Comparative analysis of WebForms MVC and MVP architecture

GU Ming-xia, TANG Keming
School of Information Science and Technology
Yancheng Teachers University
Yancheng, China
yctc_gmx@126.com, tkmchina@126.com

Abstract—*Introduced WebForms, MVC and MVP architecture works under the NET platform,,the characteristics of the three structures were analyzed and compared, especially described WebForms's ViewState, the performance and client ID pollution problems and give solutions; combing their respective advantages, noted suitable scenarios of three kinds framework, aiming to supply reference for developing Web system.*

Keywords-Software architecture, Web system, MVC, MVP, ASP.NET

I. INTRODUCTION

Software architecture has taken tremendous and obvious changes to system design and maintenance. The combination of software architecture and mainstream software development methods can enhance the abstraction level of software reuse and the efficiency of software development. Currently, there is a variety of technology system of Web application at home and abroad. One of the most representative and widely-used development technologies is ASP.NET from Microsoft. At present, ASP.NET has provided the framework of WebForms, MVC and MVP, etc. for Web system development. This paper will conduct a comparative analysis on the characteristics and usage scenarios of the three architectures supported by ASP.NET and give alternative reference to the three architectures.

II. OVERVIEW OF FRAMEWORK

A. WebForms

WebForms is the framework developed by ASP.NET applying Web, which creatively introduces form model utilized in Window Desktop into the development of Web application program. Moreover, it also applies the event-driven method, static front-end code and back-end procedures completely being separated in two files. In accordance with control model, the front page developers can easily drag and drop controls, while back-end users can use any kind of language in the NET platform to carry out back-end programming in order to respond for events of page and controls, which all can accelerate the application and development of Web.

B. ASP.NET MVC

MVC (Model-View-Controller) design pattern divides this software into 3 basic parts: Model, View and Controller in order to promote Web development. ASP.NET MVC is a framework of Microsoft's official written Web applications using MVC pattern and it latest version is 2.0.

ASP.NET MVC framework is helpful for us to develop a program in loose coupling way. View part is responsible to generate the users' interface of the application, which is only used to fill the HTML template of application data transferred from the controller part; model part is responsible to realize the data logic of application and it describes application's business objects; control part is corresponding to a set of processing function and it is controller which is used to respond to user input and interaction situation. Besides, controller will handle all the requests and decide which model can be used and what kinds of view can be generated. Work principle is shown in Figure 1.

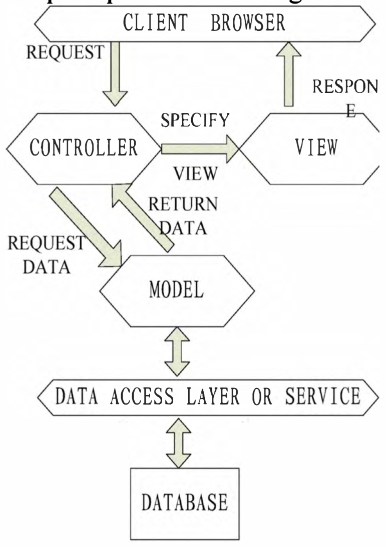

Figure 1. ASP.NET MVC

C. ASP.NET MVP

User interface layer is often reduced to a container containing the logic that should have belonged to the other layer when developing system, which makes it different to debug the code of UI layer. In addition, there is a lot of repeat code in the public view of application and it's difficult to find a good optional reconstruction method. MVP (Model-View-Presenter) design mode can separate logical code and static code in UI layer. MVP is first put forward by Mike Potel in Taligent and it is mainly used to solve the problems of MVC mode that structure is too complex and coupling of model-view is too high.

Presentation layer can be divided into UI(User Interface)and P Logic(Presentation Logic）. UI is mainly responsible for accepting user input and display output to users, while UI itself doesn't contain any logic. P Logic is the logical content which should be involved in presentation layer. For example, a text cannot be empty, which content should be attained when something occurs, etc; all of these belong to P Logic. P Logic is more abstract than specific UI, for its essence is logic and it can be used in any UI which is consistent with this logic. The link between UI and P Logic is the incident; UI can trigger various incidents according to users' action, while P Logic will respond to incident and implement the appropriate logic [4]. The structure of UI and P Logic and their

Interaction principles are shown in Figure 2.

