

## CS 230 Dropwizard Hello World Tutorial

**Note:** Images in this tutorial may vary depending on Eclipse versions and may not be an exact match.

**Overview:** For this assignment, you will create a Hello World app using Maven and Dropwizard.

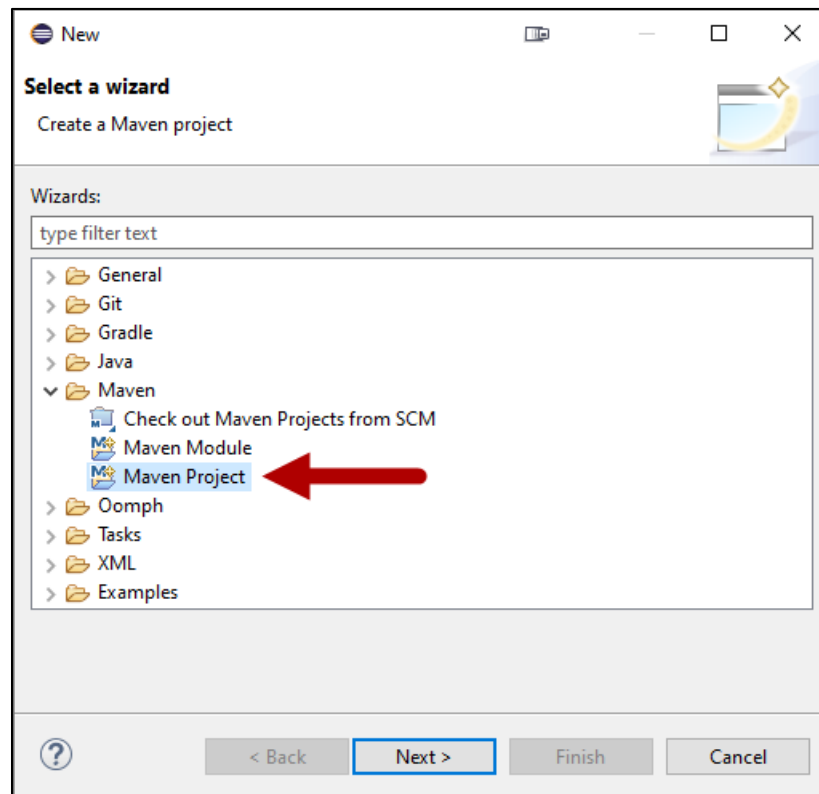
Dropwizard is a framework for implementing a distributed application. It uses RESTful web services and Java to develop production-ready web applications. This tutorial guides you through creating a Maven project using a Dropwizard archetype, adding source code, then compiling and running the Hello World application on a local server.

Most programming languages and frameworks begin with the typical "Hello World" example, and Dropwizard is no exception. This example will help you create a working prototype application. Follow the directions below to create a Hello World program in the browser using Dropwizard.

**Note:** Ensure you are connected to the internet as you complete this assignment, as specific libraries will need to be downloaded from the Eclipse Cloud Repository as you complete steps (this happens automatically).

### Creating the Maven Project and Source Code

1. In Eclipse, create a Maven Project by selecting **File, New, Other, Maven, and Maven Project**, then click **Next**.



2. In the **New Maven Project** dialog box for **Select Project Name and Location**, check **Use the Default Workspace Location** and click **Next**.

New Maven Project

New Maven project

Select project name and location

☐ Create a simple project (skip archetype selection)

☒ Use default Workspace location

Location:  Browse...

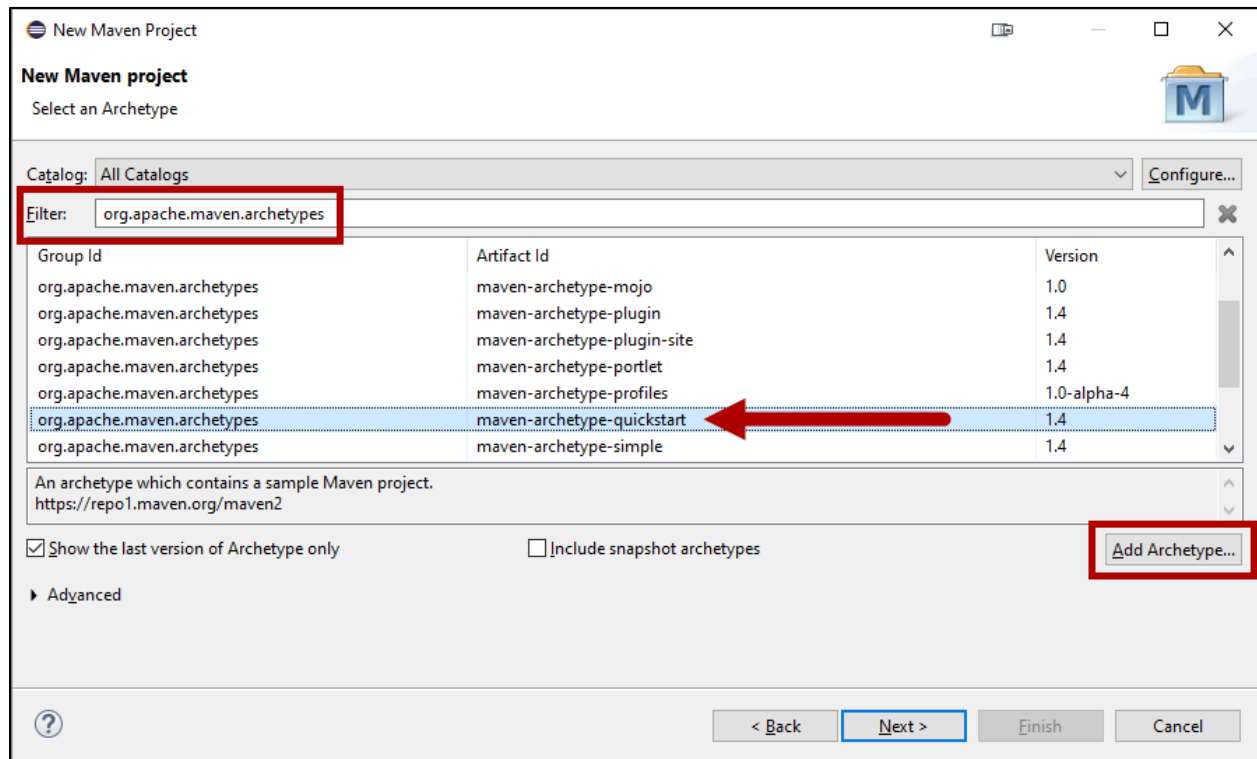
☐ Add project(s) to working set

Working set:  More...

