## SharePoint Foundation Development

**Lab Time**: 45 minutes

**Lab Folder**: C:\Student\Labs\SharePointDevelopment

**Lab Overview**: In this lab you will begin to write and test C# code using the SharePoint Foundation 2010 object model. Note that we are purposefully delaying any work with the new SharePoint Developer Tools in Visual Studio 2010 Tools until the next lab because this new tool set is the focus of the next lecture.

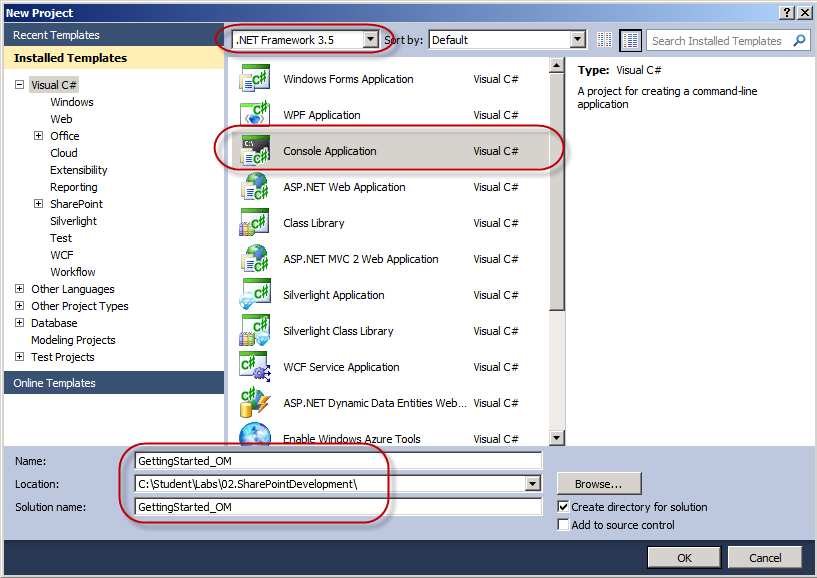
Lab Setup Requirements

* Before you begin this lab, you must run the batch file named **SetupLab.bat**. This batch file creates a new blank site collection at the location **http://intranet.wingtip.com/sites/DevSite1**. This is the site you will use to test and debug a Console application you are going to create with the Visual Studio 2010.

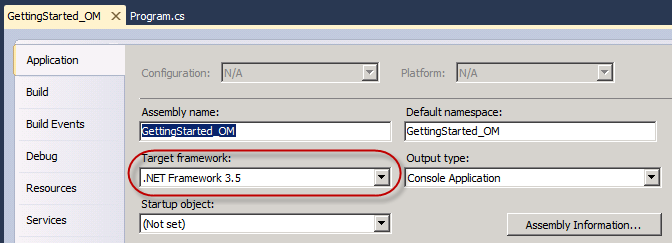
### Exercise 1: Programming with the SharePoint 2010 object model

In this exercise you will create a new console application that programs against the SharePoint Foundation 2010 object model to create new lists and add items to a SharePoint 2010 site. This will illustrate some on the aspects of the SharePoint Foundation 2010 object model as well as show some of the issues about which versions of assemblies to use when you do this type of programming.

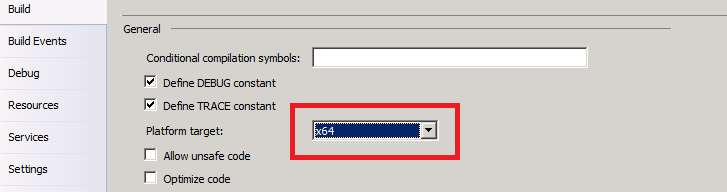
1. The script you just ran should have launched the Internet Explorer and navigated to the home page of the new site at http://intranet.wingtip.com/sites/DevSite1/default.aspx. You should see that this new site has been created as a blank site that has no existing lists. In the following steps you are going to write a program that uses the SharePoint 2010 object model to create a few new lists in this site and to populate these lists with sample data.
2. Launch Visual Studio 2010 from Windows Start menu. The shortcut you need can be found at Start » All Programs » Microsoft Visual Studio 2010 » Microsoft Visual Studio 2010.
3. Create a new Console application named **GettingStarted\_OM**. Create this new project at a location of [[LAB FILES]]\. Make sure to create the new project based on .NET Framework 3.5 instead of the default which will be .NET Framework 4.0. This is controlled by one of the drop downs found above the list of project templates as shown in the following figure:



1. Once the new project has been created, right-click on the GettingStarted\_OM project in the Solution Explorer and click on Properties to see the Project Properties view. Go to the Application tab of project properties and verify that .NET Framework Version 3.5 is selected as the Target Framework. It is important for you to acknowledge that SharePoint 2010 is based on .NET Framework version 3.5 and not version 4.0.



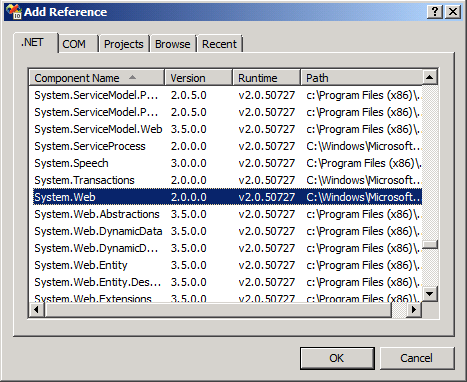
1. Click on the Build tab and verify that the target Platform is x64. If not, make sure to change the platform to x64 or Any CPU.



1. Now add a reference to the assembly Microsoft.SharePoint.dll which contains the core classes of the SharePoint Foundation object model. The \*.dll should be displayed in the .NET tab.   
   However, if you have trouble finding the assembly in the **.NET** tab, you might have to select the **Browse** tab and navigate to the following path to add the reference.

C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\ISAPI\Microsoft.SharePoint.dll

1. Next, add a reference to the main assembly for ASP.NET named System.Web as shown below.



1. Now it's time to write some code against the SharePoint object model to enable the Developer Dashboard. Modify Program.cs to look like this.

using System;

using Microsoft.SharePoint;

using Microsoft.SharePoint.Administration;

namespace GettingStarted\_OM {

class Program {

static void Main() {

SPWebService contentService = SPWebService.ContentService;

SPDeveloperDashboardSettings developerDashboard =

contentService.DeveloperDashboardSettings;

developerDashboard.DisplayLevel = SPDeveloperDashboardLevel.On;

developerDashboard.Update();

Console.WriteLine(“Developer Dashboard updated.”);

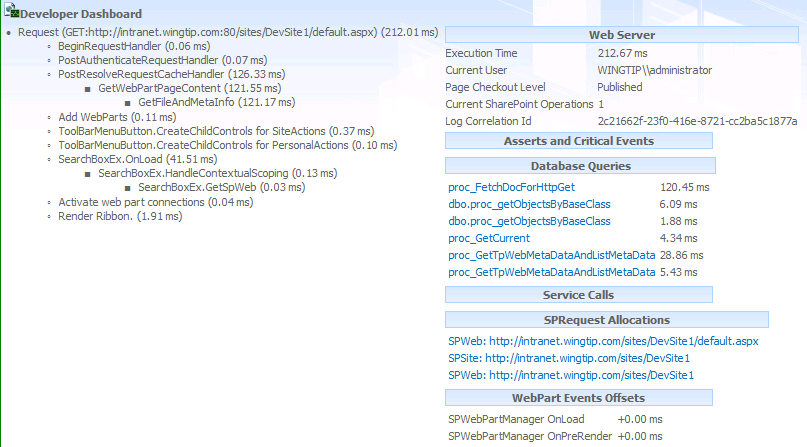
}

}

}

**Note**: If your **contentService** variable turns out to be null, you have to check your project properties. Good chance you haven’t selected the x64 platform.

