Arkesel

Internal

NSP TRAINING STRUCTURE FOR PHP BEGINNERS

Purpose for this structure:

This roadmap aims to break down the learning process into manageable phases and tasks, ensuring a gradual and comprehensive understanding of PHP concepts and practical skills.

Goal at the End of the Studies:

At the end of this journey, the individual should have achieved a deep understanding of PHP programming, enabling them to confidently build dynamic and interactive web applications. They should be equipped with the skills to develop complex backend functionalities, interact with databases, and integrate with front-end technologies. Additionally, the roadmap hopes to foster the ability to troubleshoot and debug code effectively, adhere to best practices, and potentially contribute to open-source projects.

Ultimately, the goal is to empower the person with the knowledge and skills needed to thrive in the PHP development field.

Recommended Resources:

1. **PHP Fundamentals**:
    - PHP Manual
    - PHP: The Right Way
    - Video: PHP Crash Course
2. **Diving Deeper into PHP**:
    - W3Schools PHP Forms
    - MySQL Tutorial
    - Video: PHP & MySQL Tutorial
3. **Object-Oriented Programming (OOP)**:
    - PHP OOP Tutorial
4. **Building Practical Applications**:
    - PHP CRUD Tutorial
5. **Security**:
    - PHP Security Best Practices

Phase 0: Setting Up the PHP Environment

1. **PHP Installation**: Understand how to install PHP on different operating systems.
2. **Local Server Setup**: Set up a local server environment using tools like XAMPP or MAMP.
3. **IDE/Text Editor**: Choose and set up an IDE or text editor suitable for PHP development, such as Visual Studio Code or PHPStorm.

Phase 1: PHP Fundamentals

Task 1: PHP Syntax and Variables

1. **Syntax**: Understand the language's syntax, which includes its rules for writing code, such as using proper punctuation, indentation, and keywords.
2. **Variables and Data Types**: Learn how to declare variables, assign values to them, and understand different data types like integers, floats, strings, and boolean.
3. . **Operators**: Master arithmetic, comparison, logical, and assignment operators, which allow you to perform calculations, make comparisons, and manipulate data.
4. Based on what you have learnt so far, create a simple PHP script that calculates yearly profits.
   a. Create a PHP script that calculates the yearly profits based on the following inputs:
      - Monthly Gross Salary
      - Monthly Net Salary
      - Monthly Household Expenditure
   b. Assumptions:
      - There are 12 months in a year.
      - Gross salary remains constant throughout the year.
      - Net salary and household expenditure are considered per month.

Task 2: Control Structure and Loops

1. **Control Structures**: Study control flow mechanisms like conditionals (if statements, switch statements) and loops (for loops, while loops), which help you manage the flow of your program.
2. Build a program that iterates through an array and displays its content.

Task 3: Functions

1. **Functions**: Understand how to define and call functions. Functions allow you to encapsulate code for reuse and modularity.
2. Build a program that employs the use of functions and control structure and display the output.

Phase 2: Diving Deeper into PHP

Task 4: Working with Forms

1. Create a form in HTML. The form could handle taking data like first name, last name, email and phone number.
2. Process the form data by getting and displaying the data submitted in the form.

Task 5: Introduction to Databases

1. Learn about MySQL and relational databases.
2. Set up a local database using tools like phpMyAdmin.

Task 6: Connecting PHP to Database

1. Connect to your newly created database via PHP.
2. With the existing form you have, try and save the data from the form to a table in the database.

Phase 3: Object-Oriented Programming (OOP)

Task 7: OOP Basics

1. Learn the fundamentals of OOP: grasp the concepts of classes, objects, methods, and encapsulation.
2. Create a simple PHP class and instantiate objects.

Task 8: Building Practical Applications

1. Build a mini ticketing system for your application which supports user authentication.
    a) create 4 endpoints in this ticketing system.
        I. an endpoint to Get all events and their tickets for users.
        II. an endpoint to Buy ticket for an event for users.
        III. an endpoint to create an event for users.
        IV. an Admin endpoint for only admins to view all event tickets.
    b) Add create, read, update and delete (CRUD) actions for event owners and ticket owners. Remember to add data integrity constraints.

This task should be implemented in PHP alongside any other stack that can aid you arrive at your solution quicker.

Note: Try to utilize all the concepts learnt from previous tasks however best you can in building this application.

Phase 4: Advanced Topics

Task 9: Error Handling and Debugging

1. **Introduction to PHP Errors**: Understand the different types of errors in PHP: Parse, Fatal, Warning, and Notice.
2. **Error Reporting**: Learn how to configure PHP to display or log errors for debugging purposes.
3. **Custom Error Handlers**: Create custom error handling functions to manage errors gracefully.
4. **Practical Application**: Implement a PHP script that intentionally produces errors and then handle them using the techniques learned.

Task 10: PHP Security Best Practices

1. **Data Validation**: Understand the importance of validating user input to ensure it's safe and meets the expected format.
2. **Data Sanitization**: Learn techniques to sanitize user input, removing any harmful data before processing.
3. **Preventing SQL Injection**: Understand what SQL injection is and how to use prepared statements to prevent it.
4. **Password Hashing**: Learn about PHP's built-in functions for securely hashing and verifying passwords.
5. **Practical Application**: Create a secure login system that validates and sanitizes user input, prevents SQL injection, and securely hashes passwords.

Task 11: Introduction to PHP Frameworks

1. **Why Use a Framework?**: Understand the benefits of using a framework over vanilla PHP.
2. **Popular PHP Frameworks**: Get an overview of popular PHP frameworks like Laravel, Symfony, and CodeIgniter.

Recommended Resources for Phase 4:

- **Error Handling**: PHP Error Handling, Custom Error Handling in PHP
- **Security**: PHP Security Best Practices, PHP Data Validation, Preventing SQL Injection in PHP, Password Hashing in PHP
- **PHP Frameworks**: Why use a PHP Framework, Getting Started with Laravel