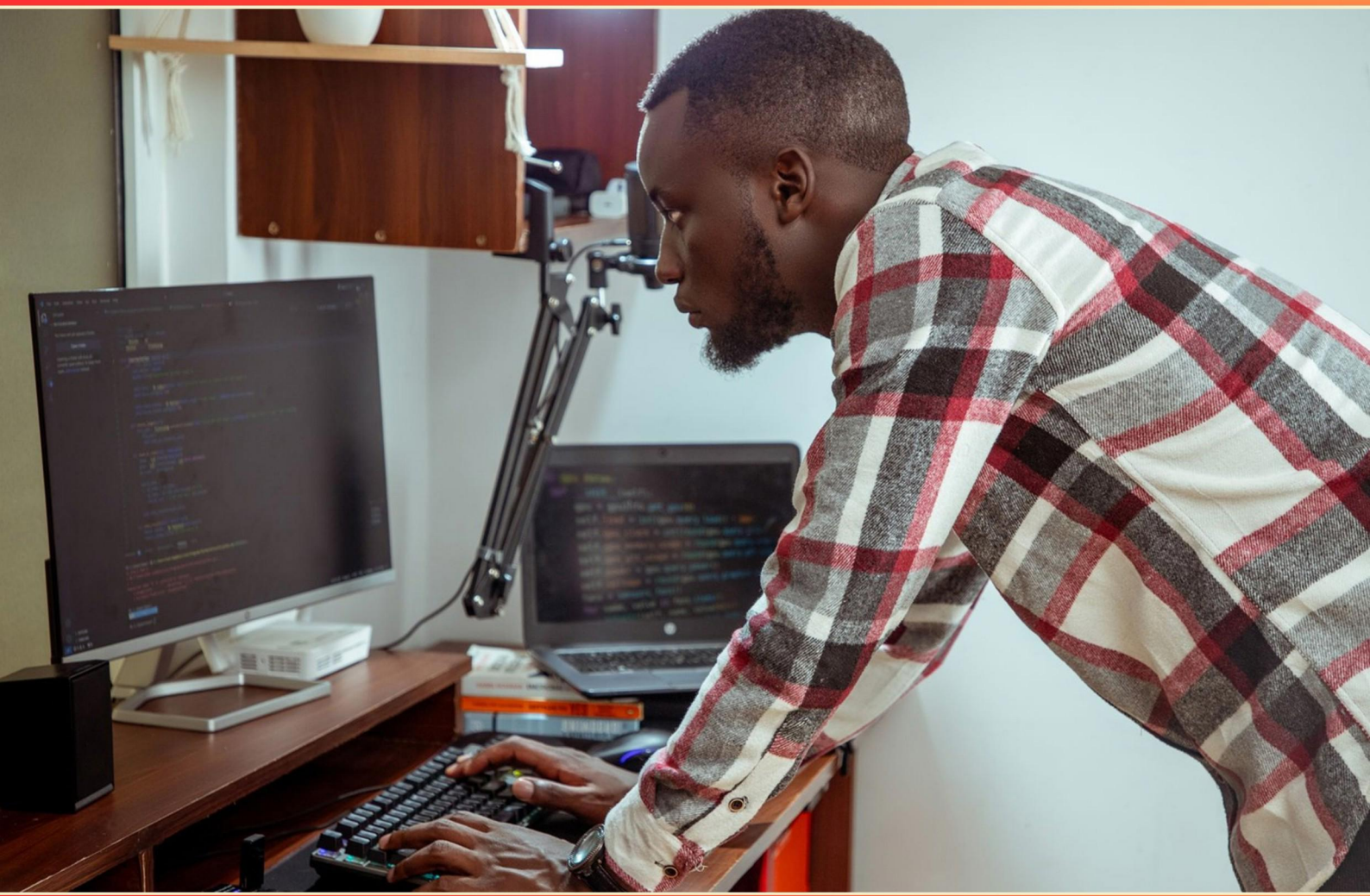
ZEND CERTIFIED PHP ENGINEER (ZCPE) PRACTICE TEST (SAMPLE)

STUDY GUIDE



SAMPLE STUDY GUIDE WITH LIMITED QUESTIONS

VISIT US ONLINE FOR THE FULL VERSION STUDY GUIDE

https://zendcertifiedphpengineer.examzify.com

Copyright © 2026 by Examzify - A Kaluba Technologies Inc. product.

