Java/Jakarta EE Lab Activity 1: FirstApp

We will work on this Activity in the first Lab period. It is worth 3% of your final grade, in total. See the Content->Lab Exercises->Submission Requirements document for instructions on how to submit and demonstrate your work.

Complete the Lab Activity and submit the zipped project folder to Brightspace for 1 mark (1% of final grade) and Demonstrate your final results of the application you write from scratch (Step 7 at the end of this document) to your Lab Professor for 2 marks (2% of course grade)

See Brightspace for due date.

High-level Overview

1. Install the Oracle JDK 8 (this is the base Java technology)
2. Install Apache Netbeans 12.0 (this is the IDE we will use for our Jakarta EE projects)
3. Install Jakarta EE (Glassfish 5.1) (this is the reference implementation Application Server, which comprises the Java Enterprise Edition)
4. Install the JEE Sample Applications (these give exemplary examples of various types of Jakarta EE application)
5. Run the Address Book example Application (the address-book application, which we study in this course, is a Java Persistence application that uses JSF pages to do CRUD operations on an Entity, and in the coming weeks you'll understand what all of that means)
6. Write and demonstrate a new Web Application based on the Contact Entity from the Address Book application (after running the address-book application, we will re-write it from scratch, to begin learning about how it is structured and how it works.)

Microsoft Windows 10 is the standard platform for most students in the course. Apple OSX and Linux should also be adequate, but the Lab Instructor may not be able to help with issues that are specific to OSX (if any) or Linux (if any). Follow the sections below to install the CST8218 development environment and example applications.

Installing the Oracle JDK 8

We will be using the Oracle JDK 8 in this course (you will need to create a free account with Oracle if you don't already have one). There are higher version numbers, but JDK 8 is the appropriate version for our Jakarta EE work in this course.

It is best to first uninstall any other JDKs that you are not using (Eclipse now has its own private JDK which is fine). Check Windows->Settings->Add and Remove Programs, and look at C:\Program Files\Java\.

The download link is https://www.oracle.com/java/technologies/downloads/#java8-windows

At the time of this writing, the most recent update is jdk-8u311, and the 64bit Windows installer is at the bottom of the list of links. The default install should be appropriate.

Install Apache Netbeans 12.0

We use the Apache NetBeans IDE to take advantage of its integrated support for the Jakarta EE platform. You can build, package, deploy, and run your Jakarta EE applications from within NetBeans IDE. We will also take advantage of features in Netbeans to write straightforward code automatically.

Download Netbeans 12.0 here:

https://netbeans.apache.org/download/nb120/nb120.html

Look for the executable (*.exe) installers on that page. You want to be sure the version is 12.0, not later, and the default install should be appropriate.

Install Jakarta EE

Find the Glassfish server zip file at the link below.

NOTE: We are installing Eclipse GlassFish 5.1.0 so download that version (not Glassfish 6.x).

NOTE: There are two versions of GlassFish 5.1.0 on the download page:

- We want the one under Jakarta EE 8 **Platform** Compatible Products
- We DO NOT want the one under Jakarta EE 8 **Web Profile** Compatible Products

