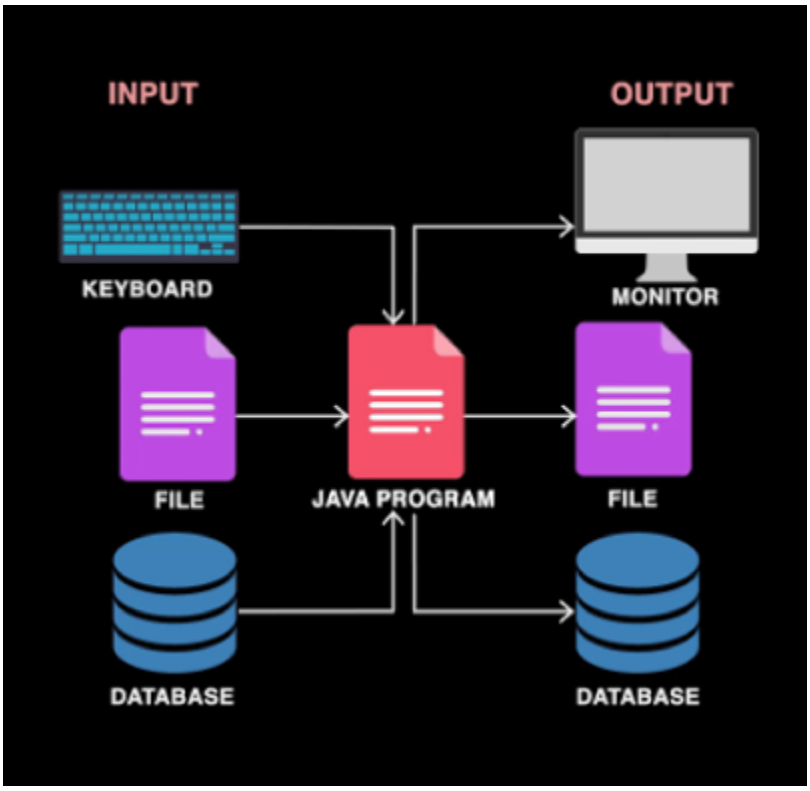
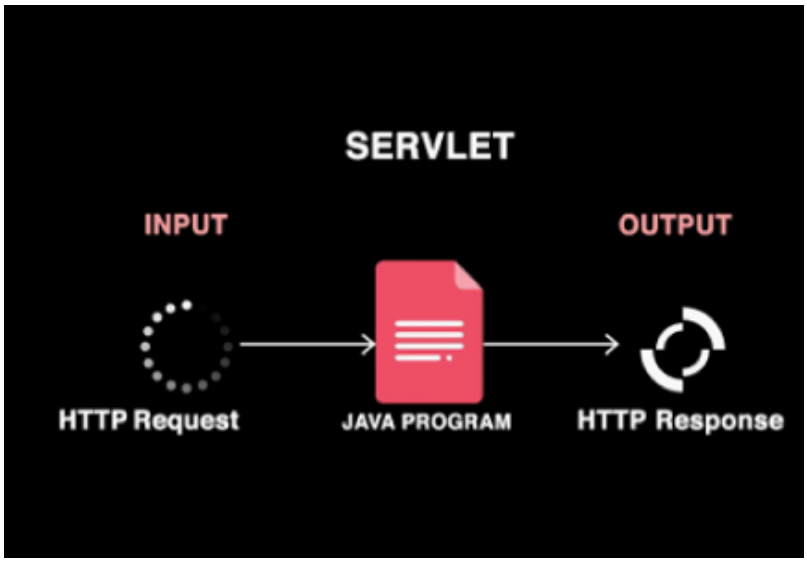
DAY - 2
Servlet Hierarchy

