LUC/FCSMM/DCS/05/0020 David Samuels 2024-07-19 08:09:05

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 10:40:11 **Number of words : 1250**

SECTION A.

1. (1) Functions are blocks of code that can be defined once and executed multiple times throughtout a script. they are very important for organizing code into reusable segments , improve readability and maintaining effeciency. To define a function, you use '**function**' keyword then the function name and parentheses'**()**'. the function block is closed in '**{}**'. When the function is defined, you can call it to execute the code inside. to call a function, write its name followed by parentheses'**()**'.
2. (2) PHP is an open source scripting language designed for web development majorly. It runs on the web server where your website is hosted. Primary uses of PHP are; Server side scripting, Web development, Command line scripting, Database Access, CMS, E-commerce applications.

2B) Echo is not a function but a language construct, so parentheses are optional when using it. it can output more than one parameters seperated by commas. it doesnt have a return value,because of this it is faster than 'print'. the little difference in speed makes it more viable when outputting large amounts of data. Whilst print is also a language construct it behaves more like a function so parentheses are required. it can output a single value at a time. it returns '1' and this can be useful in expressions. it is slower than echo.

3. (6) It handles form data by capturing it from the HTML form submitted by the user. it can access the form data through predefined variables

4. (7) 'include' evaluates the specified file during script execution. if a file cant be include a warning is issued but the script will continue to execute. If a file is included multiple times, PHP will include each time. 'require' evaluates the specified file during script execution too. but if the file can not be found a fatal error is issued and the script stops. if a a file is required multiple times PHP will show fatal error. 'include_once' includes and evaluates the specified file during script execution but only if it has not been included before. if its been included before in the script, it will not include it again. It does not support multiple inclusion. 'require_once' incudes and evaluates the specified file during script execution, but only if hasnt been included already. If a file has been included already it wont include it again. also doesnt support multiple inclusion

5. (8) Error reporting functions, error logging functions, exception handling, error suppression, custom error handlers. Error can be managed in php script by always logging errors and exceptions to a file for debugging purpose, instensive testing your php scripts, to identify and handle errors effectively before deploying to production.

SECTION B.

1. (3) In user authentication, sessions are crucial for implementing user authentication system. upon successful login, PHP can store user credentials or IDs in session variables, allowing server to validate subsequent request without requiring the user to re-authenticate on each page. In e-commerce, session are used to maintain shopping carts contents across pages. product selections and quantities can be stored in session variables until the user completes their their purchase. In customization session and cookies enable wesites to customize content based on user preferences and previous interactions.

a) Starting and managing sessions in PHP involves a series of steps to initialize, store, and use session data throughout a users interaction with the website. the steps are as follows; session_start() -to initiate a session, to set session variables- $_session , to access session data simply acess '$_SESSION' with corrsponding key, for unsetting session variable use 'unset()',destroy a session by using 'session_destroy()'.

b) sessions are stored in server side, they are temporary, sessions are larger and more secured, they are used for storing user authentication status and other user specific state information, they are managed automatically by PHP. while cookies are stored in the client side, you can set a particular expiration time, the storage capacity is up to 4kb of data per domain, it is less secure, its used for tracking and personlization, cookies need to be explicitly set, updated and delted by the application.

c) Use HTTPS, set secure and Httponly flags, limit cookie scope, encrypt sensitive data, validate and sanitze data, use samesite attribute

d) sessions are commonly used to manage user logins. when a user logs into a website, a session is created to maintain their authenticated state accross different pages until the log out.

website use cookies to remember user preferences such as language settings or display preferences such as language setting or display preferences.

2. (2) Request initiation:

- HTTP Request: The lifecycle begins when a client sends an HTTP request to the web server.
- Web Sever Handling: The web server recieves the request and identifies it as needing PHP processing

PHP Initialization:

- PHP Interpreter: Once the web server identifies the request as requiring PHP, it intiliazes the PHP interpreter.
- Script Execution: The PHP interpreter starts executing the requested script line by line.

Preprocessing:

- Configuration: PHP reads and applies the configuration settings.
- Session initialization: If sessions are used, PHP initiliazes session data.

- Auto loading: If using namespace and classes, PHP may autoload class files if not already loaded.

Script Execution:

- Code Execution: PHP executes the script according to its logic.
- Includes and Requires: PHP processes 'include', 'require', 'include_once', 'require_once' statements by including and executing code from other files as necessary.
- Error Handling: PHP handles errors and exceptions based on the configured error handling settings.

Output Generation:

- Content Generation: As the script executes, it generates HTML, text, JSON or other content to be setn back as the response.
- Buffering: PHP may buffer output to optimize performance, especially when generating large amounts of content.

Response Sending:

- HTTP Headers
- Output

Cleanup and termination:

- Session Cleanup:
- Script Termination
- Log activity

Client Reception:

- Client Response
- Browser Rendering

b) PHP interpreter and Script execution

c)

- Session cleanup: If sessions are used, PHP may update session data and clean up resources associated with the session.
- script termination: PHP script execution ends. Resources like database connections, file handles, and memory allocations are released.
- log activity: PHP may log activities and errors as configured.

d) You can influence the lifecycle of PHP script with custom code by leveraging PHP's various hooks, functions and features that allow you to interven at different stges of script executing. Some ways to do this are as follows;

Using PHP hooks and functions, Using Object-Oriented Programming(OOP) principles, utilizing PHP extensions and libraries, handling sessions and cookies.

LUC-INT-NGA-002-100040 Abdulrahman Yari Yahaya 2024-07-19 08:11:55

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 10:00:24 **Number of words : 420**

SECTION A

Q2

a. PHP (Hypertext preprocessor) is a widely used open source script language that is especially suited for web development and can be embedded into HTML. PHP is primarily used for creating dynamic web pages and web application.

b. echo can take multiple parameters, seperated by commas and does not have a return value while

Print can only take one argument and always return 1, making it suitable for expressions.

Q4

1. Integer e.g"php $num=10;"
2. Float e.g "php price= 5;"
3. String e.g "php $name="Yari","
4. Array e.g "php $fruit = array("banana,mango");

SECTION B

Q2

a. During the initialization phase, the system prepares itself for operation. this phase typically involvevsetting up variables, allocating memory, establishing data structure, and performing any neccessary tasks to get the system ready to execute its main functions.

b.

1. PHP handle execution by parsing, compiling and executing the script to generate the desire output.

c. During the termination phase the system completes its tasks, releases allocated resources, and prepare to shutdown. This phase involves cleaning up any remaining processes , closing open connections, deallocating memory, and ensuring that all resources used during the system operationare properly released.

d. You can influence the lifecycle with custom code by using functions or hooks provided by the programming language or frame work you are working with.

Q3

a. To start a session in php , use 'session_start()', set session variables with '$_SESSION', access them with '$_SESSION['variable_name'], end a session with 'session_destroy()', and manage session settings using PHP function or

b. Session store data on the server, are more secure, have larger capacity, and expire after a period of inactivity, While cookies store data on client side, are less secure, have limited capacity, and can have a specific expiration time or be store indefinitely.

c. To store data in cookies use 'setcookies()' function with the desired parameters, and to retrieve data, access the '$_COOKIE' superglobal array.

d. a. COOKIES

1. Remember me functionality
2. Personalization
3. Tracking

b. SESSION

1. User authentication
2. Shopping cart
3. Form data persistence

LUC-INT-NGA-002-EL-APTECH-100042 Yahaya Muhammed Ad-dahuk 2024-07-19 08:13:50

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 10:44:53 **Number of words : 1565**

Yahaya Muhammed Ad-dahuk

Section A

1a) PHP (Hypertext Preprocessor) is an open-source scripting language that is es[ecially suited for web development and can be embedded into HTML. PHP is primarily used for *Web Development, server side scripting, Database interaction,* and *E-commerce development.*

1b) DIFFERENCES BETWEEN ECHO AND PRINT:

i) Print can only output one string, Echo can output multiple strings.

ii) Echo doesn't return a value, Print returns a value. Hence, Echo is faster than Print.

iii) Print always uses parentheses when used with a function, Echo doesn't need parentheses.

2a) TYPES OF VARIABLES WITH EXAMPLES:

i) Interger ($age = 30;)

ii) Float e.g $price = 25.50;

iii) String e.g $name = "Felix";

iv) Boolean e.g $isValid = true;

v) Array e.g $fruits = array("Apple", "Banana", "Orange");

vi) Null e.g $var = NULL;

vii) Objects.

viii) Associative array.

2b) HOW PHP HANDLES DATA TYPES:

i) Dynamic Typing: PHP determines the type of the variable based on its value. e.g $variable = 42;

ii) Type Juggling: PHP automatically convertstpes as needed in different contexts e.g $number = "5"; $result = $number + 10; ("5" is an int now)

iii) Type casting: Covert a variable from one type to another using casting e.g $number = "10.3"; $int_number = (int)$number;

iv) Type functions: Check and convert types using built-in functions.

v) Type declaration: Declare types for function argumentts and return values to enforce type safety

3) in PHP, CONSTANTS are identifiers for simple values that, when defined, cannot be changed during the execution of a script. They are used for defining values that should remain constant throughout the application. Constants can be defined by either using the **define(),** or using **const**.

4) THE DIFFERENCE BETWEEN INCLUDE, REQUIRE, INCLUDE_ONCE AND REQUIRE_ONCE:

i) Include: This includes the file and continues the execution with a warning on faliure.

ii) Require: This includes the file and stops the execution with a fatal error on faliure.

iii) Include_once: This includes the file only once, and continues the execution with a warning on faliure.

iv) Require_once: This includes the file only once, and stops the execution with a fatal error on faliure.

