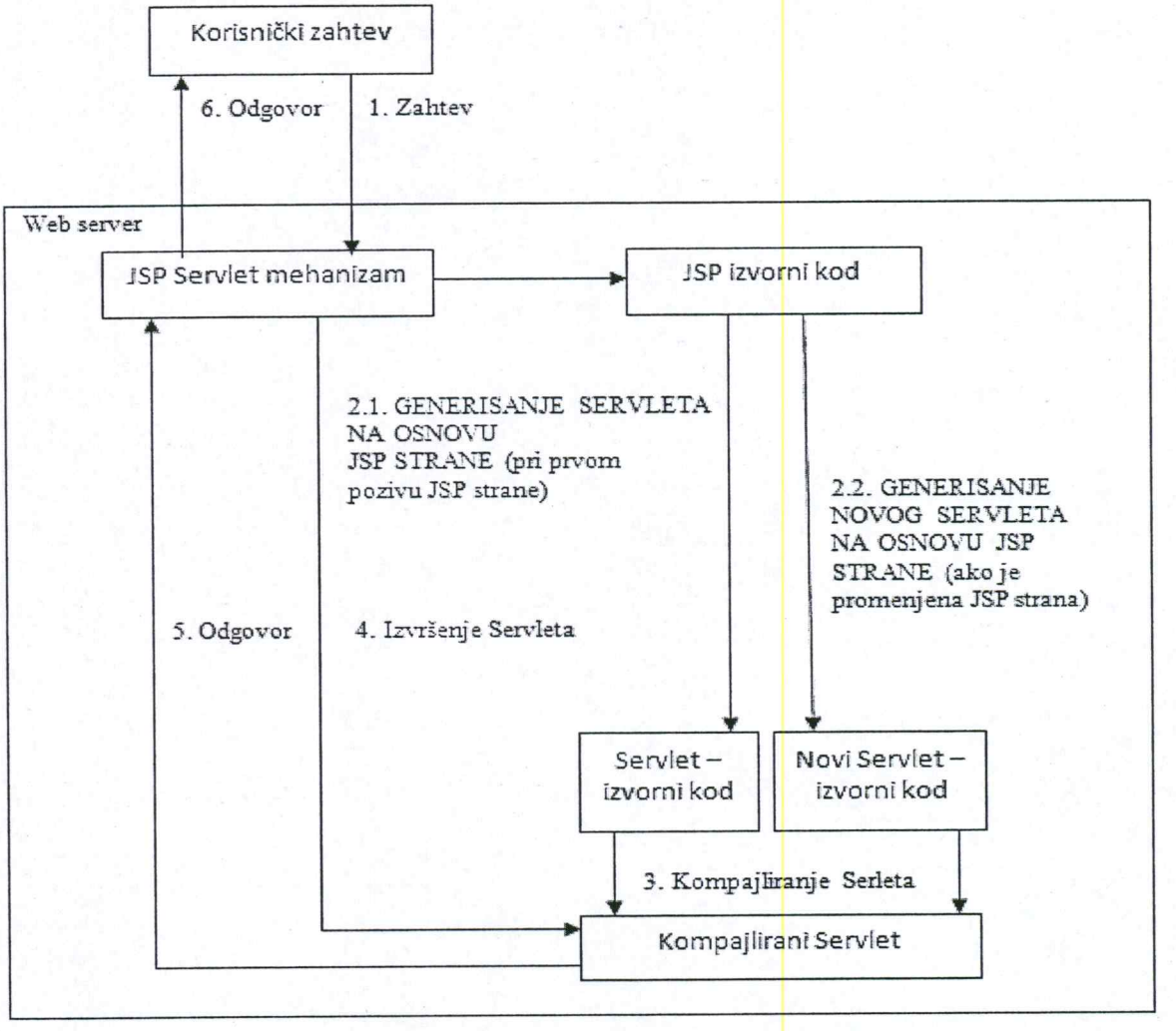
Kôd u Primeru 2 se prilikom kompilacije prevodi u Servlet i ono što zapravo server obraduje jeste Servlet kod prikazan u Primeru 1. Naravno, prilikom kreiranja složenih aplikacija problem pisanja kôda u Servlet tehnologiji je

znatno veci tako da se razmatranjem ovog primera jednostavne aplikacije jasno vide prednosti JSP tehnologije u pogledu brzine i jednostavnosti kreiranja aplikacije.

6.1 JSP Athitektura

Postoje tri scenarija izvršenja JSP stranice koji su prikazani na slici 3:

1. Klijent prvi put poziva JSP stranicu. Klijent šalje zahtev do Web servera(1). Web server pokrece mehanizam koji na osnovu JSP strane generiše Servlet (2.1), koga nakon toga kompajlira (3). JSP mehanizam izvršava Servlet (4), koji vraca neki rezultat (5). Rezultat se vraca nazad do klijenta(6).
2. Klijent pozove JSP stranicu za koju je vec formiran Servlet. Klijent šalje zahtev do Web servera (1). JSP mehanizam izvršava Servlet (4), koji vraca neki rezultat (5). Rezultat se vraca nazad do klijenta (6).
3. Promenjena je JSP stranica za koju je bio formiran Servlet (Servlet nije validan). Klijent šalje zahtev do Web servera (1). Web server pokrece mehanizam koji, na osnovu JSP strane, generiše novi Servlet (2.2), koga nakon toga kompajlira (3). JSP mehanizam izvršava Servlet (4), koji vraca neki rezultat (5). Rezultat se vraca nazad do klijenta (6).



Slika 3: *Scenariji izvršenja JSP strane*

6.2 Prednosti korišćrnja JSP-a

- Brzo kreiranje dinamicke web stranice.
- JSP je veoma jednostavna a ujedno veoma mocna tehnologija.
- JSP ima pun pristup Java API - ju, zapravo se JSP u celosti kompajlira i pokrece kao Java Servlet.
- Nezavisnost od platforme kao Javina aplikacija.
- JSP API pruža nam mogucnost za kreiranje delova koda koji mogu biti ponovo upotrebljeni korišćenjem jednostavnih tagova u JSP-u.
- Kao što možemo da ugradimo cist Java kôd u stranicu, možemo takode i JavaScript i tradicionalne tagove.

6.3 Komponente JSP-a

Pravila koja se najcešce primenjuju u svim JSP tagovima su slicna HTML pravilima:

- Etikete (tag-ovi) imaju jedan pocetni tag (npr. <body>) sa artibutima koji su neobavezni, neobavezno telo i odgovarajuci krajnji tag (npr. </body>).
- Vrednosti atributa u tagovima su navedeni.

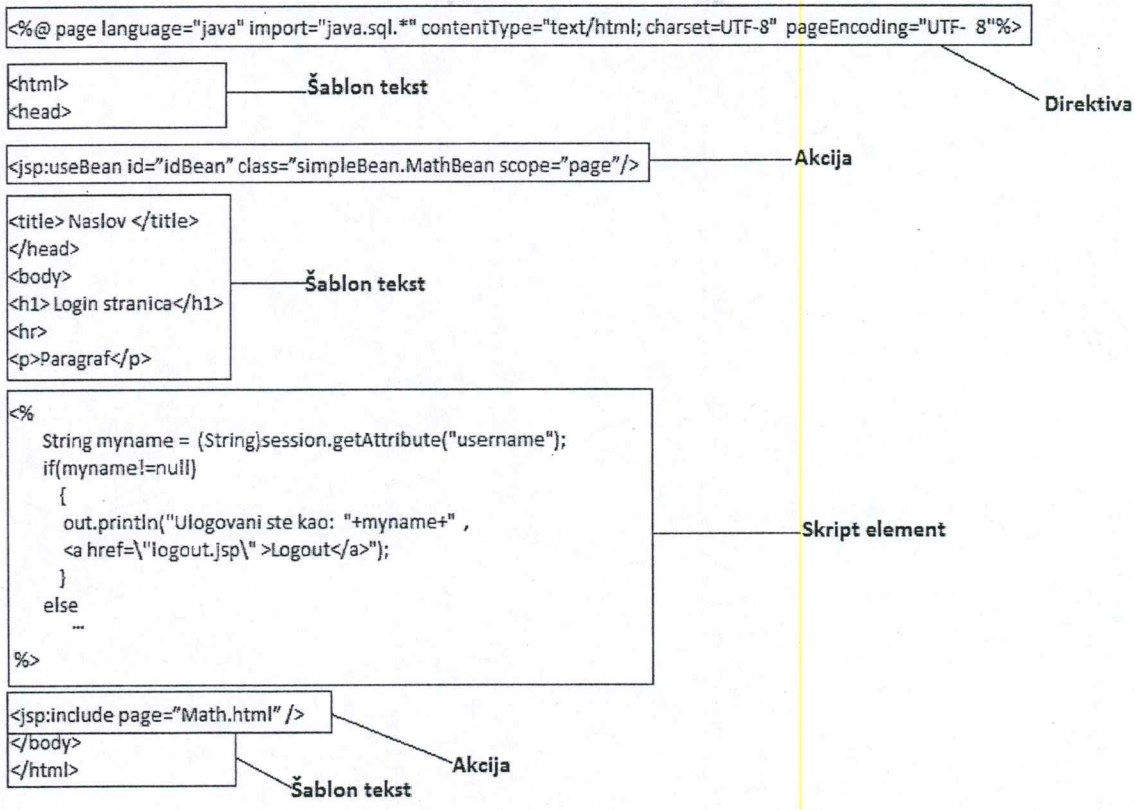
JSP podseca na HTML, ali kada se prvi put pozove, kompajlira se u Java Servlet. Rezultujuci Servlet je kombinacija HTML-a iz JSP datoteke i ugradenog dinamickog sadržaja navedenog od strane novih tagova. Sve što se nalazi na JSP stranici se može podeliti u dve kategorije. Prva se sastoji od elemenata koji se obraduju na serveru. Druga se sastoji od šablon podataka (template data) koji zapravo predstavljaju bilo koji tekst (HTML,XML, ili obican tekst). Šablon tekst se uvek direktno prosleduje internet citacu.

6.4 JSP elementi

JSP elementi, izuzev šablon podataka predstavljaju dinamicki deo JSP stranica. Sve što se prevodi i izvršava od strane JSP servera se naziva JSP element. JSP elementi se mogu klasifikovati u sledece kategorije:

- Direktive (directives)
- Akcije (actions)
- Skript elemente (scripting elements)
- Šablon podaci (template data)

Anatomija JSP stranice je prikazana na slici 4.



Slika 4: *Anatomija JSP stranice*

6.5 JSP Direktive

JSP direktiva je izraz koji daje JSP kontejneru instrukcije kako da realizuje odredene aspekte JSP obrade. JSP directive se mogu naci na skoro svakoj JSP stranici.

Sintaksa: Može biti u XML formi i ne-XML formi.

- Ne-XML: <%@ tip_direktive naziv_atributa_direktive=vrednost %>
  Primer: <%@ page language="java"%>
- XML: <jsp:directive.tip_direktive
  naziv_atributa_direktive=vrednost />
  Primer: *<jsp:directive.page import=java.util.*" />*

Postoje tri tipa direktiva:

1. **Page** – ovom direktivom možemo da uvezemo klase, postavimo tip sadržaja, atribute sesije, itd.

