Write a program to set and print the cookies.

```
import javax.servlet.http.*;
import java.io.IOException;

public class CookieCreatorServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        // Create a cookie named "user" with the value "John Doe"
        Cookie cookie = new Cookie("user", "John Doe");

        // Set the cookie to expire in 1 hour
        cookie.setMaxAge(3600);

        // Add the cookie to the response
        response.addCookie(cookie);

        // Create a PrintWriter object for writing to the response body
        PrintWriter out = response.getWriter();

        // Access the cookie from the request object
        Cookie userCookie = request.getCookies()[0];

        // Print the cookie value
        out.println("Cookie name: " + userCookie.getName());
        out.println("Cookie value: " + userCookie.getValue());
    }
}
```

What is MVC Architecture? Explain the role and purpose of every layer with example.

Model-View-Controller (MVC) is a software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements: the model, the view, and the controller.

Model:

- **Role:** Represents the application's data and business logic. It is responsible for managing the data, processing user inputs, and enforcing business rules.
- **Purpose:** Ensures the integrity of the application's data and provides methods for accessing and manipulating that data.
- **Example:** In a web-based task management application, the Task class could be part of the model. It would contain attributes such as task name, description, due date, and methods for updating and retrieving task information.

View:

- **Role:** Represents the presentation and visualization of the data. It is responsible for displaying the user interface and presenting information to the user.

- **Purpose:** Ensures a clear and user-friendly presentation of the data without containing business logic.
- **Example:** In the same task management application, a view could be a web page that displays a list of tasks, their details, and user interface elements for adding, editing, and deleting tasks.

Controller:

- **Role:** Acts as an intermediary between the model and the view. It receives user inputs, processes them, and updates the model and view accordingly.
- **Purpose:** Manages the flow of data between the model and the view, handles user interactions, and implements application logic.
- **Example:** A controller in the task management application would handle requests from the user interface, update the model based on user actions (e.g., adding or completing tasks), and instruct the view to update accordingly.

Differentiate between Java Bean and Enterprise Java Bean.

	Java Beans	**Enterprise Java Beans**
Purpose	Creating reusable software components	Building server-side components for enterprise apps
Usage	GUIs and reusable components	Enterprise-level applications
Complexity	Lightweight and simple	Sophisticated and feature-rich
Features	Basic property handling	Transaction management, security, concurrency, etc.
Deployment	Versatile	Java EE containers
Scalability	Suitable for small to medium-scale applications	Designed for large-scale enterprise applications
Concurrency Control	Limited support	Advanced concurrency control mechanisms
Remote Access	Limited support	Provides remote access

With the help of Servlet Program, differentiate between doGet() and doPost() function

	doGet()	doPost
HTTP method	GET	POST
Data transmission	URL parameters	HTTP request body
Data size	Limited	Unlimited
Idempotent	Yes	No
Caching	Can be cached	Should not be cached
Usage	Retrieving data, navigating between pages	Submitting forms, sending large amounts of data
Visibility	Data is visible in the URL	Data is not visible in the URL

```java
import java.io.IOException;
import javax.servlet.http.*;

public class SimpleDoGetAndDoPostServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException {
        response.setContentType("text/html");
        response.getWriter().println("<h1>doGet() method</h1>");
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException {
        response.setContentType("text/html");
        response.getWriter().println("<h1>doPost() method</h1>");
    }
}
```

Design a HTML Page for a Employee Management System containing key fields like:: Empld, Emp Name, Address, Salary, Designation etc. Write appropriate Java Script for it.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Employee Management System</title>
</head>
<body>
  <h1>Employee Management System</h1>
  <form id="employeeForm">
    <label for="empID">Employee ID:</label>
    <input type="text" id="empID" name="empID" required>

    <label for="empName">Employee Name:</label>
    <input type="text" id="empName" name="empName" required>

    <label for="empSalary">Employee Salary:</label>
    <input type="number" id="empSalary" name="empSalary" min="0" required>

    <label for="empDesignation">Employee Designation:</label>
    <select id="empDesignation" name="empDesignation" required>
      <option value="">Select Designation</option>
      <option value="Software Engineer">Software Engineer</option>
      <option value="Project Manager">Project Manager</option>
      <option value="Product Manager">Product Manager</option>
      <option value="QA Engineer">QA Engineer</option>
    </select>

    <button type="submit">Submit</button>
  </form>
```

```
  <script>
    const form = document.getElementById('employeeForm');

    form.addEventListener('submit', (event) => {
      event.preventDefault(); // Prevent form submission to a server

      const empID = document.getElementById('empID').value;
      const empName = document.getElementById('empName').value;
      const empSalary = parseFloat(document.getElementById('empSalary').value);
      const empDesignation = document.getElementById('empDesignation').value;

