Prism for Windows Runtime CodeGen

Monday, September 9, 2013 4:08 PM

Summary:
'PrismRT CodeGen' is a code generation system for setting up Line of Business Windows Store applications that use the 'Prism for Windows Runtime' guidance from the Microsoft Patterns and Practices group. It helps to reduce the repetitious procedures for starting such Windows Store applications.

Motivation:
I wanted to write Windows Store LOB apps that use Windows Azure hosted WebAPI services and SQL Azure hosted databases. I wanted my design to use best practices and loosely coupled components. I found that the 'Prism for Windows Runtime' gives my apps a very good MVVM structure and that the Prism base classes included a lot of LOB functionality for free. I wanted to write a lot of smaller test apps that target specific areas e.g. DataValidation but I wanted my test apps to have the same structure as my production apps. It takes time to set up all the projects and include all the classes I need and is error prone. So I'm using Visual Studio Project Templates, T4 Code Templates, Visual Studio Automation and SQL Server Management Objects to solve the problem.

PrismRT CodeGen objective:
Create a simple 'Prism for Windows Runtime' Starter app that is driven by SQL Server Domain database tables. The app:

1. Contains a List Page for each table and presents each table's rows in a ListView
2. Contains a Details Page that presents a single table row when selected from the ListView
3. Supports CRUD
4. Designed with MVVM pattern
    a. UILayer
    b. UILogic
    c. WebAPI
    d. DataRepository
    e. Unity Dependency Injection
    f. EF4

Is PrismRT CodeGen for you?

1) Do you write Windows Store Apps using C# and XAML? If yes, continue.
2) Are your Windows Store Apps mainly Line of Business CRUD type apps? If yes, continue.
3) Do you use the 'Prism for Windows Runtime' guidance? If yes, continue.
4) Do you use the Unity Dependency Injection Container? If yes, continue.
5) Do your Windows Store Apps follow a similar pattern e.g. CRUD apps that get data from a REST service and present that data in Windows Runtime pages? If yes, continue.
6) Do you find that most of your .cs files are similar from app to app but just that the data fields and namespaces are different? If yes, continue.
7) Do your Windows Store Apps use the MVVM Pattern? If yes, continue.
8) Do you use RESTful services to access backend data? If yes, continue.
9) Do you use SQL Server 2012 as your database? If yes, continue.
10) Do you use Entity Framework 4 as your ORM? If yes, continue.
11) Do you use a Repository Pattern in your Data Access Layer? If yes, continue.
12) You are comfortable with the fact that PrismRT CodeGen does not support roundtripping. i.e. Running PrismRT CodeGen a second time will overlay any changes made to the Target application after the 1st time it is run. If yes, then continue.

If PrismRT CodeGen is not for you, there are some things it demonstrates that may have some value to you:

1) Example of Visual Studio 2012 Project Templates
2) Example of Visual Studio Add-in
3) Example of Visual Studio 2012 Automation including ExecuteCommand event handler
4) Example of a SQL Server Database driven app using SSMO (SQL Server Management Objects)
5) Example of EF4 CRUD
6) Example of WebAPI accessing SQL Server 2012 data
7) Example of T4 Templates for code generation
8) Example of a simple 'Prism for Windows Runtime' app

PrismRT CodeGen components:

1) PrismStarterTemplates solution - VS Project Templates that generate a 'Prism for Windows Runtime' shell app. This solution contains all the projects and folders for a MVVM based Line of Business application but that does not contain any Domain specific data objects

or code. Each project in this solution is Exported to a Visual Studio project template (zip file) that is stored at C:\Users\...\Documents \Visual Studio 2012\My Exported Templates.

2) PrismStarter Visual Studio 2013 Add-in - uses VS Automation to create a Windows Store app shell from the PrismStarterTemplates solution's projects.
3) PrismTableTemplates solution - VS Project Templates that contain T4 code templates that generate code for the MVVM layers specific to one table in a SQL Server 2012 Database.
4) PrismTable Visual Studio 2013 Add-in - uses VS Automation and SQL Server Management Objects (SMO) to generate code based on one table and then plug that code into the PrismStarter shell app.
5) PrismCodeGen metadata - 2 SQL Server 2012 database tables:
    i. CodeGenRules - model class Validation Attributes
    ii. CodeGenStrings - used to create .resx and .resw files
6) Excel metadata spreadsheet - makes working with PrismCodeGen metadata easier.

