ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 8(S7), 618-623, April 2015* DOI: 10.17485/ijst/2015/v8iS7/70449

A Design of Infographics by using MVC Design Patterns Based on N-Tier Platform

Myeong-Ho Lee*

Department of e-Commerce, Semyung University, Korea; mhlee@semyung.ac.kr

Abstract

In IT environments, computers and the Internet are used just as water and air are. Furthermore, the time enters upon another phase with new potential growth engines such as cloud computing, big data, and business intelligent. In terms of various interactions and infographics, demands among users continue to increase, and the necessity of additional functions that the current versions of web browsers do not provide is ever more emphasized than before. Hence, this study suggests a design of infographics in utilization of N-Tier platform based MVC design patterns.

Keywords: Business Intelligent, Component, Infographics, MVC Design Patterns, N-Tier Platform

1. Introduction

As semi-structured data such as SNS, web log, social media, email, image, and video are rapidly increasing, interest in big data continues to increase up to the point of introducing a new era of cloud computing environment. According to the theory of the waves of the internet revolution, it is predicted that web 3.0 will advance centering on keywords such as N-screen, cloud computing, big data, digital contents, smart work, social services platform, smart device, mobile webapp, HTML[5], etc[1]. In line with such change in platforms, the area of software in IT industry involves the problem of increasing expenses for distribution, and thus it turns its attention to web application platforms.

Internet-based web application systems, however, require a large quantity of computing on the part of servers although they are of distributed computing systems, and the complexity increases as services continue to be diversified. In addition, demands for various interactions and user interfaces continue to increase among users. Hence, in response to such requirements among customers, it is necessary to complement the limit of web-based platforms, utilize the advantages of client-server environments, and accommodate various, complicated requirements appropriately by adopting a new model. To this end, infographics were created. This study aims to suggest a design of infographics by utilizing MVC (Model-View-Controller) design patterns based on N-Tier platform.

2. Investigation on Existing Research

2.1 History of Infographics

Infographics is to provide desired information out of such a tremendous amount of information and to advertise it with efficiency in information delivery by minimizing the contents. Infographics are an visual representation of information, data, or knowledge. It is distinguished from common pictures or photos in that it delivers information in a specific, superficial, and practical manner. This can be used for symbols, maps, and technical documents which require fast and clear explanation on complicated information. Infographics include such elements as chart, fact box, map, diagram, flow chart, logo, calendar, illustration, and TV program schedule[2].

*Author for correspondence

As for the history of infographics, it traces back to Lascaux Cave Painting created in France between 17,000 B.C. and 15,000 B.C. and pictographs in Egypt Luxor Temple in 3,000 B.C. More specifically, William Playfair, an economist, is regarded as forerunner of data visualization as he used such means as line graph, bar graph, and circle graph to express statistical data in his work, 'Commercial and Political Atlas,' written in 1786. Thereafter, Otto Neurath others created ISOTYPE, a visual communication model that explains concepts with icons and pictures, to deliver information in a simplified format. This can be viewed as the original form of icons that are widely used in Infographics today[3].

Customers can design static, interactive, or motion infographics for themselves, and it is also possible to visualize data by means of charts, maps, etc. In addition, specific contents may be differentiated with elements such as font, map, chart, and icon and then shared through well-known social networks, home and abroad.

Types of infographics may be divided to statistical infographics, timeline infographics, process-based infographics, location and geography-based infographics, comparative infographics, and visual storytelling-based infographics[4].

2.2 MVC Design Patterns

As shown in Figure 1, MVC design patterns may be divided to three core component - model, view, and controller - so that each can handle their given tasks[5,6].

