PHP FRAMEWORK FOR DATABASE MANAGEMENT BASED ON MVC PATTERN

Chanchai Supaartagorn

Department of Mathematics Statistics and Computer, Faculty of Science,
Ubon Ratchathani University, Thailand
scchansu@ubu.ac.th

ABSTRACT

PHP is a powerful language to develop dynamic and interactive web applications. One of the defining features of PHP is the ease for developers to connect and manipulate a database. PHP prepares the functions for database manipulation. However, database management is done by the Structure Query Language (SQL). Most novice programmers often have trouble with SQL syntax. In this paper, we present the PHP framework for database management based on the MVC pattern. The MVC pattern is very useful for the architecture of web applications, separating the model, view and controller of a web application. The PHP framework encapsulated, common database operations are INSERT, UPDATE, DELETE and SELECT. Developers will not be required to consider the specific SQL statement syntax, just to call it the method in the model module. In addition, we use White-Box testing for the code verification in the model module. Lastly, a web application example is shown to illustrate the process of the PHP framework.

KEYWORDS

SQL, Framework, MVC, White-Box testing

1. INTRODUCTION

PHP is a server-side scripting language designed specifically for web-based applications. There are many advantages of the PHP language, for example performance, scalability, open source, portability, etc. One of the difficult issues in web application development is coding the program for manipulates of the database. Indeed, several studies suggest that traditional database query languages are not very simple to use, for non skilled users of database technologies, as a consequence of the fact that interaction is based on textual language such as SQL [1].

One way to solve the problem is develop a web application framework that is designed to support the development of dynamic website, web application and web services. The framework aims to alleviate the overhead associated with common activities performed in web development. For example, many frameworks provide libraries for database access, templating frameworks and session management, and they often promote code reuse [2]. The software frameworks significantly reduce the amount of time, effort, and resources required to develop and maintain web applications. Moreover, a framework is an open architecture based on commonly accepted standards [3].

Web application framework usually implements the Model View Controller (MVC) design pattern. The MVC pattern is a proven, effective way for the generation of organized modular applications [4]. The MVC pattern breaks an application into three modules: model, view and controller. The model module contains the underlying classes whose instances are to be used for manipulating the database. The programmers learn how to use each class and what the output is, rather than on SQL syntax. This advantage can reduce the syntax error of SQL commands. In addition, the programmers create the controller module merely to handle the user events and

DOI : 10.5121/ijcsit.2011.3219

create the view module to render the appearance of the data in the user interface. By decoupling the module, MVC helps to reduce the complexity in architectural design and to increase flexibility and reuse of code [5]. This research will propose a development of the PHP framework for database management, based on the MVC design model, which will be an effective separation of event handling, underlying classes and user interface.

The rest of this paper is organized as follows: Section 2 describes the structural design of the framework. Section 3 presents the testing solution. Section 4 shows an example of web application based on this PHP framework. Section 5 draws the conclusions and proposals for future research.

2. THE STRUCTURAL DESIGN OF THE PHP FRAMEWORK

In this section we describe the basic concept of MVC that was used for creating the PHP framework. The design of model module that contains the underlying classes for manipulating the database.

2.1. The architecture of the framework

The MVC pattern breaks an application into three modules: model, view and controller. Figure 1 illustrates the architecture of the PHP framework for database management based on MVC.

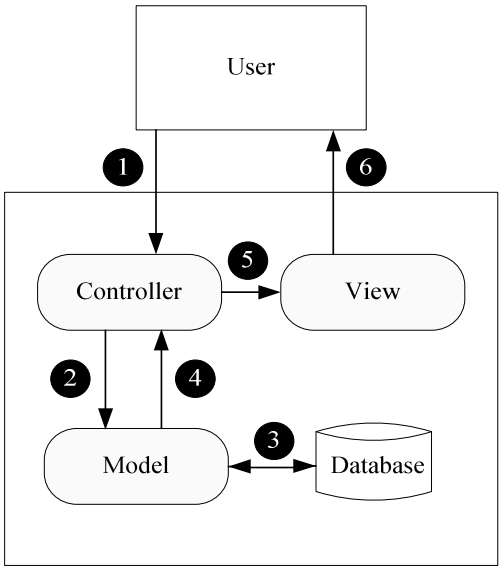

Figure 1. Framework architecture

The model means the business logic of the application and is the core of the application [6]. There are five classes in the model for this framework: connectDB class, insert class, update class, delete class and select class. These classes are to be used for manipulating the database.

