Module 1: [4L] Introduction to Web:

1. What do you mean by Static and Dynamic Web Pages?
2. What are the advantages of dynamic pages over static web pages?
3. Write down the advantages of 3 tier web architecture over 2 tier web architecture.
4. Write short notes on any one:

a) J2EE architecture

b) WWW

c) W3C

d) Internet vs Intranet vs extranet

e) Dynamic IP

f) HTTP Protocol

**1. Static and Dynamic Web Pages:**

**Static Web Pages:** Static web pages are fixed and display the same content to every user. The content is hardcoded in HTML and does not change unless the webmaster manually edits the page. They are faster to load and simpler to create but lack interactive features.

**Dynamic Web Pages:** Dynamic web pages generate content on-the-fly based on user interactions, inputs, or other variables. Content can be pulled from databases, allowing for personalized and interactive user experiences. Dynamic pages are typically created using server-side scripting languages like PHP, Python, or Ruby.

**2. Advantages of Dynamic Pages over Static Web Pages:**

* **Interactivity:** Dynamic pages allow user interaction, such as submitting forms, updating content without refreshing the entire page, and providing personalized experiences.
* **Real-time Updates:** Content on dynamic pages can change dynamically, reflecting real-time data from databases or other sources.
* **Database Integration:** Dynamic pages can connect to databases, enabling the storage and retrieval of dynamic information, making them suitable for content-rich websites.
* **Customization:** Dynamic pages can adapt content based on user preferences, behavior, or profile information.

**3. Advantages of 3-Tier Web Architecture over 2-Tier Web Architecture:**

**3-Tier Web Architecture:**

* **Scalability:** The architecture separates the presentation layer, business logic layer, and data storage layer, making it easier to scale and maintain each component independently.
* **Modularity:** It promotes a modular design where changes in one layer don't necessarily affect the others, facilitating easier maintenance and updates.
* **Security:** By separating the layers, sensitive data and business logic can be better protected, enhancing overall system security.

**2-Tier Web Architecture:**

* **Simplicity:** 2-tier architectures are simpler to design and implement, suitable for small-scale applications with limited complexity.
* **Performance:** In some cases, a 2-tier architecture might provide better performance for simple applications as there are fewer layers and processing steps.

**4. Short Notes:**

a) J2EE Architecture:

Java 2 Platform, Enterprise Edition (J2EE) is a set of specifications, APIs, and technologies for developing multi-tiered, scalable, and secure enterprise applications. It provides a framework for developing and deploying distributed Java-based applications.

b) WWW (World Wide Web):

The World Wide Web (WWW) is an information space on the internet where documents and resources are identified by Uniform Resource Locators (URLs) and can be accessed via the internet. It is the collection of web pages and related content.

c) W3C (World Wide Web Consortium):

W3C is an international community that develops open standards to ensure the long-term growth of the Web. It is responsible for specifications like HTML, XML, and other web-related technologies, aiming to make the web accessible and interoperable.

d) Internet vs Intranet vs Extranet:

* **Internet:** A global network of interconnected computers accessible to the public.
* **Intranet:** A private network within an organization, using internet technologies to share information internally.
* **Extranet:** An extension of an intranet that allows limited access to external users, such as business partners or clients.

e) Dynamic IP:

Dynamic IP addresses are assigned by Internet Service Providers (ISPs) and can change each time a device connects to the network. This contrasts with static IP addresses, which remain constant.

f) HTTP Protocol:

Hypertext Transfer Protocol (HTTP) is the foundation of data communication on the World Wide Web. It defines how messages are formatted and transmitted, facilitating the transfer of hypertext between web browsers and servers.

Module -2: [9L] HTML, DHTML & CSS & Module -3: Extended Markup Language (XML)

1. Write a single html program through which you can explain

a) anchor tag,

b)’img’ tag with ‘src’ attribute,

c) paragraph

d) heading

e) href

f) form

g) frame and frameset

h) table,th,tr,td

2. Write a single html program which implements image map concept using ‘usemap’ and

3. What is CSS? Why it is important in web design? Provide coding example for external, embedded, imported and inline style of CSS? What is the difference between class selector and id selector in CSS?

