MVVM best practices: folders and namespaces

General discussion

Hi, I am about to cleanup a big solution from another developer to a more clean MVVM pattern. 2 questions regarding best practices (I am curious what most of you use):

1. Most samples seem to have Views and ViewModels in separate folders (mine hasn't). When the solution and projects grow large, the projects usually already have subfolders. What would you recommended for the folder structure?
   Solution
   + Project 1
   + Properties
   + References
   + Converters
   + Views
   + Module 1
   Module1FirstView.cs
   Module1SecondView.cs
   + Module 2
   + ViewModels
   + Module 1
   Module1FirstViewModel.cs
   Module1SecondViewModel.cs
   MyModule1.cs
   + etc
   + Project 2
   Or:
   Solution
   + Project 1
   + Properties
   + References
   + Converters
   + Module 1
   + Views
   Module1FirstView.cs
   Module1SecondView.cs
   + ViewModels
   Module1FirstViewModel.cs
   Module1SecondViewModel.cs
   MyModule1.cs
   + Module 2
   + Views
   + ViewModels
   + etc
   + Project 2
2. Most MVVM projects I've come across kept the namespaces proposed by Visual Studio, so the Views come in namespace [somespace].Views and the ViewModels in [somespace].ViewModels. I've always found this a bit overkill. Do you agree that its ok to cut of the Views/ViewModel distinction?
3. Are there any other naming conventions that are considered to be 'best practice'? Any good reference sites?

Thanks!

1

Sign in to vote

Changed type Dropmans Thursday, October 20, 2011 12:03 PM Best practices are more discussion related

Thursday, October 20, 2011 12:02 PM

Reply | Quote

All replies

2

Sign in to vote

1. No, I do not have separate folders for Views and ViewModels instead I place a view and it's related interfaces and ViewModels in a folder.

Module 1
    + Views
        +ViewA
            ViewA.xaml
            ViewA.cs
            IVewA.cs
            IViewAViewModel.cs
            ViewAViewModel.cs
        + ViewB
            .....

This makes it much easier to navigate all code relate to a View you are working on. Instead of hopping around disperse folder structure in your project.

2. I never keep the namespaces proposed by VS. Namespaces are for API, folders are for organization. They shouldn't match.

3. You conventions will come as you develop code. You will start to identify how you like things to be.

Edited by Brian Lagunas MVP Thursday, October 20, 2011 2:31 PM

Thursday, October 20, 2011 2:29 PM

Reply | Quote

0

Sign in to vote

Agreed that placing Views and ViewModels in the same folder makes for easier navigation, but it feels less clean to me. I don't really like it.

I also agree with your comment about namespaces. But do you put views and viewmodels in different namespaces?

Friday, October 21, 2011 10:14 AM

Reply | Quote

1

Sign in to vote

Personally, I find that when I'm working on some screen I want to work on the view and the viewmodel.

So I keep them together.

If the app is small then I might not bother with extra folders.

I either stick everything in a views folder and use naming to group them or where that's not possible I have a subfolder per form.

Much like Brian.

Personally, I don't like loads of namespaces.

The less the better, unless there's some good reason.

Friday, October 21, 2011 10:49 AM

Moderator

Reply | Quote

0

Sign in to vote

I don't put views and ViewModels in different namespaces, as there is no point to it. The main reason for namespaces is to avoid class name collisions. Of course, they are also used to organize code from an API perspective, not a folder perspective.

I'm with Andy on this one. You shouldn't go overboard on namespaces. Keep them simple and easy to discover. Keep related classes in the same namespace. As in this example, the Views and ViewModels have a close relationship and I like to keep them in the same namespace. It's bad enough trying to find ViewModels that are in different folders in a large application let alone trying to find them through a separate API.

Of course, everyone has their preferences. The key is to find a convention you like and stick to it for the life of the app.

Friday, October 21, 2011 2:06 PM

Reply | Quote