The view is the user interface of the controller. It is the public face of the user event's response [7]. A user can design a view with HTML, cascading style sheets (CSS), Javascript, etc. Multiple views can exist for a single model for different purposes.

The controller component implements the flow of control between the view and the model [8]. It contains code to handle the user actions and invoke changes on the model.

The advantages of this pattern are: 1) loosely-couples: many kinds of components can interact in a flexible way; 2) parallel development: the duty is clear and it is possible to partition the whole system into components so that different person could develop at the same time. The structure is clear so as to easily integrate and maintain; 3) expansibility: Controller can expand with the module; 4) reusable quality: it can improve the reusable quality by encapsulated the business logic in the component [9].

The operation a process of the framework can be broken down into six steps. Following these steps shows the workflow of the framework.

1) A user sends a request to the controller.
2) The controller analyses the request and calls the model (method in the class).
3) The model will perform the necessary business logic and connect the database.
4) The model transmits the result to the controller.
5) The controller forwards the request to the view.
6) The request is complete when the result responds to the user.

2.2. The design of model

We will use a class diagram to illustrate the model module. A Class diagram is a graphical model that shows the relationship between classes and shows what attributes reside in the class. They are useful for showing how models work. There are five classes in the model module: connectDB class, insert class, update class, delete class and select class. The Class diagram of the model module is shown in Figure 2**.**

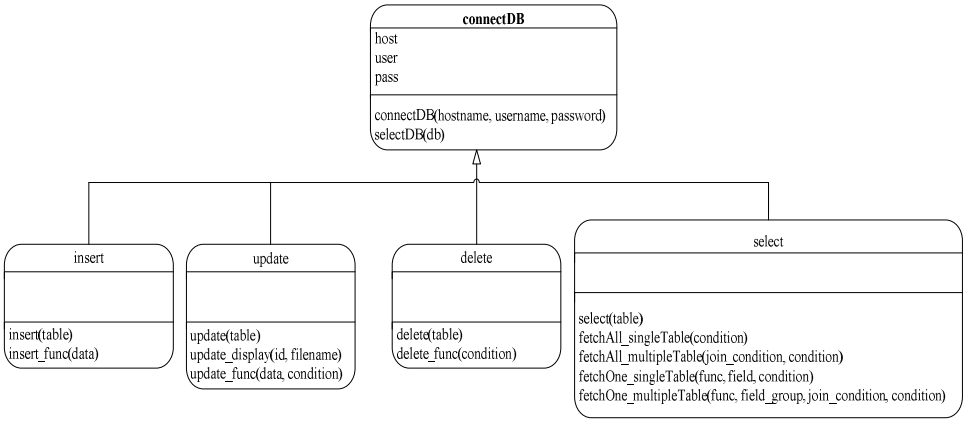

Figure 2. Class diagram showing the model module

The connectDB class is superclass or parent class. There are three attributes (host, user and pass) and two methods (connectDB() and selectDB()). The connectDB() method is the constructor that automatically executes at the time of object instantiation. It is used to initialize the host name attribute, user name attribute and password attribute. The selectDB() method is used to connect the hosting and select the database.

The insert class is a subclass or child class that inherits from the connectDB class. There are two methods (insert() and insert_func()). The insert method is the constructor that initializes the table name for insert statement. The insert_func() method is used to operate the inserting record inside table. In addition, the insert class also inherits the attributes and method from its superclass.

The update class is a subclass or child class that inherits from the connectDB class. There are three methods (update(), update_display() and update_func()). The update() method is the constructor that initializes the table name for update statement. The update_display() method is

used to show the record that matches with the condition specified. The update_func() method is used to operate the updating record inside table. In addition, the update class also inherits the attributes and method from its superclass.

The delete class is a subclass or child class that inherits from the connectDB class. There are two methods (delete() and delete_func()). The delete() method is the constructor that initializes the table name for delete statement. The delete_func() method is used to operate the deleting record inside table. In addition, the delete class also inherits the attributes and method from its superclass.