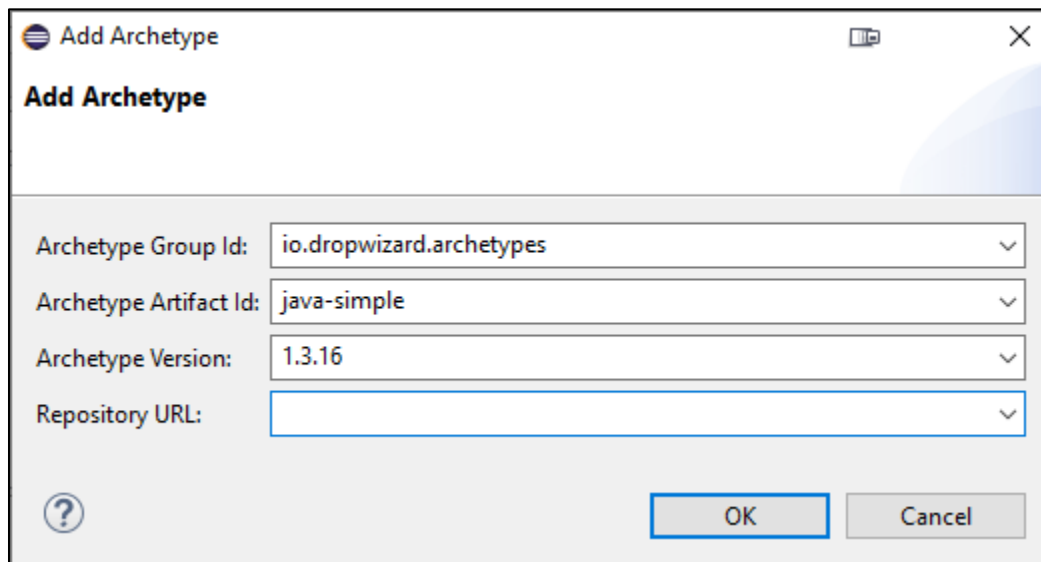
Advanced

< Back Next > Finish Cancel

3. In the **Archetype** filter box, search for **org.apache.maven.archetypes**. Under Artifact ID select **maven-archetype-quickstart**, then click the **Add Archetype** button.

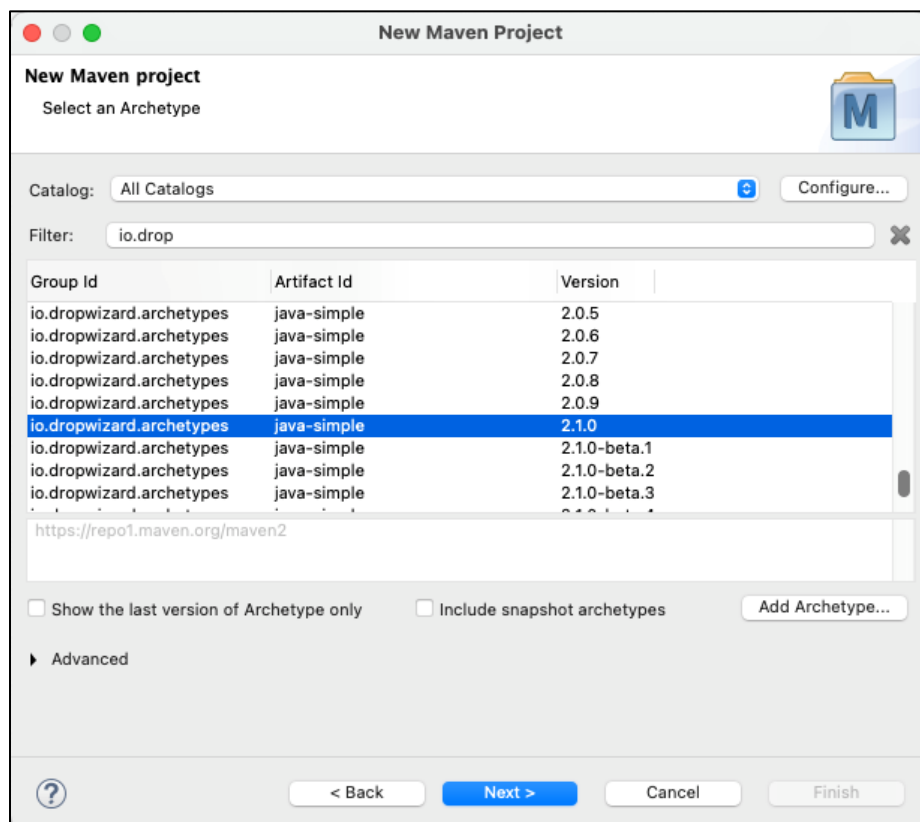
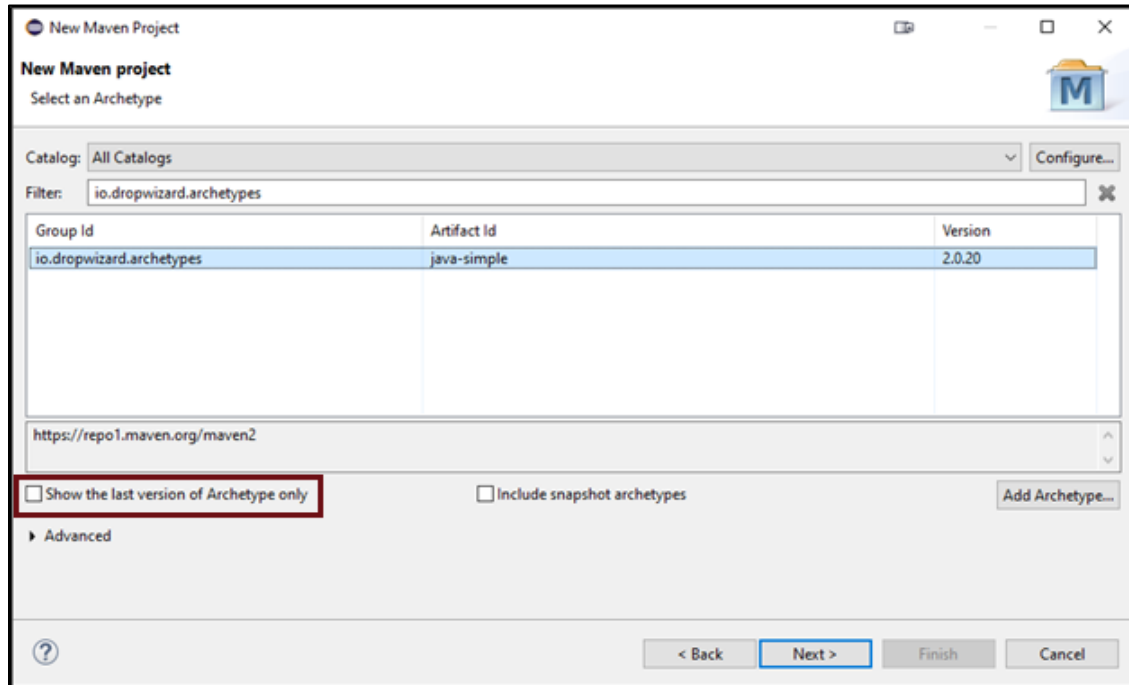


4. Type in the Archetype Group Id, Artifact Id, and the version, and click **OK**.



Search for, and select the archetype that you just created; deselect 'Show the last version of Archetype only' checkbox, and select the most recent non-beta version:

5. Click **Next**.



6. Add the Group Id, Artifact Id, and project name value for the archetype and click **Finish** to create the Maven Dropwizard project.