      // Validate input fields
      if (!empID || !empName || !empSalary || empDesignation === '') {
        alert('Please fill in all required fields');
        return;
      }

      if (isNaN(empSalary)) {
        alert('Invalid salary. Please enter a valid number');
        return;
      }
    });
  </script>
</body>
</html>
```

Write a JDBC Program to insert a record in the student table of the Access/Oracle database having fields like: EmpId, Emp Name, Address, Salary, Designation etc.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class SimpleInsertStudentRecord {

    public static void main(String[] args) {
        try {
            // Load the JDBC driver
            Class.forName("com.mysql.jdbc.Driver");

            // Establish a connection to the database
            Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/mydatabase",
"username", "password");

            // Prepare the SQL statement to insert a record
            String sql = "INSERT INTO student (empID, empName, address, salary,
designation) VALUES (?, ?, ?, ?, ?)";
            PreparedStatement preparedStatement =
connection.prepareStatement(sql);
```

```
            // Set the values for the prepared statement
            preparedStatement.setInt(1, 102); // Replace with actual employee
ID
            preparedStatement.setString(2, "Jane Smith"); // Replace with
actual employee name
            preparedStatement.setString(3, "456 Elm Street"); // Replace with
actual address
            preparedStatement.setDouble(4, 75000.00); // Replace with actual
salary
            preparedStatement.setString(5, "Project Manager"); // Replace with
actual designation

            // Execute the prepared statement and check if the insertion was
successful
            int rowsAffected = preparedStatement.executeUpdate();
            if (rowsAffected > 0) {
                System.out.println("Record inserted successfully!");
            } else {
                System.out.println("Record insertion failed.");
            }

            // Close the prepared statement and connection
            preparedStatement.close();
            connection.close();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

What are Servlets? Write a Servlet Program to Print the Request headers of the Web browser.

Servlets are Java classes that extend the capabilities of servers and respond to requests from web clients. They are part of the Java EE (Enterprise Edition) platform and provide a component-based, platform-independent method for building web-based applications.

```
import java.io.IOException;
import javax.servlet.http.*;

public class PrintRequestHeadersServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        // Get the request headers
        Enumeration<String> headerNames = request.getHeaderNames();

        // Print the headers and their values
        while (headerNames.hasMoreElements()) {
```

```
            String headerName = headerNames.nextElement();
            String headerValue = request.getHeader(headerName);
            response.getWriter().println(headerName + ": " + headerValue);
        }
    }
}
```

Write a JSP Program to print the string "GGSIP University" and IP Address of a Remote Client Machine.

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>GGSIP University</title>
</head>
<body>
    <h1>GGSIP University</h1>
    <p>Remote Client IP Address: <%= request.getRemoteAddr() %></p>
</body>
</html>
```

Design a XML Page for Selling an item on the Web Site. Assume appropriate attribute of item like: Item ID, Item Name, Item Price, Item Manufacture Name, Item Quantity etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<item>
  <itemID>12345</itemID> // Unique identifier for the item
  <itemName>Laptop</itemName> // Name of the item
  <itemPrice>1299.99</itemPrice> // Price of the item
  <itemManufacturer>Apple</itemManufacturer> // Manufacturer of the item
  <itemQuantity>50</itemQuantity> // Available quantity of the item
  <itemDescription>A sleek and powerful laptop</itemDescription> // Detailed
description of the item
  <itemImageURL>https://example.com/laptop.jpg</itemImageURL> // URL of the
item's image
</item>
```

Difference between HTTP and HTTPS.

HTTP	HTTPS
HTTP stands for HyperText Transfer Protocol	HTTPS for HyperText Transfer Protocol Secure
URL begins with “http://”	URL starts with “https://”
It works at Application Layer	It works at Transport Layer
Encryption is absent	Encryption is present
It does not require any certificate	It needs SSL Certificate

It is faster	It is slower
It does not improve search ranking	It improves search ranking
It should be avoided	It should be preferred

CSS sheets

CSS (Cascading Style Sheets) is a style sheet language used to style HTML elements.

Types of CSS:

- **Inline CSS:** It is defined directly within HTML elements using the style attribute.

```
<p style="color: red; font-size: 20px;">Hello World!</p>
```

- **Internal CSS:** It is defined within the <style> tag in the <head> section of an HTML document.

```
<head>
  <style>
    p {
      color: blue;
      font-weight: bold;
    }
  </style>
</head>
```

- **External CSS:** It is defined in a separate .css file that is linked to the HTML document using the <link> tag in the <head> section.

```
<head>
  <link rel="stylesheet" href="style.css">
</head>
```

Cookies

Cookies are small pieces of data that are stored on a user's computer by a website. They are used to remember information about the user, such as their login credentials and browsing preferences. Cookies can be used to improve the user experience by making it easier for users to access and personalize websites.

There are two main types of cookies:

- **Session cookies:** These cookies are temporary and are deleted when the user closes their browser. They are typically used to remember information, such as login credentials or shopping cart items.
- **Persistent cookies:** These cookies are stored on the user's computer for a longer period of time. They are typically used to remember information, such as their language settings or time zone.