Jaeseung Lee

CS408

Final Project - term paper

Due Tuesday, June 6, 2017

Final Project: PHP

Learning one language within few weeks was challenging, especially, since it was scripting language. I didn’t have any knowledge about scripting language and it was big first step on it, with one of the most popular scripting languages, PHP. Despite the short learning period, I was able to obtain lots of knowledge about PHP, and me and my partner was able to come out with some cool program. Before I actually started to look up for specific features of PHP, and all the detail tutorials for PHP, it was unarguably important for me to understand the programming language versus scripting language. And after weeks of researching on this topic, I have concluded that those two are not clearly separated as many people thought, but only main difference between those two were the compiler – For the case of scripting language, it reads each line of code, interpret, and decide what to do each line of code, and it tends to be vastly simpler compared to the programming language. However, for the case of Programming language, it requires to compile literally every time you make a change to your code, meaning it takes longer time. (However, this line got significantly faded in nowadays, due to the remarkable technology development) Another remarkably interesting software I found during the researching process was Nessus, the software that scans vulnerability of other systems for hacking, which I am fully motivated to check it out later.

While I was in the process of preparing the presentation for me and my partner, doing this final project, I was able to not only how to write PHP code, but also learned about the detail of historical facts about PHP. The very first question from the people who absolutely know nothing about PHP would be: “What is PHP? Why is it created, by who, and where do we use it?” As I have mentioned previously, PHP is the one of the scripting language that was developed by Rasmus Lerdorf in 1995. The first PHP was a simple set of Common Gateway Interface (CGI) binary written in C programming language. He originally developed it for personal homepage (PHP) tools, such as tracking his personal info. It automatically the form of variables, it has HTML embedded syntax. The very first PHP, PHP 2.0, was the first release that could actually be characterized as PHP, and in PHP 4.1, it introduced the concept of super global variables, PHP 5.3 started to support namespaces, late static binding, garbage collection for circular references, and improved Windows and SQL support as well. And finally, the current version, PHP 7.0/7.1, incorporates more modern OOP principles, void return types, and catching multiple exception types.

There is no such a programming language that has perfect and flawless advantages with no disadvantages. Then for the case of PHP, what would be some advantages? Some advantages are, having the garbage collection, regular expression support, and it also runs on many operating system. The most well-known advantage is that it is easy to learn – this means it has fairly good readability (setting the default value in the function argument is also part of this). For the writability, it supports abstraction, since PHP is aiming for simplicity, the uniformity is unarguably good.

Comparing PHP with Java and C++, many similar features were found, but there were also great number of features that I have never seen before – one of them were the syntax part, such as loosely type, where variables don’t need to declare a type, and everything other than variables are not case-sensitive, echo instead of print for better efficiency for the execution time. For data type, while everything else, such as integers, floats... and etcetera, were really similar, resource was the special variable that was really interesting to me, which holds a reference to an external source. Associative Array was also one of the remarkable features that PHP had, which is an array with named keys (string as index). Another impressive feature was setting the default value in the function argument. And other features, such as OOP supports, constructor and destructor, exception handling, are really similar as Java.

Because it is always good to have solid pseudo-code for any programming projects, we have made the blueprint of our project first. Completing it was pretty much straight forward – In the main, we load LoadAccount() and then LoggedOnUI() in the while loop, meaning in html, load the login screen for username and password and option to create account. In LoggedOnUI(), the user will be able to perform: CheckBalance, CheckHistory, Deposit/Withdraw, Logout, DeleteAccount. And in LoadAccount(), we’ll check if the account exist or not, and perform proper action depends on that.

Getting into our own PHP program, before we start to write code for our final project program, we were required to install XAMPP, which allows us to run our own web server. After XAMPP started to work properly, me and my partner has implemented online bank account, which provides panels to let the user login their bank account. If they don’t have online bank account, give them an option to create own account (three account options – Gold, Platinum, Unlimited), withdraw/deposit money, and an option to see the record of the withdraw/deposit summary. I also have to mention that we have used CSS, the style sheet language which provides nice default design, layout and variations. We have three php files in total – Home.php, NewAccount.php, Panel.php. For Home.php, it handles the very first login screen with the possible case of errors(where user doesn’t have account and tries to login, or ridiculously long id or password) For NewAccount, it handles creating the account for the new users. test\_input() function takes the user’s input and apply appropriate functions -- trim(),stripslashes(), and then take care of all the possible errors.(missing info, username already taken, etc.) For the Panel.php, we have three in total – Panel1 for the page that has login button, create account button, and Panel2 for deposit/withdraw buttons and other functions, with the exception handling case code(invalid deposit/withdraw amount, insufficient funds, and Panel3 for account history. Because my partner knew how to do html and css work, I was more focusing on the <?php?> codes.

Overall, it was definitely worthy project that gave me numerous lessons and infomration. As I previously mentioned, it was my first step into the scripting language. Also, me and my partner weren’t only learning how to code PHP codes, but also the detail of historical facts, which allowed us to understand better about the questions we had. (Such as the main motivation of creating PHP and etc.) I also believe our program was challenging enough that, once someone, who is trying to learn PHP, fully understand how our code works, they will be able to make any projects by modifying or changing little bit from our code. Many students might remember implementing the *Car class* in OOP features, which emphasize on the encapsulation concept I OOP programming. (Where all the variables should be private and limit the direct access, and use getter and setter for the access) Just like practicing the *Car class* example for OOP learning, practicing and understanding our project, and being able to recognize the css and html for UI, how those get applied to the actual execution, will be really informative to anyone who are motivated to learn PHP in a short period of time.

I have also learned that PHP is at everywhere – no other language is easy as PHP. The languages such as Java, C++, Ruby, Python requires the knowledge of OOP and frameworks and etc. On the every web hosting server, there is PHP, and provides abundant extensions.