13th homework assignment; JAVA, Academic year 2018/2019; FER

Napravite prazan Maven projekt, po uzoru na projekt koji smo koristili na ovom predavanju (pazite na `packaging` te na ovisnosti): u Eclipsovom workspace direktoriju napravite direktorij `hw13-0000000000` (zamijenite nule Vašim JMBAG-om) te u njemu oformite Mavenov projekt `hr.fer.zemris.java.jmbag0000000000:hw13-0000000000` (zamijenite nule Vašim JMBAG-om), postavite `packaging` na `war`, dodajte potrebne ovisnosti i na kraju provjerite da ste u `<build>` sekciji postavili `finalName` na `webapp2`. Stvorite i potrebnu strukturu direktorija koja uključuje i `src/main/webapp`, te u taj direktorij u `WEB-INF` obavezno postavite `web.xml` verzije `3.1`. Tek kad ste to sve učinili, importajte projekt u Eclipse (u suprotnom moglo bi se dogoditi da Eclipse počne prijavljivati svakakve pogreške). Sada možete nastaviti s rješavanjem zadataka.

Introduction

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 Maven project. **You must check that the ZIP archive DOES NOT contain target folder; if it does, open it in any ZIP archiver and delete the target directory from it.**
Then you can upload it.

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.

Add on `index.jsp` also the following form:

```
<form action="trigonometric" method="GET">
  Početni kut:<br><input type="number" name="a" min="0" max="360" step="1" value="0"><br>
  Završni kut:<br><input type="number" name="b" min="0" max="360" step="1" value="360"><br>
  <input type="submit" value="Tabeliraj"><input type="reset" value="Reset">
</form>
```

The `form` tag tells the browser to compose a `GET` (see `method` attribute) request to server with URL determined from resolving `action` attribute and URL of the page containing this form (so if the form was in http://localhost:8080/webapp2/index.jsp, and `action="trigonometric"`, the request will be sent to: `resolve(/webapp2/index.jsp, trigonometric)=/webapp2/trigonometric`. It will have two parameters (`a` and `b`, see two input tags with `name="a"` and `name="b"`) whose values will be set to whatever user left in displayed edit-boxes.

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`. Important: this JSP must be available directly at defined location; do not put it in `WEB-INF/pages/stories/funny.jsp` or some other location!

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. Google: "`maven jfreechart`", follow the first link: http://mvnrepository.com/artifact/jfree/jfreechart, it will tell you that the artefact was moved – click on new artifact; it will send you to page http://mvnrepository.com/artifact/org.jfree/jfreechart; there select the last version (currently: 1.0.19 (Jul, 2014)). You will reach the page for this version with dependency-snippet which you have to copy into your `pom.xml` in order to use this library.
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 components. Your job is to adjust the shown code and to modify it as needed. Do not create frames or swing components; using *jfreechart* find out how to define a graph of specified type, and how to render it directly as image (hint: google "jfreechart servlet pie chart" or "jfreechart servlet bar chart")! The exact numbers based on which the chart will be created you can set to anything you like.

Since your servlet 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. Google "`apache poi maven`"; you will eventually reach page http://mvnrepository.com/artifact/org.apache.poi/poi, select current version 3.16 and copy dependency-snippet into your `pom.xml`.

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.

Optional (but nice feature): If you want to specify a filename that browser will automatically set in Save As… dialog when asking to save XLS generated by servlet, after setting the `Content-Type` in your servlet, add another header: `Content-Disposition`. For example, if you want the file to be named "`tablica.xls`", you will use:

```
resp.setHeader("Content-Disposition", "attachment; filename=\"tablica.xls\"");
```

Problem 6.

