CS425 – Spring 2021

Class Project: Create a Java web project in Eclipse using Maven

Credit: Many online resources

What is Maven?
Maven is a build automation tool used primarily for Java projects
The Maven project is hosted by the Apache Software Foundation

PostgreSQL JDBC driver and Maven repositories
There are different approaches to design and configure your project in Java.
One of the most common is using Apache Maven. The benefit of Apache Maven is that you only need to add dependencies in the POM file; the rest is taken care of by Maven. You do not need to add the JAR files manually yourself. You can simply add the PostgreSQL JDBC dependency to the project's POM file.

What does a POM file do?
POM stands for Project Object Model. It is an XML file that stores crucial information about a Maven project, including configuration details used by Maven.
It's the place where we can specify plugins, dependencies, and project versions.

Below is a sample POM file where we can see a few basic XML tags.

*mavenwebexample/pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.cs425.javaweb</groupId>
  <artifactId>mavenwebexample</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>mavenwebexample Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>

  </dependencies>

  <build>
```

Eclipse IDE for Enterprise Java Developers

382 MB | 646,847 DOWNLOADS

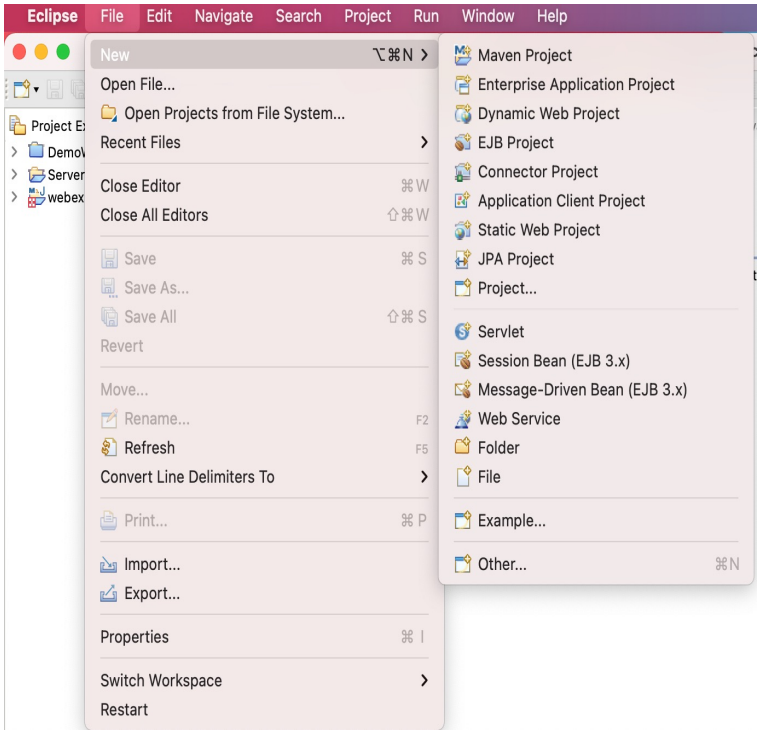
Tools for developers working with Java and Web applications, including a Java IDE, tools for Web Services, JPA and Data Tools, JavaServer Pages and Faces, Mylyn, Maven and Gradle, Git, and more.

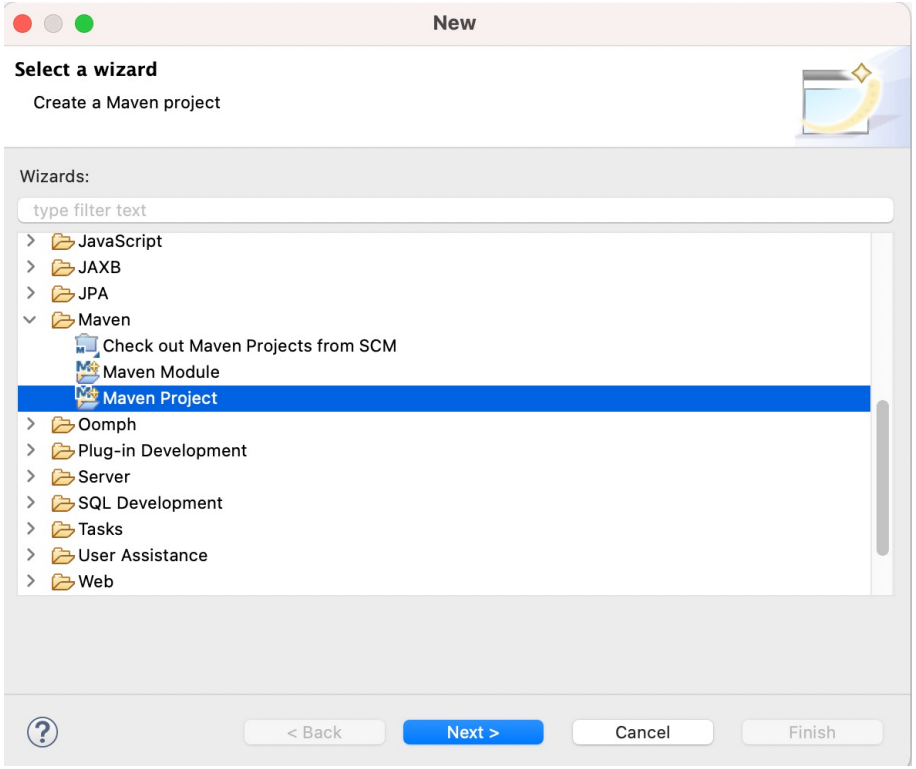
Click here to file a bug against Eclipse Web Tools Platform.
Click here to file a bug against Eclipse Platform.
Click here to file a bug against Maven integration for web projects.
Click here to report an issue against Eclipse Wild Web Developer (incubating).

Windows x86_64
macOS x86_64
Linux x86_64

Create Maven web project

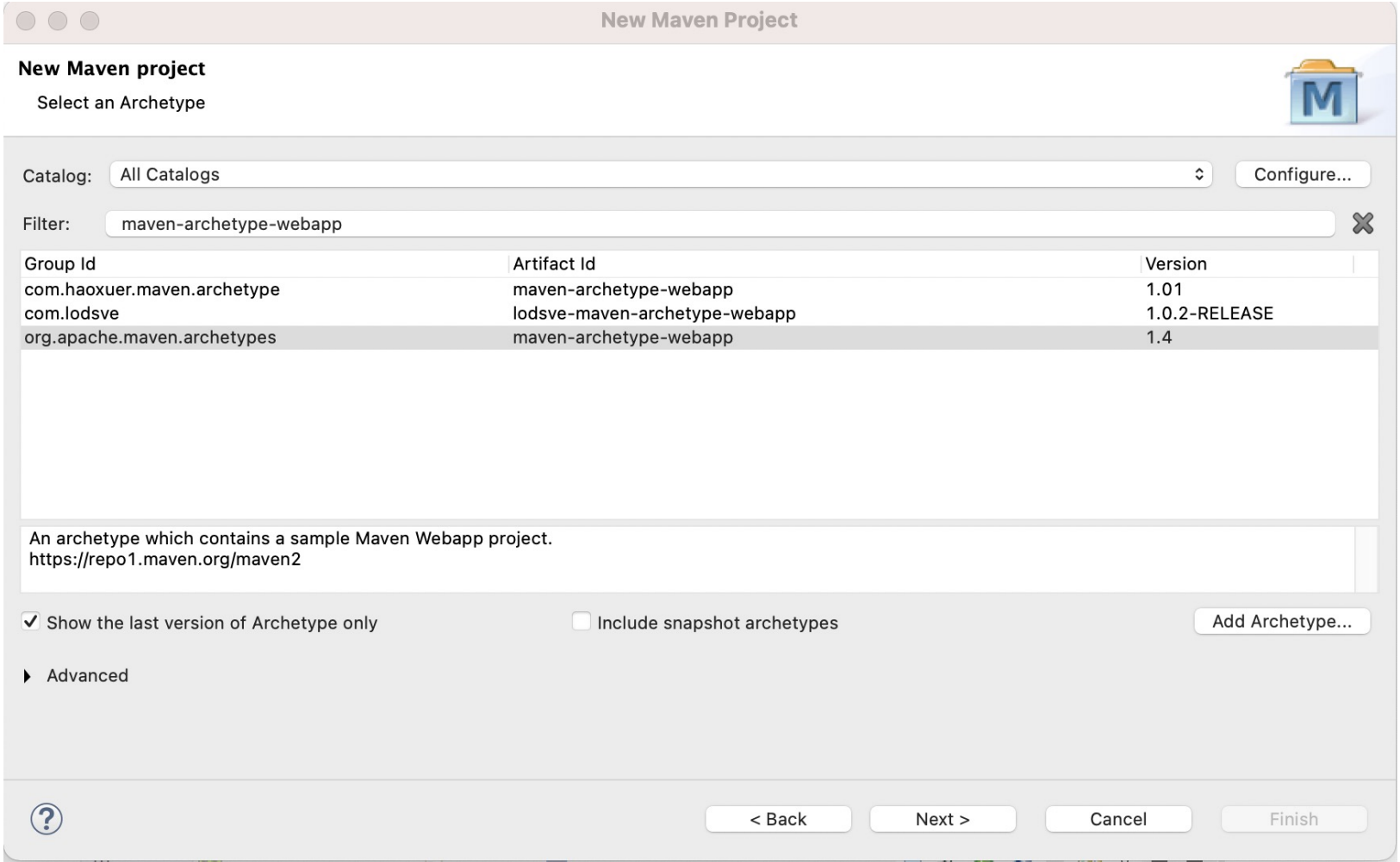
- Create a new Maven Project called **DemoJavaWebApp** via the File>New>Other...>Maven>Maven Project





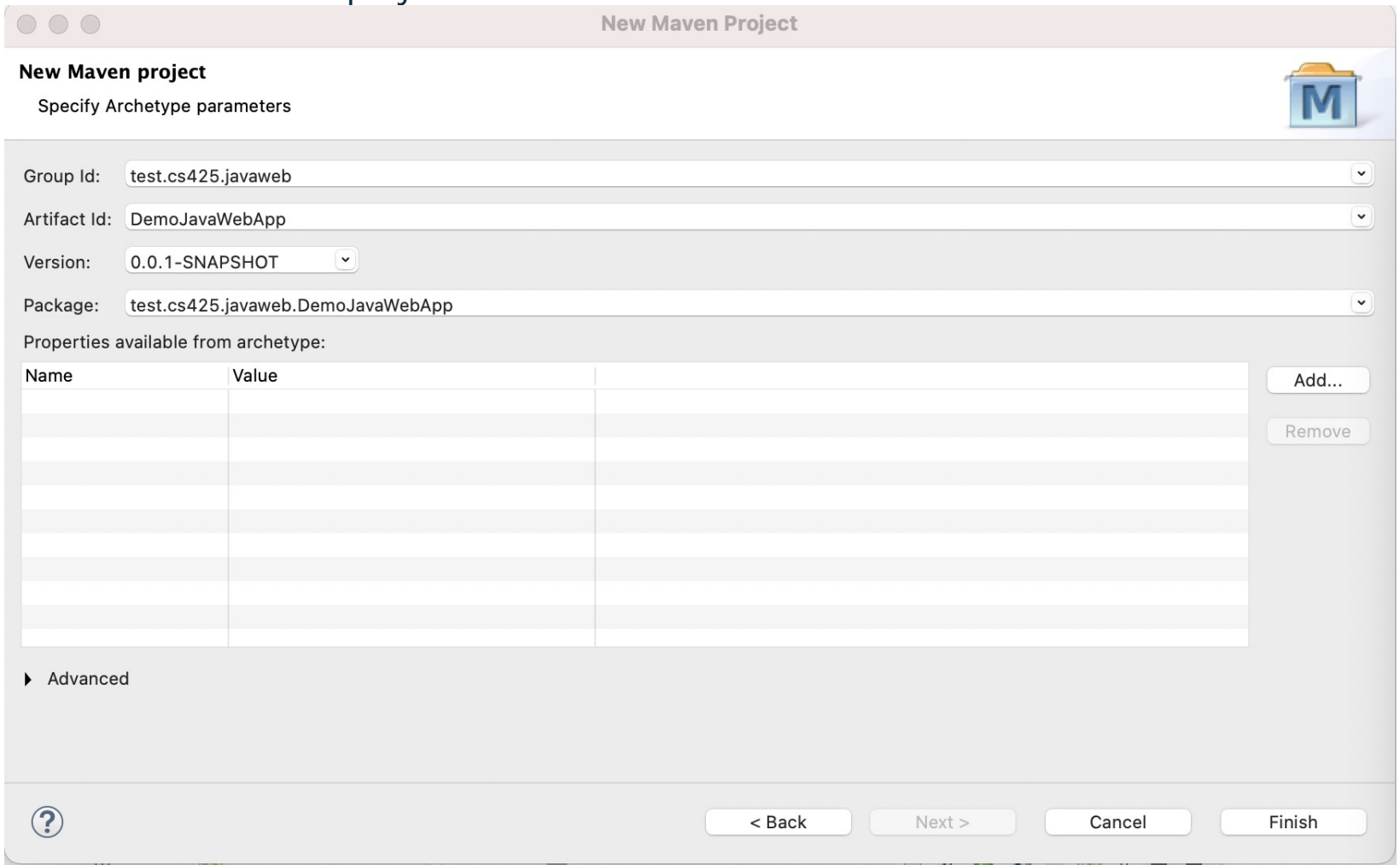
Create Maven web project

- Press next, filter for the " **maven-archetype-webapp**" archetype and select the **org.apache.maven.archetypes** entry.
  - This is the classical Maven example archetype for project creation.
- Click the Next button.



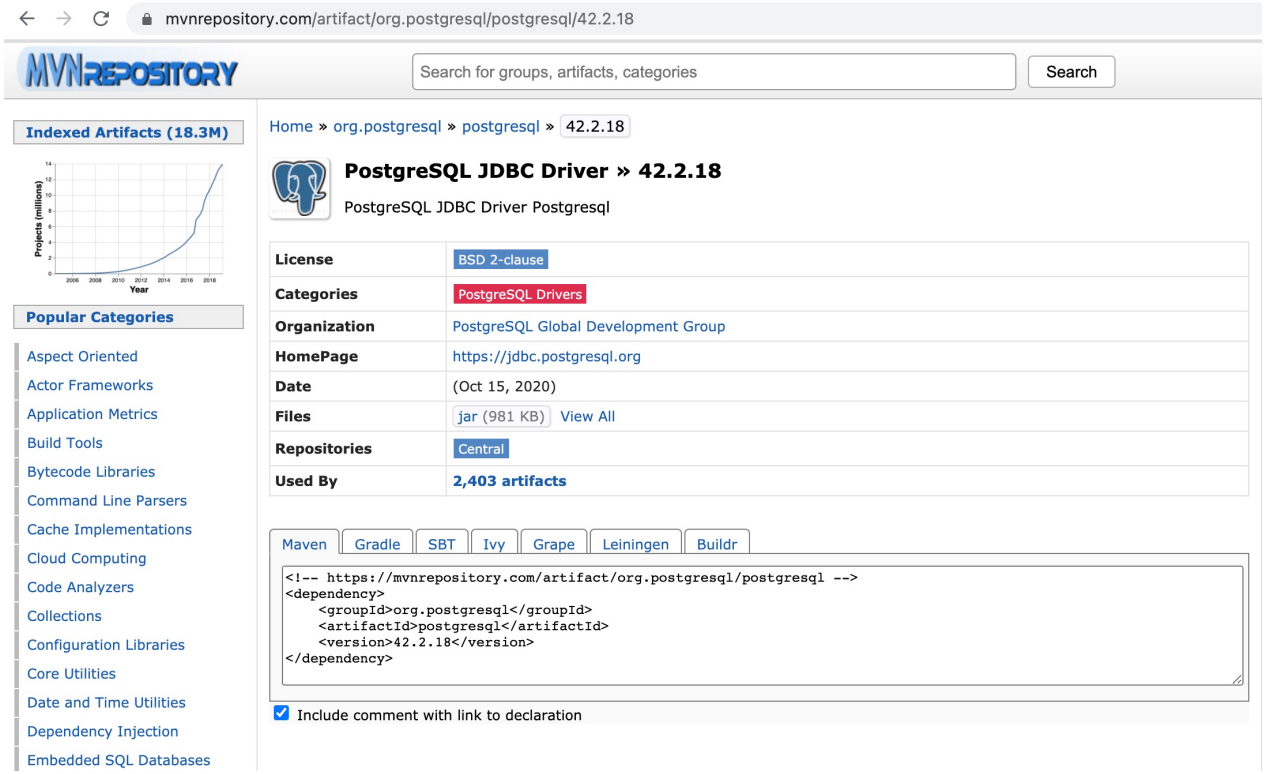
Create Maven web project

- Enter the group, artifact and version of your new Maven component.
  - <groupId> is the name of the company or group that created this project, <artifactId> is the name of the project, <version> as the name suggest is the version of the project



Create a Java web project in Eclipse using Maven

- To add the PostGreSQL dependency
- It assumes that you have already configured Eclipse for the creation of web applications
- On Maven Central (https://mvnrepository.com/artifact/org.postgresql/postgresql) you will be able to find the tag for the dependency, as shown below:
- https://mvnrepository.com/artifact/org.postgresql/postgresql/42.2.18



Create a Java web project in Eclipse using Maven

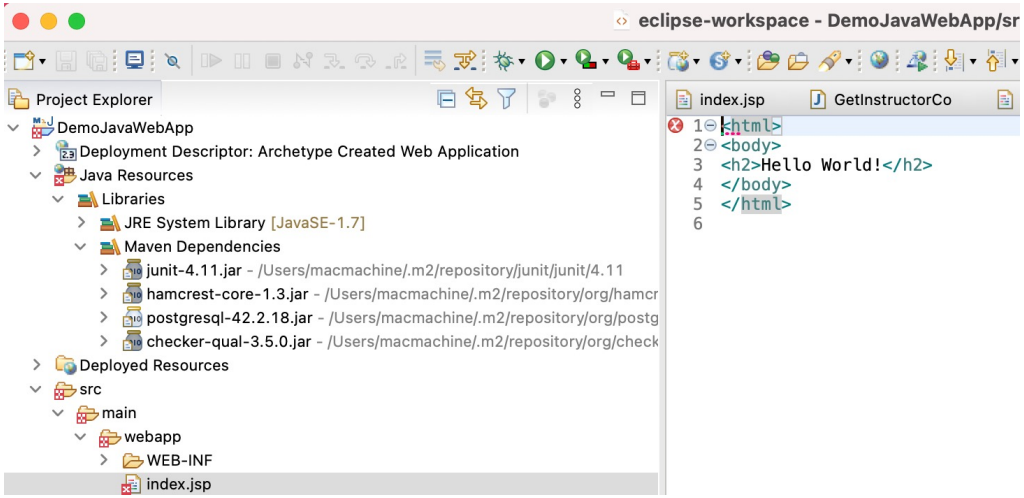
- To add the PostGreSQL dependency
- Copy the dependency tag into your project **pom.xml**, as shown below:

```
22  <dependencies>
23    <dependency>
24      <groupId>junit</groupId>
25      <artifactId>junit</artifactId>
26      <version>4.11</version>
27      <scope>test</scope>
28    </dependency>
29    <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
30  <dependency>
31      <groupId>org.postgresql</groupId>
32      <artifactId>postgresql</artifactId>
33      <version>42.2.18</version>
34  </dependency>
35    </dependencies>
```

- As soon as you save your POM file, you will notice that the Dependencies tree is updated, and PostgreSQL JDBC is displayed.

Create a Java web project in Eclipse using Maven

- To Install tomcat apache

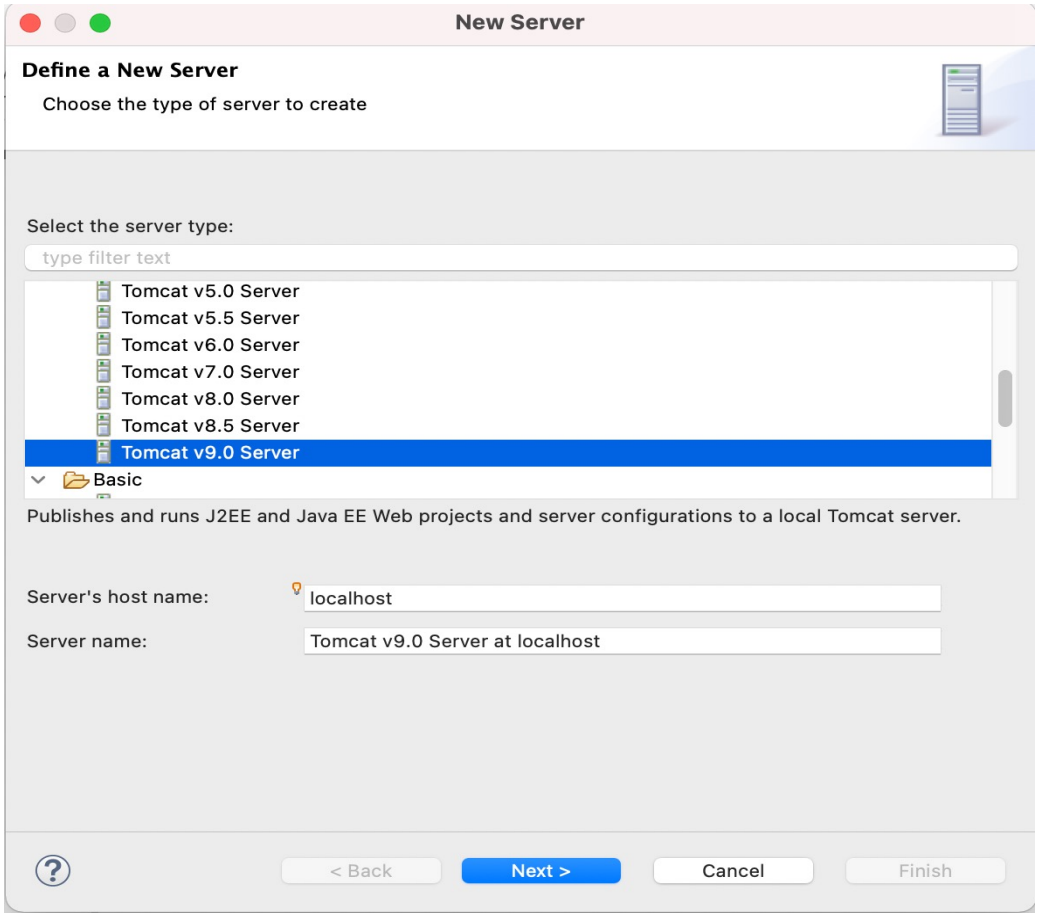


- Step-1:
  - Download Apache Tomcat from:
    - https://tomcat.apache.org/download-90.cgi

- Step-2:
  - Extract it to **Document** folder.

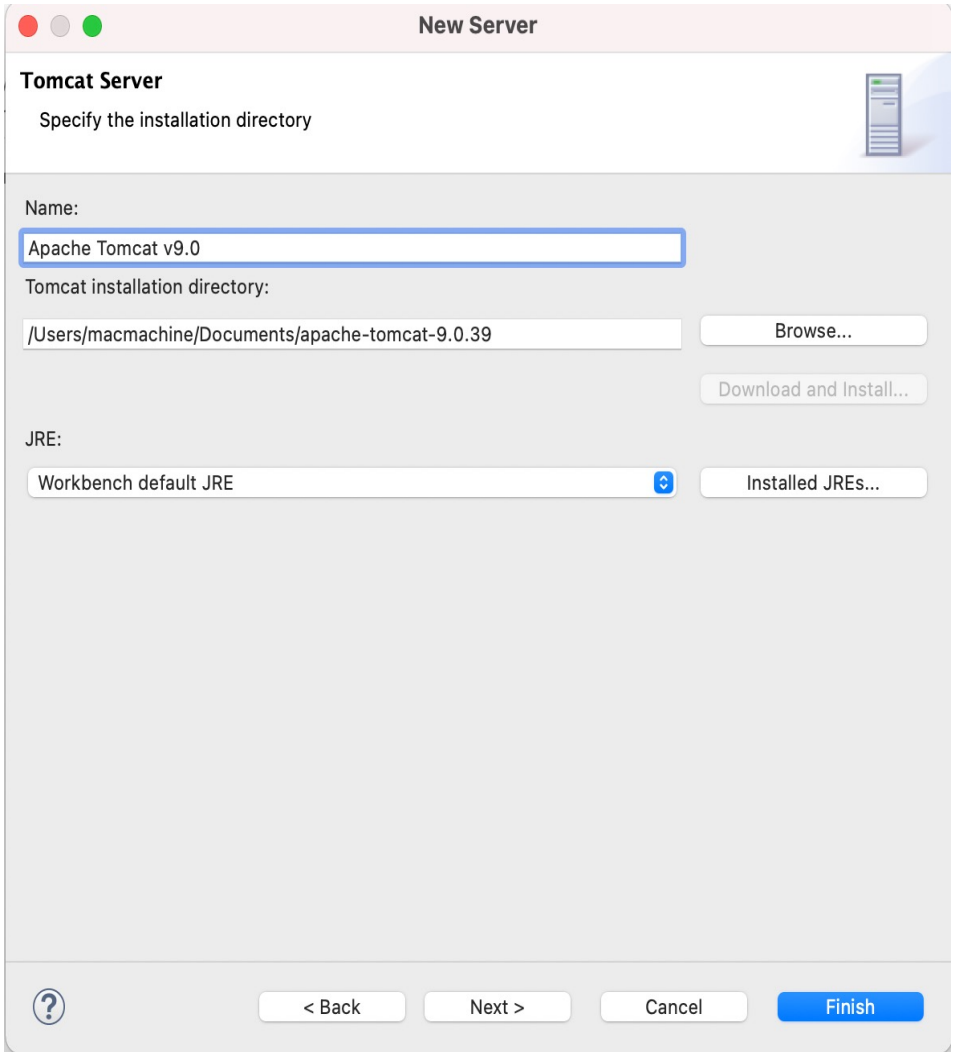
Create a Java web project in Eclipse using Maven

- To Install tomcat apache
- Step-3
  - Open Eclipse Environment
  - Click on Servers Tab
  - Click on No servers are available. Click this link to create a new server...
  - Click Tomcat v9.0 Server and Next



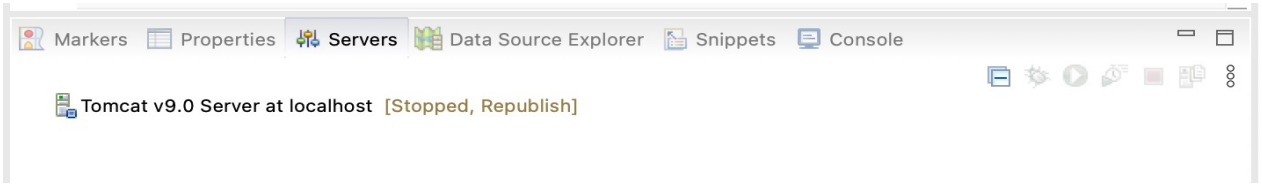
Create a Java web project in Eclipse using Maven

- To Install tomcat apache
- Step-4
  - Select Apache installation Directory and click Finish.



