13th homework assignment; JAVA, Academic year 2014/2015; FER

As usual, please see the last page. I mean it! You are back? OK. Here we have several problems for you to solve. Please start by new empty Eclipse project. You will solve all of problems in same project. For this project you will again use Apache Ant (and not Apache Maven). Your ant will have to support all tasks which we have previously used with ant.

Introduction

For this homework you will create a simple web application. So start by creating a new Eclipse project just the way we have done it during the last lecture class. Name this application *aplikacija2*. In each problem that follows I will describe a functionality that you are required to add into this application. Once done, you will prepare a single ZIP archive containing your Eclipse project. However, you will have to write appropriate ant WAR task so that user can run "ant war" and get aplikacija2.war archive which would be ready for deployment.

Problem 1.

Create a page `index.jsp`. Render on it a link "`Background color chooser`" that will lead to a new page `colors.jsp`. Render on that page four links with texts `WHITE`, `RED`, `GREEN` and `CYAN`. Clicking each of these links will trigger a servlet mapped to `/setcolor` and will remember the selected color as user's session attribute `pickedBgCol`. Hint: this "map" is available in servlet via `request.getSession()`.
In order to access session data in JSP page, you must declare `session="true"` in page directive (`<%@ page … session="true" … %>` , and then you will have access to implicitly defined variable `session` (which is the same variable which you obtained in servlet by calling `request.getSession()`).
You will use this information for rendering the background for each of web pages in this homework including the `index.jsp` and `colors.jsp`. If there is no information on selected color in users session data, use the white as default.

Hint: to specify a background color for a HTML page you can either use appropriate attribute of BODY tag or you can link each page to use a *css* file which you then dynamically create (either by JSP or servlet and in which you dynamically write the color to be used).

Problem 2.

Allow the user to obtain a table of values of trigonometric functions sin(x) and cos(x) for all *integer* angles (in degrees, not radians) in a range determined by URL parameters *a* and *b* (if *a* is missing, assume *a*=0; if *b* is missing, assume *b*=360; if *a* > *b*, swap them; if *b* > *a*+720, set *b* to *a*+720). This action must be accessible on local URL `/trigonometric` and must be mapped to a servlet that will do all the calculations. Once the data are calculated, forward rendering of a resulting page to a JSP `/WEB-INF/pages/trigonometric.jsp`, by using `request.getRequestDispather(...).forward(...)` call.
Add a link to this action in `index.jsp` with parameters *a*=0 and *b*=90.

Problem 3.

Create a page `stories/funny.jsp` that contains some not too long but funny story. The color of the font used for stories text must be each time randomly chosen (you can randomly pick some color from predefined selection of colors). Add a link to this in `index.jsp`.

Problem 4.

Create a page `report.jsp` that contains a heading "*OS usage*", a paragraph "*Here are the results of OS usage in survey that we completed.*" and with a dynamically created image showing Pie Chart. For the creation of the image you will utilize a servlet mapped to `/reportImage`. The servlet will create the requested image by using `jfreechart` library which is freely available. The pie chart should look like the one in simple tutorial available at address:

http://www.vogella.com/articles/JFreeChart/article.html

This tutorial shows how to display the chart using Swing. Your job is to adjust the shown code and to modify it as needed. The libraries that are needed you will have to download and place in `WEB-INF/lib` directory in order to make them available to servlet container, and you will have to include them in your classpath as well in Eclipse ("Add to build path"). Since your sevlet will now generate an image, do not forget to set appropriate response's content type, depending on the image format you will use – I recommend *png*.

Problem 5.

Create an action mapped to `/powers` that accepts a three parameters *a* (integer from [-100,100]) *b* (integer from [-100,100]) and *n* (where *n*>=1 and *n*<=5). If any parameter is invalid, you should display appropriate message (forward request to some prepared JSP).

Your action should dynamically create a Microsoft Excel document with *n* pages. On page *i* there must be a table with two columns. The first column should contain integer numbers from *a* to *b*. The second column should contain *i*-th powers of these numbers.

For a creation of XLS document you should use a library developed as part of Apache POI project which is also freely available for download. There are many tutorials available on Internet. For a quick and simple please check:

http://www.roseindia.net/answers/viewqa/Java-Beginners/14946-how-to-create-an-excel-file-using-java.html

Add a link to this action on `index.jsp` with a parameters *a*=1, *b*=100, *n*=3.

Problem 6.

