WebForms vs. MVC

<https://www.codeproject.com/Articles/528117/WebForms-vs-MVC>

## Introduction

It’s been a long time I have been working as an ASP.NET developer and I do enjoy writing web apps using web forms.

In 2008 Microsoft came up with something called as **ASP.NET MVC** and I was quite amazed about the fact **“Why one more ASP.NET technology required”** and many people still pondering same.

|  |  |
| --- | --- |
| Many people say ASP.NET MVC replaced ASP.NET Web Forms. But that’s not true. Both have their own pros and cons Nobody can tell or teach us what to use and when, but we can discuss about some facts which will help us to make choice between both of them. And this article contains same. | Image 1 for WebForms vs. MVC |

|  |  |
| --- | --- |
| Image 2 for WebForms vs. MVC | We also try to find answers for some questions like,   * What is ASP.NET? * What is ASP.NET Web Forms? * What is MVC? * What is ASP.NET MVC? |

If you are advanced or experienced ASP.NET MVC developer, who already know MVC very well then this article will help you to revise your concepts.

## Index/Contents

* [Talk about Visual](https://www.codeproject.com/Articles/528117/WebForms-vs-MVC#Talk_about_Visual)
* [Web Technologies](https://www.codeproject.com/Articles/528117/WebForms-vs-MVC#Web_Technologies)
* [Visual in Web](https://www.codeproject.com/Articles/528117/WebForms-vs-MVC#Visual_in_Web)
* [What is ASP.NET?](https://www.codeproject.com/Articles/528117/WebForms-vs-MVC#What_is_ASP.NET)
* [What is Web Forms?](https://www.codeproject.com/Articles/528117/WebForms-vs-MVC#What_is_Web_Forms)
* [ASP.NET 4.0](https://www.codeproject.com/Articles/528117/WebForms-vs-MVC#ASP.NET_4.0)
* [What is MVC?](https://www.codeproject.com/Articles/528117/WebForms-vs-MVC#What_is_MVC)
* [What is ASP.NET MVC?](https://www.codeproject.com/Articles/528117/WebForms-vs-MVC#What_is_ASP.NET_MVC)
* [Why ASP.NET Web Forms and Why ASP.NET MVC?](https://www.codeproject.com/Articles/528117/WebForms-vs-MVC#Why_ASP.NET_Web_Forms_and_Why_ASP.NET_MVC)
* [Conclusion](https://www.codeproject.com/Articles/528117/WebForms-vs-MVC#Conclusion)

## Talk about Visual

|  |  |
| --- | --- |
| Image 3 for WebForms vs. MVC | Microsoft initially came up with something called **Visual**. With the help of technologies like Visual Basic, Visual C++ Microsoft enabled RAD (rapid application development) of Graphical User applications. Features such as Drag and Drop and intellisense made developers to focus more on the business functionality of the application rather than on UI design.  But this technologies were limited to desktops, when it comes to web the only option left with Microsoft was ASP. |

## Web Technologies

When we say web technologies, we have classic ASP, php, jsp,ROR, ASP.NET Web Forms, ASP.NET MVC and many more. Classic ASP is one of the web technology introduced by Microsoft. Biggest pain point with the classic ASP was spaghetti code and maintainability. Let assume a scenario where you have some text boxes and a button. On button click you validate the data with the server and if validation succeeds data will be stored into database and in case it fails, error message will be shown to user in the form of red colored label. You know what’s the biggest problem with this scenario is? You have to do lots of stuffs by your own.

1. First make the self-post back by setting form’s action attribute’s value to same page.
2. Text box values are going to be cleared on button click, so only choice left will be retrieving values from posted data.
3. In case validation fails you have to explicitly
   1. Set all values back to the corresponding text boxes from posted data
   2. Show error message.

(Using Ajax is an alternate method. Here I was trying to explain the manual work need to be done with classic ASP)

## Visual in Web

Technologies like Visual Basic are limited to standalone applications, when it comes to web, only option left with Microsoft is classic ASP.

