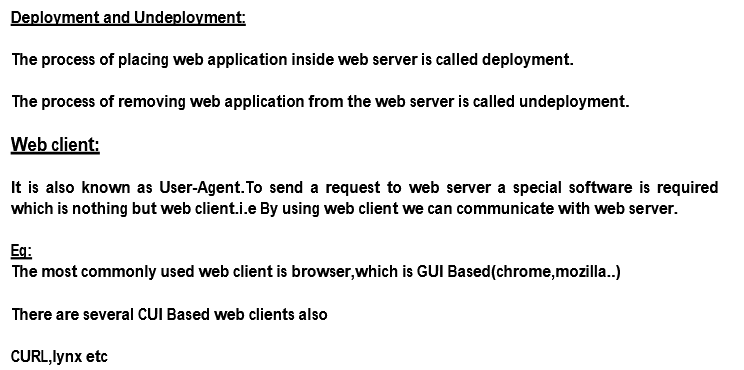
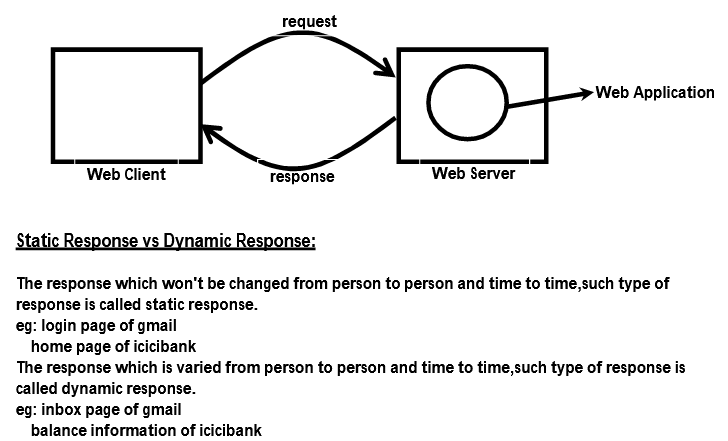
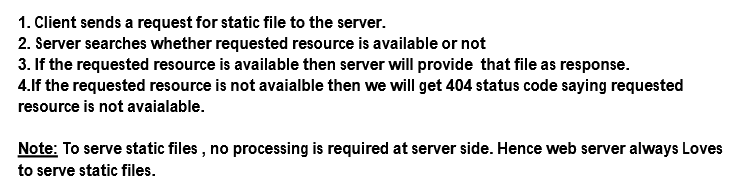
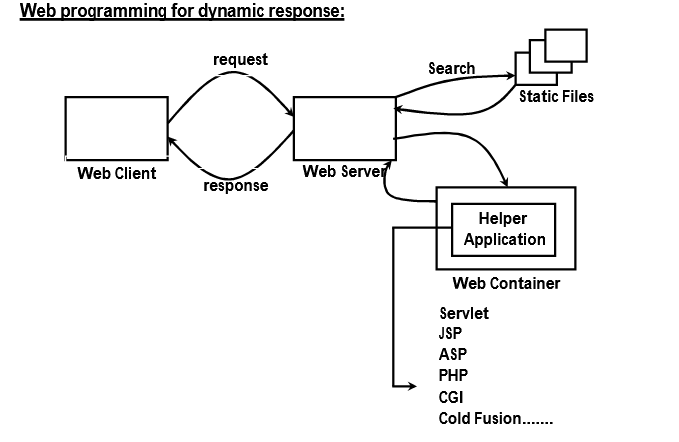
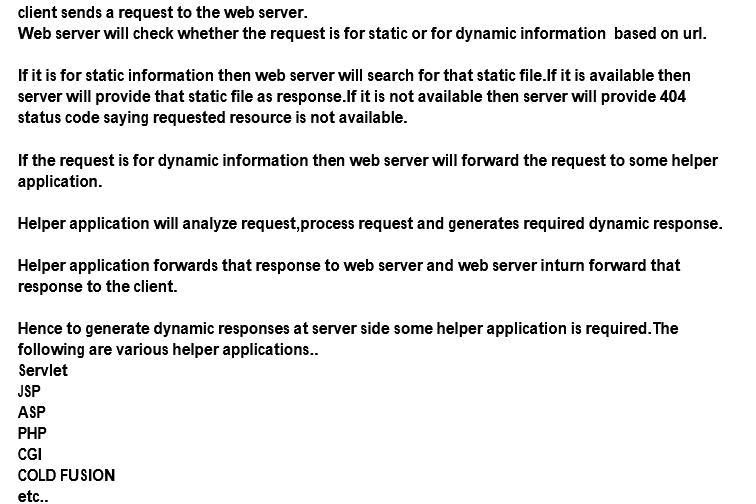