How PrismRT CodeGen works

Thursday, September 26, 2013 3:38 AM

Let's say that I work in Utah for the NSA and I'm assigned the task to write an app that will keep track of all the information on a person. This information includes personal, work, financial, medical, email, phone etc. I'm going to need a database with many tables and I'm going to need an app that has many forms to view and edit the information. Each type of information will have many pieces of data. For example a phone call has an Id, SSN of the person who owns the call, FromPhoneNumber, ToPhoneNumber, DateAndTime, Lat/Long (if mobile then a series if moving), LengthOfCall, Audio, Text etc. Some types of information could involve hundreds of data items. If my app is design for MVVM then I'm going to be moving the data in code at least from the data layer to the business layer to the service layer to the presentation layer. I'll be basically doing the same thing for each table of data and I could have hundreds of tables.

This is where Code Generation comes in handy. With Code Generation I have a T4 (Text Template) file with placeholders for the data items I working with. For example a PageView form will have xaml for each data item - a title label, a textbox, an error message label. This will be repeated for each data item in a form which could have 10 to 100 data items. This repetition will occur in each layer of code as data is transferred from one layer to another. The combination of T4 Templates and SQL Server Management Objects (SMO) helps will this boring and error prone coding. With SMO you can retrieve the metadata for each table i.e. the data fields's name, type, keys etc. (See the Skill Links page for more info) and then within the T4 Template logic can loop through the metadata to generate a line of code for each data field. Besides T4 Templates PrismRTCodeGen uses FindAndReplace to modify generic-to-domain specific code. If you had 20 code files with a generic namespace of say 'PrismStarter' and your app's namespace should be 'BigBrother', you could read the whole file in, use C# string functions to find and replace 'PrismStarter' with 'BigBrother' and write the whole file out. These 2 methods of code generation (T4 and FindAndReplace) are used by PrismRT CodeGen.

To simplify the CodeGen process, let's say that my app will be called BigBrother and that my database contains just one table (PhoneCall) and that the table contains only 2 fields (Id and SSN). I want to code generate an app that will present a ListView of phone calls on the main page and when I select a phone call from the ListView, the user is taken to a Details page with label and text controls for each phone call data field and options to create a new call, update an existing call or delete a phone call (CRUD). Since this app follows a pattern (a list page and a corresponding detail page) I can create a set of templates that will do the same thing for each table in the database.

In the downloads I've included the 2 Templates solutions that contains the .tt files (PrismStarterTemplates and PrismTableTemplates). PrismStarterTemplates contains the T4 templates that generate Windows Store app shell. PrismTableTemplates contains the T4 templates that generate code specific to one table in the Domain database. After I wrote PrismStarterTemplates and PrismTableTemplates I Exported each project within each of the solutions to Visual Studio Project Template zip files that are included in the downloads. Take these zip files and put them in the place Visual Studio looks for them (see What you have to do). You do not need to touch the zip files after they are placed.

In the downloads I've included a VS Addin project to match each of the Templates solutions. PrismStarter addin matches and uses PrismStarterTemplates to create a shell Windows Store app. PrismTable addin matches and uses PrismTableTemplates to create code based on one table in the Domain database and then plugs that code into the shell app.

Note that PrismStarterTemplates and PrismTableTemplates are generic and reusable. The 2 Addins will

generate Domain specific code based on the AppSettings section of their App.config files. That is where you specify the names for the shell app and which tables in the Domain database that will be used to generate the Domain specific code. Examples are included in the download.

After PrismAddin finishes all its steps you need to do some domain specific things. (see What you have to do).

Now we are ready to test the WebAPI to see if we are getting SQL Server data. Note that PrismAddin will generate a WebAPI project that uses the Visual Studio Development Server at port 56789. If not getting data, use debugging (see Debugging using 2 VS Instances and Debug WebAPI with Fidder2).

After the WebAPI test is successful, run the Target app to see if CRUD is working. Make sure that an instance of Visual Studio Development Server is running.

That's how PrismRT CodeGen works in some detail.

