mte answers

1. What is meaning of Synchronous request and Asynchronous request? Mention the working of AJAX Write the ways using that we can deploy an application. Write the steps.

Synchronous vs Asynchronous Requests

Tabular Form:

Feature	Synchronous Request	Asynchronous Request
Execution	Blocks execution until complete	Continues execution without waiting
User Interaction	User cannot interact during request	User can interact during request
Page Behavior	Entire page reloads	No page reload
Use Case Example	Form submission with page reload	Live search suggestions

Bullet Points:

Synchronous Request:

- Blocks further execution until the request is complete.
- The user cannot interact with the page during the request.
- Typically results in a full page reload.
- Example: Submitting a form that reloads the page to show the results.

Asynchronous Request:

- Allows execution to continue while waiting for the server's response.
- The user can continue to interact with the page during the request.
- The page does not reload; only part of the page updates.
- Example: Live search suggestions that appear as you type.

Working of AJAX

Bullet Points:

1. **Event Occurs:** A user action triggers a JavaScript event (e.g., button click).
2. **Create XMLHttpRequest Object:** JavaScript creates an `XMLHttpRequest` object.
3. **Send Request:** The `XMLHttpRequest` object sends a request to the server.
4. **Server Processes Request:** The server processes the request and sends a response.
5. **Receive Response:** JavaScript receives the response and updates the web page dynamically.

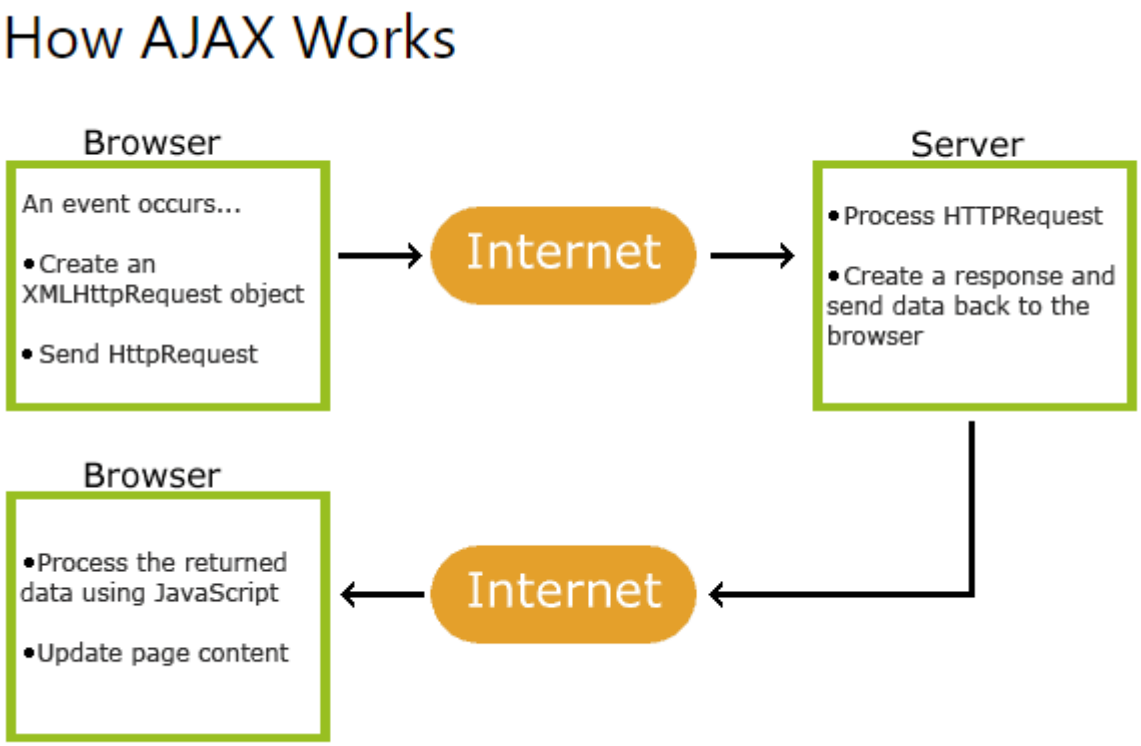
Deploying an Application

Tabular Form:

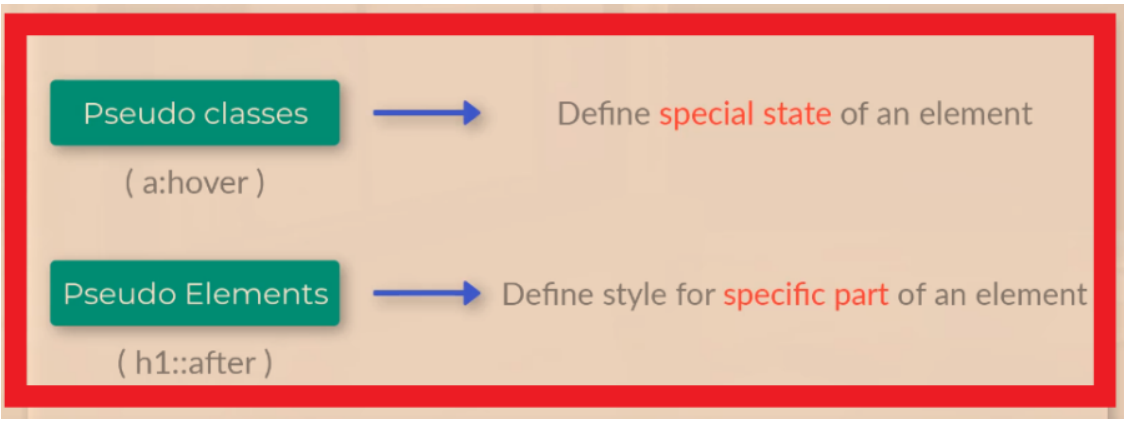
Step	Description
Prepare the Application	Ensure the app is complete and ready; bundle files.
Choose Hosting Provider	Select a hosting service (e.g., AWS, Heroku).
Set Up the Server	Configure server environment to meet app requirements.
Transfer Files	Use FTP/SFTP, Git, or deployment service to upload files to the server.
Configure the Application	Set up configurations like database connections and environment variables.
Run the Application	Start the application with appropriate commands.

mte answers

Step	Description
Testing	Access and test the application to ensure it works correctly.
Monitor and Maintain	Monitor for issues and perform regular updates and maintenance.



2. How can we use pseudo-classes and pseudo-elements in CSS? Provide examples. What is the meaning of HTTP Status Code. Explain codes 200, 500,400, 403 & 404. Explain jQuery with a suitable example.



HTTP Status Codes in Tabular Form

Status Code	Description	Example Scenario
200 OK	The request has succeeded.	Successfully retrieved a web page.
400 Bad Request	The server could not understand the request due to invalid syntax.	Sending a malformed request.
403 Forbidden	The client does not have access rights to the content.	Trying to access a restricted page without authorization.
404 Not Found	The server cannot find the requested	Requesting a URL that does not exist

mte answers

Status Code	Description	Example Scenario
	resource.	on the server.
500 Internal Server Error	The server encountered a situation it doesn't know how to handle.	A bug in server-side code causing the server to crash.

jQuery

- **jQuery**: A JavaScript library for simplifying HTML DOM manipulation, event handling, and animation.
- **Example**: Toggles paragraph visibility on button click.

Example Code:

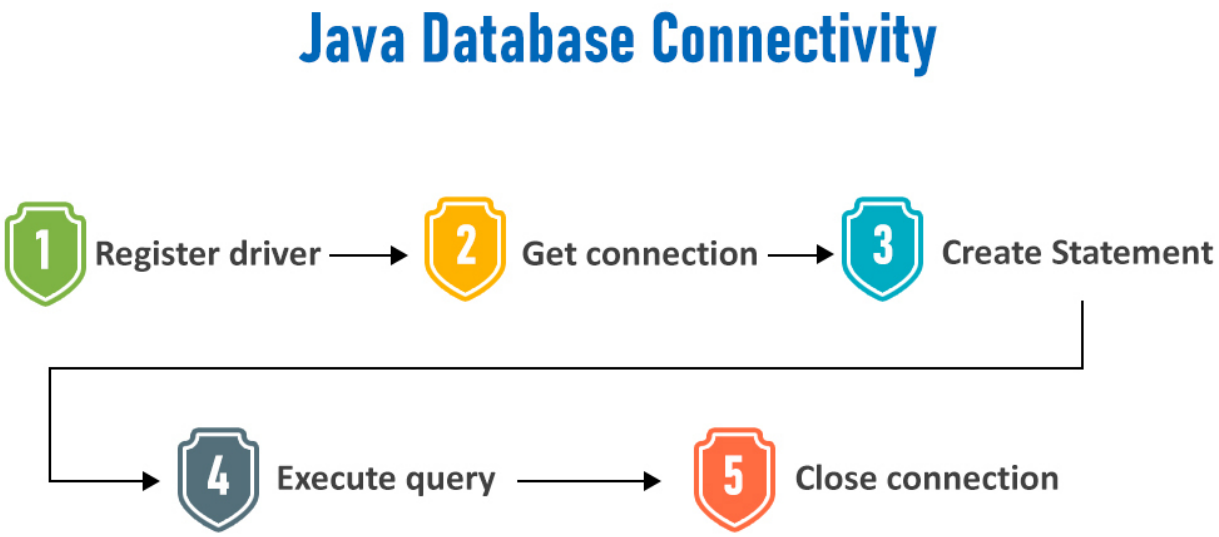
```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery Example</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script>
        $(document).ready(function() {
            $("button").click(function() {
                $("p").toggle();
            });
        });
    </script>
</head>
<body>
    <button>Toggle Paragraph</button>
    <p>This is a paragraph that will be toggled.</p>
</body>
</html>
```

3. What is JDBC? Write the steps to fetch data from a database. What is the purpose of <Java Server Pages (JSP):setProperty >? What is a bean ? use setter and setter method in a bean program.

JDBC (Java Database Connectivity)

JDBC is an API in Java that allows applications to interact with databases. It provides methods for querying and updating data in a database.

mte answers



Purpose of `<jsp:setProperty>`

jsp:setProperty is used in JavaServer Pages (JSP) to set the properties of a JavaBean. It allows you to set the value of a bean property from a request parameter or a specified value.

What is a Bean?

A **JavaBean** is a reusable software component that follows certain conventions:

- It should have a public default constructor.
- Properties should be accessed using getter and setter methods.
- It should be serializable.