5) PHP ERROR HANDLING FUNCTIONS:

i) error_reporting(): This sets which PHP errors are reported.

ii) set_error_handler(): This sets a user_defined error handler function.

iii) trigger_error(): This generates user level error/warning.

iv) restore_error_handler(): This restres the previous error handler function.

v) set_exception_handler(): This sets a user-defined error handler function.

vi) restore_exception_handler(): This restores previous error handlers

vii) error_get_last(): This gets the last error that occured.

HOW ERRORS CAN BE MANAGED IN A PHP SCRIPT:

i) Displaying errors for Deployment, Log errors for production: In development, it is helpful to display errors to quickly identify issues. Also, in production, errors can be logged to avoid exposing sensitive information to users

ii) Set Error Reporting level: Define the level of error reporting based on the enviroment.

iii) Custom error and exception handlers: Define custom handlers to manage errors and exceptions in a controlled manner.

iv) Error suppression operator: Use the @ operator to suppress errors for specific expressions. This can make debugging difficult.

v) Using try-catch blocks: Wrap code that might throw exceptions in try-catch blocks.

Section B

1) Error handling in php is the process of catching and managing errrors that occure during the execution of a script. It helps in identifying bugs, maintaining application staqbility, and provides user friendly error messages.

1a) TYPES OF ERROR IN PHP:

i) Parse Errors(Syntax Errors): This Occurs when PHP encounters syntax issues in the script.

ii) Fatal Errors: Occurs when PHP cannot continue the execution due to critical issues.

iii) Warning errors: Occurs for non-critical issues, script execution is not halted, but a warning message is displayed.

iv) Notice Errors: Occurs for minor issues. They do not halt script execution.

v) Exceptions: These are special objects representing an error condition in a program typically thrown and caught with try-catch blocks.

1b) HOW YOU CAN USE TRY, CATCH AND FINALLY BLOCKS FOR EXCEPTION HANDLING:

i) Try Block: Contains the code that may throw the exception

ii) Catch Block: Catches and handles the excpetion if one is thrown in the try block.

iii) Finally Block: Contains code that will always execute regardless of whether an exception is thrown.

1c) ROLES OF set_error_handler() AND set_exception_handler() FUNCTIONS:

- set_error_handler() registers a custom function to handle errors. it allows you to define how errors are reported and managed, but it does not handle fatal errors.
- set_exception_handler() registers a custom function to handle uncaught exceptions. It provides a way to manage exceptions globally when they are not caught within the try-catch block.

2) The Lifecycle of a PHP script refers to the various stages that the script ges through from the moment it was requested by a user to its final execution and cleanup. The lifecycle includes; *Request initiation, PHP interpreter initialization, script parsing, script execution, output generation, response transmission, script termination,* and *post-execution.*

2a) WHAT HAPPENS DURING THE INITIALIZATION PHASE:

i) Request Handling: The web browser receives and processes the request for the php script.

ii) PHP Interpreter: The PHP interpreter is invoked and initialized.

iii) Configuration: PHP loads configuration setting from php.ini and other sources.

iv) Resource initialization: Internal components and resources are set up.

v) Error Handling: PHP sets up error and exception handling mechanisms.

vi) Extensions: PHP loads necessary extensions and modules.

2b) HOW PHP HANDLES EXCEPTIONS:

i) Throwing Exceptions: This uses throws to raise exceptions

ii) Catching exceptions: This uses try to enclose the code that may throw exceptions and catch to handle them.

iii) Exception object: Provides methods for accessing exception details.

iv) Finally Block: Ensures that code always runs regardless of whether an exception was thrown.

2c) THE TERMINATION PHASE:

i) Script Execution Completion: The script finishe s executing and sends output to the client.

ii) SHutdown Functions: Functions registered with register_shutdown_function() are executed.

iii) Error and Exceptiom Handling: Uncaught exceptions are handled and errors are logged if configured.

iv) Session Management: Session data is saved and sessions are closed.

v) Resource Deallocation: Resources are automatically closed and memory is freed.

vi) Output buffering: Buffered output is flushed and sent to the client.

vii) Finalization: Final cleanup tasks are performed, and script execution is terminated.

2d) INFLUENCE THE LIFECYCLE WITH CUSTOM CODE:

i) Custom Error and exception handling: Use set_error_handler() and set_exception_handler() to manage errors and exceptions.

ii) Shutdown Functions: Register functions to run at the end of script execution using register_shutdown_function().

iii) Output buffering: Control and modify output using output buffering functions.

iv) Sessions and resources Management: Customize sessions handling and manage resources explicitly

v) Error logging and reporting: Configgure error reporting and logging settings.

LUC0001110213362019 Saliu Muhammad 2024-07-19 08:15:33

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 10:30:16 **Number of words : 1799**

Name: SALIU MUHAMMAD

Student ID: LUC0001110213362019

Course code: CSE251

SECTION A

QUESTION 1

A function is a piece of code that takes another input in the form of a parameter, processes it, and then returns a value. A PHP function feature is a piece of code that can used over and over again, and accepts argument lists as input, and returns a value.

How do you define and call a function?

i. Function Declaration: Use the "function" keyword, followed by the function name and block of code enclosed within curly braces "{ }" to define a function.

ii. Function Parameters: Define any parameters the function may accept within the parenthesis "( )".

iii. Function Body: Write the code that the function will execute within the curly braces "{ }".

iv. Function Call: Invoke the function by its name followed by parenthesis "( )" containing any required arguments.

QUESTION 2

2A. PHP ia an open-source, server-side programming language that can be used to create websites, applications, customer relationship management systems and more. It is a widely-used general-purpose language that can be embedded into HTML.

What is PHP Primarily used for?

PHP is mostly used for making web servers. it runs on the browser and is also capable of running in the command line.

2B. Below listed are some basic differences between "echo" and "Print" in PHP:

i. Echo does not return a value. **While** Print returns a value of 1.

ii. Echo takes multiple parameters separated by commas. **While** Print only takes one parameter.

iii. Echo can display multiple strings separated by commas. **While** Print can only display one string at a time.

iv. Echo is faster and more efficient than print. **While** Print is slower than echo.

v. Echo is used for displaying simple text or HTML code. **While** Print is used in more complex expressions or functions where a return value is needed.

vi. Echo can be used in control structures such as if, while, and for. **While** Print cannot be used for control structures.

QUESTION 3

An Array is a special variable that we use to store or hold more than one value in a single variable without having to create more variables to store those values. To create an array in PHP, we use the array function "array( )". By default, an array of any variables starts with an index of 0.

Difference between Indexed arrays, Associative arrays, and Multidimensional array:

Indexed arrays can be define as arrays with a numeric key. While an associative array is a type of array where each key has its own specific value. And lastly, a multidimensional array is a type of array containing one or more arrays within itself.

QUESTION 6

How does PHP handle form data?

When using PHP with an HTML form, we specify the URL of the PHP script in the action attribute of the form tag. The data passed by the form is then accessed by the target PHP code using PHP's "$_POST" or

"$_GET" variables, depending on the value of the form's action attribute ("get" or "post").

Basic Example of PHP $_POST:

A HTML form submits information via the HTTP POST method if the form's method attribute is set to "POST".

When a user clicks the submit button, the form data is sent to a PHP file specified in the action attribute of the "<form>" tag.

In the action file we can use the "$_POST" variable to collect the value of the input field.

QUESTION 7

i. include() : if an error occurs, the include() function generates a warning if an error occurs, but the script will continue execution.

ii. require() : If an error occurs, the require() generates a fatal error, and the script will stop.

iii. include_once: the include_once keyword is used to embed PHP code from another file. If the file is not found, a warning is shown and the program continues to run. If the file was already included previously, this statement will not include it again.

iv. require_once: the require_once function can be used to include a PHP file in another one, when you may need to include the called file more than once. If it is found that the file has already been included, calling script is going to ignore further inclusions.

SECTION B

QUESTION 1

1A. Below are different types of errors in PHP:

i. Praise error or Syntax error.

ii. Fatal error.

iii. Warning errors.

iv. Notice errors.

1B. i. The **try** block is used to specify a block of code that may throw an exception.

ii. The **catch** block is used to handle the exception if it is thrown.

iii. The **finally** block is used to excute the code after the **try** and **catch** blocks have been executed.

1C. i. set_error_handler(): the set_error_hanlder() function is used to tell PHP how to handle standard engine errors that are not instances of the Error exception class. The standard PHP error handler is completely bypassed if this function is used, and the user-defined error handler must terminate the script.

ii. set_exception_handler(): the set_exception_handler() function sets a user-defined exception handler function. The script will stop executing after the exception handler is called.

1D. Since wwe want our custom function to handle all errors, the set_error_handler() only needed one parameter, a second parameter could be added to specify an error leel.

Example:

```
<?php
      //error handler function
     function customError($errno, $errstr) {
          echo "<b>Error:</b> [$errno] $errstr";
      }
     //set error handler
    set_error_handler("customError");
    //trigger error
    echo($test);
?>
```

QUESTION 3

3A. You can start a session in PHP by using the session_start() function. This function will by default, first check for an existing session. If a session already exists, it will do nothing, but it will create one if there is no pre-existing session aailable.

3B. Below listed are some differences between sessions and cookies:

i. Sessions are stored on the server. **While** Cookies are stored on the client's browser.

ii. sessions are accessible only through PHP on server. **While** Cookies are Accessible by client-side scripts and PHP.

iii. Sessions lasts until the browser is closed or session times out. **While** cookies expires based on the specified duration, can persist beyound browser sessions.

iv. Sessions generally has no practical size limit, depends on server's memory. **While** cookies are limited to about 4KB per cookie.

v. Sessions are more secure, as it is not exposed to the client-side. **While** cookies are less secure, vulnerable to client-side access and manipulation.

