Research Article

# A Framework for Web-based Data Visualization Using Google Charts Based on MVC Pattern

Chanchai Supaartagorn*
Department of Mathematics Statistics and Computer, Faculty of Science, Ubon Ratchathani University, Warin Chamrap, Ubon Ratchathani, Thailand

* Corresponding author. E-mail: chanchai.s@ubu.ac.th

DOI: 10.14416/j.ijast.2016.11.002
Received: 23 May 2016; Accepted: 13 July 2016; Published online: 7 November 2016

**Abstract**
Data is recognized as an important corporate asset. Organizations use data in planning and decision making. However, data interpretation is the most important issue. In order to interpret the data clearly and efficiently, data visualization concept helps people understand the significance of data by placing it in visual context. This paper aims to provide a framework to create a web-based data visualization with Google Charts. The framework was designed and developed based on MVC pattern. The MVC pattern is very useful for the architecture of web applications, separating the model, view and controller of a web application. In addition, the framework also was designed to connect a database that users can retrieve data from database to create web-based data visualization. Through this experimental development, we use White-Box testing for the code verification in the model module. Lastly, a web page example is shown to illustrate the process and result from the use of this framework.

**Keywords**: Data visualization, Google Charts, MVC

## 1 Introduction

Organizations today are beginning to realize the important of data in achieving goals. Corporations have increasingly come to realize that data are important corporate assets. However, it is difficult to interpret the data clearly and efficiently to users. The concept of data visualization comes from visualization in scientific computing. It can reflect the information pattern, data relation, data change trend, and help scientific researchers and government officers to view and analyze data in the style of direct graphs to find the principles hiding behind the data [1]. Not only does data visualization help us communicate information better, it also allows us to detect patterns, trends and correlations in our businesses that might otherwise go undetected [2]. Currently, there are several open source tools that can help to create data visualization, such as Google Charts, D3.js, FusionCharts, etc. Ying Zhu has a survey of more than 20 data visualization courses taught in North American universities [3]. The outcome survey concluded that Google Charts tool was very positive. There are many advantages such as stable, up-to-date, cross-platform, flexible and versatile. According, Krishna Sumanth [4], Google Charts are the basic and simple tool for data visualization. It has a very easy-to-use Application Programming Interface (API).

Although, Google Charts is the ease for developers to create data visualization. Developers use Google Charts with simple JavaScript embedded in a web page. However, in the case of retrieving data from database to create Google Charts, most novice developers are often troubled by SQL syntax. Indeed, several studies suggest that traditional database query languages are not very friendly to use, for novice users of database technologies [5].

Please cite this article as: C. Supaartagorn, "A framework for web-based data visualization using google charts based on MVC pattern," *KMUTNB Int J Appl Sci Technol*, vol. 9, no. 4, pp. 235–241, Oct.–Dec. 2016.

**Table 1**: Comparison of data visualization tools, adapted from [8], [9]

| Visualization Tool | Summary | Licensing | Charts Rendered in | Chart Types | Documentation |
|---|---|---|---|---|---|
| FusionCharts | Delightful JavaScript charts for and mobile application | Free for personal and non-commercial use. Paid for commercial applications. | JavaScript (HTML5) Charts using SVG and VML | 15 | ✓ |
| D3.js | Powerful JavaScript library for maipulating documents based on data | BSD-3 | SVG | 9 | ✓ |
| Chart.js | Easy, object oriented client side graphs for designers and developers | Free under MIT license | Canvas based charts | 6 | ✓ |
| Google Charts | Display live data on your site | Free for all usage. | HTML 5 charts using SVG and VML | 9 | ✓ |

One way to solve the problem is develop a framework that is designed to support the development of dynamic web pages. The framework connects and integrates with Google Charts API. 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 [6]. 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. Developers learn how to use each class and what the output is, rather than on SQL syntax. This advantage can reduce the syntax errors of SQL commands. In addition, developers create the controller module merely to handle the user events and 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 [7]. This research will propose a Google Charts framework based on MVC design model. The framework was developed with PHP language that connects and integrates with Google Charts API. Moreover, a MySQL was used for database management system. In conclusion, the research provides a framework to create a data visualization with Google Charts, 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 related works. Section 3 presents the architecture of the framework. Section 4 discusses the Model testing. Section 5 shows the examples of data visualization based on the framework. Section 6 draws the conclusions and proposals for future research.

