Prof. dr Boško Nikolić

Dražen Drašković

PROGRAMIRANJE INTERNET APLIKACIJA

udžbenik sa zbirkom zadataka

ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

maj 2017. godine

Prof. dr Boško Nikolić, Dražen Drašković

PROGRAMIRANJE INTERNET APLIKACIJA - udžbenik sa zbirkom zadataka

Recenzenti: Doc. dr Zaharije Radivojević, Doc. dr Miloš Cvetanović

Izdavač: Elektrotehnički fakultet Univerziteta u Beogradu

Akademska misao, Beograd

Na osnovu člana 42. Statuta Elektrotehničkog fakulteta i člana 14. Pravilnika o udžbenicima i drugoj nastavnoj literature, kao i pozitivne recenzije dr Zaharija Radivojevića i dr Miloša Cvetanovića, Nastavno-naučno veće Elektrotehničkog fakulteta je na svojoj sednici održanoj dana 22. marta 2016. godine donelo Odluku o odobravanju nastavnog materijala za štampu pod nazivom “Programiranje internet aplikacija - udžbenik sa zbirkom rešenih zadataka” (broj 2154/2).

ISBN: XXXXXXXXXXXXX

Štampano u Srbiji.

Štampa: Akademska misao Beograd

Sadržaj

Predgovor

Ovaj udžbenik sa zbirkom rešenih zadataka predstavlja osnovni materijal za učenje i pripremu gradiva iz predmeta Programiranje internet aplikacija. Predmet Programiranje internet aplikacija drži se na osnovnim akademskim studijama na Elektrotehničkom fakultetu u Beogradu, na Odseku za računarsku tehniku i informatiku i Odseku za softversko inženjerstvo. Sadržaj ove knjige potpuno obuhvata gradivo predviđeno za predavanja i vežbe koje se izvode u okviru tog predmeta. Autori veruju da će ova knjiga biti od koristi i mnogim studentima drugih fakulteta, diplomcima, softverskim inženjerima i svima onima koji žele da nauče ili se podsete osnova veb programiranja.

Udžbenik je podeljen u četiri poglavlja koja obuhvataju Java veb programiranje: programski jezik Java Script, tehnologije Java Servleti, JavaServer Pages i JavaServer Faces. Svako poglavlje obuhvata teorijske osnove sa primerima i zadatke. Cilj udžbenika je da studenti savladaju gradivo, razumeju osnovne koncepte i načine projektovanja veb aplikacija i nauče da primene takve principe u industrijskim projektima.

Autori su svesni da ova knjiga može još da se doradi i da postoje eventualne greške, jer je ovo tek prvo izdanje, pa iz tog razloga molimo sve čitaoce da upute primedbe, predloge i pohvale slanjem poruke autorima knjige putem elektronske pošte draskovic@etf.bg.ac.rs.

Zbog čestih unapređenja i pojave novih tehnologija u oblasti internet aplikacija i u cilju podizanja kvaliteta nastavnih materijala Elektrotehničkog fakulteta u Beogradu, autori ovu knjigu objavljuju u elektronskom obliku i žele studentima da im ova knjiga bude od koristi i u učenju i u poslu. Srećno!

Beograd, Autori

29. maja 2017. godine

1 Programski jezik JavaScript

1.1 Uvod u programski jezik JavaScript

Korišćenje HTML tehnologije je u počecima svoje upotrebe dala zadovoljavajuće rezultate, ali se kod ovakvih Internet strana počeo sve izraženije pojavljivati nedostatak dinamičke obrade unetih podataka od strane korisnika. Zato se sa razvojem Interneta došlo do zaključka da HTML postaje ograničavajući faktor i da je potrebna nova tehnologija za realizaciju dinamičkih delova aplikacije.

Prvi pokušaj je bio pomoću serverskih komponenti, od kojih je jedan od najpopularnijih bila CGI (*Common Gateway Interface*). Ova vrsta tehnologije je u početku dala zadovoljavajuće rezultate, jer su veb programeri dobili mogućnost obrade unetih korisničkih podataka i različite odgovore u zavisnosti od tih podataka. Ipak, problem je predstavljala česta klijent-server komunikacija. Primer takve komunikacije može da bude forma u kojoj korisnik treba da unese datum svog rođenja. Korisnik može da pogreši i da u tekst polje predviđeno za mesec unese dan rođenja, na primer 28. Ovaj podatak se šalje na server, tamo serverska komponenta pre unosa u bazu proverava podatke i utvrđuje da je došlo do greške. Korisniku se šalje obaveštenje da je pogrešio i da mora ponovo uneti vrednost za mesec. Znači, korisnik je pogrešio, čekao je da se podaci prenesu na server, čekao je da se tamo obrade, da se vrati odgovor i konačno dobio je obaveštenje da je pogrešio.

Tako se došlo do zaključka da je potrebna tehnologija koja će doneti mogućnost obrade određenih podataka na klijentskoj strani, pa su decembra 1995. godine, *Netscape* i *Sun* predstavili jezik *JavaScript 1.0*, originalno nazvan *LiveScript*. *Netscape* je omogućio da se kod pisan na tom jeziku mogao izvršavati u okviru *Netscape Navigator 2* veb pregledača. Već u početku je ovaj jezik omogućio ne samo formatiranje podataka na klijentskoj strani, već i obradu i dinamičko izvršavanje stranica.

Sledeći korak u popularnosti JavaScript jezika je bila *Microsoft*-ova implementacija u okviru *Internet Explorer 3* pregledača, pri čemu je ova verzija od strane *Microsoft*-a nazvana *JScript*. *JScript* je bio baziran na javnoj dokumentaciji *Netscape*-a i bio je skoro identičan JavaScript jeziku. Konačno, telo ECMA (*European Computer Manufacturers Association*) je usvojilo standard za ovaj jezik nazvan ECMAScript 1.0, kasnije ISO/IEC 16262, pa je JavaScript postao Nestcape-ova implementacija ovog standarda, a JScript *Microsoft*-ova.

Danas, JavaScript je objektno bazirani, platformski neutralan, višekorisnički jezik koji programeru omogućava mnogo veću funkcionalnost na klijentskoj strani. Primer naveden kao ilustracija nedostataka serverskih tehnologija, sada se rešava pomoću JavaScript-a koda. Onog trenutka kada korisnik unese pogrešnu vrednost za mesec rođenja, JavaScript kod tu grešku

odmah ustanovi i prikaže obaveštenje korisniku. Korisnik je i u ovom slučaju pogrešio, ali je sada korekciju dobio odmah.

Pojam objektno bazirani, znači da nisu svi koncepti objektno orijentisanih jezika realizovani u ovom jeziku, da je veoma limitiran rad sa nasleđivanjem, važenjem i funkcionalnošću samih objekata. Sa druge strane postoje hijerarhija ugrađenih objekata i oni se mogu koristiti, sa već definisanim metodama i osobinama (eng. *property*). Ovakvim pristupom dobijeno je na jednostavnosti samog jezika, a pomoću ugrađenih objekata nije izgubljena potrebna funkcionalnost.

Navedeno je da je *JavaScript* platformski neutralan jezik (kao i HTML), što znači da bi njegov kod, ako je pisan po standardu, trebalo da se izvršava u okviru pregledača klijenta, bez obzira koja je vrsta hardvera u računaru ili koje je softversko okruženje u pitanju. Zato je i veličina programa pisanih u ovom jeziku bila dovoljno mala da je mogla da se izvršava i na računarima sa lošijim perfomansama.

Radi efikasnijeg i jednostavnijeg programiranja i održavanja samog koda *JavaScript* omogućava modularno programiranje, jer je moguće kreirati svoje sopstvene objekte, definisati opšte funkcije koje će realizovati uobičajene zadatke i čuvati i izvršavati kod pomoću posebnih dokumenata (sa .js ekstenzijom). Primer efikasne upotrebe ovakvog načina programiranja je funkcija koja proverava da li je adresa elektronske pošte korisnika u dozvoljenom formatu. Verovatno je da je takva funkcija potrebna više puta tokom jedne aplikacije. Ako se realizuje spoljašnji dokument koji će sadržati ovu funkciju, koja kao argument prihvata uneti tekst, na različitim mestima upotrebe je dovoljno samo pozvati realizovanu funkciju. Ako se želi promeniti format adrese, sve promene se izvršavaju samo na jednom mestu, u *JavaScript* dokumentu (eksternom fajlu).

Još jedna od prednosti *JavaScript* jezika je njegova integrisanost sa HTML-om. U okviru jedne HTML strane je moguće je na proizvoljan način kombinovati *JavaScript* i HTML kod. Takođe iz *JavaScript*-a moguće je generisati sam HTML kod, u zavisnosti od određene akcije korisnika.

1.2 Osnove programskog jezika JavaScript

U ovom odeljku se razmatraju osnove samog *JavaScript* jezika.

Programski kod ovog jezika se može uključuti u okviru HTML stranice na dva načina. Prvi je direktnim pisanjem koda u okviru stranice. Tada se koristi HTML <script> tag. Uobičajeno je da se u početnom tagu definiše atribut *language* sa vrednošću *JavaScript*, iako trenutno ako se i ne navede, veb pregledači podrazumevaju da je korišćeni skript jezik *JavaScript*.

Znači struktura koda u okviru HTML stranice je:

```
<script language="JavaScript">
...JavaScript kod…
</script>
```

Drugi način korišćenja *JavaScript* koda u okviru HTML stranice je poziv js dokumenta. Opet se koristi <script> tag, samo što se u okviru taga definiše i spoljašnji dokument u okviru atributa *src*. Struktura ove vrste koda je:

```
< script language ="JavaScript" src=" JSkod.js">
</script>
```

gde je JSkod.js dokument koji sadrži željene *JavaScript* funkcije.

Za razliku od mnogih viših jezika (kao što su C, C++, Java) u *JavaScript* jeziku nije potrebno na kraju svake naredbe pisati simbol ";" (tačka-zarez), mada neće dolaziti do greške ako se tačka-zarez upotrebljava. Jedini izuzetak, kada se obavezno mora koristiti tačka-zarez je ako se navodi više naredbi u istom redu. Tada se svaka pojedinačna naredba mora odvojiti sa tačkom-zarez.

Za komentar jedne linije koda se koristi oznaka „//" , na primer:

```
// komentar u jednoj liniji ...
```

Za komentarisanje više redova koriste se oznake „/*" za početak bloka pod komentarom i oznake „*/" za kraj bloka pod komentarom:

```
/*
komentar u više redova...
*/
```

HTML tekst se prikazuje pomoću *JavaScript* koda na stranici korišćenjem metoda document.write("neki tekst koji se prikazuje na stranici"). Argument ovog metoda je string koji može biti proizvoljan HTML kod. Na primer:

```
< script language="JavaScript">
document.write("<b>Prvi red</b><br /><I>Drugi red</I>")
</script>
```

Dobija se sledeći tekst u okviru stranice:

Prvi red

Drugi red

Imena promenljivih mogu da sadrže brojeve i slova engleske abecede, ali prvi znak mora da bude slovo engleske abecede ili simbol "_" . *JavaScript* je *case sensitive* jezik, što znači da se velika i mala slova razlikuju, pa je promenljiva *Aaa* različita promenljiva od promenljive *AAA*. Takođe se ključne reči (for, if, else, class, int...) ne mogu koristiti u imenu promenljivih.

Tipovi podataka koji su podržani su celobrojni brojevi, racionalni brojevi, stringovi i logički tip.

Celobrojni brojevi se mogu koristiti sa brojnom osnovom 10, sa osnovom 8 i osnovom 16. Uobičajena je predstava pomoću osnove 10. Ovakvi brojevi imaju cifre od 0 do 9, s tim da početna cifra ne sme biti 0. Brojevi prikazani u oktalnom brojnom sistemu sa osnovom 8 moraju počinjati sa cifrom 0, a ostale cifre su od 0 do 7. Brojevi prikazani u heksadecimalnom brojnom sistemu sa osnovom 16 moraju počinjati sa 0x ili 0X, a ostale cifre su od 0 do 15, s tim da se cifre 10-15 prikazuju slovima A - F. Znači ako je broj prikazan sa 0716, u decimalnom brojnom sistemu to je broj $7*8^2 + 1*8 + 6 = 462$, a broj prikazan sa 0x716 u decimalnom brojnom sistemu je $7*16^2 + 1*16 + 6 = 1814$.

Racionalni brojevi se mogu prikazati na dva načina. Jedan je pomoću decimalne tačke, na primer 3.14, a drugi je pomoću eksponencijalne prezentacije, na primer 314E-2 ili 314e-2.

String predstavlja proizvoljan niz karaktera između navodnika (“neki tekst”) ili apostrofa (‘neki tekst’). U stringovima se mogu koristiti i specijalni karakteri kao što su:

****b = jedno mesto levo (*backspace*)

****f = jedan red nadole (*form feed*)

****n = početak novog reda (*new line character*)

****r = return (*carriage return*)

****t = tabulator (*tab*)

Logički tip podataka obuhvata dve vrednosti *true* (tačno) i *false* (netačno). Prilikom rada ako je potrebno može se izvršiti konverzija logičke vrednosti *true* u broj 1, odnosno vrednosti *false* u broj 0.

Tokom rada sam *JavaScript* jezik automatski izvršava promenu jednog tipa u drugi, jer se dozvoljava da promenljiva ima različite tipove podataka u različito vreme izvršavanja programa. Na primer, neka je dat sledeći deo kod:

```
a = 5;
b = 8;
b = “broj ” + a;
```

Tokom izvršavanja ovog programskog koda promenljiva *a* predstavlja broj 5, promenljiva *b* predstavlja broj 8, a u poslednjoj naredbi potrebno je sabrati vrednost tipa String i vrednost tipa ceo broj, što je nemoguće izvršiti bez konverzije obe vrednosti u isti tip. Kako tip String ima prednost broj 5 se konvertuje u tip String, odnosno “5”, i izvršava se sabiranje (tzv. operacija konkatenacije) dva Stringa, tako da se na kraju u promenljivoj *b* nalazi String “broj 5”.

1.3 Operatori u JavaScript-u

U *JavaScript*-u postoje sledeće grupe operatora: aritmetički operatori, operatori na nivou bita, logički operatori i operatori poređenja. U daljem tekstu će se detaljnije razmotriti svaka od navedenih grupa.

Aritmetički operatori

Ovi operatori se koriste za matematičke operacije. Ukoliko je jedan od operanada tipa String za sve operatore, osim za sabiranje, pokušaće se da se izvede konverzija Stringa u broj i da se tako izvrši definisana operacija. Ako se ne uspe kao rezultat se dobija specijalna vrednost NaN (*Not A Number*). Kod sabiranja podatak koji nije tipa String konvertuje se u String i izvršava se sabiranje dva Stringa. Pregled svih aritmetičkih operatora je dat u sledećoj tabeli.

Operator	Naziv operatora	Operator	Naziv operatora
+	sabiranje	+=	sabiranje dodela
-	oduzimanje	-=	oduzimanje dodela
*	množenje	*=	množenje dodela
/	deljenje	/=	deljenje dodela
%	moduo	%=	moduo dodela
++	inkrement	--	dekrement

Sabiranje, oduzimanje, množenje i deljenje obavljaju naznačene operacije nad numeričkim operandima. Unarni minus negira operand kojem prethodi. Operator moduo kao rezultat vraća ostatak pri deljenju.

Operatori na nivou bita

Operatori iz ove grupe obavljaju operacije nad celobrojnim brojevima, i to dužine 32 bita. Ukoliko neki od operanada nije celobrojni broj dužine 32 bita, pokušaće se izvršiti konverzija u traženi tip, pa tek onda primeniti operacija. Ovi operatori celobrojne brojeve razmatraju na nivou bita i obavljaju operacije nad bitovima.

Operator	Upotreba	Opis
Logičko I (AND)	a & b	Rezultat se dobija 1, jedino ako su oba bita 1, u ostalim slučajevima rezultat je 0.
Logičko ILI (OR)	a | b	Rezultat se dobija 0, jedino ako su oba bita 0, u ostalim slučajevima rezultat je 1.
Logičko eksluzivno ILI (XOR)	a ^ b	Rezultat se dobija 1, ako biti imaju različite vrednosti, u slučaju da imaju iste vrednosti, rezultat je 0.
Logičko NE (NOT)	~ a	Komplementira bitove operanda a.
Pomeranje ulevo	a << b	Pomera binarni sadržaj operanda a za b mesta ulevo. Prazna mesta popunjava sa vrednošću 0.
Pomeranje udesno sa znakom	a >> b	Pomera binarni sadržaj operanda a za b mesta udesno. Prazna mesta popunjavju sa vrednošću najstarijeg bita.
Pomeranje udesno sa nulama	a >>> b	Pomera binarni sadržaj operanda a za b mesta udesno. Prazna mesta popunjavju sa vrednošću 0.

Logički operatori

U okviru izraza koji se koriste sa ovom vrstom operatora, operandi moraju biti logičkog tipa.

Operator	Upotreba	Opis
I (&&)	expr1 && expr2	Logičko I
ILI (||)	expr1 || expr2	Logičko ILI
NE (!)	!expr	Negacija

Operatori poređenja

Operatori iz ove grupe obavljaju poređenje dve vrednosti i kao rezultat vraćaju vrednost logičkog tipa. Svaki dozvoljeni tip podataka, celobrojan, racionalni, karakter, String i logički tip mogu se upoređivati koristeći operatore == i !=. Samo numerički tipovi koriste ostale operatore.

Operator	Upotreba

Jednakost (==)	Rezultat je true, ako su operandi jednaki - vrši se konverzija podataka
Nejednakost (!=)	Rezultat je true, ako su operandi različiti - vrši se konverzija podataka
Veće (>)	Rezultat je true, ako je levi operand veći od desnog operanda
Veće ili jednako (>=)	Rezultat je true, ako je levi operand veći ili jednak desnom operandu
Manje (<)	Rezultat je true, ako je levi operand manji od desnog operanda
Manje ili jednako (<=)	Rezultat je true, ako je levi operand manji ili jednak desnom operandu
Jednako bez konverzije tipova (===)	Rezultat je true, ako su operandi jednaki bez konverzije podataka
Različito bez konverzije tipova (!==)	Rezultat je true, ako su operandi različiti bez konverzije podataka

Operatori == i != obavljaju potrebnu konverziju podataka pre poređenja, ukoliko su operandi različitog tipa. Znači za ove operatore vrednosti 5 i "5" su iste, pa će posle njihovog poređenja rezultat sa operatorom == biti true, a sa operatorom != false.

S druge strane operatori === i !== ne obavljaju potrebnu konverziju podataka pre poređenja, ukoliko su operandi različitog tipa. Znači za ove operatore vrednosti 5 i "5" su različite, pa će posle njihovog poređenja rezultat sa operatorom === biti false, a sa operatorom !== true.

Ternarni if-then-else operator

Generalna forma ovog operatora je:

expression ? statement1 : statement2

gde je izraz *expression* bilo koji izraz čiji rezultat je vrednost logičkog tipa. Ako je rezultat izraza true, onda se izvršava *statement1*, u suprotnom *statement2*.

1.4 Kontrole toka

Kontrola toka omogućava tok programa željenom putanjom u zavisnosti od određenih uslova. Osnovne izjave kontrole toka koje se koriste u *JavaScript*-u su veoma slične osnovnim kontrolama toka u višim programskim jezicima.

if-else

if-else konstrukcija omogućava izvršenje odredjenog bloka instrukcija ako je uslov konstrukcije ispunjen. Opšti oblik konstrukcije je:

```
if (boolean_izraz) blok1;
[else blok2;]
```

gde je else klauzula opciona, a boolean_izraz može biti bilo koji izraz čiji rezultat je vrednost logičkog tipa. blok1 i blok2 su proizvoljni blokovi instrukcija, koji mogu da budu i samo jedna instrukcija. Takođe, svaki od blokova, bilo u if ili u else delu može biti nova if-else konstrukcija.

switch

Opšti oblik ove naredbe je:

```
switch (izraz) {
  case vr1: blok1; [break];
  ...
  case vrN: blokN; [break];
  [default: blok_def]
}
```

Svi navedeni blokovi instrukcija, kao i break klauzule, su opcioni.

Na početku switch-a se izračunava vrednost izraza izraz i dobijena vrednost se poredi sa vrednostima vr_1, vr_2 ... vr_N. Tamo gde dođe do uparivanja izvršava se blok instrukcija u odgovarajućem case delu naredbe. Naredba break dovodi do skoka na kraj tekućeg bloka što u ovom slučaju znači na kraj switch konstrukcije. Ukoliko se vrednost izraza izraz ne nalazi medju vrednostima vr_1,..., vr_N tada se izvršava blok naredbi blok_def;

Treba napomenuti da ukoliko se izostavi neka od break klauzula dolazi do propagiranja kontrole u sledeći case.

while petlja

while petlja funkcioniše na taj način što se blok instrukcija unutar nje ponovljeno izvršava sve dok je uslov za ostanak u petlji, koji se nalazi na ulasku u petlju, ispunjen. Opšti oblik petlje izgleda ovako:

```
[inicijalizacija;]
while(uslov_ostanka){
      telo_petlje;
  }
```

do-while petlja

Za razliku od prethodne petlje koja je imala uslov na svom početku, do-while petlja ima uslov na kraju. Prema tome, telo petlje će se sigurno izvršiti bar jednom. Opšti oblik petlje izgleda ovako:

```
[inicijalizacija]
do {
      telo_petlje
     [iteracija]
} while (uslov);
```

for petlja

Opšti oblik for petlje izgleda ovako:

```
for( inicijalizacija; uslov; iteracija){
    telo_petlje;
}
```

Unutar zaglavlja petlje se nalazi kod za inicijalizaciju, uslov terminacije petlje i kod za iteraciju. Ukoliko uslov na početku nije ispunjen telo petlje se neće izvršiti nijednom.

break

Naredba break se koristi za skok na kraj bloka koji je označen labelom uz break ili na kraj bloka u kome se i break nalazi ako break stoji bez labele. Labele, pomoću kojih se označavaju blokovi, formiraju se kao i svi ostali identifikatori, s tim što iza njih mora stajati dvotačka (:).

return

Naredba return se koristi za povratak iz funkcije na mesto poziva. Ukoliko funkcija vraća neku vrednost tada return mora slediti izraz čiji je tip kompatibilan sa povratnim tipom funkcije (na primer: ako je povratni tip funkcije ceo broj, nakon return mora biti celobrojna promenljiva). U suprotnom return naredba može stajati sama.

continue

continue klauzula se koristi kada je potrebno preći na sledeću iteraciju petlje, a da se deo koda pre njenog kraja ne izvrši.

Specijalne naredbe

Do sada navedene kontrole toka i petlje se mogu pronaći kod većine viših programskih jezika. U okviru *JavaScript* jezika postoje i neke naredbe koje su specifične za ovaj programski jezik:

for...in:

Izvršava iteraciju po specifičnoj promenljivoj za svaku osobinu (*property*) u okviru određenog objekta.

function:

Deklariše *JavaScript* funkciju sa specificiranim parameterrima. Tipovi podataka mogućih parametara obuhvataju stringove, brojevi i objekte.

var:

Deklariše promenljivu, opciono moguće je izvršiti i njenu inicijalizaciju.

with:

Definiše tip objekta za niz izraza. U okviru izraza dodeljuje specifične vrednosti za određene osobine objekta.

1.5 Objekti Date i String

U okviru *JavaScript* jezika postoje ugrađeni određeni objekti koji umnogome olakšavaju pojedine akcije. U okviru ovog odeljka razmatraće se takvi objekti, njihova upotreba, njihovi metodi i korišćenje metoda.

Date objekat

Ovaj objekat se koristi kada je potrebno primeniti određene operacije u kojima se koriste vremenske promenljive. Svaki datum koji se pojavi u okviru nekog *JavaScript* programa se pamti kao broj koji predstavlja broj milisekundi između dobijenog datuma i ponoći 1. januara 1970. godine po UTC vremenu. Na primer argument 5000 će kreirati datum koji predstavlja 5 sekundi posle ponoći 1.1.1970.

U programu kreiranje promenljive od ovog objekta se postiže na jedan od sledećih načina:

```
dateObjectIme = new Date()
dateObjectIme = new Date("month day, year hours:minutes:seconds")
dateObjectIme = new Date(year, month, day)
dateObjectIme = new Date(year, month, day, hours, minutes, seconds)
```

Ako se koristi bez argumenata konstruktor Date() kreira objekat koji predstavlja trenutno vreme i datum. Ako se prosleđuje jedan numerički argument, on se prihvata kao datum u milisekundama po ranije navedenoj konvenciji. Ako je string argument on predstavlja datum u formatu koji je prihvatljiv za računar na kome se izvršava aplikacija (tačan format se može dobiti izvršavanjem metoda Date.parse( )). U okviru konstruktora se može definisati između 2 i 7 numeričkih argumenata, koji onda specificiraju pojedinačna polja datuma i vremena. U tom slučaju, svi argumenti, osim prva dva koji predstavljaju godinu i mesec, su opcioni. Treba naglasiti da se vreme definiše na osnovu lokalne mašine, a ne UTC ili GMT vremena.

Konstruktor Date( ) se može pozivati i u obliku funkcije, bez navođenja operatora new. U tom slučaju se ignorišu svi eventualni argumenti i kao rezultat se vraća string prezentacija trenutnog vremena i datuma.

Primeri korišćenja navedenih konstruktora:

```
today = new Date()
birthday = new Date("December 17, 1995 03:24:00")
birthday = new Date(95,12,17)
birthday = new Date(95,12,17,3,24,0)
```

U okviru ovog predefinisanog objekta postoje ugrađene metode koje se mogu koristiti. U daljem delu teksta će se razmatrati najvažnije metode, njihova primena i primeri.

Date.parse(datum)

Ovaj metod vraća broj milisekundi do navedenog datuma po lokalnom vremenu (od 1.1.1970. 00:00:00). Primer:

```
datum.setTime(Date.parse("Aug 9, 2015")
```

Date.UTC(gg,mm,dd [,hh][,mh][,sec])

Ovaj metod vraća broj milisekundi od 1.1.1970. 00:00:00 do datuma, prema Universal Coordinate Time (GMT). Primer:

```
gmtDatum = new Date(Date.UTC(96, 11, 1, 0, 0, 0))
```

datum.getDate()

Ovaj metod vraća dan u mesecu (1-31) za navedeni datum. Primer:

```
datum = new Date("December 19, 2011 23:15:00");
dan = datum.getDate()
```

Nakon izvršavanja primera promenljiva dan dobija vrednost 19.

datum.getDay()

Ovaj metod vraća dan u nedelji (0-nedelja 1-ponedeljak, 2-utorak, ... 6-subota) za navedeni datum. Primer:

```
datum = new Date("December 19, 2015 23:15:00");
dan = datum.getDay()
```

Nakon izvršavanja primera promenljiva dan dobija vrednost 6, jer 19.12.2015. godine pada u subotu.

datum.getHours()

Ovaj metod vraća sat za navedeni datum, moguće vrednosti su brojevi u opsegu od 0 do 23. Primer:

```
datum = new Date("December 25, 2015 23:15:00");
sati = datum.getHours()
```

Nakon izvršavanja primera promenljiva sati dobija vrednost 23.

datum.getMinutes()

Ovaj metod vraća minute za navedeni datum, moguće vrednosti su brojevi u opsegu od 0 do 59. Primer:

```
datum = new Date("December 25, 2015 20:15:00");
minuti = datum.getMinutes()
```

Nakon izvršavanja primera promenljiva minuti dobija vrednost 15.

datum.getMonth()

Ovaj metod vraća mesec za navedeni datum (0-januar, 1-februar, 2-mart, 3-april, 4-maj, ... 10-novembar, 11-decembar). Primer:

```
datum = new Date("December 25, 2015 20:15:00");
mesec = datum.getMonth()
```

Nakon izvršavanja primera promenljiva mesec dobija vrednost 11.

datum.getSeconds()

Ovaj metod vraća sekunde za navedeni datum, moguće vrednosti su brojevi u opsegu od 0 do 59. Primer:

```
datum = new Date("December 25, 2001 23:15:08");
sekunde = datum.getSeconds()
```

Nakon izvršavanja primera promenljiva sekunde dobija vrednost 8.

datum.getTime()

Ovaj metod vraća vreme do navedenog datuma u milisekundama (od 1.1.1970. 00:00:00). Primer:

```
datum = new Date("December 25, 2001 23:15:00");
proteklo = datum.getTime()
```

Nakon izvršavanja primera promenljiva proteklo dobija vrednost 1009318500000, koja odgovara broju milisekundi od 1.1.1970 00:00:00 do 25.12.2001. 23:15:00.

datum.getTimezoneOffset()

Ovaj metod vraća razliku lokalnog vremena i GMT u minutama. Primer:

```
datum = new Date();
razlikaSati = datum.getTimezoneOffset()/60
```

Nakon izvršavanja primera promenljiva razlikaSati dobija vrednost -1.

datum.getFullYear()

Ovaj metod vraća godinu iz navedenog datuma. Primer:

```
datum = new Date();
godina = datum.getFullYear()
```

Nakon izvršavanja primera promenljiva godina dobija vrednost tekuće godine (u našem slučaju izvršavanja ovog koda, to je 2015. godina).

datum.setDate(brojDana)

Ovaj metod postavlja dan u mesecu za navedeni datum. Argument metoda je broj u opsegu od 1 do 31. Primer:

```
datum = new Date("July 27, 1960 23:30:00");
datum.setDate(24)
```

Nakon izvršavanja primera promenljiva datum dobija vrednost 24.7.1960 23:30:00.

datum.setHours(brojSata)

Ovaj metod postavlja broj sati za navedeni datum. Argument metoda je broj u opsegu od 0 do 23. Primer

```
datum = new Date("July 27, 1960 23:30:00");
datum.setHours(7)
```

Nakon izvršavanja primera promenljiva datum dobija vrednost 27.7.1960 07:30:00.

datum.setMinutes(brojMinuta)

Ovaj metod postavlja broj minuta za navedeni datum. Argument metoda je broj u opsegu od 0 do 59. Primer:

```
datum = new Date("July 27, 1960 23:30:00");
datum.setMinutes(35)
```

Nakon izvršavanja primera promenljiva datum dobija vrednost 27.7.1960 23:35:00.

datum.setMonth(brojMeseca)

Ovaj metod postavlja broj meseci za navedeni datum. Argument metoda je broj u opsegu od 0 do 11. Primer:

```
datum = new Date("July 27, 1960 23:30:00");
datum.setMonth(11)
```

Nakon izvršavanja primera promenljiva datum dobija vrednost 27.12.1960 23:30:00.

datum.setSeconds(brojSekundi)

Ovaj metod postavlja dan u mesecu za navedeni datum. Argument metoda je broj u opsegu od 0 do 59. Primer:

```
datum = new Date("July 27, 1960 23:30:00");
datum.setSeconds(35)
```

Nakon izvršavanja primera promenljiva datum dobija vrednost 27.7.1960 23:30:35.

datum.setTime(vreme)

Ovaj metod definiše novi datum. Argument metoda je broj milisecundi od 1.1.1970 00:00:00 do željenog datuma.

```
datum.setTime(1009318500000)
```

Nakon izvršavanja primera promenljiva datum dobija vrednost 25.12.2001. 23:15:00.

datum.setYear(brojGodine)

Ovaj metod postavlja godinu za navedeni datum. Argument metoda je broj u opsegu od 0 do 99 za godine koje počinju sa 19, za ostale je 4 cifre.

```
datum = new Date("July 27, 1960 23:30:00");
datum.setYear(2010)
```

Nakon izvršavanja primera promenljiva datum dobija vrednost 27.7.2010 23:30:00.

datum.toGMTString()

Ovaj metod vrši konverziju datuma u GMT string iz lokalne vremenske zone. Primer:

```
datum = new Date("December 25, 2001 23:15:00");
datum.toGMTString()
```

Nakon izvršavanja primera promenljiva datum dobija vrednost ”Tue, 25 Dec 2001 22:15:00 UTC”

datum.toLocaleString()

Ovaj metod vrši konverziju datuma u lokalni datum string iz GMT. Primer:

```
datum.toLocaleString()
```

String objekat

Ovaj objekat se koristi da bi se efikasnije obradio niz karaktera. I u okviru ovog objekta postoje dostupni metodi koji se mogu koristiti. U daljem tekstu sledi njihov opis, sintaksa i primeri upotrebe.

parseInt(stringBroj [,*osnova*])

Ova funkcija kao rezultat vraća ceo broj dobijen konverzijom argumenta stringBroj koji je tipa String u brojnom sistemu sa osnovom koju definiše argument osnova (8, 10 ili 16). Ovaj argument je opcioni i ako se ne navede podrazumeva se osnova 10. Primer:

```
x = parseInt("17", 8);
y = parseInt("15", 10);
// isti rezultat za y daje i naredba: y = parseInt("15")
```

Nakon izvršavanja primera i promenljiva x i promenljiva y dobijaju vrednost 15.

string.charAt(broj)

Ovaj metod kao rezultat vraća znak na navedenoj poziciji. Pozicije unutar stringa se računaju sa leve na desnu stranu i prva pozicija ima indeks 0. U okviru svakog objekta tipa String postoji i osobina (*property*) length koja je jednaka broju karaktera u posmatranom stringu. Korišćenjem ovog podatka može se odrediti i indeks poslednjeg karakera u stringu, a to je vrednost string.length - 1. Primer:

```
x= "Doba" + "r" + " dan!".charAt(4)
y= "Dobar " + "d" + "an!".charAt(6)
```

Nakon izvršavanja primera promenljiva x dobija vrednost ’r’, a promenljiva y je ’d’.

string.indexOf(traziString, [odPozicije])

Ovaj metod vraća broj pozicije na kojoj je prvi put pronađen argument tipa String definisan kao arguement traziString. U slučaju da se traženi string ne nalazi u početnom stringu kao rezultat se vraća vrednost -1. Ako postoji i drugi argument odPozicije, tada će se pretraga izvršavati od zadate pozicije. Primer:

```
x ="Dobar dan!".indexOf("r")
y ="Dobar dan!".indexOf("a",4)
```

Nakon izvršavanja primera promenljiva x dobija vrednost 4, a promenljiva y je 7.

string.lastIndexOf(traziString, [doPozicije])

Ovaj metod vraća broj pozicije na kojoj se poslednji put pojavljuje argument tipa String traziString. U slučaju da se traženi string ne nalazi u početnom stringu kao rezultat se vraća

vrednost -1. Ako postoji i drugi argument doPozicije, tada će se pretraga izvršavati do zadate pozicije. Primer:

```
x = "Dobar dan!".lastIndexOf("a")
y = "Dobar dan!".lastIndexOf("a", 6)
```

Nakon izvršavanja primera promenljiva x dobija vrednost 7, jer je to poslednje pojavljivanje stringa "a", a promenljiva y je 3, jer je to poslednje pojavljivanje stringa "a" do pozicije 6.

string.substring(prvi, poslednji)

Ovaj metod vraća deo stringa počev od pozicije prvi do pozicije poslednji, tj. uzima redom karaktere na pozicijama prvi, prvi + 1, prvi + 2, ..., poslednji - 2, poslednji - 1.

```
x = "Dobar dan!".substring(6,9)
```

Nakon izvršavanja primera promenljiva x dobija vrednost "dan", jer su to karakteri na pozicijama 6, 7 i 8.

string.toLowerCase()

Ovaj metod izvrši konverziju svih karaktera u okviru stringa u mala slova. Primer:

```
x = "Dobar dan!".toLowerCase()
```

Nakon izvršavanja primera promenljiva x dobija vrednost "dobar dan!", jer je izvršena konverzija svih karaktera u mala slova.

string.toUpperCase()

Ovaj metod izvrši konverziju svih karaktera u okviru stringa u velika slova. Primer:

```
x = "Dobar dan!".toUpperCase()
```

Nakon izvršavanja primera promenljiva x dobija vrednost "DOBAR DAN!", jer je izvršena konverzija svih karaktera u velika slova.

1.6 JavaScript i forme

Programski jezik *JavaScript* je svoju popularnost stekao mogućnošću da pristupa elementima forme, čita njihove vrednosti, obrađuje ih i definiše nove vrednosti elemenata. Takođe iskorišćena je i osobina čitača da prepozna korisnikovu akciju i reaguje na nju. U ovom poglavlju će se razmatrati ove osobine *JavaScript*-a, njihova primena i primeri upotrebe.

Čitač može da prepozna svaku akciju korisnika, bilo da ona potiče od miša ili tastature. Svaki HTML objekat, kao što je dugme, tekst polje, polje za potvrdu, slika, itd. ima listu

događaja koji su povezani sa datim objektom. Na primer tekst polje prepoznaje kada korisnik promeni tekst, koji se trenutno nalazi u okviru polja, slika prepozna kada se kursor miša nalazi na njenoj površini, dugme prepozna kada korisnik mišem klikne na njega, itd. U sledećoj tabeli su dati najvažniji događaji koji se mogu desiti, na koje objekte mogu da se primene i kako se pozivaju u okviru taga koji definiše određeni objekat.

Događaj	Nastaje kada korisnik...	Programski kod
blur	izađe iz fokusa elementa forme	onBlur
click	klikne na element forme ili link	onClick
change	promeni vrednost izabranog elementa forme	onChange
focus	uđe u fokus nekog elementa forme	onFocus
load	učita stranicu u browser	onLoad
mouseover	pređe pokazivačem miša preko linka i sl.	onMouseOver
mouseout	izađe kurzorom miša sa određene površine ili linka	onMouseOut
select	izabere polje elementa forme	onSelect
submit	izvrši slanje forme	onSubmit
unload	napusti stranicu	onUnload
reset	resetuje sadržaj forme	onReset
error	dobije grešku prilikom učitavanja slike ili stranice	onError
abort	prekine učitavanja slike ili stranice	onAbort

U okviru taga elementa forme se može definisati šta da se izvršava kada se desi određeni događaj tj. koja *JavaScript* funkcija da se pozove. Na primer, ako se želi da se pozove funkcija "proveri()" kada korisnik klikne na dugme sa natpisom "Poslati" tada je HTML kod sledeći

```
<INPUT TYPE="button" VALUE="Poslati" NAME="dugme" onClick="proveri()">
```

Funkcija proveri() mora postojati u okviru *JavaScript* koda na datoj stranici. U okviru sledećeg primera prepoznaje se prelazak miša preko slike i korisniku se prikazuje druga slika.

Takođe, *JavaScript* može i da pročita vrednost proizvoljnog elementa forma. Vrednosti elementa forme se prilazi u opštem slučaju na sledeći način:

```
document.imeForme.imeElementa.value
```

gde je document službena reč, imeForme ime forme u okviru koje se nalazi element čijoj se vrednosti pristupa, imeElementa ime elementa i value je službena reč koja dohvata vrednost tog elementa.

Ako je u okviru HTML stranice definisana sledeća forma sa jednim tekst poljem:

```
<form name="PrimerForme">
<input type="text" name="PrimerTekstPolja">
</form>
```

tada se definisanom tekst polju u okviru *JavaScript* koda prilazi na sledeći način:

```
document.PrimerForme.PrimerTekstPolja.value
```

Treba naglasiti još jednom da je *JavaScript* case-sensitive jezik, pa kako je definisano ime u okviru elementa forme, tako se mora koristiti i u okviru *JavaScript* koda. U prethodnom primeru greška bi bila ako bi se ime tekst polja navelo kao PRIMERTEKSTPOLJA ili Primertekstpolja, jer *JavaScript* ne bi prepoznao da se radi o definisanom elementu sa nazivom PrimerTekstPolja.

Ako se želi pročitati vrednost navedenog tekst polja u promenljivu x, to se rešava sa:

```
x =  document.PrimerForme.PrimerTekstPolja.value
```

i analogno, ako se želi u tekst polje upisati vrednost promenljive x:

```
document.PrimerForme.PrimerTekstPolja.value =  x
```

1.7 Rad sa više prozora

JavaScript omogućava da se iz jednog prozora formira, kontroliše ili menja sadržaj u okviru drugog prozora. U ovom poglavlju se razmatraju mogućnosti takvog rada. Prvo se razmatraju specijalne vrste prozora, kao što su upozorenja (alerti), a kasnije se razmatraju mogućnosti rada sa uobičajenim prozorima.

Upozorenja (alerti)

Alerti se koriste unutar HTML strane kada se želi prikazati određeno obaveštenje. To su ugrađeni predefinisani prozori, koji sadrže tekst koji se prikazuje korisniku i dugme OK. U primeru alert se pojavljuje kada se prepozna da je korisnik pritisnuo dugme i poziva se funkcija za prikazivanje alerta.

```
<form action="">
   <input type="button" value="Pritisni me" onClick="alert()" />
</form>

<script type="text/javascript">
   <!--//
         function alert() {
               alert ("Prvi red "+ "i ove je prvi red - \nDrugi red!");
         }
   //-->
</script>
```

Rad sa prozorima

JavaScript omogućava da se iz jednog prozora otvara drugi prozor i da se uspostavlja međusobna komunikacija između njih.

Novi prozor se otvara pomoću naredbe:

```
deteProz = open("noviProzor.html", "deteProz")
```

Ovom naredbom se otvara novi prozor, u okviru koga se prikazuje stranica *noviProzor.html.* Novi prozor je definisan u okviru objekta *deteProz.* Nekom objektu na novom prozoru se može pristupiti pomoću naredbe:

```
deteProz.deteForma.deteObjekat.value
```

gde je *deteForma* - forma definisana u okviru novog prozora, a *deteObjekat* - objekat definisan u okviru forme *deteForma* na novom prozoru.

Pored pristupa objektima na novom prozoru iz starog, može se uspostaviti komunikacija i u drugom smeru, tj. može se objektima na starom prozoru pristupiti iz novog prozora. Primer je:

```
window.opener.document.otacForma.otacObjekat.value
```

gde je *otacForma* - forma definisana u okviru starog prozora, a *otacObjekat* - objekat definisan u okviru forme *otacForma* na starom prozoru.

1.8 Cookies podaci

Cookie su tekstualni fajlovi koji se mogu zapamtiti na računaru klijenta. Internet aplikacije, zbog poboljšane bezbednosti, imaju niz ograničenja, kao što je zabrana da se čita, piše ili brišu datoteke na sistemu na kome se aplikacija izvršava, tj. zabranjen je svaki pristup lokalnom fajl sistemu. Jedini izuzetak od ovog pravila predstavljaju upravo *cookie* fajlovi. Iz razloga bezbednosti, ovi fajlovi moraju da imaju tačno definisan format.

Format koji *cookie* fajl mora da zadovolji je:

```
ime=vrednost[;EXPIRES=datum][;DOMAIN=imeDomena][;PATH=putanja] [;SECURE]
```

gde je:

ime - ime koje definiše upisani cookie. Pomoću njega vrši se čitanje i upis vrednosti u ovaj fajl.

vrednost - je upravo informacija koja se želi zapamtiti na klijentskoj mašini. Ovo polje se čita i definiše u okviru cookie.

datum - je datum koji definiše do kada cookie ostaje upisan na klijentskoj mašini.

imeDomena - definiše jedini domena sa kog cookie može da se čita i da mu se menja vrednost.

putanja - definiše jedinu putanju sa koje cookie može da se čita i da mu se menja vrednost.

SECURE je službena reč koja definiše da se upis i čitanje kukija izvršava preko posebnih, bezbednijih linija.

Opcije EXPIRES, DOMAIN, PATH, SECURE su opcione i nije bitan redosled u kom se pojavljuju. Tekst koji definiše cookie ne sme sadržavati razmake.

Objektu koji predstavlja cookie fajl pristupa se kao objektu u okviru document objekta.

Primer čitanja cookie je:

```
var citamCookie=document.cookie
```

U prethodnom primeru su u promenljivoj citamCookie upisani svi *cookies* koji postoje na datom klijentskom računaru. Sada su ove informacije zapamćene u obliku tekst promenljive. Da bi se pročitao baš određeni *cookie*, ovaj tekst se mora parsirati i pronaći deo koji počinje sa imenom traženog *cookie*-ja.

S druge strane, *cookie* sa određenom vrednošću se upisuje, tako što se formira string u formatu koji je naveden na početku poglavlja. Tako da je primer *cookie* koji se zove primerCookie, čija vrednost je upisana u promenljivu vrednostKojuPamtim i koji se šalje klijentu preko bezbednijih linija bio:

```
document.cookie = "primerCookie="+vrednostKojuPamtim+";secure"
```

Nakon izvršavanja ove naredbe na klijentskom računaru je upisan *cookie* sa imenom *primerCookie* i vrednošću *vrednostKojuPamtim*.

1.9 Rad sa uzorcima (pattern maching)

Česta upotreba *JavaScript* funkcija je provera unetih podataka od strane klijenta. Razlog tome je što *JavaScript* upravo ima razvijenu podršku za razne vrste provera i one se obavljaju na klijentskoj strani, tako da korisnik veoma brzo dobija obaveštenje ako neki podatak nije unet u propisanom formatu. Primer takvog korišćenja ovog jezika može da bude polje za poštanski broj, gde korisnik treba da unese petocifreni broj, koji ne sme počinjati nulom. Ako nisu ispunjeni ovi uslovi treba korisnika obavestiti da je došlo do greške.

Ovaj i slični zadaci se rešavaju upotrebom uzoraka (eng. *pattern*) i pozivanjem određenih metoda koje uneti tekst upoređuju sa definisanim uzorkom. Uzorak se još naziva i regularni izraz (*regular expression*) i može se definisati na dva načina:

```
var uPrimer = new RegExp(„HTML”)
```

ili

```
var uPrimer = /HTML/
```

Na oba načina se formira objekat uzorka koji se naziva uPrimer i kome odgovara svaki string koji u sebi sadrži podstring HTML. Kao primer složenijeg uzorka mogu se navesti sledeće deklaracije:

```
var uPrimer = new RegExp(„s$”)
```

ili

```
var uPrimer = /s$/
```

Sada je promenljiva uPrimer uzorak koji odgovara bilo kom stringu koji se završava sa s. Ovakva vrednost je dobijena, jer u okviru uzorka simbol s predstavlja samog sebe, a simbol $ predstavlja metakarakter koji označava kraj stringa.

U sledećoj tabeli su dati mogući karakteri koji se mogu koristiti u okviru uzorka.

Karakter	Predstavlja
Slovo ili broj	Vrednost tog slova ili broja
`\0`	Specijalni NUL karakter
`\t`	Tab znak
`\n`	Nova linija
`\v`	Vertikalni tab znak
`\f`	Form feed

\r	Carriage return
\uxxxx	Unicode karakter definisan pomoću heksadecimalnog boja *xxxx*; na primer, `\u0009` ima isti efekat kao i `\t`

U okviru uzorka se mogu koristiti specijalni simboli sa posebnim značenjem. U okviru sledeće tabele su definisani takvi simboli.

Karakter	Predstavlja pojavljivanje
[...]	Bilo kog karaktera od onih koji su navedeni između `[ i ]`
[^...]	Bilo kog karaktera koji nije naveden između `[ i ]`
.	Bilo kog karaktera osim nove linije
\w	Bilo kog ASCII definisanog slova
\W	Bilo kog karaktera koji nije ASCII definisano slovo.
\d	Bilo koje ASCII definisane cifre
\D	Bilo kog karaktera koji nije ASCII definisana cifra
[\b]	Blanko znak

Na osnovu ove tabele može se zaključiti da uzorak Primer za poštanski broj koji predstavlja petocifreni broj, može da izgleda ovako:

```
Primer = /\d\d\d\d\d/
```

Pomoću ovog uzorka se definiše broj koji se sastoji od 5 cifara. Uzorak bi bio još nepregledniji da je potrebno definisati broj koji se sastoji od 18 ili 28 cifara. Za rad sa ovakvom vrstom uzoraka postoje i specijalne oznake koje su date u sledećoj tabeli.

Oznaka	Značenje

{*n*,*m*}	Ponavljanje prethodne grupe najmanje *n* puta, ali najviše *m* puta
{*n*,}	Ponavljanje prethodne grupe *n* ili više puta
{*n*}	Ponavljanje prethodne grupe tačno *n* puta
?	Ponavljanje prethodne grupe jednom ili nijednom. Isto dejstvo kao i `{0,1}`
+	Ponavljanje prethodne grupe jednom ili više puta. Isto dejstvo kao i `{1,}`
*****	Ponavljanje prethodne grupe nijednom ili više puta. Isto dejstvo kao i `{0,}`
|	Alternative. Pojavljivanje dela izraza sa desne ili pojavljivanje izraza sa leve strane
(...)	Grupisanje simbola u jedan objekat nad kojim se mogu koristiti oznake `*`, `+`, `?`, `|`, itd.
^	Pretraga uzorka se obavlja na početku stringa
$	Pretraga uzorka se obavlja na kraju stringa

Uzorak /\d\d\d\d\d/ se sada može definisati i kao /\d{5}/.

Ispitivanje da li određeni uzorak odgovara stringu može se dodatno definisati i sa određenim atributima. Atributi mogu biti:

`i` – Izvršava *case-insensitive* ispitivanje.

`g` – Izvršava globalno ispitivanje, odnosno pronaći će se sva pojavljivanja definisanog uzorka, a neće se ispitivanje zaustaviti nakon prvog pronađenog uzorka.

M - Rad sa više linija. `^` označava početak linije ili stringa, a `$` predstavlja kraj linije ili stringa.

U dosadašnjem delu poglavlja razmatrani su načini definisanja uzorka. Rečeno je da je primena uzorka u okviru ispitivanja da li određeni string odgovara uzorku ili ne. Takva ispitivanja se mogu izvršavati na dva načina: pomoću metoda koje poziva String koji se ispituje ili pomoću metoda koje poziva definisani uzorak.

Najjednostavniji metod je **search( )**. Ovaj metod ispituje da li u okviru stringa postoji defnisani uzorak, i kao rezultat vraća poziciju njegovog prvog pojavljivanja, ili -1 ako ne pronađe uzorak. Primer:

```
x = /script/
y = "JavaScript".search(x,i);
```

Kao rezultat izvršavanja ovog primera promenljiva y će dobiti vrednost 4.

Ovaj metod ne podržava globalnu pretragu, znači on ignoriše upotrebu atributa g u okviru definicije uzorka.

Sledeći metod koji se koristi je metod **replace()**. Ovaj metod obavlja ispitivanje da li u stringu postoji uzorak i ako postoji zamenu uzorka unutar stringa sa nekom drugom vrednošću. Metod replace() ima dva argumenta, prvi je uzorak, a drugi je string koji treba da zameni uzorak. Ako se u okviru uzorka navede i g atribut, ovaj metod će izvršiti zamenu svakog uzorka koji pronađe u okviru stringa. Ako se ne navede g atribut izvršiće se zamena samo prvog pojavljivanja uzorka.

```
str = "Lana ima 5 pomorandzi i 135 limuna"
promena3u5 = new RegExp("[3-5]", "g")
str.replace(promena3u5, "9")
```

Nakon izvršavanja ovog primera promenljiva str će imati vrednost "Lana ima 9 pomorandzi i 199 limuna", jer je uzorkom definisano da se svako pojavljivanje cifri 3, 4 i 5 zameni sa cifrom 9.

Sledeći metod **match()** je najopštiji metod od mogućih metoda objekta tipa String. Ovaj metod ima samo jedan argument, i to argument tipa uzorka. Rezultat njegovog izvršavanja je niz koji sadrži rezultate ispitivanja. Ako je u okviru uzorka definisan atribut g, rezultat je niz sa svim pojavljivanjem definisanog uzorka. Na primer:

```
"1 plus 2 equals 3".match(/\d+/g)
```

Rezultat izvršavanja metode match() u primeru je niz ["1", "2", "3"], jer je uzorak definisan kao pojavljivanje 1 ili više puta cifre, i to u celom stringu. U okviru navedenog stringa prvo se pronalazi cifra 1, zatim cifra 2, pa cifra 3 i kao rezultat se dobija niz koji kao članove sadrži ove cifre.

U slučaju da nije naveden g atribut, pomoću metode match( ) se ne izvršava globalno ispitivanje, već se ispituje prvo pojavljivanje definisanog uzorka. I u ovom slučaju rezultat je niz, iako se ne sprovodi globalno ispitivanje, i to niz čiji je prvi element ponađeni deo stringa, a ostali članovi niza su delovi uzorka. Znači, ako je rezultat niz a, a[0] je kompletno pronađeni uzorak, a[1] je deo stringa koji odgovara prvom delu uzorka, itd.

Poslednji metod koji postoji kod objekta String, a koji koristi uzorke je metod **split()**. Ovaj metod ima jedan argument i to tipa uzorak. Rezultat izvršavanja ovog metoda je niz koji se dobija kada se string podeli argumentom tj. uzorkom kao separatorom. Na primer:

```
"123,456,789".split(",");
```

Rezultat izvršavanja ovog primera je niz ["123","456","789"], jer je string podeljen pomoću separatora ",".

U ovom primeru uzorak je bio jednostavan i predstavljao je samo jedan karakter. Treba naglasiti da argument metoda split( ) može biti bilo koji regularno definisani uzorak. Na primer:

```
"1,2, 3 , 4 ,5".split(/\s*,\s*/);
```

Rezultat ovog primera je niz ["1","2","3","4","5"], jer je uzorak definisan sa određenim brojem blanko znakova pre i posle zareza, uključujući i zarez.

Najvažniji metod objekta tipa uzorak koji se koristi prilikom ispitivanja je **exec()**. Ovaj metod je veoma sličan String metodu match( ) koji je opisan u prethodnom delu teksta. Razlike je u tome što je kod ovog metoda argument string, a primenjuje se na uzorak, dok je kod metoda match() argument bio uzorak, a primenjivao se na String. Znači rezultat izvršavanje metoda exec() je niz koji sadrži rezultate ispitivanja, definisane na isti način kao i metod match().

Za razliku od metoda match() metod exec() vraća isti rezultat ako postoji ili ako ne postoji atribut g, i to uvek prvo poklapanje i sve relaventne informacije o njemu. Primer:

```
var pattern = /Java/g;
var text = "JavaScript je mnogo zabavniji nego Java!";
var result;
while((result = pattern.exec(text)) != null) {
    alert("Pronadjen `" + result[0] + "'" +
          " na poziciji " + result.index +
          "; sledeca pretraga pocinje od " +
          pattern.lastIndex);
}
```

Rezultat izvršavanja navedenog primera su dva alerta. Prvi sa tekstom "Pronadjen `Java' na poziciji 0; sledeca pretraga počinje od 4", a drugi je sa tekstom "Pronadjen `Java' na poziciji 34; sledeca pretraga počinje od 38".

Sledeći metod iz ove grupe je metod **test()**. Ovaj metod je mnogo jednostavniji nego prethodni. Njegov argument je string, a rezultat true, ako string odgovara uzorku. Na primer:

```
var pattern = /java/i;
pattern.test("JavaScript");
```

Rezultat se dobija true, jer je uzorak pronađen u okviru stringa.

1.10 JSON i JavaScript

JSON (**J**ava**S**cript **O**bject **N**otation) je format za smeštanje i razmenu podataka. Često se koristi u slučajevima kada se podaci šalju od servera ka klijentskoj stranici. Pogodan je za ovakvu vrstu razmene podataka, jer ne zahteva resurse, veoma je razumljiv i intuitivan. JSON sintaksa je nasleđena od sintakse notacije *JavaScript* objekata, ali je JSON format u suštini običan tekst. Čitanje i generisanje JSON podataka je moguće realizovati u proizvoljnom programskom jeziku. Primer JSON objekta:

```
{
"student":[
    {"ime":"Lana", "prezime":"Lanic"},
    {" ime ":"Mina", "prezime ":"Minic"},
    {" ime ":"Petar", "prezime ":"Petrovic"}
]
}
```

Podaci u okviru JSON objekta se definišu kao par ime/vrednost, sa navođenjem navodnika i razdvojeni dvotačkom: "ime":"Lana", dok se nizovi definišu između [] zagrada.

U okviru *JavaScript* koda, podatak koji je dobijen u JSON formatu se može obraditi pomoću parse() funkcije (u okviru primera se pretpostavlja da je JSON objekat upisan u promenljivu text):

```
var obj = JSON.parse(text);
```

Iz objekta obj preneti podaci se mogu koristiti na jednostavan način:

```
<p id="primer"></p>
<script>
document.getElementById("primer").innerHTML =
obj. student [1].ime + " " + obj. student [1].prezime;
</script>
```

Metod document.getElementById() kao rezultat vraća HTML element koji ima ID atribut sa specificiranom vrednošću. Ovaj metod je jedan od najviše korišćenih metoda i moguće ga je koristiti svaki put kada se želi obrađivati ili čitati podatak iz nekog elementa u okviru dokumenta. U slučaju kada na stranici ne postoji element sa specificiranom vrednošću atributa ID dobija se vrednost null.

Zadaci iz programskog jezika JavaScript

Zadatak 1

Napisati primer ispisivanja teksta, formatiranog teksta kao boldovanog tekst, italik teksta i podvučenog tekst i u istom tekstu prikazati upotrebu komentara u JavaScript-u.

REŠENJE

Zadatak pišemo u HTML fajlu, tako što ćemo dodati u okviru tela HTML strane JavaScript blok sa oznakom <script language=”JavaScript”>. Kod novijih veb pregledača dovoljno je napisati i samo <script>. Sadržaj stranice zadatak1.html izgleda ovako:

```
<html>
<head>
   <title>JavaScript</title>
</head>
<body>
  <script language="JavaScript">
   <!--
   //komentar
   /* Komentar u
      vise linija*/
   document.write("Primer");
   document.write("<br><b>Jedan</b><br><i>Dva</i><br><u>Tri</u><br>");
   -->
  </script>
</body>
</html>
```

Zadatak 2

Napisati primer funkcije koja se učitava po učitavanju HTML stranice i ispisuje tekstualnu poruku u vidu upozorenja (alerta).

REŠENJE

```
<html>
<head>
   <script type="text/javascript">
   function poruka()
   { //komentar: pocetak bloka funkcije
         alert("This alert box was called with the onload event");
         //Funkcija poruka ima u sebi alert (upozorenje).
         //Alert mozete da koristite i kada treba
         //da ispitate neku vrednost.
   } //komentar: kraj bloka funkcije
   </script>
</head>

   <body onload="poruka()">
         HTML stranica
         <!--
               po ucitavanju stranice poziva se
               funkcija poruka iz JavaScript-a
         -->
   </body>
</html>
```

Zadatak 3

Napisati primer JavaScript fajla koji se učitava u okviru HTML strane.

REŠENJE

U prvom fajlu, koji ćemo nazvati **Primer3.js** (ekstenzija js potiče od JavaScript) napisaćemo samo programski kod u jeziku JavaScript.

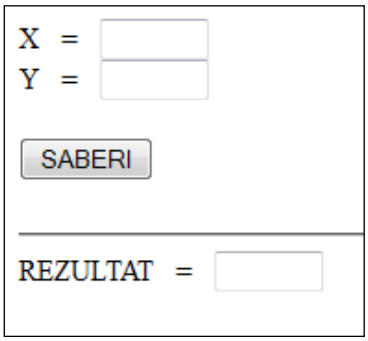
```
document.write("This script is external")
```

U drugom fajlu, koji ćemo nazvati **Primer3.html**, u okviru taga <script> referenciraćemo se na već napravljeni JavaScript fajl, tako što atributu src (skraćenica od source) dodelimo vrednost sa imenom napravljenog JavaScript fajla.

```
<html>
   <head>
   </head>
   <body>
         <script src="Primer3.js">
         </script>

         <p>
         The actual script is in an external script file
         called "Primer3.js".
         </p>
   </body>
</html>
```

Zadatak 4

Napisati HTML formu i JavaScript funkciju koja služi za sabiranje dva broja.



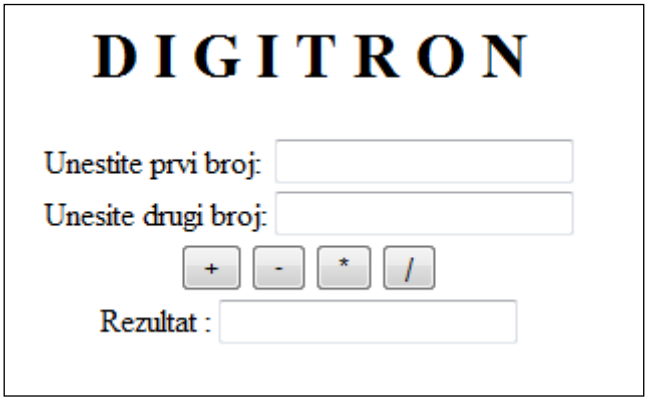
REŠENJE

U ovom zadatku prikazano je izvršavanje događaja miša onClick i dohvatanje vrednosti elemenata forme, a zatim izračunavanje zbira dve vrednosti korišćenjem funkcije Saberi:

```
<html>
   <head>
         <title>Dogadjaji sa misem</title>

         <SCRIPT LANGUAGE="JavaScript">
         function Saberi() {
               var br1 = document.mojaforma.X.value - 0;
               // konverzija u ceo broj
               var br2 = document.mojaforma.Y.value - 0;
               var ukupno = br1 + br2;
               //sabiranje i smeštanje rezultata u promenljivu ukupno
               document.mojaforma.zbir.value = ukupno;
         }
         </SCRIPT>
    </head>

   <body>
      <FORM METHOD="post" NAME="mojaforma">
               X   =  
               <INPUT TYPE="text" NAME="X" SIZE=5> <br>
               Y   =  
               <INPUT TYPE="text" NAME="Y" SIZE=5> <br><br>
               <INPUT TYPE="button" VALUE="SABERI"
               NAME="dugme" onClick="Saberi()"> <br><br>
               <hr>
               REZULTAT   =  
               <INPUT TYPE="text" NAME="zbir" SIZE=5> <br>
         </FORM>
    </body>
    </html>
```

Zadatak 5

Napisati JavaScript funkciju koja na osnovu operacije koju odaberemo izvršava sabiranje, oduzimanje, množenje ili deljenje dva broja iz HTML forme, kao što je prikazano na slici, i rezultat izvršavanja prikazuje u tekstualnom polju.



