Building a Monarch Application from Multiple Gameplans
ASNA Monarch® 3.1

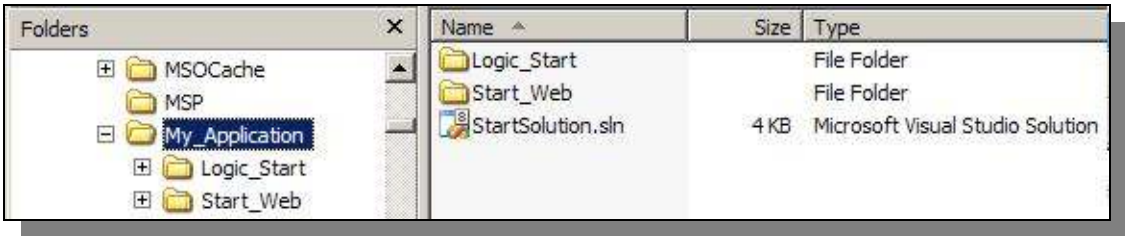
One of the challenges to a Monarch *Migrator* is to build a .NET application from a number of migrated class libraries. Generally, this would not be considered a major challenge with a Windows application. However, the structure and security imposed upon a Web Application increases the complexity of development a little bit.

This document seeks to provide some guidance to the Migrator for converting GamePlans to a meaningful collection of Class Libraries under a single Web Site. Be aware the illustrations employed here do not describe the only mechanism for migration. Rather, it is the hope that the revelations contained here will assist the Migrator to better understand the environment.

By design, Monarch migrates Website GamePlans for testing using the *File System* location (Cassini).

Startup Solution

One mechanism for migration of the Web portion of the application is to begin with a startup or "root" web site that can be built upon with newly migrated GamePlans. A startup Website is provided in Monarch examples[1] which will be used in this document. All you need to do is to copy the folder to the location where you wish to develop your application. In this example, ***Monarch_Start*** was renamed to ***My_Application***.

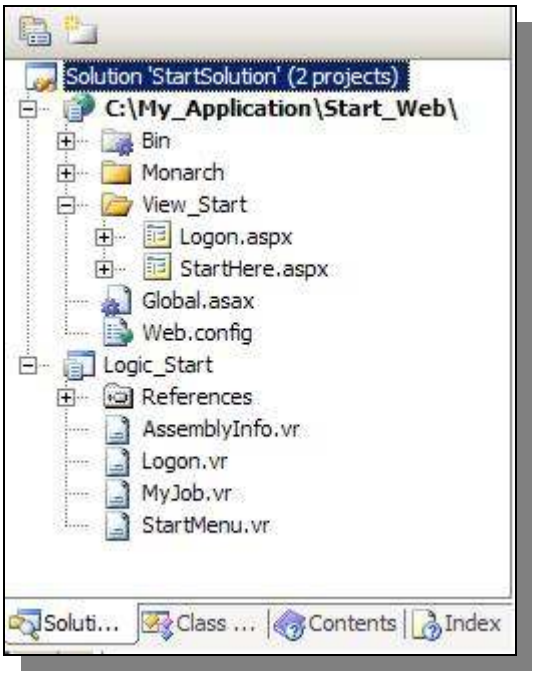


My_Application

StartSolution consists of two programs.

1. **Logon.vr** - provides a shell for user logon.
2. **StartMenu** - provides a shell for initiating tests of major portions of your application.

