Holly Robertson

WEBD236

Midterm

March 23, 2019

Question 1

> Christopher Alexander, a structural architect, described patterns in his 1977 book as:
>
> "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice" (Pitt, 2012).

Christopher wasn't speaking about software development here, rather he was describing patterns followed in building physical structures and towns, but his description fits perfectly into the field of designing software. Pattern theory, largely influenced by Christopher Alexander himself, is a universal way to process problems using a series of proven steps to reach a clear and concise conclusion. Though the final outcomes of each challenge will be different, the paths followed to get there will contain an overall similarity to each other – and that's the point. Patterns don't detail how to solve a problem explicitly, but rather how to think about solving a problem. Since the injection of patterns into software development, numerous different structures have been created to help solve each new problem. In web development, the Model-View-Controller (MVC) has been one of the most widely used patterns since its creation in the late 1970's by Trygve Reenskaug. This paper will discuss what the Model-View-Controller is and why it is needed in software development, examples of how it is used in real-world applications, and its shortcomings.

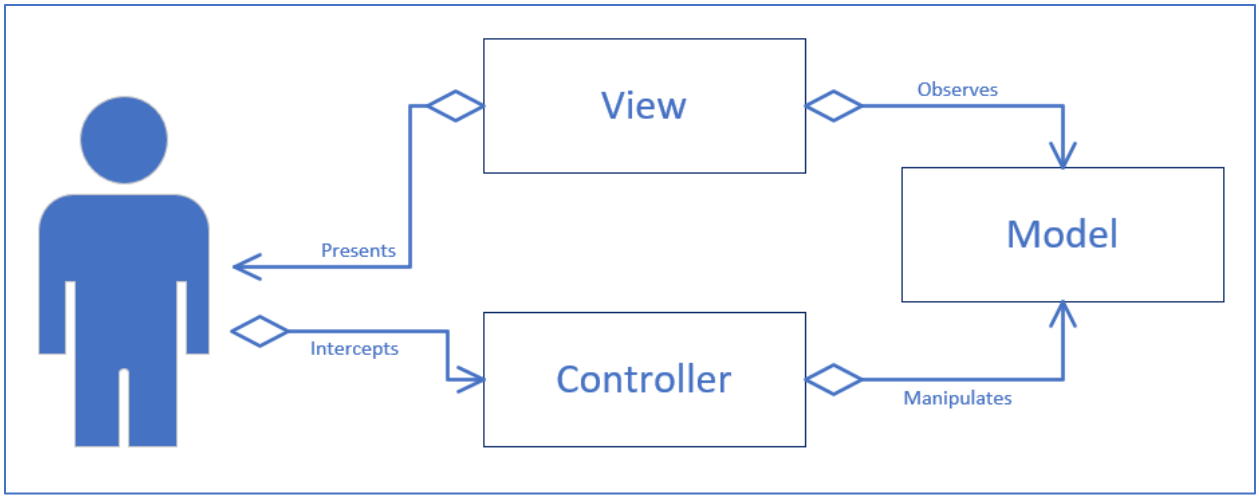
There are generally two types of websites: static and dynamic. Static sites depend on client-side technology – browsers – there is no back-end logic to develop. These sites don't reach into a database and pull information, they display the code of a page to the user. With dynamic sites, you have a more

complicated approach to development. There is usually a database to configure, manage, and update. Also, your pages usually change per user, whether they're logged in or logged out, what they have in their shopping cart, or if they have sent an email via a contact page. Though the complexity of the code changes with static vs dynamic, the issue of designing the application remains the same. How you structure your code to make it easier for continued development, troubleshoot issues, and easy to understand? MVC attempts to solve that dilemma. It decouples the various components of an application, making each section almost autonomous, while seamlessly working together to give the user and developer a structured and organized product. The Model-View-Controller pattern has three parts:

Model – "Where all the business logic is kept" (Pitt, 2012). This section is responsible for representing and persisting application data. It receives input, mainly user input, from the controller and usually sends updates to the database.

View – Is what the user sees. It's responsible for rendering data from the model to the user. It is the final output that the user sees. It is the "display of the application's current state" (Krasner & Pope, 1988).

Controller – The middle ground between the model and the view. It intercepts user interaction and converts it into instructions for the model and view sections. It is also responsible for updating the model. The general concept: The user inputs data, the controller takes that data, updates the model, the model tells the view it's been updated, the view pulls the new data, and then pushes the view back to the user.



