SDL Web 8 Developer (.Net) Content Interaction Library (CIL)

Quick Start Guide

Updated: 14 September 2016

Version 1.1 – Document owner: Richard Hamlyn

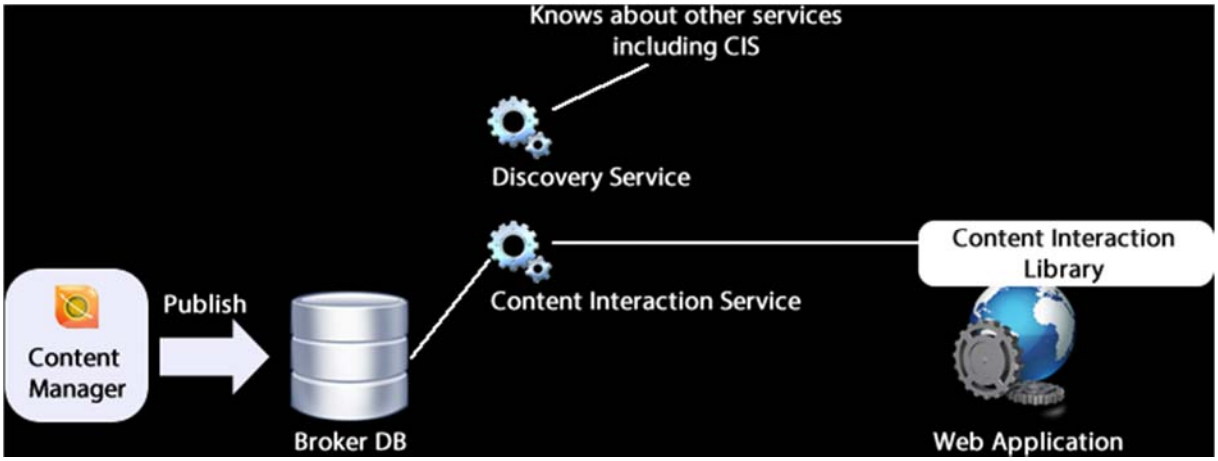
Content

Introduction

This short introduction to development with SDL Content Interaction Library (CIL) will introduce you to the basic concepts and knowledge required to commence development work. CIL is a client API that allows you to interact with an SDL Web 8 Content Interaction Service.

The latest CIL is available through Nuget and this guide will show you how to implement it in your project. Ref: https://www.nuget.org/packages/Sdl.Web.Cil/

Overview:



This short guide will:

Introduce the Content Interaction Library (CIL).
Setup a basic .Net project in Visual Studio.
Teach you how to consume content that was Published from an SDL Content Manager.

Pre-requisites:

.Net 4.5.
Visual Studio.

Questions or help

If you have any questions or you require any further help then please post to the SDL Community (http://community.sdl.com). Please remember to identify the guide, including the fact that it is the Quick Start Guide (CIL) you are referring to.

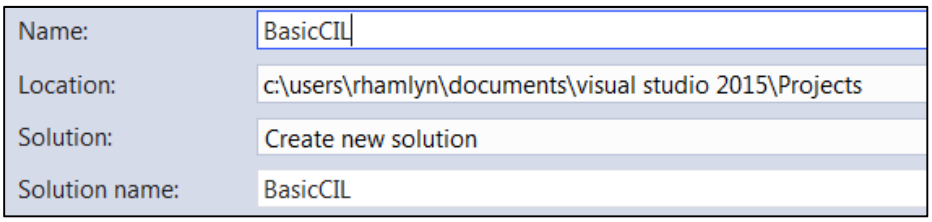
Setting up your .Net project

In this practical session, you will setup a basic .Net Web Application that you will later use to present content through SDL Web's RESTfull Content Interaction Service.