A model encapsulates all essential data and business logics necessary for an application rather than relying on expression methods. A view is for visual expression. It is an interface through which users can see things and interact with each other. A controller may call on models and views when necessary to interpret and implement a user's request and thus control the relation between a model and a view. MVC design patterns are advantageous in that they separate visualization from data and logic so that multiple views that share the same model enhance a code's reusability. In addition, embodiment, testing, and maintenance also become easy. A model encapsulates all essential data and business logics necessary for an application rather than relying on expression methods. When a user requests data through a view or a controller, a model gives data to that view. Since it is impossible or difficult for a web application to do this, a pull type method through which a client pulls one from a model is used instead of a push type method through which a modified aspect is pushed into a model[7].

2.3 PAC Model

The basic structure of PAC (Presentation-Abstract-Controller) is of an architectural pattern of the Presentation-Abstract-Controller. Figure 2 illustrates this structure[6,8].

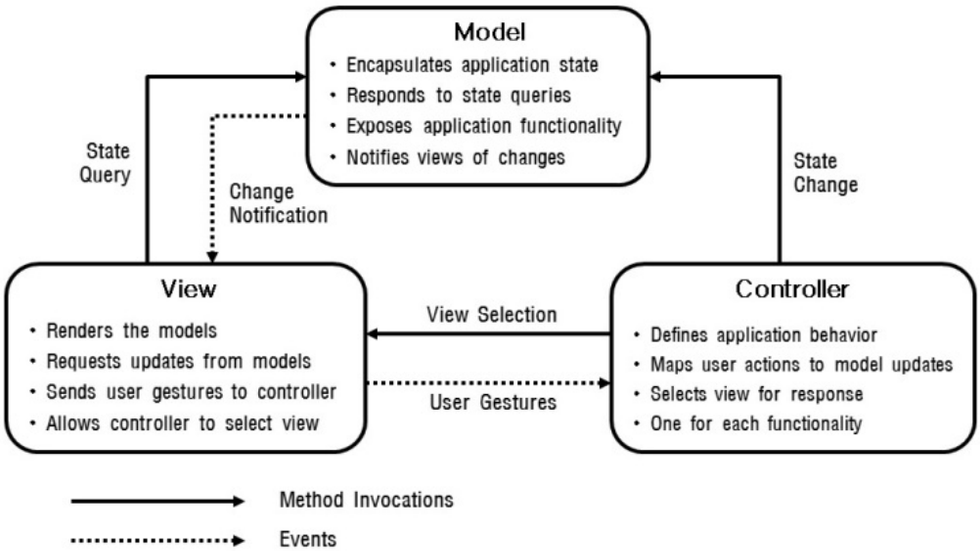

Figure 1. Architecture of MVC design patterns.

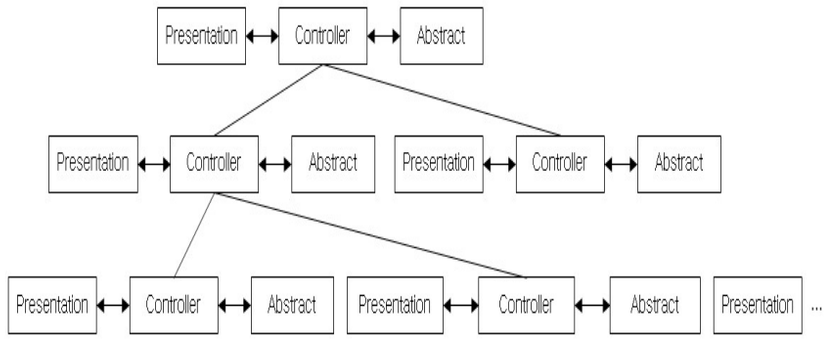

Figure 2. Hierarchical Structure of PAC.

A presentation is an object that functions as an interface, an abstract is a model to be processed, and a controller is an object that controls processing. The structure is similar to that of MVC, but the controller's role is different: The controller of MVC is to abstract input devices around GUI for data from mouse events or keyboard inputs while that of PAC is to embody an application's logic. Hence, the view and controller of MVC are embodied by a presentation in the case of PAC. Recently, views and controllers are often provided as part of a package called 'widget.'