[1] C:\Program Files\ASNA\Examples\Monarch 3.1\Monarch_Start



StartSolution

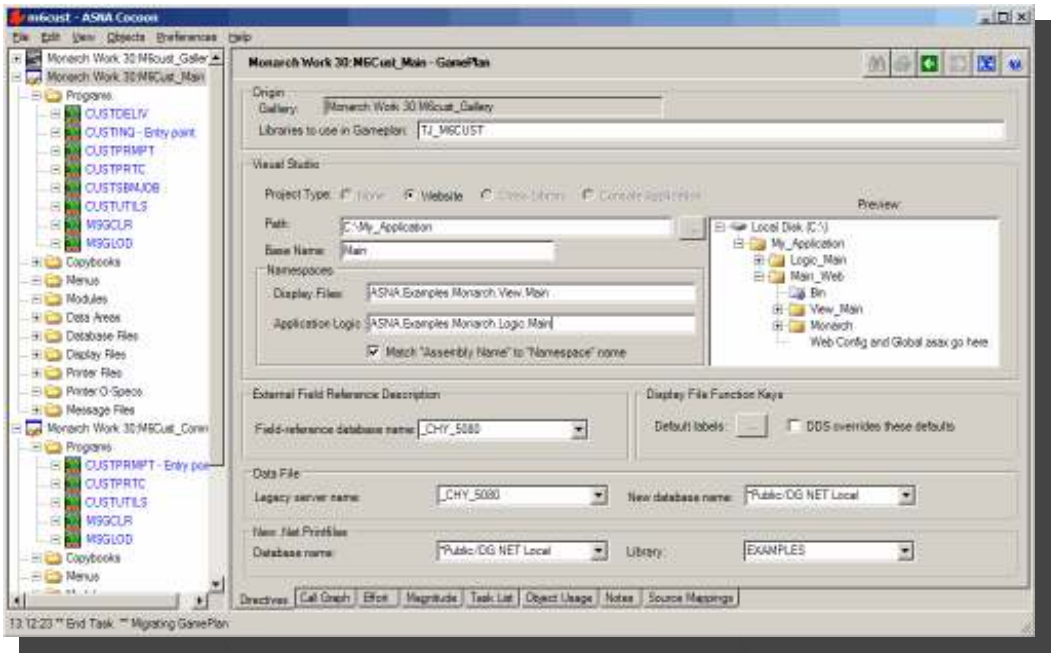
This exercise migrates two imaginary interactive GamePlans:

1. *M6Cust_Main* (“Main”)
2. *M6Cust_Common* (“Common”)

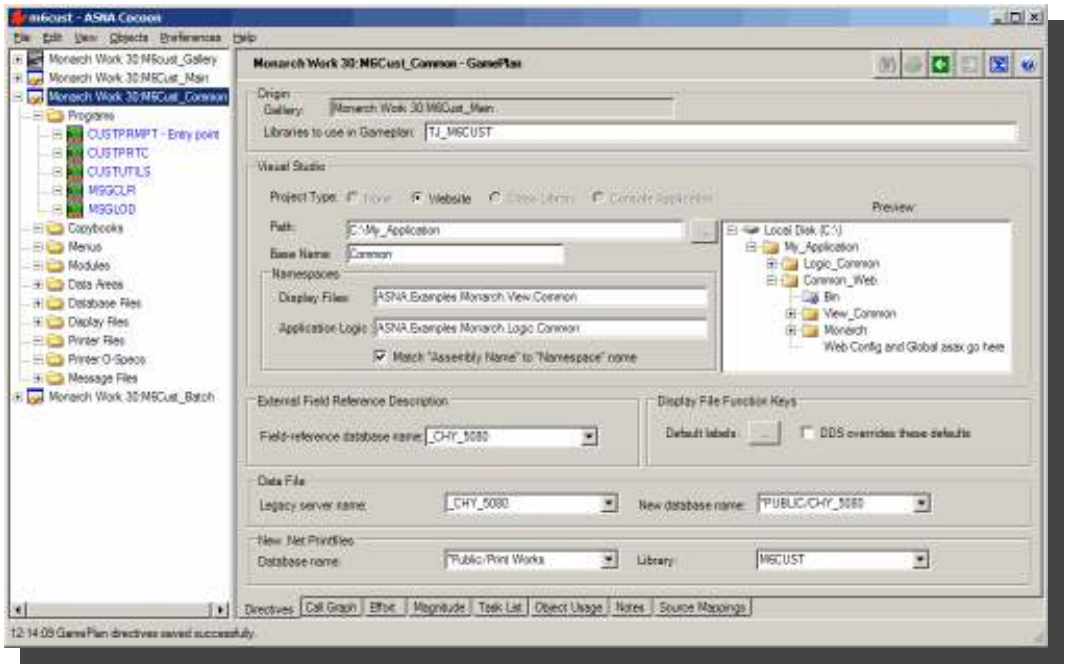
The figures below show the Monarch directives for “Main” and “Common”, in addition to the Video Studio solution migration directives defaults.