2. **Include** – doslovno služi za ukljucivanje fajlova
3. **Taglib** – koristi se da obavesti JSP kontejner koju tag biblioteku zaheva odredena JSP strnica

6.5.1 Direktiva – page

Page direktiva definiše atribute koji važe za celu JSP stranicu. Može imati sledece atribute: language, extends, import, buffer, session, autoFlush, isThreadSafe, info, errorPage, isErrorPage i contenetType.([7])

6.5.2 Direktiva – include

JSP stranice mogu da obuhvataju druge fajlove. Ovo je moguce korišćenjem JSP direktive include. Koristi se za unos teksta i koda u JSP pre kompilacije. Ukljucivanje se uglavnom dešava u vreme kompilacije. Svaka izmena u okviru ukljucenog fajla nece imati uticaja sve dok se sledeci put JSP fajl ne kompajlira.
Sintaksa:

- Ne-XML forma: <%@ include file=nekiURL"%>
- XML forma: <jsp:directive. include file="nekiURL"/>

Zapravo postoje dva načina za izvršavanje direktive include. Jedan način je da se izvrši u vreme kompajliranja a drugi u trenutku kada se zahteva fajl koji ukljucujemo, tako da postoje dva različita pristupa za korišćenje JSP direktive include.

- Ako želimo da koristimo direktivu include u vreme kompajliranja, koristimo vec videnu sintaksu <%@ include file="nekiURL"%>
  - Ako se bilo šta promeni u uključenom fajlu (dete fajlu), roditelj fajl mora da se rekompajlira.
  - Generiše se samo jedan Servlet kada se koristi ovaj tip direktive include.

- Delovi JSP-a koji predstavljaju dete fajl odnosno fajl koji ce da bude ukljucen ne moraju biti samostalno kompajlirani.

- Ako želimo da ukljičimo fajl u vreme izvršavanja, koristimo drugaciju etiketu (tag) <jsp:include>. Ovako korišćen element include predstavlja standardnu akciju.
  - Ako se bilo šta promeni u uključenom fajlu (dete fajlu) automatski se reflektuje u roditeljskom fajlu, odnosno JSP fajlu.
  - Odvojeni Servleti se generišu za stranice koje ukljucujemo i ukljucene JSP stranice.
  - Ukljucen JSP mora biti kompletan i mora biti u mogucnosti da se kompajlira sam.

Pored <jsp:include> etikete, cesto korišćena etiketa je <jsp:forward> koja prosleduje zahtev klijenta nekom HTML ili JSP fajlu.([7])

6.5.3 Taglib direktiva

Taglib direktiva generalno ima dva oblika. Prvi navodi *uri* i *prefix* atribute, dok drugi navodi tag fajlove. ([1] i [2])

Primer korišćenja URI za povezivanje sa tag bibliotekom

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

Primer direktnog navodenja direktorijuma tag fajlova

```
<%@ taglib prefix="t" tagdir="/WEB-INF/tags" %>
```

6.6 Akcije

Akcije su dešavanja opisana JSP elementima koja se direktno ukljucuju u obradu zahteva. U najvecem broju slucajeva, akcije nam omogucavaju pristup podacima i manipulisanje ili transformisanje podataka koji ucestvuju u kreiranju dinamickog izlaza. Akcije mogu biti standardne ili prilagodene korisniku (custom).

Standardne akcije su dostupne u svakom JSP kontejneru i to su npr. <jsp:useBean>, <jsp:getProperty>, <jsp:setProperty>, <jsp:include>, <jsp:foward>,... Akcije prilagodene korisniku su akcije kreirane korišćenjem JSP tag proširenog mehanizma. Ovaj mehanizam omogucava programerima da kreiraju sami svoj skup akcija za manipulisanje podacima.

JSP akcije se izvršavaju kada se zahteva JSP stranica, tj. u fazi obrade zahteva. Drugim recima, JSP akcije predstavljaju dinamicke akcije koje se koriste u vreme izvršavanja što je suprotno JSP direktivama koje se koriste samo u fazi prevodenja. Sinonim za akciju je tag jer je svaka akcija u XML-u zapravo tag pa se ova dva naziva cesto poistovecuju. Akcioni elementi su grupisani u biblioteke (tag biblioteke). Naziv akcije se sastoji iz dva dela: prefiksa i naziva akcije iz tag biblioteke. Na primer <jsp:useBean>. Sve akcije iz standardne JSP biblioteke imaju prefiks jsp dok akcije prilagodene korisniku mogu imati bilo koji prefiks, osim jsp, jspx, java, javax, servlet, sun, ili sunw. Potrebni podaci za akcije obezbeduju se kroz par attribute/value.

6.7 Skript elementi

Skript elementi nam daju sintaksno korektan nacin za ugradivanje dinamickog koda u HTML dokument. Skript elementi se koriste da ukljuce skriptovani kôd (Java kôd) u JSP. Oni omogucavaju deklarisanje promenljivih i metoda, ukljucivanje skriptovanog koda i obradu izraza.
Postoje tri tipa skript elemenata:

- deklaracija (declaration)
- skriptleti (sriptlets)
- izrazi (expressions)

6.7.1 JSP Deklaracije

Deklaracija predstavlja blok Java koda u JSP-u koji se koristi da definiše promenljive na nivou klase i metode koje mogu da se koriste u samoj stranici.

- Deklaracija promenljivih – promenljive postaju promenljive instance Servlet klase u koju se JSP prevodi i kompajlira.
- Deklaracija metoda – metode postaju metode Servlet klase izvan service() metoda.

Sintaksa:

- Ne-XML forma : <%! Java kod%>

  Primer: <%! int brojac=0; %> , <%! int dodaj(int brojac) { return brojac++; }%>
- XML forma: <jsp:declaration> Java kod </jsp:declaration>

6.7.2 JSP Skriptleti

Skriptlet sadrži jedan ili više Javinih iskaza. Najčešće skriptlet predstavlja blok Javinog koda koji se izvršava u vreme obrade zahteva. Ovaj kod se dodaje u service() metod generisanog Servleta. Sa JSP script elementima možemo da kontrolišemo koji delovi JSP stanice će biti poslati u internet čitač kao odgovor korisniku.

Sintaksa:

- Ne-XML forma : <% Java kod%>
- XML forma: <jsp:scriptlet> Java kod </jsp:scriptlet>

Unutar skriptleta se mogu:

- Deklarisati promenljive i metode koje ce se kasnije koristiti u JSP stranici.
- Pisati validne Java naredbe.
- Koristiti implicitni objekti (deklarisani izvan skriptleta) ili drugi objekti deklarisani kao <jsp:useBean> objekti.

Iz skriptleta se može pristupiti sledecim implicitnim objektima([5]) koji ce biti generisani u Servletu:

- request – obezbeduje metode koje nam daju pristup informacijama koje su dostupne u vezi sa trenutnim zahtevom, kao što su parametri, atributi, cookies,..
- response – odgovor klijentu.

- pageContext – kontekst strane, iz koga se dobijaju brojne informacije o aplikaciji, Servletu, sesiji, objektu izlaznog toka.
- session – HTTP session objekat koji omogucava da se unutar njega kreiraju promenljive sesije.
- application – na osnovu pageContext.getServletContext() dobija se objekat same aplikacije unutar koje se koristi JSP strana.
- out – izlani tok, koji se povezuje sa response objektom i pomocu koga se šalju podaci do klijenta.
- config – konfiguracioni objekat za stranu.
- page – objekat koji sadrži referencu na samog sebe.
- exception – objekat koji se prosleduje do strane koja obraduje greške koje se dešavaju u toku izvršenja programa.

6.7.3 JSP Izrazi

Izrazi se koriste za ugradivanje vrednosi iz Java izraza u HTML sadržaj. Izraz se konvertuje u String i prikazuje se u izlaznom toku zajedno sa ostatkom HTML-a.

Sintaksa:

- Ne-XML forma : <%= Java izraz%>

  Primer: <input name="Drzava" type="text"
  value="<%=(String)lista.get(10) %>" />
- XML forma: <jsp:expression> Java izraz </jsp:expression>

6.8 JSP komentari

Postoje dva tipa komentara u JSP jeziku:

- Izlazni komentari - omogucavaju prikazivanje komentara u HTML-u u internet citacu. Ovi komentari mogu biti staticki i dinamicki.
  - Staticki: <! -- Ovo je komentar -->
  - Dinamicki: <! -- Korisničko ime je <%=name%> -->

- Skriveni komentari – omogucavaju programerima da dokumentuju svoj rad u okviru JSP stranice. Skriveni komentari se izostavljaju prilikom kompilacije i prikazivanja klijentu. <%-- Ovo je komentar-- %>

6.9 Java zrna (JavaBeans)

Dinamcki sadržaj je sadržaj generisan od strane nekog serverskog procesa, npr. rezultat SQL upita. Pre nego što se dinamički sadržaj pošalje internet citacu, dinamicki sadržaj mora da se spoji sa regularnim HTML elementima. Rezultat serverske obrade -dinamicki sadržaj- je u JSP tehnologiji najčešće predstavljen komponentom Java zrno.

Java zrna (JavaBeans) su Java klase koje se mogu višestruko upotrebljavati i komponovati zajedno u aplikacijama. Svaka Java klasa koja prati odgovarajici obrazac implementiranja može biti Java zrno komponenta. JSP tehnologija direktno podržava upotrebu komponente Java zrno korišćenjem JSP elemenata. Java zrno (u daljem tekstu samo zrno) se najčešće koristi u JSP-u kao kontejner za dinamicki sadržaj koji ce biti prikazan na internet stranici. Zrno je uvek kreirano od strane serverskog procesa i predaje se JSP stranici gde se koriste JSP elementi za ubacivanje podataka iz zrna u HTML šablon tekst. Tip elemenata koji se koriste za pristup zrnu u stranici se nazivaju JSP standardne akcije. JSP stranica može da napravi i koristi bilo koji Java objekat korišcenjem deklaracija i skriptleta. Medutim, pomocu zrna kreiraju se daleko mnogo konciznije JSP stranice. Pored toga, prednost korišćenja zrna je i u tome što se sve informacije nalaze u jednom paketu zahvaljujući tome što se zrno koristi i za prikaz podataka a i za uzimanje podataka koje je korisnik uneo. Dakle, sa zrno specifikacijom, možemo da kreiramo višestruko upotrebljivu, od platforme nezavisnu komponentu koja je laka za održavanje.

6.9.1 Struktura zrna

Komponenta zrno reguliše svojstva (proprerty) klase i ureduje public metode koje omogucevaju pristup svojstvima. Zrno je klasa sa podrazumevanim konstruktorom što omogućava JSP kontejneru kreiranje instance zrno klase. Takode, zrno sadrži metode za citanje i pisanje vrednosti svojstva. set() i get() metode se direktno pozivaju kada koristimo vrednost svojstva zrna u skript elementu.

Dakle, zrno sadrži:

- Svojstva (properties) - Svojstvima se smatraju atributi konkretne instance zrna u Java zrno komponenti. Svojstva mogu biti za citanje/pisanje (read/write), samo citanje(read-only), ili samo pisanje (write-only) a mogu biti i jednostavna u smislu da sadrže pojedinacnu vrednost ili indeks.
- Ponašanje definisano metodama - ovo je glavni funkcionalni interfejs zrna. Ove metode mogu biti kombinacija private i public metoda.
- Dogadaje (events).

6.9.2 Korišćenje zrna u JSP-u

Postoje tri jako jednostavne a u isto vreme jako korisne etikete zrna, odnosno JSP etikete koje nam omogucavaju interakciju sa podacima zrna. Te etikete su:

- `<jsp:useBean/>` - omogućava nam da deklarišemo da ce JSP stranica koristiti komponentu zrno.
- `<jsp:getProperty/>` - omogucava nam da uzmemo vrednost nekog svojstva zrna i da je prikažemo u internet stranici.
- `<jsp:setProperty/>` - omogucava nam da promenimo/postavimo vrednost nekog svojstva zrna.

