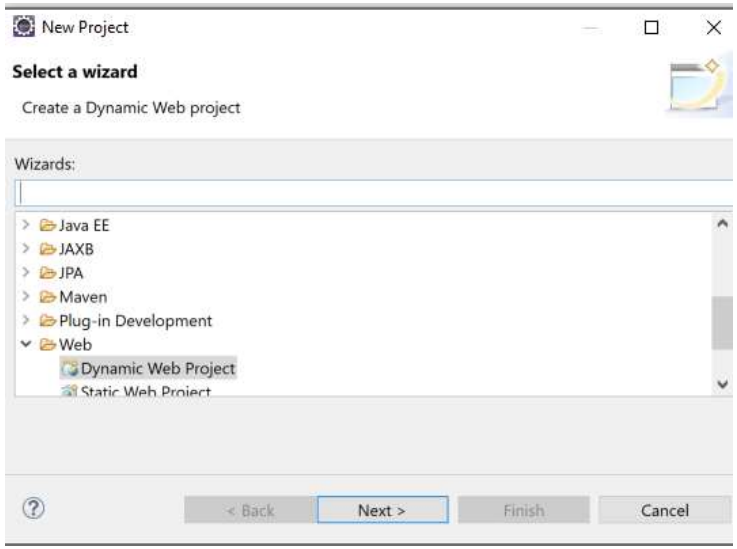
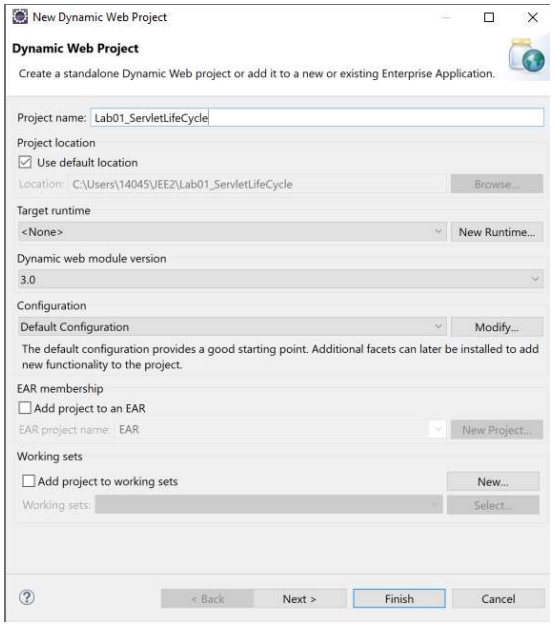
Lab 1 – Servlet Lifecycle

- **We'll start by creating a basic servlet. Launch Eclipse, selecting a default workspace**
- **Create a new Dynamic Web Project**
  - Use the menu bar to select a new project (Files>New>Project…)
  - Search for Dynamic Web Project in the search bar, select the Dynamic Web Project and then "Next >"
  - Enter "Lab01_ServletLifeCycle as the Project Name and then select "Finish >" [1]



