DigitalHeaven.co WebShop

Igor Štrbac
Smer: Java Development
Generacija: 2017
Datum predaje rada: 10.01.2019
Mentor: prof.Vladimir Bošković

Sadržaj

Uvod

Kako je projekat u celini obiman, akcenat će biti stavljen na profilsku stranu koja je ujedno i najbogatija sadržajem pa samim tim i najkompleksnija uzimajući u obzir i sam dizajn HTML-a. Ostali delovi aplikacije biće opisani u letu, najčešce u delovima rada u kojima je predstavljena neka komponenta profilske strane.

Poglavlje *Korišćene tehnologije* detaljnije opisuje i daje primere tehnologija i alata koji su korišćeni u implementaciji projekta.

Poglavlje *Model baze* opisace model baze koji je korišćen,iz koga je kasnije upotrebom OR mapiranja izveden model biznis logike.

U nastavku sledi poglavlje *Opis implementacije upotrebom Java Server Faces 2.2* u kojoj se detaljno opisuje specifikacija i implementacija korisničkog interfejsa i frontend dela aplikacije.

Nakon poglavlja *Opis implementacije upotrebom Java Server Faces 2.2* sledi poglavlje *Logika aplikacije* u kojem se detaljno opisuje backend logika projekta.

Poglavlje *GWeb xHTML stranice* opisuje komponente korisničkog interfejsa profilske strane kao i detaljniji pregled svih xHMTL stranica.

Pregled korišćenih tehnologija

Java Enterprise Edition

Java Enterprise Edition[1] predstavlja široko korišćenu platformu za serversko programiranje zasnovanu na Java programskom jeziku. Ona nadgrađuje standardnu Javu dodavanjem biblioteka koje omogućuju razvijanje web i enterprise aplikacija koje se izvršavaju na aplikativnom serveru. Kako je Java EE definisana svojom specifikacijom, provajder usluga (kao što je Glassfish[2] aplikativni server) mora da ispuni određene uslove kako bi svoje proizvode mogao da proglasi usaglašenim sa Java EE specifikacijom. Java EE uključuje nekoliko API specifikacija kao što su JDBC[3], RMI[4], servleti, Java Server Pages i druge. Java EE aplikativni server rukuje transakcijama, bezbednošću, skalabilnošću, konkurentnošću i upravljanjem komponenti koje se na njemu izvršavaju, olakšavajući i ubrzavajući razvoj aplikacije.

Glassfish

Glassfish je aplikativni server baziran na Java Enterprise Edition platformi. On ne samo da implementira server koji se izvršava na Java virtuelnoj mašini, nego i Enterprise deo Java specifikacije. Kako je baziran na Javi, Glassfish server se može koristiti na svim operativnim sistemima koji podržavaju i Javu. Neke od glavnih karakteristika Glassfish-a su:

- Podrška za aspekt-orijentisano programiranje
- Podrška za Enterprise JavaBeans 3 i 2.1
- Integracija Hibernate-a
- Podrška za Java Server Pages i Java Servleta
- Balansiranje opterećenja

Servleti

Servlet[5] predstavlja Java klasu u Java EE okruženju koja je u skladu sa Java Servlet API-jem, protokolom po kome Java klasa može da odgovara na zahteve. Servleti nisu vezani za specifičan klijent-server protocol, ali se najčešće koriste sa HTTP protokolom. Da bi se servlet koristio, potreban je Web kontejner. Web kontejner je komponenta Web servera koja upravlja životnim ciklusom servleta.

Filteri

Filter[6] predstavlja Java klasu koja može da izmeni zaglavlje i sadržaj (ili oba) nekog zahteva ili odgovora. Filteri obično ne kreiraju odgovore, za razliku od servleta. Umesto toga oni pružaju funkcionalnost koja se može ‘nakačiti’ na bilo koji web resurs.

Neki od scenarija u kojima su filteri pogodni za upotrebu su: autentifikacija, logovanje, kompresija, enkripcija, itd.

Java Server Faces (JSF)

Java Server Faces[7] predstavlja web framework zasnovan na Model-View-Controller[8] paradigmi i na HTTP zahtev/odgovor principu. JSF framework sadrži jedan master Controller, takozvani Faces servlet koji procesira dolazeće zahteve, učitava odgovarajući View na osnovu parametara zahteva, i isporučuje odgovarajući odgovor klijentu. Slede glavne karakteristike JSF framework-a.

Managed Beans[9] - predstavljaju java klase proglašene kao 'managed' (upravljan), u konfiguracionom fajlu samog framework-a. Da bi klasa bila managed potrebno joj je dodeliti neki od definisanih opsega (scope). JSF 2.2 implementacija definiše 6 opsega,od kojih su najčešće korišćeni:

- *Request* - Životni vek managed objekta je ograničen na dolazni zahtev. JSF proizvodi novu instancu klase za svaki dolazeći zahtev, a na kraju zahteva ih uništava.
- *View* - Životni vek objekta je ograničen na trajanje samo dok smo na toj strani
- *Session* - Životni vek objekta ograničen je na trajanje sesije jednog korisnika. Svaki korisnik poseduje jedinstvenu sesiju, tako da su i objekti sesije jedinstveni za svakog korisnika. Dokle god je sesija aktivna JSF će koristiti objekat asociran sa tom sesijom. Nakon isteka sesije JSF će uništiti sve objekte koji se nalaze u opsegu sesije.
- *Application* - Životni vek objekta vezan je za životni vek same aplikacije. Managed bean-ovi čiji je opseg aplikacija, biće kreirani prilikom podizanja same aplikacije na serveru, a uništeni kada se ugasi sama aplikacija na serveru. Ove objekte dele svi korisnici.

Request, Session i Application opsezi su implementirani kao mape koje sadrže parove ključ/vrednost.

Managed Beans klase najčešće enkapsuliraju model podataka i poseduju metode koje se pozivaju od strane klijenta.