Bean Example with Getter and Setter Methods

UserBean.java:

```
package com.example;

import java.io.Serializable;

public class UserBean implements Serializable {
    private String username;
    private String email;

    // Default constructor
    public UserBean() {}

    // Getter for username
    public String getUsername() {
        return username;
    }

    // Setter for username
    public void setUsername(String username) {
        this.username = username;
    }

    // Getter for email
    public String getEmail() {
```

```
        return email;
    }

    // Setter for email
    public void setEmail(String email) {
        this.email = email;
    }
}
```

4. With the help of Java script code block which checks the contents entered in a text box of a form. If the test entered is in the lower case, convert to upper case using the builtin function.

```
<!DOCTYPE html>
<html>
<head>
    <title>Text Box Uppercase Conversion</title>
    <script type="text/javascript">
        function convertToUpperCase() {
            // Get the text box element
            var textBox = document.getElementById("textInput");

            // Get the value entered in the text box
            var textValue = textBox.value;

            // Convert the value to uppercase
            var upperCaseValue = textValue.toUpperCase();

            // Set the converted value back to the text box
            textBox.value = upperCaseValue;
        }
    </script>
</head>
<body>
    <form>
        <label for="textInput">Enter Text:</label>
        <input type="text" id="textInput" name="textInput"
onblur="convertToUpperCase()">
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

5. Explain the use of JSP/ Servlet in web application. Write down the directory structure of the application using a servlet.

Use of JSP/Servlet in Web Applications

JavaServer Pages (JSP) and **Servlets** are server-side technologies used to create dynamic web applications in Java.

JSP (JavaServer Pages):

- **Purpose**: Simplifies the creation of dynamic web content.
- **Usage**:
  - Embeds Java code within HTML using special tags (`<% ... %>`).
  - Facilitates the separation of presentation and business logic.
  - Ideal for creating the view layer in MVC architecture.

mte answers

Servlets:

- **Purpose**: Handles client requests and generates responses.
- **Usage**:
  - Extends the `HttpServlet` class and overrides methods like `doGet()` and `doPost()`.
  - Ideal for processing business logic and managing control flow in web applications.

Directory Structure of a Web Application Using Servlet

A typical Java web application has a well-defined directory structure to organize its components. Here's a common structure:

```
MyWebApp/
├── build/
│   └── classes/
│       └── com/
│           └── example/
│               └── HelloServlet.class
├── src/
│   └── main/
│       └── java/
│           └── com/
│               └── example/
│                   └── HelloServlet.java
├── web/
│   ├── WEB-INF/
│   │   ├── classes/
│   │   ├── lib/
│   │   └── web.xml
│   ├── index.html
│   └── styles.css
└── build.xml
```

6. How can we handle the exceptions in Java Server Pages (JSP)? Explain with a suitable example. How is HTML 4.0 different from HTML 5.0.

- **Handling Exceptions in JSP**:
  - Use `errorPage` directive in JSP to handle exceptions.
  - Example:
    - Create `error.jsp` with error message display.
    - Define error handling in `web.xml` with `<error-page>`.
- **HTML 4.0 vs HTML 5.0**:
  - **HTML 4.0**: Older version, introduced frames, tables, forms.
  - **HTML 5.0**: Latest version, adds semantics, multimedia, graphics, forms, storage, APIs, and accessibility improvements.

7. Why do we use action tags in JSP? Explain with a suitable example.

mte answers

We use action tags in JavaServer Pages (JSP) to perform dynamic actions like including other resources, forwarding requests, and looping through collections. One of the commonly used action tags is the `<jsp:include>` tag.

Example Using `<jsp:include>` Action Tag

Suppose you have a JSP page (`main.jsp`) that includes a header and a footer. You can use the `<jsp:include>` tag to include these sections dynamically:

1. **main.jsp**:

```
jsp
Copy code
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Main Page</title>
</head>
<body>
    <!-- Include Header -->
    <jsp:include page="header.jsp" />

    <h1>Welcome to Main Page!</h1>
    <p>This is the main content of the page.</p>

    <!-- Include Footer -->
    <jsp:include page="footer.jsp" />
</body>
</html>
```

2. **header.jsp**:

```
jsp
Copy code
<header>
    <h1>Header Section</h1>
    <nav>
        <a href="home.jsp">Home</a>
        <a href="about.jsp">About</a>
        <a href="contact.jsp">Contact</a>
    </nav>
</header>
```

3. **footer.jsp**:

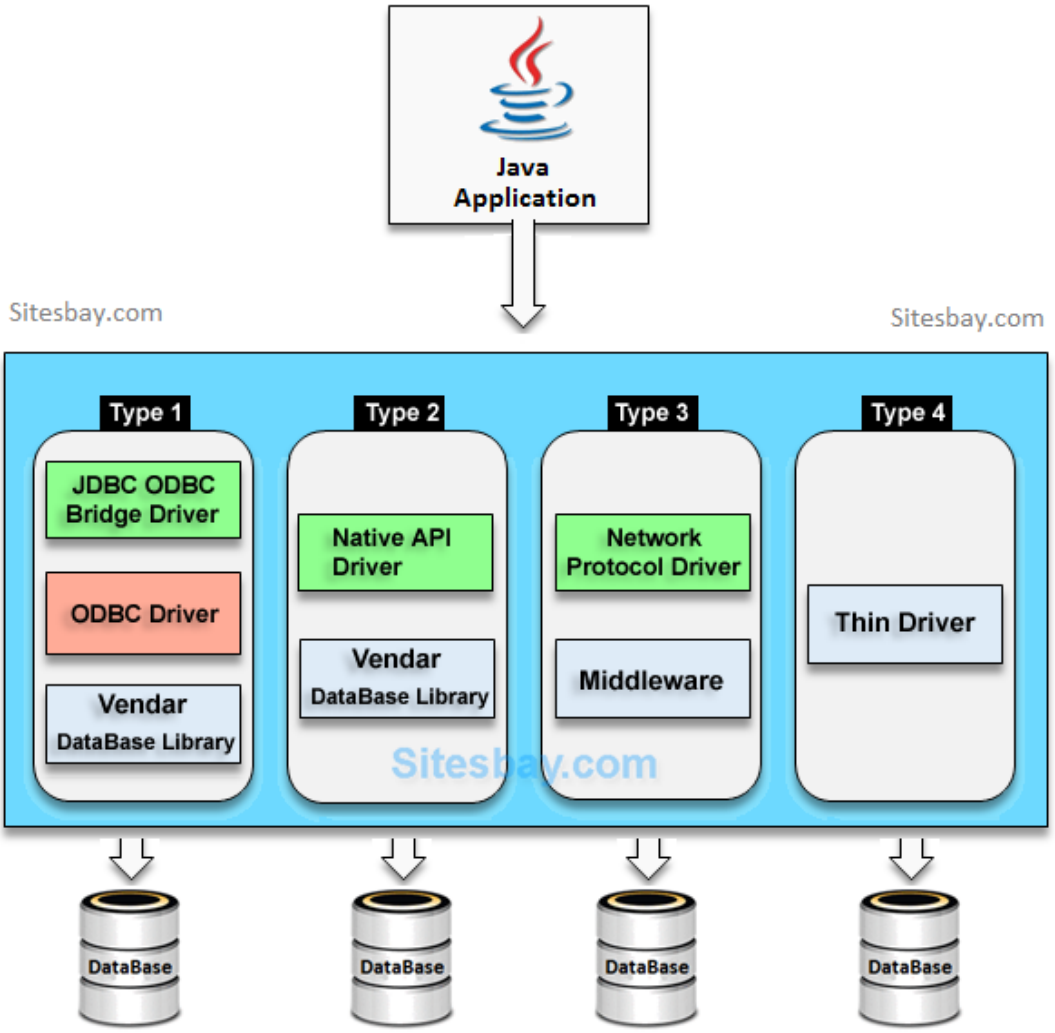
```
jsp
Copy code
<footer>
    <p>&copy; 2024 MyWebsite. All rights reserved.</p>
</footer>
```

8. What is the use of JDBC drivers? Explain different types of JDBC drivers.

9. different types of JDBC drivers:

mte answers

JDBC Driver Type	Description	Advantages	Disadvantages
Type 1: JDBC-ODBC Bridge	Uses ODBC (Open Database Connectivity) to communicate with databases.	Easy to use for simple applications.	Performance overhead due to translation from JDBC calls to ODBC calls. Limited portability.
Type 2: Native-API	Uses native API libraries provided by the database vendor.	Improved performance compared to Type 1 drivers. Direct interaction with the database's native API.	Platform-dependent. Requires client-side installation of database-specific libraries.
Type 3: Network Protocol	Communicates with a middle-tier server using a network protocol.	Platform-independent. Supports multiple databases through middleware layer.	Performance overhead due to additional network communication. Requires separate middleware server installation.
Type 4: Thin Driver	Communicates directly with the database using a vendor-specific protocol.	Platform-independent. Better performance due to direct communication. No client-side installation.	May not support all database features if vendor-specific protocol is limited. Database-specific, requires driver for each database vendor.



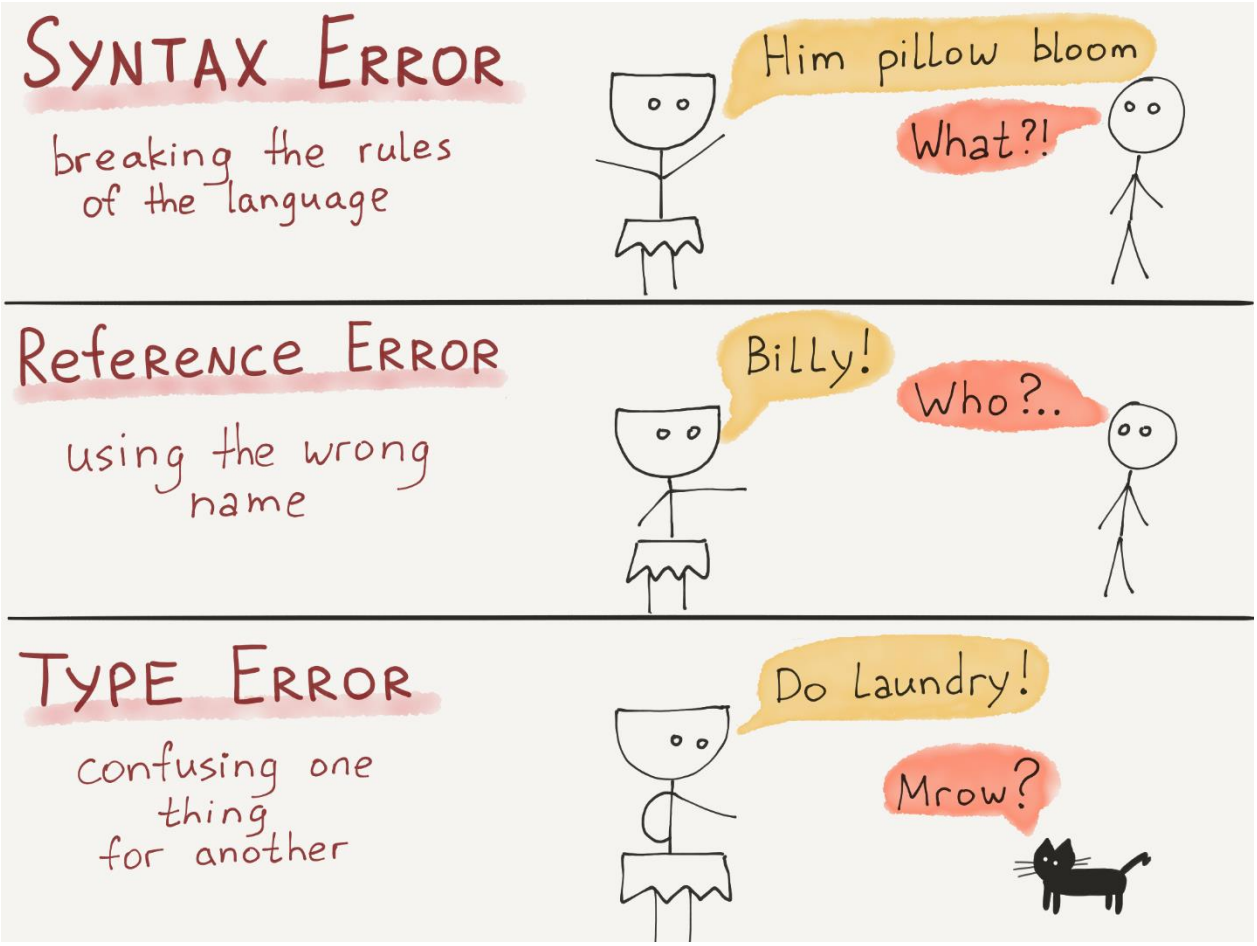
mte answers

10. How can we create a dedicated error page in Java Server Pages (JSP)?

Create the Error JSP Page (error.jsp): Create a JSP page named `error.jsp` in your web application's directory. This page will display the error message to the user.