3D. i. Sessions: Used for user authentication, managing shopping cart data in ecommerce websites, maintaining user preferences, and storing temporary user-specific information during a user's visit.

ii. Cookies: Employed for persisting login sessions, remembering user preferences (language, theme), tracking user behaviour for analytics or advertising, and storing session tokens for authentication.

LUC-INT-NGA-002-EL-APTECH-100038 Uthman Dantata 2024-07-19 08:31:34

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 10:22:37 **Number of words : 566**

SECTION A

1.

Functions help in reusability, modularity, and improving the readability of the code.

A function is defined using the function keyword followed by the function name and parentheses.

2.

A. PHP is a widely used open-source server-side scripting language designed specifically for web development.

B. echo can output multiple strings seperated by commas without parentheses and has no return value while print can only output one string and has a return value of 1.

4.

A constant is an identifier for a simple value. The value cannot be changed during the script.

They can be defined by 2 ways:

- using const keyword
- using define() function

5.

A. Session in PHP is a way of temporarily storing and making data accessible across all the website pages. It will create a temporary file that stores

SECTION B

1.

Error handling in PHP involves managing errors and exceptions that occur during the execution of a script. It allows developers to handle errors gracefully and ensure the application runs smoothly even when unexpected issues arise.

a.

- Fatal errors
- warning errors
- notice errors
- syntax errors

b.

try: the code that may throw an exception is placed inside the try block

catch: the code tht handles the exception is placed inside the catch block. It catches exceptions of a specific class or its subclass

finally: the finally block contains code that will always run, regardless of whether an exception was thrown or not

c.

the set_error_handler() function is used to tell PHP how to handle standard engine errors that are not instances of the Error exception class.

the set_exception_handler() function is used to set user-defined function as the default exception handler for uncaught exceptios

d.

set_error_handler("customError")

3.

Sessions and cookies are used to maintain state and keep track of information about user's interaction with a website. This is crucial because HTTP is stateless.

a.

- Starting a Sesion: session_start();
- Managing a session: Set session variable ($_SESSION['username']) => access session variable (echo $_SESSION['username']) => destroy a session(session_destroy)

b.

- Data is stored on the server in sessions while it is stored on the client's browser in cookies.
- Sessions are generally more secure
- Sessions can store large amounts of data while cookies are limited to about 4KB each
- Sessions usually expire when the browser is closed or after a set timeout period while cookies can last until manually deleted.

c.

To securely store and retrieve data in cookies, consider the following practices

- Encryption
- HTTP-Only and Secure Flags

LUC-INT-NGA-002-EL-APTECH-100040 John Albert 2024-07-19 08:40:29

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 10:29:32 **Number of words : 2167**

SECTION A

1. Function in PHP is a reusable block of code used to perform specific tasks.

   It makes writing code in PHP that much easier because:

   Reusability: If you have a block of code in your program that does a task

   that you would like to use in differents parts of your program, you can

   contain it as a function and call it whenever required.

   Error handling: It makes error handling that much easier to detect because

   you would know which function is giving the error

   Syntax for function:

   function function_name(){

   executable code;

   }

   Where "function" is the keyword used when defining a function, "function_

   name" is what name you would like to give the function.

2a. PHP is an acronym for: Hypertext Processor. PHP is a server-side scripting language that is primarily used for web development.

b. Echo is a language construct used to output multiple strings seperated by commas without parentheses while print is a language construct that can

only output one string and always returns 1.

3. Arrays in PHP are a data structure used to store multiple elements of similar or dissimilar data types under a single variable.

Indexed arrays uses a numeric index for their values to be stored linearly while associative arrays uses a string index and instead of their values being stored linearly, each value can be assigned to a specific key; however, multidimensional arrays contains a single or multiple arrays within itself and can be accessed with multiple indices.

4a. Scalar Variables:

Boolean

Integer

Double

String

Compound Variables:

Array

Object

Special Variables:

NULL

Resource

b. Pre-defined data types:

Boolean

Integer

Double

String

User-defined data types:

Array

Object

Special data types:

NULL

Resource

4. PHP constants are either identifiers or simple names that can be assigned fixed values. The define() function is used to create them as shown below:

   define(name, value, case_insensitive)

   Where "name" is the name of the constant, "value" is the value to be stored in the constant and "case_insensitve" defines whether it is case_ insensiteve(it is false by default).

5a. Session refers to a frame of communication between two medium. PHP session is used to store data on a server rather than the client side.

b. $_GET is a superglobal variable used to collect form data sent with the HTTP GET method.

   $_POST is a superglobal variable used to collect form data sent with the

HTTP POST method.

SECTION B

1. Error handling in PHP is how faults or mistakes are prevented from your program.

a. Syntax/Parse Error

Warning Error

Fatal Error

Notice Error

b. Try: This encapsulates a block of code in which an exception can arise

Catch: This represents a block of code that will be executed if when a particular exception has been thrown.

Finally: This contains a code that will execute regardless of the exception thrown; basically, it is put for clean up activity in PHP code.

c. The set_error_handler() sets a user-defined error handler function which will be called whenever an error occurs.

The set_exception_handler sets a user-defined exception handler function which will be called whenever an exception occurs.

d.
```
<?php
//function that will catch  the error
function myerror($error_no, $error_msg){
```

```
    echo "Error: [$error_no] $error_msg@;
    echo "\n Now the script will end";
    die();
  }
  set_error_handler("myerror");
  //code for the error to occur
 $a = 1
 $b = 0
 echo($a/$b);
  ?>
```

2. The life cycle of a PHP script encompasses the stages from the moment a PHP script is requested by the server until the response is sent back to the client. Understanding this concept optimizes performance, manages resources and aids in error handling.

a. Client Request: A PHP script is typically requested by a client through an HTTP request during initialization phase.
   Web Server: The web server then receives the request and identifies it as a request for a PHP script based on the file extension.

b. Memory allocation: PHP manages memory allocation and deallocation during execution, including variable storage and garbage collection.

c. Resource cleanup: PHP performs cleanup operations during termination

phase.

Session Handling: If session handling is used, PHP saves session data at this point.

Destructors: PHP calls destructors for objects that need cleanup before the script execution ends.

d.

```
<?php
//request initiation
require 'init.php';
//script exection
try{
  //session handling
  $action = filter_input(INPUT_GET, 'action', FILTER_SANITIZE_STRING);
  //routing
  switch($action) {
    case 'home':
      require 'home.php';
      break;
    case 'about':
      require 'about.php';
      break;
    default:
      require '404.php';
```

```
  }
 } catch (Exception $e) {
        //error handling
        echo "An error occurred: ". $e ->getMessage();
  }
 //Resource cleanup
require 'cleanup.php';
 ?>
```

LC000111000494 Talha Tanveer 2024-07-19 08:51:32

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 12:00:24 **Number of words : 2679**

SECTION A

Q1. A) In PHP, function are used to encapsulate a block of code that can be resused multiple times throughout a script. Functions help to organize and modularize your code making it more readable and maintainable.

Defining a Function: To define a function in PHP, you use the function keyword followed by the name of the function, parentheses and curly braces containg the code to be executed when the function is called.

Calling a Function: To call a function, you simply use the function name followed by parentheses. if the function takes parameters, you pass the arguments within the parentheses.

Q2. a) PHP (Hypertext Preprocessor) is a widely used open source scripting language especially suited for web development. It is embedded within HTML and is executed on the server, making it ideal for creating dynamic web pages and applications.

PHP is primarily used for server side web development. Its main purposes include:
1. Dynamic Web Page Creation: PHP can generate dynamic content on web pages allowing for personalized user experiences and real time data updates.

2. Server-Side Scripting: PHP scripts are executed on the server with the output sent to the client's web browser as HTML. This makes PHP iseal for tasks like form processing and session management.

3. Databse Interaction: PHP can connect to and interact with various databses (e.g. MySQL, PostgreSQL) enabling the creation, retrieval, updating and deletion of data. This capability is essential for building database driven applications like CMSs, e-commerce sites and forums.

b) In PHP, echo and print are both used to output data to the screen but there are some difference between them. here are the key distinctions:

1. Performance:
echo: Slightly faster than print since it dose not return a value.
print: Slightly slower due to returning a value, although the differece is generally negligible.

2. Use Cases:
echo: Preferred for genral output purposees due to its simplicity and slightly better performance.

print: useful when you need to use it within an expression because it return a value.

Q3. A) Arrays in PHP are versatile and powerful data structure used to store multiple values in a single variable. They are essential for managing collection of data allowing for efficient data manipulation and access. Here are the primary uses of arrays in PHP:

1. Storing multiple Values: Arrays allow you to store multiple values in a single variable, which is useful for managing related data.

2. Indexed Arrays: Indexed Arrays use numeric indicies to access elemnets. They are useful when the order of element is important.

3. Associative Arrays: Associative arrays use named keys to access elements. They are useful for storing data with a clear relationship between keys and values.

4. Multidimensional Arrays: Multidimensional arrays contain or more arrays. They are useful for storing complex data structures such as tables or matrices.

5. Iterating Over Arrays: Arrays can be easily iteratedover using loops like foreach, for and while. This is useful for processingor displaying all elements.

Difference Between Indexed, Associative, and Multidimensional Arrays: In PHP, arrays can be categorized into three main types:

Indexed Arrays: Indexed arraysuse nueric indices to store and access elemnets. These indices start from 0 and increase sequentially.
Usage: Indexed arrays are useful when you need to store a list of items where the order matters such as a list of colors or numbers.

Associative Arrays: Associative arrays use named keys to store and access elemnets. Each key is unique and is associated with a specific value.
Usage: Associative arrays are useful, when you need to map keys to values such as storing user informationor configuration settings.

