UNIVERZITET
METROPOLITAN
BEOGRAD

CS230 - DISTRIBUIRANI SISTEMI

JavaServer Faces

Lekcija 05

PRIRUČNIK ZA STUDENTE

CS230 - DISTRIBUIRANI SISTEMI

Lekcija 05

JAVASERVER FACES

UVOD

Cilj lekcije je analiza i demonstracija specifičnog Java okvira za razvoj veb aplikacija.

JavaServer Faces (JSF) predstavlja specifičan Java okvir (framework) namenjen za razvoj veb aplikacija. Upravo, u ovoj lekciji će biti cilj pokazivanje načina na koji primena JSF pojednostavljuje razvoj veb aplikacija. U tu svrhu će biti obrađene značajne teme za analizu i demonstraciju ovog specifičnog Java okvira za razvoj veb aplikacija:

- Kreiranje JSF projekta primenom izabranog razvojnog okruženja;
- Postavljanje JSF tagova za korišćenje prednosti koju obezbeđuje primena taga JSF <h:panelGrid>;
- Korišćenje statičke i dinamičke navigacije za definisanje navigacije između stranica;
- Razvoj CDI (Contexts and Dependency Injection for the Java EE Platform) zrna za enkapsulaciju podataka i programske logike;
- Implementacija JSF validatora;
- Kreiranje JSF šablona;
- Rad sa JSF temama.

Posebno će lekcija insistirati na analizi i demonstraciji primenom odabranih primera koji će na najbolji način istaći i pokazati prednosti primene JSF stranica u Java veb aplikacijama. Od posebnog značaja će biti fokusiranost primera na rad sa šablonima stranica i temama za pokazivanje veoma jednostavnog načina kreiranja JSF stranica sa predefinisanim atraktivnim izgledom. Konkretan primer će pokazati kako jedna šablon JSF datoteka na veoma jednostavan način upravlja nizom JSF datoteka klijenata šablona.

Savladavanjem ove lekcije studenti će biti osposobljeni da koriste JSF tehnologiju za razvoj Java veb aplikacija.

Poglavlje 1

Uvod u JSF

UVODNA RAZMATRANJA

JSF je standardna komponenta Java EE platforme.

Pre pojave JSF okvira, većina Java veb aplikacija je razvijana primenom nestandardnih razvojnih okvira za veb aplikacije (koji ne predstavljaju integralno deo Java EE platforme) kao što su: Apache Struts, Tapestry i Spring Web MVC, kao i mnogi drugi. Navedeni okviri su izgrađeni na servlet i JSP standardima i automatizuju brojne funkcionalnosti koje bi morale, inače, ručno da se kodiraju kada bi se navedeni API - ji direktno koristili.

Uvođenje JSF u Java EE specifikaciju dalo je za rezultat standardan, veoma moćan aplikativni razvojni okvir dostupan u bilo kojem Java EE aplikativnom serveru. Sa standarizovanjem JSF kao integracionog dela Java EE platforme, brojni programeri Java Enterprise aplikacija upravo koriste JSF za kreiranje korisničkog interfejsa u aplikaciji.

Kao što je postavljeno kao praksa, u ovim materijalima će svako izlaganje biti praćeno odgovarajućim primerom. Tako će i u ovom slučaju, uvodna razmatranja biti podržana razvojem jednostavne JSF aplikacije.

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

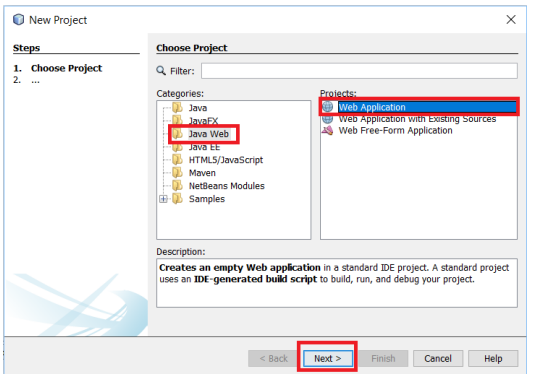
KREIRANJE I RAZVOJ JSF PROJEKTA

Tekuća razmatranja će biti podržana razvojem jednostavne JSF aplikacije.

Iz ugla aplikativnog programera JSF aplikacija se sastoji iz jedne ili više XHTML stranica koje sadrže JSF tagove, jedan ili više CDI zrna i opciono konfiguracione datoteke pod nazivom faces-config.xml.

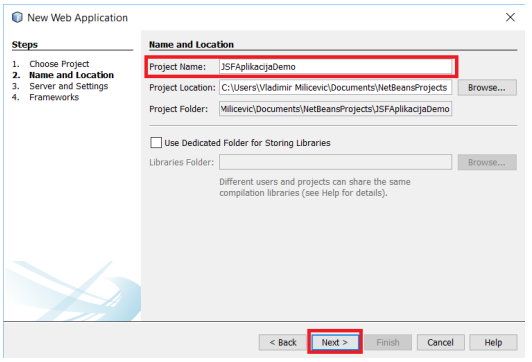
Starije verzije JSF su insistirale na primeni konfiguracija (JSF upravljanim zrnima) za implementaciju funkcionalnosti serverske strane. Zbog kompatibilnosti unazad, primena JSF konfiguracija je i dalje podržana ali je kao dobra praksa, i olakšanje, dodata podrška za anotacije (CDI zrna) koje se dominantno koriste u razvoju savremenih JSF veb aplikacija.

Za demonstraciju rada sa JSF projektima biće kreiran konkretan. U razvojnom okruženju (neka to bude NetBeans IDE) u meniju File i izborom opcije New Project, otvoriće se prozor u kome će biti izabrano da se razvija Java veb aplikacija (sledeća slika).



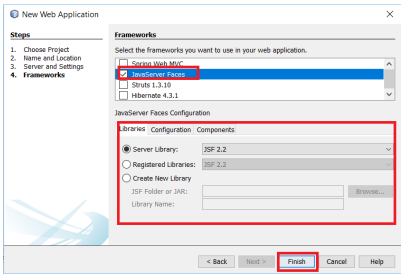
Slika 1.1 Započinjanje JSF projekta [izvor: autor]

Klikom na dugme Next otvara se nov prozor u kojem će biti određen naziv projekta, na primer JSFAplikacijaDemo.



Slika 1.2 Definisane naziva projekta [izvor: autor]

Klikom na Next, otvoriće se prozor gde se bira verzija Jave EE i tip aplikacionog servera. Klikom na Next se otvara poslednji prozor u kojem se bira JSF framework za razvoj kreiranog veb projekta.

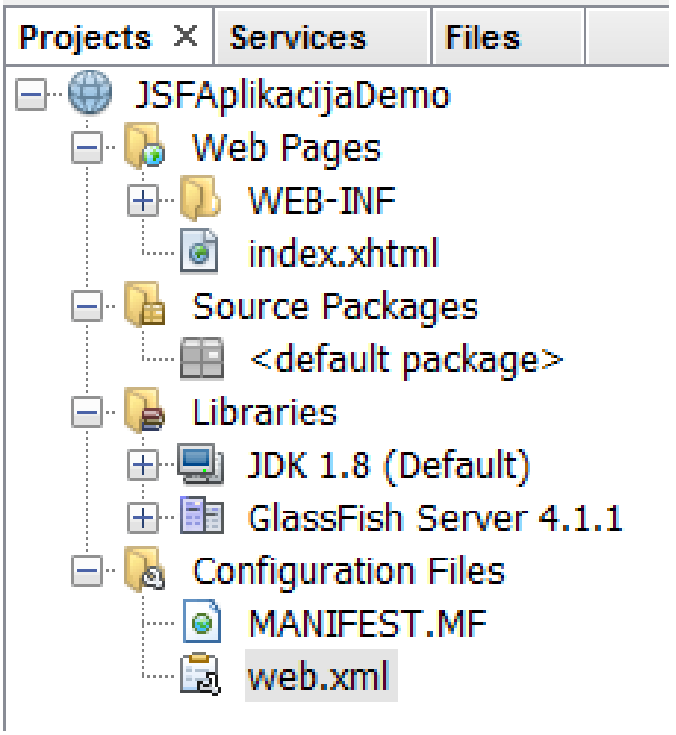


Slika 1.3 Izbor JSF razvojnog okvira [izvor: autor]

STRUKTURA JSF PROJEKTA

Klikon na Finish, nakon izbora JSF okvira, vrši se kreiranje projekta.

Prethodne aktivnosti se završavaju klikom na dugme Finish prozora u okviru kojeg je izabran JSF okvir za korišćenje u tekućem primeru. Nakon toga razvojno okruženje je generisalo projekat sa strukturom koja je prikazana na sledećoj slici.



Slika 1.4 Struktura JSF projekta [izvor: autor]

U pokazanoj strukturi je moguće uočiti datoteku web.xml koja predstavlja standardni, opcioni konfiguracioni fajl kojeg koriste Java veb aplikacije. Fajl je postao opcionalan od verzije Servlet API 3.0 koji je uveden Java EE 6 platformom. U većini slučajeva, u savremenim Java veb aplikacijama, ovaj fajl više nije potreban budući da se podešavanja obavljaju putem anotacija. Međutim, za JSF aplikacije je dobra ideja da se kreira ova datoteka budući da se u okviru nje mogu specificirati faze razvoja aplikacije. Kreiranjem projekta, automatski je generisan sledeći kod za web.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/
javaee/web-app_3_1.xsd">
    <context-param>
        <param-name>javax.faces.PROJECT_STAGE</param-name>
        <param-value>Development</param-value>
    </context-param>
    <servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>/faces/*</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
```

```
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>faces/index.xhtml</welcome-file>
    </welcome-file-list>
</web-app>
```

FAZE JSF PROJEKTA

Razvojno okruženje je automatski podesilo JSF fazu projekta na Development.

Kao što je moguće primetiti u prethodno priloženom kodu za deskriptor veb razvoja web.xml, razvojno okruženje je automatski podesilo JSF fazu projekta na vrednost ***Development***. Na ovaj način razvojno okruženje je omogućilo primenu dopunskih alata za pomoć debagovanju koji nisu dostupni u ostalim fazama JSF projekta. Na primer, standardni problem kod razvoja JSF stranice može biti da, tokom razvoja stranice, provera jednog ili više polja na stranici "pada" iz razloga što programer nije dodao neki od elemenata <h:message> ili <h:messages> u stranicu. Kada se ovo desi i odgovarajuća forma je potvrđena (submit) moguće je uočiti da se ništa ne dešava ili se navigacija između stranica ne obavlja na odgovarajući način. Podešavanjem JSF faze projekta na vrednost Development, odgovarajuće greške validacije biće automatski dodate u stranicu, bez potrebe da ih programer eksplicitno dodaje u kod odgovarajućim tagovima. Pre konačnog objavljivanja koda moguće je dodati odgovarajuće tagove u stranicu budući da korisnik, u suprotnom, neće biti u stanju da vidi automatski generisane informacije o greškama validacije.

U narednom izlaganju je neophodno navesti i obrazložiti dozvoljene vrednosti za javax.faces.PROJECT_STAGE:

- ***Development***
- ***Production***
- ***SystemTest***
- ***UnitTest.***

Kao što je već napomenuto u fazi Development se dodaju dopunske informacije koje pomažu prilikom debagovanja. Faza Production se bavi performansama. Ostale dve faze omogućavaju da se kroz testiranje uposli odgovarajuće ponašanje.

Klasa javax.faces.application.Application poseduje metodu getProjectStage() kojom je moguće dobiti informaciju o trenutnoj fazi JSF projekta. Navedeno je moguće ilustrovati sledećim kodom:

```
public void NekaMetoda() {
FacesContext facesContext = FacesContext.getCurrentInstance();
Application application = facesContext.getApplication();
ProjectStage projectStage = application.getProjectStage();
if (projectStage.equals(ProjectStage.Development)) {
```

```
//faza razvoja
} else if (projectStage.equals(ProjectStage.Production)) {
//faza produkcije
} else if (projectStage.equals(ProjectStage.SystemTest)) {
// faza sistemskog testiranja
} else if (projectStage.equals(ProjectStage.UnitTest)) {
//faza jediničnog destiranja
}
}
```

Kao što je ilustrovano u prethodnom listingu, moguće je implementirati kod koji će se izvršiti u bilo kojoj od faza JSF projekta u zavisnosti od povratne vrednosti metode getProjectStage() klase Application.

FACELET

Facelet nije ništa drugo do XHTML datoteka sa specifičnim JSF tagovima.

Sa kreiranjem projekta facelet je automatski generisan. Facelet nije ništa drugo do XHTML datoteka sa specifičnim JSF prostorom naziva (namespace). Automatski generisan facelet ima sledeći oblik:

```
<?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://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Facelet Title</title>
    </h:head>
    <h:body>
        Hello from Facelets
    </h:body>
</html>
```

U ovom automatski generisanom kodu definicija prostora naziva koristi oznaku"h" (za HTML) JSF biblioteke komponenata. To je prikazano sledećim izolovanim kodom:

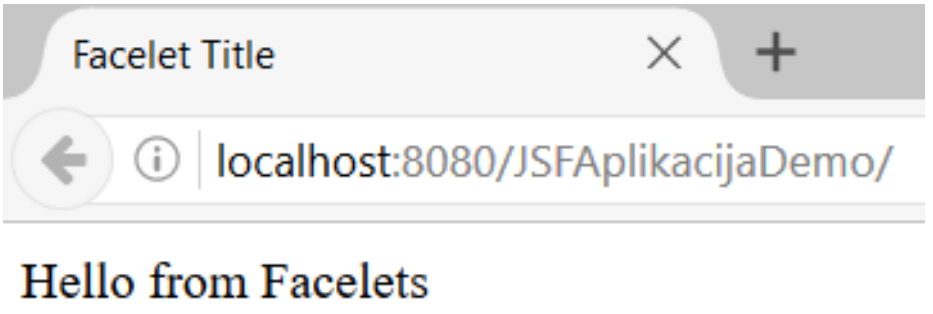
```
xmlns:h="http://xmlns.jcp.org/jsf/html"
```

Takođe, moguće je primetiti da su korišćeni i specijalni JSF tagovi <h:head>i <h:body> koji predstavljaju zamenu za standardne HTML/XHTML <head> i <body> tagove respektivno.

Još jedan prostor naziva koji se često koristi u JSF stranicama određen je prefiksom "f". Ovom oznakom su određeni JSF tagovi koji se ne ugrađuju direktno u JSF stranice već specificiraju elemente drop - down listi, instance, bind akcije i tako dalje. Primer definisanja ovog prostora naziva je priložen sledećim izolovanim kodom:

```
xmlns:f= "http://xmlns.jcp.org/jsf/core"
```

Sada je moguće pokrenuti ovu najjednostavniju JSF veb aplikaciju. U razvojnom okruženju NetBeans IDE je dovoljno kliknuti na opciju Run Project (ili F6), aplikacioni server se pokreće i aplikacija se angažuje. U veb pregledaču je moguće pogledati rezultate izvršavanja kreirane aplikacije. To je prikazano sledećom slikom.



Slika 1.5 Prva JSF aplikacija [izvor: autor]

Poglavlje 2

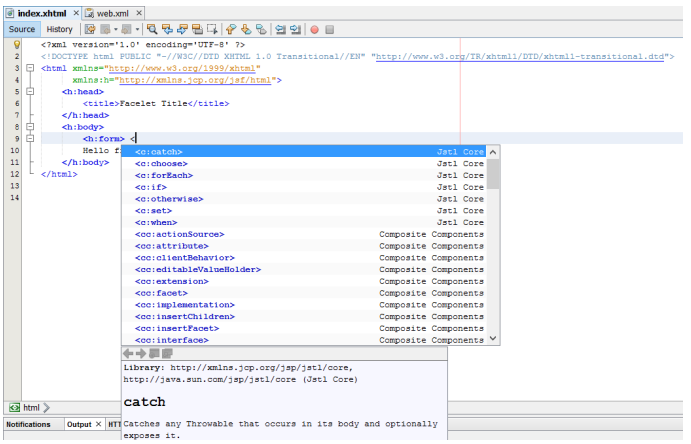
Forme u JSF stranicama

MODIFIKOVANJE AUTOMATSKI GENERISANE JSF STRANICE

Cilj je konkretizovanje zadataka koje bi JSF stranica trebalo da obavi.

Kada se kreira JSF projekat dobija se veoma prosta aplikacija koja u veb pregledaču prikazuje jedan string: "Hello from Facelets" i pored toga ne radi ništa konkretno. Za dobijanje više informacija o načinima funkcionisanja i elementima JSF stranica, moguće je izvršiti određene modifikacije na stranici index.xhtml. Na primer, na ovoj stranici je moguće kreirati formu koja može da pokupi izvesne korisničke podatke.

Svaka JSF form se realizuje tagom <h:form> koji je ekvivalentan tagu <form> standardne HTML stranice. Kucanjem prvih nekoliko karaktera u kodu stranice za <h:form> razvojno okruženje, konkretno NetBeans, započeće davanje sugestija o upotrebi mogućih tagova. Navedeno je pokazano sledećom slikom:



Slika 2.1 Početak kreiranja JSF forme [izvor autor]

Odmah po obeležavanju odgovarajućeg JSF taga razvojno okruženje prikazuje njegovu dokumentaciju (kao na slici). Sledećim kodom je moguče realizovati konkretnu formu u okviru JSF stranice:

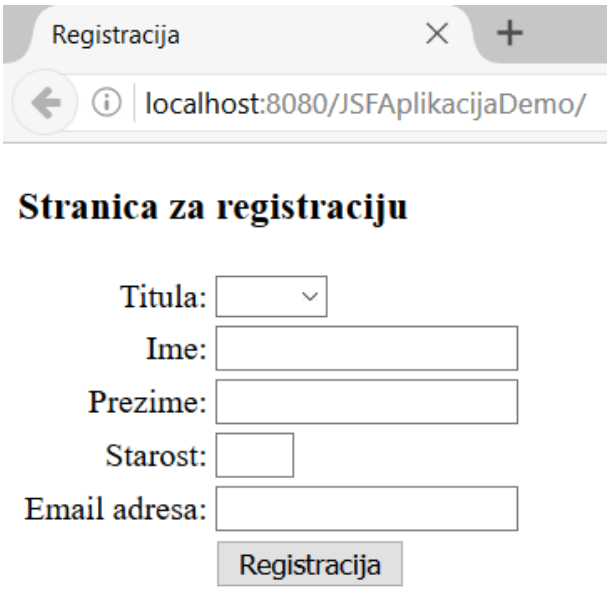
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
```

```
    <h:head>

        <title>Registracija</title>
        <h:outputStylesheet  name="styles.css"/>
</h:head>
    <h:body>
        <h3>Stranica za registraciju</h3>
        <h:form>
            <h:panelGrid columns="3"
                         columnClasses="rightalign,leftalign,leftali
                         gn">
                <h:outputLabel value="Titula: " for="titula"/>
                <h:selectOneMenu id="titula" label="Titula"
                                 value="#{registrationBean.
                                          titula}" >
                    <f:selectItem itemLabel="" itemValue=""/>
                    <f:selectItem itemLabel="St." itemValue="ST"/>
                    <f:selectItem itemLabel="Ma." itemValue="MA"/>
                    <f:selectItem itemLabel="MSci" itemValue="MSCI"/>
                    <f:selectItem itemLabel="Dr" itemValue="DR"/>
                </h:selectOneMenu>
                <h:message for="titula"/>
                <h:outputLabel value="Ime:" for="ime"/>
                <h:inputText id="ime" label="Ime"
                             required="true"
                             value="#{registrationBean.ime}" />
                <h:message for="ime" />
                <h:outputLabel value="Prezime:" for="prezime"/>
                <h:inputText id="prezime" label="Prezime"
                             required="true"
                             value="#{registrationBean.prezime}" />
                <h:message for="prezime" />
                <h:outputLabel for="starost" value="Starost:"/>
                <h:inputText id="starost" label="Starost" size="2"
                             value="#{registrationBean.starost}"/>
                <h:message for="starost"/>
                <h:outputLabel value="Email adresa:" for="email"/>
                <h:inputText id="email" label="Email adresa"
                             required="true"
                             value="#{registrationBean.email}">
                </h:inputText>
                <h:message for="email" />
                <h:panelGroup/>
                <h:commandButton id="registracija" value="Registracija"
                                 action="potvrda" />
            </h:panelGrid>
        </h:form>
    </h:body>
</html>
```

MENADŽER RASPOREDA I CSS

Neophodno je u narednom izlaganju objasniti priloženi JSF kod.

JSF kod, koji je priložen u prethodnoj sekciji, tokom vremena izvršavanja će generisati sledeću stranicu.



Slika 2.2 Kreirana JSF forma [izvor autor]

Sva polja za unos podataka morala su da dudu definisana u okviru JSF taga <h:form>. Takođe, stranica je veoma jednostavno uređena primenom menadžera rasporeda <h:panelGrid> o formi matrice (rešetke) koja je izdeljena na ćelije. Budući da se radi o dobro poznatom menadžeru rasporeda, o njemu ovde neće biti detaljno govora. Dovoljno je napomenuti da se svaka JSF komponenta pakuje u posebnu ćeliju, a da je atributom ***columns,*** taga <h:panelGrid>, određeno koliko će kolona matrica imati - u konkretnom slučaju 3. Kada broj komponenata u vrsti matrice dostigne broj definisanih kolona, nova vrsta za pakovanje novih elemenata automatski se kreira. Svaki red u matrici, koja uređuje redosled komponenata na kreiranoj JSF stranici, sastoji se od: <h:outputLabel> taga, polja za unos i <h:message> taga.