Create `/appinfo.jsp` that displays how long is this web application running. In order to do so, create a servlet context listener that adds information when it was called into servlet context's attributes (for example, store the result of `System.currentTimeMillis()`). In your JSP, check what is the current time, look up the stored time by your listener and calculate the difference. Format the duration nicely (for example, instead of writing a huge number of milliseconds, output something like 2 days 11 hours 25 minutes 17 seconds and 342 milliseconds).

Add a link to this action on `index.jsp`.

For help see:
http://docs.oracle.com/javaee/5/tutorial/doc/bnafi.html#bnafj
http://docs.oracle.com/javaee/6/api/javax/servlet/ServletContextListener.html

Problem 7.

You will prepare a simple voting application. In `/WEB-INF` directory you will put a voting definition file named `glasanje-definicija.txt` that will, in each line, contain a unique ID (integer), musical band name, and a link to one representative song of that band. In each line, elements are separated by TAB. Example of such file is given below (some lines are broken due to limited page width).

```
1	The Beatles	http://www.geocities.com/~goldenoldies/TwistAndShout-Beatles.mid
2	The Platters	http://www.geocities.com/~goldenoldies/SmokeGetsInYourEyes-Platters-
ver2.mid
3	The Beach Boys	http://www.geocities.com/~goldenoldies/SurfinUSA-BeachBoys.mid
4	The Four Seasons	http://www.geocities.com/~goldenoldies/BigGirlsDontCry-
FourSeasons.mid
5	The Marcels	http://www.geocities.com/~goldenoldies/Bluemoon-Marcels.mid
6	The Everly Brothers	http://www.geocities.com/~goldenoldies/All.I.HaveToDoIsDream-
EverlyBrothers.mid
7	The Mamas And The Papas	http://www.geocities.com/~goldenoldies/CaliforniaDreaming-
Mamas-Papas.mid
```

You will prepare several servlets and JSP-s that will enable user to view a list of bands, click a link to vote for a band, and then get the report with voting results.

You will keep voting results in a file that must also be placed in `/WEB-INF` directory: name it `glasanje-rezultati.txt`. Its is a simple textual file having in each line band ID and a total number of votes that band received so far. Here is an example (separator is also TAB).

```
1	150
2	60
3	150
4	20
5	33
6	25
7	20
```

For accessing these files from your servlets, you will need to know where on disk these files actually reside, in order to open them for reading/writing. For this purpose, servlet containers offer appropriate method `getRealPath` that is available in `ServletContext`. From a servlet's `doGet` method, you can access it like this:

```
String fileName = req.getServletContext().getRealPath(
   "/WEB-INF/glasanje-definicija.txt"
);
// Now open the file with the obtained fileName...
```

Observe that the argument is an absolute path relative to your current web-application. The result will be an absolute path on disk. For example, on my computer, the `fileName` would become:

```
d:\usr\apache-tomcat-7.0.54\webapps\aplikacija2\WEB-INF\glasanje-definicija.txt
```

So now that we understand the basics, here are the servlets and JSP-s that you must create.

GlasanjeServlet. Map it to `/glasanje`. Its `doGet` method should be like:

```
// Učitaj raspoložive bendove
String fileName = req.getServletContext().getRealPath("/WEB-INF/glasanje-definicija.txt");
// ...
```

```
// Pošalji ih JSP-u...
req.getRequestDispatcher("/WEB-INF/pages/glasanjeIndex.jsp").forward(req, resp);
```

GlasanjeGlasajServlet. Map it to `/glasanje-glasaj`. Its `doGet` method should be like:

```
// Zabiljezi glas...
String fileName = req.getServletContext().getRealPath("/WEB-INF/glasanje-rezultati.txt");
// Napravi datoteku ako je potrebno; ažuriraj podatke koji su u njoj...
// ...

// ...
// Kad je gotovo, pošalji redirect pregledniku
resp.sendRedirect(req.getContextPath() + "/glasanje-rezultati");
```

GlasanjeRezultatiServlet. Map it to `/glasanje-rezultati`. Its `doGet` method should be like:

```
// Pročitaj rezultate iz /WEB-INF/glasanje-rezultati.txt
String fileName = req.getServletContext().getRealPath("/WEB-INF/glasanje-rezultati.txt");
// Napravi datoteku ako je potrebno; inače je samo pročitaj...
// ...

// Pošalji ih JSP-u...
req.getRequestDispatcher("/WEB-INF/pages/glasanjeRez.jsp").forward(req, resp);
```

You will also need two JSP pages that these servlets use.

