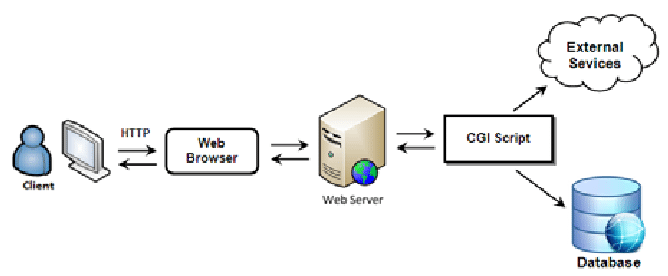
1. Explain how CGI server-side script execution works. [2080 Baisakh QN.4]

Ans. The Common Gateway Interface (CGI) is a standard (protocol) for providing an interface, or a gateway, between an information server and an external process (that is, a process external to the server).

The web server fetches the CGI script, and activates it as a process, passing to the process input data transmitted by the web client. The web script executes and transmits its output to the web server, which returns the web-script-generated data as the body of a response to the web client. CGI scripts are executables that will execute on the server to produce dynamic and interactive web pages.

Working Principle:

* The browser sends an HTTP GET or POST command to the web server with the parameters, along with the path to the requested document (which in this case is a CGI program, although the browser doesn't know or care about this distinction).
* The web server notes that the requested document is a CGI program rather than a static HTML document.
* The web server initializes certain environment variables with the passed parameters, runs the CGI program in a newly-created OS process, and the CGI program in turn produces an HTML document as its output, making use of the parameters as necessary.
* The web server returns the generated HTML document to the browser, passing along the header information as before.

1. Write the difference between b) Client-Side Scripting and Server-Side Scripting. [2080 Baisakh QN.5 b)]

Ans. The differences between Client and server-side scripting is given below:

|  |  |  |
| --- | --- | --- |
| Aspect | Client-Side Scripting | Server-Side Scripting |
| Execution Location | Executes on the client's browser | Executes on the web server |
| Languages | JavaScript, HTML, CSS | PHP, Python, Ruby, ASP.NET, Node.js, Java |
| Interaction with Server | Limited to sending requests to the server | Full access to server resources and databases |
| Response Time | Immediate response, as it doesn't require server communication for execution | May take longer due to server processing and database interactions |
| Security | Less secure, as code is exposed to the client | More secure, as code is hidden from the client |
| Use Cases | Form validation, animations, interactive content | Database operations, user authentication, dynamic content generation |
| Resource Access | Limited to the resources loaded in the browser | Full access to server resources, including databases and files |
| Examples | React.js, Angular, Vue.js | Django, Flask, Laravel, Express.js |
| Dependency | Depends on the client's browser and its capabilities | Depends on the server's configuration and resources |
| Performance Impact | Affects the client's device performance | Affects the server's performance and scalability |
| Data Storage | Temporary storage using cookies, local storage | Persistent storage using server-side databases |
| Environment Control | Controlled by the client | Controlled by the server |

1. Write the program to connect the database and insert data using server-side scripting language [Assume database\_name "employee", table\_name "info" with columns id, firstname, lastname, address, post, and contact]. [2079 Bhadra QN.4]

Ans. The script of a server-side uing PHP to connect to a MySQL database is given below:

PHP Script: ‘insert.php’

<?php

// Database connection parameters

$servername = "localhost";

$username = "root";

$password = "yourpassword";

$dbname = "employee";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Sample data to insert

$firstname = "John";

$lastname = "Doe";

$address = "123 Main St";

$post = "Manager";

$contact = "555-1234";

// Prepare and bind

$stmt = $conn->prepare("INSERT INTO info (firstname, lastname, address, post, contact) VALUES (?, ?, ?, ?, ?)");

$stmt->bind\_param("sssss", $firstname, $lastname, $address, $post, $contact);

// Execute the statement

if ($stmt->execute()) {

echo "New record created successfully";

} else {

echo "Error: " . $stmt->error;

}

// Close the statement and connection

$stmt->close();

$conn->close();

?>

1. What do you mean by server-side processing? Explain Multitier architecture. [2078 Bhadra QN. 3]

Ans. **Server-side processing** refers to operations that are performed on the server in a client-server architecture. When a client (e.g., a web browser) sends a request to a server, the server processes the request, performs the necessary computations or data manipulations, and then sends the response back to the client. Server-side processing is typically used for tasks that require secure access to resources, complex computations, or interactions with databases and other services.

Characteristics:

* **Security**: Sensitive operations and data handling are done on the server, reducing exposure to the client.
* **Complexity**: Handles complex logic and data processing that is impractical to perform on the client side.
* **Performance**: Can leverage the server's resources for intensive tasks, reducing the load on the client's device.
* **Database Interaction**: Provides secure access to databases, enabling data retrieval, storage, and management.

N-tier architecture (with N more than 3) is a 3-tier architecture in which the middle tier is split up into new tiers. The application tier is broken down into separate parts. What these parts are differs from system to system.

The primary advantagesof N-tier architectures:

* They make load balancing possible.
* Since the application logic is distributed between several servers, processing can then be more evenly distributed among those servers.
* N-tiered architectures are also more easily scalable, since only servers experiencing high demand, such as the application server, need to be upgraded.

The primary disadvantage of N-tier architectures is:

* It is also more difficult to program and testan N-tier architecture due to its increased complexity.

1. Explain the CGI server-side script execution mechanism. [2076 Chaitra QN.3]

Ans. *(Repeated question)*

1. Explain any three server-side scripting languages used in the web. Explain multi-tier architecture. [2074 Chaitra QN.5]

Ans. Any three server-side scripting languages are explained below:

1. PHP

* Overview: PHP (Hypertext Preprocessor) is an open-source server-side scripting language designed for web development, and it can be embedded into HTML.
* Key Features: Easy to learn, integrates well with databases (e.g., MySQL), and has a large community with many frameworks (e.g., Laravel) and content management systems (e.g., WordPress).
* Common Use Cases: Dynamic websites, form handling, user authentication, and CMSs.

1. Python

* Overview: Python is a high-level, interpreted language known for its readability and versatility, often used in web development with frameworks like Django and Flask.
* Key Features: Emphasizes readable and clean code, versatile beyond web development (e.g., data analysis, AI), and has an extensive library ecosystem.
* Common Use Cases: Web applications, RESTful APIs, data processing, and automation scripts.

1. Node.js

* Overview: Node.js is a runtime environment that allows JavaScript to be executed on the server side, built on Chrome's V8 engine, and known for its event-driven, non-blocking I/O model.
* Key Features: Handles multiple connections concurrently, uses JavaScript for both client and server and has a vast repository of packages via npm.
* Common Use Cases: Real-time applications (e.g., chat apps), RESTful APIs, single-page applications (SPAs).