# **Environment Description**

Operating System: Microsoft Windows 10

Integrated Development Environment: NetBeans IDE version 8.2

Application Server: GlassFish Server version 4.1.1

Web Browser: Chrome

# **Instructions how to set up and run your project.**

1. Obtain NetBeans and configure it

* Go to <https://netbeans.org/downloads/> and download the Java EE version of NetBeans
* The downloaded file is called: netbeans-8.2-javaee-windows.exe. Run this file.
* During the installation, make sure to install GlassFish Server 4.1.1 and Apache Tomcat 8.0.27 as well

1. Load the javatutorials from the GlassFish distribution

* Go to <https://javaee.github.io/glassfish/download> and download glassfish: “[GlassFish 4.1.2 - Full Platform](http://download.java.net/glassfish/4.1.2/release/glassfish-4.1.2.zip)”
* Unarchive the downloaded file and copy it to a folder of your liking, let’s call it: <**downloaded GlassFish**>
* Navigate to <**downloaded GlassFish**>\glassfish4\docs\javaee-tutorial\examples\ and copy the contents of this folder to a work folder of your linking, let’s call it <**work folder**>.

1. Load the modified dukeetf code into NetBeans

* Navigate to <**work folder**>\web\servlet\ and unarchive here the provided **dukeetf.zip**
* Override all contents
* Start-up your NetBeans
* Go to File->New project->Maven->double-click Project with existing POM->Finish
* Navigate to <**work folder**> and click Load Project
* In the list of projects in NetBeans now we have a javaeetutorial group. Expand it until you find the dukeetf module:

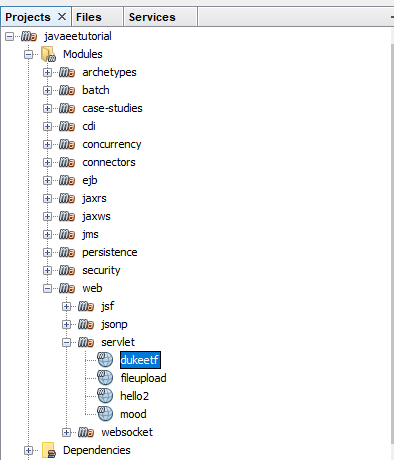


Figure dukeetf project in the javaeetutorials

* Double click the dukeetf. This loads the modified dukeetf project
* Right-click the loaded project and click Run
* In the pop-up, select the GlassFish server:

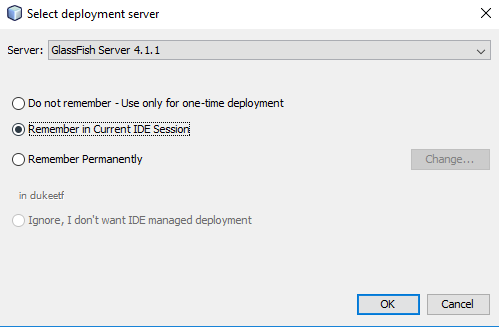


Figure The server selection pop-up

# **Description of all the changes made to the dukeetf project.**

1. Added 1 resource file to the project

For this I navigated to the <**work folder**>\web\servlet\dukeetf\src\main\ and added there a new folder called “resources”. Inside this folder I added the project4input.txt file. Going back to NetBeans, I refresh to see this structure:

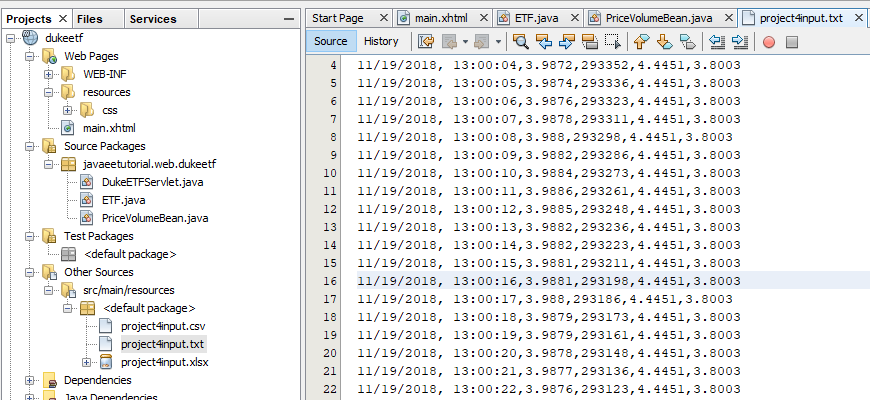


Figure The project4input.txt

1. Updated PriceVolumeBean.java

* I added 2 new variables

*private List<String> linesOfPrices;* - to hold the lines of text from the input file

*private volatile int count = 0;* - to hold the number of lines displayed so far

* Inside the init() method:

*linesOfPrices = readFileLines("project4input.txt");* - to read the input file line by line and store the results in a list of lines

* Inside the timeout() method: - to read current line and send it to the servlet

*if (linesOfPrices.size() > count) {*

*String line = linesOfPrices.get(count++);*

*servlet.send(line);*

*}*

* Added the new private method: - to read all the prices from the input file line by line