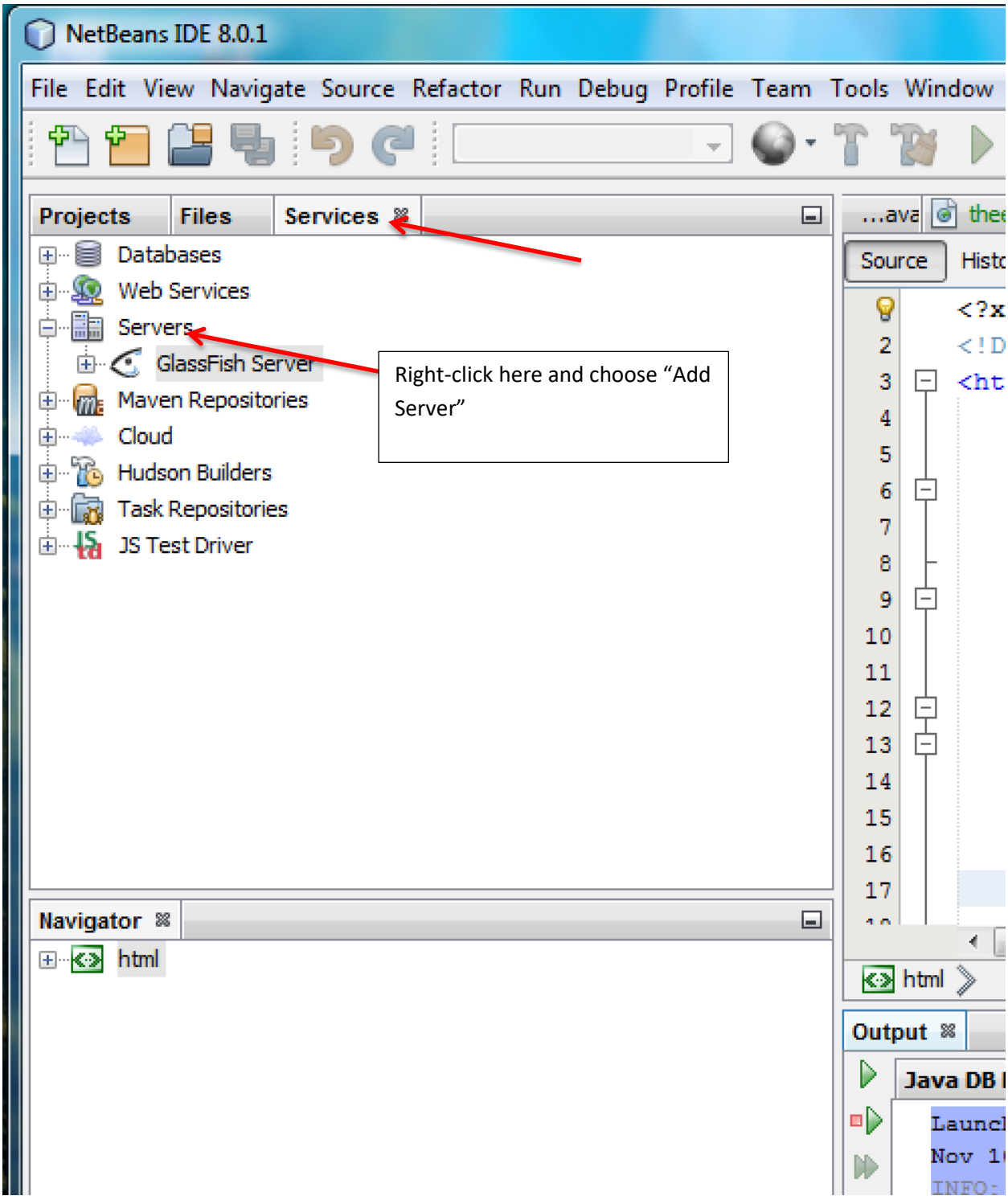
https://jakarta.ee/compatibility/#tab-8

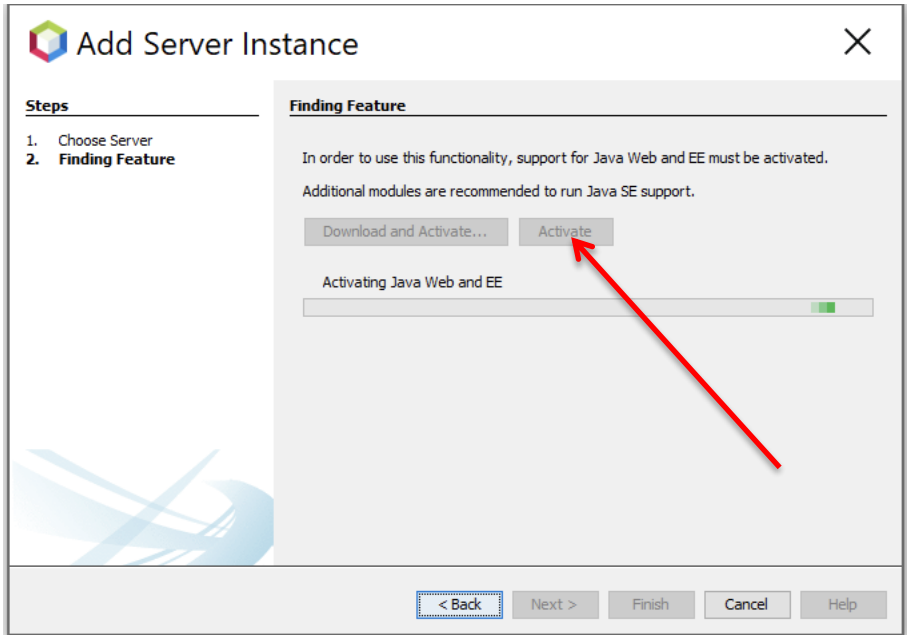
Unzip the downloaded file to C:\, so that after unzipping you have C:\glassfish5\README, C:\glassfish5\glassfish, C:\glassfish5\bin, etc. On windows, this means “Extract All” to C:\

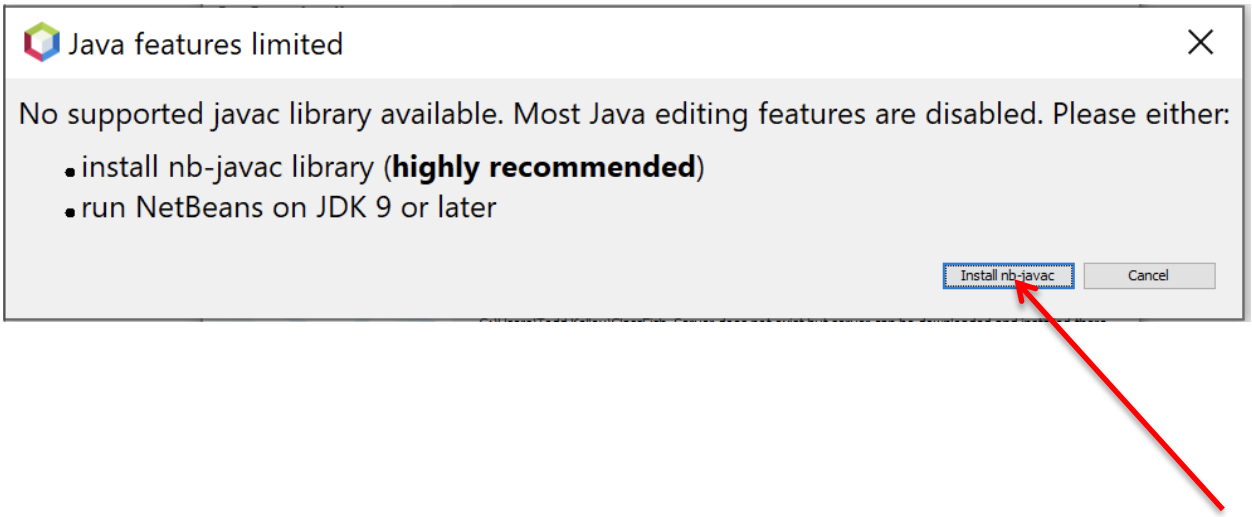
Run Netbeans, dismiss the Start Page, select the Services tab (see first screenshot below), right-click on Servers, select Add Server, then follow the wizard for "Glassfish Server" , and when prompted, activate the necessary modules (see second screenshot below). You will need to restart Netbeans after installing nb-javac, then restart the "Add Server" wizard.