Java Server Pages[10] - predstavlja java tehnologiju za generisanje dinamičkog sadržaja zasnovanog na XHTML. JSF 2.2 koristi ovu tehnologiju kao podrazumevanu za generisanje prikaza tj. View dela aplikacije. JSF sadrži set XHTML i web orijentisanih UI komponenti koje se lako ugrađuju u JSP strane i olakšavaju razvoj korisničkog interfejsa.

```
<h:panelGroup styleClass="price_buy_holder" layout="block">
    <h:panelGroup styleClass="price_holder" layout="block">
        <h:panelGroup styleClass="price_wrapper" layout="block">
            <h:outputText value="#{currentShoeBean.price }"></h:outputText>
        </h:panelGroup>
    </h:panelGroup>

    <h:panelGroup styleClass="buy_holder" layout="block">
        <h:form>
            <a4j:jsFunction immediate="true" name="bindSplash"
                            oncomplete="goToBuyPage();">
                <f:setPropertyActionListener
                            target="#{currentUser.splashLink}"
                            value="#{currentShoeBean.buyLink }" />
            </a4j:jsFunction>
        </h:form>

        <h:outputLink target="_blank"
                      value="/jsp/splash.jsf?tgt=#{currentShoeBean.buyLink }">
            <h:panelGroup layout="block" styleClass="buy_btn buy_btn_unhovered" />
        </h:outputLink>
    </h:panelGroup>
</h:panelGroup>
```

Listing 2. Ilustracija JSF tagova unutar Java Server Pages strane.

Unified Expression Language (EL)[11] - predstavlja programski jezik posebne namene koji se koristi za ugrađivanje izraza u JSP stranice. EL sintaksom moze se pozvati proizvoljna metoda nekog managed objekta ili sam objekat. Novije implementacije EL mogu da zovu i metode sa parametrima, prosledjujući te parametre direktno iz JSP-a (ili neke druge tehnologije za generisanje prikaza). Objektima se pristupa preko sintakse #{*izraz*}.

```
<h:outputText value ="#{article.title }"rendered="#{not empty article.title }"/>
```

Listing 3. Primer pozivanja metoda objekta EL sintaksom

JSON

JSON (JavaScript Object Notation)[12] predstavlja lak format za razmenu podataka. Lak za ljude da ga pišu i čitaju, lak za mašine da ga parsiraju i koriste. Primer JSON formata:

{ "**id**": 1, "**name**": "Foo", "**price**": 123, "**tags**": ["B","E"], "**stock**": { "**warehouse**": 300, "**retail**":

20 } }

Vitičaste zagrade označavaju jedan objekat, pravougle zagrade predstavljaju niz. Vrednosti pre i posle dvotačke predstavljaju par ključ/vrednost.

AJAX

Ajax (Asynchronous JavaScript and XML)[13] predstavlja grupu tehnologija koje se zajedno koriste na klijentskoj strani za kreiranje asinhronih web aplikacija. Upotrebom ove tehnologije web aplikacije mogu da šalju podatke serveru i da ih prime od servera 'u pozadini', bez mešanja u trenutan prikaz stranice. Ovim mehanizmom se mogu osvežiti delovi stranice. Stranica samim tim postaje bogatija i prijatnija za korišćenje.

Iako sadrži XML u svom imenu, sam XML nije neophodan za korišćenje Ajax-a, čak se u najvećem broju slučajeva koristi JSON format.

```
$.getJSON('/brands', function(result) {
  console.log('Fetched brands: ' + result);
});
```

Listing 4. Primer asinhronog GET zahteva koji gadja java Servlet mapiran na "/brands" url. Metoda getJSON() jQuery biblioteke automatski parsira odgovor u JSON objekat.

EJB Stateless Session Beans

EJB predstavlja server-side model koji enkapsulira biznis logiku aplikacije. EJB kontejner sadrži dve vrste bean-ova:

- Session Beans – koji mogu biti "Stateful", "Stateless" ili "Singleton"
- Message Driven Beans

Stateless Session Beans[14] su biznis objekti koji ne poseduju stanje. Korisnik nema garancije da će se za sledeći poziv metode koristiti instanca stateless bean-a koja je i prvi put korišćena. Korišćenje jedne instance bean-a je ograničeno na jednog korisnika istovremeno. Ukoliko se pokuša konkurentan pristup jednom bean-u, kontejner jednostavno preusmerava zahteve na drugu instancu. Session bean-u se pristupa preko Local ili Remote interfejsa, u zavisnosti da li se konkrentna implementacija session bean-a nalazi na istoj java virtuelnoj mašini ili na nekom drugom (udaljenom) sistemu.