Your actual GamePlans will be different from those used in this narrative and they will also have different names. “Main” and “Common” represent a typical migration scenario. Therefore, substitute your own GamePlan and folder names for those used in this illustration.

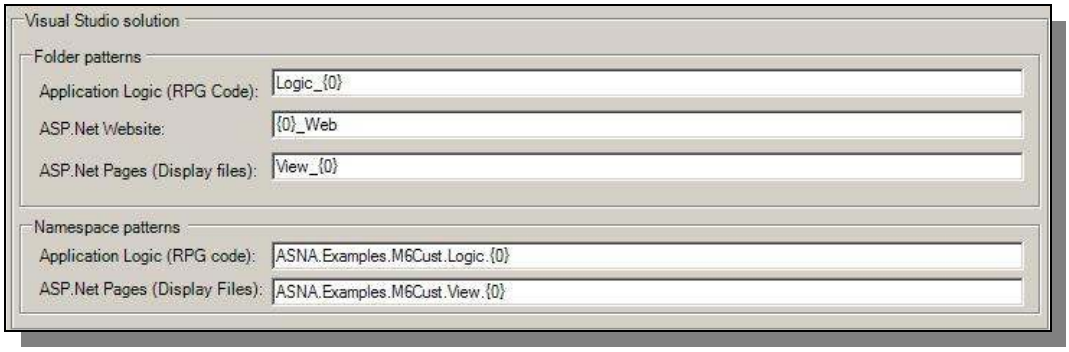
Make note of the Website path directive for the “Main” and “Common” GamePlans. Specifying C:\My_Application was deliberate. We want “Main” and “Common” to be *web modules* that are composite members of the My_Application Website. *Currently, Monarch 3.1 does not yet handle this nicely. So, this is what we will be doing manually from here.*



