1. **Servlets**

Servlets runs and enabled in a java server which is also a Java class. In the form of an HTML page, one of the special type of servlet is a HTTP servlet which provides an HTTP response and handles an HTTP request. A java servlets is a java objects which aims to play the role in client-server communications of a server components.

1. **1 Java Servlet Versions**

Java Servlet has these versions:

* J2EE 1.2 (December 1999) (**Java Servlet 2.2**, JSP 1.1, EJB 1.1, JDBC 2.0)
* J2EE 1.3 (September 2001) (**Java Servlet 2.3**, JSP 1.2, EJB 2.0, JDBC 2.1)
* J2EE 1.4 (November 2003) (**Java Servlet 2.4**, JSP 2.0, EJB 2.1, JDBC 3.0)
* Java EE 5 (May 2006) (**Java Servlet 2.5**, JSP 2.1, JSTL 1.2, JSF 1.2, EJB 3.0, JDBC 3.0)
* Java EE 6 (December 2009) (**Java Servlet 3.0**, JSP 2.2/EL 2.2, JSTL 1.2, JSF 2.0, EJB 3.1, JDBC 4.0)
* Java EE 7: end of 2012.

1. **Servlets life cycle**

* Servlet is initialized in **init()** method by calling.
* To process or proceed to a client’s request ,the servlet calls **service()** method.
* For terminating the servlets, call **destroy()** method.
* Collect the garbage in servlet by the garbage collector of the JVM (Java Virtual Machine).

**2.1 init() Method**

* Called only once.
* One-time initializations
* Called only once the servlet is created.
* Not called for any user requests on the succeeding time or afterwards.
* It will be used through the entire life of the servlet that loads or creates some data.

public void init() throws ServletException {

// Initialization code...

}

**2.2 service() Method**

* Main method.
* Handle requests from the client browsers
* Write the arrangement or formatted response to the client.
* It checks the HTTP request type such as GET,DELETE,POST,PUT, etc.
* it also calls doPost, doGet, doDelete, doPut, etc.

public void service(ServletRequest request, ServletResponse response)

throws ServletException, IOException {

}

* called by service and container method that supports doPost, doGet, doPut, doDelete methods as appropriate.

**2.3 doGet() Method**

* it supports the HTTP GET requests.
* Used when a small amount of data and insensitive data.
* Results from a normal request from HTML form or URL that has no method being specified.

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// Servlet code

}

**2.4 doPost() Method**

* it supports the HTTP POST requests.
* Used when almost large amount of a sensitive data has to be sent.

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// Servlet code

}

**2.5 destroy() Method**

* Called once at the end of the servlet life cycle.
* It contribute the servlet to close database connections, write cookie lists, terminate background threads and other cleanup activities.

public void destroy() {

// Finalization code...

}

1. **Servlet Container**

Servlet Container is also called servlet engine which handles components of a Java Servlet that can be run in different procedure.



Figure 1 : Servlet’s user requests process

**3.1 Services**

Following services of a Servlet Container:

* It manages or handles the life cycle of servlet.
* The servlet container handles resources like JSP pages, servlets and HTML files.
* To preserve session, it appends or attaches session ID to the URL path.
* It has security service.
* File systems such as local file system or remote file system, network services can load in a servlet class.

**3.2 Container Configurations of Servlet**

Servlets container have three ways on configuring the web server that manages servlets:

* Standalone container
* In-process container
* Out-process container

*Standalone container –* a Java-based server which is the functionality of the Web Server is received by the Servlet container.

*In-Process container -*  the web server is separated because the plug-in runs in another program between the main server’s address space.

*Out-Process container –* different programs of the servlet container and web server runs in a different procedure or process.

**4 Servlet Demo**

**Step 1:**

Open *NetBeansIDE >> File >> New Project >> WebApplication >>* Set the Project Name as *ServletDemo*

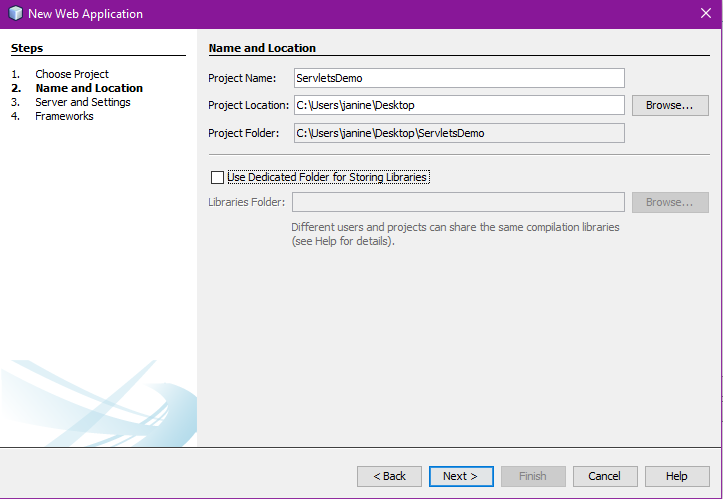


Figure 2 : Creating new Web Application project  
NetBeansIDE : ServletsDemo

**Step 2:**

Click *Next >>* as shown in Figure 2. It will now create new project with the following different directory structure.

