PHP7 From SCRATCH

PHP7 PROGRAMMING

THOMAS PICHLER

PHP 7 from Scratch

A Streamlined Approach to Modern PHP Mastery

Thomas Pichler

This book is for sale at http://leanpub.com/php7fromscratch

This version was published on 2017-09-25

Tweet This Book!

Please help Thomas Pichler by spreading the word about this book on Twitter!

The suggested tweet for this book is:

Just bought @Sepixus' new book, PHP 7 from Scratch!

The suggested hashtag for this book is #php7fromscratch.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#php7fromscratch

Contents

Introduction

Before we get started, let me say thank you for choosing this book. There are probably a million books about PHP out there and the fact that you ended up with mine means a lot to me. I am confident that you will not be disappointed!

Why Another PHP Book?

I decided to write another PHP Book because most PHP books out there are outdated, contain too much information about things you never need or are written for computer science majors with lots of jargon and mathematical examples that have no value in real world applications. I believe that I can do better and what better time to do so than a new major PHP Release after 11 years?

Is This Book For Me?

This book is written for PHP beginners and people with little to no programming experience. If you used PHP in the past and forgot most of it this book might also serve you well. However, if you are a seasoned developer or a PHP programmer who just wants to learn about the new features in PHP 7 then you should go with another book.

How to Consume This Book

I am using a different approach in this book than most other people. Therefor the best way to read this book is from start to finish. The reason for that is that most chapters build on top of the previous chapters and you might get in trouble if you skip one. Beside that I would like you to know that we are not going the same route here as everybody else and you won't be bombarded with all kind of stuff before you are able to do something.Every few chapters we will have a Practical Project that deals with all we have learned previously.

How this book is different

Most PHP programming books these days are outdated. They teach you things which are useless these days. There is no need to know how to handle certain things from scratch anymore. Modern PHP Developers rarely code their own file handling solutions but use cloud storage and API's to communicate with them. They rarely code their own calendar and date time functions. They use

Carbon, Chronos or other libraries for that. This book will focus on the things that are actually important to get you up to speed as fast as possible and send you on your way from there. In case you ever need anything not covered in this book you will be able to look it up yourself without much trouble.

Exercises

Every Chapter in this book comes with 3 to 5 exercises that are labeled with a difficulty rating. The solutions to these challenges can be found in the Solutions section in the end of this book. When viewing said solutions keep in mind that most programming problems have a wide range of possible solutions so don't be too shocked if my solution looks different than yours.

Some of these challenges will require you to use Google to find the methods needed to complete them. This has the purpose of preparing you for your own development career after this book. Knowing how to find the answers you need is already half the battle!

Code Files

All the code required for challenges and practice chapters is available in a zip file, that is included with the purchase of this book and as a Github repository. That way, you can skip the copy and paste part and get straight to work. The code can be found under the following link:

https://github.com/Sepix/php7fromscratch-code

Errata

I am trying my best to create this book as perfect and error free as possible but I am only human. That means that from time to time you may encounter some errors, typos and other buggy things during your reading session. If you encounter something that strikes you as especially annoying or threatens the honor of the english language please let me know about it and I will fix it as fast as possible.

How to Get in Touch

In case you want to report a bug, typo, or problem with the book, or have any kind of feedback for me you can reach me in the following ways:

Email: thomas@thomaspichler.me

Just saying Hi is okay too. I won't bite. :)

Chapter 1: The Setup

All right! Let's get started with the first and probably worst chapter of the book. This Chapter is about setting up your development environment and the famous Hello World example that every programming book needs to have. Depending on your operating system you will either follow the "Setup for Mac" or "Setup for Windows Section". If you are on a Linux computer just follow the "Setup for Mac Section". It will be close enough and you should be able to figure out the missing links yourself. :)

The Shortcut Route

If you are completely new to the developer side of things and have never dealt with a more complex environment, or if you feel just eager to get started right away, then you can use one of the quick solutions below. That way you can get a PHP server on your system in a few clicks and don't have to deal with any of the configuring that is required for the more advanced options in this chapter. Keep in mind that by doing so you will trade comfort for flexibility but it will not stop you from following along this book by any means. The choice is yours!