PAC is advantageous in that it defines PAC layers. Since each layer of an application is structured by PAC, these form the general structure of an application, and this structure is quite useful when an application is established based on a component.

2.4 N-Tier Platform

The environmental structure of an N-Tier platform seeks a large-capacity distributed computing system, which consists of the pc-based presentation tier where a web browser is operated, the middle tier where web applications and business logic are operated, and the data tier where all information generated from business areas is stored and managed. The presentation tier is designed to include Unit Task user interface objects for a process handler that manages highest level processes and Unit Tasks that form a process. Unit Task business object components that correspond to unit task user interface objects form the middle tier. Unit Task components of this layer are embodied as EJB components and function to handle and process business data just as the existing components did. These Unit Task components communicate with data objects in the actual data tier. As for the suggested type of development, an application is developed in a way that arranges unit functions. Currently, most applications are developed this way, they have limitations in coping with changes in businesses and processes although the generic technology is based on OOP.

Hence, this study suggests a design method for user interface elements of the presentation tier that will match business, process, and code units to overcome this type of limitation. Business processes are examined, and tasks that form a given process are drawn out. A development is initiated for such task units of an examined process, and finally, a combination of these tasks becomes an application system. Instead of using common MVC design patterns as they are, the structure is modified specifically for J2EE (Java EE) environment[9,10].

3. Design Architecture of Infographics

3.1 Design of Framework

The logical structure of Infographics suggested in this study includes multiple processes necessary for a business in the highest level as shown in Figure 3. Each process consists of Unit Tasks, and each Unit Task carries out formalized activities.

When a system is formed by extracting certain Unit Tasks from a certain process, the physical structure of infographics is as in Figure 4. In the highest level are the process menu webapp and process webapp that handle the process.

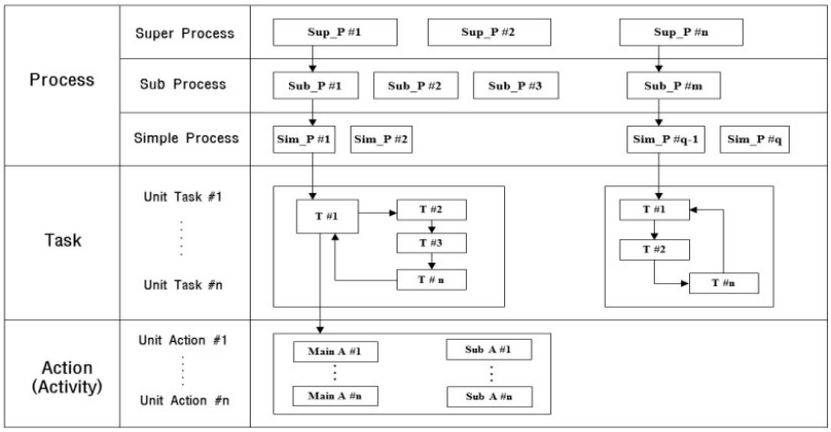

Figure 3. Logical Architecture of Infographics.

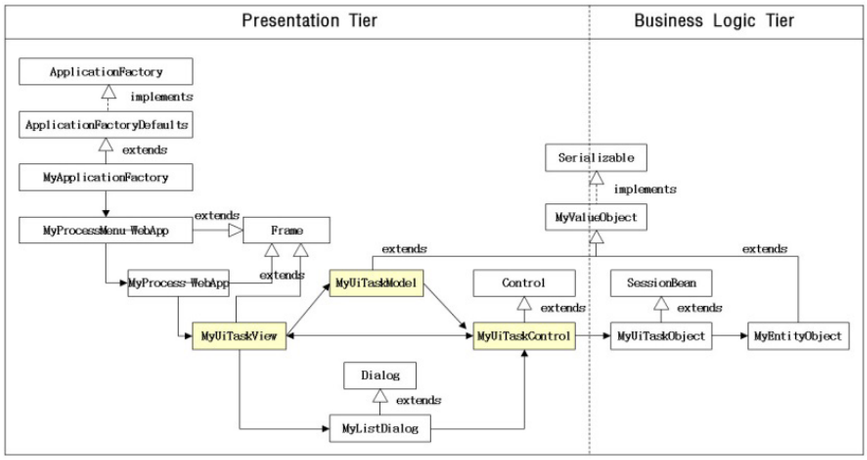

Figure 4. Physical architecture of infographics.

