14. domaća zadaća; JAVA, Akademska godina 2013./2014.; FER

Uvod

U okviru ove domaće zadaće razvit ćete web-aplikaciju *radionice* koja koristi bazu podataka koja sve svoje podatke čuva u nekoliko tekstovnih datoteka. Konceptualno, zadaća je slična aplikaciji koju smo razvili na prethodnom predavanju – jedino je mogućnostima nešto bogatija. Za potrebe rješavanja napravite novi Eclipseov projekt i u njemu definirajte standardnu strukturu projekta koju smo koristili kod svih dosadašnjih web-aplikacija. Opremite projekt odgovarajućom skriptom `build.xml` koja između ostaloga nudi i cilj `war` čijim će pokretanjem nastati web-arhiva `radionice.war`. Ručnim kopiranjem te arhive u Tomcatov direktorij `webapps` mora se dobiti web-aplikacija koja radi! Pročitajte na zadnjoj stranici upute **više o roku za predaju**.

Struktura baze podataka

Prilikom rada s bazom podataka očekuje se da na svim mjestima gdje Vam je baza potrebna slijedite isti obrazac uporabe: (1) učitaj bazu, (2) pretraži/izmijeni bazu, (3) po potrebi pohrani bazu.

Osnovni razred koji predstavlja bazu podataka ove aplikacije je `hr.fer.zemris.web.radionice.RadioniceBaza`.

Razred treba imati jednu statičku metodu:

```
public static RadioniceBaza ucitaj(String direktorij);
```

Razred također treba imati nestatičku metodu zaduženu za snimanje baze:

```
public void snimi();
```

Ideja je da s bazom možete raditi na sljedeći način:

```
String direktorij = ... // neka staza
RadioniceBaza baza = RadioniceBaza.ucitaj(direktorij);
 ... sada koristimo bazu ...
 .. pretpostavimo da smo radili izmjene: tada trebamo:
baza.snimi();
```

Da bi ovo bilo moguće, implementacija baze mora biti takva da u konstruktoru zapamti stazu kako bi se kasnije mogla snimiti na isto mjesto pozivom metode `snimi()`. Razlog zašto metode za učitavanje baze primaju stazu do direktorija leži u činjenici da se podatci baze zapisuju u više datoteka. Ako prilikom čitanja baze datoteke ne postoje, postupiti kao da su prisutne ali prazne. Prilikom snimanja baze, datoteke se ili stvaraju (ako ne postoje) ili gaze (ako postoje). Također, dodajte razredu RadioniceBaza metodu `snimi(String direktorij)` kojom ćete tražiti snimanje baze ali u direktorij koji ste upravo dali a ne u onog iz kojeg je baza početno učitana.

Dio aplikacije koji gradimo treba omogućiti ovlaštenom korisniku da definira radionice (engl. *Workshops*) koje se nude polaznicima neke znanstvene konferencije. Pri tome svaka radionica ima sljedeći skup podataka:

Podatak	Opis

ID	Jedinstveni identifikator radionice; tip: pozitivni Long.
Naziv	Naziv radionice: string maksimalne duljine 40 znakova; ne smije biti prazan.
Datum	Datum održavanja radionice; mora biti formata godina-mjesec-dan, npr. 2014-06-15. Mora biti zadan.
Oprema	**Nula, jedan ili više** elemenata opreme koju radionica treba; primjerice, radionica bi mogla trebati projektor, laserski pokazivač te bijelu ploču.
Trajanje	Koliko je predviđeno trajanje radionice. Mora biti zadano, i može biti **jedna** od triju vrijednosti: dvosatna, poludnevna ili cjelodnevna.
Publika	Tko je ciljana publika koja se smije prijaviti na radionicu. Mora biti **jedna ili više** od sljedećih opcija: početnici, prosječno iskusni, vrlo iskusni.
MaksPolaznika	Koliko se najviše polaznika smije prijaviti za radionicu. Mora biti zadano, i mora biti broj iz raspona [10, 50].
EMail	E-mail adresa na koju polaznici mogu dobiti više informacija. Obavezno prisutno.
Dopuna	Tekst koji daje dodatne upute o radionici. Ne mora biti zadan.