REŠENJE

U ovom zadatku realizovana je jedna funkcija izracunaj, kojoj se prosleđuje kao argument tip operacije koja se izvršava na digitronu. Svako dugme u okviru HTML forme realizuje jednu operaciju, pa se za operaciju sabiranja poziva funkcija izracunaj sa argumentom 1, za operaciju oduzimanja poziva se ista funkcija sa argumentom 2, itd. Funkcija prihvata vrednosti iz dva tekstualna polja, koja zatim konvertuje u cele brojeve (metodom parseInt), a zatim na osnovu argumenta zna koju aritmetičku operaciju korisnik želi da izvrši i u okviru switch strukture izvršava se jedna grana. Grana, odnosno case koji će se izvršiti, zavisi upravo od prosleđenog argumenta. Pre samog ulaska u switch, koristeći ugrađenu funkciju isNaN, ispitujemo da li su vrednosti oba tekstualna polja, koja su konvertovana u cele brojeve (int) stvarno brojevi. Ako funkcija isNaN za prvi ili drugi broj vraća false, to znači da oba broja jesu brojevi, ako bar jedna vraća true, to znači da bar u nekom od ova dva tekstualna polja za prihvatanje brojeva, nismo imali broj, već možda tekstualni podatak (tipa String) koji nije mogao da se konvertuje u ceo broj (tip int).

Rezultat odabrane operacije, izvršene na ovom digitronu, na kraju smeštamo u treće tekstualno polje, predviđeno za rezultat.

```
<html>
    <head>
            <title>Digitron</title>
            <script language="JavaScript">

            function izracunaj(operacija) {
                var broj1 = parseInt(document.racunaljka.br1.value);
                var broj2 = parseInt(document.racunaljka.br2.value);
                var rezultat = 0;
```

```
            if ((isNaN(broj1) == false) && (isNaN(broj2) == false)) {
                    switch (operacija) {
                        case 1: rezultat = broj1 + broj2;
                                break;
                        case 2: rezultat = broj1 - broj2;
                                break;
                        case 3: rezultat = broj1 * broj2;
                                break;
                        case 4: rezultat = broj1 / broj2;
                                break;
                    }
                    document.racunaljka.rez.value = rezultat;
                } else {
                    alert("Digitron radi samo sa brojevima");
                }
            }
        </script>
 </head>

<body>
<h1 align='center'>D I G I T R O N</h1>
<table border='0' align='center'>
      <form name="racunaljka" action="" method="">
      <tr>
            <td>Unestite prvi broj:</td>
            <td><input type="text" name="br1" value="" /> </td>
      </tr>
      <tr>
            <td>Unesite drugi broj: </td>
            <td><input type="text" name="br2" value="" /> </td>
      </tr>
      <tr>
            <td colspan='2' align='center'>
                  <input type="button" name="saberi"
                  value=" + " onClick="izracunaj(1);" />

                  <input type="button" name="oduzmi"
                  value=" - " onClick="izracunaj(2);" />

                  <input type="button" name="pomnozi"
                  value=" * " onClick="izracunaj(3);" />

                  <input type="button" name="podeli"
                  value=" / " onClick="izracunaj(4);" />
            </td>
      </tr>
      <tr>
            <td colspan='2' align='center'>
```

```
                        Rezultat :
                        <input type="text" name="rez"
                        value="" readonly="readonly"/>
                    </td>
            </tr>
            </form>
        </table>
        </body>
</html>
```

Zadatak 6

a) Napisati JavaScript funkciju koja izračunava faktorijel broja 5.

REŠENJE

Faktorijel je matematička funkcija kojom se izračunava proizvod prirodnih brojeva od 1 do nekog određenog prirodnog broja n, i označava se kao n!. Ova funkcija se koristi u statistici, u zakonima verovatnoće i u kombinatorici. Primer:

n	**n!**
0	1
1	1
2	2
3	6
4	24
5	120

Mi ćemo ovu funkciju rešavati kao rekurzivnu funkciju koju ćemo nazvati factorialFunction(n), odnosno dokle god imamo pozitivan broj n kao argument funkcije, pozivaćemo tu istu funkciju, sa argumentom koji je za jedan manji (n-1) i taj proizvod množiti u jednoj promenljivoj počev od broja n, sve dok ne stignemo do broja 0.

```
<html>
   <head>
         <title>Faktorijel</title>
   </head>
   <body>
         <script language="JavaScript">
         function factorialFunction(n) {
               return n == 0 ? 1 : n*factorialFunction(n - 1)
         }
         document.write("Faktorijel broja 5 je: ", factorialFunction(5))
         </script>
   </body>
</html>
```

b) Napisati funkciju u *JavaScript*-u koja izračunava faktorijel bilo kog broja. Unos broja realizovati preko HTML forme.

REŠENJE

Za razliku od prethodnog slučaja pod stavkom a), sada moramo napraviti jednu HTML formu, koja će nam služiti za prihvatanje broja od strane korisnika, koji želi da izračuna operaciju faktorijel nad tim brojem. Forma treba da ima jedno tekstualno polje za prihvatanje broja od strane korisnika, jedno dugme koje će služiti za pokretanje funkcije factorialFunction, koja će izvršiti operaciju faktorijel i jedno tekstualno polje kao rezultujuće.

Funkcija factorialFunction ima jedan argument n, koji predstavlja broj, nad kojim želimo izvršiti operaciju faktorijel. Zatim funkcijom isNaN treba da ispitamo da li je vrednost argumenta broj ili nije. Takođe, ovde smo stavili i ispitivanje n<0, jer ne postoji faktorijel od negativnog broja.

```
<html>
<head>
   <title>
   Primer faktorijela i formi
   </title>
   <script language="Javascript">
         function factorialFunction(n){
               if (isNaN(n) || n < 0) return "Greska";
               return( n == 0 ? 1 : n*factorialFunction(n - 1));
         }
         function racunaj() {
               n = parseInt(document.faktorijel.argument.value, 10);
               document.faktorijel.result.value=factorialFunction(n);
         }
    </script>
</head>

<body>
<table cellspacing="0" cellpadding="6" border="0"
bgcolor="#99AA99" width="350">
   <tr> <td colspan="2">  </td> </tr>
   <form name="faktorijel">
   <tr>
         <td align="right">
               Unesite <i>n</i> da bi izracunali <i>n!</i>
         </td>
         <td>
               <input type="text" name="argument"
               value="" size="20" maxlength="3">
         </td>
   </tr>
```

```
    <tr>
          <td align="right">
                <input type="button"
                value="Izracunaj" onclick="racunaj()">
                <!-- na dogadjaj klik misem na dugme se
                poziva funkcija racunaj-->
          </td>
          <td> </td>
    </tr>
    <tr>
          <td align="right">
                Rezultat je
          </td>
          <td align="left">
                <input name="result" value="" size="20">
          </td>
    </tr>
    <tr>
          <td colspan="2"> </td>
    </tr>
    </form>
</table>
</body>
</html>
```

Zadatak 7

Napisati funkciju u *JavaScript*-u koja na osnovu dva niza sa istim brojem elemenata, u koje su smešteni različiti citati, u prvom nizu, i autori, u drugom nizu, ispisuje na stranici po jedan citat, ali tako da se citat promeni kada se stranica ponovo osveži (koristiti ugrađenu *Math.random()* funkciju).

REŠENJE

U ovom zadatku imamo dva niza, sa citatima, niz quotes, i sa autorima, niz authors. U primeru smo dali da oba niza imaju po 6 elemenata. Pozivanjem funkcije Math.random() dobijamo realan broj između 0 i 1. Nama je potrebno da za niz sa 6 citata dobijemo broj između 0 i 5 (prvi indeks niza kreće od 0), pa ćemo broj koji vrati funkcija random() pomnožiti sa dužinom niza, odnosno sa quotes.length (u našem slučaju sa 6), a zatim ćemo izvršiti operaciju zaokruživanja floor:

index = Math.floor(Math.random() * quotes.length);

Funkcija floor zaokružuje realan broj na prvi donji ceo broj (na primer: 1.75 na 1).

```
<HTML>
<HEAD>
   <TITLE>Slucajni citati</TITLE>
   <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
   <!--  Ukljucivanje meta taga sa UTF-8 dobijate i
         latinicna srpska slova-->
</HEAD>
<BODY>
   <H1>Slučajni citati</H1>
   <HR>
   <SCRIPT LANGUAGE="JavaScript">
   //citati i autori su smesteni u 2 posebna niza
   quotes = new Array(6);
   authors = new Array(6);
   quotes[0] = "Toliko je bilo stvari u □□životu kojih smo se bojali.
               A nije trebalo. Trebalo je živeti.";
   authors[0] = "Ivo Andrić";
   quotes[1] = "Prijateljstvo se na bira, ono biva, ko zna zbog čega,
               kao ljubav";
   authors[1] = "Meša Selimović";
   quotes[2] = "Zdrav covek ima hiljadu želja,
               a bolestan samo jednu - da ozdravi.";
   authors[2] = "Narodna izreka";
   quotes[3] = "Nemojte da hendikepirate svoju decu time što ćete im
               život učiniti suviše lakim";
   authors[3] = "Duško Radović";
```

```
    quotes[4] = "Čast se ne može oduzeti, ona se može samo izgubiti";
    authors[4] = "Čehov";
    quotes[5] = "Nema sunca bez svetlosti, ni čoveka bez ljubavi";
    authors[5] = "Gete";

    //izracunavanje slucajnog broja, izmedju 0 i 1
    index = Math.floor(Math.random() * quotes.length);

    //prikaz citata u vidu definicione liste
    document.write("<DL>\n");
    document.write("<DT>" + "\"" + quotes[index] + "\"\n");
    document.write("<DD>" + "- " + authors[index] + "\n");
    document.write("</DL>\n");
    </SCRIPT>
    <HR>
    Slucajan citat se prikazuje na ekranu.
    Probajte osvezavanje stranice (Refresh/Reload F5)
    za ponovno ucitavanje stranice i mozda prikaz drugog citata.
    <HR>
</BODY>
</HTML>
```

Diskusija:

Ovo rešenje ne garantuje da će se citat promeniti baš uvek, jer funkcija Math.random() može u nekom slučaju vratiti isti broj dva puta uzastopno. Zato je dobro poslednji citat beležiti kao cookie, proveriti da li je poslednji citat isti kao tekući, pa ako jeste, onda ponovo pozvati funkciju Math.random(), sve dok ne vrati neki citat koji je različit kao poslednji.

Zadatak 8

a) Napisati funkciju u *JavaScript*-u koja prikazuje odloženo izvršavanje naredbe (timeout) i to nakon što kliknemo na dugme. Koristiti ugrađenu funkciju setTimeout.

REŠENJE

U ovom zadatku prikazaćemo primenu funkcije setTimeout. Data je HTML forma sa jednim tekstualnim poljem, sa početnim sadržajem "Pritisni KLIK!" i jedno dugme. Klikom na dugme pokreće se funkcija novitekst(), koja nema argumente, ni povratnu vrednost. Ta funkcija menja vrednost tekstualnog polja na novu vrednost "Cekaj malo!", a zatim poziva ugrađenu funkciju setTimeout, sa argumentima "drugitext()" i 1000.

Prvi argument setTimeout funkcije predstavlja naziv funkcije koja se poziva, nakon isteka 1000 milisekundi, što je navedeno kao drugi argument u pozivu ove funkcije. Nakon 1000 milisekundi, odnosno jedne sekunde, pozvaće se druga metoda drugitext(), koja će ponovo promeniti sadržaj tekstualnog polja.

```
<html>
<head>
<title>Timeout</title>
   <script language="javascript">

   function novitekst()
   {
         document.myform.mytext.value="Cekaj malo!";
         setTimeout("drugitext()",1000);
         //Funkcija kojom se za 1000 milisekundi odlaze pozivanje
         //sledece funkcije, u nasem slucaju drugitext()
   }

   function drugitext()
   {
         document.myform.mytext.value="Promenio sam se!";
   }
   </script>
</head>
<body>
   <form name="myform">
         <input type="text" name="mytext"
               value="Pritisni KLIK!" size="30">
           
         <input type="button" name="but1" value="KLIK!"
               onclick="novitekst()">
   </form>
</body>
</html>
```

b) Napisati funkciju u *JavaScript*-u koja u tekstualnom polju inkrementira brojač za 1, nakon svake 2 sekunde. Koristiti ugrađenu funkciju setInterval.

REŠENJE

Za razliku od setTimeout funkcije koja se poziva samo jednom, nakon isteka vremenskog intervala, funkcija setInterval se periodično poziva. Ona prihvata dva argumenta: prvi je naziv funkcije koja se poziva, a drugi argument je vreme u milisekundama, nakon koga želimo da se željena funkcija (navedena u prvom argumentu), ponovo pozove.

```
<html>
   <head>
   <title> Timer </title>
   <script language="JavaScript">
      var brojac = 0;
      function period() {
            brojac++;
            document.mojaforma.time.value = brojac;
      }
   </script>
   </head>

   <body>
      <form name="mojaforma">
            <input type="text" name="time">
      </form>
      <br>Promena brojaca na svake 2 sekunde.
      <script language="JavaScript">
            setInterval("period()", 2000);
      </script>
   </body>
</html>
```

Zadatak 9

Napisati program koji ispisuje trenutno vreme u formatu SATI:MINUTI:SEKUNDE PM/AM.

REŠENJE

U ovom zadatku prikazaćemo kreiranje novog objekta tipa Date() i dohvatanje određenih atributa tog objekta, korišćenjem ugrađenih metoda getHours() za sate, getMinutes() za minute i getSeconds() za sekunde. Ovde treba obratiti pažnju da za jednocifrene brojeve, sati, minuti, odnosno sekunde se prikazuju bez vodeće nule, pa ukoliko želimo da se vodeća nula vidi, treba ispitati da li su sati (ili minuti ili sekunde) manje od 10. Za ispisivanje oznake AM ili PM, treba ispitati sate, da li imaju vrednost veću od 12 ili nemaju, pošto metoda getHours() vraća vrednost od 0 do 23, kao što je rečeno u teorijskom delu.

```
<HTML>
<HEAD>
<TITLE> Trenutno vreme </TITLE>
<SCRIPT language="JavaScript">
   function Vreme(){
         time = new Date()
         cas=time.getHours()
         minuti=time.getMinutes()
         sekunde=time.getSeconds()
         temp = "" + ((cas>12)?cas-12:cas)
         temp += ((minuti<10)?":0":":")+minuti
         temp += ((sekunde<10)?":0":":")+sekunde
         temp += ((cas>=12)?" P.M.":" A.M.")
         document.vremeForma.cifre.value=temp
         setTimeout("Vreme()",1000)
         //posle svakih 1000milisekundi, odnosno 1 sekunde
         //ponovo se ucitava funkcija Vreme()
   }

</SCRIPT>
</HEAD>

<BODY BGCOLOR="#FFFFFF" onLoad="Vreme()">
<FORM name="vremeForma">
   Trenutno vreme je   <INPUT TYPE="text" NAME="cifre" SIZE=12>
</FORM>
</BODY>
</HTML>
```

Zadatak 10

Napisati program koji u okviru alerta ispisuje kog dana se ove godine proslavlja bivši Dan republike Jugoslavije i koji u tekućem prozoru korišćenjem objekta Date ispisuje trenutni datum, dan u nedelji, ispisan na srpskom jeziku, i trenutno vreme.

REŠENJE

Metoda getDay() koja vraća dan u nedelji, vraća broj između 0 i 6 (0 za nedelju,… i 6 za subotu). Ne postoji ugrađena metoda koja vraća dan u nedelji na srpskom jeziku, pa moramo da napravimo jedan niz, koji ćemo nazvati daniunedelji. Poželjno je da taj niz napravimo tako da na nultom indeksu (prvi element niza) bude nedelja, na indeksu sa vrednošću 1 da bude "ponedeljak", na indeksu 2 "utorak",… i na indeksu sa vrednošću 6 "subota", jer ćemo na taj način moći prilikom pozivanja funkcije getDay(), da dohvatimo indeks korišćenjem te funkcije: daniunedelji[danas.getDay()]).

U ovom zadatku, treba obratiti pažnju da metoda getMonth() vraća vrednosti od 0 (za januar) do 11 (za decembar), pa vrednost promenljive mesec treba da uvećamo uvek za 1, ukoliko želimo da dobijemo broj iz opsega od 1 do 12 za mesec. Takođe, treba obratiti pažnju i na metode za vraćanje godine, jer postoji getYear() i getFullYear(). Metoda getYear() vraća 0 za 1900, 99 za 1999. godinu, a za 2015. godinu vraća 115. Iz tog razloga, na vrednost te metode uvek treba dodavati 1900. Zato je bolje u ovom slučaju koristiti noviju funkciju getFullYear(), koja vraća vrednost godine sa četiri cifre.

```
<html>
   <head>
         <title>Danasnji dan</title>
   </head>
   <body bgcolor="white">
   <script language="javascript">
   daniunedelji = new Array("nedelja", "ponedeljak", "utorak", "sreda",
                            "cetvrtak", "petak", "subota")
   var danas = new Date();

   //jos neki nacini predstavljanja vremena
   dan1 = new Date("November 29, 2015 06:37:00");
   dan2 = new Date(2015,10,29);
   dan3 = new Date(2015,10,29,6,37,0);
   alert("Bivsi dan republike proslavljen je ove godine: " + dan2);

   setTimeout("test()",1000);
   //ova funkcija odlaze izvrsavanje nastavka programskog koda
   //u ovom slucaju funkcije test() za 1000 milisekundi
```

```
   function test() {
         document.write("Danas je " + daniunedelji[danas.getDay()]);
         document.write("<br> Od 1.1.1970. godine je proteklo " +
                           danas.getTime()+" milisekundi " );

         var godina = danas.getYear();
         if(godina < 1000){
               godina += 1900
         }

         var  mesec=danas.getMonth()+1;
         document.write("<br>" + danas.getDate() +  "." +mesec + "." +
                        (godina+" ").substring(2,4));

         var sat = danas.getHours();
         var min = danas.getMinutes();
         var sec = danas.getSeconds();
         document.write("<br>Tacno je: " +sat + " casova " + min +
                        " minuta i " + sec+ " sekundi" );

}

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

Zadatak 11

Napisati program koji pravi tajmer sa minutima i sekundama, a preko *JavaScript*-a napraviti funkcije kojima možete da upravljate štopericom: da je pokrenete, zaustavite vreme ili da resetujete štopericu.

REŠENJE

Zadatak smo realizovali pisanjem jedne glavne i tri pomoćne funkcije. Po učitavanju veb stranice poziva se Reset() funkcija koja postavlja vrednost štoperice na 00:00. Klikom na dugme Start pokrećemo funkciju Start() kojom počinje merenje vremena, štoperica dobija početnu vrednost 00:00 i svake sekunde se u režimu rada štoperica inkrementira za 1 sekundu (ovde koristimo ugrađenu metodu setTimeout sa argumentom 1000 milisekundi). Vreme koje se štopuje meri se kao razlika između trenutnog vremena i startnog vremena. Funkcijom Stop() štoperica se trenutno zaustavlja, ali se može ponovo nastaviti merenje vremena, klikom na dugme Start. Funkcija UpdateTimer() je glavna funkcija ovog programa koju poziva funkcija Start, da bi prikazala izmenjeno vreme na veb stranici, u tekstualnom polju u kome se prikazuje štoperica.

```
<html>
<head>
   <title>Javascript timer</title>
   <script language="JavaScript">
   var timerID = 0;
   var tStart  = null;
   function UpdateTimer() {
         if(timerID) {
               clearTimeout(timerID);
               timerID  = 0;
         }
         if(!tStart) tStart = new Date();
         var tDate = new Date();
         var tDiff = tDate.getTime() - tStart.getTime();
         tDate.setTime(tDiff);
         document.theTimer.theTime.value = "" + tDate.getMinutes() +
                                            ":" + tDate.getSeconds();
         timerID = setTimeout("UpdateTimer()", 1000);
   }

   function Start() {
         tStart   = new Date();
         document.theTimer.theTime.value = "00:00";
         timerID  = setTimeout("UpdateTimer()", 1000);
   }
```

```
   function Stop() {
         if(timerID) {
               clearTimeout(timerID);
               timerID  = 0;
         }
         tStart = null;
   }

   function Reset() {
         tStart = null;
         document.theTimer.theTime.value = "00:00";
   }
   </script>
</head>

<body onload="Reset()" onunload="Stop()">
<center>
   <form name="theTimer">
   <table>
   <tr>
         <td colspan="3" align="center">
               <input type="text" name="theTime" size=5>
         </td>
   </tr>
   <tr>  <td></td>   </tr>
   <tr>
         <td>
               <input type="button" name="start" value="Start"
                onclick="Start()">
         </td>
         <td>
               <input type="button" name="stop" value="Stop"
                onclick="Stop()">
         </td>
         <td>
               <input type="button" name="reset" value="Reset"
                onclick="Reset()">
         </td>
   </tr>
   </table>
   </form>
</center>
</body>
</html>
```

Zadatak 12

Napisati *JavaScript* kod koji u rečenici: “JavaScript is computer language” nalazi pojavljivanje reči “Script”, korišćenjem regularnih izraza i metode *search*.

REŠENJE

```
<html>
   <head></head>

   <script language="JavaScript">
         x = /Script/
         y = "JavaScript is computer language".search(x);
         document.write("Metodom SEARCH pronadjen je uzorak
                        na poziciji:" + y );
   </script>

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

Zadatak 13

Napisati *JavaScript* kod koji u nekom tekstu reč “HTML” zamenjuje rečju “Java” korišćenjem regularnih izraza i metode *replace*.

REŠENJE

```
<html>
   <head></head>
   <script language="JavaScript">
         x1 = /HTML/
         y1 = "html: HTML ili HyperText Markup Language.
               HTML se uci na IP1.
               HtmL je jednostavan jezik".replace(x1, "JAVA");
         document.write("Metodom REPLACE promenjen je uzorak HTML u
                        JAVA pa string izgleda ovako:<br>" + y1 );
         document.write("<br>");

         x2 = /HTML/i
         y2 = "html: HTML ili HyperText Markup Language.
               HTML se uci na IP1.
               HtmL je jednostavan jezik".replace(x2, "JAVA");
         document.write("<br>Upotrebom atributa i nad uzorkom,
                        string izgleda ovako:<br>" + y2 );
         document.write("<br>");
```

```
        x3 = /HTML/ig
        y3 = "html: HTML ili HyperText Markup Language.
              HTML se uci na IP1.
              HtmL je jednostavan jezik".replace(x3, "JAVA");
        document.write("<br>Upotrebom atributa ig nad uzorkom,
                        string izgleda ovako:<br>" + y3 );
   </script>

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

Zadatak 14

Napisati *JavaScript* kod koji prikazuje korišćenje uzoraka i metode match() tako što će u nizu godina:

```
1980-olimpijada 1987 1993 1996-olimpijada  1998 1995 2002 2005 2006 2008-
olimpijada 2008
```

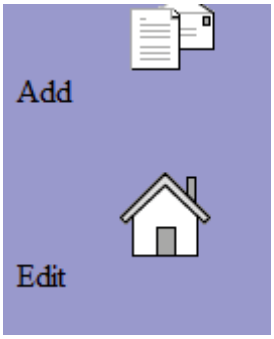
pronaći sve godine koje pripadaju 21.veku.

REŠENJE

```
<html>
   <head></head>

   <script language="JavaScript">
   document.write("Zadatak: U nizu godina, naci sve godine 21.veka");

   x1 = /2\d\d\d/g
   y1 = "1980-olimpijada 1987 1993 1996-olimpijada  1998 1995 2002
         2005 2006 2008-olimpijada 2008".match(x1);
   document.write("<br>Metodom MATCH nad stringom,
                   uzorak je pronadjen kod: <br>" + y1 );

   document.write("<br>");
   </script>

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

Zadatak 15

Napisati *JavaScript* kod koji prilikom prelaska kursora miša na stavke menija “Add” ili “Edit” menja te dve slike u neke druge slike.



REŠENJE:

U ovom zadatku demonstrirana je upotreba događaja miša onMouseOver i onMouseOut. Date su dve slike ADD i EDIT, koje se prelaskom kursora miša preko njih promene, a odlaskom kursora miša sa njih, vrate u prvobitno stanje. Za svrhu promene slika, napravljena je metoda change(id, name) koja ima dva argumenta tipa String. Prvi argument id predstavlja identifikator slike, prvu smo nazvali “add”, a drugu “edit”. Drugi argument predstavlja vrednost i ona može biti za prvu sliku “add” ili “add_o” kada se kursor miša nađe preko prve slike. Za drugu sliku vrednosti koje ona može imati su “edit” ili “edit_o”, u zavisnosti da li je kursor miša iznad te druge slike ili nije. Slike su napravljene kao novi objekti nad klasom Image, koja postoji u programskom jeziku JavaScript i preko atributa src (skraćeno od source) im je dodeljena putanja na kojoj se te slike nalazi. Niz document.images sadrži sve slike u dokumentu, a slikama koje se nalaze u tom nizu se može pristupiti tako što se kao indeks tog niza pozove identifikator slike.

```
<html>
   <head>
         <script language="JavaScript">
          if (document.images) {
               var add_o = new Image();
               add_o.src = './images/add_o.gif';
               var edit_o = new Image();
               edit_o.src = './images/edit_o.gif';

               var add = new Image();
               add.src = './images/add.gif';
               var edit = new Image();
               edit.src = './images/edit.gif';
               alert("Postoje slike u dokumentu");
           }
```

```
            else
            {
                alert("Ne postoje slike u dokumentu!");
            }

            function change(id,name){
            if (document.images) {
              document.images[id].src= eval(name+".src");
            }
            else
            {
                alert("nema promene");
            }
          }
          </script>
          <title>Promena slike</title>
    </head>

    <body bgcolor="#9999CC">
          Add   

          <a href="#" onMouseOut="change('add', 'add')"
                      onMouseOver="change('add', 'add_o')">
          <img name="add" border="0" src="./images/add.gif" Alt="Add">
          </a><br><br>

          Edit   
          <a href="#" onMouseOut="change('edit', 'edit');"
                      onMouseOver="change('edit', 'edit_o');">
          <img name="edit" border="0" src="./images/edit.gif" Alt="Edit">
          </a>
    </body>
</html>
```

Zadatak 16

Napisati *JavaScript* kod koji omogućava da upišete vrednost kolačića (cookie) i da ga posle toga iščitate.

REŠENJE:

U zadatku je data HTML forma sa jednim tekstualnim poljem i dva dugmeta: za postavljanje nove vrednosti cookie-ja i za prikazivanje već upisane vrednosti u cookie promenljivu. Metodom onClick, obrađuje se događaj klik mišem, pa prvo dugme u tom slučaju pokreće funkciju postaviCookie(), a drugo dugme pokreće funkciju prikaziCookie(). Funkcija postaviCookie() pamti podatak iz tekstualnog polja i postavlja ga u promenljivu document.cookie. Funkcija prikaziCookie() prikazuje postojeću vrednost iz promenljive cookie kao upozorenje (alert).

```
<html>
   <head></head>
   <script language="JavaScript">
         function postaviCookie(){
               document.cookie=document.mojaforma.polje.value;
         }
         function prikaziCookie(){
               alert("Cookie je: " + document.cookie);
         }
   </script>

   <body>
         <form name="mojaforma">
               <input type="text" name="polje" size="20">
               <br>
               <input type="button" value="Postavi cookie" name="dugme1"
               onClick="postaviCookie()">
               <input type="button" value="Prikazi cookie" name="dugme2"
               onClick="prikaziCookie()">
         </form>
   </body>
</html>
```

Zadatak 17

Napisati *JavaScript* kod koji kada kliknete na određeni link otvara novi prozor, a nakon toga omogućiti da taj prozor može da se zatvori, odnosno da se nastavi rad sa prvim prozorom.

REŠENJE

U ovom zadatku prikazano je korišćenje objekta Window, koji prikazuje otvaranje novog prozora iz postojećeg. Data je stranica popup.html, koja ima jedan link i nad tim linkom funkciju noviprozor() koja poziva metodu open objekta window. Prvi i najbitniji argument je link (adresa) postojećeg prozora koji želimo da otvorimo. Ostali argumenti ove metode open objašnjeni su u teorijskom delu.

```
<HTML>
<HEAD><TITLE>Otvaranje novog prozora</TITLE>
<SCRIPT LANGUAGE="JavaScript">
   function noviprozor()  {
      iwin = window.open("NoviPopup.html","IWIN",
      "status=no,toolbar=no,location=no,menu=no,width=400,height=300");
   }
</SCRIPT>
</HEAD>
<BODY>
   <H1>Otvaranje novog prozora</H1>
   <HR>
   Ovaj primer ilustruje otvaranje pop-up prozora.
   Kada kliknete na link prozor ce se otvoriti, a kada kliknete na OK,
   vraticete se na ovaj prozor.
   <HR>
   Ovo je trazeni
   <A HREF="#" onClick="noviprozor();">
   link ka novom prozoru</A>.
   <HR>
</BODY>
</HTML>
```

U nastavku je prikazan programski kod druge stranice NoviPopUp.html, koja je rezultat izvršavanja prve stranice, odnosno kada kliknemo na link za otvaranje novog prozora.

```
<HTML>
<HEAD>
   <TITLE>Novi pop-up</TITLE>
</HEAD>
<BODY>
<H1>Novi pop-up</H1>
Dobrodosli na novu stranicu, u novi prozor.
<BR>
```

```
Kliknite na dugme OK ukoliko zelite da zatvorite prozor i
vratite se u prethodni.
<FORM NAME="form1">
   <INPUT TYPE="button" VALUE="OK" onClick="window.close();">
</FORM>
</BODY>
</HTML>
```

Zadatak 18

a) Napisati *JavaScript* kod koji omogućava da prelazite na stranicu napred i stranicu nazad, kao i da ispisuje koji veb pregledač koristite.

REŠENJE:

U ovom zadatku opisano je korišćenje history objekta u okviru objekta window. Objekat history sadrži posećene URL adrese od strane korisnika (u okviru tog prozora veb pregledača). History objekat je sastavni deo window objekta i pristupa mu se kao property-ju window.history. Ovde ćemo demonstrirati povratak na prethodno posećenu stranicu korišćenjem window.history.back() i odlazak na sledeću stranicu, korišćenjem window.history.forward(). Preduslov da bi se izvršile ove metode je da u nizu posećenih stranica imamo prethodno posećenu (za operaciju back) i sledeću posećenu (za operaciju forward). Ukoliko to ne postoji, izvršavanje ovih metoda nije moguće.

Da bismo demonstrirali rad ovog zadatka, prvo je neophodno posetiti neku stranicu, recimo http://www.etf.bg.ac.rs, a zatim u okviru istog prozora da u polju za veb adresu unesemo putanju do ove naše veb stranice. Tada ako kliknemo na dugme Nazad, izvršiće se poziv window.history.back() i ponovo ćemo dobiti stranicu ETF-a, jer je ona prethodna u nizu (istorijatu) naših stranica. Slično može da se demonstrira i rad sa forward() metodom.

```
<html>
   <script language="JavaScript">
         function prikaz() {
               var koristis = window.navigator.appName;
               alert("Vi koristite: " + koristis);
         }
         function nazad() {
               window.history.back();
         }
         function napred() {
               window.history.forward();
         }
   </script>
   <body>
         <form name="podaci">
         <input type="button" value="Sta koristite?" onClick="prikaz()">
         <br>
         <input type="button" value="Nazad <<" onClick="nazad()">
         <input type="button" value=">> Napred" onClick="napred()">
         </form>
   </body>
</html>
```

b) Napisati *JavaScript* kod koji će proširiti prethodni zadatak sa navigacijom, dodavanjem još dva dugmeta: za novi prozor, koji se otvara kao pop-up, i za novi prozor koji se otvara umesto postojećeg prozora.

REŠENJE:

U ovom zadatku pod stavkom b) napisane su još dve dodatne funkcije: noviprozor(), koja se aktivira kada se klikne na "NoviProzor" i fakultet(), koja se aktivira kada se klikne na "ETF". Funkcija noviprozor() je realizovana tako da nema prva dva argumenta: prvi argument za URL adresu prozora koji se otvara i drugi argument, koji specificira ime prozora ili ciljni atribut (_blank, _parent, _self, _top). Druga funkcija fakultet() je realizovana da kao prvi argument ima adresu ETF-a, a kao drugi argument da ima atribut "_self", koji označava da se sadržaj veb stranice otvara u tom već otvorenom prozoru.

```
<html>
<script language="JavaScript">
function prikaz(){
   browser = window.navigator.appName;
   alert("Vi koristite " + browser);
}
function napred(){
   window.history.forward();
}
function nazad(){
   window.history.back();
}
function noviprozor(){
   myWindow = window.open('','','width=200,height=100')
   myWindow.document.write("<p>This is 'myWindow'</p>")
   myWindow.focus()
}
function fakultet(){
   window.open("http://www.etf.bg.ac.rs", "_self")
}
</script>

<form name="forma">
   <input type="button" value="Koji browser koristite?"
   onClick="prikaz()">
   <input type="button" value="NAZAD" onClick="nazad()">
   <input type="button" value="NAPRED" onClick="napred()">
   <input type="button" value="NoviProzor" onClick="noviprozor()">
   <input type="button" value="ETF" onClick="fakultet()">
</form>
</html>
```

Zadatak 19

a) Napisati *JavaScript* funkciju koja kreira 12 dugmića, tako da svaki ima vrednost između 1 i 12.

REŠENJE:

HTML strana poziva funkciju generisiDugme() koja u FOR petlji 12 puta poziva funkciju kreirajDugme(vrednost), sa vrednošću argumenta od 1 do 12. Time se formira 12 dugmića, koji imaju ispisane brojeve od 1 do 12. Funkcija za kreiranje dugmadi poziva zatim metodom onClick drugu funkciju ime(obj), koja će ispisati vrednost argumenta obj kao alert.

```
<html>
    <head>
            <title>Generisi dugmice</title>
            <script language="JavaScript">
                function kreirajDugme(vrednost) {
                    document.write("<input type='button' name='mesec" +
                            vrednost + "' value='" +
                            vrednost + "' onClick='ime(" +
                            vrednost + ")' />");
                }

                function generisiDugmice() {
                    var i;
                    for(i = 1; i<= 12; i++) {
                        kreirajDugme(i);
                    }
                }

                function ime(obj) {
                    alert(obj);
                }
                generisiDugmice();
            </script>
    </head>
    <body>
    </body>
</html>
```

b) Proširiti prethodni primer, tako da kada se klikne na određeni broj, otvara se novi prozor koji sadrži kao naslov ime meseca koji odgovara tom broju (npr. 3-mart, 8-avgust, 12-decembar i slično).

REŠENJE:

Za razliku od stavke a) ovde je kreiran niz imenaMeseci, tako da se Januar nalazi kao prvi element niza na indeksu 0, Februar kao drugi element na indeksu 1,… Decembar kao poslednji element na indeksu 11. Kada se novi prozor otvori, kao naslov nivoa 1 (header 1) ispisuje se ime meseca iz tog niza, tako što se pristupi određenom indeksu.

```
<html>
    <head>
        <title>Zadatak 19</title>
        <script language="JavaScript">
            var imenaMeseci = new Array("Januar", "Februar", "Mart",
                                "April", "Maj", "Jun",
                                "Jul", "Avgust", "Septembar",
                                "Oktobar", "Novembar", "Decembar");

            function noviProzor(broj) {
                var novi = window.open();

                novi.document.write("<html>");
                novi.document.write("<head></head>");
                novi.document.write("<body>");
                novi.document.write("<h1 align='center'>" +
                                    imenaMeseci[broj - 1] + "</h1>");
                novi.document.write("</body>");
                novi.document.write("</html>");
            }

            function kreirajDugme(broj) {
                document.write("<input type='button' value='" + broj +
                             "' onClick='noviProzor(" + broj + ");'/>");
            }

            function kreirajDugmad() {
                var i;
                for (i = 1; i <= 12; i++) {
                    kreirajDugme(i);
                }
            }
            kreirajDugmad();
        </script>
    </head>
    <body> </body>
</html>
```

Zadatak 20

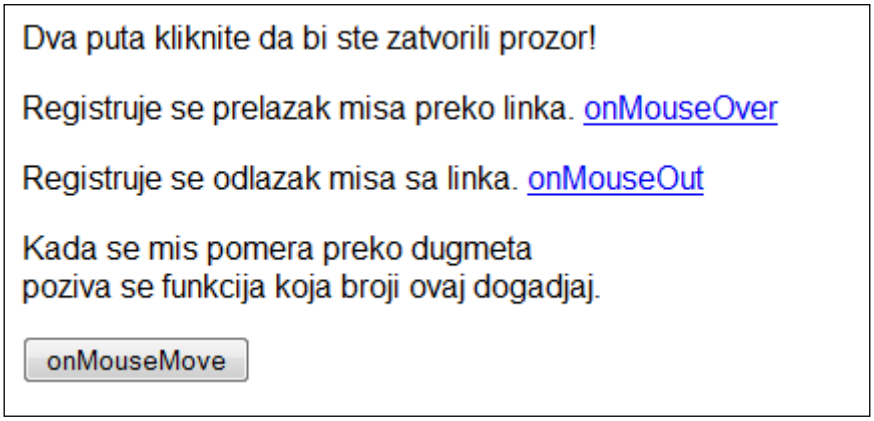
Napisati *JavaScript* program koji ima dva linka i jedno dugme i izvršava sledeće događaje miša: onMouseOver, onMouseOut, onMouseMove. Kada se pređe mišem preko prvog linka, treba obraditi događaj onMouseOver, kada se mišem kursor skloni sa drugog linka treba obraditi događaj onMouseOut, a kada se pomera kursorom miša preko dugmeta, treba obraditi događaj onMouseMove. Takođe, realizovati funkciju za zatvaranje postojećeg prozora, ukoliko se dva puta klikne na pozadinu (bilo koji prazan prostor na stranici).

REŠENJE:

Dvoklik je obrađen događajem miša ondblclick, napisanim u okviru body taga. Ovaj događaj poziva funkciju closeWindow u kojoj smo ispisali alert sa porukom “Gotovo je!” i pozvali metodu close() objekta window.

Događaji onMouseOver i onMouseOut realizovani su u linkovima, i oni takođe ispisuju alert sa porukom o tom događaju koji je obrađen.

Funkcija mouseOverCounter poziva se uvek kada se kursorom miša pređe preko dugmeta sa nazivom “onMouseMove”. Na početku, van ove funkcije, definisan je brojač sa nazivom counter, čija je inicijalna vrednost postavljena na 0. Unutar funkcije, vrednost ovog brojača se inkrementira i kao alert ispisuje poruka sa brojem prelazaka kursora preko ovog dugmeta.




```
<html>
   <head>
   <title>Dogadjaji sa misem</title>

      <script language="JavaScript">
          var counter=0;

          function closeWindow(){
              alert("Gotovo je!");
              window.close();
           }
```

```
        function mouseOverCounter(){
            counter++;
             if(counter==1){
                document.title = counter + " prelazak preko dugmeta!";
             }
             else{
                document.title = counter + " prelaska preko dugmeta!";
             }
         }
      </script>
   </head>


   <body ondblclick="closeWindow()";>
      <p>
            <font face="arial" size=3>
            Dva puta kliknite da bi ste zatvorili prozor!
            </font>
      </p>
      <p>
            Registruje se prelazak misa preko linka.
            <a href="#" onMouseOver="alert('Event: onMouseOver');">
            onMouseOver
            </a>
      </p>
      <p>
            Registruje se odlazak misa sa linka.
            <a href="#" onMouseOut="alert('Event: onMouseOut');">
            onMouseOut
            </a>
      </p>
      <p>
            Kada se mis pomera preko dugmeta <br>
            poziva se funkcija koja broji ovaj dogadjaj.
      </p>
      <form>
            <input type="button" value="onMouseMove"
            onMouseMove="mouseOverCounter();">
      </form>
   </body>
</html>
```

Zadatak 21

Napisati *JavaScript* funkciju koja proverava ispravnost adrese e-pošte. Adresa e-pošta treba da bude u sledećem formatu:

a) ime, koje sadrži najmanje jedan karakter koji je slovo;
b) opciono tačku ili srednju crtu, nakon imena;
c) prezime, koje sadrži najmanje jedan karakter koji je slovo;
d) stavke b) i c) mogu da se ponavljaju 0 ili više puta;
e) znak @;
f) domen koji sadrži najmanje 3 karaktera, koji su alfabetski karakteri ili numerici;
   *alfabetski karakteri i numerici su: A-Z, a-z, 0-9 i _ (donja crta)
g) opciono tačku ili srednju crtu, nakon domena;
h) poddomen, koji sadrži najmanje jedan alfabetski karakter ili numerik;
i) stavke g) i h) mogu da se ponavljaju 0 ili više puta;
j) domen za zemlju ili opšti domen, koji sadrži 2 ili 3 slovna karaktera, i koji se na kraju e-mail adrese može ponavljati najmanje jednom, ili više puta.

REŠENJE:

```
<html>
<head>
<title>email</title>

<script language="javascript">
function proveri()
{
   var test=/^[A-Za-z]+([\.-]?[A-Za-z]+)*@\w{3,}([\.-]?\w+)*
               (\.[A-Za-z]{2,3})+$/;
   var tekst=document.forma.izraz.value;
   alert(tekst);
   var rezultat = tekst.match(test);
   if (rezultat != null)
         alert("E-mail adresa je OK.")
   else
         alert("E-mail adresa nije u redu.");
}
</script>
</head>
```

```
<body>
<form name="forma" method="post" action="">
  <table width="50%"  border="0" align="center">
   <tr>
         <th scope="col">Unesite izraz za proveru:
               <input type="text" name="izraz" size="20">
         </th>
   </tr>
   <tr>
         <td align="center">
               <input type="submit" name="Submit" value="Submit"

               onClick="proveri()">
         </td>
    </tr>
  </table>
</form>
</body>
</html>
```

Adrese koje zadovoljavaju traženi regularni izraz:

draskovic@etf.rs
draskovic@etf.bg.ac.rs
n.kojic@etf.rs
jelica.protic@etf.bg.ac.rs
dr.bosko.nikolic@etf-bg.rs
bosko-nikolic@etf.rs

Adrese koje ne zadovoljavaju traženi regularni izraz:

drazen?draskovic@etf.rs, zbog znaka pitanja
bosko_nikolic@etf.rs, zbog donje crte
petar_petrovic_njegos@lovcen123.cg, zbog donje crte
draskovic123@etf.rs, zbog brojeva 123
jovan.jovanovic.zmaj@novi_sad-com, zbog nekorektnog domena (iza @ nema tačke)

Zadatak 22

Napisati *JavaScript* funkciju koja proverava ispravnost telefonskog broja u Beogradu. Primeri koji treba da zadovolji regularni izraz su:
0112345678,
011 23 45 678,
011-23-45-678,
011-234-5678,
(011) 234-5678,
011/234-5678.

REŠENJE:

```
<html>
<head>

   <title>Provera telefonskog broja</title>

   <script language="javascript">
   function proveri()
   {

      var tel=/^(\d{3}(\s?|-)\d{2}(\s?|-)\d{2}(\s?|-)\d{3}|
                  \d{3}-\d{3}-\d{4}|[(]\d{3}[)]\s*\d{3}-\d{4}|
                  \d{3}\/\d{3}-\d{4})$/;

      var tekst=document.forma.izraz.value;

      if (tel.test(tekst))
      {
            var rezultat=tekst.match(/\d+/g);
            var i=0;
            var novirez="";
            while (rezultat[i] ) {
            novirez+=rezultat[i++];
            }
            document.forma.rezultat.value=novirez;
      }
      else
      {
            document.forma.rezultat.value="Pogresan unos";
      }
    }
   </script>
</head>
```

```
<body>
<form name="forma">
   <table width="50%"  border="0" align="center">
      <tr>
            <td>Unesite izraz za proveru:
                  <input type="text" name="izraz" size="20">
            </td>
      </tr>
      <tr>
            <td align="center">
                  <input type="button" name="Submit"
                  value="Submit" onClick="proveri()">
            </td>
      </tr>
      <tr>
            <td>Rezultat:
                  <input type="text" name="rezultat" size="20">
            </td>
      </tr>
   </table>
</form>
   </body>
   </html>
```

Zadatak 23

Napisati *JavaScript* funkcije koje proveravaju da li su datum i vreme ispravno unešeni. Datum mora biti u formatu: dd.mm.gggg. gde dd i mm mogu biti i jednocifreni brojevi. Takođe obratiti pažnju na broj dana svakog meseca i da li je prestupna godina (prestupna godina ima i 29. februar!). Godina treba da bude iz opsega od 2000. do 2100.

REŠENJE:

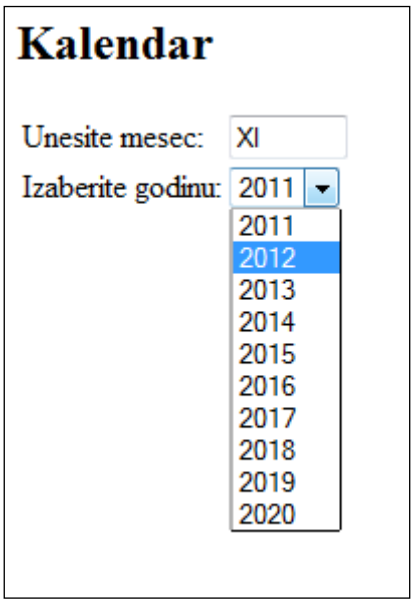
```
<html>
<head>
<title>DatumVreme</title>
<script language="javascript">
   function provera() {
      var poruka="";
         if (ProveriDatum(document.forma.datum.value))
            poruka += "Datum je ok. \n";
      else
            poruka+="Datum nije korektno unesen. \n";

         if (ProveriVreme(document.forma.vreme.value))
               poruka+="Vreme je ok. \n";
         else
               poruka+="Vreme nije korektno uneseno. \n";
      alert(poruka);
    }

   function ProveriDatum(date){
      var provera = /^([1-2][0-9]|[3][0-1]|[1-9]|[0][1-9])\.
                     ([0][1-9]|[1][0-2]|[1-9])\.([2][0|1][0-9]{2})\.$/;
         var danstr="";
         var dan=0;
         var godinastr="";
         var godina=0;
         var mesecstr="";
         var godinastr=0;

      if (provera.test(date)) {
               if(date.substr(2,1)== ".") {
                  danstr=date.substr(0,2);
                     if (date.substr(5,1)=="." ) {
                        mesecstr=date.substr(3,2);
                           godinastr=date.substr(6,4);
                     } else if(date.substr(4,1)==".") {
                           mesecstr=date.substr(3,1);
                           godinastr=date.substr(5,4);
                     }
               } else {
```

```
                if (date.substr(1,1)== ".") {
                        danstr=date.substr(0,1);
                        if (date.substr(4,1)=="." ) {
                           mesecstr=date.substr(2,2);
                              godinastr=date.substr(5,4);
                        } else if(date.substr(3,1)==".") {
                              mesecstr=date.substr(2,1);
                              godinastr=date.substr(4,4);
                        }
                     }
               }
               dan=parseInt(danstr,10);
               mesec=parseInt(mesecstr,10);
               godina=parseInt(godinastr,10);
               if (mesec==2){
                     if((godina%4 == 0)&&(!(godina%100==0)||
                        (godina%400==0))){
                        if (dan > 29) return false;
                     } else {
                        if (dan>28) return false;
                     }
               } else {
                  if ((dan > 30) && ((mesec == 4) || (mesec == 6) ||
                         (mesec == 9) || (mesec == 11)))
                  return false;
            }
      return true;
   } else {
      return false;
   }
}

function ProveriVreme(vreme)
{
   var provera = /^([0-1][0-9]|[0-9]|[2][0-3]):([0-5][0-9]|[0-9])$/;

   if (provera.test(vreme))
      return true;
   else
         return false;
}
</script>
</head>
<body>
<form name="forma" method="post" action="">
   <table bgcolor="#FFCC99">
   <tr>
      <td colspan="2"> </td>
   </tr>
```

```
   <tr>
      <td>Unesite datum :</td>
      <td> <input type="text" name="datum" value="dd.mm.gggg."></td>
   </tr>
   <tr>
      <td>Unesite vreme :</td>
      <td> <input type="text" name="vreme" value="hh:mm"></td>
   </tr>
   <tr>
      <td colspan="2" align="center">
            <input type="submit" name="Submit"
         value="Submit" onClick="provera()">
      </td>
   </tr>
   <tr>
      <td colspan="2"> </td>
   </tr>
   </table>
</form>
</body>
</html>
```

Zadatak 24

Napisati *JavaScript* program koji na osnovu unešenih podataka u formu, meseca koji je predstavljen rimskim brojem, i godine, koja se odabira iz padajuće liste, iz opsega od 2011. do 2020. godine, u novom prozoru otvara kalendar za izabrani mesec i izabranu godinu.



REŠENJE:

```
<html>
<head>
<script language="JavaScript">
   function otvori(){
         rim_mesec = document.forma.mes.value;
         godina = document.forma.god.value;
         switch(rim_mesec){
               case 'I': mesec = 1; br_dana=31; break;
               case 'II': mesec = 2;
                     if(godina%400==0 || (godina%100!=0 && godina%4==0))
                     br_dana=29; else br_dana=28; break;
               case 'III': mesec = 3; br_dana=31; break;
               case 'IV': mesec = 4; br_dana=30; break;
               case 'V': mesec = 5; br_dana=31; break;
               case 'VI': mesec = 6; br_dana=30; break;
               case 'VII': mesec = 7; br_dana=31; break;
               case 'VIII': mesec = 8; br_dana=31; break;
               case 'IX': mesec = 9; br_dana=30; break;
               case 'X': mesec = 10; br_dana=31; break;
               case 'XI': mesec = 11; br_dana=30; break;
               case 'XII': mesec = 12; br_dana=31; break;
         }
```

```
            datum = new Date(godina,mesec-1,1);
            dan = datum.getDay();
            prozor = window.open('','','width=300, height=250');
            prozor.document.write("<h2>Kalendar za: " + rim_mesec + "/" +
                                    godina + "</h2>");
            prozor.document.write("<table border=1>");
            prozor.document.write("<tr><td>N</td> <td>P</td> <td>U</td>
                  <td>S</td> <td>C</td> <td>P</td> <td>S</td> </tr>");
            prozor.document.write("<tr>");

            for(i=0; i<dan; i++)
                  prozor.document.write("<td></td>");
            for(j=1; j<=7-i; j++)
                  prozor.document.write("<td>"+j+"</td>");

            prozor.document.write("</tr>");

            while(j<=br_dana){
                  prozor.document.write("<tr>");
                  for(i=0; i<7 && j<=br_dana; i++, j++){
                        prozor.document.write("<td>"+j+"</td>");
                  }
                  prozor.document.write("</tr>");
            }
            prozor.document.write("</table>");
            prozor.focus();
   }
</script>
</head>

<body>
   <h2>Kalendar</h2>
   <form name="forma">
   <table>
   <tr>
      <td>Unesite mesec:</td>
      <td><input type="text" name="mes" size="5" value="XI"></td>
   </tr>
   <tr>
      <td>Izaberite godinu:</td>
      <td>
            <select name="god" onChange="otvori()">
                  <option>2011</option>
                  <option>2012</option>
                  <option>2013</option>
                  <option>2014</option>
                  <option>2015</option>
                  <option>2016</option>
                  <option>2017</option>
```

```
                <option>2018</option>
                <option>2019</option>
                <option>2020</option>
            </select>
        </td>
    </tr>
    </table>
    </form>
</body>
</html>
```

Zadatak 25

Data je HTML strana koja sadrži tri tekstualna polja (za šifru, potvrdu šifre i poštanski kod mesta), tri čekboksa i jedno dugme. Kada se pritisne dugme poziva se *JavaScript* funkcija proveri(). Napisati kod ove funkcije koja proverava da li su vrednosti tekstualnog polja šifra i potvrda šifre iste, i da li je u tekstualno polje poštanski kod upisan petocifreni broj koji počinje sa cifrom 1, 2 ili 3. U slučaju da je provera uspešna prikazati poruku »U redu je«, a ako provera nije uspešna prikazati poruku »Nije u redu!«. U okviru funkcije proveriti i stanje čekboksova, pa ukoliko su sva tri čekboksa potvrđena prikazati poruku »Da li ste sigurni?«.

REŠENJE:

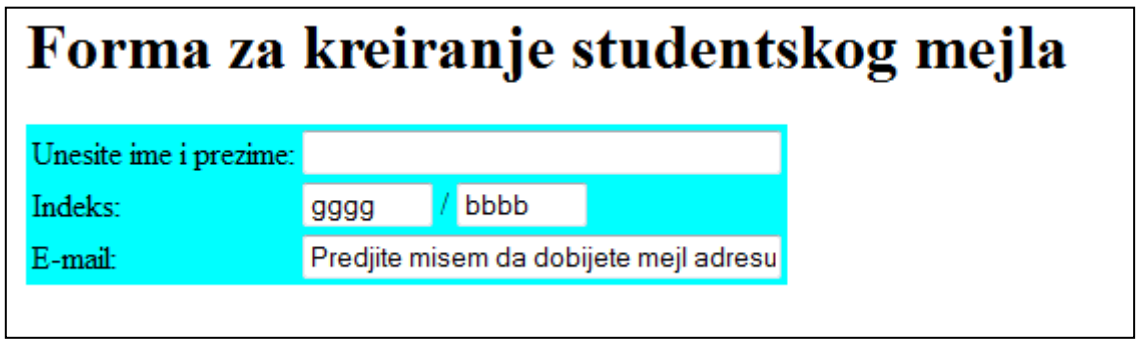
```
<html>
<script language="JavaScript">
   function proveri(){
         var uzorak = /^[1|2|3]\d{4}$/
         if(document.forma.sifra1.value==document.forma.sifra2.value)
               if(uzorak.test(document.forma.ptt.value))
                     alert("U redu je!");
               else alert("Nije u redu! PTT broj nije
                           u trazenom formatu");
         else alert("Nije u redu! Sifre nisu iste!");

         if(document.forma.izbor1.checked &&
            document.forma.izbor2.checked &&
            document.forma.izbor3.checked)
               alert("Da li ste sigurni?");
   }
</script>
<body>
   <form name="forma">
         <table>
         <tr>
               <td>Sifra: </td>
               <td><input type="password" name="sifra1" size="20"></td>
         </tr>
         <tr>
               <td>Potvrda sifre:</td>
               <td><input type="password" name="sifra2" size="20"></td>
         </tr>
         <tr>
               <td>Postanski broj:</td>
               <td><input type="text" name="ptt" size="10"></td>
         </tr>
         <tr>
               <td colspan="2">
               Predmet1
```

```
                <input type="checkbox" name="izbor1">
                Predmet2
                <input type="checkbox" name="izbor2">
                Predmet3
                <input type="checkbox" name="izbor3">
                </td>
          </tr>
          <tr>
                <td>
                      <input type="button" value="PROVERA"
                      onClick="proveri()">
                </td>
                <td></td>
          </tr>
          </table>
    </form>
</body>
</html>
```

Zadatak 26

Data je HTML strana koja sadrži tekstualno polje za unos imena i prezimena, tekstualna polja za unos godine upisa fakulteta i broja indeksa i tekstualno polje za adresu elektronske pošte. Na osnovu imena, prezimena, godine upisa i broja indeksa studenta, potrebno je formirati studentsku adresu e-pošte, koja treba da bude u formatu: piGGbbbbd@student.etf.rs

gde su:
p - inicijal za prezime,
i - inicijal za ime,
GG - poslednje dve cifre godine upisa na fakultet,
bbbb - broj indeksa (ukoliko se upiše sa manje od 4 cifre,
onda se dopunjuje vodećim nulama na početku),
d - standardna oznaka da je student osnovnih akademskih studija,
@student.etf.rs - faklultetski domen za studentske mejl adrese.



REŠENJE:

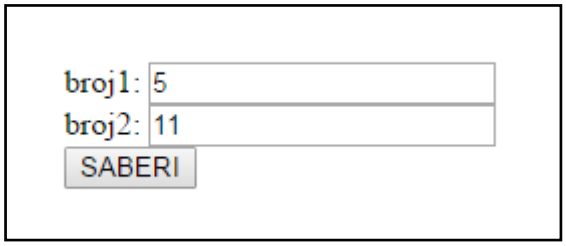
```
<html>
<head>
<script language="JavaScript">
function formirajMejl(){
   document.mojaForma.email.value = "";
   student = document.mojaForma.ime.value;
   var inicijali;
   if(student!="") {
         niz = student.split(/\s+/);
         ime = niz[0];
         prezime = niz[1];
         if(niz[0])
               inicijalime = niz[0].charAt(0).toLowerCase();
         if(niz[1])
               inicijalprezime = niz[1].charAt(0).toLowerCase();
         inicijali = inicijalprezime + inicijalime;
   }
```

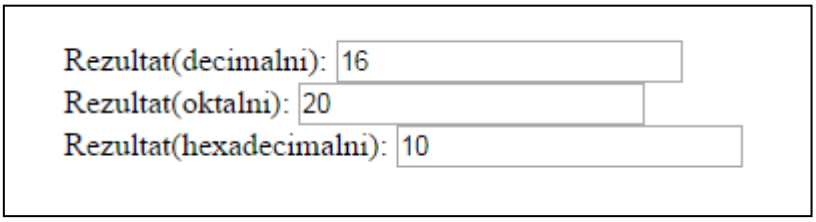
```
   godina = document.mojaForma.god.value;
   indeks = document.mojaForma.brindeks.value;
   uzorak_godina = /20([0][0-9]|[1][0-2])/
   rezultatgod = godina.match(uzorak_godina);
   var brojindeksa;

   if(indeks.length==1)
         brojindeksa = "000" + indeks;
   else if(indeks.length==2)
         brojindeksa = "00" + indeks;
   else if(indeks.length==3)
         brojindeksa = "0" + indeks;
   else brojindeksa = indeks;
   document.mojaForma.brindeks.value = brojindeksa;
   uzorak_indeks = /((\d){3}[1-9])|(\d{2}[1-9]\d)|
                    (\d[1-9]\d{2})|([1-9]\d{3})/
   rezultatind = brojindeksa.match(uzorak_indeks);

   if(inicijali!=null && rezultatgod!=null && rezultatind!=null){
         godina = godina.substring(2,4);
         email = inicijali + godina + brojindeksa + "d@student.etf.rs";
         document.mojaForma.email.value = email;
   } else {
         document.mojaForma.email.value = "Greska!";
   }
}
</script>
</head>

<body>
<h1>Forma za kreiranje studentskog mejla</h1>

<form name="mojaForma">
<table border="0" bgcolor="cyan">
   <tr>
         <td>
               Unesite ime i prezime:
         </td>
         <td>
               <input type="text" name="ime" size="35">
         </td>
   </tr>
   <tr>
         <td>
               Indeks:
         </td>
         <td>
               <input type="text" name="god" placeholder="gggg"
               maxlength="4" size="6"> /
```

```
                <input type="text" name="brindeks" placeholder="bbbb"
                maxlength="4" size="6">
            </td>
        </tr>
        <tr>
            <td>
                E-mail:
            </td>
            <td>
                <input type="text" name="email" size="35"
                onMouseOver="formirajMejl();"
                value="Predjite misem da dobijete mejl adresu!">
            </td>
        </tr>
    </table>
    </form>
    </body>
    </html>
```

Zadatak 27

Napisati *JavaScript* funkciju koja na osnovu dva unešena cela broja sa osnovom 10, kao što je prikazano na prvoj slici, izračunava zbir ta dva broja, i taj zbir prikazuje sa osnovom 10, 8 (oktalni brojevi) i 16 (heksadecimalni brojevi) i zatim taj rezultat prikazuje u formi u novom prozoru, kao što je prikazano na drugoj slici.





REŠENJE:

Kada funkcija prihvati dva podatka iz tekstualnih polja sa HTML forme, potrebno je da proverimo da li su ti podaci brojevi ili nisu. Prvo ćemo pokušati metodom parseInt da izvršimo konverziju tih podataka, dobijenih iz forme, u cele brojeve. Zatim ćemo da ispitamo da li je konverzija u ceo broj uspešno izvršena, a to možemo da uradimo pozivom funkcije isNaN. Funkcija isNaN će vratiti true, ako podatak nije broj, i false, ako je podatak broj. Nakon što utvrdimo da su brojevi u redu, onda ćemo sabrati ta dva broja i dobiti zbir sa osnovom 10, a zatim ćemo taj zbir metodom toString sa argumentima 8, i 16, pretvoriti u oktalni broj, odnosno heksadecimalni broj.

```
<html>
    <head>
        <title>Forme u novom prozoru</title>

        <script language="JavaScript">
            function otvori() {
                var br1 = parseInt(document.formaA.TekstA1.value);
                var br2 = parseInt(document.formaA.TekstA2.value);
                var rez = 0;
                var noviProzor;

                if ((isNaN(br1) == false) && (isNaN(br2) == false)) {
                     rez = br1 + br2;

                     noviProzor = window.open();

                     noviProzor.document.write("<html>");
                     noviProzor.document.write("<head>");
                     noviProzor.document.write("</head>");
                     noviProzor.document.write("<body>");
                     noviProzor.document.write("<form name='formaB'>");
                     noviProzor.document.write("Rezultat(decimalni):
                           <input  type='text'  name='TekstB1'  value=''>
                           <br>");
                     noviProzor.document.write("Rezultat(oktalni):
                           <input type='text' name='TekstB2' value='' >
                           <br>");
                     noviProzor.document.write("Rezultat(hexadecimalni):
                           <input type='text' name='TekstB3' value='' >
                           <br>");
                     noviProzor.document.write("</form>");
                     noviProzor.document.write("</body>");
                     noviProzor.document.write("</html>");

                     noviProzor.document.formaB.TekstB1.value = rez;
                     noviProzor.document.formaB.TekstB2.value =
```

```
                                rez.toString(8);
                        noviProzor.document.formaB.TekstB3.value =
                                rez.toString(16);
                    } else {
                        alert("Unesi oba broja,
                                ako zelis da se izracuna zbir!");
                     }
                }
            </script>
        </head>

        <body>
            <form name="formaA">
                    broj1: <input type="text" name="TekstA1" value=""/> <br/>
                    broj2: <input type="text" name="TekstA2" value=""/> <br/>
                    <input type="button" name="SABERI"
                    value="SABERI" onClick="otvori()">
            </form>
        </body>
</html>
```

+ JOŠ NEKI ZADACI SA ROKOVA!