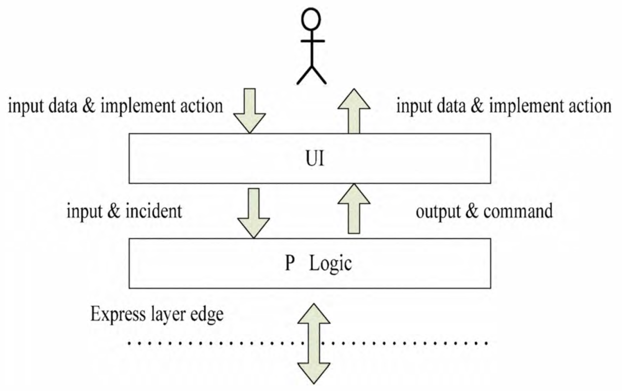

Figure 2. UI and P Logic

The core idea of MVP is to separate UI into View and P Logic into Presenter, while both business logic and field-related logic are separated into Model. View and Model coupling will be completely lifted, no longer the same as in MVC to realize the Observer mode, and the communication between them relying on Presenter. Presenter responses to user's action accepted by View, transfers business logic in Model and finally return the information required by users to View. MVP work principles are shown in Figure 3.

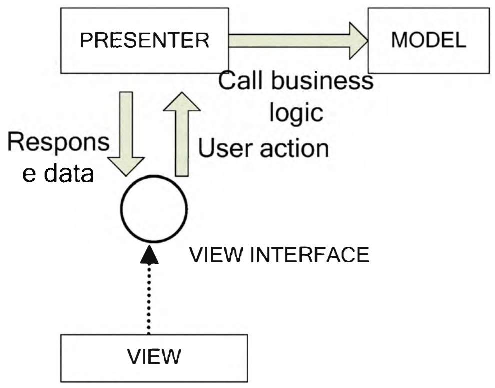

Figure 3. MVP mode

III. Performance Comparison Analysis

A. *Features of Web Forms*

1. View State

As the HTTP protocol is stateless, pages is instantiated, executed, emerged and processed in the process of round-trip between servers. In order to maintain the page's UI state, WebForms has put forward the concept of View State, which means when the data being posted to the page, UI state can be recovered in order to support certain of complex control events. While if the View State has been used improperly, the size of the page will increase too much and the network transmission performance also will be affected. Therefore, in some cases, such as jumping instantly from the current page to a new page that has nothing to do with the former, users can turn off View State.

2. Effects on performance

WebForms components model has a life cycle, and sometimes life cycle is implemented over and over again "pointlessly", affecting performance. But in most Web applications, performance bottlenecks lie in access to database. And performing another database query can be worth 100 million times quotation copy [5]. For this bottleneck, it can be solved through optimizing database query and the adoption of cache.

3. Chaotic HTML

Some server controls in WebForms will generate ugly HMTL, difficult to process a style control. For example, Grid View cannot generate <th />,bringing about chaoses,but the key of WebForms is Control Model, such as <div> <uc1:DemoControl ID="DemoControl1" runat="server" /></div>, generated html<div> Hello World! </div>, very neat, and Master Page, <asp:ContentPlaceHolder />, basic control, displaying batch data Repeater, etc and without generating any excessive codes.

