5. laboratorijska vježba: Java Servlet i PHP

i domaća zadaća

Cilj vježbe

Izrada dinamičke Web stranice koja dohvaća XML podatke isporučene od strane izrađenog Servlet poslužiteljskog programa.

Priprema za vježbu

Proučiti – Java Servlete:

- Osnove izrade Java Servleta
  - Java Servlet Programming (O'Reilly) – poglavlja 1 – 5
    http://www.unix.com.ua/orelly/java-ent/servlet/index.htm
  - The Java EE 5 Tutorial - Java Servlet Technology, Chapter 4 – dijelovi koji se odnose na vježbu (stvaranje, inicijalizacija, odgovor)
- Konfiguriranje web.xml za Servlete
- Pokretanje Servleta na poslužitelju

Zadatak za vježbu

Potrebno je izraditi **Java Servlet** koji **isporučuje podatke u obliku XML**. Java Servlet je potrebno izraditi na temelju gotove aplikacije iz 4. laboratorijske vježbe koju je potrebno preraditi prema Servlet specifikaciji kako bi odgovarala na potrebne HTTP metode.

Dodatno je potrebno doraditi **PHP skriptu** (koristeći rješenje 3. laboratorijske vježbe) za pretraživanje strukturiranog zapisa podataka u XML obliku koje sad dobiva od Java Servleta, a ne iz XML datoteke.

Java Servlet bi trebao na poziv GET metode HTTP zahtjeva dohvatiti podatke iz tekstualne datoteke, transformirati ih u XML podatke (sve iz 4. lab. vježbe), te ih u XML obliku proslijediti PHP skripti (bez bilo kakvog filtriranja/sortiranja podataka).

Napomena: Ako niste adekvatno riješili zadatak iz 4. lab. vježbe, sada bi ga trebalo doraditi s obzirom da se većinom isti kôd koristi za dohvat podataka iz tekstualne datoteke i prosljeđivanje PHP skripti u ovoj vježbi.

Za izradu metode ***doGet()*** možete iskoristiti programski kôd iz prije implementiranog razreda *Citac*, odnosno njegove metode *main()*. U toj metodi (kao o prije) treba otvoriti datoteke čija imena su sad zadana **kao konstante** zapisane u samom Servletu i definirane u metodi *init()* (a ne kao parametar komandne linije kako je bilo u 4. lab. vježbi).

Na osnovu učitanih podataka iz datoteka, metoda (isto kao u 4. lab. vježbi) treba stvoriti kolekciju objekata (*java.util.ArrayList*) koja treba sadržavati sve objekte tipa osnovnog elementa. Za svaki zapis iz datoteke treba stvoriti odgovarajući objekt i dodati ga u kolekciju ili dopuniti postojeći objekt (koji se već nalazi u kolekciji) novim informacijama. U kolekciji podataka smije postojati samo jedan isti objekt osnovnog elementa ako se radi o istom zapisu

(npr. kod osobe Ivo Ivić samo jedna takva osoba koja treba sadržavati sve podatke o osobi kao što je primjerice više telefonskih brojeva koji su u tekstualnoj datoteci bili zapisani u više redaka). Razrede objekata koje treba međusobno uspoređivati (isto kao u 4. lab. vježbi) trebaju nadjačati javnu metodu *equals(Object obj)* razreda *java.lang.Object* koja vraća istinosnu vrijednost (*boolean*). Sve ulazne podatke (kao i dosad) treba pri pretvorbi u XMl pohraniti u formatu definiranom inačicom laboratorijskih vježbi. Nedostajuće obvezne podatke nadomjestiti po volji odabranim vrijednostima.

Kad su svi podaci uneseni u kolekciju, treba generirati izlaznu XML strukturu podataka slijednom obradom (pomoću metode *toXML()*) svih objekata pohranjenih u kolekciji objekata i poslati je na *HttpServletResponse*. U metodi je naravno potrebno u izlaznu XML strukturu podataka zapisati i korijenski element nadređen osnovnom elementu svake inačice (npr. za inačicu 'Imenik' to je element <imenik> koji sadrži elemente <osoba>).

Predaja vježbe i predaja domaće zadaće

Predaja domaće zadaće podrazumijeva:

1. stvaranje na poslužitelju *pinus* odgovarajuće strukture direktorija
2. stvaranje konfiguracijske datoteke `web.xml` i postavljanje u odgovarajući direktorij na poslužitelju *pinus*
3. opcionalno stvaranje testne datoteke test.jsp postavljanje u odgovarajući direktorij na poslužitelju *pinus*

Nakon što su predane domaće zadaće, asistent će restartati poslužitelj kako bi se osvježile sve konfiguracije (objašnjeno kasnije).