```
<%@ page isErrorPage="true" %>
<!DOCTYPE html>
<html>
<head>
    <title>Error Page</title>
</head>
<body>
    <h1>Error Occurred!</h1>
    <p><strong>Error Message:</strong> <%= exception.getMessage() %></p>
</body>
</html>
```

11. What are the different types of Javascript errors?



Error Type	Description	Example
Syntax Errors	Occur due to syntax issues in the code.	Missing semicolon at the end of a statement: `var x = 10` instead of `var x = 10;`.
Referenc	Occur when	Accessing an undefined variable: `console.log(y);` where `y` is not

mte answers

Error Type	Description	Example
e Errors	trying to access undefined variables or object properties.	declared.
Type Errors	Occur when an operation is performed on an inappropriate data type.	Trying to call a non-function as a function: `myFunction();` where `myFunction` is not a function.
Range Errors	Occur when using a value outside the allowable range.	Using an index that is out of bounds in an array: `var arr = [1, 2, 3]; console.log(arr[3]);`
Eval Errors	Occur when there is an issue with the `eval()` function.	Incorrect syntax or logic within the `eval()` function: `eval("alert('Hello World!')");`
URI Errors	Occur when there are issues with encoding or decoding URIs.	Using illegal characters in a URI: `window.location.href = "http://example.com/?name=John@Doe";`
Network Errors	Occur when there are issues with network requests.	Failed network request due to server issues: `fetch('https://example.com/api/data').then(...).catch(...);`
Async Errors	Occur in asynchronous code execution, such as promises or async functions.	Promise rejection without proper error handling: `fetch('https://example.com/api/data').then(...).catch(...);`

12. Compare iteration and recursion using Javascript.

- **Iteration** uses loops (`for`, `while`, etc.) to repeatedly execute code until a condition is met.

- **Recursion** involves calling a function within itself, breaking down problems into smaller subproblems until a base case is reached.

mte answers

- **Iteration** is often more efficient for known iterations or simple tasks.

- **Recursion** is elegant for certain problems but may use more memory due to function call stack frames.

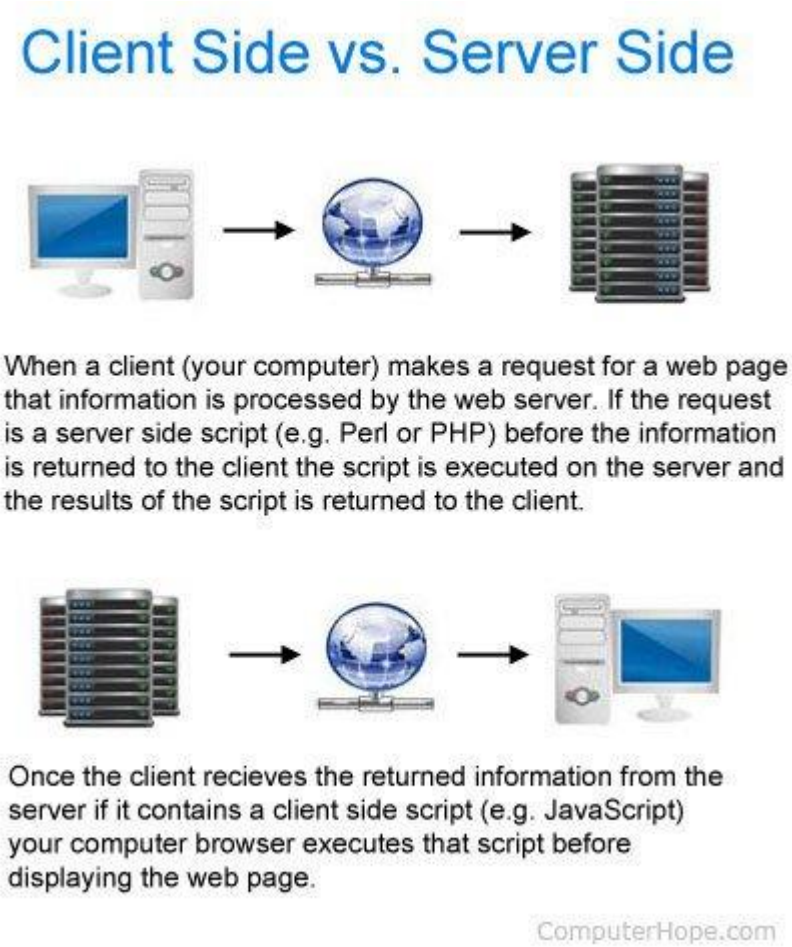
- **Example (Iteration)**:

```
for (let i = 0; i < 5; i++) {
    console.log(i);
}
```

- **Example (Recursion)**:

```
function factorial(n) {
    if (n === 0 || n === 1) return 1;
    return n * factorial(n - 1);
}
console.log(factorial(5)); // Outputs: 120
```

13. Why do we use client side scripting? Also explain the importance of server side scripting.



tabular comparison of client-side scripting and server-side scripting:

Aspect	Client-Side Scripting	Server-Side Scripting
Purpose	Enhances user experience and	Handles backend logic and processing on

mte answers

Aspect	Client-Side Scripting	Server-Side Scripting
	interactivity on the client side (web browser).	the server before sending data to the client.
Execution Environment	Executes on the user's web browser.	Executes on the server where the web application is hosted.
Key Uses	- Dynamic Content (AJAX)	- Database Interaction
	- Form Validation	- User Authentication
	- User Interface Enhancements	- Dynamic Content Generation
	- Cookies Handling	- Security (Access Control, Encryption)
Importance	Improves responsiveness, reduces server load, enhances user interaction.	Ensures data integrity, security, scalability, efficient server-client communication.
Examples	- Updating web page content dynamically.	- Retrieving data from databases.
	- Validating user inputs in real time.	- Authenticating users and managing sessions.
	- Creating interactive elements (sliders, dropdowns).	- Generating HTML content based on user requests.

Both client-side scripting and server-side scripting play critical roles in web development, each focusing on different aspects of the web application and contributing to a seamless user experience.

14. Write a Servlet which includes the content of another Servlet (ie data of other servlet is coming to your servlet).

to include the content of another Servlet within your Servlet, you can use the `RequestDispatcher` interface in Java Servlets. Here's an example Servlet code that demonstrates how to do this:

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.RequestDispatcher;

public class IncludeServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // Content of another Servlet (e.g., "OtherServlet") is included
here
        RequestDispatcher dispatcher =
request.getRequestDispatcher("/OtherServlet");
        dispatcher.include(request, response);
    }
}
```

15. Write a Servlet which forwards request to another servlet

mte answers

To forward a request from one Servlet to another Servlet, you can use the `RequestDispatcher` interface in Java Servlets. Here's an example Servlet code that demonstrates how to do this:

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.RequestDispatcher;

public class ForwardServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // Forward the request to another Servlet (e.g., "OtherServlet")
        RequestDispatcher dispatcher =
request.getRequestDispatcher("/OtherServlet");
        dispatcher.forward(request, response);
    }
}
```

16. Write a Servlet which receives initialization values from web.xml

17. Construct an HTML code using various text formatting

    tags List some features of jQuery.

HTML code that includes various text formatting tags:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Text Formatting Example</title>
</head>
<body>
    <h1>This is a Heading</h1>
    <p>This is a paragraph with <strong>strong</strong>, <em>emphasized</em>,
and <u>underlined</u> text.</p>
    <p>You can also use <sup>superscript</sup> and <sub>subscript</sub>
text.</p>
    <p>Text can be <span style="color: red;">colored</span> and <span
style="font-size: 20px;">styled</span>.</p>
    <p>You can create <a href="https://www.example.com">links</a> and add
<abbr title="Hypertext Markup Language">HTML</abbr> abbreviations.</p>
    <p>Additionally, you can use <code>code</code> and <kbd>keyboard
input</kbd> formatting.</p>
</body>
</html>
```

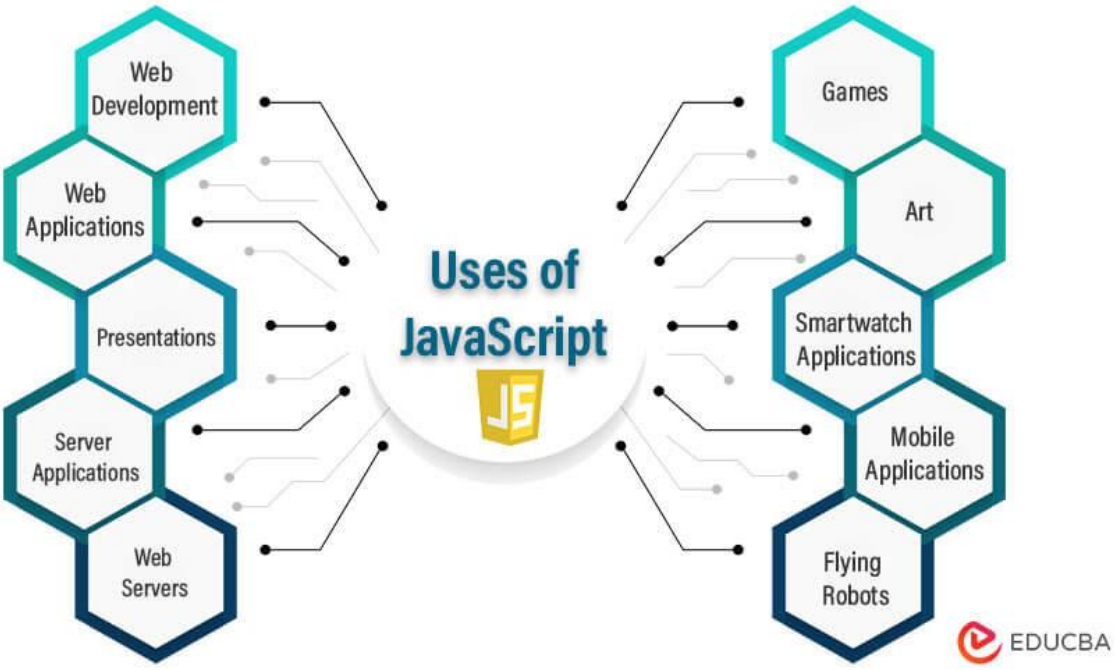
This HTML code demonstrates the use of various text formatting tags such as `<strong>`, `<em>`, `<u>`, `<sup>`, `<sub>`, `<span>`, `<a>`, `<abbr>`, `<code>`, and `<kbd>` to format text in different ways.