For MacOSX: MAMP[1] For Windows: EasyPHP[2]

If you decided to use one of the above 2 versions, then you can go ahead and move along to Chapter 2. For everybody else, time to get dirty!

Setup for Mac

Welcome to the Mac section!

Since you are on a Mac I will assume that you know at least a little about the command line which should make this whole process a little easier. Should you have any problems with any of the steps please contact me with the problem you are having so I can help you fix it and include the common pitfalls into this book for future reference.

Step 1: Installing Sublime

There is a ton of Text Editors and you can basically choose whichever one you like. However Sublime is free, has all the functions we need and is incredibly flexible which is the reason why I will use it in this book.

[1] https://www.mamp.info/en/downloads/
[2] http://www.easyphp.org/easyphp-devserver.php

To get started just visit [the Sublime Website)(http://www.sublimetext.com/3) and download the latest version for your operating system. After everything is installed open a terminal window and type the following:

```
ln -s /Applications/Sublime\ Text.app/Contents/SharedSupport/bin/subl /usr/local/bin/sublime
```

This makes it possible for us to open files via the command line which will make your life a lot easier if you have never used the vi-editor before.

Step 2: Installing Composer

Open the Terminal App on your Mac and copy and paste the following:

```
curl -sS https://getcomposer.org/installer | sudo php – –install-dir=/usr/local/bin –filename=composer
```

Just type in your password when you are prompted to do so and the terminal will do the rest. After the installation is complete either close your Terminal Window and reopen it or open a new tab with "Command + T", then type the following:

```
composer
```

If everything went fine you should see a list of commands and options. That means we are done and move on to the next Step!

Step 3: Installing Laravel Homestead

Laravel Homestead is just … wonderful. It will create a virtual server for you that will always have the newest versions of PHP and everything else you need. Completely for free and without being effected by any external changes on your machine. Updating to a new OSX version without affecting your current development environment is more than sweet. To get it running however we need to do a few things first.

Installing Virtual Box

Visit the VirtualBox website[3] and download the latest version for your operating system. Then just follow the installation manager. Easy Peasy Lemon Squeezy!

Installing Vagrant

Just visit the Vagrant website[4] and download the latest version for your operating system. After that execute it and follow the installation manager as usual. To make sure everything went well open a new terminal window and type:

```
vagrant
```

You should see a list of commands and options.

Time for Homestead

First we need a so called box. These are pretty much blueprints for virtual servers. However the technical details really don't matter to us here so all we need to do is type the following into our terminal window:

```
vagrant box add laravel/homestead
```

This might take a while to download. Go ahead and have a well deserved break, you are almost done.

Once the download finishes we will use composer to pull in the homestead composer repository. This will make our life a lot easier. Again we are going to skip the technical details for now and type the following into our terminal window:

```
composer global require “laravel/homestead=∼2.0”
```

This should only take a few moments to complete. To test if everything went fine open a new terminal and type:

[3] https://www.virtualbox.org/wiki/Downloads
[4] https://www.vagrantup.com/downloads.html

```
homestead
```

As usual you will see a list of commands and options. Time to configure!

Step 4: Configuring Laravel Homestead

Good Job for making it here. You are almost there! The first thing we need to do is create a configuration file. This is done by typing the following into our terminal window:

```
homestead init
```

This will create a homestead.yaml file which will hold our configuration options. Next we need a so called SSH key. If you already have one you can ignore this step, otherwise just type the following into your terminal window:

```
ssh keygen -t rss -C “your email that you want to use”
```

Make sure you use your email address and not my placeholder. :) After using the command just follow the instructions until the key is generated. That is all!

