KORIŠTENJE NAPREDNIH JAVA TEHNOLOGIJA PRI IZRADI WEB APLIKACIJA

Adnan Ramakić
Univerzitet u Bihaću, Pape Ivana Pavla II 2/2, adnan.ramakic@unbi.ba

Ključne riječi: Java tehnologije, objektno – relaciono mapiranje (ORM), Hibernate, JavaServer Faces (JSF), fejsleti, baza podataka.

SAŽETAK:

U radu se razmatraju i opisuju napredne Java tehnologije za izradu web aplikacija. Na početku je opisan programski jezik Java. Prilikom izrade web aplikacija pomoću Java tehnologija koriste se različite komponente i okviri (Framework), naravno u kombinaciji sa bazom podataka. U vezi s tim opisan je jedan od najpoznatijih Java okvira pod imenom Hibernate. Radi se o ORM (object-relational mapping) biblioteci za Java programski jezik koji pruža okvir za mapiranje objektno-orijentiranih domena modela u tradicionalne relacione baze podataka. Predstavljajući jedno od najfleksibilnijih i najmoćnijih objektno/relacionih rješenja na tržištu Hibernate se brine o mapiranju iz Java klasa u tabele baze podataka i iz Java tipova podataka u SQL tipove podataka. Nakon toga opisana je JavaServer Faces (JSF) tehnologija. Radi se o tehnologiji koja utvrđuje standarde za izradu server-side korisničkih interfejsa (sučelja) Java baziranih web aplikacija. Uz JSF opisani su i Java fejsleti (Java Facelets). Radi se o moćnom, ali laganom jeziku deklaracije stranica(page declaration language) koji se koristi za izgradnju JSF pogleda (JSF views) koristeći HTML stilske templejte (HTML styletemplate)i za izgradnju komponentnih stabala. Na kraju je predstavljen primjer web aplikacije koja objedinjuje sve navedene tehnologije u kombinaciji sa MySQL bazom podataka, a omogućuje CRUD (create, read, update, delete) funkcionalnostiautoriziranim korisnicima.

1.UVOD

Web aplikacije danas predstavljaju neminovnost koja se ne može izbjeći i koja se naširoko koristi na Internetu. Postavlja se pitanje šta one predstavljaju i o čemu se tu radi. Aplikacije na webu su programi dizajnirani za upotrebu u potpunosti unutar web preglednika. Upotrebom ovih aplikacija mogu se stvarati dokumenti, uređivati fotografije i slušati muzika, bez potrebe za instaliranjem kompliciranih softvera. U današnje vrijeme, web-lokacije mogu obavljati dinamičke funkcije koje se očekuju od aplikacija za radnu površinu na računaru. Te robusne web-lokacije nazivaju se web aplikacijama ili, kratko, aplikacije[2].
Web aplikacije su programska rješenja kojima se pristupa putem web preglednika koristeći Internet ili Intranet. Vrtoglavi rast i razvoj web aplikacije trebaju zahvaliti činjenici da su dostupne u bilo koje vrijeme s bilo kojeg mjesta, s računara,mobilnih telefona, tableta i sl. Osim toga, web poslovne programe nije potrebno periodički nadograđivati na svakom računaru s kojeg im se pristupa, jer im se pristupa identično kao i ostalim web stranicama, s bilo kojeg računara putem web preglednika[3]. Ako se koriste usluge kao što su Gmail ili Google karte, već se upotrebljavaju web aplikacije.