As for some features of jQuery:

mte answers

- **DOM Manipulation**: jQuery simplifies DOM traversal and manipulation tasks, making it easier to interact with HTML elements.
- **Event Handling**: jQuery provides efficient methods for event handling, such as `click`, `hover`, `submit`, etc., improving interactivity in web applications.
- **AJAX Support**: jQuery simplifies AJAX calls, allowing developers to fetch data from servers and update parts of a web page without reloading the entire page.
- **Animations**: jQuery includes animation methods to create smooth transitions and effects on web elements, enhancing user experience.
- **Utilities**: jQuery offers various utility functions for tasks like string manipulation, array manipulation, and working with asynchronous tasks.
- **Cross-Browser Compatibility**: jQuery handles cross-browser compatibility issues, ensuring consistent behavior across different web browsers.
- **Plugins**: jQuery has a vast ecosystem of plugins that extend its functionality, providing solutions for various tasks such as form validation, sliders, and more.

18. Discover the benefits of using JavaScript



19. Outline the various http methods used to send an html form's data to server in details.Explain with the help of a program.

HTTP provides several methods to send HTML form data to a server. The most commonly used methods are GET and POST

- **GET Method**:
  - Used for retrieving data from the server.
  - Appends form data to the URL as query parameters.
  - Suitable for small amounts of data and non-sensitive information.
  - Visible in the browser's address bar.
- **POST Method**:
  - Used for sending large amounts of data and sensitive information.

mte answers

- Sends form data in the body of the HTTP request.
- Not visible in the browser's address bar.
- More secure than GET for sensitive data.

HTML Form (index.html):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>HTML Form Example</title>
</head>
<body>
    <h1>HTML Form</h1>
    <form action="processFormData" method="GET">
        <label for="name">Enter your name:</label>
        <input type="text" id="name" name="name">
        <input type="submit" value="Submit">
    </form>
    <form action="processFormData" method="POST">
        <label for="email">Enter your email:</label>
        <input type="email" id="email" name="email">
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

20. What is the utility of Spring boot in web programming? Explain its working

Aspect	Spring Boot in Web Programming
Simplified Development	- Provides defaults and reduces configuration overhead.
	- Focuses on writing business logic rather than setup.
Auto-Configuration	- Automatically configures components based on dependencies.
	- Saves time by setting up common functionalities like database connectivity.
Embedded Server	- Includes an embedded web server (Tomcat, Jetty, etc.).
	- Simplifies deployment as standalone JAR files.
Dependency Management	- Offers starter dependencies for common libraries and frameworks.
	- Eases dependency inclusion and management.
Spring Boot Actuator	- Provides monitoring and management capabilities for applications.
	- Includes features like health checks, metrics, and environment information.
Integration with Spring	- Seamlessly integrates with Spring MVC, Spring Security, Spring Data, etc.
Ecosystem	- Leverages the power of the entire Spring ecosystem for building robust web apps.

mte answers

21. Write an Servlet / Java Server Pages (JSP) program to calculate Perimeter and area using post & get method

RectangleServlet.java:

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class RectangleServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        int length = Integer.parseInt(request.getParameter("length"));
        int width = Integer.parseInt(request.getParameter("width"));

        int perimeter = 2 * (length + width);
        int area = length * width;

        response.setContentType("text/html");
        response.getWriter().println("<html><body>");
        response.getWriter().println("<h1>Rectangle Perimeter and Area
(POST)</h1>");
        response.getWriter().println("<p>Length: " + length + "</p>");
        response.getWriter().println("<p>Width: " + width + "</p>");
        response.getWriter().println("<p>Perimeter: " + perimeter +
"</p>");
        response.getWriter().println("<p>Area: " + area + "</p>");
        response.getWriter().println("</body></html>");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        int length = Integer.parseInt(request.getParameter("length"));
        int width = Integer.parseInt(request.getParameter("width"));

        int perimeter = 2 * (length + width);
        int area = length * width;

        response.setContentType("text/html");
        response.getWriter().println("<html><body>");
        response.getWriter().println("<h1>Rectangle Perimeter and Area
(GET)</h1>");
        response.getWriter().println("<p>Length: " + length + "</p>");
        response.getWriter().println("<p>Width: " + width + "</p>");
        response.getWriter().println("<p>Perimeter: " + perimeter +
"</p>");
        response.getWriter().println("<p>Area: " + area + "</p>");
        response.getWriter().println("</body></html>");
    }
}
```

22. Write a Servlet code snippet using PreparedStatement to insert data into a table in a MySQL/ Oracle database.

Here's a Java Servlet code snippet using PreparedStatement to insert data into a table in a MySQL database:

```
import java.io.IOException;
import java.io.PrintWriter;
```

mte answers

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/InsertDataServlet")
public class InsertDataServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final String JDBC_URL =
"jdbc:mysql://localhost:3306/mydatabase";
    private static final String JDBC_USER = "username";
    private static final String JDBC_PASSWORD = "password";

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String name = request.getParameter("name");
        int age = Integer.parseInt(request.getParameter("age"));

        Connection conn = null;
        PreparedStatement pstmt = null;
        PrintWriter out = response.getWriter();
        try {
            // Register JDBC driver and establish connection
            Class.forName("com.mysql.cj.jdbc.Driver");
            conn = DriverManager.getConnection(JDBC_URL, JDBC_USER,
JDBC_PASSWORD);

            // Prepare SQL statement with placeholders
            String sql = "INSERT INTO users (name, age) VALUES (?, ?)";
            pstmt = conn.prepareStatement(sql);

            // Set values for placeholders
            pstmt.setString(1, name);
            pstmt.setInt(2, age);

            // Execute the SQL statement
            int rowsInserted = pstmt.executeUpdate();
            if (rowsInserted > 0) {
                out.println("Data inserted successfully!");
            } else {
                out.println("Failed to insert data.");
            }
        } catch (ClassNotFoundException | SQLException e) {
            out.println("Error: " + e.getMessage());
        } finally {
            // Close PreparedStatement and Connection
            try {
                if (pstmt != null) pstmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                out.println("Error in closing resources: " +
e.getMessage());
            }
        }
    }
}
```

23. What is the use of Spring and Spring Boot in web application. Explain with a suitable example

mte answers

tabular comparison of Spring and Spring Boot in web application development:

Aspect	Spring Framework	Spring Boot
Purpose	Comprehensive framework for enterprise-level applications	Simplified framework for rapid development
Dependency Injection	Provides Dependency Injection (DI) for managing dependencies	Uses auto-configuration for managing dependencies
Aspect-Oriented	Supports Aspect-Oriented Programming (AOP)	AOP capabilities for handling cross-cutting concerns
Web Development	Provides Spring MVC for web application development	Includes embedded servers and auto-configuration
Transaction Management	Offers declarative transaction management	Supports transaction management with annotations
Integrations	Integrates with various technologies (Hibernate, JPA, JDBC)	Offers starter dependencies for common tasks
Production Features	Provides features like security, messaging, and more	Includes production-ready features like metrics

Both frameworks are part of the larger Spring ecosystem and are used based on project requirements. Spring is suitable for complex applications requiring extensive configuration and integration, while Spring Boot is preferred for fast development, microservices, and cloud-native applications.

24. Write a Servlet/ Java Server Pages (JSP) program to validate user login. User data is available in database

LoginServlet.java:

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static final String JDBC_URL =
"jdbc:mysql://localhost:3306/mydatabase";
    private static final String JDBC_USER = "username";
    private static final String JDBC_PASSWORD = "password";

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        PrintWriter out = response.getWriter();
```

```
        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            conn = DriverManager.getConnection(JDBC_URL, JDBC_USER,
JDBC_PASSWORD);
            String sql = "SELECT * FROM users WHERE username=? AND
password=?";
            pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, username);
            pstmt.setString(2, password);
            rs = pstmt.executeQuery();
            if (rs.next()) {
                out.println("<html><body>");
                out.println("<h1>Login Successful!</h1>");
                out.println("</body></html>");
            } else {
                out.println("<html><body>");
                out.println("<h1>Login Failed! Invalid credentials.</h1>");
                out.println("</body></html>");
            }
        } catch (ClassNotFoundException | SQLException e) {
            out.println("Error: " + e.getMessage());
        } finally {
            try {
                if (rs != null) rs.close();
                if (pstmt != null) pstmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                out.println("Error in closing resources: " +
e.getMessage());
            }
        }
    }
}
```

25. Write a Servlet/ Java Server Pages (JSP) program to maintain session using session object, Hidden Form Field and URL Rewriting

SessionManagementServlet.java:

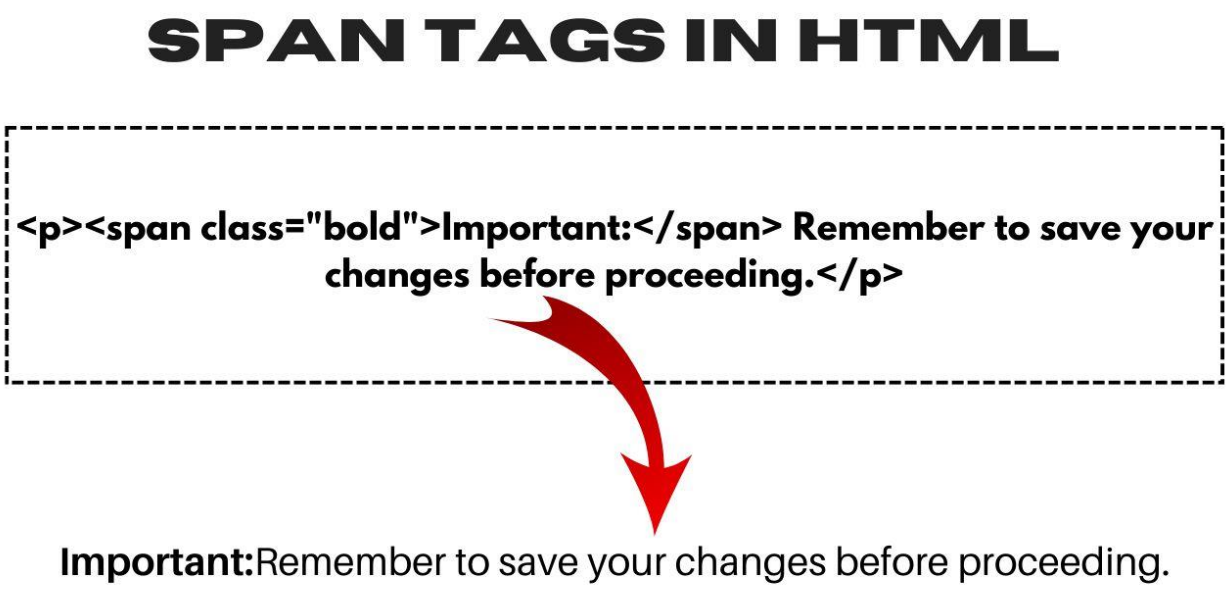
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/SessionManagementServlet")
public class SessionManagementServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpSession session = request.getSession();
        PrintWriter out = response.getWriter();

        // Using Session Object
        session.setAttribute("username", "JohnDoe");
```