Skill Links

Thursday, September 26, 2013 4:10 AM

Links:

How to: Create a Project Template
http://msdn.microsoft.com/en-us/library/xkh1wxd8.aspx
Project Template locations:
C:\Users\Ron\Documents\Visual Studio 2013\My Exported Templates
C:\Users\Ron\Documents\Visual Studio 2013\Templates\ProjectTemplates
C:\Users\Ron\AppData\Roaming\Microsoft\VisualStudio\12.0\ProjectTemplatesCache

How to: Create a Multi-Project Template
Note: Multi-Project Template (PrismStarter.zip) did not work out. When creating a New Project with it, the ProjectName e.g. BigBrother did not flow down to the child projects. Just including it here for reference.
http://msdn.microsoft.com/en-us/vstudio/cc315061.aspx (video)
http://msdn.microsoft.com/en-us/library/ms185308.aspx

How to: Write a T4 Text Template
http://msdn.microsoft.com/en-us/library/bb126478.aspx
http://www.olegsych.com/2007/12/how-to-create-a-simple-t4-template/
http://visualstudiogallery.msdn.microsoft.com/a42a8538-8d6e-491b-8097-5a8a00174d37
http://www.devart.com/t4-editor/download.html (DevArt T4 Editor - Free)

How to: Work with Microsoft.SqlServer.Management.Smo
http://www.codeproject.com/Articles/127065/SMO-Tutorial-1-of-n-Programming-data-storage-objec
http://ittecture.wordpress.com/2009/01/04/tip-of-the-day-82/

How to: Automate Visual Studio 2012
http://msdn.microsoft.com/en-us/library/vstudio/1xt0ezx9.aspx
Professional Visual Studio 2012 -> Part XI -> Chapter 51: The Automation Model -> TheVisual Studio Automation Model
(see this book to understand DTE, DTE2, EnvDTE80, EnvDTE90, EnvDTE100)

How to: Create a WebAPI Service
http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api
http://www.asp.net/web-api/overview/creating-web-apis/creating-a-web-api-that-supports-crud-operations

How to: Deploy WebAPI to Azure Cloud
http://www.windowsazure.com/en-us/develop/net/tutorials/get-started/
(web page has link to download Windows Azure SDK for Visual Studio 2013)

How to: Deploy SQL Server 2012 Database to Azure Cloud
http://channel9.msdn.com/posts/SQL11UPD00-REC-02 (video)

Simple way to import data into SQL Server
http://www.mssqltips.com/sqlservertutorial/203/simple-way-to-import-data-into-sql-server/

Run .dtsx from command prompt (on 64-bit machine needs to run 32-bit dtexec)
http://stackoverflow.com/questions/8787007/how-to-execute-dtsx-packages-through-command-line
"C:\Program Files (x86)\Microsoft SQL Server\110\DTS\Binn\DTExec.exe" /File "C:\Dev\Metro.2013.2\CodeGen\Docs\DataTypes2.dtsx"

How to: Create an Add-in
http://msdn.microsoft.com/en-us/library/ms228767.aspx
http://msdn.microsoft.com/en-us/library/80493a3w.aspx

How to: Remove Add-in
http://msdn.microsoft.com/en-us/library/vstudio/ms228765.aspx
devenv /resetaddin PrismAddin.Connect

How to: Access App.config from Add-in
http://stackoverflow.com/questions/3925308/is-there-a-config-type-file-for-visual-studio-add-in