4. What are the needs of XML? What is XML parser? What is XML Shema?

Write short notes on DTD. What is DOM? What are the needs for DTD?

5. What is web service? Explain service oriented architecture with proper diagram.

6. What is AJAX? How does it work

7. Differentiate between:

a) XML and HTML

b) SAX and DOM Model

c) HTML vs DHTML

### 1. HTML Program:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>HTML Basics</title>

</head>

<body>

<!-- Anchor Tag -->

<a href="https://www.example.com">Visit Example Website</a>

<!-- Image Tag -->

<img src="image.jpg" alt="Description of the image">

<!-- Paragraph -->

<p>This is a sample paragraph.</p>

<!-- Heading -->

<h1>Main Heading</h1>

<!-- Href Attribute -->

<a href="#section">Link to Section</a>

<!-- Form -->

<form action="/submit" method="post">

<label for="username">Username:</label>

<input type="text" id="username" name="username">

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

</form>

<!-- Frame and Frameset -->

<frameset cols="25%, 75%">

<frame src="menu.html" name="menu">

<frame src="content.html" name="content">

</frameset>

<!-- Table -->

<table border="1">

<tr>

<th>Header 1</th>

<th>Header 2</th>

</tr>

<tr>

<td>Row 1, Cell 1</td>

<td>Row 1, Cell 2</td>

</tr>

<tr>

<td>Row 2, Cell 1</td>

<td>Row 2, Cell 2</td>

</tr>

</table>

</body>

</html>

### 2. HTML Program with Image Map:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Image Map Example</title>

</head>

<body>

<img src="worldmap.jpg" alt="World Map" usemap="#worldmap">

<map name="worldmap">

<area shape="rect" coords="0,0,100,100" href="northamerica.html" alt="North America">

<area shape="rect" coords="100,0,200,100" href="europe.html" alt="Europe">

<!-- Add more areas for other continents as needed -->

</map>

</body>

</html>

**3. CSS (Cascading Style Sheets):**

**What is CSS?** CSS stands for Cascading Style Sheets. It is a style sheet language used for describing the look and formatting of a document written in HTML or XML.

**Importance in Web Design:** CSS is crucial in web design as it separates the structure and content from the presentation. It allows designers to control the layout, colors, fonts, and other stylistic aspects of a web page. This separation enhances consistency, maintainability, and flexibility in web development.

**CSS Coding Examples:**

* **External Style:**

<link rel="stylesheet" type="text/css" href="styles.css">

**Embedded Style:**

<style>

body {

font-family: Arial, sans-serif;

background-color: #f0f0f0;

}

</style>

**Imported Style:**

<style>

@import url("styles.css");

</style>

* **Inline Style:**

<p style="color: blue; font-size: 16px;">This is a styled paragraph.</p>

**Difference between Class Selector and ID Selector:**

**Class Selector (.)**: Used to select elements with a specific class. Can be applied to multiple elements. .highlight {

background-color: yellow;

}

**ID Selector (#)**: Used to select a single element with a specific ID. Should be unique on a page.

#main-heading {

font-size: 24px;

}

**4. XML (eXtensible Markup Language):**

**Needs of XML:**

* **Data Exchange:** Facilitates the exchange of data between different platforms and applications.
* **Platform-Independence:** XML is platform-independent and can be used across various operating systems and programming languages.

**XML Parser:** An XML parser is a software component that reads XML documents and provides an interface for programmatic access. It parses the XML structure and allows applications to interact with the data.

**XML Schema:** XML Schema is a specification that defines the structure of an XML document. It serves as a blueprint for the structure and data types that should be present in the XML document.

**Short Notes on DTD (Document Type Definition):** DTD is a formal definition of the structure and the legal elements and attributes of an XML document. It defines the document's structure, specifying the elements and their relationships. However, DTD lacks some features compared to XML Schema.

**DOM (Document Object Model):** DOM is a programming interface for web documents. It represents the document as a tree of objects, allowing programs to manipulate the structure, style, and content of web documents dynamically.

**Needs for DTD:**

* **Validation:** DTD helps validate the structure of an XML document, ensuring it adheres to predefined rules.
* **Interoperability:** Provides a common structure that allows different systems to understand and exchange data.