New Maven Project

Specify Archetype parameters

Group Id: com.dropwizard

Artifact Id: helloworld

Version: 0.0.1-SNAPSHOT

Package: com.dropwizard.helloworld

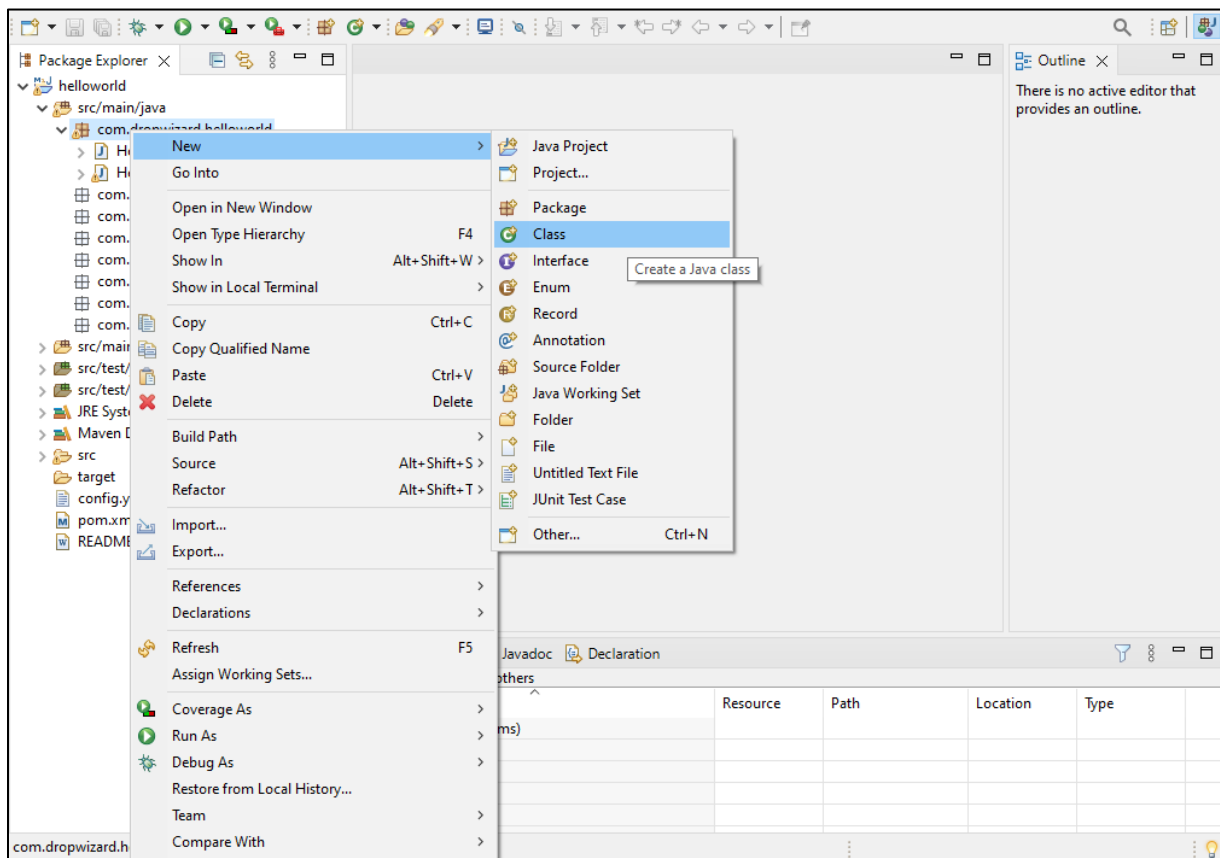
Properties available from archetype:

Name	Value
name	HelloWorld
description	null
shaded	true


Advanced

< Back Next > Finish Cancel

- Open the folder 'src/main/java', and create a class by right-clicking on the com.dropwizard.helloworld package created. Select **New**, then **Class**, and click **Enter**.



- In the **Java Class Dialog**, name the class "HelloWorldResource" and click **Finish**.

**Java Class**

Create a new Java class.

Source folder:

helloworld/src/main/java

Browse...

Package:

com.dropwizard.helloworld

Browse...

☐ Enclosing type:

Browse...

Name:

HelloWorldResource

Modifiers:

☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:

java.lang.Object

Browse...

Interfaces:

Add...


Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)  
☐ Constructors from superclass  
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☒ Generate comments



Finish

Cancel

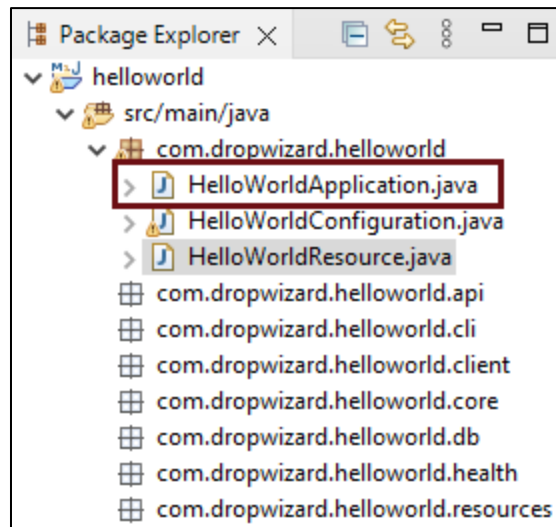
9. Type in the code for the **HelloWorldResource** class. Use the Path annotation to define the local host:8080 hello world path and the GET/Produces annotation to get and display the text in the browser.

```
package com.dropwizard.helloworld;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

/*
 * The @Path annotation provides the URL path for this API end-point.
 *
 *For example, if you are running this on your local machine with
 *at the TCP/IP port 8080, the @Path ("hello") annotation would
 *make this end-point available at http://localhost:8080/hello
 *
 *This annotation must appear immediately before the definition
 *of your class.
 */
@Path("hello")
public class HelloWorldResource {
    /*
     * This @GET annotation indicates that the end-point is
     * accessed via an HTTP GET operation. While the @Produces
     * annotation indicates that the data returned to the user
     * will be in plain text.
     *
     * These annotations should immediately proceed the method
     * definition.
     *
     * The getGreeting method will execute when the "hello"
     * end-point is accessed.
     */
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String getGreeting() {
        return "Hello World";
    }
}
```

10. Double click the **Application** JAVA file to open the file in the editor.



11. In the HelloWorldApplication JAVA file **and** the HelloWorldResource class, import the resource. This makes the class that you created in the previous step available to the HelloWorldApplication.

```
import com.dropwizard.helloworld.HelloWorldResource;
```