2.TEHNOLOGIJE KOJE SE KORISTE PRI IZRADI WEB APLIKACIJA

2.1. Java programski jezik

Java programski jezik je objektno-orijentirani jezik opšte namjene posebno pogodan za razvoj konkurentnih, mrežnih i distribuiranih programa. Razvoj jezika pod inicijalnim nazivom Oak počeo je 1991. godine, a vodio ga je James Gosling iz kompanije Sun Microsystems. Prva verzija jezika pod nazivom Java objavljena je 1995. godine. Jedna od najvažnijih osobina programskog jezika Java jeste nezavisnost od platforme. Ova osobina omogućava da se programi pisani u ovom programskom jeziku mogu kompajlirati na jednoj računarskoj platformi, a izvršavati na različitim platformama. Početkom 2010. godine kompanija Oracle preuzima Sun Microsystems, a samim tim i Javu. Java interpreter, kompajler kao i brojni razvojni alati grupisani su u JDK (Java Development Kit). U okviru JRE (Java Runtime Environment) se nalazi interpeter, ne uključujući kompajler i razvojne alate. Kompanija Oracle redovno objavljuje JDK i JRE pakete za različite platforme kao što su Linux, Windows, Solaris itd[1].

2.2. ORM i Hibernate

ORM (Object-relational mapping) je način postizanja trajnosti (engl.persistence) objekata unutar relacijskih baza podataka. On djeluje kao posrednik na podatkovnoj razini aplikacije, automatski preslikavajući podatke iz objekata u bazu i natrag, na zahtjev aplikacije. Iako sam pojam posrednika za sobom povlači problem smanjenih performansi (jer se komunikacija ne odvija direktno), prednosti koje ovakav pristup donosi (u pogledu vremena kodiranja, količine i kvalitete kôda, održavanja) definitivno nadjačavaju bilo kakve probleme s performansama. Aplikacija komunicira s ORM-om preko njegovog vlastitog API-ja (Application programming interface) i pripadnih mu objekata te je potpuno odvojena od komunikacije koja se odvija između ORM-a i baze podataka preko SQL-a i DBC-a. Takav princip se naziva transparentnost (engl. Transparency). Hibernate je jedno od najpopularnijih ORM rješenja današnjice, dizajniran specifično za Javu. Projekt je započeo računarski inženjer Gavin King 2001. godine, s namjerom da nadomjesti dotadašnje popularno rješenje za očuvanje objekata, CMP Entity Beans, koje nije bilo dovoljno moćano ni fleksibilano da zadovolji potrebe osim najjednostavnijih aplikacija. Projekt je započet kao neprofitni, open-source projekt, koji se prije svega razvijao s obzirom na zahtjeve korisnika i tvrtki koje su ga upotrebljavale[4].
Hibernate predstavlja objektno/relacioni perzistentni i upitni servis visokih performansi. Radi se o jednom o najfleksibilnijih i najmoćnijih rješenja na tržištu koji se brine za mapiranje iz Java klasa u tabele baza podataka i iz Java tipova podataka u SQL (engl. Structured Query Language) tipove podataka. On pruža upite za podatke i povratne sadržaje koji uvelike skraćuju vrijeme razvoja aplikacija. Glavni cilj Hibernate-a je oslobađanje programera od perzistentno baziranog programiranja za 95% zadataka, eliminirajući potrebu za ručnom obradom podataka koristeći SQL i JDBC (engl. Java Database Connectivity) [5].

2.3. JavaServer Faces (JSF) tehnologija

Razvijena kroz Java Community Process, pod JSR -314, JSF tehnologija utvrđuje standarde za izgradnju server-side korisničkih interfejsa. JavaServer Faces tehnologija uključuje:

- Skup API-jaza predstavljanje UI komponenti i upravljanje njihovim stanjima, rukovanje događajima(engl. handling events), validaciju unosa, definiranje navigacije stranica i podršku za internacionalizaciju i dostupnost,
- JSP(JavaServer Pages) biblioteku oznaka za predstavljanje JSF interfejsa unutar JSP stranice.

