14th homework; JAVA, Academic year 2017./2018.; FER

Introduction

For this homework you will integrate the structured web-application which is described in document "`12-uputa.txt`" which is available in Ferko's repository with a JDBC-based reimplementation of voting application which you have already developed as part of 13th homework.

Make a copy of web-application project that we developed during last lecture (based on Sekcija 5 from `12-uputa.txt`. Let the directory name of the project in Eclipse's workspace directory be `hw14-0000000000` (replace zeros with your JMBAG). Change the `pom.xml` so that groupID:artefactID be `hr.fer.zemris.java.jmbag0000000000:hw14-0000000000`. Set the `finalName` to `voting-app`. Leave the support for jetty-plugin, so that your app can be easily started. Import this project into Eclipse.

Once you have the project imported into Eclipse, remove:

- class Unos
- declared methods in DAO and implementation in SQLDAO for working with Unos
- servlet ListajKratko
- JSP ListajKratko.jsp

This way you will have all the infrastructure "set-up" but the project will be blank from any concrete data model and service implementation.

Problem 1.

Assume you have on your disposal a database on which you connect using the following URL:
`jdbc:derby://localhost:1527/votingDB;user=ivica;password=ivo`

The exact database name, host, port, user and password will be provided in properties file `dbsettings.properties` which must be directly in `WEB-INF` folder. This file must take the following form (the right side can vary):

```
host=localhost
port=1527
name=votingDB
user=ivica
password=ivo
```

To allow easier homework review, you must include this file in your homework with the exact content as given in the example above. Your application should read this file during connection-pool setup and use provided information; if this file is not present or any of properties is missing, throw an exception which will cause the web-application to be stopped (verify this). For obtaining the actual path to this file from your web-app, use:

http://docs.oracle.com/javaee/6/api/javax/servlet/ServletContext.html#getRealPath(java.lang.String)

Initialization of connection-pool and its destroying must be performed in appropriate web-application listener.

Assume you will work with two tables created in this database by the following commands:

```
CREATE TABLE Polls
    (id BIGINT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    title VARCHAR(150) NOT NULL,
    message CLOB(2048) NOT NULL
);

CREATE TABLE PollOptions
    (id BIGINT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    optionTitle VARCHAR(100) NOT NULL,
    optionLink VARCHAR(150) NOT NULL,
    pollID BIGINT,
    votesCount BIGINT,
    FOREIGN KEY (pollID) REFERENCES Polls(id)
);
```

During the web-application startup, verify if these tables exists in database; if not, send appropriate CREATE statements to create them (*but only if they do not already exists*); if they exist, do not try to delete them. You can google it up ("apache derby create table if not exists").

The first table (**Polls**) models concrete polls. Each row represents a different poll. Each poll will have its own poll ID. For example, to mimic your previous voting-application homework problem where everything was written in files, you would have a single entry in table **Polls** with *ID*=1 (or any other), *Title*="Glasanje za omiljeni bend:" te *Message*="Od sljedećih bendova, koji Vam je bend najdraži? Kliknite na link kako biste glasali!".

If during startup web-application determines that appropriate database tables do not exist, you should create them (as previously described). If you create missing tables (or determine that the table `Polls` is empty), you should also populate them with poll data which mimics the data used in homework 13, and some other poll data (so that you end up with two initial polls). Do not assume that the first INSERT will create a record with key 1; always ask for created keys and use this information. To check this behavior, once your code successfully creates the tables (and populates them), stop servlet container (Tomcat/Jetty), log to database using ij-console and delete all table contents (first verify how it works when using DELETE statements, and if your application on next start populates everything OK, try DROP TABLE statements and verify that tables are automatically created).

The table **PollOptions** contains options for each defined poll. If we assume that our poll had the ID with value 1, in **PollOptions** table we would have 7 rows with pollID set to 1 (attribute `votesCount` is not shown):

id	OptionTitle	optionLink	pollID
1	The Beatles	http://...	1
2	The Platters	http://...	1
3	The Beach Boys	http://...	1
4	The Four Seasons	http://...	1
5	The Marcels	http://...	1
6	The Everly Brothers	http://...	1
7	The Mamas And The Papas	http://...	1

Problem 2a.

Create a servlet which is mapped to `/servleti/index.html`. This servlet must obtain a list of defined polls and render it to user as a list of clickable links. When user clicks on a Poll title, the link that will be followed must be `/servleti/glasanje?pollID=x` where `x` is selected poll ID.

Problem 2b.