**5. Web Service and Service-Oriented Architecture (SOA):**

**Web Service:** A web service is a software system designed to support interoperable machine-to-machine communication over a network. It provides a standardized way for different software applications to communicate with each other.

**Service-Oriented Architecture (SOA):** SOA is an architectural style that organizes software as a set of services. These services are loosely coupled, and they communicate with each other to achieve a specific business goal. SOA promotes reusability and flexibility in software design.

**Diagram:**

+------------------+ +------------------+ +------------------+

| Service 1 | | Service 2 | | Service 3 |

| (Functionality)| <--> | (Functionality)| <--> | (Functionality)|

+------------------+ +------------------+ +------------------+

**6. AJAX (Asynchronous JavaScript and XML):**

**What is AJAX?** AJAX is a technique used in web development to create asynchronous web applications. It allows updating parts of a web page without requiring a full page reload. AJAX uses a combination of HTML, CSS, JavaScript, and XML or JSON for data exchange.

**How does it work?** AJAX works by sending asynchronous requests to the server using JavaScript. The server processes the request and sends back data (usually in XML or JSON format), which is then dynamically updated on the web page.

**7. Differentiation:**

a) XML vs HTML:

* **XML (eXtensible Markup Language):**
  + Purpose: Designed to transport and store data.
  + Tags: User-defined tags.
  + Structure: Hierarchical and flexible.
  + Use: Suitable for data interchange.
* **HTML (Hypertext Markup Language):**
  + Purpose: Used to display data and create a structured document.
  + Tags: Predefined tags for document structure.
  + Structure: Fixed, follows a tree-like structure.
  + Use: Designed for creating web pages.

b) SAX vs DOM Model:

* **SAX (Simple API for XML):**
  + Parsing: Event-driven parsing, reads XML sequentially.
  + Memory Usage: Uses less memory.
  + Use: Suitable for large XML documents.
* **DOM (Document Object Model):**
  + Parsing: Tree-based parsing, loads the entire XML document into memory.
  + Memory Usage: Uses more memory.
  + Use: Suitable for small to medium-sized XML documents.

c) HTML vs DHTML:

* **HTML (Hypertext Markup Language):**
  + Nature: Static, defines the structure of a webpage.
  + Interaction: Limited interactivity.
* **DHTML (Dynamic HTML):**
  + Nature: Dynamic, uses HTML, CSS, and JavaScript to create interactive web pages.
  + Interaction: Allows dynamic content changes and user interactivity.

Module 3: JavaScript,CGI,PHP

1.Write a JavaScript to validate the mail id for @,.com etc?

2.What is CGI? Explain CGI architecture with suitable

3. Differentiate between:

a) GET ( ) and POST( ) method

b) CGI and Servlet

c) EJB and JavaBeans

4.How can PHP interact with HTML?

5.What did the acronym PHP originally stand for?

6.What are some of the common applications of PHP?

7.Name the popular Content Management Systems (CMS) in PHP.

8.What are the characteristics of PHP variables?

9.How do you define a constant in PHP?

10. What is the main difference between asp net and PHP?

11. Write a html program to find out m to the power n (m, n valid integer no) using a function using JavaScript.

12. Write a simple PHP program through which you can implement the concept of GET & POST method w.r.t PHP Form handling each field should be validated by JavaScript.

13. Write a simple PHP program through which you can find out maximum and minimum among three no’s specified by the user.

**1. JavaScript Email Validation:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Email Validation</title>

<script>

function validateEmail() {

var email = document.getElementById("email").value;

var emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

if (emailPattern.test(email)) {

alert("Email is valid!");

} else {

alert("Invalid email address!");

}

}

</script>

</head>

<body>

<label for="email">Email:</label>

<input type="text" id="email">

<button onclick="validateEmail()">Validate Email</button>

</body>

</html>

**2. CGI (Common Gateway Interface):**

**What is CGI?** CGI is a standard protocol for web servers to execute programs that execute like console applications running on a server to generate dynamic web content. It stands for Common Gateway Interface.