Dizajnirana da bude fleksibilna, JSF tehnologija iskorištava postojeće, standardne UI i web-tier koncepte bez ograničavanja razvojnih programera na određene markup jezike, protokole ili klijentske uređaje. UI komponentne klase uključene u JSF tehnologiju enkapsuliraju komponent

funkcionalnosti, a ne klijent specifične prezentacije, čime se omogućuje JSF UI komponentama da budu renderirane na različitim klijentskim uređajima. Kombinirajući UI komponent funkcionalnosti sa proizvoljnim prikazivačima(engl. renderers), koji definiraju renderiranje atributa za specifičnu UI komponentu, razvojni programeri mogu konstruirati prilagođene oznake (engl. tags) za pojediniklijentski uređaj. JSF arhitektura jasno definira razdvojenost između aplikacijske logike i prezentacije čineći jednostavnim spajanje prezentacijskog sloja sa kôdom aplikacije.Ovakav dizajn omogućuje svakom članu razvojnog tima koji radi na web aplikaciji da se fokusira na svoj dio zadatka, a također pruža i programski modul za povezivanje svih tih dijelova zajedno[6].

2.4.Java fejsleti (Java Facelets)

Izraz fejsleti (engl. Facelets) odnosi se na deklaracijski jezik pregleda za JSF tehnologiju. JSP tehnologija se prvotno koristila kao prezentacijska tehnologija za JSF, međutim JSP nije podržavao sve nove pogodnosti koje je pružao JSF u Java EE 6 platformi. S toga se JSP tehnologija smatra zastarjelom prezentacijskom tehnologijom za JSF, te se danas najčešće koriste fejsleti. Fejsleti su dio JSF specifikacije i trenutno preferirana prezentacijska tehnologija za izgradnju JSF baziranih aplikacija[7].
Fejsleti su moćan, ali lagan jezik deklaracije stranica (engl. page declaration language) koji se koristi za izgradnju JSF pogleda (engl. JSF views) koristeći HTML stilske templejte (engl. HTML style template) i za izgradnju komponentnih stabala.Pogodnosti fejsleta uključuju sljedeće:

- Korištenje XHTML-a za kreiranje web stranica,
- podršku za fejslet biblioteku oznaka u dodacima na JSF i JSTL biblioteku oznaka,
- podršku za ekspresni jezik (engl. Expression Language – EL),
- templejting za komponente i stranice.

Prednosti fejsleta za velike razvojne projekte uključuju sljedeće:

- Podršku za ponovno korištenje koda kroz templejting i kompozitne komponente,
- funkcionalno proširivanje komponenti i drugih server-side objekata kroz prilagodbu(engl. customization),
- brže vrijeme kompajliranja,
- kompajliranje-vrijeme EL validacija,
- visokoperformansno renderiranje.

Ukratko, korištenje fejsleta smanjuje vrijeme i trud koji je potreban uložiti na razvoj i implementaciju. Fejslet pogledi (engl. Facelets views) su obično kreirani kao XHTML stranice. JSF tehnologija podržava različite biblioteke oznaka za dodavanje komponenti na web stranicu. Za podršku JSF mehanizma biblioteke oznaka, fejsleti koriste XML imenske prostore deklaracije[8].

3.PRIMJER JEDNOSTAVNE WEB APLIKACIJE REALIZIRANE KORIŠTENJEM NAVEDENIH TEHNOLOGIJA

U ovom dijelu rada predstavljen je primjer web aplikacije koja objedinjuje sve navedene tehnologije u kombinaciji sa MySQL bazom podataka, a omogućuje CRUD (create, read, update, delete) funkcionalnosti autoriziranim korisnicima. CRUD funkcije predstavljaju četiri osnovne funkcije prilikom rada sa perzistentno pohranjenim podacima.Radi se o aplikaciji koja omogućuje autoriziranim korisnicima, sa odgovarajućim korisničkim imenom i šifrom, da pristupe bazi podatka koja sadrži podatke o određenim osobama. Nad tim podacima se mogu vršiti gore navedene funkcionalnosti, tj. autorizirani korisnik je u mogućnosti da dodaje nove, pregleda, mijenja ili briše postojeće podatke. Podaci su prikazani u tabelarnom obliku. Za tu namjenu poslužila je tabela iz biblioteke komponenti PrimeFaces, namijenjenoj posebno za JSF. Pored tabele iz ove biblioteke korišteni su stilski meniji i dugmadi, paneli, dijalozi, uključujući i login dijalog na početnoj stranici (*index.xhtml*). Podaci o autoriziranim korisnicima također se čuvaju u bazi podataka. U nastavku je opisan detaljniji način kreiranja same web aplikacije. Prije svega kreirana je MySQL baza podataka sa dvije tabele, koja će čuvati unesene podatke i nad kojim će se vršiti manipulacija. Jedna tabela je

