**SERVLETS**

Requirements: Any Server(Tomcat,GlassFish), IDE

Web.xml – deployment descriptor which tells what to render.

Context root – name which will be called when we will deploy the application on server.

Ex – context root was simpleservletproject. Then localhost:8080/ simpleservletproject will be called.

In web.xml we will be having multiple welcome-files, if the project is able to find any of these, server

Will render it.

**URL Mappings while creating Servlet-** this is auto populated with the same name as class, while creating

Servlets. What this actually means is that , tomcat comes to servlet to ask which pattern need to

Execute. We can have multiple url mappings, whichever pattern matches will be executed.

This URL Mapping can be found out in the annotation above Servlet

@WebServlet(urlPatterns ={“/SimpleServletPath”}).

doGet method is called as its default for any URL Request.

**Understanding Servlet**

http://localhost:8080/SimpleServletProject/SimpleServletPath

so here in the above url, SimpleServleProject – ContextRoot

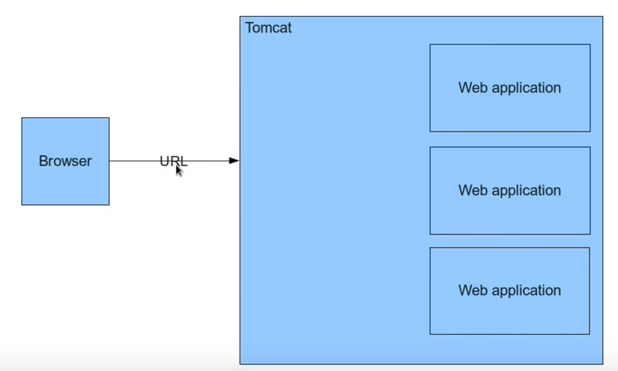
SimpleServletPath – Url Mapping that is specified at the time of servlet creation.

@webServlet – is the annotation which takes a lot of properties, but the important

One is **urlPatterns**  which actually tells that whenever this urlPatterns is called , then the particular

Class needs to be executed.

How Servlet Work?



As we can see in the above diagram we have a browser and there is a tomcat server where 3 web applications are deployed. Now we will discuss how the calls are made.

As soon as a browser makes the call to the URL, tomcat will create two objects – one request & one

response object. The url tells the browser what needs to be done. In our case the url is **localhost:8080**. Request object contains details about the request that the browser made.

It could have query parameters, or post variables. Response object will be empty for the time. It

Is actually meant for web application to process and put the response back ,so that it can be feeded to

the browser.

Tomcat examines the url and find out which application needs to be run. Here in url-**Localhost:8080/SimpleServletProject**, Tomcat finds out that **SimpleServletProject** need to be

run . Once it identifies that SimpleServletProject needs to be run, It will go to Web.xml of the project or it goes to the annotations and find out the servlet in the urlPatterns attribute.

Now once it find out which servlet needs to run, tomcat passes the req/resp object to Servlet and use the req. parameters and send back the response to the Response object.

The response object will now be having the processed html sent by Web Application. It is then sent back to the browser and displayed to the client.

**Servlet-Xml Configuration**

Another way of configuring servlet is by using Xml Configuration. This is an old way, before

Using annotations Xml’s were configured for Servlet. The Xml that we will be configuring here is

**Web.xml** which is also known as deployment descriptor.

Ex- <servlet>

<servlet-name>XmlServlets</servlet-name>

<servlet-class>org.Kamal.Servlets.XmlServlets</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>XmlServlets</servlet-name>

<url-pattern>/XmlServletPath</url-pattern>

</servlet-mapping>

# Here we are specifying the url pattern for servlet file. When web.xml will

**be called it will check the url pattern specified for the xml.**

Xml configuration is an old way of doing it. Annotation support came from java 5, before that xml

Configuration was done. So we should be aware of it as well.

## Post and passing parameters in servlet

We can use get request to pass parameters, but the point here is that those variables can be

seen in url. Its not safe to use get request for a potential secure site.

Using post for passing parameters require a html file and these variables are secure as it cannot

be seen in the url.

By default form method is doGet, to change it into doPost we need to explicitly set the method

as doPost.

**doGet & doPost**

doGet method is called when get method is issued to the servlet and doPost method is called when

a post method is issued to the servlet.

doGet and doPost methods are standard HTTP methods as per the protocol and each method has

a particular functions. Use get method when u want something from the server, and post is used

when u want to post/write something to ther server.

Lets take a very short example to understand **GET** & **POST.**  So here when we use get and submit any

Variables from form. After that if we refresh the page, the server will receive the

Value each time from the server. At the same time if we use post, it will send us a

Warning saying the operation will be repeated.

So the understanding that we get from here is that post operation is basically

used when we want the data to be changed or updated. Otherwise get needs to

be used.

It doesn’t mean that we can not use post operation for getting the data from

Server, we can use it but that’s not needed specifically.

**Request, session & context Objects**

So here we will be talking about the scopes of the object that are used, There are two objects- request & Response.

Tomcat creates these objects whenever any request is made. Whole application runs inside a container.

**Request and response objects are created - per access.**

**Servlet object -Not created per access, they are reused. Ex – if there are 100 users and a servlet, then there will not be 100 objects for the servlet. What tomcat do here is that it created 100 threads for servlet and give it to each user.**

Request object is created every time for every request. Question here is why its not reused.

Request object is a HTTP stateless protocol. What does it mean? Http protocol expect that the username and data that we sent for one request will not be remembered the next time we make request.

Ex- run the application which is accepting only a username. You will see the data that is sent as a username, now remove the username from the passed url . You will get a null entry which clearly means that username is not remembered for second request.

Now there are some use cases where u want the server to remember the data. Lets say a login page, u want it to be remembered so that u can customize the user experience. So here comes the concept of **Session** object which is provided by tomcat. Its basic use is to make the server remember the particular user.