****

****

****

****

****

**Servlets:**

**Servlets are Java classes that run entirely on**

**(and/or within) an HTTP (web) server.**

Simply put, a Servlet is a class that handles requests, processes them and reply back with a response.

For example, we can use a Servlet to collect input from a user through an HTML form, query records from a database, and create web pages dynamically.

Servlets are under the control of another Java application called a **Servlet Container.**When an application running in a web server receives a request, the Server hands the request to the Servlet Container – which in turn passes it to the target Servlet.

**Servlet Container:**

**It is also known as Servlet Engine.**

**It is responsible to manage and execute Servlet components.**

**In Tomcat servlet container's name is CATALINA**

Servlets are useful for running code on the

server as a result of client interaction with a

web page. For example, if a client submits

information in a web page form, the servlet can

process the information, add it to a database,

and send back a customized, confirmation

response page

**SERVER-SIDE JAVA:**

Like applets, servlets are also Java classes.

Unlike applets, which are used to implement client-side of a client–server application, servlets are used at

the server-side.

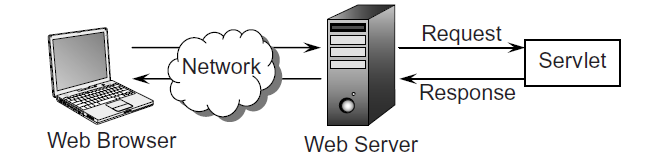
They usually run inside a Java-enabled web server and extend its capabilities.

it has become the most popular technology **for building interactive and dynamic web applications.**

servlets are most commonly used with HTTP. That is why, the word *servlet* is often used to mean *HTTP servlet*

Execution of a servlet consists of four basic steps and is shown in the following

* The client sends a request to the web server.
* The web server interprets it and forwards it to the corresponding servlet.
* The servlet processes the request, generates the output (if any), and sends it back to the web server.
* The web server sends the response back to the client. The browser then displays it on the screen.



**(packages--🡪Interfaces---🡪classes-🡪methods)**

**SERVLET ARCHITECTURE:**

Servlet architecture consists of two packages:

**1. javax.servlet package**

**2. javax.servlet.http package**

**1. javax.servlet:**

**This package defines several classes and interfaces used for developing protocol independent servlets(Generic Servlets).** It contains top-level interfaces and classes that are used and extended by all other servlets either directly or indirectly

**2. javax.servlet.http :**

**This package defines several classes and interfaces which can be used to develop Http protocol based servlets. So** The second package is provided for servlets that can handle HTTP requests.