`/WEB-INF/pages/glasanjeIndex.jsp` must render the list of bands that servlet `/glasanje` prepared and stored in request's attributes. For a bands given in this document's example file, this JSP should generate a HTML that looks like the following (you do not have to mimic it literally).

```
<html>
   <body>
       <h1>Glasanje za omiljeni bend:</h1>
     <p>Od sljedećih bendova, koji Vam je bend najdraži? Kliknite na link kako biste glasali!
</p>
     <ol>
       <li><a href="glasanje-glasaj?id=1">The Beatles</a></li>
       <li><a href="glasanje-glasaj?id=2">The Platters</a></li>
       <li><a href="glasanje-glasaj?id=3">The Beach Boys</a></li>
       <li><a href="glasanje-glasaj?id=4">The Four Seasons</a></li>
       <li><a href="glasanje-glasaj?id=5">The Marcels</a></li>
       <li><a href="glasanje-glasaj?id=6">The Everly Brothers</a></li>
       <li><a href="glasanje-glasaj?id=7">The Mamas And The Papas</a></li>
     </ol>
   </body>
</html>
```

Please observe that the bands are sorted according to IDs from definition file.

`/WEB-INF/pages/glasanjeRez.jsp` must render the voting results that servlet `/glasanje-rezultati` prepared and stored in request's attributes. For a bands given in this document's example file, this JSP should generate a HTML that looks like the following (you do not have to mimic it literally but all conceptual parts must be present).

```
<html>
   <head>
     <style type="text/css">
       table.rez td {text-align: center;}
     </style>
   </head>
   <body>

       <h1>Rezultati glasanja</h1>
     <p>Ovo su rezultati glasanja.</p>
     <table border="1" cellspacing="0" class="rez">
       <thead><tr><th>Bend</th><th>Broj glasova</th></tr></thead>
       <tbody>
       <tr><td>The Beach Boys</td><td>150</td></tr>
       <tr><td>The Beatles</td><td>150</td></tr>
       <tr><td>The Platters</td><td>60</td></tr>
       <tr><td>The Marcels</td><td>33</td></tr>
       <tr><td>The Mamas And The Papas</td><td>28</td></tr>
       <tr><td>The Everly Brothers</td><td>25</td></tr>
       <tr><td>The Four Seasons</td><td>20</td></tr>
       </tbody>
     </table>

     <h2>Grafički prikaz rezultata</h2>
     <img alt="Pie-chart" src="/glasanje-grafika" width="400" height="400" />

     <h2>Rezultati u XLS formatu</h2>
     <p>Rezultati u XLS formatu dostupni su <a href="/glasanje-xls">ovdje</a></p>

     <h2>Razno</h2>
     <p>Primjeri pjesama pobjedničkih bendova:</p>
     <ul>
       <li><a href="http://www.geocities.com/~goldenoldies/TwistAndShout-Beatles.mid"
target="_blank">The Beatles</a></li>
       <li><a href="http://www.geocities.com/~goldenoldies/SmokeGetsInYourEyes-Platters-
ver2.mid" target="_blank">The Platters</a></li>
     </ul>
   </body>
</html>
```