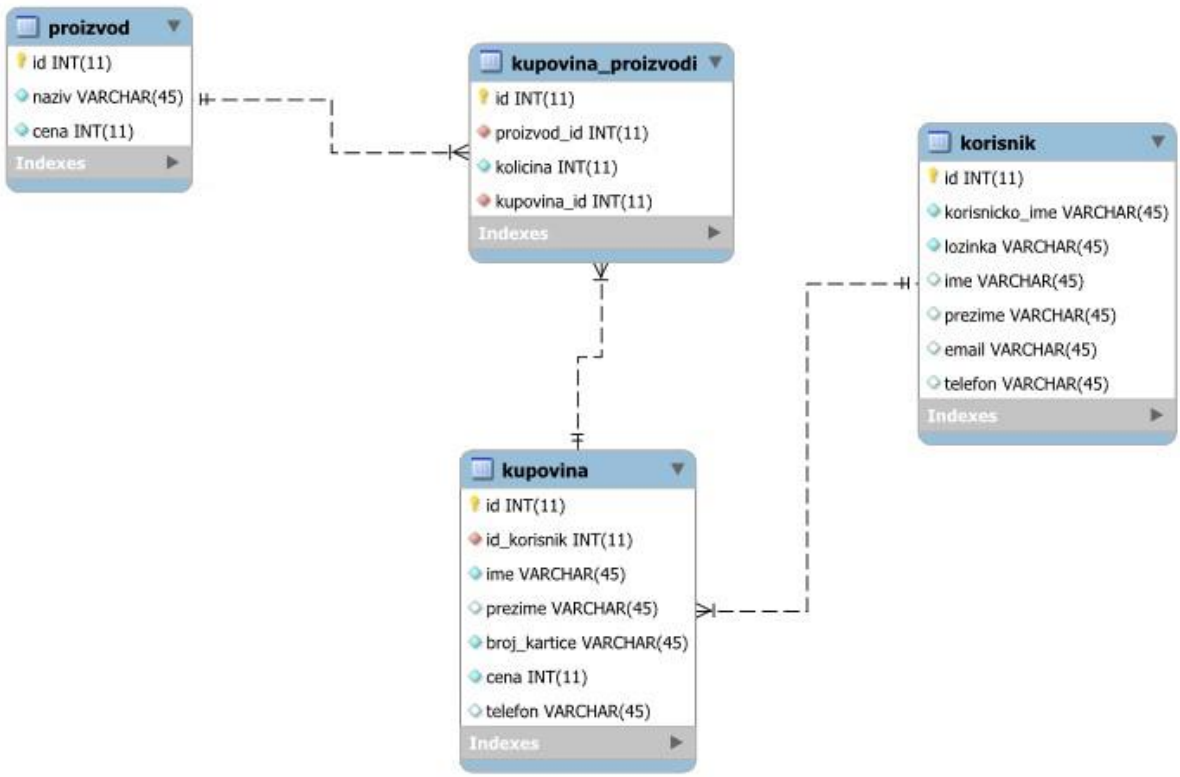
Hibernate

Hibernate[15] predstavlja objektno-relacioni maper za Java programski jezik, koji je zadužen za mapiranje objektno-orijentisanih modela u tradicionalne relacione baze.

Hibernate-ova primarna funkcija je mapiranje iz Java klasa u tabele baza. Takodje pruža alate za perzistovanje i pronalaženje Java objekata. Hibernate biblioteka je jednostavna za korišćenje jer se mapiranje klasa svodi na upotrebu Java anotacija. Anotirane klase nose naziv *entiteti (Entities).*

Model baze

Model baze ilustrovan je sledećom slikom:



Slika 1. Model baze

Kao platforma za bazu podataka uzeta je MySQL baza podataka sa trenutno najaktuelnijom verzijom 8.0 zajedno sa implementacijom odgovarajućeg MySQL JDBC 8.0 drajvera. Projekat je realizovan tako što je najpre kreirana baza podataka pa je putem OR mapiranja - JPA/Hibernate-a kreiran model u biznis logici. Zamisao je da postoje četiri tabele:

- Korisnik – tabela koja će čuvati podatke o svakom korisniku,koja će imati sledeće kolone:
  - ID koji je tipa integer i koji je primarni ključ
  - Korisničko ime koje je tipa VARCHAR(45)
  - Ime koje je tipa VARCHAR(45)
  - Prezime koje je tipa VARCHAR(45)
  - Email koje je tipa VARCHAR(45)
  - Telefon koje je tipa VARCHAR(45)

- Proizvod – tabela koja će čuvati sve proizvode koji će biti na raspolaganju za prodaju,i koja će imate sledeće kolone
    - ID koji je tipa INT i koji je primarni ključ
    - Naziv koje je tipa VARCHAR(45)
    - Cenu koja je tipa INT

- Kupovina koja je tabela koja će čuvati podatke o svakoj kupovini i sadrži
    - ID koji je tipa integer i koji je primarni ključ i koja označava redni broj kupovine
    - ID korisnika koji je strani ključ iz tabele korisnik
    - Ime koje je tipa VARCHAR(45)
    - Prezime koje je tipa VARCHAR(45)
    - Broj kartice koje je tipa VARCHAR(45)
    - Cena koje je tipa INT i koja će se automatski sračunavati u ukupnom iznosu i koja će nam omogućiti da kasnije lakše radimo izveštaje
    - Telefon korisnika koja je tipa VARCHAR(45)

- Kupovina-prozvodi je pomoćna tabela koja će čuvati podatke o svakoj kupovini,kako bi omogućila da se svaki proizvod može kupiti više puta kao i da se može kupiti više različitih proizvoda, i koja sadrži kolone
    - ID koji je tipa integer i koji je primarni ključ i označava redni broj kupovine
    - Proizvod_id koja je tipa INT i koja je strani ključ iz tabele proizvod
    - Količina koja je tipa INT
    - Kupovina_id koja je tipa INT i koja strani ključ iz tabele kupovina

Opis implementacije upotrebom JSF 2.2

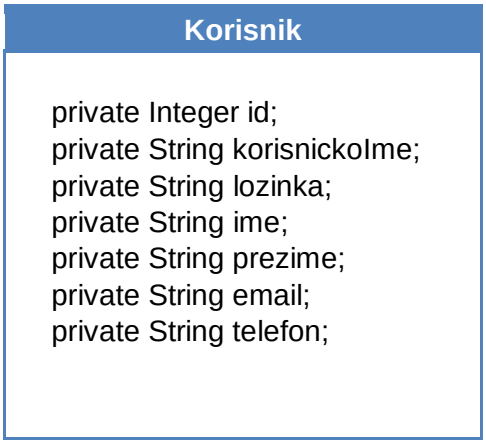
DigitalHeaven projekat je podeljen u dva web projekta koji su postavljeni na Glassfish aplikativni server kao jedna enterprise arhiva.
Prvi deo projekta predstavlja korisnički interfejs i sadrži biblioteke, resurse, Java Server Faces strane i managed bean-ove koji se tiču samog korisničkog interfejsa.
Drugi projekat predstavlja backend logiku aplikacije i sadrži modele klasa mapirane pomoću Hibernate biblioteke na odgovarajuće tabele kao i session bean-ove koji sadrže biznis metode za rad nad tim modelima. Session bean-ovi imaju ulogu fasada preko čijih interfejsa se pozivaju biznis metode od strane korisničkog dela projekta.