Kako bi se izbjeglo da se konkretni tekstovi opcija koje se nude u bazu zapisuju više puta (i time naprave različite pogreške), svaki skup opcija koji se nudi za pojedini atribut (podatak) radionice modeliran je zasebnom datotekom – konkretno, `oprema.txt`, `publika.txt` i `trajanje.txt`. Svaka od ovih datoteka u jednom retku definira jednu opciju. Pri tome je u prvom stupcu naveden jedinstveni identifikator opcije a u drugom stupcu naziv opcije. Elementi jednog retka odvojeni su znakom tab. Ogledni primjer ovih datoteka dan je u nastavku.

`oprema.txt`

```
12	Laptop
21	Stolno računalo
22	Projektor
27	Laserski pokazivač
29	Bijela ploča
37	Grafoskop
38	Zvučnici
```

`publika.txt`

```
1	Početnici
2	Prosječno iskusni
3	Vrlo iskusni
```

`trajanje.txt`

```
2	Dvosatna radionica
6	Poludnevna radionica
7	Cjelodnevna radionica
```

Jednu opciju bilo koje od ovih datoteka modelirajte razredom `Opcija` koji ima dva `property`-ja: `String id` te `String vrijednost` - prvi je samo za čitanje a drugoga je moguće i čitati i mijenjati. Iako se id ovdje pohranjuje kao String, morate provjeriti/osigurati da je on ujedno legalan pozitivni long (da ga se može konvertirati).
Zapis o jednoj radionici modelirajte razredom `Radionica`. Taj bi razred trebao imati sljedeće *property*-je:

```
Long id;
String naziv;
String datum;
Set<Opcija> oprema;
Opcija trajanje;
Set<Opcija> publika;
Integer maksPolaznika;
String email;
String dopuna;
```

Opcije moraju imati definiran prirodni poredak prema njihovim identifikatorima. Implementacije skupova koje se koriste u razredu `Radionica` morale bi biti takve da omogućavaju iteriranje elemenata njihovim prirodnim poretkom.

Pohrana podataka o konkretnim radionicama rješena je uporabom nekoliko datoteka. Datoteka `radionice.txt` za svaku radionicu sadrži jedan redak u kojem su navedeni svi atributi koji mogu imati nula ili jednu vriejdnost (dakle, atributi čija vrijednost može biti nezadana, ili koji mogu imati jednu konkretnu vrijednost). Struktura jednog retka datoteke `radionice.txt` stoga je:

ID tab *Naziv* tab *Datum* tab *MaksPolaznika* tab *Trajanje* tab *EMail* tab *Dopuna*

Pri tome stupac *Trajanje* sadrži samo identifikator opcije iz datoteke `trajanja.txt` – ne i vrijednost te opcije.

Za pohranu informacija koja je publika prikladna za radionicu odnosno koju opremu treba radionica napravljene su dvije pomoćne datoteke: `radionice_publika.txt` i `radionice_oprema.txt`; obje sadrže retke oblika *identifikator radionice* tab *identifikator opcije*. Primjer ovih datoteka prikazan je u nastavku.

`radionice_publika.txt`

```
1	1
1	3
2	3
3	1
3	3
```

`radionice_oprema.txt`

```
1	12
1	22
2	21
2	22
2	29
3	12
3	27
3	37
3	38
```

Primjerice, iz datoteke `radionice_publika.txt` vidimo vidimo da je radionica čiji je identifikator 1 prikladna za publiku definiranu opcijama čiji su identifikatori 1 i 3 (prva dva retka), da je radionica čiji je identifikator 2 prikladna za publiku definiranu opcijom čiji je identifikator 3 (treći redak) te da je radionica čiji je identifikator 3 prikladna za publiku definiranu opcijama čiji su identifikatori 1 i 3 (četvrti i peti redak). Slično možemo interpretirati i datoteku `radionice_oprema.txt`.