Create `/appinfo.jsp` that displays how long is this web application running. In order to do so, create a **servlet context listener** (I talked about listeners and there is example in text-file we followed) 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. You can prepare a different set of bands, but make sure there are at least 7 of them, with links to representative songs. ID's do not have to start with 1, and they do not have to be consecutive. In example below, instead of 1, 2, 3, 4, 5, 6, 7 in the first column, there could have been 34, 55, 17, 251, 999, 185, 352.

```
1	The Beatles https://www.youtube.com/watch?v=z9ypq6_5bsg
2	The Platters	https://www.youtube.com/watch?v=H2di83WAOhU
3	The Beach Boys	https://www.youtube.com/watch?v=2s4slliAtQU
4	The Four Seasons	https://www.youtube.com/watch?v=y8yvnqHmFds
5	The Marcels https://www.youtube.com/watch?v=qoi3TH59ZEs
6	The Everly Brothers	https://www.youtube.com/watch?v=tbU3zdAgiX8
7	The Mamas And The Papas https://www.youtube.com/watch?v=N-aK6JnyFmk
```

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). If there were no votes for some band, this file could have had either a row with a value 0 for votes, or no record at all (both solutions are on; the latter is even easier to implement).

```
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. Please be aware – your web application can be deployed by any user where ever he chooses, in any servlet container that user has installed. So you can not hard-code the absolute path on disk for this files. Instead, 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 old Windows computer, the `fileName` could become:

```
d:\usr\apache-tomcat-7.0.54\webapps\webapp2\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 I dalje NE generiraj odgovor
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="https://www.youtube.com/watch?v=z9ypq6_5bsg" target="_blank">The
Beatles</a></li>
       <li><a href="https://www.youtube.com/watch?v=H2di83WAOhU" 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.

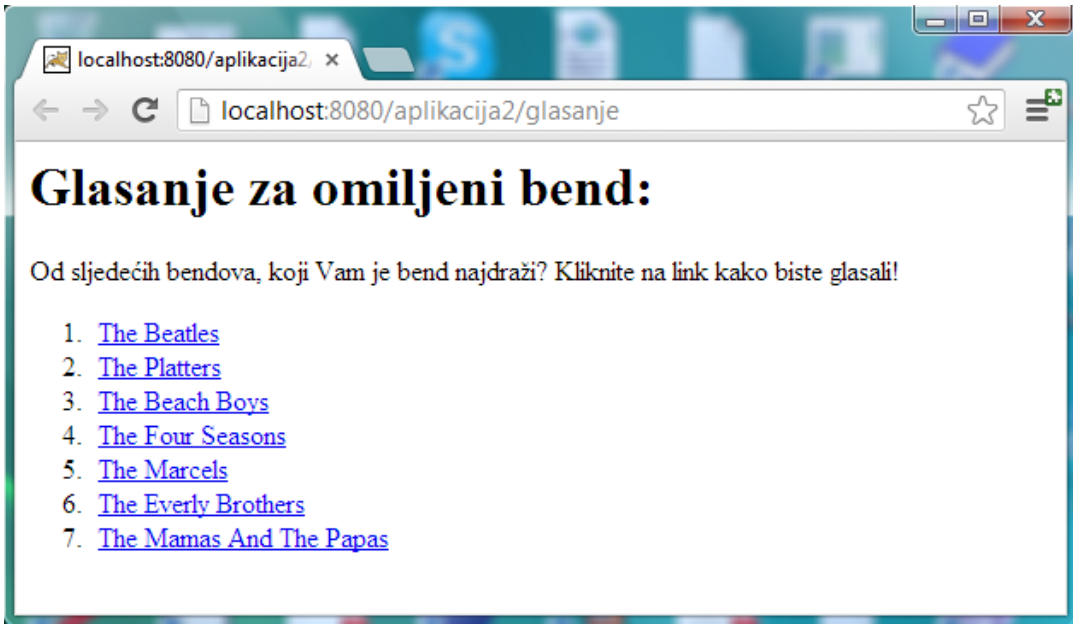
Third section contains a link for obtaining voting results in XLS format. For this, you must create another servlet (map it to `/glasanje-xls`) that will create the requested document.

