14] Describe the servelet life cycle with life cycle methods. Explain role of web container.

→

The life cycle contains the following steps :

- Load servelet class.
- Create Instance of servelet.
- Call the servelets init() Method.
- Call the servelets service() Method.
- Call the servelets destroy() Method.

- Now, we will discuss about each steps in details.

* Loading servelet class :-
- A servelet class is loaded when first request for the servelet is received by the web container.

* Create servelet instance :-
- After the servelet class is loaded, web container creates the instance of it. Servelet instance is created only once in the life cycle.

* Call the service() Method :-
- The container calls the init() Method each time the to initialize the servelet.

* Call the Service() Method :-
- The container calls the Service() Method each time the request for servelet is received.

* Call destroy() Method :-

- The web container calls the destroy() Method before removing servelet instance giving a chance for cleanup activity.

→ Method :-

* init() Method :-

- The servelet is initialized by calling the init() Method.

- The init Method is called only one time, when servelet is first created, never calls again for each user request.

Public void init() throw servletException
{
 	
}

- This Method allows the servelet to perform any setup processing such as opening files or establishing connections to their servelets.

* service() Method :-

- The service() Method is the heart of the servelet. Each request message from a client results in a single call to the servelet's service() Method.

- The service() Method's job is conceptually simple - it creates a Response for each client Request sent to it from the host servlet.

- The service() Method checks the HTTP Request type such as GET, POST etc and calls doGet, doPost etc

```
Public void service (ServletRequest, ServletResponse Response)
throws ServletException, IOException
{

}
```

* doGet() Method :-

- GET is HTTP method. It wants to tell servlet to get a Resource and send it. It should be handled by doGet() Method.

```
Public void doGet (HttpServletRequest Request, HttpServlet-
Response Response)

throws ServletException, IOException
{

}
```

* The doPost() Method :-

With Post, You can request something and at same time send form data to server. It should be handled by doPost() method.

```
Public Void doPost(HttpServletRequest request, HttpServlet
Response response)

throws ServletException, IoException
{

}
```

* destroy() Method :-

The destroy() method is called to allow your servlet to clean up any resources before the servlet is unloaded.

If you do not require any clean-up operations, this can be an empty method.

The server waits to call the destroy() method until either all service calls are complete, or a certain amount of time has passed.

```
Public Void destroy()
{
  // finalization code
}
```

→ Web container :

- Web container is responsible for managing execution of servlet. The life cycle of every is controlled by the web container in which the servlet has been declared.

- The container performs the following steps :

1. If an instance of the servlet does not exist, the web container loads the servlet class.

2. Invokes the service method, passing request and response objects.

3. If the container needs to remove the servlet, it finalizes the servlet by calling the servlet's destroy method.

- Servlet don't have a main() method. web container manages the life cycle of a servlet instance.

15] Explain step by step procedure to execute servlet program.

→

These steps are as follows :

Step 1 :- Create a directory structure under Tomcat for Java application.

Step 2 :- write the servlet source code.

* TestServlet.Java

```
IMPORT Javax.Servlet.*;
IMPORT javax.Servlet.http.*;
IMPORT Java.io.*;
IMPORT Java.util.*;

Public class TestServlet extends HttpServlet
{
 Public Void doGet(HttpServletRequest request, HttpServletRes-
 Ponse response)

 throws ServletException, IoException
  {

 PrintWriter out = response.getWriter();
 out.println("<html>");
 out.println("<body>");
 out.println("Hello world!");
 out.println("</body>");
 out.println("</html>");
  }
}
```

Step 3 :- Compile Your Source Code

- For your servlet source code to compile, you need to include in your CLASSPATH environment variable the Path to the servlet.jar file.

- Now change directory to your working directory :

```
javac -d ..\WEB-INF\classes\ TestServlet.java
```

Step 4 :- Create the Deployment Descriptor

```
<!DOCTYPE web-app
Public "-//Sun Microsystems, Inc.//DTD web Application
2.3//" http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
 <servlet>
 <servlet-name> Test </servlet-name>
 <servlet-class> TestServlet </servlet-class>
 </servlet>
</web-app>
```

Step 5 :- Run Tomcat

- If it is not already running, you need to start Tomcat. For information on how to do that, see Appendix B "Tomcat Installation and Configuration".

