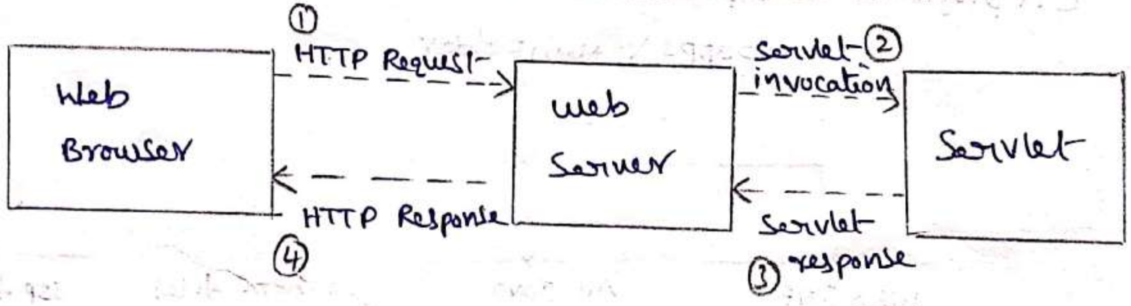
* Introduction to servlets :-

↳ A Servlet is a Java program, that was executed by the webServer. The Servlet is used to execute the process at the server side.

↳ A servlet can be thought of as a server side applet.

↳ Servlet accepts a request from a client, performs some task and returns results to client (browser).

↳ The Servlet runs on the server and will respond to a request from the client either in the form of HTML pages.



To develop and run servlets we need following :

→ Java Development kit (JDK) installation

→ Java Servlet Development kit (JSDK) installation

→ A web Server

→ A client application (generally a web browser).

↳ Basic Servlet Structure :-

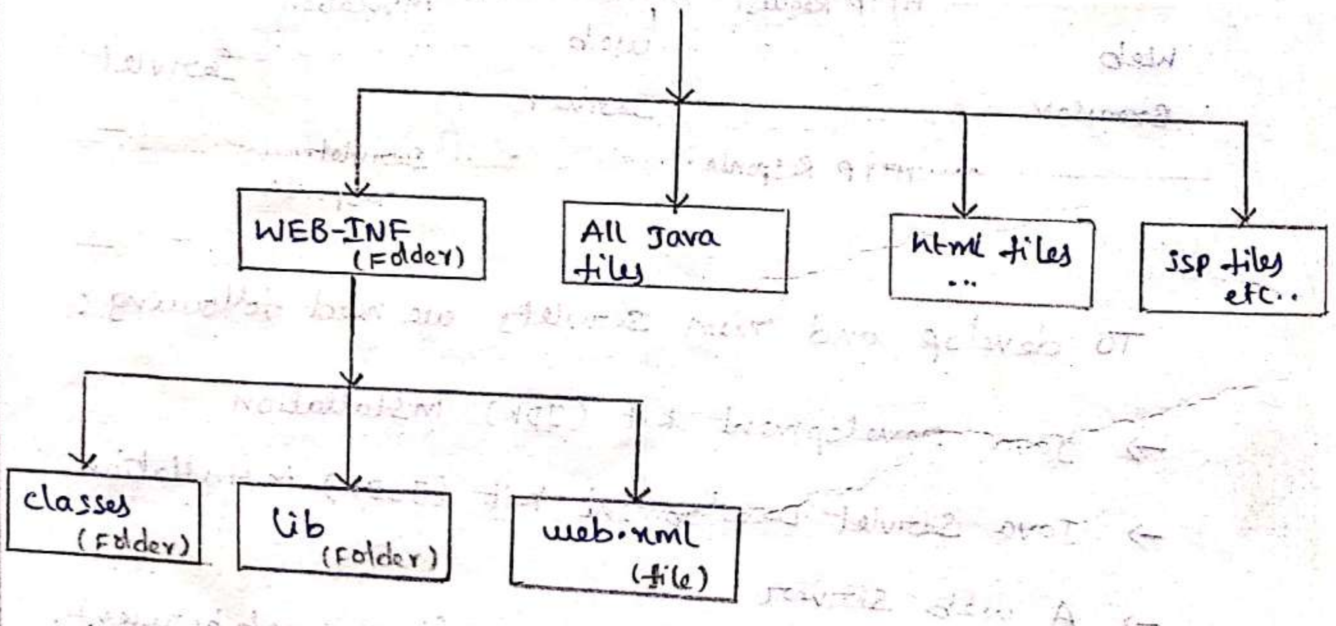
```
    import Java.io.*;
    import javax.Servlet.*;
    import javax.Servlet.http.*;
```