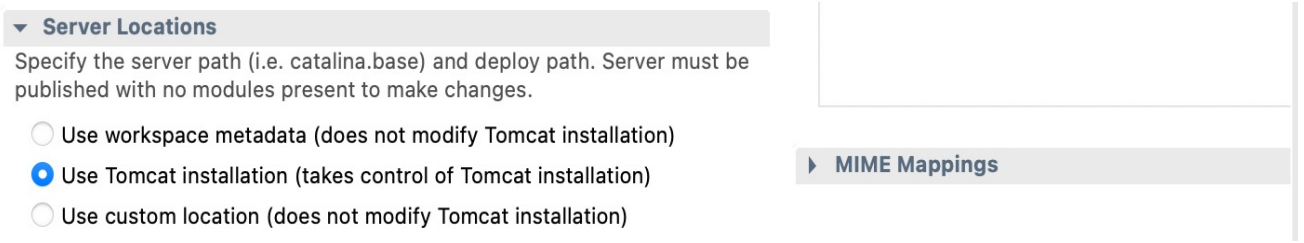
Create a Java web project in Eclipse using Maven

- To Install tomcat apache
- You should see Tomcat v9.0 Server at localhost [Stopped, Republish] under Servers tab.



Create a Java web project in Eclipse using Maven

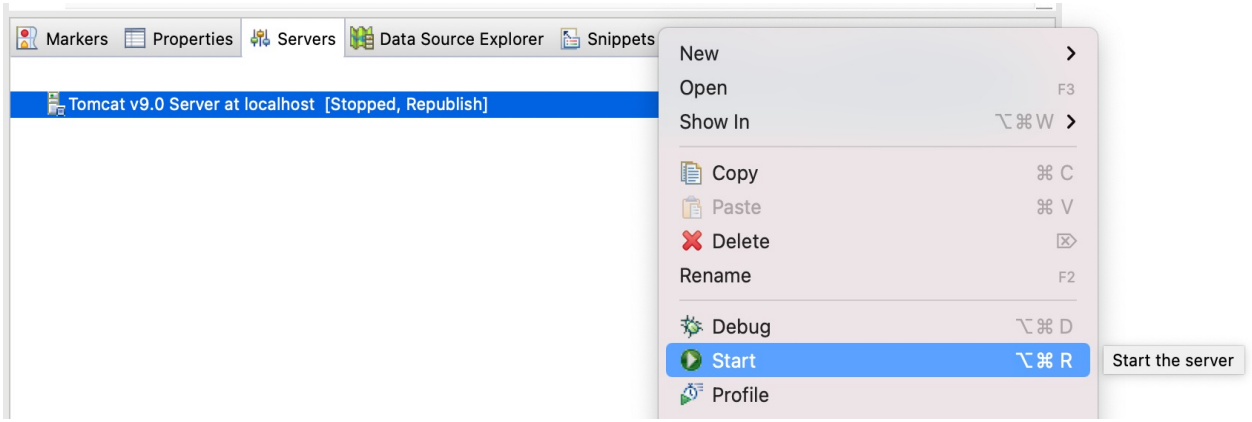
- To Install tomcat apache
- Step-6
  - Double click on Tomcat v9.0 Server at localhost .
  - New Apache Tomcat configuration page will open
  - Go to Server Location section
  - Select **Use Tomcat installation (takes control of Tomcat installation)**



  - Save configuration

Create a Java web project in Eclipse using Maven

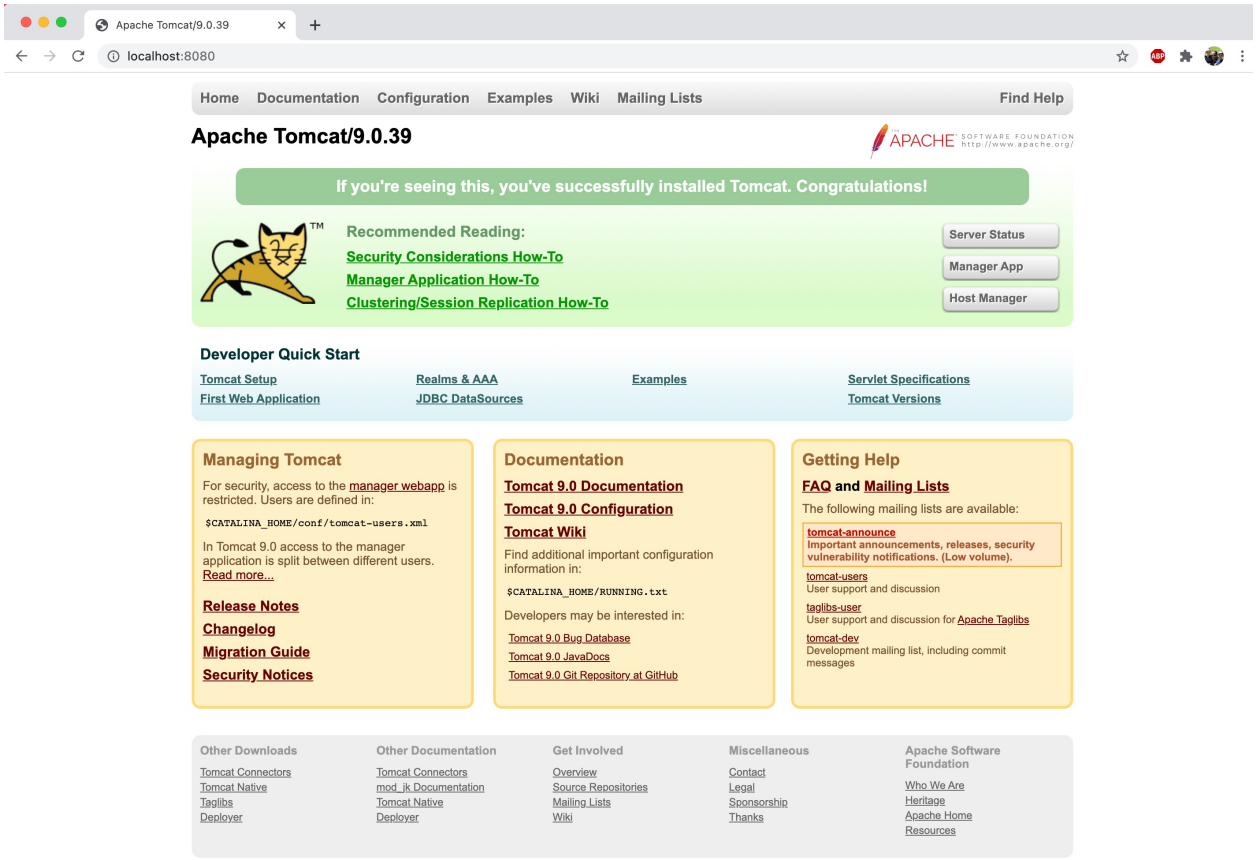
- To Install tomcat apache
- Step-7
  - Now right click on Server and click Start Double click on Tomcat v9.0 Server at localhost .



  - It should be up and running on port 8080 and you could visit default page using URL: **http://localhost:8080/**

Create a Java web project in Eclipse using Maven

- To Install tomcat apache
- Step-6
  - It should be up and running on port 8080 and you could visit default page using URL: **http://localhost:8080/**



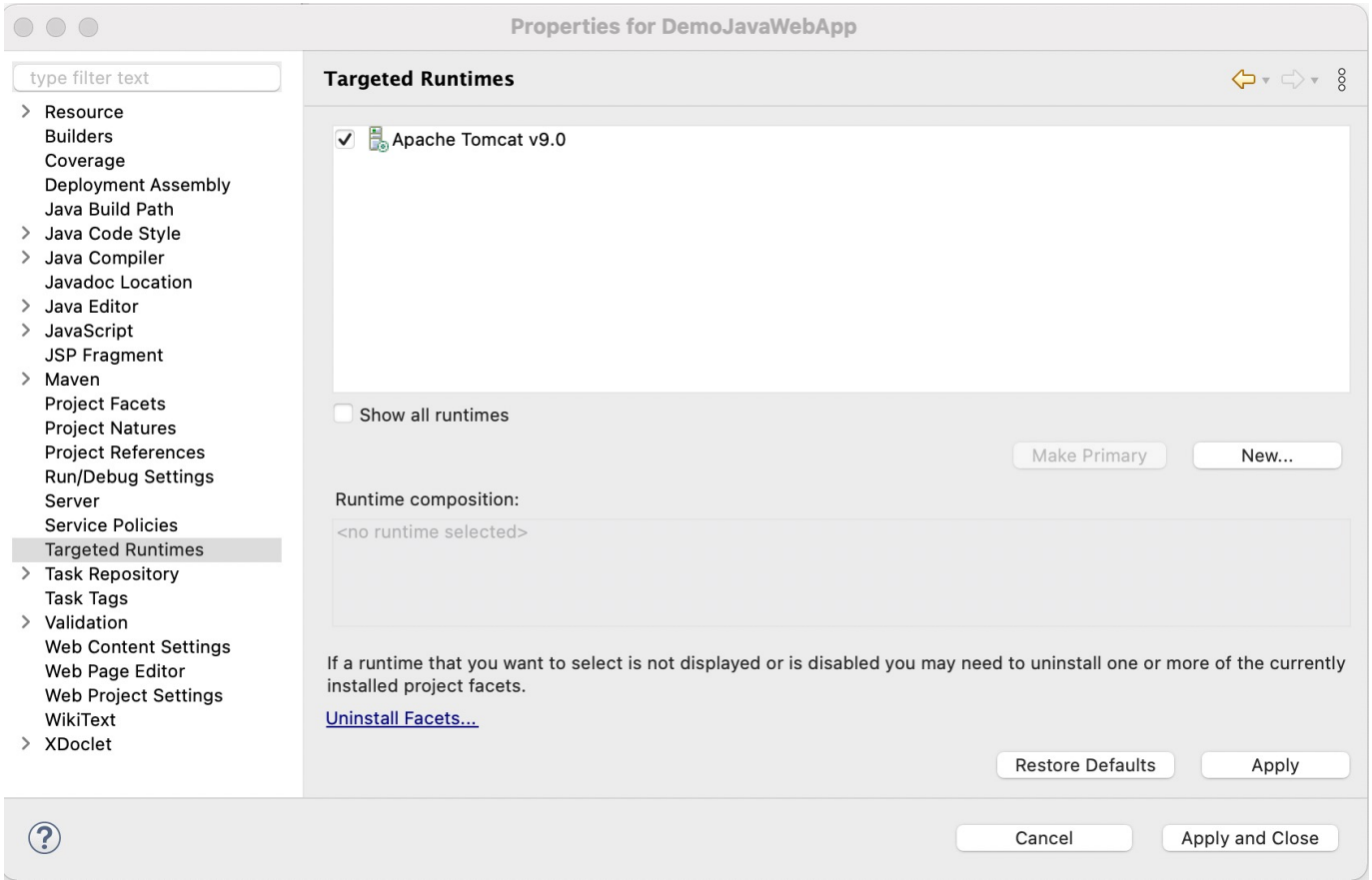
Create a Java web project in Eclipse using Maven

- Connect the project with Apache Tomcat

webexample/pom.xml | index.jsp