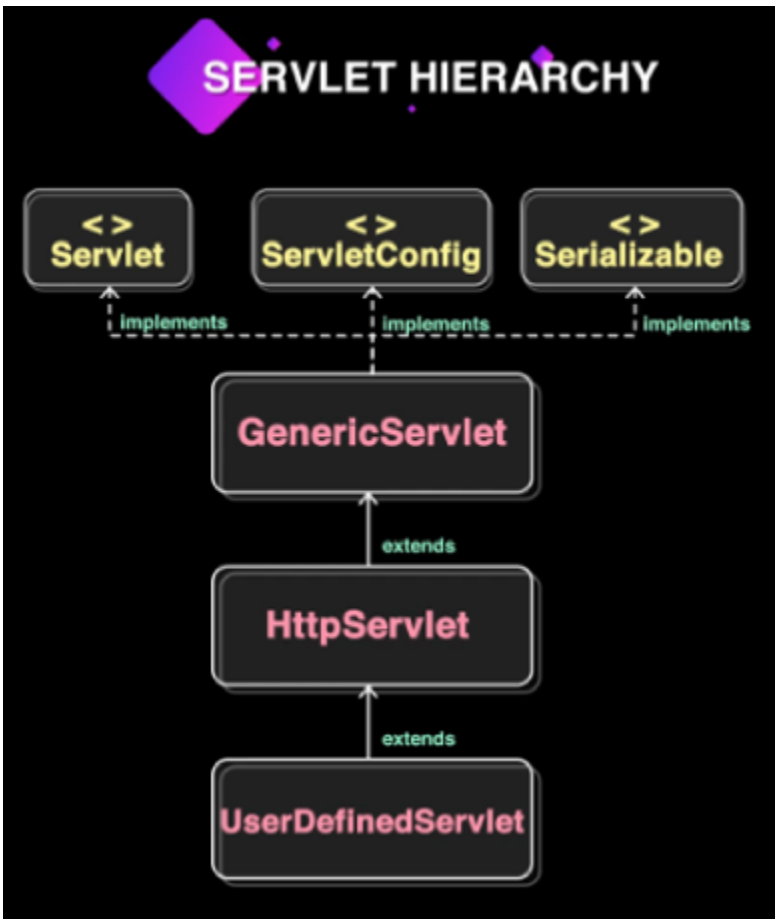
In this session we will be discussing a very important topic, which is **“SERVLETS”.**
Previously we have seen that the java programs which we have written, when executed the output can be displayed in a Monitor or it can be saved inside a file or a database.
Similarly, input can also be taken from a file or database.



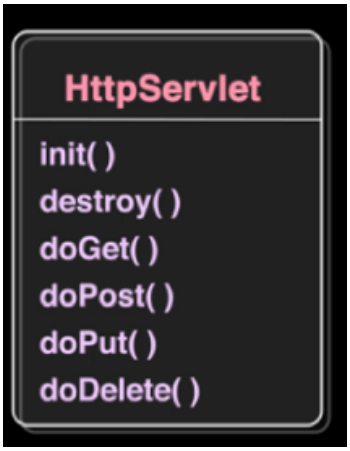
Now we will see a different kind of Java program, which can accept **HTTP REQUEST** as input and give **HTTP RESPONSE** as output.
A file which takes input as Request and gives output as Response is called **SERVLET.**



So how do we convert a regular java program into a Servlet?
For that first we have to understand,**Servlet Hierarchy**
We have **GenericServlet** class which implements 3 interfaces namely, **Servlet, ServletConfig and Serializable** interface.
HttpServlet class inherits from **GenericServlet** class and everything present inside GenericServlet and interfaces will be inherited.
HttpServlet is the class which we must inherit from if we want to convert our regular java program into a Servlet.



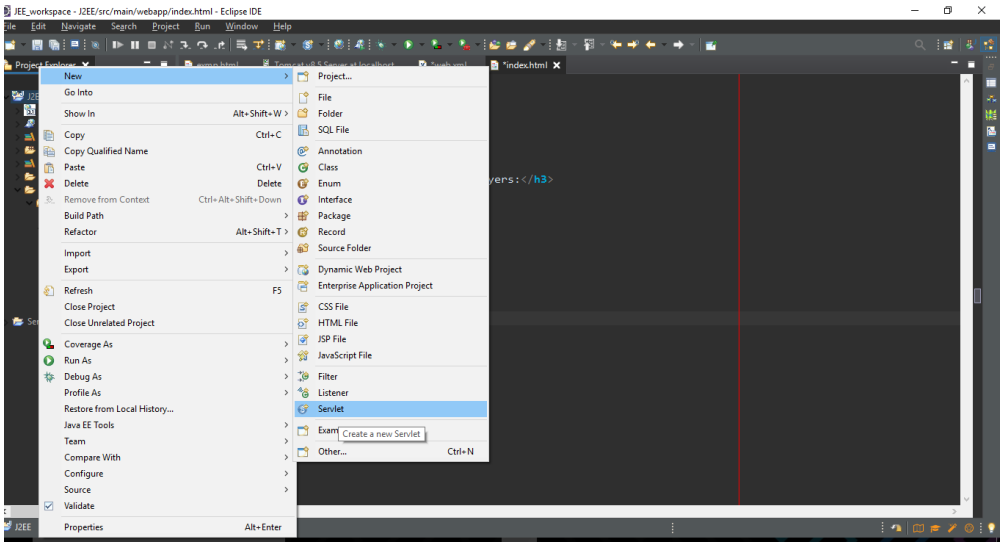
So we have to define a UserDefinedServlet and make it inherit from the HttpServlet class.
When we inherit from HttpServlet class, we inherit these methods-