<jsp:useBean../> etiketa

<jsp:useBean /> etiketa se u JSP stranici koristi da locira postojecu instancu zrna ili da kreira novu ako instanca nije pronadena.

Postoje dve sintaksne forme ove etikete:

- <jsp:useBean id="beanName" class="className scope="scope"/>
- <jsp:useBean id="beanName" class="className scope="scope">
  // inicijalizovan kod
  <jsp:useBean />

Druga forma etikete se koristi kada želimo da ukljucimo <jsp:setProperty/> iskaze za inicijalizovanje svojstva zrna. Atribut scope definiše kako ce zrno biti podeljeno sa drugim JSP stranicamu u aplikaciji, odnosno odredjuje njegovu vidljivost. Atribut scope može da ima vrednost, odnosno zrno može biti vidljivo u okviru stranice (page), sesije (session), zahteva (request), aplikacije (application).

Primer: <jsp:useBean id="idBean" class="bean.Korisnik
scope="page"/>

Atribut *id* se koristi da dodeli zrnu jedinstveno ime, dok atribut *class* sadrži puno kvalifikaciono ime klase koja predstavlja zrno. Pored ovih atributa, <jsp:useBean> akcija podržava dodatna dva atributa: type i beanName.

Podaci zrna su predstavljeni preko njegovih svojstva (properties). Kada se jednom kreira zrno i dodeli mu se ime, mogu se uzeti vrednosti njegovih svojstava pomocu druge JSP standardne akcije <jsp:getProperty>.

<jsp:getProperty/> etiketa

Postoji nekoliko nacina da se preuzmu svojstva komponente zrno. Jedna od metoda jeste korišćenje <jsp:getProperty> elementa, odnosno <jsp:getProperty> etikete.

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

Atribut *name* se odnosi na instancu zrna koju smo definisali sa <jsp:useBean/> standardnom akcijom. Vrednost nekog svojstva se može preuzeti i korišćenjem izraza kao npr. <%= *beanName.getPropName*() %>. Oba nacina konvertuju vrednosti svojstva u String i ubacuju vrednost u tenutni implicitni *out* objekat.

Ako je potrebno preuzeti vrednost nekog svojstva bez konvertovanja i ubacivanja vrednosti u out objekat, mora se koristiti skriptlet kao npr.
<% Object o = *beanName*.get*PropName*(); %>

<jsp:setProperty/> etiketa

Svojstva zrna mogu biti postavljena ili modifikovana korišćenjem ove etikete. Etiketa zahteva vrednosti tri atributa: *name*, *property* i *value*.
Sintaksa:
<jsp:setProperty name="beanname" property="propertyName"/>

Atribut *name* kao i kod <jsp:getProperty/> akcije mora da odgovara atributu *id* standardne akcije <jsp:useBean>. Atribut *property* odreduje kom svojstvu ce biti postavljena vrednost. Standardna akcija <jsp:setProperty/> ima dva dodatna opciona atributa, *parm* i *value*.

<jsp:setPropertyname="beanName"
property="propName" value="string constant"/>

<jsp:setProperty name="*beanName*" property="*propName*"
param="*paramName*"/>

Postoji varijacija <jsp:setProperty/> etikete koja nam omogucava da umesto korišćenja atributa *value* koristimo samo atribut *property*.

<jsp:setProperty name="beanName" property=" * " /> - ovo znaci da ce se za svako svojstvo u primljenom zahtevu, naci odgovarajuce svojstvo u zrnu i pozvati odgovarajuca set etiketa koja ce vrednost svojstva u primljenom zahtevu

postaviti za vrednost svojstva zrna na osnovu istog atrinuta *name*. Za korišćenje ove varijacije <jsp:setProperty/> etikete važno je da se nazivi elemenata forme poklapaju sa nazivima svojstva zrna.

Postoji i drugi nacin da se u JSP stranici postave vrednosti svojstva zrna a to je pomocu skriptleta. Npr. <% *beanName.setPropName(value)*; %>

6.9.3 Primer klase zrno i korišćenje zrna u JSP stranici

Kao primer klase zrno naveden je deo koda klasa Korisnik.java koja se koristi u aplikaciji opisanoj u odeljku 7.0.

```
package bean;
public class Korisnik {
    // Svojstva
        private int id;
        private String ime;
        private String prezime;
        ...
        public int getId() {
            return id; }
        public void setId(int id) {
            this.id = id; }
    public String getIme() {
            return ime; }
        public void setIme(String ime) {
            this.ime = ime;}
        public void setPrezime(String prezime) {
            this.prezime = prezime;}
        public String getPrezime() {
            return prezime; }
        ...
}
```

Zrno "Korisnik" se može upotrebiti u više jsp stranica u situacijama kada je potrebno uneti, odnosno prikazati podatke o korisniku. Tako se na primer prilikom logovanja korisnika svojstvima zrna se mogu dodeliti vrednosti ulogovanog korisnika korišćenjem <jsp:setProperty/> etikete.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
...
<sql:query  var="korisnik"
dataSource="jdbc:derby://localhost:1527/c:/DerbyDatabase/test,org.apache
.derby.jdbc.ClientDriver,admin,admin">
SELECT ime,prezime FROM KORISNIK
   WHERE KORISNICKO_IME = ? AND LOZINKA = ?
<sql:param value="${param.userName}" />
<sql:param value="${param.password}" />
</sql:query>
...
<% Row oneRow = (Row) korisnik.firstElement();  %>
    <jsp:useBean id="validUser" scope="session"
     class="bean.Korisnik" >
    <jsp:setProperty name="validUser" property="id"
      value='<%= oneRow.getInt("ID") %>' />
    <jsp:setProperty name="validUser" property="ime"
      value='<%= oneRow.getString("IME") %>' />
    <jsp:setProperty name="validUser" property="prezime"
      value='<%= oneRow.getString("PREZIME") %>' />
...
```

Na stranici gde je potrebno prikazati podatke o ulogovanom korisniku podaci se mogu prikazati na sledeci nacin:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
```

```
...
<jsp:useBean id="validUser" scope="session" class="bean.Korisnik"/>
<html>
<head>
<title>Moj nalog</title>
</head>
<body>
<fieldset>
<legend>Lične informacije</legend>
<p><label>Ime: </label>
<input type="text" value="<%=
StringFormat.toHTMLString(validUser.getIme())%>" name="ime" id="ime"
disabled="false"> </p>
<p><label>Prezime: </label>
<input type="text" value="<%=
StringFormat.toHTMLString(validUser.getPrezime())%>" name="prezime"
id="prezime" disabled="false"></p>
</fieldset>
</body>
</html>
```

6.10 Etikete prilagodene korisniku (Custom Tags)

Standardne JSP etikete (tags) koje pozivaju operacije nad zrnima i otpremaju obavljeni zahtev u mnogome pojednostavljuju razvoj i održavanje JSP stranica. Pored toga, JSP takode omogucava mehanizam za ugradivanje drugih tipova dinamickih funkcionalnosti u okviru etiketa prilagodenih korisniku. Ove etikete predstavljaju dodatak JSP jeziku i najčešće su distribuirane u formi tag biblioteke koja definiše skup etiketa prilagodenih korisniku i sadrži objekte koje implementiraju etikete.

Etikete prilagodene korisniku su JSP elementi definisani od strane korisnika (programer). Ukoliko JSP sadrži ove etikete, one se prevodi u Servlet,

gde se konvertuju u operacije nad objektom koji se zove tag handler. Kontejner pokreće ove operacije kada se izvršava JSP stranica.

Etikete prilagodene korisniku imaju bogat skup karakteristika:

- Imaju pristup svim objektime dostupnim u JSP stranici
- Mogu da modifikuju odgovor generisan od pozvane stranice
- Mogu da komuniciraju medusobno
- Mogu da budu ugnježdene omogucavajuci kompleksne interakcije u okviru JSP stranice.

(Detaljnije o kastomizovanim akcijama i kako se one kreiraju može se naci u [2] i [14]).

6.11 JSP Expression Language

Expression Language (EL) je programski jezik koji se koristi u konstrukciji izraza. EL omogucava pristup Java komponentama (zrnima) u okviru JSP-a. EL koriste JSP programeri za pristup i upotrebu podataka aplikacije bez korišćenja Java koda. Od JSP 2.0 verzije, EL se zvanicno koristi u okviru JSP etiketa kako bi se odvojio Java kod od JSP koda i omogućio lakši pristup Java komponentama. Glavni razlog uvodenja EL bio je da se obezbedi kreiranje prezentacionog nivoa JSP stranice bez oslanjanja na skript elemente. Svi EL izrazi su ogradeni u okviru $ { ... } notacije. Izrazi se izračunavaju pre nego što se pocne sa obradom ostalog dela JSP stranice. EL izraze u JSP stranicu možemo koristiti na dva nacina, odnosno možemo ih smestiti u okviru:

- šablon podataka
- atributa JSP akcija

EL izrazi se najčešće upotrebljavaju za predstavljanje teksta u okviru šablon podataka. Na primer, deo JSP koda `<b>Godina ima ${11 + 1} meseci.</b>` rezultirace u `<b> Godina ima 12 meseci.</b>` HTML kod. Takode, EL izrazi mogu da se smeste u okviru atributa odredenih etiketa, JSTL etiketa, standardnih JSP akcija i akcija kreiranih od strane programera. Npr. EL izraz u okviru JSTL etikete upotrebljaba se na sledeci nacin:

```
<c:if test="${godine > 18}">
<b>Punoletan!</b><br/>
</c:if>
```

JSP Expression language definiše skup implicitnih objekata koji se mogu koristiti bez bilo kakvog eksplicitnog kodiranja i deklaracije. Postoji ukupno 11 implicitnih objekata koji nisu isti kao implicitni objekti dostupni za JSP skript elemente. Jedini EL implicitni objekat koji je isti kao i za skript elemente je pageContext.
(Detaljnije o EL implicitnim objektima može se naci u [1] i [14]).

6.12 Deljenje podataka izmedu JSP stranica

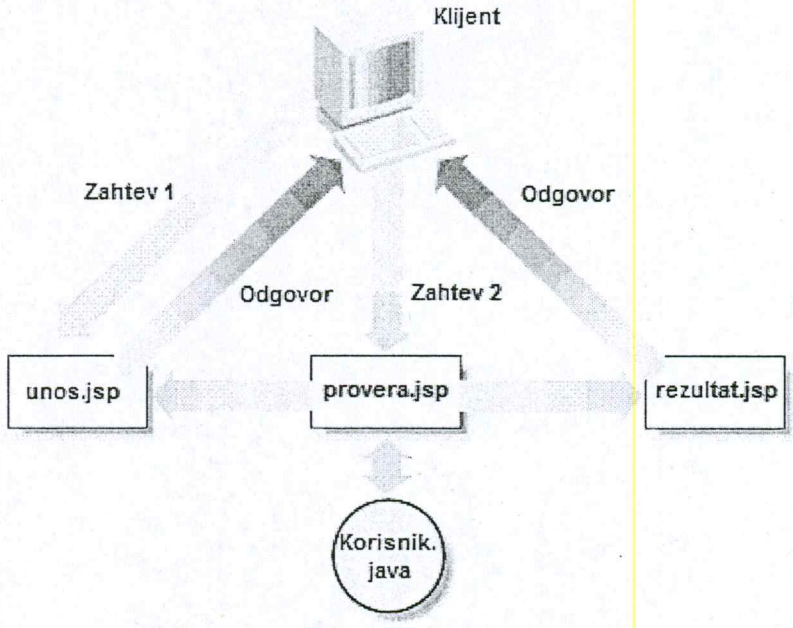
Svaka aplikacija se uglavnom sastoji od više stranica i cesto je slucaj da nekoliko njih ili sve zahtevaju pristup istim informacijama i resursima na serverskoj strani. Kada se koristi više stranica za obradu istog zahteva, cest je slucaj da jedna stranica uzima podatke koje korisnik zahteva a druga ih prikazuje korisniku što zahteva prosleđivanje podataka sa jedne stanice na drugu.