Source: Holly Robertson

The Model-View-Controller pattern has only gained in popularity over the last several years with big companies, such as Microsoft, adapting its structure into their applications for web development. ASP.NET MVC "was a fresh of breath air for most Microsoft web developers. It took us into a new web era […] bringing a lot of developers behind MVC 110%" (Danylko, 2016). Many of the web's most-visited websites were designed around this ASP.NET MVC pattern: StackOverflow, Microsoft, GoDaddy, and Dell. These sites require teams of developers who need to actively manage, monitor, update, and troubleshoot 24 hours, 7 days a week. By using the MVC architecture, simultaneous development is possible. The pattern abstracts the web application, allowing front-end developers to work on the view without touching the database (model) code. Also, permitting the back-end developers to migrate a complete database, without the front-end ever being affected. By extracting these sections of a web application through the model-view-controller pattern, developers are not restrained to wait on other developers or teams to manage their own code or implement new features into a website.

Though the Model-View-Controller is widely popular throughout web development, it isn't without its flaws. In real-world development, it's not always clear which section a file should belong too. For example, in an invoice creation application, date formatting is a necessary and common task. The

model would handle creating the invoice, while the view would display the invoice with the formatted date, but where does the code to format the data go? An argument could be made that manipulating information from the database could still fall under the model's responsibility, but another argument could fight that formatting the date is for the benefit of the user's view, so it needs to go in the view. But why should the model be responsible for formatting data that is related to the front-end, and the view doesn't need to understand what it is displaying to the user. This file doesn't really fit into the controller either because it's not manipulating user-input to the model, it's manipulating data from the model to the user. Unfortunately, most of the time these types of ambiguous files will get stuck in the controller section. "After thousands of lines of code, you end up with a bunch of overweight controllers, ready to burst and impossible to test" (Jacobs, 2016). Alternatives to the Model-View-Controller have been developed, but all equally flawed in its own way.

The fact that the Model-View-Controller pattern isn't perfect is precisely why we need the Model-View-Controller in web development. Nothing is perfect, and everything eventually fails, but without a good foundation, like the MVC pattern provides, nothing can be built.

Bibliography

Danylko, J. (2016, February 2016). *10 Reasons to Start Using ASP.NET MVC*. Retrieved March 23, 2019, from https://www.danylkoweb.com: https://www.danylkoweb.com/Blog/10-reasons-to-start-using-aspnet-mvc-E9

Jacobs, B. (2016, July 5). *What is Wrong with Model-View-Controller*. Retrieved March 23, 2019, from Cocoacasts: https://cocoacasts.com/what-is-wrong-with-model-view-controller

Krasner, G. E., & Pope, S. T. (1988). *A Description of the Model-View-Controller User Interface Paradign in the Smalltalk-80 System.* Mountain View: ParcPlace Systems. Retrieved March 23, 2019, from https://www.researchgate.net/profile/Stephen_Pope/publication/248825145_A_cookbook_for_using_the_model_-_view_controller_user_interface_paradigm_in_Smalltalk_-_80/links/5436c5f30cf2643ab9888926/A-cookbook-for-using-the-model-view-controller-user-interface

Pitt, C. (2012). *Pro PHP MVC.* New York, NY, United States: Springer Science & Business Media New York. Retrieved March 23, 2019

