TASK

PHP - Overview

Visit our website

Introduction

JUST WHEN YOU THOUGHT THE WEB COULDN'T GET ANY WEIRDER

PHP is a scripting language with an usual history. Originally intended to be no more than a limited set of convenience methods to extend HTML, PHP has today become a fully-fledged programming language and one of the most popular tools used for creating web apps. This task covers the basics of how PHP works and its syntax for what should be already-familiar programming concepts.

THE FLEXIBLE ECLIPSE

As discussed in the Introduction to Java task, the Eclipse IDE supports many more languages than just Java. One of these is PHP, and to get it going all you need to do is install the PHP Development Tools extension for Eclipse. Details are at the bottom of this document in the first Compulsory Task.

Note that there are plenty of PHP IDEs out there, as a quick Google search will show. Any one of them will do the job.

WHAT MAKES PHP SPECIAL?

PHP was created specifically for the web. In fact, its original 1.0 release had no other use case. Even today it has tons of convenience methods and global variables that make coding for the web a breeze compared to other languages.

PHP code you write executes server-side. Its job is to interpret a client's request and send back a response. Sometimes this is as easy as sending "Hello World", but it can get about as complicated as you like — from session and cookie management to database connections, to reverse proxy calls. PHP has everything you need to make a fully-functional web application.

Unlike JavaScript, CSS, and HTML, which execute on a user's browser, PHP executes on the server and returns resulting HTML, which then executes client-side. This allows dynamic web content to be sent to the user based on their interactions with the website. Later in this task, we'll look at how this works.

Note that PHP major version updates often change a lot of things about the language. This course covers the latest version at the time of writing, which is

PHP7, but the content is also compatible with PHP8 (released in November 2020). Understanding PHP's fundamentals will make it easy for you to use any modern version.

CREATING A SIMPLE PHP SCRIPT

While it isn't necessary, PHP code is often embedded inside HTML for convenience purposes. The following code generates a webpage that shows the current time.

```
<h1><?php echo "Hello there!" ?></h1>
<p>The current time is
   <?php echo date('H:m') ?>
</p>
```

You will recognise the **`h1`** and **`p`** tags from earlier in this course. The **`php`** tags (much like the **`style`** and **`script`** tags) indicate that what is between them is PHP code. However, unlike **`style`** and script **`tags`**, **`php`** tags are evaluated server-side. By the time the client receives the result, there is no PHP left:

```
<h1>Hello there!</h1>
<p>The current time is 20:05</p>
```

`echo` is a special keyword in PHP that prints the expression to its right to the document (the output being returned to the client). **`date()`** is a function that gives the current date and time formatted however you specify. To use **`date`** (and all other PHP built-in functions) you don't need to explicitly import anything like with Java. PHP's built-ins are always ready for action.

RUNNING PHP

When clicking "Run" in Eclipse on a PHP file, Eclipse will ask you how it should be run. Select "PHP Web Application" or equivalent. This will open a browser window to the appropriate location and make a request to your PHP script and show its response directly in-browser.

This requires that we set up a mini web server first. After PHP is installed on your system, open a command line or terminal and navigate to your project's directory. Run the following:

```
php -S localhost:8080
```

We'll be making use of this methodology for testing our PHP code since it best emulates how PHP typically runs in production environments.

INPUT

Input for PHP scripts usually come from parameters or other data attached to a client's request. It isn't obvious from our PHP script above, but what really happened was this:

1. A client (your browser) requested the webpage.
2. The server executed the appropriate PHP script.
3. The server sent the result of the PHP script back to the client.
4. The client rendered the result in-browser.

Let's have our PHP code make use of client request parameters.

```php
<?php
   echo "<h1>Hello, " . $_GET['user_name'] . '!</h1>';
?>
```

As you can see, not only can PHP be embedded in HTML, but HTML can also be embedded in PHP! The dot operator is used to concatenate strings together. Note PHP supports both single and double quotes to wrap strings. Versatile, isn't it?

`$_GET` is a superglobal — basically, a variable that is automatically managed by PHP and is available to your code everywhere and anywhere. **`$_GET`** specifically holds the parameters that come with a GET request. That is the default kind of request your browser sends when you visit a page. We'll look at other kinds of requests in later tasks. **`$_GET`** is structured like a map and square brackets are used to access one of its elements by name.

Note that the key "user_name" will only be defined in **`$_GET`** if the client adds it as a parameter to its request. So let's do that now. Navigate to the following URL in your browser:

```
localhost:8080/index.php?user_name=David
```

Depending on how your IDE handles its web serving, the URL may look different for you. The important thing is to add `?user_name=David` to what it was. This specifies the appropriate parameter PHP is expecting. Your browser should now be saying “Hello, David!” Change the request to contain your name to convince yourself it works properly.

