1.What is a directive in JSP? Name and very briefly explain the need of different JSP directives. How is a JSP different from that of a Servlet? Write a simple JSP to count and display the number of times a page has been visited in a single session..

Ans.

Directives in JSP:

In JSP (JavaServer Pages), a directive provides global information about how a JSP page should be processed by the JSP container. Directives do not produce any output themselves but affect the overall structure of the resulting servlet.

Ans.

Types of JSP Directives:

1. **Page Directive** (`<%@ page %>`)
   It defines page-level instructions, such as the scripting language used, error handling, buffering, and importing packages.
   Example:

   <%@ page language="java" contentType="text/html" %>

2. **Include Directive** (`<%@ include %>`)

   It includes a static file (e.g., header, footer) at compile time into the current JSP. This is useful for reusing common parts across multiple pages.

   Example: <%@ include file="header.jsp" %>

3. **Taglib Directive** (`<%@ taglib %>`)

   It declares a tag library containing custom JSP tags that can be used in the page. This is used to extend the functionality of JSP pages.

Example: <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

Ans. Difference between JSP and Servlet:

- JSP: JSP is a high-level abstraction of Servlets. It allows embedding Java code directly into HTML, making it easier to create dynamic web content. It is used mainly for presentation purposes.
- Servlet: Servlets are Java classes that handle HTTP requests and responses at the server-side. They are typically used for business logic and complex request handling.

Ans. Simple JSP to Count Page Visits in a Single Session:

<%@ page session="true" %>

<html>

```
<head>
    <title>Page Visit Counter</title>
</head>
<body>
    <h1>Page Visit Counter</h1>

    <%
        // Get the current session
        Integer visitCount = (Integer) session.getAttribute("visitCount");

        if (visitCount == null) {
            // First visit
            visitCount = 1;
        } else {
            // Increment visit count
            visitCount++;
        }

        // Store the updated visit count in the session
        session.setAttribute("visitCount", visitCount);
    %>

    <p>You have visited this page <%= visitCount %> times in this session.</p>
</body>
</html>
```

2. Assume a table EMP [fields: ID, NAME, SALARY, AGE] in mySql in your localhost. Now do the following from your Java file:

i. Register the JDBC driver

ii. Open a connection

iii. Execute a query to find out the Employees above 40 years of age who get salary above Rs 40,000 per month

iv. Display the result

v. Close the connection.

Ans.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.SQLException;

public class EmployeeQuery {
    public static void main(String[] args) {
        // JDBC URL, username, and password of MySQL server
        String jdbcURL = "jdbc:mysql://localhost:3306/CSE188";
        String username = "CSE188";
        String password = "CSE188";

        // JDBC variables for opening and managing connection
        Connection connection = null;
        Statement statement = null;

        try {
            // 1. Register the JDBC driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // 2. Open a connection
            System.out.println("Connecting to the database...");
            connection = DriverManager.getConnection(jdbcURL, username, password);

            // 3. Execute a query
            String query = "SELECT ID, NAME, SALARY, AGE FROM EMP WHERE AGE > 40 AND SALARY >
40000";
            statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(query);

            // 4. Display the result
            System.out.println("Employees above 40 years of age with salary above Rs 40,000:");
            while (resultSet.next()) {
                int id = resultSet.getInt("ID");
                String name = resultSet.getString("NAME");
                double salary = resultSet.getDouble("SALARY");
                int age = resultSet.getInt("AGE");
```

```
            System.out.println("ID: " + id + ", Name: " + name + ", Salary: " + salary + ", Age: " + age);
        }

        // 5. Close the resultSet, statement, and connection
        resultSet.close();
        statement.close();
        connection.close();