2 Java Servleti

2.1 Uvod u Java Servlete

Ako se pogleda istorijski razvoj, priča o Javi kao programerskom jeziku na serverskoj strani aplikacija je počela 1997. godine, kada je Sun Microsystems realizovao "Java Web Server" beta i Java Servlet Developers Kit. Servleti su specifične Java klase koje imaju mogućnost da se izvršavaju na strani servera. Servleti dinamički prihvataju i obrađuju mrežne zahteve i odgovore, najčešće koristeći HTTP protokol.

U junu 1999. godine, Sun je predstavio Java Server Pages (JSP), koje su omogućile realizaciju Java koda zajedno sa dizajniranim JavaScript i HTML stranicama. JSP stranice, koje su kasnije još razvijane sa JSP verzijom 2.0, dizajnirane su da izvrše enkapsulaciju domenske logike u okviru standardnih i dodatih tagova, i tako izvrše odvajanje ovog sloja od komponenti prezentacione logike na JSP strani. Kasniji koncept je nazvan "*the stuff that people see*" kada se programeru omogućava da radi sa gotovim HTML-baziranim šablonima. Idealno, JSP strane koriste tagove da bi pristupile bazama podataka i zaštitile pravila domena, i od statičkih ili dinamički generisanih tekstualnih šablona, kao što su XML ili XHTML, da bi se generisale krajnje stranice za korisnika. Pojava Jave kao serverske komponente je omogućila samim programerima upotebu alata koji je apsolutno objektno orijentisan i modularan, i na taj način jednostavnije i efikasnije organizovanje projekata različitih veličina. Pored toga, dobila se mogućnost korišćenja i velikog skupa pomoćnih Java klasa, kao što su klase za rad sa Stringovima, sa ulazno-izlaznim fajlovima, matematičke kalkulacije,... Sledeća velika prednost ove tehnologije je mogućnost korišćenja u okviru više platformi. Moguće je realizovati Internet aplikaciju, zapakovati je u specijalne JAR fajlove za veb komponente, koji se nazivaju Web Application Archive (WAR) fajlovi, pa postaviti dobijene WAR fajlove na različite servere, pod različitim operativnim sistemima. Java veb komponente nisu vezane za određene operativne sisteme, ili za određeni serverski softver, kao neke druge softverske tehnologije za razvoj ove vrste aplikacija. Od kraja 2003, Java je već dobila status predvodnika razvoja serverske strane. Servleti i JSP stranice su uključene u Java 2 Enterprise Edition (J2EE), veoma prihvaćenu „enterprise“ tehnologiju za mrežno bazirano i distribuirano programiranje. Tako su stotine hiljada programera širom sveta počeli da razvijaju "veb sloj" u okviru J2EE-baziranih tehnologija, koristeći servlete i JSP stranice, kao i specijalne veb frejmvorke, kao što su Strust i JavaServer Faces.

2.2 O tehnologiji

Servleti su Java komponente za rad sa serverima koji su orijentisani na komunikaciju tipa request/response. Izvršavaju se pomoću veb servera i sa podrškom za Javu [slika 1]. Ove Java

klase mogu da se izvršavaju na mnogim različitim serverima, jer servlet API, koji se koristi za pisanje servleta, ne koristi ništa iz serverskog okruženja ili protokola. Servleti su postali sastavni deo skoro svih HTTP servera koji na taj način podržavaju servlet tehnologiju. Servleti se koriste kod:

- Komunikacije između korisnika - Servlet može da obradi više zahteva konkurentno i da ih ujedno sinhronizuje. Na ovaj način servleti podržavaju sisteme kao što su on-line konferencije;
- Prosleđivanja zahteva - Servleti mogu da proslede zahteve ostalim serverima i servletima. Na taj način se odražava bolji balans opterećenja nekoliko servera koji imaju istu funkciju.

Njihov osnovni zadatak je:

- Prihvatanje i čitanje eksplicitnih podataka poslatih od strane klijenta, u suštini podaci dobijeni sa forme klijenta;
- Prihvatanje i čitanje implicitnih podataka poslatih od strane klijenta, podaci dobijeni na osnovu hedera zahteva;
- Generisanje rezultata u zavisnosti od primenjene poslovne logike;
- Slanje eksplicitnih podataka klijentu, u formi HTML stranica;
- Slanje implicitnih podataka klijentu u formi kodova statusa i samog heder-a odgovora.

2.3 Osnovna struktura servleta

Naglašeno je da su servleti Java klase koje moraju da imaju određeni format. Primer jednog najosnovnijeg servleta bi bio:

```
import java.io.*;
import java.servlet.*;
import java.servlet.http.*;

public class Zdravo extends HttpServlet{
   public voig doGet(HttpServletRequest request,
                     HttpServletResponse response)
                     throws ServletException, IOException {
         PrintWriter out = response.getWriter();
         out.println("Zdravo!");
   }
}
```

U ovom osnovnom primeru, u okviru servleta se generiše tekst “Hello World” koji se prikazuje na strani klijenta, u trenutku kada se zatraži definisani servlet. Ovako definisanu klasu nije moguće kompajlirati samo sa osnovnim Java okruženjem, već je potrebno imati dodatne pakete. I nakon uspešnog kompajliranja datu klasu nije moguće samostalno pokretati, već određenim pozivom preko veb servera.

Današnja popularna vizuelna okruženja ovaj postupak u mnogome olakšavaju. Kod većine Java baziranih razvojnih okruženja potrebno je samo pokrenuti odgovarajući wizard (na primer u okviru novog veb projekta izabrati opciju Insert => New => Servlet) i dobija se automatski generisani programski kod u koji treba ubaciti programski kod iz prethodnog primera.

Ako je aplikacija pravilno instalirana, sam poziv servleta se može obaviti na više načina. Servleti se mogu pozivati direktno upisom njihove URL putanje unutar veb pregledača. Format URL putanje generalno zavisi od servera koji se koristi. Primer za Jakarta Tomcat veb server kod koga je u direktorijumu ,,TOMCAT_HOME/webapps“ instalirana aplikacija izgleda ovako:

```
http://machine-name:port/Direktorijum_Aplikacije/Ime_Servleta
http://localhost:8080/Proba/ObradaPostMetodaServlet
```

Ako se želi simulirati GET metoda poziva servleta:

```
http://localhost:8080/Proba/ObradaGetMetodaServlet?par1=88
```

Servlet se može pozivati i sa klijentske HTML stranice. U okviru HTML stranice se pojavljuje sledeći tag:

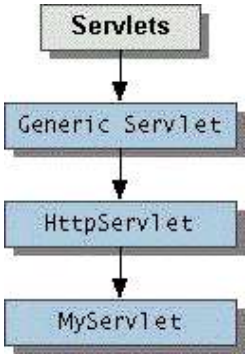
```
<a href="/Proba/ObradaPostMetodaServlet"> Poziv servleta </a>
```

Tada se u okviru servleta generiše HTML kod koji izgleda slično:

```
public class PozivHTMLServlet extends HttpServlet {
        public void doGet (HttpServletRequest request,
                           HttpServletResponse response)
                           throws ServletException, IOException {
                ...
                out.println(... +
        "<a href=\"" +
            response.encodeURL("/Proba/ObradaPostMetodaServlet") +
            "\">Poziv servleta</a>" +
                            ...);
                ...
        }
        ...
}
```

Čitalac je mogao da primeti da servleti koriste paket javax.servlet, koji sadrži interfejse i klase za njihovo pisanje. Svi servleti implementuju interfejs Servlets, ili nasleđuju klasu koja je već implementirala ovaj interface. Najkorišćenija klasa ove vrste je klasa HttpServlet. Kada servlet dobije poziv od strane klijenta, on prihvata dva objekta:

- ServletRequest (zahtev), koji održava komunikaciju od klijenta do servera.
- ServletResponse (odgovor), koji održava komunikaciju od servera ponovo ka klijentu.



Slika 1 - Hijerahija klasa servleta

ServletRequest i ServletResponse su takođe interfejsi definisani u navedenom javax.servlet paketu.

ServletRequest interfejs dozvoljava servletu da priđe:

- informacijama kao što su imena parametara koje šalje klijent, protokol koji koristi klijent, ime udaljenog računara koji je poslao zahtev i koji je server primio, i drugim informacijama;
- objektu klase ServletInputStream, koji pruža servletu da dobije podatke sa ulaznog toka, odnosno koji su od klijenta poslati preko protokola, na primer korišćenjem HTTP POST ili GET metode.

ServletResponse interfejs:

- Dozvoljava servletu da definiše sadržaj odgovora i tip podataka (MIME);
- Obezbeđuje izlazni tok podataka, odnosno objekat klase ServletOutputStream i objekat klase Writer, preko kojih će servlet poslati podatke.

U okviru klase koja predstavlja servlet i koja nasleđuje klasu HttpServlet klasu postoje metode koje mogu obrađivati zahteve klijenta. Da bi se olakšala sama komunikacija sa klijentom postoje dva osnovna argumenta koja sadrže ove metode:

1. HttpServletRequest objekat, koji obuhvata podatke *od* klijenta;
2. HttpServletResponse objekat, koji obuhvata podatke koji se šalju *ka* klijentu.

HttpServletRequest objekat omogućava pristup HTTP podacima u zaglavlju. Takođe, omogućava primanje argumenata koje klijent šalje kao deo svog zahteva.

U zavisnosti od dobijenih podataka i tražene poslovne logike, potrebno je klijentu generisati odgovarajući odgovor. Najjednostavniji način, koji će biti primenjen u sledećim primerima ovog poglavlja je da se dati odgovor generiše u formi HTML stranice. Tada je potrebno obavestiti veb pregledač da se šalju podaci koji su tipa HTML i to na sledeći način:

```
response.setContentType("text/html");
```

Da bi se dobila korektna veb stranica potrebno je primeniti određeni broj puta metod println, tako što se kao argumenti u vidu Stringova pojavljuju HTML tagovi. Primer:

```
import java.io.*;
import java.servlet.*;
import java.servlet.http.*;

public class Zdravo extends HttpServlet {
   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                     throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String docType ="\n";
      out.println(docType +
      "<html>\n" +
      "<head><title>Zdravo</ title ></ head>\n"+
      "<body bgcolor=\"#FDF5E6\">\n" +
      "<h1> Zdravo </h1>\n" +
      "</body></html>");
   }
}
```

U datom primeru je u okviru String promenljive docType definisano zaglavlje HTML stranice i kasnije izvršena konkatenacija sa samim sadržajem, koji je takođe definisan u obliku Stringa. Još jedan sličan primer je :

```
public class PrviServlet extends HttpServlet {
   public void doGet(HttpServletRequest zahtev,
                     HttpServletResponse odgovor)
                     throws ServletException, IOException {
         PrintWriter out;
         String title = "Jednostavan odgovor Servera";
         odgovor.setContentType("text/html");
         out = odgovor.getWriter();
         out.println("<html><head><title>");
         out.println(title);
         out.println("</title></head><body>");
         out.println("<h1>" + title + "</h1>");
         out.println("<p>Ova strana je generisana pomocu Servleta.");
         out.println("</body></html>");
         out.close();
   }
}
```

U datom primeru klasa PrviServlet nasleđuje klasu HttpServlet, koja je implementirala Servlet interfejs. PrviServlet definiše svoj doGet metod. Ovaj metod se poziva kada klijent generiše GET zahtev sa svoje forme i kao rezultat se jednostavna HTML stranica vraća klijentu. Zahtevi od strane klijenta su sadržani u HttpServletRequest objektu, a odgovor

klijentu je u objektu tipa HttpServletResponse. Zato što se u ovom primeru tekstualni podaci vraćaju klijentu, odgovor je poslat pomoću objekta Writer koji je sadržan u okviru HttpServletResponse objekta.

Naglašeno je da HttpServletRequest objekat omogućava pristup podacima koji su sadržani u HTTP header-u zahteva. Ovaj objekat omogućava prihvatanje argumenata koje klijent šalje kao deo svog zahteva. Podacima koje šalje klijent može se pristupiti pomoću getParameter() metoda, koji vraća vrednost imenovanog parametra. Ako parametar ima više od jedne vrednosti, koristi se metod getParameterValues(). Ovaj metod kao rezultat vraća niz vrednosti za imenovani parametar. Postoji i metod getParameterNames() koji vraća imena svih parametera.

Objekat HttpServletResponse daje dve mogućnosti za vraćanje podataka klijentu, upotrebom metoda getWriter(), kada se dobija objekat tipa Writer, ili upotrebom metoda getOutputStream(), kada se dobija objekat tipa ServletOutputStream. Metoda getWriter() se koristi kada su podaci koji se vraćaju klijentu tekstualnog tipa, a metoda getOutputStream() za binarne podatke. Pozivanje metoda close() u okviru dobijenih objekata nakon slanja odgovora omogućava serveru da zna kada je odgovor kompletiran.

Sa strane klijenta zahtevi mogu doći ili metodom GET ili POST. Metoda kojom se šalju podaci je definisana u okviru taga forme na stranici klijenta i njegovog atributa METHOD, kome se definiše jedna od datih vrednosti.

GET zahtev se obrađuje preklapanjem metoda doGet().

```
public class ObradaGetMetodaServlet extends HttpServlet {
   public void doGet(HttpServletRequest zahtev,
                     HttpServletResponse odgovor)
                     throws ServletException, IOException {
         odgovor.setContentType("text/html");
         PrintWriter out = odgovor.getWriter();
         out.println("<html>" +
                     "<head><title>Primer       citanja       vrednosti
                     parametra</title></head>" +
                     ...);
         String servletPar1 = zahtev.getParameter("par1");
         if (servletPar1 != null) {
               out.println("<body>" + servletPar1);
        }
        out.println("</body></html>");
        out.close();
   }
}
```

Unutar metoda doGet(), pomoću metoda getParameter() objekta koji predstavlja HttpServletRequest se dobijaju vrednosti očekivanih parametara. Za generisanje odgovora klijentu doGet() metod koristi objekat Writer iz objekta HttpServletResponse da bi poslao tekstualne podatke klijentu. Pre pristupa ovom objektu definiše se tip sadržaja. Na kraju doGet() metoda, kada je odgovor poslat, Writer objekat se zatvara.

POST zahtev se obrađuje na identičan način preklapanjem metoda doPost().

```
public class ObradaPostMetodaServlet extends HttpServlet {
   public void doPost(HttpServletRequest request,
                      HttpServletResponse response)
                      throws ServletException, IOException {
         response.setContentType("text/html");
         PrintWriter out = response.getWriter();
         out.println("<html>" +
                     "<head><title>Primer       citanja       vrednosti
                     parametra</title></head>" +
                     ...);
         String servletPar1 = request.getParameter("par1");
         if (servletPar1 != null) {
            // prikazivanje procitane vrednosti
            out.println("<body>" + servletPar1);
         }
         out.println("</body></html>");
         out.close();
   }
    ...
}
```

Kao i kod metoda doGet(), unutar metoda doPost(), pomoću metoda getParameter() objekta koji predstavlja HttpServletRequest se dobijaju vrednosti očekivanih parametara. Za generisanje odgovora klijentu doGet() metod koristi Writer iz objekta HttpServletResponse da bi poslao tekstualne podatke klijentu. Pre pristupa ovom objektu definiše se tip sadržaja. Na kraju doPost() metoda, kada je odgovor poslat, Writer objekat se zatvara.

Upotreba metoda doGet() ili metoda doPost() je potpuno identična. Svi ugrađeni objekti i njihovi metodi se mogu koristiti na potpuno identičan način. Kao što je moglo da se primeti iz prethodnih primera jedna od najbitnijih osobina servleta je mogućnost da se priđe parametrima koji su definisani i poslati na strani klijenta.

Najvažniji metodi u okviru objekta HttpServletRequest koji se mogu koristiti su:

- getParameter(“ime") - dobija se URL-dekodirana vrednost prvog elementa koji se zove ime. Rezultat je null, ako ne postoji takav parametar u elementima forme;

- getParameterValues(“ime") - dobija se niz URL-dekodiranih vrednosti za sve elemente koji se zovu ime. U slučaju da se ime pojavljuje samo jednom dobija se niz sa jednim elementom. Rezultat je null ako ne postoji takav parametar u elementima forme;
- getParameterNames() ili getParameterMap() - dobija se Enumeration ili Map objekti od poslatih elemenata. Uobičajeno je da je koristi u procesu debagovanja.

2.4 Životni ciklus servleta

Pored opisanih metoda doGet() i doPost() čijim preklapanjem se definišu odgovori nakon dobijenog zahteva od strane klijenta, postoji još metoda koji su definisani u osnovnoj klasi servleta i definišu načine izvršavanja i upotrebe ovih Javinih klasa. Osnovni metodi su:

- **init() -** Izvršava se jednom kada se servlet prvi put učitava, pa se poziva tačno jednom. Ne postoje prepreke za postojanje konstruktora u samoj servlet klasi u kome će se izvršiti određene inicijalizacije, ali je to upravo zadatak ove metode *Ne* poziva se za svaki zahtev.
- **service() -** Poziva se kod novog thread-a od strane servera za svaki zahtev. Tipičan scenario poziva je već opisan: postavi se tip sadržaja HTTP odgovora, definše se objekat PrintWriter, pomoću koga se šalje dinamički kreiran HTML Ne preporučuje se da se u izvedenoj klasi preklapa ovaj metod.
- **doGet, doPost, doXxx -** Obrađuju GET, POST, ... zahteve. Upravo preklapanje ovih metoda omogućava željeno ponašanje servleta.
- **destroy()** Poziva se kada server briše instancu servleta. Namenjen je zadatku brisanja nepotrebnih podataka neposredno pre uništenja servleta, izvršava se oslobađanje resursa koje je servlet zauzimao, kao što su otvorene datoteke, konekcija sa bazom podataka,... Obično se izvršava prilikom zaustavljanja Web servera *Ne* poziva se posle svakog zahteva.

2.5 Rad sa cookie-ijima

Cookie-iji omogućavaju da se na klijentski računar zapiše određena informacija koja se može iskoristiti prilikom sledećeg pristupa istog klijenta. Na primer servlet generiše jedinstveno ime i vrednost koje dodeli datom klijentu. Klijent vraća isto ime i vrednost kada se ponovo konektuje na isti sajt, ili isti domen u zavisnosti od podešavanja vrednosti cookie-ija. Ovakva mogućnost se može iskoristiti na više načina, pa su tipične upotrebe cookie-ija: mogućnost identifikacije korisnika tokom sesije elektronskog poslovanja(treba napomenuti da sami servleti imaju svoj API na višem nivou za ovaj zadatak i o njemu više reči u narednom odeljku), mogućnost izbegavanja upotrebe korisničkog imena i šifre, prilagođavanje sajta korisniku, ciljane reklame, ...

Sa strane servleta cookie se veoma jednostavna stvara i prosleđuje klijentu. Prvo je potrebno kreirati objekat koji predstavlja cookie. Treba pozivati Cookie konstruktor sa željenim imenom i vrednosti, pri čemu su oba argumenta tipa String:

```
Cookie c = new Cookie("userID", "a1234");
```

Preporuka je i da se postavi maksimalno vreme života cookie-ija, i na taj način se naglasiti čitaču da smesti cookie na disk umesto samo u memoriju. Data operacija se postiže pomoću metoda setMaxAge() i argumenta koji predstavlja broj sekundi.

```
c.setMaxAge(60*60*24*7); // Jedna nedelja
```

Kada se kreira željeni Cookie objekat, tada ga je potrebno smestiti u objekat koji predstavlja HTTP odgovor, koristeći response.addCookie. Tek izvršavanjem ove naredbe, cookie se šalje na stranu klijenta

```
response.addCookie(c);
```

Druga operacija koja se može izvršiti je čitanje podataka koji su upisani na strani klijenta u obliku cookie-ija. Da bi se izvršila ova operacija potrebno je pozvati metod request.getCookies(). Kao rezultat izvršavanja ovog metoda dobija se niz objekata tipa Cookie. Da bi se pronašla određene vrednost potrebno je kretati se po nizu objekata, pozvati metod getName() za svaki element niza sve dok se ne pronađe željeni cookie. Nakon toga pomoću getValue() može se koristiti upisana vrednost.

```
String cookieName = "userID";
Cookie[] cookies = request.getCookies();
if (cookies != null) {
   for(int i=0; i<cookies.length; i++) {
      Cookie cookie = cookies[i];
      if (cookieName.equals(cookie.getName())) {
         doSomethingWith(cookie.getValue());
      }
   }
}
```

Cookie se može iskoristiti i za proveru da li je korisnik već pristupao datoj aplikaciji. Sledi jedan takav primer:

```
public class RepeatVisitor extends HttpServlet {
   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                     throws ServletException, IOException {
         boolean newbie = true;
         Cookie[] cookies = request.getCookies();
         if (cookies != null) {
               for(int i=0; i<cookies.length; i++) {
                     Cookie c = cookies[i];
                     if((c.getName().equals("repeatVisitor"))&&
                        (c.getValue().equals("yes"))) {
                           newbie = false;
```

```
                    break;
                }
            }
        }
        String title;
        if (newbie) {
            Cookie returnVisitorCookie =
                  new Cookie("repeatVisitor", "yes");
            returnVisitorCookie.setMaxAge(60*60*24*365);
            response.addCookie(returnVisitorCookie);
            title = "Dobrodosli prvi put";
        } else {
            title = "Dobrodosli ponovo";
        }
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        … // (Izlaz sa razlicitim naslovom)
```

Pored navedenih, objekat Cookie ima i sledeće metode

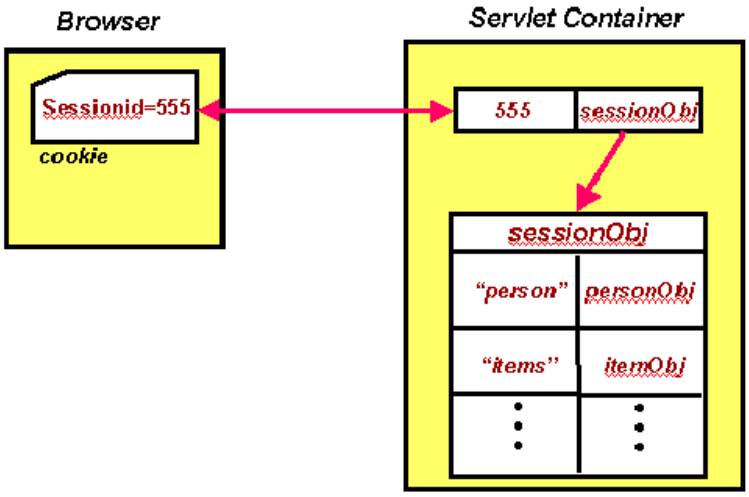
- **getDomain()/setDomain()** - Specificira se domen koji obrađuje cookie. Trenutni host mora biti deo specificiranog domena.
- **getMaxAge()/setMaxAge()** - Čita/postavlja vreme života cookie-ija u sekundama. Ako se ova vrednost ne postavi, podrazumeva se da je vreme života cookie-ija samo trenutna sesija.
- **getName()** - Dobija se ime cookie-ija. Ne postoji setName metod, jer se ime definiše u okviru konstruktora. U okviru niza cookie-ija koji se dobija od strane klijenta, metod getName se koristi da bi se pronašao željeni cookie.
- **getPath()/setPath()** - Čita/postavlja putanju koja obrađuje cookie. Ako se ne navede, cookie pripada URLu koji je u okviru ili direktorijum iznad trenutne stranice.
- **getSecure()/setSecure()** - Čita/postavlja flag koji definiše da li se cookie izvršava samo pomoću SSL konkecije ili bilo koje konekcije.
- **getValue()/setValue()** - Čita/postavlja vrednost koja se želi pamtiti u okviru cookie-ija. Za nove cookie-ije, ova vrednost se postavlja u konstruktoru, a ne pomoću setValue. getValue se koristi da bi se dobila upisana vrednost. Ako se postojećem cookie-iju menja vrednost, potrebno je poslati tu novu vrednost sa response.addCookie.

2.6 Rad sa sesijom

Tokom rada određenog korisnika sa aplikacijom, često je potrebno čuvati određene podatke tokom jednog ili više zahteva. U okviru Java internet aplikacije moguće je organizovati čuvanje podataka na 4 nivoa. Prvi nivo je kada se podaci čuvaju na nivou stranice. Tada ovi podaci nisu deljeni i ne može im se pristupati van date stranice. Drugi nivo je na nivou zahteva, kada su podaci vezani za pojedinačni zahtev korisnika. Tada su podaci vidljivi samo komponentama koje su direktno vezane za dati zahtev. Treći nivo je na nivou sesije. U ovom slučaju podaci su deljeni tokom sesije jednog korisnika i vidjivi su svim stranicama datog

korisnika, bilo da su u pitanju serverske komponente ili klijentske stranice. Četvrti nivo je nivo aplikacije i podaci su tada dostupni svim stranicama svih korisnika koji pripadaju datoj aplikaciji. U ovom odeljku će se detaljnije razmotriti čuvanje podataka na nivou sesije, a o ostalim tehnikama biće reči kasnije.

Sama sesija je bitan pojam i definiše ponašanje korisnika. Simulaciju sesije moguće je izvesti alternativnim metodama, kao što su čuvanje informacija u cookie-iju i provera informacija pri svakom pristupu korisnika. Sesiju je moguće simulirati i pomoću HIDDEN promenljive u okviru forme korisnika u koju se upisuje vrednost koja se kasnije proverava.



Slika 6. Realizacija sesije

Dalje moguće je u okviru pozivnog URL-a sa strane klijenta dodavati određene vrednosti za proveru. Zajedničko ovim pristupima je da zahtevaju dosta programerskog znanja i vremena. Da bi se olakšao ovaj veoma bitan zahtev, razvijen je poseban deo klasa i objekata koji se mogu koristiti i umnogome olakšati ovaj proces. Odlučeno je da se sami objekti sesije svakog klijenta čuvaju na serveru. S druge strane, sesije su automatski povezane sa klijentom pomoću cookie-ija ili promene URLa (slika N).

Programerima je jednostavno pomoću metoda request.getSession() omogućeno da se dobije objekat sesije datog klijenta. U pozadini, sistem pregleda cookie ili URL dodatne informacije i pretražuje da li se ključ poklapa sa nekim od prethodno smeštenih objekata sesije. Ako pronađe poklapanje, kao rezultat vraća pronađeni objekat. Ako ne pronađe, kreira novi, povezuje cookie ili URL informacije sa njegovim ključem i vraća kao rezultat novi objekat sesije. Mehanizmi poput Hashtable dozvoljavaju da se smeštaju neophodni objekti unutar sesije. Koriste se metodi objekta koji predstavlja sesiju i to setAttribute() smešta vrednost u sesiju, odnosno getAttribute() čita datu vrednost iz sesije. Primer korišćenja sesije:

```
HttpSession session = request.getSession();
```

```
SomeClass value = (SomeClass)session.getAttribute("someID");
if (value == null) {
   value = new SomeClass(...);
   session.setAttribute("someID", value);
}
doSomethingWith(value);
```

Treba napomenuti da nije potrebno pozivati ponovo metod setAttribute(), nakon promene vrednosti, ako je promenjena vrednost isti objekat. Ali ako je vrednost nepromenljiva, tada će promenjena vrednost biti novi objekat i mora se ponovo pozvati setAttribute().

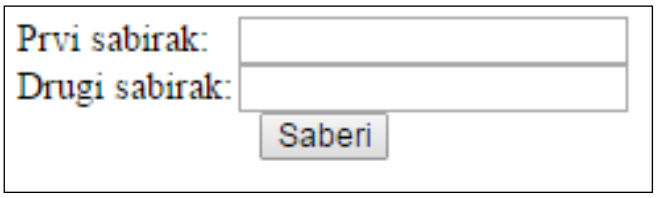
Objekat koji predstavlja sesiju i koji je HttpSession tipa ima sledeće metode:

- **getAttribute()** – Vraća prethodno smeštenu vrednost iz objekta sesije. Rezultat je null ako nijedna vrednost nije povezana sa datim imenom.
- **setAttribute()** – povezuje vrednost sa navedenim imenom. Vrši se nadgledanje promena: vrednosti implementiraju HttpSessionBindingListener.
- **removeAttribute()** – uklanja vrednosti koje su povezane sa navedenim imenom.
- **getAttributeNames()** – vraća imena svih atributa u okviru sesije.
- **getId()** – vraća jedinstveni identifikator sesije.
- **isNew()** – prepoznaje da li je sesija nova za klijenta (ne za *stranicu*)
- **getCreationTime()** – vraća vreme kada je sesija prvi put kreirana
- **getLastAccessedTime()** – vraća kao rezultat vreme kada je sesija poslednji put poslata klijentu
- **getMaxInactiveInterval(), setMaxInactiveInterval()** – pročita ili postavi vremenski interval kada je sesije bez pristupa i postavlja se kao nevalidna
- **invalidate()** – postavi trenutnu sesiju kao nevalidnu

Zadaci iz tehnologije Java Servlet

Zadatak 1

Korišćenjem Java Servlet tehnologije napisati servlet koji vrši sabiranje dva broja. Unos podataka treba izvršiti kroz HTML formu.



REŠENJE

Formu možemo pisati u HTML fajlu, pri čemu će action atribut forme imati vrednost naziva servleta koji pozivamo, bez ekstenzije java. Servlet koji ćemo napraviti naknadno ćemo mapirati u XML fajlu.

```
<!-- web page: index.html -->
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <meta http-equiv="Content-Type" content="text/html;
              charset=UTF-8">
    </head>
    <body>
        <form action="pozoviServlet" method="POST">
            <table>
                <tr>
                    <td>Prvi sabirak:</td>
                    <td><input type="text" name="prvi" value=""/> </td>
                </tr>
                <tr>
                    <td>Drugi sabirak:</td>
                    <td><input type="text" name="drugi" value=""/> </td>
                </tr>
                <tr>
                     <td colspan="2" align="center">
                     <input type="submit" value="saberi" />
                 </td>
                </tr>
            </table>
        </form>
    </body>
</html>
```

Servlet sadrži metodu za obradu HTTP GET i HTTP POST zahteva je processRequest. Ovu metodu pozivaju i doGet i doPost metoda. U metodi processRequest prvo se prihvataju dva podatka sa forme koja je pozvala servlet. Oba podatka su tekstualnog tipa (String) i dohvatamo ih u servletu pozivom metode getParametar nad zahtevom (request). Pošto želimo da sabiramo dva podatka, ova dva tekstualna podatka treba da konvertujemo u cele brojeve. Konverzija u cele brojeve iz stringa radi se pozivom metode parseInt(string). S obzirom da postoji mogućnost da se neuspešno izvrši konverzija, ova konverzija mora da se uradi u try-catch bloku sa obradom izuzetka tipa NumberFormatException.

```
//source file: PrviServlet.java

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class PrviServlet extends HttpServlet {
 //metoda za obradu i HTTP GET i HTTP POST zahteva
    protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
    throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String prvi=request.getParameter("prvi");
        String drugi=request.getParameter("drugi");
        int a=0, b=0;
        boolean ispravno=true;
        //double c=0.0;

        try{
            a=Integer.parseInt(prvi);
            b=Integer.parseInt(drugi);
            //c = Double.parseDouble(prvi);
        }catch(NumberFormatException nfe){ispravno=false;}

     int zbir=a+b;

        try {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Primer2</title>");
            out.println("</head>");
            if(ispravno){
             out.println("<body bgcolor=\"00ffff\">");
             out.println("<h1>Zbir je: " + zbir + "</h1>");
            }
            else{
             out.println("<body bgcolor=\"ff0000\">");
```

```
                out.println("<h1>Neispravni brojevi!</h1>");
            }
            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }

    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    public String getServletInfo() {
        return "Short description";
    }
}
```

Slično kao sabiranje dva cela broja, mogli bi da uradimo i sabiranje dva realna broja, jedina izmena bila bi kod konverzije iz stringa, gde će umesto metode parseInt(string) biti pozvana metoda parseDouble(string).

Sada ćemo prikazati i fajl web.xml. To je XML fajl koji generiše naše okruženje Netbeans prilikom kreiranja servleta. Ovaj fajl se stvara dinamički, ukoliko prilikom pravljenja novog servleta čekiramo opciju “Add information to deployment descriptor (web.xml)”.

```
   //XML file: WEB-INF/web.xml

   <?xml version="1.0" encoding="UTF-8"?>
   <web-app     version="2.5"     xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
        <servlet>
            <servlet-name>imeServleta</servlet-name>
            <servlet-class>PrviServlet</servlet-class>
        </servlet>
        <servlet-mapping>
            <servlet-name>imeServleta</servlet-name>
```

```
            <url-pattern>/pozoviServlet</url-pattern>
        </servlet-mapping>
        <session-config>
            <session-timeout>
                30
            </session-timeout>
        </session-config>
        <welcome-file-list>
            <welcome-file>index.html</welcome-file>
        </welcome-file-list>
    </web-app>
```

Zadatak 2

Korišćenjem Java Servlet tehnologije napisati HTML stranu koja preko linka poziva prvi servlet, koji zatim vrši prosleđivanje na drugi servlet, korišćenjem metode sendRedirect. Na drugom servletu napraviti formu koja ima tekstualno polje za unos imena korisnika i dugme za potvrdu, kojim se poziva treći servlet. Na trećem servletu pročitati podatak (GET parametar), koji je prosleđen tom servletu od strane drugog.

REŠENJE

```
//web page: index.html

<!DOCTYPE html>
<html>
    <head>
        <title>Primer sa prosledjivanjem izmedju servleta</title>
        <meta http-equiv="Content-Type" content="text/html;
              charset=UTF-8">
    </head>
    <body>
        <a  href="prviServlet">Kliknite  na  link  da  pozovete  prvi
servlet!</a>
        <!-- pozivom servleta preko linka poziva se doGet() metod-->
    </body>
</html>


//source file: servleti/PrviServlet.java

package servleti;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
public class PrviServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
    throws ServletException, IOException {
        PrintWriter izlaz = response.getWriter();
        try {
            izlaz.println("<html>");
            izlaz.println("<head>");
            izlaz.println("<title>Servlet prviServlet</title>");
            izlaz.println("</head>");
            izlaz.println("<body>");
            izlaz.println("<h1>Ovo je rezultat prvog!</h1>");

            izlaz.println("</body>");
            izlaz.println("</html>");
        } finally {
        }
        response.sendRedirect("drugiServlet");
    }
}
//source file: servleti/DrugiServlet.java

package servleti;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class DrugiServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter izlaz = response.getWriter();
        try {
            izlaz.println("<html>");
            izlaz.println("<head>");
            izlaz.println("<title>Servlet drugiServlet</title>");
            izlaz.println("</head>");
            izlaz.println("<body>");
            izlaz.println("<h1>Ovo je rezultat drugog servleta,
                          a ne prvog!</h1>");
            izlaz.println("<form                 action=\"treciServlet\"
method=\"get\">");
            izlaz.println("Unesite     ime:     <input     type=\"text\"
name=\"ime\">");
            izlaz.println("<input type=\"submit\" value=\"Pozovi treci
```

```
                        servlet\">");
            izlaz.println("</form>");
            izlaz.println("</body>");
            izlaz.println("</html>");
        } finally {
            izlaz.close();
        }
    }
}
```

```
//source file: servleti/TreciServlet.java

package servleti;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class TreciServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
    throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            String ime=request.getParameter("ime");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet treciServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Dobrodosli " + ime+ "</h1>");
            out.println("<br/>");
            out.println("<a href=\"index.html\">Povratak na
                        prvu stranu!</a>");
            out.println("</body>");
            out.println("</html>");
         } finally {
            out.close();
        }
    }
}
```

```
//XML file: WEB-INF/web.xml
//fajl koji generise okruzenje Netbeans prilikom kreiranja servleta

<?xml version="1.0" encoding="UTF-8"?>
```

```
   <web-app     version="2.5"     xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
       <servlet>
           <servlet-name>prviServlet</servlet-name>
           <servlet-class>servleti.PrviServlet</servlet-class>
       </servlet>
       <servlet>
           <servlet-name>drugiServlet</servlet-name>
           <servlet-class>servleti.DrugiServlet</servlet-class>
       </servlet>
       <servlet>
           <servlet-name>treciServlet</servlet-name>
           <servlet-class>servleti.TreciServlet</servlet-class>
       </servlet>
       <servlet-mapping>
           <servlet-name>prviServlet</servlet-name>
           <url-pattern>/prviServlet</url-pattern>
       </servlet-mapping>
       <servlet-mapping>
           <servlet-name>drugiServlet</servlet-name>
           <url-pattern>/drugiServlet</url-pattern>
       </servlet-mapping>
       <servlet-mapping>
           <servlet-name>treciServlet</servlet-name>
           <url-pattern>/treciServlet</url-pattern>
       </servlet-mapping>
       <session-config>
           <session-timeout>
               30
           </session-timeout>
       </session-config>
       <welcome-file-list>
           <welcome-file>index.html</welcome-file>
       </welcome-file-list>
   </web-app>
```

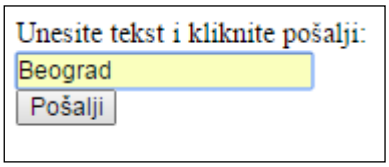
Zadatak 3

Korišćenjem Java Servlet tehnologije napisati HTML stranu koja ima formu sa tekstualnim poljem i dugme za potvrdu. HTML strana treba da pozove prvi servlet, koji će da taj uneti parametar (tekst unet kroz formu) prosleđuje drugom servletu korišćenjem:

a) sendRedirect metode

b) objekta klase RequestDispatcher i forward metode

U prvom servletu osim ovog parametra koji unosi korisnik, treba postaviti još dva parametra, jedan koji se prosleđuje drugom servletu preko zahteva (*request*-a) i jedan koji se prosleđuje drugom servletu preko sesije (odnosno objekta klase *HttpSession*).



REŠENJE

```
//web page: index.html

<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <meta http-equiv="Content-Type" content="text/html;
              charset=UTF-8">
    </head>
    <body>
        <form action="Servlet1" method="get">
            Unesite tekst i kliknite pošalji:<br>
            <input type="text" name="mojparametar"/><br/>
            <input type="submit" value="Pošalji"/>
        </form>
    </body>
</html>


//source file: servleti/Servlet1.java

package servleti;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Servlet1 extends HttpServlet {
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
    throws ServletException, IOException {


        String ime="ETF";
        request.setAttribute("param2", ime);

        HttpSession sesija = request.getSession();
```

```
            sesija.setAttribute("param3", ime);

         //zadatak pod a
            response.sendRedirect("Servlet2");

            //zadatak pod b
            //RequestDispatcher rd=request.getRequestDispatcher("Servlet2");
            //rd.forward(request, response);
        }
    }
```

//source file: servleti/Servlet2.java

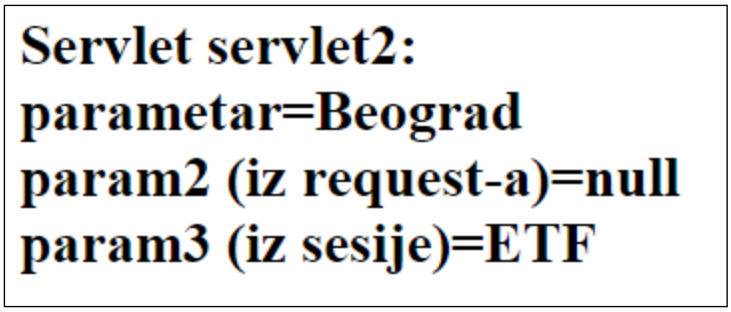
```
    package servleti;

    import java.io.*;
    import javax.servlet.*;
    import javax.servlet.http.*;

    public class Servlet2 extends HttpServlet {

        protected void doGet(HttpServletRequest request,
                             HttpServletResponse response)
        throws ServletException, IOException {
            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();
            try {
                HttpSession sesija = request.getSession();
                String parametar=request.getParameter("mojparametar");
                String param2 = request.getParameter("param2");
                String param3 = (String)sesija.getAttribute("param3");
                out.println("<html>");
                out.println("<head>");
                out.println("<title>Servlet servlet2</title>");
                out.println("</head>");
                out.println("<body>");
                out.println("<h1>Servlet  servlet2:  <br/>  parametar="  +
parametar +
                        "<br/> param2 (iz request-a)=" + param2  +
                        "<br/> param3 (iz sesije)="  + param3 + "</h1>");
                out.println("</body>");
                out.println("</html>");
            } finally {
                out.close();
            }
        }
    }
```

```
    //XML file: WEB-INF/web.xml
    //fajl koji generise okruzenje Netbeans prilikom kreiranja servleta
    <?xml version="1.0" encoding="UTF-8"?>
    <web-app      version="2.5"      xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
        <servlet>
            <servlet-name>Servlet1</servlet-name>
            <servlet-class>servleti.Servlet1</servlet-class>
        </servlet>
        <servlet>
            <servlet-name>Servlet2</servlet-name>
            <servlet-class>servleti.Servlet2</servlet-class>
        </servlet>
        <servlet-mapping>
            <servlet-name>Servlet1</servlet-name>
            <url-pattern>/Servlet1</url-pattern>
        </servlet-mapping>
        <servlet-mapping>
            <servlet-name>Servlet2</servlet-name>
            <url-pattern>/Servlet2</url-pattern>
        </servlet-mapping>
        <session-config>
            <session-timeout>
                30
            </session-timeout>
        </session-config>
        <welcome-file-list>
            <welcome-file>index.html</welcome-file>
        </welcome-file-list>
    </web-app>
```

Rezultat izvršavanja sa sendRedirect metodom:

Servlet servlet2:
parametar=null
param2 (iz request-a)=null
param3 (iz sesije)=ETF

Rezultat izvršavanja sa RequestDispatcher i forward metodom:



Zadatak 4

Korišćenjem Java Servlet tehnologije napisati HTML stranu koja ima formu sa tekstualnim poljem za dodavanje proizvoda u korpu za kupovinu i dugme za potvrdu. HTML strana treba da pozove servlet, koji će da taj uneti parametar (tekst unet kroz formu) ubaciti u niz proizvoda koji su do tada dodati u korpu. Kada se proizvod doda, treba prikazati spisak svih naručenih proizvoda koji se trenutno nalaze u korpi i mogućnost da se korisnik vrati na početnu stranicu i nastavi kupovinu, odnosno dodavanje novih proizvoda u korpu. Trenutno stanje korpe čuvati u sesiji (objektu klase *HttpSession*).

REŠENJE

```
//web page: index.html
<!DOCTYPE html>
<html>
    <head>
      <title>E-korpa</title>
    </head>
    <body bgcolor="#FDF5E6">
        <center>
            <h1>Narudžbenica</h1>
            <form action="prikaz" method="GET">
                Unesite artikal koji zelite da porucite:
                <input type="TEXT" name="noviProizvod"
              value="Sok od breskve"><P>
                <input type="SUBMIT" value="Dodaj u korpu">
            </form>
        </center>
    </body>
</html>
```

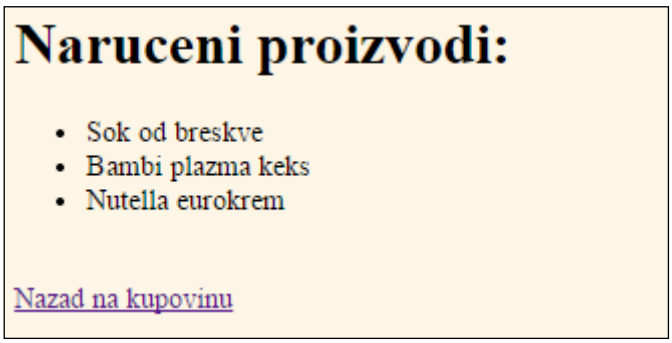
```
//source file: servleti/PrikazKorpe.java
```

```
package servleti;

import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

/** Servlet prikazuje listu narucenih artikala.
 *  Svi artikli u korpi cuvaju se kao ArrayList.
 *  Posto servlet moze vise puta da se pozove,
 *  sadrzaj korpe se prenosi preko sesije.
 */

@SuppressWarnings("unchecked")
public class PrikazKorpe extends HttpServlet {

  public void doGet(HttpServletRequest request,
          HttpServletResponse response)
          throws ServletException, IOException {
    HttpSession sesija = request.getSession();
    ArrayList<String> prethodniProizvodi =
            (ArrayList<String>) sesija.getAttribute("prethodniArtikli");
    if (prethodniProizvodi == null) {
      prethodniProizvodi = new ArrayList<String>();
      sesija.setAttribute("prethodniArtikli", prethodniProizvodi);
    }
    String novi = request.getParameter("noviProizvod");
    if ((novi != null)
            && (!novi.trim().equals(""))) {
      prethodniProizvodi.add(novi);
    }
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String naslov = "Naruceni proizvodi:";
    String docType =
            "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 "
            + "Transitional//EN\">\n";
    out.println(docType
            + "<html>\n"
            + "<head><title>" + naslov + "</title></head>\n"
            + "<body bgcolor=\"#FDF5E6\">\n"
            + "<h1>" + naslov + "</h1>");
    if (prethodniProizvodi.isEmpty()) {
      out.println("<i>Nema nijednog artikla.</i>");
    } else {
      out.println("<ul>");
      for (String stavka : prethodniProizvodi) {
        out.println("  <li>" + stavka);
      }
```

```
            out.println("</ul>");
        }
        out.println("<br/>");
        out.println("<a href=\"e-korpa.html\"> Nazad na kupovinu </a>");
        out.println("</body></html>");
      }
    }
```

```
//XML file: WEB-INF/web.xml

<?xml version="1.0" encoding="UTF-8"?>
<web-app     version="2.5"     xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
        <session-config>
            <session-timeout>
                30
            </session-timeout>
        </session-config>
        <servlet>
            <servlet-name>prikaz</servlet-name>
            <servlet-class>servleti.PrikazKorpe</servlet-class>
        </servlet>
        <servlet-mapping>
            <servlet-name>prikaz</servlet-name>
            <url-pattern>/prikaz</url-pattern>
        </servlet-mapping>
        <welcome-file-list>
            <welcome-file>e-korpa.html</welcome-file>
        </welcome-file-list>
    </web-app>
```

Prikaz korpe sa naručenim proizvodima:



Zadatak 5

Korišćenjem Java Servlet tehnologije napraviti HTML formu koja služi za pretraživanje pojmova korišćenjem različitih internet pretraživača. Korisnik treba da ima mogućnost da u formi unese u tekstualno polje pojam koji želi da pronađe i da u listi ponuđenih internet pretraživača (eng. search engines) odabere od kog pretraživača želi da dobije pojam. Lista internet pretraživača treba da bude prikazana kao lista radio dugmadi, po jedan za svaki pretraživač (Google, Bing i Yahoo).

REŠENJE

Servlet *FormaPretrazivanja* treba da napravi HTML formu koja služi za unos parametara pretrage - pojma koji pretražujemo i izbor pretraživača. Pojam koji pretražujemo treba da se unese u tekstualno polje, koje će biti prosleđeno upitu pretrage, a upit će biti formiran u zavisnosti koji pretraživač smo odabrali. Izbor pretraživača izvršićemo odabirom jednog radio dugmeta, a nazivi pretraživača dobijaju se iz klase *VrstePretrazivaca*.

```
//source file: FormaPretrazivanja.java

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class FormaPretrazivanja extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String naslov = "Internet pretrazivanje";
    String URLadresa = "odgovor";
    String docType =
      "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
```

```
      "Transitional//EN\">\n";
    out.println
      (docType +
       "<HTML>\n" +
       "<HEAD><TITLE>" + naslov + "</TITLE></HEAD>\n" +
       "<BODY BGCOLOR=\"#FDF5E6\">\n" +
       "<CENTER>\n" +
       "<H1>" + naslov + "</H1>\n" +
       "<FORM ACTION=\"" + URLadresa + "\">\n" +
       "  Pojam za pretragu: \n" +
       "  <INPUT TYPE=\"TEXT\" NAME=\"pojamzapretragu\"><P>\n");

 Pretrazivac[] vrste = VrstePretrazivaca.getPretrazivaci();
    for(int i=0; i<vrste.length; i++) {
      String naziv = vrste[i].getIme();
      out.println("<INPUT TYPE=\"RADIO\" " +
                  "NAME=\"tipPretrazivaca\" " +
                  "VALUE=\"" + naziv + "\">\n");
      out.println(naziv + "<BR>\n");
    }
    out.println
      ("<BR>  <INPUT TYPE=\"SUBMIT\" VALUE=\"PRONADJI\">\n" +
       "</FORM>\n" +
       "</CENTER></BODY></HTML>");
  }
}
```

Klasa *Pretrazivac* služi da poveže ime pretraživača sa njegovom URL adresom. Metoda *napraviURL* treba da izgradi rezultujuću stranicu za svaki pretraživač posebno, tako što na osnovnu URL adresu pretraživača i sufiksa za ključ koji taj pretraživač koristi u svom upitu (http://...?nekiKljuc=") doda pojam koji se traži u URL-enkodovanom formatu (internet+programiranje).

```
//source file: Pretrazivac.java

public class Pretrazivac {
  private String ime, URLadresa;

  public Pretrazivac(String ime,
                     String URLadresa) {
    this.ime = ime;
    this.URLadresa = URLadresa;
  }
```

```
  public String napraviURL(String pojam) {
    return(URLadresa + pojam);
  }

  public String getIme() {
    return(ime);
  }
}
```

Klasa *VrstePretrazivaca* ima statički niz pretraživača, koji sadrži njihove adrese (poziva se konstruktor prethodne klase *Pretrazivac* koji pravi po jedan objekat za svaki internet pretraživač).

```
//source file: VrstePretrazivaca.java

public class VrstePretrazivaca {
  private static Pretrazivac[] pretrazivaci =
    { new Pretrazivac("Google",
                      "http://www.google.com/#q="),
      new Pretrazivac("Bing",
                      "http://www.bing.com/search?q="),
      new Pretrazivac("Yahoo",
                      "http://search.yahoo.com/bin/search?p=")
    };

  public static Pretrazivac[] getPretrazivaci() {
    return(pretrazivaci);
  }

  public static String napraviURL(String searchEngineName,
                                  String searchString) {
    Pretrazivac[] searchSpecs = getPretrazivaci();
    String URLadresa = null;
    for(Pretrazivac spec : searchSpecs) {
      if (spec.getIme().equalsIgnoreCase(searchEngineName)) {
        URLadresa = spec.napraviURL(searchString);
        break;
      }
    }
    return(URLadresa);
  }
}
```

Servlet *OdgovorPretrazivaca* dobija se kao rezultat izvršavanja forme u servletu *FormaPretrazivanja*. Cilj ovog servleta je da prihvati pojam za pretragu i vrstu pretraživača iz forme koju popunjava korisnik i da pošalje upit internet pretraživaču koji je odabran u toj formi. Ako je pretraživač izabran u listi radio dugmadi, šalje se odgovor

302 korišćenjem *sendRedirect* metode, a ako pretraživač nija izabran, šalje se odgovor 404 korišćenjem *sendError*.

URL enkoder menja blanko (space) znak sa znakom + i sve druge nealfabetske karaktere konvertuje u "%XY" gde je XY heksadecimalna vrednost ASCII (ili ISO Latin-1) karaktera. Veb pregledači uvek enkoduju vrednosti forme, ali *getParameter* metoda ih automatski dekoduje.

```
//source file: OdgovorPretrazivaca.java

import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class OdgovorPretrazivaca extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {

    String kljuc = request.getParameter("pojamzapretragu");
     if (kljuc == null || kljuc.isEmpty()) {
       reportProblem(response,  "Niste  uneli  pojam  koji  zelite  da
pronadjete.");
       return;
     }

     kljuc = URLEncoder.encode(kljuc, "UTF-8");
     String imepretrazivaca = request.getParameter("tipPretrazivaca");
     if (imepretrazivaca == null || imepretrazivaca.isEmpty()) {
       reportProblem(response, "Niste uneli pretrazivac koji koristite.");
       return;
     }
     String  pretragaURL  =  VrstePretrazivaca.napraviURL(imepretrazivaca,
kljuc);
     if (pretragaURL != null) {
       response.sendRedirect(pretragaURL);
     } else {
       reportProblem(response, "Nije prepoznat pretrazivac.");
     }
  }

  private  void  reportProblem(HttpServletResponse  response,  String
message)
       throws IOException {
     response.sendError(response.SC_NOT_FOUND, message);
  }
```

```
    }
    //XML file: WEB-INF/web.xml

    <?xml version="1.0" encoding="UTF-8"?>
    <web-app     version="2.5"     xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
        <servlet>
            <servlet-name>odgovor</servlet-name>
            <servlet-class>OdgovorPretrazivaca</servlet-class>
        </servlet>
        <servlet-mapping>
            <servlet-name>odgovor</servlet-name>
            <url-pattern>/odgovor</url-pattern>
        </servlet-mapping>
        <servlet>
            <servlet-name>index</servlet-name>
            <servlet-class>FormaPretrazivanja</servlet-class>
        </servlet>
        <servlet-mapping>
            <servlet-name>index</servlet-name>
            <url-pattern>/index</url-pattern>
        </servlet-mapping>
        <session-config>
            <session-timeout>
                30
            </session-timeout>
        </session-config>
        <welcome-file-list>
            <welcome-file>index</welcome-file>
        </welcome-file-list>
</web-app>
```

Zadatak 6

Korišćenjem Java Servlet tehnologije napraviti servlet koji čita početno stanje brojača iz tekstualnog fajla brojacposeta.txt i inkrementira taj brojač. Servletu treba omogućiti da se za svaki ponovni poziv tog servleta uvećava taj brojač u tekstualnom fajlu. Koristiti metode *init* za čitanje početne vrednosti brojača i *destroy* za ažuriranje vrednosti brojača u fajlu. Ukoliko tekstualni fajl inicijalno ne postoji na lokaciji sa koje čitamo, treba ga kreirati, a brojač postaviti na početnu vrednost 0.