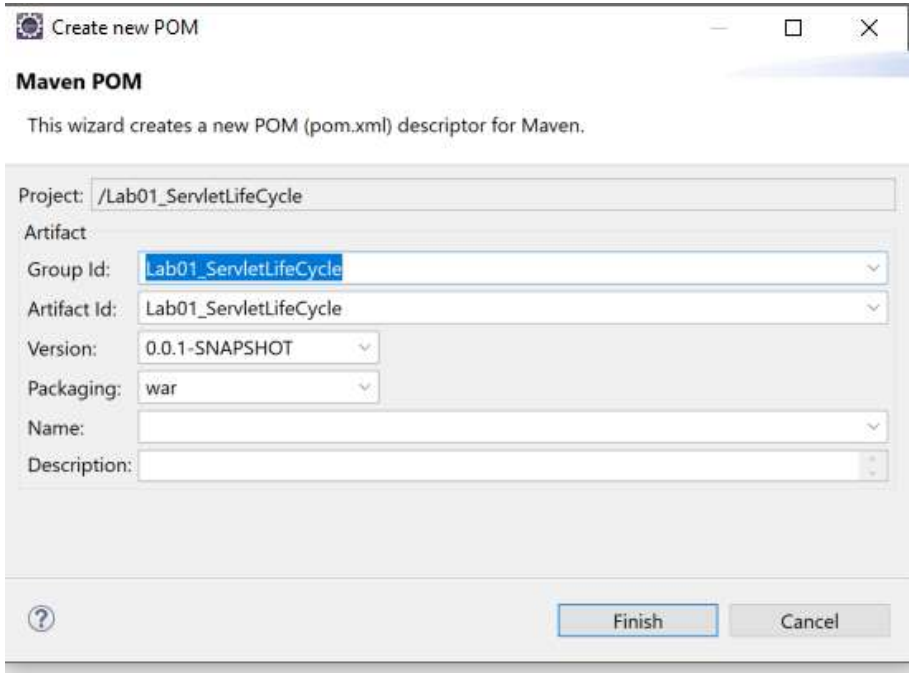


  - Open the EE perspective if prompted (otherwise select Window>Perspective>Open Perspective>Java EE to open the EE perspective)

Lab 1 – Servlet Lifecycle

Convert the project to a Maven project

- Within the Project Explorer, configure the project as a Maven project (right click on the project> Configure>Convert to Maven Project)
- Within the Create new POM window, select Finish to convert the project



Lab 1 – Servlet Lifecycle

Add the Servlet API maven dependency to the pom.xml file

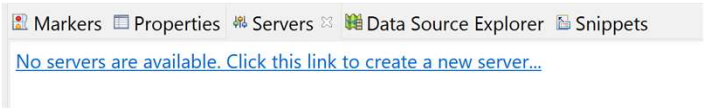
- Open the pom.xml file and add the following (immediately before the Build tag):

```
<dependencies>
        <dependency>
                <groupId>org.apache.tomcat</groupId>
                <artifactId>tomcat-servlet-api</artifactId>
                <version>9.0.1</version>
        </dependency>
</dependencies>
```

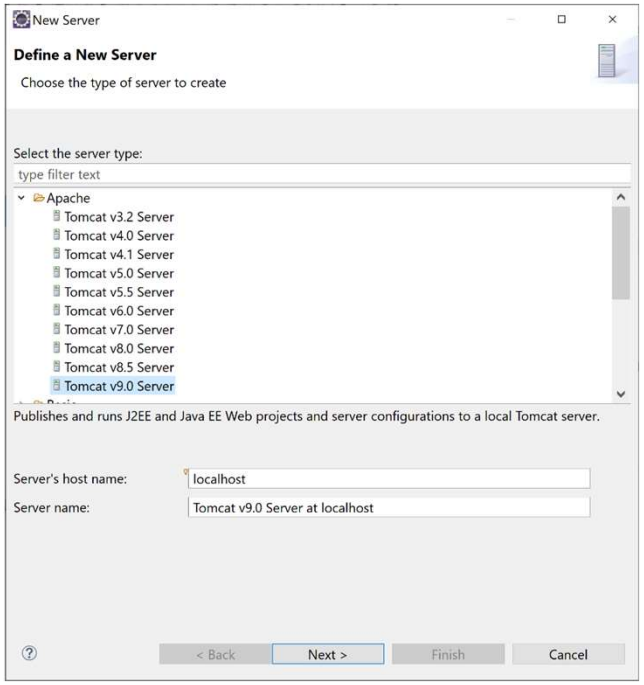
- Save the pom.xml file
- Update the Maven Project
  - ✓ Within the Explorer window, right click on the project > Maven > Update Project

Lab 1 – Servlet Lifecycle

- **Add a Server**
  - Within the server tab, click on "Click this link to create a new server…" (if this window isn't visible, then use the menu bar to make it visible Window>Show Window>Servers…