namijenjena za korisnike, koja će čuvati podatke o korisnicima. Ta tabela se sastoji od sljedećih atributa: *ime*, *prezime*, *username*, *password.* Druga tabela je tabela namijenjena čuvanju podataka o određenim osobama(sastoji se od atributa kao što su ime, prezime, adresa, mobitel, telefon itd.). U ovom slučaju nije konkretno namijenjeno čuvanju nekih određenih osoba neke organizacije, nego je tu prikazano kao primjer. Naravno kod praktične upotrebe čuvat će se podaci o osobama neke organizacije, firme itd. Bitno je napomenuti još jedanput da se ovdje radi samo o primjeru rada sa podacima u bazi podataka putem web aplikacije, te se ne treba nužno vezati za sadržaj baze podataka. U praktičnoj implementaciji niti se ne mora raditi o osobama, tu može biti neka druga tabela koja će čuvati podatke o automobilima, računarskoj opremi itd.Nakon što je kreirana baza podataka, potrebno je kreirati web aplikaciju korištenjem nekih od Javinih razvojnih okruženja, kao što je NetBeans, Eclipse, IntelliJ IDEA ili neki slični. U ovom slučaju web aplikacija je kreirana sa NetBeans razvojnim okruženjem. On sadrži unutar sebe već okvire (engl. Framework) za rad za Hibernate-om i JSF-om, koje je prilikom postupka kreiranja web aplikacije potrebno dodati. Sam postupak kreiranja web aplikacije putem NetBeans-a ogleda se u tome da se iz meni-a odabere sekcija *File*, pod kojom se odabere *New Project*, iz ponuđenih kategorija sa lijeve strane selektira se opcija *Java Web*, te tip projekta *Web Application*. Nakon toga da se odgovarajuće ime aplikacije, odabere se server s kojim će se raditi, u ovom slučaju to je GlassFish server, iako se može koristiti i neki drugi, kao što je Apache Tomcat.Nakon toga potrebno je još uključiti i okvire za rad sa aplikacijom, što su u ovom slučaju Hibernate i JavaServer Faces (JSF). Ovako kreirana web aplikacija nema posebnih funkcionalnosti, do ispisivanja pozdravne poruke („Hello from Facelets“). Naravno sada je ovu novokreiranu aplikaciju potrebno oblikovati tako da radi ono što programer želi.Potrebno je aplikaciju povezati sa bazom podataka, izvršiti mapiranje podataka pomoću Hibernate-a, te kreirati klase u Javi koje će omogućiti manipulaciju sa podacima u bazi. Osnovna stranica *index.xhtml*, koja je početno kreirana prilikom kreiranja aplikacije ima zadatak samo da ispisuje gore navedenu pozdravnu poruku, te je potrebno izvršiti njenu modifikaciju, a u ovom slučaju kreirati i dodatnu stranicu koja će se otvarati poslije početne.Kôd početne stranice nakon što je kreirana izgleda kao u nastavku:

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html">
<h:head>
<title>Facelet Title</title>
</h:head>
<h:body>
     Hello from Facelets
</h:body>
</html>
```

Prilikom uključivanja Hibernate okvira automatski se kreira i Hibernate konfiguracijski fajl (engl.Hibernate configuration file) sa ekstenzijom .cfg.xml (hibernate.cfg.xml). On pruža informacije neophodne za povezivanje sa bazom podataka, a također i podatke o mapiranju domenskih objekata u tabele baze podataka. U ovom slučaju izgleda kao što ilustrira sljedeći kôd:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/zaposleni</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password">adox</property>
<property name="hibernate.show_sql">true</property>
<property name="hibernate.format_sql">true</property>
```