Multidimensional Arrays: Multidimensional arrays are arrays that contain one or mpre arrays as their elements. These arrays can be indexed or associative.
Usage: Multidimensional arrays are useful for stioring complex data structures such as a table of data or hierarchical information.

Q5. a) Session in PHP are a way to store information to be used across multiple pages. Unlike cookies, session data is stored on the server rather than the client side. This makes sessions more secure and suitable for storing sensitive information such as user authentication data.

Key Concepts of PHP Sessions:

1. Session ID: Each session is assigned a unique identifier called a session ID. This ID ids tyoically storedin a cookie on the client's machine and sent to the server with each request allowing the sever to link the client to the stored session data.

2. Session Storage: Session dat is stored on the server often in temporary files or in a database. This data is accessible across multiple pages within the same session.

3. Session Variables: Session variables are used to store user specific information that needs to persist across different pages. These variables are stored in the global $_SESSION superglobal array.

Using Sessions in PHP:

* Before you can store or retrive session data, you need to start the session using the session_start() function. This function must be called at the begining of your script before any HTML or output is sent to the browser.
* We can set session variables by assigning values to the $_SESSION array.
* We can access session variables on any page that has session_start() called.
* We can modify session variables by resigning valuesto the $_SESSION array.
* We can remove specfic session variables using the sunset() function.

b) The $_GET and $_POST superglobals in PHP are used to collect data from HTML forms and to interact with URLs. They are associative arrays that store dat sent via HTTP GET and POST methods, respectively. Here's an in depth look at thiuer purposess:

$_GET Superglobal:

Purpose:

* $)POST is used to collect form data after submiting an HTML form with the method="post" attribute.
* It is commonly used for sending sensitive or large amounts of data since the data is not displayed in the URL.

$_POST Superglobal:

Purpose:

* $_POST is used to collect form data after submitting an HTML Form with the method="post" attribute.
* It is commonly used for sending sensitive or large amounts of data since the data is not displayed in the URL.

Q6. A) PHP handles form data using the $_GET and $_POT superglobals, depending on the form's method attribute. Here are basic example to illustrate how PHP handles form data.

Handling Form Data with PHP:

* Create an HTML form that allows users to input their name and age . This form will use the POST method to submit data to PHP script.
* Create PHP script that will allow handle the form data sent via the POST method.

SECTION B

Q2. The lifecycle of a PHP script involves several stages from the moment a request is madeby a client to the completion of the script's execution and the delivery of the response back to the client. Understanding this lifecycle is essesntial fr developing efficient and effective PHP applications. Here's a detailed ovrview of the lifecycle of a PHP script:

a) 1. Request Initialization: The lifecycle begins when a client makes an HTTP request to a server to acccess a PHP script.

2. PHP Initialization: The web server passes the request to the PHP interpreter. This can happen through various methods like CGI, FastCGI or as a module in the webserver.

3. Script Execution: The PHP interpreter parses the PHP script. This involves converting to the PHP code into a series of tokens and then building an abstract syntax tree.

4. Output Handling: PHP can use output buffering to collect script output before sending it to the client.

b) PHP handles execution through a series of well defined steps that involve interpreting, compiling and running the code. Here's detailed explanation of how PHP processes and executes scripts:

1. Request Reception: When a client (e.g. a web browser) sends an HTTP request to a server for a PHP script the web server such as Apache or Nginx receives this request for a PHP script based on the file extension or URL configuration.

2. PHP Initialization: The web server forwards the request to the PHP interpreter. This can be done through interfaces such as CGI, FastCGI or as aserver module.

3. Parsing: The PHP interpreter reads the PHP script the file system. The script's code is parsed into tokens using a lexical analyzer. These tokens represent the smllest elemnts of the script such as keywords, variables, operators and literals.

4. Compilation: The tokens are used an Abstract Sybtac Tree (AST) which represents the hierarchical structure of the code. The AST is then converted intpo intermediate code known as opcodes.

5. Execution: The Zend Engine executes the opcodes.

c) During the termination phase of a PHP script several important tasks occur to ensure that the script completes execution cleanly and resources are properly managed.

d) We can influence the lifecylce of a PHP script with custom code at various stages, from initialization to termination. hee are some ways we can do that:

Ways to do Influence the Lifecycle:

1. Initilizaation Phase: We can influence the initial configuration of your PHP environment using the PHP.

2. Script Execution Phase: Use the spl_autoload_register() function to automatically load classes, helpung manage dependencies and improve organization.

3. Termination Phase: Register shutdown functions to execute custom code at the end of the script execution.

Q3. In web development maintaing state between different requests from the same user is a critical aspect. PHP uses sessions and cookies to achieve this as HTTP itself is a stateless protocol.

a) starting and managing a session in PHP involves several steps incluidng initiating the session, storing and retrieving session session data and properly ending the session when its no longer needed. Here's a detailed guide on how to start and manage a session in PHP:

1. Starting a Session: To start a session we use the session_start() function. This function either creates a new session or resumes an existing session based on th session ID passed by the client.

2.Storing Data in a Session: We can store data in the session by assigning values to the $_SESSION superglobal array. This data will be available across different pages for the same user.

3 Retrieving Data from a Session: To retrieve session data we simply access the values stored in the $_SESSION array.

4. Checking if a Session Variable is Set: Before accessing session variables, its often a good idea to check if they are set to avoid errors.

5. Unsetting a Specific Session Variable: If we need to remove a specific session variable, we can use the unset() function.

6. Ending Session: To completely end a sesson and delete all session data, we use the session_unset() and session_destroy() functions.

b) Session and cookies are both used to store information about users and maintain state across different pages of a web application butthey work differently and have distinct characteristics. here are the key difference between sessions and cookies:

Storage Location:
Sessions: Store data on the server.
Cookies: Store data on te clients machine.

Security:
Sessions: Generally more secure because data is stored on the server.
Cookies: Less secure because data is stored on the clients machine.

Data Size:
Sessions: Can store large amounts of data since the data is stored on the server.
Cookies: Limited to a few killobytes (typically 4KB).

Lifetime:
Sessions: Exist only as long as the browser is open unless explicity.
Cookise: Can be set to expire at a specific time allowing them to persist.

Access:
Sessions: Data is accessed on the server side through the $_SESSION superglobal array.
Cookies: Data is accessed on the client side through the document. cookie object in Javascript or the $_COOKIE superglobal array in PHP.

c) To securely store and retrieve data in cookies we need to follow best practices to protect the data from being accessed or manipulated by unauthorized users.

d) Practical Use Cases for Sessions and Cookies:

Session:
User Authentication and Login Systems:
* Maintaing user login state across different pages.
Example:

```
session_start();

$_session['user_id'] = $user_id;

$_SESSION['username'] =

echo "Welcome", $SESSION['usefrname'];
```

Cookies:

Remember Me Functionality:
Example:

```
if

(isset($_POST['remeber_me])) {
  setcookie('username',

$username, time() + (86400 * 30), "/");
}

session_start();
$_SESSION["username] = $username;
```

LUC-INT-NGA-003-EL-APTECH100048 Usman Ginsau 2024-07-19 09:05:11

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 10:10:38 **Number of words : 486**

Usman Ginsau (LUC-INT-NGA-003-EL-APTECH-100048)

SECTION A

1) The use of functions in PHP is it serves as building blocks that can be used repeatedly and returns a value.

2)A) PHP is a coding language that is used on a web server.

3) An array is a special variable used to hold more than one value in a single variable.

-Indexed arrays are represented by a number.

-There are only 2 ways to define associative array

-Multidimentional array can be represented in the form of a matrix.

4) A constant in PHP is a name or an identifier for a simple value. They do not change.

8) PHP error handling functions are functions used to solve errors and logging.

Some common PHP error handling functions include :

-error_reporting

-display_errors

-log_errors

-report_memleaks

-track_errors

SECTION B

1) Error handling is catching errors generated by your program/code and taking the steps to solve the error.

A) Types of errors in PHP include :

-Notice error

-Fatal error

-Parse error

-Error

-E_STRICT

B) The try, catch and finally blocks are used :

-Try block : to specify a block of code that may throw an exception.

-Catch block : to handle the exception if it is thrown.

-Finally block : to execute the code after thr try and catch blocks have been executed.

C)

D) An example of custom error handling in PHP would be for example failing to read a file properly yet continuing to use would result in problems.

3) Sessions and cookies are used by sites to store a users data from a visit to a site in order to provide the user with a more tailored experience.

A) To start a PHP session you simply use the session_start() function.

B) Some common differences between sessions and cookies are :

- Cookies are used to store user information on the users computer, whilst sessions store user information on the user's server side.
- Cookies have an expiration time, whilst sessions ends only when the user closes the browser.
- Cookies store information in a text file whilst session stores data in an encrypted format.

C) You can securely store and retrieve data in cookies by using document.cookies.

D) Examples of practical uses cases for sessions and cookies include :

-Information of products customers viewed whilst online shopping is stored.

-Username can be stored once you visit a sitr, thus making logging in next time smoother.

-Make personalized content recommendations.

TEST Test test 2024-07-19 09:32:04

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 10:18:06 **Number of words : 0**

---

test test test

2024-08-03 18:47:37

LUC-INT-NGA-002-EL-APTECH-100037 Umar Michika 2024-07-19 09:59:39

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 22:23:06 **Number of words : 679**

SECTION A

1

A function can be define by starting with the *Function* keyword, followed by the functions name. It is called by writting the function name follow by a parenthesis ().

2

A) It is a popular general-purpose scripting language that is well suited for wed development and can be embedded into HTML. It cna be used to generate dynamic page content, collect form data, control user-access and encrypt data.