When we talk about Web and Desktop, two things which should be considered are

1. How state management works?
2. How request/response works?

Web works on HTTP which is completely a stateless protocol. We will be having request and response independent of previous request and response. Unlike desktop no variable values will be preserved and no complete event driven programming model. Like desktop it will wait for user inputs but every user input and interaction act as a new request (get/post) to server.

Finally Microsoft came up with something called ASP.NET Web Forms, considering rapid application development and easy learning in priority.

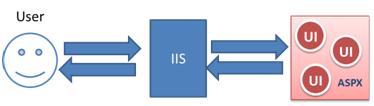
## What is ASP.NET?

ASP.NET is a Microsoft’s Web application framework built on Common language runtime for building dynamic web sites using one of the programming languages like C#, VB.NET etc. It supports 2 models Web Forms and ASP.NET MVC.



## What is Web Forms?

Microsoft first brought out ASP.NET Web Forms from ASP which solved lots of problems by creating higher level abstraction over stateless web and simulated stateful model for Web developers. In web forms concepts like self postback (post form data to same page) and ViewState (maintain control values during postbacks) are introduced. And the most interesting part is it’s not required to write even a single line of code. With Web Forms Microsoft tried to bring the Visual Basic model into web.

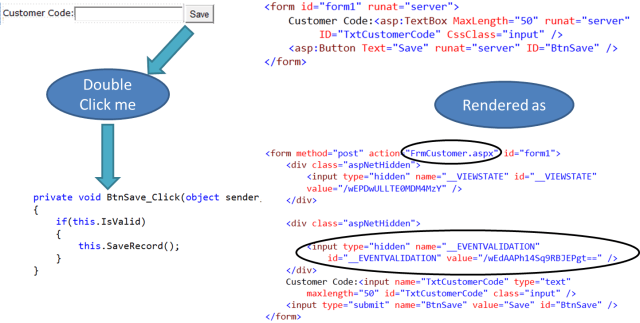


Let’s talk about advantages and disadvantages of Web Forms.

### Advantages:

* Web Forms supports **Rich server controls.**

|  |  |
| --- | --- |
| * Image 6 for WebForms vs. MVC | * + While working with pure HTML you might have noticed, Things are not always same at all place. A UI which looks great in IE might distract in Firefox or vice versa. ASP.NET server control detects the browser and generates appropriate html and if required JavaScript also.   + Many server controls like GridView and ListView comes up with data binding capabilities reducing lots of efforts and codes being written. |

* **Support for ViewState -** You might have heard couple of times “HTTP is a stateless protocol”. Normally controls will not retain their values between requests. But in Web Forms statefulness is achieved by storing last known state of every control within the client page itself in the form of hidden field called ViewState.
* **Event driven programming -**   
    
    
  With the help of
  + Code behind
  + Self postback mechanism (posting back form to the same page)
  + ViewState

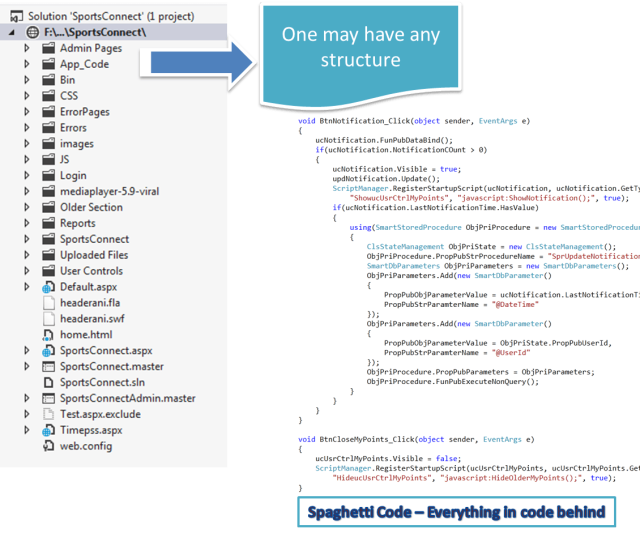
Microsoft introduced event driven programming in internet world.