## 2 Related Works

Data visualization is the presentation of data in the style of graphs or images. It enables the viewer with a qualitative understanding of the information contents. In a business environment, data visualization can help decision makers to see analyzed data presented graphically, so the viewers can grasp difficult concepts or identify new patterns. There are several data visualization tools. We are providing the list of four best data visualization tools: FusionCharts, D3.js, Chart.js and Google Charts. Table 1 is a comparison of which some features are available in each tool.

FusionCharts support many chart types. In addition, there are many options to customize a chart. However, it is costly for commercial use. D3.js is a preferred data visualization tool among developers, but DOM manipulation very slow for large entry numbers. Chart.js renders charts with the HTML5 canvas element. Therefore, user need to include a polyfill to support older browsers, like ExplorerCanvas. We decide to use Google Charts for designing and developing a framework because it is free for all usage. Moreover, charts rendering in both Scalable Vector Graphics (SVG) and Vector Markup Language (VML). Overall, Google Charts is a good choice for those who want simple and basic graphs without wasting time for learning.

Google Charts is one of the most widely used for data visualization. The advantages of Google Charts are powerful, simple to use, and free of charge. There are several researches that use Google Charts. For example, biometric system [10] that relies on measurable physical characteristics that can be automatically checked which is called u-RPMS (USN Remote Patient Monitoring System). The system receives and monitors biometric information of patients in real-time. The system implementation has used Google Charts API which can express graph with diverse colors and shapes. General users can easily understand patient's biometric information using the smart mobile devices. Data mining research [11] that proposed a service oriented framework (SO-MSR) for conducting the MSR (Mining Software Repositories). The MSR is an active research area that utilizes data mining techniques to software projects' historical data in order to gain a better understanding of the software development [12]. This research has developed a consumer mashup application, named MetricViewer, which integrates MetricsWebAPI for metrics calculation and Google Chart Tools for metrics visualization. The visualization process flow starts from the MetricsViewer invokes the MetricsWebAPI. Then, the MetricsWebAPI invokes the Google Chart Tools, which visualizes software evolution as a graph object from a source control system repository.

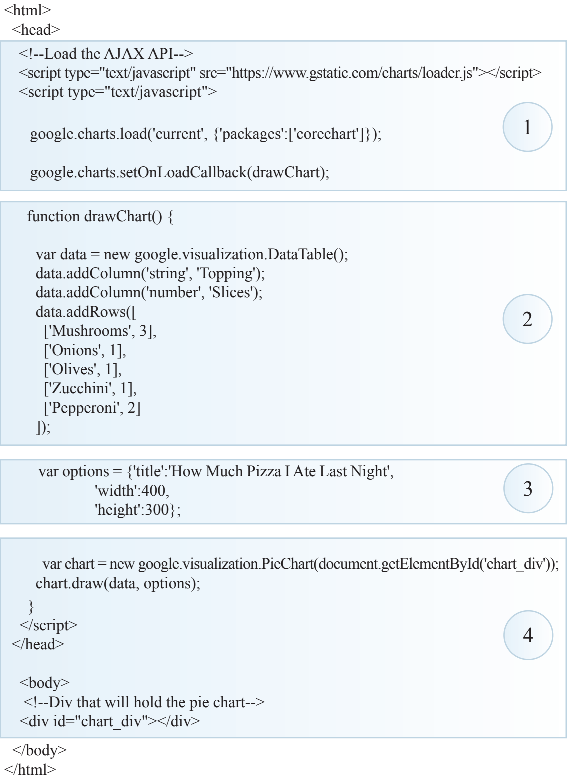
For creating Google Charts [13], there are four steps are as follows:

**1) Load the Libraries.** Google Charts consist of codes for libraries loading. Firstly, loading the AJAX API. Secondly, loading the visualization API and the corechart package. Finally, setting a callback to run when the Google Visualization API is loaded. All steps must be included in the <head> of the web page.