Holly Robertson

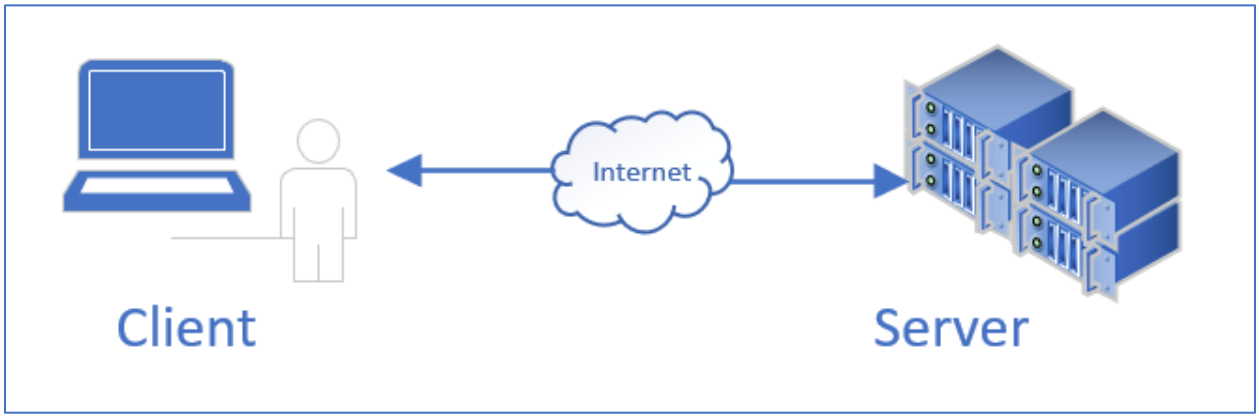
WEBD236

Midterm

March 23, 2019

Question 2

The Internet is vast and diverse, but it does follow some specific structural rules for how it communicates. The most popular is the client-server architecture, which is an “architecture of a computer network in which many clients (remote processors) request and receive service from a centralized server (host computer)” (The Editors of Encyclopaedia Britannica, 2019). A simplified definition: a user requests a website, through a browser (client), the host computer (server) listens for requests and processes them in the order received. The website data is sent back to the user and is rendered within their browser.



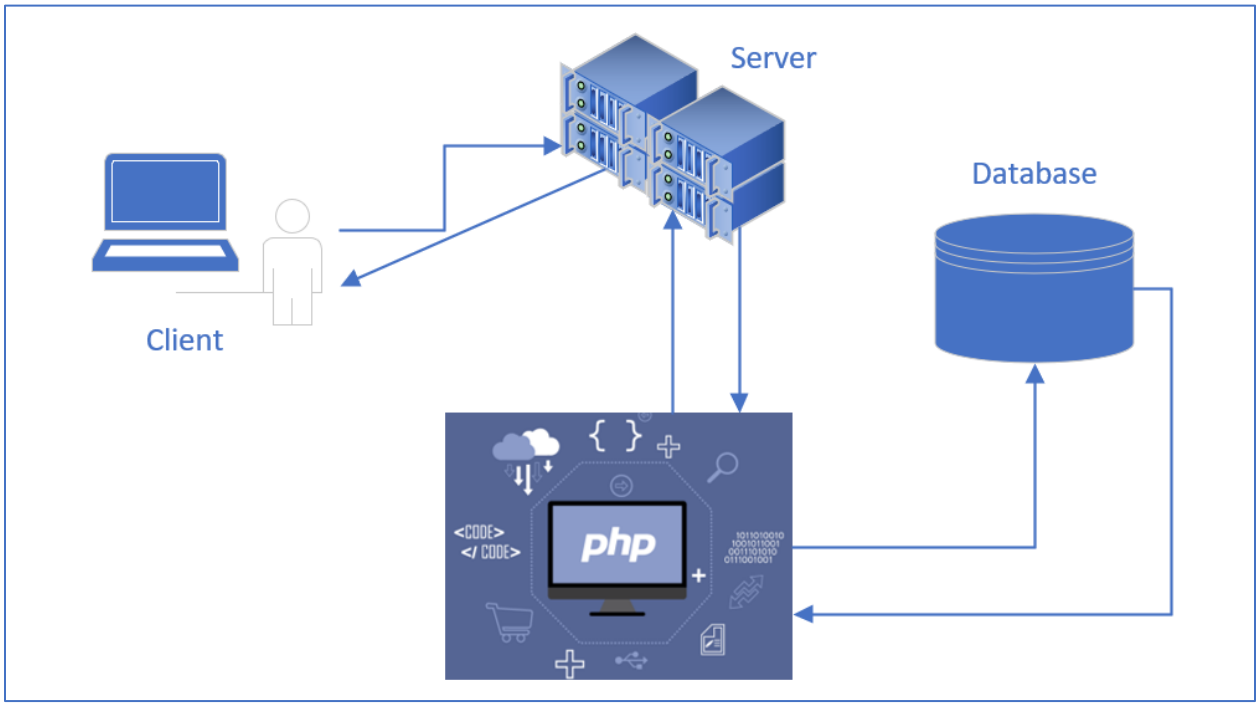
Source: Holly Robertson

There are different languages for client and server-side processing. HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript are examples of scripting languages that are processed on the client-side, within the browser. This means they don’t need an internet connection to be rendered on a user’s computer. PHP (Hypertext Processor) is a general-purpose programming language that is processed on the server-side and then sent back to the client, browser, usually in HTML

format. Developed by Rasmus Lerdorf in 1994, PHP is an open-sourced, widely-used “server scripting language and a powerful tool for making dynamic and interactive web pages” (W3 Schools, 2019). By integrating PHP in with HTML, CSS, and JavaScript you can design a highly-responsive and efficient web application.