What has been provided

Wednesday, September 25, 2013 5:04 PM

1) **Docs->PrismRTCodeGen_v2.pdf -** documentation
2) **PrismStarterTemplates and PrismTableTemplates** - Visual Studio 2013 Solutions that were used to create VS Project Template zip files.
    a. File->Export Template… for each project in each solution
3) **My Exported Templates** - my VS Project Template zip files
4) **PrismStarter and PrismTable** - Visual Studio 2013 Addin solutions that use Visual Studio Automation (EnvDTE) and SQL ServerManagement Objects (SMO) to generate code.
    a. VS->New Project->Other Project Types->Extensibility->Visual Studio Add-in (see Skill Links)
5) **Data->BigBrother.bak** - A sample SQL Server database called BigBrother that contains the Domain tables. The generated code is based on these tables:
    a. PhoneCall
    b. CodeGenRules
    c. CodeGenStrings
6) **Data->BigBrother.CodeGenTables_2003.xlsx** - A sample Excel spreadsheet that contains the data for the CodeGen tables. Excel's Find and Replace makes it easy to create CodeGen data for a new project. Also included are .dtsx files and .cmd to Import excel data into SQL Server (see Skill links)
7) **BigBrother Target Solution** which is the result of running PrismStarter(addin) and PrismTable(addin). Not including PrismForWindowsRuntime dlls, EntityFramework dlls, Unity Container dlls.
    a. BigBrother **-** Target solution (Windows Store app that runs on user device)
        i. BigBrother.DalInterface project
        ii. BigBrother.DalEF4 project
        iii. BigBrother.DalMemory project
        iv. BigBrother.WebAPI project
        v. BigBrother.UILogic project
        vi. BigBrother.UILayer project
8) **Data->BigBrother.bak** - sample SQL Server 2012 Database

What you have to do

Wednesday, September 25, 2013 5:05 PM

1) Copy PrismStarterTemplate.zip files and PrismTableTemplate.zip files to Visual Studio Project Template locations
   C:\Users\<your name>\Documents\Visual Studio 2012\My Exported Templates
   C:\Users\<your name>\Documents\Visual Studio 2012\Templates\ProjectTemplates

2) SQL Server 2012 CodeGenTables configuration
   Create a SQL Server 2012 database that contains tables similar to the ones provided in the sample database:
   PhoneCall
   CodeGenRules
   CodeGenStrings
   Cut and paste blocks of spreadsheet cells into VS Server Explorer -> Data Connections -> Tables.

   Spreadsheet to Database steps

Run Reset Page T-Sql commands to clean out SQL Server 2012 BigBrother database	
Copy Rules Excel page rows to CodeGenRules DB Table	
Copy Strings Excel page rows to CodeGenStrings DB Table	

3) Addin Configuration. Change the <AppSettings> section in the App.config files of PrismStarter(addin) and PrismTable(addin) solutions with Domain specific values.

4) Install the addins by building them in Visual Studion 2013. (see Skill Links - How to: Create an Add-in). To run an Add-in, start a new Visual Studio. Go to Tools->Add-in Manager. Check the PrismAddin checkbox and press OK. Add-in runs and displays a MessageBox when done.
   Note: <appSettings> can be dynamically changed without having to rebuild the addin. Change them in C:\...\...\...\PrismAddin \PrismAddin\bin\PrismAddin.dll.config then save them. Start new Visual Studio and run addin.

5) Target solution fixups after PrismStarter addin is run:
   BigBrother.DalEF4
   Add ADO.NET Entity Data Model - BigBrother.edmx
   BigBrother.WebAPI -> Web.config
   Copy BigBrother.DalEF4->App.config->BigBrotherEntities connectionString to Web.API->Web.config
   Set BigBrother.UILayer as StartUp Project

   Target solution fixups after each PrismTable addin is run
   App.xaml.cs
   Using BigBrother.UILogic.Repositories

   App.xaml.cs->OnLaunchApplication
   NavigationService.Navigate("PersonList", null);

   App.xaml.cs->OnInitialize

```
// Register repositories
_container.RegisterType<IPersonRepository, PersonRepository>(new ContainerControlledLifetimeManager());
_container.RegisterType<IStateRepository, StateRepository>(new ContainerControlledLifetimeManager());

// Register web service proxies
_container.RegisterType<IPersonServiceProxy, PersonServiceProxy>(new ContainerControlledLifetimeManager());
_container.RegisterType<IStateServiceProxy, StateServiceProxy>(new ContainerControlledLifetimeManager());
```

   App.xaml.cs->OnRegisterKnownTypesForSerialization

```
SessionStateService.RegisterKnownType(typeof(Person));
SessionStateService.RegisterKnownType(typeof(State));
```

DalEF4
Update Model from Database

6) For each provided .sln check for missing references (to save space on github I removed packages like Prism for Windows Runtime, Entity Framework, Unity3 etc). Right click each project -> Manage NuGet packages… to restore them. Use the names from the Project -> References folder.