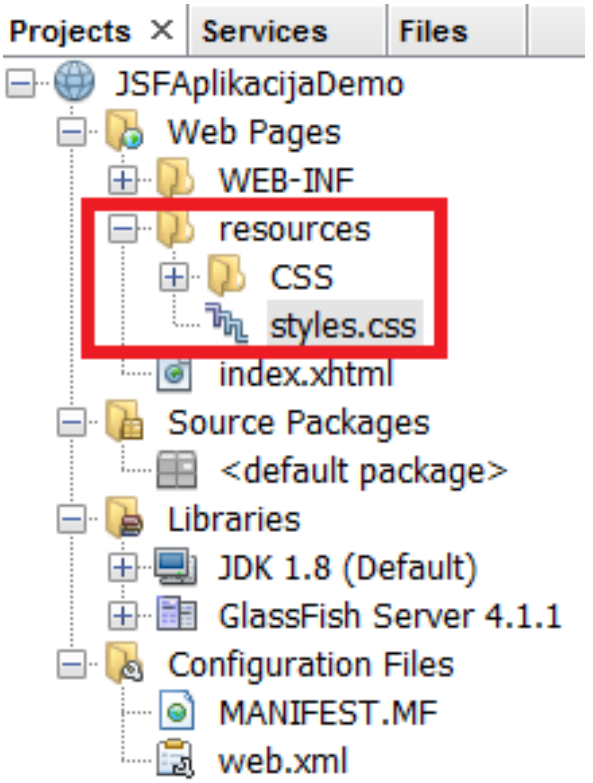
Atribut columnClasses, taga <h:panelGrid>, omogućava primenu CSS stilova na svaku kolonu iz matrice određene navedenim menadžerom rasporeda.
Vrednosti ovog atributa su razdvojene zarezima i definisane su u datoteci styles.css koja je u JSF stranicu uključena elementom *<h:outputStylesheet name="styles.css"/>*.

Pošto matrica ima tri kolone (pogledati kod) prvi stil se primenjuje na prvu, drugi na drugu, a treći na treću kolonu. Za slučaj postojanja četvrte kolone na nju bi bio primenjen prvi, na petu drugi stil i tako dalje. Stilove je moguće primeniti i na vrste, ali bi u tom slučaju bio angažovan atribut rowClasses.

CSS DOPUNSKA RAZMATRANJA

Tag za uključivanje CSS datoteke u JSF stranicu je dostupan od JSF verzije JSF 2.0.

Obavezan položaj CSS datoteke, u Web Pages folderu projekte, prikazan je sledećom slikom.



Slika 2.3 CSS datoteka u JSF projektu [izvor autor]

Tag za uključivanje CSS datoteke u JSF stranicu je dostupan od JSF verzije JSF 2.0. Resursi, kao što su CSS, JavaScript datoteke i slike, mogu biti čuvani u top - level direktorijumu pod nazivom ***resources.*** Ovim resursima JSF tagovi mogu automatski da pristupaju. Da bi kreirana JSF stranica mogla da pristupa datoteci styles.css, kreiran je prvo pomenuti folder resources, a onda u okviru njega odgovarajući podfolder CSS. Datoteka sa stilovima je smeštena u folder CSS.

Tag <h:outputStylesheet> može da sadrži i atribut ***library*** čija vrednost mora da se podudara sa lokacijom na kojoj se CSS fajl čuva. Vrednost atributa ***name*** mora da odgovara nazivu CSS datoteke.

Po analogiji, ukoliko bi u projektu bile korišćene JavaScript datoteke i slike, datoteke bi bile čuvane u podfolderima foldera resources, javascript i images, respektivno.

JSF ELEMENTI

U narednom izlaganju će biti opisani JSF elementi koji su korišćeni u primeru.

Kada se pogleda priloženi kod moguće je uočiti brojne JSF elemente kojima je realizovana forma, odnosno odgovarajući korisnički interfejs koji je dostavljen korisniku učitavanjem kreirane JSF stranice u veb pregledač.

Tagom <h:outputLabel> su određene labele koje označavaju polja za unos teksta koja postoje na formi stranice. Vrednost za atribut ***for*** ovog taga, odgovara vrednosti ***id*** odgovarajućeg polja za unos teksta.

```
<h:outputLabel value="Titula: " for="titula"/>
```

Tagom <h:message> je generisana greška unosa za tekst polje. Vrednost za atribut ***for*** ovog taga, odgovara vrednosti ***id*** odgovarajućeg polja za unos teksta.

```
<h:message for="titula"/>
```

Na formi je moguće uočiti i jednu kontrolu koja je u brojnoj literaturi i implementacijama poznata kao padajuća lista (padajući meni) ili ComboBox. Ovom kontrolom se bira "titula" osobe čiji se podaci unose u formu. Kontrola je realizovana JSF tagom <h:selectOneMenu> koji je ekvivalentan standardnom HTML tagu <select>.

Dakle. moguće je primetiti da brojni JSF tagovi sadrže atribut ***id.*** Ovaj atribut je predstavljen stringom koji jednoznačno označava tag pomoću identifikatora. Ukoliko programer nije specificirao vrednost za atribut id, ona će automatski biti generisana.

Prilikom korišćenja taga <h:label> za definisanje polja za unos teksta ili taga <h:message> za definisanje greške validacije, neophodno je eksplicitno definisati ***id*** atribut polja za unos koji mora da se podudara sa atributima ***for*** navedenih tagova.

Svaki JSF tag za unos sadrži atribut ***label.*** Ovaj atribut se koristi za generisanje greške provere na kreiranoj stranici. Ukoliko ovaj atribut nije specificiran za polje, ono će automatski biti prepoznato, u poruci greške. na osnovu njegovog ***id*** atributa.

Takođe, Svaki JSF tag za unos sadrži atribut ***value.*** U slučaju taga <h:selectOneMenu> ovaj atribut ukazuje na to koja će vrednost iz generisanog HTML taga <select> biti izabrana. Vrednost ovog atributa mora da se podudara sa vrednošću ***itemValue*** atributa nekog od ugnježdenih <f:selectItem> tagova, unutra taga <h:selectOneMenu>.

Vrednost ***itemLabel*** atributa taga <f:selectItem> predstavlja vrednost koju će korisnik moći da vidi u listi (meniju) iz koje vrši izbor. Kada korisnik izabere tu vrednost, vrednost određena atributom ***itemValue*** se šalje na server kada se forma popuni i potvrdi.

DODATNO O ATRIBUTIMA TAGOVA

Neki od inputText tagova zahtevaju vrednost.

Neki od <h:inputText> tagova zahtevaju vrednost. Ovi tagovi poseduju atribute ***required*** postavljene na vrednost ***true***. To znači da je obaveza korisnika da unese vrednost u polje koje odgovara ovom JSF elementu. Ovaj atribut je opcionalan i ukoliko nije obavezan unos u polje on može da se izostavi ili da bude podešen na ***false*** (ovo je i podrazumevana vrednost atributa)***.*** Navedeno je pokazano sledećim izolovanim listingom:

```
<h:inputText id="email" label="Email adresa"
                              required="true"
                              value="#{registrationBean.email}">
                </h:inputText>
```

Ako se dalje pogleda priloženi kod moguće je primetiti da je tag <h:panelGroup/> dodat u poslednji red matrice koja određuje raspored GUI elemenata. Svhra navedenog je dodavanje više JSF tagova u jednu ćeliju <h:panelGrid> rasporeda. U konkretnom slučaju na ovaj način je omogućeno preskakanje jedne ćelije (dodavanjem prazne ćelije), dodavanje kontrole <h:commandButton> i njeno poravnanje sa poljima za unos teksta na kreiranoj formi. Zadatak dugmeta je da potvrdi popunjenu formu i omogući slanje podataka na server. Atribut dugmeta ***value*** ima zadatak da pokaže korisniku tekst na kreiranom dugmetu, a atribut ***action*** je zadužen za određivanje stranice koja će biti prikazana nakon klika na dugme.

U konkretnom slučaju, radi se o statičkoj navigaciji. Klikom na dugme "Registracija" ka datoteci pod nazivom potvrda.xhtml.

```
<h:commandButton id="registracija" value="Registracija"
                                  action="potvrda" />
```


```
<h:commandButton id="registracija" value="Registracija"
                                  action="potvrda" />
```


Alternativno, moguće je koristiti dinamičku navigaciju. U tom slučaju, vrednost atributa ***action,*** dugmeta za potvrdu forme, odgovara vrednosti izraza koji odgovara metodi i povratnom stringu CDI zrna. To znači da povratna vrednost navedene i izvršene metode mora da odgovara nazivu stranice ka kojoj se vrši navigacija.

Poglavlje 3

Kreiranje CDI zrna

DODAVANJE KLASE ZRNA U JSF APLIKACIJU

CDI zrna prihvataju korisničke podatke.

CDI (Context and Dependency Injection) zrna su klasična Java zrna čiji je zadatak prihvatanje podataka koje je korisnik uneo u JSF aplikaciji. Budući da se radi Java klasi, njeno kreiranje se odvija po istom scenariju kao i za ostale Java klase.

U razvojnom okruženju, desnim klikom na Source Package aktuelnog projekta, se bira opcija New i nakon toga se bira opcija Java Class. Otvara se prozor u kojem se unosi naziv klase i paketa: RegistrationBaan i com.metropolitan, respektivno. Generisana je prazna klasa priložena sledećim listingom:

```
package com.metropolitan;

/**
 *
 * @author Vladimir Milicevic
 */
public class RegistrationBean {

}
```

Da bi kreirana Java klasa postala šablon za CDI (obeleženo, imenovano) zrno neophodno je izvršiti obeležavanje klase anotacijom ***@Named.*** Po osnovnim podešavanjima naziv CDI zrna u potpunosti odgovara nazivu klase sa razlikom što počinje mail slovom. Dakle, ovoj klasi odgovara CDI zrno registrationBean kojeg je moguće uočiti u priloženom kodu datoteke index.xhtml. Podrazumevano ime može biti i redefinisano kada se u anotaciji ***@Named*** podesi atribut ***value*** na željenu vrednost.

CDI zrna mogu pokrivati različite oblasti (sledeća tabela):

Oblast	Anotacija
Request	@RequestScoped
Session	@SessionScoped
Conversation	@ConversationScoped
Application	@SessionScoped
Dependent	@Dependent
Flow	@FlowScoped

Slika 3.1 Oblasti CDI zrna [izvor autor]

Sledi objašnjenje za pomenute oblasti i odgovarajuće anotacije:

- Request - znači da je zrno dostupno isključivo u jednom HTTP zahtevu;
- Session - zrno dostupno isključivo u jednoj korisničkoj HTTP sesiji;
- Conversation - zrno dostupno preko više HTTP zahteva;
- Application - zrno je dostupno svim korisnicima aplikacije preko njihovih sesija;
- Dependent - zrno može imati zavisan pseudo - kod, a to podrazumeva umetanje nove instance svaki puta kada se zahteva;
- FLow - zrno je dostupno kroz specifične JSF tokove.

KODIRANJE KLASE CDI ZRNA

Na osnovu prethodnih izlaganja neophodno je konkretizovati definiciju kreirane klase.

Ako se vrati fokus na prethodno izložene anotacije trebalo bi napomenuti i da su sve anotacije, izuzev ***@FlowScoped***, definisane u okviru paketa javax.enterprise.context. Anotacija ***@FlowScoped*** je definisana unutar paketa pod nazivom javax.faces.flow. Dakle, neophodno je kreiranu klasu obeležiti odgovarajućim anotacijama i obezbediti odgovarajuće ***import*** instrukcije za podršku. Kada se dodaju anotacije, trebalo bi imati na umu da klasa implementira CDI zrno ***(@Named***) i da osluškuje zahtev korisnika koji popunjava formu (***@RequestScope***).