VARIABLES

In PHP, variables work similarly to other languages you’ve used before, but there are two remarkable differences: firstly, all variable names start with a dollar sign ($). Secondly, they are weakly typed, meaning that, unlike Java, you don’t need to declare a variable’s type when creating it.

```php
<?php
   $var_num = 5;
   $var_str = 'The following is a number: ';
   $var_concat = $var_str . $var_num;
   echo $var_concat;
?>
```

Unlike in Java and JavaScript, it is customary to use snake case for the names of functions and variables in PHP (that is, `php_var` vs `javaVar`). Like in Java, PHP statements are semicolon-separated.

CONDITIONAL STATEMENTS

Conditionals work exactly the same in PHP as in all other languages. The syntax is exactly the same as in Java.

```php
if ($age < 18) {
   echo 'You are a minor.';
} else
   echo 'You are an adult.';
```

Just like in Java, using curly braces is not required for single-statement blocks. It is good programming practice to do it nonetheless.

Boolean expressions can be chained using boolean operators (and, or, not, etc.). In PHP you can use either the Java equivalent symbols or the plain English names.

```
if ( ($age > 18 && $age < 29) or !($height > 1.7) ) {}
```

A very unique feature of PHP is its ability to function almost anywhere in between HTML code. You can define conditionals at arbitrary places in HTML and anything between the curly braces will become part of the script's output if the conditional is true. In the following example, only one of the three paths will output, depending on the username.

```
<?php if ($_GET['user_name'] === 'David'){ ?>
   <h1 style="color: blue">Hello, David!</h1>
<?php } else if ($_GET['user_name'] === 'Mike'){ ?>
   <h1>Hi, Michael.</h1>
   <p>Glad you logged in.</p>
<?php } else { ?>
   <p>Could not recognise your username.</p>
<?php } ?>
```

Try it out to make sure it works as you'd expect with different usernames.

EQUALITY

To check equality of variables in PHP, always use three equal signs. Using two equal signs also does an equality check, but utilises type coalescing, which is usually not what you want.

```
if ($name === 'David'){
   // Hi David!
}
```

In the PHP OOP task we'll look at object equality.

ARRAYS

Arrays in PHP are a lot more versatile than in Java. Declaring them is super quick, they can easily change length, and can even hold different variable types in different slots (although this generally isn't recommended).

```
$arr = ['a', 'b', 'c'];
array_push($arr, 'd'); // adds 'd' onto the array
$len = sizeof($arr); // will be 4
$first_el = $arr[0]; // arrays are 0-indexed, like Java and JS
```

Unlike in Java, PHP variables generally don't have member functions (like `.length()`, `.add()`, etc.) To do things like this with variables, you call built-in functions (like `array_push()` and `sizeof()`) and pass in the relevant variable.

To get the length of a string, you should use `strlen()` instead of `sizeof()`, the latter of which is only for arrays.

LOOPS

The PHP *for* and *while* loops work and look exactly like their Java counterparts. The following loops each double the numbers in a given array. Read through them and make sure you understand how they work.

```
$arr = [1, 2, 3];

$i = 0;
while($i < sizeof($arr)){
   $arr[$i] = $arr[$i] * 2;
   $i++;
}

for($i = 0; $i < sizeof($arr); $i++){
   $arr[$i] = $arr[$i] * 2;
}
```

Note that `continue` and `break` statements work exactly the same as in Java.

The *foreach* loop in PHP looks a little different but works the same as in Java.

```
$arr = [1, 2, 3];
foreach($arr as $element){
   echo $element * 2;
}
```

Note you cannot do assignments in a foreach loop. It is meant for reading an array only.

FUNCTIONS

Functions in PHP work the same as in any other programming language but are much easier and quicker to define. You don't have to specify argument or return types.

```
function concatenate($text, $more_text){
   return $text . $more_text;
}
function output($content){
   echo $content;
}

output(concatenate('This is ', 'PHP!'));
```

Unlike in Java, functions have to be declared before they can be used. That is, literally higher in the PHP file.

A NOTE ON ERRORS

In a future task, we'll look at how to deal with errors in PHP. An important thing worth noting at this stage is that PHP reports errors to a different place than its output. When an error occurs (like an invalid array index access) the script stops executing, but the response generated so far is still sent to the client. Depending on your script this could result in half-baked pages, or even completely blank pages. In this case, PHP should have printed some information about the error to its webserver's console output.