REŠENJE

```
//web page: index.html
```

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <title>Brojac</title>
    </head>
    <body>
        <h2>Kliknite na servlet!</h2>
        <br/>
        <a href="brojacservlet">click</a>
    </body>
</html>
```

//source file: paket/MyServlet.java

```
package paket;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyServlet extends HttpServlet {

  int count;

  @Override
  public void init(ServletConfig config) throws ServletException {
    // Prvo pozivamo init metod bazne klase
    super.init(config);

    // Trazimo fajl u kome stoji ranija vrednost brojaca
    try {
      File file = new File("c:/brojacposeta.txt");
      FileReader fileReader = new FileReader(file);
      BufferedReader bufferedReader = new BufferedReader(fileReader);
      String pocetno = bufferedReader.readLine();
      count = Integer.parseInt(pocetno);
      return;
    }
    catch (FileNotFoundException e1) { }  // Ako stanje nije sacuvano
    catch (IOException e2) { }              // Ako ima problem prilikom
citanja
    catch (NumberFormatException e3) { }   // Ako sacuvano stanje nije
broj

    // Nismo nasli nigde zapisano inicijalno stanje brojaca, pa ga
resetujemo
```

```
        count = 0;
      }

      @Override
      public void doGet(HttpServletRequest req, HttpServletResponse res)
                                    throws ServletException, IOException {

            res.setContentType("text/html;charset=UTF-8");
            PrintWriter out = res.getWriter();
            count++;
            try {
                out.println("<html>");
                out.println("<head>");
                out.println("<title>Brojac poziva</title>");
                out.println("</head>");
                out.println("<body>");
                out.println("Od kad je sveta i veka, ovaj servlet je pozvan
" +
                    count + " puta.");
                out.println("<a href=\"brojacservlet\">klikni ponovo</a>");

                out.println("</body>");
                out.println("</html>");
            }
            finally{
                out.close();
            }
      }

      @Override
      public void destroy() {
        saveState();
      }

      public void saveState() {
        // Pokusavamo da zapisemo stanje brojaca na perzistentnu memoriju
(fajl)
        try {
          File file = new File("c:/brojacposeta.txt");
          FileWriter fileWriter = new FileWriter(file);
          String stanje = Integer.toString(count);
          fileWriter.write(stanje, 0, stanje.length());
          fileWriter.close();
          return;
        }
        catch (IOException e) {  // ako postoji problem prilikom upisa
        }
      }
```

```
    }

    //XML file: WEB-INF/web.xml

    <?xml version="1.0" encoding="UTF-8"?>
    <web-app     version="2.5"     xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
        <servlet>
            <servlet-name>moj_servlet</servlet-name>
            <servlet-class>paket.MyServlet</servlet-class>
        </servlet>
        <servlet-mapping>
            <servlet-name>moj_servlet</servlet-name>
            <url-pattern>/brojacservlet</url-pattern>
        </servlet-mapping>
        <session-config>
            <session-timeout>
                30
            </session-timeout>
        </session-config>
        <welcome-file-list>
            <welcome-file>index.html</welcome-file>
        </welcome-file-list>
</web-app>
```

Zadatak 7

Korišćenjem Java Servlet tehnologije napraviti servlet koji korisniku prikazuje 7 slučajno odabranih brojeva na igri loto, koristeći metodu *Math.random()*. Brojeve odabrati prilikom inicijalizacije servleta (metoda *init()*). Brojevi neka budu u opsegu od 1 do 37.

REŠENJE

```
    //source file: Loto.java

    import java.io.*;
    import javax.servlet.*;
    import javax.servlet.http.*;

    /** Primer koriscenja init i
     *  getLastModified metoda kod servleta.
     */

    public class Loto extends HttpServlet {
      private long modTime;
```

```
private int[] brojevi = new int[7];

/** Ova init metoda se poziva samo kada se servlet
 *  prvi put poziva, pre obrade prvog request-a.
 */



public void init() throws ServletException {
  // zaokruzuje na pribliznu vrednost
  modTime = System.currentTimeMillis()/1000*1000;
  for(int i=0; i<brojevi.length; i++) {
    brojevi[i] = randomNum();
  }
}

/** Metoda doGet vraca listu koji je izracunala metoda init. */

public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
  response.setContentType("text/html");
  PrintWriter out = response.getWriter();
  String naslov = "Tvoja loto kombinacija";
  String docType =
    "<!DOCTYPE HTML>\n";
  out.println(docType +
              "<HTML>\n" +
              "<HEAD><TITLE>" + naslov + "</TITLE></HEAD>\n" +
              "<BODY BGCOLOR=\"yellow\">\n" +
              "<H1 ALIGN=CENTER>" + naslov + "</H1>\n" +
              "<B>Izracunali smo slucajnu kombinaciju " +
              brojevi.length +" loto brojeva najboljih za Vas: " +
              ".</B> <OL>");

  for(int i=0; i<brojevi.length; i++) {
    out.println("  <LI>" + brojevi[i]);
  }
  out.println("</OL>" +
              "</BODY></HTML>");
}

/** Standarda servisna metoda koja uporedjuje ovaj datum
 *  sa datumom u If-Modified-Since u headeru requesta.
 *  Ako je getLastModified datum kasniji ili ako ne postoji
 *  polje If-Modified-Since u headeru, doGet metoda se poziva
 *  uobicajno. Ako getLastModified datum predstavlja isti
 *  ili raniji, servisna metoda ce kao odgovor
 *  poslati 304 (Not Modified) i nece pozvati doGet metodu.
```

```
   *  Pretrazivac ce u tom slucaju iskoristiti kesiranu
   *  verziju te stranice (tj tog servleta).
   */

  public long getLastModified(HttpServletRequest request) {
    return(modTime);
  }

  // Slucaj ceo broj izmedju 1 i 37
  // random metoda vraca broj izmedju 0 i 1

  private int randomNum() {
    return((int)(1 + Math.random() * 37));
  }
}
```

Zadatak 8

Korišćenjem Java Servlet tehnologije napraviti servlet koji proverava da li u internet pregledaču postoji kolačić (*cookie*) koji pokazuje da li je korisnik već posećivao datu veb stranicu, pa ako ne postoji da ispiše „Dobrodošli!“, a ako kolačić postoji, odnosno korisnik je već posećivao datu veb stranicu, da se ispiše „Lepo je videti Vas opet“.

REŠENJE

```
//source file: cookie/MyCookie.java

package cookie;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyCookie extends HttpServlet {

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
         throws ServletException, IOException {

    boolean newuser = true;
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
      for (int i = 0; i < cookies.length; i++) {
        Cookie c = cookies[i];
        if ((c.getName().equals("stariPosetilac"))
         && (c.getValue().equals("yes"))) {
          newuser = false;
```

```
            break;
          }
        }
      }
      String title;
      if (newuser) {
        Cookie posetilacCookie =
                new Cookie("stariPosetilac", "yes");
//              posetilacCookie.setMaxAge(60 * 60 * 24 * 365);
        posetilacCookie.setMaxAge(60);
        response.addCookie(posetilacCookie);
        title = "Dobrodosli!";
      } else {
        title = "Lepo je videti Vas opet";
      }

      response.setContentType("text/html;charset=UTF-8");
      PrintWriter izlaz = response.getWriter();

      try {
        izlaz.println("<html>");
        izlaz.println("<head>");
        izlaz.println("<title>" + title + "</title>");
        izlaz.println("</head>");
        izlaz.println("<body>");
        izlaz.println("<h1>" + title + "</h1>");
        izlaz.println("</body>");
        izlaz.println("</html>");
      } finally {
        izlaz.close();
      }
    }
}
```

Zadatak 9

Korišćenjem Java Servlet tehnologije napraviti servlet koji prikazuje HTML stranu sa formom za logovanje korisnika, u koju korisnik treba da unese svoje korisničko ime i svoju lozinku (kredencijali), i ukoliko su korisnički kredencijali ispravni (postoje u bazi podataka u tabeli *KorisnikInfo*) korisniku prikazati stranicu sa svojim ličnim podacima (servlet *Prikaz*), a ukoliko podaci nisu dobri korisniku prikazati stranicu sa greškom (servlet *Greska*). Takođe, potrebno je realizovati još tri servleta: servlete za promenu ličnih podataka (servlet *Promena*, koji prikazuje formu za izmenu korisničkih podataka i koji poziva za ažuriranje tabele u bazi drugi servlet *Izmena*) i servlet koji služi prekidanje sesije i izlogovanje iz sistema (servlet *Logout*).

Izgled tabela u bazi *dbkorisnik*:

KorisnikInfo		
#	Naziv kolone	Tip
1	username (PK)	varchar(32)
2	password	varchar(32)
3	first_name	varchar(32)
4	last_name	varchar(32)
5	email	varchar(32)
6	phone	varchar(32)

REŠENJE

//source file: util/DB.java

```
package util;

import java.sql.Connection;
import java.sql.DriverManager;

public class DB {

    private static DB instance;
    private static final int MAX_CON = 5;
    private static final Connection[] bafer = new Connection[MAX_CON];
    private int first = 0, last = 0, free = MAX_CON;

    private DB() { //za MySQL
        try {
            Class.forName("com.mysql.jdbc.Driver");
            for (int i = 0; i < MAX_CON; i++) {
                bafer[i] =
                     DriverManager.getConnection("jdbc:mysql://
                     localhost:3306/dbkorisnik", "root", "");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    /* za Access
     private DB(){
     try{
```

```
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            for(int i=0; i<MAX_CON; i++)
            bafer[i]                                                    =
   DriverManager.getConnection("jdbc:odbc:dbkorisnik");
     }catch(Exception e){}
     }
     */
    public static DB getInstance() {
        if (instance == null) {
            instance = new DB();
        }
        return instance;
    }

    public synchronized Connection getConnection() {
        if (free == 0) {
            return null;
        }
        free--;
        Connection con = bafer[first];
        first = (first + 1) % MAX_CON;
        return con;
    }


    public synchronized void putConnection(Connection con) {
        if (con == null) {
            return;
        }
        free++;
        bafer[last] = con;
        last = (last + 1) % MAX_CON;
    }
}
```

```
//source file: beans/Korisnik.java
//Java bean

package beans;

public class Korisnik {

    private String username = "";
    private String password = "";
    private String ime = "";
    private String prezime = "";
    private String email = "";
    private String telefon = "";
```

```
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    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 String getEmail() {
        return email;
    }

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

    public String getTelefon() {
        return telefon;
    }

    public void setTelefon(String telefon) {
        this.telefon = telefon;
    }
}
```

```
//source file: servleti/Index.java

package servleti;

import beans.Korisnik;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Index extends HttpServlet {

    protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        HttpSession sesija = request.getSession();
        Korisnik korisnik = (Korisnik) sesija.getAttribute("korisnik");
        String username = "", password = "";
        if (korisnik != null) {
            username = korisnik.getUsername();
            password = korisnik.getPassword();
        }
        String poruka = (String) request.getAttribute("poruka");
        if (poruka == null) {
            poruka = "Dobrodosli!";
        }
        try {
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Korisnicka aplikacija</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h3>" + poruka + " </h3><br/><br/>");
            out.println("Molimo, ulogujte se: ");
            out.println("<table><tr><td>Korisnicko ime:</td>");
            out.println("<form action=\"login\" method=\"POST\">");
            out.println("<td><input    type=\"text\"    name=\"username\"
value=\""
                    + username + "\" width=\"20\"></td></tr>");
            out.println("<tr><td>Lozinka:</td>");
            out.println("<td><input type=\"password\" name=\"password\"
                    value=\""       +       password       +       "\"
               width=\"20\"></td></tr>");
            out.println("<tr><td><input type=\"submit\"
                     value=\"Ulogujte se\"</td>");
            out.println("<td><input type=\"reset\"
                    value=\"Ponistite\"</td><tr></table>");
```

```
            out.println("</form>");
            out.println("</body>");
            out.println("</html>");
        } finally {
            out.close();
        }
    }

    @Override
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

//source file: servleti/Login.java

```
package servleti;

import beans.*;
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import util.DB;

public class Login extends HttpServlet {

  protected void processRequest(HttpServletRequest request,
                                HttpServletResponse response)
          throws ServletException, IOException {
    HttpSession sesija = request.getSession();
    String poruka = null;
    String username = request.getParameter("username");
    String password = request.getParameter("password");

    Korisnik korisnik = new Korisnik();
    korisnik.setUsername(username);
    korisnik.setPassword(password);

    sesija.setAttribute("korisnik", korisnik);
```

```
        if (username.isEmpty() || password.isEmpty()) {
          poruka = "Niste popunili sva polja!";
          request.setAttribute("poruka", poruka);
          RequestDispatcher rd = request.getRequestDispatcher("index");
          rd.forward(request, response);
        }

        Connection con = null;
        Statement stmt = null;
        ResultSet rs = null;
        String address = "prikaz";
        try {
          con = DB.getInstance().getConnection();
          stmt = con.createStatement();
          String upit = "select * from KorisnikInfo where username='" +
username +
                        "' and password='" + password + "';";
          rs = stmt.executeQuery(upit);
          if (rs.next()) {
            String email = rs.getString("email");
            String ime = rs.getString("first_name");
            String prezime = rs.getString("last_name");
            String telefon = rs.getString("phone");
            korisnik.setEmail(email);
            korisnik.setIme(ime);
            korisnik.setPrezime(prezime);
            korisnik.setTelefon(telefon);
            stmt.close();
          } else {
            poruka = "Neispravno korisnicko ime i/ili lozinka! Pokusajte
ponovo.";
            request.setAttribute("poruka", poruka);
            korisnik.setPassword("");
            address = "index";
          }
        } catch (SQLException ex) {
          sesija.invalidate();
          String errormsg = ex.getMessage();
          request.setAttribute("errormsg", errormsg);
          address = "error";
        } finally {
          DB.getInstance().putConnection(con);
        }

        request.getRequestDispatcher(address).forward(request, response);
      }

      @Override
      protected void doGet(HttpServletRequest request,
```

```
                        HttpServletResponse response)
            throws ServletException, IOException {
      processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request,
                        HttpServletResponse response)
            throws ServletException, IOException {
      processRequest(request, response);
    }
}
```

//source file: servleti/Prikaz.java

```
package servleti;

import beans.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Prikaz extends HttpServlet {
  protected void processRequest(HttpServletRequest request,
                                HttpServletResponse response)
          throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    HttpSession sesija = request.getSession();
    Korisnik korisnik = (Korisnik) sesija.getAttribute("korisnik");
    String poruka = (String) request.getAttribute("poruka");

    if (poruka == null) {
      poruka = "Vasi podaci (prethodna stranica je \"promena\")";
    }

    try {
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Korisnicka aplikacija</title>");
      out.println("</head>");
      out.println("<body>");
      out.println("<h3>" + poruka + "</h3><br/><br/>");
      out.println("<table width=\"40%\" border=\"3\">"
              + "<tr><td width=\"50%\">Korisnicko ime:</td>"
              + "<td>" + korisnik.getUsername() + "</td></tr>");
      out.println("<tr><td>Ime:</td>"
              + "<td>" + korisnik.getIme() + "</td></tr>");
      out.println("<tr><td>Prezime:</td>"
              + "<td>" + korisnik.getPrezime() + "</td></tr>");
```

```
      out.println("<tr><td>Email:</td>"
              + "<td>" + korisnik.getEmail() + "</td></tr>");
      out.println("<tr><td>Telefon:</td>"
              + "<td>" + korisnik.getTelefon() + "</td></tr>");
      out.println("</table><br/>");
      out.println("<a href=\"promena\">Promenite podatke</a>");
      out.println("<a href=\"logout\">Izlogujte se</a>");
      out.println("</body>");
      out.println("</html>");
    } finally {
      out.close();
    }
  }

  @Override
  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
          throws ServletException, IOException {
    processRequest(request, response);
  }

  @Override
  protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
          throws ServletException, IOException {
    processRequest(request, response);
  }
}
```

//source file: servleti/Greska.java

```
package servleti;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Greska extends HttpServlet {

  protected void processRequest(HttpServletRequest request,
                                HttpServletResponse response)
          throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    String errormsg = (String) request.getAttribute("errormsg");
    PrintWriter out = response.getWriter();
    try {
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Greska!</title>");
```

```
      out.println("</head>");
      out.println("<body>");
      out.println("<h1>Greska u radu sa bazom podataka</h1>");
      out.println("<br/>" + errormsg + "<br/>");
      out.println("</body>");
      out.println("</html>");
    } finally {
      out.close();
    }
  }

  @Override
  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
          throws ServletException, IOException {
    processRequest(request, response);
  }

  @Override
  protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
          throws ServletException, IOException {
    processRequest(request, response);
  }
}
```

//source file: servleti/Promena.java

```
package servleti;

import beans.Korisnik;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Promena extends HttpServlet {

  protected void processRequest(HttpServletRequest request,
                                HttpServletResponse response)
          throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    HttpSession sesija = request.getSession();
    Korisnik korisnik = (Korisnik) sesija.getAttribute("korisnik");

    try {
      out.println("<html>");
```

```
            out.println("<head>");
            out.println("<title>Korisnicka aplikacija</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h3>Vasi podaci koje mozete menjati</h3><br/><br/>");
            out.println("<form action=\"izmena\" method=\"post\" >");
            out.println("<table width=\"40%\" border=\"3\">"
                    + "<tr><td width=\"50%\">Ime:</td>"
                    + "<td><input type=\"text\" name=\"ime\" value=\""
                    + korisnik.getIme() + "\"/></td></tr>");
            out.println("<tr><td>Prezime:</td>"
                    + "<td><input type=\"text\" name=\"prezime\" value=\""
                    + korisnik.getPrezime() + "\"/></td></tr>");

            out.println("<tr><td>Email:</td>"
                    + "<td><input type=\"text\" name=\"email\" value=\""
                    + korisnik.getEmail() + "\"/></td></tr>");
            out.println("<tr><td>Telefon:</td>"
                    + "<td><input type=\"text\" name=\"telefon\" value=\""
                 + korisnik.getTelefon() + "\"/></td></tr>");
            out.println("</table><br/>");
            out.println("<input       type=\"submit\"       value=\"Prihvatite
izmenu\"/>");
            out.println("<input type=\"reset\" value=\"Odbacite izmenu\"/>");

            out.println("</form></body>");
            out.println("<a href=\"prikaz\">Nazad</a>");
            out.println("</html>");
          } finally {
            out.close();
          }
        }

        @Override
        protected void doGet(HttpServletRequest request,
                             HttpServletResponse response)
                throws ServletException, IOException {
          processRequest(request, response);
        }

        @Override
        protected void doPost(HttpServletRequest request,
                              HttpServletResponse response)
                throws ServletException, IOException {
          processRequest(request, response);
        }
    }
```

//source file: servleti/Izmena.java

```
package servleti;

import beans.Korisnik;
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
import util.DB;

public class Izmena extends HttpServlet {
  protected void processRequest(HttpServletRequest request,
                                HttpServletResponse response)
          throws ServletException, IOException {
    HttpSession sesija = request.getSession();
    Korisnik korisnik = (Korisnik) sesija.getAttribute("korisnik");
    String ime = request.getParameter("ime");
    String prezime = request.getParameter("prezime");
    String telefon = request.getParameter("telefon");
    String email = request.getParameter("email");

    korisnik.setIme(ime);
    korisnik.setPrezime(prezime);
    korisnik.setTelefon(telefon);
    korisnik.setEmail(email);
    String upit = "update KorisnikInfo "
            + "set first_name='" + ime + "',last_name='" + prezime
         + "', phone='" + telefon + "', email='" + email + "'"
            + "where username='" + korisnik.getUsername() + "';";

    Connection con = null;
    Statement stmt = null;
    String address = "prikaz";
    try {
      con = DB.getInstance().getConnection();
      stmt = con.createStatement();
      stmt.executeUpdate(upit);
      stmt.close();
    } catch (SQLException ex) {
      sesija.invalidate();
      String errormsg = ex.getMessage();
      request.setAttribute("errormsg", errormsg);
      address = "error";
    } finally {
      DB.getInstance().putConnection(con);
    }
    request.setAttribute("poruka", "Podaci su uspesno izmenjeni");
    request.getRequestDispatcher(address).forward(request, response);
  }
```

```
  @Override
  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
          throws ServletException, IOException {
    processRequest(request, response);
  }

  @Override
  protected void doPost(HttpServletRequest request,
                      HttpServletResponse response)
          throws ServletException, IOException {
    processRequest(request, response);
  }
}
```

//source file: servleti/Logout.java

```
package servleti;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Logout extends HttpServlet {

  protected void processRequest(HttpServletRequest request,
                                HttpServletResponse response)
          throws ServletException, IOException {
    HttpSession sesija = request.getSession();
    sesija.invalidate();
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Dovidjenja</title>");
      out.println("</head>");
      out.println("<body>");
      out.println("<h1>Dovidjenja</h1>");
      out.println("<a href=\"index\">nazad na login stranu</a>");
      out.println("</body>");
      out.println("</html>");
    } finally {
      out.close();
    }
  }

  @Override
```

```
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
            throws ServletException, IOException {
      processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
            throws ServletException, IOException {
      processRequest(request, response);
    }
  }
```

//XML file: WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app     version="2.5"     xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <servlet>
        <servlet-name>index</servlet-name>
        <servlet-class>servleti.Index</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>login</servlet-name>
        <servlet-class>servleti.Login</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>greska</servlet-name>
        <servlet-class>servleti.Greska</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>promena</servlet-name>
        <servlet-class>servleti.Promena</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>izmena</servlet-name>
        <servlet-class>servleti.Izmena</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>prikaz</servlet-name>
        <servlet-class>servleti.Prikaz</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>logout</servlet-name>
```

```
        <servlet-class>servleti.Logout</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>index</servlet-name>
        <url-pattern>/index</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>login</servlet-name>
        <url-pattern>/login</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>greska</servlet-name>
        <url-pattern>/error</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>promena</servlet-name>
        <url-pattern>/promena</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>izmena</servlet-name>
        <url-pattern>/izmena</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>prikaz</servlet-name>
        <url-pattern>/prikaz</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>logout</servlet-name>
        <url-pattern>/logout</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index</welcome-file>
    </welcome-file-list>
</web-app>
```

Zadatak 10

Napraviti HTML stranu sa formom za logovanje korisnika, u koju korisnik treba da unese svoje korisničko ime i svoju lozinku (kredencijali), a zatim pre pozivanja servleta treba da se izvrši validacija na klijentskoj strain, uz pomoć JavaScript tehnologije. JavaScript kod treba da proveri da li korisničko ime ima najmanje 8 karaktera (pri čemu korisničko ime ne sme da počinje cifrom) i da li lozinka ima najmanje 8 karaktera, od kojih mora da bude bar jedno veliko slovo, jedno malo slovo, jedna cifra i jedan specijalni karakter iz sledećeg skupa: .

U slučaju da kredencijali ne ispunjavaju navedene zahteve, korisnik aplikacije treba da ostane na formi za logovanje, uz ispisivanje greške u prozoru sa upozorenjima (JavaScript alert), a u slučaju da kredencijali ispunjavaju navedene zahteve, potrebno je podatke iz forme proslediti servletu.

REŠENJE

```
//web page: index.html

<!DOCTYPE html>
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
                                       initial-scale=1.0">
        <script>
            function provera() {
                uzorakUser = /^[A-Za-z]\w{7,}$/
                uzorakPass = /^(?=.*?[A-Z])(?=.*?[a-z])
                              (?=.*?[0-9])(?=.*?[#?!@$%^&*-]).{8,}$/
                if (!uzorakUser.test(document.imeForme.user.value)) {
                    alert("Korisničko ime nema 8 karaktera!");
                } else
                    if(!uzorakPass.test(document.imeForme.pass.value)) {
                           alert("Lozinka ne ispunjava uslove - najmanje
                           8 karaktera od toga bar:
                           \njedno     veliko     slovo,\njedno     malo
                           slovo,\njedna     cifra,\njedan     specijalni
                           karakter #?!@$%^&*-");
                } else {
                    document.imeForme.submit();
                    //Drugi nacin, prema ID:
                    //document.getElementById("idForme").submit();
                }
            }
        </script>
    </head>
```

```
    <body>
        <div>Forma</div>
        <div>
            <form name="imeForme" id="idForme"
                  action="MojServlet" method="POST">
                Unesite korisničko ime:
                <input type="text" name="user" size="20"/>
                <br/>
                Unesite lozinku:
                <input type="password" name="pass" size="20"/>
                <br/>
                <input type="button" value="POTVRDI"
                       onClick="provera()"/>
            </form>
        </div>
    </body>
</html>
```

//source file: servleti/MojServlet.java

```
package servleti;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MojServlet extends HttpServlet {
    @Override
    protected void doPost( HttpServletRequest request,
                           HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        String us = request.getParameter("user");
        String pass = request.getParameter("pass");

        //serverska provera preko RegEx ako bi bio iskljucen JavaScript
        String uzorak1 = "^[A-Za-z]\\w{7,}$";
        String uzorak2 = "^(?=.*?[A-Z])(?=.*?[a-z])
                         (?=.*?[0-9])(?=.*?[#?!@$%^&*-]).{8,}$";

        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html>");
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet MojServlet</title>");
            out.println("</head>");
```

```
            out.println("<body>");
            if(us.matches(uzorak1) == true)
                out.println("<h2>Vaše korisničko ime: " + us + "</h2>");
            else
                out.println("Korisničko ime nije u skladu
                             sa regularnim izrazom. ");
            if(pass.matches(uzorak2) == true)
                out.println("<h2>Vaša lozinka: " + pass + "</h2>");
            else
                out.println("Lozinka nije u skladu
                             sa regularnim izrazom. ");
            out.println("</body>");
            out.println("</html>");
        }
    }
}
```

//XML file: WEB-INF/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">
    <servlet>
        <servlet-name>MojServlet</servlet-name>
        <servlet-class>servleti.MojServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>MojServlet</servlet-name>
        <url-pattern>/MojServlet</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

3 JavaServer Pages tehnologija

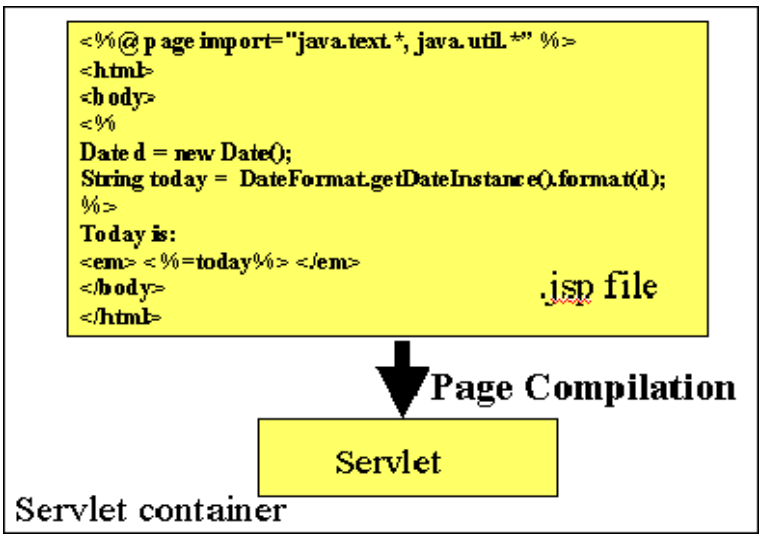
3.1 Uvod u JavaServer Pages

Servleti su predstavljali početak razvoja Javinih internet aplikacija. Ako se izvrši njihova kritička analiza može se zaključiti da je pomoću servleta jednostavno čitati podatke sa forme, čitati zaglavlja (eng. *header*) HTTP zahteva, postavljati HTTP status kod i zaglavlje odgovora, koristiti kolačiće (eng. *cookies*) i rad sa sesijom (eng. *session*), deliti podatke između servleta, zapamtiti podatke između različitih zahteva. S druge strane veoma je naporno koristiti println naredbe da bi se generisao HTML kod i održavati i menjati generisani HTML. Zato se krenulo sa novom tehnologijom JavaServer Pages, čije su osnovne prednosti:

- Odvojen statički sadržaj od dinamičkog sadržaja: kada se klijentski sadržaj generiše u okviru servleta, logika stvaranja dinamičkog sadržaja je sastavni deo koda servleta. Takođe statički deo klijentske strane je ,,sakriven“ u okviru servleta. Pored otežanog praćenja samog koda, nakon svake, pa i najmanje promene, servlet se mora ponovo kompajlirati. Sve ovo prouzrokuje stvaranje aplikacije čija se logika veoma teško razume, a svaka promena zahteva određeno vreme. Sa JSP stranicama statički deo se odvaja od dinamičkog korišćenjem specijalnih tagova i skriptleta (biće naknadno opisani). Kada dizajner strane napravi bilo koju promenu, JSP stranica se automatski rekompajlira i ponovo učita na veb serveru.
- Princip „piši jednom izvršavaj bilo gde“: JSP stranice se mogu prenositi sa jedne platforme na drugu, ili usled promene veb servera, ponašanje aplikacije će biti potpuno isto.
- Dinamički sadržaj se može predstaviti u različitim formatima: ne postoji pravilo u kom formatu mora da bude statički (eng. *template*) deo JSP stranice.
- Dobijanje veb pristupnog nivoa kod aplikacija n-slojne arhitekture.
- Potpuna usklađenost sa servlet tehnologijom: jer su JSP stranice u stvari servleti pisani na velikom nivou apstrakcije. Sve što se može uraditi sa servletima, može i pomoću JSP stranica, ali mnogo jednostavnije.

Svrha postojanja JSP stranica je definisanje deklarativnog, prezentacionog metoda za pisanje servleta. Kao što je već navedeno, JSP specifikacija je definisana kao standardna tehnologija na vrhu Servlet API klasa. Uobičajeno je da JSP stranica prolazi kroz dve faze: translacionu i fazu odgovora na zahtev. Translaciona faza se izvršava samo jednom, sve dok se u kod stranice ne unesu neke izmene, u kom slučaju se prevođenje ponavlja. Ako u kodu na stranici ne postoji greška, rezultat prevođenja je fajl koji predstavlja klasu koja implementira interfejs Servlet, kao što je prikazano na slici.

Ova faza se izvršava na veb serveru automatski, kada se primi prvi put zahtev za stranicom. Takođe, JSP stranica se može iskompajlirati u ,,class“ fajl pre instalacije aplikacije na server čime se izbegava kašnjenje pri prvom pozivu stranice od strane korisnika. Gde se postavlja kod stranice ili iskompajlirana klasa zavisi od korišćenog veb servera. Dobijena klasa od JSP stranice nasleđuje osnovnu klasu HttpJspBase, čime se vrši implementaciju Servlet interfejsa. Kada se jednom klasa učita u okviru servlet kontejnera, _jspService() metod je odgovoran za odgovor na klijentske zahteve. Ovaj metod se ne može preklapati i izvršava se u posebnoj niti u okviru servlet kontejnera.



Slika 1 - Translaciona faza JSP stranice

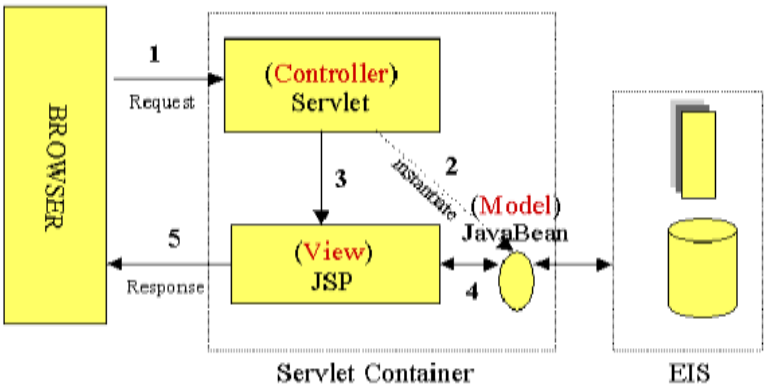
Kada se izvrši prevođenje koda JSP strane navedenog na prethodnoj slici dobija se sledeći složen i veoma nepregledni servlet, koji je je s druge strane prilagođen veb serverima i njihovom izvršavanju:

```
package jsp;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.PrintWriter;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import java.util.Vector;
import org.apache.jasper.runtime.*;
import java.beans.*;
import org.apache.jasper.JasperException;
import java.text.*;
import java.util.*;
public class _0005cjsp_0005cjsptest_0002ejspjsptest_jsp_0
extends HttpJspBase {
   static {
```

```
        }
public _0005cjsp_0005cjsptest_0002ejspjsptest_jsp_0( ) {
}
private static boolean _jspx_inited = false;
public final void _jspx_init() throws JasperException {
}
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
                        throws IOException, ServletException {
      JspFactory _jspxFactory = null;
      PageContext pageContext = null;
      HttpSession session = null;
      ServletContext application = null;
      ServletConfig config = null;
      JspWriter out = null;
      Object page = this;
      String _value = null;
      try {
            if (_jspx_inited == false) {
                  _jspx_init();
                  _jspx_inited = true;
            }
            _jspxFactory = JspFactory.getDefaultFactory();
            response.setContentType("text/html");
            pageContext = _jspxFactory.getPageContext(this,
                        request,response, "", true, 8192, true);
            application = pageContext.getServletContext();
            config = pageContext.getServletConfig();
            session = pageContext.getSession();
            out = pageContext.getOut();
            //begin
            out.write("\r\n<html>\r\n<body>\r\n");
            //end
            //begin [file="E:\\jsp\\jsptest.jsp";from=(3,2);to=(5,0)]
            Date d = new Date();
            String today = DateFormat.getDateInstance().format(d);
            // end
            // begin
            out.write("\r\nToday is: \r\n<em> ");
            // end
            //begin
      [file="E:\\jsp\\jsptest.jsp";from=(7,8);to=(7,13)]
            out.print(today);</b>
            // end
            // begin
            out.write(" </em>\r\n</body>\r\n</html>\r\n");
            // end
      } catch (Exception ex) {
            if (out.getBufferSize() != 0)
```

```
                    out.clear();
                pageContext.handlePageException(ex);
            } finally {
                out.flush();
                _jspxFactory.releasePageContext(pageContext);
            }
        }
    }
```

U okviru predložene arhitekture podrazumeva se da se zahtev šalje servletu, koji ima ulogu kontrolera i pozicije gde je smeštena celokupna poslovna logika. Dobijeni rezultat se šalje JSP stranici koja u ovom slučaju predstavlja prezentacioni nivo.



Slika 2 - Model 2 arhitekture pristupa JSP stranicama

3.2 Razvoj aplikacija uz pomoć JSP

Tehnologija JSP je tokom vremena omogućila više pristupa pomoću kojih se mogla razviti određena internet aplikacija. Najvažniji su sledeći:

- U okviru stranica skript elementi pozivaju kod servleta direktno;
- U okviru stranica skript elementi pozivaju kod servleta indirektno, korišćenjem pomoćnih klasa;
- Realizacija pomoću upotrebe JavaBeans-a;
- Servlet/JSP aplikacija (MVC pristup);
- MVC pristupa (Servlet-JSP aplikacija) uz upotrebu JSP EL (eng. *expression language*);
- Upotreba posebnih (*custom*) tagova;
- MVC pristup (Servlet-JSP aplikacija) upotrebom *JavaBean*-ova, pomoćnih tagova, i frejmvorka, kao što su *Struts* ili *JavaServer Faces*.

Tehnologije su poređane po složenosti, ali i po korisnosti. Prve tehnologije su najjednostavnije, ali se najmanje upotrebljavaju, zbog nemogućnosti razvijanja kompleksnih aplikacija. S druge strane određeni frejmvork je veoma kompleksna tehnologija i zahteva

vreme da se sam frejmvork savlada, ali s druge strane razvoj složenih aplikacija je pojednostavljen.

3.3 Osnovna sintaksa JSP

JSP stranice se upotrebljavaju da bi se na jednostavniji način prikazao klijentski interfejs. Kako je HTML bio i ostao standard za prikaz internet stranice, osnovna prednost JSP tehnologije je mogućnost kombinovanja HTML teksta sa Java kodom. Na ovaj način se omogućava i dizajnerima i programerima podela poslova. Tako se u okviru JSP stranica HTML tagovi mogu koristiti, kao i u običnim HTML stranicama. Na primer kod:

```
<h1>Neki tekst</h1>
```

je dozvoljen i može se poslati dalje klijentu. Pri tome on se prevodi u kod servleta koji je sličan sledećem:

```
out.print("<h1> Neki tekst </h1>");
```

Dozvoljeno je koristiti i HTML komentare koji se isto kao u HTML-u šalju dalje klijentu

```
<!-- Komentar -->
```

Moguće je koristiti i posebne JSP komentare, koji se za razliku od prethodnih, ne šalju samom klijentu:

```
<%-- Komentar --%>
```

U oviru JSP stranica moguće je koristiti i dinamičke elemente, kao što su Java izrazi, skripleti, deklaracije i direktive. Pomoću posebne sintakse moguće je koristiti regularne Java izraze za prikaz određenih rezultata. Opšti tip sintakse je:

```
<%= java_izraz %>
```

Primer:

```
<%= new java.util.Date() %>
```

Evo još jednog primera gde se može primetiti kombinovanje običnog HTML teksta sa mogućnostima programskog jezika Java:

```
<html>

  <h4>Dobrodošli, <%= username %></h4>
  Danas je <%= new java.util.Date() %>.

</html>
```

Pored samih izraza moguće je koristiti i ostale osobine i mogućnosti Jave, poput kontrola toka (if-then-else strukture, switch), petlji (while-do, for, do-while) i mnogih drugih. Deo JSP stranice koji sadrži Java kod se naziva skriptlet (eng. *scriptlet*) i ima sledeću sintaksu:

```
<% java_kod %>
```

Primer:

```
<% for (int i = 0; i < 10; i++) ... %>
```

ili

```
<html>
...
   <% if (Math.random() < 0.5) { %>
   Dobar dan!
   <% } else { %>
   Dobro veče!
   <% } %>
...
</html>
```

```
<html>
...
<table border=1>
   <tr>
     <td>R.br.</td>
     <td>Ime</td>
   </tr>
   <%
   String names[] = {"Marko", "Nikola", "Igor", "Vladimir", "Dejan"};
   for (int i = 0; i < names.length; i++) {
   %>
   <tr>
     <td><%= i %></td>
     <td><%= names[i] %></td>
   </tr>
   <% } %>
</table>
...
</html>
```

Moguće je koristiti i Java deklaracije (eng. *declarations*):

```
<%! java_deklaracija %>

<%! int a; %>
```

Postoje i posebne direktive (eng. *directives*) koje samoj JSP stranici daju dodatne mogućnosti, kao što je upotreba dodatnih fajlova, kodiranje stranice, reagovanje na grešku i slično.

```
<%@ direktiva attr="..."%>

<%@ page contentType="text/plain" %>
```

U okviru JSP stranice moguće je koristiti i neke predefinisane promenljive. Neke od ovih promenljivih su definisane u sledećoj tabeli:

ime	tip
request	HttpServletRequest
response	HttpServletResponse
out	JspWriter
session	HttpSession
application	ServletContext
page	(this)

Objekat tipa *HttpServletResponse* je povezan sa odgovorom klijentu. Dozvoljeno je postavljanje HTTP statusnih kodova i zaglavlja odgovora (*response headers*). *PrintWriter* se koristi za slanje odgovora klijentu, pri čemu je *out* baferovana verzija *PrintWriter* pod nazivom *JspWriter*. Moguće je podešavanje veličine bafera, kao i njegovo potpuno isključivanje pomoću *buffer* atributa *page* direktive. Promenljiva *out* se koristi skoro isključivo u skriptletima, jer se JSP izrazi automatski smeštaju u izlazni tok. *HttpSession* objekat je povezan sa samim zahtevom. Sesije se kreiraju automatski, tako da je ova varijabla već instancirana čak iako nema ulazne reference na sesiju. Izuzetak je ako se koristi session atribut u okviru page direktive sa vrednošću false kako bi se isključilo praćenje sesija. U tom slučaju pokušaj pristupanja session promenljivoj rezultuje generisanjem poruke o grešci od strane servera u momentu prevođenja JSP strane u servlet. Objekat application je objekat klase ServletContext koji služi za komunikaciju servleta i aplikacionog servera. Uobičajena upotreba je za smeštanje globalnih promenljivih uz pomoć metoda setAttribute()/getAttribute(). Objekat page je sinonim za ključnu reč this. Primer upotrebe predefinisanih promenljivih:

```
<html>
...
<% if (request.getParameter("username") != null) { %>
Vrednost parametra: <%= request.getParameter("username") %>
<% } %>
...
</html>
```

Kod JSP stranica, kao i kod regularnih servleta, ponekad postoji potreba za korišćenjem *init* i *destroy* metoda. Problem može nastati kada servlet koji napravljen od JSP stranice možda već koristi svoje init i *destroy* metode. Tada njihovo preklapanje može da prouzrokuje probleme, pa je nelegalno koristiti JSP deklaracije da bi se deklarisale *init* ili *destroy* metode. I u ovom slučaju rešenje postoji - treba koristiti metode *jspInit()* i *jspDestroy()*. Samom specifikacijom se garantuje da će auto-generisani servlet pozivati ove metode u okviru svojih *init* i *destroy*

metoda, ali je standardna verzija *jspInit* i *jspDestroy* metoda prazna, sa mogućnošću preklapanja.

3.4 Bean-ovi i njihova upotreba u internet aplikacijama

Prilikom realizacije komunikacije između servleta i JSP stranice koriste se objekti koji se dobijaju od Java bean-ova. To su Java klase koje, u ovom slučaju, moraju ispunjavati nekoliko pravila:

- Moraju imati zero-argument (prazan) konstruktor;
    - Ovaj zahtev se može ispuniti eksplicitnim definisanjem takvog konstruktora ili izostavljanjem svih konstruktora.
- Ne bi trebalo da ima javne (eng. *public*) promenljive instanci (polja);
- Neposredne vrednosti treba da se dobijaju pomoću metoda nazvanih getX i setX (gde je X naziv polja);
    - Ako klasa ima metod getTitle koji kao rezultat vraća String, kaže se da klasa poseduje String property title.

Treba istaći da se aplikacija može realizovati korišćenjem samo JSP stranica, kada se one koristite za jednostavniji razvoj i održavanje HTML sadržaja, ali se to danas izbegava. U tom slučaju za jednostavniji dinamički kod, moguće je pozvati servlet kod pomoću skript elemenata. Za nešto kompleksnije aplikacije mogu se koristiti pomoćne klase koje se pozivaju iz skript elemenata. Za srednje kompleksne aplikacije, koriste se bean-ovi i pomoćni tagovi. Ali se lako može uočiti da to nije dovoljno. Za kompleksnija procesiranja, rad samo sa JSP stranama nije najprihvatljivije rešenje. I pored odvajanja Java koda u odvojene klase, bean-ove i pomoćne tagove, važi da pojedinačna stranica pruža pojedinačni osnovni izgled.

Zaključak je da se za pojedinačni zahtev koji dolazi sa strane klijenta moguće koristiti samo servlete. Ovaj pristup funkcioniše dobro kada je izlaz binarnog tipa (na primer slika) ili ne postoji izlaz (na primer obavlja se prosleđivanje ili redirekcija) ili se izgled (eng. *layout*) stranice drastično menja (kao kod portala). Takođe moguće je koristiti samo JSP stranice. Ovaj pristup funkcioniše dobro kada je izlaz većim delom tipa karaktera (HTML) i kada je format/layout statičan bez promena.

U praksi je najbolje koristiti kombinaciju JSP strana i servleta (MVC arhitektura). Ona je neophodna kada je rezultat pojedinačnog zahteva više neposrednih stranica različitog izgleda. Ovaj pristup se preporučuje i u slučajevima kada postoji veliki razvojni tim sa različitim podtimovima koji realizuju veb razvoj i poslovnu logiku, i kada se izvršava komplikovana obrada podataka, ali je rezultat relativno statični izgled.

Ovakav pristup je moguće unaprediti, tako što se određeni frejmvork (Struts, JavaServer Faces) može koristiti. Ponekada je upotreba frejmvorka korisna, ali se ne zahteva njihovo

korišćenje, jer je moguće implementirati MVC pristup pomoću objekta tipa *RequestDispatcher* i dobiti sasvim zadovoljavajuće rezultate za jednostavne i srednje kompleksne aplikacije. Da bi se realizovao navedeni pristup potrebno je sprovesti sledeće korake:

1. Definisati bean-ove da bi prezentovali podatke.

2. Koristiti servlet da bi se prihvatio podatak, pri čemu servlet čita parametre zahteva, proverava unete podatke, itd.

3. Postaviti bean-ove. Servlet realizuje poslovnu logiku (kod specifičan za aplikaciju) ili kod za pristup podacima da bi došao do rezultata. Resultati se smeštaju u bean-ove koji su definisani u okviru koraka 1.

4. Postaviti bean u okviru zahteva, sesije ili sadržaja servleta. Servlet poziva *setAttribute()* u okviru objekata zahteva, sesije ili sadržaja servleta da bi smestio referencu bean-ova koji predstavljaju rezultat zahteva.

5. Proslediti zahtev JSP stranici. Servlet prepoznaje koja JSP stranica odgovara datoj situaciji i koristi metod *forward()* od *RequestDispatcher* objekta da bi prebacio kontrolu datoj stranici.

6. Preuzeti podatke iz bean-a. JSP stranica pristupa beanu, pri čemu u ovom slučaju JSP stranica ne kreira ili modifikuje bean; ona samo prihvata i pokazuje podatke koje servlet kreira.

Na strani servleta koristi se objekat *RequestDispatcher* da bi se definisala stranica koja se naredna izvršava. Upotreba ovog objekta je definisana u sledećem primeru:

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response     )
                  throws ServletException, IOException {

  String operation = request.getParameter("operation");
  if (operation == null) {
    operation = "unknown";
  }
  String address;
  if (operation.equals("order")) {
    address = "/WEB-INF/Order.jsp";
  } else if (operation.equals("cancel")) {
    address = "/WEB-INF/Cancel.jsp";
  } else {
    address = "/WEB-INF/UnknownOperation.jsp";
  }
    RequestDispatcher dispatcher = request.getRequestDispatcher(address);
    dispatcher.forward(request, response);
}
```

Ovde treba napomenuti još jednu mogućnost objekta tipa *RequestDispatcher*. Pomoću metode *forward* se dalje izvršavanje prosleđuje određenoj klijentskoj stranici i to tako da se sam URL stranice ne vidi, već korisnik vidi samo URL servleta. Ovakav pristup ima i određene

nedostatke, na primer korisnik ne može da izvrši obeležavanje dobijene JSP stranice i dodavanje u omiljene stranice, jer se njena URL adresa i ne vidi.

Navedeni objekat ima i mogućnost upotrebe metoda *response.sendRedirect* umesto *RequestDispatcher.forward*. Tada i sam korisnik vidi URL dobijene JSP stranice.

Prednosti upotrebe metoda *sendRedirect* su da:

- korisnik može pristupiti JSP stranici odvojeno;
- korisnik može da zapamti adresu JSP stranice.

Nedostaci upotrebe metoda sendRedirect su da:

- Korisnik može da poseti JSP stranicu bez pristupanja servletu, pa JSP podaci mogu da budu nedostupni;
- U okviru JSP stranice trebalo bi detektovati ovu situaciju;
- Ne može se pristupati objektima koji su smešteni u okviru zahteva, jer se pomoću *sendRedirect* pravi nova instanca ovog objekta.

Ako se razmatra navedena MVC arhitektura i preporuka da se podaci smeštaju u okviru bean-ova, ostaje pitanja gde i koliko čuvati napravljene objekte. Kao i u ranijim razmatranjima postoje 3 mogućnosti:

- na nivou zahteva,
- na nivou sesije,
- na nivou aplikacije
- i pristup koji se ne koristi u okviru MVC arhitekture - na nivou stranice.

Za svaki od navedenih nivoa deljenje podataka važi veoma sličan pristup. U sledećim primerima demonstriraće se čitanje parametara na osnovu onog što je do sada navedeno. U kasnijem delu ovog poglavlja obradiće se poseban deo ove tehnologije - jezik za izraze (*expression language*, u daljem tekstu EL) koji omogućava efikasniji i razumljiviji pristup podacima smeštenim u okviru određene oblasti važenja.

3.5 Jezik za izraze - Expression language

Jedno od najviše korišćenih proširenja JSP specifikacije je EL. Glavne prednosti novih opcija su:

- Koncizniji pristup smeštenim objektima - da bi se pristupilo promenljivama čije je stanje zabeleženo (to su objekti zabeleženi metodom *setAttribute* u okviru *PageContext*, *HttpServletRequest*, *HttpSession* ili *ServletContext* objekta); na primer nekoj promenljivoj osoba, koristi se **${osoba}**.

- Skraćena notacija za svojstva bean - da bi se pristupilo svojstvu imeKompaije (na primer, rezultat dobijen od metoda getImeKompanije()) zabeležene promenljive pod imenom kompanija, koristi se **${kompanija.imeKompanije}**. Da bi se pristupilo svojstvu ime od svojstva predsednik zabeležene promenljive kompanija, koristi se **${kompanija.predsednik.ime}**, a to ime se dohvata pozivom metode getIme().
- Jednostavniji pristup elementima kolekcije - pristup elementima tipa *Array*, *List*, ili *Map*, koristi se ${variable[indexOrKey]}. Treba primetiti da je potrebno realizovati prikaz indeksa ili ključa u formi legalnoj za imena Java promenljivih.
- Jednostavan pristup parametrima zahteva, kolačićima (*cookies*), i drugih podataka zahteva - da bi se pristupilo standardnim tipovima podataka zahteva, moguće je koristiti jedan od nekoliko predefinisanih implicitnih objekata.
- Mali, ali koristan skup jednostavnih operatora - za rad sa objektima u okviru EL izraza, može se koristiti bilo koji od nekoliko aritmetičkih, relacionih, logičkih operatora ili operatora za testiranje operanda bez dodeljene vrednosti.
- Uslovni rezultati - da bi se izabrao prikaz između nekoliko opcija, nije potrebno koristiti Java skriplete. Moguće je koristiti ternarni operator: ${uslov ? opcija1 : opcija2}.
- Automatska konverzija tipova - EL uklanja potrebu za korišćenjem konverzije tipova i omogućava ovaj proces automatskim.
- Ne prikazuje se nikakva vrednost, umesto poruke o greškama - u većini slučajeva, nedefinisana promenljiva ili *NullPointerExceptions* prikazaće se kao prazni stringovi, a neće se dogoditi izuzetak.

Kako ovaj deo JSP tehnologije nije deo početnih verzija, nije ga moguće koristiti u svim slučajevima. EL je moguće upotrebljavati samo u okviru servera koji imaju podršku za verziju JSP 2.0 i kasnije (odnosno verzije servleta 2.4 i kasnije).

Osnovna forma koja se koristi je sledeća definicija EL elementa:

```
${expression}
```

Ovako definisani EL elementi se mogu pojaviti kako u običnom HTML tekstu tako i u okviru JSP tag atributa, i omogućiti izvršavanje regularnih JSP izraza. Na primer:

```
<ul>
      <li>Ime: ${expression1}
      <li>Adresa: ${expression2}
</ul>
<jsp:include page="${expression3}" />
```

Sam EL je veoma fleksibilan, pa se može koristiti više izraza u okviru jednog atributa (moguće ih je kombinovati i sa statičkim tekstom), tako da se na kraju konvertuju u stringove i izvrši njihova konkatenacija. Na primer:

```
<jsp:include page="${expr1}nesto${expr2}" />
```

Treba naglasiti da ako se navede ${varName}, tada se izvršava pretraga objekata tipa *PageContext*, *HttpServletRequest*, *HttpSession*, *ServletContext*, u navedenom redosledu i prikaz objekta koji sadrži navedeno ime. Opet treba naglasiti da upotreba *PageContext* ne odgovara MVC pristupu.

Sledeći primer ilustruje redosled pretragu imena navedenog svojstva:

```
public class ScopedVars extends HttpServlet {
   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                     throws ServletException, IOException {

         request.setAttribute("attribute1", "First Value");
         HttpSession session = request.getSession();
         session.setAttribute("attribute2", "Second Value");
         ServletContext application = getServletContext();
         application.setAttribute("attribute3", new java.util.Date());
         request.setAttribute("repeated", "Request");
         session.setAttribute("repeated", "Session");
         application.setAttribute("repeated", "ServletContext");
         RequestDispatcher dispatcher =
               request.getRequestDispatcher("/el/redosled.jsp");
         dispatcher.forward(request, response);
   }
}
```

Ako se na navedenoj klijentskoj strani (redosled.jsp) pristupi definisanim parametrima:

```
<table border=5 align="CENTER">
   <tr><th>Pristup parametrima
</table>
<p>
<ul>
   <li><B>attribute1:</b> ${attribute1}
   <li><B>attribute2:</b> ${attribute2}
   <li><B>attribute3:</b> ${attribute3}
   <li><B>Gde se nalazi "repeated" atribut:</b> ${repeated}
</ul>
</body>
</html>
```

Do sada su predstavljeni primeri kada je pristupano samim parametrima. Pomoću EL može se na veoma koncizan i efikasan način pristupiti i složenijim podacima, kao što su svojstva u okviru bean-a. Sintaksa je sledeća:

```
${varName.propertyName}
```

Okruženje traži promenljivu pod imenom varName i to u okviru objekata tipa *PageContext*, *HttpServletRequest*, *HttpSession*, *ServletContext*, u navedenom redosledu. Kada je pronađe, uzima vrednost njenog svojstva propertyName i dalje obrađuje.

Jedna od prednosti EL je što ima mogućnost da prikaže članove određene kolekcije (niza, liste, heš mape, …). Sintaksa je:

```
${attributeName[entryName]}
```

Može se upotrebiti kod:

- Nizova - ekvivalentno je sa theArray[index]
- Lista - ekvivalentno je sa theList.get(index)
- Mapa - ekvivalentno je sa theMap.get(keyName)

3.6 JSTL (JSP standard tag library)

JSTL (skraćeno od *JSP Standard Tag Library*) je baziran na tagovima frejmvorka *Struts* za kontrolu petlji i logičkih operacija. Nije bio deo JSP 1.2 ili 2.0 specifikacije, već je zahtevao posebnu specifikaciju i podešavanje. JSTL EL je sada deo JSP 2.0 specifikacije.

Pored ostalih mogućnosti JSTL nudi tagove pomoću kojih se mogu na jednostavan način prikazati (obraditi) članovi određene kolekcije i to u okviru klijentske strane. Treba napomenuti da sam JSTL obuhvata i čitav niz dodatnih opcija klijentskoj JSP strani, kao što su pristup bazi podataka, tagovi za uslovne realizacije, za rad sa XML fajlovima, itd. Jedan od razloga zašto se ovi tagovi neće obrađivati u ovoj knjizi je i taj što se njihova upotreba ne uklapa u MVC pristup, koji se želi realizovati.

Da bi se JSTL tagovi mogli koristiti potrebno je na početku JSP strane uključiti JSTL biblioteku pomoću sledećeg taga:

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

Tag koji ima mogućnost obrade kolekcije podataka je c:forEach i njegova sintaksa je:

```
<c:forEach var="element" items="${kolekcija}">
   <c:out value="${element}"/>
</c:forEach>
```

U navedenoj strukturi lokalna promenljiva element u svakoj iteraciji dobija vrednost određenog člana promenljive kolekcija i u okviru c:forEach taga se obrađuje na određeni

način, pri čemu se može koristiti, kao u navedenom primeru, tag c:out koji prikazuje trenutnu vrednost promenljive element.

Ako se pretpostavi da je promenljiva kolekcija niz prethodni primer se može uraditi i pomoću skripleta, ali treba opet napomenuti da ovakav pristup ne odgovara MVC arhitekturi:

```
<ul>
   <%
   for(int i=0; i<kolekcija.length; i++) {
         String element = kolekcija[i];
   %>
         <li><%= element %>
   <% } %>
</ul>
```

Pored odstupanja od MVC arhitekture može se primetiti da upotreba c:forEach taga dovodi do konciznijeg i jednostavnijeg koda, a izbegava se i eksplicitna upotreba samog Java programiranja. Navedeni tag c:forEach se može koristiti i za prikaz statičkih kolekcija kao što je slučaj u sledećem primeru:

```
<%@ taglib prefix="c"uri="http://java.sun.com/jsp/jstl/core" %>
<ul>
   <c:forEach var="i" begin="1" end="10">
         <li><c:out value="${i}"/>
   </c:forEach>
</ul>
```

Zadaci iz tehnologije JavaServer Pages

Zadatak 1

Korišćenjem Java Server Pages tehnologije napisati JSP stranu koja preko HTML forme prihvata dva broja, prosleđuje ih drugoj JSP strani i upoređuje ih. Ukoliko je prvi broj manji od drugog, treba ispisati „Razlika je manja od nule“ i obojiti pozadinu u plavu boju, ukoliko je prvi broj veći od drugog, treba ispisati „Razlika je veća od nule“ i obojiti pozadinu u zelenu boju, a ukoliko su brojevi isti, treba ispisati „Razlika je nula“ i obojiti pozadinu u žutu boju. Sve provere treba realizovati na JSP stranama.

REŠENJE

```
//web page: index.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <title>Poredjenje</title>
    </head>
    <body>
       <form action="rezultat.jsp" method="post">
           Unesite prvi broj: <input type="text" name="prvi"/><br/>
           Unesite drugi broj: <input type="text" name="drugi"/><br/>
           <input type="submit" value="Uporedi"/>
       </form>
     </body>
</html>


//web page: rezultat.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!-- direktiva -->
<%@ page errorPage="error.jsp" %>

<!-- deklaracija -->
<%! int a, b, razlika;
    String prvi, drugi;
%>

<!-- skriplet -->
<%
```

```
        prvi = request.getParameter("prvi");
        drugi = request.getParameter("drugi");
        a = Integer.parseInt(prvi);
        b = Integer.parseInt(drugi);
        razlika = a - b;
    %>




    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
            <title>
                <%if (razlika < 0) {%>
                Manje
                <%} else if (razlika > 0) {%>
                Veće
                <%} else {%>
                Nula
                <%}%>
            </title>
        </head>
        <body bgcolor="<%if (razlika < 0) {%>
              blue
              <%} else if (razlika > 0) {%>
              green
              <%} else {%>
              red
              <%} %>">

            <h2>
                Razlika je
                <%if (razlika < 0) {%>
                manja od nule.
                <%} else if (razlika > 0) {%>
                veća od nule.
                <%} else {%>
                nula.
                <%}%>
            </h2>
        </body>
    </html>

    //web page: error.jsp

    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <%@page isErrorPage="true" %>
```

```
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
            <title>Error Page</title>
        </head>
        <body>
            <h2>Greska!</h2>
            <h3><%=exception%></h3>
        </body>
    </html>
```

Zadatak 2

Korišćenjem Java Server Pages tehnologije napisati JSP stranu koja preko HTML forme prihvata dva broja, prosleđuje ih Java Servletu, koji ih upoređuje. Ukoliko je prvi broj veći od drugog, servlet nas preusmerava na stranu *vece.jsp*, ukoliko je prvi broj manji od drugog, servlet nas preusmerava na stranu *manje.jsp*, a ukoliko je razlika jednaka 0, servlet nas preusmerava na stranu *nula.jsp*. Sve provere treba realizovati na servletu.

REŠENJE

```
//web page: index.jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
              charset=UTF-8">
        <title>MVC prvi primer</title>
    </head>
    <body>
        <H2>MVC pristup resavanja problema - prednosti i mane</H2>
       <form action="Uporedi" method="post">
           Unesite prvi broj: <input type="text" name="prvi"/><br/>
           Unesite drugi broj: <input type="text" name="drugi"/><br/>
        <input type="submit" value="Uporedi"/>
       </form>
     </body>
</html>

//web page: vece.jsp

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

```
        <head>
            <meta http-equiv="Content-Type" content="text/html;
                  charset=UTF-8">
            <title>Vece</title>
        </head>
        <body bgcolor="#00ff00">
            <h1>Razlika je veca od nule.</h1>
        </body>
    </html>
```

//web page: manje.jsp

```
    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
            <title>Manje</title>
        </head>
        <body bgcolor="#0000ff">
            <h1>Razlika je manja od nule.</h1>
        </body>
    </html>
```

//web page: nula.jsp

```
    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
            <title>Nula</title>
        </head>
        <body bgcolor="#ff0000">
            <h1>Razlika je nula.</h1>
        </body>
    </html>
```

//web page: error.jsp

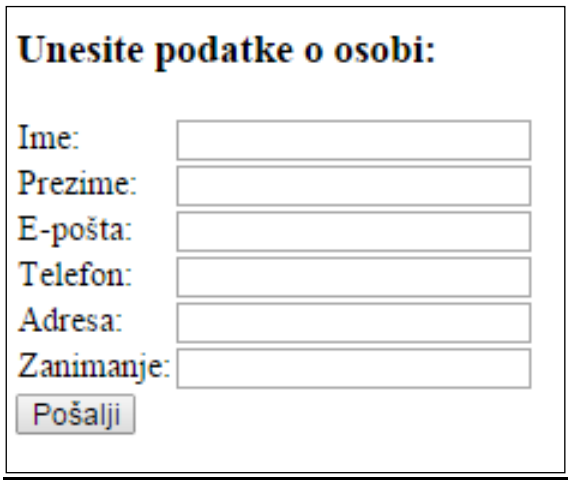
```
    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
            <title>Error Page</title>
        </head>
        <body bgcolor="#ff0000">
            <h2>Greska!</h2>
        </body>
```

```
</html>

//source file: servlets/UporediServlet.java

package servlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class UporediServlet extends HttpServlet {
    @Override
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
            throws ServletException, IOException {
        String prvi = request.getParameter("prvi");
        String drugi = request.getParameter("drugi");
        String address;
        int a, b;
        try {
            a = Integer.parseInt(prvi);
            b = Integer.parseInt(drugi);
            int razlika = a - b;
            if (razlika > 0) {
                address = "/vece.jsp";
            } else if (razlika < 0) {
                address = "/manje.jsp";
            } else {
                address = "/nula.jsp";
            }
        } catch (NumberFormatException nfe) {
            address = "/error.jsp";
        }
        RequestDispatcher                  dispatcher                  =
request.getRequestDispatcher(address);
        dispatcher.forward(request, response);
    }
}


//XML file: WEB-INF/web.xml

<?xml version="1.0" encoding="UTF-8"?>
<web-app     version="2.5"     xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <servlet>
```

```
        <servlet-name>Uporedi</servlet-name>
        <servlet-class>servlets.UporediServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Uporedi</servlet-name>
        <url-pattern>/Uporedi</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

Zadatak 3

Korišćenjem Java Server Pages tehnologije napisati JSP stranu koja preko HTML forme prihvata podatke o osobi: ime, prezime, adresu e-pošte, telefon, adresu i zanimanje i prosleđuje ih drugoj JSP strani, koja prikazuje te podatke. Za prikaz podataka potrebno je koristiti Java binove (*Java beans*).



REŠENJE

```
//web page: index.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML>
<html>
    <head>
```

```
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
            <title>Rad sa binovima</title>
        </head>
        <body>
            <h3>Unesite podatke o osobi:</h3>
            <form action="prikaz.jsp" method="post">
                <table>
                    <tr>
                        <td>Ime:</td>
                        <td><input type="text" name="ime"/></td>
                    </tr>
                    <tr>
                        <td>Prezime:</td>
                        <td><input type="text" name="prezime"/></td>
                    </tr>
                    <tr>
                        <td>E-pošta:</td>
                        <td><input type="text" name="email"/></td>
                    </tr>
                    <tr>
                        <td>Telefon:</td>
                        <td><input type="text" name="telefon"/></td>
                    </tr>
                    <tr>
                        <td>Adresa:</td>
                        <td><input type="text" name="adresa"/></td>
                    </tr>
                    <tr>
                        <td>Zanimanje:</td>
                        <td><input type="text" name="zanimanje"/></td>
                    </tr>
                </table>
                <input type="submit" value="Pošalji">
            </form>
        </body>
    </html>

    //source file: beans/Korisnik.java

    package beans;
    public class Korisnik {
        private String ime = "";
        private String prezime = "";
        private String email = "";
        private String telefon = "";
        private String adresa = "";
        private String zanimanje = "";
```

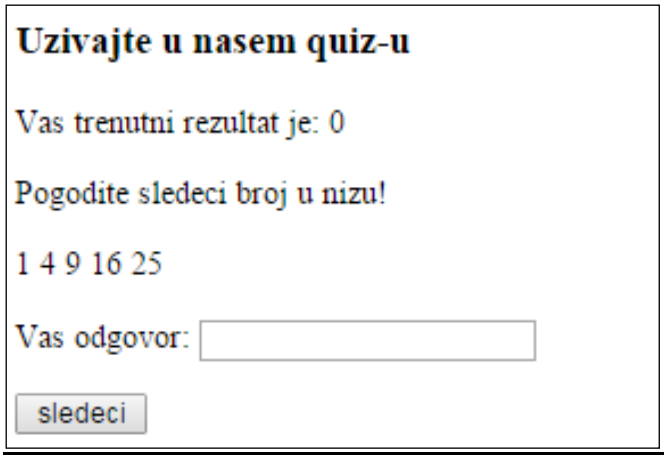
```
    public String getIme() {
        return ime;
    }

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

    public String getPrezime() {
        return prezime;
    }

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

    public String getEmail() {
        return email;
    }

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

    public String getTelefon() {
        return telefon;
    }

    public void setTelefon(String telefon) {
        this.telefon = telefon;
    }

    public String getAdresa() {
        return adresa;
    }

    public void setAdresa(String adresa) {
        this.adresa = adresa;
    }

    public String getZanimanje() {
        return zanimanje;
    }

    public void setZanimanje(String zanimanje) {
        this.zanimanje = zanimanje;
    }
}
```

```
//web page: prikaz.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <title>Prikaz</title>
    </head>
    <jsp:useBean id="korisnik" class="beans.Korisnik"  scope="session"/>
    <jsp:setProperty name="korisnik" property="*"/>
    <body>
        <table>
            <tr>
                <td>Ime:</td>
                <td>${korisnik.ime}</td>
            </tr>
            <tr>
                <td>Prezime:</td>
                <td>${korisnik.prezime}</td>
            </tr>
            <tr>
                <td>e-mail:</td>
                <td>${korisnik.email}</td>
            </tr>
            <tr>
                <td>Telefon:</td>
                <td>${korisnik.telefon}</td>
            </tr>
            <tr>
                <td>Adresa:</td>
                <td>${korisnik.adresa}</td>
            </tr>
            <tr>
                <td>Zanimanje:</td>
                <td>${korisnik.zanimanje}</td>
            </tr>
        </table>
    </body>
</html>
```

Zadatak 4

Korišćenjem tehnologija Java Servlet i Java Server Pages napraviti mini aplikaciju za kviz matematičkog znanja. Korisnik treba da ima mogućnost da na veb strani vidi pitanje, polje za unos odgovora i dugme za potvrdu, kao i trenutni rezultat koji je postigao u kvizu.

Prikaz pitanja i unos odgovora treba da se izvrši na JSP strani, a sve provere treba da se izvrše na servletu.