*private List<String> readFileLines(String fileName) {*

*List<String> lines = new ArrayList<>();*

*//Get file from resources folder*

*ClassLoader classLoader = getClass().getClassLoader();*

*File file = new File(classLoader.getResource(fileName).getFile());*

*try (Scanner scanner = new Scanner(file)) {*

*while (scanner.hasNextLine()) {*

*String line = scanner.nextLine();*

*lines.add(line);*

*}*

*scanner.close();*

*} catch (IOException e) {*

*e.printStackTrace();*

*}*

*return lines;*

*}*

1. Updated DukeETFServlet.java

* Added new public method: - to send updates about the current prices

*public void send(String etf) {*

*for (AsyncContext acontext : requestQueue) {*

*try {*

*String msg = etf;*

*PrintWriter writer = acontext.getResponse().getWriter();*

*writer.write(msg);*

*logger.log(Level.INFO, "Sent: {0}", msg);*

*acontext.complete();*

*} catch (IOException ex) {*

*logger.log(Level.INFO, ex.toString());*

*}*

*}*

*}*

1. Updated main.xhtml

* Updated the updatePrice method:

*function updatePage() {*

*if (ajaxRequest.readyState === 4) {*

*var arraypv = ajaxRequest.responseText.split(","); document.getElementById("price").innerHTML = arraypv[2]; document.getElementById("volume").innerHTML = arraypv[3]; document.getElementById("date").innerHTML = arraypv[0]; document.getElementById("time").innerHTML = arraypv[1]; document.getElementById("highest52").innerHTML = arraypv[4]; document.getElementById("lowest52").innerHTML = arraypv[5];*

*makeAjaxRequest();*

*}*

*}*

* Added new elements date, time, 52-week high and 52-week low to the main page:

*<tr>*

*<td style="font-size:18pt;font-weight:bold;" width="100px"></td>*

*<td align="left">Date</td>*

*<td id="date" align="right">--/--/----</td>*

*</tr>*

*<tr>*

*<td style="font-size:18pt;font-weight:bold;" width="100px"></td>*

*<td align="left">Time</td>*

*<td id="time" align="right">--:--:--</td>*

*</tr>*

*<tr>*

*<td style="font-size:18pt;font-weight:bold;" width="100px"></td>*

*<td align="left">52-week high </td>*

*<td id="highest52" align="right">--.--</td>*

*</tr>*

*<tr>*

*<td style="font-size:18pt;font-weight:bold;" width="100px"></td>*

*<td align="left">52-week low </td>*

*<td id="lowest52" align="right">--.--</td>*

*</tr>*

# **Screen capture demonstrating the successful compilation and running of the project on your platform.**

1. Run the app

* In NetBeans, find the main.xhtml, right-click and “Run file”. This will deploy the application to GlassFish and run it.
* You can see the user interface by accessing: <http://localhost:8080/dukeetf/main.xhtml>. Load this url in Chrome. I’ve had issues with Internet Explorer not refreshing and getting blocked.

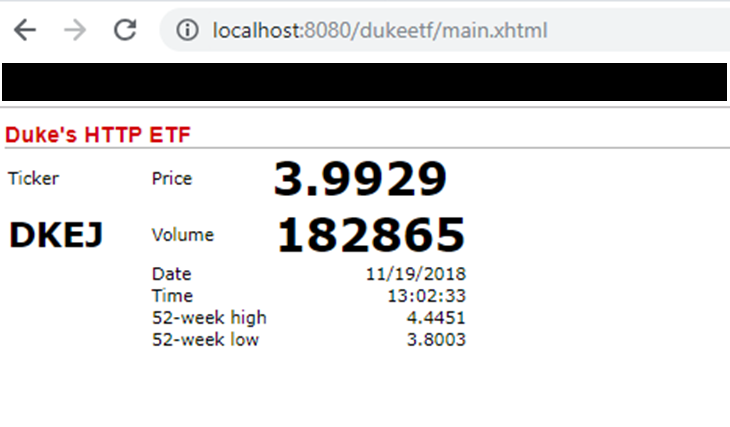


Figure Screen capture 1

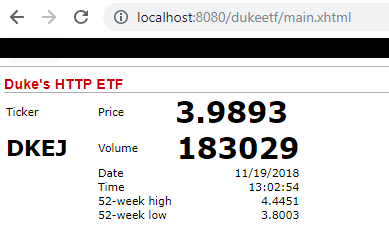


Figure Screen capture 2

# **Lessons Learned**

1. The most important lesson: Keep it simple!

In the code there are evidences of more complex try-outs. For example, the modified code contains an ETF.java class, containing all 6 attributes of the displayed prices: date, time, price, volume, 52-week high and 52-week low.

I created this class in order to store the information read from the input file. See the code below.

The image shows the commented code, which represents the original attempt at solving the problem.

The idea was to:

* read each line from the file
* interpret each value as what it was: LocalDate, LocalTime, double, int
* group this data into an instance of class ETF and send it to the servlet

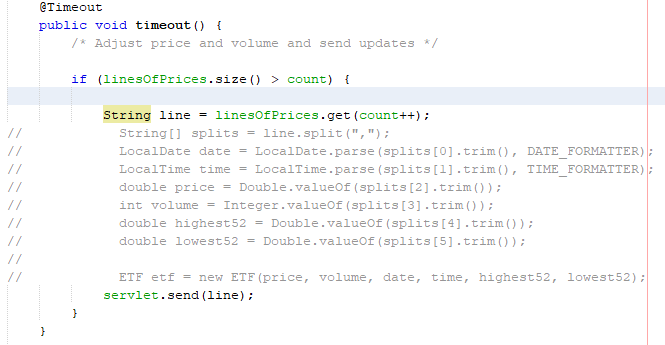


Figure The contents of the timeout method in the PriceVolumeBean.java

This proved more complicated than necessary, because the user interface displays everything as strings anyway. The simple solution was to just get the current line from the file and send it to the servlet so that it can be displayed, avoiding unnecessary processing.