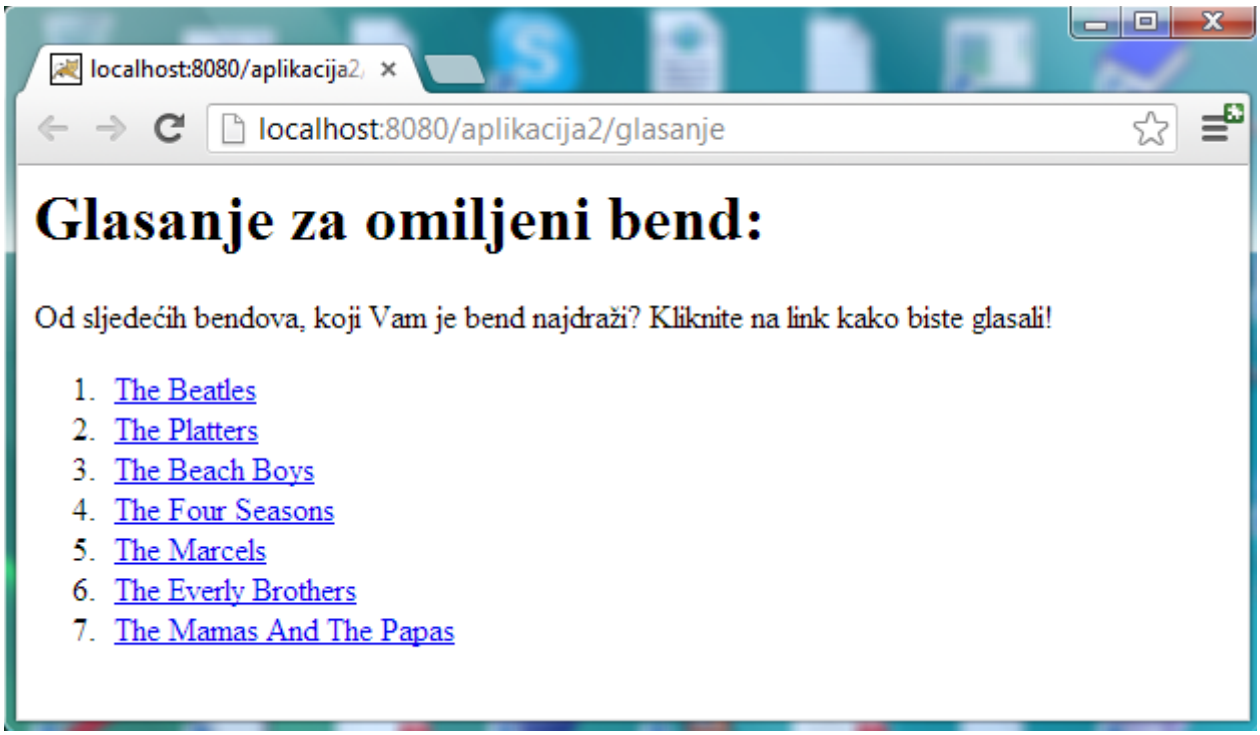
The first section of the page must render voting results as a HTML table in which bands are sorted according to the number of votes.

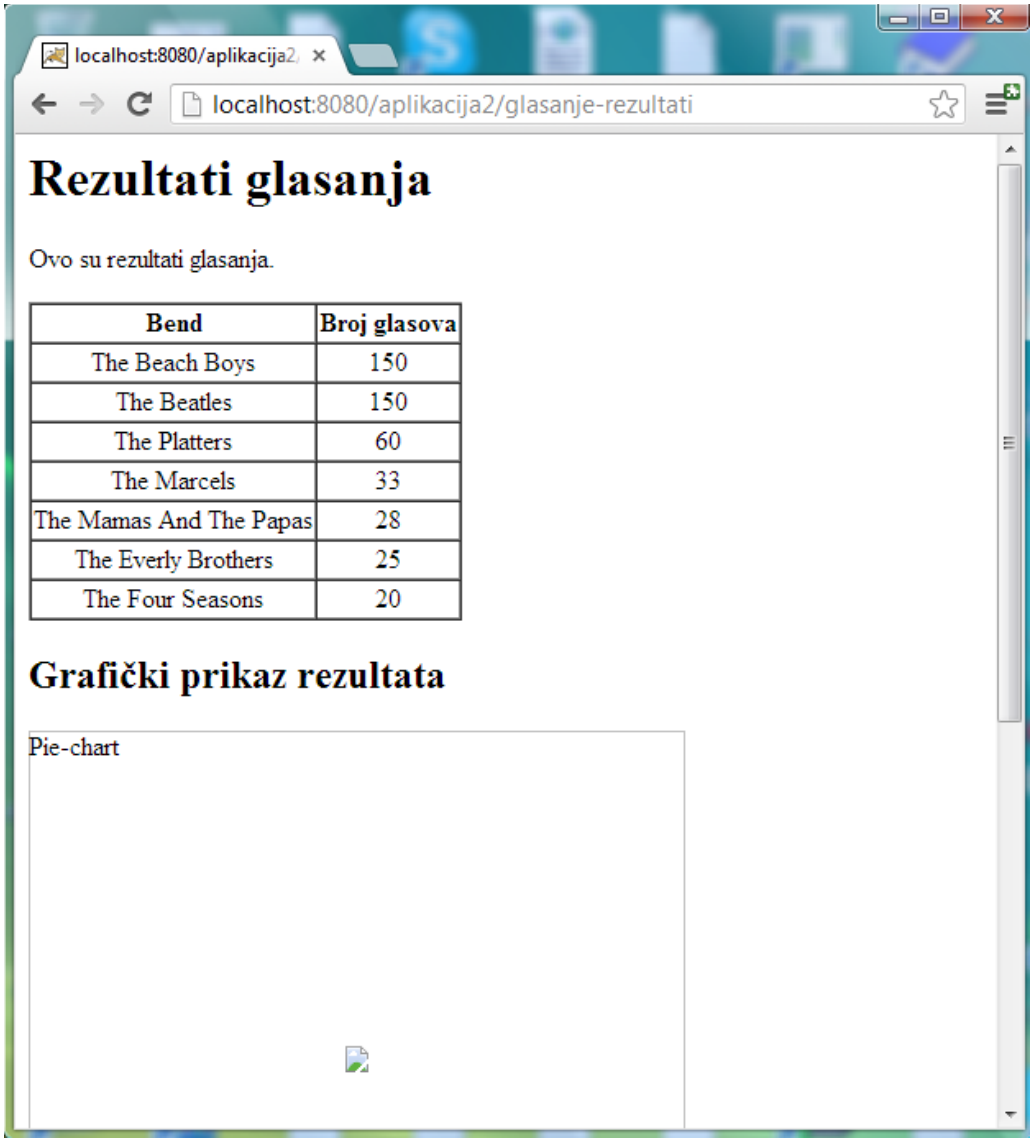
Second section contains a graphical representation of results. For this, you must create another servlet (map it to `/glasanje-grafika`) that will open the file with results and using jFreeChart produce a PNG of PieChart.