Contd..

```
public class ServletTemplate extends HttpServlet
{
public void doGet(HttpServletRequest req,
                  HttpServletResponse res) throws
          ServletException, IOException
 {
    printWriter out = res.getWriter();
 }
}
```

↳ Folder Structure :-

C:\ program Files \ Apache Software Foundation \ Tomcat 5.5 \
webapps \ New Folder



↳ Servlet Container :-

A servlet container is a specialized web server that supports servlet execution. The servlet container have ability to dynamically add and remove servlets from the system.

Example

* write a program to print "Hello world" using Servlets.

```
import java.io.*;
import javax.Servlet.*;
import javax.servlet.http.*;
public class Hello extents HttpServlet
{
 public void doGet (HttpServletRequest req,
                    HttpServletResponse res) throws
          ServletException, IOException
  {
    printwriter pw = res.getwriter ();
        pw.println ("Helloworld");
  }
}
```

* Life cycle of a servlet :-

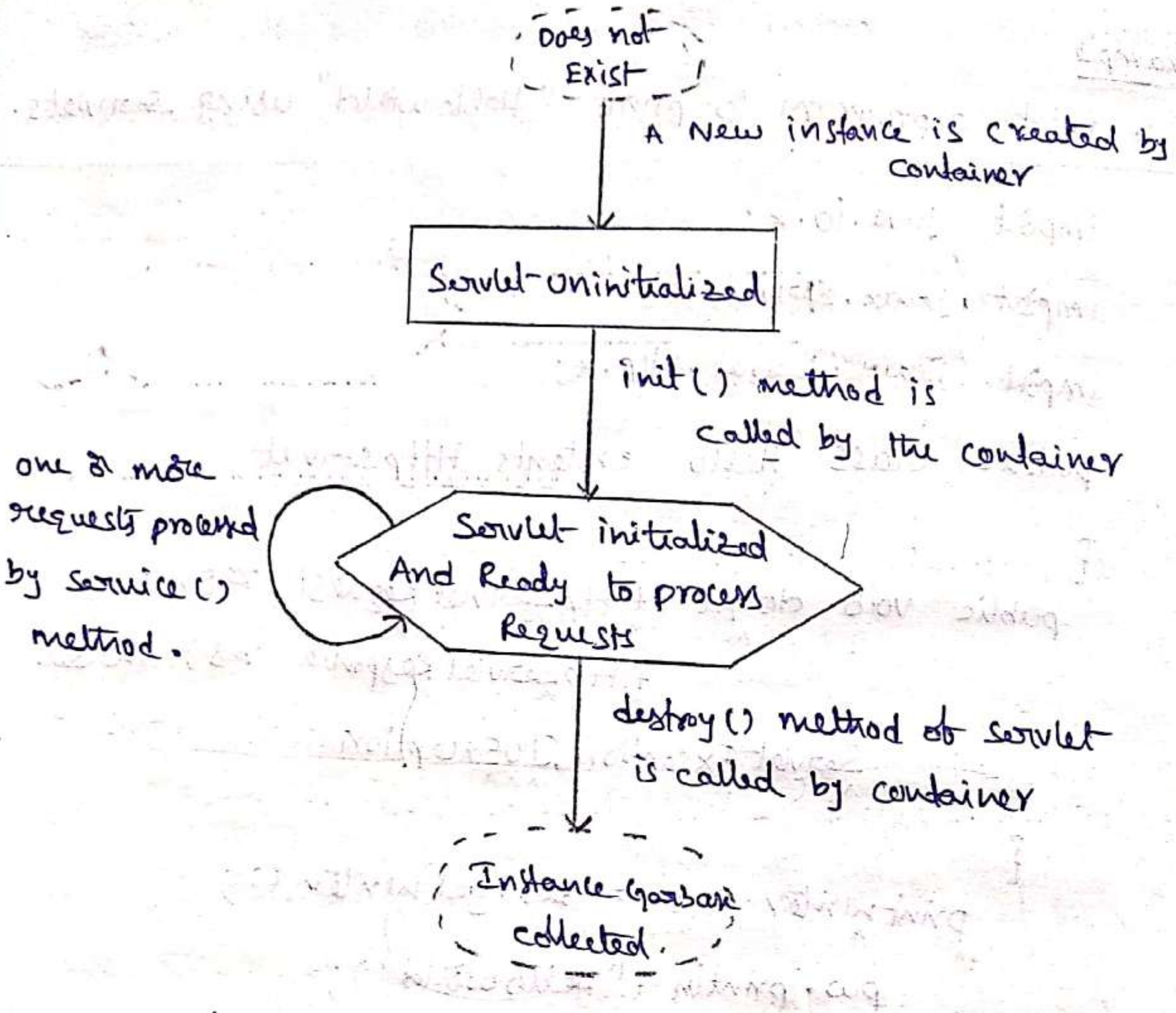
↳ Lifecycle of a servlet describes how and when a servlet is loaded, initialized, able to handle requests and unloaded (destroyed).