Privremeni oblik klase RegistrationBean priložen je sledećim listingom:

```
package com.metropolitan
;
import javax.enterprise.context.RequestScoped;
import javax.inject.Named;

@Named
@RequestScoped
public class RegistrationBean {
}
```

Ukoliko razvojno okruženje ne može da pronađe anotaciju @RequestScope, neophodno je, desnim klikom na Libraries, izborom opcije Add JAR /Folder, izabrati cdi-api.jar iz foldera modules GlassFish instalacije.

Konačno, moguće je konkretizovati i finiširati definiciju posmatrane klase dodavanjem osobina koje će čuvati vrednosti koje je korisnik uneo.

Sledećim listingom je priložen konačan kod klase koja odgovara CDI zrnu:

```
package com.metropolitan;

/**
 *
 * @author Vladimir Milicevic
 */
import javax.enterprise.context.RequestScoped;
import javax.inject.Named;

@Named
@RequestScoped
public class RegistrationBean {
private String titula;
private String ime;
private String prezime;
private Integer staros;
private String email;
//geteri i seteri su izostavljeni zbog preglednosti
//bice prilozeni u materijalima za vezbe
}
```

Moguće je primetiti da nazivi osobina zrna u potpunosti odgovaraju nazivima korišćenim u JSF stranici index.xhtml za podatke koje unosti korisnik. Ovi nazivi moraju da se podudaraju da bi JSF stranica znala koje vrednosti može da mapira.

Poglavlje 4

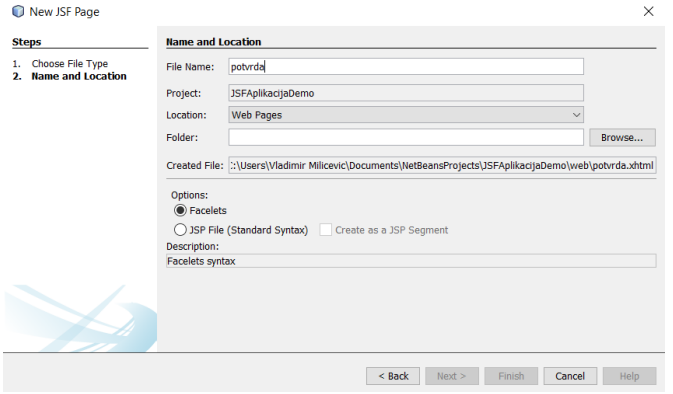
JSF stranica za potvrdu unosa

KREIRANJE STRANICE ZA POTVRĐIVANJE

Kreiranje stranice za prikaz unetih korisničkih podataka.

Kada korisnik unese sve potrebne podatke na početnoj stranici i potvrdi unos na formi stranice, neophodno je na odgovarajućoj stranici za potvrđivanje prikazati podatke koje je korisnik uneo. Vrednosti koje odgovaraju atributima ***values*** svakog polja za unos teksta forme, biće pridruženi odgovarajućim osobinama kreiranog CDI zrna. U nastavku, zadatak je jednostavan. Stranica za potvrđivanje će biti generisana kao serija <h:outputText> JSF tagova koji prikazuju vrednosti koje kreirano CDI zrno čuva.

U razvojnom okruženju, u konkretnom slučaju NetBeans, klikom na File, a zatim izborom opcije New File, pa kategorije JavaServer Faces i tipom datoteke JSF Page, otvara se prozor za kreiranje nove JSF stranice kojoj će biti dodeljen naziv potvrda.xhtml (sledeća slika).



Slika 4.1 Nova JSF stranica [izvor autor]

Još jednom je neophodno proveriti da li dodeljeni naziv ovoj datoteci odgovara vrednosti atributa ***action*** dugmeta forme koja se nalazi na stranici index.xhtml. U nastavku, automatski generisana stranica je modifikovana u skladu sa istaknutim zahtevima u prethodnom izlaganju. Nakon modifikacije stranicu sačinjava kod koji je priložen sledećim listingom:

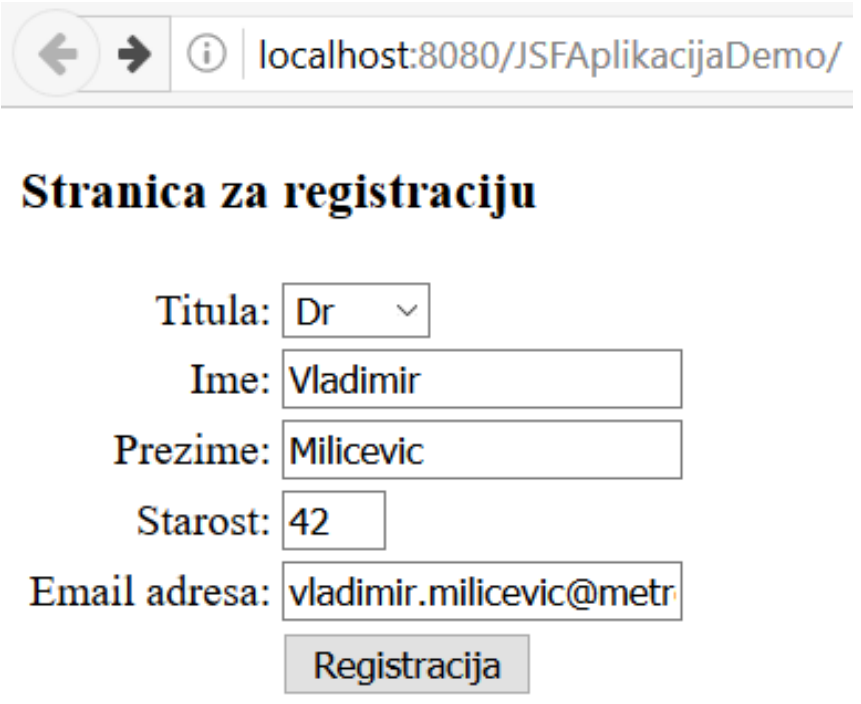
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Stranica za potvrdu</title>
        <h:outputStylesheet name="styles.css"/>
```

```
    </h:head>
    <h:body>
        <h2>Stranica za potvrdu</h2>
        <h:panelGrid columns="2"
                     columnClasses="rightalign-bold,leftalign">
            <h:outputText value="Titula:"/>
            ${registrationBean.titula}
            <h:outputText value="Ime:"/>
            ${registrationBean.ime}
            <h:outputText value="Prezime:"/>
            ${registrationBean.prezime}
            <h:outputText value="Starost:"/>
            ${registrationBean.starost}
            <h:outputText value="Email adresa:"/>
            ${registrationBean.email}
            </h:panelGrid>
        </h:body>
</html>
```

IZVRŠAVANJE APLIKACIJE I PRIKAZIVANJE UNETIH PODATAKA

Na kraju je neophodno proveriti funckionalnost kreirane JSF aplikacije.

Sa stranicom za potvrđivanje unetih podataka zaokružena je konstrukcija kreirane veb aplikacije. Najlakši način da se aplikacija testira je klik na Run opciju u razvojnom okruženju. Aplikacija se pokreće, učitava se stranica index.xhtml sa odgovarajućom formom koju je neophodno popuniti kao na sledećoj slici.



Slika 4.2 Popunjena forma sa početne stranice [izvor autor]

Klikom na dugme pod nazivom "Registracija", svi uneti podaci se poveravaju na čuvanje CDI zrnu čija klasa predstavlja sastavni deo aplikacije. Zrno se ponaša kao posrednik između dve stranice, za registraciju i potvrđivanje. Stranica za potvrđivanje, određena je vrednošću action, dugmeta "Registracija" dobija od zrna odgovarajuće podatke i prikazuje ih na ekranu na način prikazan sledećom slikom.

Slika 4.3 Prikazivanje podataka na stranici za potvrđivanje [izvor autor]

Ukoliko se u veb pregledaču pojave podaci na način prikazan prethodnom slikom, posao je uspešno obavljen i može se preći na izučavanje naprednijih JSF tema.

Poglavlje 5

JSF Validacija

KLASA VALIDATOR

Klasa validator mora da implementira interfejs javax.faces.validator.Validator.

Da bi forma pokupila podatke koji su pravilno formatirani i odgovarajućeg tipa, neophodno je u projekat dodati validator klasu. Ova klasa mora da implementira interfejs javax.faces.validator.Validator. Navedeni interfejs sadrži jednu metodu validate() koja preuzima sledeća tri parametra, instance klasa: javax.faces.context.FacesContext, javax.faces.component.UIComponent (sadrži JSF komponentu koja se proverava) i java.lang.Object (sadrži vrednost koju je korisnik uneo za komponentu), respektivno.

Sledećim listingom je moguće ilustrovati email validator:

```
package com.metropolitan;

/**
 *
 * @author Vladimir Milicevic
 */
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.component.html.HtmlInputText;
import javax.faces.context.FacesContext;
import javax.faces.validator.FacesValidator;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;

@FacesValidator(value = "emailValidator")
public class EmailValidator implements Validator {

    @Override
    public void validate(FacesContext facesContext,
            UIComponent uiComponent, Object value) throws
            ValidatorException {
        Pattern pattern = Pattern.compile("\\w+@\\w+\\.\\w+");
        Matcher matcher = pattern.matcher(
                (CharSequence) value);
        HtmlInputText htmlInputText
                = (HtmlInputText) uiComponent;
```

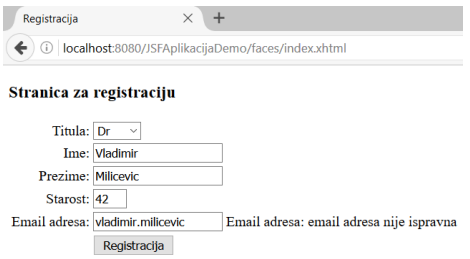
```
        String label;
        if (htmlInputText.getLabel() == null
                || htmlInputText.getLabel().trim().equals("")) {
            label = htmlInputText.getId();
        } else {
            label = htmlInputText.getLabel();
        }
        if (!matcher.matches()) {
            FacesMessage facesMessage
                    = new FacesMessage(label
                            + ": email adresa nije ispravna");
            throw new ValidatorException(facesMessage);
        }
    }
}
```

U ovom primeru metoda validate() vrši upoređivanje regularnog izraza sa vrednošću JSF komponente koja se proverava. Ako je prisutno poklapanje, validacija je uspešna. U suprotnom, biće izbačena instanca izuzetka javax.faces.validator.ValidatorException.

Takođe, važno je napomenuti da klasa validator mora da bude obeležena anotacijom ***@FacesValidator.*** Vrednost anotacije predstavlja ***id*** koji će biti upotrebljen da referencira na validator - a u JSF stranici.

Da bi ova klasa obavila svoju ulogu, neophodno je vratiti se u datoteku index.xhtml, i za polje za unos teksta email, obaviti modifikaciju koja je priložena sledećim listingom. Ukoliko email adresa nije korektno uneta, JSF stranica će reagovati kao na slici broj 4.