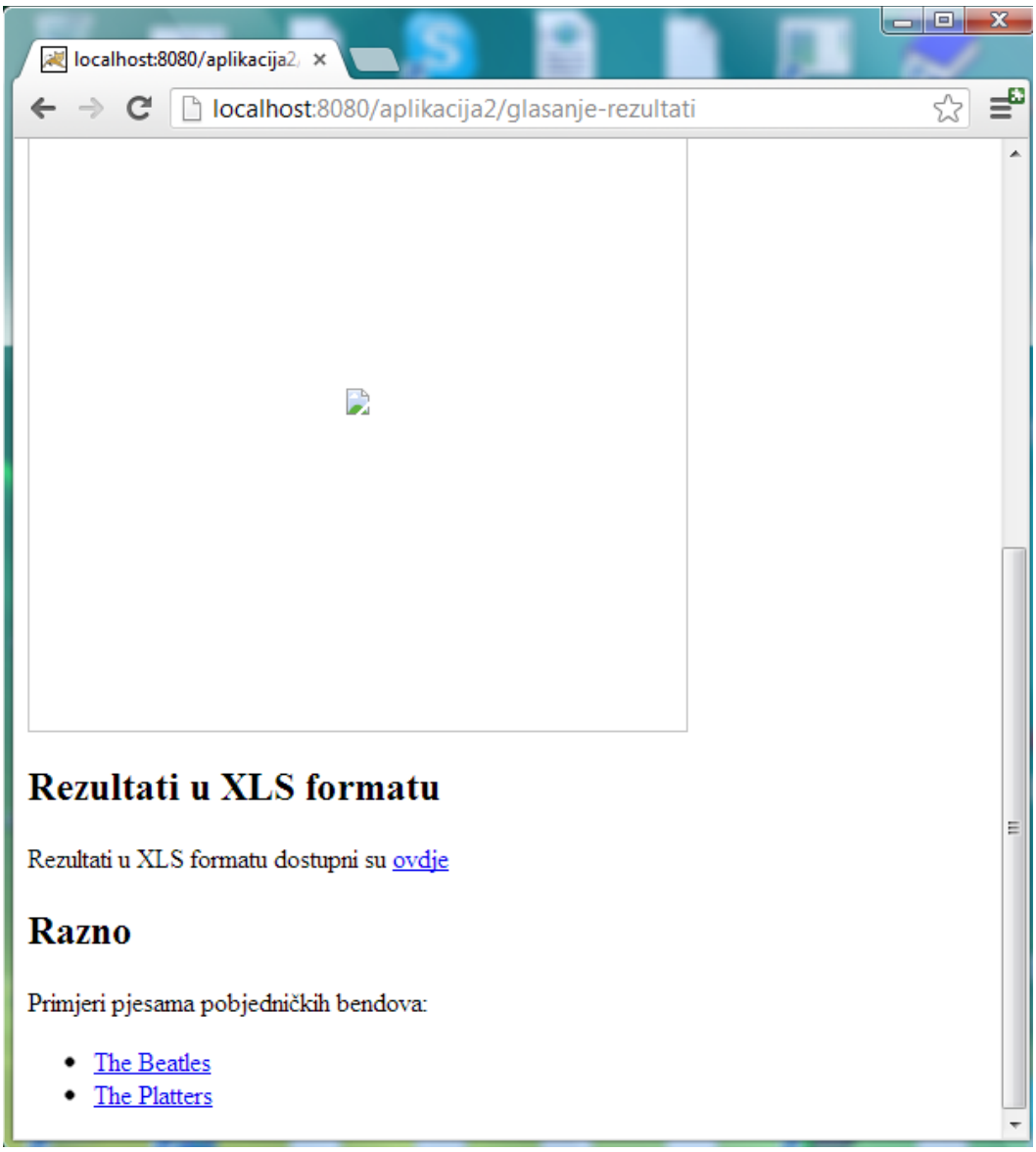
Third section contains a link for obtaining voting results in XLS format. For this, you must create another servlet (map it to `/glasanje-xls`) that will create the requested document.