```
1 <html>
2 <body>
3 <h2>Hello World!</h2>
4 </body>
5 </html>
```

- Open the project Properties>Target Runtimes>Checked the Apache Tomcat.



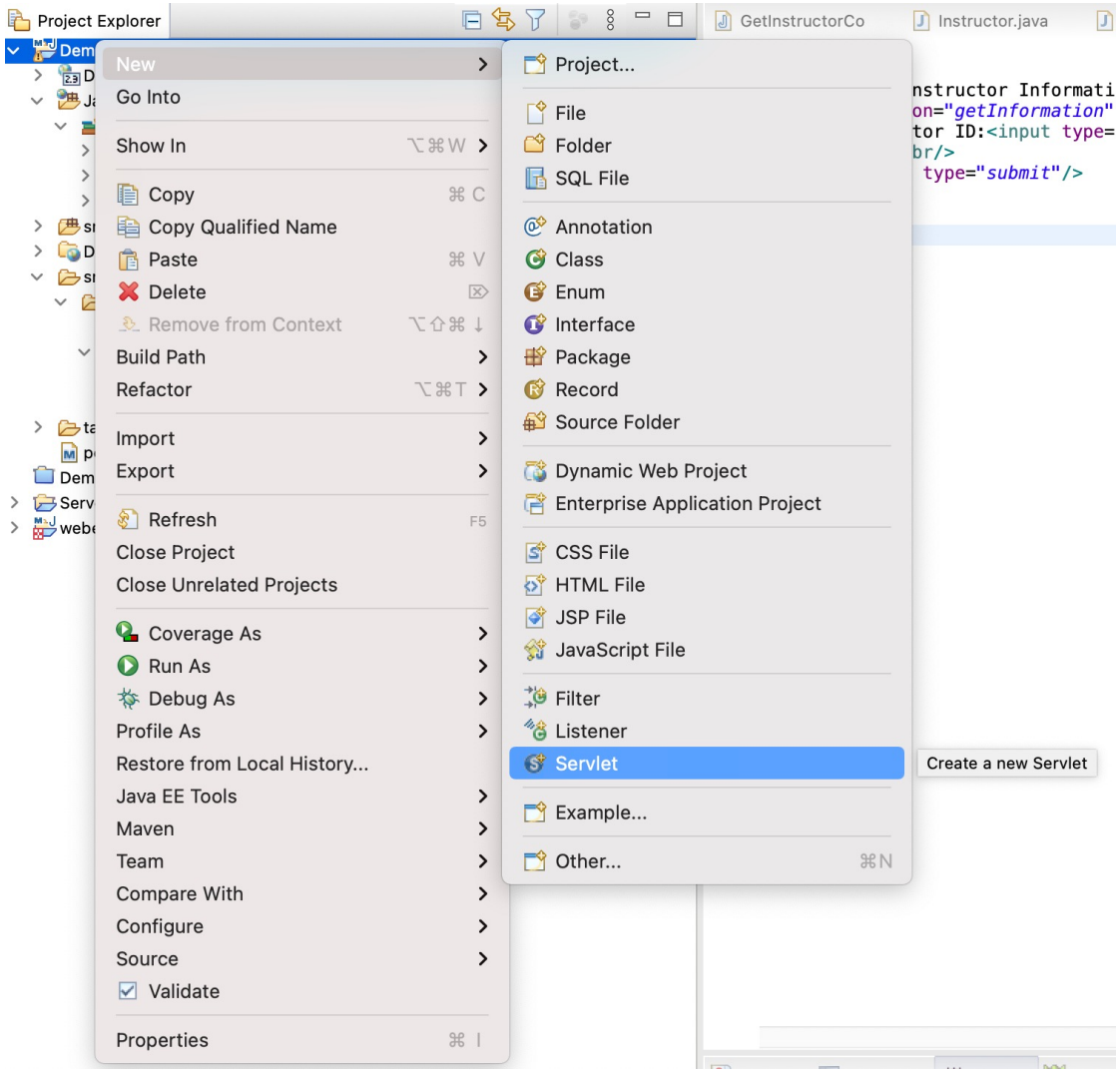
Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
- Add form to the **index.jsp** file