**CGI Architecture:**

1. **Client Request:** The client (browser) sends a request to the webserver for a CGI script.
2. **Server Launches CGI Script:** The web server launches the CGI script as a separate process.
3. **Execution of CGI Script:** The CGI script processes the request, performs necessary actions, and generates output.
4. **Server Sends Response:** The server sends the output generated by the CGI script back to the client.

**3. Differentiation:**

a) GET() and POST() Method:

* **GET():**
  + Data is sent in the URL.
  + Limited data transfer (about 2048 characters).
  + Less secure as data is visible in the URL.
  + Suitable for non-sensitive data.
* **POST():**
  + Data is sent in the request body.
  + Can handle large data sets.
  + More secure as data is not visible in the URL.
  + Suitable for sensitive data.

b) CGI and Servlet:

* **CGI (Common Gateway Interface):**
  + Older technology.
  + Each request creates a new process, which can be resource-intensive.
  + Language-independent (can be written in Perl, Python, etc.).
* **Servlet:**
  + Java-based technology.
  + Servlets run in the same address space as the web server, more efficient.
  + Requires a servlet container, like Apache Tomcat.

c) EJB and JavaBeans:

* **EJB (Enterprise JavaBeans):**
  + Used for building distributed, scalable, and transactional enterprise applications.
  + Typically used in large-scale, enterprise-level projects.
* **JavaBeans:**
  + Used for creating reusable components in Java.
  + Generally used in smaller, standalone applications.

**4. PHP and HTML Interaction:**

PHP can interact with HTML in several ways:

* **Embedding PHP in HTML:**

<html>

<body>

<?php

// PHP code here

?>

<p>This is an HTML paragraph.</p>

</body>

</html>

**Using HTML Forms:**

<form action="process.php" method="post">

<input type="text" name="username" />

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

</form>

**5. PHP Acronym:**

Originally, PHP stood for "Personal Home Page." It has since been redefined to "PHP: Hypertext Preprocessor."

**6. Common Applications of PHP:**

* Web development for creating dynamic websites.
* Server-side scripting for handling form data and user authentication.
* Command-line scripting for automation tasks.
* Developing e-commerce applications.
* Content Management Systems (CMS) development.

**7. Popular CMS in PHP:**

* WordPress
* Joomla
* Drupal

**8. Characteristics of PHP Variables:**

* **Dynamic Typing:** Variable types are determined at runtime.
* **Variable Scope:** Variables can be local or global.
* **Variable Naming:** Case-sensitive, starts with a dollar sign ($).

**9. Defining a Constant in PHP:**

define("PI", 3.14);

echo PI; // Outputs 3.14

**10. Difference between ASP.NET and PHP:**

* **ASP.NET:**
  + Developed by Microsoft.
  + Primarily used with C# or VB.NET.
  + Runs on Windows servers.
  + Integrated with the .NET framework.
* **PHP:**
  + Open-source scripting language.
  + Platform-independent.
  + Runs on various web servers.
  + Works well with databases like MySQL.

**11. HTML Program for m to the power n using JavaScript:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Power Calculation</title>

<script>

function calculatePower() {

var base = parseInt(document.getElementById("base").value);

var exponent = parseInt(document.getElementById("exponent").value);

var result = Math.pow(base, exponent);

alert(base + " to the power " + exponent + " is: " + result);

}

</script>

</head>

<body>

<label for="base">Base:</label>

<input type="text" id="base">

<label for="exponent">Exponent:</label>

<input type="text" id="exponent">

<button onclick="calculatePower()">Calculate Power</button>

</body>

</html>

### 12. PHP Program for Form Handling:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>PHP Form Handling</title>

</head>

<body>

<?php

// Check if form is submitted

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Validate and process form data

$username = test\_input($\_POST["username"]);

$email = test\_input($\_POST["email"]);

// Further processing or database operations can be done here

echo "Form submitted successfully! Username: $username, Email: $email";

}

// Function to sanitize input data

function test\_input($data) {

$data = trim($data);

$data = stripslashes($data);

$data = htmlspecialchars($data);

return $data;

}

?>

<form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>">

<label for="username">Username:</label>

<input type="text" name="username" required>

<br>

<label for="email">Email:</label>

<input type="email" name="email" required>

<br>

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

</form>

</body>

</html>

### 13. PHP Program to Find Maximum and Minimum:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Max and Min Finder</title>