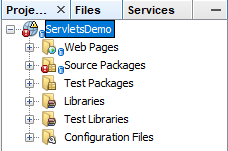


Figure 3: Project directory structure after creating new project.

**Step 3:**

By Right Clicking on the new servlet application. *Project Directory (ServletsDemo) >> New >> Servlet…*

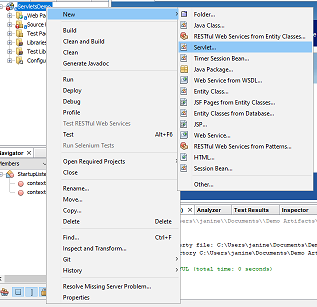


Figure 4: Adding Servlet file

**Step 4:**

Add Class name of the Servlet as “ServletDemo” and click on *Next.*

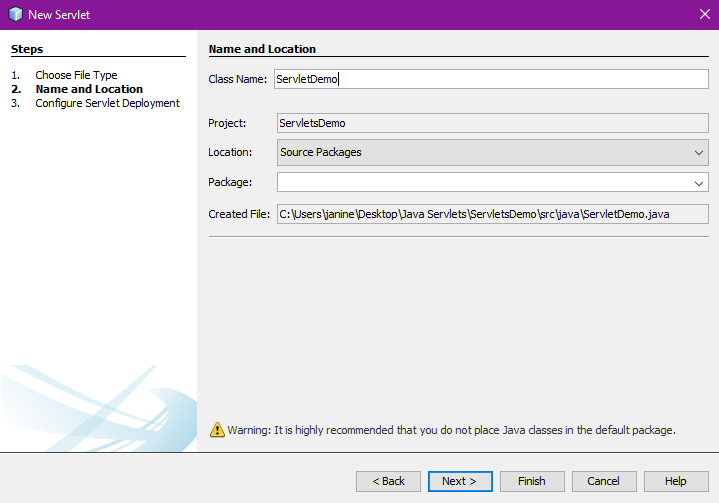


Figure 5: Adding Class Name of Servlet

**Step 5:**

By checking the “*Add information to deployment descriptor (web.xml)”* to Configure Servlet Deployment. Next add the URL Pattern as “ServletsDemonstration”. This will generate the *web.xml* file in the *WEB-INF* folder.

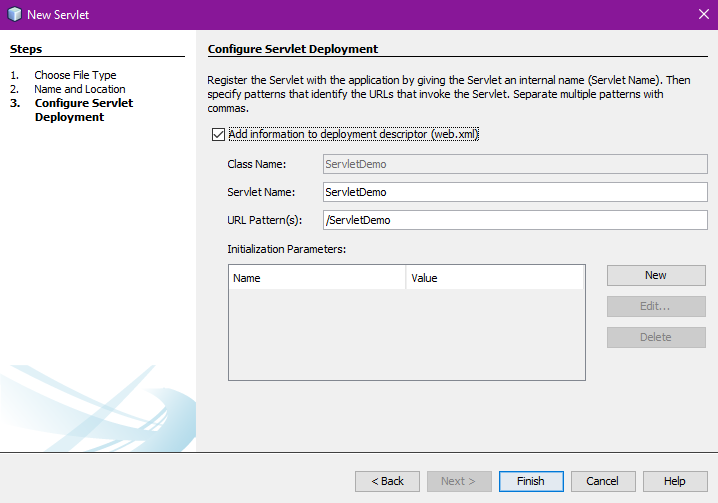


Figure 6: Configuring Servlet Deployment

**Step 6:**

And click the Finish button as shown in Figure 6. And it will add the *ServletDemo.java* servlet under the source packages of the project directory.