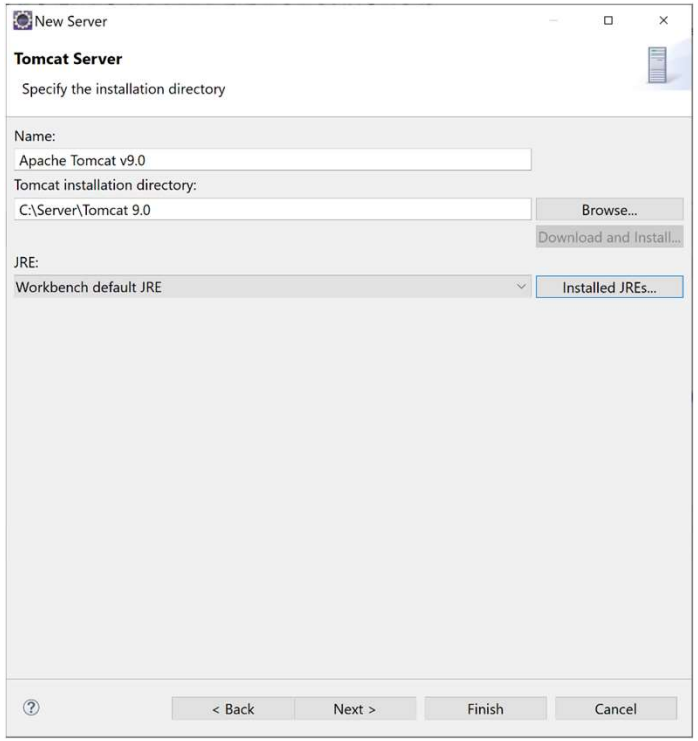
  - Within the New Server window, select Apache>Tomcat v9.0 Server and then select "Next >"



Lab 1 – Servlet Lifecycle

Add a Server

- Within the New Server window, use Browse… to locate and set the Tomcat Installation directory [1]
- Select "Finish" to create the Tomcat Server

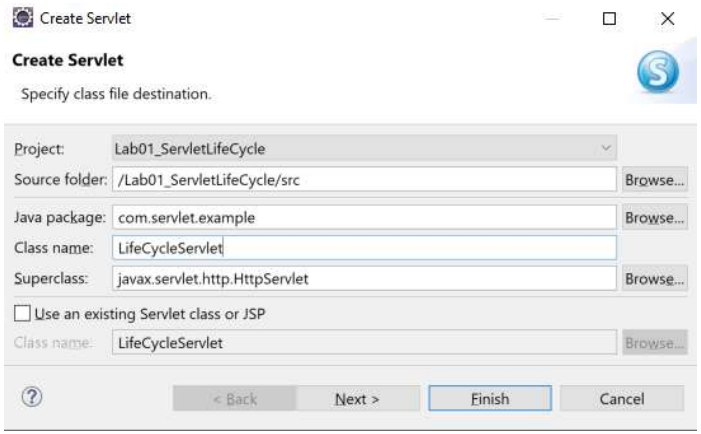


Lab 1 – Servlet Lifecycle

Create a Servlet

- Within the Project Explorer right click on the Java Resources, select New>Servlet

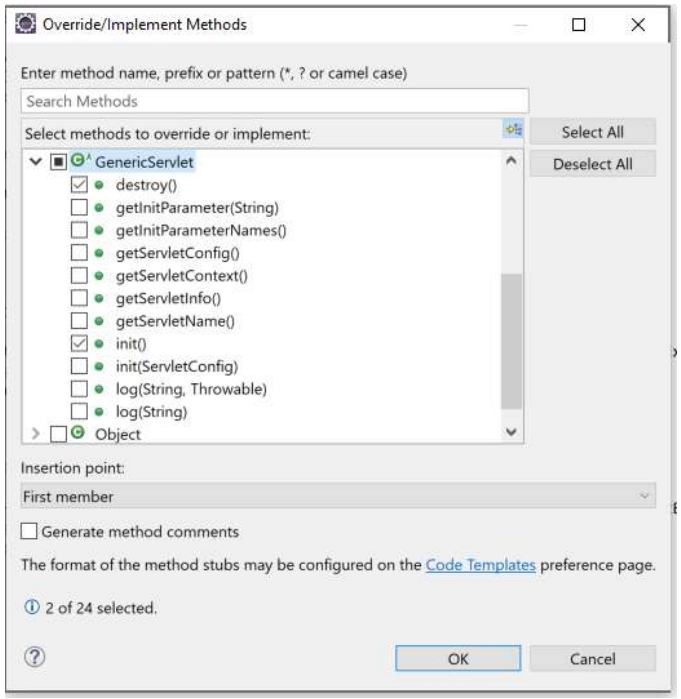
- Within the Create Servlet window enter a Java package name of "com.servlet.example", enter a Class name of "LifeCycleServlet", and then select "Finish"