Primer jednog takvog poziva bilo bi logovanje na sajt na profilskoj strani index.html. Managed bean-u koji je zadužen za logovanje prosledjuju se parametri korisnicko ime I loznika klijenta. Managed bean preuzima podatke sa xHTML strane i poziva metodu fasade zaduženu za proveru tih podataka u bazi.

Logika aplikacije

Entiteti

Korisnik entitet – predstavlja model koji sadrži sve potrebne informacije vezane za jednog korisnika. Klasa je anotirana kao entitet i mapirana na odgovarajuću tabelu u bazi. Kako Korisnik klasa predstavlja model, ona ne sadrži nikakve metode koji manipulišu njenim atributima.
Pregled atributa Korisnik klase dat je klasnim dijagramom:



Slika 2. Klasni dijagram Korisnik entiteta

Sledi opis atributa Korisnik klase:

- Mapiranje same Korisnik klase:

```
@Entity
@Table(name = "korisnik")
@XmlRootElement
```

- **id** – predstavlja primarni ključ Korisnik objekta, mapiran kao:

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Basic(optional = false)
@Column(name = "id")
```

gde @Column(name=”ID”) predstavlja ime kolone u tabeli na koje će se mapirati atribut.

- **korisnickoIme** – ovaj atribut predstavlja samo ime korisnika mapirano kao:
  @Basic(optional = false)
  @NotNull
  @Size(min = 1, max = 45)
  @Column(name = "korisnicko_ime")
- **lozinka** – predstavlja lozinku korisnika, mapiranu kao:
  @Basic(optional = false)
  @NotNull
  @Size(min = 1, max = 45)
  @Column(name = "lozinka")

- **ime** – predstavlja ime korisnika, mapirano kao:
  @Size(max = 45)
  @Column(name = "ime")

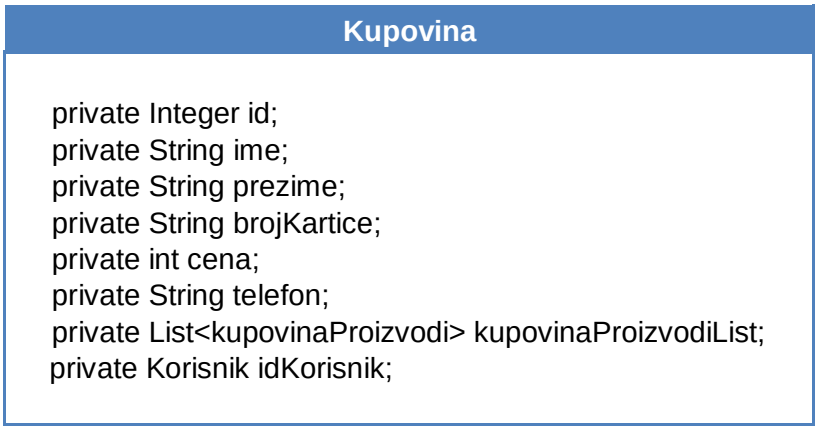
- **prezime** – predstavlja prezime korisnika, mapirano kao:
  @Size(max = 45)
  @Column(name = "prezime")

- **email** – predstavlja email adresu korisnika, mapiran kao:
  @Size(max = 45)
  @Column(name = "email")

- **telefon** – predstavlja broj telefona korisnika, mapiran kao:
  @Size(max = 45)
  @Column(name = "telefon")

Kupovina entitet – predstavlja model koji sadrži sve potrebne informacije vezane za jednu kupovinu osim kolicine i tipa proizvoda koja će biti realizovana KupovinaProizvodi modelom. Klasa je anotirana kao entitet i mapirana na odgovarajuću tabelu u bazi. Kako kupovina klasa predstavlja model, ona ne sadrži nikakve metode koji manipulišu njenim atributima.

Pregled atributa Korisnik klase dat je klasnim dijagramom:



Slika 3. Klasni dijagram Kupovina entiteta

Sledi opis atributa Kupovina klase:

- Mapiranje same Korisnik klase:

```
@Entity
@Table(name = "kupovina")
@XmlRootElement
```

- **id** – predstavlja primarni ključ Korisnik objekta, mapiran kao:

```
@Id
@GeneratedValue(strategy =  GenerationType.IDENTITY)
@Basic(optional = false)
@Column(name = "id")
```

  gde @Column(name="ID") predstavlja ime kolone u tabeli na koje će se mapirati atribut.

- **ime** – ovaj atribut predstavlja samo ime korisnika mapirano kao:

```
@Basic(optional = false)
@NotNull
@Size(min = 1, max = 45)
@Column(name = "ime")
```

- **prezime** – predstavlja prezime korisnika, mapirano kao:

```
@Size(max = 45)
@Column(name = "prezime")
```

- **brojKartice** – predstavlja broj kartice korisnika, mapiran kao:

```
@Size(max = 45)
@Column(name = "broj_kartice")
```

- **cena** – predstavlja ukupnu cenu, mapiranu kao:

```
@Basic(optional = false)
@NotNull
@Column(name = "cena")
```

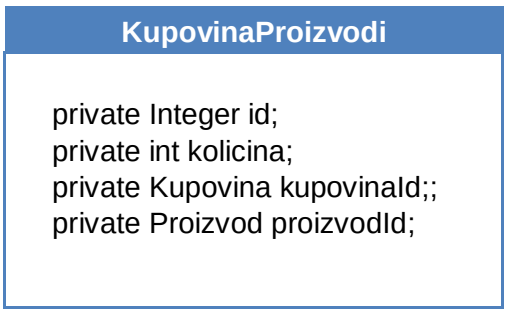
- **telefon** – predstavlja telefon korisnika, mapiran kao:

```
@Size(max = 45)
@Column(name = "telefon")
```

- **kupovinaProizvodiList** – predstavlja "one-to-many" relaciju,jer jedna kupovina_proizvodi tabela može imati više reda sa istim kupovinaId-em,a ovo je projektovano tako da bi se realizovalo da jedna kupovina moze da ima više razlicitih proizvoda kao i više istih proizvoda pa je ovo realizovano pomoćnom tabelom kupovina_proizvodi koja će beležiti sve prodaje,a kako je prilikom kreiranja entiteta iz baze podataka odabrana opcija da se strani ključevi realizuju preko list-a ovaj atribut je tipa List. kolona mapirana kao:

```
@OneToMany(cascade = CascadeType.ALL, mappedBy = "kupovinaId")
```

KupovinaProizvodi entitet – predstavlja model koji sadrži sve potrebne informacije vezane za jednu kupovinu. Klasa je anotirana kao entitet i mapirana na odgovarajuću tabelu u bazi. Kako kupovina klasa predstavlja model, ona ne sadrži nikakve metode koji manipulišu njenim atributima.
Pregled atributa Korisnik klase dat je klasnim dijagramom:



Slika 4. Klasni dijagram KupovinaProizvodi entiteta

Sledi opis atributa KupovinaProizvodi klase:

- Mapiranje same Korisnik klase:

```
@Entity
@Table(name = "kupovina_proizvodi")
@XmlRootElement
```

- **id** – predstavlja primarni ključ KupovinaProizvodi objekta, mapiran kao:

```
@Id
@GeneratedValue(strategy =  GenerationType.IDENTITY)
@Basic(optional = false)
@Column(name = "id")
```

gde @Column(name=”ID”) predstavlja ime kolone u tabeli na koje će se mapirati atribut.

- **kolicina** – ovaj atribut predstavlja količinu datog proizvoda korisnika mapirano kao:

```
@Basic(optional = false)
@NotNull
@Column(name = "kolicina")
```

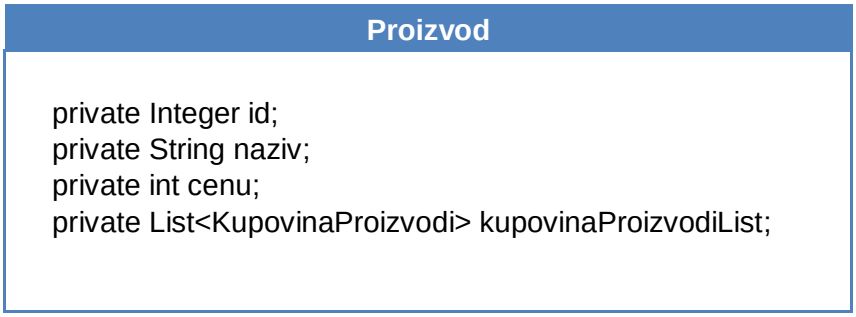
- **kupovinaId** – ovaj atribut predstavlja inicijalnu kupovinu za koju se vezuje putem stranog ključa, mapiranu kao:

```
@JoinColumn(name = "kupovina_id", referencedColumnName = "id")
@ManyToOne(optional = false)
```

- **proizvodId** – ovaj atribut predstavlja proizvod za koju se vezuje putem stranog ključa, mapiranu kao:

```
@JoinColumn(name = "kupovina_id", referencedColumnName = "id")
@ManyToOne(optional = false)
```

Proizvod entitet – predstavlja model koji sadrži sve potrebne informacije vezane za jedan proizvod. Klasa je anotirana kao entitet i mapirana na odgovarajuću tabelu u bazi. Kako kupovina klasa predstavlja model, ona ne sadrži nikakve metode koji manipulišu njenim atributima. Pregled atributa Korisnik klase dat je klasnim dijagramom:



Slika 5. Klasni dijagram Proizvod entiteta

Sledi opis atributa KupovinaProizvodi klase:

- Mapiranje same Korisnik klase:

```
@Entity
@Table(name = "proizvod")
@XmlRootElement
```

- **id** – predstavlja primarni ključ KupovinaProizvodi objekta, mapiran kao:

```
@Id
@GeneratedValue(strategy =  GenerationType.IDENTITY)
@Basic(optional = false)
@Column(name = "id")
```

  gde @Column(name="ID") predstavlja ime kolone u tabeli na koje će se mapirati atribut.

- **naziv** – ovaj atribut predstavlja naziv datog proizvoda mapiranog kao:

```
@Basic(optional = false)
@NotNull
@Column(name = "naziv")
```

- **cena** – ovaj atribut predstavlja cenu datog proizvoda mapiranog kao:

```
@Basic(optional = false)
@NotNull
@Column(name = "cena")
```

- **kupovinaProizvodiList** – predstavlja "one-to-many" relaciju,jer jedna kupovina_proizvodi tabela može imati više reda sa istim proizvodId-em, kolona mapirana kao:

```
@OneToMany(cascade = CascadeType.ALL, mappedBy = "kupovinaId")
```

Klase

LoginBean klasa

Predstavlja Stateless Session Bean koji sadrži logiku vezanu za aspekt logovanja na aplikaciju. Metodama se pristupa preko Local interfejsa pošto se sama implementacija nalazi unutar iste Java virtuelne mašine. Sadrži jednu metodu koja vraća objekat tipa Korisnik i kojoj se prosledjuju dva parametra "username" i "password" i koja vrši ispitivanje prema bazi putem named query-a "Korisnik.findByKorisnickoImeLozinka" pretražuje bazu za prosledjeno korisničko ime i lozinku i u slučaju poklapanja vraća entitet korisnik,i sve to u jednom try-catch bloku,i u slučaju ne poklapanja ispaljuje NoResultException i u tom slučaju vraća vrednost NULL.

ProductBean klasa

Predstavlja Stateless Session Bean koji sadrži logiku za iščitavanje svih proizvoda iz baze i ima jednu metodu listAllProducts() koja kao rezultat vraća array list-u proizvoda putem named query-a "Proizvod.findAll".

PurchaseBean klasa

Predstavlja Stateless Session Bean koji sadrži logiku vezanu za registrovana kupovine. Ona dobija od PurchaseManagedBean-a kao ulazne parametre objekat tipa Korisnik, List<ShoppingCartItem> -listu svih proizvoda koji se nalaze u korpi i broj kreditne kartice koju korisnik unosi prilikom kupovine. Zatim formira entitet Kupovina i setuje sve potrebne attribute uzimajuci parametre iz prosledjenih vrednosti.Zatim prelazi na konvertovanje liste proizvoda iz korpe u listu List<KupovinaProizvodi> i “for each” petljom prolazi kroz sve stavke korpe i setuje sve attribute i na kraju sracuna totalnu cenu kako bi popunila i polje “total_price”. Posle toga u “try-catch” bloku pokusava da izvrši perzistovanje,tj.da putem entity-manager-a i hibernate-a kreiran objekat smesti u MySql bazu.

RegisterBean klasa

Predstavlja Stateless Session Bean koji sadrži logiku sa registraciju novih korisnika, i ima jednu metodu:

```
public boolean register(String username, String password, String name, String surname,
String email, String phoneNumber)
```

Ova metoda prima od RegisterManagedBean-a ulazne parametre korisnicko ime,ime i prezime korisnika,njegov email kao i broj telefona. Ova metoda najpre proverava da li se za uneto korisnicko ime u bazi možda već nalazi i u tom slučaju hvatamo Excpetion,a u slučaju da nije nastavljamo sa kreiranjem novog Korisnik objekta setujući mu sve prosledjene parametre i perzistujemo ga u bazu.

LoginFilter klasa

LoginFIlter klasa je pomoćna klasa koja implementira Filter iz servlet paketa,koja nam služi da bi zaštitili deo sajta koji je namenjen potvrdi kupovine i koju mogu samo ulogovani korisnici da posećuju. Prilikom implementacije interfejsa Filter dobijamo na implementaciju tri metode od kojih koristimo samo prvu,

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
```

Tu iz sesije izvlačimo podatke o logovanom korisniku putem LoginManagedBean-a,tačnije gledamo da li je korisnik logovan,i ako nije preusmeravamo ga na login.xhtml stranu,a u suprotnom nastavljamo tamo gde smo stali. Kada smo napravili ovaj filter potrebno je u web.xml konfiguracionom fajlu da ga registrujemo,dodavajući sledeće parametre:

```
<filter>
    <filter-name>LoginFilter</filter-name>
    <filter-class>webshop.filter.LoginFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>LoginFilter</filter-name>
    <url-pattern>/faces/secured/*</url-pattern>
</filter-mapping>
```

Slika 6. Dodati parametric u web.xml konfiguracijonom fajlu .

Java Server Faces komponente

Sve Java Server Faces komponente smeštene su u “beans” podpaket glavnog “webshop” paketa,i podeljene su u dve grupe. Prva i glavna grupa sadrži sve “Managed bean-ove” a druga grupa koja se nalazi u podpaketu ovog paketa koji se naziva “model” sadrži pomoćne klase koje koristimo da obezbedimo funkcionisanje svih funkcionalnosti.

EmailManagedBean klasa

Predstavlja managed bean koji je će prilikom svake kupovine da bude aktiviran i koji je zadužen da prilikom svake kupovine sakupi podatke o prodaji i prosledi ih na željeni email. Da bi funkcionisao ovaj “Managed bean” ,u projekat je potrebno uključiti javax.mail biblioteku za rad sa email-ovima. Ovaj bean ima jednu metodu u okviru koje je enkapsulirano sve:

public void sendEmail(String listaproizvoda)

i ova metoda dobija kao parametar listu proizvoda u vidu stringa,a koji joj prosledjuje PurchaseManagedBean prilikom poziva. Ova metoda u sebi ima sve potrebne parametre za rad sa mail-ovima,kao što su podešavanje zaglavlja mail-a, sadržaja, i podešavanja smtp mail servera koji se koristi za isporuku željenih podataka. Da bi ispunila željeni zahtev ona kao “ManagedProperty” koristi LoginManagedBean iz koga dobija podatke o kupcu.

KorpaManagedBean klasa

Predstavlja managed bean koji enkapsulira logiku vezanu za jednu korpu,i ona je “Session Scope”-a jer se vezuje za jednu sesiju i jednog korisnika. Od atributa ima array list-u objekata pomoćne klase ShoppingCartItem koja će biti u nastavku detaljnije objašnjena,i sadrži productNum koji je tipa Integer i koji nam služi kako bi sračunavali količinu proizvoda.
Poseduje tri metode. Prva metoda je:

addProduct(Proizvod proizvod)

Ova metoda kao ulazni parametar dobija objekat klase modela Proizvod i koja prvo proverava da li se u array list-i već nalazi taj proizvod i ako se nalazi onda uvećava njen broj za jedan,ili za prosleđenu vrednost sa index.html strane iz h:selectOneMenu tag-a.U slučaju da se ne nalazi,ona formira novi objekat tipa ShoppingCartItem sa potrebnim atributima i dodaje ga u listu.Druga metoda je:

getTotalPrice()

Ona ima za zadatak da računa ukupnu cenu i to postiže upotrebom "for each" petlje,gde prolazi kroz sve stavke korpe i sabira sve cene u total promenljivu koja je tipa int.Posledja metoda je metoda:

emptyCart()

Ova metoda ima za zadataka da posle uspešno obavljene kupovine resetuje sadržaj korpe kako bi omogućila korisniku da krene ceo process kupovine iz početka.

