<https://www.javatpoint.com/life-cycle-of-a-servlet>

-If we dont use the @WebServlet("/NewServlet") annotation in servlet then we should declare the servlets in web.xml file using the following code. Of course you can add extra information to the servlet like load on startup order.

<servlet>

<servlet-name>NewServlet</servlet-name>

<servlet-class>NewServlet</servlet-class>

</servlet>

-**init:** We use System.out in init because init writes to console. Thats why we dont see it in webpage. We see it one time when the website is deployed.

Servlet interface uses init method with servletConfig parameter. GenericServlet interface provides an init method without any parameters.

public void init()

throws ServletException

{

System.out.println("NewServlet");

}

**-startup order:** you can choose to deploy servlet in a certain order to make sure some of them start working as soon as possible. Also we want some servlets to be running before servlets. You change their order in web.xml file. First order is 0 not 1.

<servlet>

<servlet-name>NewServlet</servlet-name>

<servlet-class>NewServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet>

<servlet-name>NewServlet2</servlet-name>

<servlet-class>NewServlet2</servlet-class>

<load-on-startup>2</load-on-startup>

</servlet>

**-service:** When a request comes to a servlet the service method executes. If there are doGet and doPost methods implemented, the appropriate one is called.

The service() method is invoked upon each request after its initialization. Each request is serviced in its own separate thread.

-service() method is the main method to perform the actual task. The servlet container (i.e. web server) calls the service() method to handle requests coming from the client( browsers) and to write the formatted response back to the client.

**-destroy:** You can call destroy explicitly but it will be called again when undeploying. Which will cause an exception. If you use destroy explicitly make sure to handle this exception.

**-get vs post:** They are both request types.

**-Get:** It requests the data from a specified resource.

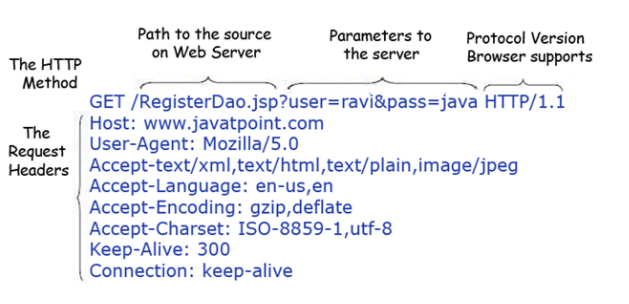
-Get sends data in url(header).

-Get is not secure because the data is in the url bar.

-Get request is idempotent. Until the response of the first request is delivered, next request will be ignored. Get does this to requests coming from a single client. So when you write a program and send two requests at the same time from a single client the get handles these one by one. While post cant but when multiple requests come from different clients both get and post are unsafe. You have to make them safe with your implementation.

-Get request is more efficient and used more than post.

-Get request can be bookmarked.



**-Post:** It submits the processed data to a specified resource.

-Post sends data in body(which is ).

-Post is secure because the data is not exposed in the url bar.

-Post request is not idempotent.

-Post request is less efficient and used less than get.

-Post request cant be bookmarked.



**-request, response:** When we request something as server we dont get it from client. We client already sent the request with the parameters. So when we get request.getParameter value we get it from server memory. The request parameters were saved in server memory when request happened.

**-getParameter:** We can get parameters from url. At the end of the url we enter and use the parameter or multiple parameters with the syntax below.

<http://localhost:8080/WebApplication1/newjsp.jsp?name=testname&amount=testamount>

String name = request.getParameter("name");

int amount = request.getParameter("amount")

**-Calling a servlet from another file:** If we are calling/sending data to a servlet from a html file, we need to use urlPattern of the servlet. Its url pattern is in the @WebServlet statement.

@WebServlet(name = "NewServlet", urlPatterns = {"/NewServlet"})

public class NewServlet extends HttpServlet {

<form action="NewServlet">

Name:<input type="text" name="name" /><br/><br/>

Tel number:<input type="text" name="tel\_number" /><br/><br/>

<input type="submit" value="add to database"/>

</form>

-We are able to refer to the servlet just with its name because when the application is deployed, all web pages and servlets are held under Web Pages folder. You can see this if you look at the “build” folder. That is the version of the program that actually runs. In the servlet file you can see the line

@WebServlet("/NewServlet")

**-Cookies and sessions:** Cookies cant have white space characters in their names or values.

-Why are cookies held in server? Because servers dont hold information with ip, they use accounts so a person without an account doesnt have any persistant information in server.

-Cookies have their own expire time. Sessions are deleted from client side when you close the browser. They are still in server side but browser gets new ones since it doesnt have any in client side. Other than that you can delete sessions using invalidate or we can set a time out just like a cookie.

You have only one session per visitor but you can have multiple cookies per visitor. Number of cookies a single website can place to clients computer is limited. It is around 20. And total number of cookies can be stored to clients computer is limited around 300 too.

**-Filters:** Lets say we have an AdminServlet which is an admin menu. We login with a username and password from index.html. This form sends the data to AdminServlet with action=”AdminServlet”. But before we let the user to the admin menu page, we need to check if he is really admin or not. We can use a filter here. That filter will intercept every request and response coming in and out of its target. We set filter’s target in web.xml in filter’s url-pattern line.

<servlet>

<servlet-name>AdminServlet</servlet-name>

<servlet-class>AdminServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>AdminServlet</servlet-name>

<url-pattern>/AdminServlet</url-pattern>

</servlet-mapping>

<filter>

<filter-name>MyFilter</filter-name>

<filter-class>MyFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>MyFilter</filter-name>

<url-pattern>/AdminServlet</url-pattern>

</filter-mapping>

In the code above we define a servlet and a filter that has the servlet as its target.