```
1  <html>
2  <body>
3   <h2>Retrieve Instructor Information!</h2>
4       <form action="getInformation" method="get">
5          Instructor ID:<input type="text" name="iID"/><br/><br/>
6           <br/><br/>
7           <input type="submit"/>
8   </form>
9   </body>
10  </html>
```

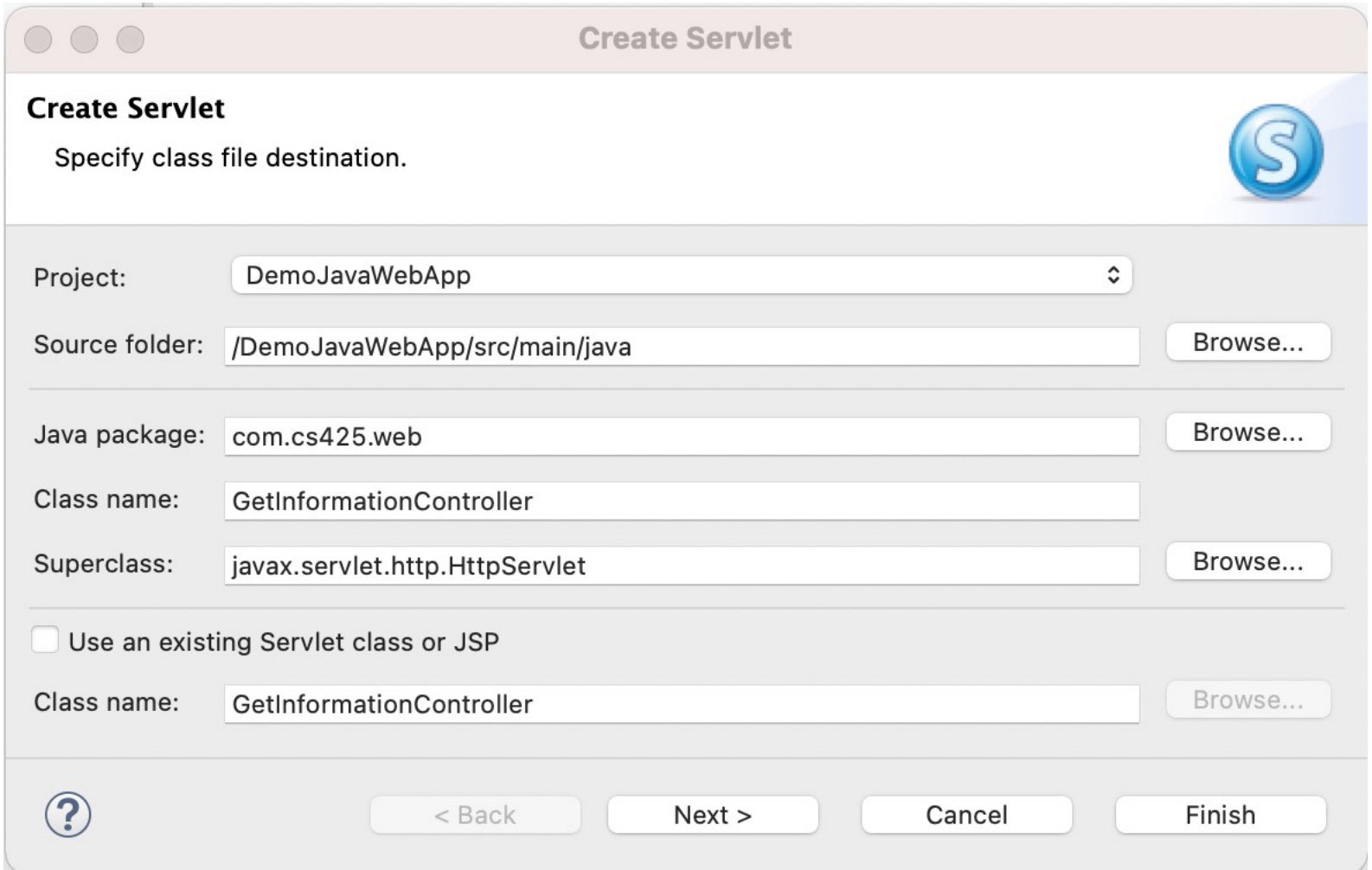
Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
- Create a **java** folder under **src>main** then create a new servlet



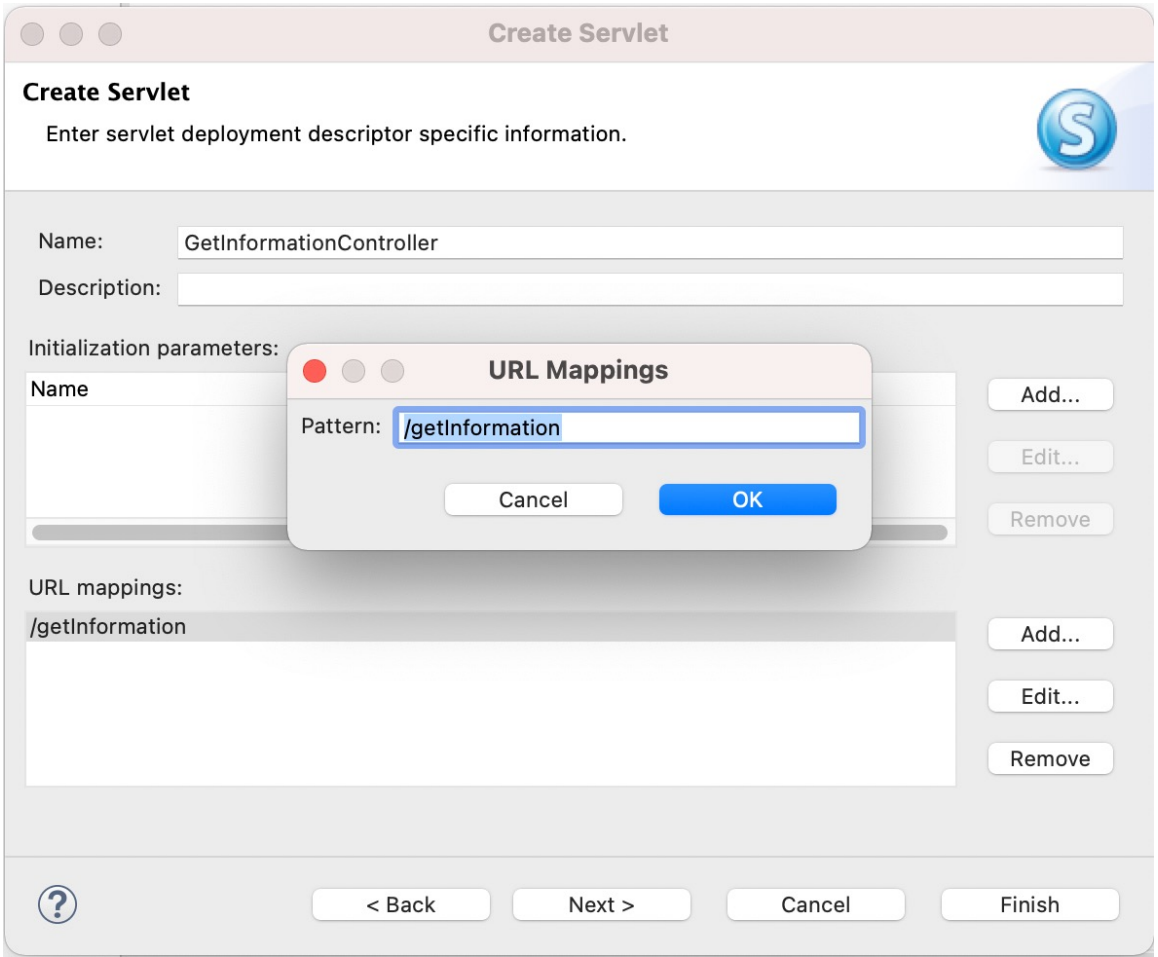
Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
  - Class name: **GetInformationController**
  - Java Package: **com.cs425.web**



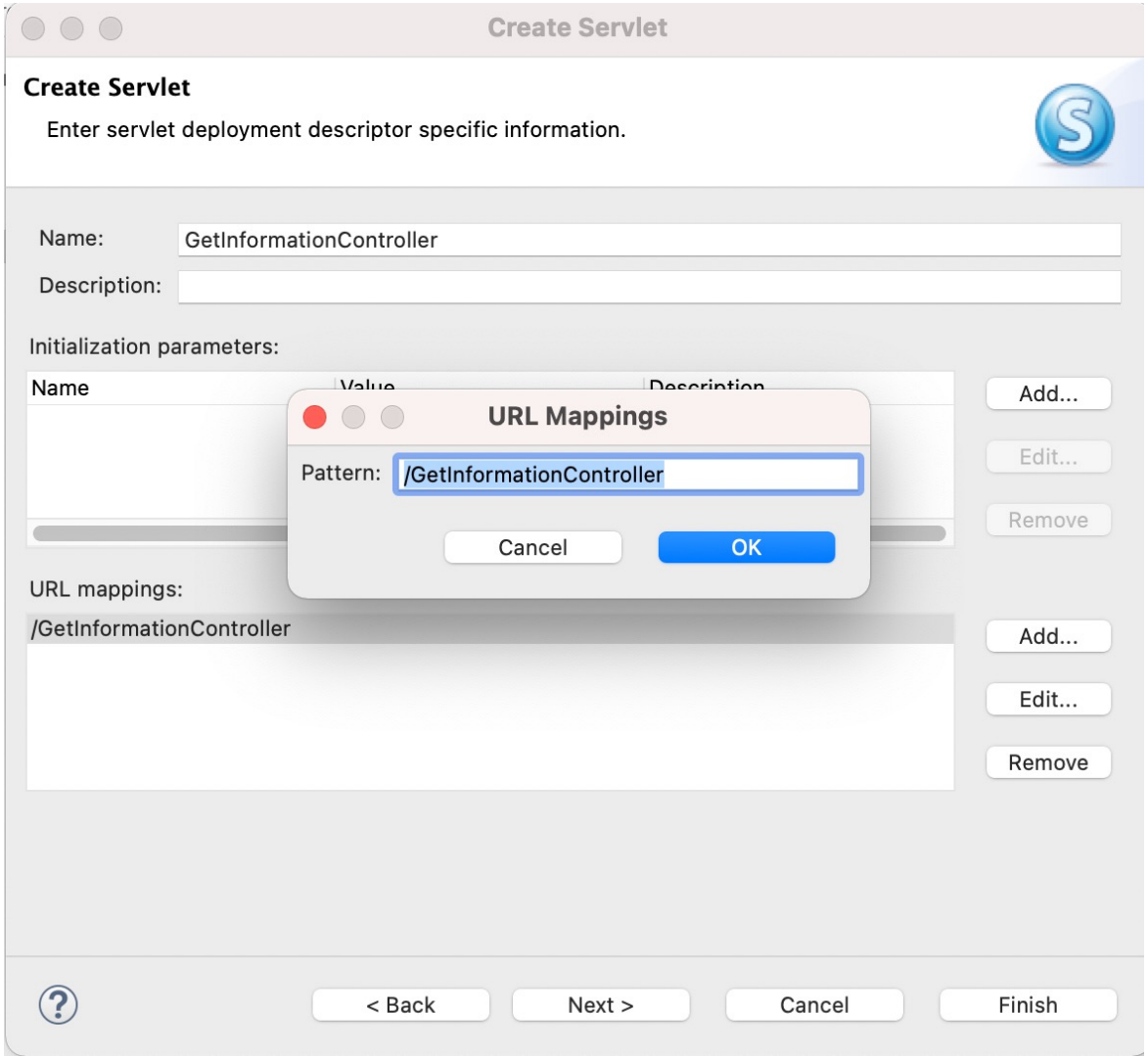
Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
  - Click Next, then click over the entry in URL mapping
  - Specify for which request you want to call this controller.
    - We want to use it for request type **getInformation**.



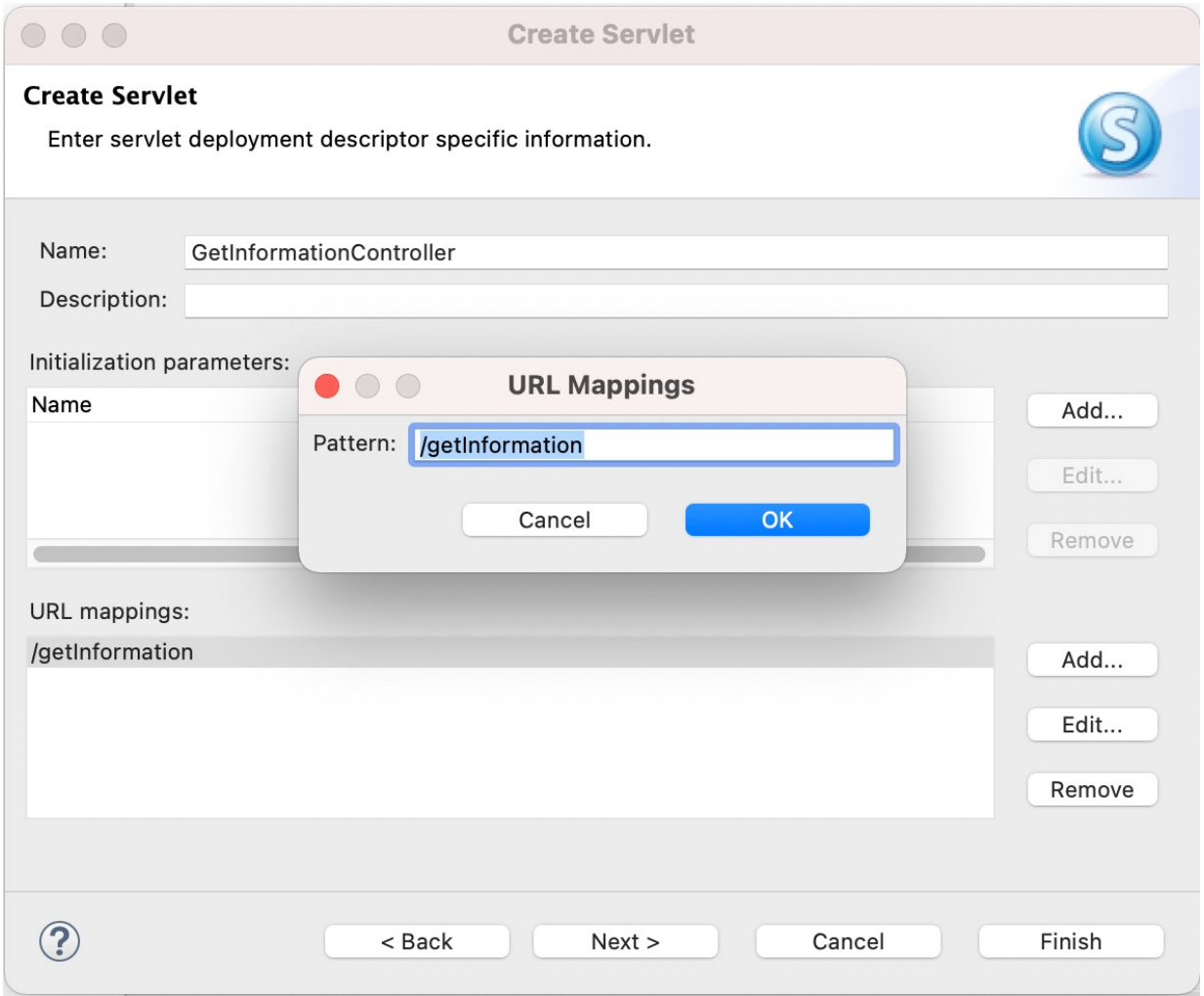
Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
  - Click Next, then click over the entry in URL mapping



Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
  - Specify for which request you want to call this controller.
    - We want to use it for request type **getInformation**. Then Ok, then Finish



Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
  - This will create a servlet as shown below

GetInformationController.java