**Important interfaces of javax.servlet package:**

**1.Servlet**

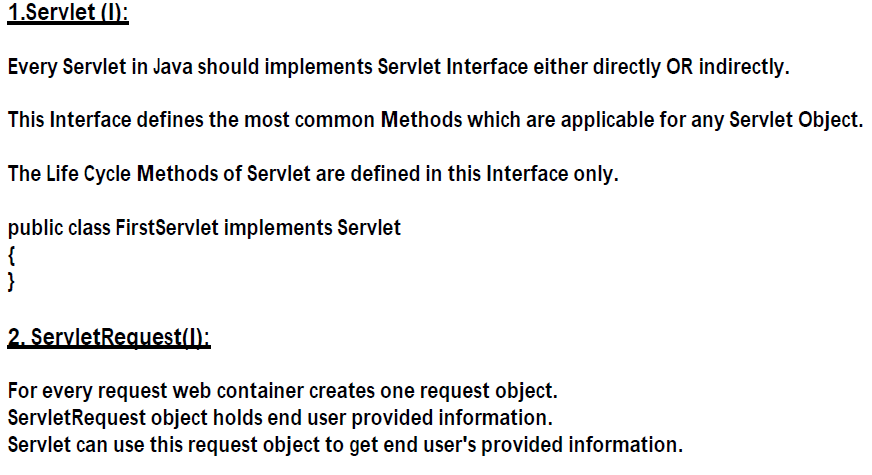
**2.ServletRequest**

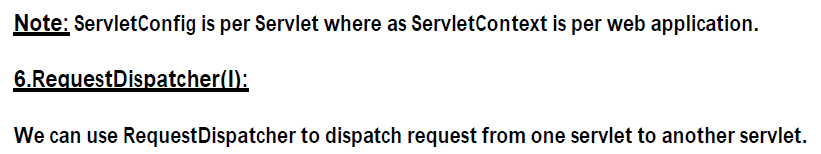
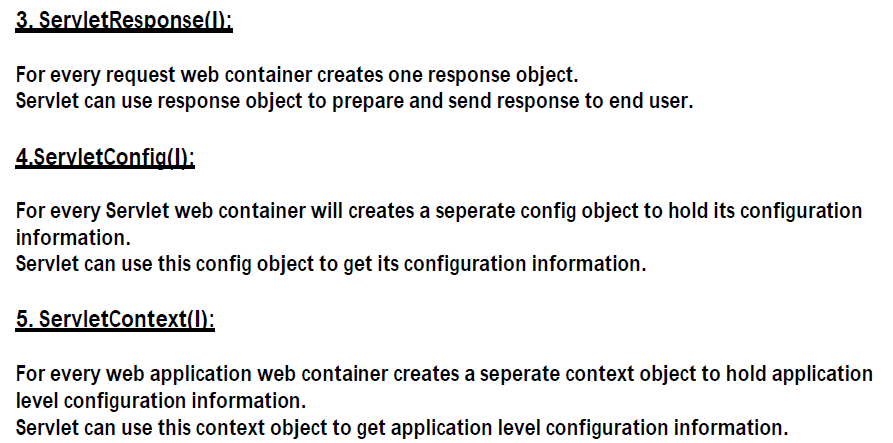
**3.ServletResponse**