PHP turns a static website into a dynamic site. A static site only contains client-side functionality, meaning it only returns the code to the browser, it doesn’t interact with a server or a database. Dynamic sites are able to be user-specific with server-side languages, like PHP. The site can intake user-input, send the data to the database, manipulate it, and send back the information.

A key feature of PHP for it’s use in an interactive web page is its available communication routes between the client, server, and the database. PHP supports numerous databases, both SQL (Structured Query Language) and NoSQL (Not only SQL) databases: SQLite, MySQL, Oracle, MongoDB, and PostGreSQL. It’s also available to different web servers: Apache, LAMP, LEMP, and Apache Tomcat. PHP can open a database connection, convert HTML/user-input into queries for that database and return the values to the server.



Source: Holly Robertson

With the natural abstraction of structures, (client, server, and database), PHP is an easy choice to use when designing an application with the Model-View-Controller (MVC) architecture. PHP works with every department in the MVC model. It opens and queries the database in the model, it sanitizes user-input in the controller, and helps format output in the view. It really is one language “to rule them all” when it comes to an MVC designed web application.

In the view, PHP integrates within HTML to display data pulled from the database or other source. For instance, if you were creating a shopping application and all of the products for sale were held in a database, PHP can query the database and then display the list of products using a ***while*** loop within a HTML unorganized/organized list.

```
<ol>
    <li><?php while($row = $statement->fetchArray(SQLITE3_ASSOC)) {
    echo '<a href="view_product.php?prod_id='.$row['prod_id'].'">;'?></li>
</ol>
```

Source: Holly Robertson

In the controller, PHP takes the user input from the view and sends it to the model. In the example project, the shopping application, the user could select the item and quantity they want to buy, the controller would then take the user input, provide form validation if desired, and then send the information onto the database. It would also provide the view with a response if an item was successfully added to a shopping cart or not.

```php
<?php
    include '../views/addItem.html';
    require '../model/db.php';
    $result = "";
    if(isset($_POST['item']) && isset($_POST['qty'])) {
        $item = $_POST['item'];
        $qty = $_POST['qty'];

        if ($db->exec("INSERT INTO forum (item, qty) VALUES('$item', '$qty')")) {
            $result = "Added";
        }
        else {
            $result = "not added";
        }
    }
?>
```

Source: Holly Robertson

In the model, PHP provides various functions to open and query a database. For the sample application, we will use SQLite3. PHP makes it extremely easy and quick to setup a database connection within a .php file. The built-in functions also allow for unit testing your application against failed database connections by providing error messages if there is a failed connection. If there is a successful connection, PHP goes on to create your database.

```php
<?php

    try {
        $db = new SQLite3('db.sqlite');
    } catch (PDOException $e) {
        $error_msg = $e->getMessage();
        include('database_error.php');
        exit();
    }

    $db->exec('CREATE TABLE IF NOT EXISTS forum (
    PROD_ID INTEGER PRIMARY KEY AUTOINCREMENT,
    postDate DATETIME default current_timestamp,
    item TEXT NOT NULL,
    qty INT NOT NULL');
?>
```

Source: Holly Robertson

Overall, PHP and the MVC pattern go hand-in-hand. Building, maintaining, and troubleshooting web applications are made easier with the built-in logic and functions of PHP and the Model-View-Controller architecture.

Bibliography

The Editors of Encyclopaedia Britannica. (2019). *Client-server architecture*. Retrieved March 23, 2019, from Encyclopaedia Britannica: https://www.britannica.com/technology/client-server-architecture

W3 Schools. (2019). *PHP 5 Tutorial*. Retrieved March 23, 2019, from W3Schools.com: https://www.w3schools.com/php/

Holly Robertson

WEBD236

Midterm

March 23, 2019

Question 3

Attached in .zip