```
1  package com.cs425.web;
2
3  import java.io.IOException;
8
9  /**
10  * Servlet implementation class GetInformationController
11  */
12 public class GetInformationController extends HttpServlet {
13     private static final long serialVersionUID = 1L;
14
15     /**
16      * @see HttpServlet#HttpServlet()
17      */
18     public GetInformationController() {
19         super();
20         // TODO Auto-generated constructor stub
21     }
22
23     /**
24      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
25      */
26     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws Se
27         // TODO Auto-generated method stub
28         response.getWriter().append("Served at: ").append(request.getContextPath());
29     }
30
31     /**
32      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
33      */
34     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws S
35         // TODO Auto-generated method stub
36         doGet(request, response);
37     }
38
39 }
40
```

Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
  - And the corresponding mapping is added to the **WEB-INF**

GetInformationController.java | web.xml

```
1  <!DOCTYPE web-app PUBLIC
2   "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
3   "http://java.sun.com/dtd/web-app_2_3.dtd" >
4
5  <web-app>
6    <display-name>Archetype Created Web Application</display-name>
7    <servlet>
8      <servlet-name>GetInformationController</servlet-name>
9      <display-name>GetInformationController</display-name>
10     <description></description>
11     <servlet-class>com.cs425.web.GetInformationController</servlet-class>
12   </servlet>
13   <servlet-mapping>
14     <servlet-name>GetInformationController</servlet-name>
15     <url-pattern>/getInformation</url-pattern>
16   </servlet-mapping>
17 </web-app>
```

Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
- Create Class **Instructor**

```java
package com.cs425.web.model;

public class Instructor {

    private String ID;
    private String name;
    private String dept_name;
    private double salary;


    public String getID() {
        return ID;
    }
    public void setID(String iD) {
        ID = iD;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDept_name() {
        return dept_name;
    }
    public void setDept_name(String dept_name) {
        this.dept_name = dept_name;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "Instructor [ID=" + ID + ", name=" + name + ", dept_name=" + dept_name + ", s
    }


}
```

Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
- The Data Access Object (DAO) pattern is a structural pattern that allows us to isolate the application/business layer from the persistence layer (usually a relational database, but it could be any other persistence mechanism) using an abstract API.

- DAO is an abstraction for accessing data, the idea is to separate the technical details of data access from the rest of the application

Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
- Create Class **InstructorDao**