The select class is a subclass or child class that inherits from the connectDB class. There are five methods (select(), fetchAll_singleTable(), fetchAll_multipleTable(), fetchOne_singleTable() and fetchOne_multipleTable()). The select method is the constructor that initializes the table name for select statement. The fetchAll_singleTable() method and fetchAll_multipleTable() method returns all fields for the supplied select statement in single table and multiple tables respectively. These two methods return an associated array. The fetchOne_singleTable() method and fetchOne_multipleTable() method is used to compute aggregating function in single table and multiple tables respectively. These two methods return a single value. In addition, the select class also inherits the attributes and method from its superclass.

3. THE SOLUTION FOR MODEL TESTING

We will use White-Box testing to analyses code in the model module. The White-Box testing can examine the design documents and the code as well as observing algorithms and their internal data [10]. Branch/Decision coverage technique is one of several techniques for White-Box testing. This testing aims to ensure that each possible branch from each decision point is executed at least once.

We will show the example of fetchAll_singleTable() method to test the quality of software as shown in Figure 3.

```
public function fetchAll_singleTable($condition='')
{       $sql = "Show Columns From ".$this->table[0].";";
        $result = mysql_query($sql);
        $count_field = 0;
        While($dbarr = mysql_fetch_row($result))
        {
                $field[$count_field] = $dbarr[0];
                $count_field++;
        }
        $sql = "Select * From ".$this->table[0];
        if ($condition != '')
        {
                $sql = $sql." Where ".$condition;
        }
        $sql = $sql.";";
        $result = mysql_query($sql);
        $count = 0;
        While($dbarr = mysql_fetch_array($result))
        {       for($i=0; $i<$count_field; $i++)
                {
                   $value[$count][$field[$i]] = $dbarr[$field[$i]];
                } // end for
                $count++;
        } // end while
        return $value;
} // end function
```

Figure 3. Source code of fetchAll_singleTable() method

To help do this systematically, we will draw a control flow graph of the code as shown in Figure 4.

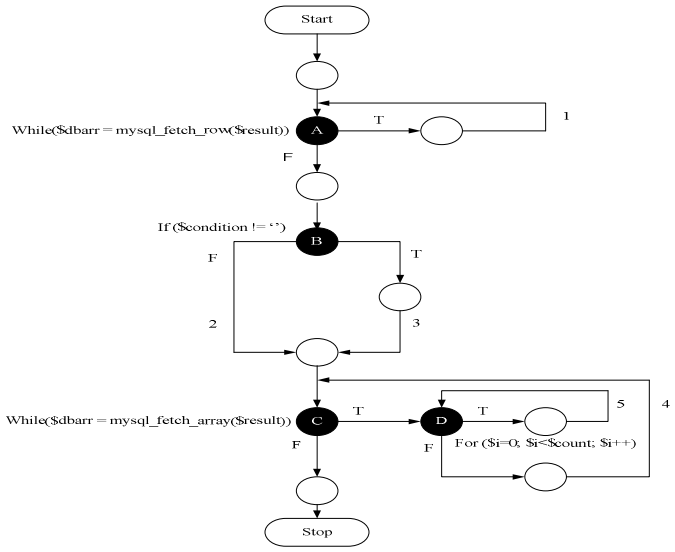

Figure 4. The control flow graph of fetchAll_singleTable() method

This graph has a shade node representing the four decisions (A, B, C and D) where the code can make the five branches (paths 1, 2, 3, 4 and 5). We created an employee database and employee table for this testing. The details of the employee table are shown in Table 1.

Table 1. The details of employee table.

employeeID	name	job	salary	departmentID
1111	Somchai	Programmer	15000	128
2222	Wichit	DBA	13500	42
3333	Somjai	Programmer	16500	128

We devised a test case to make sure that every decision and branch was taken. The following tests in Table 2 ensure branch/decision coverage.

Table 2. Test case of fetchAll_singleTable() method

Test Case	$condition	$this->table[0]	$value	Decision	Branch
1	Null	employee	$value = array(0=>array(employeeID=>1111, name=>Somchai,job=>Programmer, salary=>15000,departmentID=128), 1=>array(employeeID=>2222,name=> Wichit,job=>DBA,salary=>13500, departmentID=>42), 2=>array(employeeID=>3333,name=> Somjai,job=>Programmer,salary=> 16500,departmentID=>128))	A, B, C, D	1, 2, 4, 5
2	Job = 'DBA'	employee	$value = array(0=>array(employeeID=>2222, name=>Wichit,job=>DBA,salary=> 13500,departmentID=>42))	A, B, C, D	1, 3, 4, 5

From the above test case, we can conclude 100% decision coverage and 100% branch coverage. We have used this same testing with the rest of methods. All reachable code in the method is executed.

