

Tutorial based in Building Web Application with PHP course on Coursera.

Santiago Hidalgo Ocampo.

Preliminaries:

The code of this project can be found in the following link, you can clone it.

There are several comments that will help you understand its structure much more.

Git: https://github.com/sanhidalgoo/binary-unary_Calculator

Also, in its README you can see how to articulate this project with XAMPP, a stack of technologies that allows me to run a PHP server on your own computer. The installation of this project will be explained for an optional Linux based Debian system, In my case I installed it on a Linux mint 19.

How to install MAMP (MAC): <https://documentation.mamp.info/en/MAMP-Mac/Installation/>

Intall XAMPP (linux or windows): <https://www.apachefriends.org/es/download.html>

File structure:

In the gitHub we can emphasize in 4 files. The index.php file contains the html and php code of the login (main page), this in turn has directly associated the styles.css file which contains the code that shapes and sets the structure in a user-friendly way html. The calculator.php file contains the html and php code of the binary and unary calculator, this in turn has associated the css code in the styles_cal.css file.

The “img” directory contains the background images used in the project as well as the logo of the basic mathematical operations

TUTORIAL

This tutorial will try to teach the basics of building web pages with php. For this purpose, we will build a very simple login and a basic calculator with binary and unary operations as shown below.

There are many ways and technologies available to create web pages as sophisticated as we see today, however, for this course three tools will be used that are fundamental in the learning (or at least in its bases) of the construction of the web, these are: HTML, CSS and PHP.

HTML

HTML consists of a series of short codes typed into a text-file by the site author — these are the tags. The text is then saved as a html file, and viewed through a browser, like *Internet Explorer* or *Netscape Navigator*. This browser reads the file and translates the text into a visible form, hopefully rendering the page as the author had intended. Writing your own HTML entails using tags correctly to create your vision. You can use anything from a rudimentary text-editor to a powerful graphical editor to create HTML pages. (Shannon, Rose. 2012).

In essence, HTML, allows me to create the layout of the information that I want to put on my web page. This technology does not allow me to give color, shape or movement. Everything is dead (for the moment)

Everything works by tags, within these the main one is `<html>`, and within this there are in turn two important components, the `<head>` and the `<body>`.

The `<head>` element is a container for metadata (data about data) and is placed between the `<html>` tag and the `<body>` tag. HTML metadata is data about the HTML document. Metadata is not displayed. Metadata typically define the document title, character set, styles, scripts, and other meta information. (W3Schools, s.f.)