    } catch (SQLException | ClassNotFoundException e) {
        e.printStackTrace();
    } finally {
        // Ensure statement and connection are closed to avoid resource leaks
        try {
            if (statement != null) statement.close();
            if (connection != null) connection.close();
        } catch (SQLException se) {
            se.printStackTrace();
        }
    }
  }
}
```

3. What is the difference between Prepared and Callable statements? Explain with example.Explain the steps of creating the web page by web server when you request a JSP page from your browser.Explain the use of forward action in JSP with an example.

Ans.

Difference between `PreparedStatement` and `CallableStatement`:

PreparedStatement:

- Purpose: Used to execute parameterized SQL queries.
- Usage: It prevents SQL injection attacks by allowing you to use placeholders for input parameters in your SQL query.
- PreparedStatement is for executing SQL queries, mainly for dynamic, parameterized queries.
- Example:
  String sql = "SELECT * FROM EMP WHERE AGE > ? AND SALARY > ?";

  PreparedStatement pstmt = connection.prepareStatement(sql);

  pstmt.setInt(1, 40);

  pstmt.setDouble(2, 40000);

  ResultSet rs = pstmt.executeQuery();

CallableStatement:

- Purpose: Used to call stored procedures from the database.
- Usage: It supports IN, OUT, and INOUT parameters to communicate with stored procedures.
- CallableStatement is used for calling stored procedures that may involve IN/OUT parameters.
- Example:
  CallableStatement cstmt = connection.prepareCall("{call GET_EMPLOYEES(?, ?)}");

  cstmt.setInt(1, 40); // IN parameter

  cstmt.setDouble(2, 40000); // IN parameter

  ResultSet rs = cstmt.executeQuery();

Ans.

Steps in Creating a Web Page by Web Server when Requesting a JSP Page:

When a browser requests a JSP page, the web server follows these steps:

1. Request Received: The web server receives the HTTP request for a JSP page.
2. JSP Compilation:
   - If the JSP page is being accessed for the first time (or if the JSP has been modified), the server translates the JSP file into a Servlet (Java class).
   - The servlet class is then compiled into bytecode, creating a `.class` file.
3. Servlet Execution:
   - The compiled JSP servlet is loaded and executed by the servlet engine.
   - The servlet generates dynamic content based on the request, typically HTML.
4. Response Sent:
   - The generated HTML content is returned as an HTTP response to the browser.
5. Browser Renders Page:
   - The browser receives the HTML response and renders the web page for the user.

Ans. `forward` Action in JSP:

The `<jsp:forward>` action is used to forward a request from one JSP page (or servlet) to another resource (another JSP page, servlet, or HTML page) within the server.

Syntax: <jsp:forward page="targetPage.jsp" />

Example:

Consider you have two JSP pages: `page1.jsp` and `page2.jsp`.

page1.jsp:

<html>

<body>

```
    <h2>Welcome to Page 1</h2>
    <%
        if (someCondition) {
            // Forward the request to page2.jsp
            RequestDispatcher dispatcher = request.getRequestDispatcher("page2.jsp");
            dispatcher.forward(request, response);
        }
    %>
</body>
</html>
```

page2.jsp:

```
<html>
<body>
    <h2>This is Page 2</h2>
    <p>Request forwarded from Page 1</p>
</body>
</html>
```

Explanation:

- In `page1.jsp`, when `someCondition` is true, the request is forwarded to `page2.jsp`.
- The client browser is not aware of the forward action, so the URL in the browser will still show `page1.jsp`, but the content will be served from `page2.jsp`.
- Forwarding is useful when processing part of the request in one resource and then passing control to another resource without the client being aware of the transition.

4. Write a simple JSP to show current Date using appropriate Page Directive. What are JSP actions? Give the general syntax for defining an action in JSP. Name some of the most widely used actions in JSP.
Ans.

Simple JSP to Show Current Date Using Page Directive:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<html>
<head>
  <title>Current Date</title>
</head>
<body>
  <h1>Current Date and Time</h1>
  <p>
    <%= new java.util.Date() %>
  </p>
</body>
</html>
```

Ans.

JSP Actions:

JSP actions are XML-based tags that control the behavior of the servlet engine. They are used to perform operations such as including other resources, forwarding requests, or generating dynamic content. JSP actions use XML syntax to embed Java functionality into JSP pages.

Ans.

General Syntax for Defining a JSP Action:

```
<jsp:actionName attribute1="value1" attribute2="value2" ... />
```

Ans.
Most Widely Used JSP Actions:

- **`<jsp:include>`**

This action is used to include the content of another resource (JSP, HTML, or servlet) at request time.

```
<jsp:include page="header.jsp" />
```

- **`<jsp:forward>`**
  This action forwards the request from one JSP page to another resource (JSP, servlet, or HTML).
  `<jsp:forward page="nextPage.jsp" />`

- **`<jsp:useBean>`**
  This action creates or locates a JavaBean object, which can be used to store and retrieve data.
  `<jsp:useBean id="myBean" class="com.example.MyBean" scope="session" />`

- **`<jsp:setProperty>`**
  This action sets the property of a JavaBean.
  `<jsp:setProperty name="myBean" property="name" value="John Doe" />`

- **`<jsp:getProperty>`**
  This action retrieves the property value from a JavaBean and displays it.