Važno je napomenuti da u datoteci `radionice.txt` stupac *Dopuna* koristi dodatno kodiranje: budući da je dopuna potencijalno višelinijski tekst a sve informacije o jednoj radionici moraju biti dane u jednom retku, svaki prijelom retka (ascii 10) zamijenjen je slijedom znakova \ i n; također, svaka pojava znaka tab zamijenjena je slijedom znakova \ i t jer se tab koristi kao graničnik stupaca u datoteci; svaka pojava znaka \ zamijenjena je slijedom znakova \ i \ jer \ koristimo kao escape. Prilikom čitanja iz datoteke pazite da ove slijedove znakova zamijenite natrag prikladnim originalnim znakovima te da prilikom pohrane u datoteku ponovno napravite opisano kodiranje.

Problem 1.

Podesite Eclipse tako da izvorne kodove sprema u `src/main/java`. Napišite sve prethodno opisane razrede. Na Ferku u dodatcima ovoj domaćoj zadaći stavio sam i početnu verziju baze (ZIP datoteka koja sadrži direktorij baza sa svim prethodno opisanim datotekama). U Vašem Eclipseovom projektu napravite raspakirajte tu arhivu direktno u korijenski direktorij (tada ćete imati direktorije `src`, `web`, `weblib` i `baza`). Dodajte još jedan direktorij za izvorne kodove: `src/test/java`. U njega smjestite *junit* test koji je opisan u nastavku.

Razred `RadionicaBaza` bi trebao imati sljedeće privatne kolekcije (koje čuvaju podatke pročitane iz datoteka):

- radionice (sve radionice koje su definirane)
- oprema (sva oprema koja je definirana u `oprema.txt`)
- publika (sva publika koje je definirana u `publika.txt`)
- trajanje (svo trajanje koje je definirano u `trajanje.txt`)

Svim ovim kolekcijama razred `RadionicaBaza` bi trebao dati pristup i svojim klijentima, ali samo kroz nepromjenjive poglede (ili kopije) tako da izvana korisnik ne može mijenjati te kolekcije (s druge strane, može mijenjati objekte koji su u kolekciji – ako neki objekt ima *property* koji je promjenjiv, naravno da ga korisnik izvana može promijeniti).

Razredu `RadionicaBaza` dodajte i privatnu metodu `void provjeriIspravnostOpcija();` koja će protrčati kroz sve radionice i provjeriti da su u trenutku poziva sve opcije postavljene na postojeće vrijednosti koje su učitane iz datoteka; ako je sve OK, metoda ne radi ništa, inače baca iznimku (napravite runtime iznimku `InconsistentDatabaseException` i prilikom njezinog izazivanja u konstruktoru dajte opis – što ne valja. Prilikom snimanja baze automatski se prvo mora pozvati ova metoda.

Sada napravite junit test čiji je ovo pseudokod:
baza = učitajBazu("./baza");
d = stvori_privremeni_direktorij // koristiti Files.createTempDirectory(...)
baza.snimi(d)
usporedite da su sve datoteke u "./baza" i d identične (uz zanemarivanje \n ili \r\n ili \r)
obrisi_privremeni_direktorij d
uspjeh ako su sve bile iste, inače fail

Potom napravite sljedeći junit test:
baza = učitajBazu("./baza");
radionica = baza.dohvatiRadionicu(1);
radionica.getOprema().add(new Opcija(101, "USB stick"));
d = stvori_privremeni_direktorij // koristiti Files.createTempDirectory(...)
baza.snimi(d)
obrisi_privremeni_direktorij d
uspjeh ako je baza.snimi bacila iznimku, fail inače

Pazite kako ćete napisati ova dva testa – nije prihvatljivo da test baci iznimku (i da kažete da je to očekivano ponašanje) jer će tada na disku ostati privremeni direktorij. Umjesto toga sami morate uhvatiti pogrešku te temeljem toga je li bilo iznimke ili ne test pustiti dalje ili ga srušiti (postoji metoda za to).

Ako ste ovo uspjeli, sada u `RadionicaBaza` dodajte još i metodu koja snima radionicu:

```
public void snimi(Radionica r);
```

Implementacija je konceptualno slična kao u adresaru s predavanja – ako `id` nije postavljen, dodjeljuje mu se novi jedinstveni, inače se ažurira objekt pod postojećim `id`-jem. Ako `r` ima postavljen `id` ali takav trenutno u bazi ne postoji, tretirati to kao dodavanje nove radionice s tim `id`-jem.