There are three methods in lifecycle of servlet those are

→ init() :- This method initializes the servlet

→ service() :- This method process the request (Http).

→ destroy() :- This method is used to destroy The servlet.



Does not Exist
A New instance is created by container
Servlet uninitialized
init() method is called by the container
one or more requests processed by service() method.
Servlet initialized And Ready to process Requests
destroy() method of servlet is called by container
Instance garbage collected.

Fig: Lifecycle of a servlet.

The servlet lifecycle consists of following steps

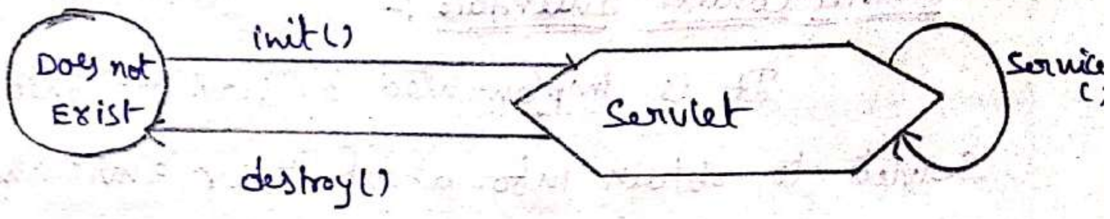
→ The servlet class is loaded by the container during start-up or the first time it is accessed.

→ The container calls the init() method. This method initialized the servlet and the init() method is called only once.

→ After initialization, the client requests can be processed by the calling of service() method. The container can call the service() method for every request.

→ And finally if the servlet is not needed more, the container calls the destroy() method to stop the service, And this metho is also called only once in the life cycle of servlet.

the simplified life cycle of servlet



---

* The Servlet API :-

The servlet API consists of classes and interfaces needed to build servlets. These classes and interfaces comes in two packages, those are:

→ javax.servlet package

→ javax.servlet.http package

*↳ Javax.servlet package :-

The Javax.servlet package is at the core of all servlet development. This package contains a number of interfaces and classes that establish the framework in which servlets operate.

The following are the core interfaces that are provided in this package.

→ Servlet Interface :-

It declares lifecycle method for a servlet, All servlet must implement servlet interface.

→ Servlet Config Interface :-

It is Implemented by servlet container. It allows a servlet to obtain config data when it is loaded.

It allows Servlets to get initialization parameters.

→ Servlet Context Interface :-

It is implemented by servlet container. It enables Servlets to obtain info. about their enviornment.

→ ServletRequest Interface :-

It is implemented by Servlet container. It is used to read data from a client request.

→ Servlet Response Interface :-

It is implemented by servlet container, It is used to write data to a client response.

The following are core classes that are provided in javax. servlet package

→ Generic Servlet class :-

The GenericServlet class implements servlet & Servlet Config interfaces. This method provides basic implementation for servlet lifecycle methods init(), service() & destroy().

→ ServletInputStream class :-

This class entends input stream. It provides an input stream that a Servlet developer can use to read data from a client request.