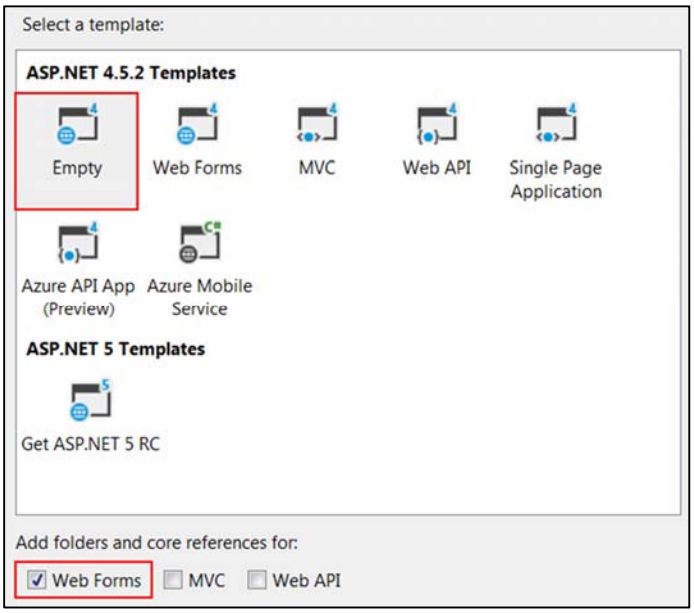
1. Open Visual Studio.

2. Create a new project as follows:
   a. From the 'File' menu select 'New' – 'Project'.
   b. Select 'ASP.Net Web Application'

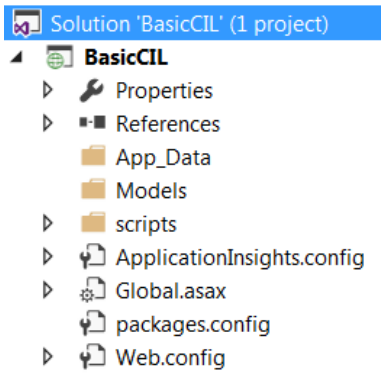
   c. Name the Project 'BasicCIL' as shown and click 'OK'.



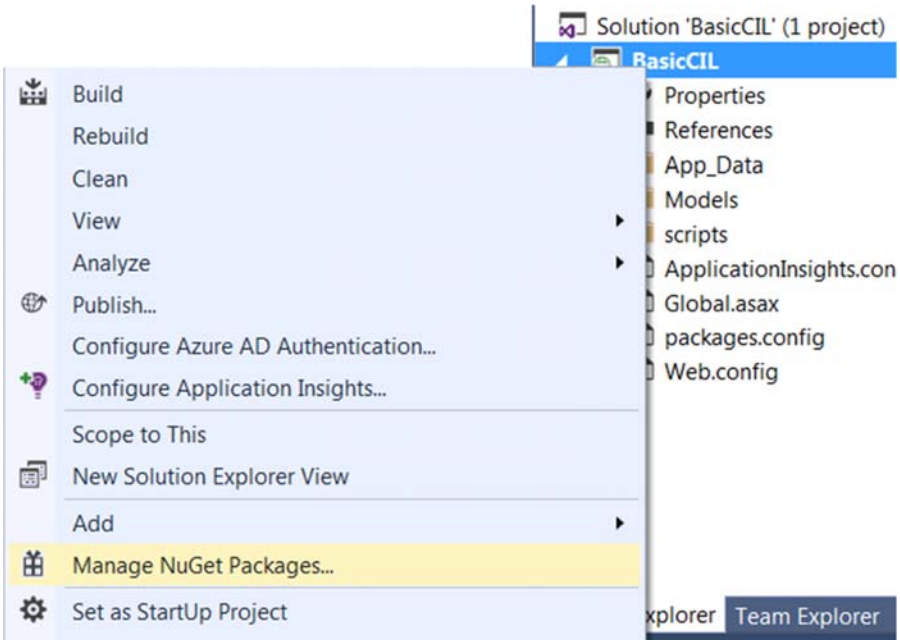
   d. Select the 'Empty' template and check the box to add Web Forms as shown:



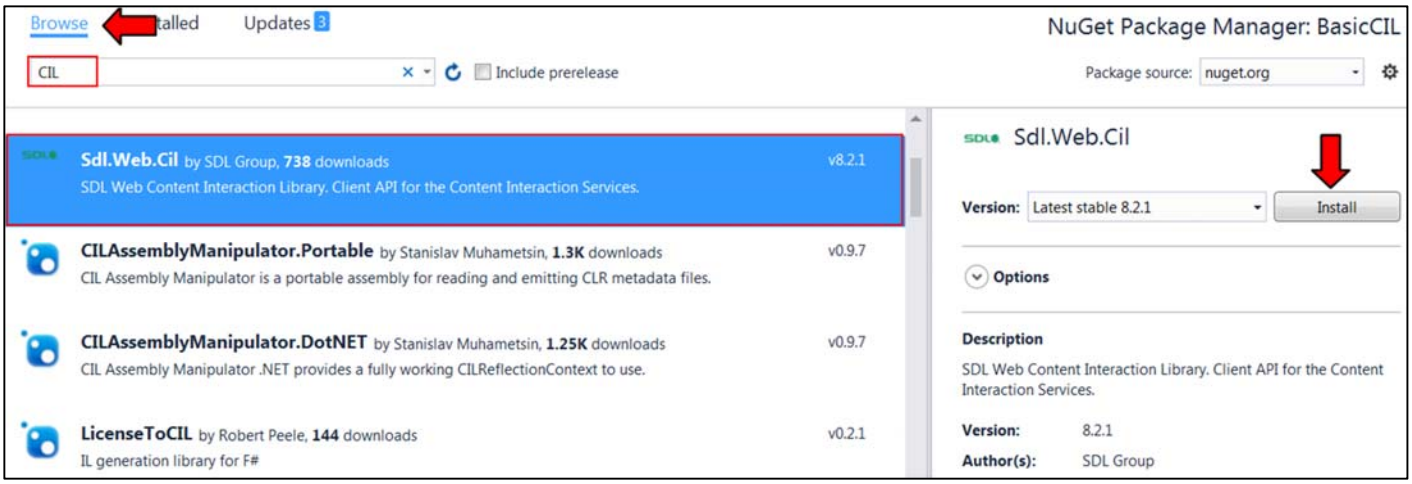
You will see the Project created as shown:



3. Add SDL's Content Interaction Library (CIL), so that you can interact with the Content Interaction Service's API, to your Project from Nuget.
    a. Right-click your Project and select 'Manage Nuget Packages…' as shown.



b. Click on the ‘Browse’ option; enter ‘CIL’ in the search box. Select ‘Sdl.Web.Cil’, as shown and click on ‘Install’.

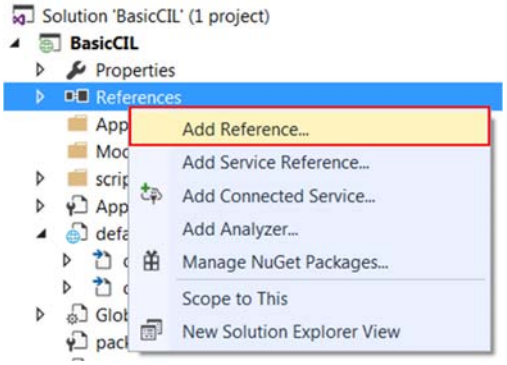


If a window pops up asking you to ‘Review Changes’ simply click ‘OK’.