12. Register the HelloWorldResource class in the run method found in the HelloWorldApplication JAVA file.

```
@Override
public void run(final HelloWorldConfiguration configuration,
                final Environment environment) {

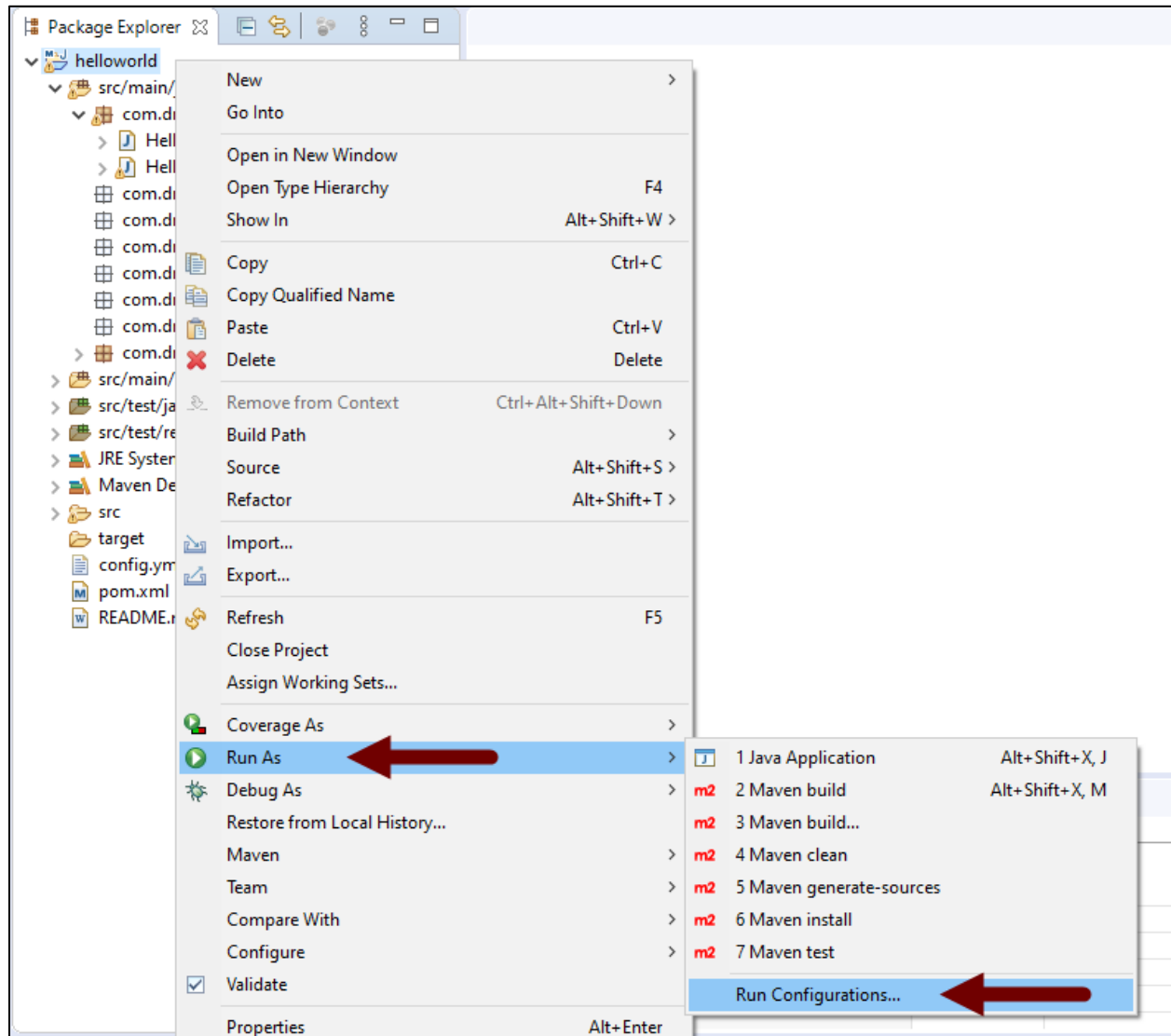
    /*
     * This section registers the HelloWorldResource, which is our
     * HTTP end-point with the application. Once this is complete, the
     * end-point will be active when the application server runs. You can
     * register more than one end-point in the environment, but it is a
     * best practice for each end-point to be developed in a separate class.
     */
    environment.jersey().register(
        new HelloWorldResource()
    );
}
```