Kao što je već rečeno, jedna od osnovnih karakteristika JSP tehnologije jeste što omogućava odvajanje obrade zahteva, biznis logiku i prezentaciju korišćenjem Model-View-Controller (MVC) modela. Uloge u MVC modelu mogu biti dodeljene razlicitim tipovima komponenata na serverskoj strani.

Korišćenjem različitih JSP stranica kao Controller i View znači da će više od jedne stranice da ucestvuje u obradi zahteva. Kako bi to bilo ostvarivo, mora da se omoguci: prosledivanje kontrole i prosledivanje podataka sa jedne na drugu stranicu.
Korišćenjem MVC modela aplikaciju možemo kategorizovati na sledeci nacin:

- Prikaz forme za korisnicki unos (prezentacija)
- Potvrdivanje unosa (obrada zahteva i biznis logika)
- Prikaz rezultata (prezentacija)

Ako bismo uzeli tri razlicite JSP stranice za svaki aspekt, ovako kreiran MVC model bi izgledao kao na slici 5.



Slika 5: *JSP stranice u MVC modelu*

Opis slike 5:

- *unos.jsp* stranica prikazuje formu za unos.
- Korisnik prosleduje ovu formu stranici provera.jsp za potvrdivanje unosa.
- *provera.jsp* obrađuje zahtev korišćenjem zrna *Korisnik.java* i prosleduje kontrolu ili stranici *unos.jsp* (ako je unos nekorektan) ili stranici *rezultat.jsp* (ako je unos korektan).
- Ako je unos korektan *rezultat.jsp* stranica prikazuje odgovor.

U ovom slucaju *Korisnik.java* predstavljea Model, *provera.jsp* predstavlja Controller, a *unos.jsp* i *rezultat.jsp* predstavljaju View.

6.12.1 Prosledivanje kontrole sa jedne na drugu JSP stranicu

Za prosledivanje kontrole sa jedne stranice na drugu u zavisnosti od rezultata provere unosa koristi se <jsp:forward> akcija:

```
<jsp:forward page="rezultat.jsp" />
```

Ova akcija zaustavlja obradu jedne stranice i pocinje obradu druge (ciljne) stranice navedene u page atributu. Ciljna stranica ima pristup svim informacijama o zahtevu, ukljucujuci sve parametre zahteva. Takode, moguce je dodati dodatne parametre zahteva kada se prosleđuje kontrola drugoj stranici korišćenjem jedne ili više ugnježdene <jsp:param> akcije.

```
<jsp:forward page="unos.jsp" >
<jsp:param name="poruka" value="Neispravno uneto ime" />
</jsp:forward>
```

6.12.2 Prosledivanje podataka sa jedne na drugu JSP stranicu

JSP obezbeduje razlicite opsege (scopes) za deljenje objekata izmedu stranica, zahteva i korisnika. Ovi opsezi definišu koliko dugo ce objekat biti dostupan i da li je dostupan jednom korisniku ili svim korisnicima aplikacije. Razlikujemo sledece opsege:

- Page
- Request
- Session
- Application

Objekti smešteni u podrazumevani opseg, page scope su dostupni akcionim elementima i skriptletima u okviru trenutne stranice. Request scope se koristi za objekte koji moraju da budu dostupni svim stranicama koji obraduju isti zahtev. Session scope se koristi za objekte koje dele više zahteva od istog korisnika. I application scope se koristi za objekte koje dele svi korisnici aplikacije.
Primer <jsp:useBean> akcije sa scope atributom:

```
<jsp:useBean id="validUser" scope="request" class="bean.Korisnik" />
```

<jsp:useBean> akcija traži zrno sa imenom navedenim u id atributu u navedenom opsegu (scope). Ako vec postoji, koristi se to zrno. Ako ne može da pronade zrno, kreira novu instancu klase navedene u class atributu i cini je dostupnom u navedenoj vrednosti navedenog opsega, odnosno scope atributa.

6.12.3 Deljenje podataka sesija i aplikacionih podataka

Kada internet čitač pošalje zahtev za neki web resurs, web server obradi zahtev i vrati nazad odgovor u internet citac. Server potom zaboravlja da se ova transakcija dogodila, tako da kada isti internet čitač pošalje novi zahtev, web server ne zna da je ovaj zahtev povezan sa prethodnim. Ovo je u potpunosti u redu ako su u pitanju staticki fajlovi, problem se javlja sa interaktivnim web aplikacijama.

Rešenje za ovaj problem jeste da se dozvoli serveru da pošalje deo informacija u internet citac, a internet citac ce potom te informacije ukljuciti u svaki naredni zahtev. Ovaj deo informacija, koji se naziva session ID se koristi od strane servera da prepozna skup zahteva sa istog web citaca koji su povezani, odnosno koji su deo iste sesije. Sesija pocinje kada internet citac prvi put zahteva JSP stranicu u konkretnoj aplikaciji. Sesija može biti završena eksplicitno od strane aplikacije, ili od strane JSP kontejnera posle odredenog perioda korisnicke neaktivnosti. Server koristi session ID da poveže zahtev sa session objektom, privremenim prostorom u memoriji gde Servleti i JSP stranice mogu smeštati informacije. JSP sakriva veliki deo detalja vezanih za to kako je session ID prenet, kako je objekat sesije kreiran i kako mu se pristupa, pružajuci nam session scope za obradivanje podataka sesije na nivou apstrakcije. Informacije sacuvane u sesija scope su dostupne svim stranicama koje se zahtevaju od strane istog internet citaca tokom trajanja sesije.

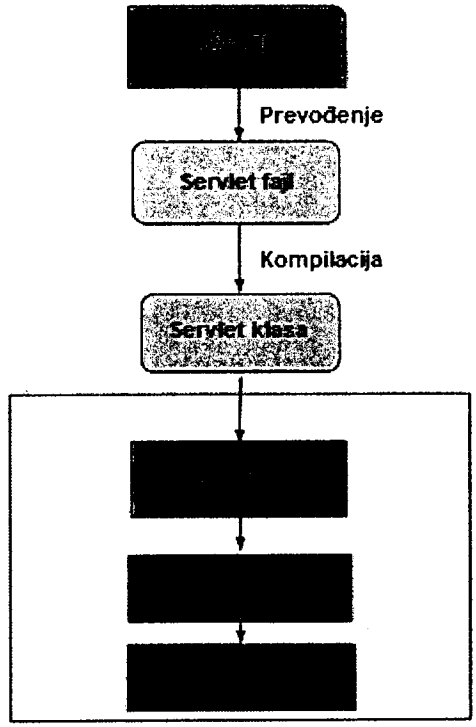
Medutim, neke informacije su potrebne vecem broju stranica nezavisno od toga ko je trenutni korisnik. JSP omogucava pristup ovom tipu deljenih informacija kroz drugi opseg, application scope. Informacijama sacuvanim u application scope od strane jedne JSP stranice, može se kasnije pristupiti nekom drugom stranicom, iako su ove dve stranice zahtevane od razlicitih korisnika. Informacije koje se najčešće dele kroz application scope su konekcije sa bazom podataka, informacije o trenutno ulogovanom korisniku,..

6.13 Životni ciklus JSP - a

JSP stranica obrađuje zahteve kao Servlet. Zbog toga su životni ciklus i mnoge sposobnosti JSP strana (posebno dinamički aspekti) određene Java Servlet tehnologijom. Kada je zahtev za JSP stranicu prosleđen, Web kontejner prvo proverava da li je Servlet JSP stranice stariji od same JSP stranice. Ukoliko je Servlet stariji, web kontejner prevodi JSP stranicu u Servlet klasu i kompajlira klasu.

Faze zivotnog ciklusa JSP-a su sledece:

1. Faza prevođenja
2. Faza kompajliranja
3. Faza izvršenja



Slika 6: *Životni ciklus JSP stranice*

Faza prevođenja i kompajliranja:

JSP fajlovi se prevode u Servlet kod, pa se zatim kompajliraju. Kontejner obavlja ove poslove automatski. JSP elementi se drugačije tretiraju:

- Direktive se koriste da kontrolišu prevođenje i izvršenje od strane Web kontejnera.
- Script elementi se ubacuju u Servletske klase JSP-a.
- Elementi forme <jsp:xxx .../> se konvertuju u metode koje pozivaju komponente zrna.

Faza izvršavanja je sacinjena od tri metode u JSP Servletu koji odgovaraju init(), service() i destroy() metodama regularnog Servleta:

- jspInit()
- _jspService()
- jspDestroy()

7.0 Opis aplikacije

Kao primer prakticne primene JSP-a u izradi web aplikacije, uz rad je priložena aplikacija koja obuhvta nekoliko slucajeva obrade korisnickih zahteva. Aplikacija za glavni cilj ima da predstavi osnovne koncepte JSP okruženja i kako se pomocu njih mogu obraditi neki zahtevi korisnika.

Aplikacija predstavlja simulaciju internet oglasa za zapošljavanje, nudeci korisniku pregled aktivnih oglasa i mogucnost da na jednostavan i brz nacin konkuriše na odabrani.

Pregled oglasa dostupan je svim korisnicima aplikacije, dok detaljan opis konkursa i mogucnost konkurisanja zahteva ragistraciju, odnosno logovanje korisnika kako bi se vodila evidencija za administratora aplikacije, a i samog korisnika. U okviru aplikacije vrši se kontrola isteka oglasa. Korisniku se u pregledu ogasa na vizuelan nacin ukazuje na konkurse koji isticu za manje od tri dana što korisniku može olakšati u vidu prioriteta detaljnog pregleda konkursa. Na administratorskoj stranici takodje postoji kontrola isteka konkursa koja funkcioniše po sličnom principu. Razlika je u tome što se vizuelno ukazuje na datum koji je istekao i za takve konkurse administratoru se otvara dodatna opcija njihive deaktivacije. Deaktiviranje konkursa na administratorskoj stranici direktno utice na pregled oglasa koji se prikazuje korisniku jer se neaktivni konkursi ne prikazuju.

Takode, aplikacijom je obuhvacen i administratorski deo u okviru kojeg administrator ima mogucnost pregleda svih (aktivnih i neaktivnih) oglasa i registrovanih korisnika uz mogucnost deaktiviranja isteklih oglasa, odnosno brisanja korisnickog naloga.

