ASP.NET MVC vs. Web client software factory (WCSF)

Asked 16 years, 3 months ago Modified 12 years, 11 months ago

Viewed 8k times

15

I have recently been doing a bit of investigation into the different types of Model View architectures, and need to decide which one to pursue for future in-house development. As I'm currently working in a Microsoft shop that has ASP.NET skills, it seems my options are between ASP.NET MVC and WCSF (Monorail is probably out of the as it wouldn't be supported by Microsoft).

After reading the ASP.NET MVC framework, using the WCSF as a yardstick, I picked up the following points:

- ASP.NET MVC cannot use web controls that rely on postbacks, whereas WCSF can.
- You have more control over the urls in an ASP.NET MVC site as opposed to a WCSF site.
- An ASP.NET MVC site will probably be easier to test than an equivalent WCSF version.
- It seems that the WCSF still uses the code behind to control UI events under some circumstances, but ASP.NET MVC doesn't allow this.

What are some of the other considerations?
What have I misunderstood?
Is there anybody out there who has used both frameworks and has advice either way?

asp.net asp.net-mvc wcsf

Share
Improve this question
Follow

edited Sep 10, 2008 at 7:14

asked Sep 10, 2008 at 5:37
Bermo
4,931 ● 1 ● 30 ● 32

7 Answers

Sorted by: Highest score (default)

15

> ASP.NET MVC cannot use web controls that rely on postbacks, whereas WCSF can.

You should think of WCSF as guidance about how to use the existing WebForms infrastructure, especially introducing Model-View-Presenter to help enforce separation of concerns. It also increases the testability of the resulting code.

> You have more control over the urls in an

> ASP.NET MVC site as opposed to a WCSF site.

If you can target 3.5 SP1, you can use the new Routing system with a traditional WebForms site. Routing is not limited to MVC. For example, take a look at Dynamic Data (which also ships in 3.5 SP1).

> An ASP.NET MVC site will probably be easier to test than an equivalent WCSF version.

This is true because it uses the new abstractions classes for HttpContext, HttpRequest, HttpResponse, etc. There's nothing inherently more testable about the MVC pattern than the MVP pattern. They're both instances of "Separated Presentation", and both increase testability.

> It seems that the WCSF still uses the code behind to control UI events under some circumstances, but ASP.NET doesn't allow this.

In Model-View-Presenter, since the outside world interacts with views (i.e., the URL points to the view), the views will naturally be responding to these events. They should be as simple as possible, either by calling the presenter or by offering events that the presenter can subscribe to.

Model-View-Controller overcomes this limitation by having the outside world interact with controllers. This means your views can be a lot "dumber" about non-presentation things.

As for which you should use, I think the answer comes down to which one best suits your project goals. Sometimes WebForms and the rich third party control vendor availability will be preferable, and in some cases, raw simplicity and fine-grained HTML control will favor MVC.

Share Improve this answer Follow

answered Sep 10, 2008 at 6:14
Brad Wilson
70.4k ● 9 ● 77 ● 85

6

Not to start a flame war but I found the WCSF to be quite convoluted. The elegance and simplicity of MVC blows away MVP which feels like a pattern that is just grafted onto webforms.

Share Improve this answer Follow

answered Dec 17, 2009 at 18:02
Jeff
1,201 ● 1 ● 11 ● 17

2

We opted for the WCSF after doing exactly the same evaluation. We felt that the MVP pattern gave us more options i.e Ability to use server controls. Our development team is mostly made up of Programmers from a myriad of disciplines i.e C++, Biztalk and web etc. but had all focused mostly on MS type development, so the learning curve in adopting the patterns was not so much for our team.

We are more than happy with our choice.

Share Improve this answer Follow

answered Feb 4, 2009 at 10:01

Gary Woodfine
361 ● 1 ● 5 ● 13

are you still happy with this decision? It seems like WCSF is phased out by MS while MVC is getting more popular. Thanks. – Yosep Kim May 28, 2014 at 15:16

1

You might also consider the background's of your developers (if any have already been identified).

If they come from a strict asp.net background, they will be more comfortable around WCSF (although in my experience, it still took them a few weeks to really be comfortable around MVP).

If they come from a java/rails background, or have used other MVC architectures before, then obviously they'll be happier there (and in my experience get very snooty about anything other than MVC).

Share Improve this answer Follow

answered Oct 9, 2008 at 2:56

Kevin Dostalek
746 ● 1 ● 5 ● 8

1

MVC is a much simpler paradigm and is more similar to how all other frameworks do web development. WebForms is simply way too much jumping through hoops and too many layers of abstraction to try and achieve simplicity. IMHO, MVC will be the default ASP.NET architecture within a few years, as more and more people realize the simplicity and ease of development and testing that it brings with us. I have been doing MVC development for a year and a half and would never even think of going back to WebForms on a new project.

Share Improve this answer Follow

answered Dec 17, 2009 at 18:13
Keith Rousseau
4,475 ● 1 ● 23 ● 28

1

> **An ASP.NET MVC site will probably be easier to test than an equivalent WCSF version.**
>
> This is true because it uses the new abstractions classes for HttpContext, HttpRequest, HttpResponse, etc. There's nothing inherently more testable about the MVC pattern than the MVP pattern. They're both instances of "Separated Presentation", and both increase testability.

This is probably debatable, but there is literature to suggest using an MVP design model is easier to unit test then an MVC design model if you have Views that are packed with logic. To summarize, in the MVP design model the Presenter is handling the work that might be handled by the View in the MVC design model. The logic that might be contained in the MVC View does not facilitate unit testing. Here are some references to literature that I have read that would cover this concept and the reasons why keeping your View light is better for many reasons including facilitating unit testing.

http://martinfowler.com/eaaDev/uiArchs.html

http://martinfowler.com/eaaDev/SupervisingPresenter.html

http://martinfowler.com/eaaDev/PassiveScreen.html

Share Improve this answer Follow

answered Jan 25, 2012 at 17:45

JC wavedrop
530 ● 1 ● 4 ● 15

0

Why not attach both to Northwind and see which fits best for you and your situation?

Share Improve this answer Follow

answered Sep 10, 2008 at 6:34

Ant
1,156 ● 11 ● 15