1. Create an index.html file containing a text input field for entering a table name and a submit button. When the button is clicked, the request should be sent to a servlet, which processes the request and sends a response back to the client

index.html

<form action="servletannotationvaluerequired" method="post">

<font face="verdana" size="2">

Enter Table Name :<input type="text" name="table">

<input type="submit" value="Display">

</font>

</form>

ServletProgram

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet(“Fetch”)

public class ServletDatabaseConnect extends HttpServlet

{

protected void doPost(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException

{

PrintWriter pw=res.getWriter();

res.setContentType("text/html");

String tb=req.getParameter("table");

try

{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");

Statement st=con.createStatement();

System.out.println("connection established successfully...!!");

ResultSet rs=st.executeQuery("Select \* from "+tb);

pw.println("<table border=1>");

while(rs.next())

{

pw.println("<tr><td>"+rs.getInt(1)+"</td><td>"+rs.getString(2)+"</td>"+

"<td>"+rs.getString(3)+"</td></tr>");

}

pw.println("</table>");

pw.close();

}

catch (Exception e){

e.printStackTrace();

} }

Create a movie.html page with a form that allows the user to enter Movie ID, Movie Name, and Movie Rating. When the user clicks the Submit button, the form data should be sent to a servlet. The servlet should process the details (e.g., save them to a database or display them back) and send a response to the client.

Hint:In Java Servlets, the **getParameter()** method is used to **retrieve form data** (or any request parameters) that the client sends to the server.

request.getParameter("fieldName")

<form action="MovieServlet" method="post">

Movie ID: <input type="text" name="movieId"><br>

Movie Name: <input type="text" name="movieName"><br>

Rating: <input type="text" name="rating"><br>

<input type="submit" value="Submit">

</form>

Sample code of Servlet

String id = request.getParameter("movieId");

String name = request.getParameter("movieName");

String rating = request.getParameter("rating");

out.println("Movie ID: " + id);

out.println("Movie Name: " + name);

out.println("Rating: " + rating);

//Add code to store above details in DB

//Load driver

//coonection

//prepared statement

//insert query

1. Create an HTML page with a text input field for entering a Movie ID and a submit button. When the user submits the form, the request should be sent to a servlet. The servlet should retrieve the details (movie name and rating) of the movie with the given Movie ID from the database and display them in the browser.

**Important Methods in Servlet**

1. Lifecycle Methods (Servlet execution flow)

These come from the javax.servlet.Servlet interface and are part of every servlet.

* init(ServletConfig config) → Called once when the servlet is first loaded.
* service(HttpServletRequest req, HttpServletResponse res) → Handles requests (usually overridden indirectly via doGet()/doPost()).
* destroy() → Called before the servlet is unloaded from memory.

2. HTTP Request Handling Methods

From HttpServlet:

* doGet(HttpServletRequest request, HttpServletResponse response) → Handles HTTP GET requests.
* doPost(HttpServletRequest request, HttpServletResponse response) → Handles HTTP POST requests.
* doPut(), doDelete() → For PUT and DELETE requests (rarely used in basic forms).

3. Request Object Methods (HttpServletRequest)

These help you retrieve data sent from the client.

* getParameter(String name) → Gets value of a form field.
* getParameterValues(String name) → Gets multiple values (e.g., checkboxes).
* getParameterNames() → Returns all parameter names.
* getMethod() → Returns HTTP method used (GET/POST).
* getRequestURI() → Returns request path.
* getSession() → Gets the session object for storing user-specific data.

4. Response Object Methods (HttpServletResponse)

These help you send data back to the client.

* setContentType(String type) → Sets the MIME type (e.g., "text/html").
* getWriter() → Returns a PrintWriter for sending text output.
* sendRedirect(String url) → Redirects client to another URL.
* setStatus(int sc) → Sets HTTP status code (e.g., 404, 200).

5. Session & Context Methods

* HttpSession setAttribute(String name, Object value) → Stores data in session.
* getAttribute(String name) → Retrieves stored data.
* ServletContext getInitParameter(String name) → Reads context-wide parameters from web.xml.

Describe the role of Web.xml and without Web.xml how can we set parameters

Note :Individually prepare Presentation for above applications and procedure of implementation.