13. Now that you have built the project and added the source code, it is time to build and run the project.

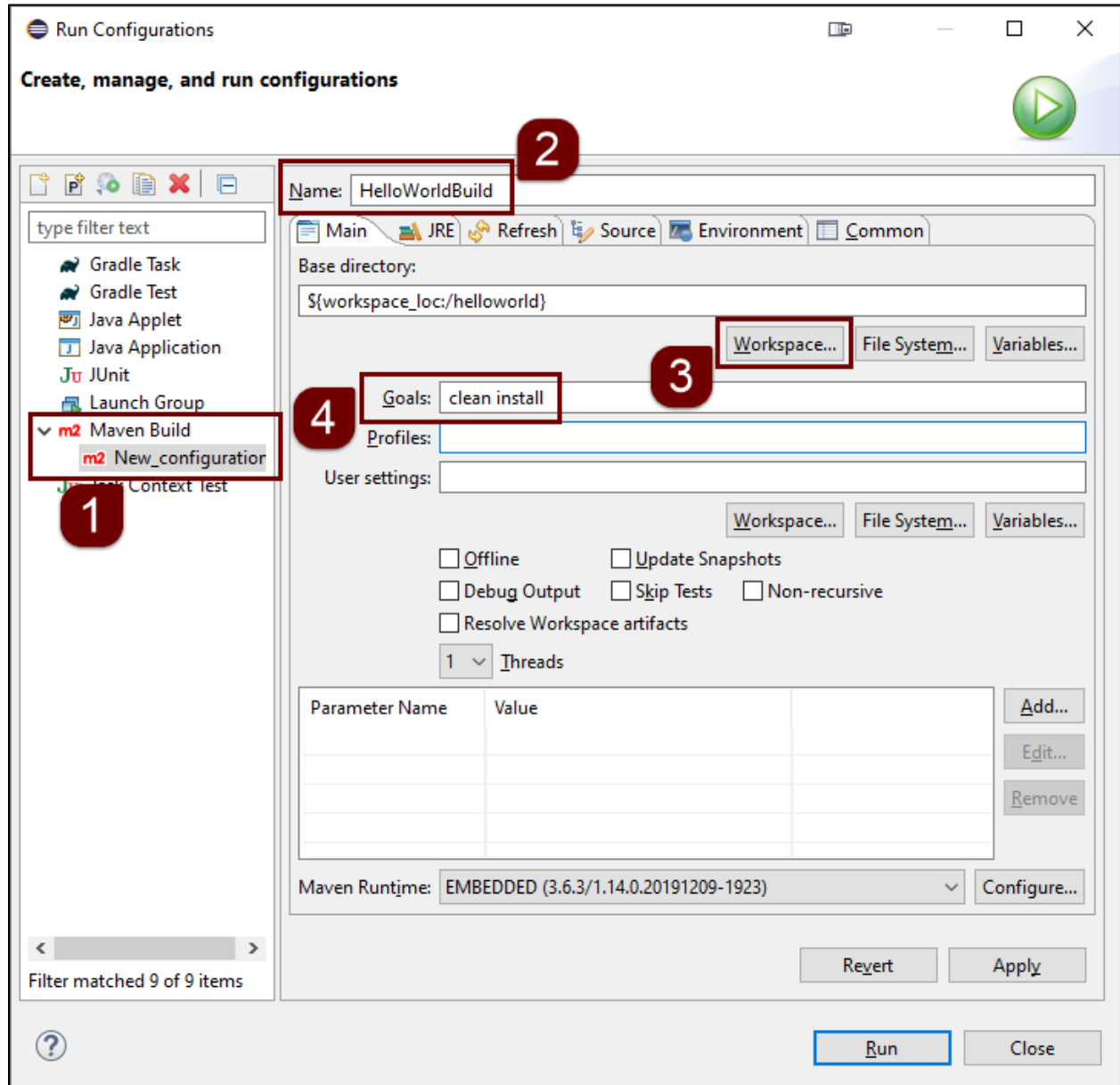


## Building and Running the Maven Project

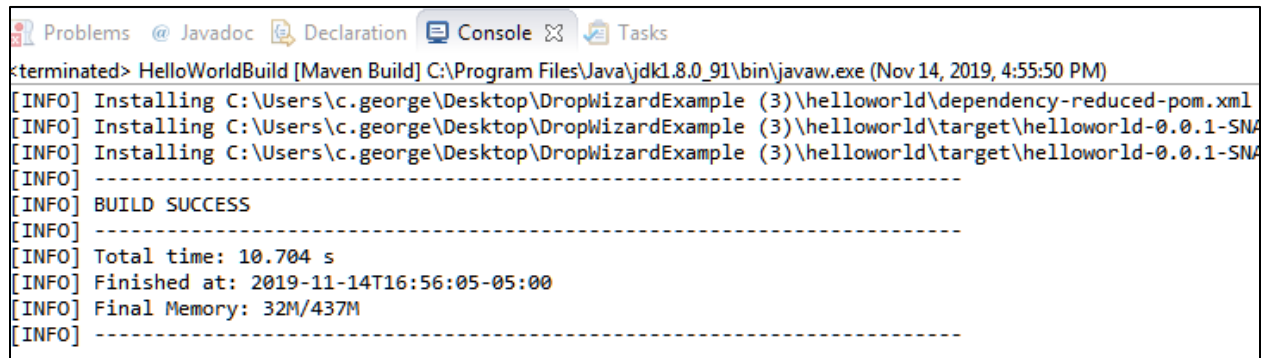
14. Create configuration to build the Maven project by right-clicking on the project name and selecting **Run As**, then **Run Configurations**.



15. See the numbering in the image below. Select **Maven Build** (#1) and click the **New** button to create a new Maven build launch configuration. Name the new configuration **HelloWorldBuild** (#2) and select the HelloWorld directory as the Base directory by selecting the **Workspace** button (#3). Add "clean install" to the Goals field (#4) and click on the **Run** button to compile the Maven project.