Step 6 :- Call Your Servlet from a web Browser.

http://domain-name/virtual-directory/Servlet/
Servlet-name.

16] Explain Process to read data from database table and display it. write a Program for it also.

→

It can be of two types namely Structural and non-Structural database. The Structural database is the one which can be Stored in Row and Columns.

- A Non-Structural database can not be Stored in form of Rows and Columns for which new concepts are Introduced which would not be discussing here.

- The Structural database is Stored in database. Hence, the Structural database warehouses and unstructured in data lakes

- Java Database Connectivity is basically a Standard API between the Java Programming language and Various databases lik MongoDB, SQL etc.

- The Syntax for importing Package to deal with JDBC:

```
import java.sql.*;
```

- The Syntax for registering drivers after loading the driver class:

```
ForName("com.mysql.jdbc.xyz");
```

* Example :-

```
import java.sql.*;

Public class JdbcResultset
{
  Public Static Void Main (String[] args)
   {
     try
      {
       class.forName("org.apache.derby.jdbc.clientDriver");
     } catch (ClassNotFoundException e)
      {
       System.out.println ("class not found" +e);
      }
    try
     {
     Connection con = DriverManager.getConnection(
"Jdbc:derby://localhost:1527//testDb","Uname","Pass");

Statement Stmt = con.createStatement();
ResultSet rs = Stmt.executeQuery("SELECT * from employee");
System.out.println("id Name Job");

while (rs.next())
{
int id = rs.getInt("id");
String name = rs.getString("name");
String Job = rs.getString("Job");
System.out.println(id+" "+name+" "+Job);
}
}
```

```
catch (SQLException e)
{
System.out.println("SQL exception occured "+e);

    }
  }
}
```

17] Explain life cycle of JSP.

→

The .JspPage interface declares only two methods - JspInit() and jspDestroy() that must be implemented by all Jsp Pages regardless of the client-server Protocol.

- However, the JSP specification has provided the HttpJspPage Interface specifically for the JSP Pages Solving HTTP Requests.

- JSP Page looks like a HTML Page but is a servlet. When presented with the JSP page the JSP engine does the following 7 Phases.

1. Translation :-
   JSP container checks the JSP code and Parses it to generate the servlet source code. If the JSP page name is home.jsp, usually the generated servlet class name is home_jsp and file and file name is home_jsp.java

2. Compilation :-
   JSP container compiles the jsp class source code and Produce class file in this Phase.

3. Class Loading :-
   Container loads the class into Memory in this Phase.

4. Instantiation :-
   Container invokes the no-args constructor of generated class to load it into Memory and instantiate it.

5. Initialization :-

Container invokes the init method of JSP class object and initializes the servlet config with init params configured in deployment descriptor.

- After this phase, JSP is ready to handle client requests.

6. Request Processing :-

This is the longest lifecycle of JSP page and JSP page process the client requests. The processing is multi-threaded and similar to servlets and for every request a new thread is spawned and ServletRequest and ServletResponse object is created and JSP service method is invoked.

7. Destroy :-

This is the last phase of JSP lifecycle where JSP class is unloaded from memory. Usually it happens when application is undeployed or the server is shut down.

→ Methods :-

1. jspInit()
2. _jspService()
3. jspDestroy()

18] what is JSP ? Explain with its architecture.

→

* JSP :-

JSP which stands for Java Server Pages. It is a server-side technology. It is used for creating web applications. It is used to create dynamic web content.

- It is used to create JSP tags are used to insert JAVA code into HTML Pages.

* JSP Architecture :-

- JSP architecture gives a high-level view of the working of JSP. JSP architecture is a 3 tier architecture. It has a client, web server, and Database.