```java
package com.cs425.web.dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import com.cs425.web.model.Instructor;

public class InstructorDao {

    final String url = "jdbc:postgresql://localhost:5432/university";
    final String user = "postgres";
    final String password = "1234";//"<add your password>";

    public Instructor getInstructor(String iID) {



        String mySQL = "SELECT id, name, dept_name, salary "
                + "FROM instructor "
                + "WHERE id = ?";

        Instructor ob1 = new Instructor();

        try {
            Class.forName("org.postgresql.Driver");
        } catch (ClassNotFoundException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        try (Connection conn = DriverManager.getConnection(url, user, password);
                PreparedStatement pStmt = conn.prepareStatement(mySQL)){

            pStmt.setString(1, iID.trim());

            ResultSet rs = pStmt.executeQuery();

            while (rs.next()) {
                /* Retrieves the value of the designated column in the current row
                   of this ResultSet object as a String in the Java programming language.
                */
                ob1.setID(rs.getString("id"));
                ob1.setName(rs.getString("name"));
                ob1.setDept_name(rs.getString("dept_name"));
                ob1.setSalary(rs.getDouble("salary"));
            }
        }catch (SQLException e) {
             System.out.println(e.getMessage());
        }

         return ob1;

    }
}
```

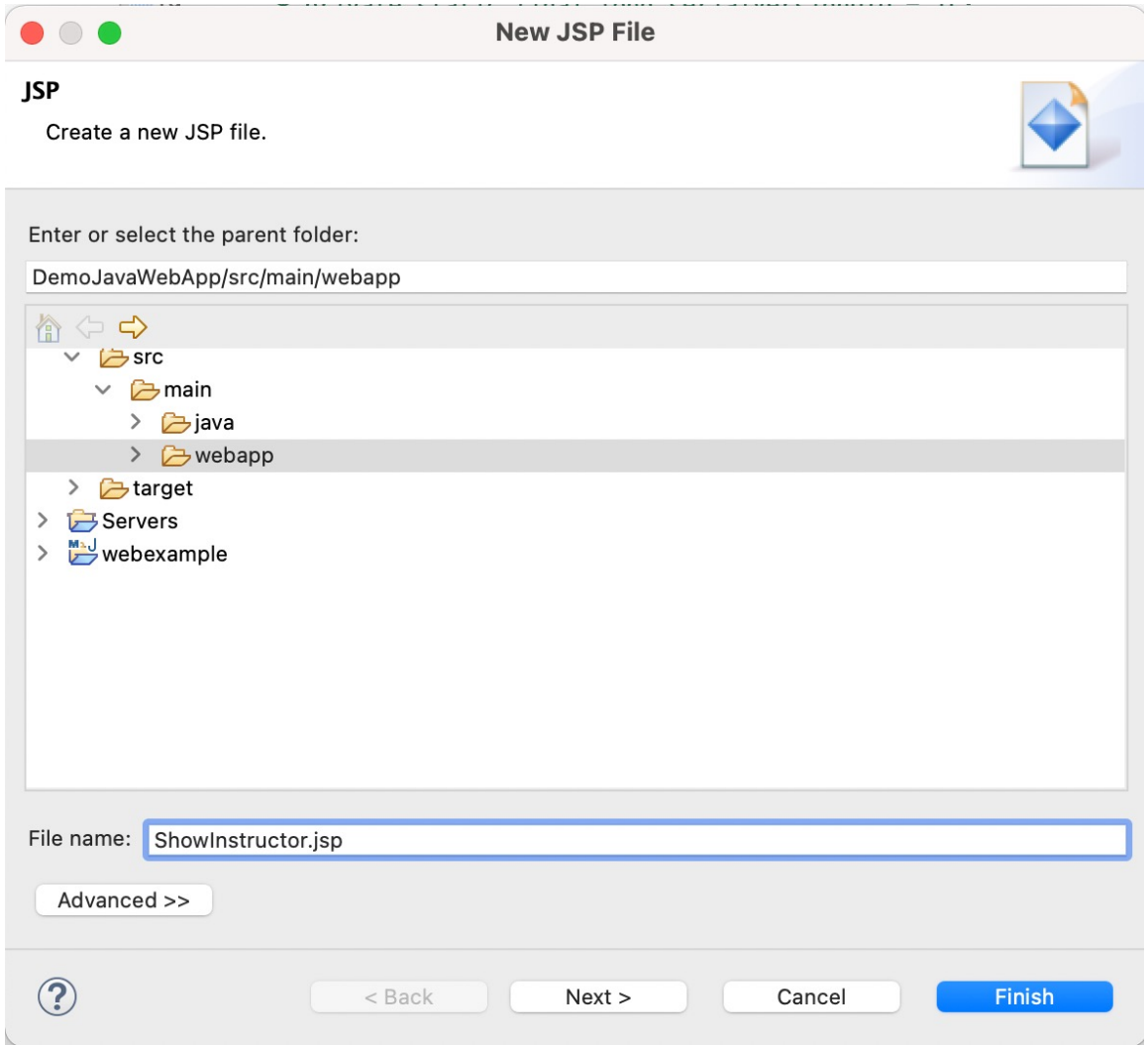
Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
- Modify the **GetInformationController** servlet

```
public class GetInformationController extends HttpServlet {
    /*
     * private static final long serialVersionUID = 1L;
     *
     *//**
         * @see HttpServlet#HttpServlet()
         *//*
             * public GetInformationController() { super(); // TODO Auto-generated
             * constructor stub }
             */

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
     *      response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // TODO Auto-generated method stub
        //response.getWriter().append("Served at: ").append(request.getContextPath());

        String ID = request.getParameter("iID").toString();
        //Servlet should only fetch and accept the request and no JDBC processing
        // Use Dao the take care of JDBC processing
        InstructorDao od1  =  new InstructorDao();

        Instructor ob1 = od1.getInstructor(ID);

        // To display the data, create a JSP file (e.g., ShowInstructor.jsp)


        // To send the Instructor object ob1 to JSP file. (e.g., ShowInstructor.jsp)
        request.setAttribute("Instructor", ob1);  // fetch this attribute in the JSP file



        // To call JSP page that will display the data either using dispatcher or send redir

        RequestDispatcher rd = request.getRequestDispatcher("ShowInstructor.jsp");
        rd.forward(request, response);


    }

    /*  *//**
             * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
             *      response)
             *//*
                 * protected void doPost(HttpServletRequest request, HttpServletResponse
                 * response) throws ServletException, IOException { // TODO Auto-generated
                 * method stub doGet(request, response); }
                 */

}
```

Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
- Create a **ShowInstructor.jsp** file to display the result



Create a Java web project in Eclipse using Maven

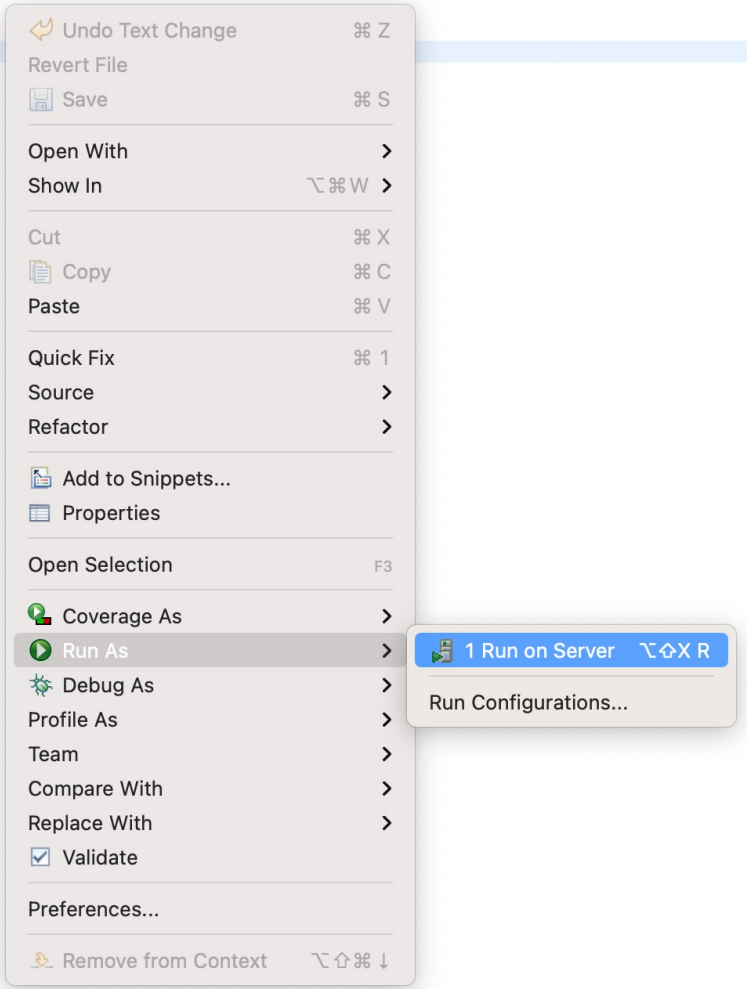
- Assume we want to retrieve a specific instructor information
- Create a **ShowInstructor.jsp** file to display the result