Only a few configurations left. Next on the list is editing our hosts file. That is the file that gets called first every time you open your browser and overwrites the IP addresses delivered by the DNS Servers of the internet. If that sounds too complicated it basically means you can have your own google.com domain if you want to, you just won’t be able to use google while you have it in your host file. :)

To get started type the following into your terminal window:

```
sudo sublime /etc/hosts
```

If you get prompted for your password again just type it in as usual. Soon after Sublime should open with the contents of your host file. In the bottom of the file add the following line:

```
192.168.10.10 php7fromscratch.app
```

Save the file and we are done here. Next is to tell homestead about our domain and what to do with it. To do so type the following into your terminal window:

```
homestead edit
```

This should open your homestead.yaml file. Inside you should see a section called sites that contains a mapping for homestead.app to give you an idea how the file should be formatted. Since we just "registered" php7fromsratch.app we can add the following into the site section below the homestead.app placeholder:

```
 map: php7fromscratch.app to: /home/vagrant/Code/php7fromscratch
```

Make sure there is one empty line between them and save the file. All that's left now is creating our folder and a test file. We can do that the following way:

```
mkdir -p Code/php7fromscratch echo "Test" > Code/php7fromscratch/index.php
```

Good! Time to see if everything works. Type the following into your terminal:

```
homestead up
```

This is your virtual server booting up and loading everything. If you get an error message check your homestead.yaml file and see if you added our php7fromscratch mapping correctly. Otherwise just wait for everything to finish and visit:

```
http://php7fromscratch.app
```

You should see the word Test in your browser.

Congratulations! You made it through the hardest part. Everything from here on will be a cakewalk. Not to mention that you are now working in an environment of a real web developer. Good Job! Time for Chapter 2.

Setup for Windows

Welcome to the Windows section. I am not going to lie, the Windows installation path is going to be a lot more complicated than you want it to be. I will try to be as clear as possible however and hope that we can make this work. If you run into any problems please message me with the issue and your windows version so that I can help you fix the problem and include the issue in this book for future students.

Step 1: Installing the required software

To get started we will need a few things installed. Please make sure you read each step carefully first and don't skip ahead. It will make it a lot easier i promise.

Git

Git is a version control software. It lets you reverse every change you ever make to your code with ease and can be used to organize projects that involve many people. The installation itself is quite straight forward.

Visit https://www.atlassian.com/git/tutorials/install-git/windows[5]

Make sure you follow the sites instructions and make sure you select "Do not use Windows Command Prompt" when installing since we will be using the Git Bash.

Git Bash

Gitbash will be used to emulate a Linux Command line in your Windows environment and is very straight forward. Just go to the Gitbash Website[6] and click on the Windows Logo under Downloads. After it's done just double click the executable and follow the instructions until everything is completed.

[5]https://www.atlassian.com/git/tutorials/install-git/windows
[6]https://git-scm.com/downloads

Vagrant

Vagrant is going to be used to create a virtual environment on your windows machine. Pick up a copy of Vagrant over here: https://www.vagrantup.com/downloads.html[7]

Before installing this be advised that Vagrant uses a programming language called Ruby to function. Ruby as trouble with spaces inside directory names so make sure there are no spaces inside the directory structure that you are planning to install it in. "CProgram Files" for example is a no go. I suggest you select something like "C:ToolsVagrant" as your installation point.

That said, the rest should be straight forward. Double click the executable and follow the instructions until everything is installed. Just make sure you are aware of the directory structure and the space problem.

Virtual Box

Virtual Box contains the virtual environment blueprints that we will be using with vagrant to create our virtual environment. These blueprints come in the form of so called "boxes" and contain standard configurations for different types of setups. Installing it is easy however.

Just visit: https://www.virtualbox.org/wiki/Downloads[8]

Select the windows version suitable for you and then just follow the installation instructions as usual. No pitfalls here.

Bitvise Tunnelier

Next we need Bitvise Tunnelier. This will be used as an SSH tunnel so that we are able to login to our server via the ssh protocol in the command line. Sounds complicated but really isn't, so don't worry! Luckily the installation isn't hard either.

Visit: https://www.bitvise.com/ssh-client-download[9]

After downloading just follow the installation instructions and you should be good to go.

Alright! That should cover the installation part of the setup.

Step 2: Configuring The Environment

Time for all the configurations that are ahead of us. Before you get started take a deep breath and know that we are already half way through this. Don't worry, we can do this!