LoginManagedBean klasa

Predstavlja managed bean koji je zadužen za logovanje korisnika,tačnije za preuzimanje parametara sa xhtml strane i prosledjivanja LoginBean session bean-u na validaciju.Ona u sebi mora da uključi session bean i ona to mora da uključi putem implementaciju njegovog local interface-a ,a pošto se i session bean i managed bean nalaze u okviru iste Java virtualne mašine ovo postiže anotacijom @EJB .Ovim je „injektovana" cela implementacija stateless session bean-a i sada je moguće pozivati njegove metode. Ovaj bean ima jednu metodu

login()

Ova metoda vraća promenljivu tipa String,i u zavisnosti od ishoda može vratiti "Success" ako je uspešno logovanje,i "Failure" ako je neuspešno. Ova metoda najpre proverava da li je korisnik već ulogovan,i u slučaju da jeste vraća odmah "Success" a ako nije,nastavlja sa logovanjem i prosleđuje korisničko ime i lozinku i proverava da li je uspešno pronadjena kombinacija i u slučaju da nije,vraća "Failure" i ispisuje poruku o grešci.

ProductsManagedBean klasa

Predstavlja managed bean koji ima za zadatak da samo preuzme podatke od ProductsBean session bean-a da bi ih dalje propagira na JSF stranu,i ima samo jednu metodu

getProducts()

Ova metoda vraća listu proizvoda,a kao izvor ima podatke koje dobija od injektovanog session bean-a.

PurchaseManagedBean klasa

Predstavlja managed bean koji enkapsulira svu logiku potrebnu za obavljanje kupovine na purchase.xhtml JSF strani. On da bi obavljao sve potrebne zadatke u sebi kao "MangedProperty" ima injektovana tri managed bean-a: LoginManagedBean, KorpaManagedBean i EmailManagedBean.Takodje,da bi izvršio perzistovanje u bazu,on u sebi ima injektovan i PurchaseBeanLocal interfejs session bean-a.

Prva metoda je:

submitPurchase()

Ova metoda ima za zadatak najpre da prosledi podatke session bean-u koji će izvršiti perzistovanje u bazu putem hibernate-a,i ako uspešno završi da podatke o datoj prodaji prosledi na predefinisani email,i po završetku toga on prazni korpu kako bi omogućio korisniku da započne novu kupovinu ukoliko to želi. Ovaj bean ima još jednu metodu:

getProductList()

Ova metoda ima za zadatak da iz uz pomoć KorpaManagedBean-a izvuče sve proizvode i napravi listu prodatih proizvoda konkatenacijom naziva u jedan duži string i to vrati.

RegisterManagedBean klasa

Predstavlja managed bean koji enkapsulira logiku vezanu za registraciju i glavni zadatak joj je da unete podatke sa register.xhtml strane propagira RegisterBean session bean-u na perzistovanje. Ovaj bean ima jednu metodu:

register()

Ova metoda vraća string koji u ovom slučaju može biti “failure” u slučaju da dodje do greške prilikom perzistovanja u bazu,i ako je kojim slučajem korisničko ime zauzeto u bazi,jer je stavljeno da bude jedinstveno i u tom slučaju ispisaće poruku da je korisničko ime zauzeto, i “success” u slučaju da je proces uspešno završen. Ove vrednosti stringova ćemo koristiti kod konfigurisanja navigacionih pravila u “faces-config.xml” konfiguracionom fajlu,kako bi u slučaju uspešnog registrovanja vratili korisnika na glavni panel,odnosno vratili na ponovnu registraciju usled greške.

Pomoćne komponente

MyCreditCardValidator klasa

Ova klasa namenjene je validaciji kreditnih kartica,i u sebi implementira “Validator” interfejs. Da bi mogli da koristimo ovu klasu kao validator,ona mora biti najpre anotirana sa @FacesValidator anotacijom,i u “faces-config.xml” konfiguracionom fajlu biti dodata na sledeći način:

```
<validator>
    <validator-id>CreditCardValidator</validator-id>
    <validator-class>webshop.beans.model.MyCreditCardValidator</validator-class>
</validator>
```

Slika 7. “validator” tag u faces-config.xml fajlu

Prilikom implementacije “Validator” interfejsa potrebno je nadjačati (override-ovati) metodu “validate” kojoj se prilikom pozivanja prosledjuje “value” objekat tipa “Object”.U ovoj metodi najpre vršimo eksplicitnu konverziju prosledjenog objekta- broja kreditne kartice u promenljivu

"card" koja je tipa String,i onda tu promenljivu šaljemo na proveru metodi validateCard() .Ova metoda najpre proverava da li je prosledjena "null" vrednost,i ako nije nastavlja dalje,pozivajući metodu calculateCheckDigit() koja će izvršiti proveru dobijene "card" string promenljive po takozvanom "Lunovom algoritmu" koji služi za generisanje brojeva kreditnih kartica po odredjenom algoritmu.

ShoppingCartItem klasa

Ova klasa je standardna java klasa,i predstavlja jednu stavku u korpi proizvoda. Ona ima dva atributa, "product" koji je tipa "Proizvod" iz modela,i "productNum" koji je tipa int i njega koristimo kako bi imali mogućnost dodavanja više proizvoda u korpu.

VerifyRecaptcha klasa

VerifyRecaptcha klasa je pomoćna klasa koju koristimo kako bi implementirali "I'm not a robot" Google recaptcha servis prilikom logovanja na sajt. Da bi ovo ostvarili potrebno je najpre kreirati nalog na Google-ovom recaptcha servisu,i tu navesti na kojim ćemo domenima koristiti ove servise.Tom prilikom dobijamo dva ključa,jedan je obelezen kao "site key" a drugi kao "secret key". "Site key" je potrebno dodati kao vrednost na našu html stranicu na kojoj želimo servis u okviru "div" taga koji dobijamo od Google-a kao i u "head" delu sajta potrebno je dodati jedan "script" tag kako bi servis bio aktivan. Drugi, "secret key" je potrebno implementirati u okviru ove klase na mesto označeno od strane Google-a. Posle ovoga ostaje još da u okviru našeg LoginMangedBean pre početka login operacije najpre pozovemo Boolean metodu koja će izvršiti proveru recaptcha vrednosti i ako dobijemo vrednost "true" –što znači da je verifikacija uspesno obavljena,nastaviti sa procesom logovanja.