ALL RIGHTS RESERVED.

No part of this book may be reproduced or transferred in any form or by any means, graphic, electronic, or mechanical, including photocopying, recording, web distribution, taping, or by any information storage retrieval system, without the written permission of the author.

Notice: Examzify makes every reasonable effort to obtain accurate, complete, and timely information about this product from reliable sources.

SAMPLE

Table of Contents

Copyright 1

Table of Contents 2

Introduction 3

How to Use This Guide 4

Questions 5

Answers 8

Explanations 10

Next Steps 15

SAMPLE

Introduction

Preparing for a certification exam can feel overwhelming, but with the right tools, it becomes an opportunity to build confidence, sharpen your skills, and move one step closer to your goals. At Examzify, we believe that effective exam preparation isn't just about memorization, it's about understanding the material, identifying knowledge gaps, and building the test-taking strategies that lead to success.

This guide was designed to help you do exactly that.

Whether you're preparing for a licensing exam, professional certification, or entry-level qualification, this book offers structured practice to reinforce key concepts. You'll find a wide range of multiple-choice questions, each followed by clear explanations to help you understand not just the right answer, but why it's correct.

The content in this guide is based on real-world exam objectives and aligned with the types of questions and topics commonly found on official tests. It's ideal for learners who want to:

- Practice answering questions under realistic conditions,
- Improve accuracy and speed,
- Review explanations to strengthen weak areas, and
- Approach the exam with greater confidence.

We recommend using this book not as a stand-alone study tool, but alongside other resources like flashcards, textbooks, or hands-on training. For best results, we recommend working through each question, reflecting on the explanation provided, and revisiting the topics that challenge you most.

Remember: successful test preparation isn't about getting every question right the first time, it's about learning from your mistakes and improving over time. Stay focused, trust the process, and know that every page you turn brings you closer to success.

Let's begin.

SAMPLE

How to Use This Guide

This guide is designed to help you study more effectively and approach your exam with confidence. Whether you're reviewing for the first time or doing a final refresh, here's how to get the most out of your Examzify study guide:

1. Start with a Diagnostic Review

Skim through the questions to get a sense of what you know and what you need to focus on. Your goal is to identify knowledge gaps early.

2. Study in Short, Focused Sessions

Break your study time into manageable blocks (e.g. 30 – 45 minutes). Review a handful of questions, reflect on the explanations.

3. Learn from the Explanations

After answering a question, always read the explanation, even if you got it right. It reinforces key points, corrects misunderstandings, and teaches subtle distinctions between similar answers.

4. Track Your Progress

Use bookmarks or notes (if reading digitally) to mark difficult questions. Revisit these regularly and track improvements over time.

5. Simulate the Real Exam

Once you're comfortable, try taking a full set of questions without pausing. Set a timer and simulate test-day conditions to build confidence and time management skills.

6. Repeat and Review

Don't just study once, repetition builds retention. Re-attempt questions after a few days and revisit explanations to reinforce learning. Pair this guide with other Examzify tools like flashcards, and digital practice tests to strengthen your preparation across formats.

There's no single right way to study, but consistent, thoughtful effort always wins. Use this guide flexibly, adapt the tips above to fit your pace and learning style. You've got this!