[7]https://www.vagrantup.com/downloads.html
[8]https://www.virtualbox.org/wiki/Downloads
[9]ttps://www.bitvise.com/ssh-client-download

The Vagrant Box

First we need to add the Vagrant Box for LaravelHomestead. To do that we open Git Bash, which we installed earlier. Make sure you open it "As Administrator" just in case so we can bypass any weird permission errors that might occur. When you have done that type the following into the shell:

```
vagrant version
```

If you installed Vagrant correctly you should see the version of your Vagrant installation. If not go back to the installation step an make sure you installed Vagrant correctly. Next we need to get our box. To do so type the following into the shell:

```
vagrant box add laravel/homestead
```

This should start downloading the box, which is several GB in size. Depending on your internet speed this might take a while so go ahead and take a break until the box is finished.

Vagrant Plugins

We got our box now but to actually use it we will need to edit our host file. You will have to add an entry to your host file for every new project you undertake and this might become bothersome over time. To solve that we can install a vagrant plugin called HostUpdater. This will update your hosts file automatically for you in the future. To do so all you have to do is type the following into your shell:

```
vagrant plugin install vagrant-hostsupdater
```

And that's all!

Installing Homestead

So far we have the box for Homestead but not Homestead itself. However we can do that really quick via the shell as well. All you have to do is type the following 2 lines:

```
cd ~ git clone https://github.com/laravel/homestead.git Homestead
```

This should install Homestead into your User directory. Next we need to initialize the Homestead configuration file. For that we use the shell again:

```
cd ~/Homestead bash init.sh
```

And thats it so far!

Generating your SSH Keys

To connect to our virtual server we will need something called an SSH key. SSH keys always come in pairs and have a public and private part. To generate them we will use Bitvise Tunnelier which we installed above. To find it open your windows menu and search for 'keypair'. All you have to do here is generate a key pair and then export it. To export a key click on the key in the list and select export. Select any location you want and make sure you export the key in "Open SSH Format".

Configuring Homestead

First we need to configure our HostUpdate plugin. To do we go to the homestead.rb file inside your homestead/script folder. If you don't know where that is you should be able to reach it under the following path:

```
%APPDATA%Composer\vendor\laravel\homestead\scripts\homestead.rb
```

In there at the following line at the bottom

```
config.hostsupdater.aliases = settings["aliases"]
```

Next we need to change our Homestead.yaml file. That is the file that contains our virtual environment configuration. You can find it under the following path:

```
%HOMEPATH%.homesteadHomestead.yaml
```

When you open that file you see a bunch of configuration options like authorize, keys, folders etc. The first thing we need to do is add our aliases as a configuration option. Just add the following line:

```
aliases: [“php7fromscratch.app”]
```

Whenever you add another url make sure to update the aliases configuration accordingly. The next thing that interests us is the folders configuration. This tells us what folder of your desktop computer is used for the code that will be used on the server. the map key points to the folder where your code is and the to key tells you where on the server the code will be. You can leave the “to” config alone and just adjust the map part of the config. For Example if all your projects are located in D:DataCode then your configuration would look like this:

```
folders: - map: D:DataCode to: /home/vagrant/Code
```

Below the folders configuration you will also find a site configuration. This maps a specific project folder to a virtual domain. In our case that is going to be php7fromscratch for now. You can add as many projects as you want however. To do so the configuration looks like this:

```
sites: - map: php7fromscratch.app to: /home/vagrant/Code/php7fromscratch
```

Make sure to update the to configuration accordingly based on the “to” configuration of your folders config.

Congratulations! All the configurations are now complete!

Launching Homestead

To launch homestead open Gitbash and type the following:

```
cd ~/Homestead homestead up
```

This should start your server. If you then open your browser and type http://php7fromscratch.app everything should work. If you don't see anything make sure you create an index.php file and fill it with some HTML code inside the php7fromscratch folder. If you are done working and want to shut down the server all you need to do is type:

```
homestead suspend
```

That should be it! See you in Chapter 2!

Chapter 2: Hello World

First of all, let me congratulate you for making it here. The worst part is over but before we can output our first Text Snippet into the browser we need to learn about a few more things. Since you are currently on a roll i won't bore you much longer and move on!

PHP Tags

PHP Code is always embedded between tags:

```
<?php ?>
```

These tags tell the web-server that he needs to send whatever is in between these tags to the PHP interpreter to be processed correctly.