```
<h:inputText id="email" label="Email Address"
required="true" value="#{registrationBean.email}">
<f:validator validatorId="emailValidator"/>
</h:inputText>
```



Slika 5.1 Angažovanje email validatora [izvor autor]

VALIDACIJA U JSF STRANICAMA

Greške validacije, za obavezna polja unosa, automatski se generišu.

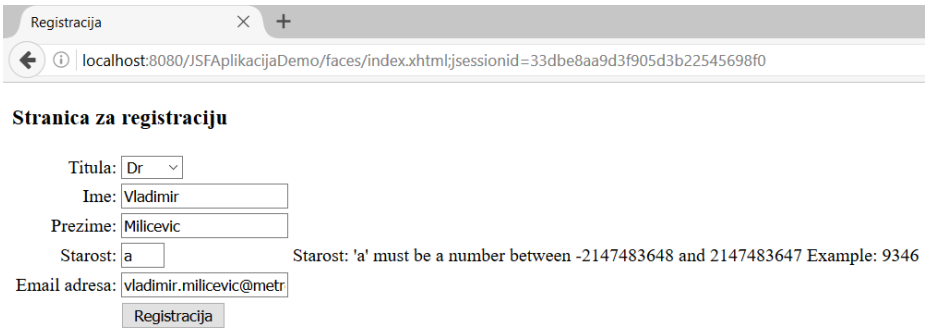
U prethodnim izlaganjima je bilo govora o jednom posebnom atributu pod nazivom ***required.*** Ovaj atribut je od posebnog značaja kada su u pitanju polja za unos teksta kreiranih formi

neke JSF stranice. Atribut označava polja koja su obavezna za unosi i ukoliko neko od njih nije popunjeno javiće se greška kao na sledećoj slici.

Slika 5.2 Obavezno polje nije popuneno [izvor autor]

Za generisanje poruke sa informacijom o grešci zadužen je JSF tag <h:message> odgovarajućeg polja koje je obavezno, a nije popunjeno prilikom potvrđivanja forme.

Dalje, ako se fokusiramo na polje forme, zaduženo za unos podatka kojim je predstavljena starost korisnima, i u njega unesemo podatak koji nije tipa ***Integer,*** javiće se sledeća greška validacije koja je, takođe, automatski generisana.



Slika 5.3 Unos podatka neodgovarajućeg tipa podataka [izvor autor]

Ovde je moguće uočiti jedan nedostatak kreirane aplikacije. Ako se za godine unese negativan broj, to će za aplikaciju biti u redu, jer se uklapa u predviđeni tip podataka, ali u realnim okolnostima to nema nikakvog smisla.

Takođe, kao vrednost za email adresu će proći bilo koji ***String*** objekat bez ikakve provere da li je on korektno unet. Ne postoji ugrađen mehanizam validacije za proveru korektnosti unosa email adrese.

Upravo u nastavku, neophodno je pokazati mehanizme i načine koji obezbeđuju da forma pokupi pravilno unete i formatirane podatke i kao takve da ih nakon potvrđivanja pošalje na server.

Poglavlje 6

Facelet šabloni

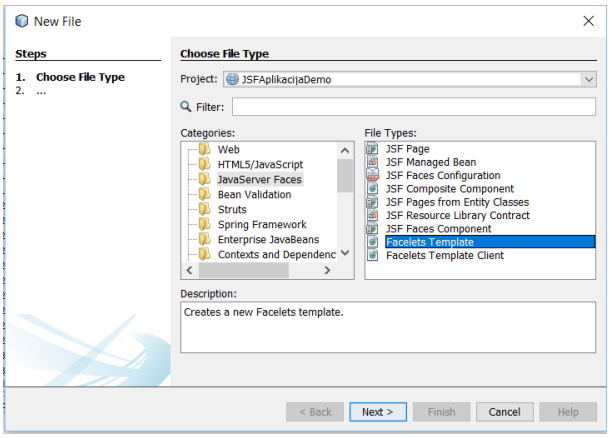
MEHANIZAM ŠABLONA

Prednost Facelet - u odnosu na JSP ogleda se u primeni mehanizma šablona.

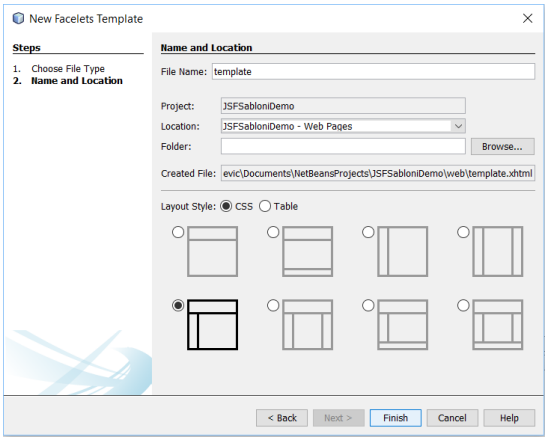
Prednost Facelet - u odnosu na JSP ogleda se u primeni mehanizma šablona. Šabloni dozvoljavaju da se izgled i raspored JSF stranice specificira na jednom mestu. Takođe, primenom šablona postiže se veći stepen održivosti kreirane veb aplikacije budući da se sve izmene u rasporedu komponenata stranice dešavaju na jednom mestu. Ukoliko se u jednoj tački stranice traži sprovođenje neke izmene, kao što je na primer dodavanje futera ili pomeranje kolone sa leve na desnu stranu stranice, neophodno je izvršiti samo izmenu u šablonu koja će se reflektovati na sve klijente šablona.

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

Neophodno je, u nastavku, demonstrirati kako se dodaje facelet šablon u JSF projekat. Klikom na opciju File, u razvojnom okruženju NetBeans, otvara se meni u okviru kojeg se bira opcija New File. Otvara se novi prozor pod nazivom New File gde se bira kategorija JavaServer Faces i konačno, u istom prozoru bira se za tip datoteke Facelets Templates. Navedeni prozor i izbor iz njega je prikazan slikom broj 1. Dalje, klikom na Next otvara se prozor u okviru kojeg se definiše naziv šablona i bira neki od predefinisanih šablona (Slika 2)



Slika 6.1 Dodavanje Facelet šablona u projekat [izvor autor]



Slika 6.2 Prozor New Facelets Template ([izvor autor]

DATOTEKA ŠABLONA

Kreiranu datoteku šablona je nophodno konkretizovati odgovarajućim modifikacijama koda.

Na prethodno priloženoj slici moguće je bilo primetiti da razvojno okruženje NetBeans IDE nudi mogućnost izbora između HTML tabela i CSS - a za određivanje rasporeda. Većina savremenih veb aplikacija podrazumeva CSS kao prirodan izbor. Neka je, u ovom slučaju, izbor pao na raspored koji uključuje polje zaglavlja (header), kolonu na levoj strani i glavno polje (izbor je prikazan Slikom 2). Konačno, u prozoru New Facelets Template, klikom na Finish se automatski generiše kod šablona koji je priložen sledećim listingom:

```
<?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:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

    <h:head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <h:outputStylesheet name="./css/default.css"/>
        <h:outputStylesheet name="./css/cssLayout.css"/>
        <title>Facelets Template</title>
    </h:head>

    <h:body>


        <div id="top" class="top">
            <ui:insert name="top">Top</ui:insert>
        </div>
        <div>
            <div id="left">
                <ui:insert name="left">Left</ui:insert>
```

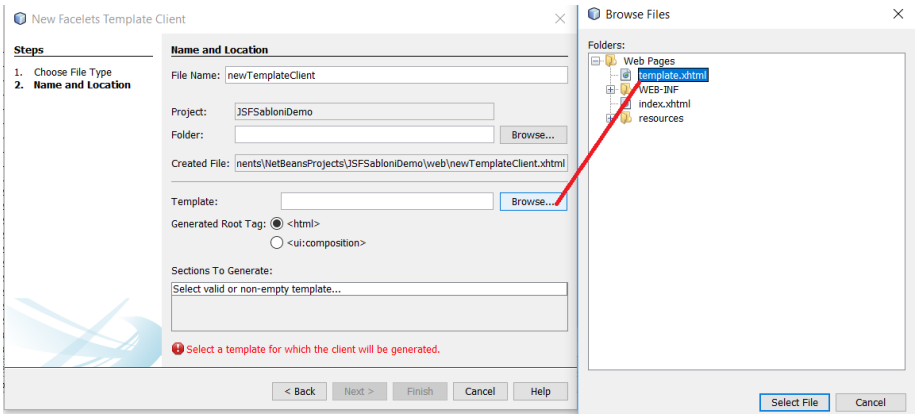
```
            </div>
            <div id="content" class="left_content">
                <ui:insert name="content">Content</ui:insert>
            </div>
        </div>
    </h:body>

</html>
```

Na prvi pogled je moguće primetiti da se šablon ne razlikuje mnogo od standardne JSF stranice. Takođe, moguće je primetiti da koristi prostor naziva *xmlns:ui=http://xmlns.jcp.org/jsf/facelets* koji omogućava primenu JSF taga <ui:insert>. Sadržaj ovog taga biće zamenjen sadržajima odgovarajućih <ui:define> tagova klijenata šablona.

UPOTREBA ŠABLONA

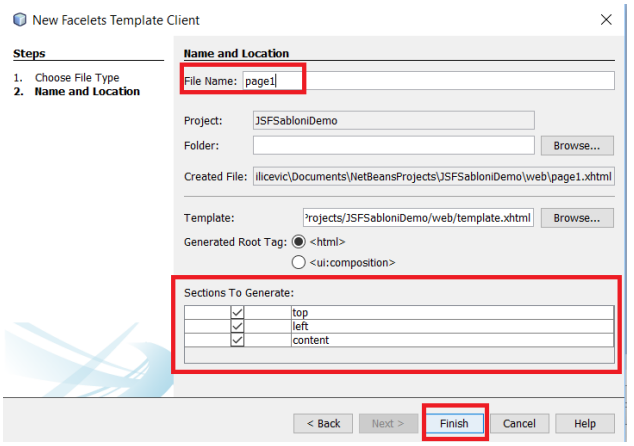
Za upotrebu šablona neophodna je implementacija klijenata šablona.

Da bi kreirani šablon mogao da bude upotrebljen neophodno je kreirati odgovarajući Facelets template client. Postupak je veoma sličan pretohodnom. Klikom na opciju File, u razvojnom okruženju NetBeans, otvara se meni u okviru kojeg se bira opcija New File. Otvara se novi prozor pod nazivom New File gde se bira kategorija JavaServer Faces i konačno, u istom prozoru bira se za tip datoteke Facelets Template Clients. Klikom na Next otvara se prozor u okviru kojeg se definiše naziv datoteke klijenta šablona, a takođe se vrši dodeljivanje kreiranog šablona ovoj datoteci (sledeća slika).



Slika 6.3 Dodela kreiranog šablona klijentu šablona [izvor autor]

U nastavku je neophodno završiti definiciju ove datoteke. Poslednji koraci su zadavanje njenog naziva, na promer page1, i klik na dugme Finish. Navedeni elementi su obeleženi na sledećoj slici, zajedno sa učitanim šablonom.



Slika 6.4 Završetak definicije klijenta šablona. [izvor autor]

Kao posledica ovih akcija javlja se sledeći automatski generisan kod:

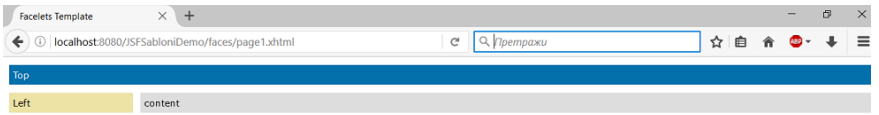
```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/
TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