4. Client ID contamination

In order to make the final client ID generated by the server controls among components unique, WebForms introduces the concept of NamingContainer. The final ID generated in the server controls of NamingContainer will use "client ID"as prefix. Such a kind of "recursion"method makes sure that the server control of client ID is unique. In additon, with the application of AJAX technology, utilizing JavaScript to operate DOM is often seen in the market. For the purpose of getting html elements in client, the server control model has provided a property setting of Client ID, through which users can obtain the final client ID from the sever control. For example, if you access to the corresponding HTML elements in TextBox and write codes as follows:

```
<%@ Control Language="C#" AutoEventWireup="true" %>
    <asp:TextBox runat="server" ID="textBox" />
    <script language="javascript" type="text/javascript">
      document.getElementById("<%=
this.textBox.ClientID %>").value = "Hello World!";
    </script>
```

At this time, after the controls are put on the pages, it will generate codes in client as follows:

```
<input name="DemoControl1$textBox" type="text" id="DemoControl1_textBox" />
<script language="javascript" type="text/javascript">
document.getElementById("DemoControl1_textBox").value = "Hello World"!;
</script>
```

Notice name and id of <input /> element. They all left traces of NamingContainer. Because the tag <%= %> is directly used on the page to output the server control ID, the JavaScript code in the client can correctly access to the corresponding client <input /> element of the server <asp:TextBox />

Such a kind of client ID difficult to predict in designer is the "cliend ID contamination"problem when using WebForms, which makes the integration of Web forms and JavaScript frame very difficult to realize.

As the project progresses, more and more complicated JavaScript codes will appear on webpage. So as to improve page loading and performance, it is usually to transfer them to js files and refert to them on webpage. The problem is that in order to quote the DOM element generated by a specific server correctly, users have to use <%= %> tag to output Client ID of controls, but <%= %> cannot be written into is files. Therefore, "client ID contamination" has always been a very serious problem when using WebForms.

5. Poor support on unit testing

All the content throughout the ASP.NET framework is close-coupled type and use only one class to in charge of displaying and processing customer input. Therefore, unit test is a mission impossible. But when following methodology of agile software and corresponding convention of developing software, unit testing is very important.

B. Features of ASP.NET MVC

MVC mode applies its given controller Action to replace Web form incident, utilizing REST-based （Representational state transfer） URL to replace file name extension method used in Web form incident in order to construct another URL which is more compatible with the SEO standard. The contrast of URL generation way between Web Forms and ASP.NET MVC is shown in Figure 4:

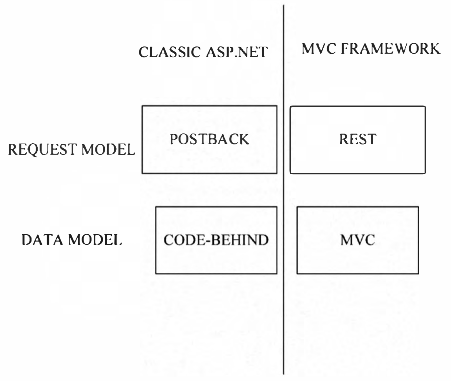

Figure 4. URL structure distinctions

Take the application of VS2008 as an example. To create a new MVC project, the file structure automatically generated will be displayed. By default, ASP.NET MVC project has six top-level directories, including:

Controls – Place Controller class and handle URL requests.

Models - Place business entity class and represent and manipulate data.

Views – Place the UI template file and display output.

ASP.NET MVC URL contains a powerful URL routing engine, which can flexibly control how URL is mapped to controller class. By default, new ASP.NET MVC project has registered a pre-configured URL routing rule, under such circumstances applications can be launched easily without any configuration. Users can see the default Register Routes of routing rule in Application class of the project.

```
public static void RegisterRoutes(RouteCollection routes)
    { routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
        routes.MapRoute(
            "Default",                          // Route name
              "{controller}/{action}/{id}",              //
URL with parameters
            new { controller = "Home", action = "Index",
id = "" } // Parameter defaults
        ); }
```

The transferred Map Route we have mentioned above is used to register a default routing rule and map required URL to controller class. The URL format: / (controller) / (action) / (id), here controller means the class name which should be instantiated, while action is the name of public method which will be transferred, id is an optional parameter embedded in URL address used to pass parameters to method. The third parameter which is passed to Map Route method is a set of controller/action/id default value, if the URL is not specified, Controller=Home、Action=Index、Id="".