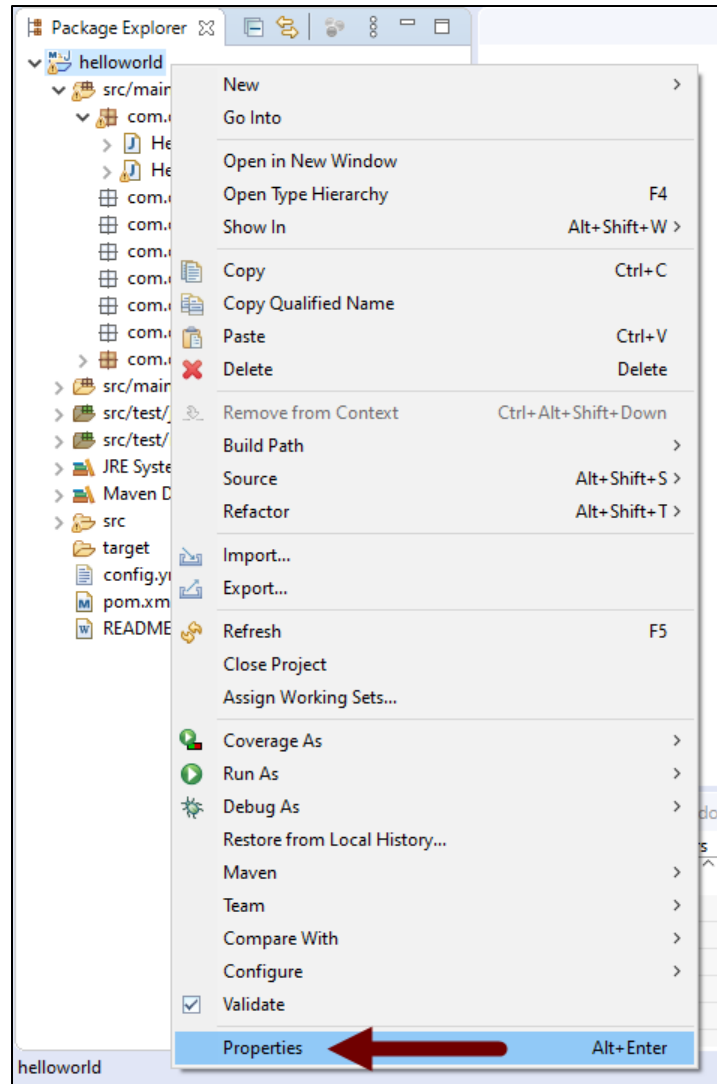


16. The project should now build successfully.

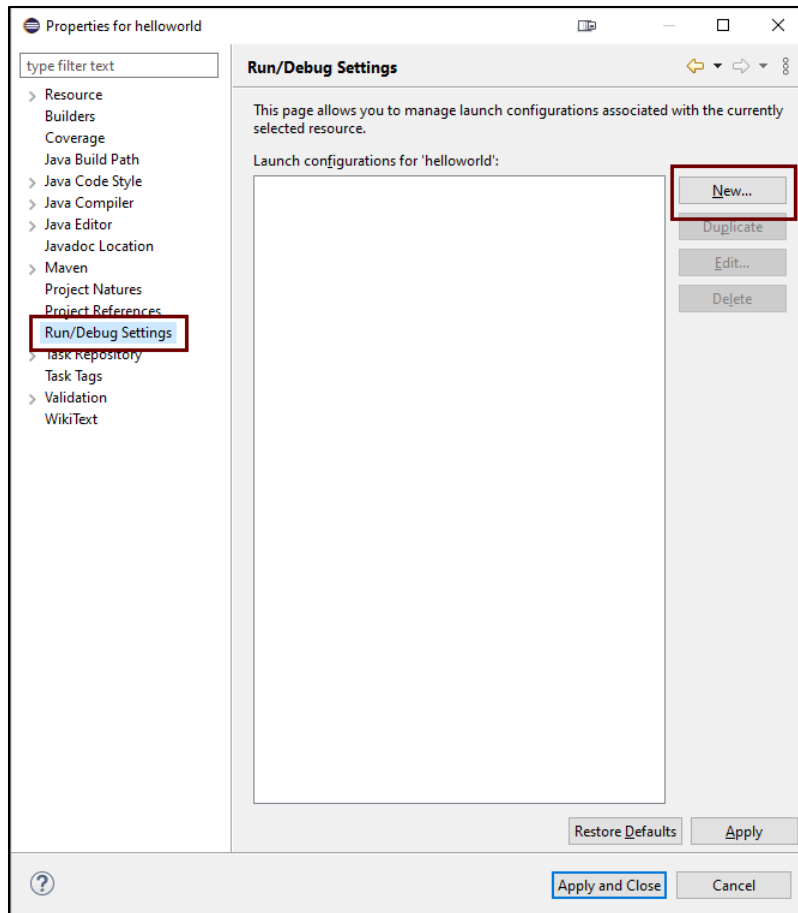


```
<terminated> HelloWorldBuild [Maven Build] C:\Program Files\Java\jdk1.8.0_91\bin\javaw.exe (Nov 14, 2019, 4:55:50 PM)
[INFO] Installing C:\Users\c.george\Desktop\DropWizardExample (3)\helloworld\dependency-reduced-pom.xml
[INFO] Installing C:\Users\c.george\Desktop\DropWizardExample (3)\helloworld\target\helloworld-0.0.1-SNAPSHOT.jar
[INFO] Installing C:\Users\c.george\Desktop\DropWizardExample (3)\helloworld\target\helloworld-0.0.1-SNAPSHOT.jar
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 10.704 s
[INFO] Finished at: 2019-11-14T16:56:05-05:00
[INFO] Final Memory: 32M/437M
[INFO]
```

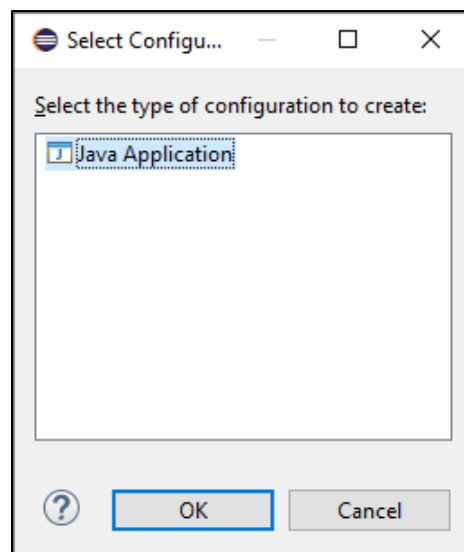
17. To run the HelloWorld app, build a configuration to run the app by first right-clicking on the project name and selecting **Properties**.



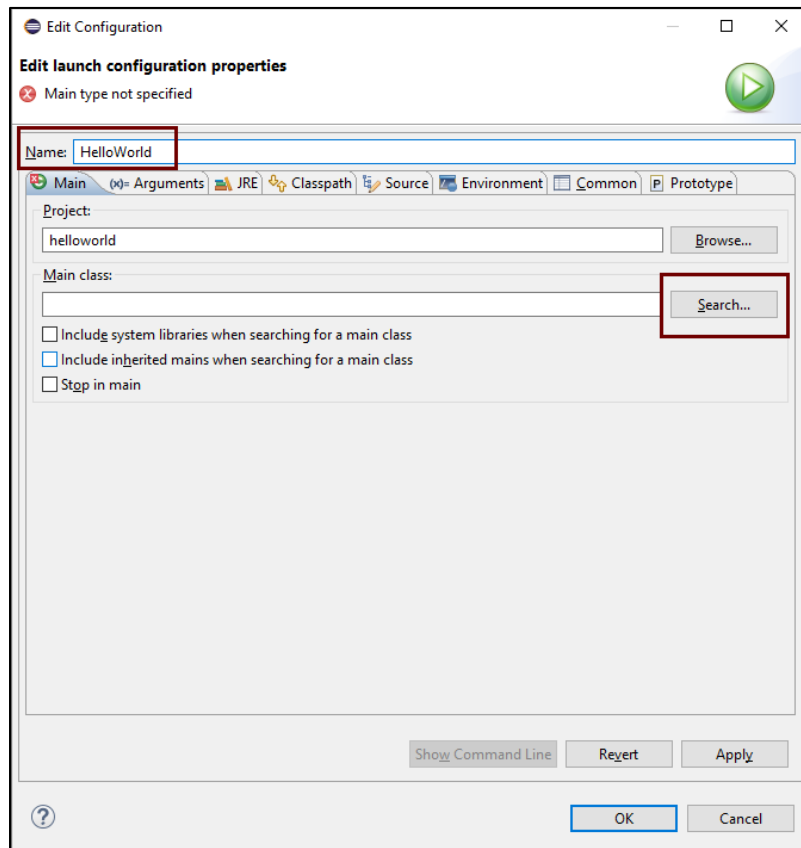
18. Select **Run/Debug Settings**, then click **New** to create a new run configuration.



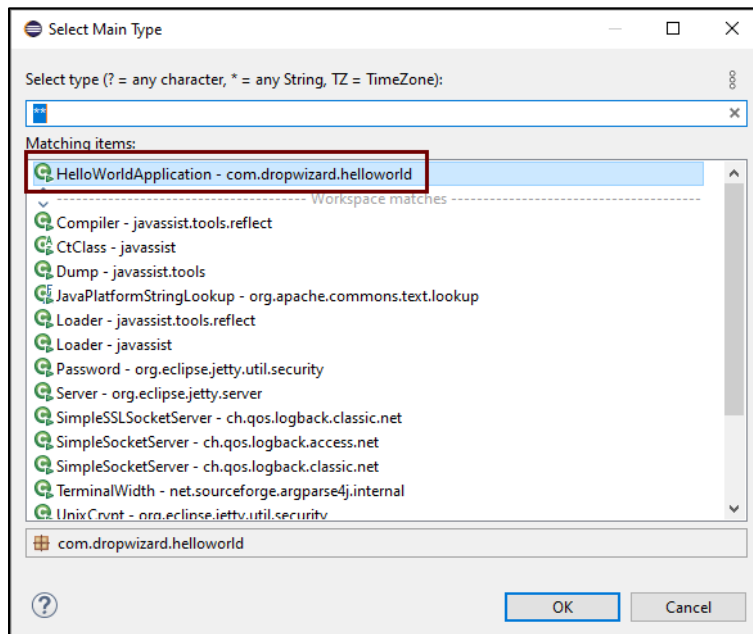
19. On the next window select **Java Application**, then click **OK**.