Developer will no more rely on POST, GET methods for handling user interactions with server. For example she/he will drag control (say button) to page, just double click it to generate the code block for handling user’s click on server, write down the logic inside it. That’s it. She/he is not concerned with what happens inside.

* **Rapid application development** - I don’t think any explanation is required for this. Rich server controls, Event driven model and ViewState increases the development speed by great extent, Developer will be abstracted from lots of the background complexities.
* **Less learning effort.** - Using strong server controls and ViewState developer can develop real world applications with minimal HTML and JavaScript skills.

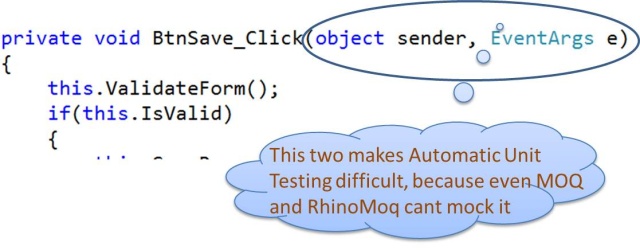
### Disadvantages:

* **Project Architecture**



There is no fixed predefined Project Architecture for creating web applications when it comes to Web Forms. Developers have full flexibility for choosing their own architecture.   
One may use basic three layered architecture dividing the system into UI, Business layer and Data access layer or a more advanced one like Model-View-Presenter. Even one may choose only code behind and write everything there which is not at all considered as good practice. Code behind is tightly connected to UI, ending up with some presentation logic.

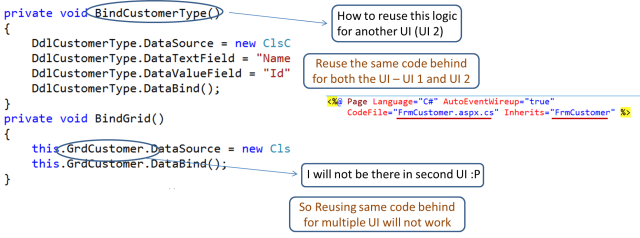
* **Unit Testing**

In Web Forms code behind ends up with lots of event handlers, making automatic unit testing an impossible task.   
**Note:** as per my knowledge even with the help of mock testing (using MOQ or rhinomoq) we can’t mock ‘sender’ and ‘eventargs’ in event handlers.  
  
And when we talk about employing TDD, unit testing code behind (presentation logic) becomes very important.

* **Performance**

ViewState becomes solution for some problems with classic ASP but it also becomes an issue. ViewState is stored in the page itself resulting increased page size so reduced performance.

* **Reusability**



Let’s talk about another example where we are supposed to build 2 UI

* + Taxable Employee screen
  + Nontaxable employee screen

Now most of the code behind logic is going to be same for both screens.   
One solution will be, add some if conditions in code behind and create a single UI.

* + This will violate Single Responsibility principle which says there should be only one reason a software entity to be changed, because in this case employee form will be changed whenever any of the taxable/nontaxable information changes.
  + Secondly it may possible that both UI
* **Less Control over HTML -** In Web Forms many times we are not sure about what html we will get at the end making integration with JavaScript frameworks like jQuery a difficult task
* **SEO** –

Image 11 for WebForms vs. MVC  
URL’s are pointing to fixed ASPX pages which might be decorated with some query string. They are not user friendly and affect SEO.

* **Less support for parallel development** - ASPX page are tightly coupled with code behind files. So it’s not possible that 2 different developers are working on one section (one on aspx and one on code behind) at same time.

## ASP.NET 4.0

ASP.NET 4.0 has come up with some good features to overcome some of the above problems

* **ViewState:** Provide the way to disable or control the size of the ViewState. (But there is no compulsion or fixed rule that will say to do so.)
* **URL routing:** Using URL routing we can provide our own URL in lieu of the physical path of the page.
* **ID:** In ASP.NET 4.0 we have better control over Id of elements and thus integration with JavaScript framework become easier. (We still don’t have complete control over HTML being generated.)