**2) Prepare the Data.** This step is creates a data table. Google Charts require data to be wrapped in a JavaScript class called google.visualization.DataTable. This class is defined in the Google Visualization library. The data object is created to call the addColumn() method and addRows() method.

**3) Customize the Chart.** Every chart has many customizable options, including title, colors, line thickness, background fill, and so on. All customize options are defined in the options variable.

**4) Draw the Chart.** First Google Charts must instantiate an instance of the chart class, and then call the draw() method that takes two values: a DataTable object, and an optional chart options object. Google Charts will be drawn in a division or a section of HTML document that was assigned with HTML element (typically a <div>).



```
<html>
 <head>
  <!--Load the AJAX API-->
  <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
  <script type="text/javascript">

   google.charts.load('current', {'packages':['corechart']});

   google.charts.setOnLoadCallback(drawChart);

   function drawChart() {

    var data = new google.visualization.DataTable();
    data.addColumn('string', 'Topping');
    data.addColumn('number', 'Slices');
    data.addRows([
     ['Mushrooms', 3],
     ['Onions', 1],
     ['Olives', 1],
     ['Zucchini', 1],
     ['Pepperoni', 2]
    ]);

    var options = {'title':'How Much Pizza I Ate Last Night',
           'width':400,
           'height':300};

     var chart = new google.visualization.PieChart(document.getElementById('chart_div'));
    chart.draw(data, options);
   }
  </script>
 </head>

  <body>
   <!--Div that will hold the pie chart-->
  <div id="chart_div"></div>
 </body>
</html>
```

**Figure 1**: Steps of creating Google Charts.

Figure 1 is a summary of the process and steps to create Google Charts.

Then, we reviewed the concepts of MVC pattern that will be used for creating Google Charts framework. The MVC pattern breaks an application into three modules: model, view and controller. Figure 2 illustrates the framework of MVC pattern.

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

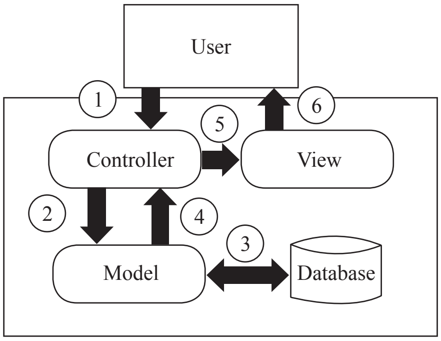


**Figure 2**: Framework of MVC pattern.

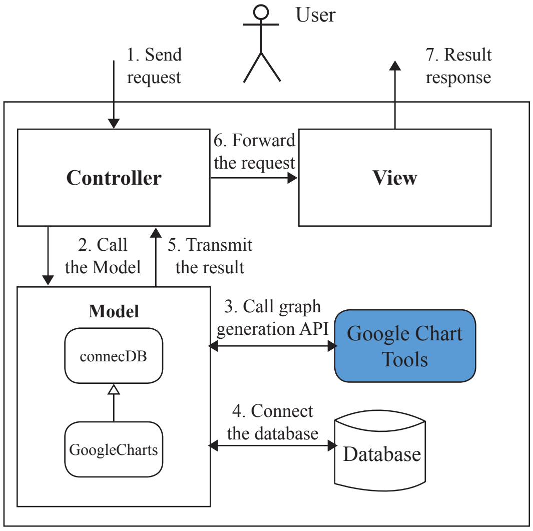


**Figure 3**: Architecture and process flow of the framework.

## 3 Architecture of the Framework

The architecture and process flow of the web-based data visualization with Google Charts is shown in Figure 3.