→ Servlet output Stream class :-

This class entends outputStream. It is implemented by Servlet container and provides an output stream that a servlet developer can use to write data to a client response.

→ Servlet Exception class :-

This class indicates that a Servlet problem has occurred.

* ↳ javax.servlet.http package :-

The javax.servlet.http package contains interfaces and classes that simplify the process of writing servlets that use the HTTP protocol. This package defines and implements much of functionality required to communicate via HTTP.

The following are interfaces thatare provided in Javax.Servlet.http package.

→ HttpServletRequest Interface :-

This interface enables servlets to read data from a HTTP request. It defines an object that provides the http servlet with service() method.

→ HttpSession Interface :-

The HttpSession interface defines an object that provides an association between a client and server persisting over multiple connections.

→ HttpServletResponse Interface :-

It enables servlets to write data to a HttpResponse. It defines an object that provides the httpServlet's service() method to return the data to client.

The following are classes that are provided in javax.servlet.http package.

Scanned by CamScanner

→ Cookie class :-

It allows to store the state information on a client machine. once a Cookie object is created, it is passed to the client using the HttpServletResponse object's addCookie() method. Important methods are getName() and getpath().

→ Http Servlet class :-

It provides methods to handle Httprequest and responses. The important methods are doDelete(), doGet(), doOption(), dopost() etc.

→ HttpSession Event Binding :-

It extends Http SessionEvent. It indicates when a listener is bound to or unbound from a session value or that a session attribute changed.

→ Http Utils :-

This class provides useful collection of HttpUtility methods. The important methods of the class are parsepostData() and getRequestURL().

---

* Reading Servlet parameters :-

The ServletRequest class includes methods that allow you to read the names and values of parameters that are included in a client request.

The following example contains two files.

A webpage is defined in login.html and a servlet is defined in param.java.

Login.html

```
<html>
 <body>
  <form name="form1" action="./param"
                          method = post >
   <b> Enter userName </b>
    <input type="text" name="uname" />
   <br/>
    <b> Enter password </b>
    <input type="password" name='pwd" /> </form>
    </body>    <input type="submit" value="Submit"/>
    </html>
```

param.java

```
import java.io.*;
import java.Util.*;
import javax.Servlet.*;
import javax.Servlet.http.*;
public class param extends GenericServlet
 {
 public void Service(ServletRequest req,
                     ServletResponse res) throws
           ServletException, IOException
 {
    res.setContentType("text/html");
    PrintWriter pw = res.getWriter();
   String name= req.getParameter("uname");
   String pass = req.getParameter("pwd");
```

```
    pw.println("username is" + name + "<br>");
    pw.println("password is" + pass);
  }
}
```

In the above program getparameter() method is used to read the content on the client request.

---

* <u>Handling Http Requests and Responses</u> :-

doGet() and dopost() methods are identical that can replace The Service() method.

The doGet() method handles GET requests (of HTML) and dopost() method handles POST requests (of HTML).

<u>Handling GET Request</u>

The doGet() method is used to handling GET requests.

The following example contains two files, ChooseColor.html defines a web page containing colors. The second file print-Color.java is a servlet that prints the color, reading it from the HTML file.

<u>ChooseColor.html</u>

```
<html>
  <body>
   <form method="get" action="./print-color">
    <b> color: </b>
     <select name="clr" >
       <option> Red </option>
    <option> Green </option>
    <option> Blue </option>
  </select>
  <br> <br>
 <input type="submit" value="choose" />
 </form>
 </body>
</html>
```

printcolor.java

```
    import java.io.*;
    import javax.servlet.*;
    import javax.servlet.http.*;
    public class printcolor entends HttpServlet
    {
 public void doGet(HttpServletRequest req,
                   HttpServletResponse res) throws
               ServletException, IOException
   {
    String c = req.getparameter("clr");
    res.setContentType("text/html");
    printwriter pw = res.getwriter();
     pw.println("<b> The selected color is</b>");
     pw.println(c);
     pw.close();
    }
 }
```

Handling POST Request

The dopost() method is used to handling POST requests.