</head>

<body>

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$num1 = $\_POST["num1"];

$num2 = $\_POST["num2"];

$num3 = $\_POST["num3"];

$max = max($num1, $num2, $num3);

$min = min($num1, $num2, $num3);

echo "Maximum: $max, Minimum: $min";

}

?>

<form method="post" action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>">

<label for="num1">Number 1:</label>

<input type="number" name="num1" required>

<br>

<label for="num2">Number 2:</label>

<input type="number" name="num2" required>

<br>

<label for="num3">Number 3:</label>

<input type="number" name="num3" required>

<br>

<input type="submit" value="Find Max and Min">

</form>

</body>

</html>

Module-4: JSP, Servlet, JDBC, .NET

JDBC

1. What do you mean by JDBC?
2. Why it is called ODBC (open database connectivity)?
3. Explain different types of JDBC drivers?
4. Write down different steps to create DSN in JDBC?
5. Differentiate between: a) PreparedStatement and prepareStatement()

**a) JDBC (Java Database Connectivity):**

**Meaning:** JDBC (Java Database Connectivity) is a Java API that allows Java programs to interact with relational databases. It provides a standard interface for connecting Java applications with databases and executing SQL queries.

**b) ODBC (Open Database Connectivity):**

**Why it is called ODBC:** ODBC (Open Database Connectivity) is called "open" because it is an open standard that allows applications to connect to any database that provides an ODBC driver. The term "connectivity" emphasizes the goal of providing a common interface for accessing various database management systems (DBMS).

**c) Different Types of JDBC Drivers:**

1. **Type 1 Driver (JDBC-ODBC Bridge Driver):**
   * Converts JDBC calls into ODBC calls.
   * Requires ODBC driver to connect to the database.
   * Limited portability, often used for migration purposes.
2. **Type 2 Driver (Native API Driver):**
   * Uses database-specific native API to interact with the database.
   * Requires database-specific native code, making it less portable than Type 4.
3. **Type 3 Driver (Network Protocol Driver):**
   * Communicates with a middle-tier server that translates JDBC calls to the database-specific protocol.
   * Platform-independent, but not as efficient as Type 4.
4. **Type 4 Driver (Thin Driver):**
   * Pure Java driver, communicates directly with the database using a protocol provided by the database.
   * Platform-independent and highly portable.
   * Most commonly used in modern JDBC applications.

**d) Steps to Create DSN in JDBC:**

Creating a Data Source Name (DSN) in JDBC involves configuring the connection between Java and the database using the ODBC driver. Here are the steps:

1. **Open ODBC Data Source Administrator:**
   * On Windows, go to Control Panel > Administrative Tools > Data Sources (ODBC).
   * Choose the appropriate ODBC driver.
2. **Add a New DSN:**
   * Click on the "System DSN" tab.
   * Click the "Add" button to add a new data source.
   * Choose the appropriate ODBC driver for your database.
3. **Configure the DSN:**
   * Enter a name for the DSN.
   * Provide necessary connection details, such as database server, port, database name, etc.
   * Test the connection to ensure it is successful.
4. **Save and Exit:**
   * Save the configuration and exit the ODBC Data Source Administrator.

**e) Differentiate between PreparedStatement and prepareStatement():**

**PreparedStatement:**

* **Definition:** **PreparedStatement** is an interface in JDBC that extends **Statement**. It is used to execute parameterized SQL queries.
* **Parameter Binding:** Allows parameter binding, which helps prevent SQL injection attacks.
* **Performance:** More efficient for repeated execution of the same SQL statement with different parameter values.
* **Compilation:** The SQL query is precompiled, and the execution plan is cached for reuse.

**prepareStatement():**

* **Definition:** **prepareStatement()** is a method of the **Connection** interface in JDBC. It is used to create a **PreparedStatement** object.
* **Usage:** The method is called on a **Connection** object to create a **PreparedStatement** for a parameterized SQL query.
* **Dynamic SQL:** Useful for executing dynamic SQL queries with varying parameter values.
* **Compilation:** The SQL query is compiled each time it is executed.