```
1  <%@ page import="com.cs425.web.model.Instructor"%>
2  <%@ page language="java" contentType="text/html; charset=UTF-8"
3      pageEncoding="UTF-8"%>
4  <!DOCTYPE html>
5  <html>
6  <head>
7  <meta charset="UTF-8">
8  <title>Instructor Information</title>
9  </head>
10 <body>
11   <%
12     Instructor ob2 = (Instructor) request.getAttribute("Instructor");
13     out.println(ob2);
14   %>
15 </body>
16 </html>
```

Create a Java web project in Eclipse using Maven

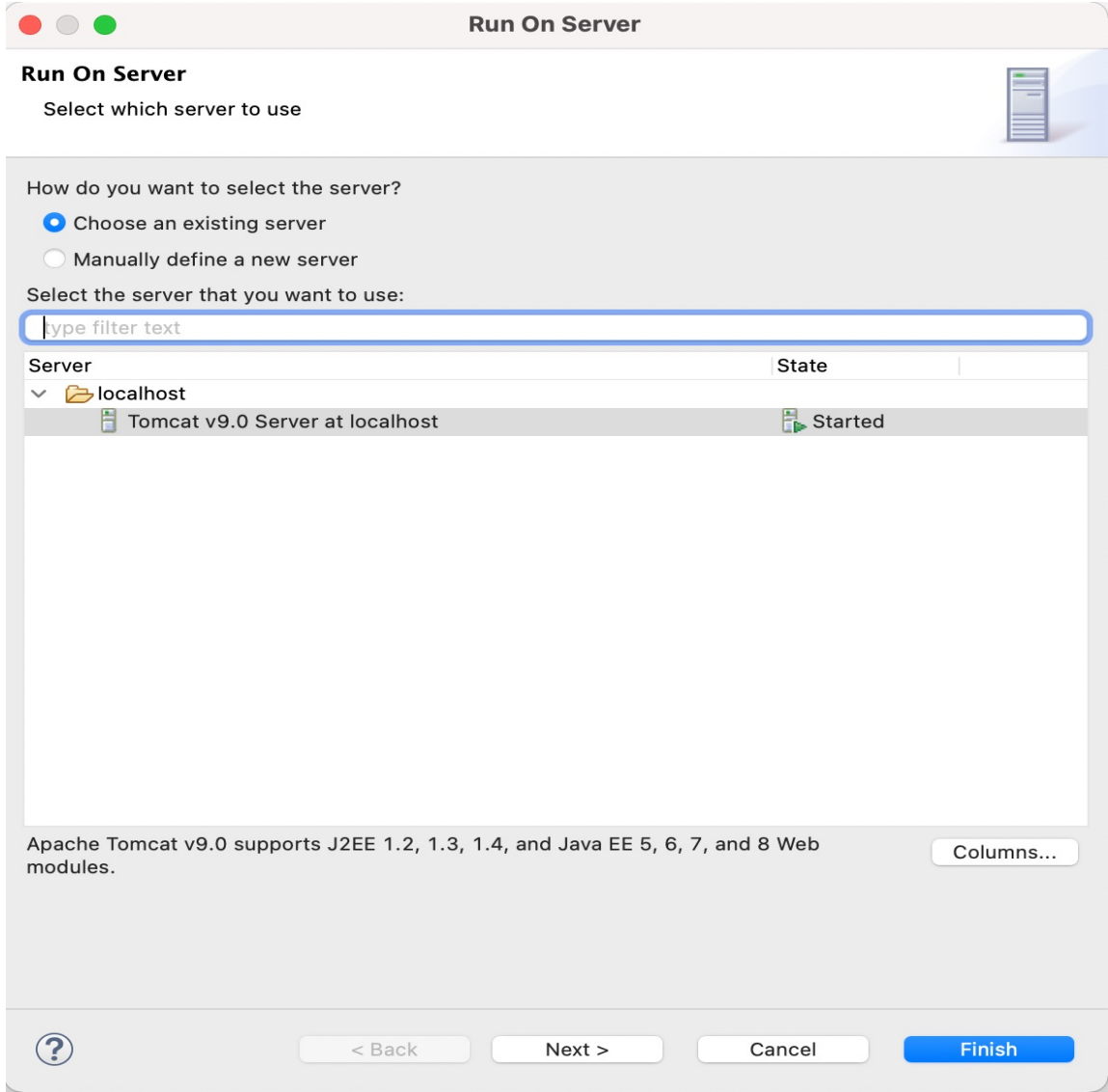
- Assume we want to retrieve a specific instructor information
- To run your project.
- Open the **index.jsp**

```
1  <html>
2  <body>
3  <h2>Retrieve Instructor Information!</h2>
4      <form action="getInformation" method="get">
5          Instructor ID:<input type="text" name="iID"/><br/><br/>
6           <br/><br/>
7           <input type="submit"/>
8  </form>
9  </body>
10 </html>
```



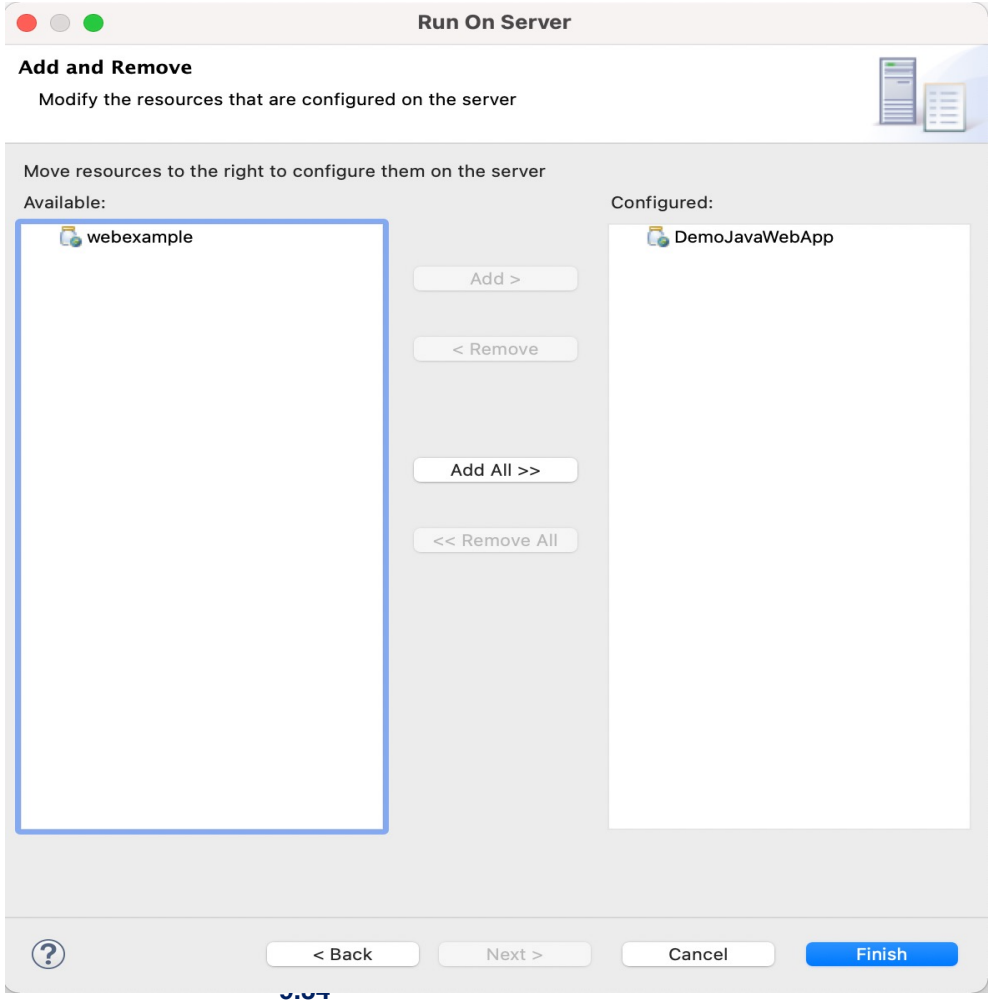
Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
- To run your project.
  - Click **Run on Server**



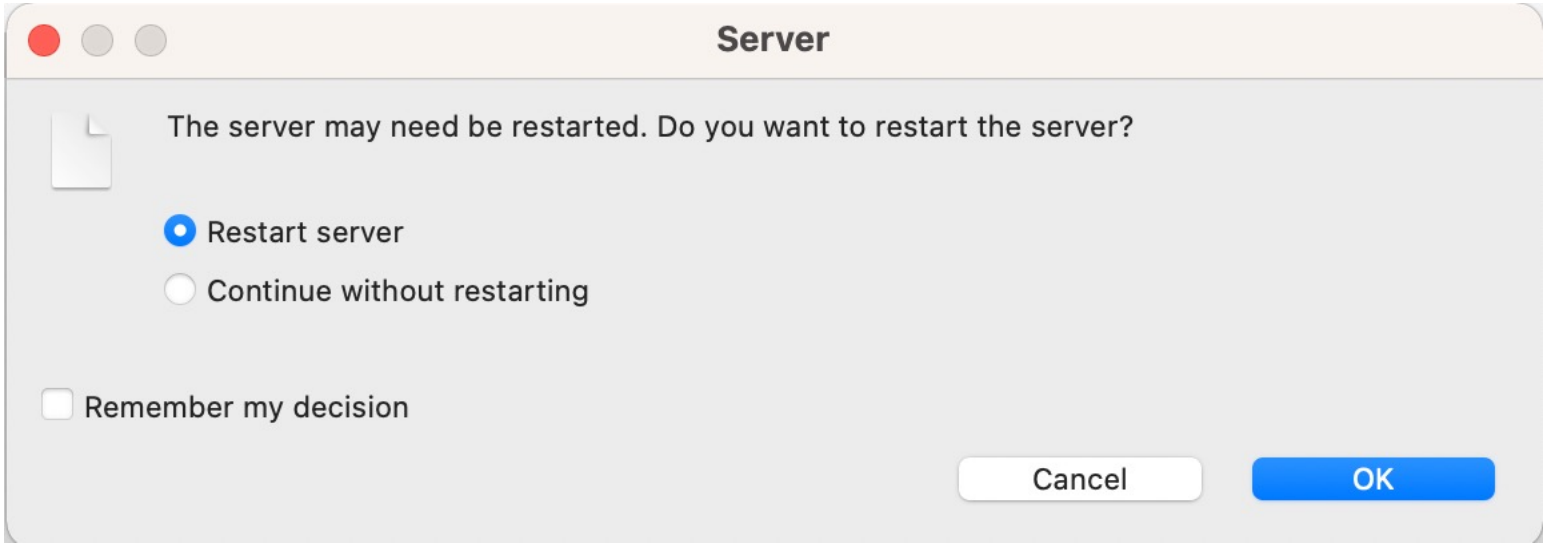
Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
- To run your project.
  - Make sure your project name is the one under the **Configured** tab.
  - Click **Finish**



Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
- To run your project.
  - Click **Ok**



Create a Java web project in Eclipse using Maven

- Assume we want to retrieve a specific instructor information
- To run your project.
  - Type any valid instructor ID, e.g., 63395, 78699, 37687. Click **Submit**

  - Result