PHP.ini Settings

Overview

The `php.ini` file is the configuration file for PHP. It controls various settings, including error reporting, file uploads, memory limits, and more.

Key Settings

1. **Error Reporting**:
   - `error_reporting` : Sets which errors are reported.
   - `display_errors` : Controls whether errors are displayed on the screen.
   - `log_errors` : Enables or disables logging errors to a file.
   - `error_log` : Specifies the file where errors are logged.
2. **File Uploads**:
   - `file_uploads` : Enables or disables file uploads.
   - `upload_max_filesize` : Sets the maximum file size for uploads.
   - `post_max_size` : Sets the maximum size of POST data.
3. **Memory Limit**:
   - `memory_limit` : Sets the maximum amount of memory a script can consume.

Example: Enabling Error Reporting

```
; php.ini
error_reporting = E_ALL
display_errors = On
log_errors = On
error_log = /var/log/php_errors.log
```

Error Handling

Overview

Error handling in PHP involves managing runtime errors, warnings, and notices. Proper error handling ensures that your application can recover gracefully from unexpected issues.

Key Concepts

1. **Error Types**:
   - **Notices**: Minor issues that do not stop script execution (e.g., accessing an undefined variable).
   - **Warnings**: More serious issues that do not stop script execution (e.g., including a missing file).
   - **Fatal Errors**: Critical issues that stop script execution (e.g., calling an undefined function).
2. **Custom Error Handlers**:
   - Use `set_error_handler()` to define a custom error handler.
3. **Error Suppression**:
   - Use the `@` operator to suppress errors.

Example: Basic Error Handling

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', 1);

echo $undefinedVariable; // Trigger a notice
?>
```

Error Reporting

Overview

Error reporting controls which errors are displayed or logged. It is essential for debugging and maintaining code quality.

Key Functions

1. `error_reporting()`:
   - Sets which errors are reported.
   - Example: `error_reporting(E_ALL)`.
2. `ini_set()`:
   - Changes configuration settings at runtime.
   - Example: `ini_set('display_errors', 1)`.

Example: Enabling Error Reporting

```
<?php
error_reporting(E_ALL);
```

```
ini_set('display_errors', 1);
?>
```

Exceptions

Overview

Exceptions are used to handle runtime errors in an object-oriented way. They allow you to separate error-handling logic from the main code.

Key Concepts

1. `try-catch` **Blocks**:
   - Handle exceptions gracefully.
2. **Throwing Exceptions**:
   - Use `throw new Exception()`.

Example: Using Exceptions

```
<?php
function divide($a, $b) {
    if ($b == 0) {
        throw new Exception("Division by zero is not allowed.");
    }
    return $a / $b;
}

try {
    echo divide(10, 0);
} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
}
?>
```

Error Suppression

Overview

Error suppression is used to ignore specific errors. It is done using the `@` operator.

Example: Suppressing Errors

```
<?php
$file = @fopen("nonexistent.txt", "r");
if (!$file) {
    echo "File not found.";
}
?>
```

Triggering Errors

Overview

You can manually trigger errors using the `trigger_error()` function. This is useful for custom error handling.

Example: Triggering an Error

```
<?php
$age = -5;
if ($age < 0) {
    trigger_error("Age cannot be negative.", E_USER_WARNING);
}
?>
```

Error Handlers

Overview

Custom error handlers allow you to define how errors are handled in your application.

Example: Custom Error Handler

```
<?php
function customErrorHandler($errno, $errstr, $errfile, $errline) {
    echo "<b>Error:</b> [$errno] $errstr in $errfile on line $errline";
}
set_error_handler("customErrorHandler");
```

```
echo $undefinedVariable; // Trigger custom error
?>
```

Error Logs

Overview

Error logs store error messages in a file for later review. This is useful for debugging in production environments.

Key Functions

1. `error_log()`:
   - Logs an error message.
   - Example: `error_log("This is a custom error message.")`.
2. `ini_set('log_errors', 1)`:
   - Enables error logging.

Example: Logging Errors

```
<?php
ini_set('log_errors', 1);
ini_set('error_log', 'php_errors.log');

error_log("This is a custom error message.");
?>
```

Summary of Key Points

FeatureDescriptionPHP.ini Settings Configuration file for PHP. Controls error reporting, file uploads, memory limits, etc.**Error Handling**Managing runtime errors, warnings, and notices.**Error Reporting**Controls which errors are displayed or logged.**Exceptions**Object-oriented way to handle runtime errors.**Error Suppression**Ignoring specific errors using the `@` operator.**Triggering Errors**Manually triggering errors using `trigger_error()`.**Error Handlers**Custom functions to handle errors.**Error Logs**Storing error messages in a file for later review.

Practical Questions

1. Enable error reporting in a PHP script and trigger a notice.
2. Create a custom error handler to log errors to a file.
3. Use a `try-catch` block to handle an exception.
4. Write a script to suppress errors when opening a non-existent file.
5. Trigger a custom warning if a user enters a negative age.