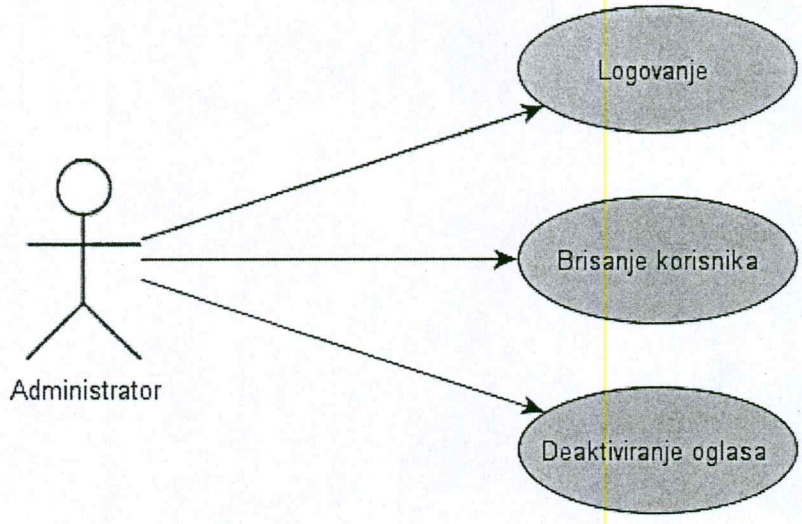
7.1. Model slučajeva korišćenja

Aplikacija obuhvata sledeće slučajeve korišćenja:

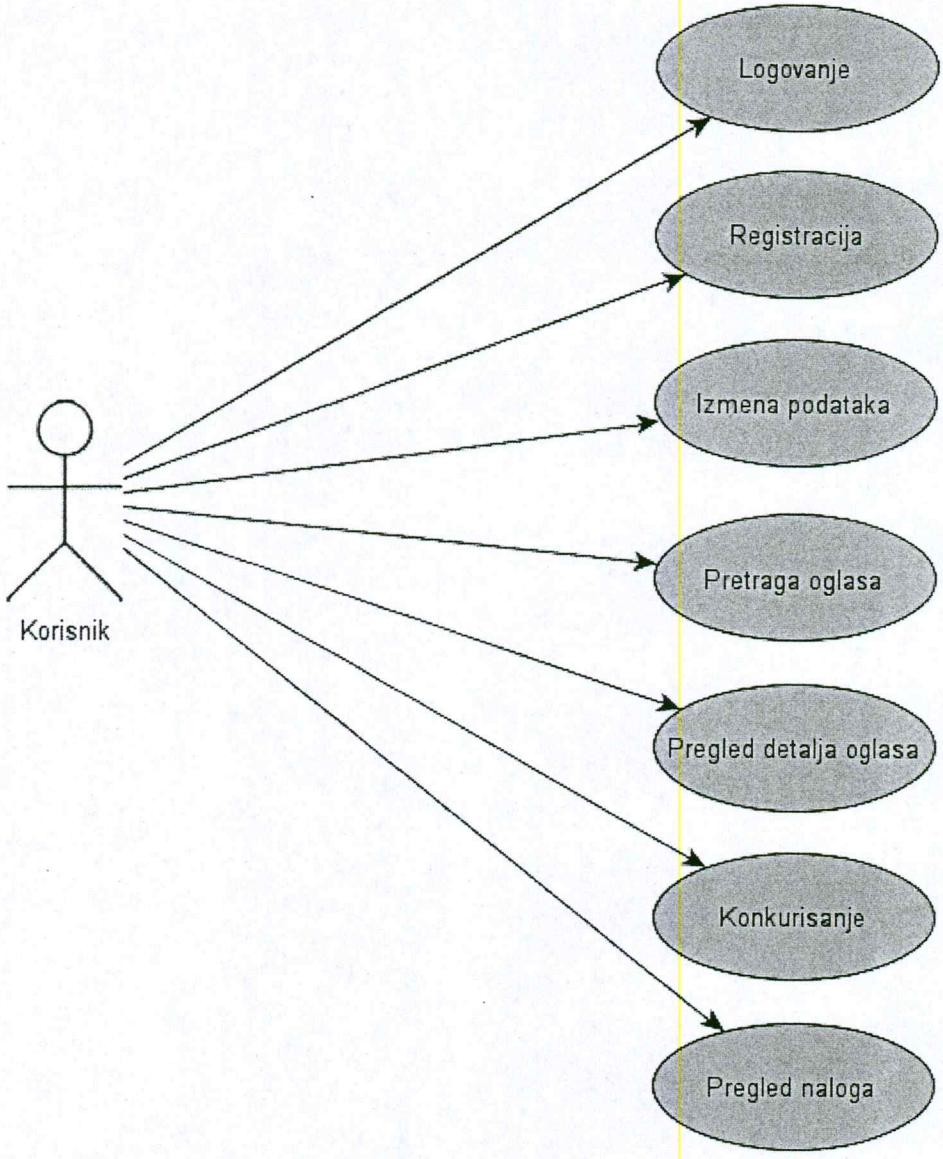
- Logovanje
- Registracija
- Izmena podataka (o korisniku)

- Pretraga oglasa
- Detaljan pregled oglasa
- Konkurisanje
- Pregled naloga
- Brisanje korisnika
- Deaktiviranje oglas

Model slučajeva korišćenja se može predstaviti preko sledećih dijagrama slučajeva korišćenja:



Slika 7: Dijagram slučajeva korišćenja (administrator)



Slika 8: Dijagram slučajeva korišćenja (korisnik)

SK1: Slucaj korišcenja – Logovanje korisnika

Naziv SK: Logovanje korisnika

Aktori SK: Korisnik

Preduslov: Sistem je uključen i prikazuje stranicu za logovanje.

Osnovni scenario SK

1. Korisnik unosi korisničko ime i lozinku.
2. Korisnik proverava da li je uneo sve potrebne podatke.
3. Korisnik poziva sistem da ga uloguje.
4. Sistem izvršava validaciju.

5. Sistem prikazuje korisniku glavnu stranicu sa potvrdom o ulogovanom korisniku.

Alternativni scenario

5.1. Ukoliko korisnik nije uneo neki od podataka ili korisnik sa takvim podacima ne postoji sistem ce obavestiti korisnika o grešci.

5.2. Ukoliko se loguje administrator, sistem ga usmerava na administratorsku stranicu.

SK2: Slucaj korišcenja – Registracija

Naziv SK: Registracija

Aktori SK: Korisnik

Preduslov: Sistem je ukljucen i prikazuje stranicu za registrovanje.

Osnovni scenario SK

1. Korisnik unosi podatke.
2. Korisnik proverava da li je uneo sve potrebne podatke.
3. Korisnik poziva sistem da zapamti podatke.
4. Sistem izvršava validaciju.
5. Sistem pamti novog korisnika.
6. Sistem prikazuje korisniku poruku o uspešnosti unosa novog korisnika.

Alternativni scenario

5.1. Ukoliko korisnik nije uneo neko od obaveznih polja sistem obaveštava korisnika.

6.1. Ukoliko korisnik sa takvim podacima vec postoji sistem obaveštava korisnika.

SK3: Slučaj korišcenja – Izmena podataka (o korisniku)

Naziv SK: Izmena podataka

Aktori SK: Korisnik

Preduslov: Sistem je ukljucen i korisnik je ulogovan pod svojom šifrom. Sistem prikazuje stranicu za prikaz podataka o korisniku.

Osnovni scenario SK

1. Korisnik poziva sistem da mu omoguci izmenu polja u kojima se nalaze podci o korisniku.

2. Korisnik menja podatke.
3. Korisnik poziva sistem da zapamti izmene.
4. Sistem ažurira podatke o korisniku.
5. Sistem prikazuje izmenjene podatke i poruku da su podaci o korisniku uspešno izmenjeni.

SK4: Slucaj korišcenja – Pretraga oglasa

Naziv SK: Pretraga oglasa

Aktori SK: Korisnik

Preduslov: Sistem je ukljucen i prikazuje glavnu stranicu. Korisnik nije odabrao nijedan kriterijum pretrage.

Osnovni scenario SK

1. Korisnik poziva sistem da prikaže sve aktivne oglase iz baze podataka.
2. Sistem nalazi sve aktivne oglase u bazi podataka.
3. Sistem prikazuje sve aktivne oglase.

Alternativni scenario

1.1. Korisnik je odabrao jedan ili sva tri kriterijuma pretrage a potom pozvao sistem da prikaže sve aktivne oglase koji zadovoljavaju kriterijume pretrage.

3.1. Sistem nije našao nijedan aktivan oglas koji odgovara kriterijumima pretrage i korisniku ispisuje odgovarajucu poruku.

SK5: Slučaj korišcenja – Pregled detalja oglasa

Naziv SK: Pregled detalja oglasa

Aktori SK: Korisnik

Preduslov: Sistem je ukljucen i prikazuje glavnu stranicu. Korisnik je izvršio pretragu aktivnih oglasa.

Osnovni scenario SK

1. Korisnik poziva sistem da prikaže detalje konkretnog oglasa.
2. Sistem proverava da li je korisnik ulogovan.
3. Korisnik nije ulogovan, sistem ga usmerava na stranicu za logovanje.
4. Korisnik se loguje.
5. Sistem prikazuje stranicu sa detaljima konkretnog oglasa.

Alternativni scenario

3.1. Korisnik je ulogovan, sistem prikazuje stranicu sa detaljima konkretnog oglasa.

5.1. Ne postoje detalji za konkretan oglas, sistem ispisuje korisniku odgovarajucu poruku.

SK6: Slučaj korišcenja – Konkurisanje

Naziv SK: Konkurisanje

Aktori SK: Korisnik

Preduslov: Sistem je ukljucen i prikazuje stranicu sa detaljima konkretnog oglasa. Korisnik je ulogovan.

Osnovni scenario SK

1. Korisnik poziva sistem da evidentira njegovo konkurisanje na konkretan oglas.
2. Sistem prikazuje podatke o korisniku i radnom mestu na koje korisnik konkuriše.
3. Korisnik proverava ispravnost podataka.
4. Korisnik obaveštava sistem da su podaci ispravni i da može da pošalje podatke.
5. Sistem evidentira da je korisnik konkurisao na konkretan konkurs.
6. Sistem usmerava korisnika na glavnu stranicu.
7. Sistem ispisuje poruku korisniku o uspešnosti konkurisanja na konkretno radno mesto.

Alternativni scenario

3.1. Korisnik menja licne podatke.

.

SK7: Slucaj korišcenja – Pregled naloga

Naziv SK: Pregled naloga

Aktori SK: Korisnik

Preduslov: Sistem je ukljucen i prikazuje pocetnu stranicu.

Osnovni scenario SK

1. Korisnik poziva sistem da prikaže podatke o korisniku i konkursima na koje je korisnik konkurisao.
2. Korisnik nije ulogovan, sistem ga usmerava na stranicu za logovanje

3. Korisnik se loguje.
4. Sistem prikazuje podatke o ulogovanom korisniku i konkursima na koje je korisnik konkurisao.

Alternativni scenario

2.1. Korisnik je ulogovan, sistem prikazuje podatke o ulogovanom korisniku i konkursima na koje je korisnik konkurisao.

SK8: Slučaj korišcenja – Brisanje korisnika

Naziv SK: Brisanje korisnika

Aktori SK: Administrator

Preduslov: Sistem je ukljucen i prikazuje pocetnu administratorsku stranicu. Administrator je ulogovan.

Osnovni scenario SK

1. Administrator poziva sistem da prikaže spisak svih registrovanih korisnika.
2. Sistem prikazuje spisak svih registrovanih korisnika.
3. Administrator poziva sistema da obriše konkretan korisnicki nalog.
4. Sistem briše korisnika iz baze podataka.
5. Sistem prikazuje ažuriran spisak korisnika.

Alternativni scenario

1.1. Administrator je uneo jedan ili sva tri kriterijuma pretrage a potom pozvao sistem da vrati sve registrovane korisnike.