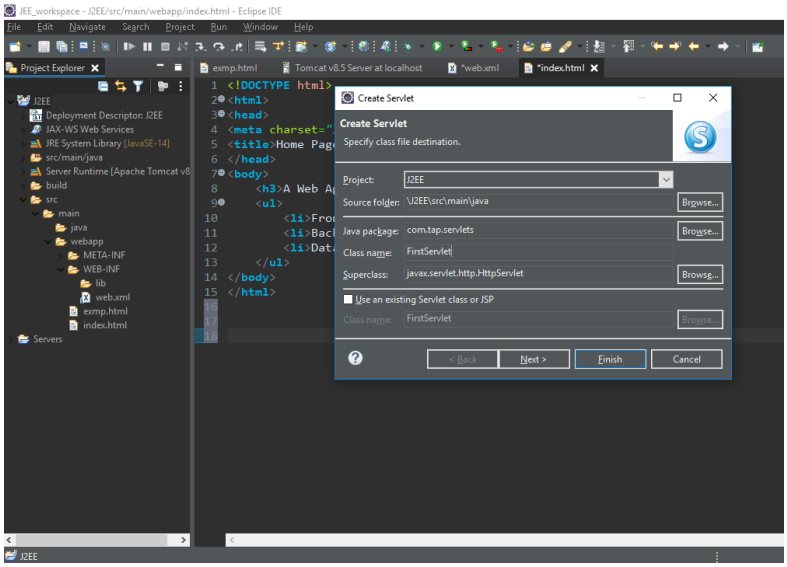
NOTE:
doGet( )
doPost( )
doPut( )
doDelete( ) are known as Service methods

And we have to override them, based on the operations we want to perform.
Let us now create our first servlet.

1. Right click on the project and click on New , select Servlet



2. Set the package name as “com.tap.servlets” and “FirstServlet” as class name



A class called FirstServlet will be created.
We will override doGet( ) service method in this program-

```java
package com.tap.servlets;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


class FirstServlet extends HttpServlet {
	@Override
	protected void doGet(HttpServletRequest req, HttpServletResponse
resp)
			throws ServletException, IOException {
		System.out.println("doGet() service method called");

	}

}
```

Now that our servlet is ready, let us see how we can deploy it.

In the previous session we had explained how we can deploy our program and with the help of a browser we can access it. But how do we access the **FirstServlet.java** file? Because when we enter the following URL in your browser-

http://localhost:8080/J2EE

The control will go to the WEB_INF folder and we have not mentioned anywhere in that folder to access the FirstServlet file.

So when a request is sent, automatically it should be able to identify to whom the request is for and automatically map to the FirstServlet.java and FirstServlet file should be called.

So how to achieve this Request Mapping?

For that we have our Deployment Descriptor (web.xml). Deployment Descriptor will take the URL and check the request in the URL and automatically activate the necessary resources.

Let us now see how we can achieve that -

1. Open web.xml file

```xml
<web-app>
	<welcome-file-list>
		<welcome-file>index.html</welcome-file>
	</welcome-file-list>


	<servlet>
```

```
            <servlet-name>Servlet1</servlet-name>
            <servlet-class>com.tap.servlets.FirstServlet</servlet-class>

      </servlet>

</web-app>
```

Note: servlet-name is the internal name which we set, so that it will be easier to access the file, and in servlet-class we have to mention the entire path of the file

2. Whenever the client requests for FirstServlet file, then we have to map it the file.That can we achieved by using the following tags-
   In url-pattern tag, we have to mention the url pattern, which if mentioned in the URL then it should map with the servlet mentioned in the servlet-name tag

