The Implementation of PHP for Dynamic Web Pages

Rory Wood

Abstract.

This paper explores the implementation of PHP for dynamic web pages, examining its evolution, features, and practical applications. Each section will explore a unique PHP language feature and provide examples of how this feature allows dynamic web pages to be created.

Introduction.

Over the past couple of months I have taken an interest in web development which started with designing static HTML web pages with a CSS file to make it look presentable. I quickly realized the limitations of static web pages with the massive limitations to what you can do with them being limited to simple web pages to display basic data in a formatted fashion. Therefore, in order to add much more advanced features like a login system and the ability to connect to a backend database, I decided to look into learning more about the PHP language to add some dynamic functionality to the website.

Overview of PHP.

PHP was originally developed in 1994 by Rasmus Lerdorf as a tool for tracking visits to Lerdorf's online resume, but gained popularity as an open-source, general-purpose scripting language fit for server-side programming. Originally, the acronym PHP stood for ‘Personal Home Page’, but now it is simply known as hypertext preprocessor. Over the years PHP has become a cornerstone of dynamic web pages and its compatibility with various web servers contribute to its widespread adoption, making it an ideal candidate for projects looking to avoid the limitations of static web pages.

The PHP language offers a versatile solution for developers, allowing them to embed PHP code directly into HTML, thus allowing for the creation of dynamic and interactive web pages. Over time, PHP has become a feature-rich language, equipped with a vast standard library and frameworks that streamline web development and make the scripting language easy to be picked up and understood by even beginner programmers. The open-source nature of PHP has also allowed for a vibrant community to contribute to its continuous enhancement and addition of features and tools to aid the needs of web developers. The exploration of PHP in this project looks at how the language is able to add dynamic functionality to web pages, offering a solution to the challenges posed by static designs.

Flexibility Across Operating Systems and Devices.

Possibly the most important feature of PHP that made it so popular to web developers worldwide is the flexibility of the language across operating systems and devices. PHP works mainly as a server-side scripting language and its compatibility with major operating systems like Linux, Windows, and macOS ensures it can be used on most devices, making it an attractive choice for developers looking for some cross-platform functionality. The discussion on PHP's flexibility is a foundation for understanding how this language is a powerful tool for dynamic web functionalities.

Loosely Typed Nature of PHP.

Similar to many modern languages like Javascript and Python, part of PHP's appeal lies in its loosely typed nature, promoting a much more flexible dynamic coding environment compared to strongly typed languages like Java and C that mandate explicit declaration of variable data types. This flexibility allows variables to be assigned without specifying their data types upfront. The variable's data type is not determined until runtime, based upon the value assigned to it. This characteristic streamlines the coding process, removing the need for somewhat redundant type declarations.

Furthermore, PHP adds another dimension by allowing dynamic changes to variable names during runtime. This dynamic change of variable names adds another layer of flexibility which allows code dynamically changed based on evolving requirements. This is achieved with a combination of using "passing by reference" and "variable variables" an example of this would be as follows:

```
// Original variable name
$originalVariable = "Original Value";

// Dynamic change of variable name
$newVariableName = "originalVariable";
$$newVariableName = "New Value";

// Output
echo $originalVariable;  // Output: New Value
```

In this example, the double dollar sign ($$) is used to reference the variable whose name is stored in $newVariableName. As a result, the value of $originalVariable is dynamically changed to "New Value" through the manipulation of variable names during runtime.

Project Overview.

The following sections will include extracts from my coding project to demonstrate unique features of the PHP language. The project I have chosen is an example of a set of dynamic web pages I designed from scratch, including the HTML, CSS and some small Javascript features as well.

Database Connection.

The two examples below showcase how PHP can seamlessly interface with a MySQL database, creating a secure method for user authentication in a dynamic web application.