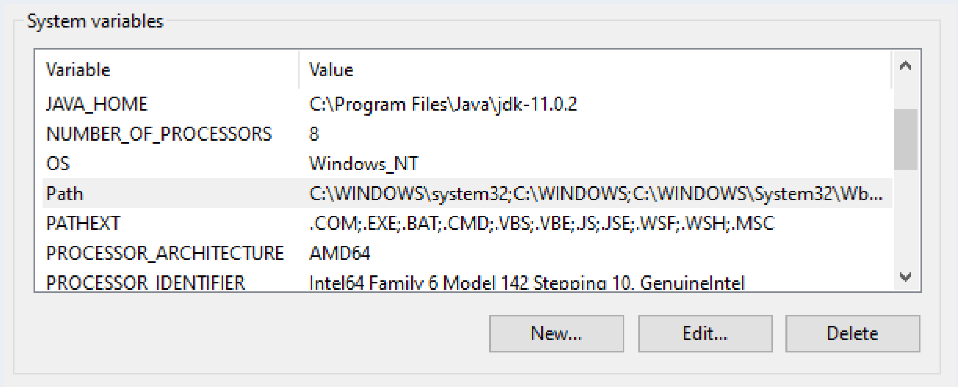
Compulsory Task 1

Install PHP on your machine. Doing this is relatively easy for macOS and Linux because it's the kind of environment where PHP typically runs as a production application. It involves no more than one or two lines on the terminal. Follow the instructions outlined in the following:

- macOS: **https://tecadmin.net/install-php-macos/**
- Linux: **https://www.php.net/manual/en/install.unix.debian.php**

The PHP installation is a bit trickier on Windows. To ease the process we created an automatic installer for a typical Windows installation of PHP 7 and is in your task folder. We recommend trying the manual installation yourself (by following the instructions below), but you are free to use the installer if you want to save time.

- Download the latest PHP zip from here: **https://windows.php.net/download/**
- Create a folder under `C:\Program Files (x86)\` called `php` and extract all the zip's contents there. You'll need to grant admin permissions when prompted.
- Press Start and type "Environment Variables". Click on the top result.
- At the bottom of the opened dialogue, click Environment Variables.
- Under System Variables, find the item with the name "Path".



System variables

Variable	Value
JAVA_HOME	C:\Program Files\Java\jdk-11.0.2
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wb...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 10, GenuineIntel

- Click on it, then click on Edit.
- In the new dialogue, click on New and then immediately on Browse.
- Use the new file selector dialogue to select the folder where you unzipped PHP. (`C:\Program Files (x86)\php`).

- Click OK three times to confirm-away all opened dialogues.
- To confirm that PHP is successfully set up, open a command line or terminal and run:

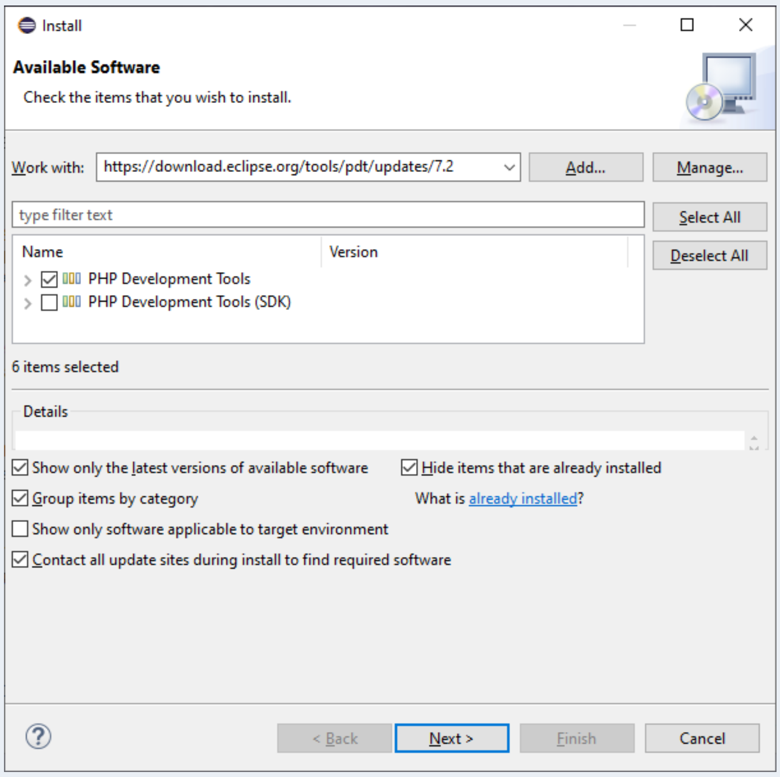
```
php --version
```

- Take a screenshot of the terminal to show that PHP has been installed correctly and paste it in this task's folder.

Optional Bonus Task

This task will walk you through how to set up Eclipse for PHP development.