SAMPLE

Questions

SAMPLE

1. What is PDO in PHP?

A. PHP Data Optimization
B. PHP Data Objects
C. Persistent Data Operations
D. PHP Direct Output

2. Which construct is not part of exception handling in PHP?

A. try
B. catch
C. throw
D. define

3. Which constant returns the trait name including namespace if defined?

A. __TRAIT__
B. get_trait_name()
C. trait_name()
D. current_trait()

4. What does PDO stand for in PHP?

A. PHP Document Object
B. PHP Data Objects
C. PHP Dynamic Output
D. PHP Database Operations

5. Which PHP function can be used to convert a string to an array?

A. implode()
B. split()
C. explode()
D. array_split()

6. How can you encrypt data in PHP?

A. Using the openssl_encrypt() function
B. Using the encrypt_data() function
C. Using the secure_encrypt() function
D. Using the crypt_data() function

SAMPLE

7. How do you declare an array in PHP?

A. By using the create_array() function

B. By using the array() function or shorthand syntax []

C. By using the list() function

D. By using the new Array() syntax

8. How can you retrieve the IP address of a user in PHP?

A. Using $_GET['ip']

B. Using $_SERVER['REMOTE_ADDR']

C. Using $_COOKIE['ip']

D. Using $_POST['ip']

9. How do you handle exceptions in PHP?

A. Using if-else statements

B. Using try, catch, and finally blocks

C. Using error_reporting()

D. Using exception_error()

10. Why might a developer create sub-namespaces in PHP?

A. To increase the size of the library

B. To improve code readability and organization

C. To increase the number of classes

D. To make variable names shorter

SAMPLE

Answers

SAMPLE

1. B
2. D
3. A
4. B
5. C
6. A
7. B
8. B
9. B
10. B

SAMPLE

Explanations

SAMPLE

1. What is PDO in PHP?

A. PHP Data Optimization

B. PHP Data Objects

C. Persistent Data Operations

D. PHP Direct Output

PDO stands for PHP Data Objects, which is a robust database access layer included in PHP. It provides a uniform method of accessing multiple databases using the same functions, which simplifies database interaction and enhances code portability. PDO supports several database management systems, including MySQL, PostgreSQL, SQLite, and others, making it versatile for developers working on different projects. One of the key features of PDO is its support for prepared statements, which help protect against SQL injection attacks by separating SQL logic from data. This is crucial for developing secure applications. Additionally, PDO allows for error handling through exceptions, which provides more control over how errors are managed during database interactions. The other options, such as PHP Data Optimization, Persistent Data Operations, and PHP Direct Output, do not accurately reflect the purpose or functionality of PDO, focusing instead on misleading or unrelated concepts that are not part of PHP's database handling features.

2. Which construct is not part of exception handling in PHP?

A. try

B. catch

C. throw

D. define

The construct that is not part of exception handling in PHP is "define." In PHP, exception handling is primarily managed using the constructs try, catch, and throw. The try block is used to wrap code that might throw an exception. If an exception occurs within this block, PHP checks for a corresponding catch block that can handle the exception. The catch block allows you to take specific actions based on the type of exception that was thrown. The throw statement is used to trigger an exception, allowing you to create custom error conditions within your code. In contrast, define is a construct used for defining constants in PHP, which serves a different purpose unrelated to error handling or exceptions. Since it does not play any role in managing or responding to exceptions, it is correctly identified as the answer to this question.

3. Which constant returns the trait name including namespace if defined?

A. __TRAIT__

B. get_trait_name()

C. trait_name()

D. current_trait()

The reason the constant that returns the trait name including the namespace when defined is __TRAIT__. This magic constant is automatically populated with the fully qualified name of the trait that is being used within a class. This feature is particularly useful for debugging, autoloading, or when you need to programmatically access trait information. When you use __TRAIT__ within the context of the trait itself, it will output the trait's name, including any associated namespace. This helps maintain clarity in applications using multiple traits, especially when they may share names across different namespaces. The other options do not provide this functionality. For instance, get_trait_name() and trait_name() are not recognized as valid built-in PHP functions or constants. Similarly, current_trait() does not exist in PHP's functionality. These alternatives would not yield the desired results, underscoring the unique utility of __TRAIT__ in identifying trait namespaces.

