Educational Framework Design

Technical Report
Presented to the Faculty of the
School of Engineering and Applied Science
University of Virginia

By

Makonnen Makonnen

May 08, 2020

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: Makonnen Makonnen

Educational Web Framework Design

Makonnen Makonnen
University of Virginia
School of Engineering and Applied Sciences
Computer Science Department
mm3xy@virginia.edu

Abstract

Currently the Engineering School at the University of Virginia has two courses that focus on developing web applications and one course that utilizes web development: 1) CS 3240, Advanced Software Development, is a course in which students analyze modern software engineering practices for multi-person projects. Although the course does not specifically focus on web application development, the lab section requires a semester long Django web application project. 2) CS 4260 (formerly CS 4501), in which students learn to build large scale web applications and gain exposure to many topics related to scalability such as load balancing, database management, API design, etc. 3) The third course is Programming Languages for the Web (commonly abbreviated as Web PL), CS 4640. Which teaches Web App fundamentals such as HTML, CSS, and JS as well as Asynchronous programming topics and using PHP. The project that is being proposed aims to supplement the materials in WebPL and introduce students to web framework design.

Keywords

Computer Science Education, Web Framework, PHP, Abstraction, Application Programming Interface, Design, Database, MVC

1 INTRODUCTION

Using frameworks in software development has been shown to improve code readability, reusability, maintainability, as well as shorten software development process. One of the most commonly used frameworks is Model-View-Controller (MVC) [4]. This project uses MVC as an underlying mechanism to help students learn and practice web development. Detailed information on MVC is presented in Section 2.

For this proposed project, students will be designing their own MVC architecture based web framework utilizing PHP and SQL. Frameworks are key tools in software development and at a higher level, computer programming. In context of this project, a web framework is one that specifically aims to support developers in creating web applications.

Students will follow an MVC architectural structure to create a web framework, consisting of the ability to: present an HTML page to a user, accept and handle requests, and being able to add and retrieve items from a database.

To integrate the proposed project into a web development course, students will be given the general directory set up of the code with the class definitions. Instructions will be provided on what functions students have to implement for a specified class. After the completion of a student's framework they can swap with other students in the course and create a simple web app using their classmate's implementations. This allows students to experience software reuse and refactor in practice. Students can then evaluate their classmate's framework designs based on criteria such as functionality and code/documentation readability. Projects should also be assessed on the fact that their framework utilizes MVC Architecture definitions mentioned in Section 2.

Integrating a web framework as part of the development process allows students to not just attempt in making some piece of code run, but to approach problems as engineers that are creating software with the intent of end-users.

2 BACKGROUND

Model-View-Controller (MVC)

A web framework is an organizational structure and means for providing various layers of abstraction to interact with databases, present information to a user, and handle logic. In addition, a framework can provide other features such as: session management, security, etc. One of the most commonly used framework patterns is the Model-view-controller (MVC) architectural pattern and that is what this project will based on [4].

The **model** in an MVC framework is the abstraction layer to the data management and structures. It is fully independent from the user interface (UI) and has a set of logic to interact with data persistence or a database such as adding and retrieving content from it.

The **view**, controls the interface or appearance the user can see. This logical separation makes it easy for maintenance, organization, and allows for res-usability of view logic. Decoupling it from the models and controllers also gives the advantage of changing the views without concern of modifying other components.

The **controller** defines a set of actions that utilize business decisions from the model and returns some piece of data or a view. It is essentially the middle man; utilizing the

model APIs for a specified action to return a specified view.

MVC Interaction

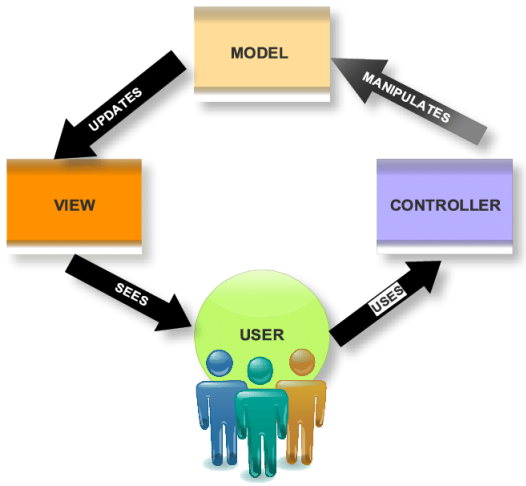

Figure 1: MVC Model [7]

Figure 1 shows a general representation of an MVC architecture illustrating how each component interacts. The user interacts with to the software through a view which presents a pre-defined interface with associated data to the user. Requests are then sent from the view to the controller. The controller handles the requests and responses, and subsequently may update the model and the view. The controller itself will be utilizing the model APIs to manipulate the data in the database and to get information from it, if it exists.

In general, a user can have multiple views, controllers, and models. Additionally, to make all these connections possible an external component is required. When a request is made from a user's browser, the framework responds to the request by using a **router**, which examines the Uniform Resource Locator (URL) of the request and determines what and how the controller should handle that request.

3 SOLUTION AND IMPLEMENTATION

This section discusses how the MVC architecture was implemented for this project. In this project, there are three main components: MVC, database, and routing. Each component consists of class and function definitions.The project source code is available in the repository for this project and is presented in Section 5.

