Web Forms is NOT Dead!

It seems like everywhere you read, everyone is talking about using ASP.NET MVC to create mobile web applications. But what about programmers still working in ASP.NET Web Forms? We want our projects done using the latest techniques, we want to build mobile web apps, and we want to use jQuery and bootstrap too. Not to fear, Web Forms is not dead, and nearly everything you can do in MVC, you can also do in Web Forms. I am creating a whole series of videos on how to use all the latest techniques in Web Forms to build modern web applications. This series of videos will be available in the summer of 2014 on www.pluralsight.com.

MVC vs Web Forms

Many programmers feel they have to choose one technology over another. But, there is really no reason you can't program in both MVC and Web Forms and use each for what they are good at.. Many MVC programmers give many reasons why you should use MVC as opposed to Web Forms. The reasons I hear more often are:

1. Fine control over the HTML generated
2. Ability to unit test
3. Can control the "id" attribute generated
4. Easier to use jQuery and JavaScript because no generated "id" attribute
5. Ability to use Friendly URLs
6. No ViewState sent to client

Of course, programmers that like Web Forms have many reasons why they want to stick with it:

1. Rapid Application Development
2. Less code to write because of rich server controls
3. Lots of server controls to choose from
4. Great third-party support
5. Easy to preserve state
6. They like the event driven model
7. Hides the complexity of the web
8. More mature so there is more information with respect to questions, problems, etc.

In this article let's explore how Web Forms will let you get very close the MVC model. There are many things you can take advantage to get Web Forms to work almost exactly the same as MVC, and you do it using the tools you already know and love.

HTML 5 and CSS 3

In any Web Form page you can use HTML 5 elements and attributes. Even if you are using a server control you can still add any attribute to the control and it will emit the attribute for you. CSS 3 is just a new version of CSS and we have always been able to use CSS with Web Forms, so there is no difference here. The TextBox control now supports the HTML 5 input types such as date, tel and email through the TextMode property.

ViewState

Most Web Form pages do not need ViewState to work property. There are just a few cases when you will need to turn on ViewState. I recommend disabling ViewState in the Web.Config and then just turn it on when you get to a page that just does seem to work right. By turning this off you will see an increase in performance because not as much HTML is sent down to the client.

Store ViewState on the Server

If you still need to use ViewState, but you do not want the extra hidden field to be sent to the client, you can easily store ViewState on the server. This functionality has been available since .NET 1.0. You simply override two events in the base page class; LoadPageStateFromPersistenceMedium and SavePageStateToPersistenceMedium. These two events will allow you to store and restore the view state on the server and not send the hidden field to the client.

Friendly URLs

Using "RESTful" or so-called SEO-friendly URLs is all the rage today. What is nice about these URLs is it makes for a cleaner query string, the user does not know the actual page name and is typically easier for users to access and remember.

Instead of these…

www.url.com/Products.aspx

www.url.com/Products.aspx?ProductId=3

Use Friendly URLs instead…

www.url.com/Products

www.url.com/Products/3

To add friendly URLs to your project simply go to **Tools** | **Nuget Package Manager** | **Manage NuGet Packages for Solution...** Search online for "Microsoft.AspNet.FriendlyUrls" and install the "Microsoft.AspNet.FriendlyUrls.Core".

Add a class called RouteConfig and add the following using statements at the top of the file:

```
using System.Web.Routing;
using Microsoft.AspNet.FriendlyUrls;
```

Now add a method named RegisterRoutes.

```
public static void RegisterRoutes(RouteCollection routes)
{
   routes.EnableFriendlyUrls();

   routes.MapPageRoute("", "Default", "~/Default.aspx");
}
```

You can add as many MapPageRoutes as you want in this method. You just have to keep the first and second parameters unique.