- The client is the web browser or application on the user side. web server uses a JSP Engine i.e. for example Apache Tomcat has a built-in JSP Engine.

- To understand the JSP Processing better which is as follows.

- JSP Engine intercepts the request for JSP and provides the runtime environment for the understanding and processing of JSP files.

- It reads, parses, build Java servlet, compiles and executes Java code, and returns the HTML page to the client.

JSP architecture process follows or perform following steps :

Step 1 :- The client navigates to a file ending with the .jsp extension and the browser initiates an HTTP request to the webserver.

Step 2 :- The JSP Engine loads the JSP file and translates the JSP to servlet (Java coded. This is done by converting all the template text into println() statements and JSP elements to Java code. This process is called "Translation."

Step 3 :- If the compiled version of JSP exists in the web server, it returns the file. Otherwise, the request is forwarded to the JSP Engine. This is done by recognizing the URL ending with .jsp extension.

Step 4 :- The JSP engine compiles the servlet to an executable .class file. It is forwarded to the servlet engine. This process is called compilation or request processing phase.

Step 5 :- The .class file is executed by the servlet engine which is a part of the web server. The output is an HTML file. The servlet engine passes the output as an HTTP response to the webserver.

Step 6 :- The web server forwards the HTML file to the client's browser.

19] Differentence between Jsp and servlet.

→

	JSP	Servlet
*	JSP can be compiled into servlet when accessed.	Servlet are Java programs that are already compiled.
*	It's easier to code in JSP than in Java servlets.	Application coding is difficult compared to JSP.
*	JSP are generally not preffered when there is much processing of data required.	Servlet are preffered when there is much processing of data required.
*	we can achieve functionality of JSP at client side by running Javascript at client side.	There are no such methods for servlets.
*	we can build custom tags using JSP.	we can not build custom tags using servlet.

20] Explain how to fetch database records using JSP.

→ We will be using MySQL as database to work with JSP

→ Create database :

```
CREATE DATABASE STUDENTS
```

→ Create Table :

```
CREATE TABLE 'STUDENTS'.'Studentinfo' (
'RollNo' INT NOT NULL,
'Name' VARCHAR(50) NOT NULL
);
```

→ Insert Records in the Table :

```
INSERT INTO 'Students'.'Studentinfo' ('RollNo','Name')
VALUES ('1','Shubham'),('2','Deep');
```

→ Displaying Database Records Using JSP :

DBAccess.jsp

```
<%@ Page language="java" import="java.sql.*"%>
<html>
<body>

<P align="center"> <b> Records from the Studentinfo
table. </b> </body> </P>
```

```
<center>

<table border="1" bordercolor="Black" width="65%"
height="63">

<% String Driver = "com.mysql.jdbc.Driver" }
class.forName(DRIVER).newInstance()

Connection con = null;
ResultSet rst = null;
Statement stmt = null;

try
{
String url = "jdbc:mysql://localhost/students";
int i = 1;

con = DriverManager.getConnection(url,"root","");
stmt = con.CreateStatement();
rst = stmt.executeQuery("SELECT * FROM studentinfo");
if (rst.next())
{
  %>
  <TABLE BORDER=1 BGCOLOR="CCCCCC">
<TH WIDTH=200 BGCOLOR="white"> Roll No </TH>
<TH WIDTH=200 BGCOLOR="white"> Name </TH>
<TR> <TD ALIGN=CENTER> <%= rst.getInt(1) %> </TD>
<TD ALIGN=CENTER> <%= rst.getString(2) %> </TD>
</TR>
<% while (rst.next())
{
  %>
```

```
<TR><TD ALIGN=CENTER><%=rst.getInt(1)%></TD>
<TD ALIGN=CENTER><%=rst.getString(2)%></TD>
</TR>

<%}
%>

</TABLE>

<%}
else
  {
   %>
   <P>Sorry, the query returned no rows!</P>

<%
}
rst.close();

stmt.close();
} catch (SQLException e)
 {
out.println("There was an error:");
out.println(e);
}
%>
</table>
</center>
</body>
</html>
```