One rule for modern PHP applications is to never close the php tag at the end of a file if the file does not contain HTML markup. This rule exists because many PHP Developers kept running into painful errors if they had empty lines below the closing tag and wasted hours to find the cause. Fun times!

Therefor if we do not have HTML in the file, leave out the closing tag in the bottom. A good rule of thumb is to always take responsibility off of ourselves if we can. We will learn a lot about that later when we hear about testing.

Comments

Comments are used to describe or, well, comment on things throughout your code. This is useful to remind yourself about stuff that still needs to be implemented or are only temporary for development purposes and need to be removed later. They are also used for reusable chunks of code in larger projects to make it easier for other developers to understand your code. Last but not least, and you might not know it yet, even you will forget what your own code is about if you don't work on it for a while. Therefor comment detailed and comment often. The faster you get used to it the better!

Comments can be implemented in 2 different ways:

```php
<?php

        // This comment has 1 line

        /*
         * This comment
         * has multiple
         * lines
         */
```

As you can see not very complicated! Comments are easy and we will be using them every chance we get throughout this book to get you used to them!

echo

It is time for the actual "Hello World" statement. For that we will be using the echo command.

Echo is pretty simple and looks like this:

```php
<?php

    echo 'Stuff to output';
```

You can use either ' ' or " " to encapsulate your output but I recommend using the single quotes. The reason for that is that most editors use the double quotes for HTML tags. Therefor if you are using HTML tags inside an echo statement you will need to use the other kind of quotes inside to make sure you do not break the string. We will look at this closer in the next chapter. For now just get used to the single quotes.

The semicolon

You might have noticed the ; at the end of the echo statement. This semicolon is very important and tells PHP that the statement is over and that it should prepare for the next one. PHP depends solely on the semicolon do determine when to stop reading and start processing before going further. It is unable to read and interpret line breaks and other formatting you are seeing. The general rule is to use a semicolon after every statement with a few exceptions that we will see later on. Missing semicolons will be the majority of your errors at the beginning of your PHP career so watch out when ever you see a "Parsing Error" Statement in your browser. :)

All together

Time to finish up with this Chapter!

Open your index.php file and insert the following into it:

```
<?php

        echo 'Hello World';
```

Save the file and visit your local website to see the word Hello World in your browser window. Congratulations you just wrote your first PHP script! Onward to the challenges!

Challenges

In this chapter you learned how to output text in your browser using PHP and these exercises will help you deepen your knowledge about it. The solutions can be found in the back of this book!

Easy

Change our output to 'I got this! I know how to output text now!'

Medium

PHP has a second command to output text into the browser called print(). It has a different syntax than echo does. We will cover the reason for this a bit later but for now all you need to know is that these functions are almost completely identical in their use. The only difference that matters for us at this point is that echo is faster than print();

Change our Hello World Script so it uses print(); instead of echo. (Google allowed)

Hard

Turn our index page into a php info page. (Google allowed)

Chapter 3: Variables and Data Types

Last chapter we learned how to output data to our browser via the echo command. While this is useful for certain things we won't use it in that form very often. Most of the time programming you will spend with juggling around variables or creating the processes to do so. You will set them once and then change, push and receive them all around your code but before we can do that we need to look at what a variable is and how it looks like. A variable in PHP always starts with the $ sign and can have any name you want it to have.

There are a few rules on how to name a variable however:

The Variable name is case sensitive. $Hello is different from $hello. Variable names cannot start with a number but the name can contain them. $hello4 is allowed. $4hello isn't. Whitespace or special characters are not allowed beside underscores. $hello_ is allowed. So is $hell_llo and so on. Variables can begin with underscores. Even $_hello works.

To save yourself the confusion just follow a simple pattern. Always name your variables with lower case letters and try to use underscores as rarely as you can. Also try to give your variables names that give a general idea of what that variable is used for. This makes it easier for other people reading your code and helps you remember what you did there after a long period of time as well.