7) Visual Studio 2013 Settings changes
Visual Studio 2013 Tools -> Options -> Projects and Solutions -> Web Projects -> Use IIS Express for new file-based web sites and projects (uncheck). We are going to test with Visual Studio Development Server.

8) Rebuild Solution
NuGet Package Manager should find missing references and load the missing packages

9) Start Debug instance of WebAPI
Right-click BigBrother.WebAPI Project -> Debug -> Start new instance
Run http://localhost:51379/api/PhoneCall to check that WebAPI is accessing Database
Stop Debugging

10) Start BigBrother.sln

Debug WebAPI with Fiddler2

Friday, September 20, 2013 7:28 AM

Http Verbs:

GET	
POST	Add
PUT	Update
Delete	

Composer Urls:

http://localhost:50420/api/<Entity>	Visual Studio Development Server
http://dell15z:8082/api/<Entity>	IIS Server
http://bigbrotherwebservice.azurewebsites.net/api/<Entity>	Windows Azure

======== BigBrother ==========

GET http://localhost:50420/api/PhoneCall
GET http://http://bigbrotherwebservice.azurewebsites.net/api/PhoneCall
GET http://dell15z:8082/api/PhoneCall
GET http://dell15z:8082/api/PhoneCall/1
GET http://dell15z:8082/api/PhoneCall?SSN=987654321

POST http://dell15z:8082/api/PhoneCall
POST http://localhost:59444/api/PhoneCall
Composer->RequestHeaders->
Accept: application/json
Content-Type: application/json
User-Agent: Fiddler
Host: dell15z:8082
Host: localhost:59444
Composer->RequestBody
{"Id":"0","SSN":"333333333"}
[{"Id":0,"FldBit":true,"FldDateTime":"1913-09-30T00:00:00","FldInt":93,"FldMoney":93.93,"FldNVarChar":"Clara Lemire"}]
[{"Id":1,"FldBit":true,"FldDateTime":"1945-11-24T00:00:00","FldInt":67,"FldMoney":67.6700,"FldNVarChar":"ron was here"}]

PUT http://dell15z:8082/api/PhoneCall/3
PUT http://localhost:59444/api/PhoneCall/3
Composer->RequestHeaders->
Accept: application/json
Content-Type: application/json
User-Agent: Fiddler
Host: dell15z:8082
Host: localhost:59444
Content-Length: 28
Composer->RequestBody
{"Id":"3","SSN":"999999999"}

DELETE http://dell15z:8082/api/PhoneCall/3
DELETE http://localhost:59444//api/PhoneCall/3
Composer->RequestHeaders->
Accept: application/json
User-Agent: Fiddler
Host: dell15z:8082
Host:localhost:59444

```
======== AdventureWorks ==========================================================================================

            // GET /api/Product
            http://http://dell15z:8084/api/Product/
            http://localhost:2112/api/Product/
            Composer->RequestHeaders-> (Accept header is the format of return data)
                  Accept: application/json
                  Accept: application/xml

    public IEnumerable<Product> GetProducts()
                  return _products.ToArray();


            // GET /api/Product/id
            http://http://dell15z:8084/api/Product/BK-R19B-58
            http://localhost:2112/api/Product/BK-R19B-58
    public Product GetProduct(string id)
                  var item = _productRepository.GetProduct(id);
                public Product GetProduct(string productNumber)


            // GET /api/Product?queryString={queryString}
            http://localhost:2112/api/Product?queryString=Mountain-500%20Red,%2042
    public SearchResult GetSearchResults(string queryString)
                  var fullsearchResult = _productRepository.GetProducts().Where(p =>
                  p.Title.ToUpperInvariant().Contains(queryString.ToUpperInvariant()));


    // GET /api/Product?categoryId={categoryId}
            http://localhost:2112/api/Product?categoryId=2
    public IEnumerable<Product> GetProducts(int categoryId)
              if (categoryId == 0)
     {
        return _productRepository.GetTodaysDealsProducts();
     }
     return _productRepository.GetProductsForCategory(categoryId);



========= PluralSight ============================================================================================

public class ValuesController : ApiController
{
      static List<string> data = initList();

      private static List<string> initList()
      {
            var ret new List<string>();
            ret.Add("value1");
            ret.Add("value2");
            return ret;
      }

      // GET api/values
      public IEnumerable<string> Get()
      {
            return data;
      }

      // GET api/values/5
      public string Get(int id)
      {
```

```
        return data[id];
}

// POST api/values
public void Post([FromBody]string value)
{
        data.Add(value);
}

// PUT api/values/5
public void Put(int id, [FromBody] string value)
{
        data[id] = value;
}

// DELETE api/values/5
public void Delete(int id)
{
        data.RemoveAt(id);
}
```