B) Echo has no return value while print has a return value of 1. echo can take multiple parameters while print can only take one argument.

3

It is used to store many values under a single variable, in php it is defined by writting $arrayName = array(value1, value2,....) ;.

Indexed arrays are arrays with a numeric index, Associate arrys are arrays that use named keys that you assign to them, while Multidimensional arrays are arrays that contain one or more arrays.

4

A constant is an identifier for a simple value. The value cannot be changed during the script. They are created using the *define(name, value, case-insensitve)* function syntax.

7

include or require statement take all text/code/markup that exist in the specified file and copies it intothe file that uses the include or require statement. The script will Execute even when the file included in include is not found while for require, the code stops when the file included with require is not found. Include_once or require_once is used when you what to include or require a file only once.

SECTION B

1

a) i) External Errors.

ii) Internal Errors.

b) try {

Code that may throw exceptions

} catch (Exeception $e) {

Code to handle exceptions

} finally {

Code that always executes

}

c) set_error_handler() is used to define your own way of handling errors during runtime.

set_exception_handler() is used to set the default exception handler if an exception is not caught within a try/ catch block.

d) <?php

function thiserror ($error_no, $error_msg){

echo "Error: [$error_no] $error_msg";

echo "Endind script";

die();

}

3

a) It is started with the session_start() function.

b) Sessionsare saved on the server side while cookies are saved on the client side.

c) by using the setcookie() function

d) it can be used to store usere login detains

It can be used to save the items on a users shopping cart on e-commerce websites.

It can be used to store user display preference on a website.

000111000458 Simran Simran 2024-07-19 10:30:42

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 13:06:54 **Number of words : 2062**

SECTION A

Q1.

Ans: In PHP, Functions are reusable blocks of code that perform a specific tast. they help organize cod, reduce duplication, and make development more efficient.

Uses of functions in PHP:

1.Reusable code blocks

2. Reduce duplication

3. Improve organization and readability

Defining a function:

1. function keyword

2. Function name

3. Parameters(if needed)

4. code within{}

Examples: function greet($name){

echo"Hello, $name!";

}

calling a function:

Function name

Arguments(if required)

examples: greet("Simone"),// output Hello, Simone!

Functions can return values using the "Return" statement.

Functions can be defined inside classes as methods.

Functions can be anonymous(lambda functions) or closures.

Q2.

Ans: (a) PHP (Hypertext Preprocessor)is a server-side scripting language used for web development. it's a popular languagefor building dynamic websites, web applications, and content management sustems(CMS).

PHP is primarily used for:

1. Web development
2. Server-side-scripting
3. Content management systems
4. E-commerce plateform
5. Social networking plateform

PHP's key features include:

1. Loose typology
2. Built-in-support for database
3. Extensive libraries and frameworks
4. Cross-platefrom compatibility

PHP is a versatile language language, but its primary focus is on web development and server-side scripting.

(b) In PHP, both ECHO and PRINT are used to output data, but there are some differences:

ECHO:

1. Language construct
2. Multiple arguments
3. No return value

4. Faster

Examples: echo 'Hello', ' ', 'World';

PRINT:

1.Function

2. Single arguments

3. Return 1(integer)

4. Slower

examples: Print 'Hello World';

Use ECHO for multiple outputs, and PRINT for single outputs with a return value.

Q3.

Ans: PHP handles form data through the $_Post and $_Get superglobals.

basic examples:

HTML Form:

```
<form action="process.php"
method="post">
<input type="text" name="name">
<input type="submit">
</form>
```

PHP (process. php):

```
<? php
$name=$_post['name'];
echo "Hello,$name!";
```

?>

When the form is submitted, the data is sent to process.php, where php accesses the form data through $_POST['name']and displays a greeting message.

Q4.

Ans: In PHP the **Include** and **Require** statements are used to include code from another file, while the **_Once** varients ensure the file in only included once.

Here's a brief breakdown:

include: Includes a file, warnings if not found

require: Includes a file, fatal error if not found

include_once: Include a file only once, warnings if not found.

require_once: Includes a file only once, fatal error if not found

Use **require** or **require_once** for critical files, and **include** or **include_once** for optional files.

Q5.

Ans: PHP error handling functions:;

1. **error_reporting()**: Sets the error reporting level.
2. **trigger_error()**: generates a user-defined error.
3. **set_error_handler()**: sets a custom error handler function.
4. **restore_error_handler()**: restores the default error handler.
5. **error_log()**: logs errors to a file or other destination.

Error management in php:

1. Error reporting
2. Custom Error Handlers
3. Try-Cash Blocks

4. Error Logging

5. Exception Handling

by using these functions and techniques, you can effectively manage errors in php and provide a better user experience.

SECTION B

Q1.

Ans: The lifecycle consists of the following stages:

1. Request
2. Parse
3. compile
4. Execute
5. Shutdown
6. Terminate

the lifecycle applies to a single request. in a web server context, the php process may persist to handle additional requests.

(a): During the initializing phase, PHP:

Initializes core functions and classes

Loads configuration settings(php.ini)

Sets up error handling and logging

Initializes extensions and modules

Defines superglobals(eg: $_Get, $_Post, $_SESSION)

PHP sets up its core environment and loads necessary components to prepare for scripts execution.

(b): PHP handles execution in following steps:

Interpreter: PHP code is interpreted into opcodes(machine-readable code).

Executor: Opcodes are executed by the Zend Engine(PHP's core execution engine).

Memory Management: PHP manages memory allocation and deallocation for variables, array, and objects.

FUnction call: PHP executes functions and methods, passing arguments and returning values.

Control Structures: PHP executes control structures likeif/else, loops (for, while, foreach), and switch statements.

Output: generates output, such as HTML, text, or other data, which is sent to the browser or client.

(c): During the termination phase, PHP:

Release memory allocated by variables, arrays, and objects.

Closes open file and network connections.

Destroys sessions and sessions data.

Logs any errors or exceptions.

Executes shutdown functions (registered using rergister_shutdown_function())

Returns control to the web server(or command line)

PHP cleans up resources, closes connections, and wraps up any loose ends before ending the script's execution.

(d): You can influence the PHP lifecycle with custom code using various techniques:

Auto_prepend/append_file

Register_shutdown_function

_destruct()

Exception handling

Output buffering

Custom error handling

Registering functions(globals, long arrays, tick function)

By using these techniques, you can tap into the php lifecycle and execute custom code at various stages, allowingyou to perform tasks like logging, profiling, and resource management.

Q2.

Ans: Sessions and Cookies play a crucial role in PHP for maintaining state by allowing you to store and retrieve data about a user's interactions with your application.

(a): To start managing sessions in PHP,

Fire up session_start() at the top of your script.

Check if a session already exists with isset($_SESSION).

Set session variables like $_SESSION['username']='simone'.

Access session variables like $_SESSION['username'].

Update session variables by reassigning values to $_SESSION array elements.

Destroy the session with session_destroy() when you're done.

that's how you manage the SESSIONS in PHP.

(b): Sessions and Cookies are both used for storing data in web applications, but they have some key differences:

Sessions:

stored on the server

data is kept in a server-side array($_SESSION)

Automatically destroyed after a certain time(usually 30 minutes)

Can store large amounts of data

More secure since data is not exposed to the client.

Cookies:

Stored on the client's browser

Data is stored in a text file on the client's device

Can be set to expire at a specific time or persist indefinitely

Limited in size (typically 4kB)

Less secure since data is exposed to the client and can be temperedwith

sessions are stored on the server, more secure and can store larger data, while cookies are stored on the client, less secure, and have size limitations.

(c): To secure and retrieve data in cookies:

Encrypt data using AES or similar algorithms.

Set secure and HttpOnly flags to protect cookies from javaScript access and ensure HTTPS transmission.

Store a secure hash of the data instead of the data itself.

Use a token-based approach to verify authenticity on the server-side.

Keep cookie data minimal and avoid sensitive information.

Utilize a reputed cookie library like PHP's setcookies() function.

Regenerate cookies periodically to limit exposure if compromised.

by following these steps, you can securely store and retrieve data in cookies.

(d): Here are some practical use cases for sessions and cookies:

Sessions:

Keep users logged in without requiring them to re-enter their credentials.

Srore shopping cart contents so users can checkout easily.

Remember user preferences, like language or fint size.

Help users fill out multi-step forms by temporarily storing their progress.

Let gamers pick up where they left off by storing their game progress.

Cookies:

Automatically log users back in when they rerturn to a site.

Show content in the user's preferred language.

Track user behavior to improve website performance and advertising.

Provide personalized recommendations based on user interets.

Deliver targeted ads that match user interests.

These examples illustrate how sessions and cookies enhance user experiences and provide convenience, personalization, and efficiency.

LC000111000486 Naqeeb Uddin 2024-07-19 10:40:21

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 11:03:37 **Number of words : 1761**

Section A

Answer (1):

In PHP, functions are used to encapsulate reusable code blocks that can be called multiple times within a script.

Defining a Function:
A function in PHP is defined using the function keyword followed by the function name, parentheses (which can include parameters), and a block of code enclosed in curly braces.

```
function functionName($parameter1, $parameter2) {
   // Code to be executed
   return $result;
}
```

Calling a Function:
To call a function, use its name followed by parentheses, optionally including arguments if the function requires them.

```
$result = functionName($arg1, $arg2);
```

Example:

```
// Define the function
function add($a, $b) {
   return $a + $b;
}

// Call the function
echo add(3, 4); // Outputs: 7
```

Answer (2 a):

PHP (Hypertext Preprocessor) is a widely-used, open-source scripting language designed for web development. It is primarily used for server-side scripting to create dynamic web pages, manage databases, session tracking, and building entire e-commerce sites. PHP can be embedded in HTML, making it a versatile tool for web development.