Even after the evolution of the revolutionary features of ASP.NET,

1. It was still not able to solve the problems of unit testing
2. I never saw any ASP.NET Web Form developer who tries to disable the ViewState. Smile | <img src=(It’s my perspective, no hard feelings please.)
3. We got some control over ID of elements, but not complete control over HTML, still have problem to implement JavaScript.

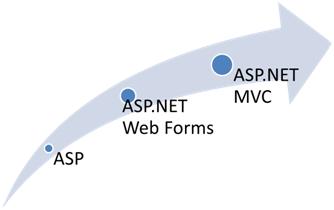
## What is MVC?

MVC is an architectural pattern which is has been around for sometimes now.  
Many are using it including Java. It’s not new concept which Microsoft brought it up. ASP.NET MVC is something we should talk about. But prior to that lets clear some terminologies including MVC.

* **Patterns -** In simple words Pattern is a solution to a problem in a context.
* **Architectural Patterns -** Architectural Pattern is something which solves our problem at sub system level or in short module level. It deals with the problem related to architecture of a project. It tells us how we can divide our systems and especially why. We make Class libraries, Components, Web services to solve the problem.
* **MVC** When we talk about application we will be having input logic, business logic and UI logic and MVC is an architectural pattern which let us develop an application having loosely coupling between each of these elements.  
  The main intention behind MVC pattern is separation of concerns. It makes presentation or UI ignorant of business and user interaction logic.  
  According to MVC system should be divided as M (Model), V (View) and C (Controller).

|  |  |
| --- | --- |
| Image 13 for WebForms vs. MVC | * Model is considered as smart and handles the Business rules, logic and data and will be independent of other parts of MVC (controller and View). * Controller receives and dispatches the request in short it handles the user interaction and input logic. It knows about both Model and View. * A view is considered as dumb and is an output representation of model data. It may be an excel sheet, a web page showing list of records or just a simple text. View knows about only Model. |

## What is ASP.NET MVC?



ASP.NET MVC is a Microsoft’s one more Web application framework designed with separation of concerns and testability in mind. It is built on CLR and completely based on MVC architecture and so we think in terms of controllers and Views. ASP.NET doesn’t have support for ViewState and server controls, so we get feel of old web here. Let’s talk about Advantages and disadvantages of ASP.NET MVC

### Advantages:

* **Project architecture -**

|  |  |
| --- | --- |
| Image 15 for WebForms vs. MVC | One of the advantages of using ASP.NET MVC is it enforces separation of concerns. So there is very less chances of getting things more complex. |

* **Test Driven development and Reusability** –

|  |  |
| --- | --- |
| * + In MVC controller is a separate class so automatic testing is possible featuring Test Driven Development.   + Controllers are not bound to any specific view and so can be reused for multiple views. | Image 16 for WebForms vs. MVC |

* **Performance** - ASP.NET MVC don’t have support for view state, so there will not be any automatic state management which reduces the page size and so gain the performance.
* **Full control over HTML** - ASP.NET MVC doesn’t support server controls, only option available is using html input controls, so we will be sure about final html rendered at the end. We will also be aware about 'id' of every element. And so integration of ASP.NET MVC application with third party JavaScript libraries like jQuery becomes easy.
* **Support for parallel development** - In ASP.NET MVC layers are loosely coupled with each other, so one developer can work on Controller ,at the same time other on View and third developer on Model. This is called parallel development.
* **SEO, URL routing and REST** - Rich routing features lets treat every URL as a resource supporting RESTful interfaces.

Image 17 for WebForms vs. MVC

Also user-friendly and readable URL improves SEO.

* **Extensibility -** ASP.NETMVC supports multiple view engines like aspx, razor and if required we can create our own.
* **Existing ASP.NET Features –** ASP.NET MVC framework is built on top of matured ASP.NET framework and thus provides developer to use many good features such as forms authentication, windows authentication, caching, session and profile state management etc.