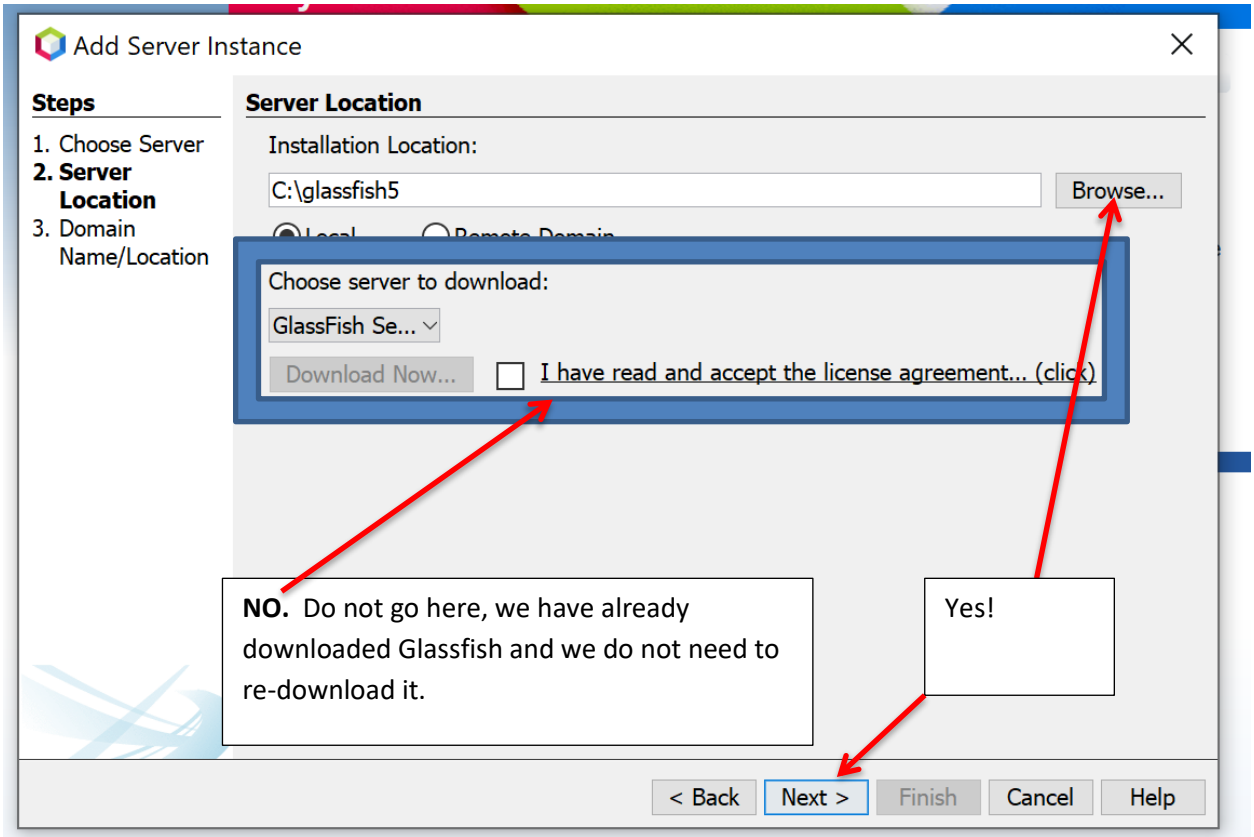
Select your C:\glassfish5 directory as the installation directory (do not re-download, refer to the fourth screenshot below). If it claims C:\glassfish5 is not a valid Glassfish installation, check to make sure you don’t have C:\glassfish5\glassfish5\.... Accept the defaults throughout the rest of the wizard. For our learning/development purposes, we will leave usernames/passwords at their defaults (blank).