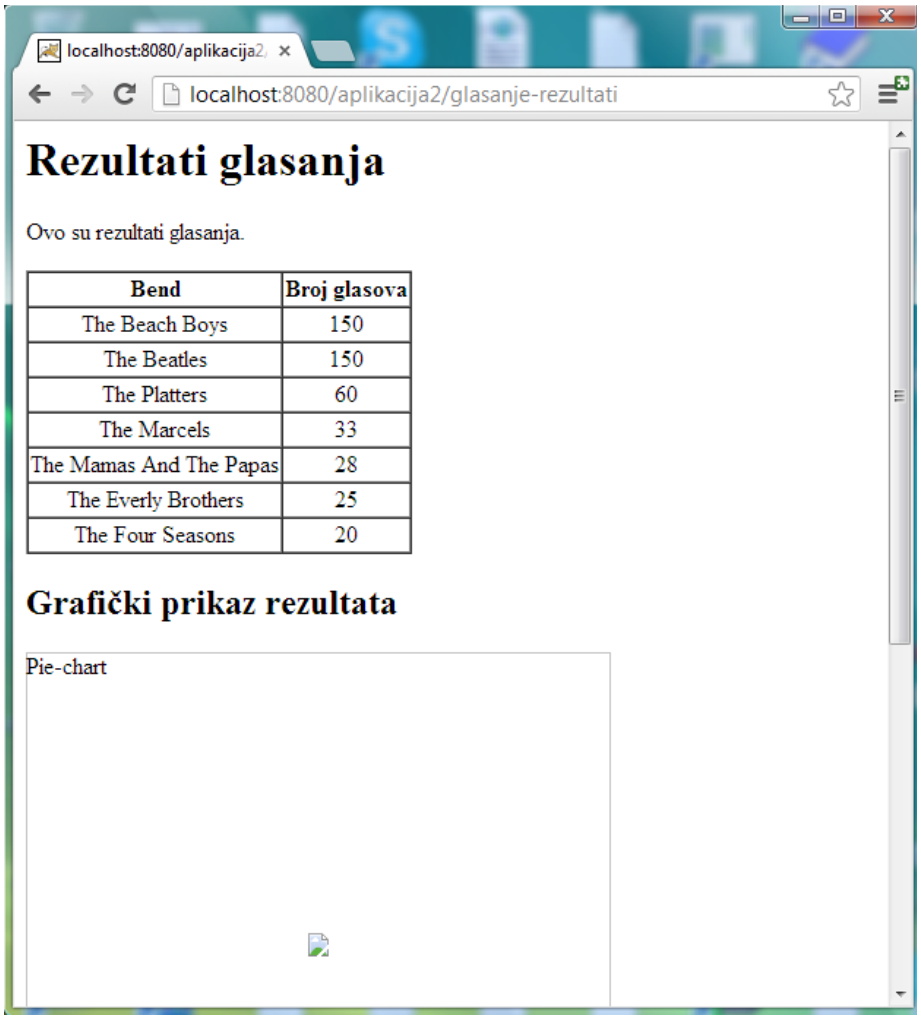
Finally, the last section contains a links to songs on Internet. The links are rendered only for current winner (or winners, in case of tie as in previous example).