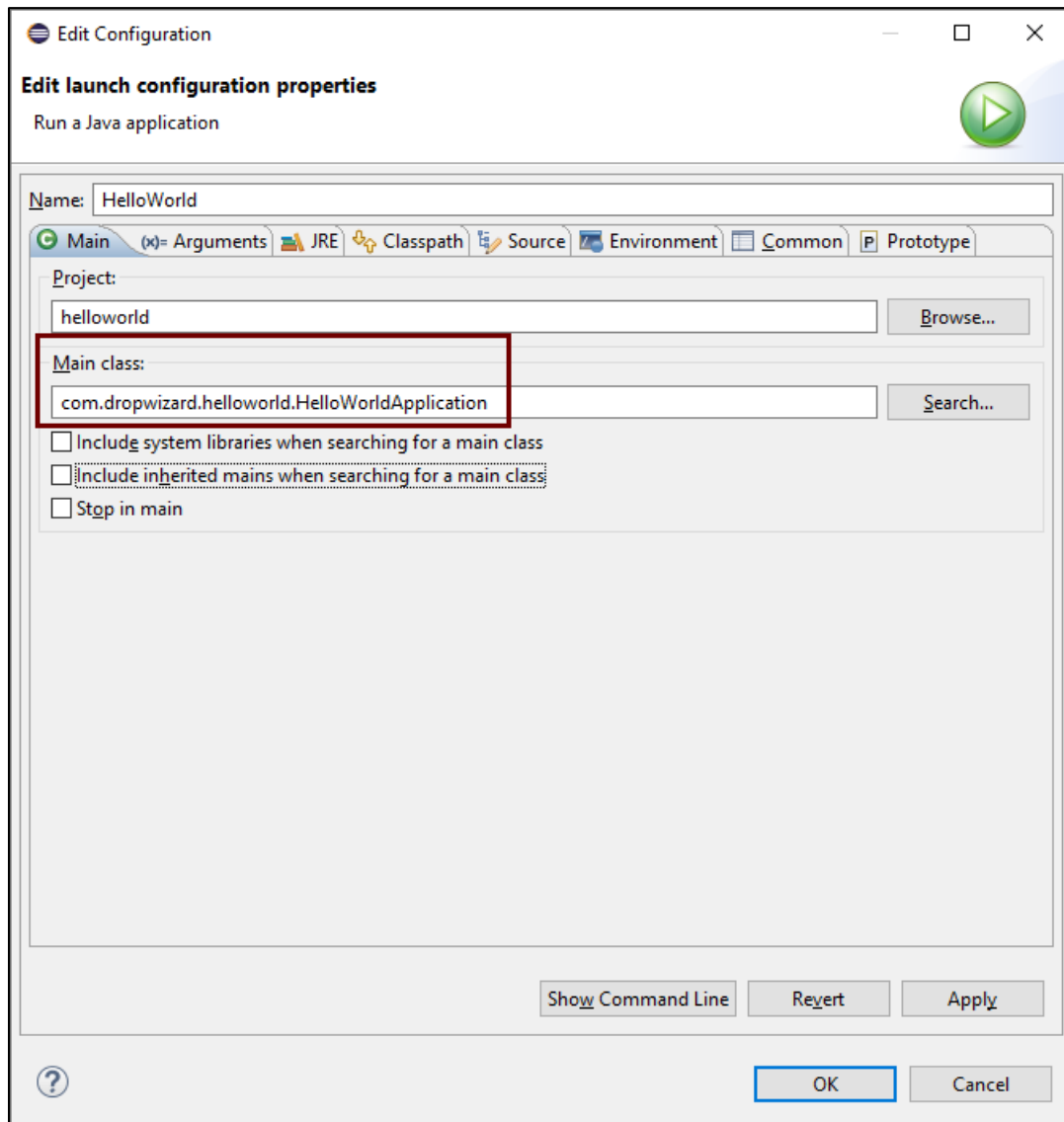
20. Name the new run configuration "HelloWorld". Click **Search...** to find the application package where the main class is located.



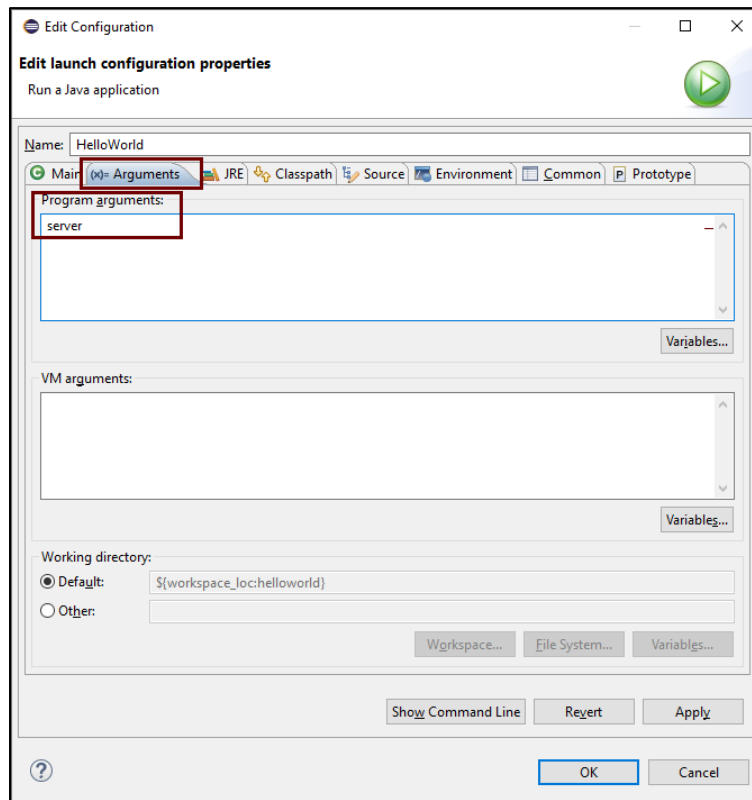
21. In the search window look for your project. Once found click on it, and click OK.



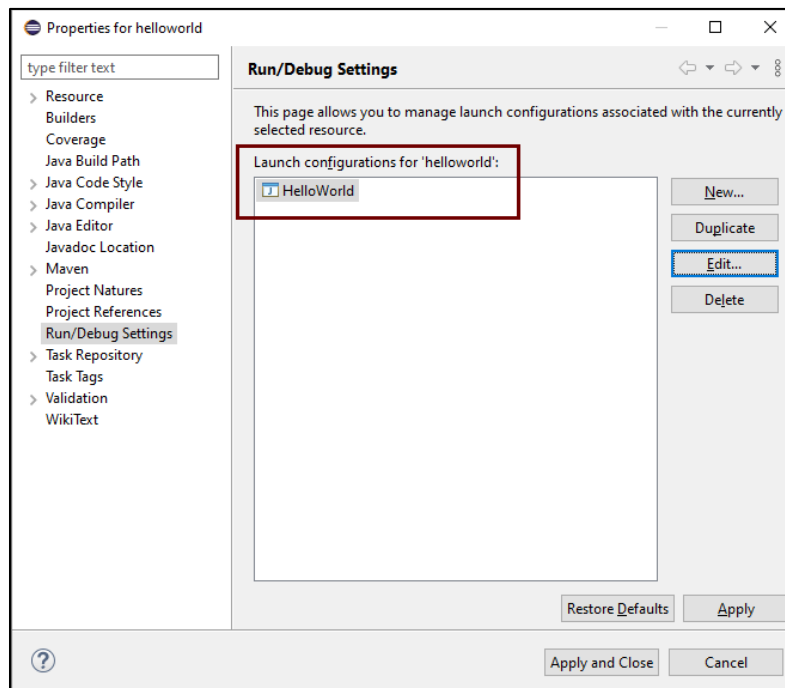
22. Back at the Edit Configuration window the **Main Class** line will now be populated.