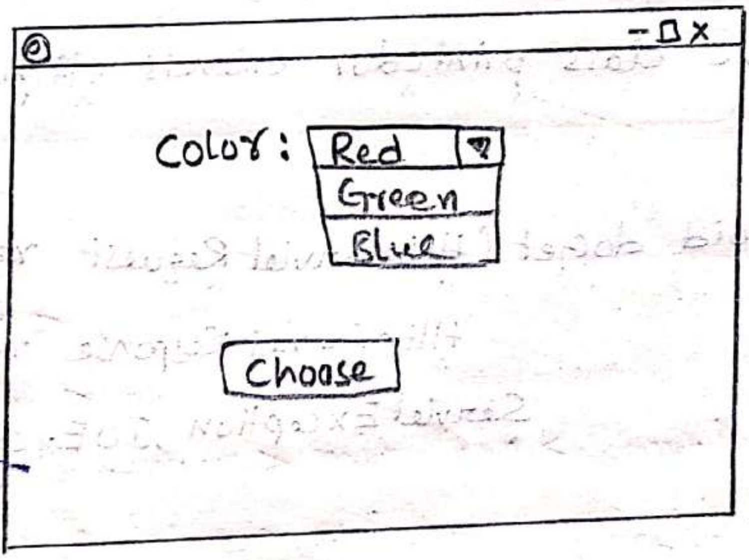
In the FORM tag, if Method is GET, we must use doGet() method and if method is POST, we must use dopost() method.

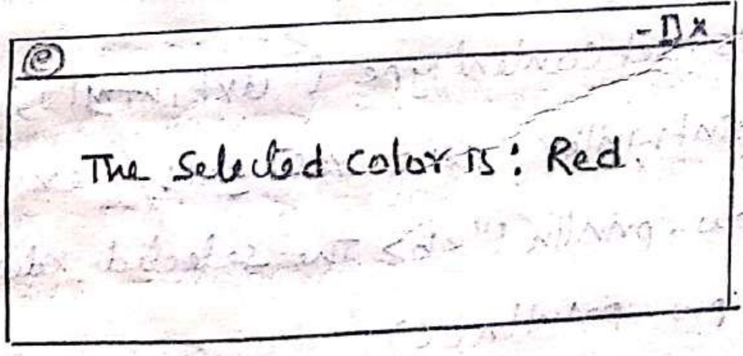
In the above program change the following

→ write method=POST instead of method = GET

→ write dopost() instead of doGet().

o/p:-





The Selected color is: Red.

Example :-

* Create a HTML form with three input fields first name, last name and e-mail. pass these values to a Servlet. In the Servlet, verify all input fields are not null and display them back to client.

one.html

```
<html>
  <body>
    <form action = "./details" method = "get" >
FirstName :<input type="text" value="" name="fname"/>
      <br/>
LastName :<input type="text" name = "lname" /> <br/>
E-mail : <input type = "text" name="email"/> <br/>
   <input type="submit" value = "submit"/>
  </form>
  </body>
  </html>
```

Valid.java

```
import javax.servlet.*;
import javax.servlet.http.*;
public class Valid extends HttpServlet
{
public void service(HttpServletRequest req,
                    HttpServletResponse res) throws
      ServletException, IOException
 {
   PrintWriter out =res.getWriter();
  String fn = req.getparameter ("fname");
String ln = req.getparameter("lname");
String em = req.getparameter("email");
if(fn.equals(" ") || ln.equals(" ") || em.equals(" "))
 {
   out.println("please enter a value to all fields");
 }
else
 { out.println("<b> your details are </b>");
   out.println(fn);
   out.println(ln);
   out.println(em);
 }
}
}
```

web.xml

```
<web-app>
  <Servlet>
    <Servlet-name> validation </Servlet-name>
    <Servlet-class> Valid</Servlet-Class>
  </Servlet>
  <Servlet-mapping>
    <Servlet-name> validation </servlet-name>
    <url-pattern>/details </url-pattern>
  </servlet-mapping>
</web-app>.
```

To deploy any Servlet Application, we need to follow following procedure

→ If we are using Tomcat webserver, set the classpath to Servlet-api.jar.

```
C:\> Set classpath= C:\program Files\Apache Software
        Foundation\Tomcat 6.0\lib\Servlet-api.jar
```

→ Next, Compile the .java file, then we got .class file that can be copied to "Tomcat6.0\web-apps\classes".

→ Next, we can write web.xml file as shown in above.