<jsp:getProperty name="myBean" property="name" />

- **`<jsp:param>`**
  This action is used inside `<jsp:include>` or `<jsp:forward>` to pass parameters to the target resource.
  <jsp:include page="header.jsp">
  <jsp:param name="user" value="John Doe" />
  </jsp:include>

Summary of Common JSP Actions:

- `<jsp:include>`: Includes content from another page.
- `<jsp:forward>`: Forwards the request to another resource.
- `<jsp:useBean>`: Instantiates or finds a JavaBean.
- `<jsp:setProperty>`: Sets properties in a JavaBean.
- `<jsp:getProperty>`: Retrieves properties from a JavaBean.
- `<jsp:param>`: Passes parameters between resources.

5. Write a JSP scriplet for displaying even numbers between 1 to 50. Explain what happens in the background when a JSP page is accessed by a user from a server that supports JSP. What is the difference between scriptlet and expression? State the main objectives of J2EE.
Ans.
JSP Scriptlet for Displaying Even Numbers Between 1 to 50:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<html>
<head>
   <title>Even Numbers</title>
</head>
<body>
   <h2>Even Numbers Between 1 and 50</h2>
   <ul>
      <%
         for (int i = 2; i <= 50; i += 2) {
            out.println("<li>" + i + "</li>");
         }
      %>
   </ul>
</body>
</html>
```

Ans.

What Happens in the Background When a JSP Page is Accessed:

1. Request Sent to Server:
   The user sends an HTTP request to the server, requesting a JSP page (e.g., `index.jsp`).
2. JSP Compilation (If Required):
   - If this is the first time the JSP is being accessed (or if the JSP file has been modified), the server translates the JSP into a Servlet (Java class).
   - The JSP code (HTML, scriptlets, etc.) is translated into Java servlet code.
3. Servlet Compilation:
   - The generated servlet is compiled into a `.class` file (bytecode) by the server.
4. Servlet Execution:
   - The servlet executes in the server, where it processes the HTTP request, interacts with the database or other services, and generates the dynamic content (HTML, XML, JSON, etc.).
5. Response Sent to Browser:
   - The servlet sends the generated response (typically HTML) back to the client as an HTTP response.
   - The browser receives the response and renders the web page.

Ans.

Difference Between Scriptlet and Expression in JSP:

Scriptlet (`<% %>`):

- A scriptlet is a block of Java code embedded inside the JSP. It can include any valid Java code and is used to write logic or manipulate data.
- Scriptlet: Can contain multiple lines of Java code (like loops, if-else, etc.), but does not automatically print anything.

  Example:
  ```
  <%
          int x = 5;
          out.println("The value of x is: " + x);
  %>
  ```

Expression (`<%= %>`):

- An expression evaluates a single Java expression and directly prints the result to the response output stream (e.g., HTML content).
- Expression: Used to display the result of an expression directly into the output stream.

  Example:
  ```
  <%= 5 * 5 %>
  ```

Ans.

Main Objectives of J2EE:

1. Enterprise-Level Development: J2EE (Java 2 Enterprise Edition) is designed to build large-scale, distributed, and transactional applications for enterprise environments.

2. Component-Based Architecture: J2EE enables developers to build reusable, modular components (such as EJBs, servlets, JSPs) that handle specific functionality within an enterprise application.
3. Platform Independence: J2EE promotes the concept of "write once, run anywhere," ensuring that applications can run on any platform that supports the J2EE standards.
4. Multi-Tiered Architecture: J2EE facilitates a multi-tiered architecture, typically consisting of client-tier, middle-tier (business logic), and data-tier (database). This separation helps in managing large-scale applications efficiently.
5. Scalability and Performance: J2EE provides a set of services (such as transaction management, security, messaging) that allow applications to scale and perform well in enterprise environments.
6. Security and Transaction Management: J2EE supports integrated security features (authentication, authorization) and transaction management, ensuring reliable and secure operations in business applications.

6.What is difference between scriptlet and expression? Explain with example. What do you understand by JDBC-ODBC bridging? Write a JSP program that shows the System date and time.
Ans.

Difference Between Scriptlet and Expression in JSP:

- Scriptlet (`<% %>`):
    - A scriptlet contains Java code inside the JSP. You can write any amount of Java code (like loops, conditions) within a scriptlet. It doesn't automatically print anything to the output unless explicitly done using `out.print()`
    - `Scriptlet: Used for writing Java code blocks and does not automatically print anything.`
    - Example:

```
<%

  // Scriptlet to calculate the sum of two numbers

  int a = 5;

  int b = 10;

  int sum = a + b;

  out.println("Sum is: " + sum);  // Must use out.print() to display content

%>
```

Expression (`<%= %>`):

- An expression is used to display the result of a single Java expression directly to the output. It is automatically evaluated, and the result is printed as part of the response.

- Expression: Used for directly outputting the result of an expression into the response.
- Example:
  <%= 5 + 10 %>
- This expression directly evaluates `5 + 10` and prints `15` in the HTML response without using `out.print()`.

Ans.

JDBC-ODBC Bridge:

The JDBC-ODBC Bridge is a driver that allows Java applications to interact with databases using ODBC (Open Database Connectivity). It serves as a bridge between JDBC (Java Database Connectivity) and ODBC.

- Functionality:
  - JDBC is a Java API that allows Java programs to interact with a database.
  - ODBC is a database access API designed to allow database interactions using C or C++.
  - The JDBC-ODBC Bridge allows a Java application using JDBC to communicate with databases that only support ODBC drivers. Essentially, the bridge converts JDBC calls into ODBC calls.
- Use Case:
  - The JDBC-ODBC Bridge was commonly used in earlier versions of Java when many databases did not provide native JDBC drivers.
- Example:
  - When using the JDBC-ODBC Bridge, the driver is typically registered like this:

    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

    Connection connection = DriverManager.getConnection("jdbc:odbc:DataSourceName");

- Limitation:
  - The JDBC-ODBC Bridge has been removed in newer versions of Java (since Java 8) due to performance issues and limited usage.

Ans.

JSP Program to Show System Date and Time:

<%@ page language="java" contentType="text/html; charset=UTF-8" %>

<html>

<head>

  <title>Current Date and Time</title>

```
</head>

<body>

   <h2>System Date and Time</h2>

   <p>

      <%= new java.util.Date() %>

   </p>

</body>

</html>
```

7.Write a JSP program to calculate factorial of an integer while the input is taken from an HTML form.Briefly explain all 4 types of JDBC Driver Explain the use of <jsp:plugin> element in JSP with an example.

Ans.

JSP Program to Calculate Factorial of an Integer (Input from HTML Form)

HTML Form (index.html):

```
<!DOCTYPE html>
<html>
<head>
   <title>Factorial Calculator</title>
</head>
<body>
   <h2>Enter a Number to Calculate Factorial</h2>
   <form action="factorial.jsp" method="post">
      <label for="number">Number:</label>
      <input type="number" id="number" name="number" required>
      <input type="submit" value="Calculate Factorial">
   </form>
</body>
</html>
```

JSP to Process Form and Calculate Factorial (factorial.jsp):

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<html>
<head>
   <title>Factorial Result</title>
</head>
<body>
```

```
    <h2>Factorial Result</h2>
    <%
        // Getting the number from the form
        String numberStr = request.getParameter("number");
        int number = Integer.parseInt(numberStr);

        // Calculating factorial
        int factorial = 1;
        for (int i = 1; i <= number; i++) {
            factorial *= i;
        }
    %>
    <p>The factorial of <%= number %> is: <%= factorial %></p>
</body>
</html>
```

Ans.

Types of JDBC Drivers:

1. Type 1: JDBC-ODBC Bridge Driver:
    - This driver translates JDBC calls into ODBC calls.
    - It requires an ODBC driver installed on the client machine and works as a bridge between JDBC and ODBC.
    - Advantage: Easy to use.
    - Disadvantage: Performance is poor because it requires an additional ODBC layer, and it's not recommended for production. Deprecated in modern Java versions.
2. Type 2: Native-API Driver (Partially Java Driver):
    - This driver converts JDBC calls into database-specific API calls, which are then sent to the database.
    - Advantage: Better performance than Type 1 since the native API is used.
    - Disadvantage: Requires database-specific native libraries installed on the client machine, making it less portable.
3. Type 3: Network Protocol Driver (Pure Java Driver):
    - The Type 3 driver translates JDBC calls into a database-independent network protocol that is forwarded to a middle-tier server, which then translates it to the database protocol.
    - Advantage: No client-side library is needed; it's fully Java-based and can connect to any database.
    - Disadvantage: Requires a middleware server for processing.
4. Type 4: Thin Driver (Pure Java Driver for Direct Database Connection):
    - This driver directly converts JDBC calls into database-specific protocols using native libraries at the database server.
    - Advantage: Best performance, fully implemented in Java, and requires no middleware.
    - Disadvantage: Database-specific, meaning you need a different driver for each database.

Ans.

Use of `<jsp:plugin>` Element in JSP:

The `<jsp:plugin>` element is used in JSP to embed Java applets or activate JavaBeans in a browser. It dynamically generates the `<object>` or `<embed>` tags based on the browser type, allowing the client to run the applet or bean.