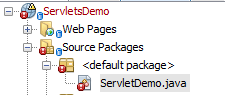
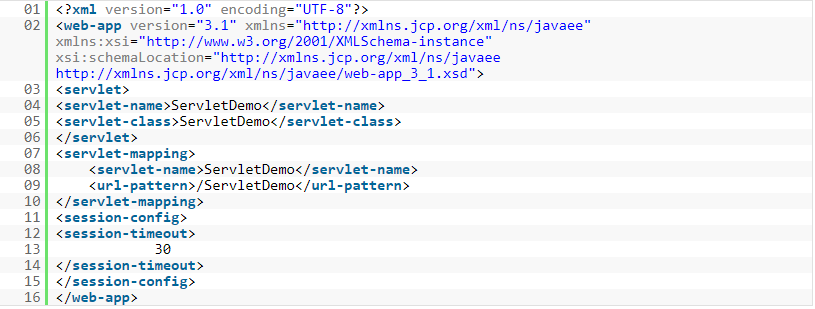
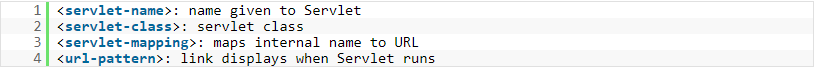


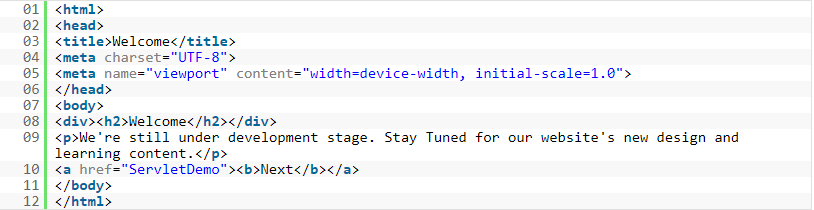
Figure 7: Project directory changes after configuring

Code for deployment descriptor in web.xml with the URL-patter as /ServletDemo.

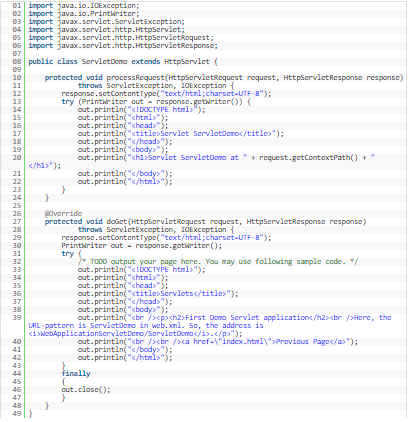




Index.html



ServletDemo.java

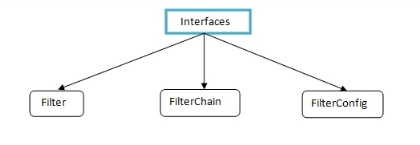


**5 Filter**

Filters can be used to convert one to another format of the content of responses, requests, and header.

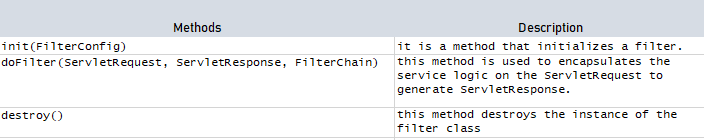
* In the deployment descriptor is where the Filter class is declared.
* The usage of filter is to write reusable components
* Usage of filter under web application for some task:
  + Compression
  + Validation
  + Internationlization
  + Verification

**5.1 Interface**



**Filter**

The basic and initial interface that should be implement in all filter class.   
*Java.servlet.Filter* interface methods:



**FilterChain**

FilterChain is a filter that collects information or data which is about more than 1 filter. All filters

before processing of a request should be used on request.

**FilterConfig**

Initialized first the filters before the object to be used. Deployment descriptor on web.xml contains

configuration information. The interface of FilterConfig’s object is used to get configuration information

that is specified under web.xml.



[**https://www.tutorialspoint.com/servlets/servlets-life-cycle.htm**](https://www.tutorialspoint.com/servlets/servlets-life-cycle.htm)

[**https://docs.oracle.com/javaee/6/tutorial/doc/bnafi.html**](https://docs.oracle.com/javaee/6/tutorial/doc/bnafi.html)

[**https://www.javatpoint.com/life-cycle-of-a-servlet**](https://www.javatpoint.com/life-cycle-of-a-servlet)

[**https://www.ntu.edu.sg/home/ehchua/programming/java/JavaServlets.html**](https://www.ntu.edu.sg/home/ehchua/programming/java/JavaServlets.html)

[**https://www.javacodegeeks.com/2014/12/java-servlet-tutorial.html**](https://www.javacodegeeks.com/2014/12/java-servlet-tutorial.html)