Create a servlet which is mapped to `/index.html`. This servlet must send redirection to clients browser so that clients browser as a follow-up action requests `/servleti/index.html` (in your web-app). For sending redirections please consult:

http://docs.oracle.com/javaee/6/api/javax/servlet/http/HttpServletResponse.html#sendRedirect(java.lang.String)

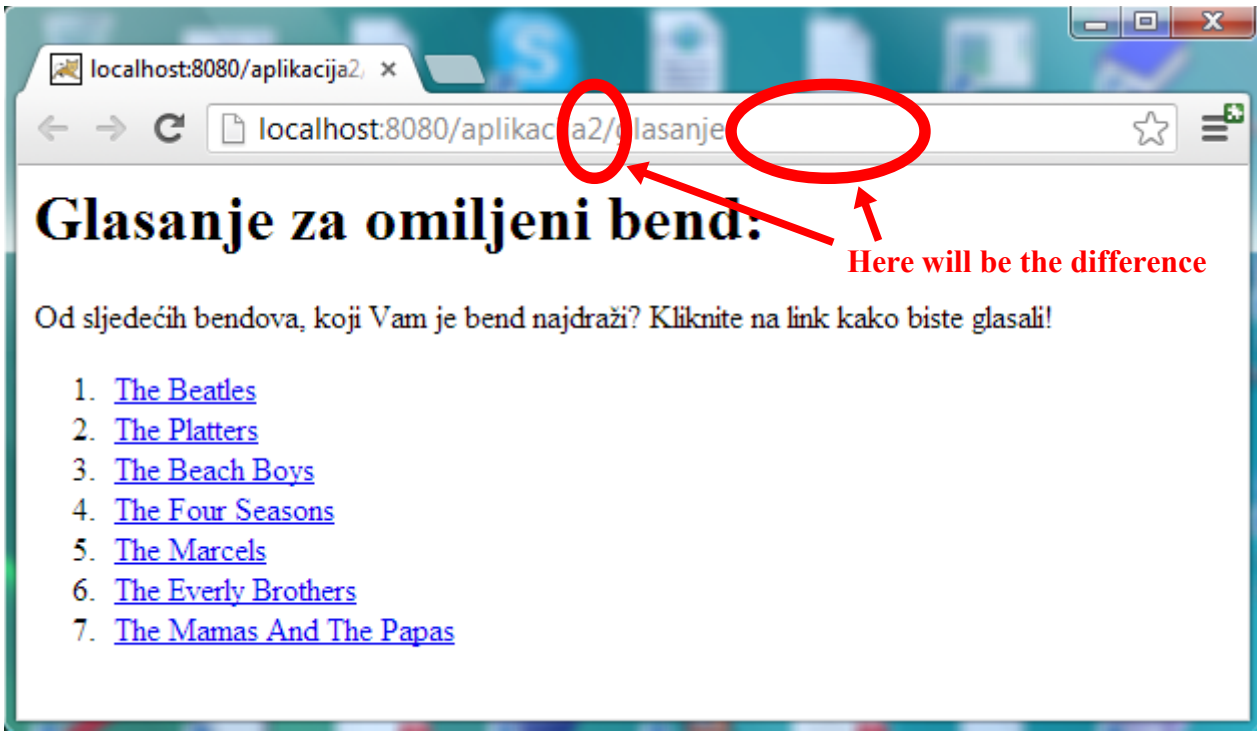
Problem 3.