Syntax:

```
<jsp:plugin type="applet" code="AppName.class" codebase="classes" width="300" height="300">
   <jsp:param name="paramName" value="paramValue" />
</jsp:plugin>
```

Attributes:

- type: Specifies whether to use an `applet` or `bean`.
- code: The name of the class that should be loaded (e.g., an applet class).
- codebase: Path to the location of the applet or bean class files.
- width and height: Specifies the size of the applet/bean on the client-side.

Example:

```
<jsp:plugin type="applet" code="MyApplet.class" codebase="classes" width="300" height="300">
   <jsp:param name="bgcolor" value="lightblue" />
</jsp:plugin>
```

Explanation:

- The `<jsp:plugin>` tag is used to embed the `MyApplet.class` applet in the JSP page. The size of the applet is set to 300x300, and a parameter (`bgcolor`) is passed to customize its background color.

8.Write a JSP program that shows the Fibonacci series up to a particular term, while the input is taken from an HTML form. What are the types of Elements in JSP? What is JSP Expression Language (EL)? Ans.

JSP Program to Show Fibonacci Series Up to a Particular Term (Input from HTML Form)

HTML Form (index.html):

```
<!DOCTYPE html>
<html>
<head>
   <title>Fibonacci Series</title>
</head>
<body>
   <h2>Enter the Number of Terms for Fibonacci Series</h2>
```

```
    <form action="fibonacci.jsp" method="post">
        <label for="terms">Number of Terms:</label>
        <input type="number" id="terms" name="terms" required>
        <input type="submit" value="Show Fibonacci Series">
    </form>
</body>
</html>
```

JSP to Process Form and Display Fibonacci Series (fibonacci.jsp):

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<html>
<head>
    <title>Fibonacci Series</title>
</head>
<body>
    <h2>Fibonacci Series</h2>
    <%
        // Getting the number of terms from the form
        String termsStr = request.getParameter("terms");
        int terms = Integer.parseInt(termsStr);

        // Variables for Fibonacci series
        int first = 0, second = 1, next;

        // Display the Fibonacci series
        out.println("Fibonacci Series up to " + terms + " terms:<br/>");
        for (int i = 1; i <= terms; i++) {
            if (i == 1) {
                out.println(first + " ");
                continue;
            }
            if (i == 2) {
                out.println(second + " ");
                continue;
            }
            next = first + second;
            first = second;
            second = next;
            out.println(next + " ");
        }
    %>
</body>
</html>
```

Ans.

Types of Elements in JSP:

JSP has several types of elements that help developers add Java code to their web pages:

1. Directive Elements:
    - These are instructions for the JSP engine about how to process the page. They do not produce any output.
    - Example:

      <%@ page language="java" contentType="text/html; charset=UTF-8" %>

The common directive elements are:

- Page Directive: Defines page attributes like content type, scripting language, error page, etc.
- Include Directive: Includes a file during the translation phase of the JSP.
- Taglib Directive: Declares a tag library for custom tags.

2. Scripting Elements:

- These allow embedding Java code in JSP pages.
- Examples:
    - Scriptlet (`<% ... %>`): Contains arbitrary Java code.
    - Expression (`<%= ... %>`): Outputs the result of an expression directly to the output.
    - Declaration (`<%! ... %>`): Declares variables or methods that become part of the servlet class.

3. Action Elements:

- These are predefined tags used to control the behavior of the JSP container.
- Example: <jsp:include page="header.jsp" />

Some common action elements include:

`<jsp:include>`: Includes content from another resource at request time.

`<jsp:forward>`: Forwards the request to another resource.

`<jsp:useBean>`: Instantiates or accesses a JavaBean.

4. Custom Tag Elements:

- These allow the use of custom tags, defined in tag libraries (taglibs), to extend the functionality of JSP.
- Example: <mytag:customTag attribute="value" />

Ans.

JSP Expression Language (EL):

JSP Expression Language (EL) simplifies accessing data stored in JavaBeans, request parameters, and attributes within the JSP without the need for explicit Java code.

- Purpose:
    - EL is designed to make JSPs cleaner and more concise by allowing easy access to Java objects, attributes, and properties without the use of complex scriptlets.
- Syntax:
    - EL expressions are enclosed in `${}`. For example, `${user.name}` accesses the `name` property of the `user` bean.
- Features:
    - Automatic Type Conversion: EL automatically handles type conversions (e.g., converting strings to numbers).
    - Accessing Objects: EL allows access to various scopes (request, session, application) as well as implicit objects such as `param`, `sessionScope`, and more.
        - Example: `${sessionScope.user}` retrieves the `user` object from the session.
- Example: Welcome, ${user.name}
- Advantages of EL:
    a) Simplifies accessing data in JSP, removing the need for scriptlet code.
    b) More readable and less error-prone compared to scriptlets for simple tasks.