SAMPLE

4. What does PDO stand for in PHP?

A. PHP Document Object

B. PHP Data Objects

C. PHP Dynamic Output

D. PHP Database Operations

The term PDO stands for PHP Data Objects, which is a database access layer providing a uniform interface for accessing multiple types of databases in PHP. It allows developers to perform database operations in a secure and efficient manner by offering prepared statements, which help mitigate SQL injection vulnerabilities. PDO is designed to support different database management systems (DBMSs), including MySQL, PostgreSQL, and SQLite, among others, without requiring changes to the PHP codebase. This abstraction makes it easier to switch between different databases if necessary while maintaining the same code structure, promoting code reusability and flexibility. Each of the other options does not accurately reflect the purpose or functionality of PDO. 'PHP Document Object' suggests a focus on document structure, which is not related to database interactions. 'PHP Dynamic Output' implies a feature oriented towards output generation rather than database handling, and 'PHP Database Operations' is somewhat close but lacks the specificity and definition provided by "Data Objects." Thus, identifying PDO as PHP Data Objects is essential for understanding its role in PHP database interaction.

5. Which PHP function can be used to convert a string to an array?

A. implode()

B. split()

C. explode()

D. array_split()

The function that is used to convert a string to an array is `explode()`. This function takes a delimiter and a string as input, and it splits the string at each occurrence of the delimiter. The result is an array of strings, with each element corresponding to a segment of the original string that was separated by the delimiter. For example, if you have a string such as "one,two,three" and you want to convert it into an array of words, you would call `explode(',', 'one,two,three')`. The output would be an array containing ["one", "two", "three"]. The other options do not serve the purpose of converting a string to an array in the same manner. `implode()` does the reverse of what `explode()` does; it takes an array and joins its elements into a single string using a specified delimiter. The function `split()` is actually deprecated as of PHP 5.3.0 and was used to split strings based on a regular expression, but is not typically used for simple delimiter-based string conversions. `array_split()` is not a built-in PHP function and does not exist in the PHP language, making it unrelated to the task of converting strings to arrays. Understanding

SAMPLE

6. How can you encrypt data in PHP?

A. Using the openssl_encrypt() function

B. Using the encrypt_data() function

C. Using the secure_encrypt() function

D. Using the crypt_data() function

The function openssl_encrypt() in PHP is used to encrypt data using a variety of encryption algorithms and modes provided by the OpenSSL library. This function supports AES, DES, Triple DES, and more, enabling developers to secure sensitive information by converting it into a format that cannot be easily understood without decryption. To use openssl_encrypt(), you need to specify the plaintext input, the encryption method, the secret key, and optionally an initialization vector (IV) for certain encryption modes. This versatility and reliance on the well-established OpenSSL library make it a robust choice for data encryption in PHP applications. The other options listed refer to functions that do not exist in the PHP standard library. Therefore, they cannot be used to encrypt data in PHP, highlighting the importance of correct function names and libraries when dealing with encryption.

7. How do you declare an array in PHP?

A. By using the create_array() function

B. By using the array() function or shorthand syntax []

C. By using the list() function

D. By using the new Array() syntax

In PHP, declaring an array can be accomplished using the array() function or the shorthand syntax, which is the square brackets []. This syntax has been available since PHP 5.4 and offers a more concise way to create arrays. Using the array() function looks like this: ```php $myArray = array("value1", "value2", "value3"); ``` With the shorthand syntax, it can be written as: ```php $myArray = ["value1", "value2", "value3"]; ``` Both methods effectively create an array with the specified values. The shorthand syntax is now preferred for its brevity and clarity. The other options do not represent valid array declaration techniques in PHP. The create_array() function is not a built-in PHP function, the list() function is used for assigning values from an array to variables rather than creating an array, and new Array() syntax is not valid in PHP for declaring arrays; instead, it derives from JavaScript, which can lead to confusion. Thus, utilizing either the array() function or the shorthand syntax is the correct approach to declare an array in PHP.