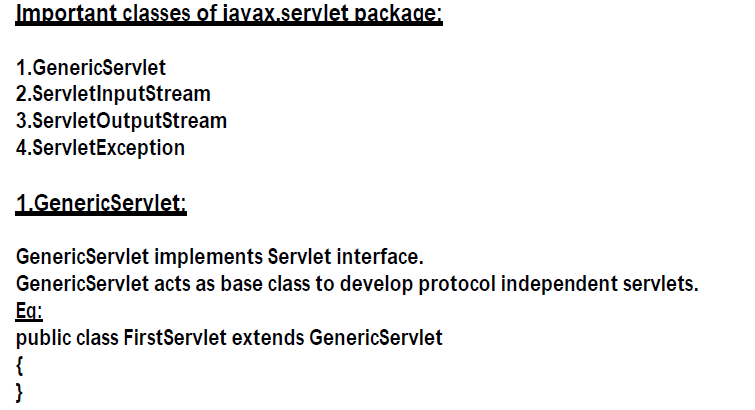
**4.ServletConfig**

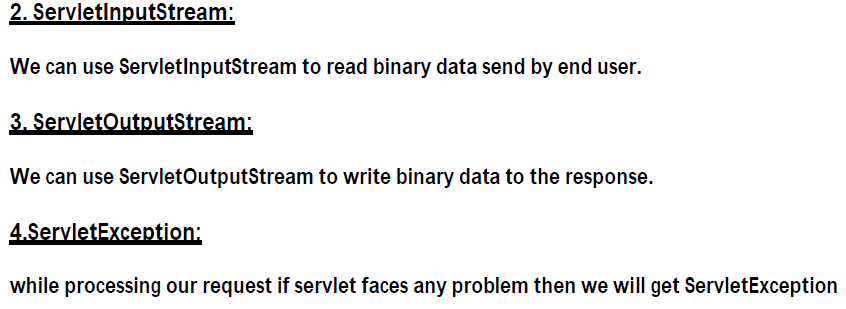
**5.ServletContext**

**6.RequestDispatcher**

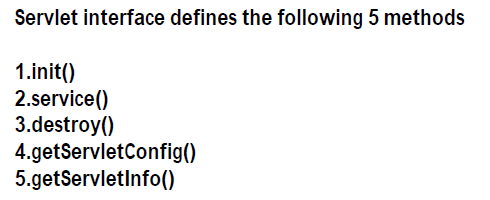


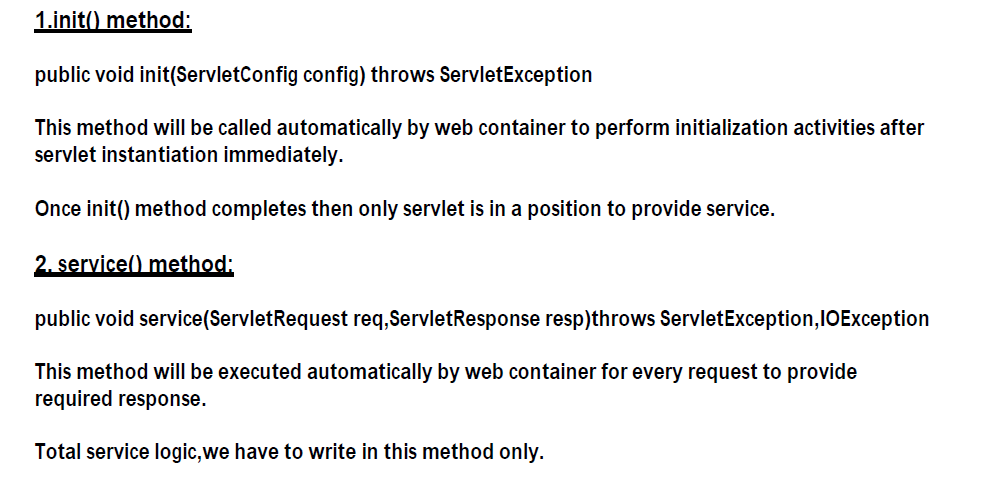


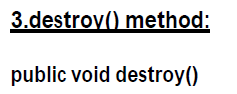


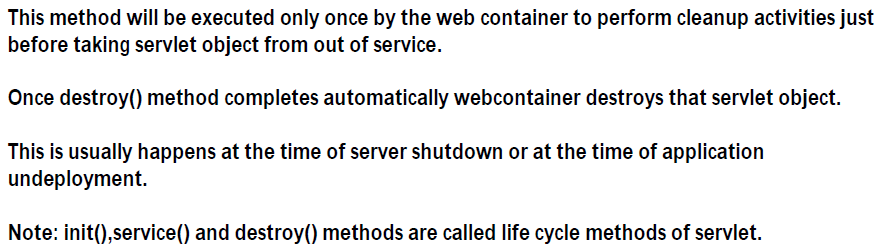


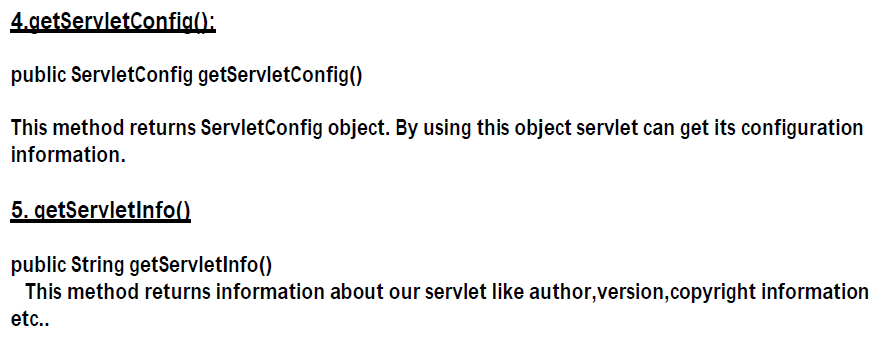
Servlet Interface methods:

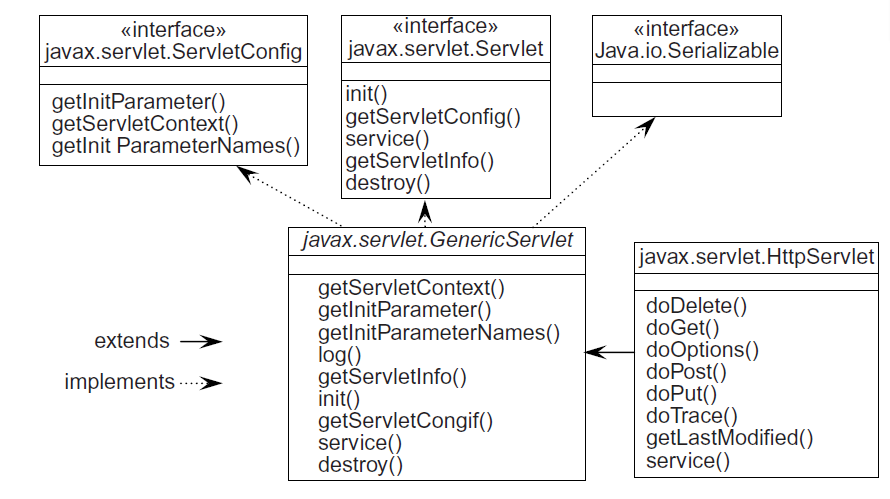










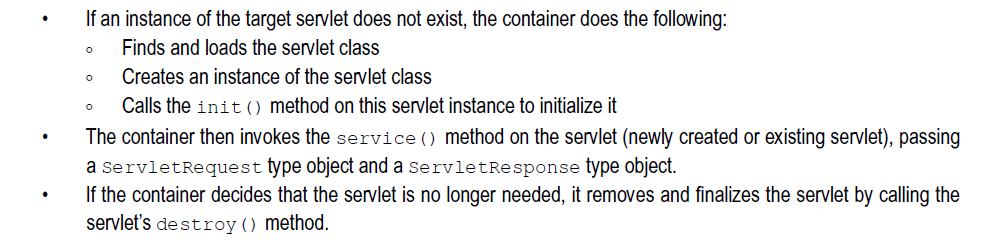


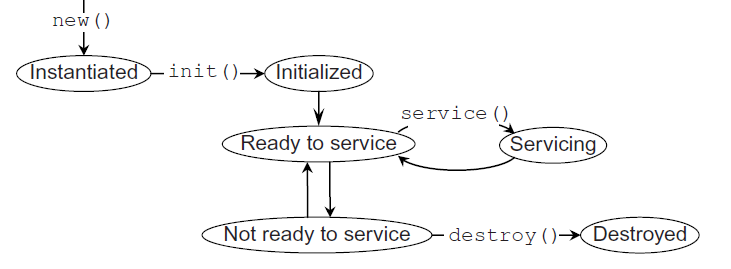
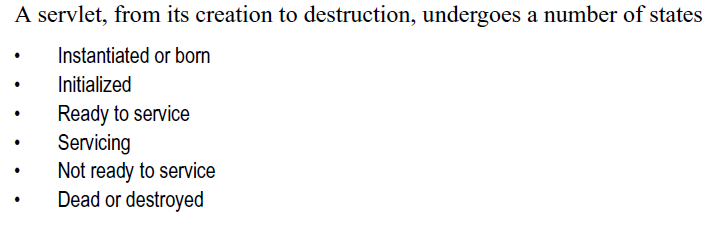
**SERVLET LIFE CYCLE:**

the servlet will be deployed to a container. this container is nothing but a web server

It supervises and controls the life cycle of a servlet.