- In the Eclipse menu (top bar) click on Help → Install new software.
- In the dialogue box that opens, paste the following into the text box next to "Work with:" and hit enter: **https://download.eclipse.org/tools/pdt/updates/7.2**
- This will let Eclipse find the PDT requirements. Once loaded, select the first item (PHP Development Tools).


Install

Available Software

Check the items that you wish to install.

Work with: https://download.eclipse.org/tools/pdt/updates/7.2

type filter text

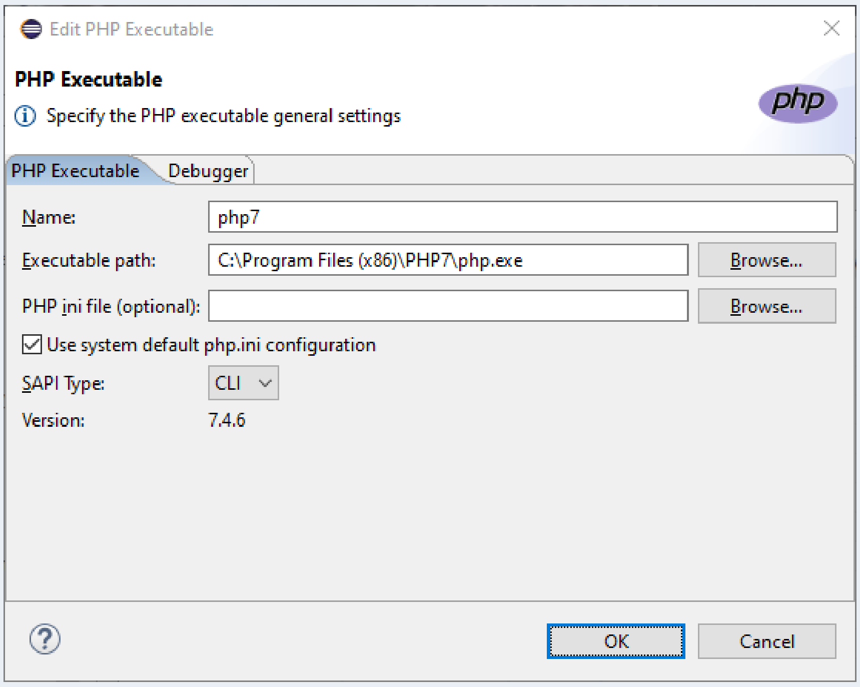
Name	Version
☑ PHP Development Tools	
☐ PHP Development Tools (SDK)	

6 items selected

Details

☑ Show only the latest versions of available software
☑ Hide items that are already installed
☑ Group items by category
What is already installed?
☐ Show only software applicable to target environment
☑ Contact all update sites during install to find required software

- Click “Next”. Eclipse will prepare the installation. Once ready, click “Next” twice, accept the terms and conditions (after reading it of course) and finally click “Finish” to do the installation.
- When complete Eclipse will ask to be restarted. Do so.
- After the restart, in the menu, click on Window → Preferences.
- In the left navbar, navigate to PHP → Installed PHPs.
- To the right of the empty list, click Add.
- In the new dialogue enter the appropriate details. It should look something like this:



Specify the PHP executable general settings

Name: php7
Executable path: C:\Program Files (x86)\PHP7\php.exe
PHP ini file (optional):
☑ Use system default php.ini configuration
SAPI Type: CLI
Version: 7.4.6

- Click OK and then Apply and Close.
- To create a new PHP Project, Click on File → New → Other, and then select PHP Project.
- After you’ve created a project, right-click on your project in the left pane and select New → PHP File. Name it anything.
- Enter some “Hello World” code, just for testing.
- Set up a mini webserver on your machine using the instructions under the “Running PHP” section above.
- Click on Run and then select PHP Web Application.

- In the dialogue box asking for server URL, enter the following (replacing the filename with your file's name):
  http://localhost:8080/newfile.php
- Voila, your PHP code should now be running through your webserver right in Eclipse.

Compulsory Task 2

- Create a new file called **names.php**
- Write a script that shows a different greeting for different usernames.
- Let your script get the username from an appropriately named GET request parameter.
- Categorise greetings as follows:
  - If the name is less than 3 characters:
    - (in green:) Hello, [name]!
  - If the name is between 3 and 6 characters (inclusive):
    - (in blue:) What's up, [name]?
  - If the name has 7 or more characters:
    - (in purple:) Top of the morning to you, [name].

Compulsory Task 3

- Create a new file called **primes.php**
- Create a function that checks if a given number is a prime number.
- Write a script that prints the first 10 prime numbers in the format of an html list (`<ul>`).

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.