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Calculator</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
  <link rel='stylesheet' href='styles_cal.css'>
</head>
```

In our example we have this metadata. We also have a "link" tag that refers to the address of a file that has css code (which we will see later).

The `<body>` tag defines the document's body. The `<body>` element contains all the contents of an HTML document, such as headings, paragraphs, images, hyperlinks, tables, lists, etc. (There can only be one `<body>` element in an HTML document).

You can consult the large number of html tags that exist. however, I want to focus on a tag that is important for reading data, and failing that for processing these with PHP.

The `<form>` element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc. The `<form>` element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc. Here are some types of inputs:

Type	Description
<code><input type="text"></code>	Displays a single-line text input field
<code><input type="radio"></code>	Displays a radio button (for selecting one of many choices)
<code><input type="checkbox"></code>	Displays a checkbox (for selecting zero or more of many choices)
<code><input type="submit"></code>	Displays a submit button (for submitting the form)
<code><input type="button"></code>	Displays a clickable button

Image taken from: https://www.w3schools.com/html/html_forms.asp

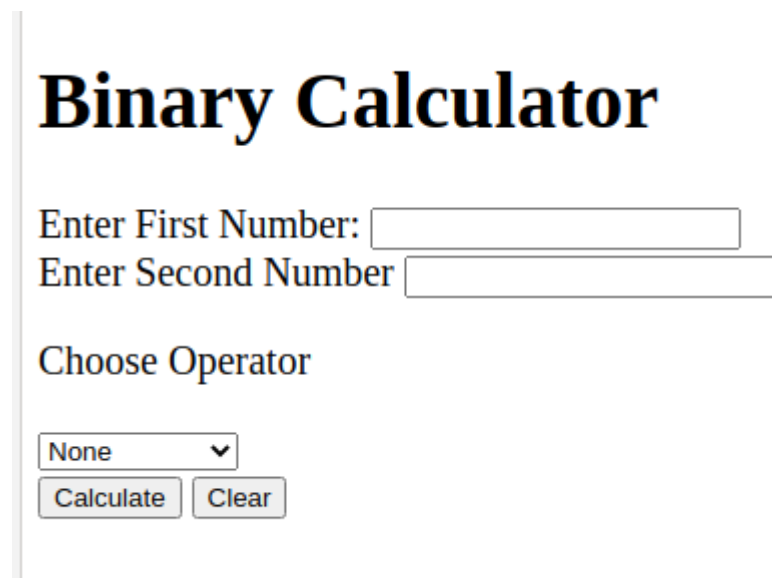
```
<form method="POST">
  <label for="Firs number">Enter First Number:</label>
  <input type="number" name="f_num" class="form-control">
  <br>
  <label for="Second number:">Enter Second Number</label>
  <input type="number" name="s_num" class="form-control">
  <br>
  <p>Choose Operator</p>
  <select name="B-operation" class="form-control">
    <option>None</option>
    <option>Addition</option>
    <option>Subtraction</option>
    <option>Multiplication</option>
    <option>Division</option>
  </select>
  <br>
  <input type="submit" name="submit" value="Calculate" class="btn btn-success">
  <input type="reset" value="Clear" class="btn btn-danger">
</form>
```

As we can see, we have two inputs of type number, which allows the user to enter only numbers in an input field. We must take into account the field that we assign to name because this value will be used to obtain the data entered by the user from our php code

In the same way we have a select tag named B-operation, this value is important, because from our php code we will obtain the user's choice to do the corresponding arithmetic operation

Finally, it is important to emphasize the functionality of the input type submit, because this element allows me to send the data via POST (since the method that we specified to the <form> tag at the beginning specifies it, don't worry , we will see this concept later) to be processed from our programming language

The following code above generates the following output (without any CSS configuration which we will see later)



The image shows a web form titled "Binary Calculator" in a large, bold, black serif font. Below the title, there are two input fields for numbers. The first is labeled "Enter First Number:" and the second is labeled "Enter Second Number". Both labels are in a blue serif font. Below these is a dropdown menu labeled "Choose Operator" in a blue serif font, with "None" selected and a downward arrow. At the bottom are two buttons: "Calculate" and "Clear", both in a light gray box with a thin border.

Binary Calculator

Enter First Number:

Enter Second Number

Choose Operator

None ▼

On the other hand, the code is very similar for the unary calculator part:

```

<div class="unary-calculator">
  
  <h1><strong>Unary Calculator</strong></h1>
  <form method="POST">
    <br>
    <label for="number">Enter Number:</label>
    <input type="number" name="number" class="form-control">
    <br>
    <p>Choose Operator</p>
    <select name="U-operation" class="form-control">
      <option>None</option>
      <option>Factorial</option>
      <option>Is_Prime?</option>
      <option>Square root</option>
      <option>Even/odd number?</option>
    </select>
    <br>
    <input type="submit" name="usubmit" value="Calculate" class="btn btn-success">
    <input type="reset" value="Clear" class="btn btn-danger">
  </form>
  <br>
  <p><strong>The Answer is: </strong> </p>
  <div class="Answer">

```

We can see that we have a h1 label (used for titles) labels (Space for a text box) and the form structure that contains input labels and selectors inside. In fact, comparing this code snippet with the binary calculator is much simpler. I also want to mention that the img tag allows me to place images in my text. At the moment, we can see this without the CSS settings:



Unary Calculator

Enter Number:

Choose Operator

The Answer is:

CSS

CSS stands for Cascading Style Sheets with an emphasis placed on “Style.” While HTML is used to structure a web document (defining things like headlines and paragraphs, and allowing you to embed images, video, and other media), CSS comes through and specifies your document’s style—page layouts, colors, and fonts are all determined with CSS. (Scott Morris, s.f).

I don't want to focus too much on this component, but essentially in CSS we specify a series of parameters in a selector that can be a class or an id. The class can be used by any tag in the html code and an id can only be used by a single vex. In order to apply the css settings, we must place the word "class" or "id" after the opening tag of the html, match it to the name of the selector.

I want to bring some examples that explain how CSS works. We will not go through all the settings to each of the tags because there are too many to address in this tutorial, however I want to show a few that represent the basic concept. You can play with this according to your creativity, you can use your favorite colors and bring out your artistic style.