mte answers

```
        // Using Hidden Form Field
        out.println("<html><body>");
        out.println("<form action='SessionManagementServlet'
method='post'>");
        out.println("<input type='hidden' name='hiddenField'
value='hiddenValue'>");
        out.println("<input type='submit' value='Submit Form'>");
        out.println("</form>");
        out.println("</body></html>");

        // Using URL Rewriting
        out.println("<a href='SessionManagementServlet?param=value'>Click
here for URL Rewriting</a>");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpSession session = request.getSession();
        String hiddenFieldValue = request.getParameter("hiddenField");
        String urlParamValue = request.getParameter("param");

        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>Session Management Example</h1>");
        out.println("<p>Username from session: " +
session.getAttribute("username") + "</p>");
        out.println("<p>Hidden Field Value: " + hiddenFieldValue + "</p>");
        out.println("<p>URL Parameter Value: " + urlParamValue + "</p>");
        out.println("</body></html>");
    }
}
```

26. Explain the use of SPAN in HTML with an example?

The `<span>` tag in HTML is used to apply styles or to group inline elements together without affecting the layout of the page.



27. What is the utility of Spring in web programming?

mte answers

28. summarizing the utility of Spring in web programming:

Utility	Description
Dependency Injection (DI)	Manages object dependencies, reducing tight coupling and promoting modular, reusable code.
Aspect-Oriented Programming (AOP)	Separates cross-cutting concerns from business logic, enhancing code organization and maintenance.
Spring MVC	Offers a robust model-view-controller architecture with features like handler mapping, view resolution, data binding, and validation.
Transaction Management	Supports declarative transaction management using annotations or XML, integrates with JDBC, Hibernate, JPA, etc., for seamless transaction handling.
Integration	Integrates with various technologies like ORM frameworks, messaging systems, and web services, providing seamless integration through configuration and API support.
Testing	Facilitates unit and integration testing with mock objects, testing annotations, and integration with popular testing frameworks.
Security	Provides robust authentication and authorization through Spring Security, offering features like user authentication, access control, CSRF protection, and more.
Simplifies Java EE Development	Offers lightweight alternatives to traditional Java EE components, simplifying enterprise development with lightweight containers and simplified configurations.

Explain Create a table inside a table

```
<!DOCTYPE html>
<html>
<body>
    <table border="1">
        <tr>
            <th>Outer Header 1</th>
            <th>Outer Header 2</th>
            <th>Outer Header 3</th>
        </tr>
        <tr>
            <td>Outer Cell 1</td>
            <td colspan="2">
                <table border="1">
                    <tr>
                        <th>Inner Header 1</th>
                        <th>Inner Header 2</th>
                    </tr>
                    <tr>
                        <td>Inner Cell 1</td>
                        <td>Inner Cell 2</td>
                    </tr>
                </table>
            </td>
        </tr>
        <tr>
            <td>Outer Cell 2</td>
            <td>Outer Cell 3</td>
            <td>Outer Cell 4</td>
        </tr>
    </table>
</body>
</html>
```

mte answers

```html
<!DOCTYPE html>
<html>
<body>
    <table border="1">
        <tr>
            <th>Outer Header 1</th>
            <th>Outer Header 2</th>
            <th>Outer Header 3</th>
        </tr>
        <tr>
            <td>Outer Cell 1</td>
            <td colspan="2">
                <table border="1">
                    <tr>
                        <th>Inner Header 1</th>
                        <th>Inner Header 2</th>
                    </tr>
                    <tr>
                        <td>Inner Cell 1</td>
                        <td>Inner Cell 2</td>
                    </tr>
                </table>
            </td>
        </tr>
        <tr>
            <td>Outer Cell 2</td>
            <td>Outer Cell 3</td>
            <td>Outer Cell 4</td>
        </tr>
    </table>
</body>
</html>
```

Outer Header 1	Outer Header 2	Outer Header 3
Outer Cell 1	Inner Header 1	Inner Header 2
	Inner Cell 1	Inner Cell 2
Outer Cell 2	Outer Cell 3	Outer Cell 4

29. Create an HTML form in which accept marks of 5 subjects and submit marks to a servlet. Calculate the division of the students based upon following criterion

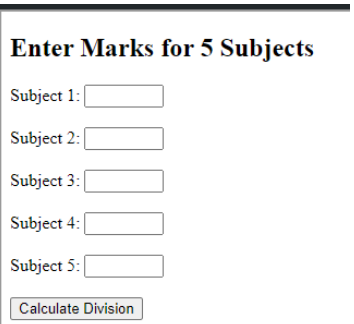
Marks >= 60 First Division
Marks <60 && >= 45 Second
Division Marks <45 && > 33
Third Division

Marks < 33 Fail

```html
<!DOCTYPE html>
<html>

<head>
    <title>Student Division Calculator</title>
</head>

<body>
    <h2>Enter Marks for 5 Subjects</h2>
    <form action="DivisionCalculatorServlet" method="post">
        <label for="subject1">Subject 1:</label>
        <input type="number" id="subject1" name="subject1" min="0" max="100" required><br><br>

        <label for="subject2">Subject 2:</label>
        <input type="number" id="subject2" name="subject2" min="0" max="100" required><br><br>

        <label for="subject3">Subject 3:</label>
        <input type="number" id="subject3" name="subject3" min="0" max="100" required><br><br>

        <label for="subject4">Subject 4:</label>
        <input type="number" id="subject4" name="subject4" min="0" max="100" required><br><br>

        <label for="subject5">Subject 5:</label>
        <input type="number" id="subject5" name="subject5" min="0" max="100" required><br><br>

        <input type="submit" value="Calculate Division">
    </form>
</body>

</html>
```

mte answers

```html
<!DOCTYPE html>
<html>

<head>
    <title>Student Division Calculator</title>
</head>

<body>
    <h2>Enter Marks for 5 Subjects</h2>
    <form action="DivisionCalculatorServlet" method="post">
        <label for="subject1">Subject 1:</label>
        <input type="number" id="subject1" name="subject1" min="0" max="100" required><br><br>

        <label for="subject2">Subject 2:</label>
        <input type="number" id="subject2" name="subject2" min="0" max="100" required><br><br>

        <label for="subject3">Subject 3:</label>
        <input type="number" id="subject3" name="subject3" min="0" max="100" required><br><br>

        <label for="subject4">Subject 4:</label>
        <input type="number" id="subject4" name="subject4" min="0" max="100" required><br><br>

        <label for="subject5">Subject 5:</label>
        <input type="number" id="subject5" name="subject5" min="0" max="100" required><br><br>

        <input type="submit" value="Calculate Division">
    </form>
</body>

</html>
```



30. State with an example the various way to add JavaScript and CSS to HTML.

In HTML, JavaScript and CSS can be added in various ways:

- **Inline**: Directly within HTML tags using attributes like `style` or `onclick`.
- **Internal**: Placed within `<style>` or `<script>` tags in the HTML file.
- **External**: Linked via `<link>` or `<script>` tags with references to external CSS/JS files.
- **CDN**: Utilizing Content Delivery Networks for external libraries like Bootstrap or jQuery using CDN URLs in `<link>` or `<script>` tags.

31. Write a Servlet/ Java Server Pages (JSP) program showing Page Navigation using Send redirect & RequestDispatcher

RequestDispatcher:

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/PageNavigationServlet")
public class PageNavigationServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String action = request.getParameter("action");

        if ("sendRedirect".equals(action)) {
            // Using sendRedirect to navigate to another page
            response.sendRedirect("redirectedPage.jsp");
        } else if ("requestDispatcher".equals(action)) {
            // Using RequestDispatcher to forward to another page

request.getRequestDispatcher("forwardedPage.jsp").forward(request,
response);
        } else {
            // Default action
            response.getWriter().println("Invalid action parameter.");
        }
    }
}
```

mte answers

32. Develop an application using AJAX and Java Server Pages (JSP) that contains html page that accept email and check and display a message email is present in the database or not. Create an user and store data in database

1. **HTML Page (index.html)**: Create an HTML form to accept the email and display the result using AJAX.

```
<!DOCTYPE html>
<html>
<head>
    <title>Email Check</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scrip
t>
    <script>
        $(document).ready(function(){
            $("#checkEmailForm").submit(function(event){
                event.preventDefault();
                var email = $("#email").val();
                $.ajax({
                    type: "POST",
                    url: "CheckEmail.jsp",
                    data: { email: email },
                    success: function(response){
                        $("#result").html(response);
                    }
                });
            });
        });
    </script>
</head>
<body>
    <h2>Email Check</h2>
    <form id="checkEmailForm">
        <label for="email">Enter Email:</label>
        <input type="email" id="email" name="email" required>
        <input type="submit" value="Check">
    </form>
    <div id="result"></div>
</body>
</html>
```

2. **Java Servlet (CheckEmailServlet.java)**: Create a servlet to handle AJAX requests and check if the email exists in the database.

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/CheckEmailServlet")
public class CheckEmailServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String email = request.getParameter("email");
        boolean emailExists = checkEmailInDatabase(email);
        PrintWriter out = response.getWriter();
        if (emailExists) {
            out.println("Email exists in the database.");
        } else {
```

```
            out.println("Email does not exist in the database.");
        }
    }

    private boolean checkEmailInDatabase(String email) {
        // Code to check if email exists in the database
        // Return true if email exists, false otherwise
        return false;
    }
}
```

3. **JSP Page (CheckEmail.jsp)**: Create a JSP page to process the AJAX request and communicate with the servlet.