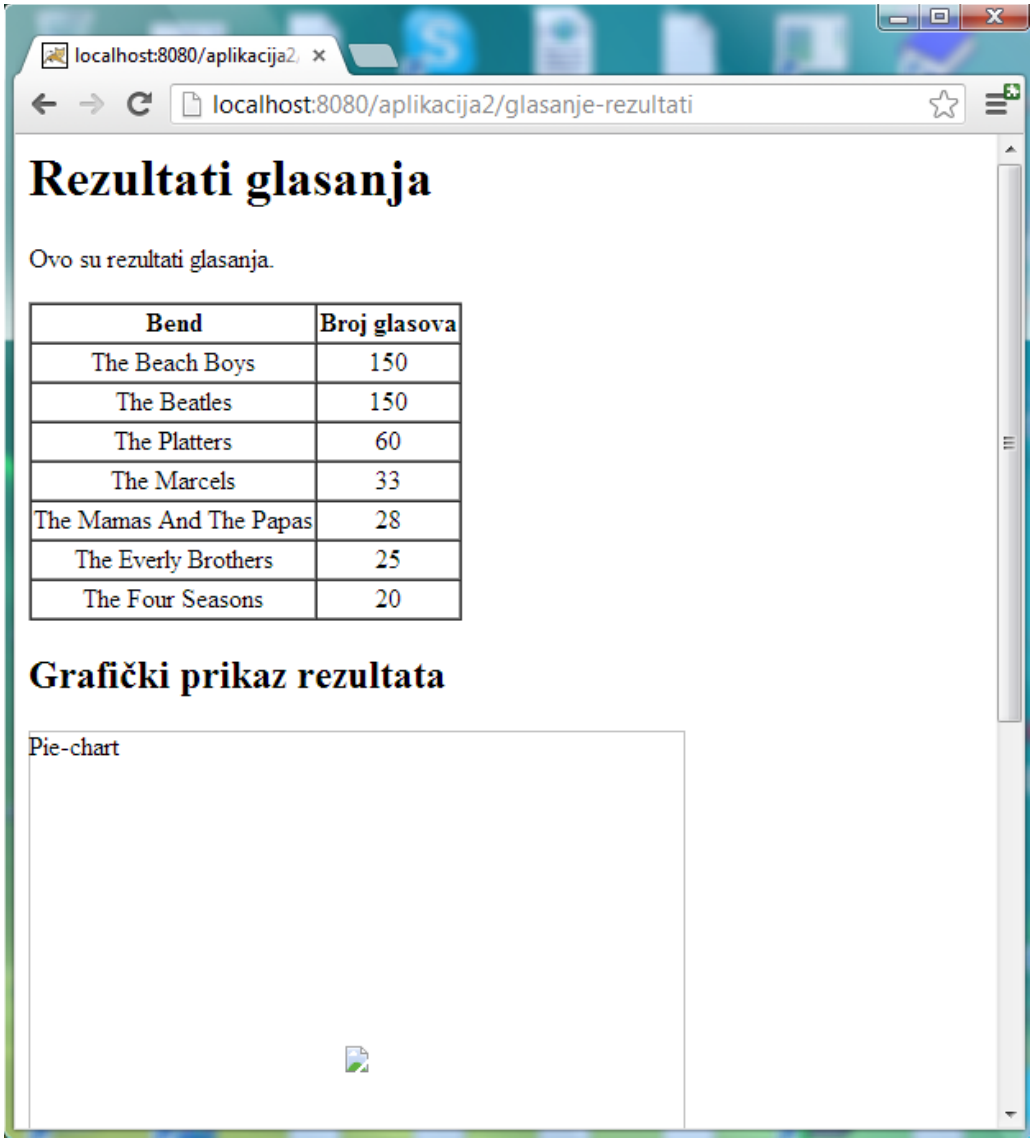
Integrate the servlets you developed in previous "voting"-homework (copy them and modify) so that they work with the poll and the poll options defined in database instead of in text files. This includes already mentioned servlet `/glasanje` which has to get the `pollID` identifier in each request so that it knows which poll to offer. Also, all of the servlets should be mapped into subpath `/servleti`; so, for example, the servlet `/glasanje` from previous homework should be now mapped to `/servleti/glasanje`.

