30-6-2020

Whenever a servlet program starts execution—

If the servlet is requested for the first time—
web container does the following tasks—

→ loads the servlet class
→ Instantiates the servlet class
→ Calls the init() method passing the ServletConfig object.

From 2nd time onwards—

→ Calls the service method passing request and response objects.

In last — web container calls the destroy() method when it needs to remove the servlet such as at time of stopping server or undeploying the project.

The web container is responsible to handle the request. Let us see how it handles the request;—

→ maps the request with the servlet in the web.xml file.
→ creates request and response objects for this request.
→ calls the service() method on the thread.
→ public service() method calls the protected service() method.
→ the protected service() method calls the doGet() ~~and~~ depending on the type of request.

→ The doGet() generates the response and it is passed to the client.

→ After sending the response, web container deletes the request and response objects.

Code inside the public service method :-

```
public void service(ServletRequest req, ServletResponse
                    res)
throws ServletException, IOException
{
    HttpServletRequest request;
    HttpServletResponse response;
  try{  request = (HttpServletRequest)req;
        response = (HttpServletResponse) res;
    }
catch(ClassCastException e){
 throw new ServletException("non-HTTP request or
                             response");
                          }
 service(request, response);
}
```

Code inside the protected service method:-

```
protected void service(HttpServletRequest req, HttpServletResponse
                       res) throws ServletException, IOException
{   String method = req.getMethod();
    if(method.equals("GET")){
```

```
        long lastModified = getLastModified(req);
        if (lastModified == -1L)
        {
            doGet(req, res);
        }
        ....
        //rest of the code here
    }
}
```

War File :— A war (web archive) file contains files of a web project. It may have servlet, xml, jsp, image, html, css, js etc. files.

How war file is created and how all files are extracted from a war file. Let us try to know :—

From inside the project directory, open the command prompt—.

- -c → for creating file
- -v → to generate verbose output
- -f → to specify archive filename.

```
jar -cvf project.war ★ ↵
```

All files & directories are added to project.war file.

Extract the files from war file :—

```
jar -xvf project.war ↵
```

How to deploy the war file

In Apache Tomcat Server, go to webapps directory and paste the war file there.

In this way you are able to access the project by the web-browser.

<web.xml> file :—

This file has some important tags :—

<welcome-file-list> tag is element of <web-app>

<welcome-file> is sub element of <welcome-file-list>

<welcome-file>, if specified then priority of opening default home page is the filename specified in this tag.

If <welcome-file> is not used then priority goes to index.html file

Load on startup in web.xml

The <load-on-startup> element of <web-app> is responsible for loading the servlet at the time of deployment or server starts if value is +ve. We can pass +ve and -ve values for the servlet. So, it will take less time for responding to first request.

```
<web-app>
...
<servlet> <servlet-name> servlet1 </servlet-name>
<servlet-class> com.ssc.Servlet </servlet-class>
<load-on-startup> 0 </load-on-startup>
</servlet>
```

```
<servlet>
<servlet-name>servlet2</servlet-name>
<servlet-class>com.ssc.LoginServlet</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
.....
</web-app>
```

There are two defined servlets, both will be loaded at the time of project deployment or server start. But servlet1 will be loaded first then servlet2.

RequestDispatcher interface use in Servlet :—

The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp.

There are two important methods defined in this interface :—

1. public void forward(ServletRequest req, ServletResponse resp) throws ServletException, IOException

— this method forwards a request from a servlet to another resource on the server.

2. public void include(ServletRequest req, ServletResponse resp) throws ServletException, IOException.

— includes the content of a resource (Servlet, JSP Page, or html) in the response.

getRequestDispatcher() — method of ServletRequest interface returns the object of RequestDispatcher.

Syntax:- public RequestDispatcher getRequestDispatcher(String resource);

Eg:- RequestDispatcher rd = request.getRequestDispatcher("Servlet2");
rd.forward(request, response);
(method may be include or forward)

→ where "Servlet2" is the <url-pattern> of the servlet where request is forwarded now.

In the example uploaded ~~yesterday~~ today using Eclipse IDE, we have following files:—

- index.html — for getting user input
- Login.java — Servlet class for processing response if password = "servlet" then WelcomeServlet
- WelcomeServlet.java — Servlet class for Welcome message
- web.xml — deployment descriptor file that contains information about the servlet.

SendRedirect() method:— This method belongs to HttpServletResponse interface and can be used to redirect response to another servlet, jsp or html file. It accepts complete URL or relative URL. It works at client side because it uses the url bar of the browser to make another request.

So, it can work inside and outside the server.

Difference between forward() and sendRedirect()

forward() method	sendRedirect() method
① forward(), works at server side.	① works at client side.
② Sends the same request and response objects to another Servlet.	② It always sends a new request.
③ It can work within the server only.	③ It can be used within and outside the server.
④ Example: request.getRequestDispatcher("Servlet2"). forward(request, response);	④ Example: response.sendRedirect("servlet

Example of sendRedirect() method in servlet:—

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class demoRedirect extends HttpServlet{
  public void doGet(HttpServletRequest req,
    HttpServletResponse res) throws ServletException, IOException
  {
    res.setContentType("text/html");
    PrintWriter pw = res.getWriter();
  response.sendRedirect("http://www.google.co.in");
   pw.close();
    }
}
```

For this example, we are using index.html file:-

```
<!DOCTYPE html>
<html> <head> <title> Send Redirect Example </title>
</head> <body>
<form action = "MySearcher">
<input type = "text" name = "name">  
<input type = "submit" value = "Google Search">
</form>
</body>
</html>
```

MySearcher.java file

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class MySearcher extends HttpServlet {
protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws Exception{
String name = request.getParameter("name");
response.sendRedirect("https://www.google.co.in/#q="+
                                        name);
}
}
```

You can also download the all above example using the download link provided in college web site today.