REŠENJE

```
//web page: index.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML>
<html>
    <head>
        <title>JSP Page</title>
    </head>

    <body>
        <H2>Dobrodosli na nas kviz!</H2><br/>
        Kliknite za pocetak <a href="QuizHandler"> kviza </a>
    </body>
</html>

//source file: servlets/QuizHandlerServlet.java

package servlets;

import beans.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class QuizHandlerServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request,
                                  HttpServletResponse response)
            throws ServletException, IOException {
```

```
            HttpSession session = request.getSession(true);
            QuizBean quizBean = (QuizBean) session.getAttribute("quizBean");

            String address = "quiz.jsp";

            if ((quizBean == null) || (quizBean.isFinished() == true)) {
                // pocetak kviza
                quizBean = new QuizBean();
                session.setAttribute("quizBean", quizBean);

            } else {
                // negde u toku kviza
                int userAnswer = 0;
                boolean ok = true;
                try {
                    userAnswer                                                    =
Integer.parseInt(request.getParameter("answer"));
                } catch (NumberFormatException nfe) {
                    ok = false;
                }

                if (ok && (quizBean.checkAnswer(userAnswer) == true)) {
                    quizBean.incrementScore();
                }


                quizBean.setCurrentIndex(quizBean.getCurrentIndex() + 1);

                if (quizBean.isFinished() == true) {
                    address = "score.jsp";
                }
            }

            response.sendRedirect(address);
            //svi objekti su sacuvani u sesiji
            //pa je bolja varijanta sa sendRedirect
        }
        @Override
        protected void doGet(HttpServletRequest request,
                             HttpServletResponse response)
                throws ServletException, IOException {
            processRequest(request, response);
        }
        @Override
        protected void doPost(HttpServletRequest request,
                              HttpServletResponse response)
                throws ServletException, IOException {
            processRequest(request, response);
        }
```

```
}

//source file: beans/ProblemBean.java

package beans;
public class ProblemBean {

    private int[] sequence;
    private int solution;

    /**
     * Kreiranje nove instance ProblemBean
     */
    public ProblemBean() {
    }

    public ProblemBean(int[] sequence, int solution) {
        this.sequence = new int[sequence.length];

        for (int i = 0; i < sequence.length; i++) {
            this.sequence[i] = sequence[i];
        }

        this.solution = solution;
    }

    // PROPERTY: sequence
    public int[] getSequence() {
        return sequence;
    }

    public void setSequence(int[] sequence) {
        this.sequence = sequence;
    }

    // PROPERTY: solution
    public int getSolution() {
        return solution;
    }

    public void setSolution(int solution) {
        this.solution = solution;
    }

    public String toString() {
        String result = "";

        for (int i = 0; i < sequence.length; i++) {
            result += sequence[i] + " ";
```

```
            }

            return result;
        }
    }

    //source file: beans/QuizBean.java

    package beans;
    import java.util.ArrayList;

    public class QuizBean {

        private ArrayList<ProblemBean> problems = new
ArrayList<ProblemBean>();
        private int currentIndex;
        private int score;

        /** pravljenje nove instance QuizBean */
        public QuizBean() {
            problems.add(new ProblemBean(new int[]{1, 4, 9, 16, 25}, 36));
             // kvadrati
            problems.add(new ProblemBean(new int[]{1, 1, 2, 3, 5}, 8));
             // fibonaci
            problems.add(new ProblemBean(new int[]{3, 1, 4, 1, 5}, 9));
         // pi
            problems.add(new ProblemBean(new int[]{2, 3, 5, 7, 11}, 13));
         // prosti
            problems.add(new ProblemBean(new int[]{1, 2, 4, 8, 16}, 32));
         // stepeni dvojke
            problems.add(new ProblemBean(new int[]{0, 7, 26, 63, 124}, 215));
         // kubovi -1

            currentIndex = 0;
            score = 0;
        }

        // PROPERTY: problems
        public void setProblems(ArrayList problems) {
            this.problems = problems;
            currentIndex = 0;
            score = 0;
        }

        // PROPERTY: score
        public int getScore() {
            return score;
        }
```

```
        public void incrementScore() {
            score++;
        }

        // PROPERTY: currentIndex
        public int getCurrentIndex() {
            return currentIndex;
        }

        public void setCurrentIndex(int currentIndex) {
            this.currentIndex = currentIndex;
        }

        // PROPERTY: currentProblem
        public ProblemBean getCurrentProblem() {
            return problems.get(currentIndex);
        }

        public boolean checkAnswer(int answer) {
            return getCurrentProblem().getSolution() == answer;
        }

        public boolean isFinished() {
            return currentIndex == problems.size();
        }
    }
```

//web page: quiz.jsp

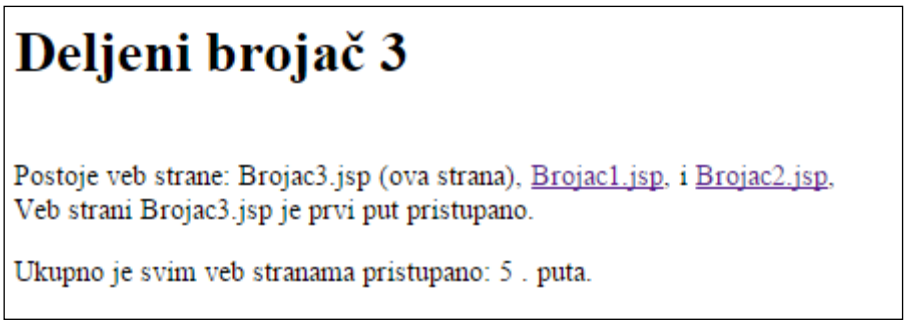
```
    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <!DOCTYPE HTML>
    <html>
        <head>
            <title> * * * Number Quiz * * *</title>
        </head>
        <body>
            <jsp:useBean         id="quizBean"         type="beans.QuizBean"
scope="session"/>

            <form name="quiz" action="QuizHandler" method="post">
                <h3>Uzivajte u nasem quiz-u</h3>
                <p> Vas trenutni rezultat je:
                    <jsp:getProperty name="quizBean" property="score"/>
                </p>
                <p> Pogodite sledeci broj u nizu! </p>
                <p><jsp:getProperty                          name="quizBean"
property="currentProblem"/>
             </p>
                <p> Vas odgovor: <input type="text" name="answer"></p>
```

```
                <p> <input type="submit" value=" sledeci "></p>
            </form>
        </body>
    </html>
    //web page: score.jsp

    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <!DOCTYPE HTML>
    <html>
        <head>
            <title> * * * Number Quiz * * *</title>
        </head>

        <body>
            <jsp:useBean         id="quizBean"         type="beans.QuizBean"
scope="session"/>

            <h3> Hvala sto ste igrali nas kviz. </h3>
            <p> Vas rezultat je:
             <jsp:getProperty name="quizBean" property="score"/>
            </p>
            <br/>
            Kliknite <a href="QuizHandler"> ovde </a>
             ako zelite ponovo da ucestvujete u kvizu.

        </body>
    </html>


    //XML file: WEB-INF/web.xml

    <?xml version="1.0" encoding="UTF-8"?>
    <web-app     version="2.5"     xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

        <servlet>
            <servlet-name>QuizHandlerServlet</servlet-name>
            <servlet-class>servlets.QuizHandlerServlet</servlet-class>
        </servlet>
        <servlet-mapping>
            <servlet-name>QuizHandlerServlet</servlet-name>
            <url-pattern>/QuizHandler</url-pattern>
        </servlet-mapping>
        <session-config>
            <session-timeout>
                30
            </session-timeout>
```

```
        </session-config>
        <welcome-file-list>
            <welcome-file>index.jsp</welcome-file>
            </welcome-file-list>
        </web-app>
```

Zadatak 5

Korišćenjem Java Server Pages tehnologije napraviti tri JSP strane Brojac1.jsp, Brojac2.jsp i Brojac3.jsp, kojima korisnik naizmenično pristupa, ali tako da se u Java binu (bean) čuva podatak o prvoj pristupanoj strani i o ukupnom broju pristupanja svim stranama.



REŠENJE

```
//web page: Brojac1.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <title>Deljeni brojač 1</title>
    </head>
    <body>
        <!-- Obratiti paznju na startnu stranicu u web.xml-->
        <H1>Deljeni brojač 1</H1><br/>
            <jsp:useBean id="counter"
                         class="beans.AccessCountBean"
                         scope="application">
            <!--Uslovno se izvrsava, samo onda kada se bean zaista kreira-
->
            <jsp:setProperty name="counter"
                             property="firstPage"
                             value="Brojac1.jsp" />
        </jsp:useBean>
        Postoje veb strane: Brojac1.jsp (ova strana),
```

```
            <A HREF="Brojac2.jsp">Brojac2.jsp</A>, i
            <A HREF="Brojac3.jsp">Brojac3.jsp</A>,
            <BR/>
            Veb strani <jsp:getProperty name="counter" property="firstPage"
/>
            je prvi put pristupano.
            <P>
                Ukupno je svim stranicama pristupano:
                <jsp:getProperty name="counter" property="accessCount" />
                . puta.
                <jsp:setProperty name="counter"
                                 property="accessCount"
                                 value="1" />
        </BODY>
    </HTML>
```

//web page: Brojac2.jsp

```
    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <!DOCTYPE HTML>

    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
            <title>Deljeni brojač 2</title>
        </head>
        <body>
            <!-- Obratiti paznju na startnu stranicu u web.xml-->
            <H1>Deljeni brojač 2</H1><br/>
                <jsp:useBean id="counter"
                             class="beans.AccessCountBean"
                             scope="application">
                <!--Uslovno se izvrsava, samo onda kada se bean zaista kreira-
->
                <jsp:setProperty name="counter"
                                 property="firstPage"
                                 value="Brojac2.jsp" />
            </jsp:useBean>
            Postoje veb strane: Brojac2.jsp (ova strana),
            <A HREF="Brojac1.jsp">Brojac1.jsp</A>, i
            <A HREF="Brojac3.jsp">Brojac3.jsp</A>,
            <BR/>
            Veb strani <jsp:getProperty name="counter" property="firstPage"
/>
            je prvi put pristupano.
            <P>
                Ukupno je svim stranicama pristupano:
                <jsp:getProperty name="counter" property="accessCount" />
```

```
                . puta.
                <jsp:setProperty name="counter"
                                 property="accessCount"
                                 value="1" />
        </BODY>
    </HTML>

    //web page: Brojac3.jsp

    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <!DOCTYPE HTML>

    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
            <title>Deljeni brojač 3</title>
        </head>
        <body>
            <!-- Obratiti paznju na startnu stranicu u web.xml-->
            <H1>Deljeni brojač 3</H1><br/>
                <jsp:useBean id="counter"
                             class="beans.AccessCountBean"
                             scope="application">
                <!--Uslovno se izvrsava, samo onda kada se bean zaista kreira-
->
                <jsp:setProperty name="counter"
                                 property="firstPage"
                                 value="Brojac3.jsp" />
            </jsp:useBean>
            Postoje veb strane: Brojac3.jsp (ova strana),
            <A HREF="Brojac1.jsp">Brojac1.jsp</A>, i
            <A HREF="Brojac2.jsp">Brojac2.jsp</A>,
            <BR/>
            Veb strani <jsp:getProperty name="counter" property="firstPage"
/>
            je prvi put pristupano.
            <P>
                Ukupno je svim veb stranama pristupano:
                <jsp:getProperty name="counter" property="accessCount" />
                . puta.
                <jsp:setProperty name="counter"
                                 property="accessCount"
                                 value="1" />
        </BODY>
    </HTML>

    //source file: beans/AccessCountBean.java
```

```
package beans;

public class AccessCountBean {
   //objekat koji se cuva u okviru objekta application
         // i traje do restartovanja web servera

    private String firstPage;
    private int accessCount = 1;

    public String getFirstPage() {
        return (firstPage);
    }

    public void setFirstPage(String firstPage) {
        this.firstPage = firstPage;
    }

    public int getAccessCount() {
        return (accessCount);
    }

    public void setAccessCount(int increment) {
        accessCount = accessCount + increment;
    }
}

//XML file: WEB-INF/web.xml

<?xml version="1.0" encoding="UTF-8"?>
<web-app     version="2.5"     xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
       <session-config>
           <session-timeout>
               30
           </session-timeout>
       </session-config>
       <welcome-file-list>
           <welcome-file>Brojac3.jsp</welcome-file>
           </welcome-file-list>
       </web-app>
```

Zadatak 6

Korišćenjem Java Servlet i Java Server Pages tehnologija napraviti JSP stranu koja prikazuje formu za logovanje korisnika, u koju korisnik treba da unese svoje korisničko

ime i svoju lozinku (kredencijali), i ukoliko su korisnički kredencijali ispravni (postoje u bazi podataka u tabeli *KorisnikInfo*) korisniku prikazati stranicu sa svojim ličnim podacima (JSP strana *prikaz.jsp*), a ukoliko podaci nisu dobri korisniku prikazati stranicu sa greškom (JSP strana *error.jsp*). Takođe, potrebno je realizovati još dve JSP strane: za promenu ličnih podataka (JSP strana *promena.jsp*) i strana sa porukom „Doviđenja“ kada korisnik pritisne dugme za odjavljivanje iz sistema (JSP strana *kraj.jsp*). Rad sa podacima treba realizovati korišćenjem servleta.

Izgled tabela u bazi *dbkorisnik*:

KorisnikInfo		
#	Naziv kolone	Tip
1	username (PK)	varchar(32)
2	password	varchar(32)
3	first_name	varchar(32)
4	last_name	varchar(32)
5	email	varchar(32)
6	phone	varchar(32)

REŠENJE

```
//source file: util/DB.java

package util;

import java.sql.Connection;
import java.sql.DriverManager;

public class DB {

    private static DB instance;
    private static final int MAX_CON = 5;
    private static final Connection[] bafer = new Connection[MAX_CON];
    private int first = 0, last = 0, free = MAX_CON;

    private DB() { //za MySQL
        try {
            Class.forName("com.mysql.jdbc.Driver");
            for (int i = 0; i < MAX_CON; i++) {
                bafer[i] =
               DriverManager.getConnection
```

```
                    ("jdbc:mysql://localhost:3306/dbkorisnik",     "root",
                "");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /* za Access
     private DB(){
     try{
     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
     for(int i=0; i<MAX_CON; i++)
     bafer[i] = DriverManager.getConnection("jdbc:odbc:dbkorisnik");
     }catch(Exception e){}
     }
     */
    public static DB getInstance() {
        if (instance == null) {
            instance = new DB();
        }
        return instance;
    }

    public synchronized Connection getConnection() {
        if (free == 0) {
            return null;
        }
        free--;
        Connection con = bafer[first];
        first = (first + 1) % MAX_CON;
        return con;
    }

    public synchronized void putConnection(Connection con) {
        if (con == null) {
            return;
        }
        free++;
        bafer[last] = con;
        last = (last + 1) % MAX_CON;
    }
}
```

//source file: beans/Korisnik.java
//Java bean

```
package beans;
```

```
public class Korisnik {

    private String username = "";
    private String password = "";
    private String email = "";
    private String telefon = "";
    private String ime = "";
    private String prezime = "";

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getEmail() {
        return email;
    }

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

    public String getTelefon() {
        return telefon;
    }

    public void setTelefon(String telefon) {
        this.telefon = telefon;
    }

    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;
        }
    }

    //source file: servleti/Login.java
    //servlet za logovanje korisnika u sistem

    package servleti;

    import beans.*;
    import java.io.*;
    import java.sql.*;
    import javax.servlet.*;
    import javax.servlet.http.*;
    import util.DB;

    public class Login extends HttpServlet {

        protected void doPost(HttpServletRequest request,
                              HttpServletResponse response)
                throws ServletException, IOException {
            HttpSession sesija = request.getSession();
            String poruka = "";
            String username = (String) request.getParameter("username");
            String password = (String) request.getParameter("password");
            Korisnik korisnik = new Korisnik();
            korisnik.setUsername(username);
            korisnik.setPassword(password);
            sesija.setAttribute("korisnik", korisnik);
            if (username.isEmpty() || password.isEmpty()) {
                poruka = "Niste popunili sva polja!";
                request.setAttribute("poruka", poruka);
                RequestDispatcher                    rd                    =
request.getRequestDispatcher("/index.jsp");
                rd.forward(request, response);
            }
            Connection con = null;
            Statement stmt = null;
            ResultSet rs = null;
            String address = "prikaz.jsp";
            try {
                con = DB.getInstance().getConnection();
                stmt = con.createStatement();
```

```
                String upit = "select * from KorisnikInfo where username='"
+
                        username + "' and password='" + password + "';";
                rs = stmt.executeQuery(upit);
                if (rs.next()) {
                    String email = rs.getString("email");
                    String ime = rs.getString("first_name");
                    String prezime = rs.getString("last_name");
                    String telefon = rs.getString("phone");
                    korisnik.setEmail(email);
                    korisnik.setIme(ime);
                    korisnik.setPrezime(prezime);
                    korisnik.setTelefon(telefon);
                    stmt.close();
                    request.setAttribute("poruka", "Vasi podaci");
                } else {
                    poruka = "Neispravno korisnicko ime i lozinka!
                           Pokusajte ponovo.";
                    request.setAttribute("poruka", poruka);
                    korisnik.setPassword("");
                    address = "index.jsp";
                    stmt.close();
                }

            } catch (SQLException ex) {
                sesija.invalidate();
                String errormsg = ex.getMessage();
                request.setAttribute("errormsg", errormsg);
                address = "error.jsp";
            } finally {
                DB.getInstance().putConnection(con);
            }

            RequestDispatcher rd = request.getRequestDispatcher(address);
            rd.forward(request, response);
        }
    }

    //source file: servleti/Logout.java
    //servlet za izlogovanje korisnika iz sistema

    package servleti;

    import java.io.*;
    import javax.servlet.*;
    import javax.servlet.http.*;

    public class Logout extends HttpServlet {
```

```
        protected void processRequest(HttpServletRequest request,
                                      HttpServletResponse response)
                throws ServletException, IOException {
            request.getSession().invalidate();
            request.getRequestDispatcher("kraj.jsp").forward(request,
response);
        }
        @Override
        protected void doGet(HttpServletRequest request,
                          HttpServletResponse response)
                throws ServletException, IOException {
            processRequest(request, response);
        }
        @Override
        protected void doPost(HttpServletRequest request,
                           HttpServletResponse response)
                throws ServletException, IOException {
            processRequest(request, response);
        }
    }

    //source file: servleti/Izmena.java
    //servlet za promenu podataka o korisniku

    package servleti;

    import beans.Korisnik;
    import java.io.*;
    import java.sql.*;
    import javax.servlet.*;
    import javax.servlet.http.*;
    import util.DB;

    public class Izmena extends HttpServlet {

        @Override
        protected void doPost(HttpServletRequest request,
                           HttpServletResponse response)
                throws ServletException, IOException {
            HttpSession sesija = request.getSession();
            Korisnik korisnik = (Korisnik) sesija.getAttribute("korisnik");
            String ime = (String) request.getParameter("ime");
            String prezime = (String) request.getParameter("prezime");
            String telefon = (String) request.getParameter("telefon");
            String email = (String) request.getParameter("email");

            korisnik.setIme(ime);
            korisnik.setPrezime(prezime);
            korisnik.setTelefon(telefon);
```

```
            korisnik.setEmail(email);
            String upit = "update KorisnikInfo "
                    + "set first_name='" + ime + "',last_name='" + prezime
                    + "', phone='" + telefon + "', email='" + email + "'"
                    + "where username='" + korisnik.getUsername() + "';";

            Connection con = null;
            Statement stmt = null;
            String address = "prikaz.jsp";
            try {
                con = DB.getInstance().getConnection();
                stmt = con.createStatement();
                stmt.executeUpdate(upit);
                stmt.close();
                con.close();
            } catch (SQLException ex) {
                sesija.invalidate();
                String errormsg = ex.getMessage();
                request.setAttribute("errormsg", errormsg);
                address = "error";
            } finally {
                DB.getInstance().putConnection(con);
            }
            request.setAttribute("poruka", "Podaci su uspesno izmenjeni");
            RequestDispatcher rd = request.getRequestDispatcher(address);
            rd.forward(request, response);
        }
    }
```

//web page: index.jsp
//prva strana za logovanje korisnika

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <title>Korisnička aplikacija</title>
    </head>
    <body>
        <h3>${(poruka!=null) ? poruka : "Dobrodošli!"} </h3><br/><br/>
        <form action="login" method="POST">
            <table>
                <tr>
                    <td>Korisničko ime:</td>
                    <td><input type="text" name="username"
                     value="${korisnik.username}" size="20"/></td>
```

```
                    </tr>
                    <tr>
                        <td>Lozinka:</td>
                        <td><input type="password" name="password"
                         value="${korisnik.password}" size="20"/></td>
                    </tr>
                    <tr>
                        <td><input type="submit" value="Ulogujte se"/></td>
                        <td><input type="reset" value="Poništite"/></td>
                    </tr>
                </table>
            </form>
        </body>
    </html>
```

```
    //web page: kraj.jsp
    //poslednja strana kada se korisnik izloguje

    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <!DOCTYPE HTML>

    <html>
        <head>
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
            <title>Doviđenja</title>
        </head>
        <body>
            <h1>Doviđenja</h1>
            <a href="index.jsp">Na početak</a>
        </body>
    </html>

    //web page: prikaz.jsp
    //prva strana nakon logovanja, koja prikazuje podatke o korisniku

    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <!DOCTYPE HTML>

    <html>
        <head>
            <title>Korisnička aplikacija</title>
        </head>
        <body>
            <h3>${poruka}</h3><br/><br/>
            <table width="40%" border="3">
                <tr><td width="50%"> Korisničko ime:</td>
                    <td>${korisnik.username} </td></tr>
                <tr><td>Ime:</td>
```

```
                <td>${korisnik.ime}</td></tr>
            <tr><td>Prezime:</td>
                <td>${korisnik.prezime}</td></tr>
            <tr><td>Email:</td>
                <td>${korisnik.email}</td></tr>
            <tr><td>Telefon:</td>
                <td>${korisnik.telefon}</td></tr>
        </table><br/>
        <a href="promena.jsp">Promenite podatke</a>
        <a href="logout">Izlogujte se</a>
    </body>
</html>
```

//web page: promena.jsp
//strana na kojoj korisnik moze promeniti svoje podatke

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML>

<html>
    <head>
        <title>Korisnička aplikacija</title>
    </head>
    <body>
       <h3>Vaši podaci koje možete menjati</h3><br/><br/>
       <form action="izmena" method="post" >
          <table width="40%" border="3">
              <tr><td width="50%">Ime:</td>
              <td><input type="text" name="ime" value="${korisnik.ime}"
               size="30"/></td></tr>
              <tr><td>Prezime:</td>
              <td><input type="text" name="prezime"
                  value="${korisnik.prezime}" size="30"/></td></tr>
              <tr><td>Email:</td>
              <td>   <input type="text" name="email"
                     value="${korisnik.email}" size="30"/>
               </td></tr>
              <tr><td>Telefon:</td>
              <td><input type="text" name="telefon"
                  value="${korisnik.telefon}" size="30" /></td></tr>
          </table><br/>
          <input type="submit" value="Prihvatite izmenu"/>
       </form>
     </body>
</html>
```

//web page: error.jsp
//strana na kojoj se prikazuje poruka o gresci

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html;
         charset=UTF-8">
        <title>Greška</title>
    </head>
    <body>
        <h1>Greška u radu sa bazom podataka</h1>
        <h3>${errormsg}</h3>
    </body>
</html>
```

Zadatak 7

Napraviti sledeću Internet aplikaciju sa MVC arhitekturom koristeći JavaServlet i JSP tehnologiju. Sistem služi za evidenciju položenih ispita za studente. U sistemu postoje studenti koji imaju ime, prezime, godinu studija i broj indeksa (primarni ključ u tabeli Student). Evidencija o položenim ispitima se vodi u tabeli Ispiti, gde se za svaki položeni ispit pamte šifra položenog predmeta, student koji je polagao, datum ispita i ocena koju je dobio.

Na polaznoj veb stranici (index.jsp) potrebno je napraviti link ka servletu, koji služi za kreiranje baze podataka. Posle izvršavanja servleta, ukoliko nije bilo greške, potrebno je prikazati veb stranicu pocetna.jsp. Ukoliko je došlo do greške u radu sa bazom podataka, prijaviti tačan opis greške na stranici greska.jsp.

Veb stranica pocetna.jsp treba da sadrži tri forme, svaka sa posebnim kontrolnim (submit) dugmetom za slanje svake forme na server. Prva forma sadrži polje za unos broja indeksa. Prosleđivanjem ove forme potrebno je generisati veb stranicu prosek.jsp sa imenom i prosekom ocena za određenog studenta. Ukoliko student ne postoji u bazi podataka, ista stranica treba da prikaže odgovarajuće obaveštenje. Stranica prosek.jsp treba da ima i link ka stranici pocetna.jsp.

Druga forma stranice pocetna.jsp treba da omogući unos dva datuma. Prosleđivanjem ove forme generiše se na stranici ocene.jsp tabelarni spisak svih studenata, šifara predmeta i ocena za ispite koje su položili u okviru tog vremenskog intervala. Ukoliko ulazni podaci nisu korektni, korisniku treba prikazati upozorenje. U slučaju da za ispravno unete datume ne postoje evidentirani položeni ispiti, na istoj rezultujućoj veb stranici treba prikazati obaveštenje o tome, bez prikazivanja tabele sa praznim sadržajem. Stranica ocene.jsp takođe treba da ima link ka stranici pocetna.jsp.

Treća forma treba da omogući unos broja indeksa, šifre predmeta i ocene studenta. Za datum ispita podrazumeva se tekući dan. Za slučaj da za uneti predmet već postoji ocena, treba ažurirati taj podatak. U suprotnom treba uneti novi podatak u bazu podataka. Rezultat operacije treba prikazati na stranici ispit.jsp tj. za tog studenta treba tabelarno prikazati sve njegove položene ispite, uključujući upravo evidentirani ispit, koji treba prikazati crvenom bojom fonta. U slučaju da student ne postoji, na istoj stranici treba obavestiti korisnika o tome. Ova stranica treba da ima i link ka stranici pocetna.jsp.

Izgled tabela u bazi *Ocene*:

Student		
#	Naziv kolone	Tip
1	indeks (PK)	int(11)
2	ime	varchar(15)
3	prezime	varchar(15)
4	godina	int (11)

Ispiti		
#	Naziv kolone	Tip
1	IDisp (PK)	int(11)
2	indeks (FK)	int (11)
3	sifra	varchar(8)
4	ocena	int(11)
5	datum	date

REŠENJE

U ovom zadatku za povezivanje na MySQL i za povezivanje sa bazom naziva *Ocene*, koristiće se klasa DB.java, iz zadatka 6.

Servlet DBServlet služi za kreiranje relacione baze podataka *Ocene*, a programski kod dat je u nastavku:

```
//source file: servlets/DBServlet.java

package servlets;

import java.io.*;
import util.DB;

import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class DBServlet extends HttpServlet {

    protected void doGet(HttpServletRequest request,
                                 HttpServletResponse response)
            throws ServletException, IOException {
        Statement stmt = null;
        Connection con=null;
        String poruka = "";
        try {
            con=DB.getInstance().getConnection();
            if(con==null){
               request.setAttribute("message", "Pokusajte kasnije");

               request.getRequestDispatcher("error.jsp").forward(request
               , response);
               return;
            }
            stmt = con.createStatement();
            try {
                stmt.executeUpdate("drop table Ispiti;");
            } catch (SQLException e) {
            } finally {
                stmt.close();
            }

            stmt = con.createStatement();
            try {
                stmt.executeUpdate("drop table Student;");
            } catch (SQLException e) {
            } finally {
                stmt.close();
            }
```

```
   stmt = con.createStatement();
   String query = "create table Student (" +
           "indeks INTEGER PRIMARY KEY," +
           "ime text(15)," +
           "prezime text(15)," +
           "godina INTEGER);";

   stmt.executeUpdate(query);
   query = "create table Ispiti (" +
           "IDisp INTEGER auto_increment PRIMARY KEY," +
           "indeks INTEGER NOT NULL," +
           "sifra text(8)," +
           "ocena INTEGER NOT NULL," +
           "datum DATE," +
           "FOREIGN KEY (indeks) references Student(indeks));";
   stmt.executeUpdate(query);

query = "insert into Student (indeks, ime, prezime, godina)
         values ";
stmt.executeUpdate(query+"(20110123,'Drazen','Draskovic',4);");
stmt.executeUpdate(query+"(20120321,'Sanja','Delcev',3);");
stmt.close();

PreparedStatement ps = con.prepareStatement("insert into Ispiti
      (indeks, sifra, ocena, datum) values (?,?,?,?);");
   ps.setInt(1, 20110123);
   ps.setString(2, "SI3AR1");
   ps.setInt(3, 9);
   ps.setDate(4, Date.valueOf("2014-05-15"));
   ps.executeUpdate();
   ps.setString(2, "SI4PIA");
   ps.setInt(3, 9);
   ps.setDate(4, Date.valueOf("2015-06-23"));
   ps.executeUpdate();
   ps.setString(2, "SI3OS2");
   ps.setInt(3, 10);
   ps.setDate(4, Date.valueOf("2014-09-15"));
   ps.executeUpdate();
   ps.setString(2, "SI4MIPS");
   ps.setInt(3, 8);
   ps.setDate(4, Date.valueOf("2015-01-20"));
   ps.executeUpdate();

   ps.setInt(1, 20120321);
   ps.setString(2, "SI2RM1");
   ps.setInt(3, 10);
   ps.setDate(4, Date.valueOf("2014-06-16"));
   ps.executeUpdate();
```

```
                ps.setString(2, "SI3ROI");
                ps.setInt(3, 9);
                ps.setDate(4, Date.valueOf("2015-06-20"));
                ps.executeUpdate();
                ps.setString(2, "SI3KDP");
                ps.setInt(3, 10);
                ps.setDate(4, Date.valueOf("2015-06-22"));
                ps.executeUpdate();

                ps.close();

               DB.getInstance().putConnection(con);
               request.getRequestDispatcher("StartServlet").forward(request,
                   response);
            } catch (SQLException sqle) {
                poruka = sqle.getLocalizedMessage();
                DB.getInstance().putConnection(con);
                request.setAttribute("message", poruka);
                request.getRequestDispatcher("error.jsp").forward(request,
                   response);
                sqle.printStackTrace();
            }
        }
}
```

Servlet DBServlet služi za kreiranje relacione baze podataka *Ocene*, a programski kod dat je u nastavku:

```
//veb stranica: index.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Sistem za evidenciju ocena</title>
  </head>
  <body>
    <h3>Polazna stranica</h3><hr/>
    <a href="dbservlet"><H4>Kreirajte bazu podataka</H4></a>
    <a href="StartServlet"><H4>Skip</H4></a>
  </body>
</html>



//izvorni fajl: servlets/StartServlet.java

package servlets;
```

```
import beans.Datum;
import java.io.IOException;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class StartServlet extends HttpServlet {

  protected void processRequest(HttpServletRequest request,
                                HttpServletResponse response)
          throws ServletException, IOException {
    ServletContext aplikacija = this.getServletContext();
    aplikacija.setAttribute("datum", new Datum());
    response.sendRedirect("pocetna.jsp");
  }

  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
          throws ServletException, IOException {
    processRequest(request, response);
  }

  protected void doPost(HttpServletRequest request,
                        HttpServletResponse response)
          throws ServletException, IOException {
    processRequest(request, response);
  }
}
```

//veb stranica: pocetna.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE HTML>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Sistem za evidenciju ocena</title>
  </head>



  <body>
    <font color="red">${messageprosek}</font>
```

```
<form action="prosek" method="POST">
  Broj indeksa: <input type="text" name="indeks"
                value="${student.indeks}"/><br/><br/>
  <input type="submit" value="Nadji prosek"/><br/>
</form>
<br/><hr>
<font color="red">${messagedat}</font>

<form action="ocene" method="POST">
  <table>
    <tr><td>Prvi datum: </td>
      <td>
        <select name="dan1">
          <c:forEach begin="1" end="31" step="1" var="i">
            <c:if test="${i == datum.dan1}">
              <option  value="${i}" selected>${i}</option>
            </c:if>
            <c:if test="${i != datum.dan1}">
              <option  value="${i}">${i}</option>
            </c:if>
          </c:forEach>
        </select>
      </td>
      <td>
        <select name="mesec1">
          <c:forEach begin="1" end="12" step="1" var="i">
            <c:if test="${i == datum.mesec1}">
              <option  value="${i}" selected>${i}</option>
            </c:if>
            <c:if test="${i != datum.mesec1}">
              <option  value="${i}">${i}</option>
            </c:if>
          </c:forEach>
        </select>
      </td>
      <td>
        <select name="god1">
          <c:forEach begin="2000" end="2016" step="1" var="i">
            <c:if test="${i == datum.godina1}">
              <option  value="${i}" selected>${i}</option>
            </c:if>
            <c:if test="${i != datum.godina1}">
              <option  value="${i}">${i}</option>
            </c:if>
          </c:forEach>

        </select>
      </td>
    </tr>
```

```
    <tr><td>Drugi datum: </td>
      <td>
        <select name="dan2">
          <c:forEach begin="1" end="31" step="1" var="i">
            <c:if test="${i == datum.dan2}">
              <option  value="${i}" selected>${i}</option>
            </c:if>
            <c:if test="${i != datum.dan2}">
              <option  value="${i}">${i}</option>
            </c:if>
          </c:forEach>
        </select>
      </td>
      <td>
        <select name="mesec2">
          <c:forEach begin="1" end="12" step="1" var="i">
            <c:if test="${i == datum.mesec2}">
              <option  value="${i}" selected>${i}</option>
            </c:if>
            <c:if test="${i != datum.mesec2}">
              <option  value="${i}">${i}</option>
            </c:if>
          </c:forEach>
        </select>
      </td>
      <td>
        <select name="god2">
          <c:forEach begin="2000" end="2016" step="1" var="i">
            <c:if test="${i == datum.godina2}">
              <option  value="${i}" selected>${i}</option>
            </c:if>
            <c:if test="${i != datum.godina2}">
              <option  value="${i}">${i}</option>
            </c:if>
          </c:forEach>
        </select>
      </td>
    </tr>
  </table>
  <br/>
  <input type="submit" value="Pronadji ocene"/>
</form>

<br/><hr>

<font color="red">${messageisp}</font>

<form action="ispit" method="POST">
```

```
      <table>
        <tr>
          <td>Godina upisa:</td>
          <td>
            <input type="text" size="4" maxlength="4" name="godina"
             value="${unos.godina}"/>
          </td>
        </tr>
        <tr>
          <td>Broj indeksa:</td>
          <td>
            <input type="text" size="4" maxlength="4" name="broj"
             value="${unos.broj}"/>
          </td>
        </tr>
        <tr>
          <td>Šifra predmeta:</td>
          <td>
            <input type="text" size="6" maxlength="6" name="sifra"
             value="${unos.sifra}"/>
          </td>
        </tr>
        <tr>
          <td>Ocena:</td>
          <td>
            <select name="ocena">
              <c:forEach begin="5" end="10" step="1" var="i">
                <c:if test="${unos.ocena==i}">
                  <option selected value="${i}">${i}</option>
                </c:if>
                <c:if test="${unos.ocena!=i}">
                  <option value="${i}">${i}</option>
                </c:if>
              </c:forEach>
            </select>
          </td>
        </tr>
      </table>
      <input type="submit" value="Unesi ocenu"/>
    </form>
  </body>
</html>
```

//veb stranica: greska.jsp

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

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Error Page</title>
  </head>
  <body>
    <h2>Dogodila se greška!</h2>
    ${message}
  </body>
</html>
```

//izvorni fajl: beans/Student.java

```
package beans;
public class Student {
  private String ime;
  private String indeks;
  private String prosek;

  public String getIme() {
    return ime;
  }

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

  public String getIndeks() {
    return indeks;
  }

  public void setIndeks(String indeks) {
    this.indeks = indeks;
  }

  public String getProsek() {
    return prosek;
  }

  public void setProsek(String prosek) {
    this.prosek = prosek;
  }
}
```

Klasa Datum je pomoćna klasa za određivanje podrazumevanih vrednosti pri izboru datuma na veb stranici pocetna.jsp. Ona nema set metode.

//izvorni fajl: beans/Datum.java

```
package beans;
import java.util.Calendar;

public class Datum {
  private String dan1 = "1";
  private String mesec1;
  private String godina1;
  private String dan2;
  private String mesec2;
  private String godina2;

  public Datum() {
    Calendar c = Calendar.getInstance();
    dan2 = Integer.toString(c.get(Calendar.DATE));
    mesec1 = mesec2 = Integer.toString(c.get(Calendar.MONTH) + 1);
    godina1 = godina2 = Integer.toString(c.get(Calendar.YEAR));
  }

  public Datum(String a, String b, String c, String d, String e, String f){
    dan1 = a;
    mesec1 = b;
    godina1 = c;
    dan2 = d;
    mesec2 = e;
    godina2 = f;
  }

  public String getDan1() {
    return dan1;
  }

  public String getMesec1() {
    return mesec1;
  }

  public String getGodina1() {
    return godina1;
  }

  public String getDan2() {
    return dan2;
  }

  public String getMesec2() {
    return mesec2;
  }

  public String getGodina2() {
    return godina2;
```

```
  }
}

//izvorni fajl: beans/Ocena.java
package beans;

public class Ocena {

  private String sifra;
  private String ocena;
  private String datum;

  public String getSifra() {
    return sifra;
  }

  public void setSifra(String sifra) {
    this.sifra = sifra;
  }

  public String getOcena() {
    return ocena;
  }

  public void setOcena(String ocena) {
    this.ocena = ocena;
  }

  public String getDatum() {
    return datum;
  }

  public void setDatum(String datum) {
    this.datum = datum;
  }

}

//izvorni fajl: beans/Ocene.java
package beans;

public class Ocene {
  private String ime;
  private String prezime;
  private String sifra;
  private String ocena;
  public Ocene(String ime, String prezime, String sifra, String ocena) {
    this.ime = ime;
    this.prezime = prezime;
```

```
    this.sifra = sifra;
    this.ocena = ocena;
  }

  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 String getSifra() {
    return sifra;
  }

  public void setSifra(String sifra) {
    this.sifra = sifra;
  }

  public String getOcena() {
    return ocena;
  }

  public void setOcena(String ocena) {
    this.ocena = ocena;
  }
}

//izvorni fajl: beans/Unos.java
package beans;

public class Unos {
  private String sifra;
  private int ocena;
  private String broj;
  private String godina;

  public String getSifra() {
    return sifra;
  }
```

```
  public void setSifra(String sifra) {
    this.sifra = sifra;
  }

  public int getOcena() {
    return ocena;
  }

  public void setOcena(int ocena) {
    this.ocena = ocena;
  }

  public String getBroj() {
    return broj;
  }

  public void setBroj(String broj) {
    this.broj = broj;
  }

  public String getGodina() {
    return godina;
  }

  public void setGodina(String godina) {
    this.godina = godina;
  }
}

//izvorni fajl: servlets/ProsekServlet.java
package servlets;

import beans.Student;
import java.io.IOException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import util.DB;


public class ProsekServlet extends HttpServlet {

  protected void doPost (HttpServletRequest request,
```

```
                    HttpServletResponse response)
      throws ServletException, IOException {
Connection con = null;
Statement stmt = null;
String poruka = null;
String address = "prosek.jsp";
String ind = request.getParameter("indeks");
if (ind.isEmpty()) {
  poruka = "Niste uneli indeks!";
}
int indeks = 0;
Student s = new Student();
s.setIndeks(ind);
request.setAttribute("student", s);
try {
  indeks = Integer.parseInt(ind);
} catch (NumberFormatException nfe) {
  if (poruka == null) {
    poruka = "Indeks mora biti u formatu ggggbbbb!";
  }
  request.setAttribute("messageprosek", poruka);
  request.getRequestDispatcher("pocetna.jsp").forward(request,
        response);
  return;
}

if (indeks < 19500000 || indeks > 20160000) {
  request.setAttribute("messageprosek",
                       "Indeks mora biti u formatu ggggbbbb!");
  request.getRequestDispatcher("pocetna.jsp").forward(request,
        response);
  return;
}

try {
  con = DB.getInstance().getConnection();
  if (con == null) {
    request.setAttribute("message", "Pokusajte kasnije");
    request.getRequestDispatcher("greska.jsp").forward(request,
        response);
    return;
  }

  stmt = con.createStatement();
  String upit = "select ime, prezime, avg (ocena) as prosek
              from ispiti i, student s where i.indeks=" + indeks +
              " and i.indeks=s.indeks GROUP BY ime, prezime;";
  ResultSet rs = stmt.executeQuery(upit);
  if (rs.next()) {
```

```
          String ime = rs.getString("ime");
          String prezime = rs.getString("prezime");
          s.setIme(ime + " " + prezime);
          String prosek = rs.getString("prosek");
          s.setProsek(prosek);
          request.setAttribute("imastudent", true);
        } else {
          request.setAttribute("message", "Ne postoji student sa indeksom " +
                    s.getIndeks());
        }
        stmt.close();
        DB.getInstance().putConnection(con);

      } catch (SQLException sqle) {
        poruka = sqle.getLocalizedMessage();
        request.setAttribute("message", poruka);
        address = "greska.jsp";
        DB.getInstance().putConnection(con);
      }
      request.getRequestDispatcher(address).forward(request, response);
    }
}

//veb stranica: prosek.jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Prosek</title>
  </head>
  <body>
    <c:if test="${imastudent}">
      Student: ${student.ime}<br/>
      Indeks: ${student.indeks}<br/>
      Prosek: ${student.prosek}<br/>
    </c:if>

    <h2>${message}</h2>

    <br/>
    <a href="pocetna.jsp">Nazad</a>
  </body>
</html>

//izvorni fajl: servlets/OceneServlet.java
package servlets;
```

```
import beans.Datum;
import beans.Ocene;
import java.io.IOException;
import java.sql.Connection;
import java.sql.Date;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import util.DB;

public class OceneServlet extends HttpServlet {

  protected void doPost(HttpServletRequest request,
                        HttpServletResponse response)
          throws ServletException, IOException {

    String dan1 = request.getParameter("dan1");
    String dan2 = request.getParameter("dan2");
    String mesec1 = request.getParameter("mesec1");
    String mesec2 = request.getParameter("mesec2");
    String god1 = request.getParameter("god1");
    String god2 = request.getParameter("god2");
    Datum d = new Datum(dan1, mesec1, god1, dan2, mesec2, god2);
    request.getSession().setAttribute("datum", d);

    String datum1 = god1 + "-" + mesec1 + "-" + dan1;
    String datum2 = god2 + "-" + mesec2 + "-" + dan2;
    Date d1 = null;
    Date d2 = null;
    String address = "pocetna.jsp";

    try {
      d1 = Date.valueOf(datum1);
      d2 = Date.valueOf(datum2);
    } catch (IllegalArgumentException ile) {
      request.setAttribute("messagedat", "Nepravilan format datuma!");
      request.getRequestDispatcher(address).forward(request, response);
    }


    if (d1.after(d2)) {
      request.setAttribute("messagedat", "Prvi datum je posle drugog!");
      request.getRequestDispatcher(address).forward(request, response);
```

```
    }

    Connection con = null;
    Statement stmt = null;
    String poruka = "";
    address = "ocene.jsp";

    try {
      con = DB.getInstance().getConnection();
      if (con == null) {
        request.setAttribute("message", "Pokusajte kasnije");
        request.getRequestDispatcher("greska.jsp").forward(request,
            response);
        return;
      }
      stmt = con.createStatement();
      String upit = "select ime, prezime, sifra, ocena from student s,
                  ispiti i where s.indeks=i.indeks and i.datum between '" +
                  d1 + "' and '" + d2 + "';";
      ResultSet rs = stmt.executeQuery(upit);
      List<Ocene> ocene = new ArrayList<>();
      while (rs.next()) {
        String ime = rs.getString("ime");
        String prezime = rs.getString("prezime");
        String sifra = rs.getString("sifra");
        String ocena = rs.getString("ocena");
        Ocene o = new Ocene(ime, prezime, sifra, ocena);
        ocene.add(o);
      }

      request.setAttribute("ocene", ocene);
      stmt.close();
      DB.getInstance().putConnection(con);

    } catch (SQLException sqle) {
      poruka = sqle.getLocalizedMessage();
      request.setAttribute("message", poruka);
      address = "greska.jsp";
      DB.getInstance().putConnection(con);
    }

    request.getRequestDispatcher(address).forward(request, response);
  }
}
```

```
//veb stranica: ocene.jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Ocene</title>
  </head>
  <body>
    <c:if test="${not empty ocene}">
      Ocene studenata: <br/>
      <table>
        <tr><th>Ime</th><th>Prezime</th><th>Šifra</th><th>Ocena</th></tr>
            <c:forEach items="${ocene}" var="i">
          <tr>
            <td>${i.ime}</td>
            <td>${i.prezime}</td>
            <td>${i.sifra}</td>
            <td>${i.ocena}</td>
          </tr>
        </c:forEach>
      </table>
    </c:if>
    <c:if test="${empty ocene}">
      Ne postoje ocene za zadati kriterijum.
    </c:if>
    <br/><br/>
    <a href="pocetna.jsp">Nazad</a>
  </body>
</html>
```

```
//izvorni fajl: servlets/IspitServlet.java
package servlets;

import beans.Ocena;
import beans.Unos;
import java.io.IOException;
import java.sql.Connection;
import java.sql.Date;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import util.DB;
```

```
public class IspitServlet extends HttpServlet {

  protected void doPost (HttpServletRequest request,
                         HttpServletResponse response)
          throws ServletException, IOException {

    String godina = request.getParameter("godina");
    String broj = request.getParameter("broj");
    String sifra = request.getParameter("sifra");
    int ocena = Integer.parseInt(request.getParameter("ocena"));
    String address = "pocetna.jsp";
    int indeks = 0;
    HttpSession sesija = request.getSession();
    Unos u = new Unos();
    u.setBroj(broj);
    u.setGodina(godina);
    u.setSifra(sifra);
    u.setOcena(ocena);
    sesija.setAttribute("unos", u);

    if (godina.isEmpty() || broj.isEmpty() || sifra.isEmpty()) {
      request.setAttribute("messageisp", "Niste popunili sva polja!!!");
      request.getRequestDispatcher(address).forward(request, response);
      return;
    }
    try {
      int god = Integer.parseInt(godina);
      int br = Integer.parseInt(broj);
      if (god < 10) {
        god += 2000;
      } else if (god < 100) {
        god += 1900;
      }
      indeks = br + 10000 * god;
    } catch (NumberFormatException nfe) {
      request.setAttribute("messageisp",
                  "Broj indeksa i godina upisa moraju biti pravilni!");
      request.getRequestDispatcher(address).forward(request, response);
      return;
    }

    if (indeks < 19500000 || indeks > 20160000) {
      request.setAttribute("messageisp", "Nepravilan broj indeksa!");
      request.getRequestDispatcher(address).forward(request, response);
      return;
    }

    Connection con = null;
```

```
Statement stmt = null;
String poruka = "";
address = "ispit.jsp";

try {
  con = DB.getInstance().getConnection();
  if (con == null) {
    request.setAttribute("message", "Pokusajte kasnije");
    request.getRequestDispatcher("greska.jsp").forward(request,
        response);
  }
  stmt = con.createStatement();

  String upit = "select * from student where indeks=" + indeks + ";";
  ResultSet rs = stmt.executeQuery(upit);
  if (!rs.next()) {
    stmt.close();
    DB.getInstance().putConnection(con);
    request.setAttribute("messageisp",
                    "Ne postoji student sa datim brojem indeksa!");
    request.getRequestDispatcher("pocetna.jsp").forward(request,
              response);
    return;
  }

  Date datum = new Date(Calendar.getInstance().getTimeInMillis());

  upit = "select ocena from ispiti where indeks=" + indeks +
         " and sifra='" + sifra + "';";
  rs = stmt.executeQuery(upit);
  if (rs.next()) {
    upit = "update ispiti set ocena=" + ocena + " where indeks=" +
            indeks + " and sifra='" + sifra + "';";
    stmt.executeUpdate(upit);
    upit = "update ispiti set datum='" + datum + "' where indeks=" +
            indeks + " and sifra='" + sifra + "';";
    stmt.executeUpdate(upit);
    request.setAttribute("messageisp", "Ocena je uspesno promenjena.");

  } else {
    upit = "insert into ispiti (indeks, sifra, ocena, datum) values"
            + " (" + indeks + ",'" + sifra + "'," + ocena + ",'" +
              datum + "');";
    stmt.executeUpdate(upit);
    request.setAttribute("messageisp", "Ocena je uspesno dodata.");
  }

  upit = "select ocena, sifra, datum from ispiti where indeks=" +
          indeks + " order by ocena desc;";
```

```
      rs = stmt.executeQuery(upit);
      List<Ocena> ocene1 = new ArrayList<>();
      int i = 0, j = 0;

      while (rs.next()) {
        Ocena o = new Ocena();
        o.setOcena(rs.getString("ocena"));
        o.setSifra(rs.getString("sifra"));
        o.setDatum(rs.getString("datum"));
        ocene1.add(o);
        if (o.getSifra().equals(sifra)) {
          j = i;
        }
        i++;
      }

      request.setAttribute("ocene1", ocene1);
      request.setAttribute("crveno", Integer.toString(j));
      stmt.close();
      DB.getInstance().putConnection(con);
      sesija.removeAttribute("unos");

    } catch (SQLException sqle) {
      poruka = sqle.getLocalizedMessage();
      request.setAttribute("message", poruka);
      address = "greska.jsp";
      DB.getInstance().putConnection(con);
    }

    request.getRequestDispatcher(address).forward(request, response);
  }
}
```

//veb stranica: ispit.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Unos ocena</title>
  </head>
  <body>
    <h3>${messageisp}</h3>
    <table border="2">
      <tr><th>Šifra ispita</th><th>Ocena</th><th>Datum ispita</th></tr>
          <c:forEach items="${ocene1}" var="i" varStatus="status">
            <c:choose>
                  <c:when test="${status.index==crveno}">
```

```
                    <tr style="color:red"> </c:when>
            <c:otherwise> <tr></c:otherwise>
            </c:choose>
            <td>${i.sifra}</td>
            <td>${i.ocena}</td>
            <td>${i.datum}</td>
          </tr>
        </c:forEach>
    </table>
    <br/><br/>
    <a href="pocetna.jsp">Nazad</a>
  </body>
</html>
```

//XML fajl: WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
    <servlet>
        <servlet-name>baza</servlet-name>
        <servlet-class>servlets.DBServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>prosek</servlet-name>
        <servlet-class>servlets.ProsekServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ocene</servlet-name>
        <servlet-class>servlets.OceneServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ispit</servlet-name>
        <servlet-class>servlets.IspitServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>StartServler</servlet-name>
        <servlet-class>servlets.StartServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>baza</servlet-name>
        <url-pattern>/dbservlet</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>prosek</servlet-name>
        <url-pattern>/prosek</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
```

```
        <servlet-name>ocene</servlet-name>
        <url-pattern>/ocene</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ispit</servlet-name>
        <url-pattern>/ispit</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>StartServler</servlet-name>
        <url-pattern>/StartServlet</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```

4 JavaServer Faces tehnologija

4.1 Uvod u JavaServer Faces

JavaServer Faces (JSF) je poslednjih godina veoma popularna tehnologija koja se koristi u okviru Java internet aplikacija. Cilj ove tehnologije je automatizacija i jednostavnost izvršavanja najkorišćenijih operacija izrade ove vrste aplikacije. Njena glavna odlika je da se sastoji od sledećih delova:

- skup ugrađenih ulazno-izlaznih komponenti;
- događajima vođen (eng. *event driven*) programski model, sa opcijama za obradu i reagovanje na događaje;
- model komponenti, koji omogućava programerima serverske strane razvoj dodatnih komponenti.

Neke od JSF komponenti su jednostavne, kao na primer ulazna polja ili dugmad. Druge su dosta sofisticiranije, na primer tabele podataka i stabla. JSF sadrži sav potreban kod za obradu događaja i organizaciju komponenti. Aplikativni programeri mogu da zanemare sve nepotrebne detalje i da se usredsrede na sam razvoj aplikacione logike.

Takođe, treba napomenuti da je JSF danas deo Java EE standarda, što znači da je uključena u svaki Java EE aplikacioni server, i da se može jednostavno izvršavati u okviru veb servera, kao što je *Jakarta Tomcat*.

Ako se razmatra JSF arhitektura na visokom nivou može se primetiti da je ovaj frejmvork odgovoran za interakciju sa klijentima i sadrži alate koji omogućavaju zajedničko izvršavanje vizuelne prezentacije, aplikacione i poslovne logike za određenu internet aplikaciju. Ipak, primena JSF-a je najčešće ograničena na prezentacioni nivo. Perzistencija baze podataka, veb servisi i ostale pozadinske (eng. *back end*) konekcije su izvan oblasti kojom se bavi JSF.

Najvažniji servisi koje JSF obezbeđuje su:

- MVC arhitektura - JSF omogućava konekciju između pogleda i modela;
- konverzija podataka - JSF omogućava da se na jednostavan način specificiraju i primene potrebna pravila konverzije;
- validacija i obrada grešaka - JSF omogućava jednostavno povezivanje validacionih pravila za sve ulazne elemente;
- internacionalizacija - JSF sadrži razne opcije za selekciju pomoćnih resursa koje se koriste i kodiranje posebnih karaktera;
- dodatne komponente - mogu se razviti veoma kompleksne i složene komponente i dati na raspolaganje programerima i dizajnerima, koji mogu da ih na jednostavan način prikažu na stranici;

- alternativni prikazi - definisano ponašanje JSF je da generiše tagove za HTML stranice. Ali veoma je jednostavno nadograditi JSF frejmvork da proizvodi oznake za neki drugi opisni jezik, kao što je WML, XUL,...

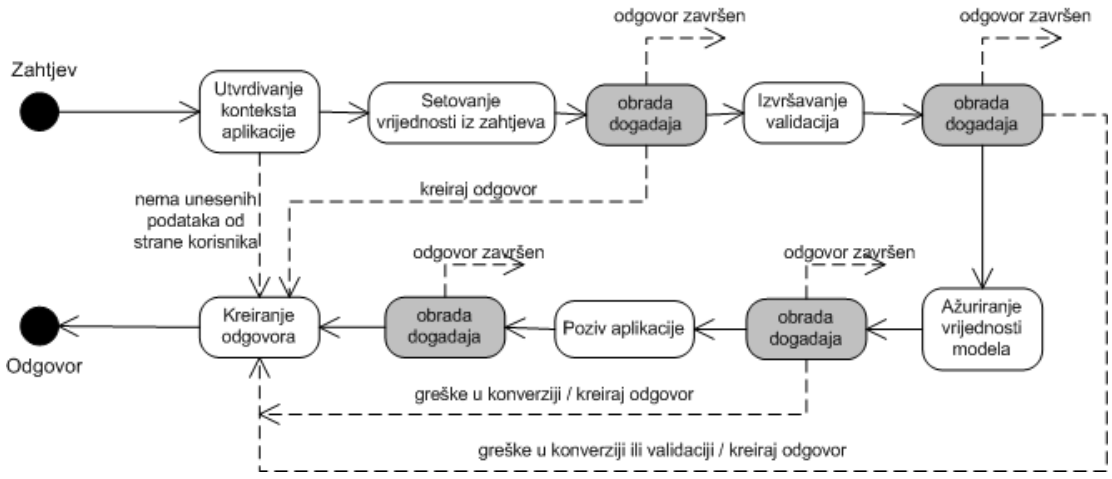
JSF je postao još popularniji frejmvork sa najznačajnijim unapređenjem kompletnog JSF okruženja koje je došlo sa pojavom verzije 2.0. U ovoj verziji ispravljeni su nedostaci otkriveni praktičnom primenom i uvedene su značajne promene u logici razvoja veb aplikacija. Specifična priroda unetih promena arhitekture za posledicu ima paralelnu egzistenciju dve verzije okruženja i nakon objavljivanja konačne specifikacije nove verzije. Određene osobine onemogućavaju jednostavno unapređenje postojećih aplikacija, pa i dalje postoje primene u kojima se starija verzija JSF okruženja zadržala kao dominantna. Zato su se autori ove knjige i odlučili da navedu najbitnije osobine i verzije 2 i prethodne verzije JSF okruženja.

Prilikom početka realizacije JSF aplikacije, da bi mogli da se koriste JSF tagovi svaka JSP strana na vrhu mora imati sledeće dve taglib direktive:

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
```

Prva uključuje html_basic, a druga jsf_core tagove. Tagovi html_basic predstavljaju kontrole HTML forme i druge osnovne HTML elemente i prikazuju podatke ili prihvataju podatke od korisnika (na primer: column, commandButton, commandLink, form, message, outputText, itd.). Tagovi jsf_core se koriste za ključne akcije koje su nezavisne od određenog render-a (na primer tagovi za rad sa događajima - npr. actionListener, tagovi za konverziju podataka - npr. converter, convertDateTime, convertNumber, tagovi za validaciju - npr. validator, validateLength, kao i loadBundle, param, subview, view itd.

JSF specifikacija definiše 6 različitih faza prilikom izvršavanja određenog poziva. Uobičajeni glavni tok izvršavanja je prikazan sa punim linijama, a alternativni tokovi su prikazani isprekidanim.



Slika 3 - Životni ciklus JSF aplikacije

Faza *Restore View* vraća stablo komponenti, za stranicu koja je ranije prikazivana, odnosno pravi novo stablo komponenti, ako se stranica prikazuje prvi put. Ako je stranica ranije prikazivana, sve komponente se postavljaju na početno stanje. To znači da JSF automatski postavlja informacije o traženoj formi. Na primer kada korisnik unese i pošalje određene nelegalne podatke, koji se neće propustiti na dalju obradu tokom njihove provere, ulazni podaci se korisniku ponovo prikazuju, tako da korisnik može da ih ispravi. Ako zahtev ne sadrži ulazne podatke, JSF implementacija prelazi na *Render Response* fazu. Ovo se događa kada se stranica prikazuje prvi put.

Sledeća faza je *Apply Request Values.* U okviru ove faze JSF implementacija prolazi kroz objekte komponenti u okviru stabla komponenti. Za svaki objekat komponenti se proverava koja mu vrednost pripada i ona mu se dodeljuje. U ovoj fazi se dodaju događaji vezani za dugme ili link u red događaja.

U okviru *Process Validation* poslati stringovi se prvo prebacuju u „lokalne“ vrednosti, koji mogu biti objekti bilo kog tipa. Kada se dizajnira JSF stranica, mogu se za klijentsku stranicu vezati validatori koji izvršavaju proveru korektnosti lokalnih vrednosti. Ako je validacija uspešna, JSF ciklus se nastavlja normalno. Kada se desi greška JSF implementacija poziva *Render Response* fazu direktno, prikazujući ponovo traženu stranicu, tako da se korisniku omogući korektan unos. Može se prikazati i odgovarajuća poruka, koja korisniku objašnjava zašto ponovo vidi iste podatke.

Kada se izvrše konverzije i validacije, pretpostavlja se da je bezbedno promeniti model podataka. Tokom *Update Model Values* faze, lokalne vrednosti se koriste da bi se promenile vrednosti u Bean-ovima povezanim sa komponentama.

U okviru *Invoke Application* faze izvršava se metod action() dugmeta ili linka koji je doveo do slanja forme. U ovom metodu se mogu izvršiti dodatne aplikacione obrade. Takođe, tu se generiše izlazni string, koji se šalje delovima odgovornim za navigaciju, gde se izvršava poziv nove stranice.

Na kraju, *Render Response* faza dekodira odgovor i šalje ga klijentu. Kada korisnik sa nove stranice pošalje novu formu, klikne na link ili na drugi način generiše novi zahtev, startuje se novi ciklus.

4.2 Java binovi

Prema *JavaBeans* specifikaciji, bin (eng. *bean*) je „softverska komponenta upotrebljiva više puta kojom je moguće manipulisati unutar razvojnog okruženja“. Unutar JSF okruženja, binovi se koriste da bi se odvojio prezentacioni sloj od sloja poslovne logike. Neke od osnovnih upotreba ovih komponenti su:

- komponente korisničkog interfejsa;

- objekti koji povezuju veb forme (eng. *backing beans*);
- objekti vezani za poslovnu logiku, čija svojstva (eng. *properties*) su prikazana na formi;
- spoljni izvori podataka koje treba konfigurisati pri sastavljanju aplikacije.

Klasa koja predstavlja bin mora da poštuje izvesna programska ograničenja i pravila da bi njene osobine bile javno dostupne za korišćenje od strane drugih alata. Najvažnije od tih osobina su svojstva (eng. *properties*). Svojstvo je svaki atribut klase koji ima: svoje ime, tip i metode za postavljanje i dobijanje vrednosti (*getter* i *setter*).

Imena i potpisi metoda se moraju pridržavati ove konvencije. Ako postoji samo prva metoda, onda se radi o *read-only* polju. Analogno tome, ako postoji samo druga metoda, radi se o *write-only* polju.

Mnoge JSF komponente korisničkog interfejsa sadrže atribut *value* kojim je moguće direktno specificirati vrednost, ili vezati komponentu za vrednost koja se dobija iz polja nekog bin-a. Zavisno od toga da li je u pitanju ulazna ili izlazna komponenta korisničkog interfejsa, na isti način je moguće i čitanje i upis vrednosti referenciranog polja.

```
<h:outputText value="#{userBean.ime}"/>
<h:inputText value="#{userBean.ime}"/>
```

Getter, u prvom slučaju, se poziva pri iscrtavanju komponente, dok se *setter*, u drugom primeru, poziva pri obradi odgovora.

U novijim JSF specifikacijama moguće je koristiti JSF EL (eng. *JSF Expression Language*) za ispisivanje vrednosti svojstava bin-ova, dok to prema ranijoj specifikaciji nije bio slučaj. Primer korišćenja outputText komponente u JSF 1.x i korišćenje JSF EL za ispis vrednosti svojstva bina:

```
// JSF 2.0
#{ userBean.usernameProperty }
// JSF 1.x
<h:outputText value="#{userBean.usernameProperty}"/>
```

Korišćenje outputText komponente u JSF 2.0 aplikacijama je i dalje omogućeno, s tim što se ona obavezno mora koristiti u slučaju kada ju je potrebno opciono prikazivati ili kada je potrebno ispisivati HTML kod u okviru JSF stranice.

Opseg važenja bin-a u okviru veb aplikacija određuje u kojem će segmentu aplikacije sam bin i podaci sadržani u njemu biti dostupni. Postoje tri osnovna opsega važenja: opseg aplikacije, opseg sesije i opseg zahteva.

Bin-ovi se najčešće kreiraju i deklarišu posebnom strukturom u okviru *faces-config.xml* fajla. Osim na ovaj način, bin-ove je moguće deklarisati u okviru *web.xml* fajla te u okviru bilo kojeg *faces-config.xml* fajla unutar neke od *jar* arhiva koje se koriste od strane aplikacije. XML struktura kojom se deklarišu bin-ovi može se prikazati sledećim primerom:

```
<managed-bean>
```

```
   <managed-bean-name>user</managed-bean-name>
   <managed-bean-class>UserBean</managed-bean-class>
   <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Pri ovakvom definisanju bin-ova, postoji i četvrta vrednost koja se može naći u okviru elementa opseg važenja (eng. *scope*). Ukoliko se neki bin deklariše sa vrednošću opsega none, to znači da se njegove vrednosti neće čuvati ni u jednom od ranije navedenih mapa opsega. Ovakvi bin-ovi se koriste kao gradivne klase pri kreiranju kompleksnih bin-ova. Ukoliko bin nije deklarisan sa ovom vrednošću opsega, tada je reč o upravljanom (eng. *managed*) bin-u.

