Web Programming Coursework 2

By Daniyar Nazarbayev, H00204990.

First and foremost, I would like to say that the coursework is far from complete. It is my fault, but I still decided to submit what I got.

What I used:

- I used XAMPP as the server.
- Sublime as the editor.
- PHP as my language.
- MVC architecture is present.
- ps: do not forget to run the sql script before running the website.

Server side.

More or less, the server side logic is 95% complete.

You can ask questions.

You can answer questions as long they are not solved and you are not the user who asked the question.

Both asking and answering requires you to be logged in.

Finally, you can view questions. There is a dropdown with 3 options (all, solved, unsolved) and a textbox.

I have a html page called "question" that should take in a GET value - *question.php?id=value*.

Currently, the problem is that the server side logic does not take in the account scenarios when an incorrect input is given.

I used the MVC architecture. Divided my files into folders view, model, controller, lib.

Model.

Originally, I wanted to have 3 tables – one for the questions, one for the answers and one mapping them together. But the more I thought about it they less it made sense. One answer cannot map to multiple questions, so in the end I figured out that the relationship is one to many. I created a foreign key for the relationship (in the answers table).

I added a users table and made a relationship with both the questions table and the answers table, by creating a foreign key in each. Again the relationship is one to many.

I also added a boolean field to questions (solved) and answers (correct). In all honesty, it might be possible to have everything working with only one of these present, but my SQL skills are not that good.

To wrap things up, altogether there are 3 tables: users, questions, answers.

AJAX and JSON.

As I said previously, I implemented a search on a server side. I extended it and made AJAX run whenever input is changed (used oninput event). AJAX would send a request to a PHP file, with the contents of the input, and php would use the already existing search function to find all the questions that have that word in it. The output of that function is an array and I had to use JSON. But it was not all that simple.

Associative arrays are very common in PHP – they are the default array used to extract from the database. Associative arrays have keys instead of indexes, but JSON only allows indexed arrays, and only objects can have keys. In the end, I gave up and just made another array that would copy only the question field from the array. I encoded it, echoed it, parsed in javascript, changed the contents of the inner html of a div.

Security.

I implemented three methods for security.

1) I used prepared SQL statements.

2) I striped tags of off every POST, GET that I receive.

3) I use password hash and password verify for the passwords.

View.

I practically skipped the view altogether. As hard as I wanted to understand bootstrap, in the end I gave up and started working on the server side. Bootstrap is a very extensive framework and it takes some time to wrap your head around it.

PHP and me.

To be honest, during my first year I really hated PHP.

First of all is that it has tons of functions, half of which are deprecated. They are not all that well documented either, or at least not for a first year student. There are at least 3 different ways to connect to a database for example, and each source on the internet will promote one of them and will discard all the others. I guess that is what I mean by poorly documented, since some of these tutorials are really outdated. The browsers outright do not allow you to use some of the sql methods anymore since they are unsecure, but php still keeps them around.

I have not heard of an IDE for PHP. No automatic indentation. Some editors provide highlighting, but not everything is properly highlighted. Eclipse's support for PHP is very poor too.

Since I came from the java's world, the lack of defining the type of a variable or the return type of a function felt very limiting. Instead of using .equals() to compare strings, you use ==, while to check for whether the objects are form the same memory addresses, you have to use ===. If you want to reference to another object, you have to explicitly state that (with &). While something like array1=array2 is just assigning array2 to array 1, compared to other languages where you have to make a copy of the object and assign it. Associative arrays instead of indexed ones when extracting from the database.

The biggest difference between java and php is the variable scope though. There are 2 types of scopes in PHP, global and local. Local is defined in a function, global is defined in anywhere but a function. Neither can communicate to each other by default. You can nest a variable deep inside multiple layers of blocks (define it there) and that variable will still be accessible outside those blocks, but if you define a variable and then right on the next line define a function and will try to call that variable, then it will not work.

Functions can only take parameters, basically they are pure functions. You can access the variable with a global keyword though, but it is not a recommended practice (since the variable can be changed).

Objectively, PHP is much simpler than Java, but that simplicity comes at a cost of less control. If I had my first language as PHP, I would definitely be weaker in some areas, since there is a lot of implicit typing in php.

The biggest offender in PHP though is that as you write code, it starts to look really messy. Especially when printing html and “ (quotations).

That said, I decided to pick it nevertheless for my second coursework. First of all, because it, more or less has its roots in C’s syntax. Semicolons and curly brackets make me feel right at home.

Second of all, I learned a lot from JSPs and Servlets. When I first looked at JSP, it had a very weird syntax for its if statements and loops. After the starting curly bracket, a closing scriplet is applied, and before the ending bracket a starting scriplet is applied. In between is the normal html code. It was very unusual at first, but it was very efficient. No more would I have to print every line of html code that I would want to be generated on the server side. The best part of that is that this method was not exclusive to JSP.

Later on I found out about another good method of printing HTML – heredocs.

As for servlets, I learned about MVC from them. Writing java code in HTML is absurd. Writing html code in java is absurd. MVC basically divides the code into sections of similar type. If the View wants to change the Model, it has to go through the Controller. If it just wants to extract the data from the Model, it can access it directly. Any input from the user is sent to the Controller.

As I was going through the lectures, I saw a screenshot. It was about MVC. How one should categorize his files into folders. In that screenshot, there was a folder named lib. One could use include or require in his php, which is similar to java’s import. All of that to avoid repetition.

All of that helped me write much better, clearer, more readable, less tedious code.

JSP/Servlet and PHP.

There is difference I noticed so far between these two – scope. Servlet has 3 scopes: request, session, application. In addition, JSP’s Bean has a page scope. Request would have a lifespan of a one HTTP request – refresh a page and the variable that was assigned to it is gone. The page scope should survive a refresh, but no more than that. Session and application are global, but session expires after some time while application does not.

PHP only has the session scope. Personally, I found the usage of the request very useful for showing errors to the user. Not having that was problematic, but in the end I figured out a way. It involves using a session. Then setting a value to a session variable, in my case named error, which is also an array, so I could store multiple error messages. After the redirect, I print out all the values of the session variable and then unset it.