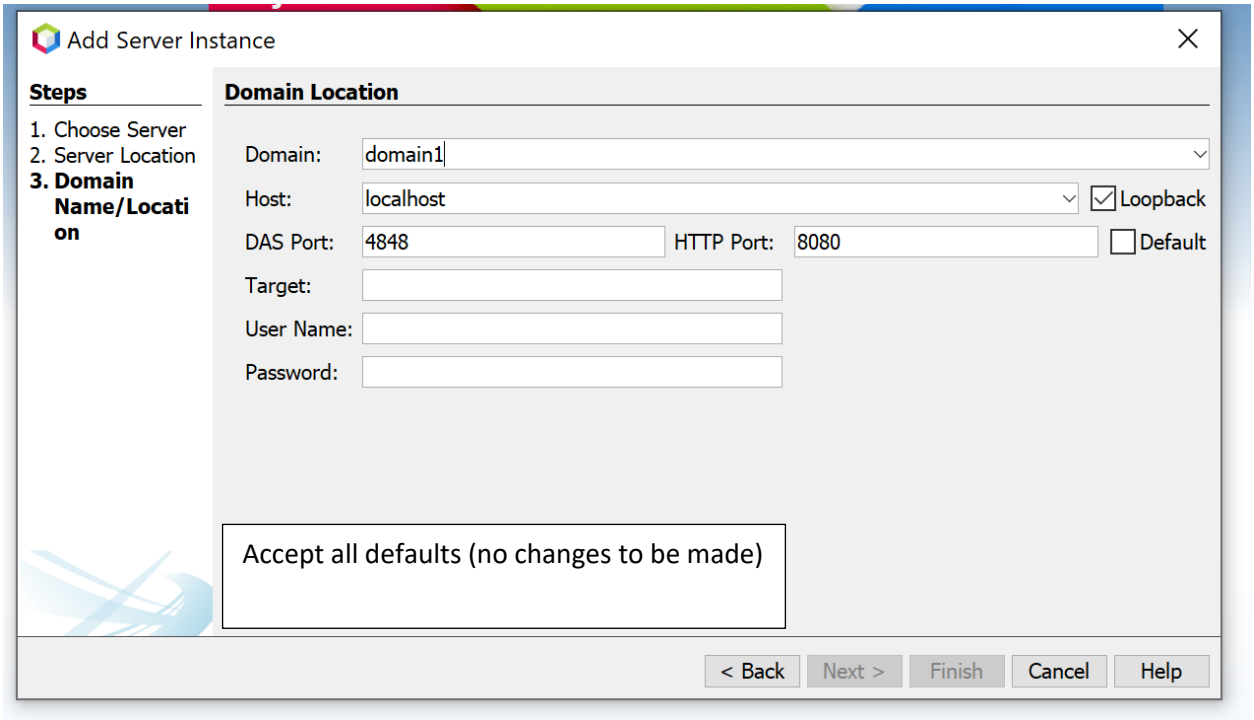
The default port used by Glassfish is 8080 (http) and 8181 (https). Some students who have worked with Oracle Database server have an Oracle-related service on Port 8080. To ensure this will not be a problem for you, enter services.msc in the Windows Start Menu search box to see services running on your computer, and look for XDB (if it's there disable that service by right-click->properties->startup-type->disabled).


NetBeans IDE 8.0.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window
Projects
Files
Services
Databases
Web Services
Servers
GlassFish Server
Maven Repositories
Cloud
Hudson Builders
Task Repositories
JS Test Driver
Right-click here and choose “Add Server”
Navigator
html
Source
Output
Java DB


Add Server Instance
Steps
1. Choose Server
2. Finding Feature
Finding Feature
In order to use this functionality, support for Java Web and EE must be activated.
Additional modules are recommended to run Java SE support.
Download and Activate...
Activate
Activating Java Web and EE
< Back
Next >
Finish
Cancel
Help


Java features limited
No supported javac library available. Most Java editing features are disabled. Please either:
install nb-javac library (highly recommended)
run NetBeans on JDK 9 or later
Install nb-javac
Cancel