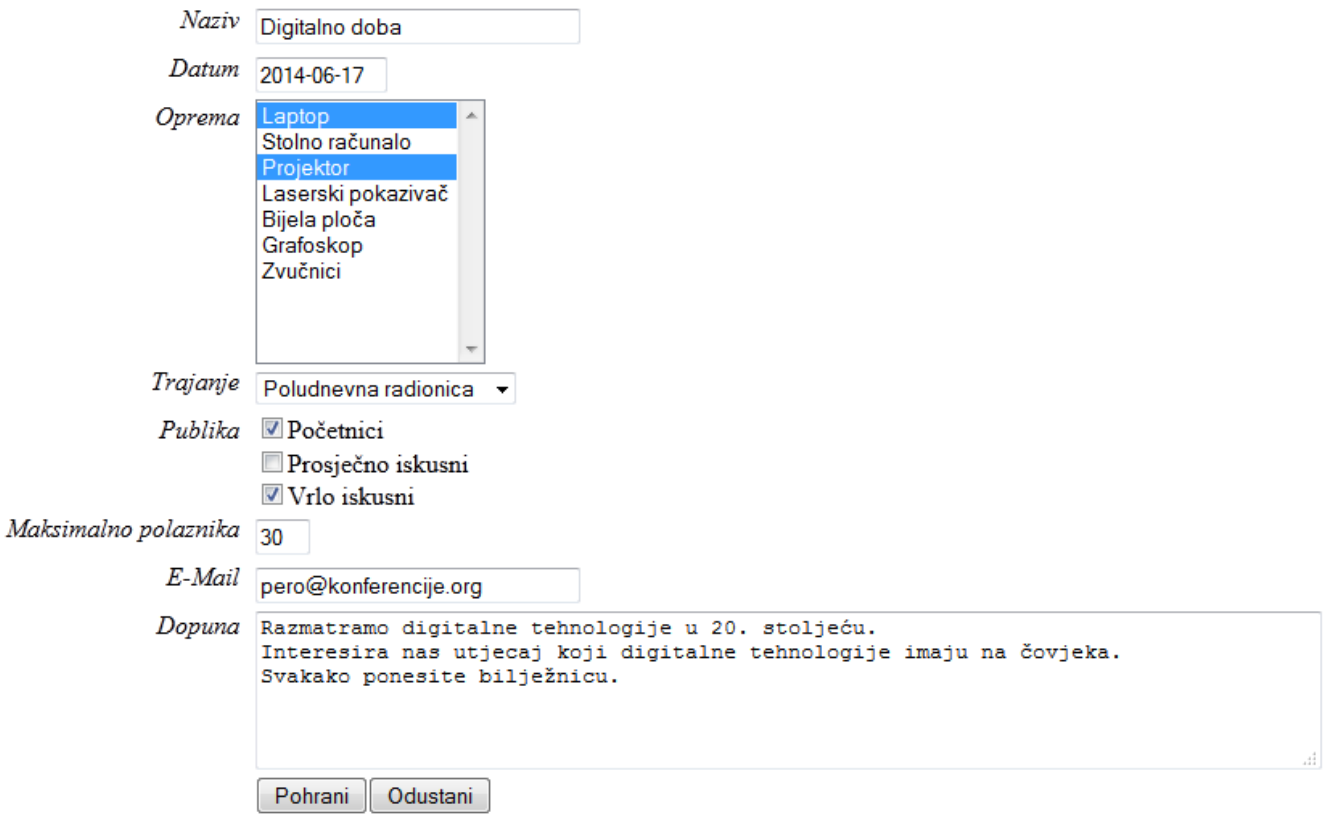
Problem 2.

Napravite servlet `/listaj` koji generira prikaz postojećih radionica. Ispisuje se samo naziv radionice i datum održavanja. Prikaz treba biti sortiran po datumu pa unutar toga po nazivu radionice.

Dodajte na stranicu i linkove na `/new` (stvaranje nove radionice) te `/edit` (uređivanje postojeće radionice, treba `id` kao parametar).