**Summary:**

* **PreparedStatement** is an interface, while **prepareStatement()** is a method.
* **prepareStatement()** is used to create a **PreparedStatement** object.
* **PreparedStatement** provides better performance for repeated executions with different parameter values.
* Both are used for parameterized SQL queries but serve different purposes.

Servlet

1. Write down the Servlet Life cycle?

2. What is the function of init(), service() and destroy() method in the life cycle of Servlet?

3. Differentiate between:

a) Session and Cookies

b) GET( ) and POST( ) method

c) CGI and Servlet

d) Servlet and Applet

**1. Servlet Life Cycle:**

The life cycle of a servlet consists of several phases, and the **javax.servlet.Servlet** interface defines the methods that represent these phases:

1. **Initialization (init):**
   * Invoked when the servlet is first created.
   * Performed only once during the servlet's lifecycle.
   * Used for one-time initialization tasks.
2. **Request Handling (service):**
   * Invoked for each client request.
   * Handles the client's request and generates the response.
   * The **service** method is called multiple times during the servlet's lifespan.
3. **Destruction (destroy):**
   * Invoked when the servlet is being taken out of service.
   * Performed only once during the servlet's lifecycle.
   * Used for cleanup tasks (closing database connections, releasing resources, etc.).
4. **Optional Methods:**
   * **init(ServletConfig config):** An overloaded version of **init** that takes a **ServletConfig** object as a parameter.
   * **service(ServletRequest request, ServletResponse response):** An overloaded version of **service** that takes **ServletRequest** and **ServletResponse** as parameters.
   * **destroy():** An overloaded version of **destroy** with no parameters.

**2. Function of init(), service(), and destroy() Method:**

* **init(ServletConfig config):**
  + **Function:** Initializes the servlet.
  + **Usage:** Typically used for one-time initialization tasks such as loading configuration parameters or connecting to a database.
* **service(ServletRequest request, ServletResponse response):**
  + **Function:** Handles client requests.
  + **Usage:** The core of the servlet's functionality where request processing and response generation occur. This method is called for each request.
* **destroy():**
  + **Function:** Cleans up resources and prepares the servlet for removal.
  + **Usage:** Used for releasing resources like closing database connections or releasing memory. Executed once during the servlet's shutdown.

**3. Differentiation:**

a) Session and Cookies:

* **Session:**
  + **Purpose:** Used to maintain state information between multiple requests from the same user.
  + **Storage:** Typically stored on the server-side.
  + **Duration:** Persists for a longer duration (configurable).
  + **Security:** Generally considered more secure than cookies.
* **Cookies:**
  + **Purpose:** Used to store small pieces of data on the client-side.
  + **Storage:** Stored on the client-side.
  + **Duration:** Can be persistent (stored after browser closure) or session-based (deleted after browser closure).
  + **Security:** May have security concerns, as data is stored on the client.

b) GET() and POST() Method:

* **GET():**
  + **Data Transmission:** Sends data as part of the URL.
  + **Security:** Less secure as data is visible in the URL.
  + **Usage:** Suitable for non-sensitive data and when idempotent operations are performed.
  + **Limitation:** Limited data transfer capability (about 2048 characters).
* **POST():**
  + **Data Transmission:** Sends data in the request body.
  + **Security:** More secure as data is not visible in the URL.
  + **Usage:** Suitable for sensitive data and when non-idempotent operations are performed.
  + **Advantage:** Can handle larger data sets than GET.

c) CGI and Servlet:

* **CGI (Common Gateway Interface):**
  + **Nature:** Stateless, each request creates a new process.
  + **Resource Usage:** May be resource-intensive as a new process is created for each request.
  + **Language:** Can be implemented in various languages (Perl, Python, etc.).
* **Servlet:**
  + **Nature:** Can be stateful, as the server may reuse the same servlet instance for multiple requests.
  + **Resource Usage:** More efficient resource usage compared to CGI.
  + **Language:** Primarily used with Java.

d) Servlet and Applet:

* **Servlet:**
  + **Execution:** Runs on the server-side.
  + **Usage:** Handles requests from clients, typically browsers.
  + **UI:** No direct UI components; interacts with HTML or other client-side technologies.
* **Applet:**
  + **Execution:** Runs on the client-side (browser).
  + **Usage:** Embedded in HTML and executed by a browser.
  + **UI:** Can have a graphical user interface (GUI) and interact directly with the user.

