Assignment No. 1

Page No.
Date: / /

Q.1) What is servlet? Explain the advantage of the servlet.

Ans:- Servlet:-

1) Servlet is a technology i.e; used to create web application.
2) Servlet is an API that provides many interfaces and classes including documentations.
3) Servlet is an interface that must be implemented for creating any servlet.
4) Servlet is a web component that is deployed on the server to create dynamic web page.

- Advantages of the servlet.

1) Better performance:-
Because it creates a thread for each request not process (like CGI).

2) Portability:-
Because it uses java language and java is robust language.

3) Robust:-
Servlet are managed by JVM so no need to worry about memory leak, garbage collection etc

4) Secure:-
Because it uses java language and java is a secure language. Java have automatic garbage collection mechanism and a lack of

pointers protect the servlets from memory management problems.

5) Inexpensive:-

There are number of free web servers available for personal use or for commercial purpose. Mostly web server are very costly. So by using free web server you can reduce project development price.

6) Efficiency:-

Servlets invocation is highly efficient as compared to any CGI programs.

7) Integration:-

Servlets are tightly integrated with the server. Servlet can use the server to translate the file path, check authorization, perform logging and MIME type mapping etc.

8) Fast:-

Since servlets are compiled into bytecodes, they can execute more quickly a compared to other scripting language. The bytecode compilation feature helps servlets to give much better performance.

Q.2) Explain generic servlet:-

Ans:- Generic servlet class implements servlet, ServletConfig, Serializable interfaces. Generic servlet is a base class of

servlet. It provides the implementation of all the methods of these interfaces except the service method. Generic servlet class can handle any type of request so it is protocol independent.

- Methods of Generic servlet class

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 void service(ServletRequest req, ServletResponse res) 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 ServletContext getServletContext() returns the parameter object of servlet context.

7) public String getServletName() returns the name of the servlet object.

Q.3) Explain the life cycle of the servlet.

Ans:- Servlets life cycle involve three important methods are:

1) init():-
Init() method is called when servlet first loaded into the web server memory.

2) Service():-
Once initialized, servlets stays in memory to process requests servlets read the data provided in the request in the service() method.

3) destroy():-
When server unbads servlets, destroy() method is called to clean up resources the servlet is consuming.

- Servlet life cycle divided into four parts:-

1) Loading and instantiation:-
In this the web container first loads the servlet class. The web container then creates an instance of this servlet. Loading of servlet is done during the startup when the first request is made, whereas the creation of instance can occure at startup or can be delayed till service to a request is required by the servlet.

2) Initialization :-

After the creation of an instance the init() method is called by the servlet to initialize the servlet instance. The init() method must be called before any of the request are serviced by the servlet. The parameters of initialization are passed to init() and persist till servlet is destroyed. If loaded successfully the servlet is available for service, if not then the servlet is unloaded by the servlet container.

3) servicing requests :-

After initialization the servlet can be used to handle client requests. Requests are represented by request objects of type servletRequest. A separate thread is created for each request.

4) Destroying the service :-

When a servlet is no longer required the destroy() method () is called by servlet container. This method removes the servlet.

Q.4) What is the use of the request dispatcher. Explain include and forward method.

Ans :- Request dispatcher is an interface that provides the facility the to forward a request to another resource or include the content of another resource - Request Dispatcher provides a way to call another resources from a servlet. Another resource can be servlet jsp or html.

- Methods of RequestDispatcher interface :-

1) forward (ServletRequest req, ServletResponse res) :- This method forwards a request from a servlet to another resource on the server.

2) include (ServletRequest req, ServletResponse res) :- This method includes the content of a resource in the response.

- Example :-

In this example, using include method. I will be changing the content of current page and when I'm ready to transfer the control to the next page. I will use forward method.

1) index.html

```
<form action="loginPage" method="post">
username :<input type="text" name="uname"/><br>
Password :<input type="password" name="upasr"/><br>
<input type="submit" value="SUBMIT"/>
</form>
```

2) MyServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class MyServlet extends HttpServlet
{
public void service (HttpServletRequest request,
```

HttpServletResponse response) throws SE, IOE
{

response.SetContentType("text/html");
print writer out= response.getWriter();
String name= request.getParameter("uname");
String pass = request.getParameter("upass");
if (name.equals("pakki") && pass.equals("gosavi")
{

RequestDispatcher dis=request.getRequestDispatcher
("welcome");
dis.forward(request, response);
}
else
{

out.println("Username or password is incorrect");
RequestDispatcher dis = request.getRequestDispatcher
("index.html");
dis.include(request, response);
}
}
}

3) NewServlet.java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class NewServlet extends HttpServlet
{

public void service(HttpServletRequest request,
HttpServletResponse response) throws SE, IOE
{

```
response.setContentType("text/html");
printwriter out = response.getWriter();
String name = request.getParameter("uname");
out.println("Hello "+name+"!,");
out.println("Welcome to srm college");
}
}
```

Q.5) What are the ways to track session explain in detail.

Ans:-

Following are the popular ways of session tracking:

1) URL rewriting :-

In this method of session tracking, some extra data is appended at the end of the URL, which identifies the session. This method is used for those browsers which do not support cookies or when the cookies are disabled by the user.

Original URL:-

http//server:port/servlet/ServletName

Rewritten URL:-

http://server:port/servlet/ServletName?session-id=7556

2) Hidden Form Fields:-

<input type="hidden" name="technology"

value = "servlet" >

Hidden fields like the above can the inserted in the webpages and information can be sent to the server for session tracking. These fields are not visible directly to the user, but can be viewed using view source option from the browsers. This type doesn't need any special configuration from the browser of server and by default available to use for session tracking.

3) Cookies :-

Cookies are the mostly used technology for session tracking.

Q.6) What is Filter ? How can we implemented?

Ans:-

A Filter is an object that is invoked at the preprocessing and post processing of a request. It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc.

The servlet Filter is pluggable. i.e. it's entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet.

We can define filter same as servlet.

For example:

```
<web-app>
 <filter>
<filter-name>.....</filter-name>
<filter-class>.....</filter-class>
 </filter>

 <filter-mapping>
<filter-name>.....</filter-name>
<url-pattern>.....</url-pattern>
 </filter-mapping>
 </web-app>
```

For mapping filter we can use, either url-pattern or servlet-name. The url-pattern elements has an advantage over servlet-name element i.e., it can be applied on servlet, jsp or HTML.