What you will need

- A text editor
- JDK

Step 1: Installing a servlet container

To run our servlet, we don't necessarily need a full-blown Jakarta EE implementation; a servlet container is enough. Luckily, there are many excellent choices out there that are free and open-source: Tomcat, Jetty, and Undertow to name a few.

For our purposes, we will be using Tomcat, which has a strong reputation in the Java community and has been widely adopted. You can grab it by visiting the downloads section of Apache Tomcat®. You will find several options there so chose the appropriate one depending on your OS and architecture.

Also, make sure you download version 10 or later, otherwise, you will have to replace the jakarta imports with javax in the code that follows. (this is because of a breaking change in Jakarta EE 9). Lastly, notice where Tomcat is located in your file system.

Step 2: Setting up the application folder

Once you have installed Tomcat, you are ready to set up your folder structure. To save us some time, we will be creating our application directly in the folder where Tomcat expects to find the deployable. This is located in

<TOMCAT_HOME>/webapps

So navigate to the base directory of Tomcat, open the webapps folder and create a folder in it, named:

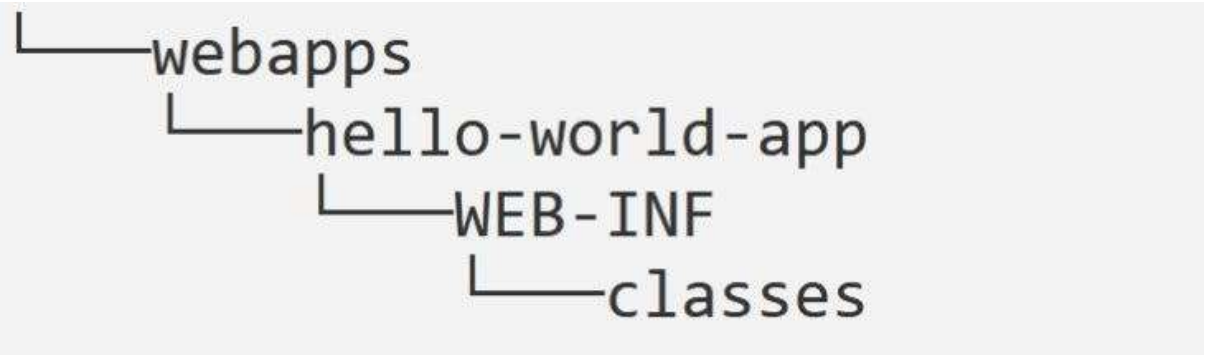
hello-world-app

This directory is where our servlet is going to be deployed and this folder name will later be part of the URL (the so-called Context Path). Note that this naming correlation is not mandatory, but for our purposes, we shall go with that. Now, within hello-world-app, we need to create a directory called:

WEB-INF

It is important to use exactly this name, as Tomcat expects to find our servlet in the /WEB-INF/classes/ folder. Therefore, within the WEB-INF folder create a folder called classes.

So, the result should be looking like this:



Step 3: Creating a servlet

Now get your boomer text editor up and running and create a file inside the classes folder, named HelloWorldServlet.java

Insert the following code, which is a "hello world" program:

```
import jakarta.servlet.annotation.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import java.io.*;

@WebServlet("/")
public class HelloWorldServlet extends HttpServlet {
   @Override
   public void doGet(HttpServletRequest request, HttpServletResponse response)
         throws IOException, ServletException {

      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<h2>Hello World!</h2>");
      out.close();
```

```
    }
}
```

In a nutshell, we are overriding the doGet method to respond to HTTP GET requests and the @WebServlet annotation maps the servlet to the application root ("/").

The println statement contains the response body.

Keep in mind that a properly formatted HTML should contain <html> <head> and <body> tags, but here we would like to keep things sweet and short. (most modern browsers will render the content anyway)

Step 4: Compiling a Servlet

Now our goal is to compile this servlet into a .class file that can be deployed. Did you notice those jakarta imports in the code above? Those packages are not in the JDK but in the servlet container. This means that we have to inform the Java compiler where to look for those, with the **-cp** or the **-classpath** argument.

```
javac -cp "<TOMCAT_HOME>/lib/servlet-api.jar" HelloWorldServlet.java
```

Don't forget to replace <TOMCAT_HOME> with Tomcat's directory! After executing this command and if everything goes well, the compiler will silently create a HelloWorldServlet.class in your classes directory.

Step 5: Deploying with Tomcat

We are almost done! All we need to do now is start the server by executing

C:\<TOMCAT_HOME>\bin\startup.bat (Windows)

or

sh <TOMCAT_HOME>/bin/startup.sh (Linux)

Let's try to figure out the URL of our servlet. Since the server is running on our machine, the first part of the URL can be localhost. Tomcat runs by default on port 8080. The Context Path of our application is hello-world-app and our servlet responds to what we defined in the @WebServlet annotation, so "/".

Put all those things together, and you have the URL which will trigger our servlet:

localhost:8080/hello-world-app

Try to enter it in a browser. Did you see "Hello World"? If yes, awesome! If not, let me know in the comments what went wrong and maybe I can help you out (I said maybe).