Servlet

Spring Web MVC is built on Servlet API and has been included in the Spring framework from the very beginning.

Consider the following architecture,

Client server [REMOTE]

Browser Tomcat

/basedir

/app1/WEB-INF/web.xml

/app1/WEB-INF/classes/servlet/Servlet.class

To run an Java EE application, we need a container. Apart from that to actually use the application we also need a basedir(default /webapps) & portnumber (/conf/server.xml), if it was done manually. Custom applications can be configured on the Context section.

Whenever we package an application through an IDE, the IDE behind-the-scenes would move the executable into the basedir for deployment.

About the file structure convention

A folder named WEB-INF is created in the ROOT of the application, within it we create web.xml file. Another folder WEB-INF/classes is created which contains the .class file (POJOs). To make the POJOs deployable we extend them by HttpServlet, which makes them suitable for Website deployment. All static html files, css files, script file are placed in ROOT folder of the application.

About configuring the application

We load the url of the format, <http://ip:8080/appname/url>, the url is configured in web.xml file, which maps the .class file with a /url.

About Servlet

HttpServlet is an abstract class which must override any of the GET, POST, PUT, DELETE methods to provide a service. These .class files are provided to the Tomcat (container) to be run.

HelloServlet.class -> /hello.do

public class HelloServlet extends HttpServlet{

public void doGet(HttpServletRequest request, HttpServletResponse response){

response.setContentType(“text/html”);

PrintWriter out = response.getWriter();

String ip = request.getRemoteAddr();

out.println(“<HTML>”);

out.println(“<BODY>”);

out.println(“ip: ”+ip);

out.println(“</BODY>”);

out.println(“<HTML>”);

}

}

We would then configure the above servlet (map an alias) in the web.xml and also add the servlet-mapping to /url. Thus, whenever Tomcat receives a request on that URL, tomcat would internally navigate the request to our defined doGet() via using web.xml.

Request -> service (GenericServlet) -> find out if doGet() or corresponding search method -> HelloServlet.class -> doGet() -> output sent

About Java

Sun & Oracle released Java EE from version 1 to 8, whereas Java version 8 and onwards were released by Eclipse foundation and is known as Jakarta EE. Tomcat 9 & 10 are compatible with different Java versions.

Tomcat 9 – only compatible with Oracle versions

Tomcat 10 – only compatible with Jakarta EE versions

Accordingly refer to correct Java docs for reading about Servlet API, Java EE & Jakarta EE -

* *docs.oracle.com/javaee*
* *jakarta.ee*

Below is the package naming convention,

Java EE – import javax.servlet.http.HttpServlet;

Jakarta EE – import jakarta.servlet.http.HttpServlet;

Thus, if we want to migrate from Java EE to Jakarta EE these import statements need to be changed and appropriate APIs. Alternatively, we could also use a third-party tool to migrate to Jakarta EE.