4. THE EXAMPLE OF WEB APPLICATION

In this section we show an example of web application that was created from the PHP framework. The database relates to employee data. This section is divided into three sub-sections: insert application, update/delete application and select application.

4.1. Insert application

The process starts by creating an input form of employee data, then this form actions to the controller. The controller calls to the insert() method and insert_func() method in the model module. The user inputs the employee data and then clicks the submit button. The input form and result is shown in Figure 5.

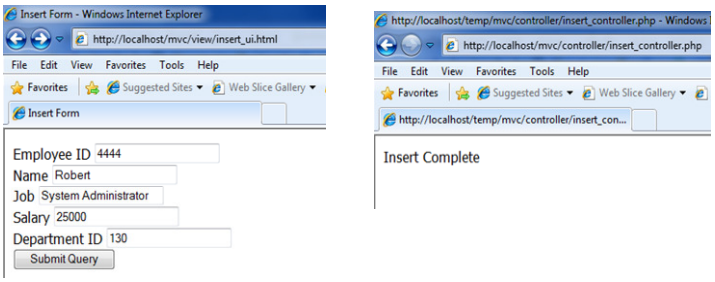

Figure 5. The insert form of employee data and the result

4.2. Update/Deleate application

Firstly, the user requests to the controller. The controller calls to the select() method and fetchAll_singleTable() method in the model module. We created the tag table to contain the employee data. In addition, we created the delete link and update link in the last column. The view of employee data is shown in Figure 6.

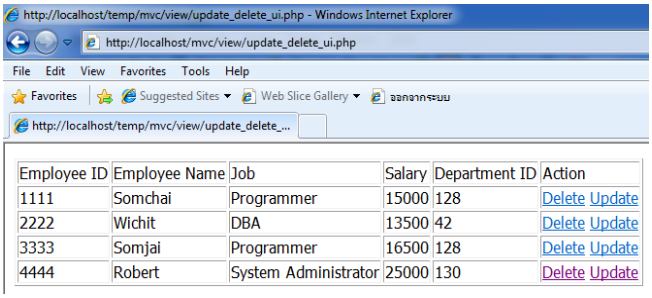

Figure 6. The view of employee data for deleting and updating

The second step, the user will click on the delete link or update link. If user clicks the delete link, the controller will call to the delete() method and delete_func() method in the model module. The view of the result after deleting is shown in Figure 7.

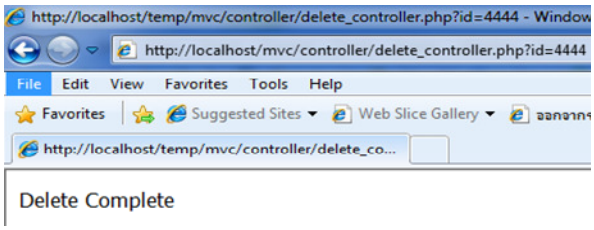

Figure 7. The result of deleting

If the user clicks the update link, the controller will call to the update() method and update_display() method. The result will show the record that matches with the condition specified. The user can modify the record data. After the user clicks the submit button, the controller will call to the update_func() method. The view of the result after updating is shown in Figure 8.

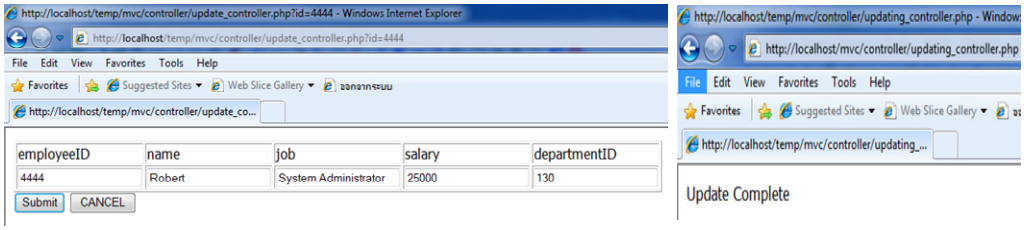

Figure 8. The update form and result

4.3. Select application

We will show an example of an employee report that joins between the employee table and department table. The user requests to the controller. The controller calls to the select() method and fetchAll_multipleTable() method for this process. In addition, this example shows the total of employees in the last line. The controller calls to the fetchOne_multipleTable() method for this process. The view of the result after selecting is shown in Figure 9.