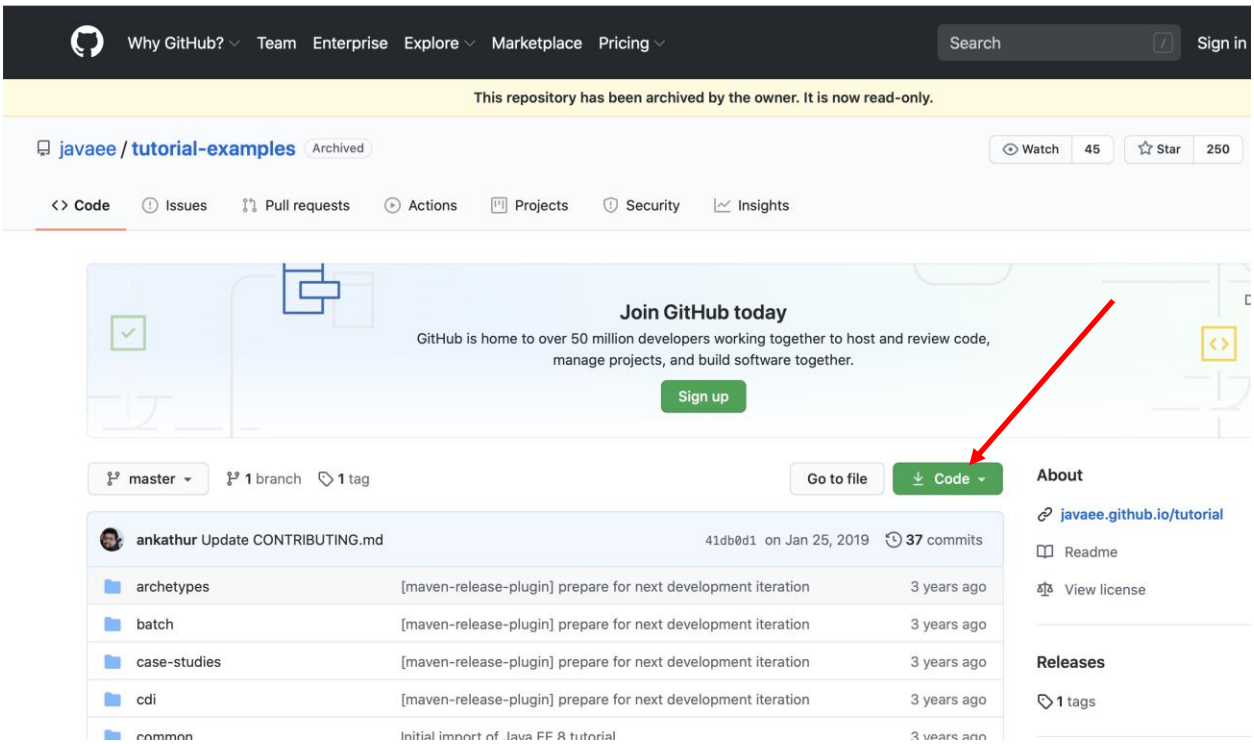

Add Server Instance
Steps
1. Choose Server
2. Server Location
3. Domain Name/Location
Server Location
Installation Location:
C:\glassfish5
Browse...
Choose server to download:
GlassFish Se...
Download Now...
I have read and accept the license agreement... (click)
NO. Do not go here, we have already downloaded Glassfish and we do not need to re-download it.
Yes!
< Back
Next >
Finish
Cancel
Help


Add Server Instance
Steps
1. Choose Server
2. Server Location
3. Domain Name/Location
Domain Location
Domain: domain1
Host: localhost
Loopback
DAS Port: 4848
HTTP Port: 8080
Default
Target:
User Name:
Password:
Accept all defaults (no changes to be made)
< Back
Next >
Finish
Cancel
Help

Java EE Tutorial Example Applications

The JEE Tutorial Example Applications can be downloaded here:

https://github.com/javaee/tutorial-examples



Run the Address Book Example Application

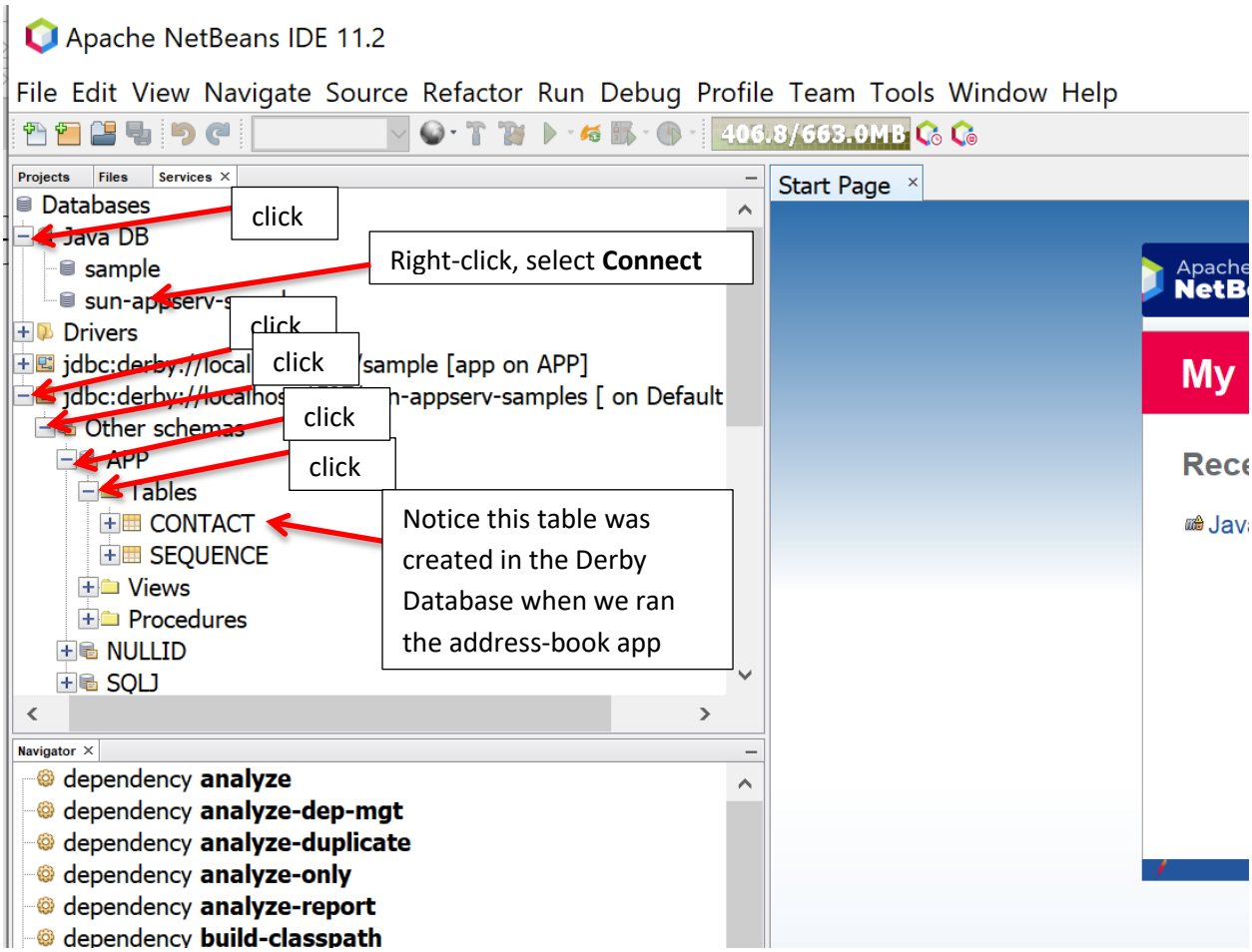
To run the same 3-tier application we discussed (or will discuss) in the classroom lecture, in Netbeans, select File->Open Project and browse to the following location. IMPORTANT TIP: in browsing to the project, click on the disclosure plus sign to the left of any folder name to open the folder, because if you click on the name itself, it will open all projects under that folder. The address-book application is at C:\where\you\put\tutorial-examples-master\persistence\address-book. Double-click the address-book name to open the project.

