PHP

VivifyAcademy

Overview

- Debugging
- Type of errors
- Dump and Die
- Exception handling in PHP

Debugging

VIVIFY
ACADEMY

Debugging

Debugging is the single most important skill in programming.

Debugging is the **process** of finding and resolving bugs or defects that prevent correct operation of computer software or a system. - Wikipedia

Process is the foundation of effective debugging.

Gain **experience** with code and tools.

Develop your **intuition**.

display_errors

It is important that you configure PHP correctly and write your code in such a way that it produces meaningful errors at the right time.

First, you should turn display_errors on. This can be done either in your php.ini file or at the head of your code like this:

```
<?php

ini_set('display_errors', 'On');
```

error_reporting

Next, you will need to set an error reporting level. Set your error reporting level either in your php.ini or amend your runtime code to look like this:

```
<?php

ini_set('display_errors', 'On');

error_reporting(E_ALL);
```

Useful links

- https://stackoverflow.com/questions/tagged/php

Type of errors

Fatal Errors

Fatal Errors sound the most painful of the four but are in fact often the easiest to resolve. What it means, in short, is that PHP understands what you've asked it to do but can't carry out the request. The most common fatal error is an undefined class or function and the error generated normally points straight to the root of the problem:

```
Fatal error: Call to undefined function create() in /Document/Root/example.php
on line 23
```

Syntax Errors

Syntactical errors or parse errors are generally caused by a typo in your code. For example a missing semicolon, quotation mark, brace or parentheses. When you encounter a syntax error you will receive an error similar to this:

```
Parse error: syntax error, unexpected T_ECHO in /Document/Root/example.php on line 6
```

In this instance it is important that you check the line above the line quoted in the error (in this case line 5) because while PHP has encountered something unexpected on line 6, it is common that it is a typo on the line above causing the error.

Warnings

Warnings aren’t deal breakers like syntax errors. PHP can cope with a warning, however, it knows that you probably made a mistake somewhere and is notifying you about it.

Notice

Notices aren’t going to halt the execution of your code either, but they can be very important in tracking down a pesky bug. Often you’ll find that code that’s working perfectly happily in a production environment starts throwing out notices when you set error_reporting to E_ALL.

Useful links

- https://www.w3schools.com/php/php_error.asp

Dump and Die

var_dump()

This function displays structured information about one or more expressions that includes its type and value. Arrays and objects are explored recursively with values indented to show structure.

```
<?php

$a = array(1, 2, array("a", "b", "c"));

var_dump($a);

?>
```

die() and exit()

These functions are special in PHP, they terminate the current script and output a message, if a message is given.

If a string is passed, the function will print the value just before exiting. If an integer is passed, it will be used as an error code.

Both `die()` and `exit()` do the exactly same thing, so it doesn’t matter which one you use.

die() and exit()

`die()` and `exit()` functions are mostly used for debugging purposes.

If an error occurs, there are better ways to handle the error than to present user with a white screen and some message.

```
<?php

$site = "https://www.example.com/";

fopen($site,"r") or die("Unable to connect to $site");

?>
```

Exception handling in PHP

Exception handling in PHP

Exception handling is used to change the normal flow of the code execution if a specified error occurs.

Exception handling in PHP

What happens when an exception is triggered?

- The current code state is saved.
- The code execution switches to a predefined exception handler function.
- Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different code location.

Exceptions should never be used just to jump to a certain part of the code.

Exception handling in PHP

Exceptions are handled like this:

1. Code that that may throw an exception is wrapped in a `try` block.
2. If no error occurs the code will continue executing in a regular way.
3. If an error occurs PHP will look for a matching `catch` block, and if it finds it, the code in the matching `catch` block will be executed. If there is no matching `catch` block, a fatal error with a message `"Uncaught exception"` will be thrown. You can also `throw` or rethrow an exception in a `catch` block in order for some other function to deal with it. Throw command actually triggers the exception.
4. A `finally` block can also be specified, which will always be executed after the `catch` blocks, even if no exception was thrown.

Exception handling in PHP

You can also write your own exception types. In order to do that, you must create a special class with functions that can be called when an exception occurs in PHP. This class must extend the `Exception` class.

Example

```php
<?php
function divide($x, $y) {
    if (!$y) {
        throw new Exception('Division by
zero.');
    }
    return $x / $y;
}
try {
    echo divide(5, 0) . "\n";
} catch (Exception $e) {
    echo 'Caught exception: ',
$e->getMessage(), "\n";
} finally {
    echo "First finally. \n";
}
...
```

```php
...
try {
    echo divide(6, 3) . "\n";
} catch (Exception $e) {
    echo 'Caught exception: ',
$e->getMessage(), "\n";
} finally {
    echo "Second finally.\n";
}
// Continue execution
echo "Hello World! \n”;
?>
```

Example Result

```
Caught exception: Division by zero.

First finally.

2

Second finally.

Hello World!
```

Throwing exceptions vs. die() & exit()

`die()` or `exit()` should never be used in production code, since it's transporting information irrelevant to end-users (a user can't do anything about "Cannot connect to database").

Exceptions are thrown if you know that at a certain critical code point, your application can fail and you want your code to recover from it.

Useful links

- http://php.net/manual/en/language.exceptions.php

Zadaci

1. Napisati funkciju koja racuna obim kruga (2*r*pi). Ukoliko poluprecnik kruga nije veci od nule, baciti izuzetak koji o tome obavestava korisnika.
2. Iskoristiti gornji primer tako da se vrednosti za poluprecnik uzimaju iz niza brojeva, i tako da se skripta ne prekida fatalnom greskom (koristiti try/catch).