    <body>

        <ui:composition template="./template.xhtml">

            <ui:define name="content">
                content
            </ui:define>

        </ui:composition>

    </body>
</html>
```

UPOTREBA ŠABLONA - DODATNA RAZMATRANJA

Aplikacija se testira i uočava se ugrađeni šablon rasporeda stranice.

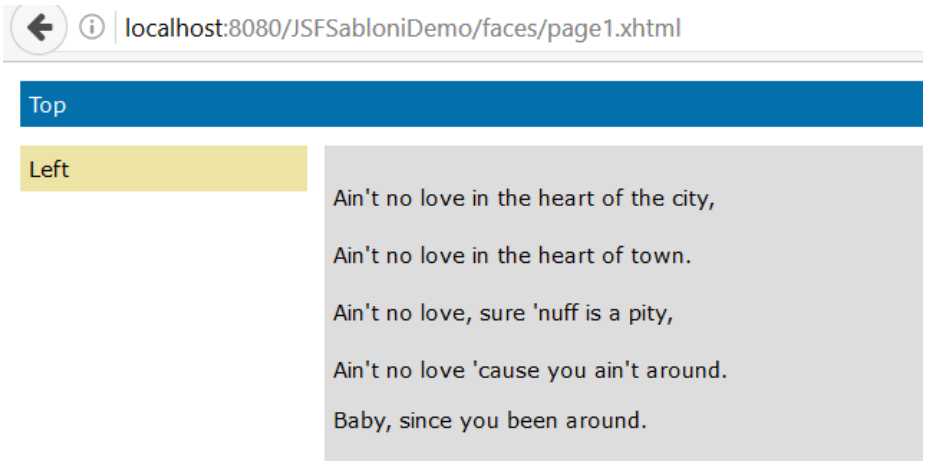
Kao što je moguće primetiti iz prethodno priloženog koda, kreirana datoteka klijenta šablona takođe uključuje prostor naziva *xmlns:ui=*"http://xmlns.jcp.org/jsf/facelets". Dalje, neophodno je napomenuti da u ovoj datoteci tag <ui:composition> mora da bude roditeljski tag za bilo koji drugi tag koji pripada ovom prostoru naziva. Sve što se nalazi izvan ovog taga neće biti postavljeno u stranicu i umesto toga će se pojaviti odgovarajući sadržaj šablona. Tagom <ui:define>, klijenta šablona, se umeće sadržaj koji odgovara sadržaju taga <ui:insert> šablona. Vrednost atributa ***name u*** tagu <ui:define> mora da se podudara sa odgovarajućom vrednošću u <ui:insert> tagu šablona.

Sada je moguće testirati kreirane datoteke projekta. Nakon prevođenja, primenom razvojnog okruženja, u veb pregledaču, u polju za unos, poziva se sledeći link http://localhost:8080/JSFSabloniDemo/faces/page1.xhtml i rezultat poziva je priložen sledećom slikom.



Slika 6.5 Primena šablona [izvor autor]

Sada je moguće uneti izmene u klijent šablona i posmatrati kako će se ponašati kreirana stranica page1. Između tagova <ui:define name = "content"> i </ui:define> ubačen je deo teksta jedne pesme i rezultat se vidi na sledećoj slici i u kodu koji sledi iza slike.



Slika 6.6 Zamena vrednosti content konkretnim sadržajem [izvor autor]

```
<?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:ui="http://xmlns.jcp.org/jsf/facelets">

    <body>

        <ui:composition template="./template.xhtml">

            <ui:define name="content">
                <br/>Ain't no love in the heart of the city,</br>
                <br/>Ain't no love in the heart of town.</br>
                <br/>Ain't no love, sure 'nuff is a pity,</br>
                <br/>Ain't no love 'cause you ain't around.</br>
                <p>Baby, since you been around.</p>
```

```
                </ui:define>

        </ui:composition>

    </body>
</html>
```

KOREKCIJE U DATOTECI ŠABLONA

Svaka korekcija u šablonu reflektovaće se na klijente šablona.

U prethodnoj aktivnosti izvršeno je korigovanje datoteke page.xhtml na način da ona prikazuje određeni tekst. Sada je moguće uraditi i određene korekcije na samom šablonu template.xhtml i posmatrati kako će se to reflektovati na klijent šablona. Budući da klijent šablona ne poseduje elemente koje odgovaraju top i left sekcijama šablona, njihovo redefinisanje je neophodno izvršiti u samom šablonu.

Top sekcija može biti redefinisana na sledeći način:

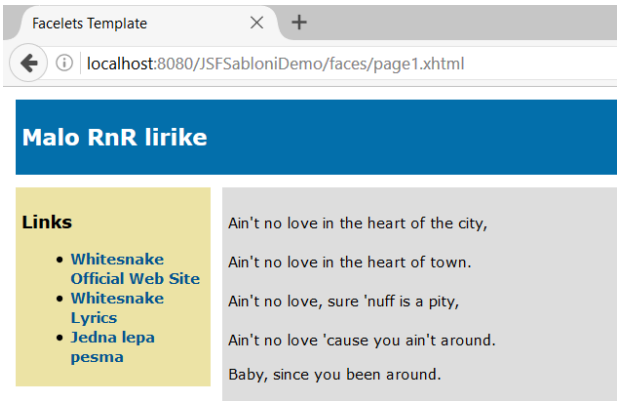
```
<div id="top" class="top">
            <ui:insert name="top">
                <h2>Malo RnR lirike</h2>
            </ui:insert>
        </div>
```

Left sekcija može biti redefinisana na malo složeniji način:

```
<div id="left">
                <ui:insert name="left"><h3>Links</h3>
                    <ul>
                        <li>
                            <h:outputLink value="http://www.whitesnake.com/">
                                <h:outputText value="Whitesnake Official Web Site"/>
                            </h:outputLink>
                        </li>
                        <li>
                            <h:outputLink value="http://www.azlyrics.com/w/
whitesnake.html">
                                <h:outputText value="Whitesnake Lyrics"/>
                            </h:outputLink>
                        </li>
                        <li>
                            <h:outputLink value="https://www.youtube.com/
watch?v=MA3DNlBMNL0">
                                <h:outputText value="Jedna lepa pesma"/>
                            </h:outputLink>
                        </li>
```

```
                    </ul></ui:insert>
            </div>
```

Budući da klijent šablone ne redefiniše sekcije top i left, nasledиće ih direktno od šablona. Nakon navedenih korekcija u šablonu i ponovnog prevođenja kreirane veb aplikacije, pozivom linka stranice page1.xhtml: http://localhost:8080/JSFSabloniDemo/faces/page1.xhtml, moguće je primetiti sasvim nov sadržaj stranice page1.xhtml. Navedeno je ilustrovano sledećom slikom.



Slika 6.7 Klijent šablona nakon korekcija šablona [izvor autor]

Poglavlje 7

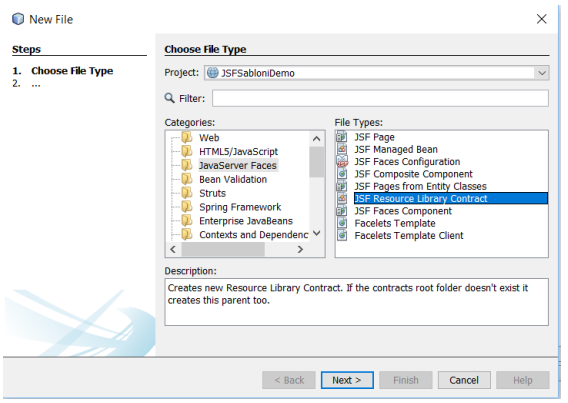
Biblioteka ugovora resursa

JSF 2.2 - NOVITET

Biblioteka ugovora resursa je Java EE novitet.

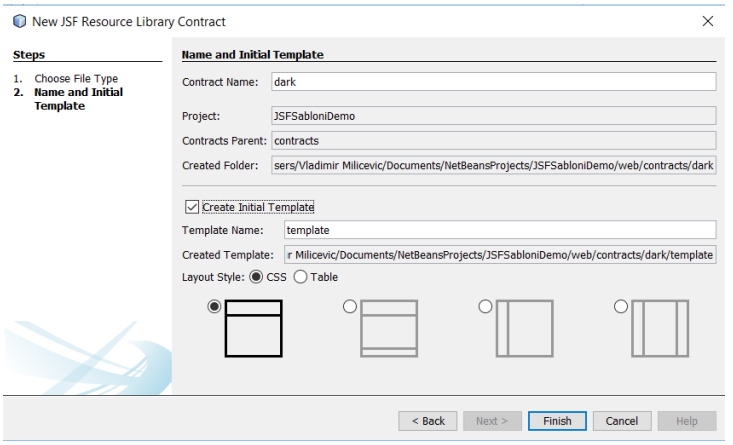
Biblioteka ugovora resursa (Resource library contracts) predstavlja potpuno nov JSF 2.2 alat. Osmišljena je kao podrška za kreiranje Facelet šablona za kreiranje "tematskih" veb aplikacija. Na primer, moguće je da postoji aplikacija koja servisira više potrošača ili je moguće kreirati korisnički interfejs tako da svaki korisnik vidi logo svoje kompanije posle logovanja na sistem. Kao alternativa, moguće je izabrati korisnika iz predefinisanog skupa tema, a to je upravo zadatak kojim će se baviti ovaj deo lekcije.

Za početak, biće kreirana biblioteka resursa, u aktuelnom projektu, izborom opcije File, a zatim New File, otvara se prozor pod nazivom New File. U ovom prozoru, bira se kategorija JavaServer Faces i tip datoteke JSF Resource Library Contracts (pogledati sledeću sliku).



Slika 7.1 Kreiranje datoteke JSF Resource Library Contracts [izvor autor]

Klikom na dugme Next, otvara se prozor u kojem se kreira inicijalni šablon teme zadaje naziv datoteke.



Slika 7.2 Izbor šablona i naziva JSF Resource Library Contracts fajla [izvor autor]

Ideja je da razvojno okruženje generiše inicijalni šablon koji će kasnije biti modifikovan na način da će rezultujuća stranica imati tamnu pozadinu sa svetlim slovima. Upravo će to predstavljati tamnu temu.

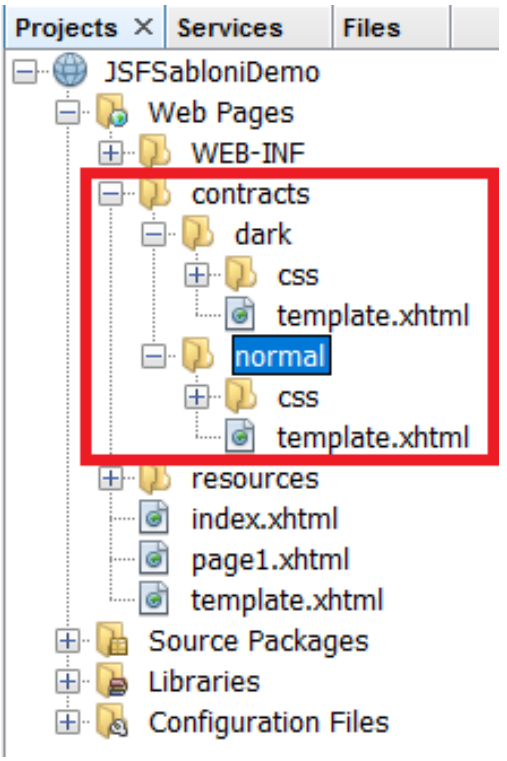
Na isti način je moguće kreirati temu koja alternativno biti korišćena u odnosu na temu dark, neka se ova tema naziva nomal.

U narednom izlaganju biće više reči o radu sa kreiranim temama i novoj hijerarhiji aktuelnog projekta koji ilustruje tekuću problematiku.

KORIŠĆENJE UGOVORA

Razvojno okruženje je kreiralo dva fajla u zajedničkom folderu contracts.

Nakon kreiranja pomenutih datoteka tema, razvojno okruženje je kreiralo dva fajla u zajedničkom folderu contracts i odgovarajućim podfolderima (sledeća slika).



Slika 7.3 Nova hijerarhija projekta [izvor autor]

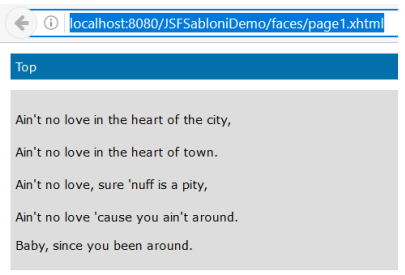
Sada je neophodno prikazati kako je moguće primeniti temu na kreiranu datoteku page1.xhtml. Za početak je u okviru taga <html> neophodno dodati sledeće: xmlns:f="http://xmlns.jcp.org/jsf/core". Zatim, u okviru bloka <body> vrše se korekcije priložene sledećim kodom:

```
<body>
<!--
        <ui:composition template="./template.xhtml">
-->
           <f:view contracts="normal">
           <ui:composition template="/template.xhtml">
            <ui:define name="content">

               ovde ide sadržaj za prikazivanje

                </ui:define>

        </ui:composition>

            </f:view>

    </body>
```

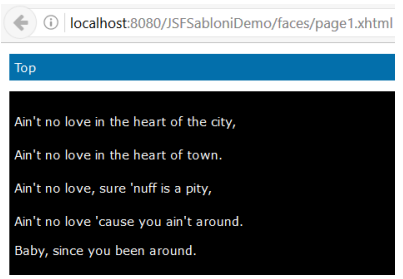
Za korišćenje ugovora, umesto prethodno kreiranih Facelet šablona, neophodno je enkapuslirati<ui:composition> u <f:view> tag.

Atribut ***contract*** taga <f:view> poseduje vrednost koja se podudara sa nazivom biblioteke ugovora resursa koji će biti upotrebljen u stranici.

Prevođenjem aplikacije i aktiviranjem linka http://localhost:8080/JSFSabloniDemo/faces/page1.xhtml, u veb pregledaču se pojavljuje sledeća stranica.



Slika 7.4 Korišćenje "normal" teme [izvor autor]



Slika 7.5 Korišćenje "dark" teme [izvor autor]

UPRAVLJANJE TEMAMA

Cilje je kreiranje stranice na kojoj je moguće izabrati željenu temu za prikazivanje sadržaja.

Da bi dark tema, sa prethodne slike, bila prikazana, napravljen su sledeće korekcije u izvornom cssLayout.css fajlu iz foldera dark.

```
#top {
    position: relative;
    background-color: #036fab;
    color: white;
    padding: 5px;
    margin: 0px 0px 10px 0px;
}

#bottom {
    position: relative;
    background-color: #c2dfef;
    padding: 5px;
    margin: 10px 0px 0px 0px;
}

#left {
    float: left;
    background-color: #ece3a5;
    padding: 5px;
```

```
    width: 150px;
}

#right {
    float: right;
    background-color: #ece3a5;
    padding: 5px;
    width: 150px;
}

.center_content {
    position: relative;
    background-color: black;
    color: white;
    padding: 5px;
}

.left_content {
    background-color: black;
    padding: 5px;
    margin-left: 170px;
}

.right_content {
    background-color: black;
    padding: 5px;
    margin: 0px 170px 0px 170px;
}

#top a:link, #top a:visited {
  color: white;
  font-weight : bold;
  text-decoration: none;
}

#top a:link:hover, #top a:visited:hover  {
  color: black;
  font-weight : bold;
  text-decoration : underline;
}
```

Sada je cilj da se kreira, za aktuelnu aplikaciju, CDI zrno koje će omogućiti dinamičko biranje teme na JSF stranici koja je zadužena za prikazivanje konkretnog sadržaja. Neka se ova klasa naziva SelektorTema.java i priložena je sledećim listingom.

```
package com.metropolitan.teme;

import javax.enterprise.context.RequestScoped;
import javax.inject.Named;

/**
 *
```

```
 * @author Vladimir Milicevic
 */
@Named
@RequestScoped
public class SelektorTema {
private String themeName = "normal";
public String getThemeName() {
return themeName;
}
public void setThemeName(String themeName) {
this.themeName = themeName;
}
}
```

MODIFIKACIJA ŠABLONA KLIJENTA

Neophodne su modifikacije šablona klijenta za korišćenje selektora tema.

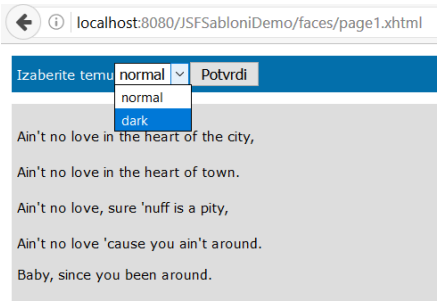
Da bi bilo moguće korišćenje selektora tema, u JSF stranici page1.xhtml, neophodno je u kodu ove stranice ukazati na kreirano CDI zrno, te izvršiti odgovarajuće korekcije. Navedeno je priloženo sledećim listingom.

```
<?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:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

    <body>
<!--
        <ui:composition template="./template.xhtml">
-->
  <!--
        <f:view contracts="dark">
  -->
          <f:view contracts="#{selektorTema.themeName}">
          <ui:composition template="/template.xhtml">
              <ui:define name="top">
                  <h:form>
                      <h:outputLabel for="themeSelector"
                                     value="Izaberite temu"/>
                      <h:selectOneMenu id="selektorTema"
                                       value="#{selektorTema.themeName}">
                          <f:selectItem itemLabel="normal"
                                        itemValue="normal"/>
                          <f:selectItem itemLabel="dark"
                                        itemValue="dark"/>
```

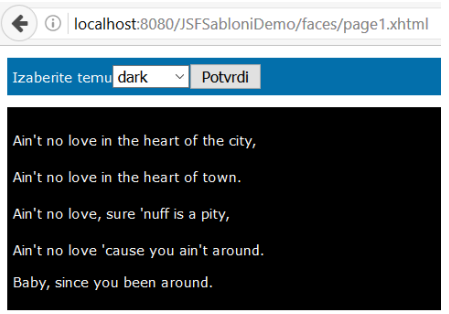
```
                    </h:selectOneMenu>
                    <h:commandButton value="Potvrdi"
                                     action="page1"/>
                </h:form>
            </ui:define>
        <ui:define name="content">

         Ovde ide sadržaj za prikazivanje


            </ui:define>

    </ui:composition>

        </f:view>

</body>
</html>
```

U priloženom kodu je moguće primetiti da je prvo uključena podrška za JSF <h:***> tagove instrukcijom *xmlns:h=*"http://xmlns.jcp.org/jsf/html" taga <html>. Dalje je u top sekciju uključeno: labela sa informacijom za izbor teme, tag <h:selectOneMenu> za listu opcija koje odgovaraju temama i dugme za potvrdu izbora teme.

Kada je prethodna procedura kreiranja koda za upravljanje temama opisana, neophodno je izvršiti i demonstriranje funkcionisanja izmenjene stranicepage1.xhtml. Prvo je neophodno izvršiti ponovno kompajliranje kreirane veb aplikacije. Nakon toga, aktivira se link http://localhost:8080/JSFSabloniDemo/faces/page1.xhtml u veb pregledaču. Sledećim slikama je prikazan početni izgled stranice sa vršenjem izbora i izgled stranice nakon što je tema promenjena u dark.



Slika 7.6 Početni izgled i izbor [izvor autor]



Slika 7.7 Promenjena tema [izvor autor]

Poglavlje 8

JSF - Pokazni primeri

ZADATAK 1 (20 MIN)

Kreira se veb apliakcija koja koristi JSF stranice i šablone.

1. Otvoriti razvojno okruženje i kreirati novi Java veb projekat
2. Kreirati veb aplikaciju koja koristi izabrani layout šablon (CSS) ili temu;
3. Kreirati klijent šablon stranicu za prikazivanje sadržaja raspoređenog na osnovu izabranog rasporeda;

Sledećim listingom je priložena datoteka kojom je definisan šablon:

```
<?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:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

    <body>
<!--
        <ui:composition template="./template.xhtml">
-->
  <!--
        <f:view contracts="dark">
  -->
           <f:view contracts="#{selektorTema.themeName}">
           <ui:composition template="/template.xhtml">
               <ui:define name="top">
                   <h:form>
                       <h:outputLabel for="themeSelector"
                                      value="Izaberite temu"/>
                       <h:selectOneMenu id="selektorTema"
                                        value="#{selektorTema.themeName}">
                           <f:selectItem itemLabel="normal"
                                         itemValue="normal"/>
                           <f:selectItem itemLabel="dark"
                                         itemValue="dark"/>
                       </h:selectOneMenu>
                       <h:commandButton value="Potvrdi"
                                        action="page1"/>
                   </h:form>
```

```
               </ui:define>
            <ui:define name="content">
                <br/>Ain't no love in the heart of the city,</br>
                <br/>Ain't no love in the heart of town.</br>
                <br/>Ain't no love, sure 'nuff is a pity,</br>
                <br/>Ain't no love 'cause you ain't around.</br>
                <p>Baby, since you been around.</p>


                </ui:define>

        </ui:composition>

            </f:view>

    </body>
</html>
```

Upravljanje temama je u domenu CDI zrna i u narednom izlaganju se prilaže njegov kod:

```
package com.metropolitan.teme;

import javax.enterprise.context.RequestScoped;
import javax.inject.Named;

/**
 *
 * @author Vladimir Milicevic
 */
@Named
@RequestScoped
public class SelektorTema {
private String themeName = "normal";
public String getThemeName() {
return themeName;
}
public void setThemeName(String themeName) {
this.themeName = themeName;
}
}
```

U nastavku je neophodno priložiti kod stranice koja će prikazivati sadržaj. Stranica po potrebi može da koristi i šablone i teme, a to je priloženo sledećim kodom.