The project should now appear under the "Projects" tab in Netbeans. To run the application, select the project name in Netbeans and click the green trianglular “Play” button to run the application (it will run as a web application in your default browser). When prompted, specify that you will deploy it on your Glassfish server (select Glassfish server permanently). There will also be a Windows Security pop-up window to confirm that you want to give access, and you should grant access. If all is well, after a while, Netbeans will finally launch your default web browser to open the URL of your Address Book JSF application.

Now, use the just-opened web-browser to explore the running Contacts web application and use the application to create and store a new contact in the database.

Writing a JSF Application

You can quickly write an application like the Address Book application, from scratch, using Netbeans, based on an existing database table. IMPORTANT: Before we do that, let's look at the database table that was created by our running the address-book application (see previous section). The Derby database engine (Java DB) is included with Glassfish, and the sun-appserv-samples database is configured to be the default data source. Refer to the screenshot below. On the **Services** tab in Netbeans, click the disclosure plus-sign next to **Databases** and then **Java DB**, and double-click on **sun-appserv-samples**. That will create a connection to the sun-appserv-samples database called **jdbc:derby://localhost:1527/sun-appserv-samples**. You can drill down to the **CONTACT** table. This table was created by the JEE system according to the fields in the **Contact.java** entity class when we ran the address-book application. You will also see a **SEQUENCE** table that was automatically created to support the auto-generated row-id of each contact in the **CONTACT** table :



To view the contents of the **CONTACT** table, right-click on its name, and select **view data…** from the context menu. Can you see the contact that you created (see previous section)?

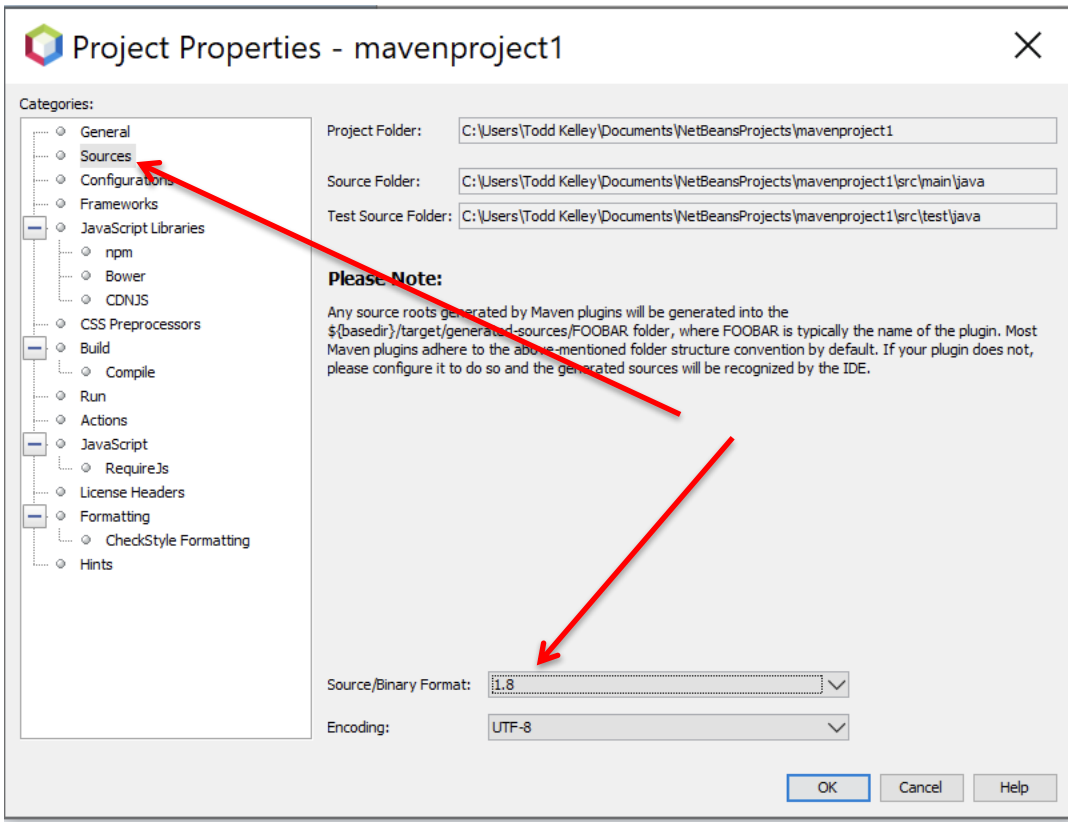
You can find the **Contact.java** file that determined the fields in the **CONTACT** table in

Projects->address-book->Source Packages->javaeetutorial.addressbook.entity->Contact.java