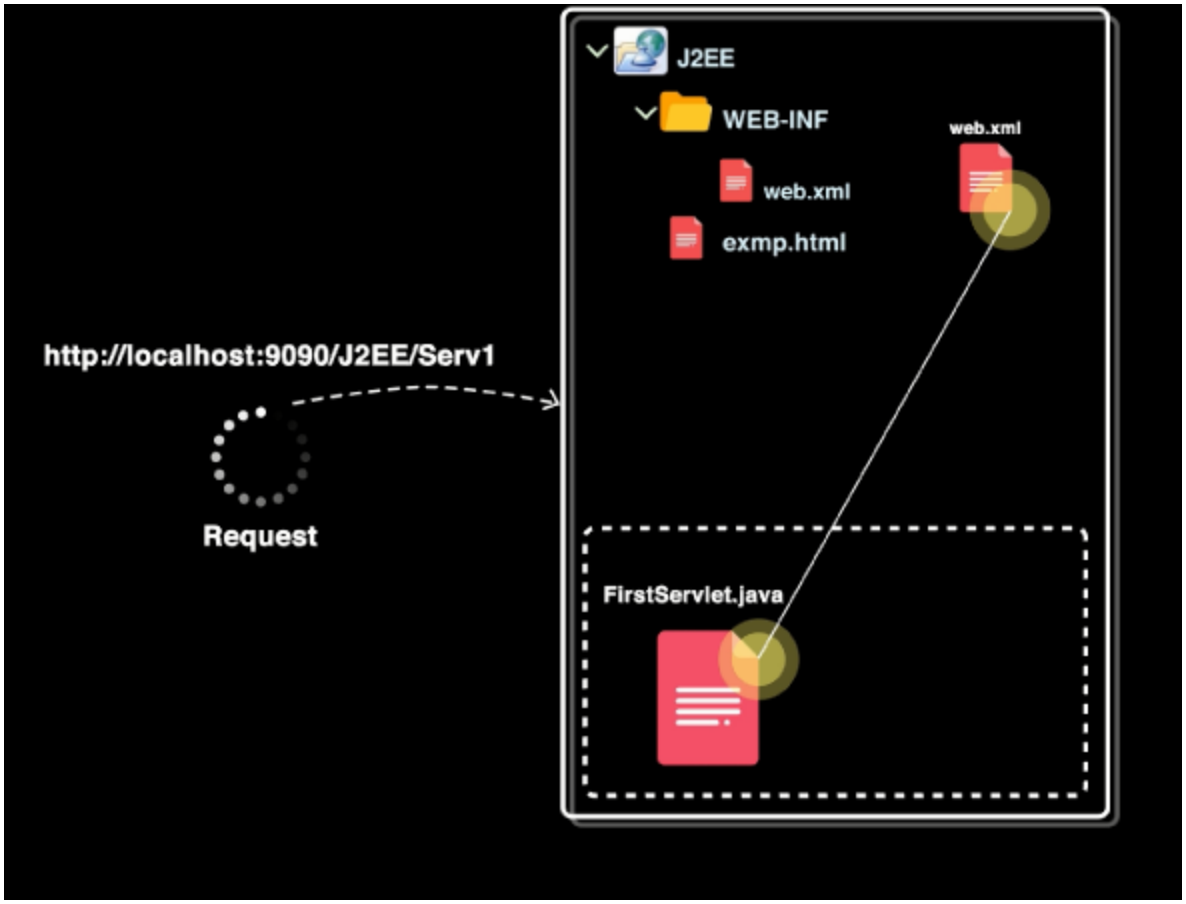
```
<web-app>
      <welcome-file-list>
            <welcome-file>index.html</welcome-file>
      </welcome-file-list>


      <servlet>
            <servlet-name>Servlet1</servlet-name>
            <servlet-class>com.tap.servlets.FirstServlet</servlet-class>
      </servlet>

      <servlet-mapping>
            <servlet-name>Servlet1</servlet-name>
            <url-pattern>/Serv1</url-pattern>
      </servlet-mapping>

</web-app>
```

So whenever a request is received, it is sent to web.xml file and web.xml file will map to the Servlet based on the url-mapping requested in the URL as shown below-



And when we try to launch the Servlet in the browser, we get an error because we had declared our FirstServlet.java class as **default** and Servlet classes should always be declared as **“public”.** If we don't declare the class as public, outsiders cannot access it.
So always remember that Servlets should always be declared as **public**.
When we execute this program, we will get the output in the console not in the browser window. We will see how we can get the output in the browser window, in the next session.