2.1. Nije pronaden nijedan korisnik koji zadovoljava kriterijume pretrage, sistem ispisuje odgovarajucu poruku.

SK9: Slučaj korišcenja – Deaktiviranje oglasa

Naziv SK: Deaktiviranje oglasa

Aktori SK: Administrator

Preduslov: Sistem je ukljucen i prikazuje pocetnu administratorsku stranicu. Administrator je ulogovan.

Osnovni scenario SK

1. Administrator poziva sistem da prikaže spisak svih oglasa iz baze podataka.
2. Sistem prikazuje spisak svih oglasa.

3. Administrator poziva sistem da promeni status konkretnog oglasa.
4. Sistem postavlja status konkretnog oglasa na „Neaktivan"(N).
5. Sistem prikazuje ažuriran spisak oglasa.

Alternativni scenario

1.1. Administrator je uneo jedan ili sva tri kriterijuma pretrage a potom pozvao sistem da vrati sve oglase iz baze podataka.

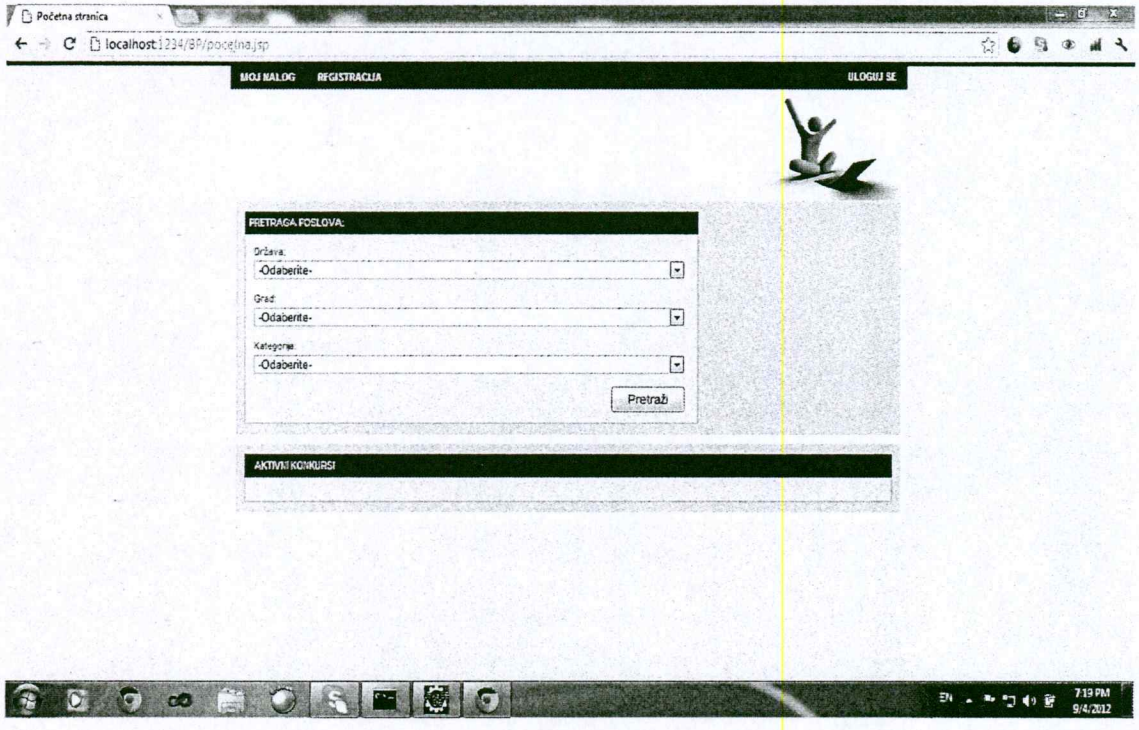
2.1. Nije pronaden nijedan oglas koji zadovoljava kriterijume pretrage, sistem ispisuje odgovarajucu poruku.

7.2. Implementacija

Aplikacija je kreirana korišćenjem MVC Model 2 arhitekture koja predstavlja obrazac projektovanja Java web aplikacija koji razdvaja prikaz podataka od logike. Za implementaciju klijent-server troslojne MVC arhitekture korišcene su JSP i Servlet tehnologije gde sloj View predstavljaju JSP stranice, ulogu Controller-a imaju Servleti, dok ulogu Modela imaju Java zrna. Baza podataka implementirana je u okviru Apache Derby baze podataka. U izradi aplikacije korišćen je Apache Tomcat web server a kao razvojno okruženje Eclipse 4.2.0.

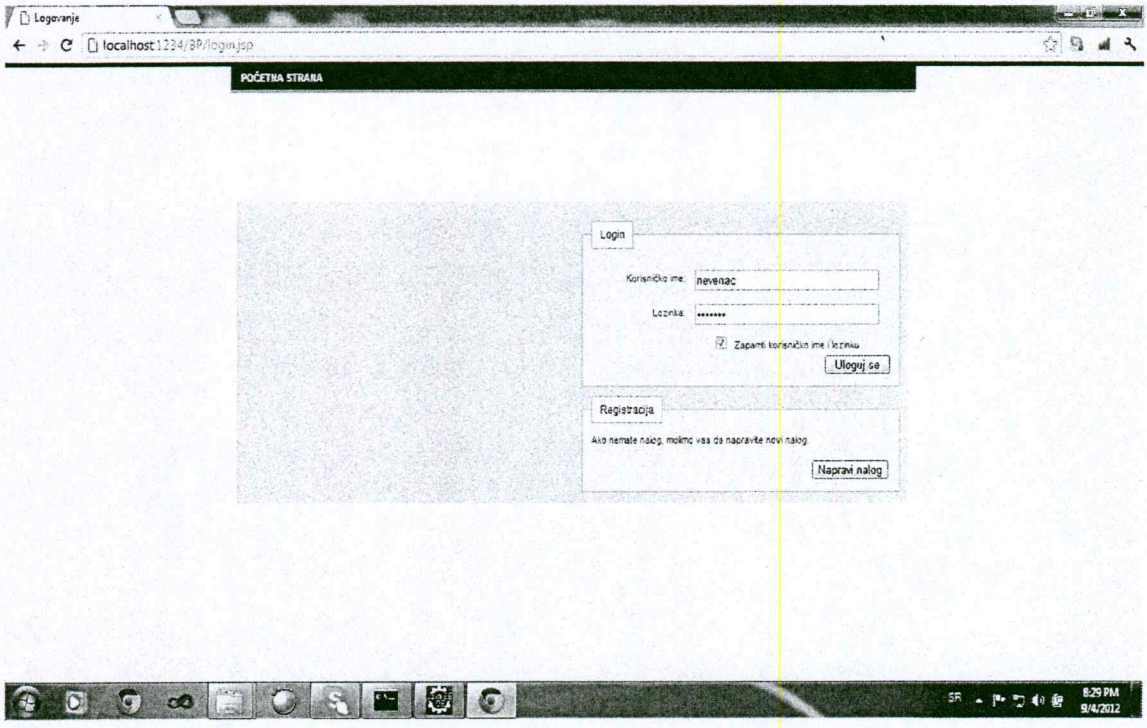
7.3. Korisnički interfejs

Kada korisnik zahteva početnu stranicu, u internet čitaču otvara se jsp stranica prikazana na slici 9.

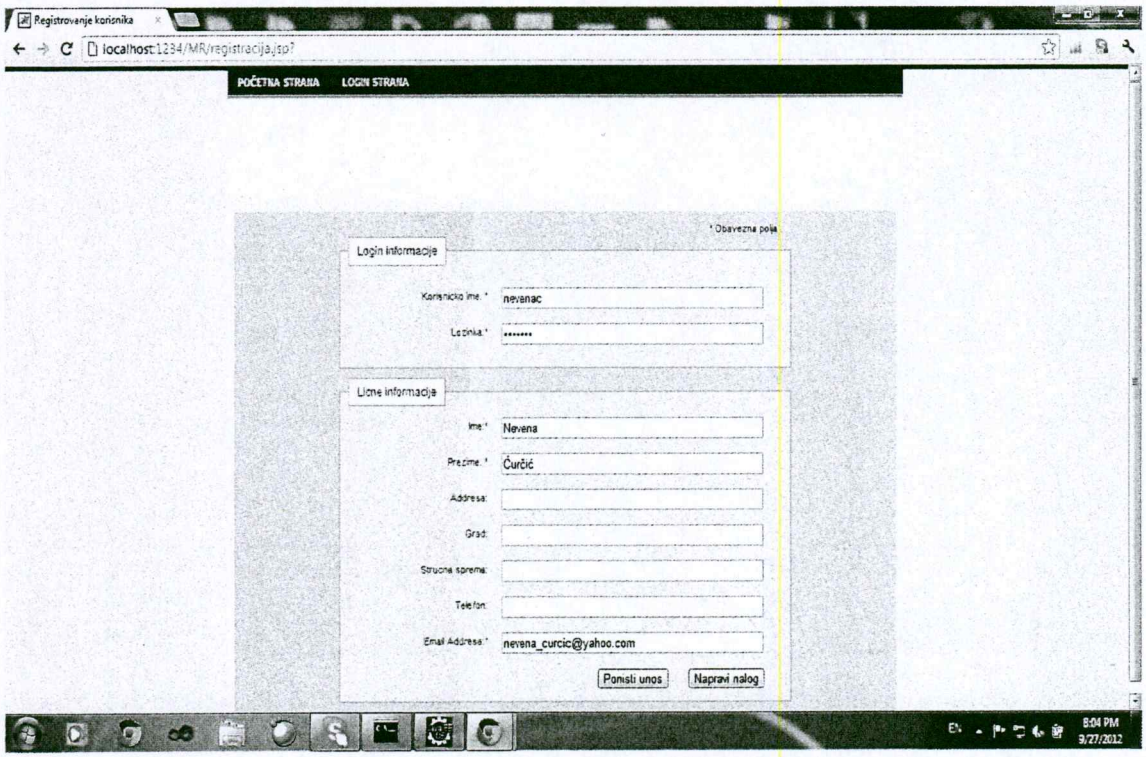


Slika 9: *pocetna.jsp stranica*

Dolaskom na početnu stranicu, korisnik ima mogućnost da se odabirom opcije "ULOGUJ SE" uloguje ukoliko ima registrovan nalog (slika 10), odnosno da se odabirom opcije "REGISTRACIJA" registruje ukoliko nema registrovan nalog (slika 11). Korisnik ima mogućnost registrovanja i na stranici za logovanje odabirom opcije "Napravi nalog". Po završetku logovanja korisnik se vraća na početnu stranicu.