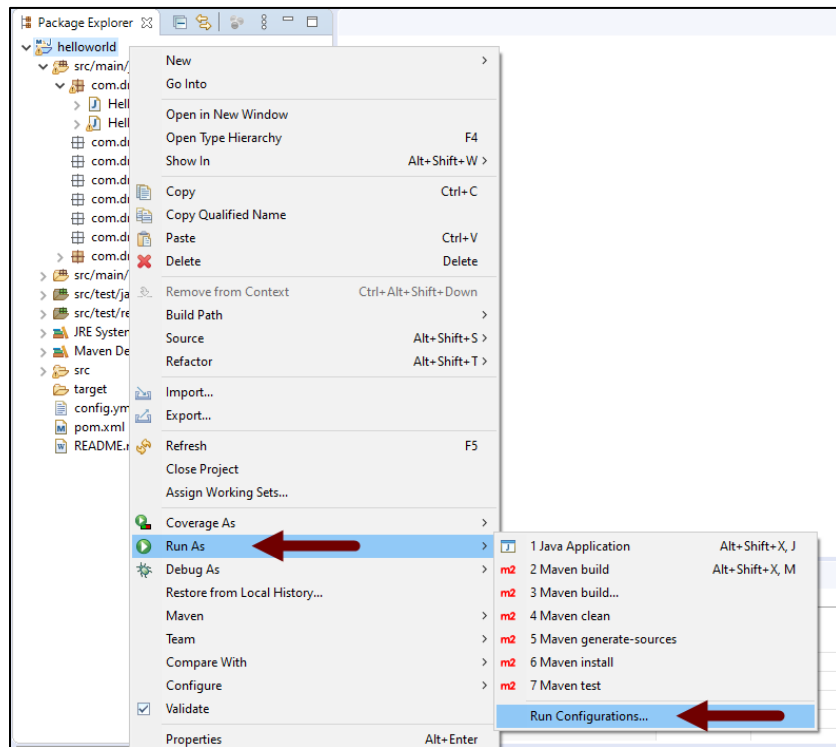
23. Click on the **Arguments** tab to add the **server** argument to build for a server environment. Click **OK**.



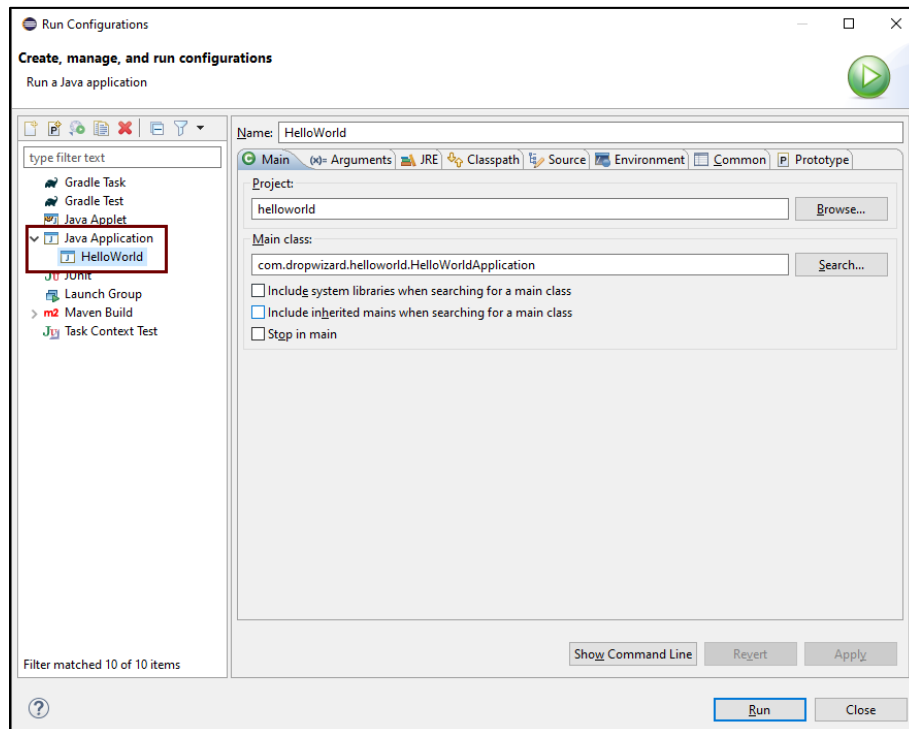
24. Back at the Properties window, you will now see your Launch Configurations for HelloWorld. Click **Apply and Close**.



25. To run the Maven project, go back to the main directory and right-click on the project name, **HelloWorld**, then select **Run As** and then select **Run Configurations**.



26. Under Java Application, select **HelloWorld**, then click **Run**.





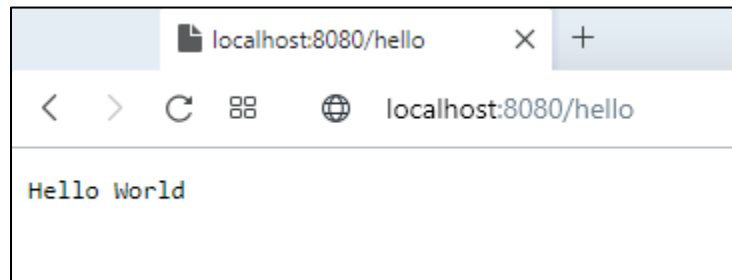
27. The following will be displayed under the console tab:

```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!   THIS APPLICATION HAS NO HEALTHCHECKS. THIS MEANS YOU WILL NEVER KNOW   !
!   IF IT DIES IN PRODUCTION, WHICH MEANS YOU WILL NEVER KNOW IF YOU'RE  !
!   LETTING YOUR USERS DOWN. YOU SHOULD ADD A HEALTHCHECK FOR EACH OF YOUR !
!   APPLICATION'S DEPENDENCIES WHICH FULLY (BUT LIGHTLY) TESTS IT.         !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
INFO [2021-04-13 19:58:37,335] org.eclipse.jetty.server.handler.ContextHandler: Started i.d.j.MutableServletContextHandler@d74bac4{/,null,AVAILABLE}
INFO [2021-04-13 19:58:37,348] org.eclipse.jetty.server.AbstractConnector: Started application@clcfcale{HTTP/1.1, (http/1.1)}{0.0.0.0:8080}
INFO [2021-04-13 19:58:37,351] org.eclipse.jetty.server.AbstractConnector: Started admin@241a53ef{HTTP/1.1, (http/1.1)}{0.0.0.0:8081}
INFO [2021-04-13 19:58:37,351] org.eclipse.jetty.server.Server: Started @2579ms

```

28. Open a browser and type in **localhost:8080/hello**. The following will be displayed:



29. Congratulations! You have successfully completed the Dropwizard HelloWorld Tutorial.