Takođe, sa novom verzijom JSF okruženja uvedena je podrška za anotacije unutar bin-ova. Uvođenjem podrške za anotacije unutar bin-ova omogućava se deklarisanje opsega važenja upravljanih bin-ova i izvan datoteke *faces-config*. Kao što je naglašeno u ranijem tekstu upravljane bin-ove je i dalje moguće registrovati u datoteci *faces-config*:

```
// JSF 2.0
@ManagedBean(name="testBean")
@SessionScoped
public class TestBean {
   ...
}
// JSF 1.X
<managed-bean>
   <managed-bean-name>testBean</managed-bean-name>
   <managed-bean-class>TestBean</managed-bean-class>
   <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Upotrebom @ManagedBean anotacije na sledeći način:

```
@ManagedBean
public class TestBean{
   private String abc;
   ...
}
```

moguće je navođenje na klijentskoj strani elementa #{testBean.abc}, gde naziv bin-a odgovara nazivu klase, s tim što je prvo slovo naziva malo slovo. Kod ovako upotrebljene anotacije opseg zahteva je podrazumevani opseg. U datom primeru abc je naziv svojstva (koji ima odgovarajuće metode za dohvatanje vrednosti i postavljanje nove vrednosti) ili naziv metode. U slučajevima kada *action controller* metoda vraća string “xyz” i ako ne postoje eksplicitno definisana navigaciona pravila, onda je rezultujuća strana xyz.xhtml.

Takođe, prilikom upotrebe anotacije moguće je koristiti i name atribut @ManagedBean-a:

```
@ManagedBean(name="testBean2")
```

```
public class TestBean{
   private String abc;
   ...
}
```

U ovom slučaju na klijentskoj strani je moguće navesti: #{testBean2.abc}, gde je testBean2 vrednost name atributa. I dalje je opseg zahteva podrazumevani opseg.

Moguće je koristiti i atribut *eager* prilikom anotacije @ManagedBean-a:

```
@ManagedBean(eager=true)
@ApplicationScoped
public class TestBean{
   ...
   ...
}
```

Ovakav vid „decentralizacije“ programske logike olakšava dalje unapređenje aplikacije i smanjuje njenu kompleksnost.

U slučajevima kada se navede da je atribut „eager=true“ i opseg je aplikacija, tada bin mora biti kreiran u trenutku učitavanja aplikacije, a ne u trenutku prvog referenciranja bin-a, tj. pre opsluživanja bilo kog zahteva:

```
<managed-bean eager="true">
   <managed-bean-name>cbbhBean</managed-bean-name>
   <managed-bean-class>etf_primer.external.beans.ETFBean
            </managed-bean-class>
   <managed-bean-scope>application</managed-bean-scope>
</managed-bean>
```

Osim anotacije i postojećih opsega dostupnosti bin-ova (opseg aplikacije, sesije i zahteva), uvedeni su i novi opsezi: *View Scope*, *Flash Scope* i *Custom Scope*. Opseg dostupnosti *View* (ista instanca bin-a se koristi dok god je korisnik na istoj stranici, npr. u slučaju AJAX zahteva, trebao bi biti serijalizabilan) i *Flash Scope* upravljanih bin-ova veći je od opsega zahteva, a manji od opsega sesije. *Custom Scope* se specificira korišćenjem EL izraza, a ne ključne reči:

```
<managed-bean>
  <managed-bean-name>testBean</managed-bean-name>
  <managed-bean-class>TestBean</managed-bean-class>
  <managed-bean-scope>#{someCustomScope}</managed-bean-scope>
</managed-bean>
```

Bin je smešten u Map objekat i programer upravlja njegovim životnim vekom.

I u slučajevima defnisanja opsega važnosti moguće je koristiti anotaciju. Uobičajeno je da se smešta iza @ManagedBean:

```
@ManagedBean
@SessionScoped
```

```
public class TestBean {
   . . .
}
```

4.3 Navigacija

Navigacija predstavlja važnu komponentu čitavog JSF okruženja. Zavisno od akcije korisnika unutar veb čitača, vrši se njegovo dalje usmeravanje u okviru veb aplikacije. U zavisnosti od načina na koji se odlučuje prema kojoj strani će se usmeriti aplikacija, postoje dve vrste navigacije: statička i dinamička.

Nakon slanja podataka prema veb serveru od strane korisnika, aplikacija analizira unos, i na osnovu te analize određuje prema kojoj strani treba dalje preusmeriti aplikaciju. Komponenta koja je odgovorna za odlučivanje o toku aplikacije se naziva *navigation handler*.

Prilikom statičke navigacije klikom na određeni link ili dugme, kao rezultat se uvek dobija ista rezultujuća stranica, bez obzira na strukturu ili sadržaj korisničkog unosa.

Navigation handler kao parametar pri odlučivanju uzima vrednost parametra *action* koji je sastavni deo bilo kog dugmeta ili linka. Veza između ovakvih ulaznih parametara i imena rezultujućih stranica se specificira u vidu navigacionih pravila, koja se nalaze u fajlu *faces-config.xml*:

```
<h:commandButton label=”Prijavi me” action=”login”>
```

Navigaciona pravila se definišu u XML formatu i imaju fleksibilno definisanu strukturu, koja programeru znatno olakšava navigaciju između pojedinih strana aplikacije. Navigaciono pravilo se definiše na sledeći način:

```
<navigation-rule>
   <from-view-id>/index.jsp</from-view-id>
   <navigation-case>
         <from-outcome>login</from-outcome>
         <to-view-id>/welcome.jsp</to-view-id>
   </navigation-case>
</navigation-rule>
```

Ovakvom strukturom se u aplikaciji definiše pravilo po kome će se za akciju *login* izvršiti usmeravanje na stranicu welcome.jsp, ako se ta akcija desila na strani login.jsp.

Pažljivim izborom stringova koji označavaju akciju, može se izvršiti grupisanje nekoliko navigacionih pravila jednom strukturom, čime se pojednostavljuje snalaženje u kompleksnijim aplikacijama, koje mogu sadržati veliki broj navigacionih pravila.

```
<navigation-rule>
   <navigation-case>
```

```
            <from-outcome>logout</from-outcome>
            <to-view-id>/logout.jsp</to-view-id>
        </navigation-case>
    </navigation-rule>
```

U slučaju ovako specifikovanog navigacionog pravila, za rezultat *logout* će se izvršiti preusmeravanje na stranu *logout.jsp*, bez obzira na kojoj strani se korisnik trenutno nalazi, jer se unutar pravila ne nalazi element *from-view-id*. Analogno tome, ako pravilo sadrži više *navigation-case* elemenata, a samo jedan *from-view-id*, svi navedeni slučajevi će se uzimati u obzir samo ako su nastali u okviru tog elementa.

Bitno je napomenuti da grupisanje navigacionih pravila nije neophodno za ispravno funkcionisanje aplikacije, već da je dostupno zbog efikasnijeg razvoja.

Kod većine veb aplikacija, navigacija je dinamička. Tok ne zavisi samo od linka na koji korisnik klikne, već i od unestih podataka. Na primer, stranica na kojoj se vrši prijavljivanje na sistem, može imati dva rezultata: uspeh i neuspeh. Ovi rezultati zavise od obrade, tj. od toga da li je korisnik upisao odgovarajuće korisničko ime i šifru ili ne.

Kod implementiranja dinamičke navigacije nije potrebno vršiti bilo kakve izmene na navigacionim pravilima u odnosu na aplikacije sa statičkom navigacijom. Potreban preduslov je da navigaciona pravila sadrže sve moguće rezultate koje metoda za obradu unosa može proizvesti, da bi se time omogućilo aplikaciji da adekvatno odreaguje na bilo koji od tih rezultata.

Za razliku od statičke navigacije, *action* parametar u ovom slučaju sadrži izraz koji specifikuje određeni bin i metodu unutar njega koja je odgovorna za navigacionu logiku i obradu podataka.

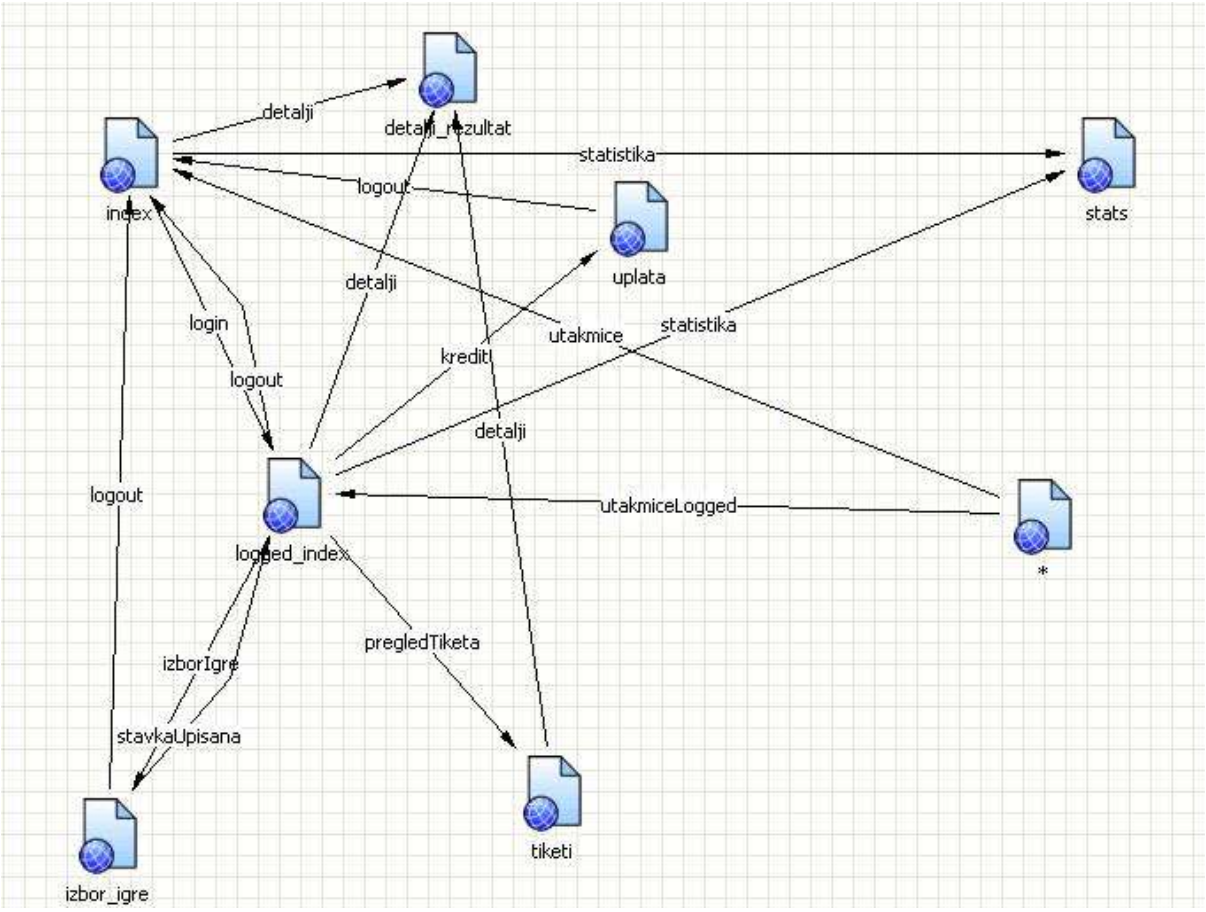
```
<h:commandButton label="Prijavi me" action="#{loginBean.prijava}"/>
```

U ovom konkretnom slučaju, *loginBean* predstavlja naziv bina koji sadrži metodu *prijava*. Sama metoda može izgledati ovako:

```
String prijava() {
   if(...) return „uspeh";
   else return „neuspeh";
}
```

Metoda vraća dva moguća rezultata na osnovu kojih se u navigacionim pravilima traže odgovarajuće vrednosti rezultujućih stranica. Ne postoji ograničenje u pogledu broja mogućih rezultata koji se mogu dobiti kao izlaz metode, kao što ne postoji ni ograničenje u pogledu broja navigacionih pravila unutar jedne aplikacije. Izbor konkretnog bin-a u kojem će se nalaziti referencirana metoda se ostavlja programeru, ali je uobičajeno da se metoda nalazi u klasi u kojoj se nalaze i podaci koji će biti potrebni za obradu.

Neka od razvojnih okruženja nude mogućnost grafičkog prikazivanja navigacionih pravila, kojima se omogućava njihovo kreiranje bez direktne manipulacije datotekom u kojoj se oni nalaze (*faces-config.xml*).



Slika 4 - Vizuelni prikaz navigacionih pravila u okruženju Eclipse IDE

Prethodno navedene tehnike navigacije su dovoljne za većinu aplikacija. Međutim, postoje još neki elementi koji se mogu pojaviti unutar *faces-config.xml* fajla:

- redirekcija - ako se navede *redirect* element iza *to-view-id* elementa, trenutni zahtev se prekida i šalje se novi HTTP *redirect* zahtev prema klijentu. URL sadržan u redirekciji govori koja je sledeća strana. Redirekcija je znatno sporiji postupak, jer je potreban jedan dodatni ciklus veb čitača. Međutim, redirekcija daje mogućnost čitaču da ažurira svoje adresno polje. Ukoliko se ne koristi redirekcija, URL koji se trenutno nalazi u adresnom polju veb čitača ostaje isti, što onemogućava korisniku dodavanje strana među omiljene;
- džoker znaci - moguće je koristiti i ovakav način zapisivanja imena strana unutar *from-view-id* elementa. Na primer, pravilo definisano elementom *<from-view-id>/secure/*</from-view-id>* se odnosi na sve strane čiji naziv počinje prefiksom */secure.* Dozvoljen je samo jedan džoker znak, koji se mora nalaziti na kraju stringa.

Sam algoritam navigacije ima tri ulaza:

- izlaz, odnosno vrednost *action* parametra ili string koji je rezultat poziva metode za obradu;

- *view ID* trenutne strane;
- akcija - doslovna vrednost *action* atributa koji je pokrenuo navigaciju.

Prva od dve faze navigacije je određivanje odgovarajućeg navigacionog pravila, koje se odvija kroz sledeće korake:

1. Ako je izlaz *null*, postupak se odmah prekida i prikazuje se trenutna strana.
2. Spajaju se sva navigaciona pravila koja imaju istu vrednost *from-view-id* parametra.
3. Traži se navigaciono pravilo čiji *from-view-id* parametar tačno odgovara *view ID* ulaznom parametru. Ako takvo pravilo postoji, ono se izvršava.
4. Razmatraju se sva pravila koja unutar *from-view-id* polja imaju džoker znak kao sufiks. Porede se ulaznim parametrima, i to tako što se džoker znak odstranjuje, a zatim se ulazni parametar poredi sa preostalim prefiksom. Ako postoji više pravila kod kojih je ovaj prefiks sadržan u ulaznom parametru *view ID*, uzima se ono kod koje je prefiks najduži.
5. Ako postoji pravilo bez *from-view-id* parametra, ono se izvršava.
6. Ako nije izvršena ni jedna od prethodnih stavki, prikazuje se trenutna stranica.

Druga od dve faze se sastoji od pregleda svih *navigation-case* elemenata unutar odabranog navigacionog pravila (koje se može sastojati od više spojenih pravila sa istim *from-view-id* vrednostima). Ova faza se sastoji od sledećih koraka:

1. Ako određeni slučaj ima i *from-outcome* i *from-action* parametre koji odgovaraju ulazu, ovo pravilo se izvršava.
2. Ako slučaj ima poklapanje samo između *from-outcome* parametra i vrednosti na ulazu, izvršava se ovo pravilo.
3. Ako slučaj ima poklapanje samo između *from-action* parametra i vrednosti na ulazu, izvršava se ovo pravilo.
4. Ako slučaj nema ni *from-outcome* ni *from-action* polja, pravilo se izvršava.

   Ako nije izvršena ni jedna od prethodnih stavki, ponovo se prikazuje trenutna stranica.

U novim verzijama JSF okruženja navigacija se može izvršiti i pomoću implicitnih navigacionih pravila. U tom slučaju, navigaciona pravila ne moraju biti eksplicitno navedena, već se na bazi rezultata *action controller* metoda implicitno mapira odgovarajući pogled (*view*) u formi fajla sa odgovarajućim imenom na fajl sistemu. Implicitno navigaciono pravilo se koristi u slučaju kada ne postoji eksplicitno navigaciono pravilo. Treba napomenuti, da i u ovom slučaju mogućnosti iz prethodne verzije ovim dodacima nisu izbačene, pa je odluka o vrsti navigacije koja se koristi ostavljena programeru:

```
public String test() {
   if (Math.random() < 0.5) {
         return("uspesno");      => vodi nas na stranicu uspesno.xhtml
   } else {
         return("neuspesno");    => vodi nas na stranicu neuspesno.xhtml
   }
}
```

Drugo poboljšanje navigacionog podsistema ogleda se u uslovnim navigacionim pravilima. Uslovna navigacija se implementira kao EL izraz korišćenjem novog <if> konfiguracionog elementa:

```
<navigation-rule>
    <display-name>page1.xhtml</display-name>
    <from-view-id>/page1.xhtml</from-view-id>
    <navigation-case>
          <from-outcome> uspesno</from-outcome>
          <to-view-id>/page2.xhtml</to-view-id>
          <if>#{testBean.nekiUslov}</if>
    </navigation-case>
</navigation-rule>
```

4.4 Rad sa pomoćnim fajlovima

Često je potrebno u okviru aplikacije menjati određene poruke ili natpise u okviru radne površine. Ranije se navedena operacija morala uraditi tako što bi se menjao kod određene klijentske stranice ili serverske komponente. Za više poruka, mora se proučavati i menjati kod više stranica. Posebne teškoće bi se desile ako je potrebno aplikaciju prebaciti na potpuno drugi jezik.

Zato se došlo do ideje da se tekstovi poruka i natpisi čuvaju u jednom posebnom tekstualnom fajlu. Promenom ovog fajla, automatski bi se promenili i tekstualni sadržaji cele aplikacije. Pomoćni fajl koji se koristi naziva se *properties* fajl i pomoću nekoliko koraka može se koristiti u okviru JSF aplikacije.

Prvi korak je kreirati fajl sa ektenzijom *.properties* u okviru direktorijuma WEB-INF/classes. Ovaj fajl će sadržati jednostavne parove imeKljuca=vrednost. Tako napravljeni fajl je moguće koristiti u okviru aplikacije pomoću f:loadBundle taga, kod koga atribut *basename* definiše osnovno ime fajla, a atribut *var* definiše promenljivu (koja je tipa Map), gde se sadrži rezultat. Pri pozivu fajla treba obratiti pažnju o fizičkoj lokaciji fajla, jer se pozicioniranje obavlja relativno u odnosu na WEB-INF/classes direktorijum, a i ekstenzija *.properties* se podrazumeva. Na primer, ako se želi pristupiti fajlu WEB-INF/classes/messages.properties treba na klijentskoj strani pozvati:

```
<f:loadBundle basename="messages" var="msgs"/>
```

Takođe za WEB-INF/classes/package1/test.properties potrebno je pozvati:

```
<f:loadBundle basename="package1.test" var="msgs"/>
```

Kada se na ovaj način ostvari konekcija sa *properties* fajlom, njegove vrednosti definisane pomoću imena ključa koriste izlazne poruke sa uobičajenjim EL konstrukcijama:

```
#{msgs.keyName}
```

Radi opštosti definisanih poruka moguće je u okviru njih definisati i parametre koji će dobiti vrednosti tokom samog pristupa određenoj poruci iz *properties* fajla. I u ovom slučaju kreira se *.properties* fajl u okviru WEB-INF/classes. Vrednosti poruka sadrže parametre {0}, {1}, {2}, ..., n.p., nekoIme= itd {0} itd {1}. Zatim je potrebno učitati fajl pomoću taga f:loadBundle kao i ranije, i pomoću atributa *basename* definisati osnovno ime fajla, a sa *var* definisati promenljivu koja će sadržati rezultat. Izlazne poruke na klijentskoj strani koriste h:outputFormat i vrednost definiše osnovnu poruku, dok ugnežđeni elementi f:param daju vrednosti za zamenu. Na primer:

```
<h:outputFormat value="#{msgs.nekoIme}">
      <f:param value="vrednost za 0ti ulaz"/>
      <f:param value="vrednost za1vi ulaz/>
</h:outputFormat>
```

4.5 Obrada događaja

U okviru veb aplikacija je često potrebno da se odgovori na korisničke akcije i događaje. Najčešće je potrebno da se u okviru komponente registruju metodi koji će obrađivati određeni događaj, na primer:

```
<h:selectOneMenu valueChangeListener="#{form.countryChanged}"...>
   ...
</h:selectOneMenu>
```

U prethodnom kodu, metod povezuje metod countryChanged koji je definisan u okviru bin forme. Sam poziv metoda će se desiti nakon korisnikove akcije selektovanja opcije iz elementa kombo liste.

JSF tehnologija podržava tri vrste događaja :

- Događaj izazvan promenom vrednosti (eng. *Value change events*)
- Događaj izazvan akcijom (eng. *Action events*)
- Fazni događaji (eng. *Phase events*)

Događaji koji su izazvani usled promene vrednosti su vezani za određene HTML elemente, kao što su h:inputText, h:selectOneRadio, i h:selectManyMenu, i to u slučajevima kada se vrednost komponente menja.

Događaji koji su izazvani usled akcije javljaju se nakon aktiviranja komandnih komponenti, kao što su elementi h:commandButton i h:commandLink, slučajevi kada je dugme ili link aktiviran.

Fazni događaji se aktiviraju u toku JSF životnog ciklusa.

Kao svi događaji koji su izazvani usled promene vrednosti, osluškivač prosleđuje vrednost događaju. Osluškivači koriste taj događaj kako bi pristupili vrednosti nove komponente. Klasa *ValueChangeEvent* nasleđuje klasu *FacesEvent*, a obe ove klase se nalaze u paketu javax.faces.event.

Metodi klase javax.faces.event.ValueChangeEvent koji mogu biti korisni su:

- UIComponent getComponent() - vraća ulaznu komponentu koju je okinuo događaj;
- Object getNewValue() - vraća novu vrednost komponente, nakon što je vrednost konvertovana i potvrđena;
- Object getOldValue() - vraća prethodnu vrednost komponente.

Klasa javax.faces.event.FacesEvent ima sledeće metode:

- void queue() - stavlja događaje u red, na kraju tekuće faze životnog ciklusa;
- PhaseId getPhaseId() - vraća fazni indentifikator u skladu sa fazom tokom koje je događaj isporučen;
- void setPhaseId(PhaseId) - postavlja fazni identifikator u skladu sa fazom tokom koje je događaj isporučen.

Događaji koji su izazvani usled akcije se javljaju nakon akitiviranja komandnih komponenti, kao što su h:commandButton i h:commandLink, kada su dugme ili link aktivirani. Oni se aktiviraju u toku faze buđenja aplikacije (eng. *Invoke Application* faza), negde pri kraju životnog ciklusa. Obično se vezuju osluškivači akcija za komandne komponente na klijentskoj stranici. Na primer, osluškivač koji se dodaje na link:

```
<h:commandLink actionListener="#{bean.linkActivated}">
   ...
</h:commandLink>
```

Kada se aktivira komanda ili link, okružujuća forma je prihvaćena i zatim JSF okruženje generiše akcioni događaj. Neophodno je razlikovati pojmove osluškivači akcija (eng. *action listener*) i same akcije (eng. *action*). U suštini, akcije su dizajnirane za logiku poslova i učestvuju u navigaciji upravljanja, dok osluškivači događaja izvode interfejsnu logiku i ne učestvuju u navigaciji upravljanja. Osluškivači događaja ponekad rade zajedno sa akcijama kada je akcijama potrebna informacija o korisničkom interfejsu.

Treba napomenuti da akcija ne može da implementira ponašanje, ali ona može da upravlja sa odgovarajućom stranicom, i ne može da određuje odgovarajuću stranicu, zato što ona ne zna ništa o dugmadima sa slikom (eng. Image button) iz korisničkog intefejsa ili o kliku miša.

Treba posebno obratiti pažnju na JSF okruženje koje uvek pokreće osluškivače akcija pre same akcije. Takođe, JSF insistira da se razdvoji korisnička i poslovna logika.

U prethodnom delu teksta demonstrirana je primena akcija i osluškivača koji su reagovali na promene vrednosti komponenti i označavali ih sa *actionListener* i *valueChangeListener* atributima.

U osnovnom JSF okruženju, pored njih, postoje i sledeći tagovi:

- f:actionListener
- f:valueChangeListener

Ovi tagovi su analogni actionListener i valueChangeListener atributima. Na primer:

```
<h:selectOneMenu value="#{form.country}" onchange="submit()"
valueChangeListener="#{form.countryChanged}">
<f:selectItems value="#{form.countryNames}"/>
</h:selectOneMenu>
```

Moguće je koristiti i tag f:valueChangeListener:

```
<h:selectOneMenu value="#{form.country}" onchange="submit()">
<f:valueChangeListener type=" CountryListener"/>
<f:selectItems value="#{form.countryNames}"/>
</h:selectOneMenu>
```

Prednost tagova u odnosu na atribute je to što oni dozvoljavaju da se vezuje više osluškivača na jednu komponentu.

JSF generiše i događaje, koji se zovu fazni događaji (eng. *phase events*), pre i posle svake faze životnog ciklusa. Ovim događajima upravljaju fazni osluškivači. Fazni događaji, za razliku od prethodna dva događaja, se specificiraju u konfiguracinom fajlu aplikacije, na sledeći način:

```
<faces-config>
   <lifecycle>
   <phase-listener>etf.PhaseTracker</phase-listener>
   </lifecycle>
</faces-config>
```

U navedenom fajlu se može dodati i više osluškivača, ali se oni bude onim redosledom kojim su navedeni u konfiguracionom fajlu.

Fazni osluškivači implementiraju interfejs *PhaseListener* koji se nalazi u paketu javax.faces.event. Taj interfejs ima sledeće tri metode:

- PhaseId getPhaseId() - govori implementaciji kako da dostavi fazne događaje osluškivaču; može da vrati vrednost PhaseId.APPLY_REQUEST_VALUES;
- void afterPhase(PhaseEvent) - poziva se jednom u životnom ciklusu, pre faze Apply Request Values;
- void beforePhase(PhaseEvent) - poziva se jednom u životnom ciklusu, posle faze Apply Request Values;

Fazni osluškivači su pogodni za debagovanje i specijalno ponašanje.

4.6 Ugrađena AJAX podrška

Današanje veb aplikacije, i pored popularnosti koje imaju, prilikom komunikacije sa korisnikom imaju i svojih nedostataka. Sam grafički korisnički interfejs (eng. GUI) može da bude spor, HTML se pokazuje kao odlično rešenje za statičke stranice, ali može biti neodgovarajuće rešenje za stranice poslovnih aplikacija, jer je suviše jednostavan i neefikasan za komunikaciju današnjih veb aplikacija.

Razlozi kao što su univerzalni pristup, već postojeći čitači na svakom računaru i automatsko ažuriranje aplikacije, jer sam sadržaj stiže od strane servera, i dalje predstavljaju dominantne faktore prilikom izbora same tehnologije izrade aplikacije.

AJAX (eng. *Asynchronous JavaScript and XML*) predstavlja novi način kako se mogu koristiti poznati standardi da bi se rešili navedeni problemi. Ova tehnologija omogućava da se efikasnije izvrše manje promene delova stranica, da sistem bude bolje sinhronizovan i da se izvršavaju promene delova bez ponovnog učitavanja celokupne stranice.

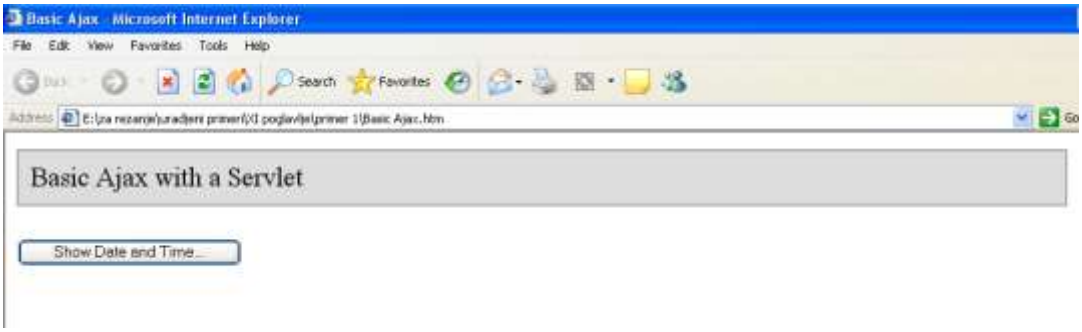
Na taj način AJAX postaje popularna tehnologija za kreiranje brzih i dinamičkih veb stranica. On dozvoljava stranicama da asinhrono menjaju svoj sadržaj razmenom manje količine podataka sa serverom, a da kranji korisnik toga nije svestan. Danas, veliki broj aplikacija koristi AJAX komponente, na primer Google Maps, Gmail, YouTube ili Facebook.

AJAX koristi sledeću kombinaciju tehnologija:

- XMLHttpRequest objekat (za asinhronu razmenu podataka sa serverom)
- JavaScript/DOM (da bi se prikazale i obrađivale informacije)
- CSS (za prikaz rezultata)
- XML (uobičajeno je da se koristi za prenos podataka)

Pomoću ovih tehnika izvršavanje AJAX koda je u tri koraka:

- pozivanje URL-a (generisanje zahteva) iz JavaScript koda na klijentskoj strani;
- obrada zahteva na serverskoj strani i slanje odgovora;
- nakon što je odgovor završen, integrisanje u DOM (Document Object Model).



Slika 5 - Prikaz stranice koja će AJAX tehnologijom da pročita serversko vreme i datum

Na slici je prikazan ekran aplikacije u okviru koga će se pomoću AJAX zahteva od strane servera dobiti trenutni datum i vreme. Dobijeni odgovor će se na klijentskoj strani integrisati u okviru iste pozivajuće stranice. Korisnik će dobiti promenjeno trenutno vreme, bez ponovnog učitavanja cele stranice.

Na klijentskoj strani je definisano dugme (pomoću taga commandButton, a ne pomoću submit) i prostor za upis rezultata pomoću DIV taga.

```
<h:commandButton type="button" value="#{msgs.buttonPrompt}"
onclick="showDateAndTime();" styleClass="button"/>

<div id="dateDIV" class="dateDiv"/>
```

Funkcija vezana za događaj *onClick*:

```
<script type="text/javascript" language="Javascript">
<!--
var xhr;

function showDateAndTime() {
   sendRequest("dateAndTime.ajax", // adresa
   processAjaxCall); // callback funkcija
}

function sendRequest(url, handler) {
   initXHR();
   xhr.onreadystatechange = handler;
   xhr.open("GET", url, true);
   xhr.send(null);
}

function initXHR() {
   if(window.XMLHttpRequest) {
         xhr = new XMLHttpRequest();
   }
   else if(window.ActiveXObject) {
         xhr = new ActiveXObject("Microsoft.XMLHTTP");
   }
```

```
}

function processAjaxCall() {
   if(xhr.readyState == 4) {
         if(xhr.status == 200) {
               var dateDiv = window.document.getElementById("dateDIV")
               dateDiv.innerHTML = xhr.responseText;
         }
   }
}
```

U datom kodu u okviru funkcije showDateAndTime kreira se objekat tipa XMLHttpRequest (u funkciji initXHR() se ispituje da li se radi o *Internet Explorer*-u verzije 7 ili nekoj starijoj verziji veb čitača, koji nema podršku za *XMLHttpRequest* objekte. U tom slučaju se kreira *ActiveXObject* objekat. Zatim se definiše takozvana *callback* funkcija, odnosno funkcija koja će se pozvati u trenutku kada stigne odgovor od strane servera. U ovom slučaju to je processAjaxCall() funkcija. Nakon definicije navedenih parametara šalje se zahtev ka serveru pomoću metode *open*, čiji argumenti su metod slanja podataka ka serveru (u ovom primeru se podaci ne šalju od klijenta ka serveru, tako da je svejedno koji se metod koristi, GET ili POST), definiše se akcija na serverskoj strani (u datom primeru to je servlet AjaxServlet) i da li je komunikacija asinhrona ili ne.

Kada server dobije zahtev od strane klijenta obrađuje ga, na primer sa sledećim kodom:

```
public class AjaxServlet extends HttpServlet {
   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
                     throws ServletException, IOException {
         response.setContentType("text/plain");
         response.setHeader("Cache-Control", "no-cache");
         response.setStatus(HttpServletResponse.SC_OK);
         response.getWriter().write(((new
                                    java.util.Date()).toString()));
   }
}
```

Servlet generiše odgovor na već standardan način. Taj odgovor se na klijentskoj strani obrađuje u okviru već pomenute *callback* funkcije. Data funkcija se poziva prilikom svake promene stanja zahteva. Zato se ispituje da li je zahtev došao u stanje 4 (oznaka za završenu obradu) i da li mu je u tom trenutku status jednak 200 (oznaka za uspešno obrađen zahtev). U tom slučajuj se prihvata odgovor od strane servera (rezultat koji je poslao servlet) i smešta u određenom delu stranice (u datom primeru to je deo stranice koji je označen DIV tagom i koji se zove dateDIV).

Podrška za AJAX komponente predstavlja značajno unapređenje u novoj verziji JSF specifikacije. JSF 1.x aplikacije su AJAX funkcionalnosti preuzimale iz velikog broja postojećih biblioteka koje su predstavljale nadogradnju osnovne arhitekture. S druge strane, JSF 2.x donosi integrisanu podršku za AJAX funkcionalnosti, najviše u domenu parcijalnog osvežavanja strane i sličnih aspekata asinhrone komunikacije:

```
<h:inputText value="#{testBean.text}">
   <f:ajax execute="@form" event="keyup" render="result"/>
</h:inputText>
...
<h:outputText value="#{testBean.text}" id="result"/>
```

Komponenta f:ajax ima sledeće atribute:

- render - id elemenata ili id-jevi liste elemenata koji treba da se osveže (izmeniti u DOM-u) nakon izvršavanja AJAX zahteva; često je to h:outputText element; specijalne vrednosti koje može dobiti su @this, @form, @none i @all - obično se ne koriste kao vrednosti render atributa;
- execute - predstavlja elemenete koji se šalju na serversku stranu radi procesiranja; često su u pitanju input elementi kao što je h:inputText ili h:selectOneMenu; specijalne vrednosti koje može dobiti su:
    - @this - element koji okružuje f:ajax - to je podrazumevana vrednost;
    - @form - h:form koji okružuje f:ajax - kada je potrebno da se pošalju vrednosti svih parametara forme;
    - @none - ništa se ne šalje;
    - @all – svi JSF UI elementi sa stranice;
- event - DOM event na koji se aktivira AJAX zahtev (npr. keyup, blur, ...);
- onevent - JavaScript funkcija koja se pokreće kada se “okida” event.

Pored ugrađene AJAX podrške, zadržana je i mogućnost integracije spoljašnjih komponenata iz bilo koje od biblioteka koje podržavaju novu specifikaciju. Zato je broj komponenata koje se mogu iskoristiti u novim aplikacijama u konstantnom porastu i već uveliko prelazi minimum koji je potreban za implementaciju čak i naprednijih aplikacija, iako se radi o novoj specifikaciji. Bitno je napomenuti da zbog prethodno navedenih razlika arhitektura sistema, postoje problemi u kompatibilnosti AJAX biblioteka razvijenih za starije aplikacije, u odnosu na zahteve koje postavlja nova specifikacija.

Zadaci iz JavaServer Faces tehnologije

Zadatak 1

a) Na početnoj JSF strani, napraviti formu za logovanje korisnika koja sadrži polja za unos kredencijala (korisničkog imena i lozinke). Kada korisnik unese svoje kredencijale, treba prikazati podatke o tom korisniku, tako što će se na drugoj JSF strani pozvati podaci iz kreiranog bean-a KorisnikBean.

b) Izmeniti početnu JSF stranu, tako da polja za unos kredencijala budu obavezna, sa ispisivanjem poruke (requiredMessage) ukoliko nedostaje korisničko ime i lozinka. U realizovanoj formi dodati još jedno polje za unos broja ulaznica koje želite da naručite. Primeniti validaciju, tako da korisnik može uneti najmanje 1, a najviše 8 ulaznica.

REŠENJE

```
//veb stranica: index.xhtml
<?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>Primer forme za logovanje korisnika</title>
    </h:head>
    <h:body>
        <h:form>
        <h3>Unesite vaše parametre za logovanje:</h3>
        <table>
            <tr>
                <td>Korisničko ime:</td>
                <td><h:inputText value="#{korisnik.username}"/></td>
            </tr>
            <tr>
                <td>Lozinka:</td>
                <td><h:inputSecret value="#{korisnik.password}"/></td>
            </tr>
```

```
        </table>
        <p>
            <h:commandButton value="Logovanje" action="dobrodosli"/>
        </p>
        </h:form>
    </h:body>
</html>

//izvorni fajl: beans/KorisnikBean.java
package beans;

import javax.inject.Named;
//ili import javax.faces.bean.ManagedBean;
import java.io.Serializable;
import javax.enterprise.context.SessionScoped;
//ili import.javax.faces.bean.SessionScoped;

@Named(value = "korisnik") //ili @Managedean(name="korisnik")
@SessionScoped
public class KorisnikBean implements Serializable{
    private String username;
    private String password;

    public String getUsername() {
        return username;
    }
    public void setUsername(String novaVrednost) {
        this.username = novaVrednost;
    }

    public String getPassword() {
        return password;
    }
    public void setPassword(String novaVrednost) {
        this.password = novaVrednost;
    }
}

//veb stranica: index.xhtml
<?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>
        <h2> Dobrodošli, #{korisnik.username} </h2>
```

```
        <h3> Ovo je JSF stranica! </h3>
    </h:body>
</html>

//Stavka b)
//nova veb stranica: index.xhtml
<?xml version
<?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>Primer forme za logovanje korisnika</title>
    </h:head>
    <h:body>
        <h:form>
        <h3>Unesite vaše parametre za logovanje:</h3>
        <table>
            <tr>
                <td>Korisničko ime:</td>
                <td><h:inputText value="#{korisnik.username}"
                                 requiredMessage="Obavezno polje za
                                                  korisničko ime"
                                 required="true"/>
                </td>
            </tr>
            <tr>
                <td>Lozinka:</td>
                <td><h:inputSecret value="#{korisnik.password}"
                                   requiredMessage="Obavezno polje za
                                                    lozinku"
                                   required="true"/>
                </td>
            </tr>
            <tr>
                <td>Unesite broj ulaznica koje želite:</td>
                <td><h:inputText requiredMessage="Morate uneti
                                                  broj ulaznica"
                                 validatorMessage="Morate uneti broj
                                                   izmedju 1 i 8"
                                 required="true">
                        <f:validateLongRange minimum="1" maximum="8"/>
                    </h:inputText>
                </td>
            </tr>
        </table>
        <p>
```

```
            <h:commandButton value="Logovanje" action="dobrodosli"/>
        </p>
        </h:form>
    </h:body>
```

Zadatak 2

Napravljena je veb aplikacija koja prikazuje kviz o poznavanju srpskih naučnika. Na početnoj JSF strani korisniku treba prikazati formu sa prikazanim pitanjem, tekstualnim poljem za unos odgovora i dugmetom za potvrdu odgovora. Ukoliko korisnik tačno odgovori na pitanje, treba mu prikazati sledeće pitanje i trenutni ukupni broj poena koje je osvojio na kvizu. Ukoliko korisnik netačno odgovori na pitanje, korisniku treba prikazati poruku sa greškom, i dati mu još jednu mogućnost da odgovori na to isto pitanje. Ukoliko korisnik i drugi put da pogrešan odgovor na pitanje, korisniku se prikazuje tačan odgovor pitanja koje nije znao, i prikazuje mu se novo pitanje. Kada korisnik prođe kroz sva pitanja iz kviza, korisniku treba prikazati ukupan broj osvojenih poena na kvizu, kao i mogućnost da odigra kviz ispočetka. Logika kompletne aplikacije treba da bude prikazana u binovima.

Realizovati i properties fajlove, tako da ova aplikacija podrži internacionalizaciju za engleski, nemački i srpski jezik.

REŠENJE

```
//veb stranica: index.xhtml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>#{por.naslovPoruka}</title>
   </h:head>
   <h:body>
      <h1>#{por.naslovPoruka}</h1>
```

```
      <h:form>
          <p>#{quizBean.pitanje}</p>
          <p><h:inputText value="#{quizBean.unetOdgovor}"/></p>
         <p>
             <h:commandButton value="#{por.proveriOdgovor}"
                              action="#{quizBean.proveraOdgovora}"/>
         </p>
      </h:form>
   </h:body>
</html>

//veb stranica: uspesno.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>#{por.naslovPoruka}</title>
   </h:head>
   <h:body>
      <h1>#{por.naslovPoruka}</h1>
      <h:form>
         <p>
            #{por.cestitamo}
            <h:outputFormat value="#{por.ukupno}">
                <f:param value="#{quizBean.poeni}"/>
            </h:outputFormat>
         </p>
          <p>#{por.sledeciProblem}</p>
         <p>#{quizBean.pitanje}</p>
         <p><h:inputText value="#{quizBean.unetOdgovor}"/></p>
         <p>
            <h:commandButton value="#{por.proveriOdgovor}"
                             action="#{quizBean.proveraOdgovora}"/>
         </p>
      </h:form>
   </h:body>
</html>

//veb stranica: neuspesno.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
```

```
      <title>#{por.naslovPoruka}</title>
   </h:head>
   <h:body>
      <h1>#{por.naslovPoruka}</h1>
      <h:form>
         <p>
            #{por.opetNetacanOdgovor}
            <h:outputFormat value="#{por.tacanOdgovor}">
                <f:param value="#{quizBean.odgovor}"/>
            </h:outputFormat>
         </p>
         <p>#{por.sledeciProblem}</p>
         <p>#{quizBean.pitanje}</p>
         <p><h:inputText value="#{quizBean.unetOdgovor}"/></p>
         <p>
            <h:commandButton value="#{por.proveriOdgovor}"
                             action="#{quizBean.proveraOdgovora}"/>
         </p>
      </h:form>
   </h:body>
</html>
```

//veb stranica: ponovo.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>#{por.naslovPoruka}"</title>
   </h:head>
   <h:body>
      <h1>#{por.naslovPoruka}</h1>
      <h:form>
         <p>#{por.netacanOdgovor}</p>
         <p>#{quizBean.pitanje}</p>
         <p><h:inputText value="#{quizBean.unetOdgovor}"/></p>
         <p>
            <h:commandButton value="#{por.proveriOdgovor}"
                             action="#{quizBean.proveraOdgovora}"/>
         </p>
      </h:form>
   </h:body>
</html>
```

```
//veb stranica: gotovo.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>#{por.naslovPoruka}</title>
   </h:head>
   <h:body>
      <h1>#{por.naslovPoruka}</h1>
      <h:form>
         <p>
            #{por.hvala}
            <h:outputFormat value="#{por.ukupno}">
               <f:param value="#{quizBean.poeni}"/>
            </h:outputFormat>
         </p>
         <p>
            <h:commandButton value="#{por.ponovoPokreni}"
                             action="#{quizBean.novaIgra}"/>
         </p>
      </h:form>
   </h:body>
</html>

//izvorni fajl: etf.kviz/Problem.java
package etf.kviz;

import java.io.Serializable;

public class Problem implements Serializable {
   private String pitanje;
   private String odgovor;

   public Problem(String pitanje, String odgovor) {
      this.pitanje = pitanje;
      this.odgovor = odgovor;
   }

   public String getPitanje() { return pitanje; }

   public String getOdgovor() { return odgovor; }

   public boolean isCorrect(String unetodgovor) {
      return unetodgovor.trim().equalsIgnoreCase(odgovor);
   }
}
```

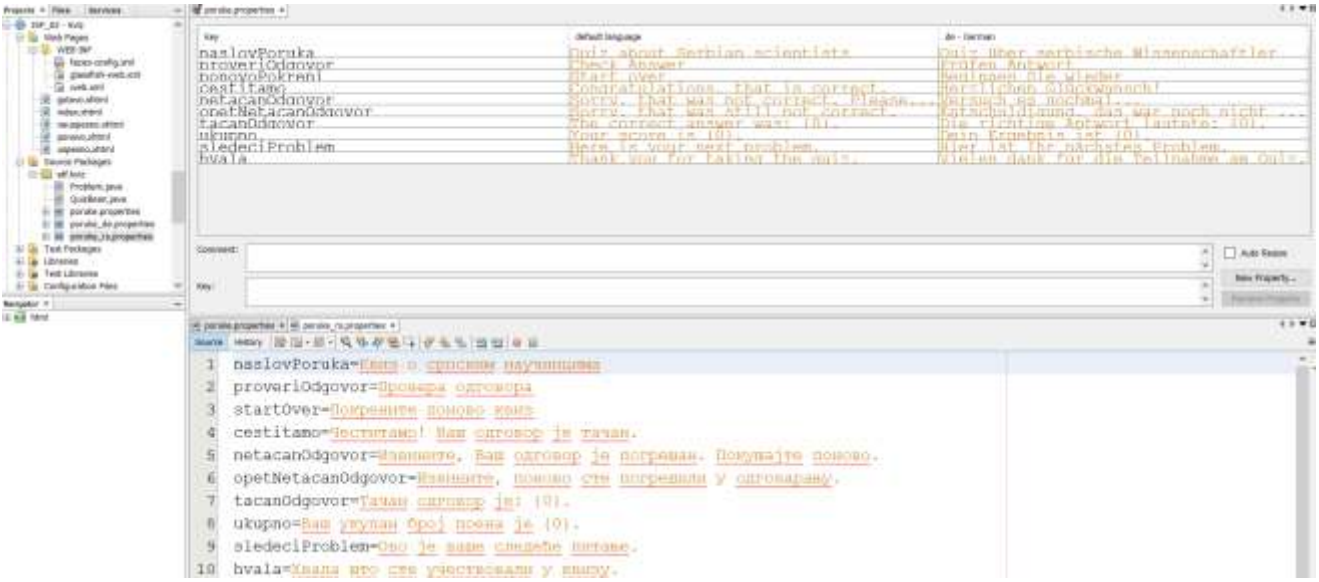
```
//izvorni fajl: etf.kviz/QuizBean.java
package etf.kviz;

import java.io.Serializable;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
// ili import javax.enterprise.context.SessionScoped;

@ManagedBean(name = "quizBean") // ili @Named
@SessionScoped
public class QuizBean implements Serializable {

    private int tekuciProblem;
    private int pokusaji;
    private int poeni;
    private String unetOdgovor = "";
    private String tacanOdgovor;

    // Ovaj deo je hardkodovan, u praksi treba iz baze podataka
    private ArrayList<Problem> problemi = new
ArrayList<Problem>(Arrays.asList(
            new Problem(
                    "Kako se zove jedini Srbin po kome je nazvana jedna
                        međunarodna jedinica mere?",
                    "Nikola Tesla"),
            new Problem(
                    "Koje godine je rodjen Nikola Tesla?",
                    "1856"),
            new Problem(
                    "Koji evropski grad ima muzej Nikole Tesle?",
                    "Beograd"),
            new Problem(
                    "Ko je autor dela \"Od pašnjaka do naučenjaka\"?",
                    "Mihajlo Pupin"),
            new Problem(
                    "Mesto rođenja Mihajla Pupina zove se?",
                    "Idvor"),
            new Problem(
                    "Koji univerzitet u SAD je nazavao laboratoriju za fiziku
u čast M.Pupina?",
                    "Kolumbija"),
            new Problem(
                    "Kako se zove Srbin tvorac najpreciznijeg kalendara
                        na svetu?",
                    "Milutin Milankovic")));
```

```
public String getPitanje() {
    return problemi.get(tekuciProblem).getPitanje();
}

public String getOdgovor() {
    return tacanOdgovor;
}

public int getPoeni() {
    return poeni;
}

public String getUnetOdgovor() {
    return unetOdgovor;
}

public void setUnetOdgovor(String noviOdgovor) {
    unetOdgovor = noviOdgovor;
}

public String proveraOdgovora() {
    pokusaji++;
    if (problemi.get(tekuciProblem).isCorrect(unetOdgovor)) {
        poeni++;
        sledeciProblem();
        if (tekuciProblem == problemi.size()) {
            return "gotovo";
        } else {
            return "uspesno";
        }
    } else if (pokusaji == 1) {
        return "ponovo";
    } else {
        sledeciProblem();
        if (tekuciProblem == problemi.size()) {
            return "gotovo";
        } else {
            return "neuspesno";
        }
    }
}
public String novaIgra() {
    Collections.shuffle(problemi);
    tekuciProblem = 0;
    poeni = 0;
    pokusaji = 0;
    unetOdgovor = "";
    return "ispocetka";
}
```

```
    private void sledeciProblem() {
        tacanOdgovor = problemi.get(tekuciProblem).getOdgovor();
        tekuciProblem++;
        pokusaji = 0;
        unetOdgovor = "";
    }
}
```

//izvorni fajl: poruke.properties

```
naslovPoruka=Quiz about Serbian scientists
proveriOdgovor=Check Answer
ponovoPokreni=Start over
cestitamo=Congratulations, that is correct.
netacanOdgovor=Sorry, that was not correct. Please try again!
opetNetacanOdgovor=Sorry, that was still not correct.
tacanOdgovor=The correct answer was: {0}.
ukupno=Your score is {0}.
sledeciProblem=Here is your next problem.
hvala=Thank you for taking the quiz.
```



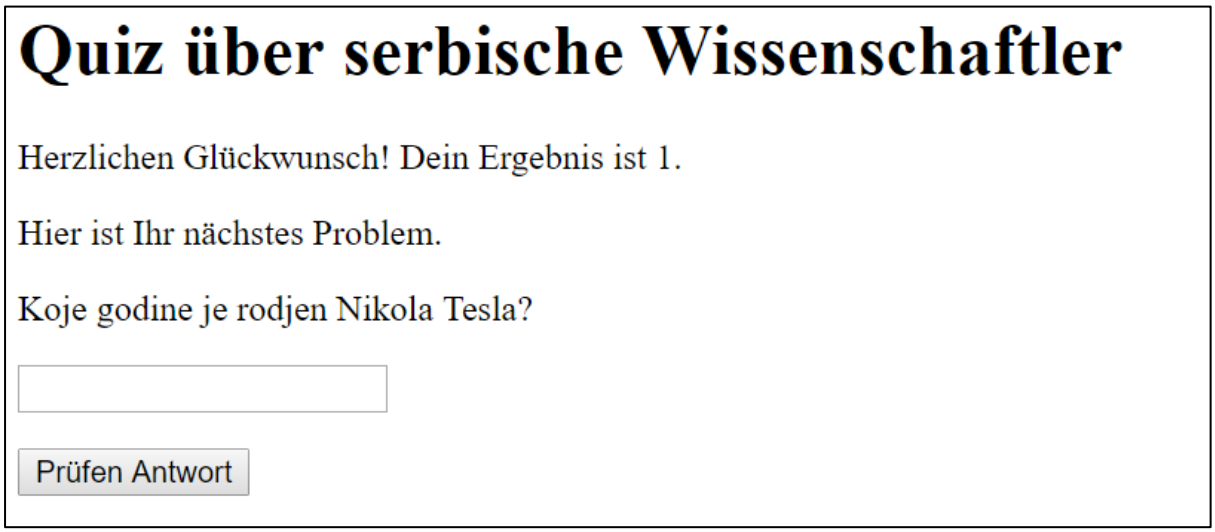
//izvorni fajl: poruke_rs.properties

```
naslovPoruka=Квиз о српским научницима
proveriOdgovor=Провера одговора
startOver=Покрените поново квиз
cestitamo=Честитамо! Ваш одговор је тачан.
netacanOdgovor=Извините, Ваш одговор је погрешан. Покушајте поново.
opetNetacanOdgovor=Извините, поново сте погрешили у одговарању.
tacanOdgovor=Тачан одговор је: {0}.
ukupno=Ваш укупан број поена је {0}.
sledeciProblem=Ово је ваше следеће питање.
hvala=Хвала што сте учествовали у квизу.
```

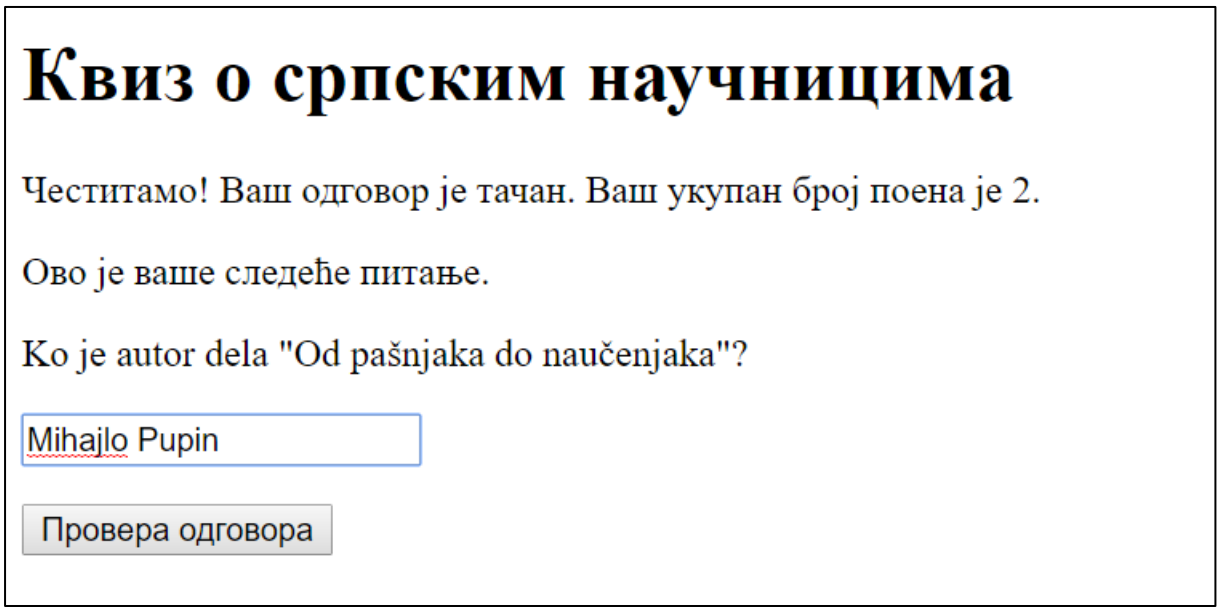
```
//XML fajl: WEB-INF/faces-config.xml
<?xml version="1.0"?>
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
   version="2.0">
   <navigation-rule>
      <navigation-case>
         <from-outcome>ispocetka</from-outcome>
         <to-view-id>/index.xhtml</to-view-id>
         <redirect/>
      </navigation-case>
   </navigation-rule>
   <navigation-rule>
      <from-view-id>/ponovo.xhtml</from-view-id>
      <navigation-case>
         <from-outcome>neuspesno</from-outcome>
         <to-view-id>/neuspesno.xhtml</to-view-id>
         <redirect/>
      </navigation-case>
   </navigation-rule>
   <navigation-rule>
      <navigation-case>
         <from-outcome>neuspesno</from-outcome>
         <to-view-id>/ponovo.xhtml</to-view-id>
         <redirect/>
      </navigation-case>
   </navigation-rule>
   <application>
      <resource-bundle>
         <base-name>etf.kviz.poruke</base-name>
         <var>por</var>
      </resource-bundle>
      <locale-config>
          <default-locale>en</default-locale>
          <supported-locale>de</supported-locale>
          <supported-locale>rs</supported-locale>
      </locale-config>
   </application>
   <navigation-rule>
        <from-view-id>/index.xhtml</from-view-id>
        <navigation-case>
            <from-outcome>case1</from-outcome>
            <to-view-id>/ponovo.xhtml</to-view-id>
            <redirect/>
        </navigation-case>
        <navigation-case>
            <from-outcome>case2</from-outcome>
```

```
            <to-view-id>/gotovo.xhtml</to-view-id>
            <redirect/>
        </navigation-case>
    </navigation-rule>
</faces-config>
```

Primer podržane lokalaziacije za nemački jezik:



Primer podržane lokalaziacije za srpski jezik:



Zadatak 3

Uraditi prethodni zadatak uz mogućnost da se pitanje u kvizu preskoči, klikom na link "Preskoči pitanje". Link treba da ima promenljive parametre, u zavisnosti do kog pitanja je korisnik stigao.

REŠENJE

```
//veb stranica: index.xhtml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
    <f:metadata>
        <f:viewParam name="q" value="#{quizBean.tekuciProblem}"/>
    </f:metadata>
    <h:head>
        <title>#{por.naslovPoruka}</title>
    </h:head>
    <h:body>
        <h1>#{por.naslovPoruka}</h1>
        <h:form>
            <p>#{quizBean.pitanje}</p>
            <p><h:inputText value="#{quizBean.unetOdgovor}"/></p>
            <p>
                <h:commandButton value="#{por.proveriOdgovor}"
                                 action="#{quizBean.proveraOdgovora}"/>
            </p>
            <p><h:link outcome="#{quizBean.preskoci}"
                       value="Preskoči pitanje">
                    <f:param name="q"
                             value="#{quizBean.tekuciProblem + 1}"/>
                </h:link>
            </p>
        </h:form>
    </h:body>
</html>


//veb stranica: uspesno.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
```

```
    <f:metadata>
        <f:viewParam name="q" value="#{quizBean.tekuciProblem}"/>
    </f:metadata>
    <h:head>
        <title>#{por.naslovPoruka}</title>
    </h:head>
    <h:body>
        <h1>#{por.naslovPoruka}</h1>
        <h:form>
            <p>
                #{por.cestitamo}
                <h:outputFormat value="#{por.ukupno}">
                    <f:param value="#{quizBean.poeni}"/>
                </h:outputFormat>
            </p>
            <p>#{por.sledeciProblem}</p>
            <p>#{quizBean.pitanje}</p>
            <p><h:inputText value="#{quizBean.unetOdgovor}"/></p>
            <p>
                <h:commandButton value="#{por.proveriOdgovor}"
                                 action="#{quizBean.proveraOdgovora}"/>
            </p>
            <p>
                <h:link outcome="#{quizBean.preskoci}"
                        value="Preskoči pitanje">
                    <f:param name="q"
                             value="#{quizBean.tekuciProblem + 1}"/>
                </h:link>
            </p>
        </h:form>
    </h:body>
</html>
```

//veb stranica: neuspesno.xhtml

```
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
    <f:metadata>
        <f:viewParam name="q" value="#{quizBean.tekuciProblem}"/>
    </f:metadata>
    <h:head>
        <title>#{por.naslovPoruka}</title>
    </h:head>
    <h:body>
        <h1>#{por.naslovPoruka}</h1>
        <h:form>
```

```
            <p>
                #{por.opetNetacanOdgovor}
                <h:outputFormat value="#{por.tacanOdgovor}">
                    <f:param value="#{quizBean.odgovor}"/>
                </h:outputFormat>
            </p>
            <p>#{por.sledeciProblem}</p>
            <p>#{quizBean.pitanje}</p>
            <p><h:inputText value="#{quizBean.unetOdgovor}"/></p>
            <p>
                <h:commandButton value="#{por.proveriOdgovor}"
                                 action="#{quizBean.proveraOdgovora}"/>
            </p>
            <p>
                <h:link outcome="#{quizBean.preskoci}"
                        value="Preskoči pitanje">
                    <f:param name="q"
                             value="#{quizBean.tekuciProblem + 1}"/>
                </h:link>
            </p>
        </h:form>
    </h:body>
</html>

//veb stranica: ponovo.xhtml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
    <f:metadata>
        <f:viewParam name="q" value="#{quizBean.tekuciProblem}"/>
    </f:metadata>
    <h:head>
        <title>#{por.naslovPoruka}"</title>
    </h:head>
    <h:body>
        <h1>#{por.naslovPoruka}</h1>
        <h:form>
            <p>#{por.netacanOdgovor}</p>
            <p>#{quizBean.pitanje}</p>
            <p><h:inputText value="#{quizBean.unetOdgovor}"/></p>
            <p>
                <h:commandButton value="#{por.proveriOdgovor}"
                                 action="#{quizBean.proveraOdgovora}"/>
            </p>
            <p>
                <h:link outcome="#{quizBean.preskoci}"
```

```
                        value="Preskoči pitanje">
                    <f:param name="q"
                             value="#{quizBean.tekuciProblem + 1}"/>
                </h:link>
            </p>
        </h:form>
    </h:body>
</html>

//veb stranica: gotovo.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>#{por.naslovPoruka}</title>
   </h:head>
   <h:body>
      <h1>#{por.naslovPoruka}</h1>
      <h:form>
         <p>
            #{por.hvala}
            <h:outputFormat value="#{por.ukupno}">
               <f:param value="#{quizBean.poeni}"/>
            </h:outputFormat>
         </p>
         <p>
            <h:commandButton value="#{por.ponovoPokreni}"
                             action="#{quizBean.novaIgra}"/>
         </p>
      </h:form>
   </h:body>
</html>
```

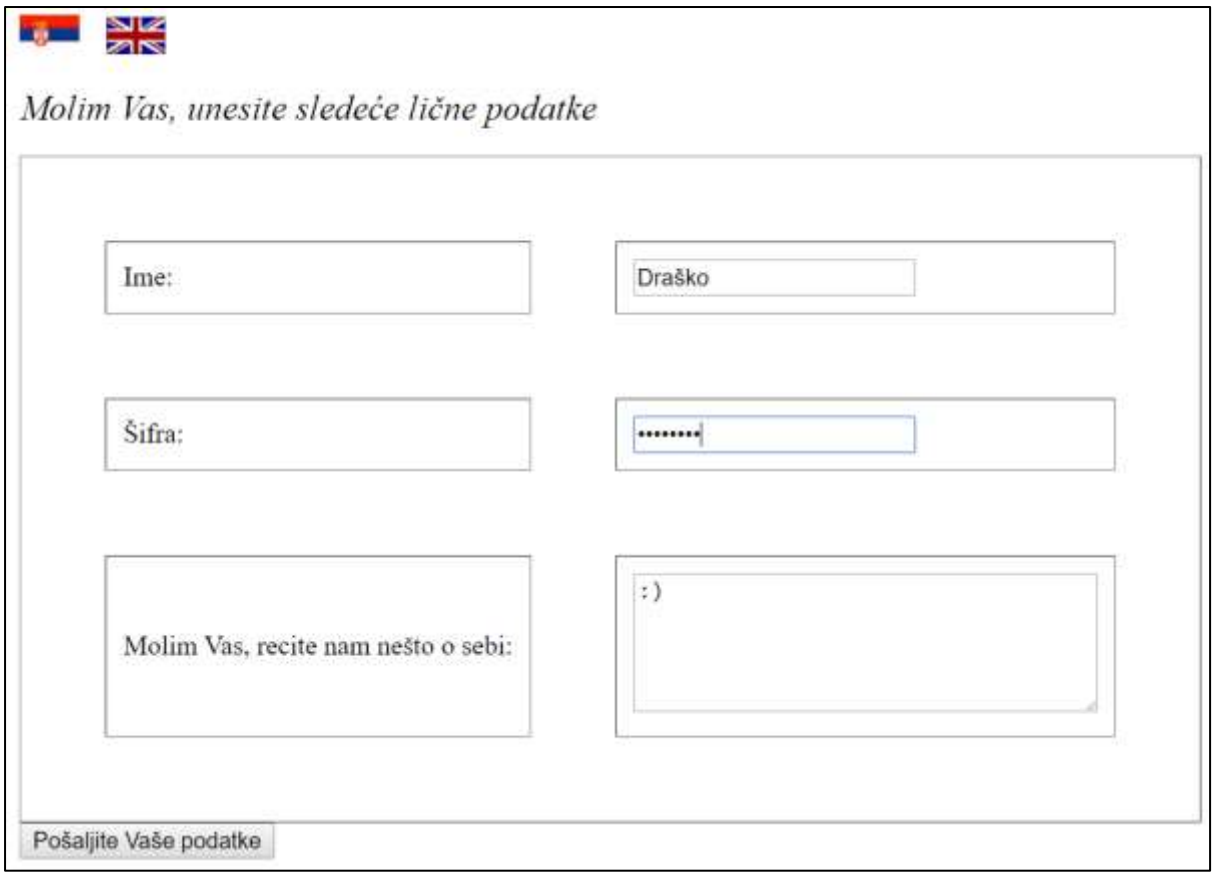
Neophodno je i u binu QuizBean, iz zadatka 2, dodati još tri metode:

```
    public int getTekuciProblem() {
        return tekuciProblem;
    }

    public void setTekuciProblem(int novaVrednost) {
        tekuciProblem = novaVrednost;
        System.out.println("Nova vrednost: " + novaVrednost);
    }

    public String getPreskoci() {
        if (tekuciProblem < problemi.size() - 1) {
            return "index";
```

```
        } else {
            return "gotovo";
        }
    }
```

Zadatak 4

Na JSF strani prikazana je forma u koju korisnik treba da unese ime, lozinku i neke informacije o sebi. Iznad forme treba da postoje i dve zastavice, britanska i srpska, tako da se odabirom određene zastavice promeni jezik date forme. Podrazumevani jezik treba da bude srpski. Druga rezultujuća JSF stranica treba da prikaže ime korisnika i informacije o tom korisniku, koje su unite preko forme.



