**-Cache:** Shift + F5: refresh while ignoring cache. Incognito mode: ignore cache.

**-Port:** C:\xampp\tomcat\conf\server.xml Connector port="8081"

**-Netbeans Connector Error:** Select the plain Chrome for the project in Netbeans.

* **Servlet API**

You can create a servlet using three different ways.

Implement 5 methods of Servlet interface.

Extend GenericServlet, this provides an implementation for all methods of Servlet, ServletConfig, and Serializable except the service method.

The HttpServlet class extends the GenericServlet class and implements the Serializable interface. It provides HTTP-specific methods such as doGet, doPost, doHead, doTrace, etc.

System.out writes to console (glassfish server console). response.getWriter() outputs to response.

* **Directory Structure**

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

If a file is in a subfolder of the web, you need to type that in the URL. If login.jsp is at web\public\login.jsp, then we access it via

http://localhost:8080/WebApplication/public/login.jsp

* **Deployment descriptor**

When you create a servlet its information is added to web.xml if you choose the “Add info to deployment descriptor” option. If you copy-paste a servlet, add the appropriate metadata to web.xml and other places.

To invoke the servlet for all files under foo,

<servlet-mapping>

<servlet-name>ServletName</servlet-name>

<url-pattern>/foo/\*</url-pattern>

</servlet-mapping>

Servlets can be used in the welcome file list.

<welcome-file-list>

<welcome-file>ServletURL</welcome-file>

</welcome-file-list>

Load on startup tag can also be used when we want some servlets to run before some other servlets. You change their order in web.xml file.

<servlet>

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

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

<load-on-startup>0</load-on-startup>

</servlet>

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

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

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

</servlet>

* **ServletRequest**

**-get, post:** get method is the default method. No explicit form method means use get method.

**-getParameter:** We can get parameters that were sent with HTTP protocol. One way to send parameters is after the file name in the URL. This is a get request.

http://localhost:8080/WebApplication/newjsp.jsp?name=testName&amount=testAmount

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

int amount = request.getParameter("amount")

You can send a get request with a form too.

<form action="ServletURL">

Enter your name: <input type="text" name="name">

<input type="submit" value="Login">

</form>

http://localhost:8080/HttpServlet2/ServletURL?name=michael // result URL

Or you can send the parameters with a post request.

<form action="ServletURL" method="post">

Enter your name: <input type="text" name="name">

<input type="submit" value="Login">

</form>

http://localhost:8080/HttpServlet2/ServletURL // result URL

**-setAttribute/getAttribute:** Uses request scope.

**-Calling a Servlet from HTML or Servlet:** We can create links or forms to Servlets from HTML or from Servlet.   
 In Servlet, use \” instead of “, to surround values of HTML attributes. Or use single quotation marks '. Or don’t use them at all since they are only mandatory in XHTML. You have to use them if the value has spaces in it.

-You dont have to use forms (action = ServletURL) to call servlets. You can just link to them,

<a href="ServletURL">View My Guestbook</a>

Here is how we do this in servlet.

out.print("<a href=ServletURL>View My Guestbook</a>");

-You can even send parameters while calling a servlet with a link,

<a href="ServletURL?operation=view">View My Guestbook</a>

Here is how we do this in servlet.

out.print("<a href=ServletURL?operation=view> Guestbook</a>");

Using concatenated strings to create the link

String operation = getParameter("operation");

out.print("<a href=ServletURL?operation=" + operation + "> Guestbook</a>");

-You can also use a form (button).

<form action="ServletURL">

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

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

<input type="hidden" value="Insert" name="operation">

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

</form>

Here is how we do this in servlet.

out.print("<form action=ServletURL>");

out.print("Name:<input type=text name=name /><br><br>");

out.print("Tel number:<input type=text name=tel\_number /><br><br>");

out.print("<input type=hidden value=insert name=operation>");

out.print("<input type=submit value='Insert to database'>");

out.print("</form>");

* **Servlet Collaboration**

**-RequestDispatcher**

**-forward:** Response ofanother resource is sent back.

**-include:** Include the response of a resource.

String p = request.getParameter("userPass");

if (p.equals("servlet")) {

RequestDispatcher rd = request.getRequestDispatcher("WelcomeServlet");

rd.forward(request, response);

}

else {

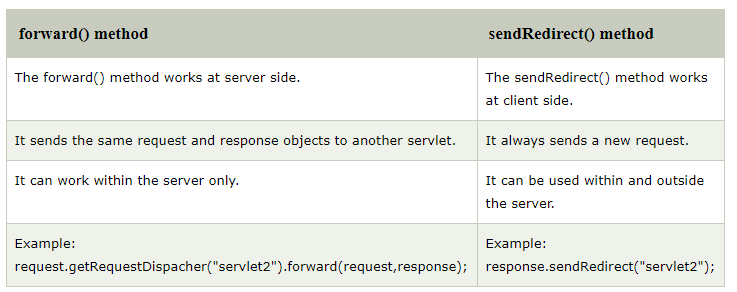
RequestDispatcher rd = request.getRequestDispatcher("index.html");

rd.include(request, response);

out.print("<br> Username or password error.");

}

**-SendRedirect:** Redirects response to another resource.



* **ServletConfig, ServletContext, Servlet Attribute**

**-ServletConfig:** There is oneweb ServletConfig object for each Servlet.

**-ServletContext:** There is one ServletContext object for each web application.

**-setAttribute/getAttribute:** Usesapplication scope.

* **Session Tracking**

**-Cookies:** Dependent on browsers (disabled cookies). Cookies have an expiration time. Cookie names can’t have white space characters. Cookies are sent to the server with every request.

**-Hidden Form Field:** Independent from browsers.

**-URL Rewriting:** Independent from browsers.

**-Sessions:** Browsersdeletes the session cookies when the browser is closed. You can delete sessions using invalidate or setting a timeout. You have only one session per visitor.

HttpSession session = request.getSession(false);

if(session == null) // session.getAttribute("userName")

response.sendRedirect("login.jsp");

**-setAttribute/getAttribute:** Uses session scope.

**-Cache-control:** If you don’t want the user to be able to see a page using the back button etc.

response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate");

* **Event and Listener**
* **Filters**

The filter will intercept every request and response coming in and out of its target. We set the filter’s target in web.xml in filter’s url-pattern line.

* **Other Notes**

**-Flush:** Normally the complete response is formed at the end of the method and sent to the client. But we can use the flush method to send the responses that were added so far. The responses added after the flush method will be included at the end of the previous response.

out.write("Message1 <br>");

out.flush();

try {

Thread.sleep(1000);

} catch (InterruptedException ex) { }

out.write("Message2");

**-Login:** The servlet that checks login should not also show the user a page (information). After you check the login of the user, send the user to another page such as profile. If you don’t do this, someone can login, then immediately logout, then press the back button and see that page. Even if you have the no-cache set up. Because it is a doPost page and the form gets resubmitted.

Does this problem affect all doPost pages? (logging out from a doPost page and then going back)

**-Checking textfield and radiobutton inputs:** When you get a textfield value and send it to the next page, you can do a check by seeing whether the parameter is an empty string or not. But for radio buttons, you are supposed to check whether they are null or not.