The = Operator

Variables exist to store things in them. Imagine you would have to write the same thing over and over throughout the code and then notice you made a mistake. You would have to change every single instance of it instead of just the value of the variable. To save data into a variable we use the = operator. The = operator binds the value on the right to the variable on the left and you cannot do it the other way around. PHP is picky that way. After the value is bound to the variable you can use the variable instead of the value for the rest of the time. The content of a variable can be changed anytime throughout your code and the variable will forget its old value if you bind a new one to it. To see how that works lets go ahead and change our Hello World example a little.

```php
<?php

$hello = 'Hello World!';
echo $hello;
```

As you can see we now bind ‘Hello World’ to the variable $hello and echo that one instead. It is that easy.

Now that we got the basics down it is time to look at what variables can contain. There are a few different data types in PHP and each of them has its use. As a start we will go over 3 Data types and introduce the rest whenever we start using them throughout this book. I just want you to know that datatypes do become a little more important with PHP 7 than they were before. That however is a good thing and nothing to be afraid of!

Data Type - String

Strings are used to save a bunch of characters together. Strings are always written between single ‘’ or double “” quotes and you have used one earlier in our Hello World example. We use strings whenever we save an output that is not a number and there is a lot you can do to them which we learn later.

```
<?php

$hello = 'Hello World';
$name = 'My name is ......';
$age = 'Iam 29 years old';
```

Data Type - Integer

Integers are used to save numbers but not all of them. Integer is only able to save natural numbers and even those are limited to certain ranges depending if you are on a 32 or 64 bit system. If that sounds too complicated do not worry because PHP takes over if you mess up and converts the number to a different data type automatically should you be out of scope. As long as you throw numbers into Integer you cannot go wrong. It is also important to mention that integer can save numbers out of any numeral system like hexadecimal, binary or even octal if you ever encounter a use case for them.

```
<?php

$number = 5;
$number2 = -25;
$number3 = 0x8C;
```

Data Type - Float

Float is used to save all those numbers Integer cannot. Be it numbers with a decimal point like 4.323 or numbers that are too big for integer to handle. Float is even able to handle numbers in exponential form like 2.4e3 but unless you are planning to create a banking or physics app you will probably never have a need for that.

```
<?php

$float = 4.32;
$float2 = -3213.555;
$float3 = 2.4e3;
```

Challenges

In this chapter you learned about variables, what they can save and how they can be used. Time to mess around with them yourself! Solutions can be found in the back of the book!

Easy

Create the output ‘This comes from the variable!’ and echo it to the browser by editing our “Hello World” example.

Medium

See what happens if you give a variable another value after it’s first declaration but before it is output to the browser.

Hard

Find out what happens if you use a variable to set the value of another variable and then echo it out to the browser!

Chapter 4: Arrays

Before we start you should know that this chapter might be a little harder to understand but stay with me here. We are getting to the more interesting stuff soon!

Data Type - Array

Arrays are a data type that enables us to save multiple values in one and the same variable. Each value gets a so called index over which it can be identified. Arrays can be created in 3 ways. Either with the array(); function, by $name[index] = value; or by putting the different segments inside [] brackets separated by a comma.

The index is always optional and will be given a number starting from 0 automatically if you do not specify one. Indexes can either be integer or strings and must be unique in the same array. Variables with strings as their index are also called associative arrays. There is an incredible amount of things that arrays are used for and the text above might be a bit confusing at first. Letâ€™s just look at a code example and go from there.

```php
<?php

// An array with the array() syntax

$foo = array('Apple', 'Banana', 'Pear');

echo $foo[0];               // will output Apple
echo $foo[1];               // will output Banana
echo $foo[2];               // will output Pear


// An array with the $name[index] = value; syntax

$foo[3] = 'Kiwi';

echo $foo[3];               // will output Kiwi


// No index specified. This will be attached to the end.

$foo[] = 'Chestnut';
```

```
echo $foo[4];               // will output Chestnut


// An array with the [] syntax

$my_array = ['a', 'b', 'c', 'd'];

echo $my_array[0];          // will output a
echo $my_array[1];          // will output b
echo $my_array[2];          // will output c
echo $my_array[3];          // will output d


// Associative Arrays

$bar = array(
    'fruit' => 'Apple',
    'vegetable' => 'Carrot',
    'animal' => 'Dog'
);

echo $bar['fruit'];         // will output Apple
echo $bar['vegetable'];     // will output Carrot
echo $bar['animal'];        // will output Dog

$bar['gender'] = 'male';

echo $bar['gender'];        // will output male
```