Conn.php

```
<?php
$serverip = "localhost";
$username = "websiteAccess";
$password = "accessPassword";
$dbname = "website";

// Create connection
$conn = new mysqli($serverip, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
```

In the above code segment from the `conn.php` file, the script initiates a connection to the MySQL database. Parameters such as `$serverip`, `$username`, `$password`, and `$dbname` are configured to match the database's specifications. The mysqli extension is then utilized to create the connection, and the script includes a validation check to ensure the connection is established successfully. If any issues arise during the connection process, an error message is generated, terminating the script.

Login.php

```
<?php
```

```
include_once 'conn.php';
session_start(); // Start the session

$error = '';
// Process login form data
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $username = $_POST['username'];
    $password = $_POST['password'];

    // Query the database to check if the provided credentials
are valid
    $query = "SELECT userid, username FROM users WHERE username
= '$username' AND password = '$password'";
    $result = mysqli_query($conn, $query);

    if (mysqli_num_rows($result) > 0) {
        // Login successful, store userid in session and
redirect to homepage.php
        $row = mysqli_fetch_assoc($result);
        $_SESSION['userid'] = $row['userid'];
        $_SESSION['username'] = $row['username'];
        header('Location: ../loginPage/home.php');
        exit;
    } else {
        $error = "Invalid credentials. Please try again.";
    }
}
```

This `login.php` script shows a practical application of the database connection. After receiving the users credentials via a login form, the script initiates a database query using the $query statement. This query attempts to match the provided username and password with the entries stored in the 'users' table of the database. If a match is found, the script stores the user's userid and username in a session, facilitating secure user authentication. In the case of unsuccessful login attempts, an error message is generated.

Form Handling.

The `login.php` script also shows an example of how the PHP language uses form handling with the `$_POST` superglobal which allows for easy retrieval of form data submitted by users. This built-in support simplifies the process of processing user inputs without the need for complex JavaScript or additional frameworks.

Embedded HTML.

PHP's unique capability to seamlessly embed HTML within its code contributes towards the creation of dynamic and data-driven web pages. The following example demonstrates how PHP integrates with HTML to display a table representing the average distance hit by various golf clubs fetched from a MySQL database.

golfStats.php

```
<table>
    <tr>
     <th>Club</th>
        <th>Average Distance(yds)</th>
    </tr>
    <?php
```

```
    $sql = "SELECT ad.club, ad.avg_distance
    FROM average_distances ad
    JOIN users u ON ad.userid = u.userid
    WHERE u.userid = '$userid'
    ORDER BY ad.avg_distance DESC;";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        while ($row = $result->fetch_assoc()) {
            echo "<tr><td>" . $row["club"] . "</td><td>" .
$row["avg_distance"] . "</td></tr>";
        }
    }
    ?>
</table>
```

The PHP block within the HTML structure dynamically retrieves data from the database and generates rows within the table for each record. The loop then iterates through the result set, echoing HTML table rows with the corresponding golf club and average distance data. This dynamic integration of PHP with HTML enables the creation of responsive and data-driven web content, showcasing PHP's ability to enhance the presentation of various web applications.

Conclusion.

In summary, looking into PHP's dynamic capabilities reveals its crucial role in modern web development. From its historical evolution to practical examples like logging into a web page and displaying data from a back end database on a web page, PHP is a versatile scripting language for creating responsive web applications, and its seamless integration with HTML to create an adaptable presentation. What I found most interesting was its seamless integration with

HTML, allowing for the creation of adaptable presentations, coupled with its robust features, such as dynamic variable handling and native support for form processing. Overall, with a flexible coding environment and cross-platform adaptability, PHP stands to be much more than just a simple scripting language.

Citations

PHP History - https://www.emizentech.com/blog/what-is-php.html#:~:text=PHP%20is%20a%20type%20of,is%20 known%20as%20 Hypertext%20Preprocessor.

PHP Unique Features - https://www.geeksforgeeks.org/php-unique-features/

PHP Manual - https://www.php.net/manual/en/index.php