Debug using 2 VS Instances

Sunday, September 15, 2013 11:15 AM

How to debug BigBrother.WebAPI using 2 instances of Visual Studio 2012:

1) Open BigBrother Solution as 1st instance

2) Set Startup Project->BigBrother.UILayer

3) In C:\Dev\Apps\BigBrother\BigBrother.UILogic\Constants.cs set ServerAddress to Visual Studio Development Server Url:

```
public const string ServerAddress = "http://localhost:50909";
```

4) In BigBrother.WebAPI->Properties->Web->Servers:

Select-> Use Visual Studio Development Server radio button

Select-> Specific port 50909

5) Delete all breakpoints

6) Set a breakpoint in UILogic->PhoneCallServiceProxy.cs->GetPhoneCallsAsync()

7) Start 1st instance using Local Machine Debugging.
Debugger should stop at breakpoint.
There should be an ASP.NET Development Server - Port 50909 started in Windows tray.

8) Open same BigBrother Solution for the 2nd instance.
(There is no need to right click BigBrother.WebServices node->Debug->Start new instance)

9) Keep Startup Project, ServerAddress, WebService port same as 1st instance

10) Delete all breakpoints in 2nd instance

11) Set a breakpoint in C:\Dev\Apps\BigBrother\BigBrother.WebServices\Controllers\PhoneCallController.cs->Get() in 2nd instance

12) Go to MainMenu->Debug->Attach to Process->WebDev.WebServer40.EXE - ASP.NET Development Server - Port 50909->press Attach button.

13) At this point there should be only 1 ASP.NET Development Server - Port 50909 in Windows tray
Debugging symbols for 2nd instance may or may not be loaded at this point.
If not they will be loaded by doing step 14).

14) In 1st instance press Continue debugging button

PrismTemplates Project Templates

Wednesday, September 11, 2013 4:13 PM

Project Name	Project Type	ProjectTemplate
PrismStarter.DalInterface	C#->Windows->Class Library	PrismStarterDalInterface.zip
PrismStarter.DalEF4	C#->Windows->Class Library	PrismStarterDalEF4.zip
PrismStarter.DalMemory	C#->Windows->Class Library	PrismStarterDalMemory.zip
PrismStarter.WebAPI	C#->Web->ASP.NET MVC Web Application->WebAPI	PrismStarterWebAPI.zip
PrismStarter.UILogic	C#->Windows Store->Class Library (Windows Store apps)	PrismStarterUILogic.zip
PrismStarter.UILayer	C#->Windows Store->Blank App (XAML)	PrismStarterUILayer.zip
CommonStarterTemplate	C#->Windows->Class Library	CommonStarterTemplate.zip

Project Name	Project Type	ProjectTemplate
PrismTable.DalInterface	C#->Windows->Class Library	PrismTableDalInterface.zip
PrismTable.DalEF4	C#->Windows->Class Library	PrismTableDalEF4.zip
PrismTable.DalMemory	C#->Windows->Class Library	PrismTableDalMemory.zip
PrismTable.WebAPI	C#->Web->ASP.NET MVC Web Application->WebAPI	PrismTableWebAPI.zip
PrismTable.UILogic	C#->Windows Store->Class Library (Windows Store apps)	PrismTableUILogic.zip
PrismTable.UILayer	C#->Windows Store->Blank App (XAML)	PrismTableUILayer.zip
CommonTableTemplate	C#->Windows->Class Library	CommonTableTemplate.zip

How to: Create Project Templates
http://msdn.microsoft.com/en-us/library/xkh1wxd8.aspx

Project Templates located at:
C:\Users\Ron\Documents\Visual Studio 2012\My Exported Templates
C:\Users\Ron\Documents\Visual Studio 2012\Templates\ProjectTemplates