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ page import="java.sql.*" %>
<%@ page import="javax.naming.*" %>
<%@ page import="javax.sql.*" %>
<!DOCTYPE html>
<html>
<head>
    <title>Email Check Result</title>
</head>
<body>
    <%
        String email = request.getParameter("email");
        boolean emailExists = checkEmailInDatabase(email);
        if (emailExists) {
            out.println("Email exists in the database.");
        } else {
            out.println("Email does not exist in the database.");
        }

        // Method to check if email exists in the database
        private boolean checkEmailInDatabase(String email) {
            boolean exists = false;
            try {
                Context context = new InitialContext();
                DataSource dataSource = (DataSource)
context.lookup("java:comp/env/jdbc/myDataSource");
                Connection connection = dataSource.getConnection();
                PreparedStatement statement =
connection.prepareStatement("SELECT * FROM users WHERE email = ?");
                statement.setString(1, email);
                ResultSet resultSet = statement.executeQuery();
                if (resultSet.next()) {
                    exists = true;
                }
                resultSet.close();
                statement.close();
                connection.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
            return exists;
        }
    %>
</body>
</html>
```

33. How do we use custom tag in Java Server Pages (JSP)? Formulate steps to create a custom tag.

mte answers

Creating a custom tag in JSP involves these steps:

1. **Tag Handler Class**: Create a Java class that extends `SimpleTagSupport` or implements `Tag`. Override `doTag()` to define the tag's behavior.
2. **Tag Library Descriptor (TLD)**: Write an XML file (.tld) defining tag names, classes, attributes, etc.
3. **Declare in JSP**: Use `<%@ taglib %>` directive to declare the tag library and specify the location of the TLD file.
4. **Use the Tag**: In JSP, use the custom tag by specifying its name and attributes as defined in the TLD.

34. Elaborate Document Object Model (DOM) with different JavaScript objects. Write a program to show system information and navigation information

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the structure of a document as a tree of objects, allowing scripts to dynamically access and manipulate the content, structure, and style of the document. In JavaScript, the DOM is accessed through various objects and methods:

1. **Document Object**: Represents the entire HTML document.
   - `document.getElementById('id')`: Gets an element by its ID.
   - `document.getElementsByTagName('tag')`: Gets elements by tag name.
   - `document.createElement('tag')`: Creates a new element.
   - `document.createTextNode('text')`: Creates a new text node.
2. **Element Object**: Represents an element in the document.
   - `element.getAttribute('attr')`: Gets the value of an attribute.
   - `element.setAttribute('attr', 'value')`: Sets the value of an attribute.
   - `element.appendChild(child)`: Appends a child node to the element.
   - `element.style.property = 'value'`: Sets inline CSS styles.
3. **Node Object**: Represents a node in the DOM tree.
   - `node.parentNode`: Gets the parent node of the current node.
   - `node.childNodes`: Gets an array-like collection of child nodes.
   - `node.firstChild`, `node.lastChild`: Gets the first/last child node.
   - `node.nextSibling`, `node.previousSibling`: Gets the next/previous sibling node.
4. **Event Object**: Represents an event triggered by user actions or browser actions.
   - `event.target`: Gets the element that triggered the event.
   - `event.preventDefault()`: Prevents the default action of an event.
   - `element.addEventListener('event', handler)`: Adds an event listener to an element.

Here's an example program demonstrating system information and navigation information using the DOM in JavaScript:

```
<!DOCTYPE html>
<html>
<head>
    <title>DOM Example</title>
    <script>
        // Function to display system information
        function displaySystemInfo() {
            var systemInfo = document.getElementById('systemInfo');
```

```
            systemInfo.innerHTML = 'User Agent: ' + navigator.userAgent +
'<br>';
            systemInfo.innerHTML += 'Platform: ' + navigator.platform;
        }

        // Function to handle navigation
        function navigate(url) {
            window.location.href = url;
        }
    </script>
</head>
<body>
    <h2>System Information</h2>
    <button onclick="displaySystemInfo()">Show System Info</button>
    <div id="systemInfo"></div>

    <h2>Navigation</h2>
    <button onclick="navigate('https://www.example.com')">Go to
Example</button>
</body>
</html>
```

35. What is the meaning of implicit objects of Java Server Pages (JSP). Show the uses with a suitable program.

Implicit objects in Java Server Pages (JSP) are pre-defined objects provided by the JSP container that can be used directly in JSP pages without needing to be declared or instantiated. These objects represent different aspects of the JSP environment and the request-processing lifecycle. Some commonly used implicit objects in JSP are:

1. **request**: Represents the client's HTTP request.
2. **response**: Represents the HTTP response that the server sends back to the client.
3. **session**: Represents the client's session, which allows maintaining state across multiple requests from the same client.
4. **out**: Represents the output stream for writing content to the response.
5. **config**: Represents the servlet configuration information for the JSP.
6. **pageContext**: Provides access to various objects and information related to the current page.
7. **page**: Represents the generated servlet class corresponding to the JSP page.
8. **exception**: Represents the exception object if an exception occurs during the execution of the JSP page.

Here's a simple program demonstrating the use of some implicit objects in JSP:

```
jsp
Copy code
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Implicit Objects Example</title>
</head>
<body>
    <h2>Request Information</h2>
    <p>Client IP Address: <%= request.getRemoteAddr() %></p>
    <p>HTTP Method: <%= request.getMethod() %></p>
```

```
    <h2>Session Information</h2>
    <p>Session ID: <%= session.getId() %></p>
    <p>Session Creation Time: <%= new
java.util.Date(session.getCreationTime()) %></p>

    <h2>Exception Handling</h2>
    <%-- Example of exception handling --%>
    <% try {
        int result = 10 / 0; // This will throw an ArithmeticException
        out.println("<p>Result: " + result + "</p>"); // This will not be
executed
    } catch (ArithmeticException e) {
        out.println("<p>Error: " + e.getMessage() + "</p>");
    } %>
</body>
</html>
```

36. Write a JSP/Servlet program to generate the exponential of a user entered value.

1. **ExponentialCalculator.jsp** (JSP Page):

```
jsp
Copy code
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Exponential Calculator</title>
</head>
<body>
    <h2>Exponential Calculator</h2>
    <form action="ExponentialCalculatorServlet" method="post">
        Enter a number:
        <input type="number" name="base" required><br>
        Enter the exponent:
        <input type="number" name="exponent" required><br>
        <input type="submit" value="Calculate">
    </form>
</body>
</html>
```

2. **ExponentialCalculatorServlet.java** (Servlet):

```
java
Copy code
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ExponentialCalculatorServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // Get the values entered by the user
        double base = Double.parseDouble(request.getParameter("base"));
        double exponent =
Double.parseDouble(request.getParameter("exponent"));
```

```
        // Calculate the exponential
        double result = Math.pow(base, exponent);

        // Set the result as an attribute to be displayed in the JSP
        request.setAttribute("result", result);

        // Forward the request to the result JSP
        request.getRequestDispatcher("ExponentialResult.jsp").forward(request,
response);
    }
}
```

3. **ExponentialResult.jsp** (JSP Page to Display Result):

```
jsp
Copy code
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Exponential Result</title>
</head>
<body>
    <h2>Exponential Result</h2>
    <p>Result: <%= request.getAttribute("result") %></p>
    <a href="ExponentialCalculator.jsp">Back to Calculator</a>
</body>
</html>
```

mte answers

37. Write a Java Server Pages (JSP) program to maintain session using session object.

Java Server Pages (JSP) program that maintains session using the session object:

```
jsp
Copy code
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Session Example</title>
</head>
<body>
    <h2>Session Example</h2>

    <%-- Check if a session exists --%>
    <% if (session.isNew()) { %>
        <p>A new session is created.</p>
    <% } else { %>
        <p>Session ID: <%= session.getId() %></p>
        <p>Session Creation Time: <%= new
java.util.Date(session.getCreationTime()) %></p>
        <p>Last Accessed Time: <%= new
java.util.Date(session.getLastAccessedTime()) %></p>
    <% } %>

    <%-- Set session attributes --%>
    <% session.setAttribute("username", "JohnDoe"); %>
    <% session.setAttribute("isLoggedIn", true); %>

    <p>Session attributes set.</p>

    <%-- Retrieve and display session attributes --%>
    <p>Username: <%= session.getAttribute("username") %></p>
    <p>Is Logged In: <%= session.getAttribute("isLoggedIn") %></p>

    <%-- Invalidate the session --%>
    <% session.invalidate(); %>
    <p>Session invalidated.</p>
</body>
</html>
```

38. Write a JSP/Servlet program using preparedStatement in entering data in MySQL/ Oracle table. Also display the result

1. **InsertData.jsp** (JSP Page):

```
jsp
Copy code
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Insert Data</title>
</head>
```

```
<body>
    <h2>Insert Data</h2>
    <form action="InsertDataServlet" method="post">
        Name: <input type="text" name="name" required><br>
        Age: <input type="number" name="age" required><br>
        <input type="submit" value="Insert">
    </form>
</body>
</html>
```

2. **InsertDataServlet.java** (Servlet):

```
java
Copy code
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/InsertDataServlet")
public class InsertDataServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        String name = request.getParameter("name");
        int age = Integer.parseInt(request.getParameter("age"));

        try {
            Class.forName("com.mysql.jdbc.Driver"); // For MySQL
            // Class.forName("oracle.jdbc.driver.OracleDriver"); // For Oracle
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/your_database",
"username", "password");
            // Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:your_sid",
"username", "password"); // For Oracle

            String sql = "INSERT INTO your_table (name, age) VALUES (?, ?)";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, name);
            pstmt.setInt(2, age);
            int rowsAffected = pstmt.executeUpdate();

            out.println("<html><body>");
            out.println("<h2>Data Inserted Successfully</h2>");
            out.println("<p>Rows Affected: " + rowsAffected + "</p>");
            out.println("</body></html>");

            pstmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException e) {
            out.println("Error: " + e.getMessage());
        }
    }
}
```

mte answers

39. Assume four users user1, user2, user3 and user4 having the passwords pwd1, pwd2, pwd3 and pwd4 respectively. Write a servlet/ JSP for doing the following. Create a Cookie and add these four user id's and passwords to this Cookie. 2. Read the user id and passwords entered in the Login form and authenticate with the values available in the cookies.

Servlet that handles creating a cookie with user IDs and passwords, and then reads and authenticates the user credentials entered in a login form with the values stored in the cookie:

```
java
Copy code
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        // User credentials
        String[] users = {"user1", "user2", "user3", "user4"};
        String[] passwords = {"pwd1", "pwd2", "pwd3", "pwd4"};