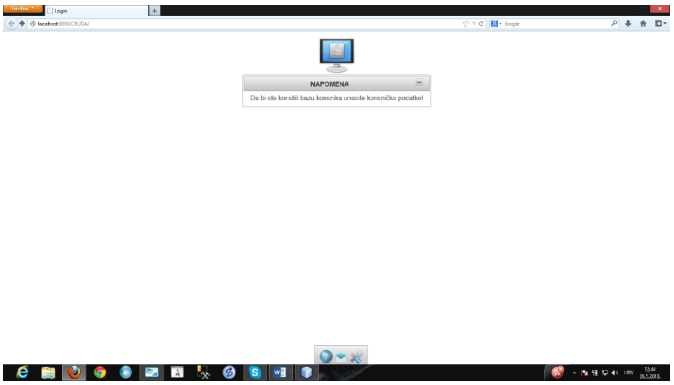
```
<property name="hibernate.current_session_context_class">thread</property>
<mapping resource="com/zaposleni/model/Osoba.hbm.xml"/>
<mapping resource="com/zaposleni/model/Korisnici.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

Pored Hibernate konfiguracijskog fajla, potrebno je kreirati i Hibernate reverse engineering fajl sa ekstenzijom .reveng.xml (hibernate.reveng.xml) i Hibernate Mapping files and POJOs (Plain Old Java Object) baziran na postojećoj bazi podataka.Hibernate reverse engineering fajl omogućuje bolju i veću kontrolu strategije mapiranja baze podataka. Hibernate Mapping files and POJOs se koristi za generiranje fajlova. Preciznije koristi se za generiranje POJO i odgovarajućih fajlova mapiranja za svaku tabelu koja se odabere. Mapirajući fajlovi su ustvari XML fajlovi koji sadrže podatke o tome kako su kolone u tabeli mapirane u polja POJO-a. Da bi se koristio potrebno je kreirati druga dva navedena fajla. Ovdje je bitno spomenuti da se ovi fajlovi kreiraju desnim klikom na navedeni projekat (ime web aplikacije) i iz kategorija sa lijeve strane odabere se Hibernate, a sa desne strane jedan od fajlova koji se želi kreirati. U nastavku je prikazan izgled Hibernate reverse engineering fajla, za primjer ove web aplikacije.

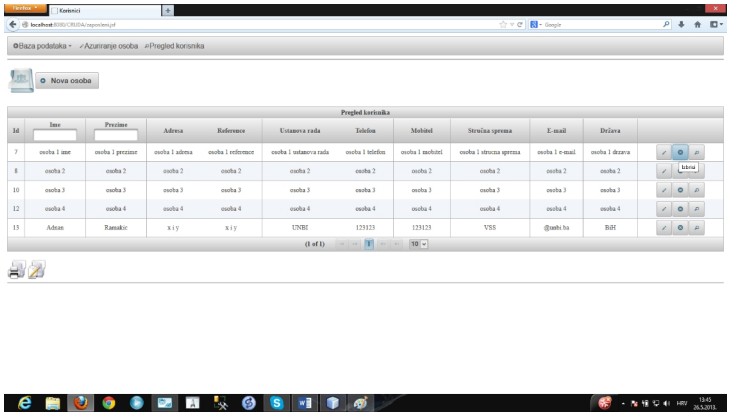
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-reverse-engineering-3.0.dtd">
<hibernate-reverse-engineering>
<schema-selection match-catalog="zaposleni"/>
<table-filter match-name="korisnici"/>
<table-filter match-name="osoba"/>
</hibernate-reverse-engineering>
```

Korištenjem Hibernate Mapping files and POJOs čarobnjaka kreirana su četiri fajla, dva ekstenzije .java i dva ekstenzije hbm.xml (Korisnici.java, Korisnici.hbm.xml i Osoba.java i Osoba.hbm.xml). Kako se može zaključiti iz ovog radi se naravno o tabelama iz baze podataka, koje su mapirane.