When using ASP.NET MVC framework to develop Web application, there is no data return and no saving view state in the page, so developers can completely control over page rendering process and it's also easy to take unit test and test driven development. ASP.NET MVC framework still supports the related properties of Web Form, such as: user controls, master pages, data binding, localization, etc. Deficiencies lie in that developers need more knowledge of html, css and java script; more codes need to be written in the server end and client.

C. Features of ASP.NET MVP

MVO mode makes View and Model completely decoupled, those two don't have any correlation and they communicate through Presenter. Presenter doesn't couple with specific View, but couples with an abstract View Interface, which is equivalent to a contract and abstracts methods corresponding to View. If this interface is realized, any View can be compatible with specified Presenter in order to achieve the seamless replacement of P Logic reusability and view. View is an "extremely thin" concept, for it only contains logic which can only maintain its own state, while other logic should be realized in the Presenter.

To separate the two concerns ---UI and P Logic using MVP mode to get a code structure which is more clean and single. Besides, this method has realized the seamless replacement of P Logic reusability and view and it's easier to realize it than MVC in structure.

The contrast of Web Forms, MVC and MVP characteristics is shown in Table 1.

IV. CONCLUSION

How to choose those three frameworks ---Web Forms, MVC and MVO, author thinks it should be decided according to the actual development needs. If you want to get more control of HTML, concerning about Web standard and accessibility, building up friendly SEO URL, integrating jQuery and other JavaScript development framework and carry on beta drive testing, you can choose MVC model. If you want to achieve a better control of system and realize state -oriented Web development of event drive, you can choose Web form model. However, the applications of MVC face with some certain technical problems, not as easy as WebForms. By disabling ViewState, WebForms can achieve the effect of a clean webpage and a small data transfer as MVC model. In addition to that, through setting up a temporary page cache, it can also realize the feature of no life cycle and fast creation speed as MVC model, too. If you do not want to have too much UI logic to avoid confusion, you can adopt MVP framework when the system demands for extension of C / S to B / S, etc.

REFERENCES

[1] Lin qing, Research on ASP.NET, MVC-based MVC design mode[J]. Computer Engineering and Design, 2008(1):167-169.

[2] Xie yan, An implementation of MVC Pattern in Web applications [J]. Computer Science, 2006 (5):

[3] Walkthrough: Creating a Basic MVC Project with Unit Tests in Visual Studio [EB/OL].[2008-10-15].

http://quickstarts.asp.net/previews/mvc/mvc_CreateMvcProjectWalkthrough.htm.

[4] Model-View-Presenter pattern practice in net platform, [EB / OL]. Zhang Yang blog: Net

http://www.cnblogs.com/leoo2sk/archive/2010/01/28/mvp-in-practice-based-on-dot-net.html

[5] ASP. NET development experience [EB / OL]. Zhao Jie blog: http://www.cnblogs. com/JeffreyZ hao/archive /2007 /12/22 /Experience-fo r-Asp- dot –net-and-WebForms.html

[6] Zhang enhui, *MVC pattern analysis under Asp.net* [J]. Silicon Valley, 2008 (18): 148.

[7] Peter D. Blackburn William (Bill) Vaughn. ADO. NET Examples and Best Practices for C# Programmers. New York: Apress,2006

[8] He lingjuan, Yi long, Liu lianchen. *To achieve web page in a kind of loosely coupled reusable MVC pattern* [J]. Computer Engineering and Applications, 2007, 43 (15).

[9] Michael Kircher,Prashant Jain.Pattern-oriented software architecture:Patterns for resource management, volume-3[M]. Hoboken, USA:John Wiley & Sons 2004.

[10] Martin Fowler. Patterns of enterprise application architecture[M].Boston,USA:Addison-Wesley,2002

Table 1 The contrast of Web Forms, MVC and MVP characteristics

	Client ID contamination	Coupling between View and Model	Support unit testing	Support multiplex in presentation layer logic	complexity
WebForms	yes	yes	no	no	low
MVC	no	yes	yes	no	high
MVP	no	no	yes	yes	Relatively high