        // Create a cookie and add user IDs and passwords
        Cookie loginCookie = new Cookie("userCredentials", "");
        loginCookie.setMaxAge(30 * 60); // Cookie expires in 30 minutes

        StringBuilder cookieValue = new StringBuilder();
        for (int i = 0; i < users.length; i++) {
            cookieValue.append(users[i]).append(":").append(passwords[i]);
            if (i < users.length - 1) {
                cookieValue.append(",");
            }
        }
        loginCookie.setValue(cookieValue.toString());
        response.addCookie(loginCookie);

        // Read user credentials from the login form
        String userId = request.getParameter("userId");
        String password = request.getParameter("password");

        // Authenticate user credentials with values in the cookie
        Cookie[] cookies = request.getCookies();
        boolean isAuthenticated = false;
        if (cookies != null) {
            for (Cookie cookie : cookies) {
                if (cookie.getName().equals("userCredentials")) {
                    String[] credentials = cookie.getValue().split(",");
                    for (String credential : credentials) {
                        String[] parts = credential.split(":");
```

```
                            if (parts.length == 2 && parts[0].equals(userId) &&
    parts[1].equals(password)) {
                                isAuthenticated = true;
                                break;
                            }
                        }
                        break;
                    }
                }
            }

            // Display authentication result
            if (isAuthenticated) {
                out.println("<html><body><h2>Login
    Successful</h2></body></html>");
            } else {
                out.println("<html><body><h2>Login Failed</h2></body></html>");
            }
        }
    }
```

40. Design a table with the following fields: `name`, `password`, `email-id`, and `phone number`. Write a program using Servlets or JavaServer Pages (JSP) to connect to the database, retrieve data from the table, and display it. Additionally, implement functionality to insert user details into the table when a new user registers by clicking the submit button on the registration page.

1. **Registration.jsp** (JSP Page):

```
jsp
Copy code
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>User Registration</title>
</head>
<body>
    <h2>User Registration</h2>
    <form action="RegistrationServlet" method="post">
        Name: <input type="text" name="name" required><br>
        Password: <input type="password" name="password" required><br>
        Email: <input type="email" name="email" required><br>
        Phone Number: <input type="tel" name="phone" required><br>
        <input type="submit" value="Register">
    </form>
</body>
</html>
```

2. **RegistrationServlet.java** (Servlet):

```
java
Copy code
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

mte answers

```
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/RegistrationServlet")
public class RegistrationServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        String name = request.getParameter("name");
        String password = request.getParameter("password");
        String email = request.getParameter("email");
        String phone = request.getParameter("phone");

        try {
            Class.forName("com.mysql.jdbc.Driver"); // For MySQL
            // Class.forName("oracle.jdbc.driver.OracleDriver"); // For Oracle
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/your_database",
"username", "password");
            // Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:your_sid",
"username", "password"); // For Oracle

            // Insert user details into the table
            String insertSql = "INSERT INTO user_details (name, password,
email, phone) VALUES (?, ?, ?, ?)";
            PreparedStatement insertStmt = conn.prepareStatement(insertSql);
            insertStmt.setString(1, name);
            insertStmt.setString(2, password);
            insertStmt.setString(3, email);
            insertStmt.setString(4, phone);
            int rowsAffected = insertStmt.executeUpdate();

            if (rowsAffected > 0) {
                out.println("<html><body><h2>Registration
Successful</h2></body></html>");
            } else {
                out.println("<html><body><h2>Registration
Failed</h2></body></html>");
            }

            insertStmt.close();

            // Retrieve data from the table
            String selectSql = "SELECT * FROM user_details";
            PreparedStatement selectStmt = conn.prepareStatement(selectSql);
            ResultSet resultSet = selectStmt.executeQuery();

            out.println("<h2>User Details</h2>");
            out.println("<table border='1'>");

out.println("<tr><th>Name</th><th>Password</th><th>Email</th><th>Phone
Number</th></tr>");
            while (resultSet.next()) {
                out.println("<tr><td>" + resultSet.getString("name") +
"</td><td>" + resultSet.getString("password") +
```

```
                        "</td><td>" + resultSet.getString("email") +
"</td><td>" + resultSet.getString("phone") + "</td></tr>");
            }
            out.println("</table>");

            resultSet.close();
            selectStmt.close();

            conn.close();
        } catch (ClassNotFoundException | SQLException e) {
            out.println("Error: " + e.getMessage());
        }
    }
}
```

41. Create a webpage that includes various form elements such as Text area, select, text box, checkboxes, and radio buttons. Implement functionality using JavaScript to display the data entered by the user.

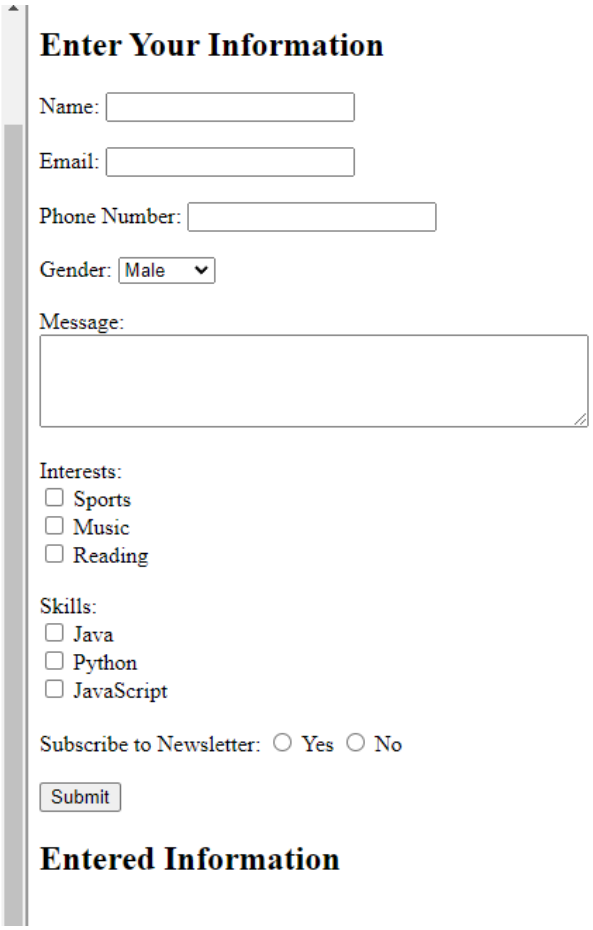
a webpage with various form elements and JavaScript functionality to display the data entered by the user:

```
html
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>Form Elements</title>
</head>
<body>
    <h2>Enter Your Information</h2>
    <form id="userForm">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name"><br><br>

        <label for="email">Email:</label>
        <input type="email" id="email" name="email"><br><br>

        <label for="phone">Phone Number:</label>
        <input type="text" id="phone" name="phone"><br><br>

        <label for="gender">Gender:</label>
        <select id="gender" name="gender">
            <option value="male">Male</option>
            <option value="female">Female</option>
        </select><br><br>

        <label for="message">Message:</label><br>
        <textarea id="message" name="message" rows="4"
cols="50"></textarea><br><br>

        <label for="interests">Interests:</label><br>
        <input type="checkbox" id="sports" name="interests" value="sports">
Sports<br>
        <input type="checkbox" id="music" name="interests" value="music">
Music<br>
        <input type="checkbox" id="reading" name="interests"
value="reading"> Reading<br><br>

        <label>Skills:</label><br>
        <input type="checkbox" id="java" name="skills" value="java">
Java<br>
```

mte answers

```
        <input type="checkbox" id="python" name="skills" value="python">
Python<br>
        <input type="checkbox" id="javascript" name="skills"
value="javascript"> JavaScript<br><br>

        <label for="newsletter">Subscribe to Newsletter:</label>
        <input type="radio" id="yes" name="newsletter" value="yes"> Yes
        <input type="radio" id="no" name="newsletter" value="no">
No<br><br>

        <input type="button" value="Submit" onclick="displayFormData()">
    </form>

    <h2>Entered Information</h2>
    <div id="userData"></div>

    <script>
        function displayFormData() {
            var formData = document.getElementById("userForm");
            var displayDiv = document.getElementById("userData");
            var displayText = "<ul>";

            for (var i = 0; i < formData.elements.length; i++) {
                var element = formData.elements[i];

                // Check if element is a text-based input
                if (element.type !== "button" && element.type !==
"checkbox" && element.type !== "radio") {
                    displayText += "<li>" + element.name + ": " +
element.value + "</li>";
                }

                // Check if element is a checkbox or radio button
                if (element.type === "checkbox" || element.type ===
"radio") {
                    if (element.checked) {
                        displayText += "<li>" + element.name + ": " +
element.value + "</li>";
                    }
                }
            }
            displayText += "</ul>";
            displayDiv.innerHTML = displayText;
        }
    </script>
</body>
</html>
```

mte answers



42. Design a shopping cart for an apparel store with session tracking API.

1. **index.html** (HTML Page):

```
html
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>Apparel Store</title>
    <script src="script.js"></script>
</head>
<body>
    <h2>Welcome to Apparel Store</h2>
    <div id="products">
        <h3>Products:</h3>
        <ul>
            <li>Product 1 - $10 <button onclick="addToCart('Product 1',
10)">Add to Cart</button></li>
            <li>Product 2 - $20 <button onclick="addToCart('Product 2',
20)">Add to Cart</button></li>
            <li>Product 3 - $30 <button onclick="addToCart('Product 3',
30)">Add to Cart</button></li>
        </ul>
    </div>
    <div id="cart">
        <h3>Shopping Cart:</h3>
        <ul id="cartItems"></ul>
        <p>Total: $<span id="cartTotal">0</span></p>
        <button onclick="checkout()">Checkout</button>
    </div>
</body>
</html>
```

2. **script.js** (JavaScript):