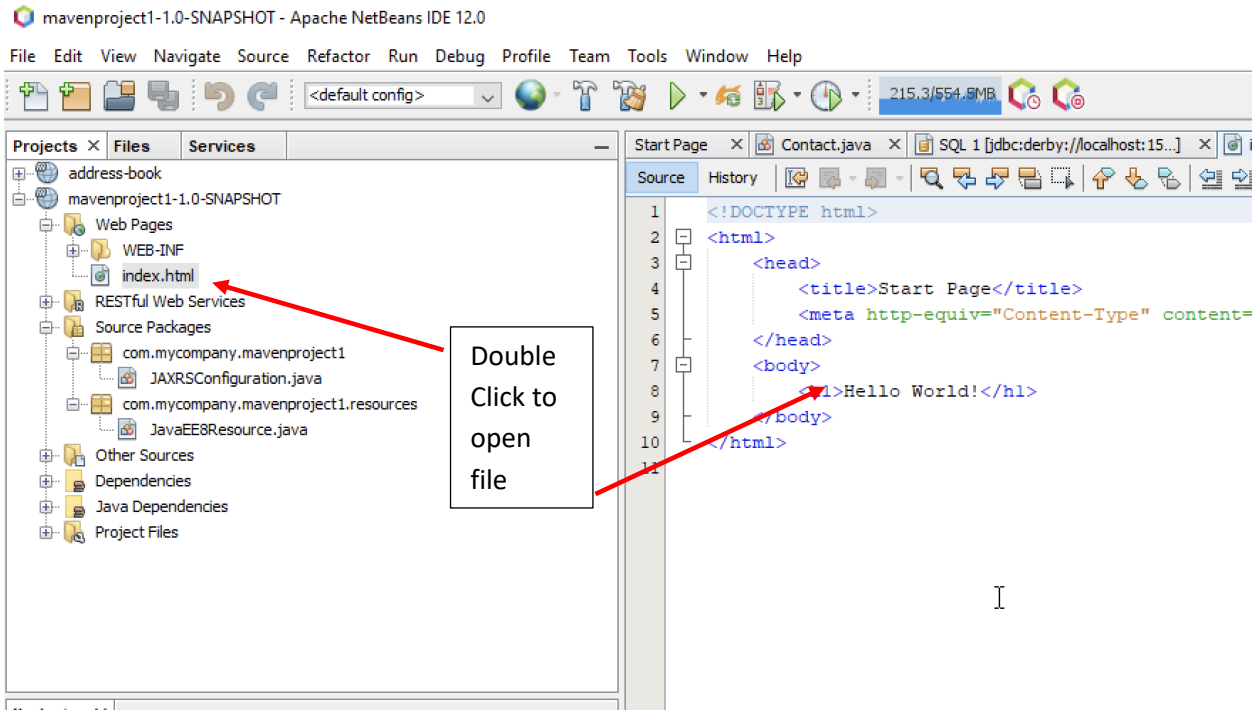
In Step 2 below, we will have Netbeans perform the opposite step, which is to create an entity class Java file from an existing database table.

Rewrite address-book from scratch:

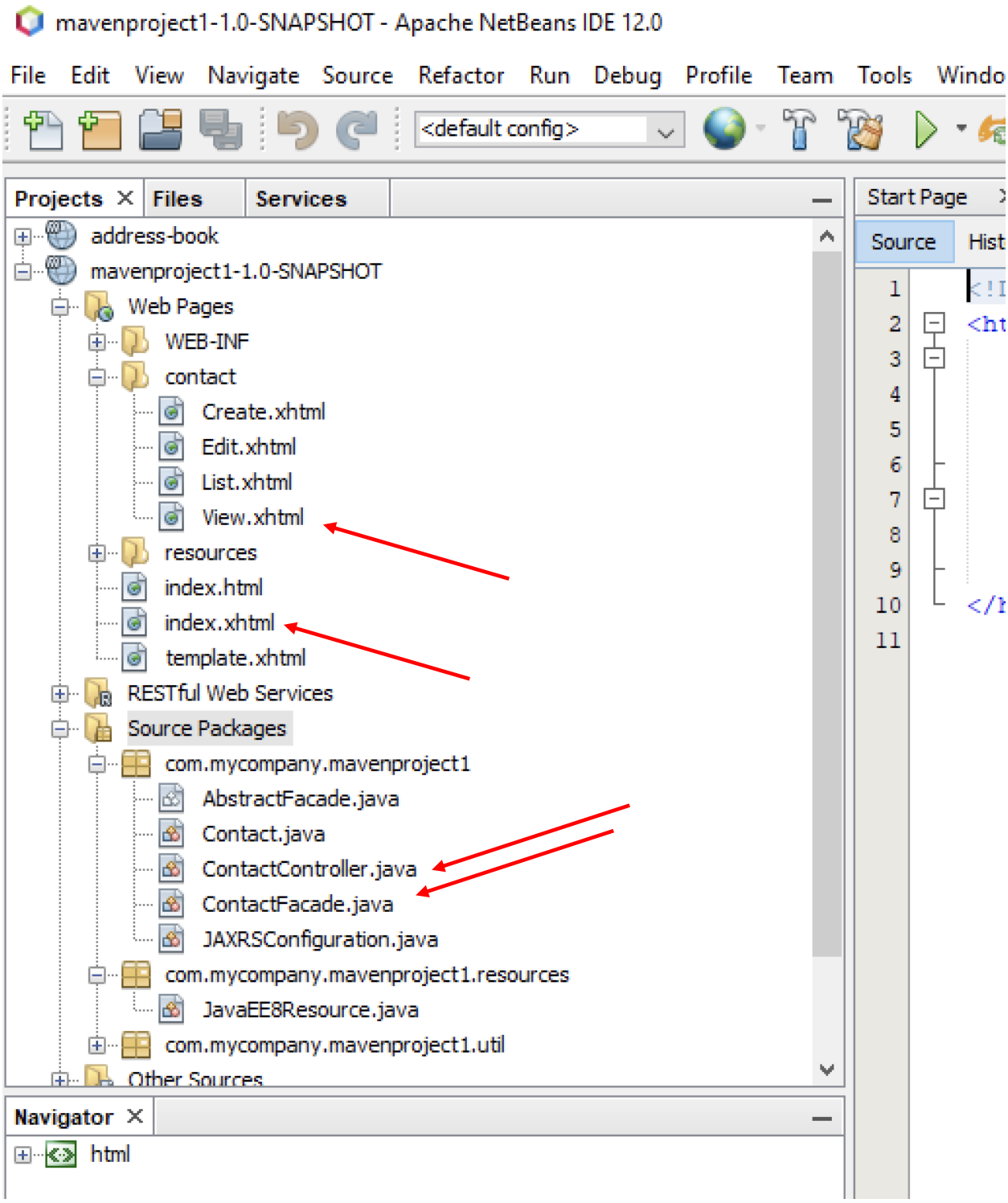
0. You are finished with this lab when you understand and can complete Steps 1-6 without referring to this document. There are three main steps to remember: 1. Create Project, 2. Create Entity class, 3. Create JSF Pages based on the Entity class.
1. Use Netbeans to create a new empty Web Application. At any point, if you want to start over or repeat the process, do not delete and recreate the project with the same name; rather, use a subsequent number on the name or use some other new name. Select **File->New Project->Java-with-Maven->Web Application**, and go through the wizard (accepting all defaults is sufficient, and that will result in a project called **mavenproject1 or 2, or 3, or 4, etc**, unless you rename it in the wizard). Right-click on the project, select properties, and verify the source format shows 1.8 (if not 1.8, it probably means you have the wrong Netbeans version, or that you have other versions of Netbeans and/or JDK installed on your computer, so you should seek help from the lab instructor to recover from that state):