When the container receives a request from a client (and determines that the request should be handled by a servlet,) it performs the following steps

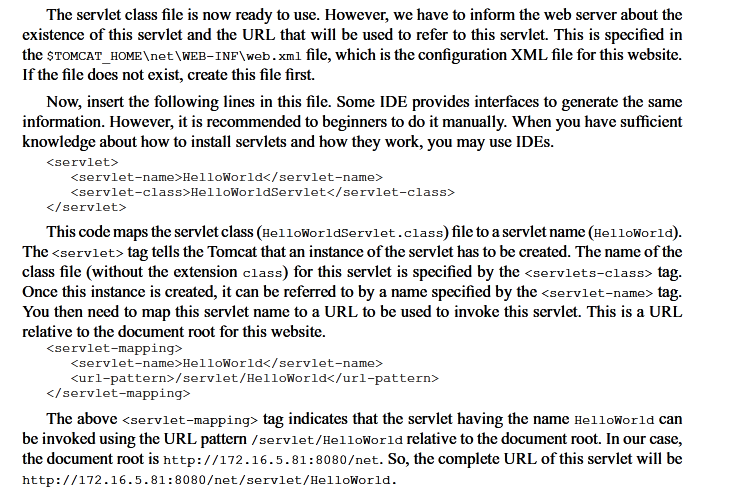


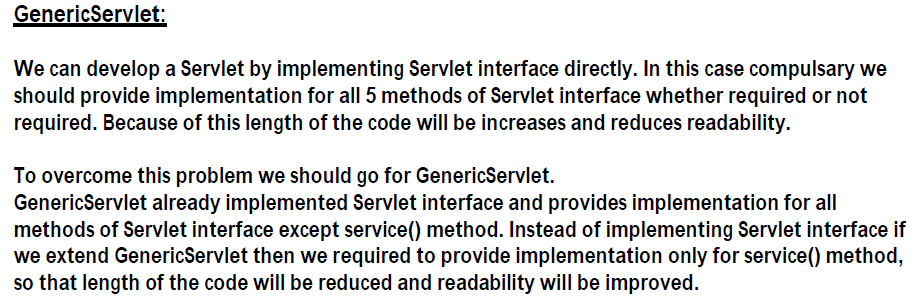


The entire period when a servlet remains alive is called *life cycle* of the servlet. The class Servlet

provides a framework where our servlets can run. It has a set of methods to control and supervise

the smooth execution of servlets





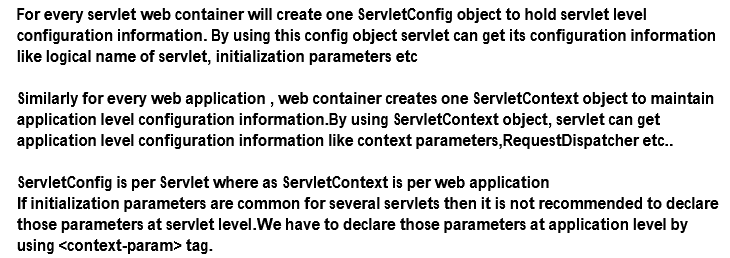
Displaying index.html when application started running

There are 2 ways

Web.xml

|  |  |
| --- | --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"* id=*"WebApp\_ID"* version=*"4.0"*>  <!—first way🡪   |  | | --- | | **<display-name>servlet4</display-name>**  **<welcome-file-list>**  **<welcome-file>index.html</welcome-file>**    **</welcome-file-list>** |       <!-- <servlet>  <servlet-name>first</servlet-name>  <servlet-class>servlet4.HomeController</servlet-class>  </servlet>    <servlet-mapping>  <servlet-name>first</servlet-name>  <url-pattern></url-pattern>      </servlet-mapping>    -->  </web-app> |

|  |  |
| --- | --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"* id=*"WebApp\_ID"* version=*"4.0"*>  <display-name>servlet4</display-name>  <!-- <welcome-file-list>  <welcome-file>index.html</welcome-file>    </welcome-file-list>    -->  <!—second way using Homecontroller🡪   |  | | --- | | **<servlet>**  **<servlet-name>first</servlet-name>**  **<servlet-class>servlet4.HomeController</servlet-class>**  **</servlet>**    **<servlet-mapping>**  **<servlet-name>first</servlet-name>**  **<url-pattern></url-pattern>**      **</servlet-mapping>** |         </web-app> |



If your application has two databases, which database needs to deal with. This can be done with servletconfig. After getting values, create DaoImpl class object.