Now in your Global.asax you will add a using statement at the top of this file:

```
using System.Web.Routing;
```

In the Application_Start() event you will call the static method you created in the RouteConfig class.

```
RouteConfig.RegisterRoutes(RouteTable.Routes);
```

MVVM

I have written a blog post and an article on using the Model-View-View-Model approach to software development. If you use MVC or Web Forms you should be using a View Model. The View Model is where all of your data access happens and all the code to control the UI should also be located. The controller in MVC calls the View Model and the code-behind in Web Forms calls the View Model. This means that all unit testing happens on the View Model and you do not need to test against a controller or the code-behind file. You can read more about this model at the following two links.

http://weblogs.asp.net/psheriff/the-basics-of-mvvm

http://www.codemag.com/article/1208031

Unit Testing

As just mentioned above, if you use the MVVM design pattern you get the benefits of being able to do unit testing and take advantage of TDD if you so desire.

jQuery and JavaScript

Using jQuery and JavaScript is an absolute must when building today's modern web applications. Web Forms used to be criticized because when it generated the HTML controls, it "munged" the id attribute. This means that the id that you used in your ASPX page was something different when it ended up on the client. This makes it hard to know what the name is when you want to reference that control in jQuery or JavaScript.

Microsoft gave us the ClientID property and the UniqueID properties to access the id and name attributes respectively. However starting with .NET 4.0 they added a ClientIDMode to the page level and to the Web.Config file. This allows you to set the ClientIDMode="Static" and the id you set in your ASPX page is what will be generated on the client. This makes integrating with jQuery and JavaScript just as easy as it is with MVC.

Bootstrap

Twitter Bootstrap is a very powerful and popular CSS and JavaScript library that allows you to create responsive web sites. We have been using bootstrap for a few years now and it works just fine in Web Forms. We have successfully implemented templates we purchased from wrapbootstrap.com and themeforest.net. We typically take these templates and integrate the navigation and other elements into our Web Forms master pages. We then build very nice looking responsive web applications using Web Forms.

GridView

Web pages love tables! However tables are not always a good thing on smaller devices like a smart phone. Using bootstrap styles in combination we can make the GridView work much better. A better approach is to use the GridView so you get the built-in paging and all the great features of the GridView but make it not look so tabular. I wrote a blog about how to create an alternate view of tabular data. Check out this approach on how to present data that will work better on a mobile device:

http://weblogs.asp.net/psheriff/an-alternate-approach-to-a-gridview

Additional Guidance

One thing I like to do is consider what other folks are saying about Web Forms vs MVC. If you look at the following names, you can see what some of the heavyweights in the industry have to say.

Scott Guthrie

“Web Forms and MVC are two approaches for building ASP.NET apps. They are both good choices.”

http://weblogs.asp.net/scottgu/archive/2010/01/24/about-technical-debates-both-in-general-and-regarding-asp-net-web-forms-and-asp-net-mvc-in-particular.aspx

Dino Esposito

“ASP.NET MVC doesn't magically transform every developer into an expert architect and doesn't prevent developers from writing bloated and poorly designed code. At the end of the day, both Web Forms and ASP.NET MVC help to build applications that are designed and implemented to deal effectively with the complexity of real-world solutions”

http://msdn.microsoft.com/en-us/magazine/dd942833.aspx

Jeffrey Palermo

“It is rarely a good idea to trash your current application and start over”

http://clear-measure.com/i-have-a-web-forms-custom-application-should-i-upgrade-to-asp-net-mvc-now-or-wait/

K. Scott Allen

"…figure out for yourself what framework will be the best for you, your team, and your business."

http://www.odetocode.com/blogs/scott/archive/2013/02/12/you-want-to-build-web-software-with-c.aspx

Microsoft

A good overview of which to use when.

http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview

Public Sites Favors MVC

What we have gathered from our own experience and from reading what others are using MVC for are the following types of sites.

- Blogs
- Web content
- Marketing
- Membership
- Shopping
- Mobile

Business Apps Favor Web Forms

We have found that for building business applications that are doing a lot of CRUD operations that Web Forms lends itself really well to these types of applications.

- Line-of-business
- SaaS
- Extranet

Summary

In the end it is up to you which approach you are going to use when developing your web applications. But you should not let folks that are clearly biased toward MVC sway your choice. Web Forms is still a great choice for developing web applications. Just because Microsoft releases a new technology does not mean that you should immediately trash everything you know and jump on it.

As you have seen in this article, Web Forms and MVC are based on the same underlying technology and both can generate fast, small, responsive web applications. At the same time both can be unit tested, take advantage of MVVM, HTML 5, CSS 3 and jQuery libraries. So don't throw away all your hard-earned skills, just take advantage of the tricks in this article and develop modern web applications with Web Forms.