The process flow is shown as follows.

**Step 1**: A user sends a request to the controller that contains codes to handle the user actions and invokes changes in the model.

**Step 2**: The controller analyses the request and calls the model. There are two classes in the model module: connecDB class and GoogleCharts class. The connectDB class is a superclass or parent class. There are three attributes (Host, UserName and Password) 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 to select the database. The Google Charts class is a subclass or child class that inherits from the connectDB class. There are six methods: The select() method is the constructor that initializes the table name for select statement. The Load_ChartLibraly() method, Initial_Data() method, Customize_Chart() method and Draw_Chart() method are based on creating Google Charts four steps. In addition, the Initial_MutipleData() method is used for fetching data from multiple tables. The class diagram of the model module is shown in Figure 4.

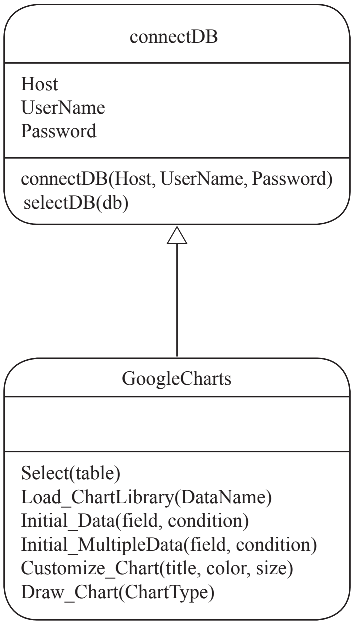

connectDB
Host
UserName
Password
connectDB(Host, UserName, Password)
selectDB(db)
GoogleCharts
Select(table)
Load_ChartLibrary(DataName)
Initial_Data(field, condition)
Initial_MultipleData(field, condition)
Customize_Chart(title, color, size)
Draw_Chart(ChartType)

**Figure 4**: Class diagram showing the model module.

**Step 3**: The model invokes graph generation API provided by Google Chart Tools. After that, the Google Chart Tools returns graph objects.

**Step 4**: The model is connected to the database and return data back in the form of an array variable.

**Step 5**: The model returns the result back to the controller. The controller prepares data in the form of variable.

**Step 6**: The controller forwards the request to the view. The view uses the command to display the data that sends it via variables.

**Step 7**: The view generates a visualization html page to the user.

## 4 The Solution for Model Testing

We will use White-Box testing to analyze 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 [14]. Brach/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 Initial_Data() method to test the quality of the software as follows.

```
public function Initial_Data($field, $condition='')
{
    if ($condition == '')     {
       $sql = "Select * From ".$this->table[0];
    }
    else {
      $sql = "Select * From ".$this->table[0]." Where
".$condition;
    }
   $sql = $sql.";";
   $query = mysql_query($sql);
   $num = mysql_num_rows($query);
   for($i=0;$i<count($field);$i++)  {
       $query = mysql_query($sql);
       for ($j = 0; $j<$num; $j++)  {
            $result = mysql_fetch_array($query);
            $data[$i][$j] = $result[$field[$i]];
       }
   }
   return $data;
}
```

To help do this systematically, we will draw a control flow graph of the code as shown in Figure 5.