8. How can you retrieve the IP address of a user in PHP?

A. Using $_GET['ip']

B. Using $_SERVER['REMOTE_ADDR']

C. Using $_COOKIE['ip']

D. Using $_POST['ip']

To retrieve the IP address of a user in PHP, utilizing the `$_SERVER['REMOTE_ADDR']` superglobal is the correct approach. This variable contains the IP address from which the user is viewing the current page. It captures the address provided by the web server and is the standard method for obtaining client IP information in PHP. This method is reliable because `$_SERVER['REMOTE_ADDR']` is set by the server environment during the user's request and reflects the direct connection point. It works effectively in most scenarios, including when running on standard shared hosting environments. Other choices, while they refer to ways data might be accessed in PHP, do not accurately capture the user's IP address. For instance, the `$_GET`, `$_POST`, and `$_COOKIE` arrays are used for capturing data sent to the server via URL parameters, form submissions, or cookie data, respectively, but they do not represent the user's IP address directly. Thus, they are not suitable for this specific requirement.

SAMPLE

9. How do you handle exceptions in PHP?

A. Using if-else statements

B. Using try, catch, and finally blocks

C. Using error_reporting()

D. Using exception_error()

Using try, catch, and finally blocks is the established way to handle exceptions in PHP, allowing for structured and effective management of error conditions. This approach enables developers to separate normal program flow from error handling, enhancing code clarity and maintainability. When an exception is thrown in PHP, execution can be transferred to a catch block where the exception can be processed, logged, or handled appropriately. The try block contains code that may throw an exception, and if an exception occurs, the script stops executing further code within that block and jumps to the associated catch block, which can deal with the exception. The finally block, if used, will execute regardless of whether an exception was thrown or caught, making it suitable for releasing resources or performing cleanup tasks. This structured error handling is crucial in building robust applications, especially when dealing with unpredictable runtime errors that can disrupt normal execution. In contrast, other options such as using if-else statements or error reporting tools do not provide this level of control or organization over error management, leading to less effective handling of exceptions.

10. Why might a developer create sub-namespaces in PHP?

A. To increase the size of the library

B. To improve code readability and organization

C. To increase the number of classes

D. To make variable names shorter

Creating sub-namespaces in PHP serves the primary purpose of enhancing code readability and organization. Namespaces are used to group related classes, interfaces, functions, and constants together, effectively preventing name collisions and making it clear to users what each component pertains to. By establishing sub-namespaces, developers can create a more structured hierarchy that reflects the architecture of their application. For instance, if an application has multiple modules or components, each can be encapsulated within its own sub-namespace, allowing developers to easily locate and manage their code. This organization aids in maintaining clarity, especially as projects scale and the number of classes grows. Moreover, when using sub-namespaces, it becomes simpler to understand the relationships between different parts of the application, which leads to improved collaboration among team members. This organizational strategy ultimately leads to a more maintainable codebase, as it establishes clear boundaries and responsibilities for each part of the application. Therefore, the use of sub-namespaces is an effective practice for structuring PHP applications in a clear and manageable way.

SAMPLE

Next Steps

Congratulations on reaching the final section of this guide. You've taken a meaningful step toward passing your certification exam and advancing your career.

As you continue preparing, remember that consistent practice, review, and self-reflection are key to success. Make time to revisit difficult topics, simulate exam conditions, and track your progress along the way.

If you need help, have suggestions, or want to share feedback, we'd love to hear from you. Reach out to our team at hello@examzify.com.

Or visit your dedicated course page for more study tools and resources:

https://zendcertifiedphpengineer.examzify.com

We wish you the very best on your exam journey. You've got this!

SAMPLE