### Disadvantages:

* **More learning effort -** Absence of event driven programming model and ViewState makes ASP.NET MVC a very difficult framework for developers with no or little experience in web application development.

## How ASP.NET MVC works



1. User makes the request for some resource in server (by putting some URL in the browser).
2. Request comes to controller first (routing engine is the one who is responsible for deciding which request will be handled by which controller. In this article we won’t talk in depth about this behavior).
3. Controller if required talk to model for data.
4. Model operates on database (or on some other data sources) and return data (in form of business objects) to controller.
5. Controller chooses the appropriate view (like say Customer view which will may contain some html tables, drop downs, textboxes…).
6. Controller passes the data (model data retrieved in step 4) to chosen view(in step 5), where data will be populated as per convenience.
7. Controller sends back view to the user.

This was for get request, same happens for post. Only instead of putting URL in the browser user will do some action on already requested page and flow start with the controller. Actions like clicking button, changing drop down value etc.

## Why ASP.NET Web Forms and Why ASP.NET MVC?

Each can be the “best choice” for a particular solution depending on the requirements of the application and the background of the team members involved. What to choose and when has more to do with business prospective than which one is better than other. When facing a decision to choose between ASP.NET Web Forms or ASP.NET MVC it is important to know that neither technology is meant to replace the other.

Two important factors you should consider while making the choice is

1. Rapid application development - If you want to develop anything rapidly ASP.NET Web Forms is the only chance you are having, you can’t even consider for ASP.NET MVC for RAD. (Reasons for RAD may be anything like client is not paying too much, or application is going to be used for only one or two months and won’t require much maintenance.)
2. Unit Testing - If automatic unit testing is most important factor for you MVC will be best for you.

Other than these, what you can do is, write down all your project requirement and try to compare them with Pros and Cons of both Web Forms and MVC and if possible try to ask yourself following questions and point MVC and Web Forms accordingly

1. Does your team have good experience with Web Forms or Windows Forms?  
   Well, if yes then probably learning ASP.NET MVC is going to be a tedious task for team, because developers have been used to with ViewState and event driven programming by now and migration is going to be a difficult task.1 point to Web Forms.
2. Does your team have good experience with ASP.NET MVC?  
   If yes ASP.NET MVC get 1 point
3. Does your team have experience on ASP or non-Microsoft technologies such as android, ios, JSP, ROR, PHP?  
   If you have been JSP or ASP developer prior then you might be familiar with HTTP get and Post and even u might have hands on with MVC because most of them use MVC by default. It gives 1 point to ASP.NET MVC.
4. Is JavaScript going to be used extensively?  
   If Yes, MVC gets the point because you get complete control over HTML. 1 point ASP.NET MVC.
5. Looking for good performance?  
   With no support for ViewState ASP.NET MVC provides good performance gain over traditional ASP.NET Web Forms.1 point ASP.NET MVC.
6. Planning to reuse the same input logic?  
   If yes stick with MVC.

## Conclusion

I think you should have equipped with enough information to make a decision what is best for your project. The complete decision depends on your team and project requirement.

Hope all of you enjoyed reading this article. Thank you for the patience.

For technical training related to various topics including ASP.NET, Design Patterns, WCF and MVC contact SukeshMarla@Gmail.com or at [www.sukesh-marla.com](http://www.sukesh-marla.com/)

For more stuff like this click [here](http://www.sukesh-marla.com/). Subscribe to [article updates](https://www.facebook.com/pages/Blogs-By-Sukesh-Marla/168078149903213?bookmark_t=page) or follow at twitter [@SukeshMarla](https://www.codeproject.com/Articles/528117/twitter.com/SukeshMarla)

Also go through our complete [step by step series of MVC(Model View Controller)](http://www.codeproject.com/Articles/207797/Learn-MVC-Model-View-Controller-step-by-step-in)

Click here for see [MVC video step by step](http://youtu.be/KAKxm4eQP24).