Lab 1 – Servlet Lifecycle

Override the init and destroy lifecycle methods

- **Start by overriding the init and destroy methods from GenericServlet**
  - Right click anywhere in your code > Source > Override/Implement Methods > GenericServlet
  - Check the boxes next to destroy() and init()



Lab 1 – Servlet Lifecycle

Add Output messages

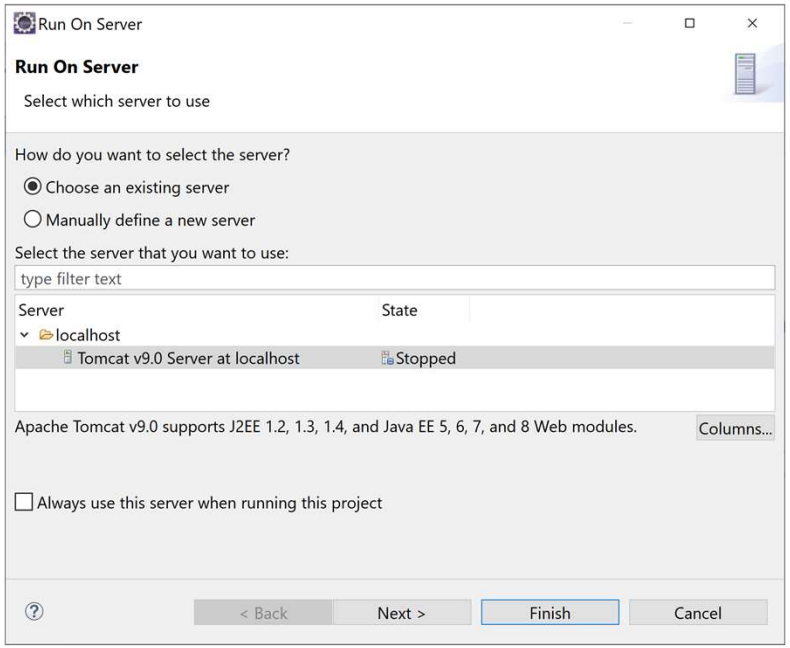
- **Put a simple System.out.println(***message to identify the method here)* **in the following places**
  - init
  - doGet
  - doPost
  - The default constructor
  - Destroy

- **init method example**

```
public void init() throws ServletException{
    System.out.println("init method in LifeCycleServlet called");
}
```

Lab 1 – Servlet Lifecycle

- **Use an external Web Browser**
  - On the menu bar select Window>Web Browser > Chrome if it exists or anything other than Internal Web Browser [1]
- **Run the Servlet on Tomcat**
  - Within the Project Explorer window, right click on the LifeCycleServlet.java file and select Run> Run As > Run on Server
  - Confirm that Tomcat v9.0 Server at localhost is selected and then select "Finish"



Lab 1 – Servlet Lifecycle

Confirm that your output is something similar to what's below

```
Default constructor in LifeCycleServlet called
init method in LifeCycleServlet called
doGet method in LifeCycleServlet called
```

Why didn't the destroy method get called?

- We'll learn more later but for now enter the following line of code at the end of your doGet method
  request.getSession.invalidate();
- Run the servlet again – you should see the 3 messages above
- Go to the Servers tab, right click on your server and choose "Stop"
- Open the console again and you should see your message from the destroy method being output

```
destroy method in LifeCycleServlet called
```