Unit Tasks formed in the process webapp are located in the presentation Tier as a form of MVC. The middle tier consists of business objects of Unit Tasks and entity objects just as in existing design methods. As MVC design patterns are introduced, a new concept of Value Object is added. Business objects of such Unit Tasks communicate with EIS (Enterprise Information System) tier ERP database servers or dissimilar data servers[11].

3.2 Design of Process Handler

The logical structure for the suggested user interface is illustrated in Figure 3. In the highest level are various Processes necessary for a business. Each Process consists of Unit Tasks, and each Unit Task carries out formalized Activities.

3.3 Design of Infographics Object

3.3.1 Design of Model

A package's model class is an abstracted class with a business value object. Upon request from the Controller (a processing command), a Value Object is received and the value is displayed in a proper format through a view. It also transmits data to VOs (Value Object) in the middle tier according to the controller's unit activity based on the modified data of the view. This class defines the Object data type's object, which is selected Object.

An object represents the attribute value of a task currently being worked on, and selected Object exists to represent the attribute value of the task that is referred to. Hence, a Unit Task user's interface is formed by adding a Model class to the name of a Unit Task that inherits a Model class in reference to the middle tier's Value Object. Figure 5 illustrates the structure of a Model class.

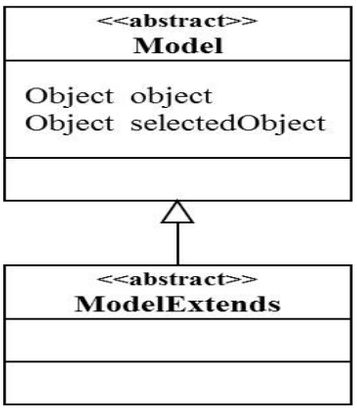

Figure 5. Architecture of model class.

3.3.2 Design of ListDialog and View

A common MVC design pattern contains various patterns of views that show model values, but in this study, the View is designed to have only one Main per Unit Task. One ListDialog is used for various types of searching.

To generate the Main View and ListDialog for ordering, the View class and ListDialog class exist as the super class as shown in Figure 6.

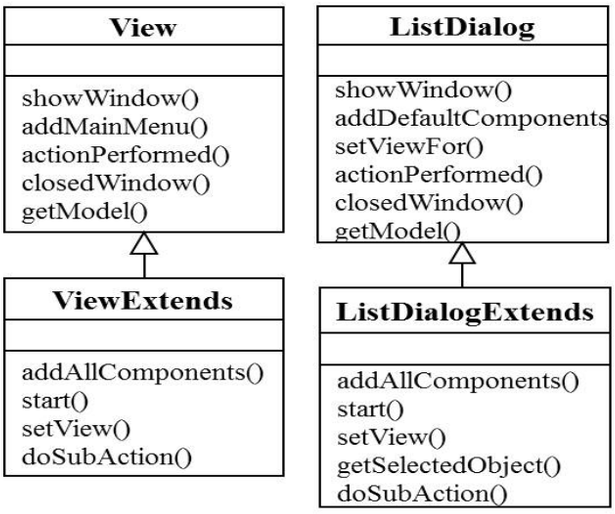

Figure 6. Architecture of view and ListDialog class.

Actions exist to receive user events. These Actions are the Main Action that exists commonly in every Unit Task infographic's Main View.

A View class represents visible views, and this class inherits a Frame. To receiver user events, designed are Main Menu Actions such as generation, reference generation, inquiry, correction, deletion, selection, storage, and end.