Napravite te servlete i pripadni JSP.

Potom napravite i servlet `/save` koji umeće novu / sprema modificiranu radionicu. Prikaz obrasca bi morao biti što bliži ovome danom u nastavku.



Uočite da se na formularu nigdje ne vidi identifikator – to je zato što se on mora slati kao skriveni parametar.

Za dohvat vrijednosti parametara koji mogu poprimiti više vrijednosti koristiti `HttpServletRequest.getParameterValues(...)` koja vraća polje vrijednosti.

http://docs.oracle.com/javaee/6/api/javax/servlet/ServletRequest.html#getParameterValues%28java.lang.String%29

Prilikom izrade web-aplikacije zadaća `ant`-cilja war bit će da automatski u `WEB-INF` direktorij *war*-arhive doda i kopiju vašeg direktorija `./baza`. Ako ručno kopirate sadržaj web-aplikacije u Tomcat, sjetite se da iskopirate i taj direktorij. Naime, očekivano je ponašanje da u web-aplikaciji postoji direktorij `WEB-INF/baza` sa svim datotekama Vaše baze.

Pomoć: kod komponenata koje rade s opcijama, kao vrijednosti opcija (ono što će natrag biti poslano kada se šalje formular) koristite identifikatore opcija. Kao opis koristite vrijednost opcije (bilo da se radi o checkboxovima, radiobuttonima kojih ovdje nema ili o listama/comboboxovima).

Pomoć 2: servleti koje namapirate na `/new`, `/edit` te `/save` (ako nastupi greška) JSP-u na crtanje ne smiju poslati samo atribut tipa `RadionicaForm` – u atribute osim ovoga treba zakvačiti i kolekciju sve postojeće opreme, kolekciju mogućih trajanja te kolekciju postojeće vrste publike. Prilikom renderiranja property-ja radionice koja poprimaju vrijednosti iz tih kolekcija tada trčite po svim stavkama kolekcije i po potrebi stavku renderirate kao označenu/selektiranu ako se njezina vrijednost poklopi s onime što je zapisano u radionici koju prikazujete. Svaka akcija koja uspješno promjeni podatke u bazi mora završiti sa *sendRedirect* – nikako ne smije generirati HTML stranicu za korisnika.

Pomoć 3: Razred `RadionicaForm` preslikava razred `Radionica` samo što su mu svi jednostavni property-ji stringovi. Ono što su u razredu `Radionica` kolekcije opcija, u `RadionicaForm` bit će kolekcije njihovih identifikatora.

Pomoć 4: nemojte si kompliciratiživot – u okviru ove zadaće treba napraviti dodavanje/uređivanje samo radionica. Ako želite isprobati kako se program ponaša ako dodate nove opcije, to modificirajte direktno u pripadnim datotekama.

U slučaju da se prilikom pohrane podataka (`/save`) ustanove bilo kakve pogreške, korisnika je potrebno vratiti na prikaz formulara kako bi pogreške mogao ispraviti. Pri tome je uz svaku stavku potrebno ispisati odgovarajući tekst pogreške (kao na predavanju).

Pokušajte formatiranje prikaza riješiti jednostavnom tablicom (dva stupca bez bordera) i malo CSS za italic font, desno poravnavanje odnosno vertikalno poravnavanje prema gore, širinu prvog stupca i slično.

Problem 3.

Zaštitite pristup stranicama za uređivanje / dodavanje radionica. Pretpostavimo sljedeće: izjavu *postoji trenutni korisnik* smatrat ćemo ispunjenom ako u sjedničkoj mapi vezanoj uz trenutni http-zahtjev koji obrađujemo postoji ključ "`current.user`" čija je vrijednost primjerak razreda `User`. Napravite razred `User` s property-jima `login`, `zaporka`, `ime`, `prezime`. Ako postoji trenutni korisnik, onda ćemo automatski smatrati da je on i autoriziran za sve potrebne operacije.