When you run this application, it will say "Hello World" in the browser. At this point we have an empty web application. You can see where the "Hello World" comes from by opening the **index.html** file in the project (see the screenshot below).



2. We want our new application to manage contacts exactly the same as the contacts of the **address-book** application. We have run the **address-book** application which created the appropriate **CONTACT** table in the database according to the address-book **Contact.java** file. In the new application (**mavenproject1**), we can do the opposite, which is to create a **Contact.java** file in **mavenproject1** which contains the fields of the **CONTACT** table in the database. To create a **Contact** entity class based on our existing **CONTACT** table, right-click on your new project, and select **New->Entity Classes from Database**, then specify the **sun-appserv-samples** database connection we mentioned above, and select the **CONTACT** table (and do not select the SEQUENCE table). Complete the wizard; defaults are fine for everything for now. There should now be a **Contact.java** entity class in your **mavenproject1** project, in the package you specified for it.
3. The app now has an entity class but it doesn't do anything with the entity (the application still just uses index.html with "Hello World"). To add the pages and code which provide Create, Read, Update, and Delete operations (CRUD operations) on the Contact Entity, we create the Java Server Faces (JSF) Pages. To do this, right-click on the project, and select **New->JSF Pages from Entity Class**, and select the **Contact** entity which we created in Step 2).
4. Observe that the project has various new files (See screenshot below).



5. Run the new JSF Application, and compare it to the Address Book example. Now the application will use the **index.xhtml** instead of the **index.html** file.
6. Congratulations, you have written a three-tier enterprise application from scratch, with JSF pages to do CRUD operations on a **Contact** entity. We have so far ignored many of the details,

but as the course proceeds, we will revisit those details to build more complete understanding. At this point, you should understand that:

a. Our web applications will deal with an Entity of some kind, which is whatever the application works with, or manages. The applications we've seen so far (address-book and mavenproject1) both work with instances of the **Contact** entity (contacts).
b. We could draw an Entity Relationship Diagram for our application's database but it would have just the one entity and no relationships.
c. Entity instances (Contacts of the Contact.java class in our examples) are stored in a database table.
d. Create, Read, Update, and Delete (CRUD) are the four operations used to manage Entities in these web applications, and we can do those four operations with the JSF pages we created.
e. The **id** field, which identifies a row in the database, appears in our new application on the Create and Edit pages for the user to manage, which is a problem that we will fix in the next lab. We want the **id** field to be managed behind the scenes by the system, as it is in the address-book application.

7. Submit the zipped Netbeans project folder, and show your new running application to your Lab Instructor. Can you find and point out any differences between the running address-book and your application?

APPENDIX A

Clearing the Netbeans Cache

It is unlikely you will need to do this at the Lab 1 stage, but if you've made some mistakes in the instructions and tried your best to recover from those mistakes, this may help. Of course getting help from the professor is appropriate in those circumstances.

When you've double-checked your work and you're convinced everything is correct but unexpected/unexplained problems still arise, usually shown by stack traces in Glassfish Server output, first try restarting Netbeans. If the problem persists, the following technique to clear the Netbeans cache might resolve the issue:

1. Shut down Netbeans

2. Clear the Netbeans cache by deleting the following directory on Windows, where <your_user_name> is your Windows user name. The appdata folder may be hidden according to your settings, but you should be able to browse there explicitly.

C:\Users\<your_user_name>\appdata\Local\Netbeans\cache\

3. Restart Netbeans (wait for Netbeans to finish scanning your projects, etc)