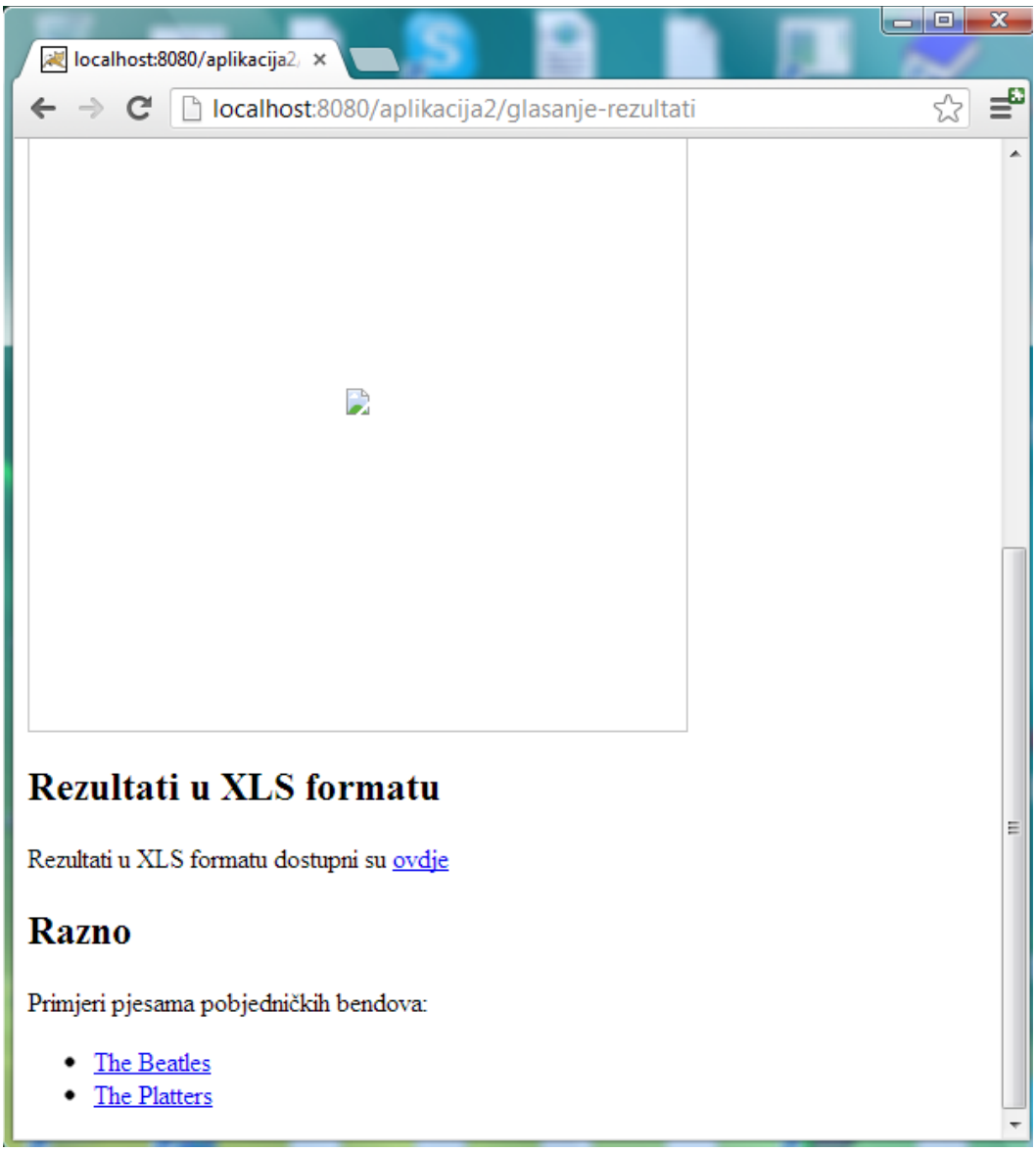
Here are screenshots. Shown URLs in address bar **are now obsolete** – the shown application was deployed

as `aplikacija2` but for this homework it should be deployed as `webapp2`. That said, the first URL in your homework will be:

`http://localhost:8080/webapp2/glasanje`







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).

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, consultations are Monday, Tuesday and Wednesday at Noon.

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 `hw13-0000000000.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 6th 2019. at 07:00 AM.

Pojednostavljenje razvoja web-aplikacije

Umjesto da stalno kopirate web-aplikaciju u Tomcata, uporabom Mavena moguće je pokrenuti Jetty web-poslužitelja koji će motriti što radite izravno u projektu, i automatski posluživati trenutno stanje projekta. Da biste to omogućili, napravite sljedeće. Otiđite u projekt koji smo napravili na predavanju pa tamo ovo najprije isprobajte. Otvorite `pom.xml` i dodajte definiciju potrebnog maven plugina u `<build>` sekciju. Ista bi morala izgledati ovako:

```
<build>
  <plugins>
    <plugin>
      <groupId>org.eclipse.jetty</groupId>
      <artifactId>jetty-maven-plugin</artifactId>
      <version>9.4.10.v20180503</version>
      <configuration>
        <scanIntervalSeconds>10</scanIntervalSeconds>
        <webApp>
          <contextPath>/webapp1</contextPath>
        </webApp>
      </configuration>
    </plugin>
  </plugins>
  <finalName>webapp1</finalName>
</build>
```