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

**Table 2**: The details of financial_summary table

| Finance_id | Year | Revenue | Expense |
|---|---|---|---|
| 1 | 2012 | 1000 | 800 |
| 2 | 2013 | 1170 | 460 |
| 3 | 2014 | 680 | 1120 |
| 4 | 2015 | 1030 | 540 |

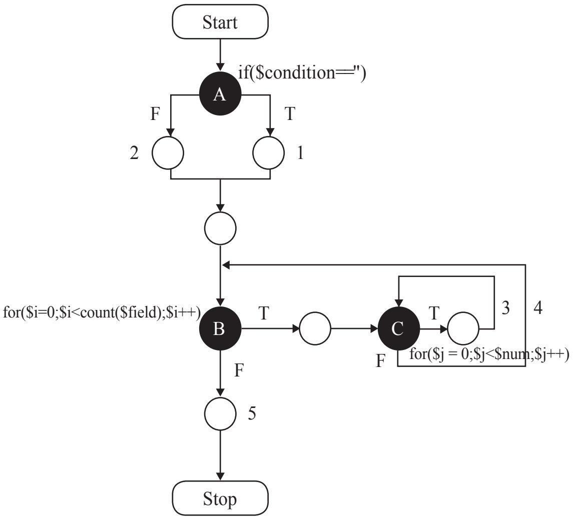


**Figure 5**: The control flow graph of Initial_Data() method.

We devised a test case to make sure that every decision and branch was taken. The following tests in Table 3 ensure branch/decision coverage.

**Table 3**: Test case of Initial_Data() method

| Test Case | $field | $condition | $data | Decision | Braanch |
|---|---|---|---|---|---|
| 1 | array("year",<br>"revenue",<br>"expense") | Null | array(<br>array(2012,1000,800),<br>array(2013,1170,460),<br>array(2014,680,1120),<br>array(2015,1030,540),<br>) | A, B, C | 1,3,4,5 |
| 2 | array("year",<br>"revenue",<br>"expense") | year=2015 | array(<br>array(2015,1030,540)<br>) | A, B, C | 1,2,3,4,5 |

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

## 5 The Examples of Data Visualization

In this section, we show examples of a web page that was created from the framework. The database is related to financial data. The codes are examples of the controller module and view module respectively.

**Financial_controller.php**

```
<?php
function __autoload($class_name) {
require_once "../class/".$class_name . '_class.php';
}
$connectdb = new connectDB('localhost','root','');
$connectdb->selectDB("finance");
$table = array("financial_summery");
$select_obj = new select($table);

$DataName = array('Year', 'Revenue', 'Expense');
$graphlibrary =
 $select_obj->Load_ChartLibrary($DataName);

$condition = "";
$field = array("year","revenue","expense");
$prepare_data =
$select_obj->Initial_Data($field, $condition);

$title = array('Financial Data','Year','Total (Million
Baht)');
$color = array('Purple','Orange');
$size = array('700','500');
$customize_data =
 $select_obj->Customize_Chart($title, $color, $size);

$charttype = "ColumnChart";
$draw_data=$select_obj->Draw_chart($charttype);
?>
```

**Financial_view.php**

```
<?php
  require_once "../controller/financial_controller.php";
  echo "<h1>Financial Data Graph</h1>";
  echo $graphlibrary;

  for($i=0;$i<=3;$i++)   {
        echo "['".$prepare_data[0][$i]."',";
        echo $prepare_data[1][$i].","
.$prepare_data[2][$i];
        echo "],";
  }
         echo $customize_data;

         echo $draw_data;
?>
 <div class="chart_div " id="chart_div"></div>
```

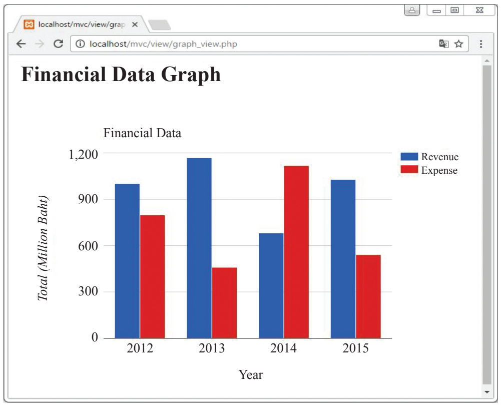


**Figure 6**: A screenshot of financial web page.

A screenshot of financial web page with Google Charts based on MVC pattern is shown in Figure 6.