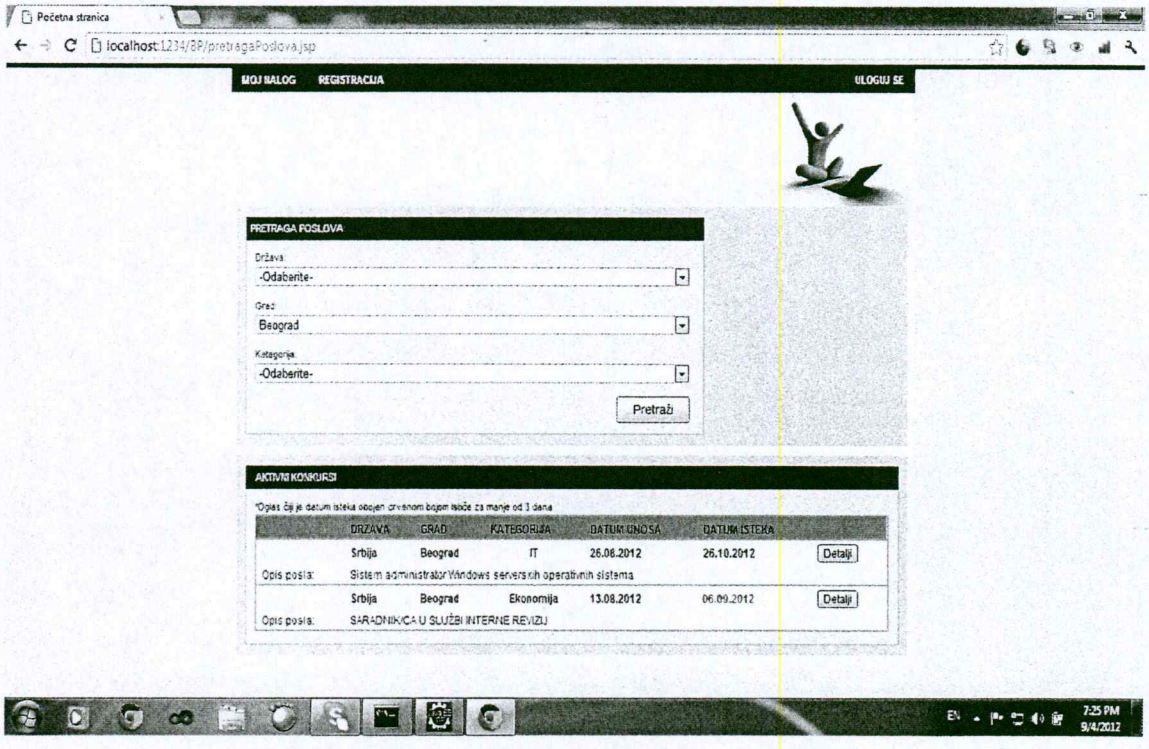


Slika 10: *Logovanje korisnika*



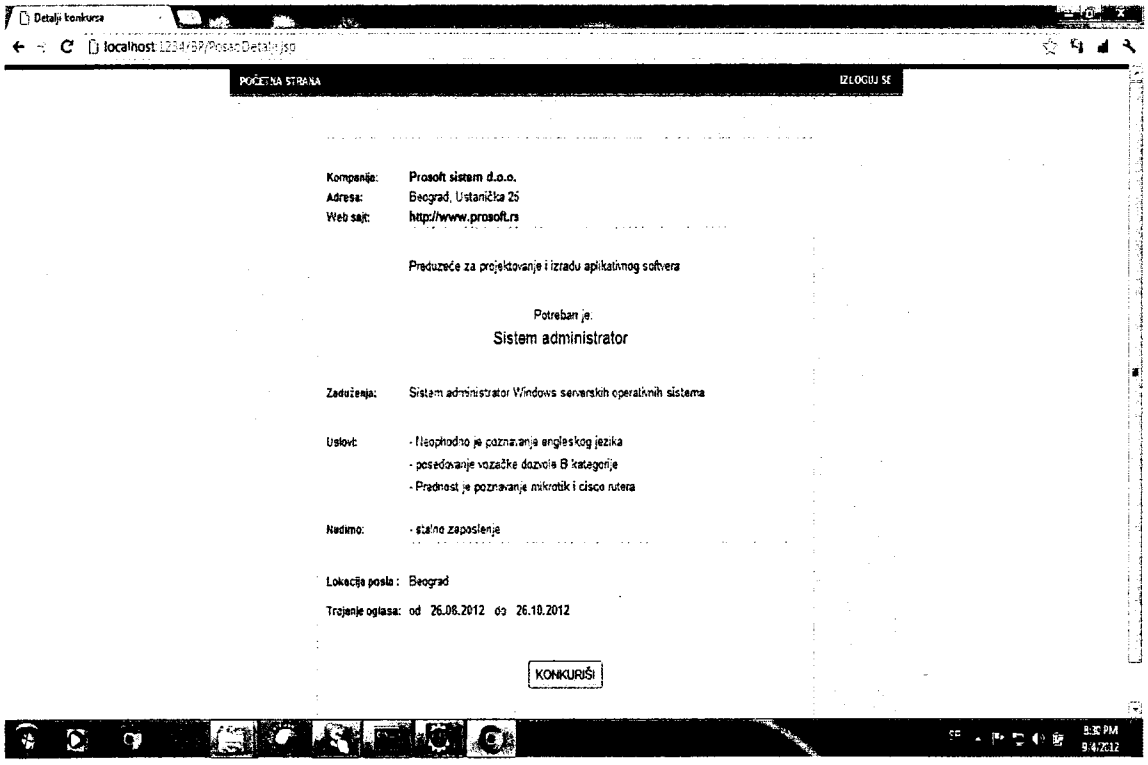
Slika 11: *Registracija korisnika*

Na početnoj stranici korisnik može da izvrši pretragu aktivnih oglasa. Na raspolaganju ima tri opciona kriterijuma pretrage. Rezultat pretrage, u slučaju da je korisnik odabrao da vidi pregled svih oglasa iz Beograda, prikazan je na slici 12.



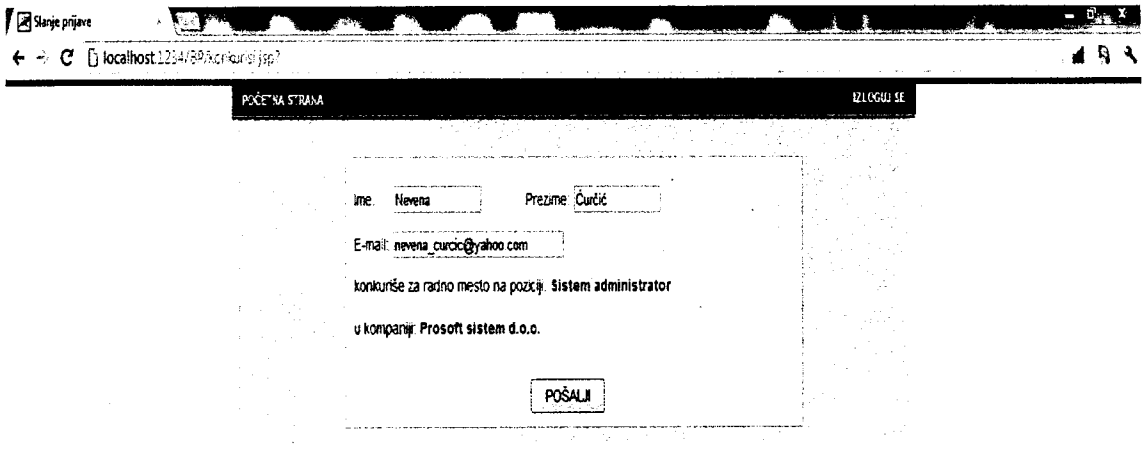
Slika 12: *Pretraga aktivnih oglasa*

Kao rezultat pretrage, korisniku se otvara opcija da pogleda detalje konkretnog konkursa klikom na dugme "Detalji". Prikaz detalja konkursa zahteva da korisnik prethodno bude ulogovan. Ukoliko korisnik nije ulogovan, odabirom ove opcije korisnik biva usmeren na stranicu za logovanje, a po završetku logovanja korisniku se otvara prethodno zahtevana stranica "Detalji". Prikaz ove stranice za slučaj da je korisnik odabrao prvi oglas predstavljen je na slici 13.



Slika 13: *Detaljan prikaz konkursa*

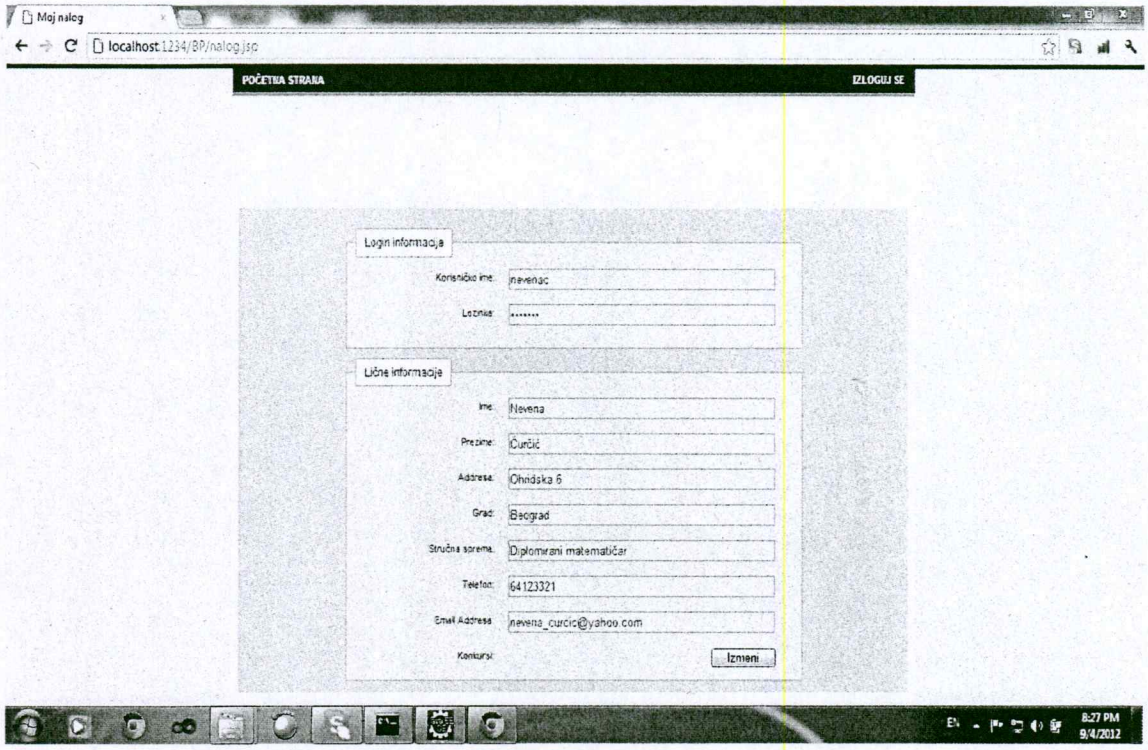
Ukoliko korisnik želi da konkuriše na konkurs, može to učiniti klikom na dugme "konkuriši" pri čemu će se otvoriti stranica prikazana na slici 14 kao krajnja potvrda da korisnik želi da konkuriše na konkurs.