Migration directives for M6Cust_Main

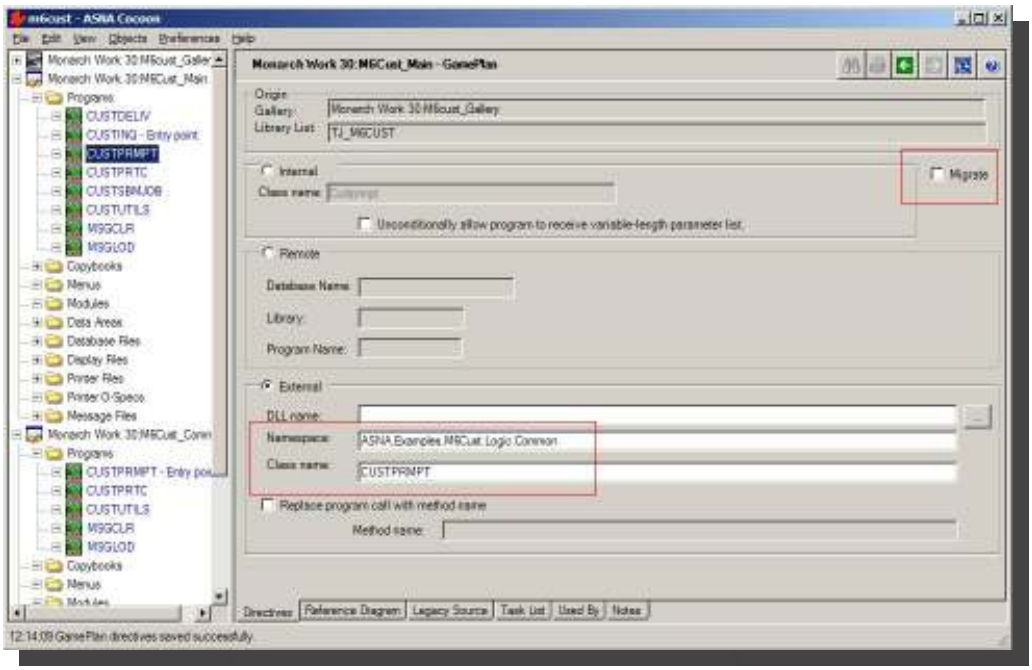


Migration directives for M6Cust_Common



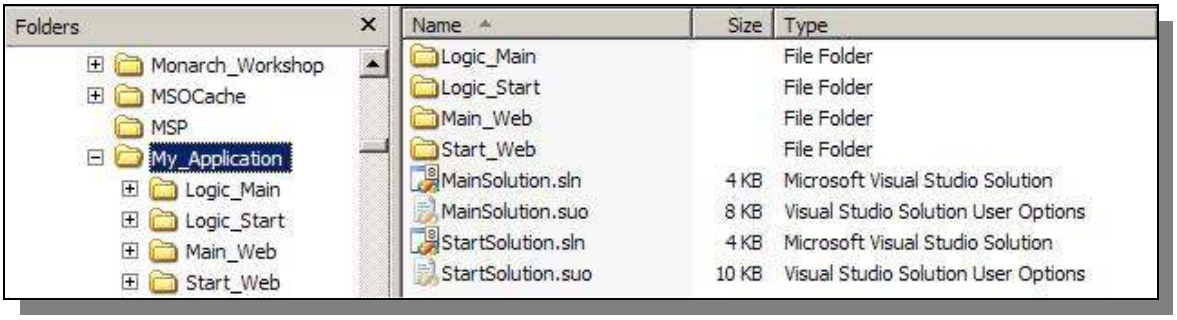
Visual Studio Migration Default Directives

Note that there are several programs present in both GamePlans; e.g., CUSTPRMPT, MSGCLR, etc. In actuality, these programs are truly being migrated in the "Common" GamePlan. However, they were included in the "Main" GamePlan when the GamePlan was created from its Exhibit cluster. When GamePlans are created from clusters, Monarch automatically sets the directives for external referencing as shown below.



Externally referenced classes (not migrated in "Main")

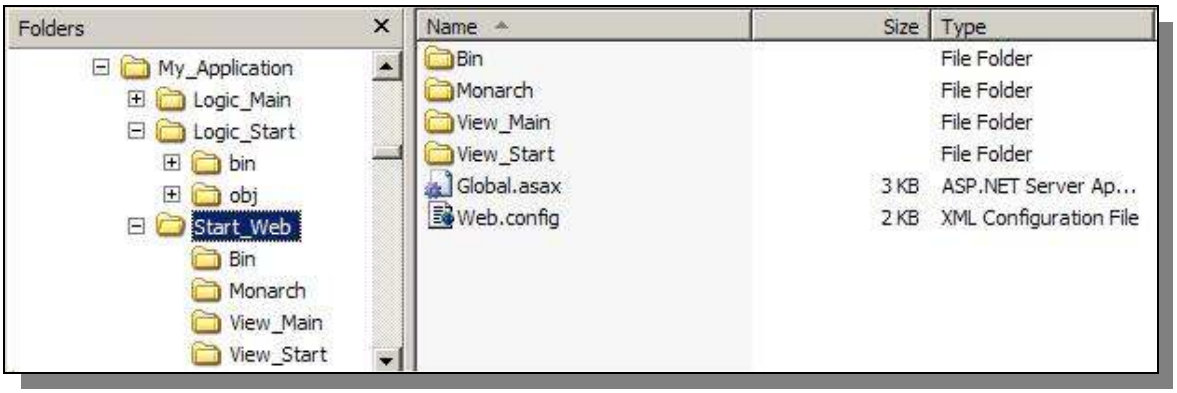
Now we are ready to migrate the "Main" and "Common" GamePlans. Remember, the Website path is going to be C:\My_Application for both GamePlans. Let's begin with the migration of "Main". Then, we will repeat the steps to migrate "Common". Be certain Visual Studio is closed after migrating.



