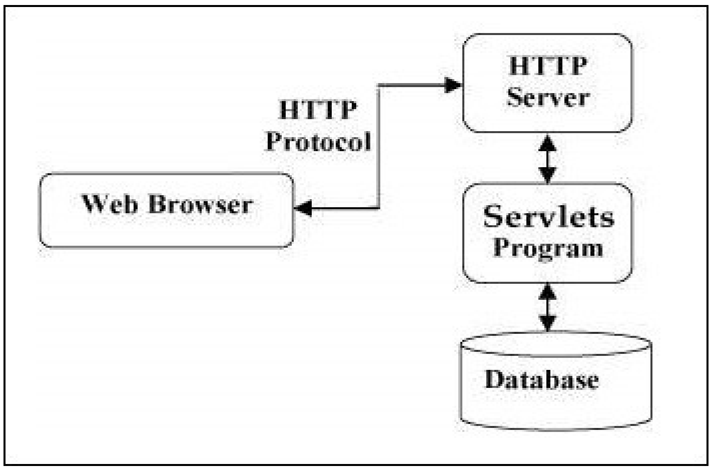
Servlet:

What are Servlets?

Java Servlets are programs that run on a Web or Application server and act as a middle layer between a requests coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.



Servlets Tasks

Servlets perform the following major tasks −

- Read the explicit data sent by the clients (browsers) - HTML form/ applet/ HTTP client program.
- Read the implicit request data sent by the clients. This includes cookies, media types and compression schemes the browser understands, and so forth.
- Process the data and generate the results.
- Send the explicit data to the clients - text (HTML or XML), binary (GIF images), Excel, etc.
- Send the implicit HTTP response to the clients. - type of document is being returned, setting cookies and caching parameters, and other such tasks.

Servlets Packages

- javax.servlet

  The javax.servlet package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.

The javax.servlet.http package contains interfaces and classes that are responsible for http requests only.

There are many interfaces in javax.servlet package. They are as follows:

- ✓ Servlet
- ✓ ServletRequest
- ✓ ServletResponse
- ✓ RequestDispatcher
- ✓ ServletConfig
- ✓ ServletContext
- ✓ SingleThreadModel
- ✓ Filter
- ✓ FilterConfig
- ✓ FilterChain
- ✓ ServletRequestListener
- ✓ ServletRequestAttributeListener
- ✓ ServletContextListener
- ✓ ServletContextAttributeListener

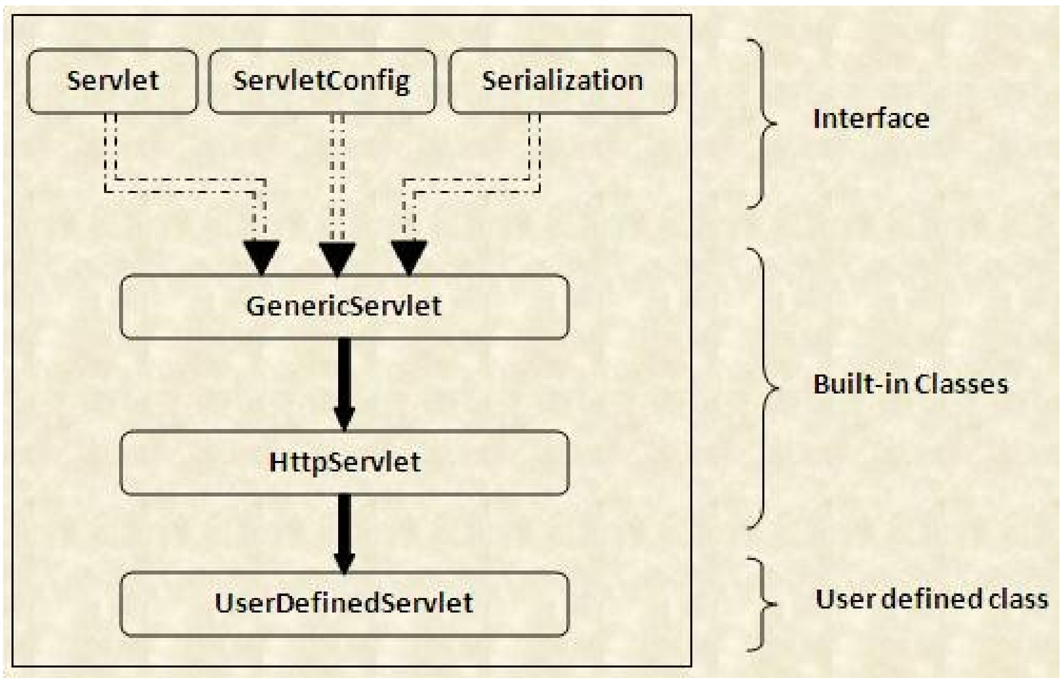
There are many classes in javax.servlet package. They are as follows:

- ✓ GenericServlet
- ✓ ServletInputStream
- ✓ ServletOutputStream
- ✓ ServletRequestWrapper
- ✓ ServletResponseWrapper
- ✓ ServletRequestEvent
- ✓ ServletContextEvent
- ✓ ServletRequestAttributeEvent
- ✓ ServletContextAttributeEvent
- ✓ ServletException
- ✓ UnavailableException

- javax.servlet.http

  There are many interfaces in javax.servlet.http package. They are as follows:

  - ✓ HttpServletRequest
  - ✓ HttpServletResponse
  - ✓ HttpSession



- ✓ HttpSessionListener
- ✓ HttpSessionAttributeListener
- ✓ HttpSessionBindingListener
- ✓ HttpSessionActivationListener
- ✓ HttpSessionContext (deprecated now)

There are many classes in javax.servlet.http package. They are as follows:

- ✓ HttpServlet
- ✓ Cookie
- ✓ HttpServletRequestWrapper
- ✓ HttpServletResponseWrapper
- ✓ HttpSessionEvent
- ✓ HttpSessionBindingEvent
- ✓ HttpUtils (deprecated now)

Servlets Life Cycle

A servlet life cycle can be defined as the entire process from its creation till the destruction. The following are the paths followed by a servlet.

- The servlet is initialized by calling the **init()** method.

```
public void init() throws ServletException {
   // Initialization code...
}
```

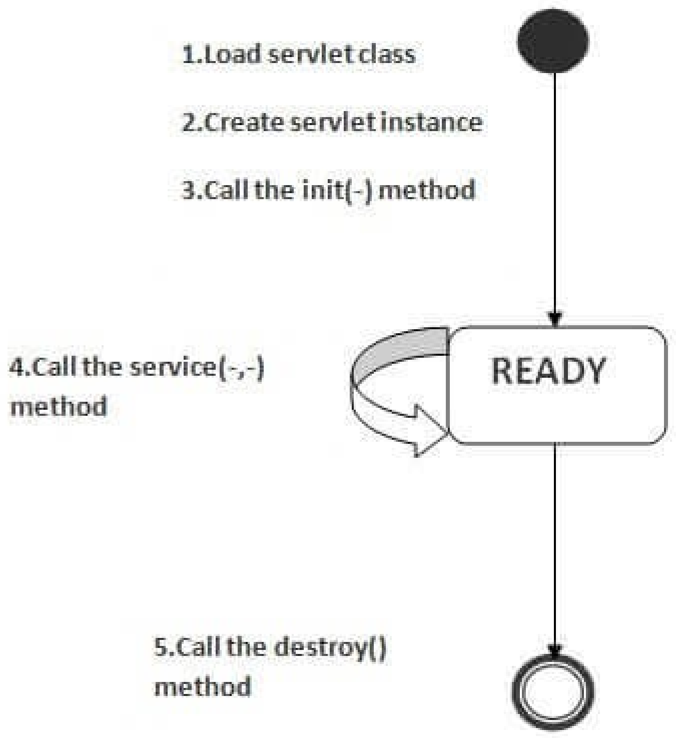
- The servlet calls **service()** method to process a client's request.

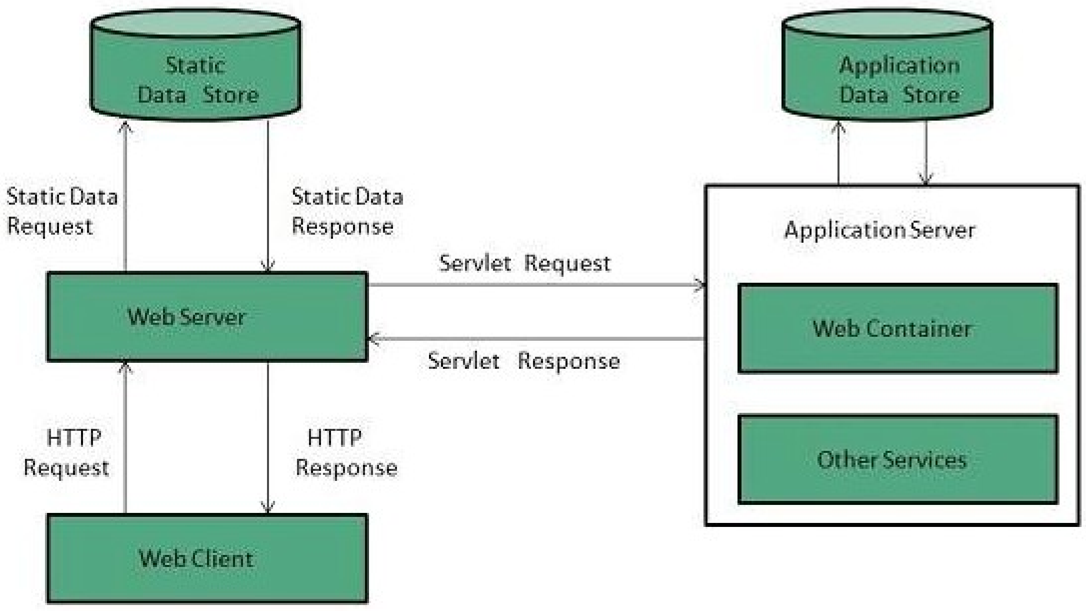
```
public void service(ServletRequest request, ServletResponse response)
   throws ServletException, IOException {
}
//************************************************************
public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
   // Servlet code
}
//************************************************************
public void doPost(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException {
   // Servlet code
}
```