List of Renamed PrismTable Template Files

Wednesday, September 11, 2013 4:11 PM

BigBrother.DalInterface
- Models
  - Rename Entity.tt -> PhoneCall.tt
- Repositories
  - Rename IEntityRepository.tt -> IPhoneCallRepository.tt

BigBrother.DalEF4
- Repositories
  - Rename EntityMapper.tt -> PhoneCallMapper.tt
  - Rename EntityRepository.tt -> PhoneCallRepository.tt

BigBrother.DalMemory
- Repositories
  - Rename EntityRepository.tt -> PhoneCallRepository.tt

BigBrother.WebAPI
- Controllers
  - Rename EntityController.tt -> PhoneCallController.tt

BigBrother.UILogic
- Events
  - Rename EntityDeletedEvent.tt -> PhoneCallDeletedEvent.tt
- Models
  - Rename Entity.tt -> PhoneCall.tt
- Repositories
  - Rename EntityRepository.tt -> PhoneCallRepository.tt
  - Rename IEntityRepository.tt -> IPhoneCallRepository.tt
- Services
  - Rename IEntityServiceProxy.tt -> IPhoneCallServiceProxy.tt
  - Rename EntityServiceProxy.tt -> PhoneCallServiceProxy.tt
- ViewModels
  - Rename EntityDetailPageViewModel.tt -> PhoneCallDetailPageViewModel.tt
  - Rename EntityListPageViewModel.tt -> PhoneCallListPageViewModel.tt)

BigBrother.UILayer
- Views
  - Rename EntityListPage.tt -> PhoneCallListPage.tt
  - Rename EntityLIstPage.xaml.tt -> PhoneCallListPage.xaml.tt
  - Rename EntityDetailPage.tt -> PhoneCallDetailPage.tt
  - Rename EntityDetailPage.xaml.tt -> PhoneCallDetailPage.xaml.tt

List of Find and Replace Words

Wednesday, September 25, 2013 7:36 PM

Do FindAndReplace

From	To
PrismTemplates	BigBrother
"Entity"	"PhoneCall"
"EntityId"	"PhoneCallId"
"Entities"	"PhoneCalls"
"entity"	"phoneCall"
"entityId"	"phoneCallId"
"entities"	"phoneCalls"
"EntityRules"	"PhoneCallRules"
"EntityServerSideRules"	"PhoneCallServerSideRules"
"EntityList"	"PhoneCallList"
"EntityListPage"	"PhoneCallListPage"
"EntityListPageViewModel"	"PhoneCallPageViewModel"
"EntityDetail"	"PhoneCallDetail"
"EntityDetailPage"	"PhoneCallDetailPage"
"EntityDetailPageViewModel"	"PhoneCallDetailPageViewModel"

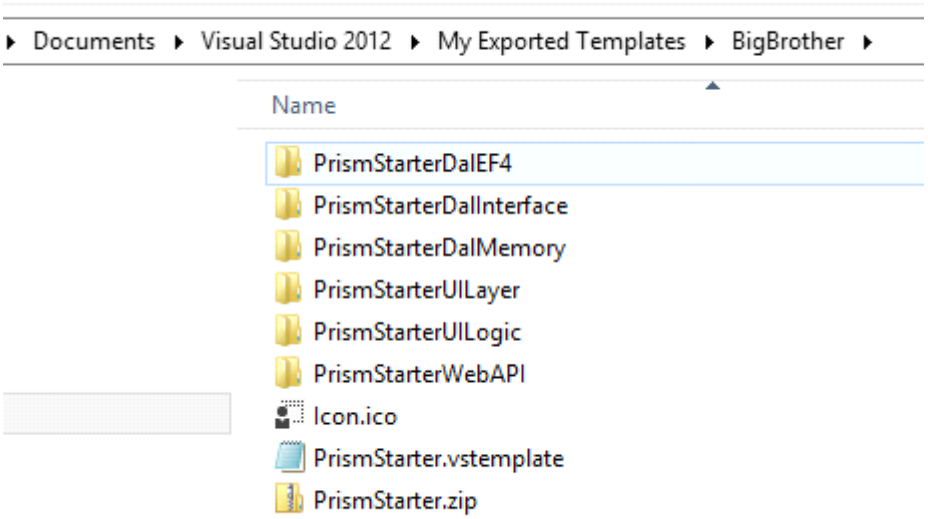
Do ReplaceAll for whole Solution to change 'PrismStarter' occurrences