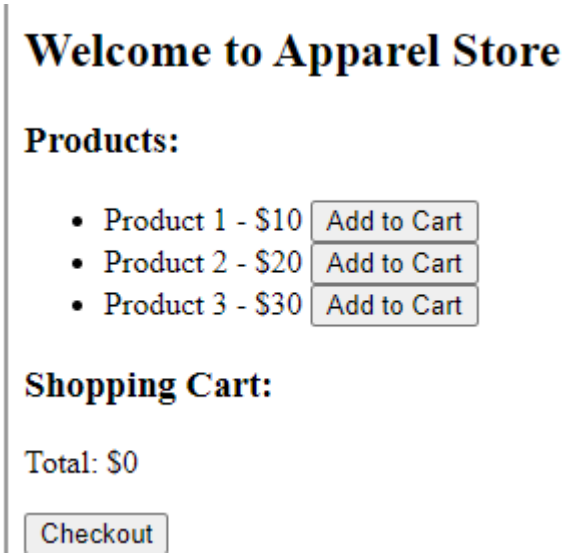
```
javascript
Copy code
function addToCart(itemName, itemPrice) {
    var cartItems = sessionStorage.getItem("cartItems");
    var cartTotal = sessionStorage.getItem("cartTotal");

    if (cartItems === null) {
        cartItems = [];
        cartTotal = 0;
    } else {
        cartItems = JSON.parse(cartItems);
        cartTotal = parseFloat(cartTotal);
    }

    cartItems.push({ name: itemName, price: itemPrice });
    cartTotal += itemPrice;

    sessionStorage.setItem("cartItems", JSON.stringify(cartItems));
    sessionStorage.setItem("cartTotal", cartTotal.toFixed(2));

    updateCartDisplay();
}

function updateCartDisplay() {
    var cartItems = sessionStorage.getItem("cartItems");
    var cartTotal = sessionStorage.getItem("cartTotal");

    if (cartItems !== null) {
        cartItems = JSON.parse(cartItems);

        var cartItemsList = document.getElementById("cartItems");
        cartItemsList.innerHTML = "";
        cartItems.forEach(function(item) {
            var li = document.createElement("li");
            li.textContent = item.name + " - $" + item.price;
            cartItemsList.appendChild(li);
        });

        document.getElementById("cartTotal").textContent = cartTotal;
    }
}

function checkout() {
    alert("Checkout completed! Total amount: $" +
sessionStorage.getItem("cartTotal"));
    sessionStorage.clear();
    updateCartDisplay();
}

window.onload = updateCartDisplay;
```

3. **CartServlet.java** (Servlet for Session Tracking):

```
java
Copy code
import java.io.IOException;
import java.util.ArrayList;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
```

mte answers

```
@WebServlet("/CartServlet")
public class CartServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        HttpSession session = request.getSession();
        ArrayList<String> cartItems = (ArrayList<String>)
session.getAttribute("cartItems");
        String cartTotal = (String) session.getAttribute("cartTotal");

        // Process cart items and total as needed
    }
}
```



43. Write a program of Servlet/ JSP in which pass three input using form element and find the greatest on in main.jsp page. If the greatest number is even then send control to Even.jsp otherwise odd.jsp

1. **index.html** (HTML Page with Form):

```
html
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>Find Greatest Number</title>
</head>
<body>
    <h2>Enter Three Numbers</h2>
    <form action="NumberServlet" method="post">
        Number 1: <input type="number" name="num1" required><br>
        Number 2: <input type="number" name="num2" required><br>
        Number 3: <input type="number" name="num3" required><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

2. **NumberServlet.java** (Servlet to Process Input):

```
java
Copy code
```

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/NumberServlet")
public class NumberServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        int num1 = Integer.parseInt(request.getParameter("num1"));
        int num2 = Integer.parseInt(request.getParameter("num2"));
        int num3 = Integer.parseInt(request.getParameter("num3"));

        int greatest = num1 > num2 ? (num1 > num3 ? num1 : num3) : (num2 >
num3 ? num2 : num3);

        if (greatest % 2 == 0) {
            response.sendRedirect("Even.jsp?greatest=" + greatest);
        } else {
            response.sendRedirect("Odd.jsp?greatest=" + greatest);
        }
    }
}
```

3. **Even.jsp** (JSP for Even Number):

```
jsp
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>Even Number Page</title>
</head>
<body>
    <h2>The Greatest Number is Even</h2>
    <p>The greatest even number entered is <%=
request.getParameter("greatest") %></p>
</body>
</html>
```

4. **Odd.jsp** (JSP for Odd Number):

```
jsp
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>Odd Number Page</title>
</head>
<body>
    <h2>The Greatest Number is Odd</h2>
    <p>The greatest odd number entered is <%= request.getParameter("greatest")
%></p>
</body>
</html>
```

44. Write a program of Servlet/ JSP in which pass three input using form element Name, Age and Marks. Use cookie / session objects to maintain session and send data to third page and print all.

mte answers

1. **index.html** (HTML Page with Form):

```
html
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>Enter Information</title>
</head>
<body>
    <h2>Enter Your Information</h2>
    <form action="InfoServlet" method="post">
        Name: <input type="text" name="name" required><br>
        Age: <input type="number" name="age" required><br>
        Marks: <input type="number" name="marks" required><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

2. **InfoServlet.java** (Servlet to Process Input and Set Session Attributes):

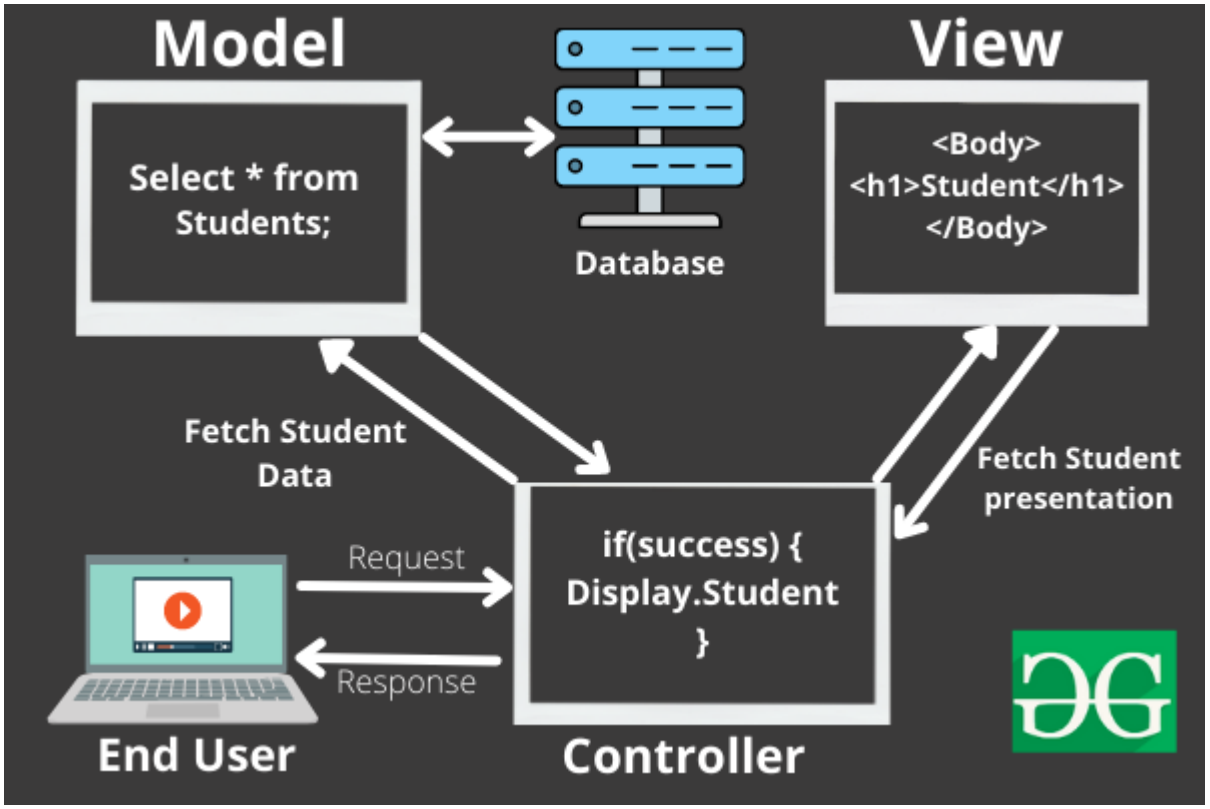
```
java
Copy code
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/InfoServlet")
public class InfoServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String name = request.getParameter("name");
        int age = Integer.parseInt(request.getParameter("age"));
        int marks = Integer.parseInt(request.getParameter("marks"));

        HttpSession session = request.getSession();
        session.setAttribute("name", name);
        session.setAttribute("age", age);
        session.setAttribute("marks", marks);

        response.sendRedirect("DisplayInfo.jsp");
    }
}
```

3. **DisplayInfo.jsp** (JSP to Display Information):

```
jsp
Copy code
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Display Information</title>
</head>
<body>
    <h2>Information Stored in Session</h2>
```

```
    <p>Name: <%= session.getAttribute("name") %></p>
    <p>Age: <%= session.getAttribute("age") %></p>
    <p>Marks: <%= session.getAttribute("marks") %></p>
</body>
</html>
```

45. Explain MVC architecture in web application. Explain with a suitable example.



MVC (Model-View-Controller) architecture is a design pattern commonly used in web applications to separate concerns and organize codebase for better maintainability and scalability. Here's a brief explanation of each component in MVC along with an example:

1. **Model**: This component represents the data and business logic of the application. It interacts with the database or any other data source to fetch or manipulate data.
2. **View**: This component represents the user interface. It is responsible for presenting data to the user and handling user interactions.
3. **Controller**: This component acts as an intermediary between the Model and the View. It receives user input from the View, processes it using the Model, and then updates the View with the results.

Example:

Let's consider a simple web application for managing a list of books. Here's how MVC can be applied:

- **Model** (Book.java):

```
java
Copy code
public class Book {
    private int id;
    private String title;
```

```
    private String author;
    private double price;

    // Getters and setters
}
```

- **View** (booklist.jsp):

```
jsp
Copy code
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Book List</title>
</head>
<body>
    <h2>Book List</h2>
    <table border="1">
        <tr>
            <th>ID</th>
            <th>Title</th>
            <th>Author</th>
            <th>Price</th>
        </tr>
        <% for (Book book : books) { %>
            <tr>
                <td><%= book.getId() %></td>
                <td><%= book.getTitle() %></td>
                <td><%= book.getAuthor() %></td>
                <td><%= book.getPrice() %></td>
            </tr>
        <% } %>
    </table>
</body>
</html>
```

- **Controller** (BookController.java):

```
java
Copy code
import java.util.List;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class BookController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) {
        // In a real application, this would fetch data from a database
        List<Book> books = BookDAO.getAllBooks();

        // Set books as an attribute in the request
        request.setAttribute("books", books);

        // Forward the request to the view (booklist.jsp)
        request.getRequestDispatcher("booklist.jsp").forward(request,
response);
    }
}
```

Difference between ServletConfig vs. ServletContext

comparison between ServletConfig and ServletContext:

Feature	ServletConfig	ServletContext
Purpose	Pass initialization parameters to a specific servlet	Provide application-wide information and functionality
Scope	Specific to a particular servlet	Global to the entire web application
Access	Accessed using getServletConfig() method	Accessed using getServletContext() method
Initialization	Parameters set in deployment descriptor (web.xml) or using annotations	Parameters set in deployment descriptor (web.xml) or using context parameters in a servlet container
Example	`java public void init(ServletConfig config) { String databaseUrl = config.getInitParameter("databaseUrl"); // Use databaseUrl parameter for initialization }`	`java public void init(ServletConfig config) { ServletContext context = config.getServletContext(); String appName = context.getInitParameter("appName"); // Use appName parameter for application-wide configuration }`

mte answers