Modificirajte JSP koji generira stranicu za `/listaj` tako da linkove za dodavanje nove radionice i za uređivanje postojećih radionica prikazuje samo postoji trenutni korisnik. Također, ako postoji trenutni korisnik, na vrhu stranice ispišite njegovo ime i prezime, te mu ponudite link na servlet `/logout`. Ako trenutni korisnik ne postoji, onda na vrhu stranice ispišite "Anonimni korisnik" i ponudite mu link na servlet `/login`.

Također, modificirajte servlete `/new`, `/edit` i `/save` tako da, ako ih se pozove a nema trenutnog korisnika, umjesto normalne obrade korisniku prikažu poruku pogreške (da nije autoriziran) i ponude mu link za povratak na naslovnicu aplikacije.

Napravite servlet `/login` koji korisnik će korisnika proslijediti na Vaš JSP na kojem ćete mu nacrtati formular koji će mu ponuditi da upiše svoj login i zaporku. Akcija koja obrađuje taj formular je opet servlet `/login`. Naime, servlet treba napisati tako da provjeri je li dobio parametar *username*. Ako ga je dobio, onda čita i parametar *password* te provjerava postoji li takav korisnik; ako ne postoji ili ako parametri nisu ispravni, registrira poruku pogreške i prosljeđuje dalje JSP-u na iscrtavanje formulara uz ispis pogreške; ako nije dobio parametar *username*, onda samo prosljeđuje dalje JSP-u na iscrtavanje formulara.

Struktura obrade u servletu `/login` je sljedeća:

```
ako imam parametar username:
  User korisnik = provjeri(username, password);
  ako korisnik != null
    u sjednicu dodaj par: (current.user, korisnik)
    redirect na /listaj
    kraj
  inače
    registriraj pogrešku "username ili password je pogrešan"
    forward na /WEB-INF/pages/Login.jsp
    kraj
  kraj-ako
inače
    forward na /WEB-INF/pages/Login.jsp
    kraj
kraj-ako
```

Struktura obrade u servletu `/logout` je sljedeća:

```
poništi trenutnu sjednicu //hint: session.invalidate()
redirect na /listaj
```

U servletu /login metodu `provjeri` napišite tako da ima jednog hardkodiranog korisnika: "aante", "tajna", "Ante", "Anić". U stvarnosti, ova bi metoda popis korisnika dohvatila iz datoteke (ili još češće, iz baze podataka, preko LDAP sustava ili na neki treći način). Za potrebe ove zadaće nećemo s time komplicirati – još je ostalo zadaća do kraja :-) Pseudokod metode stoga je:

```
ako username="aante" i password="tajna"
  return new User("aante", "tajna", "Ante", "Anić");
else
  return null
kraj
```

Konačno, osigurajte da ako korisnik umjesto *http://server:port/radionice/listaj* pozove samo *http://server:port/radionice/*, da će to automatski obraditi isti servlet (to smo već objasnili – sjetite se što treba napraviti).

Please note. You can consult with your peers and exchange ideas about this homework *before* you start actual coding. Once you open you IDE and start coding, consultations with others (except with me) will be regarded as cheating. You can not use any of preexisting code or libraries for this homework (whether it is yours old code or someones else), unless it is one of the libraries I explicitly mentioned in previous problems. Document your code!

In order to solve this homework, create a blank Eclipse Java Project and write your code inside. Once you are done, export project as a ZIP archive and upload this archive on Ferko before the deadline. Do not forget to lock your upload or upload will not be accepted.

Equip the project with appropriate `build.xml`. You must add `war` target that will automatically create complete WAR file.

You are required to create at least two unit tests which I have described in this document and your homework must pass them both.

Before uploading, please make double sure that a working WAR can be build from console by `ant`. Please take special care not to embed any absolute paths in your code or in scripts – different users will have tomcat installed at different places. Your project name must be `HW14-`*yourJMBAG*.

The official deadline for uploading and locking this homework **is June, 14th 2014. at 08:00 AM**, since the next lecture will build upon the material you learned while solving this homework. However, considering the fact that the following week is the last week of lectures for regular courses and lab-exercises, I will accept homework uploads until Monday, 16th 2014. at 11:59 PM and this will be the date displayed by Ferko. Please note, I strongly suggest that you finish this homework before official deadline.