REŠENJE

U ovom zadatku biće prikazana upotreba komandnih linkova, tako što se lokalizacija menja u zavisnosti na zastavicu na koju se klikne.

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

```
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>#{por.porukaProzor}</title>
      <link type="text/css" href="stilovi.css"/>
   </h:head>
   <h:body>
      <h:form>
         <h:commandLink action="#{promenaLokalizacije.srpski}">
            <h:graphicImage library="images" name="serbian_flag.gif"
                            style="border: 0px; margin-right: 1em;"/>
         </h:commandLink>
         <h:commandLink action="#{promenaLokalizacije.engleski}">
            <h:graphicImage library="images"
                            name="en_flag.gif" style="border: 0px"/>
         </h:commandLink>
         <p><h:outputText value="#{por.naslovPoruka}"
         style="font-style: italic; font-size: 1.3em"/></p>
         <h:panelGrid columns="2" border="1"
                     cellpadding="10px" cellspacing="50">
            #{por.ime}
            <h:inputText value="#{korisnik.korime}"/>
            #{por.lozinka}
            <h:inputSecret value="#{korisnik.lozinka}"/>
            #{por.biografija}
            <h:inputTextarea value="#{korisnik.biografija}"
                             rows="5" cols="35"/>
         </h:panelGrid>
         <h:commandButton value="#{por.potvrda}" action="drugaStrana"/>
      </h:form>
   </h:body>
</html>

//veb stranica: drugaStrana.xhtml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>#{por.hvalaPorukaProzor}</title>
   </h:head>
   <h:body>
      <h:outputText value="#{por.ime}" style="font-style: italic"/>
      #{korisnik.korime}
      <br/>
      <h:outputText value="#{por.biografijaRezultat}"
                    style="font-style: italic"/>
      <br/>
      <pre>#{korisnik.biografija}</pre>
```

```
   </h:body>
</html>

//izvorni fajl: etf.beans/KorisnikBean.java
package etf.beans;

import java.io.Serializable;
import javax.faces.bean.ManagedBean;
   // ili import javax.inject.Named;
import javax.faces.bean.SessionScoped;
   // ili import javax.enterprise.context.SessionScoped;

@ManagedBean(name="korisnik") // ili @Named("korisnik")
@SessionScoped
public class KorisnikBean implements Serializable {
   private String korime;
   private String lozinka;
   private String biografija;

    public String getKorime() {
        return korime;
    }

    public void setKorime(String korime) {
        this.korime = korime;
    }

    public String getLozinka() {
        return lozinka;
    }

    public void setLozinka(String lozinka) {
        this.lozinka = lozinka;
    }

    public String getBiografija() {
        return biografija;
    }

    public void setBiografija(String biografija) {
        this.biografija = biografija;
    }
}

//izvorni fajl: etf.beans/PromenaLokalizacije.java
package etf.beans;
import java.io.Serializable;
import java.util.Locale;
import java.util.ResourceBundle;
```

```
import javax.faces.bean.ManagedBean;
   // ili import javax.inject.Named;
import javax.faces.bean.SessionScoped;
   // ili import javax.enterprise.context.SessionScoped;
import javax.faces.context.FacesContext;

@ManagedBean // ili @Named
@SessionScoped
public class PromenaLokalizacije implements Serializable {
   public String srpski() {
      FacesContext context = FacesContext.getCurrentInstance();
      context.getViewRoot().setLocale(new Locale("rs"));
      ResourceBundle paket =
            context.getApplication().getResourceBundle(context, "por");
      String poruka = paket.getString("porukaProzor");
      return null;
   }

   public String engleski() {
      FacesContext context = FacesContext.getCurrentInstance();
      context.getViewRoot().setLocale(Locale.ENGLISH);
      return null;
   }
}
```

//izvorni fajl: poruke.properties

```
porukaProzor=Using Command Links
hvalaPorukaProzor=Thank you for submitting your information
hvala=Thank you!
naslovPoruka=Please enter the following personal information
ime=Name:
lozinka=Password:
biografija=Please tell us about yourself:
biografijaRezultat=Some information about you:
potvrda=Submit your information
```

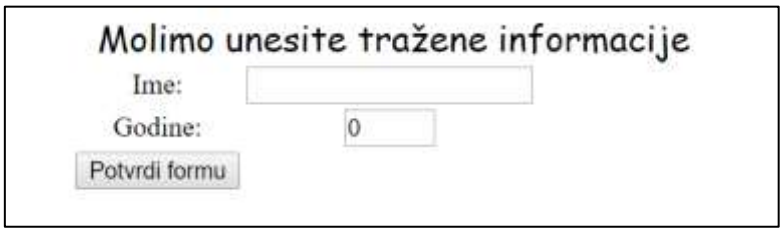
//XML fajl: WEB-INF/faces-config.xml

```
<?xml version="1.0"?>
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
   version="2.0">
   <application>
      <resource-bundle>
         <base-name>etf.beans.poruke</base-name>
         <var>por</var>
      </resource-bundle>
```

```
      <locale-config>
          <default-locale>en</default-locale>
          <supported-locale>rs</supported-locale>
      </locale-config>
   </application>
</faces-config>
```

Zadatak 5

Na JSF strani prikazana je forma u koju korisnik treba da unese svoje ime i koliko godina ima. Forma može biti potvrđena samo ukoliko je korisnik uneo ime i uneo godine. Ukoliko korisnik nije uneo jedan ili oba ova podatka, treba prikazati ispis poruke sa greškom.



REŠENJE

U ovom zadatku biće prikazana upotreba poruka pomoću JSF tagova: h:message i h:messages. Poruke treba prikazati u koloni u okviru elementa h:panelGrid.

```
//veb stranica: index.xhtml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>#{por.porukaProzor}</title>
        <h:outputStylesheet library="css" name="styles.css"/>
    </h:head>
    <h:body>
        <div class="mojaklasa">
            <h:form>
                <h:outputText value="#{por.porukaPozdravna}"
                              styleClass="emphasis"/>
                <br/>
                <h:messages errorClass="greske" layout="table"/>
                <h:panelGrid columns="3">
                    #{por.unosImena}:
                    <h:inputText id="mojeime" value="#{korisnik.ime}"
                                 required="true"
                                 label="#{por.unosImena}"/>
```

```
                    <h:message for="mojeime" errorClass="greske"/>
                    #{por.unosGodina}:
                    <h:inputText id="mojegod" value="#{korisnik.godine}"
                                 required="true" size="3"
                                 maxlength="2" label="#{por.unosGodina}"/>
                    <h:message for="mojegod" errorClass="greske" />
                    <h:commandButton value="#{por.potvrda}"/>
                </h:panelGrid>
            </h:form>
        </div>
    </h:body>
</html>
```

//izvorni fajl: etf.beans/KorisnikBean.java

```
package etf.beans;

import java.io.Serializable;
import javax.faces.bean.ManagedBean;
   // or import javax.inject.Named;
import javax.faces.bean.SessionScoped;
   // or import javax.enterprise.context.SessionScoped;

@ManagedBean(name="korisnik") // or @Named("korisnik")
@SessionScoped
public class KorisnikBean implements Serializable {
   private String ime;
   private int godine;

    public String getIme() {
        return ime;
    }

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

    public int getGodine() {
        return godine;
    }

    public void setGodine(int godine) {
        this.godine = godine;
    }
}
```

//izvorni fajl: etf.beans/poruke.properties

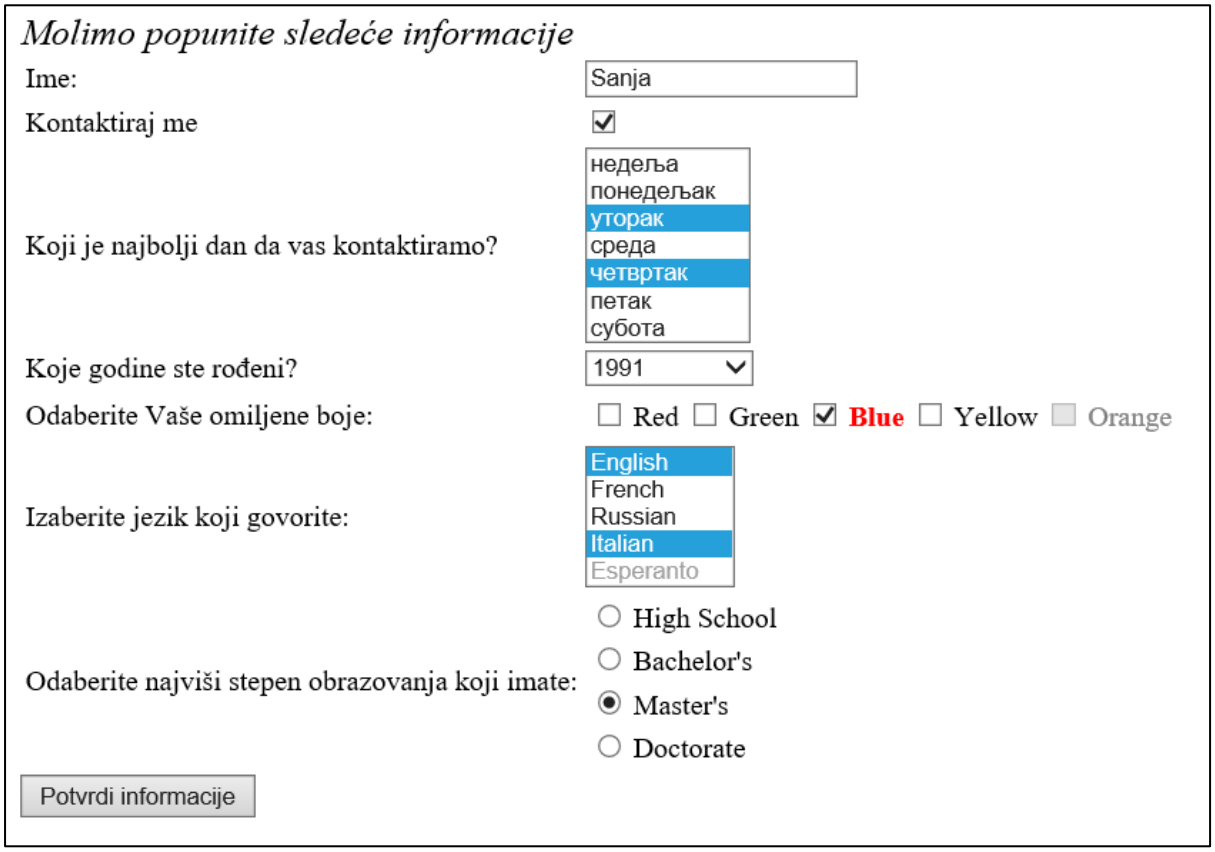
```
porukaProzor=Koriscenje h:messages i h:message
porukaPozdravna=Molimo unesite tražene informacije
unosImena=Ime
```

```
unosGodina=Godine
potvrda=Potvrdi formu

//CSS fajl: resources/styles.css
.errors {
   font-style: italic;
   color: red;
}
.emphasis {
   font-size: 1.3em;
   font-family: Comic Sans MS;
}

body {
    text-align: center;
}

.mojaklasa {
    width: 30%;
    margin: 0 auto;
}

//XML fajl: WEB-INF/faces-config.xml
<?xml version="1.0"?>
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
   version="2.0">
   <application>
      <resource-bundle>
         <base-name>etf.beans.poruke</base-name>
         <var>por</var>
      </resource-bundle>
   </application>
</faces-config>
```

Zadatak 6

Na JSF strani realizovati formu prikazanu kao na slici. Forma treba da sadrži sledeće JSF elemente:

- tekstualno polje za unos imena (h:inputText)
- čekboks za kontaktiranje (h:selectBooleanCheckbox)
- listu sa mogućnošću izbora više opcija za dane u nedelji (h:selectManyListbox)
- padajuća lista za izbor godine rođenja (h:selectOneMenu)
- čekboksove za izbor omiljenih boja (h:selectManyCheckbox)
- listu sa mogućnošću izbora više opcija za jezike (h:selectManyListbox)
- radio dugmad za izbor jedne opcije (h:selectOneRadio)



Polja za unos imena, godine rođenja i jezika su obavezna polja. Rezultujuća stranica treba da prikaže rezultate koji se proslede preko ove forme.

REŠENJE

```
//veb stranica: index.xhtml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
```

```
  xmlns:f="http://java.sun.com/jsf/core">
<h:head>
    <h:outputStylesheet library="css" name="styles.css"/>
    <title>#{poruke.naslovProzora}</title>
</h:head>

<h:body>
    <h:outputText value="#{poruke.labelaNaslovna}"
                  styleClass="emphasis"/>
    <h:form>
        <h:panelGrid columns="2">
            #{poruke.labelaIme}
            <h:inputText value="#{forma.name}"
                required="true" requiredMessage="Niste uneli ime" />

            #{poruke.kontaktiraj}
            <h:selectBooleanCheckbox value="#{forma.contactMe}"/>

            #{poruke.najboljiDan}
            <h:selectManyListbox value="#{forma.bestDaysToContact}">
                <f:selectItems value="#{forma.daysOfTheWeek}" var="w"
                       itemLabel="#{w.dayName}"
                       itemValue="#{w.dayNumber}"/>
            </h:selectManyListbox>

            #{poruke.godinaRodjenja}
            <h:selectOneMenu value="#{forma.yearOfBirth}"
                             required="true">
                <f:selectItems value="#{forma.yearItems}"/>
            </h:selectOneMenu>

            #{poruke.porukaUnosBoje}
            <h:selectManyCheckbox value="#{forma.colors}"
                      selectedClass="selected" disabledClass="disabled"
                      onchange="submit()" immediate="true">
                <f:selectItems value="#{forma.colorItems}"/>
            </h:selectManyCheckbox>

            #{poruke.porukaUnosJezika}
            <h:selectManyListbox size="5" value="#{forma.languages}"
                                 required="true">
                <f:selectItems value="#{forma.languageItems}"/>
            </h:selectManyListbox>

            #{poruke.porukaUnosObrazovanje}
            <h:selectOneRadio value="#{forma.education}"
                              layout="pageDirection">
                <f:selectItems value="#{forma.educationItems}"/>
            </h:selectOneRadio>
```

```
            </h:panelGrid>
            <h:commandButton value="#{poruke.porukaDugme}"
                             action="prikaziInfo"/>
        </h:form>
        <h:messages/>
    </h:body>
</html>
```

```
//veb stranica: prikaziInfo.xhtml
<?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:f="http://java.sun.com/jsf/core"
xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>#{poruke.naslovProzora}</title>
   </h:head>
   <h:body>
      <h:form>
         <h:outputStylesheet library="css" name="styles.css"/>
         <h:outputFormat value="#{poruke.hvalaLabela}">
            <f:param value="#{forma.name}"/>
         </h:outputFormat>
         <h:panelGrid columns="2">
            #{poruke.kontaktLabela}
            <h:outputText value="#{forma.contactMe}"/>
            #{poruke.najboljidanLabela}
            <h:outputText value="#{forma.bestDaysConcatenated}"/>
            #{poruke.godrodjenjaLabela}
            <h:outputText value="#{forma.yearOfBirth}"/>
            #{poruke.jezikLabela}
            <h:outputText value="#{forma.languages}"/>
            #{poruke.bojaLabela}
            <h:outputText value="#{forma.colorsConcatenated}"/>
            #{poruke.obrazovanjeLabela}
            <h:outputText value="#{forma.education}"/>
         </h:panelGrid>
         <h:commandButton value="#{poruke.nazad}" action="index"/>
      </h:form>
   </h:body>
</html>
```

```
//izvorni fajl: beans/RegistracionaForma.java
package etf.beans;

import java.awt.Color;
import java.io.Serializable;
import java.text.DateFormatSymbols;
```

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collection;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Set;
import java.util.TreeSet;

import javax.faces.bean.ManagedBean;
   // or import javax.inject.Named;
import javax.faces.bean.SessionScoped;
   // or import javax.enterprise.context.SessionScoped;
import javax.faces.model.SelectItem;

@ManagedBean(name="forma") // or @Named("forma")
@SessionScoped
public class RegistracionaForma implements Serializable {
   public enum Education { HIGH_SCHOOL, BACHELOR, MASTER, DOCTOR };

   public static class Weekday {
      private int dayOfWeek;
      public Weekday(int dayOfWeek) {
         this.dayOfWeek = dayOfWeek;
      }

      public String getDayName() {
         DateFormatSymbols symbols = new DateFormatSymbols();
         String[] weekdays = symbols.getWeekdays();
         return weekdays[dayOfWeek];
      }

      public int getDayNumber() {
         return dayOfWeek;
      }
   }

   private String name;
   private boolean contactMe;
   private int[] bestDaysToContact;
   private Integer yearOfBirth;
   private int[] colors;
   private Set<String> languages = new TreeSet<String>();
   private Education education = Education.BACHELOR;

   public String getName() { return name; }
   public void setName(String newValue) { name = newValue; }

   public boolean getContactMe() { return contactMe; }
```

```
public void setContactMe(boolean newValue) { contactMe = newValue; }

public int[] getBestDaysToContact() { return bestDaysToContact; }
public void setBestDaysToContact(int[] newValue) {
            bestDaysToContact = newValue;
}

public Integer getYearOfBirth() { return yearOfBirth; }
public void setYearOfBirth(Integer newValue) { yearOfBirth = newValue; }

public int[] getColors() { return colors; }
public void setColors(int[] newValue) { colors = newValue; }

public Set<String> getLanguages() { return languages; }
public void setLanguages(Set<String> newValue) { languages = newValue; }

public Education getEducation() { return education; }
public void setEducation(Education newValue) { education = newValue; }

public Collection<SelectItem> getYearItems() { return birthYears; }

public Weekday[] getDaysOfTheWeek() { return daysOfTheWeek; }

public SelectItem[] getLanguageItems() { return languageItems; }

public SelectItem[] getColorItems() { return colorItems; }

public Map<String, Education> getEducationItems() {
            return educationItems; }

public String getBestDaysConcatenated() {
   return Arrays.toString(bestDaysToContact);
}

public String getColorsConcatenated() {
   StringBuilder result = new StringBuilder();
   for (int color : colors)
         result.append(Integer.toHexString(color) + ",");
   return result.toString();
}

private SelectItem[] colorItems = {
   new SelectItem(Color.RED.getRGB(), "crvena"), // value, label
   new SelectItem(Color.GREEN.getRGB(), "zelena"),
   new SelectItem(Color.BLUE.getRGB(), "plava"),
   new SelectItem(Color.YELLOW.getRGB(), "žuta"),
   new SelectItem(Color.ORANGE.getRGB(), "narandžasta", "", true)
                                                        // disabled
};
```

```
private static Map<String, Education> educationItems;
static {
   educationItems = new LinkedHashMap<String, Education>();
   educationItems.put("High School", Education.HIGH_SCHOOL);
                                                  //label, value
   educationItems.put("Bachelor's", Education.BACHELOR);
   educationItems.put("Master's", Education.MASTER);
   educationItems.put("Doctorate", Education.DOCTOR);
};

private static SelectItem[] languageItems = {
   new SelectItem("engleski"),
   new SelectItem("francuski"),
   new SelectItem("ruski"),
   new SelectItem("italijanski"),
   new SelectItem("esperanto", "esperanto", "", true) // disabled
};

private static Collection<SelectItem> birthYears;
static {
   birthYears = new ArrayList<SelectItem>();
   // Prva stavka ne može da se odabere
   birthYears.add(new SelectItem(null, "Odaberite godinu:", "", false,
                                 false, true));
   for (int i = 1900; i < 2020; ++i) birthYears.add(new SelectItem(i));
}

private static Weekday[] daysOfTheWeek;
static {
   daysOfTheWeek = new Weekday[7];
   for (int i = Calendar.SUNDAY; i <= Calendar.SATURDAY; i++) {
      daysOfTheWeek[i - Calendar.SUNDAY] = new Weekday(i);
   }
}

private static Collection<SelectItem> daysItems;
static {
   daysItems = new ArrayList<SelectItem>();
   // Prva stavka ne može da se odabere
   daysItems.add(new SelectItem(null, "Odaberite dan", "", false,
                                false, true));
   for (int i = 0; i < 7; ++i)
         daysItems.add(new SelectItem(daysOfTheWeek[i].getDayNumber(),
         daysOfTheWeek[i].getDayName()));
}

public Collection<SelectItem> getDaysItems() {
  return daysItems;
}
```

```
  public void setDaysItems(Collection<SelectItem> daysItems) {
    RegistracionaForma.daysItems = daysItems;
  }
}

//izvorni fajl: poruke.properties
naslovProzora=Primena checkbox-ova, radio dugmadi, menija i listi
labelaNaslovna=Molimo popunite sledeće informacije

labelaIme=Ime:
kontaktiraj=Kontaktiraj me
najboljiDan=Koji je najbolji dan da vas kontaktiramo?
godinaRodjenja=Koje godine ste rođeni?
porukaDugme=Potvrdi informacije
nazad=Nazad
porukaUnosJezika=Izaberite jezik koji govorite:
porukaUnosObrazovanje=Odaberite najviši stepen obrazovanja koji imate:
porukaEmailUnos=Izaberite vaš e-mail:
porukaUnosBoje=Odaberite Vaše omiljene boje:

hvalaLabela=Hvala {0}, za vaše informacije
kontaktLabela=Kontaktirajte me:
najboljidanLabela=Najbolji dan za kontaktiranje:
godrodjenjaLabela=Vaša godina rođenja:
bojaLabela=Boje:
jezikLabela=Jezici:
obrazovanjeLabela=Obrazovanje:

//CSS fajl: poruke.properties
.emphasis {
   font-style: italic;
   font-size: 1.3em;
}
.disabled {
   color: gray;
}
.selected {
   font-weight: bold;
   color: #FF0000;
}

//XML fajl: WEB-INF/faces-config.xml
<?xml version="1.0"?>
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
   version="2.0">
```

```
   <application>
      <resource-bundle>
         <base-name>etf.beans.poruke</base-name>
         <var>poruke</var>
      </resource-bundle>
   </application>
</faces-config>
```

Zadatak 7

Realizovati JSF stranu košarkaške ABA lige, korišćenjem Facelets tehnologije. Potrebno je napraviti šablonsku JSF stranicu koja ima gornje zaglavlje (header), meni sa leve strane (left side-bar) i sadržaj. Na početnoj strani, korisniku treba prikazati izgled ekrana kao što je prikazan na slici, sa formom za logovanje:

Kada korisnik unese korisničko ime i lozinku (nije potrebno vršiti nikakve dodatne provere) i potvrdi formu dugmetom “Uloguj se” dobija novo gornje zaglavlje i novi meni sa leve strane, ali korišćenjem iste šablonske stranice. U meniju sa leve strane treba prikazati sve grbove košarkaških klubova koji su učesnici ove regionalne košarkaške lige. Klikom na određeni grb košarkaškog kluba, u delu stranice koji prikazuje sadržaj treba otvoriti stranicu tog kluba i u okviru te zasebne stranice treba ispisati pun naziv kluba, mesto odakle taj klub dolazi, kao i državu. Sve vreme dok se menja sadržaj centralnog dela u kome se nalazi sadržaj, gornje zaglavlje i meni sa leve strane moraju da ostanu neizmenjeni, kao što je prikazano na sledećoj slici:

Grbove klubova čuvati u posebnom direktorijumu u okviru projekta. Dizajn same aplikacije realizovati korišćenjem CSS fajla. Sve poruke koje se ispisuju korisnicima treba čuvati u okviru zasebnog properties fajla.

REŠENJE

```
//veb stranica: index.xhtml
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets">
   <head><title>Ovaj naslov se ne prikazuje</title></head>
   <body>
      <ui:composition template="/templates/masterLayout.xhtml">
         <ui:define name="windowTitle">
            #{poruke.loginTitle}
         </ui:define>

         <ui:define name="heading">
            <ui:include src="/sections/login/header.xhtml"/>
         </ui:define>
```

```
         <ui:define name="sidebarLeft">
            <ui:include src="/sections/login/sidebarLeft.xhtml"/>
         </ui:define>

         <ui:define name="content">
            <h:form>
               <h:panelGrid columns="2">
                  #{poruke.unosKorisnickogImena}
                  <h:inputText id="name" value="#{korisnik.kime}"/>
                  #{poruke.unosLozinke}
                  <h:inputSecret id="password"
                                 value="#{korisnik.lozinka}"/>
               </h:panelGrid>
               <p>
                  <h:commandButton value="#{poruke.loginDugme}"
                                   action="abaliga"/>
               </p>
            </h:form>
         </ui:define>
      </ui:composition>

   </body>
</html>

//izvorni fajl: etf.facelets/Korisnik.java
package etf.facelets;

import java.io.Serializable;
import java.util.Date;
import javax.faces.bean.ManagedBean;
   // or import javax.inject.Named;
import javax.faces.bean.SessionScoped;
   // or import javax.enterprise.context.SessionScoped;

@ManagedBean(name="korisnik") // or @Named("korisnik")
@SessionScoped
public class Korisnik implements Serializable {
  private String kime;
  private String lozinka;

    public String getKime() {
        return kime;
    }

    public void setKime(String kime) {
        this.kime = kime;
    }
```

```
    public String getLozinka() {
        return lozinka;
    }

    public void setLozinka(String lozinka) {
        this.lozinka = lozinka;
    }
}
```

//šablonska stranica: templates/MasterLayout.xhtml

```
<!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://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html">

   <h:head>
      <title>
        <ui:insert name="windowTitle">
          Podrazumevani naslov
        </ui:insert>
      </title>
      <h:outputStylesheet library="css" name="styles.css"/>
   </h:head>

   <h:body>
      <div id="heading">
         <ui:insert name="heading">
            <ui:include src="/sections/klubovi/header.xhtml"/>
         </ui:insert>
      </div>


      <div id="sidebarLeft">
         <ui:insert name="sidebarLeft">
            <ui:include src="/sections/klubovi/sidebarLeft.xhtml"/>
         </ui:insert>
      </div>

      <div id="content">
         <ui:insert name="content"/>
      </div>
      <ui:debug/>
   </h:body>
</html>
```

Sada ćemo prikazati zaglavlje i meni početne stranice:

```
//prvi deo šablonske stranice pre logovanja: sections/login/header.xhtml
<!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://java.sun.com/jsf/facelets">
   <head><title>Ovaj naslov se ne prikazuje</title></head>
   <body>
      <ui:composition>
         <div class="header">
            #{poruke.loginHeading}
         </div>
      </ui:composition>
   </body>
</html>

//drugi deo šablonske stranice pre logovanja:
sections/login/sidebarLeft.xhtml
<!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://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html">
   <head><title>Ovaj naslov se ne prikazuje</title></head>
   <body>
      <ui:composition>
         <div class="welcome">
            #{poruke.loginWelcome}
            <div class="welcomeImage">
               <h:graphicImage library="slike" name="aba.jpg"/>
            </div>
         </div>
      </ui:composition>
   </body>
</html>
```

Veb stranica nakon uspešnog logovanja takođe koristi šablon, i izgleda ovako:

```
//veb stranica: abaliga.xhtml
<!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://java.sun.com/jsf/facelets">
   <head><title>Ovaj naslov se nece prikazivati</title></head>
   <body>
      <ui:composition template="/templates/masterLayout.xhtml">
         <ui:define name="windowTitle">
            #{poruke.kluboviNaslov}
         </ui:define>
```

```
         <ui:define name="content">
            #{poruke.klubovi Dobrodosli}
         </ui:define>
      </ui:composition>
   </body>
</html>
```

Sada ćemo prikazati zaglavlje i meni stranice nakon uspešnog logovanja:

//prvi deo šablonske stranice pre logovanja: sections/klubovi/header.xhtml

```
<!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://java.sun.com/jsf/facelets">
   <head><title>Naslov koji se ne prikazuje</title></head>
   <body>
      <ui:composition>
         <div class="header">
            #{poruke.klubovi Podnaslov}
         </div>
      </ui:composition>
   </body>
</html>
```

//drugi deo šablonske stranice pre logovanja: sections/klubovi/sidebarLeft.xhtml

```
<!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://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:mojtag="http://corejsf.com/facelets">
    <head><title>Ovaj naslov se ne koristi</title></head>
    <body>
        <ui:composition>
            <h:form>
                <mojtag:klub name="Buducnost"
                        image="#{resource['slike:Buducnost.jpg']}"/>
                <mojtag:klub name="Cedevita"
                        image="#{resource['slike:Cedevita.jpg']}"/>
                <mojtag:klub name="Cibona"
                        image="#{resource['slike:Cibona.jpg']}"/>
                <mojtag:klub name="CrvenaZvezda"
                        image="#{resource['slike:CrvenaZvezda.jpg']}"/>
                <mojtag:klub name="Igokea"
                        image="#{resource['slike:Igokea.jpg']}"/>
                <mojtag:klub name="MZTSkopje"
                        image="#{resource['slike:MZTSkopje.jpg']}"/>
```

```
                <mojtag:klub name="MegaLeks"
                        image="#{resource['slike:MegaLeks.jpg']}"/>
                <mojtag:klub name="MetalacValjevo"
                        image="#{resource['slike:MetalacValjevo.jpg']}"/>
                <mojtag:klub name="Partizan"
                        image="#{resource['slike:Partizan.jpg']}"/>
            </h:form>
        </ui:composition>
    </body>
</html>
```

Svaki klub treba da ima svoju stranicu oblika ime_kluba.xhtml, koja izgleda ovako:

```
<!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://java.sun.com/jsf/facelets">
   <head><title>naslov koji se ne prikazuje</title></head>
   <body>
      <ui:composition template="/templates/masterLayout.xhtml">
         <ui:define name="windowTitle">
            #{poruke.buducnost}
         </ui:define>

         <ui:define name="content">
            KK Budućnost Podgorica <br/>
            CRNA GORA
         </ui:define>
      </ui:composition>
   </body>
</html>
```

Sada ćemo prikazati kako se prave Custom tagovi:

```
//XHTML fajl sa definisanim elementom: tags/corejsf/klub.xhtml
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets">
   <h:head><title>IGNORED</title></h:head>
   <h:body>
      <ui:composition>
         <div class='#{name == liga.odabraniKlub ?
                     "klubGrbOdabran" : "klubGrb"}'>
            <h:commandLink action="#{liga.promeniKlub(name)}">
               <h:graphicImage value="#{image}"/>
            </h:commandLink>
            <ui:insert name="content1"/>
```

```
        </div>
      </ui:composition>
   </h:body>
</html>
```

```
//izvorni fajl: etf.facelets/Liga.java
package etf.facelets;

import java.io.Serializable;
import java.util.Date;
import javax.faces.bean.ManagedBean;
   // or import javax.inject.Named;
import javax.faces.bean.RequestScoped;
   // or import javax.faces.bean.RequestScoped;

@ManagedBean // or @Named
@RequestScoped
public class Liga implements Serializable {
  private String odabraniKlub; //selectedPlanet

  public String getOdabraniKlub() { return odabraniKlub; }

  public String promeniKlub(String novaVrednost) {
     odabraniKlub = novaVrednost;
     return odabraniKlub;
  }
}
```

```
//XML fajl: tags/corejsf.taglib.xml
<!DOCTYPE
<?xml version="1.0"?>
<!DOCTYPE facelet-taglib PUBLIC
    "-//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN"
    "http://java.sun.com/dtd/facelet-taglib_1_0.dtd">

<facelet-taglib>
  <namespace>http://corejsf/facelets</namespace>

  <tag>
    <tag-name>klub</tag-name>
    <source>tags/corejsf/klub.xhtml</source>
  </tag>
</facelet-taglib>
```

Lokaciju ove tab biblioteke treba dodati u fajlu web.xml:

```
    <context-param>
        <param-name>facelets.LIBRARIES</param-name>
        <param-value>/WEB-INF/corejsf.taglib.xml</param-value>
    </context-param>
```

CSS fajl izgleda ovako:

```
#heading {
   padding: 20px;
   height: 30px;
   margin-bottom: 10px;
   background: #dddddd;
   border: 1px solid darkGray;
}

#sidebarLeft {
   border: thin solid lightGray;
   padding: 5px;
   width: 200px;
   background: #eeeeee;
}

#content {
   padding: 5px;
   border: thin solid lightGray;
   background: #fefeef;
   position: absolute;
   right: 8px;
   left: 230px;
   height: 400px;
   top: 90px;
}

.header {
   font-size: 1.4em;
   font-style: italic;
   font-family: Palatino;
   text-align: center;
}

.klubGrb {
   padding: 5px;
   text-align: center;
}

.klubGrbOdabran {
   padding: 5px;
   text-align: center;
   background: #fefeef;
   border: thin solid lightGray;
}

.welcome {
   font-style: italic;
```

```
   font-family: Palatino;
}

.welcomeImage {
   margin-top: 10px;
   text-align: center;
}
```

Fajl faces-config.xml izgleda ovako:

```
<?xml version="1.0"?>
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
      version="2.0">
  <application>
    <resource-bundle>
      <base-name>etf.facelets.poruke</base-name>
      <var>poruke</var>
    </resource-bundle>
  </application>
</faces-config>
```

Fajl poruke.properties izgleda ovako:

```
loginTitle=ABA LIGA
loginHeading=Dobrodošli na ABA!
unosKorisnickogImena=Korisničko ime
unosLozinke=Lozinka
loginDugme=Uloguj se
loginWelcome=Dobrodošli na sajt košarkaške lige. Molimo da se ulogujete.
kluboviPodnaslov=SASTAV ABA LIGE U SEZONI 2016/2017.
kluboviNaslov=ABA Jadranska kosarkaska liga
kluboviDobrodosli=Dobrodošli na stranicu regionalne košarkaške lige
buducnost=Buducnost
cedevita=Cedevita
cibona=Cibona
cz=Crvena Zvezda
igokea=Igokea BL
mzt=MZT Skoplje
mega=Mega Leks
metalac=Metalac
part=Partizan
```

Zadatak 8

Katedra za računarsku tehniku i informatiku (RTI) ima određen broj zaposlenih nastavnika i saradnika. Svaki zaposleni ima svoje ime, zvanje, broj predmeta koje trenutno drži i godine radnog staža. Korišćenjem JSF tabela, prikazati zaposlene na Katedri za RTI, kao što je prikazano na datoj slici. Za dizajn redova/kolona koristiti CSS fajl.

Katedra za računarsku tehniku i informatiku

Ime i prezime	Zvanje	Broj predmeta	Radni staž
Boško Nikolić	redovni profesor	5	15.75
Milo Tomašević	redovni profesor	5	27.5
Jelica Protić	vanredni profesor	4	26.75
Nemanja Kojić	asistent	6	5.5
Dražen Drašković	asistent	9	5.5
Katarina Milenković	asistent	7	2.333
Sanja Delčev	saradnik u nastavi	4	0.4

REŠENJE

```
//veb stranica: index.xhtml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core">
   <h:head>
      <title>Katedra za RTI - spisak zaposlenih</title>
      <h:outputStylesheet library="css" name="styles.css"  />
   </h:head>
   <h:body>
   <h2>Katedra za računarsku tehniku i informatiku</h2>
   <h:form>
      <h:dataTable value="#{korisnici.kolektiv}" var="osoba"
         styleClass="zaposleniTable"
         headerClass="zaposleniTableHeader"
         rowClasses="zaposleniTableOddRow,zaposleniTableEvenRow">
         <h:column>
            <f:facet name="header">Ime i prezime</f:facet>

            #{osoba.ime}
         </h:column>
         <h:column>
            <f:facet name="header">Zvanje</f:facet>
            #{osoba.zvanje}
```

```
            </h:column>
            <h:column>
               <f:facet name="header">Broj predmeta</f:facet>
               #{osoba.brojPredmeta}
            </h:column>
            <h:column>
               <f:facet name="header">Radni staž</f:facet>
               #{osoba.radniStaz}
            </h:column>
         </h:dataTable>
      </h:form>
      </h:body>
</html>
```

```
//izvorni fajl: etf.rti/Zaposleni.java
package etf.rti;

public class Zaposleni{
   private String ime;
   private String zvanje;
   private int brojPredmeta;
   private double radniStaz;
   private boolean mozeEditovati;

    public Zaposleni(String ime, String zvanje, int brojPredmeta,
                     double radniStaz) {
        this.ime = ime;
        this.zvanje = zvanje;
        this.brojPredmeta = brojPredmeta;
        this.radniStaz = radniStaz;
        this.mozeEditovati = false;
    }

    public String getIme() {
        return ime;
    }

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

    public String getZvanje() {
        return zvanje;
    }

    public void setZvanje(String zvanje) {
        this.zvanje = zvanje;
    }
```

```
    public int getBrojPredmeta() {
        return brojPredmeta;
    }

    public void setBrojPredmeta(int brojPredmeta) {
        this.brojPredmeta = brojPredmeta;
    }

    public double getRadniStaz() {
        return radniStaz;
    }

    public void setRadniStaz(double radniStaz) {
        this.radniStaz = radniStaz;
    }

    public boolean isMozeEditovati() {
        return mozeEditovati;
    }

    public void setMozeEditovati(boolean mozeEditovati) {
        this.mozeEditovati = mozeEditovati;
    }
}
```

```
//izvorni fajl: etf.rti/Korisnici.java
package etf.rti;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Arrays;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean(name = "korisnici", eager = true)
@SessionScoped
public class Korisnici implements Serializable {

   private static final long serialVersionUID = 1L;

   private String ime;
   private String zvanje;
   private int brojPredmeta;
   private double radniStaz;

   private static final ArrayList<Zaposleni> kolektiv
      = new ArrayList<Zaposleni>(Arrays.asList(
      new Zaposleni("Boško Nikolić", "redovni profesor", 5, 15.75),
```

```
   new Zaposleni("Milo Tomašević", "redovni profesor", 5, 27.5),
   new Zaposleni("Jelica Protić", "vanredni profesor", 4, 26.75),
   new Zaposleni("Nemanja Kojić", "asistent", 6, 5.5),
   new Zaposleni("Dražen Drašković", "asistent", 9, 5.5),
   new Zaposleni("Katarina Milenković", "asistent", 7, 2.333),
   new Zaposleni("Sanja Delčev", "saradnik u nastavi", 4, 0.4)
));

public ArrayList<Zaposleni> getKolektiv() {
   return kolektiv;
}

public String addEmployee() {
   Zaposleni osoba = new Zaposleni(ime, zvanje,
                                   brojPredmeta, radniStaz);
   kolektiv.add(osoba);
   return null;
}

public String deleteEmployee(Zaposleni osoba) {
   kolektiv.remove(osoba);
   return null;
}

public String editZaposleni(Zaposleni osoba){
   osoba.setMozeEditovati(true);
   return null;
}

public String saveKolektiv(){
   for (Zaposleni osoba : kolektiv){
      osoba.setMozeEditovati(false);
   }
   return null;
}

 public String getIme() {
     return ime;
 }

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

 public String getZvanje() {
     return zvanje;
 }
```

```
    public void setZvanje(String zvanje) {
        this.zvanje = zvanje;
    }

    public int getBrojPredmeta() {
        return brojPredmeta;
    }

    public void setBrojPredmeta(int brojPredmeta) {
        this.brojPredmeta = brojPredmeta;
    }

    public double getRadniStaz() {
        return radniStaz;
    }

    public void setRadniStaz(double radniStaz) {
        this.radniStaz = radniStaz;
    }
}

//CSS fajl: resources/css/styles.css
.zaposleniTable{
   border-collapse:collapse;
   border:1px solid #000000;
}

.zaposleniTableHeader{
   text-align:center;
   background:none repeat scroll 0 0 #B5B5B5;
   border-bottom:1px solid #000000;
   padding:2px;
}

.zaposleniTableOddRow{
   text-align:center;
   background:none repeat scroll 0 0 #FFFFFF;
}

.zaposleniTableEvenRow{
   text-align:center;
   background:none repeat scroll 0 0 #D3D3D3;
}
```

Zadatak 9

Korišćenjem dinamičke tabele, realizovati JSF stranu u kojoj će biti prikazane sledeće JSF komponente:

- polja za unos teksta (h:inputText),
- dugmad (h:commandButton),
- čekboksovi (h:selectBooleanCheckbox),
- linkovi (h:commandLink),
- slike (h:graphicImage),
- meniji (h:selectOneMenu),
- radio dugmad (h:selectOneRadio),
- liste sa izborom jedne opcije (h:selectOneListbox).

Sadržaj tabele prikazati u najmanje 6 redova, kao što je prikazano na datom primeru.

REŠENJE

```
//XML fajl: WEB-INF/faces-config.xml
<?xml version="1.0"?>
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
```

```
      http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
   version="2.0">
   <application>
      <resource-bundle>
         <base-name>etf.rti.mojpaket.poruke</base-name>
         <var>por</var>
      </resource-bundle>
   </application>

   <managed-bean>
      <managed-bean-name>mojaLista</managed-bean-name>
      <managed-bean-class>java.util.ArrayList</managed-bean-class>
      <managed-bean-scope>session</managed-bean-scope>
      <list-entries>
         <value>1</value>
         <value>2</value>
         <value>3</value>
         <value>4</value>
         <value>5</value>
         <value>6</value>
      </list-entries>
   </managed-bean>
</faces-config>

//veb stranica: index.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{por.windowTitle}</title>
   </h:head>
   <h:body style="background: #eee">
      <h:form>
         <h:dataTable value="#{mojaLista}" var="broj">
            <h:column>
               <f:facet name="header">#{por.numberHeader}</f:facet>
               #{number}
            </h:column>
            <h:column>
               <f:facet name="header">#{por.textfieldHeader}</f:facet>
               <h:inputText value="#{broj}" size="3"/>
            </h:column>
            <h:column>
               <f:facet name="header">#{por.buttonHeader}</f:facet>
               <h:commandButton value="#{broj}"/>
```

```
            </h:column>
            <h:column>
               <f:facet name="header">#{por.checkboxHeader}</f:facet>
               <h:selectBooleanCheckbox value="false"/>
            </h:column>
            <h:column>
               <f:facet name="header">#{por.linkHeader}</f:facet>
               <h:commandLink>#{broj}</h:commandLink>
            </h:column>
            <h:column>
               <f:facet name="header">#{por.graphicHeader}</f:facet>
               <h:graphicImage library="images" name="domina#{broj}.gif"
                               style="border: 0px"/>
            </h:column>
            <h:column>
               <f:facet name="header">#{por.menuHeader}</f:facet>
               <h:selectOneMenu>
                  <f:selectItem itemLabel="#{broj}" itemValue="#{broj}"/>
               </h:selectOneMenu>
            </h:column>
            <h:column>
               <f:facet name="header">#{por.radioHeader}</f:facet>
               <h:selectOneRadio layout="lineDirection" value="nextMonth">
                  <f:selectItem itemValue="da" itemLabel="da"/>
                  <f:selectItem itemValue="ne" itemLabel="ne"/>
               </h:selectOneRadio>
            </h:column>
            <h:column>
               <f:facet name="header">#{por.listboxHeader}</f:facet>
               <h:selectOneListbox size="4">
                  <f:selectItem itemValue="da" itemLabel="da"/>
                  <f:selectItem itemValue="mozda" itemLabel="mozda"/>
                  <f:selectItem itemValue="ne" itemLabel="ne"/>
               </h:selectOneListbox>
            </h:column>
         </h:dataTable>
      </h:form>
   </h:body>
</html>

//CSS fajl: styles.css
.allColumns {
   text-align: center;
}
.headers {
   text-align: center;
}
```

```
//izvorni fajl: poruke.properties
windowTitle=Koriscenje JSF kompnenti u tabelama

numberHeader=Broj
textfieldHeader=Tekstualna polja
buttonHeader=Dugmad
checkboxHeader=Čekboks-ovi
linkHeader=Linkovi
menuHeader=Meniji
graphicHeader=Slike
radioHeader=Radio dugmad
listboxHeader=Liste
```

Zadatak 10

Korišćenjem dinamičke tabele, realizovati JSF stranu u kojoj će biti prikazani srpski književnici. Tabela treba da ima dve kolone, za prezime i za ime. U Tabeli treba iskoristiti JSF tagove za gornje i donje zaglavlje i naslov (*caption*) tabele.

Srpski književnici

Prezime	Ime
Andrić,	Ivo
Selimović,	Meša
Dučić,	Jovan
Šantić,	Aleksa
[footer]	[footer]

REŠENJE

```
//veb stranica: index.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{poruke.windowTitle}</title>
   </h:head>
   <h:body>
      <h:form>
         <h:dataTable value="#{tableData.pisci}" var="osoba"
```

```
                    captionStyle="font-size: 0.95em; font-style:italic"
                    style="width: 250px;">

            <f:facet name="caption">Srpski književnici</f:facet>

            <h:column headerClass="columnHeader"
                      footerClass="columnFooter">
               <f:facet name="header">#{poruke.lastnameColumn}</f:facet>

               #{osoba.prezime},

               <f:facet name="footer">#{poruke.alphanumeric}</f:facet>
            </h:column>

            <h:column headerClass="columnHeader"
                      footerClass="columnFooter">
               <f:facet name="header">#{poruke.firstnameColumn}</f:facet>

               #{osoba.ime}

               <f:facet name="footer">#{poruke.alphanumeric}</f:facet>
            </h:column>
         </h:dataTable>
      </h:form>
   </h:body>
</html>
```

//izvorni fajl: etf.pisci/Osoba.java

```
package etf.pisci;

import java.io.Serializable;

public class Osoba implements Serializable {
   private String ime;
   private String prezime;

   public Osoba(String ime, String prezime) {
      this.ime = ime;
      this.prezime = prezime;
   }

    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;
    }

}
```

//izvorni fajl: etf.pisci/TableData.java

```
package etf.pisci;
import java.io.Serializable;
import javax.inject.Named;
   // or import javax.faces.bean.ManagedBean;
import javax.enterprise.context.SessionScoped;
   // or import javax.faces.bean.SessionScoped;

@Named // or @ManagedBean
@SessionScoped
public class TableData implements Serializable {
   private static final Osoba[] pisci = new Osoba[] {
       new Osoba("Ivo", "Andrić"),
       new Osoba("Meša", "Selimović"),
       new Osoba("Jovan", "Dučić"),
       new Osoba("Aleksa", "Šantić")
   };

   public Osoba[] getPisci() { return pisci;}
}
```

//izvorni fajl: etf.pisci/poruke.properties

```
windowTitle=Primena headera, footera i zaglavlja
lastnameColumn=Prezime
firstnameColumn=Ime
editColumn=Ažuriraj
alphanumeric=[footer]
```

//XML fajl: WEB-INF/faces-config.xml

```
<?xml version="1.0"?>
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
   version="2.0">
      <application>
            <resource-bundle>
                  <base-name>etf.pisci.poruke</base-name>
                  <var>poruke</var>
```

```
            </resource-bundle>
      </application>
</faces-config>

//CSS fajl: styles.css
.columnHeader {
   font-style: bold;
}
.columnFooter {
   font-size: 0.75em;
}
```

Zadatak 11

Korišćenjem dinamičke tabele, realizovati JSF stranu u kojoj će biti prikazani srpski slikari. Tabela treba da ima tri kolone, za prezime i za ime slikara, kao i za čekboks za ažuriranje. U tabeli kada je čekboks polje čekirano omogućiti ažuriranje, a kada nije čekirano, onda ažuriranje ne treba da bude omogućeno. Takođe, ispod tabele treba da postoji dugme "Snimi promene", kojim se sve promene u tabeli snimaju.

REŠENJE

```
//veb stranica: index.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>#{poruke.naslov}</title>
   </h:head>
   <h:body>
      <h:form>
         <h:dataTable value="#{tableData.slikari}" var="slikar">
            <h:column>
               <f:facet name="header">
                  <h:outputText value="#{poruke.azurirajKolona}"
                                style="font-weight: bold"/>
               </f:facet>
                <h:selectBooleanCheckbox value="#{slikar.azuriraj}"
                        onclick="submit()"/>
            </h:column>
            <h:column>
               <f:facet name="header">
                  <h:outputText value="#{poruke.prezimeKolona}"
```

```
                              style="font-weight: bold"/>
                </f:facet>
                 <h:inputText
                    value="#{slikar.prezime}" rendered="#{slikar.azuriraj}"
                             size="10"/>
                 <h:outputText value="#{slikar.prezime}"
                        rendered="#{not slikar.azuriraj}"/>
             </h:column>
             <h:column>
                <f:facet name="header">
                   <h:outputText value="#{poruke.imeKolona}"
                              style="font-weight: bold"/>
                </f:facet>
                 <h:inputText value="#{slikar.ime}"
                        rendered="#{slikar.azuriraj}"
                            size="10"/>
                 <h:outputText value="#{slikar.ime}"
                        rendered="#{not slikar.azuriraj}"/>
             </h:column>
          </h:dataTable>
          <h:commandButton value="#{poruke.snimiDugme}"
                           action="#{tableData.snimi()}"/>
       </h:form>
    </h:body>
</html>
```

Ovde treba obratiti pažnju da za razliku od prethodnog zadatka (zad. 10), ovde klasa Osoba mora imati i dodatni atribut za ažuriranje, čija vrednost može da bude *true* ili *false*.

```
//izvorni fajl: etf.slikari/Osoba.java
package etf.slikari;

import java.io.Serializable;

public class Osoba implements Serializable {
   private String ime;
   private String prezime;
   private boolean azuriraj = false;

    public Osoba(String ime, String prezime) {
        this.ime = ime;
        this.prezime = prezime;
    }

    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 boolean isAzuriraj() {
        return azuriraj;
    }

    public void setAzuriraj(boolean azuriraj) {
        this.azuriraj = azuriraj;
    }
}

//izvorni fajl: etf.slikari/TableData.java
package etf.slikari;

import java.io.Serializable;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean
@SessionScoped
public class TableData implements Serializable {
   private static final Osoba[] slikari = new Osoba[] {
       new Osoba("Paja", "Jovanović"),
       new Osoba("Uroš", "Predić"),
       new Osoba("Sava", "Šumanović"),
       new Osoba("Petar", "Lubarda")
   };

   public Osoba[] getSlikari() { return slikari;}

   public String snimi() {
      for (Osoba o : slikari) o.setAzuriraj(false);
      return null;
   }
}
```

```
//izvorni fajl: etf.slikari/poruke.properties
naslov=Ažuriranje ćelija tabele
prezimeKolona=Prezime
imeKolona=Ime
azurirajKolona=Ažuriraj
snimiDugme=Snimi promene
```

Zadatak 12

Korišćenjem JSF tabele napraviti tabelu sa nacionalnim parkovima Srbije, koja će imati naziv nacionalnog parka, regiju kojoj pripada, sliku i link za brisanje datog nacionalnog parka. Sve podatke o nacionalnim parkovima čuvati u nekoj od Javinih kolekcija.

Nacionalni parkovi:

Regija	*Naziv*	*Regija*	*Kliknuti za brisanje*
Vojvodina	Fruska-gora		Obriši
Istočna Srbija	Djerdap		Obriši
Zapadna Srbija	Tara		Obriši
Centralna Srbija	Kopaonik		Obriši

REŠENJE

```
//veb stranica: index.xhtml
<?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://java.sun.com/jsf/facelets"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{poruke.naslov}</title>
   </h:head>
   <h:body>
      <h1>#{poruke.podnaslov}</h1>
      <h:form>
          <h:dataTable value="#{tableData.nparkovi}"
                  var="park" styleClass="nazivi"
                  headerClass="naziviHeader" columnClasses="regija,naziv"
                  rendered="#{tableData.nparkovi.size()!=0}">
            <h:column>
               <f:facet name="header">#{poruke.kolonaRegija}</f:facet>
               #{park.regija},
            </h:column>
            <h:column>
               <f:facet name="header">#{poruke.kolonaNaziv}</f:facet>
               #{park.naziv}
            </h:column>
            <h:column>
               <f:facet name="header">#{poruke.kolonaRegija}</f:facet>
               <h:graphicImage value="resources/images/#{park.naziv}.jpg"
                               alt="Nema slike: #{park.naziv}"/>
            </h:column>
            <h:column>
                <f:facet name="header">#{poruke.brisanje}</f:facet>
                <h:commandLink value="#{poruke.link}"
                              action="#{tableData.obrisiRed(park)}"/>
            </h:column>
         </h:dataTable>
      </h:form>
   </h:body>
</html>
```

```
//izvorni fajl: etf.nacionalni/NPark.java
package etf.nacionalni;
import java.io.Serializable;

public class NPark implements Serializable {
   private String naziv;
   private String regija;

    public NPark(String naziv, String regija) {
        this.naziv = naziv;
        this.regija = regija;
    }

    public String getNaziv() {
        return naziv;
    }

    public void setNaziv(String naziv) {
        this.naziv = naziv;
    }

    public String getRegija() {
        return regija;
    }

    public void setRegija(String regija) {
        this.regija = regija;
    }
}

//izvorni fajl: etf.nacionalni/TableData.java
package etf.nacionalni;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Arrays;

import javax.faces.bean.ManagedBean;
      // or import javax.inject.Named;
import javax.faces.bean.SessionScoped;
      // or import javax.enterprise.context.SessionScoped;

@ManagedBean
@SessionScoped
public class TableData implements Serializable {
   private ArrayList<NPark> nparkovi = new ArrayList<NPark>(Arrays.asList(
      new NPark("Fruska-gora", "Vojvodina"),
      new NPark("Djerdap", "Istočna Srbija"),
      new NPark("Tara", "Zapadna Srbija"),
```

```
        new NPark("Kopaonik", "Centralna Srbija"),
        new NPark("Sar-planina", "Kosovo i Metohija"),
        new NPark("Prokletije", "Kosovo i Metohija"),
        new NPark("Kucaj-Beljanica", "Istočna Srbija")
    ));

    public ArrayList<NPark> getNparkovi() {
        return nparkovi;
    }

    public String obrisiRed(NPark brisem) {
        nparkovi.remove(brisem);
        return null;
    }
}
```

//izvorni fajl: etf.nacionalni/poruke.properties

```
naslov=Brisanje reda iz tabele
podnaslov=Nacionalni parkovi:
kolonaNaziv=Naziv
kolonaRegija=Regija
brisanje=Kliknuti za brisanje
link=Obriši
```

//CSS fajl: resources/css/styles.css

```
.nazivi {
    border: thin solid black;
}
.naziviHeader {
    text-align: center;
    font-style: italic;
    color: Snow;
    background: Teal;
}
.regija {
    height: 25px;
    text-align: center;
    background: MediumTurquoise;
}
.naziv {
    text-align: left;
    background: PowderBlue;
}
```

Zadatak 13

Korišćenjem JSF konvertera i validatora realizovati veb stranu prikazanu kao na slici. Veb strana treba da sadrži formu koja treba da ima tri tekstualna polja:

- polje za unos iznosa uplate, koje ima konverziju u valutu EUR sa najmanje 2 decimale;
- polje za unos broja kreditne kartice, koje ima validator tako da broj kartice ima najmanje 13 cifara, a najviše 15 cifara;
- polje za unos datuma isteka kreditne kartice, u formatu MM/GG (inicijalno u formi staviti trenutni mesec i trenutnu godinu).

Forma treba da sadrži i dugme za potvrdu, preko koga će se dobiti rezultujuća veb strana, nakon izvršenih standardnih konverzija i validatora, koji su navedeni.

REŠENJE

```
//veb stranica: index.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{poruke.naslov}</title>
   </h:head>
   <h:body>
      <h:form>
         <h1>#{poruke.info}</h1>
         <h:panelGrid columns="3">
            #{poruke.iznos}
            <h:inputText id="amount" label="#{poruke.iznos}"
                         value="#{placanje.iznos}">
               <f:convertNumber minFractionDigits="2"/>
            </h:inputText>
            <h:message for="amount" styleClass="errorMessage"/>
```

```
            #{poruke.brkartice}
            <h:inputText id="card" label="#{poruke.brkartice}"
                         value="#{placanje.kartica}"
                         required="true"
                         requiredMessage="#{poruke.obavezanUnos}">
                <f:validateLength minimum="13"
                                  maximum="15"/>
            </h:inputText>
            <h:message for="card" styleClass="errorMessage" />

            #{poruke.istek}
            <h:inputText id="date" label="#{poruke.istek}"
                         value="#{placanje.datum}">
               <f:convertDateTime pattern="MM/yyyy"/>
               <!-- java.text.SimpleDateFormat -->
            </h:inputText>
            <h:message for="date" styleClass="errorMessage"/>
         </h:panelGrid>
         <h:commandButton value="#{poruke.potvrda}" action="rezultat"/>
      </h:form>
   </h:body>
</html>
```

//veb stranica: rezultat.xhtml

```
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{poruke.naslov}</title>
   </h:head>
   <h:body>
      <h:form>
         <h1>#{poruke.info}</h1>
         <h:panelGrid columns="2">
            #{poruke.iznos}
            <h:outputText value="#{placanje.iznos}">
                <f:convertNumber type="currency"
                                 currencyCode="EUR"/>
            </h:outputText>



            #{poruke.brkartice}
```

```
            <h:outputText value="#{placanje.kartica}"/>

            #{poruke.istek}
            <h:outputText value="#{placanje.datum}">
               <f:convertDateTime pattern="yyyy/MM"/>
            </h:outputText>
         </h:panelGrid>
         <h:commandButton value="#{poruke.nazad}" action="index"/>
      </h:form>
   </h:body>
</html>

//izvorni fajl: etf/PaymentBean.java
package etf;

import java.io.Serializable;
import java.util.Date;
import javax.faces.bean.ManagedBean;
// or import javax.inject.Named;
import javax.faces.bean.SessionScoped;
// or import javax.enterprise.context.SessionScoped;

@ManagedBean(name="placanje")
@SessionScoped
public class PaymentBean implements Serializable {
private double iznos;
   private String kartica = "";
   private Date datum = new Date();

    public double getIznos() {
        return iznos;
    }

    public void setIznos(double iznos) {
        this.iznos = iznos;
    }

    public String getKartica() {
        return kartica;
    }

    public void setKartica(String kartica) {
        this.kartica = kartica;
    }

    public Date getDatum() {
        return datum;
    }
```

```
    public void setDatum(Date datum) {
        this.datum = datum;
    }
}

//izvorni fajl: etf/poruke.properties
naslov=Aplikacija za testiranje konverzije podataka u JSF
info=Molimo unesite informacije o plaćanju
iznos=Iznos
brkartice=Broj kreditne kartice
istek=Kartica ističe (mesec/godina)
potvrda=Procesiraj
nazad=Nazad
paymentInformation=Payment information
obavezanUnos=Molimo unesite broj kreditne kartice

//CSS fajl: resources/css/styles.css
.errorMessage {
   color: red;
}
```

Bean Validation frejmvork

Od verzije 2.0 Java Server Faces integriše *Bean Validation Framework*. To je frejmvork koji uvodi ograničenja kod validacije podataka. Koristi se u Java klasi i pridodaje se uz atribut ili metodu get atributa, na primer:

```
public class KorisnikBean {
    @Size(min = 5) private String korisnickoIme;
    @Future public Date getDate() { ... }
...
}
```

Anotacija	Atribut	Značenje
@Null, @NotNull	nema	Provera vrednosti na *null* ili na *not null*.
@Min, @Max	int, long, short	Provera vrednosti da li je iznad donje granice (min) ili ispod gornje granice (max). Tip atributa je *int*, *long*, *short*, *byte* i njihovi omoti *BigInteger* i *BigDecimal*. Nisu podržani *float* i *double*.
@DecimalMin, @DecimalMax	String	Isto kao prethodno, ali se može primeniti i na *String*.
@Digits	integer, fraction	Proverava vrednost koliko ima cifara ili frakcionih (razlomačkih) cifara. Može se primeniti na *int*, *long*, *short*, *byte* i njihovi omoti *BigInteger* i *BigDecimal*, kao i *String*.
@AssertTrue, @AssertFalse	nema	Provera vrednosti tipa *Boolean* na *true* ili *false*.
@Past, @Future	nema	Provera datuma da li je u prošlosti ili u budućnosti.
@Size	min, max	Provera da li je veličina tekstualnog niza (*String*), niza (*array*), kolekcije ili mape, manja od gornje zadate granice (max) ili veća od donje zadate granice (min).
@Pattern	regexp, flags	Provera određenog regularnog izraza.

Zadatak 14

Potrebno je realizovati formu za registraciju korisnika na forum Elektrotehničkog fakulteta u Beogradu. Korisnik treba da unese korisničko ime i adresu e-pošte. Korisničko ime mora imati najmanje 3 karaktera, a adresa e-pošte treba biti realizovana korišćenjem regularnih izraza, tako da podrži studentske naloge sa domena @etf.rs, @gmail.com i @yahoo.com.

REŠENJE

```
//veb stranica: index.xhtml
<?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>Primer validacije beana</title>
    </h:head>
    <h:body>
        <h2>Primer validacije bean-a</h2>
        <h:form>
            <h:panelGrid columns="3">
                <h:outputLabel value="Ime: " for="ime"/>
                <h:inputText id="ime" value="#{korisnikBean.ime}"/>
                <h:message for="ime" style="color:red"/>

                <h:outputLabel value="E-pošta: " for="eposta"/>
                <h:inputText id="eposta" value="#{korisnikBean.eposta}"/>
                <h:message for="eposta" style="color:red"/>
            </h:panelGrid>
            <h:commandButton value="POTVRDI" action="rezultat"/>
        </h:form>
    </h:body>
</html>

//veb stranica: rezultat.xhtml
<?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>Primer validacije beana</title>
    </h:head>
    <h:body>
        <h2>Uspešno ste se registrovali:</h2>
        <h:form>
            <h:panelGrid columns="2">
```

```
                <h:outputLabel value="Ime: " for="ime"/>
                <h:outputText id="ime" value="#{korisnikBean.ime}"/>