From	To
PrismStarter.UILogic.Services	BigBrother.UILogic.Services
PrismStarter.UILogic.Models	BigBrother.UILogic.Models

Multi-Project Template

Saturday, September 14, 2013 7:04 AM

Multi-Project Template (PrismStarter.zip) did not work out. When creating a New Project with it, the ProjectName e.g. BigBrother did not flow down to the child projects.




```
<VSTemplate Version="2.0.0" Type="ProjectGroup"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>PrismStarter</Name>
    <Description>An example of a multi-project template</Description>
    <Icon>Icon.ico</Icon>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <ProjectCollection>
      <ProjectTemplateLink ProjectName="PrismStarterDalEF4">
        PrismStarterDalEF4\MyTemplate.vstemplate
      </ProjectTemplateLink>
      <ProjectTemplateLink ProjectName="PrismStarterDalInterface">
        PrismStarterDalInterface\MyTemplate.vstemplate
      </ProjectTemplateLink>
      <ProjectTemplateLink ProjectName="PrismStarterDalMemory">
        PrismStarterDalMemory\MyTemplate.vstemplate
      </ProjectTemplateLink>
      <ProjectTemplateLink ProjectName="PrismStarterUILayer">
        PrismStarterUILayer\MyTemplate.vstemplate
      </ProjectTemplateLink>
      <ProjectTemplateLink ProjectName="PrismStarterUILogic">
        PrismStarterUILogic\MyTemplate.vstemplate
      </ProjectTemplateLink>
      <ProjectTemplateLink ProjectName="PrismStarterWebAPI">
        PrismStarterWebAPI\MyTemplate.vstemplate
```

```
        </ProjectTemplateLink>
      </ProjectCollection>
    </TemplateContent>
</VSTemplate>
```

T4 Editors

Saturday, September 14, 2013 5:51 PM

Visual Studio 2013

I like DevArt T4 Editor but it did not work with Visual Studio 2013.
Instead using free version of 'tangible T4 Editor'
http://t4-editor.tangible-engineering.com/Download_T4Editor_Plus_ModelingTools.html

Visual Studio 2012

http://www.devart.com/t4-editor/download.html (DevArt T4 Editor - Free)
VS2012->Tools->Options...->Environment->Fonts and Colors->Display Items:

T4(Devart) - Attribute	Magenta
T4(Devart) - C# Keyword	Lime
T4(Devart) - C# String	Cyan
T4(Devart) - String	White

Add-in Notes

Thursday, October 17, 2013 3:51 PM

Addins stored in:
C:\Users\Ron\Documents\Visual Studio 2012\Addins

To remove an add-in from the integrated development environment (IDE)

1. Delete the .addin XML registration file for the add-in that you want to remove.
   The default location is ..\Users*username*\My Documents\Visual Studio 2012\Addins\.
2. At a Visual Studio command prompt, type **devenv /resetaddin** *Namespace*.*ClassName*, where *Namespace* is the name of your add-in project and *Classname* is its class name, for example, **devenv /resetaddin MyAddin1.Connect**.

From <http://msdn.microsoft.com/en-us/library/ms228765(v=vs.110).aspx>

Note: Use VS2012 x86 Native Tools Command Prompt to remove add-in
devenv/resetaddin PrismAddin.Connect

To change PrismDTE configuration use:

C:\Dev\Metro.2013.2\CodeGen\PrismDTE\PrismDTE\bin\PrismDTE.dll.config

Do not use:

C:\Dev\Metro.2013.2\CodeGen\PrismDTE\PrismDTE\bin\Debug\PrismDTE.dll.config
C:\Dev\Metro.2013.2\CodeGen\PrismDTE\PrismDTE\App.config

To disable in 'Project properties' go to Project properties -> Build > "Errors and warnings" (section), Suppress Warnings (textbox), add 1591 (comma separated list) – Nick Josevski Dec 23 '09 at 6:29

From <http://stackoverflow.com/questions/203863/missing-xml-comment-for-publicly-visible-type-or-member>