3.3.3 Design of Controller

As shown in Figure 7, the Controller class accompanies all Unit Activities among the Model, View, and ListDialog. It embodies methods that generate Model, View, and ListDialog, methods that receive or return Model values, methods that process Unit Activities, etc.

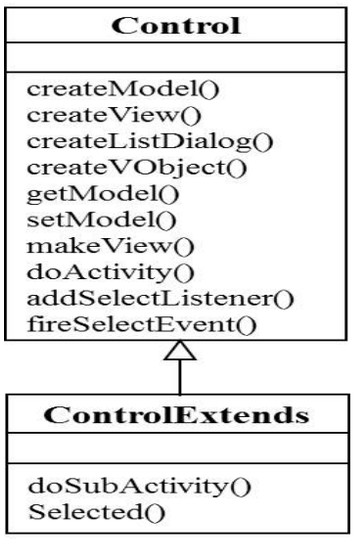

Figure 7. Architecture of Controller Class.

A Controller class generates each object based on the information received from the generator such as application, Unit Activity, sub task existence, Model, View, List Dialog, Value Object, and Unit Task name. It also embodies processing logic for Main activities. Other controllers produce records of Unit Tasks, serialize objects and save in the database to restore them in case of unexpected system disorders.

Each Unit Task name is attached to the Controller class. Since the Unit Task's model, view, and List Dialog need to be generated, its class is declared with member variables and used as the generator's parameter. The generator's parameter is also used to designate the values that indicate the unit activity and sub task existence.

Basically, a Controller class defines the processing of Main Activity, and the Controller is attached to each Unit Task's name as aspects of Sub Activity are reflected in the method of do Sub Activity.

4. Conclusions

Since the user interface suggested in this study involves specified units called a Unit Task as a development unit, development scheduling and role dividing are easier than in existing development methods.

In addition, developing standardized Unit Tasks contributes to shortening an actual development period and improving system software quality. This interface makes it possible to reuse Unit Tasks as well as objects, and thus using Unit Tasks developed in previous large-scale projects will greatly improve software productivity.

Since this type of design method synchronizes a business process and a system, the need for user education is minimized. When a process is to be modified, affected Unit Tasks only are to be modified. Hence, it is possible to cope with frequent changes in a process flexibly.

The future study needs to include examining practical cases through a detailed design and researches on design patterns of a middle tier for the establishment of e-Commerce and m-Commerce based on web services.

5. References

1. Lee MH, Han JS. Comparison of development productivity of Spring 2.5 and EJB 3.0 with lightweight container architecture. The Society of Digital Policy and Management. 2012; 10(3):137–42.
2. Available from: https://en.wikipedia.org/wiki/Infographic
3. Oh BK, Kang SJ. Text book of information design. AhnGraphics; 2008.
4. Kim MY. Infographic. Gilbut; 2004.
5. Krasner GE, Pope ST. A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. Journal of Object-Oriented Programming. 1988; 1(3):26–49.
6. Lee MH, Park JS. A design of productive life cycle of controller by using MVC models on X-internet environment. Journal of the Korea Location Based Service Society. 2005; 3(1):77–83.
7. Lee MH. A design of N-Tiers platform for building enterprise framework with development productivity. The Society of Digital Policy and Management. 2013; 11(10):411–7.
8. Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M, Sommerlad P, Stal M. Pattern-Oriented Software Architecture. Volume 1: A System of Patterns. Hoboken, NJ, United States of America: John Wiley and Sons; 1996.
9. Alur D, Malks D, Crupi J. Core J2EE Patterns: Best Practices and Design Strategies. Santa Clara, California, USA: Sun Microsystems Press; 2003.
10. Lee MH. A design of business tier by using MVC design patterns for plant information. Journal of the Korean Institute of Plant Engineering. 2006; 11(2):99–107.
11. Lee MH. A design of N-screen convergence presentation tier by using infographics based on N-Tier platform. Journal of the Korea Convergence Society. 2014; 5(4):9–13.