Answer (2 b):

In PHP, both echo and print are used to output data to the screen, but they have a few differences:

echo: Can output one or more strings, does not return a value, and is slightly faster.

echo "Hello, ", "World!";

print: Can only output one string, returns a value of 1, and is slower compared to echo.

print "Hello, World!";

Answer (3):

In PHP, arrays are used to store multiple values in a single variable. They can be categorized into three types:

Indexed Arrays:
Indexed arrays use numeric keys to store and access elements.

```
$indexedArray = array("Apple", "Banana", "Cherry");
echo $indexedArray[1]; // Outputs: Banana
```

Associative Arrays:
Associative arrays use named keys to store and access elements.

```
$assocArray = array("first" => "Apple", "second" => "Banana", "third" => "Cherry");
echo $assocArray["second"]; // Outputs: Banana
```

Multidimensional Arrays:
Multidimensional arrays are arrays containing one or more arrays.

```
$multiArray = array(
   array("Apple", "Banana"),
   array("Cherry", "Date")
);
echo $multiArray[1][0]; // Outputs: Cherry
```

Answer (6):

PHP handles form data using the superglobal arrays $_GET, $_POST, and $_REQUEST. These arrays store data submitted through HTML forms using the GET or POST methods.

Example of Handling Form Data:
HTML Form:

```
<!DOCTYPE html>
<html>
<body>
   <form method="post" action="process_form.php">
      Name: <input type="text" name="name">
      Age: <input type="text" name="age">
      <input type="submit">
   </form>
</body>
</html>
```

PHP Script (process_form.php):

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $name = htmlspecialchars($_POST['name']); // Sanitize input
   $age = htmlspecialchars($_POST['age']);   // Sanitize input

   echo "Name: " . $name . "<br>";
   echo "Age: " . $age;
}
?>
```

In this example:

The HTML form uses the POST method to submit data to process_form.php.
The PHP script checks if the request method is POST.
Form data is accessed using the $_POST array.
The htmlspecialchars() function is used to sanitize the input, preventing XSS attacks.
The script outputs the sanitized name and age.

Answer (8):

PHP error handling functions and techniques include:

1. Error Reporting Functions:
error_reporting(): Sets the level of error reporting.

```
error_reporting(E_ALL);  // Report all types of errors
```

ini_set('display_errors', '1'): Controls whether errors are displayed to the user.
ini_set('display_errors', '1');

2. Error Handling Functions:
set_error_handler(): Sets a custom function to handle errors.

```
function customError($errno, $errstr) {
   echo "Error: [$errno] $errstr";
}
set_error_handler("customError");
```

restore_error_handler(): Restores the previous error handler.

```
restore_error_handler();
```

3. Exception Handling:
try and catch: Catches exceptions thrown during script execution.

```
try {
   if (!file_exists("file.txt")) {
      throw new Exception("File not found");
   }
} catch (Exception $e) {
   echo 'Caught exception: ',  $e->getMessage(), "\n";
}
```

4. Custom Error Logging:
error_log(): Sends error messages to a specified log file.
error_log("Custom error message", 3, "errors.log");

Section B

Answer (1):

Error Handling in PHP
Error handling in PHP involves detecting and managing errors during script execution to ensure smooth operation and debugging.

Answer (1 a):

Types of Errors in PHP:
Parse Errors: Syntax errors that prevent the script from running.
// Missing semicolon
echo "Hello, World"

Fatal Errors: Critical errors that halt script execution, like calling undefined functions.
undefinedFunction();

Warning Errors: Non-fatal errors that allow the script to continue, such as including a non-existent file.

include 'file.php'; // File does not exist

Notice Errors: Minor errors indicating issues like accessing undefined variables.
echo $undefinedVar; // Notice: Undefined variable

Answer (1 b):

Exception Handling with try, catch, and finally:
try: Block where exceptions may occur.
catch: Block to handle exceptions thrown in the try block.
finally: Block that executes after try and catch, regardless of whether an exception was thrown.
Example:

```
try {
    $file = fopen("nonexistentfile.txt", "r");
    if (!$file) {
        throw new Exception("File not found");
    }
} catch (Exception $e) {
    echo 'Caught exception: ', $e->getMessage();
} finally {
    echo 'This is the finally block';
}
```

Answer (1 c):

Role of set_error_handler() and set_exception_handler() Functions:
set_error_handler():

Sets a custom function to handle errors.
Allows for customized error handling instead of default PHP behavior.

Example:

```
function customError($errno, $errstr) {
   echo "Error [$errno]: $errstr";
}
set_error_handler("customError");
```

set_exception_handler():

Sets a custom function to handle uncaught exceptions.
Allows for centralized exception handling.
Example:

```
function customException($exception) {
   echo "Exception: ", $exception->getMessage();
}
set_exception_handler("customException");
```

Answer (1 d):

Example of Custom Error Handling in PHP:

```
// Custom error handler function
function customError($errno, $errstr) {
   echo "Custom Error [$errno]: $errstr<br>";
}

// Custom exception handler function
function customException($exception) {
   echo "Custom Exception: ", $exception->getMessage();
}

// Set custom error and exception handlers
set_error_handler("customError");
set_exception_handler("customException");

// Trigger an error
echo $undefinedVar;  // Custom Error handling for notices
```

```
// Throw an exception
throw new Exception("This is a custom exception message");
```
In this example:

set_error_handler() is used to handle errors.
set_exception_handler() is used to handle exceptions.
Both handlers output custom messages for errors and exceptions.

Answer (2):

The lifecycle of a PHP script involves several key phases from the time the script is requested until it is completed. Here's a detailed breakdown of each phase:

Answer (2 a):

Initialization Phase:

Script Loading: PHP loads the script from the server.
Configuration: PHP settings (from php.ini or .htaccess) are applied.
Preprocessing: Server-side processing begins, including parsing the PHP code and setting up the environment.

Answer (2 b):

Execution Phase:

Compilation: PHP compiles the script into an internal bytecode format.
Execution: The compiled bytecode is executed. This includes processing all PHP statements, expressions, and function calls.
Output Generation: PHP generates the output, typically HTML, which is sent to the web server or client.

Answer (2 c):

Termination Phase:

Cleanup: Resources like open files, database connections, and memory are released.
Final Output: The final output is sent to the client.
Shutdown: PHP executes any registered shutdown functions, register_shutdown_function(), and cleans up the environment.

Answer (2 d):

Influencing the Lifecycle with Custom Code:

Custom Initialization: Use php.ini settings, auto_prepend_file, or auto_append_file for setup.
Custom Execution: Implement error handling, logging, and session management within the script.
Custom Termination: Register shutdown functions using register_shutdown_function() to perform tasks like logging or cleanup.

Example of Custom Termination:

```
function shutdownFunction() {
   echo "Script has ended. Cleanup actions here.";
}
register_shutdown_function('shutdownFunction');
```

LC000111000487 Syed Muhammad Haris Habib 2024-07-19 10:43:20

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 11:03:07 **Number of words : 1670**

Section A

Ans 1 : In PHP, functions are used to encapsulate reusable code blocks that can be called multiple times within a script.

Defining a Function:
A function in PHP is defined using the function keyword followed by the function name, parentheses (which can include parameters), and a block of code enclosed in curly braces.

```
function functionName($parameter1, $parameter2) {
   // Code to be executed
   return $result;
}
```

Calling a Function:
To call a function, use its name followed by parentheses, optionally including arguments if the function requires them.

```
$result = functionName($arg1, $arg2);
```

Example:

```
// Define the function
function add($a, $b) {
   return $a + $b;
}

// Call the function
echo add(3, 4); // Outputs: 7
```

Ans 2 a : PHP (Hypertext Preprocessor) is a widely-used, open-source scripting language designed for web development. It is primarily used for server-side scripting to create dynamic web pages, manage databases, session tracking, and building entire e-commerce sites. PHP can be embedded in HTML, making it a versatile tool for web development.

Ans 2 b : In PHP, both echo and print are used to output data to the screen, but they have a few differences:

echo: Can output one or more strings, does not return a value, and is slightly faster.

```
echo "Hello, ", "World!";
```

print: Can only output one string, returns a value of 1, and is slower compared to echo.

```
print "Hello, World!";
```

Ans 3 : In PHP, arrays are used to store multiple values in a single variable. They can be categorized into three types:

Indexed Arrays:
Indexed arrays use numeric keys to store and access elements.

```
$indexedArray = array("Apple", "Banana", "Cherry");
echo $indexedArray[1]; // Outputs: Banana
```

Associative Arrays:
Associative arrays use named keys to store and access elements.

```
$assocArray = array("first" => "Apple", "second" => "Banana", "third" => "Cherry");
echo $assocArray["second"]; // Outputs: Banana
```

Multidimensional Arrays:
Multidimensional arrays are arrays containing one or more arrays.