Sada možete u terminalu (u direktoriju projekta) zadati:

```
mvn compile
mvn jetty:run
```

Ova posljednja naredba pokrenut će jetty-poslužitelj (i zamrznuti terminal; poslužitelj možete prekinuti s CTRL+C). Poslužitelj će web-aplikaciju posluživati na `localhost:8080/webapp1` (ovo posljednje određeno je atributom `contextPath` koji ste upravo iskopirali u `pom.xml`; pronađite to!). Također, kako jetty po defaultu želi slušati na portu `8080`, posljedica je da Tomcat ne može istovremeno biti pokrenut jer i on želi taj isti port za sebe. Pokrenuti jetty periodički će (svaki 10 sekundi – pronađite i to u `pom.xml`) skenirati projekt da vidi je li bilo noviteta.

Sada možete slobodno raditi izravno u Eclipse-u, modificirati sve što želite, i izravno pozivati adrese iz web-preglednika; sve bi trebalo raditi automatski. Ako nešto zašteka, ugasite jetty pa ga samo ponovno pokrenite.

Za detalje konfiguracije vidjeti:
http://www.eclipse.org/jetty/documentation/9.4.x/jetty-maven-plugin.html

Debugiranje web-aplikacije

Također isprobajte najprije u projektu koji smo napravili na predavanju. Stavite u neki servlet breakpoint. Zaustavite jetty. U konzoli zadajte naredbu:

```
export MAVEN_OPTS="-Xdebug -Xnoagent -Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n"
```

ako ste na Linuxu, odnosno

```
SET "MAVEN_OPTS=-Xdebug -Xnoagent -Djava.compiler=NONE
-Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=n"
```

ako ste na windowsima (u normalnom Command Promptu; isprobano u PowerShellu – ne radi, ako ga koristite, treba izgoogleati kako se u njemu podešavaju varijable okruženja). **Ovo sve je jedna naredba** gdje iza `compiler=NONE` dolazi jedan razmak pa `-Xrun...`, smo što nije sve stalo u jedan redak ovog dokumenta.

Ponovno pokrenite `jetty:run` i primjetite kako će Vam u prvom retku ispisa sada ispisati da sluša na portu `8000` i tamo prihvaća veze od debuggera koji žele debugirati aplikacije koje se izvode na virtualnom stroju (koji trenutno izvodi jetty-poslužitelja i Vašu web-aplikaciju).

Sada još morate Eclipseu reći da se za potrebe debugiranja spoji na ovaj virtualni stroj. Postupak je sljedeći.

Izbornik "`Run`" --> "`Debug configuration`".

Otvorit će se dijalog koji s lijeve strane navodi sve što se pokretali u Eclipseu. Ako pogledate malo bolje, vidjet ćete da je to sve stavljeno pod kategoriju "`Java Application`", i da ima još kategorija. Označite kategoriju "`Remote Java Application`" (kliknite na nju), pa u toolbaru iznad klik na gumb "`New`" (bijeli dokument sa znakom "`+`"). Desno će se otvoriti kartica s podatcima o Vašem projektu. Odmah na prvoj kartici vidjet ćete i "`Connection Properties`". Tamo prikazani `port` mora biti jednak (ako nije, promijenite ga) portu koji ste postavili u `MAVEN_OPTS` (postoji dio `address=xxxx`).

Kad je sve OK, pritisnite gumb `Debug`. Sada možete otići u Debug-perspektivu i vidjet ćete da je Eclipse zakvačen na Jetty-poslužitelja. Ako u nekom servletu sada stavite breakpoint i u pregledniku zatražite taj

servlet, Eclipse će zaustaviti izvođenje na tom breakpointu, i moći ćete korak po korak debugirati što se događa. Pritiskom na gumb "`Disconnect`" Eclipse se otpaja od Jetty-ja, i više neće zaustavljati izvođenje na označenim breakpointima.

Otiđite u preglednika, zatražite primjerice http://localhost:8080/webapp1/prvi (stavite prije toga u njega breakpoint) i uvjerite se da debugiranje radi. Jednom kada ste ovo savladali, napravite isto u Vašoj domaćoj zadaći!