Nakon što se završi rad sa Hibernate-om, potrebno je izvršiti modifikaciju početne stranice (index.xhtml) i naravno kreirati drugu stranicu koja će sačinjavati prikaz podataka i dozvoliti manipulaciju nad njima. Za tu namjenu potrebno je kreirati odgovarajuće Java klase. Prije svega potrebno je kreirati tzv. managed bean klase. Tipična JSF aplikacija uključuje jednu ili više navedenih klasa, gdje se svaka može povezati sa komponentama korištenim na određenim stranicama. U ovom slučaju kreirane su tri ovakve klase (KorisnikBean, OsobaBean i LoginBean). KorisnikBean sadrži getter i setter metode, kao i metode za manipulaciju korisnicima. Te metode su metode za pregled, brisanje, izmjenu i dodavanje podataka. Na istim temeljima je i OsobaBean, dok je LoginBean nešto drugačiji, naime on sadrži metodu za provjeru korisnika koji se logiraju u aplikaciju. Provjera se vrši na osnovu podataka koji se nalaze u bazi. U nastavku slijede slike koje ilustriraju web aplikaciju.



Slika 1: Početna stranica (Index.xhtml)



Slika 2: Izgled web aplikacije

4. ZAKLJUČAK

Ukoliko se pogleda današnji Internet onda se može vidjeti da se on dosta promijenio u odnosu na njegov sami početak, gdje su inače dominirale statičke web stranice. Danas to više nije slučaj, i može se reći da je većina današnjih web stranica u biti web aplikacija. Postavlja se pitanje šta su ustvari web aplikacije? U suštini one predstavljaju programe dizajnirane da se koriste u potpunosti unutar web preglednika (Mozilla Firefox, Google Chrome, Internet Explorer i dr.). Njihovim korištenjem korisnici mogu raditi različite stvari, kao što su manipulacija dokumentima, uređivanje fotografija, slušanja muzike, registriranje i prijavu korisnika, bankovne transakcije, online kupovinu, objavljivanje vlastitog sadržaja i mnoge druge stvari, a da im za to nije potrebna nikakva komplikovana instalacija. Tu u suštini leži ljepota web aplikacija. Njihova osnovna prednost, i kojoj ustvari mogu i zahvaliti za ovako brz razvoj, jeste da su dostupne u bilo koje vrijeme i sa bilo kojeg mjesta, i sa različitih uređaja, da li sa računara, mobilnog telefona, tableta i sl. Druga prednost jeste da da ih nije potrebno nadograđivati na svakom uređaju sa kojeg im se pristupa. Primjer najpoznatijih web aplikacija su Gmail i Facebook. Za njihovu izgradnju mogu se koristiti različite tehnologije, od kojih je jedna od poznatijih i korištenjem Java tehnologija. Java tehnologije se danas naširoko koriste na Internetu, i s njima je moguće izgraditi moderne i vizualno atraktivne web aplikacije koje mogu odgovoriti svim zahtjevima korisnika u današnje vrijeme. Pored danas ionako velike raširenosti i upotrebe web aplikacija, u budućnosti će se one još više koristiti, pružajući krajnjem korisniku još ugodnije korištenje, i vizualno naprednija i modernija sučelja.

5. LITERATURA

[1] Zoran Đurić: *Korak u Java svijet, Elektrotehnički fakultet Banja Luka*, 2010.
[2] https://support.google.com/chrome/answer/1050586?hl=hr
[3] http://www.dimedia.hr/web-aplikacije/im-11-sto_su_web_aplikacije
[4] http://java.zemris.fer.hr/seminari/SEM0036419177/Seminar.pdf
[5] http://www.hibernate.org/about
[6] http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html
[7] http://docs.oracle.com/javaee/6/tutorial/doc/giepx.html
[8] http://docs.oracle.com/javaee/6/tutorial/doc/gijtu.html
[9] http://docs.oracle.com/javaee/6/tutorial/doc/bnaqm.html