```
.binary-calculator .login-image {  
  width: 100px;  
  height: 100px;  
  border-radius: 50%;  
  position: absolute;  
  top: -50px;  
  left: calc(50% - 50px);  
}
```

This selector allows me to configure the tag of my login.php file that is inside a div that contains a class value called .login-calculator and an image tag that has a class with value .login-image. This configuration is simple, I determine the width and height of this element (each with 100 pixels), we shape it as a button, specifying that its border had a radius of 50% and we place it specifically in the div tag. Note: div tags are usually represented as containers or boxes in which I want to place a specific structure.

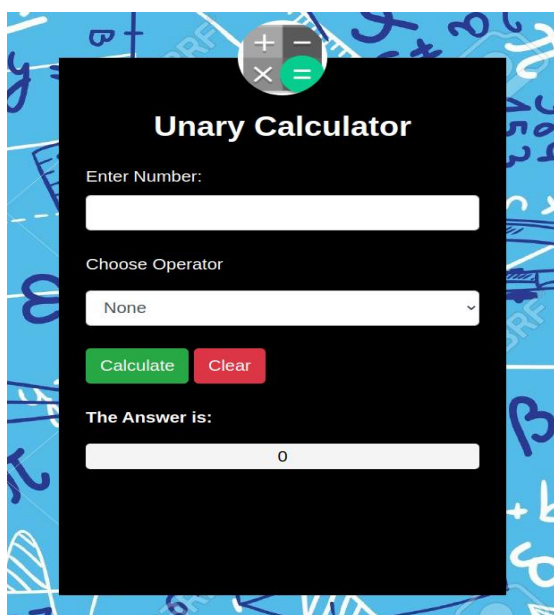


From the raw image that I show previously we arrive at something more aesthetic

```
.unary-calculator{  
  width: 500px;  
  height: 80%;  
  color: #fff;  
  background: black;  
  position: relative;  
  box-sizing: border-box;  
  top: 10%;  
  left: 55%;  
  padding: 70px 30px;  
}
```

Giving another example of how to configure a specific tag I want to highlight this class that parameterizes the way my main structure of the unary calculator should look like.

We give this structure a specific height and width, we say that the color of the letter inside this element must be white, we define its background to be black and we also position this component in a specific space of my model, in this case it we configure it so that it is exactly to the right and at the same height of the binary calculator



With these configurations seen we have this basic structure, (the image and the black background, do not take into account the input and the text)

The other elements that are inside, have much simpler and easier to understand settings, you can change the style of the font, the size etc. Please review these elements by associating them with the classes referenced in the html code.

For more information consult the following link:
https://www.w3schools.com/whatis/whatis_css.asp

HTTP and PHP

PHP is the universal language of the web applications. It was created by Rasmus lerdof. To understand PHP you must understand the HTTP protocol at least in a general way.

You can check the styles.css (index.php) and styles_cal.css (calculator.php) files to check the parameters given to the selectors

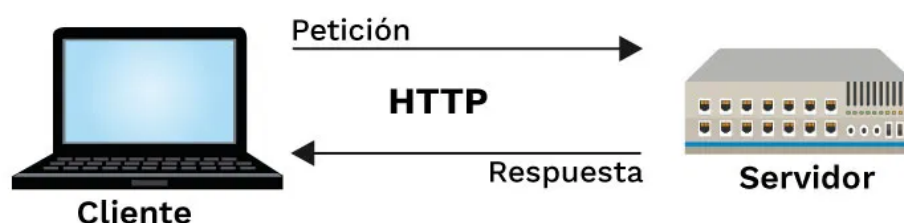


Image taken from: https://www.google.com/url?sa=i&url=https%3A%2F%2Funprogramador.com%2Fcrear-interfaz-retrofit2%2F&psig=AOvVaw3Lv_ZhNkC9fTGQ2H6ukFpN&ust=1597293907933000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCNCNIdjtIoScFQAAAAAdAAA AABAV

Through the http protocol I can make requests with various parameters. The most important for these starts are GET and POST. Next we will see when to use one or the other:

Rules of the POST/GET Choice

- POST is used when data is being created or modified.
- GET is used when you are reading or searching things.
- Web search spiders will follow GET URLs but generally not POST URLs.
- GET URLs should be “idempotent” - the same URL should give the “same thing” each time you access it.
- GET has an upper limit of the number of bytes of parameters and values (think about 2K).

Image taken from resources of Building Web Applications with PHP on coursera.

Now, let's talk about PHP. This language is interpreted and weakly typed, its syntax is very simple and has elements very similar to C-based languages. If you know a programming language like C, C++ or Java. It will be very easy for you to learn this language.