1. Run the application by pressing [CTRL]+[F5]. Once the program runs, return to the Internet Explorer and site at http://intranet.wingtip.com/sites/DevSite1/default.aspx. Refresh the page and you should see the SharePoint 2010 Developer Dashboard at the bottom of the page.



1. Go back to Visual Studio and update the code in the static Main method to assign a value of SPDeveloperDashboardLevel.Off instead of SPDeveloperDashboardLevel.On and run the program again. Go back to the Internet Explorer and refresh the page again. You should see that the Developer Dashboard is now gone. Now you know how to make the Developer Dashboard appear and disappear just by running code against the SharePoint 2010 object model.
2. Now, it is time to write code that programs against the top-level site you created at the beginning of this exercise. Modify the Main method in program.cs to create a new SPSite object to program against the new site collection. Make sure to structure your code inside a using construct so that your code makes an implicit call to the Dispose() method to prevent memory leakage. As a first step, obtain a reference to the SPWeb object for the top-level site and print the Title property of the site to the console windows. Your code should look something like the code below. When you are done writing this code, run your program to make sure it runs without errors and displays the proper site title.

static void Main() {

string targetSiteUrl = "http://intranet.wingtip.com/sites/DevSite1";

using (SPSite siteCollection = new SPSite(targetSiteUrl)) {

SPWeb site = siteCollection.RootWeb;

Console.WriteLine(site.Title);

}

Console.WriteLine("Press ENTER to continue");

Console.ReadLine();

}

1. In this next step you will add an existing source file to your project that contains utility classes which program against the SharePoint 2010 object model and you will use these classes to create lists and add Web Parts. Right-click on the project in the Solution Explorer and select the **Add Existing Item** command. Add the existing source file inside the [[LAB FILES]]\StarterFiles folder named ListUtilties.cs.

**Note:** You may need to add an additional **using** statement reference to the namespace **ListUtilities** to get everything working.

1. The new source file named ListUtilties.cs contains three utility classes named TasksListFactory, AnnouncementsListFactory and WebPartPageDesigner. The first two classes contain code to create new list instances and to populate them with sample data. The third class named WebPartPageDesigner programs against a class named SPLimitedWebPartManager which is provided by the SharePoint 2010 object model to delete and add Web Part instance from a target Web Part Page. Take a moment to examine the three classes inside ListUtilties.cs in order to understand how it is leveraging the SharePoint 2010 object model.
2. Now it’s time to modify the Main method in Program.cs to use the three utility classes in ListUtilties.cs. Modify the code within the using construct in Main that you created in a step so that it looks like the code below.

using (SPSite siteCollection = new SPSite(targetSiteUrl)) {

SPWeb site = siteCollection.RootWeb;

Console.WriteLine(site.Title);

string TasksListTitle = "Tasks";

string TasksListDescription = "Wingtip Tasks";

TasksListFactory.Create(site, TasksListTitle, TasksListDescription);

string AnnouncementsListTitle = "Announcements";

string AnnouncementsListDescription = "Wingtip Announcements";

AnnouncementsListFactory.Create(site,

AnnouncementsListTitle,

AnnouncementsListDescription);

WebPartPageDesigner.ClearAll(site, "default.aspx");

WebPartPageDesigner.AddXsltListViewWebPart(site, "default.aspx",

"Tasks", "Left");

WebPartPageDesigner.AddXsltListViewWebPart(site, "default.aspx",

"Announcements", "Left");

}

1. Once you have modified the Main method with the code shown above, run the program by pressing **[CTRL]+[F5]**. When the program runs, it should add two new lists to the target Blank site and add two new Web Parts to the home page (**http://intranet.wingtip.com/sites/DevSite1/default.aspx**) to display the contents of these two lists as shown in the following figure:



In this exercise you learned how to enable the Developer Dashboard and how to work with the SharePoint object model to modify an existing SharePoint site.