**Note:** Applets have become less common due to security concerns and the prevalence of other web technologies.

JSP

4. a. Explain JSP life cycle with proper diagram?

b. What do you mean JSP translation and JSP Compilation?

c. What are the different JSP Scripting tags? Explain with examples.

d. Differentiate between: JSP Declaration tag vs JSP Scriptlet tag

e. What are different types of JSP Action tag? Explain with example.

f. Differentiate between:

a) JSP VS ASP

b) JSP vs JavaScript

c) JSP VS PHP VS .NET

d) JSP VS JAVA APPLICATION

g. Write short notes on: JSP Implicit object and JSP Explicit object

a. **JSP Life Cycle:**

JavaServer Pages (JSP) have a life cycle that involves several phases:

1. **Translation:** In this phase, the JSP engine translates the JSP page into a servlet class. This servlet class is responsible for generating dynamic content during runtime.
2. **Compilation:** The translated servlet class is then compiled into bytecode by the Java compiler. This bytecode is then executed by the Java Virtual Machine (JVM).
3. **Initialization:** During this phase, the servlet engine initializes the servlet instance. The **init** method is called, and any necessary resources are set up.
4. **Execution:** This is the main phase where the **service** method is called for each request, and the servlet generates the dynamic content based on the client's request.
5. **Cleanup:** Finally, when the servlet is no longer needed or the server is shutting down, the **destroy** method is called to perform cleanup activities.

b. **JSP Translation and Compilation:**

* **JSP Translation:** This is the process of converting JSP code into a servlet. The translation phase generates a servlet source file from the JSP file.
* **JSP Compilation:** Once the translation is done, the generated servlet source code is compiled into bytecode by the Java compiler. This bytecode can be executed by the JVM.

c. **JSP Scripting Tags:**

JSP provides three main types of scripting tags:

* **Declaration Tag:**

<%! int count = 0; %>

**Scriptlet Tag:**

<% int result = 10 + 20; %>

**Expression Tag:**

<%= result %>

d. **Difference between JSP Declaration Tag and JSP Scriptlet Tag:**

* **Declaration Tag:** Used for declaring variables or methods. Declarations are outside the service method and are part of the generated servlet class.

<%! int count = 0; %>

* **Scriptlet Tag:** Used for embedding Java code within the **service** method. It is executed every time the JSP page is requested.

<% int result = 10 + 20; %>

e. **Types of JSP Action Tags:**

JSP Action tags include:

* **<jsp:include>:** Includes another resource, such as a file or servlet.

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

**<jsp:forward>:** Forwards the request to another resource, like another JSP or servlet.

<jsp:forward page="newPage.jsp" />

f. **Differences:**

* **JSP vs ASP:**
  + JSP is Java-based, while ASP (Active Server Pages) is Microsoft's server-side scripting technology.
  + JSP uses Java for server-side scripting, while ASP uses languages like VBScript or JScript.
* **JSP vs JavaScript:**
  + JSP is a server-side technology for dynamic content generation, while JavaScript is a client-side scripting language for enhancing interactivity on the user's browser.
* **JSP vs PHP vs .NET:**
  + JSP and PHP are server-side scripting technologies, while .NET is a framework that supports multiple languages, including C# and VB.NET.
* **JSP vs Java Application:**
  + JSP is specifically designed for web development and generating dynamic content, while a Java application is a standalone program that runs on the Java Virtual Machine (JVM).

g. **JSP Implicit and Explicit Objects:**

* **JSP Implicit Objects:**
  + Implicit objects are automatically available in JSP pages without declaration.
  + Examples include **request**, **response**, **session**, **application**, **out**, **config**, **pageContext**, etc.
* **JSP Explicit Objects:**
  + Explicit objects are created or referenced using JSP tags.
  + Examples include **page**, **pageContext**, **request**, **response**, **session**, and **application**. They provide more control and flexibility in managing data and interactions in JSP pages.