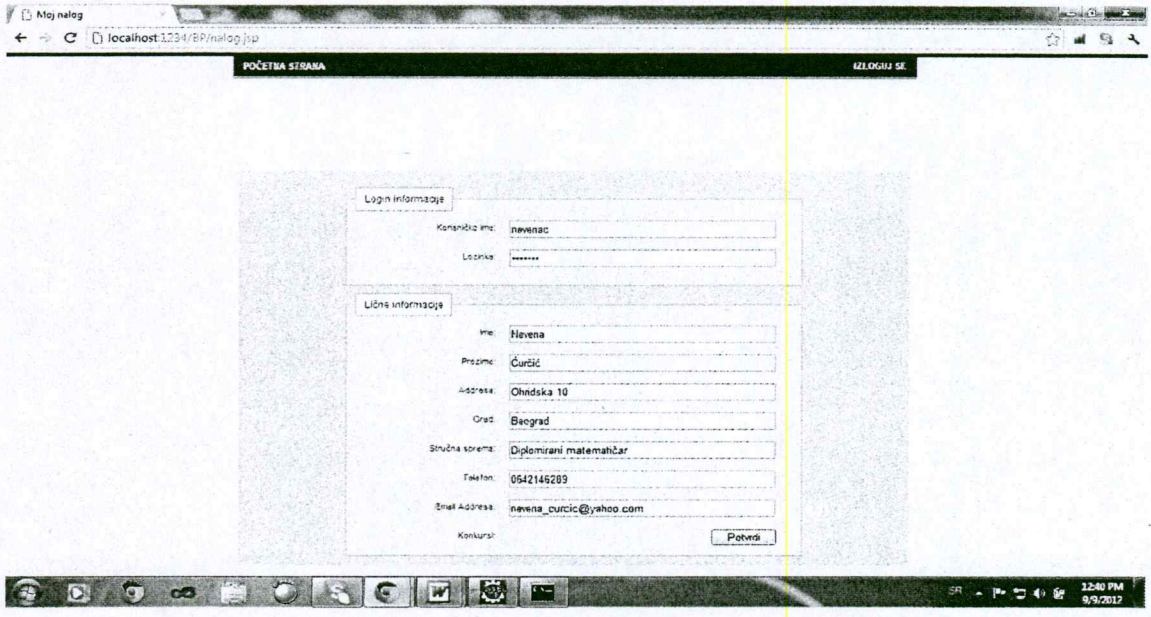
Slika 14: *Provera podataka*

Na ovoj stranici korisnik može proveriti tačnost informacija koje se šalju i eventualno ih izmeniti. Slanjem se podaci šalju u bazu podataka pri čemu se ažuriraju podaci sa konkursima na koje je korisnik konkurisao.

Na početnoj stranici korisnik može da pogleda svoj nalog (slika 15), što takođe zahteva da korisnik prethodno bude ulogovan. Na stranici nalog.jsp korisnik može da izmeni svoje podatke (slika 16). Potvrdom na dugme "Promeni" podaci se ažuriraju u bazi podataka.



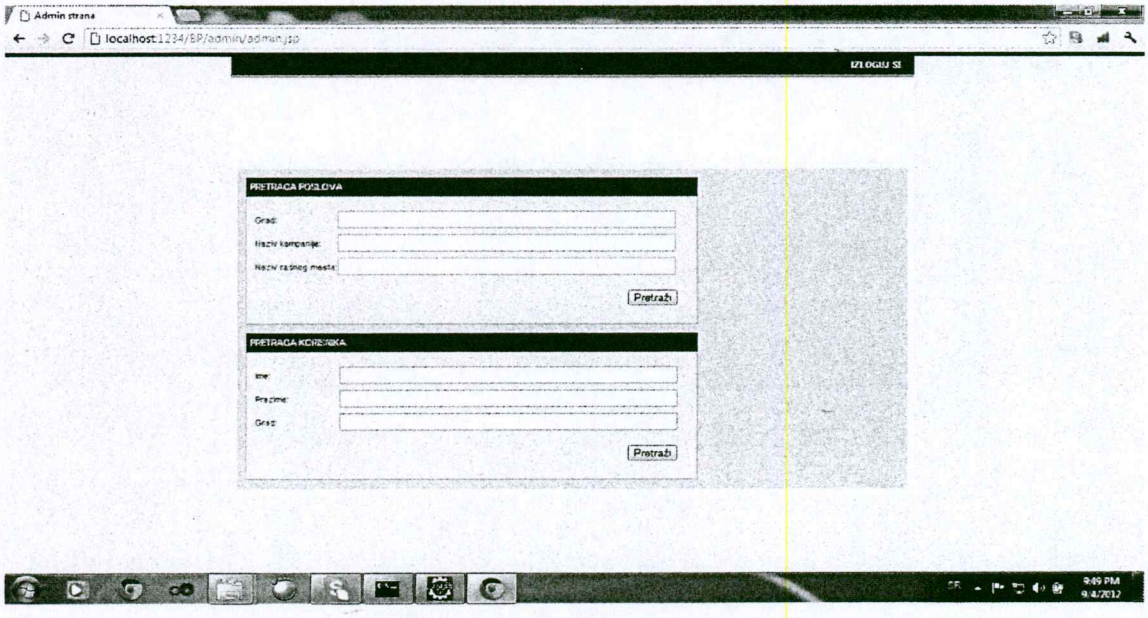
Slika 15: *Nalog korisnika*



Slika 16: *Izmena podataka*

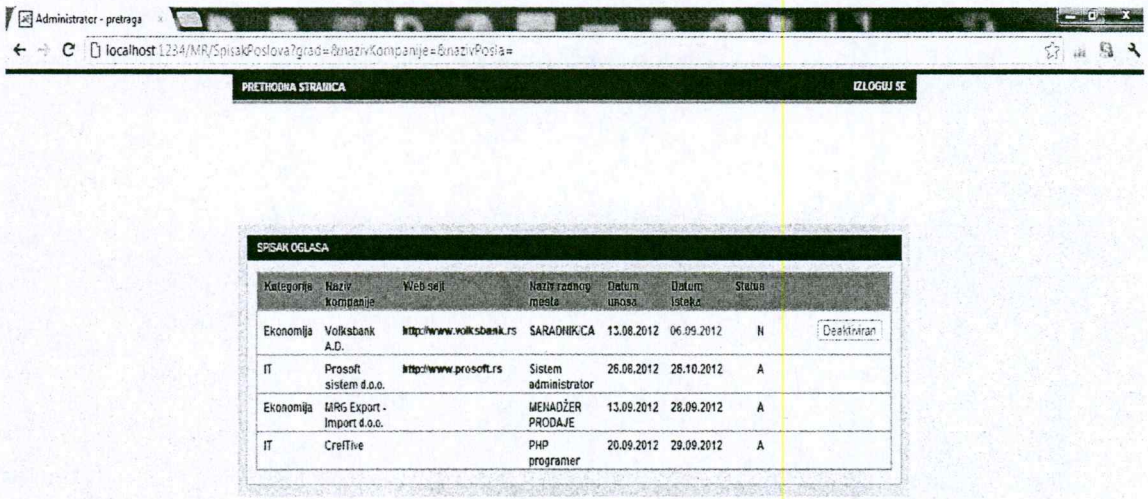
Administratorski deo

Na početnoj stranici (slika 17) administrator ima mogućnost pretrage svih objavljenih oglasa i registrovanih korisnika uz tri opciona kriterijuma pretrage.

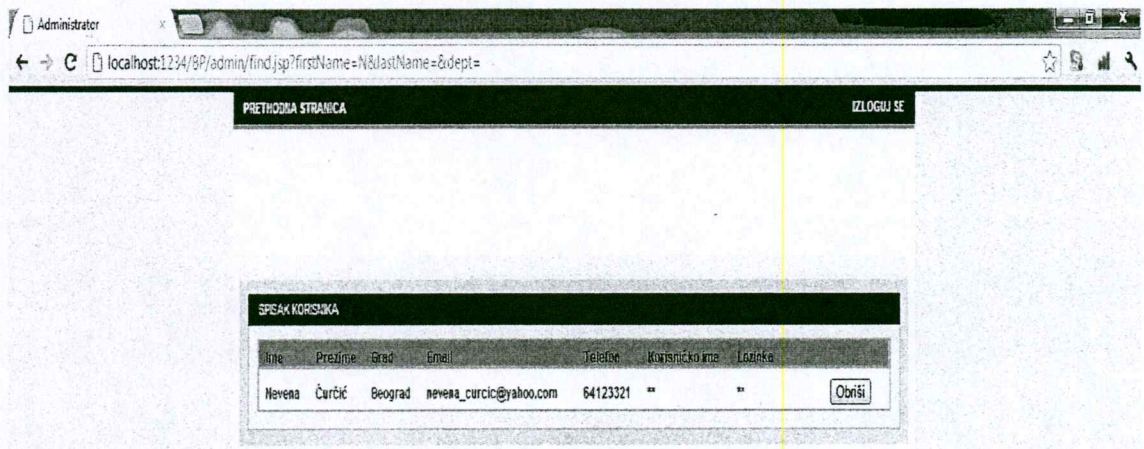


Slika 17: *Administratorska početna stranica*

Pretragom oglasa otvara se stranica (slika 18) sa pregledom svih oglasa koji zadovoljavaju kriterijume pretrage ukoliko je neki od njih naveden gde administrator ima uvid u sve objavljene konkurse, konkurse koje je deaktivirao jer su istekli i konkurse koje treba deaktivirati. Pretragom registrovanih korisnika otvara se stranica (slika 19) sa pregledom svih registrovanih korisnika koji zadovoljavaju kriterijume pretrage gde administrator ima uvid u sve registrovane korisnike sa mogućnošću brisanja korisničkog naloga.



Slika 18: *Administratorski deo – pegled oglasa*



Slika 19: *Administratorski deo – pegled registrovanih korisnika*

8.0 Zakljucak

Jedan od glavnih razloga zašto se JavaServer Pages tehnologija razvila u ono što je danas jeste velika tehnicka potreba za jednostavno projektovanim dinamickim web aplikacijma. JavaServer Pages omogucava web programerima da na efikasan nacin kreiraju upravo takve aplikacije.

U ovom radu su predstavljni osnovni koncepti JSP tehnologije kao samostalne tehnologije u kreiranju web stranica i kao takva ona predstavlja veoma mocan alat. Medutim, iako JSP može u potpunosti da zameni Servlet u smislu funkcionalnosti, za kreiranje složenih web aplikacija preporucije se kombinovanje JSP tehnologije sa drugim Java tehnologijama kao što su Servleti. Uz pun pristup Java EE platformi, pokrenute od strane servera na siguran nacin, mogucnosti JSP aplikacija su neogranicene. Mnogi proizvodaci serverskih aplikacija vec uveliko koriste JSP u izradi svojih proizvoda, a rezultat toga jeste razvoj složenih poslovnih web aplikacija.

9.0 Literatura

[1] Vivek Chopra, Sing Li, Rupert Jones, Jon Eaves, John T. Bell: *Beginning JavaServer Pages, Wiley 2005.*

[2] Hans Bergsten: *JavaServer Pages, 3rd Edition, O'Reilly 2003.*

[3] Larne Pekowsky: *JavaServer Pages, 2nd Edition, Addison Wesley 2003.*

[4] Simon Brown, Sam Dalton, Daniel Jepp, Dave Johnson, Sing Li, Matt Raible: Pro JSP 2, *4th Edition, Apress 2005.*

[5] Duane K. Fields,Mark A. Kolb, Shawn Bayern: *Web Development with JavaServer Pages, 2nd Edition, Manning 2002.*

[6] *Java Enterprise Best Practices*: The O'Reilly Java authoers

[7] James Goodwill: *Pure JSP -- Java Server Pages: A Code-Intensive Premium Reference, Sams 2000.*

[8] Jennifer Ball, Debbie Carson, Ian Evans, Scott Fordin, Kim Haase, Eric Jendrock: *Java EE 6 Tutorial, Sun Microsystems, 2012.*

[9] Marty Hall, Larry Brown: *Core Servlets and JavaServer Pages, 2nd Edition, Sun Microsystems, 2003.*

[10] Martin Keen, Rafael Coutinho, Sylvi Lippmann, Salvatore Sollami, Sundaragopal Venkatraman, Steve Baber, Henry Cui, Craig Fleming: *Developing Web Applications using JavaServer Pages and Servlets, IBM Corp, 2012.*

Internet izvor

[11] http://www.oracle.com

[12] http://en.wikipedia.org

[13] http://java.sun.com/products/jsp/syntax/1.2/syntaxref12.html

[14] http://java.sun.com/j2ee/tutorial/1_3-fcs/index.html

[15] http://www.jsptut.com/

[16] http://www.roseindia.net/