Predaja vježbe podrazumijeva:

1. predočavanje datoteka s izvornim kôdom, eventualna objašnjenja
2. pokretanje Servleta ručnim pokretanjem dohvata GET metodom HTTP zahtjeva sa zadanim izvornim podacima
3. pokretanje PHP skripte koja koristi XML podatke dobivene iz Servleta

Vježba se predaje na poslužitelju *pinus* korištenjem aplikacijskog poslužitelja Tomcat. Poslužitelj Tomcat pokreće asistent te ga po potrebi restarta.

Da bi vježba radila potrebno je konfigurirati poslužitelj Tomcat za korištenje odgovarajućeg Servleta. Za to je potrebno vlastitom direktoriju **na poslužitelju *pinus* kreirati** sljedeću strukturu poddirektorija:

web-app
/WEB-INF
/classes **ovdje idu .class fileovi (sa strukturom paketa)**
/lib **ovdje idu .jar fileovi**

Direktorij `WEB-INF` sadrži konfiguraciju aplikacije, a u poddirektoriju `classes` se trebaju nalaziti *.class* datoteke potrebne za rad aplikacije. Ako ste pakirali aplikaciju u *.jar* datoteku, kreirajte i poddirektorij `lib` unutar `WEB-INF` direktorija i tamo stavite *.jar* datoteku. Napomena: U slučaju problema, možete otpakirati *.jar* datoteku i razrede postaviti u direktorij `classes`.

Za test da li ste dobro izradili direktorij i da li im poslužitelj Tomcat može pristupiti, u direktoriju `web-app` možete postaviti datoteku `test.jsp` sljedećeg sadržaja, koja služi kao proba:

```
<HTML><BODY>
Vrijeme na posluzitelju: <%= new java.util.Date() %>
</BODY></HTML>
```

Da bi vaš Servlet bio dostupan, potrebno je **u direktorij `WEB-INF` postaviti datoteku `web.xml`** koja konfigurira web aplikaciju (odnosno vaš Servlet) u poslužitelju Tomcat. Ta datoteka sadrži opis Servleta, spoj na odgovarajući razred i mapiranje Servleta na URL.

Sadržaj navedene datoteke `web.xml` bi trebao biti sličan sljedećem (obojane dijelove nadopunite proizvoljnim nazivima razreda Servleta, samog naziva Servleta i mapiranja):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_3.
dtd">
<web-app>
        <servlet>
                <servlet-name>NazivMojegServleta</servlet-name>
                <servlet-class>hr.fer.rasip.or.MojServlet</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name> NazivMojegServleta </servlet-name>
                <url-pattern>/servlet</url-pattern>
        </servlet-mapping>
</web-app>
```

Nakon postavljanja konfiguracije u `web.xml` datoteci, potrebno je restartati poslužitelj Tomcat (**što ne možete sami, već će se to obavljati nekoliko puta dnevno od strane asistenta**).

Jednom podešena web aplikacija (vaš Servlet) može se osvježavati, znači ako su promijenjeni razredi mogu se stavljati nove inačice istih razreda u direktorij `classes`, ali **NE** i ako je promijenjena datoteka `web.xml`, odnosno ako su promijenjeni nazivi razreda (potreban je restart poslužitelja Tomcat).

Osvježavanje web aplikacije (Servleta)

Stranica za osvježavanje web aplikacije nalazi se na adresi http://www.fer.hr/predmet/otvrac_a/upravljanje_web_aplikacijom . Za osvježavanje aplikacije potrebno je prijaviti se na FER web, nakon čega bi sustav trebao prikazati korisnički račun na *pinusu* – ako oznaka računa nije ispravna, javite se asistentima.

Nakon promjene stanja aplikacije gumbom **List** možete u popisu podešenih aplikacija provjeriti stanje svoje. Veza Link na aplikaciju vodi do direktorija vaše aplikacije na poslužitelju Tomcat. Ako dobijete grešku 5xx, vaša aplikacija nije podešena unutar poslužitelja Tomcat.

Ako ste izradili testnu datoteku test.jsp prema gornjem primjeru, pristupom URL-u http://pinus.cc.fer.hr:8080/~vase_korisnicko_ime/test.jsp, trebali bi dobiti stranicu sa ispisom trenutnog vremena poslužitelja.