```
<?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:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:h="http://xmlns.jcp.org/jsf/html">
```

```
    <body>
<!--
        <ui:composition template="./template.xhtml">
-->
  <!--
        <f:view contracts="dark">
  -->
           <f:view contracts="#{selektorTema.themeName}">
           <ui:composition template="/template.xhtml">
               <ui:define name="top">
                   <h:form>
                       <h:outputLabel for="themeSelector"
                                      value="Izaberite temu"/>
                       <h:selectOneMenu id="selektorTema"
                                        value="#{selektorTema.themeName}">
                           <f:selectItem itemLabel="normal"
                                         itemValue="normal"/>
                           <f:selectItem itemLabel="dark"
                                         itemValue="dark"/>
                       </h:selectOneMenu>
                       <h:commandButton value="Potvrdi"
                                        action="page1"/>
                   </h:form>
               </ui:define>
            <ui:define name="content">
                <br/>Ain't no love in the heart of the city,</br>
                <br/>Ain't no love in the heart of town.</br>
                <br/>Ain't no love, sure 'nuff is a pity,</br>
                <br/>Ain't no love 'cause you ain't around.</br>
                <p>Baby, since you been around.</p>


                </ui:define>

        </ui:composition>

            </f:view>

    </body>
</html>
```

Za rad sa temama koriste se opisane biblioteke resursa. Neophodno je modifikovati odgovarajuće cssLayout.css fajlove u zavisnosti od toga kako želite da vam tema izgleda (pogledati u predavanjima).

Urađen i testiran zadatak je priložen u dodatnim aktivnostima materijala.

ZADATAK 2 (25 MIN)

Vežbanje korišćenja formi na JSF stranicama

1. Otvoriti razvojno okruženje i kreirati nov Java veb projekat;
2. Kreirati JSF stranicu na kojoj se nalazi forma za unos korisničkih podataka (titula, ime, prezime, starost, email);
3. Kreirati JSF stranicu na kojoj će biti prikazani uneti podaci;
4. Kreirati CDI klasu zrna za obezbeđivanje setter i getter metoda za upravljanje podacima sa forme;
5. Kreirati jednu validator klasu za proveru proizvoljnog polja sa forme.

Forma je realizovana sledećim programskim kodom početne stranice:

```
<?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://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">

    <h:head>

        <title>Registracija</title>
        <h:outputStylesheet  name="styles.css"/>
</h:head>
    <h:body>
        <h3>Stranica za registraciju</h3>
        <h:form>
            <h:panelGrid columns="3"
                         columnClasses="rightalign,leftalign,leftali
                         gn">
                <h:outputLabel value="Titula: " for="titula"/>
                <h:selectOneMenu id="titula" label="Titula"
                                 value="#{registrationBean.
                                          titula}" >
                    <f:selectItem itemLabel="" itemValue=""/>
                    <f:selectItem itemLabel="St." itemValue="ST"/>
                    <f:selectItem itemLabel="Ma." itemValue="MA"/>
                    <f:selectItem itemLabel="MSci" itemValue="MSCI"/>
                    <f:selectItem itemLabel="Dr" itemValue="DR"/>
                </h:selectOneMenu>
                <h:message for="titula"/>
                <h:outputLabel value="Ime:" for="ime"/>
                <h:inputText id="ime" label="Ime"
                             required="true"
                             value="#{registrationBean.ime}" />
                <h:message for="ime" />
                <h:outputLabel value="Prezime:" for="prezime"/>
                <h:inputText id="prezime" label="Prezime"
                             required="true"
                             value="#{registrationBean.prezime}" />
                <h:message for="prezime" />
                <h:outputLabel for="starost" value="Starost:"/>
                <h:inputText id="starost" label="Starost" size="2"
                             value="#{registrationBean.starost}"/>
```

```
                <h:message for="starost"/>
                <h:outputLabel value="Email adresa:" for="email"/>

                <h:inputText id="email" label="Email adresa"
                             required="true"
                             value="#{registrationBean.email}">
                    <f:validator validatorId="emailValidator"/>
                </h:inputText>
                <h:message for="email" />
                <h:panelGroup/>
                <h:commandButton id="registracija" value="Registracija"
                                 action="potvrda" />
            </h:panelGrid>
        </h:form>
    </h:body>
</html>
```

Stranica za prikazivanje rezultata je realizovana sledećim JSF kodom:

```
<?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://xmlns.jcp.org/jsf/html">
    <h:head>
        <title>Stranica za potvrdu</title>
        <h:outputStylesheet name="styles.css"/>
    </h:head>
    <h:body>
        <h2>Stranica za potvrdu</h2>
        <h:panelGrid columns="2"
                     columnClasses="rightalign-bold,leftalign">
            <h:outputText value="Titula:"/>
            ${registrationBean.titula}
            <h:outputText value="Ime:"/>
            ${registrationBean.ime}
            <h:outputText value="Prezime:"/>
            ${registrationBean.prezime}
            <h:outputText value="Starost:"/>
            ${registrationBean.starost}
            <h:outputText value="Email adresa:"/>
            ${registrationBean.email}
            </h:panelGrid>
        </h:body>
</html>
```

JAVA KLASE

Neophodno je kreirati CDI klasu i klasu validator.

Klasa koja obezbeđuje CDI zrno za podršku podacima forme data je sledećim listingom:

```
package com.metropolitan;

/**
 *
 * @author Vladimir Milicevic
 */
import javax.enterprise.context.RequestScoped;
import javax.inject.Named;

@Named
@RequestScoped
public class RegistrationBean {
private String titula;
private String ime;
private String prezime;
private Integer starost;
private String email;
public String getTitula() {
        return titula;
    }

    public void setTitula(String titula) {
        this.titula = titula;
    }

    public String getIme() {
        return ime;
    }

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

    public String getPrezime() {
        return prezime;
    }

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


    public Integer getStarost() {
        return starost;
    }

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

    public String getEmail() {
        return email;
    }
```

```
    public void setEmail(String email) {
        this.email = email;
    }

}
```

Konačno, klasa validator zaključuje kreiranje ove veb aplikacije.

```
package com.metropolitan;

/**
 *
 * @author Vladimir Milicevic
 */
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.component.html.HtmlInputText;
import javax.faces.context.FacesContext;
import javax.faces.validator.FacesValidator;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;

@FacesValidator(value = "emailValidator")
public class EmailValidator implements Validator {

    @Override
    public void validate(FacesContext facesContext,
            UIComponent uiComponent, Object value) throws
            ValidatorException {
        Pattern pattern = Pattern.compile("\\w+@\\w+\\.\\w+");
        Matcher matcher = pattern.matcher(
                (CharSequence) value);
        HtmlInputText htmlInputText
                = (HtmlInputText) uiComponent;
        String label;
        if (htmlInputText.getLabel() == null
                || htmlInputText.getLabel().trim().equals("")) {
            label = htmlInputText.getId();
        } else {
            label = htmlInputText.getLabel();
        }
        if (!matcher.matches()) {
            FacesMessage facesMessage
                    = new FacesMessage(label
                            + ": email adresa nije ispravna");
            throw new ValidatorException(facesMessage);
        }
    }
}
```

Urađen i testiran zadatak je priložen u dodatnim aktivnostima lekcije.

VIDEO MATERIJAL

Naslov "8 - Facelets" - Trajanje: 21:45

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

Poglavlje 9

Individualna vežba 5

INDIVIDUALNA VEŽBA (135 MIN)

Pokušajte sami

1. Kreirajte vlastitu JSF aplikaciju po uzoru na Zadatak 2
2. Unesite modifikaciju u Zadatak 2,
3. Forma preuzima dodatne podatke o korisniku kao što je naziv fakulteta koji studira, skraćenica fakulteta i studijski program.
4. Obezbediti da ove informacije budu obrađene i prikazane.
5. Angažujte aplikaciju i proverite korektnost vašeg rada.
6. U potpunosti urađenu aplikaciju dokumentujte i prezentujte predmetnom asistentu

Poglavlje 10

Domaći zadatak 5

ZADATAK 1

Unapređenje i proširenje primera urađenog na vežbama.

1. Kao osnovu za domaći zadatak izabrati Zadatak 2 sa vežbi;
2. Umesto kategorije titula, dodati kategoriju "Vrsta studiranja" sa opcijama: tradicionalno i internet;
3. Polja za unos teksta su za podatke o studentu;
4. Dodati još jednu padajuću listu čiji su elementi fakulteti našeg Univerziteta;
5. Obezbediti mehanizme validacije za polja;
6. Realizovati CDI zrno za pomoć prilikom prihvatanja unetih podataka forme;
7. Na stranci za prikazivanje podataka, prikazati sve podatke o studentu, uključujući vrstu studiranja i fakultet koji studira.
8. Za realizovanje aplikacije poželjno je korišćenje nekog css šablona ili teme.

Nakon urađenog obaveznog zadatka, studenti dobijaju posebne zadatke od predmetnog asistenta na mail.

Poglavlje 11

Zaključak

ZAKLJUČAK

Lekcija se bavila kreiranjem i upotrebom JSF stranica u Java veb aplikacijama.

Lekcija se bavila kreiranjem i upotrebom JSF stranica u Java veb aplikacijama. Lekcija je posebno istakla da JavaServer Faces (JSF) predstavlja specifičan Java okvir (framework) namenjen za razvoj širokog spektra Java veb aplikacija. Vodeći se navedenim, u ovoj lekciji je bio cilj pokazivanje načina na koji primena JSF pojednostavljuje razvoj veb aplikacija. U tu svrhu su bile obrađene neke značajne teme za analizu i demonstraciju ovog specifičnog Java okvira za razvoj veb aplikacija:

1. Kreiranje JSF projekta primenom izabranog razvojnog okruženja, konkretno je korišćen NetBeans IDE razvojno okruženje;
2. Postavljanje JSF tagova za korišćenje prednosti koju obezbeđuje primena taga JSF <h:panelGrid>;
3. Korišćenje statičke i dinamičke navigacije za definisanje navigacije između stranica;
4. Razvoj CDI (Contexts and Dependency Injection for the Java EE Platform) zrna za enkapsulaciju podataka i programske logike;
5. Implementacija JSF validatora za proveru valjanosti unosa pojedinih polja forme;
6. Kreiranje JSF šablona i isticanje prednosti njihove primene;
7. Rad sa JSF temama i isticanje prednosti njihove primene.

Posebno značajno u prethodnom izlaganju predstavlja činjenica da je lekcija insistirala na analizi i demonstraciji primenom odabranih primera koji su na najbolji način istakli i pokazali prednosti primene JSF stranica u Java veb aplikacijama. Od posebnog značaja je bila fokusiranost primera na rad sa šablonima stranica i temama za pokazivanje veoma jednostavnog načina kreiranja JSF stranica sa predefinisanim atraktivnim izgledom. Konkretan primer je, takođe, pokazao kako jedna šablon JSF datoteka na veoma jednostavan način upravlja nizom JSF datoteka klijenata šablona.

Izlaganje, započeto u teorijskom delu lekcije, posebno je zaokruženo zadacima koji su kompletirani, testirani i demonstrirani u materijalima za vežbe. Svaki od navedenih zadataka, odgovara kompletno urađenoj i funkcionalnoj veb aplikaciji čiji je kod priložen kao dodatni materijal u Objektu učenja "JSF - vežbe" u okviru kategorije Shared Resources.

Savladavanjem ove lekcije studenti su osposobljeni da koriste JSF tehnologiju za razvoj Java veb aplikacija pored detaljno obrađene JSP tehnologije iz prethodnih lekcija. Studenti su kroz ove dve tehnologije stekli znanja koja im omogućavaju savladavanje naprednijih koncepata

Java EE veb aplikacija čija analiza i demonstracija sledi kroz lekcije koje će biti izučavane odmah nakon ove lekcije.

LITERATURA

U pripremi lekcije korišćena je najnovija literatura.

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