As you can see they are kind of like variables with assigned seats. Just call the seat number or keyword in the [] brackets and the right stuff comes out of it. :)

The var_dump() function

Arrays can get very complex. To work better with them it is sometimes required to see their structure on paper right in front of us. To do so we have the var_dump(); function. var_dump(); is used as a debug tool. You can feed it any type of variable and it will spit out its structure in your browser window.

```php
<?php

// Array

$foo = array('Apple', 'Banana', 'Pear');

$foo[3] = 'Kiwi';

var_dump($foo);

/* array(4) {
 *    [0]=> string(5) "Apple"
 *    [1]=> string(6) "Banana"
 *    [2]=> string(4) "Pear"
 *    [3]=> string(4) "Kiwi"
 * }
 */
```

This tells us that the array has 4 values, which indexes they have and what is inside them. It even tells us the data type and how many characters it has. var_dump(); will help us a great deal from here on and you should use it every time when you aren't sure how the data your are currently working with looks on the inside.

Challenges

This was a rather short chapter but we will be working with arrays from here on out. Make sure you take your time to play around with them before moving on.

Easy

Create your own Array and throw it into the var_dump() function. See what happens when you change things inside of it.

Medium

Create an associative array and throw it into the var_dump() function. Play around again until you get the hang of it.

Hard

See what happens if you save an array inside an array and var_dump() the result. You might be surprised. :)

Chapter 5: How to work with Strings

In Chapter 2 we covered the Data Type String that is used to save a bunch of Characters together. In this chapter we will learn how to manipulate them to our liking with different PHP functions. We will learn how to split them, change them, glue them together and even how to search and replace parts of them.

The . Operator

The . operator is used to chain strings together. I could try to explain this with a really long text or just show it to you and save us the trouble. It is a lot easier to understand after you see it.

```
<?php

// Outputs My name is Thomas

echo 'My Name is ' . ' Thomas';


// Outputs My name is Thomas
```

```
$text = 'My name is ';
$name = 'Thomas';

echo $text . $name;
```

As you can see it is quite simple. Keep in mind that those strings get glued together right where the . operator is. You will have to add the spaces yourself in between. Still gets me sometimes after all these years!

Using Variables inside Strings

In the above examples of the . Operator we only glued pure strings and pure variables together but did not touch on using a variable in the middle of a string. While it is possible to solve that problem with the solution above there is a better way. Take the following as an example.

```
<?php

// Outputs My name is Thomas

$name = ‘Thomas’;

echo 'My Name is ' . $name;


// Using the Variable inside a String instead

$name = 'Thomas';

echo “My Name is {$name}”;
```

As you can see the second options looks a lot cleaner and is less error prone. There are a few things to consider however. First of all you absolutely have to use double quotes “” and never single quotes ‘’ for this to work. Second the curly braces {} are optional but do help with 2 things. They make it easier to read since the variable is identifiable much easier and they help PHP distinguish between a normal word and a variable name inside a string. In our case here that doesn’t matter since there is a space between the variable and the rest of the string but what happens if we have something like this:

```
<?php

// Outputs She is beautiful

$adjective = ‘beautiful’;

echo “She is {$adjective}”

// But what about if she sings beautifully?

echo “She sings {$adjective}ly”;
```

If you look at the second echo statement then php wouldn’t know that the variable is only $adjective and not $adjectively without the brackets and that is why we use them whenever we can! :)

The implode() function

implode() has the purpose of turning an array into a string. It does so by creating an empty string and then appends each array value to it one after another. The indexes will be ignored.

It takes 2 arguments. The first one is the so called glue and the second one is the array that we want to implode.