Web xHTML stranice

Glavna stranica, index.xhtml

Ovo je glavna stranica koja je ujedno i početna stranica celog projekta. Za stilizaciju je korišćena CSS 3.0 tehnologija. Cela stranica je najpre smeštena u jedan glavni "div" koji je nazvan "wrapper",i koji je kasnije podeljen u više divova,"header","nav","logo", "main", i "footer". Dalje se u okviru ovih div-ova nalaze još specifičniji divovi koji nam pomažu kako bi rasporedili komponente kako želimo.

Glavna stranica sadrži listu svih proizvoda koji su na raspolaganju u webshop-u kao i mogućnost dodavanja u korpu. Na dnu stranice nalazi se izlistan sadržaj korpe zajedno sa ukupnom cenom,i ovaj deo se renderuje asihrono putem ajax tehnologije koja je uključena u JSF 2.2 tehonologiju kao podrazumevana i aktiviramo je jednostavnim tagom <f:ajax> koji uključujemo u <h:commandButton> tag i podešavamo da reaguje prilikom pritiska na dugme „dodaj" kako bi u svakom trenutku imali ažuriran sadržaj korpe bez tzv. „refresh"-ovanja stranice.Ova stranica je kreirana putem JSF tehnologije i nalazi se u glavnom „web" folderu koji je podfolder korenom direktorijumu projekta.Koristi za stilizaciju style.css fajl koji se nalazi u resources podfolderu,koji u sebi ima i „images" podfolder koji sadrži sliku koja predstavlja pozadinu stranice.

Stranica za logovanje korisinka, login.xtml

Ova stranica je zadužena za logovanje korisnika na aplikaciju,i uz podršku LoginManagedBean-a preuzima podatke koje dalje propagira session bean-u koji vrši validaciju unetih podataka u MySql bazi i u odnosu na rezultat nas preusmerava na glavnu stranu ako je uspešno ili nas vraća na istu stranu u slučaju da su uneti neispravni podaci,kako bi moglo da se ponovo pokuša. Kao zaštitu od zlonamernih korisnika,ova stranica je zaštićena Google recaptcha tehnologijom koja opisana u odeljku “Pomoćne komponente”.

Stranica za registraciju novih korisnika, register.xhtml

Ova stranica je zadužena za registraciju novih korisnika i implementira u sebi RegisterManagedBean. Prilikom unosa parametara u određena polja,koriste se odredjeni validatori,pa tako prilikom unosa lozinke koristimo

```
<f:validateRegex pattern="((?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,20})" />
```

koji nam sprečava korisnika da unese lozinku ako ne zadovoljava naše standarde a to su da lozinka mora da sadrži od 6 do 20 karaktera,da mora da sadrži najmanje jedno veliko i jedno malo slovo kao i mora da sadrži najmanje jedan broj. Kod unosa email-a se takodje vrši validacija formata email-a i koristi se sledeći tag:

```
<f:validateRegex pattern="^[_\w-\+]+(\.[_\w-]+)*@[\w-]+(\.[\w]+)*(\.[A-Za-z]{2,})$">
</f:validateRegex>
```

Prilikom uspešne registracije faces-config.xml konfiguracioni fajl je podešen tako da se automatski korisnik preusmerava na glavnu stranicu.

Stranica za potvrdu kupovine, purchase.xhtml

Ova stranica je smeštena u “secured” podfolder glavnog foldera web dela projekta,služi za potvrdu kupovine i njoj je moguće pristupiti samo ako je korisnik ulogovan,što smo objasnili prilikom objašnjavanja “LoginFilter” klase. Ona sadrži izlistane podatke o korisniku kao i sadržaj korpe i jedno polje za unos broja kreditne kartice.Za ovo polje vezan je “custom made” validator koji će validirati broj kartice,a koji smo detaljno objasnili prilikom objašnjenja klase MyCreditCardValidator u odeljku “Pomoćne komponente”. Posle uspešne kupovine,korisnik se preusmerava na početnu stranu,kako bi mogao da nastavi sa kupovinom novih proizvoda ako želi.

Reference

1. http://docs.oracle.com/javaee/6/tutorial/doc/
2. http://www.wildfly.org
3. http://www.oracle.com/technetwork/java/overview-141217.html
4. http://docs.oracle.com/javase/6/docs/technotes/guides/rmi/index.html
5. http://docs.oracle.com/javaee/6/tutorial/doc/bnafd.html
6. http://docs.oracle.com/javaee/6/tutorial/doc/bnagb.html
7. http://docs.oracle.com/javaee/6/tutorial/doc/bnaph.html
8. http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
9. http://docs.oracle.com/javaee/6/tutorial/doc/bnatx.html
10. http://docs.oracle.com/javaee/5/tutorial/doc/bnagx.html
11. http://java.sun.com/products/jsp/reference/techart/unifiedEL.html
12. http://www.json.org/
13. http://en.wikipedia.org/wiki/Ajax_(programming)
14. http://en.wikipedia.org/wiki/Session_Beans
15. http://docs.jboss.org/hibernate/core/4.0/devguide/en-US/html_single/
16. http://ocpsoft.com/docs/prettyfaces/3.3.2/en-US/html_single/
17. https://sites.google.com/site/gson/gson-user-guide
18. http://jsoup.org/
19. http://facelets.java.net/nonav/docs/dev/docbook.html
20. http://docs.jboss.org/weld/reference/latest/en-US/html/
21. http://docs.jboss.org/weld/reference/latest/en-US/html/interceptors.html
22. http://docs.jboss.org/weld/reference/latest/en-US/html/decorators.html
23. http://docs.jboss.org/weld/reference/latest/en-US/html/injection.html#alternatives