Models

Figure 2 shows a parent `Model` class. This class will be referred to as a *parent Model*. There is one parent Model class

```
public class Model{
   public $table_name;
   public $db_cols=array();
   public $sql_params=array();
   function init($table_name){
       // Initializes the databases
   }
   function char_field($name,$length){
       // Intiailizes a char field with a specified
            name and length
   }
   function int_field($name){
       // Initializes an integer field with a
            specified name
   }
   function create_table(){
       // Creates the table utilizing PHP PDO to
            execute the query
   }
   function add($vals){
       // Adds to the table utilizing PHP PDO to
            execute the query
   }
   function delete($cols,$vals){
       // Deletes from the table utilizing PHP PDO to
            execute the query
   }
   function retrieve($cols,$vals){
       // Gets data from the table utilizing PHP PDO
            to execute the query
   }
}
```

Figure 2: Parent Model Class

as shown in 2. This parent class uses the database connection to create tables with `CHAR` or `INT` fields. Functions to add or retrieve content from the database are included. The creation of the database is done outside of the parent Model in the database set up of the framework which will be discussed later.

This Model class should be the only thing with direct access to the database. Executing SQL queries for the database should not be available in the views or controller. All queries should be executed through utilizing the APIs the models exposes to create, add, delete, and retrieve from the database. This is due to the goal of having a loosely coupled system. Each component is reserved for one task, which in this case is handling the data. Overall, this allows developers to easily test the code, swap functions out without breaking the whole framework (e.g. swapping database drivers for this project), and adding more features to the system, which is referred to as scalability.

```
public class View{
     $twig_fs = new
         \Twig_Loader_Filesystem(dirname(__DIR__) .
         '/myapp/views/');
     $twig_load = new \Twig_Environment($twig_fs);

     echo $twig_load->render($view,$params);
     return;
}
```

Figure 3: Parent View Class

```
abstract class Controller{
   public function __call($name,$args){
      if(method_exists($this,$name)){
         call_user_func_array($this,$method,$args);
      }
      else{
         throw new \Exception("This method --> $name
             <-- was not found in " .
             get_class($this));

      }
}
```

Figure 4: Parent Controller Class

Views

The View class as presented in Figure 3, has one functionality to render the specified interface. It is implemented using the twig engine rather than PHP templating due to PHP templating being quite verbose and a lot of functionality like iterating over arrays is not supported by default.

Controller

The main controller class as shown in Figure 4, is essentially used for invoking user defined controller functions in the context of an application. If the specified method exists, it will be called. Otherwise, an exception is triggered, stating that the controller method does not exist.

Database

The database logic is set in db.php. This is where the students will specify the database hostname, database name, and the IP:port of the database as well as the proper credentials (username and password) to access the database.

This project uses PHP Data Object (PDO) [1]. PDO is a database abstraction layer for interacting with various database engines. It can be used to establish database connections and run database specific commands [1]. For example, students may use PDO to connect to a MySQL [8] database and runs SQL queries. If students choose to use other databases

```
class Router{
   public function create($route,$controller,$view){
      //Add specified route to map of routes with
          values being the controller and specified
          view
   }
   public function execute($url){
       //If the route exists , find the specified
            controller and call the view.
       //If it is a post request with data being
            passed, past that as well
      }
}
```

Figure 5: Router Class

such as MySQLi [1], they can simply specify the driver supporting MySQLi instead of MySQL. Moreover, PDO can be used to create a database and sets the PDO object to use that specific database for all future queries.

Routing

The router as presented in Figure 5, has two functions. The create function creates a specific route which takes some URL path, a controller, and a view (which is a function in the specified controller). The `execute` function calls (and thus executes) a route that was created using the Router execute function. From here the controller the route was pointing to will execute what it was specified to do, such as returning a View or using the Models API to retrieve data.

4 CONCLUSION

Solution Usage

This project was intended to help students learn and practice web development using an MVC architecture. The proposed prototype, although has not been empirically validated, can potentially be used as a skeleton to help students get started with their web software development. The skeleton presented in this paper is a set of files containing class and function definitions, allowing students to fill code specifically to their project.

To use the proposed prototype, the skeleton should include the directory structure, twig, and an initial router. This basic information, serving as a template, will provide students an indication of where they need to write their code.

Limitations & Refactoring

Since this project is in an early stage, an empirical evaluation is needed to analyze how it may help students understanding the concepts of web development. Aspects such as students'

learning styles may be taken into account for future improvement. Additional features such as programming hints and automated testing functionalities may be incorporated to promote self-paced learning. Security should be included to expose students to a real-world web software scenario, in which security aspect is crucial. Features that support multiple users and form handling should also be added to the prototype. Since web applications normally support multiple users and utilize forms, having these features as part of the prototype allows students to simulate real-world web software specification and development.

5 CODE

The project source code is available at https://github.com/mmako1/mvc-capstone-s20. Access will be given upon request.

References

[1] PHP Documentation *https://www.php.net/docs.php* Accessed Apr 29 2020.

[2] UVA WebPl *http://www.cs.virginia.edu/ up3f/cs4640/* Accessed Apr 29 2020.

[3] Django Documentation *https://docs.djangoproject.com* Accessed Apr 30 2020.

[4] Glenn E. Karsner, Stephen T. Pope *A Cookbook For Using the Model-View-Controller User Interface Paradigm* Accessed Apr 29 2020.

[5] Scikit Documentation *https://scikit-learn.org/stable/* Accessed May 01 2020.

[6] Laplante, Phillip *What Every Engineer Should Know About Software Engineering* Accessed Apr 30 2020.

[7] Rashidah Olanrewaju, Thouhedul Islam, Norashikin Ali *An Empirical Study of the Evolution of PHP MVC Framework* Accessed May 02 2020.

[8] MySQL Documentation *https://www.mysql.com/* Accessed May 03 2020