- The servlet is terminated by calling the **destroy()** method.

```
public void destroy() {
// Finalization code...
}
```

- Finally, servlet is garbage collected by the garbage collector of the JVM.


1.Load servlet class
2.Create servlet instance
3.Call the init(-) method
4.Call the service(-,-) method
READY
5.Call the destroy() method


Static Data Store
Application Data Store
Static Data Request
Static Data Response
Application Server
Servlet Request
Web Server
Web Container
Servlet Response
HTTP Request
HTTP Response
Other Services
Web Client

Servlet Interface

Following are the methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

public void init(ServletConfig config)

initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.

public void service(ServletRequest request,ServletResponse response)

provides response for the incoming request. It is invoked at each request by the web container.

public void destroy()

is invoked only once and indicates that servlet is being destroyed.

public ServletConfig getServletConfig()

returns the object of ServletConfig.

public String getServletInfo()

returns information about servlet such as writer, copyright, version etc.

Example

First.java

```
import java.io.*;
import javax.servlet.*;

public class First implements Servlet{
ServletConfig config=null;

public void init(ServletConfig config){
this.config=config;
System.out.println("servlet is initialized");
}

public void service(ServletRequest req,ServletResponse res)
throws IOException,ServletException{

res.setContentType("text/html");

PrintWriter out=res.getWriter();
out.print("<html><body>");
out.print("<b>hello simple servlet</b>");
out.print("</body></html>");

}
public void destroy(){System.out.println("servlet is destroyed");}
public ServletConfig getServletConfig(){return config;}
public String getServletInfo(){return "copyright 2007-1010";}

}
```

Web.xml

```
<web-app>

        <servlet>

                <servlet-name>s1</servlet-name>

                <servlet-class>First</servlet-class>

        </servlet>

        <servlet-mapping>

                <servlet-name>s1</servlet-name>

                <url-pattern>/hello</url-pattern>

        </servlet-mapping>

</web-app>
```

GenericServlet Class

GenericServlet class implements **Servlet**, **ServletConfig** and **Serializable**interfaces. It provides the implementation of all the methods of these interfaces except the service method.

GenericServlet class can handle any type of request so it is protocol-independent.

You may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

There are many methods in GenericServlet class. They are as follows:

1. **public void init(ServletConfig config)** is used to initialize the servlet.
2. **public abstract void service(ServletRequest request, ServletResponse response)** provides service for the incoming request. It is invoked at each time when user requests for a servlet.
3. **public void destroy()** is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
4. **public ServletConfig getServletConfig()** returns the object of ServletConfig.
5. **public String getServletInfo()** returns information about servlet such as writer, copyright, version etc.
6. **public void init()** it is a convenient method for the servlet programmers, now there is no need to call super.init(config)
7. **public ServletContext getServletContext()** returns the object of ServletContext.
8. **public String getInitParameter(String name)** returns the parameter value for the given parameter name.
9. **public Enumeration getInitParameterNames()** returns all the parameters defined in the web.xml file.
10. **public String getServletName()** returns the name of the servlet object.
11. **public void log(String msg)** writes the given message in the servlet log file.
12. **public void log(String msg,Throwable t)** writes the explanatory message in the servlet log file and a stack trace.

Example:

First.java

```
import java.io.*;
import javax.servlet.*;

public class First extends GenericServlet{
public void service(ServletRequest req,ServletResponse res)
throws IOException,ServletException{
```

```
res.setContentType("text/html");

PrintWriter out=res.getWriter();
out.print("<html><body>");
out.print("<b>hello generic servlet</b>");
out.print("</body></html>");

}
}
```

HTTPServlet Class

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

There are many methods in HttpServlet class. They are as follows:

1. **public void service(ServletRequest req,ServletResponse res)** dispatches the request to the protected service method by converting the request and response object into http type.
2. **protected void service(HttpServletRequest req, HttpServletResponse res)** receives the request from the service method, and dispatches the request to the doXXX() method depending on the incoming http request type.
3. **protected void doGet(HttpServletRequest req, HttpServletResponse res)** handles the GET request. It is invoked by the web container.
4. **protected void doPost(HttpServletRequest req, HttpServletResponse res)** handles the POST request. It is invoked by the web container.
5. **protected void doHead(HttpServletRequest req, HttpServletResponse res)** handles the HEAD request. It is invoked by the web container.
6. **protected void doOptions(HttpServletRequest req, HttpServletResponse res)** handles the OPTIONS request. It is invoked by the web container.
7. **protected void doPut(HttpServletRequest req, HttpServletResponse res)** handles the PUT request. It is invoked by the web container.
8. **protected void doTrace(HttpServletRequest req, HttpServletResponse res)** handles the TRACE request. It is invoked by the web container.
9. **protected void doDelete(HttpServletRequest req, HttpServletResponse res)** handles the DELETE request. It is invoked by the web container.
10. **protected long getLastModified(HttpServletRequest req)** returns the time when HttpServletRequest was last modified since midnight January 1, 1970 GMT.

Example
HelloWorld.java

```
import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;



// Extend HttpServlet class

public class HelloWorld extends HttpServlet {
   private String message;


   public void init() throws ServletException {
      // Do required initialization
      message = "Hello World";
   }


   public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {


      // Set response content type
      response.setContentType("text/html");


      // Actual logic goes here.
      PrintWriter out = response.getWriter();
      out.println("<h1>" + message + "</h1>");
   }


   public void destroy() {
      // do nothing.
   }
}
```

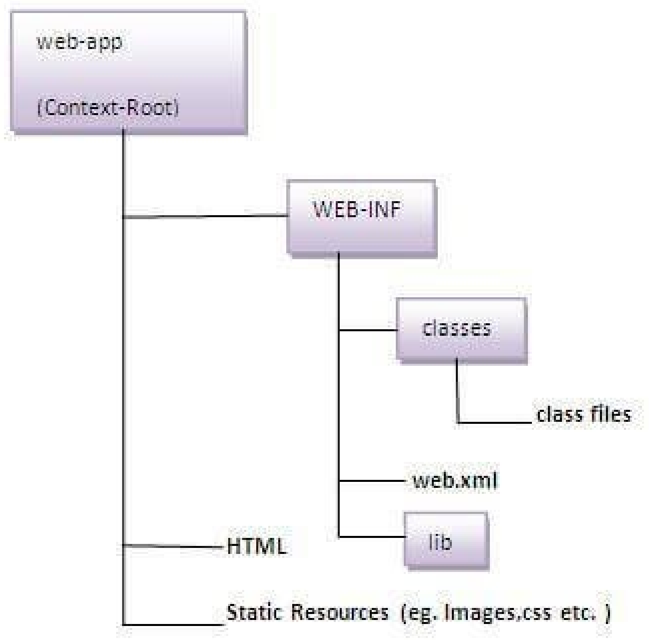
Content Types/ MIME Types:

Content Type is also known as **MIME (Multipurpose internet Mail Extension)**Type. It is a **HTTP header** that provides the description about what are you sending to the browser.

- text/html
- text/plain
- application/msword
- application/vnd.ms-excel
- application/jar
- application/pdf
- application/octet-stream
- application/x-zip
- images/jpeg
- images/png
- images/gif
- audio/mp3
- video/mp4
- video/quicktime etc.

Steps for creating a Servlet:

1. Create a directory structure



2. Create a Servlet

   - ✓ By implementing the Servlet interface
   - ✓ By inheriting the GenericServlet class
   - ✓ By inheriting the HttpServlet class

3. Compile the Servlet

For compiling the Servlet, jar file is required to be loaded. Different Servers provide different jar files:

Jar file	Server
1) servlet-api.jar	Apache Tomcat
2) weblogic.jar	Weblogic
3) javaee.jar	Glassfish
4) javaee.jar	JBoss

Two ways to load the jar file

1. set classpath
2. paste the jar file in JRE/lib/ext folder

Put the java file in any folder. After compiling the java file, paste the class file of servlet in **WEB-INF/classes** directory.

4. Create a deployment descriptor

   web.xml file

   There are too many elements in the web.xml file. Here is the illustration of some elements that is used in the above web.xml file. The elements are as follows:

   - ✓ <web-app>
     represents the whole application.
   - ✓ <servlet>
     is sub element of <web-app> and represents the servlet.
   - ✓ <servlet-name>
     is sub element of <servlet> represents the name of the servlet.
   - ✓ <servlet-class>
     is sub element of <servlet> represents the class of the servlet.
   - ✓ <servlet-mapping>
     is sub element of <web-app>. It is used to map the servlet.
   - ✓ <url-pattern>
     is sub element of <servlet-mapping>. This pattern is used at client side to invoke the servlet.
   - ✓ <welcome-file-list> is used to define a list of welcome files.

5. Start the server and deploy the project
6. Access the servlet