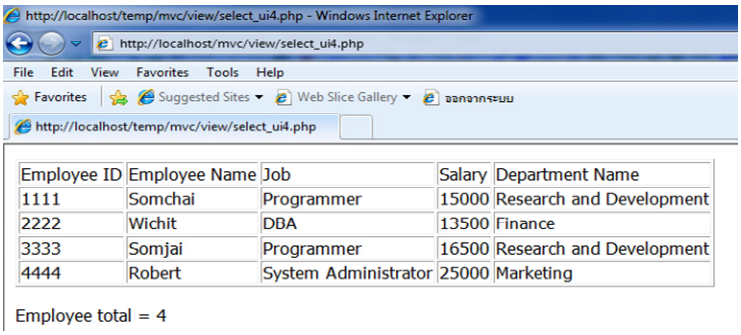

Figure 9. The view of select application

5. CONCLUSION AND FUTURE RESEARCH

The critical problem of novice programmers in web application development is coding the program for manipulates of the database. They often have trouble with SQL syntax. To remedy this problem is the main objectives of this research. The PHP framework for database management based on MVC pattern is proposed. It separates the business logic, user interface and controller. It prepares the basic methods and clarity in the accomplishment of the design of the framework that can develop a web application for database manipulation. The user or the

programmer is able to easily program web application software projects or projects of their own by customized development. This PHP framework takes the full advantages of its loosely-coupled, expansibility, and reusable quality.

In addition, we can use the White-Box testing to examine the model module in orders to guarantee the web application quality. Lastly, we show an example of web application that created from the PHP framework.

In future research, we will develop the web authoring tools that contain basic functions to create web content. Moreover, this PHP framework will be embedded in the web authoring tools. A developer will be able to create a web application for database management based on the MVC pattern.

REFERENCES

[1] Avensano L, Canfora G, De Lucia A, Stefanucci S (2002) "Understanding SQL through iconic interfaces", Computer Software and Applications Conference (COMPSAC), pp.703-708.

[2] DocForge, Web application framework, http://docforge.com/wiki/Web_application_framework, 2010.

[3] Tony C Shan, Winnie W Hua (2006) "Taxonomy of Java Web Application Frameworks", IEEE International Conference on e-Business Engineering (ICEBE'06).

[4] Hofmeister C, Nord R.L, Soni D (2000) Applied Software Architecture, Addison-Wesley.

[5] Wei Cui, Lin Huang, LiJing Liang, Jing Li (2009) "The Research of PHP Development Framework Based on MVC Pattern", Fourth International Conference on Computer Sciences and Convergence Information Technology (ICCIT), pp.947-949.

[6] J.M. Li, G..S Ma, G..Feng, Y.Q Ma (2006) "Research on Web Application of Struts Framework Based on MVC Pattern", International Workshop on Web-Based Internet Computing for Science and Engineering.

[7] Armando Padilla (2009) Beginning Zend Framework, Apress.

[8] Karam M, Keirouz W, Hage R (2006) "An Abstract Model for Testing MVC and Workflow Based Web Applications", International Conference on Telecommunications and International Conference on Internet and Web Applications (AICT/ICIW), pp. 206-212.

[9] Zhang Yanqiu, Chen Chuan (2001) "Designing JSP/Servlet+EJB Web Applications Based on MVC Design Mode", Computer Engineering, Vol.27, No.11, pp.71-73

[10] Timothy C, Robert L (2005) Object-Oriented Software Engineering, McGraw-Hill.

Author

Chanchai Supaartagorn received B.Sc computer Science from Kasetsart University, Thailand in 1994, M.Sc Computer Science from Mahidol University, Thailand in 1998. He had 12 years of teaching experience. His research interests include Model-View-Controller (MVC), Business Data Model and Web-Based Technology.