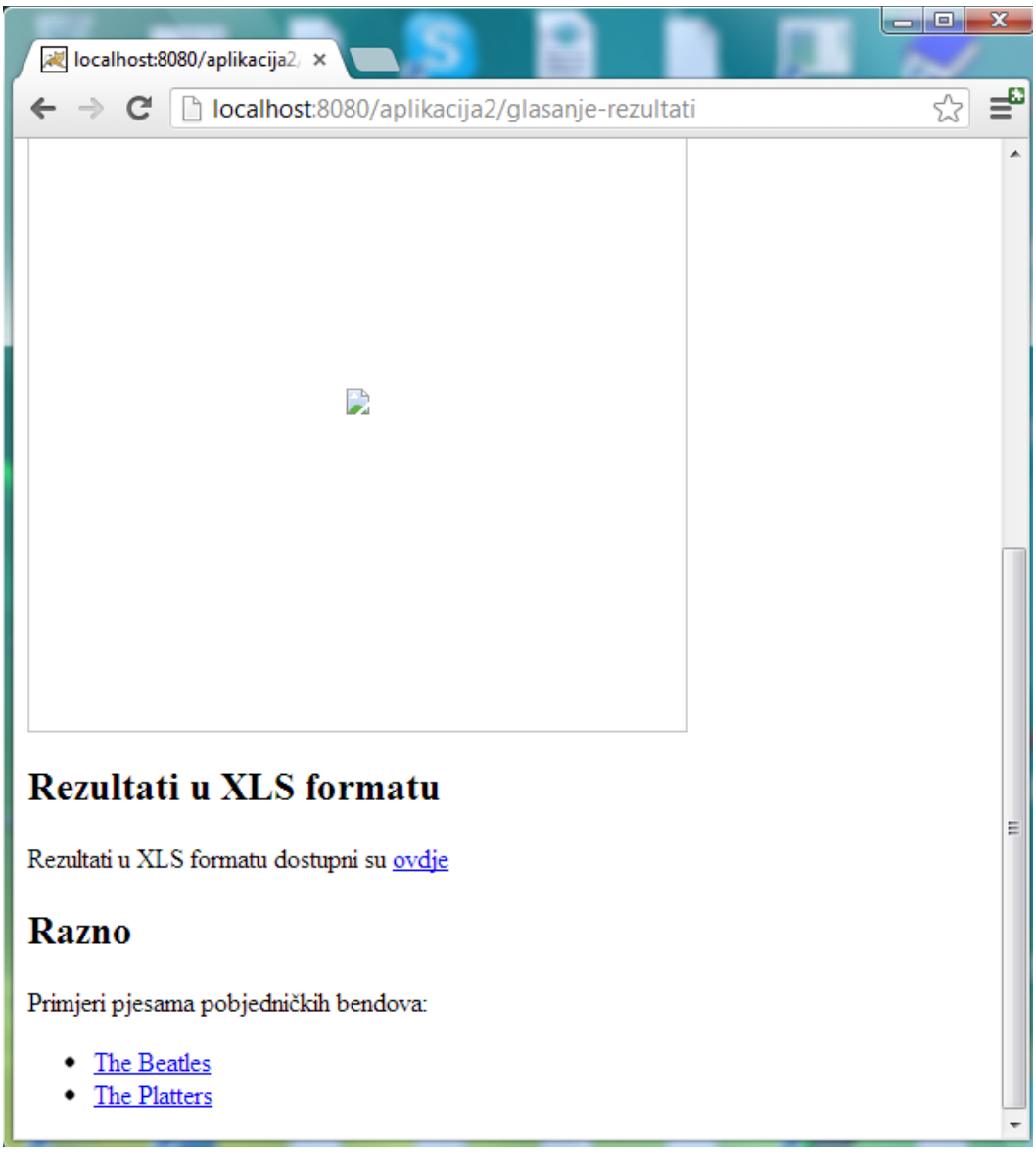
Once completed, the application screenshots should look the same as in old homework (screenshots are repeated here). The only difference will be in URL that will contain `/voting-app/servleti/...` and additional parameter that defines a concrete poll we work with. For recording the number of obtained votes use attribute `votesCount` of table **PollOptions**.

Here are the screenshots. **Please note:** the screenshot is from older version of this homework and shown URL is invalid; in this homework it should be something like:

`http://localhost:8080/voting-app/servleti/glasanje?pollID=2`


localhost:8080/aplikacija2/
localhost:8080/aplikacija2/glasanje
Here will be the difference
Glasanje za omiljeni bend:
Od sljedećih bendova, koji Vam je bend najdraži? Kliknite na link kako biste glasali!
1. The Beatles
2. The Platters
3. The Beach Boys
4. The Four Seasons
5. The Marcels
6. The Everly Brothers
7. The Mamas And The Papas


localhost:8080/aplikacija2/
localhost:8080/aplikacija2/glasanje-rezultati
Rezultati glasanja
Ovo su rezultati glasanja.
Bend | Broj glasova
The Beach Boys | 150
The Beatles | 150
The Platters | 60
The Marcels | 33
The Mamas And The Papas | 28
The Everly Brothers | 25
The Four Seasons | 20
Grafički prikaz rezultata
Pie-chart



The generation of XLS document and graphics must also work as expected.

Important: you must not at any time during the creation of XLS and Pie Chart save any files to disk; the servlet responsible for its creation must create everything in memory and write it directly to client's OutputStream. Also, you must not create and/or open and top level Swing containers (such as frames of dialogs) when creating Pie Chart!

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). You can use Java Collection Framework and other parts of Java covered by lectures; if unsure – e-mail me. Document your code!

If you need any help, I have reserved a slot for consultations as follows: Monday 2PM, Tuesday at 10AM, Wednesday at 9AM. Feel free to drop by my office (after e-mail).

All source files must be written using UTF-8 encoding. All classes, methods and fields (public, private or otherwise) must have appropriate javadoc.

You are not required to unit test code in this homework.

Important: acceptibility of implemented homework will be tested in Apache Tomcat. So, if you are using Jetty for development, before uploading your homework to Ferko, please deploy it on Apache Tomcat and CHECK that everything works as expected.

When your complete homework is done, pack it in zip archive with name `hw14-0000000000.zip` (replace zeros with your JMBAG); make sure there is **NO** `target` directory present. Upload this archive to Ferko before the deadline. Do not forget to lock your upload or upload will not be accepted. Deadline is June 14th 2018. at 07:00 AM.