```
$multiArray = array(
    array("Apple", "Banana"),
    array("Cherry", "Date")
);
echo $multiArray[1][0]; // Outputs: Cherry
```

Ans 6 : PHP handles form data using the superglobal arrays $_GET, $_POST, and $_REQUEST. These arrays store data submitted through HTML forms using the GET or POST methods.

Example of Handling Form Data:
HTML Form:

```
<!DOCTYPE html>
<html>
<body>
   <form method="post" action="process_form.php">
      Name: <input type="text" name="name">
      Age: <input type="text" name="age">
      <input type="submit">
   </form>
</body>
</html>
```

PHP Script (process_form.php):

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $name = htmlspecialchars($_POST['name']); // Sanitize input
   $age = htmlspecialchars($_POST['age']);   // Sanitize input

   echo "Name: " . $name . "<br>";
   echo "Age: " . $age;
}
?>
```

In this example:

1. The HTML form uses the POST method to submit data to process_form.php.

2. The PHP script checks if the request method is POST.

3. Form data is accessed using the $_POST array.

4. The htmlspecialchars() function is used to sanitize the input, preventing XSS attacks.

5. The script outputs the sanitized name and age.

Ans 8 : **PHP error handling functions and techniques include:**

1. Error Reporting Functions:

error_reporting(): Sets the level of error reporting.
error_reporting(E_ALL); // Report all types of errors

ini_set('display_errors', '1'): Controls whether errors are displayed to the user.
ini_set('display_errors', '1');

2. Error Handling Functions:

set_error_handler(): Sets a custom function to handle errors.

```
function customError($errno, $errstr) {
   echo "Error: [$errno] $errstr";
}
set_error_handler("customError");
```

restore_error_handler(): Restores the previous error handler.

```
restore_error_handler();
```

3. Exception Handling:

try and catch: Catches exceptions thrown during script execution.

```
try {
   if (!file_exists("file.txt")) {
      throw new Exception("File not found");
   }
} catch (Exception $e) {
   echo 'Caught exception: ',  $e->getMessage(), "\n";
}
```

4. Custom Error Logging:

error_log(): Sends error messages to a specified log file.
error_log("Custom error message", 3, "errors.log");

Section B

Ans 1 a : **Error Handling in PHP:**
Error handling in PHP involves detecting and managing errors during script execution to ensure smooth operation and debugging.

a. Types of Errors in PHP:

Parse Errors: Syntax errors that prevent the script from running.

```
// Missing semicolon
echo "Hello, World"
```

Fatal Errors: Critical errors that halt script execution, like calling undefined functions.

```
undefinedFunction();
```

Warning Errors: Non-fatal errors that allow the script to continue, such as including a non-existent file.

```
include 'file.php';  // File does not exist
```

Notice Errors: Minor errors indicating issues like accessing undefined variables.

```
echo $undefinedVar;  // Notice: Undefined variable
```

Ans 1 b : **Exception Handling with try, catch, and finally:**

try: Block where exceptions may occur.
catch: Block to handle exceptions thrown in the try block.
finally: Block that executes after try and catch, regardless of whether an exception was thrown.

Example:

```
try {
    $file = fopen("nonexistentfile.txt", "r");
    if (!$file) {
        throw new Exception("File not found");
    }
} catch (Exception $e) {
    echo 'Caught exception: ', $e->getMessage();
} finally {
    echo 'This is the finally block';
}
```

Ans 1 c :

Role of set_error_handler() and set_exception_handler() Functions:

set_error_handler():

Sets a custom function to handle errors.
Allows for customized error handling instead of default PHP behavior.

Example:

```
function customError($errno, $errstr) {
   echo "Error [$errno]: $errstr";
}
set_error_handler("customError");
```

set_exception_handler():

Sets a custom function to handle uncaught exceptions.
Allows for centralized exception handling.

Example:

```
function customException($exception) {
   echo "Exception: ", $exception->getMessage();
}
set_exception_handler("customException");
```

Ans 1 d : **Example of Custom Error Handling in PHP:**

```
// Custom error handler function
function customError($errno, $errstr) {
   echo "Custom Error [$errno]: $errstr<br>";
}

// Custom exception handler function
function customException($exception) {
   echo "Custom Exception: ", $exception->getMessage();
```

```
}

// Set custom error and exception handlers
set_error_handler("customError");
set_exception_handler("customException");

// Trigger an error
echo $undefinedVar;  // Custom Error handling for notices

// Throw an exception
throw new Exception("This is a custom exception message");
```
In this example:

set_error_handler() is used to handle errors.
set_exception_handler() is used to handle exceptions.
Both handlers output custom messages for errors and exceptions.

Ans 2 a : **Initialization Phase:**

Script Loading: PHP loads the script from the server.
Configuration: PHP settings (from php.ini or .htaccess) are applied.
Preprocessing: Server-side processing begins, including parsing the PHP code and setting up the environment.

Ans 2 b : **Execution Phase:**

Compilation: PHP compiles the script into an internal bytecode format.

Execution: The compiled bytecode is executed. This includes processing all PHP statements, expressions, and function calls.

Output Generation: PHP generates the output, typically HTML, which is sent to the web server or client.

Ans 2 c :

Termination Phase:

Cleanup: Resources like open files, database connections, and memory are released.

Final Output: The final output is sent to the client.

Shutdown: PHP executes any registered shutdown functions, register_shutdown_function(), and cleans up the environment.

Ans 2 d :

Influencing the Lifecycle with Custom Code:

Custom Initialization: Use php.ini settings, auto_prepend_file, or auto_append_file for setup.

Custom Execution: Implement error handling, logging, and session management within the script.

Custom Termination: Register shutdown functions using register_shutdown_function() to perform tasks like logging or cleanup.

Example of Custom Termination:

```
function shutdownFunction() {
    echo "Script has ended. Cleanup actions here.";
}


register_shutdown_function('shutdownFunction');
```

LC000111000489 Sana khan 2024-07-19 13:00:58

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 15:05:54 **Number of words : 795**

ANS. 1 :They enhance code organisation ,promote reusability ,and improve readability by encapsulation logic into named units .Declaring and calling functions in PHP involves defining the function using the function keyword,specifying any parameters it may accept,and then invoking the function when needed within the script.

ANS 2: A:

PHP is an open source,server side programming language that can be used to create websites ,applications ,customer relationship management systems and more.It is a widely used general purpose language that can be embadded into HTML.

ANS 2: B:

They are both used to output data to the screen .The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument .echo is marginally faster than print.

ANS 6:

When the user fills out the form above and clicks the submit button,the form data is sent for processing to a PHP file named "welcome .php"The form data is sent for processing to a PHP file named "welcome .php".The form data is data is sent with the HTTP POST method .To display the submitted data you could simply echo all the variables.

ANS 7:

Include includes and evaluates a specified file,and only emits a warning if the file cannot be included .require once works like require ,but it will only include the file if it has not been included before include once works like include ,but it will only include the file if it has not been include before.

ANS 8:

The default error handler for PHPis the built in error handler.We are going to make the function above the default error handler for the duration of the script .It is possible to change the error handler to apply for only some errors ,that way the script can handle different errors in different ways.

SECTION B:

ANS :1 (A) PHP error and exception handling in PHP are powerful mechanisms.While exception handling helps change the usual flow of the code execution if a specified error occurs,procedural error handling in PHP detects errors raised by your program and then takes the necessary course of action .

(B)The code in the try block is executed first ,and if throws an exception ,the code in the catch block will be executed .The code in the finally block will always be executed before control flow exits the entire construct.

(c)The set error handler ()function sets a user defined error handler function .Note :The standard PHP error handler is completely bypassed if this function is used, and the user defined error handler must terminate the script ,die(),necessary .

(d)However,in this example we are going to use our custom error handler (custom error):since we want our custom function to handle all errors ,the set error handler ()only needed one parameter could be added to specify an error level.

ANS 2:(A)

The PHP script life cycle refers to the sequence of events that occur when a PHP script is executed .The life cycle consists of several stages,including parsing ,compilation ,execution and clean up .

(A)The initializing phase marks the beginning of a new project .During this stage ,the organisation identifies the objectives ,scope,purpose ,and deliverables to be produced .It then obtain the authorzation to do the actual work.

(B)The PHP interpreter will read the PHP file, parse it (and other included files)and then execute it.Once the PHP file,it will take that output and send it back as a response on browser.

(C)The termination phase of the nurse client relationship comes at the end,and durring this phase you will summarize goals that were achieved during the relationship,discuss incorporation of new coping mechanisms and problem solving skills into the patients life ,and discuss their discharge plans.

(D)Custom code extend the standard SAP software .They implement customer specific business requirements and fill functional gaps by using custom Development ,Modification and Enhancements.

The custom code lifecycle management (CCLM)application is the central location where customers can manage their collection of custom code all the way from creation to clearing of unused custom code object across the entire SAP landscape .Once identified,the custom code objects in the landscape can be managed more effectively because you can document ownership ,version,data quality ,and usage patterns .

LC000111000488 Waseem Ahmed 2024-07-19 13:21:47

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 14:53:20 **Number of words : 1818**

Section A:

Q1.

Ans: In PHP, functions are used to group a set of statements that perform a specific task. To define a function, you use the function keyword followed by the function name and parentheses.

For example:

function sayHello() { echo "Hello, World!"; }

sayHello();

Q4.

Ans: PHP constants are variables whose values cannot be changed once defined. They are useful for storing data that shouldn't be altered during the execution of the script, like configuration settings.To define a constant, use the define() function, specifying the name of the constant and its value. Constants are typically written in uppercase letters by convention.

For Example:

define("SITE_NAME", "MyWebsite");

echo SITE_NAME;

Q8.

Ans: PHP error handling functions help manage and respond to errors that occur during script execution. Some common error handling functions include:

1) error_reporting(): Sets the error reporting level.
2) set_error_handler(): Sets a custom error handler function.
3) trigger_error(): Generates a user-level error.
4) error_log(): Sends an error message to a log file or a remote server.

Managing Errors in PHP:

1) Using error_reporting(): Control which errors are reported.
```
error_reporting(E_ALL); // Report all errors
```

2) Custom Error Handler: Define a function to handle errors.
```
function customError($errno, $errstr) {
    echo "Error: [$errno] $errstr";
}
set_error_handler("customError");
```

3) Triggering Errors: Generate an error manually.
```
if ($age < 18) {
    trigger_error("Age must be 18 or older", E_USER_WARNING);
}
```

4) Logging Errors: Write errors to a log file.
```
error_log("Error message", 3, "/var/log/php_errors.log");
```

By using these functions, you can handle errors gracefully, log them for later analysis, and display user-friendly error messages.