Finally, the last section contains a links to songs on Internet. The links are rendered only for current winner (or winners, in case of tie as in previous example).

Here are screenshots.

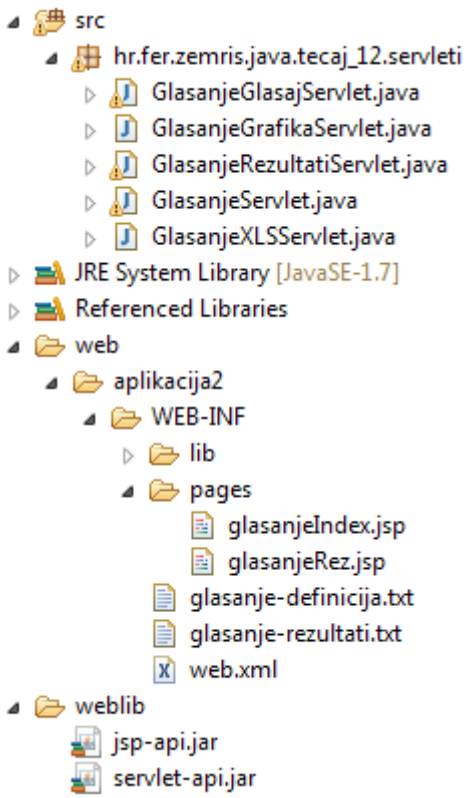

localhost:8080/aplikacija2/
localhost:8080/aplikacija2/glasanje
Glasanje za omiljeni bend:
Od sljedećih bendova, koji Vam je bend najdraži? Kliknite na link kako biste glasali!
1. The Beatles
2. The Platters
3. The Beach Boys
4. The Four Seasons
5. The Marcels
6. The Everly Brothers
7. The Mamas And 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 PieChart in above example is not shown because I wanted to leave you a freedom to style it and implement as you see fit. The style and the exact content of the generated XLS document is also for you to decide (will you present bands in rows or columns, how will you present votes, etc).

Since this is the most complex problem in this homework, I have also prepared a screenshot of Eclipse project (with everything not considering this problem removed).



In your `/WEB-INF/lib` you will have to add all the JAR-s needed for jFreeChart and Apache POI, as well as JSTL JAR-s if you use `<c:xxx />` tags.

Problem 8.

Equip your `build.xml` with a `war` target. In the book, I have described how to implement a build cycle init → compile (chapter 2, section "Automatizacija razvojnog ciklusa", "Uporaba alata ant"). Now add additional target `war` that will create "`dist/aplikacija2.war`". Ant already contains appropriate task. Here is a helpful link:

https://ant.apache.org/manual/Tasks/war.html

This task must depend on `compile`.

Once you produce correct WAR, just drop it into tomcat's `webapps` directory and try using your application. While you experiment with WAR creation, keep in mind that it is a (renamed) ZIP archive, so you can always take a look at its content using any ZIP archive browser to check if everything is as it should be.

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`. Read carefully the last problem – you must add `war` target that will automatically create complete WAR file.

You are required to create at least one unit test (for whatever you wish).

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 `HW13-`*yourJMBAG*.

The deadline for uploading and locking this homework is June, 13th 2015. at 07:00 AM.