.NET

1. 5.What is LINQ?
2. How does the .NET framework work?
3. Explain about major components of the .NET framework.
4. What is CTS and CLS? What does CLR stand for in .NET?
5. What is JIT?
6. Explain Microsoft Intermediate Language?
7. 11.Differentiate: ASP vs ASP.NET
8. 12. What is .NET Middleware Engine?
9. 13. Write a short note on: MVC architecture

a. **LINQ (Language Integrated Query):**

* LINQ is a feature in .NET that enables developers to query data from various data sources using a uniform syntax. It integrates query capabilities directly into the C# and VB.NET languages.
* LINQ supports querying databases, XML, collections, and other data sources in a type-safe manner.

b. **How .NET Framework Works:**

* The .NET Framework is a platform developed by Microsoft for building, deploying, and running applications. It works through the following steps:
  1. **Compilation:** Source code (written in languages like C# or VB.NET) is compiled into Microsoft Intermediate Language (MSIL) or Common Intermediate Language (CIL).
  2. **Common Language Runtime (CLR):** MSIL code is executed by the CLR, which provides services like memory management, security, and exception handling.
  3. **Just-In-Time Compilation (JIT):** MSIL code is compiled into native machine code by the JIT compiler at runtime.
  4. **Execution:** The native code is executed by the computer's processor.

c. **Major Components of the .NET Framework:**

* **Common Language Runtime (CLR):** Manages memory, security, and exception handling.
* **Class Library:** A set of pre-built classes and functions that developers can use.
* **ASP.NET:** A framework for building web applications.
* **ADO.NET:** Provides data access capabilities.
* **Windows Forms:** Used for creating desktop applications with a graphical user interface (GUI).
* **Windows Presentation Foundation (WPF):** For building modern user interfaces.
* **Windows Communication Foundation (WCF):** Facilitates building distributed and interoperable applications.
* **ASP.NET MVC:** A framework for building web applications using the Model-View-Controller architectural pattern.

d. **CTS (Common Type System), CLS (Common Language Specification), and CLR (Common Language Runtime):**

* **CTS:** Defines the types that can be used and shared across .NET languages.
* **CLS:** A subset of CTS that all .NET languages are expected to support, ensuring language interoperability.
* **CLR:** Manages the execution of .NET programs. It includes the JIT compiler, garbage collector, and other services.

e. **JIT (Just-In-Time Compilation):**

* JIT is a part of the CLR that compiles MSIL code into native machine code at runtime, just before execution.
* This helps in achieving platform independence as the same MSIL code can be executed on any system with the appropriate CLR.

f. **Microsoft Intermediate Language (MSIL):**

* MSIL, also known as CIL (Common Intermediate Language), is an intermediate language used by .NET.
* It is a low-level, platform-agnostic representation of code generated during compilation.
* MSIL code is later compiled into native machine code by the JIT compiler.

g. **Differentiate: ASP vs ASP.NET:**

* **ASP (Active Server Pages):**
  + Technology for building dynamic web pages using server-side scripting.
  + Uses VBScript or JScript for server-side scripting.
  + Lacks built-in support for object-oriented programming.
* **ASP.NET:**
  + Framework for building web applications.
  + Supports multiple languages (C#, VB.NET, etc.).
  + Utilizes the .NET framework, providing better scalability, maintainability, and robustness.
  + Introduces a more structured and event-driven programming model.

h. **.NET Middleware Engine:**

* .NET Middleware Engine typically refers to components like ASP.NET Core Middleware.
* Middleware components in ASP.NET Core process requests and responses. Examples include authentication middleware, logging middleware, etc.
* Middleware allows developers to configure and extend the request/response processing pipeline.

i. **MVC Architecture:**

* MVC (Model-View-Controller) is an architectural pattern widely used in web development.
* **Model:** Represents the application's data and business logic.
* **View:** Displays the data to the user and handles user input.
* **Controller:** Manages the flow of data between the model and the view, handling user input and updating the model accordingly.
* MVC promotes separation of concerns, making applications more modular, maintainable, and scalable. It is a fundamental concept in ASP.NET MVC.