```
<?php

$stored_hash = "e34a60e8c38f3cc26930e8b04c51ac1741225342dd168c41f33f310fb5b1d65e"; //the password
$salt = 'XyZzy12*_*';

$message = false; // If we have no POST data

// Check to see if we have some POST data, if we do process it
if (isset($_POST['username']) && isset($_POST['password'])) {
    $check = hash('sha256', $_POST['password'] . $salt);
    if ($check == $stored_hash) { //Redirect to game.php
        header("Location: calculator.php?username=" . urlencode($_POST['username']));
        return;
    } else {
        $message = "Incorrect password";
    }
}

?>
```

Let's explain something simple about this language. In our HTML we have two input tags that receive the username and password, thus simulating a login. Variables in PHP always start with the dollar sign, it is a rule that we cannot violate, in our case we have \$ stored_hash and \$ salt as a string of Strings, which can be represented with single or double quotes. Now, there are global variables that are essential to take advantage of this programming language. These variables are \$ _GET AND \$ _POST. As we can see, the POST variable allows me to know what parameter was given by the user in our input "username" and "password" (which we

specify with HTML), when we know that there is a piece of data we begin to do the redirection process (acceptance of the login or authentication successful) with a conditional structure. It should be noted that the header (Location: calculator.php = username = USERNAME) instruction redirects me to the specified location while making a GET request.

Now we are going to see another structure such as the switch-case.

```
if (isset($_POST['submit'])) {
    $number = $_POST['number'];
    $operator = $_POST['U-operation'];
    $uResult = "";
    switch ($operator) {
        case 'None':
            $uResult = "Please choose operation";
            break;

        case "Factorial":
            if (is_numeric($number)) {
                $uResult = "The factorial of $number is: " . factorial($number);
            }
            break;

        case "Is Prime?":
            $uResult = isPrime($number) ? "$number is prime" : "$number is not prime";
            break;

        case "Square root":
            $uResult = "Square root of $number is " . sqrt($number);
            break;

        case "Even/odd number?":
            $uResult = $number % 2 == 0 ? "$number is even" : "$number is odd";
            break;
    }
}
```

How it can be observed depending on the operation that the user wants to perform, a basic mathematical operation will be executed, storing its result in a variable that will then be displayed within a label as follows:

```
<br>
<p><strong>The Answer is: </strong> </p>
<div class="Answer">
    <?= htmlentities($uResult) ?>
</div>
```

The variable is displayed within htmlentities to prevent malicious users from entering code in the form of "plain text". This is a good practice and we avoid security problems.

```
// Returns the factorial of a number
function factorial($num)
{
    $factorial = 1;
    for ($i = 1; $i <= $num; $i++) {
        $factorial = $factorial * $i;
    }
    return $factorial;
}
```

On the other hand, cycles are a very common structure when programming, as we can see, and assuming you know a programming language, the syntax of this structure is very similar to other languages such as Java and C ++. This should not be a problem.

Structure and a simple approach to MVC

One of the simplest approaches to writing neat and readable code (good programming practices) is the Model-View-Controller (MVC). For these simple projects the suggestion is that our PHP code goes in the first part of the file with the .php extension, and in the lower part everything that has to do with the layout goes, that is, our HTML code. In other words, our programming code and our layout should be separated by an imaginary line. On the other hand, everything that has to do with CSS goes in a separate file, and to apply it to our layout it must be referenced with a link tag in our HTML head.

References:

- What is CSS?. (s.f). Obtained from: <https://skillcrush.com/blog/css/>
- HTML. (s.f.). Obtained from: <https://www.w3schools.com/html/>