C:\My_Application following "Main" Website Migration

When you view C:\My_Application from Windows Explorer, you can see that Start_Web and Main_Web are at the same hierarchy. When you drill down into C:\My_Application\Start_Web you see the View_Start folder containing StartHere and Logon .aspx files. Similarly, when you drill down into C:\My_Application\Main_Web you see the View_Main folder containing the workstation .aspx files for that GamePlan.

1. What we want to do now is to move the View_Main folder into the Start_Web folder so that both *view* folders are within the Start website (C:\My_Application\Start_Web). At this time, you can also delete the Main_Web folder as it is no longer needed[2]. Below is how it would appear in Windows Explorer.



Start_Web with View_Main and View_Start

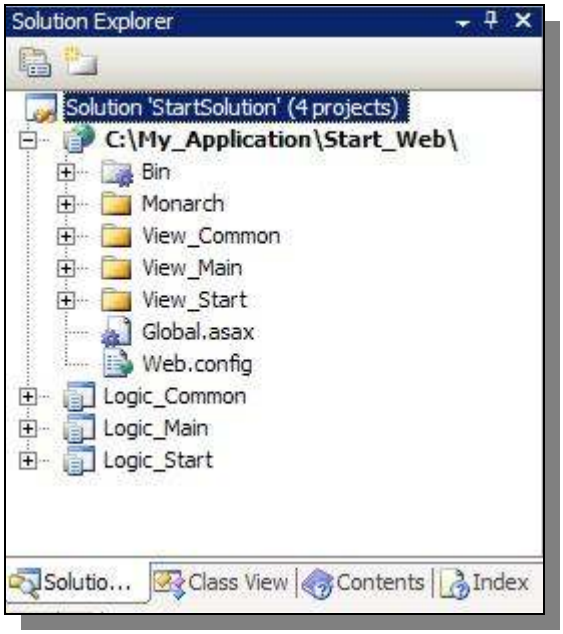
[2] The web site Monarch folder, Global.asax and Web.Config already reside in C:\My_Application\Start_Web.

The next step is to open the original StartSolution and incorporate the “Main” module into the solution. To do this, follow these steps:

2. Start Visual Studio.
3. Open **StartSolution.sln** from C:\My_Application. (notice that View_Main folder is now part of the solution).
4. In the Solution Explorer, right-mouse click on the solution and select **Add > Existing Project** and navigate to Logic_Main. Select the .vrproj file.
5. Right-mouse click on **Logic_Main** project and select **Properties**. Change the Display File Root setting to “C:\My_Application\Start_Web”. (*Remember, earlier we had moved the View_Main folder from Main_Web to Start_Web*).
6. Right-mouse click on **Logic_Start** project and select **Add Reference**. Select **Logic_Main** from the Projects tab. The Solution should look like this:

7. You can now build **Logic_Main**. Most likely, you will have compiler errors; if for no other reason than CALLs to “Common” classes cannot be found. This is due to the fact that we have neither migrated “Common”, nor added it to the solution.
8. One final, but optional step is to remove the unneeded code from Logic_Main\Myjob.vr. (However, it doesn’t hurt anything to leave it the way it is).

Now you can migrate the next GamePlan. In this case, it's "Common". After repeating the steps above used for "Main", the Solution should look like this:



Start_Web now contains both **View_Main** and **View_Common** folders. Start_Web is the repository for all the .aspx pages in the application. You do not need to copy them anywhere else. *This is where they will reside and where you will maintain them during development.*