```
Vrijeme na posluzitelju: Mon May 12 10:31:14 MEST 2008
```

Serlvetu pristupate preko adrese http://pinus.cc.fer.hr:8080/~korisnicko_ime/servlet (prema parametru <`url-pattern`> u datoteci `web.xml`).

Dodatno: Aplikacija naravno može sadržavati i više Servleta, pa se tada u datoteci `web.xml` navode više puta elementi `<servlet>` i `<servlet-mapping>` za svaki Servlet, kao u sljedećem primjeru:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_3.
dtd">
<web-app>
        <servlet>
                <servlet-name>ImeMojegServleta</servlet-name>
                <servlet-class>hr.fer.rasip.or.RazredMojegServleta</servlet-class>
        </servlet>
        <servlet>
                <servlet-name>ImeDrugogServleta</servlet-name>
                <servlet-class>hr.fer.rasip.or.RazredDrugogServleta</servlet-class>
        </servlet>

        <servlet-mapping>
                <servlet-name>ImeMojegServleta</servlet-name>
                <url-pattern>/mojservlet</url-pattern>
        </servlet-mapping>
        <servlet-mapping>
                <servlet-name>ImeDrugogServleta</servlet-name>
                <url-pattern>/drugiservlet</url-pattern>
        </servlet-mapping>
</web-app>
```

Implementacija Servleta

Servlet razred treba naslijediti `HttpServlet` kao u primjeru:

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

public class MyServlet extends HttpServlet {
...
```

Servlet ne treba prihvaćati nikakve parametre, niti ih preuzimati s dobivenog zahtjeva (HttpServletRequest).

Servlet treba u metodi `init`() definirati nazive datoteka iz koji se učitavaju podaci, slično kao u primjeru:

```
       String input1Filename;
       String input2Filename;

       public void init(ServletConfig config) throws ServletException {
              super.init(config);
              input1Filename = "aaa.txt";
              input2Filename = "bbb.txt";
       }
}
```

Servlet vraća XML podatke (XML dokument), a.povratni MIME tip za XML se postavlja pomoću metode `setContentType` kao u primjeru:

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
throws ServletException, IOException {

    ...

    response.setContentType("text/xml; charset=UTF-8");
    ...
```

Također je potrebno dohvatiti objekt `PrintWriter` nad odgovorom (`HttpServletResponse`) kako bi se mogli „ispisivati“ XML podaci te ispravno poslati podatke i na kraju zatvoriti tok podataka, kao u primjeru:

```
PrintWriter out = response.getWriter();
out.println("<?xml version=\"1.0\" encoding=\"utf-8\" standalone=\"no\"?>");
...
out.close();
```

Dohvat XML podataka GET metodom i PHP –a

Za dohvat podataka s nekog HTTP izvora pomoću GET metode u PHP-u možete koristiti metodu `file_get_contents` , npr.:

```
$a = file_get_contents("http://pinus.cc.fer.hr:8080/~korisnicko_ime/servlet");
```

Za inicijalizacija DOM i učitavanje XML dokumenta kao DOM stabla prihvatom podataka iz String-a možete (umjesto dosadašnje metode `load` koja je čitala datoteku s diska) koristiti metodu `loadXML`, npr.:

```
$dom = new DOMDocument();
$dom->loadXML( $a );
```

Ispitno gradivo vježbe

Ispitno gradivo uključuje sve navedeno u pripremi za vježbu, te detaljno razumijevanje napisanog rješenja i snalaženja u prepravcima istog.

Primjeri pitanja:

1. Objasnite implementaciju vašeg Servleta
2. Objasnite konfiguraciju u datoteci web.xml
3. Objasnite životni ciklus Servleta
4. Objasnite preinake u PHP-u

Linkovi i literatura

The Java EE 5 Tutorial - Java Servlet Technology (Chapter 4) - http://java.sun.com/javaee/5/docs/tutorial/doc/bnafd.html

Java Servlet Programming - http://www.unix.com.ua/orelly/java-ent/servlet/index.htm

Drugi zanimljivi linkovi:
Story of a Servlet: An Instant Tutorial - http://java.sun.com/products/servlet/articles/tutorial/

A Hello, World Servlet - http://www.caucho.com/resin-3.0/servlet/tutorial/helloworld/index.xtp

Servlet Essentials - http://www.novocode.com/doc/servlet-essentials/

Servlet-Tutorial - http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/

Servlet Basics - http://courses.coreservlets.com/Course-Materials/pdf/csajsp2/02-Servlet-Basics.pdf