                <h:outputLabel value="E-pošta: " for="eposta"/>
                <h:outputText id="eposta" value="#{korisnikBean.eposta}"/>
            </h:panelGrid>
        </h:form>
    </h:body>
</html>

//izvorni fajl: etf/KorisnikBean.java
package etf;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.validation.constraints.*;

@ManagedBean
@RequestScoped
public class KorisnikBean {
    @Size(min = 3, message = "Korisničko ime mora biti dužine najmanje
                              tri karaktera!")
    private String ime;



    @Pattern(regexp = "^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z]{2,3}$",
      message = "Adresa e-pošte ne odgovara traženom regularnom izrazu.")
    private String eposta;

    public String getIme() {
        return ime;
    }

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

    public String getEposta() {
        return eposta;
    }

    public void setEposta(String eposta) {
        this.eposta = eposta;
    }
}
```

Zadatak 15

Realizovati veb formu koja korišćenjem JSF validacionih provera, procesira formu, samo ukoliko su unete sledeće vrednosti u polja:

- u polje za iznos potrebno je uneti vrednost između 1000 RSD i 20 500 RSD;
- broj kreditne kartice mora biti ispravnog formata*;
- datum isteka je obavezno polje, u obliku MM/GG i mora biti u budućnosti.

Ukoliko neka od validacija ne prođe, neophodno je prikazati pored svakog tekstualnog polja odgovarajuću validacionu poruku sa greškom (kao što je prikazano na slici).

Napomena:

U ovom rešenju primenjena je Luhn formula, napravljena kasnih 60-ih godina od strane grupe matematičara, sa ciljem da verifikuje validne brojeve kreditnih kartica. Formula može detektovati kada je cifra uneta pogrešno ili ako su dve cifre transponovane. Za proces testiranja ove forme, možete koristiti ispravan broj kartice: 4111 1111 1111 1111.

Takođe, u ovom zadatku kreiran je zaseban validator koji proverava standardne kreditne kartice, broj cifara u njihovom broju i njihove formate, prema sledećoj tabeli:

Tip kartice	Broj cifara	Format
MasterCard	16	5xxx xxxx xxxx xxxx
Visa	16	4xxx xxxx xxxx xxxx
Visa	13	4xxx xxx xxx xxx
Discover	16	6xxx xxxx xxxx xxxx
American Express	15	37xx xxxxxx xxxxx
American Express	22	3xxxxx xxxxxxxx xxxxxxxx

REŠENJE

```
//veb stranica: index.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{poruke.title}</title>
   </h:head>
   <h:body>
      <h:form>
         <h1>#{poruke.enterPayment}</h1>
         <h:panelGrid columns="3">
            #{poruke.amount}
            <h:inputText id="amount" label="#{poruke.amount}"
                         value="#{placanje.iznos}">
               <f:convertNumber minFractionDigits="2"/>
            </h:inputText>
            <h:message for="amount" styleClass="errorMessage"/>

            #{poruke.creditCard}
            <h:inputText id="card" label="#{poruke.creditCard}"
                         value="#{placanje.brkartice}" required="true">
               <f:converter converterId="etf.Kartica"/>
               <f:validator validatorId="etf.Kartica"/>
            </h:inputText>
            <h:message for="card" styleClass="errorMessage"/>

            #{poruke.expirationDate}
            <h:inputText id="date" label="#{poruke.expirationDate}"
                         value="#{placanje.datum}">
               <f:convertDateTime pattern="MM/yyyy"/>
            </h:inputText>
            <h:message for="date" styleClass="errorMessage"/>
         </h:panelGrid>
         <h:commandButton value="#{poruke.process}" action="rezultat"/>
      </h:form>
   </h:body>
</html>

//veb stranica: rezultat.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{poruke.title}</title>
   </h:head>
   <h:body>
      <h:form>
         <h1>#{poruke.paymentInformation}</h1>
         <h:panelGrid columns="2">
            #{poruke.amount}
            <h:outputText value="#{placanje.iznos}">
               <f:convertNumber type="currency"
                                currencyCode="RSD"/>
            </h:outputText>

            #{poruke.creditCard}
            <h:outputText value="#{placanje.brkartice}"
                          converter="etf.Kartica"/>

            #{poruke.expirationDate}
            <h:outputText value="#{placanje.datum}">
               <f:convertDateTime pattern="MM/yyyy"/>
            </h:outputText>
         </h:panelGrid>
         <h:commandButton value="#{poruke.back}" action="index"/>
      </h:form>
   </h:body>
</html>

//izvorni fajl: etf/PaymentBean.java
package etf;
import java.io.Serializable;
import java.util.Date;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.validation.constraints.Future;
import javax.validation.constraints.Max;
import javax.validation.constraints.Min;

@ManagedBean(name="placanje")
@SessionScoped
public class PaymentBean implements Serializable {
   @Min(1000) @Max(20500)
   private double iznos;
   private CreditCard brkartice = new CreditCard("");
   @Future
   private Date datum = new Date();
```

```
    public double getIznos() {
        return iznos;
    }

    public void setIznos(double iznos) {
        this.iznos = iznos;
    }

    public CreditCard getBrkartice() {
        return brkartice;
    }

    public void setBrkartice(CreditCard brkartice) {
        this.brkartice = brkartice;
    }

    public Date getDatum() {
        return datum;
    }

    public void setDatum(Date datum) {
        this.datum = datum;
    }
}

//izvorni fajl: etf/CreditCard.java
package etf;

import java.io.Serializable;

public class CreditCard implements Serializable {
   private String broj;

   public CreditCard(String broj) { this.broj = broj; }
   public String toString() { return broj; }
}

//izvorni fajl: etf/CreditCardConverter.java

package etf;

import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;
import javax.faces.convert.ConverterException;
import javax.faces.convert.FacesConverter;
```

```
@FacesConverter("etf.Kartica")
public class CreditCardConverter implements Converter {
   public Object getAsObject(
      FacesContext context,
      UIComponent component,
      String newValue)
      throws ConverterException {
      StringBuilder builder = new StringBuilder(newValue);
      boolean foundInvalidCharacter = false;
      char invalidCharacter = '\0';
      int i = 0;
      while (i < builder.length() && !foundInvalidCharacter) {
         char ch = builder.charAt(i);
         if (Character.isDigit(ch))
            i++;
         else if (Character.isWhitespace(ch))
            builder.deleteCharAt(i);
         else {
            foundInvalidCharacter = true;
            invalidCharacter = ch;
         }
      }

      if (foundInvalidCharacter) {
         FacesMessage message
            = etf.util.Messages.getMessage(
               "etf.messages", "badCreditCardCharacter",
               new Object[] { new Character(invalidCharacter) } );
         message.setSeverity(FacesMessage.SEVERITY_ERROR);
         throw new ConverterException(message);
      }

      return new CreditCard(builder.toString());
   }

   public String getAsString(
      FacesContext context,
      UIComponent component,
      Object value)
      throws ConverterException {
      // length 13: xxxx xxx xxx xxx
      // length 14: xxxxx xxxx xxxxx
      // length 15: xxxx xxxxxx xxxxx
      // length 16: xxxx xxxx xxxx xxxx
      // length 22: xxxxxx xxxxxxxx xxxxxxxx
      String v = value.toString();
      int[] boundaries = null;
      int length = v.length();
      if (length == 13)
```

```
            boundaries = new int[] { 4, 7, 10 };
         else if (length == 14)
            boundaries = new int[] { 5, 9 };
         else if (length == 15)
            boundaries = new int[] { 4, 10 };
         else if (length == 16)
            boundaries = new int[] { 4, 8, 12 };
         else if (length == 22)
            boundaries = new int[] { 6, 14 };
         else
            return v;
         StringBuilder result = new StringBuilder();
         int start = 0;
         for (int i = 0; i < boundaries.length; i++) {
            int end = boundaries[i];
            result.append(v.substring(start, end));
            result.append(" ");
            start = end;
         }
         result.append(v.substring(start));
         return result.toString();
      }
}

//izvorni fajl: etf/CreditCardValidator.java
package etf;

import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.validator.FacesValidator;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;

@FacesValidator("etf.Kartica")
public class CreditCardValidator implements Validator {
   public void validate(FacesContext context, UIComponent component,
            Object value) {
         if(value == null) return;
         String cardNumber;
         if (value instanceof CreditCard)
            cardNumber = value.toString();
         else
            cardNumber = value.toString().replaceAll("\\D", "");
            //uklanjanje karaktera koji nisu cifre
         if(!luhnCheck(cardNumber)) {
            FacesMessage message
               = etf.util.Messages.getMessage(
                  "etf.messages", "badLuhnCheck", null);
```

```
            message.setSeverity(FacesMessage.SEVERITY_ERROR);
            throw new ValidatorException(message);
         }
      }

      private static boolean luhnCheck(String cardNumber) {
         int sum = 0;

         for(int i = cardNumber.length() - 1; i >= 0; i -= 2) {
            sum += Integer.parseInt(cardNumber.substring(i, i + 1));
            if(i > 0) {
               int d = 2 * Integer.parseInt(cardNumber.substring(i - 1, i));
               if(d > 9) d -= 9;
               sum += d;
            }
         }
         return sum % 10 == 0;
      }
   }
```

//izvorni fajl: etf.util/Messages.java

```
package etf.util;

import java.text.MessageFormat;
import java.util.Locale;
import java.util.MissingResourceException;
import java.util.ResourceBundle;
import javax.faces.application.Application;
import javax.faces.application.FacesMessage;
import javax.faces.component.UIViewRoot;
import javax.faces.context.FacesContext;

public class Messages {
   public static FacesMessage getMessage(String bundleName,
                                         String resourceId,
      Object[] params) {
      FacesContext context = FacesContext.getCurrentInstance();
      Application app = context.getApplication();
      String appBundle = app.getMessageBundle();
      Locale locale = getLocale(context);
      ClassLoader loader = getClassLoader();
      String summary = getString(appBundle, bundleName, resourceId,
         locale, loader, params);
      if (summary == null) summary = "???" + resourceId + "???";
      String detail = getString(appBundle, bundleName,
                                resourceId + "_detail",
                                locale, loader, params);
      return new FacesMessage(summary, detail);
   }
```

```
public static String getString(String bundle, String resourceId,
      Object[] params) {
   FacesContext context = FacesContext.getCurrentInstance();
   Application app = context.getApplication();
   String appBundle = app.getMessageBundle();
   Locale locale = getLocale(context);
   ClassLoader loader = getClassLoader();
   return getString(appBundle, bundle, resourceId, locale,
                    loader, params);
}

public static String getString(String bundle1, String bundle2,
      String resourceId, Locale locale, ClassLoader loader,
      Object[] params) {
   String resource = null;
   ResourceBundle bundle;

   if (bundle1 != null) {
      bundle = ResourceBundle.getBundle(bundle1, locale, loader);
      if (bundle != null)
         try {
            resource = bundle.getString(resourceId);
         } catch (MissingResourceException ex) {
         }
   }

   if (resource == null) {
      bundle = ResourceBundle.getBundle(bundle2, locale, loader);
      if (bundle != null)
         try {
            resource = bundle.getString(resourceId);
         } catch (MissingResourceException ex) {
         }
   }
   if (resource == null) return null; // no match
   if (params == null) return resource;

   MessageFormat formatter = new MessageFormat(resource, locale);
   return formatter.format(params);
}

public static Locale getLocale(FacesContext context) {
   Locale locale = null;
   UIViewRoot viewRoot = context.getViewRoot();
   if (viewRoot != null) locale = viewRoot.getLocale();
   if (locale == null) locale = Locale.getDefault();
   return locale;
}
```

```
   public static ClassLoader getClassLoader() {
      ClassLoader loader = Thread.currentThread().getContextClassLoader();
      if (loader == null) loader = ClassLoader.getSystemClassLoader();
      return loader;
   }
}
```

```
//izvorni fajl: etf/poruke.properties
badCreditCardCharacter=Nevalidan broj kartice!
badCreditCardCharacter_detail=Broj kartice sadrži nevalidan karakter {0}.
badLuhnCheck=Broj kartice nije u skladu sa Luhn formulom.
badLuhnCheck_detail=Jedna ili više cifara u broju nisu korektni.
title=Aplikacija za testiranje validacije bankovnih kartica
enterPayment=Molimo unesite informacije o placanju
amount=Iznos
creditCard=Broj kreditne kartice
expirationDate=Datum isteka kartice (MM/GG)
process=POTVRDI
back=Nazad
paymentInformation=Informacije o plaćanju
```

```
//CSS fajl: resources/css/styles.css
.errorMessage {
   color: red;
}
```

Zadatak 16

Realizovati veb formu kao JSF veb stranu preko koje korisnik unosi marku telefona (obavezan podatak), boju telefona (obavezan) i model telefona (opciono). Nakon što potvrdi formu, model telefona treba validirati, na dva načina, korišćenjem bean-a i pomoću faces validatora (<f:validator>). Validacije izvršiti na zasebnim veb stranama.

REŠENJE

```
//veb stranica: index.xhtml
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html">
<h:head>
</h:head>
<h:body>
    <h3>Unos informacija o telefonu</h3>
    <h:form>
        <h:panelGrid columns="2">
            <h:outputLabel for="mname">Marka telefona:</h:outputLabel>
            <h:inputText id="mname"
                required="true"
                requiredMessage="Marka mobilnog telefona je
                                obavezna!"></h:inputText>
            <h:outputLabel for="color">Boja:</h:outputLabel>
            <h:inputText id="color" required="true"
                        requiredMessage="Boja telefona je
                                        obavezna!"></h:inputText>
            <h:outputLabel for="model">Model telefona:</h:outputLabel>
            <h:inputText id="model"></h:inputText>
            <h:commandButton value="Potvrdi" action="mob"/>
        </h:panelGrid>
    </h:form>
</h:body>
</html>

//veb stranica: mob.xhtml
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html">
<h:head>
</h:head>
<h:body>
    <h3>Primer validacije korišćenjem bean-a</h3>
    <h:form>
        <h:outputLabel for="mno">Model telefona:</h:outputLabel>
        <h:inputText id="mno" value="#{mobile.mno}"
            required="true" size="4"
            disabled="#{mobile.mno}" validator="#{mobile.validateModelNo}">
        </h:inputText>
        <h:commandButton value="Submit" action="mobvalidator"/>
    </h:form>
```

```
</h:body>
</html>

//veb stranica: mobvalidator.xhtml
<?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://java.sun.com/jsf/core">
<h:head>
</h:head>
<h:body>
    <h:form>
        <h3>Primer validacije korišćenjem Faces validator</h3>
        <h:outputLabel for="mno" value="Model telefona: " />
        <h:inputText id="mno" value="#{mobileValidator.mno}"
            required="true" requiredMessage="Obavezno popuniti!">
        <f:validator validatorId="mobileValidator"  />
        </h:inputText>
        <h:message for="mno" style="color:blue" />
        <p></p>
        <h:commandButton value="Submit"/>
    </h:form>
</h:body>
</html>

//izvorni fajl: mobilni/Mobile.java
package mobilni;

import java.io.Serializable;
import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.component.UIComponent;
import javax.faces.component.UIInput;
import javax.faces.context.FacesContext;
import javax.validation.constraints.NotNull;

@ManagedBean
@SessionScoped
public class Mobile implements Serializable {
    @NotNull(message="Molimo unesite oznaku modela")
    private String mno;

    public String getMno() {
        return mno;
    }
```

```
    public void setMno(String mno) {
        this.mno = mno;
    }

    public void validateModelNo(FacesContext context, UIComponent comp,
            Object value) {
        String mno = (String) value;

        if (mno.length() < 4) {
            ((UIInput) comp).setValid(false);

            FacesMessage message = new FacesMessage(
                    "Minimalna duzina naziva modela je 4 karaktera");
            context.addMessage(comp.getClientId(context), message);
        }
    }
}

//izvorni fajl: mobilni/MobileValidator.java
package mobilni;

import javax.faces.application.FacesMessage;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.validator.FacesValidator;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;

@ManagedBean
@SessionScoped
@FacesValidator("mobileValidator")
public class MobileValidator implements Validator {

    private String mno;

    public String getMno() {
        return mno;
    }

    public void setMno(String mno) {
        this.mno = mno;
    }

    int maximumlength = 6;

    public MobileValidator() {
    }
```

```
    @Override
    public void validate(FacesContext fc, UIComponent uic, Object obj)
            throws ValidatorException {
        String model = (String) obj;

        if (model.length() > 6) {
            FacesMessage msg = new FacesMessage(
                    "Maksimalna dužina od 6 karaktera je prekoračena.
                        Molimo unesite do 6 karaktera!");
            msg.setSeverity(FacesMessage.SEVERITY_ERROR);

            throw new ValidatorException(msg);
        }
    }
}
```

Zadatak 17

Realizovati veb aplikaciju korišćenjem JSF tehnologije i formu koja je prikazana na slici. Forma sadrži 3 tekstualna polja i padajuću listu. Padajuća lista treba da obrađuje događaj promene vrednosti (*ValueChange event*) i da promenom vrednosti iz liste se menjaju labele forme, u skladu sa lokalizacijom (npr. ako se odabere za zemlju SAD, promeniti jezik na engleski).

REŠENJE

```
//veb stranica: index.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
```

```
      <title>#{poruke.naslov}</title>
   </h:head>

   <h:body>
      <h:form>
         <span class="emphasis">#{poruke.unosforme}</span>
         <h:panelGrid columns="2">
            #{poruke.ulica}
            <h:inputText value="#{form.ulica}"/>

            #{poruke.grad}
            <h:inputText value="#{form.grad}"/>

            #{poruke.regija}
            <h:inputText value="#{form.regija}"/>

            #{poruke.drzava}
            <h:selectOneMenu value="#{form.zemlja}" onchange="submit()"
                  valueChangeListener="#{form.promena}">
               <f:selectItems value="#{form.zemlje}" var="loc"
                  itemLabel="#{loc.displayCountry}"
                  itemValue="#{loc.country}"/>
            </h:selectOneMenu>
         </h:panelGrid>
         <h:commandButton value="#{poruke.potvrda}"/>
      </h:form>
   </h:body>
</html>

//izvorni fajl: etf/RegisterForm.java
package etf;

import java.io.Serializable;
import java.util.Locale;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;
import javax.faces.event.ValueChangeEvent;

@ManagedBean(name="form")
@SessionScoped
public class RegisterForm implements Serializable {
   private String ulica;
   private String grad;
   private String regija;
   private String zemlja;

   private static final Locale[] zemlje = { Locale.ROOT, Locale.US,
```

```
                                                Locale.CANADA};
   public Locale[] getZemlje() { return zemlje; }

   public void setUlica(String novaVr) { ulica = novaVr; }
   public String getUlica() { return ulica; }

   public void setGrad(String novaVr) { grad = novaVr; }
   public String getGrad() { return grad; }

   public void setRegija(String novaVr) { regija = novaVr; }
   public String getRegija() { return regija; }

   public void setZemlja(String novaVr) { zemlja = novaVr; }
   public String getZemlja() { return zemlja; }

   public void promena(ValueChangeEvent event) {
      for (Locale loc : zemlje)
         if (loc.getCountry().equals(event.getNewValue()))
            FacesContext.getCurrentInstance().getViewRoot().setLocale(loc);
   }
}
```

```
//izvorni fajl: etf/poruke.properties
naslov=Koriscenje dogadjaja za promenu vrednosti
unosforme=Molimo popunite adresu

ulica=Adresa
grad=Grad
regija=Pokrajina
drzava=Zemlja
potvrda=Potvrdi adresu
```

```
//izvorni fajl: etf/poruke_en_CA.properties
naslov=Using Value Change Events
unosforme=Please fill in your address

ulica=Address
grad=City
regija=Province
drzava=Country
potvrda=Submit address
```

```
//izvorni fajl: etf/poruke_en_US.properties
naslov=Using Value Change Events
unosforme=Please fill in your address

ulica=Address
grad=City
regija=State
```

```
drzava=Country
potvrda=Submit address
```

Zadatak 18

Realizovati JSF veb aplikaciju koja prikazuje mapu Srbije i koja putem osluškivača reaguje na klik miša, pa na osnovu koordinate koje su zadate otvara određeni region Srbije i podatke o tom regionu.

Date koordinate su:

- Grad Beograd (116,160,176,220)
- Podunavski (200,180,260,240)
- Kolubarski (40,180,90,230)
- Šumadijski (134,253,235,285)
- Zlatiborski (63,282,117,365)

REŠENJE

```
//veb stranica: index.xhtml
<?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>Regioni Srbije</title>
    </h:head>
    <h:body>
        <span class="instrukcije">#{poruke.instrukcije}</span>
        <h:form>
            <h:commandButton image="/resources/images/srbija-mapa.png"
                             styleClass="imageButton"
                             actionListener="#{srbija.uhvatiKlik}"
                             action="#{srbija.navigacija}"/>
        </h:form>
    </h:body>
</html>

//izvorni fajl: etf/Srbija.java
package etf;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.bean.RequestScoped;
import java.awt.Point;
import java.awt.Rectangle;
import java.util.Map;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;

@ManagedBean(name="srbija")
@RequestScoped
public class Srbija {
    private String izlaz = null;
    private Rectangle okrug1 = new Rectangle(116,160,176,220);
    private Rectangle okrug2 = new Rectangle(200,180,260,240);
    private Rectangle okrug3 = new Rectangle(40,180,90,230);
    private Rectangle okrug4 = new Rectangle(134,253,235,285);
    private Rectangle okrug5 = new Rectangle(63,282,117,365);

    public void uhvatiKlik(ActionEvent dogadjaj){
        FacesContext context = FacesContext.getCurrentInstance();
        String klijent = dogadjaj.getComponent().getClientId(context);
        Map<String, String> parametri =
                context.getExternalContext().getRequestParameterMap();
```

```
        int x = new Integer((String) parametri.get(klijent +
                                ".x")).intValue();
        int y = new Integer((String) parametri.get(klijent +
                                ".y")).intValue();

        izlaz = null;

        if(okrug1.contains(new Point(x, y)))
            izlaz = "beograd";

        if(okrug2.contains(new Point(x, y)))
            izlaz = "smederevo";

        if(okrug3.contains(new Point(x, y)))
            izlaz = "valjevo";

        if(okrug4.contains(new Point(x, y)))
            izlaz = "kragujevac";

        if(okrug5.contains(new Point(x, y)))
            izlaz = "uzice";
    }

    public String navigacija(){
        return izlaz;
    }
}
```

//izvorni fajl: etf/poruke.properties

```
instrukcije=Kliknite na određeni region (Beograd, Podunavski, Kolubarski,
Šumadijski, Zlatiborski)
mojasrbija=Moja majka Srbija
imeregiona1=Grad Beograd
grad1=Beograd
imeregiona2=Podunavski
grad2=Smederevo
imeregiona3=Kolubarski
grad3=Valjevo
imeregiona4=Šumadijski
grad4=Kragujevac
imeregiona5=Zlatiborski
grad5=Uzice
tekst1=Beograd, glavni i najveći grad Srbije...
tekst2=Smederevo se nalazi na obalama Dunava...
tekst3=Valjevo je grad u Srbiji Kolubarskog upravnog okruga...
tekst4=Kragujevac je sedište Šumadijskog okruga...
tekst5=Užice leži na obalama reke Đetinje...
indexLinkTekst=Vrati se na mapu Srbije
prestonica=Glavni grad regiona
```

```
//veb strana: beograd.xhtml
<?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>#{poruke.mojasrbija}</title>
    </h:head>
    <h:body>
        <h:form>
            <span class="region">#{poruke.imeregiona1}</span>
            <br/>
            <h:graphicImage library="images" name="#{poruke.grad1}.jpg"
                            styleClass="leftImage"/> <br/><br/>
            <h2>#{poruke.prestonica}: #{poruke.grad1}</h2>
            <span class="opis">#{poruke.tekst1}</span>
            <br/>
            <h:commandLink action="index"
                           styleClass="backLink">
                               #{poruke.indexLinkTekst}
            </h:commandLink>
        </h:form>
    </h:body>
</html>
```

Preostale veb strane: smederevo.xhtml, valjevo.xhtml, kragujevac.xhtml i uzice.xhtml izgledaju isto kao strana beograd.xhtml, uz izmenu podataka koji se čitaju iz *properties* fajla.

```
//CSS fajl: resources/css/styles.css
.body {
   background: #eee;
}
.instructions {
      font-style: italic;
      font-size: 1.2em;
}
.imageButton {
      margin: 10px;
      border-left: thin solid darkGray;
      border-bottom: thin solid darkGray;
      border-top: thin solid lightGray;
      border-right: thin solid lightGray;
}
.region {
   vertical-align: top;
   text-align: middle;
```

```
   font-style: italic;
   font-size: 2em;
}
.opis {
   vertical-align: top;
   text-align: left;
   font-size: 1em;
}

.leftImage {
   float: left;
   margin-right: 1em;
}

.backLink {
      font-style: italic;
}
```

Zadatak 19

U okviru JSF strane napraviti formu pomoću koje korisnik unosi korisničko ime, lozinku, dodatni tekst o sebi. Na istoj veb strani i napraviti dve ikonice u obliku zastava Velike Britanije i Nemačke, tako da odabirom određene zastave se menja prevod te veb strane, na osnovu fajlova za lokalizaciju ova dva jezika. Izborom britanske zastave dobija se engleski jezik, a izborom nemačke zasteve, veb strane se prevodi na nemački jezik. Prilikom pozivanja metode za promenu jezika, prosleđivanje parametra komponenti uraditi korišćenjem **<f:attribute>** taga.

REŠENJE

```
//veb stranica: index.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{poruke.naslovpocetna}</title>
   </h:head>
   <h:body>
      <h:form>
         <h:commandLink immediate="true"
                        actionListener="#{localeChanger.changeLocale}">
```

```
            <f:attribute name="atributJezik" value="de"/>
            <h:graphicImage library="images" name="de_flag.gif"
                            style="border: 0px; margin-right: 1em;"/>
         </h:commandLink>
         <h:commandLink immediate="true"
                        actionListener="#{localeChanger.changeLocale}">
            <f:attribute name="atributJezik" value="en"/>
            <h:graphicImage library="images" name="en_flag.gif"
                            style="border: 0px"/>
         </h:commandLink>
         <p style="font-style: italic;
                   font-size: 1.3em">#{poruke.indexPageTitle}</p>
         <h:panelGrid columns="2">
            #{poruke.labelaIme}
            <h:inputText value="#{korisnik.kime}"/>
            #{poruke.labelaLozinka}
            <h:inputSecret value="#{korisnik.lozinka}"/>
            #{poruke.labelaDodatniKomentar}
            <h:inputTextarea value="#{korisnik.osebi}" rows="5" cols="35"/>
            <h:commandButton value="#{poruke.potvrdiDugme}"
                             action="prosledi"/>
         </h:panelGrid>
      </h:form>
   </h:body>
</html>

//veb stranica: prosledi.xhtml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>#{poruke.naslovhvala}</title>
   </h:head>
   <h:body>
      <em>#{poruke.labelaIme}</em>
      #{korisnik.kime}
      <br/><br/>
      <em>#{poruke.labelaDodatniKomentar}</em>
      <pre>#{korisnik.osebi}</pre>
      <h:form>
        <h:commandLink action="index">#{poruke.labelaNazad}</h:commandLink>
      </h:form>
   </h:body>
</html>
```

```
//izvorni fajl: rs.etf/UserBean.java
package rs.etf;

import java.io.Serializable;
import javax.faces.bean.ManagedBean;
// or import javax.inject.Named;
import javax.faces.bean.SessionScoped;
// or import javax.enterprise.context.SessionScoped;

@ManagedBean(name="korisnik")
@SessionScoped
public class UserBean implements Serializable {
   private String kime;
   private String lozinka;
   private String osebi;

    public String getKime() {
        return kime;
    }

    public void setKime(String kime) {
        this.kime = kime;
    }

    public String getLozinka() {
        return lozinka;
    }

    public void setLozinka(String lozinka) {
        this.lozinka = lozinka;
    }

    public String getOsebi() {
        return osebi;
    }

    public void setOsebi(String osebi) {
        this.osebi = osebi;
    }
}

//izvorni fajl: rs.etf/LocaleChanger.java
package rs.etf;

import java.util.Locale;
import java.util.Map;
import javax.faces.bean.ApplicationScoped;

import javax.faces.bean.ManagedBean;
```

```
import javax.faces.bean.RequestScoped;
import javax.faces.bean.SessionScoped;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;

@ManagedBean // or @Named
@ApplicationScoped
public class LocaleChanger {
   public void changeLocale(ActionEvent event) {
      UIComponent komponenta = event.getComponent();
      String jezik = getLanguageCode(komponenta);
      FacesContext.getCurrentInstance()
         .getViewRoot().setLocale(new Locale(jezik));
   }
   private String getLanguageCode(UIComponent komponenta) {
      Map<String, Object> atributi = komponenta.getAttributes();
      return (String) atributi.get("atributJezik");
   }
}

//izvorni fajl: rs.etf/poruke.properties
naslovpocetna=Using Command Links
naslovhvala=Thank you!
indexPageTitle=Please enter the following personal information
labelaIme=Name:
labelaLozinka=Password:
labelaDodatniKomentar=Please tell us about yourself:
osebi=Some information about you:
potvrdiDugme=Submit your information
labelaNazad=Back

//izvorni fajl: rs.etf/poruke_de.properties
naslovpocetna=Ein Beispiel für Textfelder und Textgebiete
naslovhvala=Vielen Dank!
indexPageTitle=Bitte geben Sie die folgenden persönlichen Daten ein
labelaIme=Name:
labelaLozinka=Paßwort:
labelaDodatniKomentar=Bitte erzählen Sie etwas über sich:
osebi=Ihre persönlichen Daten:
potvrdiDugme=Daten absenden
labelaNazad=Zurück
```

Zadatak 20

U okviru JSF strane napraviti formu pomoću koje korisnik unosi korisničko ime, lozinku, dodatni tekst o sebi. Na istoj veb strani i napraviti dve ikonice u obliku zastava Velike Britanije i Nemačke, tako da odabirom određene zastave se menja prevod te veb strane, na osnovu fajlova za lokalizaciju ova dva jezika. Izborom britanske zastave dobija se engleski jezik, a izborom nemačke zasteve, veb strane se prevodi na nemački jezik. Prilikom pozivanja metode za promenu jezika, prosleđivanje parametra komponenti uraditi korišćenjem **<f:param>** taga.

REŠENJE

U odnosu na rešenje zadatka 19, gde smo koristili f:attribute tag, tako što smo komponenti korisničkog interfejsa na serveru prosledili atribut komponente, sada ćemo prikazati prenos podataka u zahtevu, putem f:param taga.

Izmene koda su u komponenti prilikom dodavanja parametra na veb strani:

```
//veb strana: index.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{poruke.naslovpocetna}</title>
   </h:head>
   <h:body>
      <h:form>
         <h:commandLink immediate="true"
                        action="#{localeChanger.changeLocale}">
            <f:param name="atributJezik" value="de"/>
            <h:graphicImage library="images" name="de_flag.gif"
                            style="border: 0px; margin-right: 1em;"/>
         </h:commandLink>
         <h:commandLink immediate="true"
                        action="#{localeChanger.changeLocale}">
            <f:param name="atributJezik" value="en"/>
            <h:graphicImage library="images"
                  name="en_flag.gif" style="border: 0px"/>
         </h:commandLink>
         <p style="font-style: italic; font-size: 1.3em">
            #{poruke.indexPageTitle}
         </p>
```

```
      <h:panelGrid columns="2">
            #{poruke.labelaIme}
            <h:inputText value="#{korisnik.kime}"/>
            #{poruke.labelaLozinka}
            <h:inputSecret value="#{korisnik.lozinka}"/>
            #{poruke.labelaDodatniKomentar}
            <h:inputTextarea value="#{korisnik.osebi}" rows="5" cols="35"/>
            <h:commandButton value="#{poruke.potvrdiDugme}"
                             action="prosledi"/>
         </h:panelGrid>
      </h:form>
   </h:body>
</html>
```

Takođe, pošto se ne koristi actionListener, parametri se dohvataju iz konteksta, a ne iz komponente:

```
//izvorni fajl: rs.etf/LocaleChanger.java
package rs.etf;

import java.util.Locale;
import java.util.Map;
import javax.faces.bean.ApplicationScoped;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.RequestScoped;
import javax.faces.bean.SessionScoped;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;

@ManagedBean
@ApplicationScoped
public class LocaleChanger {
   public void changeLocale() {
    FacesContext context = FacesContext.getCurrentInstance();
    String jezik = getLanguageCode(context);
    context.getViewRoot().setLocale(new Locale(jezik));
   }
   private String getLanguageCode(FacesContext context) {
      Map<String, String> parametri =
                  context.getExternalContext().getRequestParameterMap();
      return (String) parametri.get("atributJezik");
   }
}
```

Zadatak 21

Isti zadatak br. 19, odnosno 20, rešiti prosleđivanjem parametra kroz metodu.

REŠENJE

```
//veb strana: index.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{poruke.naslovpocetna}</title>
   </h:head>
   <h:body>
      <h:form>
         <h:commandLink action="#{localeChanger.changeLocale('de')}">
            <h:graphicImage library="images" name="de_flag.gif"
                            style="border: 0px; margin-right: 1em;"/>
         </h:commandLink>
         <h:commandLink action="#{localeChanger.changeLocale('en')}">
            <h:graphicImage library="images" name="en_flag.gif"
                            style="border: 0px"/>
         </h:commandLink>
         <p style="font-style: italic; font-size: 1.3em">
                  #{poruke.indexPageTitle}
         </p>
         <h:panelGrid columns="2">
            #{poruke.labelaIme}
            <h:inputText value="#{korisnik.kime}"/>
            #{poruke.labelaLozinka}
            <h:inputSecret value="#{korisnik.lozinka}"/>
            #{poruke.labelaDodatniKomentar}
            <h:inputTextarea value="#{korisnik.osebi}" rows="5" cols="35"/>
            <h:commandButton value="#{poruke.potvrdiDugme}"
                             action="prosledi"/>
         </h:panelGrid>
      </h:form>
   </h:body>
</html>
```

```
//izvorni fajl: rs.etf/LocaleChanger.java
package rs.etf;

import java.util.Locale;
import java.util.Map;
import javax.faces.bean.ApplicationScoped;
import javax.faces.bean.ManagedBean;
import javax.faces.context.FacesContext;

@ManagedBean // or @Named
@ApplicationScoped
public class LocaleChanger {
   public void changeLocale(String izmenaJezika) {
    FacesContext context = FacesContext.getCurrentInstance();
    context.getViewRoot().setLocale(new Locale(izmenaJezika));
   }
}
```

Zadatak 22

Isti zadatak br. 19, odnosno 20, rešiti korišćenjem f:setPropertyActionListener taga.

REŠENJE

```
//veb strana: index.xhtml
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{poruke.naslovpocetna}</title>
   </h:head>
   <h:body>
      <h:form>
         <h:commandLink immediate="true"
                        action="#{localeChanger.changeLocale}">
            <f:setPropertyActionListener
                  target="#{localeChanger.kodjezika}" value="de"/>
            <h:graphicImage library="images" name="de_flag.gif"
                            style="border: 0px; margin-right: 1em;"/>
         </h:commandLink>
          <h:commandLink immediate="true"
                         action="#{localeChanger.changeLocale}">
```

```
            <f:setPropertyActionListener
                  target="#{localeChanger.kodjezika}" value="en"/>
      <h:graphicImage library="images" name="en_flag.gif"
                      style="border: 0px"/>
         </h:commandLink>
         <p style="font-style: italic;
                   font-size: 1.3em">#{poruke.indexPageTitle}</p>
         <h:panelGrid columns="2">
            #{poruke.labelaIme}
            <h:inputText value="#{korisnik.kime}"/>
            #{poruke.labelaLozinka}
            <h:inputSecret value="#{korisnik.lozinka}"/>
            #{poruke.labelaDodatniKomentar}
            <h:inputTextarea value="#{korisnik.osebi}" rows="5" cols="35"/>
            <h:commandButton value="#{poruke.potvrdiDugme}"
                             action="prosledi"/>
         </h:panelGrid>
      </h:form>
   </h:body>
</html>

//izvorni fajl: rs.etf/LocaleChanger.java
package rs.etf;

import java.util.Locale;
import java.util.Map;
import javax.faces.bean.ApplicationScoped;
import javax.faces.bean.ManagedBean;
import javax.faces.context.FacesContext;

@ManagedBean // or @Named
@ApplicationScoped
public class LocaleChanger {
    private String kodjezika;

    public void changeLocale() {
        FacesContext context = FacesContext.getCurrentInstance();
        context.getViewRoot().setLocale(new Locale(kodjezika));
   }

    public void setKodjezika(String novikod) {
        this.kodjezika = novikod;
    }
}
```

Diskusija oko zadataka 19 - 22:

U kontekstu ovog primera, metod prosleđivanja parametara putem metode je prividno najbolji izbor (zad. 21). Najlakši za implementaciju i razumevanje je pristup preko sPAL taga (setPropertyActionListener taga, zad. 22). Tagom sPAL mi postavljamo vrednost atributa unutar pozadinskog Java bina. Ipak. sPAL tag nalazi manju primenu kod JSF2.0 + aplikacija. Međutim, f:param i f:attribute imaju njihovu primenu, tako što se setuju parametri preko zahteva serveru ili se u komponentu dodaju atributi. Jedini nedostatak kod ova dva pristupa za slanje podataka serveru je što mi moramo manuelno da izvučemo podatke iz zahteva servera (parametre) ili iz komponente (atribut).

Zadatak 23

Realizovati JSF aplikaciju pomoću koje korisnik može da se uloguje na sistem. Korisnik putem forme na veb strani index.xhtml unosi korisničko ime i lozinku (tekstualna polja) i pritiska dugme "ULOGUJ SE". Kada korisnik unese neke podatke na početnoj formi, taj korisnik postaje ulogovan na sistem, sve dok ne pritisne dugme izloguj se.

a) Na početnoj strani, pre forme, realizovati događaj, koji proverava da li je korisnik možda već ulogovan na sistem i ako jeste preusmeriti ga na veb stranu login.xhtml, a ako nije, ostaviti korisnika na istoj početnoj veb strani.

b) Nakon logovanja (uraditi samo logiku logovanja, bez provere korisničkog imena i lozinke) korisnik treba da ima mogućnost da se izloguje iz sistema i da ode na drugu veb stranu preko koje će proveriti da li je određeni datum validan ili nije. Forma za validaciju datuma treba da ima tri zasebna tekstualna polja za dan, mesec i godinu i dugme "POTVRDI". Na strani za unos datuma realizovati sistemski događaj koji radi validaciju datuma pozivom Javinog bina, koji je takođe potrebno realizovati sa svim pratećim metodama koji proveravaju uneti datum.

REŠENJE

```
//veb strana: index.xhtml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
   <f:view>
```

```
      <f:event type="preRenderView" listener="#{user.checkLogin}"/>
      <h:head>
          <title>#{poruke.welcome}</title>
      </h:head>
      <h:body>
         <h3><h:outputText value="Dobrodošli, #{user.name}!" /></h3>
         <h:form>
            <h:commandButton value="#{poruke.logout}"
               action="#{user.logout}" />
            <h:commandButton value="#{poruke.next}" action="enterDate" />
         </h:form>
      </h:body>
   </f:view>
</html>

//izvorni fajl: rs.etf/UserBean.java
package rs.etf;
import java.io.Serializable;

import javax.faces.application.ConfigurableNavigationHandler;
import javax.inject.Named;
import javax.enterprise.context.SessionScoped;
import javax.faces.context.FacesContext;
import javax.faces.event.AbortProcessingException;
import javax.faces.event.ComponentSystemEvent;

@Named("user") // or @ManagedBean(name="user")
@SessionScoped
public class UserBean implements Serializable {
   private String name = "";
   private String password;
   private boolean loggedIn;

   public String getName() { return name; }
   public void setName(String newValue) { name = newValue; }

   public String getPassword() { return password; }
   public void setPassword(String newValue) { password = newValue; }

   public boolean isLoggedIn() { return loggedIn; }

   public String login()  {
      loggedIn = true;
      return "index";
   }

   public String logout() {
      loggedIn = false;
      return "login";
```

```
    }

    public void checkLogin(ComponentSystemEvent event) {
        if (!loggedIn) {
            FacesContext context = FacesContext.getCurrentInstance();
            ConfigurableNavigationHandler handler =
                    (ConfigurableNavigationHandler)
                    context.getApplication().getNavigationHandler();
                    handler.performNavigation("login");
        }
    }
}
```

//veb strana: login.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>#{poruke.welcome}</title>
    </h:head>
    <h:body>
        <h:form>
            <h3>Molimo unesite Vaše korisničko ime i lozinku:</h3>
            <table>
                <tr>
                    <td>Korisničko ime:</td>
                    <td><h:inputText id="name" value="#{user.name}" /></td>
                </tr>
                <tr>
                    <td>Lozinka:</td>
                    <td>
                      <h:inputSecret value="#{user.password}" required="true" />
                    </td>
                </tr>
            </table>
            <p><h:commandButton value="Uloguj se" action="#{user.login}" /></p>
        </h:form>
    </h:body>
</html>
```

//veb strana: enterDate.xhtml

```
<?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:f="http://java.sun.com/jsf/core"
      xmlns:h="http://java.sun.com/jsf/html">
```

```
   <h:head>
      <h:outputStylesheet library="css" name="styles.css"/>
      <title>#{poruke.title}</title>
   </h:head>
   <h:body>
      <h:form>
         <h1>#{poruke.enterDate}</h1>
         <h:panelGrid id="date" columns="2">
            <f:event type="postValidate" listener="#{bb.validateDate}"/>
            #{poruke.day}
            <h:inputText id="day" value="#{bb.day}" size="2"
                         required="true"/>

            #{poruke.month}
            <h:inputText id="month" value="#{bb.month}"
                         size="2" required="true"/>

            #{poruke.year}
            <h:inputText id="year" value="#{bb.year}"
                         size="4" required="true"/>
         </h:panelGrid>
         <h:message for="date" styleClass="errorMessage"/>
         <br/>
         <h:commandButton value="#{poruke.submit}" action="result"/>
         <h:commandButton value="#{poruke.back}"
                          action="index" immediate="true"/>
      </h:form>
   </h:body>
</html>

//izvorni fajl: rs.etf/BackingBean.java
package rs.etf;

import java.io.Serializable;
import javax.faces.application.FacesMessage;
import javax.inject.Named;
import javax.enterprise.context.SessionScoped;
import javax.faces.component.UIComponent;
import javax.faces.component.UIForm;
import javax.faces.component.UIInput;
import javax.faces.context.FacesContext;
import javax.faces.event.ComponentSystemEvent;
import javax.faces.validator.ValidatorException;

@Named("bb") // or @ManagedBean(name="bb")
@SessionScoped
public class BackingBean implements Serializable {
   private int day;
   private int month;
```

```
   private int year;

   public int getDay() { return day; }
   public void setDay(int newValue) { day = newValue; }

   public int getMonth() { return month; }
   public void setMonth(int newValue) { month = newValue; }

   public int getYear() { return year; }
   public void setYear(int newValue) { year = newValue; }

   public void validateDate(ComponentSystemEvent event) {
      UIComponent source = event.getComponent();
      UIInput dayInput = (UIInput) source.findComponent("day");
      UIInput monthInput = (UIInput) source.findComponent("month");
      UIInput yearInput = (UIInput) source.findComponent("year");
      int d = ((Integer) dayInput.getLocalValue()).intValue();
      int m = ((Integer) monthInput.getLocalValue()).intValue();
      int y = ((Integer) yearInput.getLocalValue()).intValue();
      if (!isValidDate(d, m, y)) {
         FacesMessage message = com.corejsf.util.Messages.getMessage(
            "rs.etf.poruke", "invalidDate", null);
         message.setSeverity(FacesMessage.SEVERITY_ERROR);
         FacesContext context = FacesContext.getCurrentInstance();
         context.addMessage(source.getClientId(), message);
         context.renderResponse();
      }
   }

   private static boolean isValidDate(int d, int m, int y) {
      if (d < 1 || m < 1 || m > 12) return false;
      if (m == 2) {
         if (isLeapYear(y)) return d <= 29;
         else return d <= 28;
      }
      else if (m == 4 || m == 6 || m == 9 || m == 11)
         return d <= 30;
      else
         return d <= 31;
   }

   private static boolean isLeapYear(int y) {
      return y % 4 == 0 && (y % 400 == 0 || y % 100 != 0);
   }
}

//veb strana: result.xhtml

<?xml version="1.0" encoding="UTF-8"?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>#{poruke.title}</title>
   </h:head>
   <h:body>
      <h:form>
         <p>#{poruke.validDate}</p>
         <p><h:commandButton value="#{poruke.back}" action="index"/></p>
      </h:form>
   </h:body>
</html>

//izvorni fajl: rs.etf/poruke.properties
welcome=Dobrodošli u primer za sistemske događaje
title=Validacija izmedju komponenti
enterDate=Molimo unesite datum.
day=Dan
month=Mesec
year=Godina
submit=Potvrdi
validDate=Uneli ste validan datum.
invalidDate=Nevalidan datum.
invalidDate_detail=Uneti datum nije validan.
logout=Izloguj se
next=Dalje
back=Nazad
```

Zadatak 24

Realizovati JSF veb stranu sa veb formom preko koje korisnik unosi korisničko ime i lozinku. Ukoliko je korisničko ime manje od 6 karaktera ili veće od 10 karaktera, korišćenjem tehnologije AJAX i taga f:ajax, prikazati odgovarajuću poruku korisniku, neposredno ispod forme.

REŠENJE

```
//veb strana: index.xhtml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
   <h:head>
      <title>#{poruke.loginWindowTitle}</title>
   </h:head>

   <h:body style="background: #fefeef">
      <h:form>
         <h3>#{poruke.loginHeading}</h3>
         <h:panelGrid columns="2">
            #{poruke.namePrompt}
                        <h:inputText value="#{korisnik.kime}" id="name">
                          <f:ajax event="keyup" render="echo"/>
                        </h:inputText>

            #{poruke.passwordPrompt}
            <h:inputSecret value="#{korisnik.lozinka}"
                           id="password" size="8"/>

            <h:commandButton value="#{poruke.loginButtonText}"
                             action="dobrodosli"/>
         </h:panelGrid>

         <p><h:outputText id="echo" value="#{korisnik.provera()}"/></p>
      </h:form>
   </h:body>
</html>

//veb strana: dobrodosli.xhtml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
   <h:head>
      <title>#{poruke.loginWindowTitle}</title>
   </h:head>
   <h:body style="background: #fefeef">
         <h3>#{poruke.greeting} #{korisnik.kime}!</h3>
           <h3>#{poruke.pass}: #{korisnik.lozinka}</h3>
   </h:body>
</html>
```

```
//izvorni fajl: rs.etf/UserBean.java
package rs.etf;
import java.io.Serializable;
import javax.inject.Named;
import javax.enterprise.context.SessionScoped;

@Named("korisnik") // or @ManagedBean(name="user")
@SessionScoped
public class UserBean implements Serializable {
   private String kime = "";
   private String lozinka;

    public String getKime() {
        return kime;
    }

    public void setKime(String kime) {
        this.kime = kime;
    }

    public String getLozinka() {
        return lozinka;
    }

    public void setLozinka(String lozinka) {
        this.lozinka = lozinka;
    }

    public String provera(){
        if(this.kime.length() < 6)
            return "Korisničko ime je prekratko! Min 6 karaktera";
        else if(this.kime.length() >= 10)
            return "Korisničko ime je predugačko! Max 10 karaktera";
        else
            return "";
    }
}

//izvorni fajl: rs.etf/poruke.properties
loginWindowTitle=JSF 2 i Ajax
loginHeading=Molimo ulogujte se
namePrompt=Korisničko ime:
passwordPrompt=Lozinka:
loginButtonText=ULOGUJ SE
greeting=Dobrodošli
pass=Vaša lozinka je
```

Zadatak 25

Realizovati JSF aplikaciju "Telefonski imenik". Korisnici sistema loguju se preko forme za logovanje korišćenjem svojih kredencijala (korisničko ime, lozinka). Ukoliko korisnik postoji u sistemu, treba mu dozvoliti dalji rad, a ukoliko ne postoji treba ispisati odgovarajuću poruku sa greškom.

Kada se uloguje, korisnik vidi svoje ime i prezime i sadržaj svog telefonskog imenika (ime, broj telefona i da li je u pitanju mobilni). Korisnik treba da ima mogućnost da za svaki broj telefona iz imenika, vidi detalje, na posebnoj veb strani, da obriše postojeći broj iz imenika i da doda novi telefonski broj (nije potrebno vršiti nikakve provere formata telefonskog broja). Takođe, korisnik treba na zasebnim veb stranama da ima mogućnost pregleda svih fiksnih telefonskih brojeva iz svog imenika i mogućnost pregleda svih mobilnih telefonskih brojeva iz svog imenika. Korisniku treba omogućiti i da se izloguje iz sistema.

Podatke o korisnicima i telefonskim imenicima čuvati u Javinim kolekcijama i mapama. Svaki korisnik treba da ima jedan telefonski imenik.

REŠENJE

```
//veb strana: index.xhtml
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>Telefoni</title>
    </h:head>
    <h:body>
        <h1>TELEFONSKI IMENIK - Forma za logovanje korisnika</h1>
        <span style="color:red">#{controlerBean.poruka}</span>
        <h:form>
            <h:panelGrid columns="2">
                <h:outputText value="Korisnicko ime"/>
                <h:inputText value="#{controlerBean.username}"/>
                <h:outputText value="Lozinka"/>
                <h:inputSecret value="#{controlerBean.password}"/>
            </h:panelGrid>
            <h:commandButton value="LogIn"
                             action="#{controlerBean.login}"/>
        </h:form>
    </h:body>
</html>

//izvorni fajl: podaci/DB.java
//Javina klasa sa kolekcijama koja se koristi umesto baze podataka
package podaci;

import beans.Korisnik;
import beans.Telefon;
import java.util.HashMap;
import java.util.LinkedList;
import javax.faces.bean.ApplicationScoped;

@ApplicationScoped
public class DB {

    private static HashMap<String, LinkedList<Telefon>> mapaTelefona
            = new HashMap<String, LinkedList<Telefon>>();
    private static HashMap<String, Korisnik> mapaKorisnika
            = new HashMap<String, Korisnik>();

    static {

        mapaKorisnika.put("drasko", new Korisnik("drasko123", "Drazen",
                          "Draskovic", "draskovic@etf.rs", null));
```

```
        mapaKorisnika.put("nbosko", new Korisnik("bosko123", "Bosko",
                          "Nikolic", "nbosko@etf.rs", null));

        LinkedList<Telefon> telefoni1 = new LinkedList<Telefon>();
        telefoni1.add(new Telefon("Baka Smiljka", "011-111-111", false, 1));
        telefoni1.add(new Telefon("Sanja", "064-123-3333", true, 2));
        telefoni1.add(new Telefon("Dr Petrovic", "011-555-333", false, 3));
        telefoni1.add(new Telefon("Dekan", "063-202-2002", true, 4));

        mapaTelefona.put("drasko", telefoni1);

        LinkedList<Telefon> telefoni2 = new LinkedList<Telefon>();
        telefoni2.add(new Telefon("Zaki", "011/222222", false, 1));
        telefoni2.add(new Telefon("Cmilos", "060/223322", true, 2));
        telefoni2.add(new Telefon("Dekan", "063/2022002", true, 3));
        mapaTelefona.put("nbosko", telefoni2);
    }

    public static Korisnik findKorisnikByUsername(String username) {
        return mapaKorisnika.get(username);
    }

    public static LinkedList<Telefon> getTelefoniByUsername(String username)
    {
        return mapaTelefona.get(username);
    }

    public static boolean isKorisnik(String username, String password) {
        Korisnik k = mapaKorisnika.get(username);
        return (k != null) && k.getPassword().equals(password);
    }
}
```

```
//izvorni fajl: beans/Korisnik.java
//Javin bean za podatke o korisniku sistema
package beans;
import java.util.LinkedList;

public class Korisnik {

    private String username;
    private String password;
    private String ime;
    private String prezime;
    private String email;
    private LinkedList<Telefon> telefoni;

    public Korisnik() {
    }
```

```
public Korisnik(String password, String ime, String prezime,
                String email, LinkedList<Telefon> telefoni) {
    this.password = password;
    this.ime = ime;
    this.prezime = prezime;
    this.email = email;
    this.telefoni = telefoni;
}

public String getEmail() {
    return email;
}

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

public String getIme() {
    return ime;
}

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

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public String getPrezime() {
    return prezime;
}

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

public LinkedList<Telefon> getTelefoni() {
    return telefoni;
}

public void setTelefoni(LinkedList<Telefon> telefoni) {
    this.telefoni = telefoni;
}
```

```
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}

//izvorni fajl: beans/Telefon.java
//Javin bean za podatke o telefonu
package beans;

public class Telefon {
    private String ime;
    private String telefon;
    private boolean mob;
    private int IDTel;

    public Telefon(String ime, String telefon, boolean mob, int IDTel) {
        this.ime = ime;
        this.telefon = telefon;
        this.mob = mob;
        this.IDTel = IDTel;
    }

    public Telefon() {
    }

    public int getIDTel() {
        return IDTel;
    }

    public void setIDTel(int IDTel) {
        this.IDTel = IDTel;
    }

    public String getIme() {
        return ime;
    }

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

    public boolean isMob() {
        return mob;
    }
```

```
    public void setMob(boolean mob) {
        this.mob = mob;
    }

    public String getTelefon() {
        return telefon;
    }

    public void setTelefon(String telefon) {
        this.telefon = telefon;
    }
}

//izvorni fajl: controlers/Controler.java
//Kontroler koji predstavlja ManagedBean
package controlers;

import beans.Korisnik;
import beans.Telefon;
import podaci.DB;
import java.util.LinkedList;
import java.util.List;
import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;
import javax.faces.context.FacesContext;
import javax.faces.model.SelectItem;
import javax.servlet.http.HttpSession;

@ManagedBean(name = "controlerBean")
@SessionScoped
public class Controler {

    private String poruka;
    private String username;
    private String password;
    private Korisnik korisnik;
    private Telefon izabraniTelefon;
    private Telefon noviTelefon = new Telefon();
    private List<SelectItem> mobItems = new LinkedList<SelectItem>();

    public Controler() {
        mobItems.add(new SelectItem(true, "DA"));
        mobItems.add(new SelectItem(false, "NE"));
    }

    public String login() {
        String sledecaStranica = null;
        if (DB.isKorisnik(username, password)) {
```

```
            korisnik = DB.findKorisnikByUsername(username);
            korisnik.setTelefoni(DB.getTelefoniByUsername(username));
            sledecaStranica = "sviTelefoni";
            poruka = "";
        } else {
            poruka = "Ne postoji traženi korisnik u sistemu!";
        }
        return sledecaStranica;
    }

    public String logOut() {
        FacesContext context = FacesContext.getCurrentInstance();
        HttpSession session = (HttpSession)
                        context.getExternalContext().getSession(false);
        session.invalidate();
        return "index";
    }

    public String detalji(Telefon tel) {
        this.izabraniTelefon = tel;
        return "detalji";
    }

    public void dodajTelefon() {
        this.korisnik.getTelefoni().add(noviTelefon);
        this.noviTelefon = new Telefon();
    }

    public void brisi() {
        this.korisnik.getTelefoni().remove(izabraniTelefon);
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
```

```
    public Korisnik getKorisnik() {
        return korisnik;
    }

    public void setKorisnik(Korisnik korisnik) {
        this.korisnik = korisnik;
    }

    public Telefon getIzabraniTelefon() {
        return izabraniTelefon;
    }

    public void setIzabraniTelefon(Telefon izabraniTelefon) {
        this.izabraniTelefon = izabraniTelefon;
    }

    public Telefon getNoviTelefon() {
        return noviTelefon;
    }

    public void setNoviTelefon(Telefon noviTelefon) {
        this.noviTelefon = noviTelefon;
    }

    public List<SelectItem> getMobItems() {
        return mobItems;
    }

    public void setMobItems(List<SelectItem> mobItems) {
        this.mobItems = mobItems;
    }

    public String getPoruka() {
        return poruka;
    }

    public void setPoruka(String poruka) {
        this.poruka = poruka;
    }

}
```

//veb strana: svitelefoni.xhtml

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

```
<h:head>
    <title>Telefoni</title>
    <h:outputStylesheet library="css" name="newcss.css"/>
</h:head>
<h:body>
    <h:form id="form">
        <h:panelGrid columns="2">
            <h:outputText value="Ime"/>
            <h:outputText value="#{controlerBean.korisnik.ime}"/>
            <h:outputText value="Prezime"/>
            #{controlerBean.korisnik.prezime}
        </h:panelGrid>

        <h:messages />

        <h:dataTable value="#{controlerBean.korisnik.telefoni}"
                     var="tel" styleClass="tabela" id="tabela">
            <f:facet name="caption">
                <h:outputText value="Vas telefonski imenik"/>
            </f:facet>
            <h:column headerClass="header">
                <f:facet name="header">
                    <h:outputText value="Ime"/>
                </f:facet>
                #{tel.ime}
                <f:facet name="footer">
                  <h:inputText value="#{controlerBean.noviTelefon.ime}"
                   required="true"  requiredMessage="Niste uneli ime!"/>
                </f:facet>
            </h:column>
            <h:column headerClass="header">
                <f:facet name="header">
                    <h:outputText value="Broj telefona"/>
                </f:facet>
                <h:inputText value="#{tel.telefon}"/>
                <f:facet name="footer">
                    <h:inputText
                       value="#{controlerBean.noviTelefon.telefon}"/>
                </f:facet>
            </h:column>
            <h:column headerClass="header">
                <f:facet name="header">
                    <h:outputText value="Mobilni"/>
                </f:facet>
                #{tel.mob ? "DA" : "NE"}
                <f:facet name="footer">
                    <h:selectOneMenu
                       value="#{controlerBean.noviTelefon.mob}">
                     <f:selectItems value="#{controlerBean.mobItems}"/>
```

```
                </h:selectOneMenu>
            </f:facet>
        </h:column>
        <h:column>
            <f:facet name="header">
                <h:outputText value="Detalji"/>
            </f:facet>
            <h:commandLink value="detalji"
                           action="detalji" immediate="true">
                <f:setPropertyActionListener
                      target="#{controlerBean.izabraniTelefon}"
                      value="#{tel}"/>
            </h:commandLink>
        </h:column>
        <h:column>
            <f:facet name="header">
                <h:outputText value="Brisi"/>
            </f:facet>
            <h:commandLink value="brisi"
                action="#{controlerBean.brisi}" immediate="true">
                <f:setPropertyActionListener
                   target="#{controlerBean.izabraniTelefon}"
                   value="#{tel}"/>
            </h:commandLink>
        </h:column>
        <h:column>
            <f:facet name="header">
                <h:outputText value="Detalji dugme"/>
            </f:facet>
            <h:commandButton value="dugme detalji"
                            action="#{controlerBean.detalji(tel)}"
                            immediate="true">
            </h:commandButton>
        </h:column>
        <h:column>
            <f:facet name="header">
                <h:outputText value="Dodaj"/>
            </f:facet>
            <f:facet name="footer">
                <h:commandButton value="dodaj"
                   action="#{controlerBean.dodajTelefon}"/>
            </f:facet>
        </h:column>
    </h:dataTable>

    <h:commandButton value="Izmenite"/>
</h:form>
```

```
        <h:form>
            <h:panelGrid columns="1">
                <h:commandLink value="Fiksni telefoni"
                   action="fiksniTelefoni" immediate="true"/>
                <h:commandLink value="Mobilni telefoni"
                   action="mobilniTelefoni"/>
                <h:commandLink action="#{controlerBean.logOut}">
                  Log out
                 </h:commandLink>
            </h:panelGrid>
        </h:form>
    </h:body>
</html>
```

//veb strana: fiksniTelefoni.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
    <h:head>
        <title>Telefoni</title>
    </h:head>
    <h:body>
        <h:dataTable value="#{controlerBean.korisnik.telefoni}" var="tel">
            <f:facet name="caption">
                <h:outputText value="Vas telefonski imenik"/>
            </f:facet>
            <h:column>
                <f:facet name="header">
                    <h:outputText value="Ime"/>
                </f:facet>
                <h:outputText value="#{tel.ime}" rendered="#{!tel.mob}"/>
            </h:column>
            <h:column>
                <f:facet name="header">
                    <h:outputText value="Broj telefona"/>
                </f:facet>
                <h:outputText value="#{tel.telefon}"
                              rendered="#{!tel.mob}"/>
            </h:column>
        </h:dataTable>
    </h:body>
</html>
```

```
//veb strana: mobilniTelefoni.xhtml
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core">
    <h:head>
        <title>Telefoni</title>
    </h:head>
    <h:body>
        <h:dataTable value="#{controlerBean.korisnik.telefoni}" var="tel">
            <f:facet name="caption">
                <h:outputText value="Vas telefonski imenik"/>
            </f:facet>
            <h:column>
                <f:facet name="header">
                    <h:outputText value="Ime"/>
                </f:facet>
                <h:outputText value="#{tel.ime}" rendered="#{tel.mob}"/>
            </h:column>
            <h:column>
                <f:facet name="header">
                    <h:outputText value="Broj telefona"/>
                </f:facet>
                <h:outputText value="#{tel.telefon}"
                              rendered="#{tel.mob}"/>
            </h:column>
        </h:dataTable>
    </h:body>
</html>

//veb strana: detalji.xhtml
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
    <h:head>
        <title>Telefoni</title>
    </h:head>
    <h:body>
        <h:panelGrid columns="2">
            <h:outputText value="Ime" />
            <h:outputText value="#{controlerBean.izabraniTelefon.ime}"/>
            <h:outputText value="Broj telefona"/>
            <h:outputText
                  value="#{controlerBean.izabraniTelefon.telefon}"/>
            <h:outputText value="Mobilni"/>
```

```
            <h:outputText
               value="#{controlerBean.izabraniTelefon.mob ? 'DA' : 'NE'}"/>
        </h:panelGrid>

        <h:form>
            <h:commandLink action="sviTelefoni" value="Svi telefoni"/>
        </h:form>
    </h:body>
</html>

//CSS fajl: resources/css/newcss.css
.tabela {
    border-width: 3px;
    border-style: solid;
}

.header {
    background-color: cadetblue;
}
```