ASSIGNMENT

Name: Umesh Khatiwada

Subject: Java

Grade :7th sem BSC-csit

1.What is meant by web container and deployment descriptor? Explain work done by web container briefly.

Web container:

Is a component of a web server that interacts with Java servlets.

Implements the web component aspect of the Java engineering architecture

creates servlet instances, loads and unloads servlets, creates and manages request and response objects, and performs other servlet-management tasks.

Deployment descriptor:

A deployment descriptor is a configuration file for a web application or EJB application which is to be deployed to web or EJB container.

The deployment descriptor should contain standard structural information for all enterprise beans in an EJB application.

Java web application use a deployment descriptor file to determine how URLs map to servlets.

For web applications, the deployment descriptor must be called web.xml and must reside the WEB-INF directory in the web application root.

Here is a simple web.xml example that maps all URL paths (/*) to the servlet

class

mysite.server.ComingSoonServlet:

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"

version="3.1">

<servlet>

<servlet-name>comingsoon</servlet-name>

<servlet-class>mysite.server.ComingSoonServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>comingsoon</servlet-name>

<url-pattern>/*</url-pattern>

</servlet-mapping>

</web-app>

2.Explain the steps used in writing servlets with suitable source code and bried description of each step?

There are three ways to create the servlet.

By implementing the Servlet interface

By inheriting the GenericServlet class

By inheriting the HttpServlet class

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
public class DemoServlet extends HttpServlet{
public void doGet(HttpServletRequest req,HttpServletResponse res)
throws ServletException,IOException
{
res.setContentType("text/html"); //setting the content type
PrintWriter pw=res.getWriter(); //get the stream to write the data
//writing html in the stream
pw.println("<html><body>");
pw.println("Welcome to servlet");
pw.println("</body></html>");
pw.close();//closing the stream
}}
```

Compile the servlet

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

In Apache Tomcat server servlet-api.jar file is required to compile a servlet class.

Create the deployment descriptor (web.xml file)

<web-app>

<servlet>

<servlet-name>sonoojaiswal</servlet-name>

<servlet-class>DemoServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>sonoojaiswal</servlet-name>

<url-pattern>/welcome</url-pattern>

</servlet-mapping>

</web-app>

Start the Server and deploy the project

To start Apache Tomcat server, double click on the startup.bat file under apache-tomcat/bin directory.

One Time Configuration for Apache Tomcat Server

You need to perform 2 tasks:

1.

set JAVA_HOME or JRE_HOME in environment variable (It is required to start server).

2.

Change the port number of tomcat (optional). It is required if another server is running on same port (8080).

3.What is cookie? How can you store and read cookies by using servlet? Explain with example.

Cookies are messages that web servers pass to your web browser when you visit Internet sites. Your browser stores each message in a small file, called cookie.txt

These files typically contain information about your visit to the web page, as well as any information you've volunteered, such as your name and interests.
There are three steps involved in identifying returning users −

Server script sends a set of cookies to the browser. For example name, age, or identification number etc.

Browser stores this information on local machine for future use.

When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.
Example:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloForm extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
// Create cookie
Cookie myCookie = new Cookie("my_cookie", request.getParameter("my_cookie"));
// Set expiry date after 24 Hrs
my_cookie.setMaxAge(60*60*24);
```

```
// Adding cookie in the response header.

response.addCookie(my_cookie);

// Set response content type

response.setContentType("text/html");

PrintWriter out = response.getWriter();

String title = "Storing Cookie Example";

String docType ="<!doctype html public \"-//w3c//dtd html 4.0 " + "transitional//en\">\n";

out.println(docType +

"<html>\n" +

"<head>

<title>" + title + "</title>

</head>\n" +

"<body bgcolor = \"#f0f0f0\">\n" +

"<h1 align = \"center\">" + request.getParameter("my_cookie")+ "</h1>\n" +

"</body>

</html>"

);

} }
```

Similarly we can read cookies by calling the getCookies() method of HttpServletRequest. Then cycle through the array, and use getName() and getValue() methods to access each cookie and associated value.

Example:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ReadCookies extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
Cookie[] cookies = null;
// Get an array of Cookies associated with this domain
cookies = request.getCookies();
// Set response content type
response.setContentType("text/html");
PrintWriter out = response.getWriter();
String title = "Reading Cookies Example";
String docType =
"<!doctype html public \"-//w3c//dtd html 4.0 " +
"transitional//en\">\n";
out.println(docType +
"<html>\n" +
"<head><title>" + title + "</title></head>\n" +
"<body bgcolor = \"#f0f0f0\">\n" );
if( cookies != null ) {
out.println("<h2> Found Cookies Name and Value</h2>");
for (int i = 0; i < cookies.length; i++) {
cookie = cookies[i];
```

```
out.print("Name : " + cookie.getName( ) + ", ");
out.print("Value: " + cookie.getValue( ) + " <br/>");
}
} else {
out.println("<h2>No cookies founds</h2>");
}
out.println("</body>");
out.println("</html>");
}
}
```

4.What is session? What are different ways of tracking sessions? Write down suitable servlet for trackiing session.

Session:

the word "session" is a reference to a certain time frame for communication between two devices, two systems or two parts of a system.

It can also be termed as the state consisting of several requests and responses between the client and the server.

Session tracking:

Session tracking is a way to maintain state (data) of a user, which is also known as session management in servlets.

Different ways of tracking sessions are:

1.User Authorization :: Basic concept is that the user will provide username and password to login to the application. Based on that the user can be identified and the session can be maintained.

2.Hidden Fields:: Hidden can be 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

3.URL rewriting: When a request is made, additional parameter is appended with the url. In general, added additional parameter will be session id or sometimes the user id. It will suffice to track the session.

4. Cookies: Cookies are the mostly used technology for session tracking. Whenever the browser sends a request to that server it sends the cookie along with it. Then the server can identify the client using the cookie.