## 6 Conclusion

To make it easier to develop web-based data visualization with Google Charts, we have designed and developed the framework based on MVC pattern. It prepares the basic methods and clarity in the accomplishment of the design of the framework. Developers can create web pages which connect and integrate with Google Charts API for visualizing data from a database. In addition, we use the White-Box testing to examine the model module in order to guarantee the web page quality. In addition, we show an example of web page that is created from the framework.

In our future work, we will continue to develop the Google Charts creator that is embedded the framework in the system. A developer will be able to create web-based data visualization with Google Charts easily and quickly. Additionally, comparative empirical evaluation with other existing tools must be conducted.

## References

[1] Z. Liang, W. Bing-fang, Z. Yue-min, M. Xin-hui, and Y. Lei-dong, “Research on eco-environmental data visualization for three gorges project,” in *Proceedings International Symposium on Information Engineering and Electronic Commerce*, 2009, pp. 590–593.

[2] B. Martinuzzi. (2016, Apr.). *Data visualization will change the way you think about your*

*business* [Online]. Available: https://www.americanexpress.com/us/small-business/openforum/articles/data-visualization-will-change-the-way-you-think-about-your-business

[3] Y. Zhu, "Introducing google chart tools and google maps api in data visualization courses," *IEEE Computer Graphics and Applications*, vol. 32, no. 6, pp. 6–9, Nov. 2012.

[4] K. Sumanth. (2016, Apr.). *D3.js vs Google Charts: A data scientist's review* [Online]. Avaiable: https://blog.socialcops.com/engineering/d3-js-versus-google-charts

[5] L. Avensano, G. Canfora, A. De Lucia, and S. Stefanucci, "Understanding SQL through iconic interfaces," in *Proceedings Computer Software and Applications*, 2002, pp. 703–708.

[6] C. Hofmeister, R. L. Nord, and D. Soni, *Applied Software Architecture*. Addison-Wesley, 2000.

[7] W. Cui, L. Huang, L. J. Liang, and J. Li, "Theresearch of PHP development framework based on MVC pattern," in *Proceedings Fourth International Conference on Computer Sciences and Convergence Information Technology*, Seoul, South Korea, 2009, pp. 947–949.

[8] Wikipedia. (2016, Jul.). *Comparison of JavaScrip charting frameworks* [Online]. Avaiable: https://en.m.wikipedia.org/wiki/Comparison_of_JavaScript_charting_frameworks

[9] EDUCBA. (2016, May.). *Top 10 best data visualization tools (Essential)* [Online]. Avaiable: https://www.educba.com/10-best-data-visualization-tools/

[10] L. Jae-Gwang, K. Young-Hyuk, L. II-Kwon, L. Jae-Pil, H. Namgung, and L. Jae-Kwang, "Implementation of u-RPMS using Google Chart in hybrid application for visualization of patient's biometric information," in *Proceedings International Conference on Information Science and Applications*, Pattaya, Thailand, 2013, pp. 1–4.

[11] Y. Sakamoto, S. Matsumoto, and M. Nakamura, "Integrating service oriented MSR framework and Google Chart tools for visualizing software evolution," in *Proceedings Fourth International Workshop on Empirical Software Engineering in Practice*, Osaka, Japan, 2012, pp. 35–39.

[12] Sunday O. Olatunji, Syed U. Idrees, Yasser S. Al-Ghamdi, and Jarallah Ali Al-Ghamdi, "Mining software respositories–a comparative analysis," *International Journal of Computer Science and Network Security*, vol. 10, no.8, pp. 161–174, 2010.

[13] Google Developers, Using Google Charts. (2016, Apr.). [Online]. Avaiable: https://google-developers.appspot.com/chart/interactive/docs/

[14] C. Timothy and L. Robert, *Object-Oriented Software Engineering*. New York: McGraw-Hill. 2005.