The glue is a string that gets put between each array value while it gets attached to the string. It can be absolutely anything. A character, a dot or semicolon. a whole sentence or just a number. Even a whitespace works. If that sounds confusing just take a look at the example below and it should be easier to understand.

```
<?php

// Our Array

$fruits = array();

$fruits[] = 'Apple';
$fruits[] = 'Banana';
$fruits[] = 'Kiwi';


/*
 * If we var_dump() this we would get
 *
   * array(3) {
   *  [0]=> string(5) "Apple"
   *  [1]=> string(6) "Banana"
   *  [2]=> string(4) "Kiwi"
   * }
 *
   */


// Our String

$string = implode('_', $fruits);


// Apple_Banana_Kiwi

echo $string;
```

The explode() function

explode() is the opposite of implode(). It takes a string and turns it into an array. The string gets split by the delimiter of your choosing. Let's see how this works!

```
<?php

// Our string (Who doesn't love Lorem Ipsum?)

$string = 'Lorem ipsum dolor sit amet';


// Let's explode it per whitespace

$parts = explode(' ', $string);


// var_dump() it to see how it looks

var_dump($parts);
```

```
/*
 * array(5) {
   *   [0]=> string(5) "Lorem"
   *   [1]=> string(5) "ipsum"
   *   [2]=> string(5) "dolor"
   *   [3]=> string(3) "sit"
   *   [4]=> string(4) "amet"
 * }
 */
```

The str_replace() function

str_replace() is used to replace so called needles inside the so called haystack. The needles are a piece of text, or some characters that we want to change and the haystack is the string we are searching in.

The best example would be the BB Code that most Forums have. Have you ever seen or used [b]BOLD TEXT[/b] in a forum post? That is probably implemented by str_replace() or one of its brethren.

```php
<?php

// Our text

$text = 'I [b]love[/b] swimming!';


// Let's translate that into HTML

$text = str_replace('[b]', '<strong>', $text);
$text = str_replace('[/b]', '</strong>', $text);


// Our text is now: "I <strong>love</strong> swimming!"

echo $text;
```

The example above is not the best implementation but it is sufficient as an example in this case.

Challenges

This pretty much wraps up our chapter. There is a lot more you can do with strings and many more functions that we did not cover yet. However since they are so rarely used you can always look them up if there is a problem you cannot solve with your current arsenal. Time for some challenges!

Easy

Create a Hello World Example. Save Hello in one variable and World in another. Output them both with the same echo command.

Medium

Create a variable with the following String: “I love to eat”

Explode the string into an array, and attach your favorite food to it. After that implode the array and echo out your finished string.

Hard

Create your own BB Code for cursive. Search a string for [i] and [/i] and replace these with <em> and </em>. Output the result.

Practical Lesson I: Dynamic Content & Navigation

Time for some real world practice! If you have never used any kind of backend language like PHP before then you are probably used to creating a new HTML file for every single page. There is no reusability of anything and if you have to change something like the navigation you have to do so in every single file.

Now imagine that you could change the content of a page with a single variable which you can add to your web address and only need a single file for all your pages! Everything on the page stays the same but the content in the middle changes every time. You have seen this done on a million websites before. Even whole systems like Wordpress use this in a more advanced form that people like to call “pretty URL’s”.

So let’s get to it! Go to your code folder that is included with the book and open “practice-1”.

An Overview

Inside the folder you should see a single file called index.php. This is the file we will be working with. If you open it via your web server you will see a very simple 3 column layout that was created with Bootstrap. If you don’t know what Bootstrap is, don’t worry!

The page contains a header, left sidebar, content area, right sidebar and a footer. We want everything but the content area to be static, which means, unchanging and only switch around whatever is displayed in the middle of the page based on what navigation link is clicked on the left sidebar.

Currently there are 3 links. Home, About and Contact. If you click on them however nothing changes. So let’s get to work!

The $_GET Array

PHP comes with many predefined arrays that all contain useful data. Some contain things like the users IP Address, others cookie information or data that got submitted in forms. These are called the Superglobals and we will talk about why at a later time. For now we only need to know that they are all associative arrays which we just heard about.

Before we explain anything else let’s go ahead and var_dump() out the $_GET variable into our content section. To do so just open the index.php file with your editor and add the following code under the “Your Code here” comment inside.

CODE COMING SOON

If you refresh your page now you should see that the $_GET array is empty. To change that go ahead and click on any of the navigation links. Take a close look at the content of the $_GET variable and compare it with the URL in your browser. Do you notice something?