Q6.

Ans: PHP handles form data using the "$_GET" and "$_POST" superglobal arrays, depending on the method used to submit the form.

Basic Example:

1. HTML Form:
```
<form action="process.php" method="post">
    Name: <input type="text" name="name">
    Email: <input type="email" name="email">
    <input type="submit" value="Submit">
</form>
```

2. PHP Script (process.php):

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = $_POST['name'];
    $email = $_POST['email'];

    echo "Name: " . $name . "<br>";
    echo "Email: " . $email;
}
```

Q7.

Ans: In PHP, "include", "require", "include_once", and "require_once" are used to insert the content of one PHP file into another.

Here are the differences:

include: Includes a file, and script continues on error.
require: Includes a file, and script stops on error.
include_once: Includes a file once, and script continues on error.
require_once: Includes a file once, and script stops on error.

Section B:

Q2.

Ans: Lifecycle of a PHP Script:

The lifecycle of a PHP script can be broken down into three main phases: initialization, execution, and termination.

Here's a detailed look at each phase:

a. Initialization Phase

During the initialization phase, PHP prepares to execute the script:

Configuration Loading: PHP reads configuration settings from php.ini and environment variables.

Module Initialization: PHP extensions and modules are loaded and initialized.

Superglobals Setup: PHP sets up superglobal arrays like $_GET, $_POST, $_SESSION, $_COOKIE, $_FILES, etc.

Server-Specific Initialization: Server-specific variables and settings are established.

b. Execution Phase

During the execution phase, PHP processes the script line by line:

Parsing and Compiling: The PHP script is parsed into tokens and compiled into bytecode.

Execution: The compiled bytecode is executed by the Zend Engine.

This involves:

Evaluating expressions and executing statements.

Handling control structures (loops, conditionals).

Interacting with databases, files, and other resources.

Sending output to the client (HTML, JSON, etc.).

c. Termination Phase

During the termination phase, PHP cleans up and finalizes the script execution:

Output Finalization: Any remaining output buffers are flushed and sent to the client.

Resource Cleanup: PHP releases resources, such as database connections and file handles.

Session Management: PHP writes session data (if any) and closes the session.

Shutdown Functions: Any registered shutdown functions are executed.

Memory Cleanup: PHP frees allocated memory and performs garbage collection.

d. Influencing the Lifecycle with Custom Code

You can influence the PHP script lifecycle with custom code:

Initialization:

Use configuration files (php.ini, .htaccess, etc.) to set up the environment.

Use auto_prepend_file in php.ini to automatically include a file before the script execution.

Execution:

Utilize require, include, require_once, and include_once to manage code modularity.

Implement custom error handlers with set_error_handler().

Use output buffering with ob_start() and ob_end_flush() to control output flow.

Termination:
Register shutdown functions using register_shutdown_function().
Handle session data with session_start() and session_write_close().
Perform cleanup tasks within the registered shutdown functions.

Q3.

Ans: Role of Sessions and Cookies in PHP:

Sessions and cookies are both used to maintain state and store user-specific information across multiple page requests. They help create a seamless user experience by remembering user data.

a. Starting and Managing a Session in PHP:

1) **Starting a Session**: Use "session_start()" at the beginning of your script to initialize a session.

2) **Storing Data in a Session**: Store data using the "$_SESSION" superglobal array.

3) **Retrieving Data from a Session**: Access session data via the "$_SESSION" array.

4) **Ending a Session**: Use "session_destroy()" to remove all session data and end the session.

b. Differences Between Sessions and Cookies:

Storage Location:
Sessions: Store data on the server, with a session ID sent to the client via a cookie or URL.
Cookies: Store data on the client's browser.

Security:
Sessions: Generally more secure because the data is stored on the server.
Cookies: Less secure as data is stored on the client side and can be tampered with.

Data Lifespan:
Sessions: Typically last until the browser is closed or the session is explicitly destroyed.
Cookies: Can be set to expire at a specific date and time or last for a set period.

Size Limitations:
Sessions: Not restricted by size, limited by server storage.
Cookies: Limited to about 4 KB per cookie.

c. Securely Storing and Retrieving Data in Cookies:

1) **Setting a Cookie**: Use "setcookie()" with parameters for security.

Parameters:
"name": Cookie name.
"value": Cookie value.
"expire": Expiration time (in seconds).
"path": Path on the server where the cookie is available.
"domain": Domain for which the cookie is valid.
"secure": "true" for HTTPS only.
"httponly": "true" to make the cookie accessible only through HTTP, not JavaScript.

2) **Retrieving a Cookie**: Access cookie values using the "$_COOKIE" superglobal array.

3) **Deleting a Cookie**: Set the cookie's expiration time to a past date.

d. Practical Use Cases for Sessions and Cookies:

Sessions:
User Authentication: Store user login information and manage user access across pages.
Shopping Carts: Track items added to a shopping cart during a single session.
User Preferences: Save user settings or preferences that need to be accessed across multiple pages during the session.

Cookies:
Remember Me: Store a persistent login state or user preferences that remain across sessions.
Tracking and Analytics: Collect data on user behavior or preferences for analytics purposes.
Language and Theme Selection: Save the user's choice of language or theme for future visits.

By using sessions and cookies appropriately, you can enhance user experience and maintain state across web interactions.

LC000111000459 Noman Babar 2024-07-19 21:11:15

Teacher :

Exam Title/Code : PHP EXAM

Submitted on : 2024-07-19 22:32:55 **Number of words : 1278**

Section-B

Ans-1

Function in PHP

In PHP, function are reuseable blocks of codethat perform specific tasks. They help to organize your code, inprove readability, and redundency. Here is how you define and call functions:

Defining a Function:

1- function Name: start with the function keyboard.

2- Function Name: chose a meaningfull name that reflects the functions purpose (e.g., calculateArea)

3- Parameters (Optional): enclose any parameters (inputs) the function accepts within parentheses. These are like placeholders for values you will provide when calling the function.

Ans-3

Arrays in PHP

Arrays store collections of data under one name in PHP

Indexed Arrays: Ordered lists accessed by numeric indexes (O-based).

Asociative Arrays: Unordered collections accessed by unique string keys (like a dictionary)

Multidimensional Arrays: Arrays that hold other arrays as elements (useful for complex data)

Ans-4

PHP constants

PHP constant are fixed values that cannot change during script execution (except for special magic constants). they:

Improve code readability and maintainability by using meaningful names instead of hardcoded values.

Help prevent accidental modification of critical data

Ans-6

PHP handle foram data

PHP handles form data with superglobals:

$_POST: store data sent through the POST method (better for sensitive info)

$_GET:stores data through the URL (less secure).

Example:

form sends data (username, email) to process.php using POST.

process.php retrieves data from $_POST and uses it (e.g., greetings

Ans-8

PHP erros

PHP offers functions to manage errors:

Catch errors: set_error_hadler lets you define a funstion to handle errors.

Get errors info: error_get_last provides details about the last error.

Control reported errors: allows filtering which errors are shown.

Error Management:

Report all errors during development (e.g., error_reporting(E_ALL))

Define a custom error handle (optisnsl for centralized handling (logging, messeges, etc)

use exceptions (recommended) for structured error handling with try...catch blocks.

Section-B

Ans-1

Error handling in PHP

error handling in php is crucial for writing robust and user-friendly applications. it allows you to fracefully manage errors that might occur during script execution, prevented unexpected crashes or incorrect behavior.

A-types of errors in php:

there are two main categories of errors in php:

fatal errors: these cause the script to halt immediately. examples include syntax errors, parse errors, and fatal memory errors.

non-fatal errors: these do not necessarily halt the script but might produce unexpected results or warnings. examples include warnings, motices, and user defined exceptions

B-exception handling with:

try...catch...finally

exceptions provide a structured way to handle errors inphp. they are objects that represents exceptional circumstances that might arise during program execution. you can use try..catch...finally blocks to manage exceptions:

php

try{

$result = 10/ 0;

echo $result;

} catch (divisionbyzeroerror $e) {

echo "Error : division by zero!"'

} finally{

echo "/nCleanup tasks (if necessary)";

c. set-error-handler()and set-exception-handler()

set-error-handler() : this function allows you to define a function to hadle php errors (both fatal and non-fatal) during script execution.

d. custom error handling example

here is an example of how you might use set-error-handler() for custom error handling:

php

function customerrorhandling ($errno, $errstr, $errfile, $errline){

error-log("[$errno] $errstr in $errfile on line $errline". 1. "/path/to/error.log");

echo"an error uccured =, pleasr try again later.";

return false;

}

Ans-2

lifecycle of a PHP script

the lifecycle of a php script

a PHP script goes through a well-defined lifecycle, from startup to termination. understanding these phases is essential for writing efficient and robust php application.

a.initialization phase(MINIT):

occurs when the php engines starts up, either as a standalone process (CLI) or as a module within the web server (e.g., apache).

b.script execution (RINIT,SCRIPT,RSHOTDOWN):

when a request arrives(e.g., user visits a webpage):

the web server(e.g., apche) parses the request and identifies the php script to be executed.\c. termination phase(MSHUTDOWN):

when the process is no longer needed or reaches a predefined shutdown condition:

the madule phase (MSHUTDOWN) sxecutes.

d.influencing the lifecycle with custom code:

you cant modify the core lifecycle phases, but you can influence them with custom code:

initiallization: use ini_set() or a custom startup script to configure specific settings at startup.

script exeution:

implement custom functions or error handlers that execute during script execution.