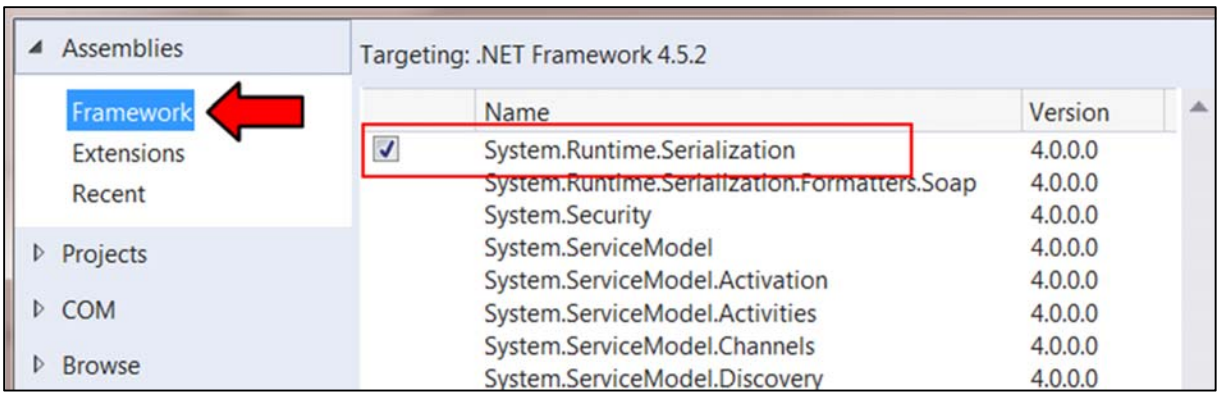
c. Accept the license agreement and wait whilst the CIL Packages are added to your Project.
    i. The ‘Output’ Window in Visual Studio should show something similar to:

```
Successfully installed 'Sdl.Web.Cil 8.2.1' to BasicCIL
========== Finished ==========
```

d. Now you need to add a reference to the Project so that you can serialize JSON.
e. Right click on the References node in your Project and select ‘Add reference’ as shown:



f. In the ‘Assemblies’ – ‘Framework’ option check ‘System.Runtime.Serialization’ and click ‘OK’.



4. Tell your Project how it can communicate with the pre-implemented SDL Web environment.

   a. Open the Project’s Web.config file.
   b. Add the following within the <configuration> node of the XML:

```
<configuration>
  <appSettings>
    <!-- SDL Web 8 CDaaS configuration settings: -->
    <add key="discovery-service-uri" value="http://sdl-training.com:8082/discovery.svc" />
    <!-- Security - Currently not being used by CDaaS -->
    <add key="oauth-enabled" value="false" />
    <add key="oauth-client-id" value="cduser" />
    <add key="oauth-client-secret" value="CDUserP@ssw0rd" />
   </appSettings>
```

Note that the security ‘OAuth’ settings have been left in but set to disabled. The service you will use for this example does not have OAuth enabled but a live service probably would and this is where you would set the credentials!

Note that the Discovery Service is associated with a read only environment set up for use with the Quick Start Guide Series. You can change this to any Web 8 Discovery Service!

The Discovery Service knows about the other services, including the Content Interaction Service, that you might want to communicate with in your project.

Create code to communicate with SDL Web through CIL.

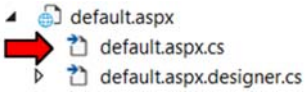
1. You will now create the initial code (not completed) to ensure you are able to communicate with the SDL Web instance and to define the response.

   a. Right-click on your Project node in Solution Explorer and select 'Add' – 'New item'.
   b. Select 'Web Form' and name the form 'default.aspx'.

   c. Add the following code default.aspx (we will use this later).

```
<body>
    <form id="form1" runat="server">
    <div>
        <h2><asp:Label ID="titleLbl" runat="server" /></h2>
        <p><asp:Label ID="headlineLbl" runat="server" /></p>
        <asp:Image ID="image" runat="server" BorderWidth="2" />
    </div>
    </form>
</body>
```

   d. Save and close default.aspx.
   e. Open default.aspx.cs.



   f. Add the following Namespace to your class:

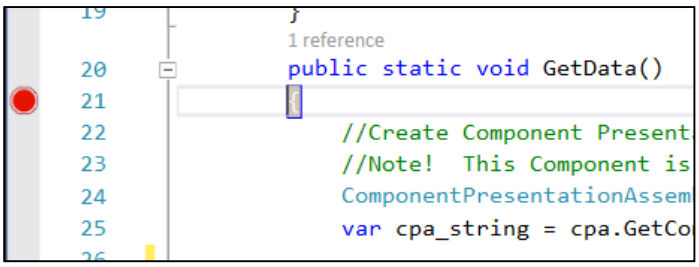
```
using Tridion.ContentDelivery.DynamicContent;
using System.IO;
using System.Text;
using System.Runtime.Serialization.Json;
```

g. Now create a new method and trigger the method when the page loads as shown below:

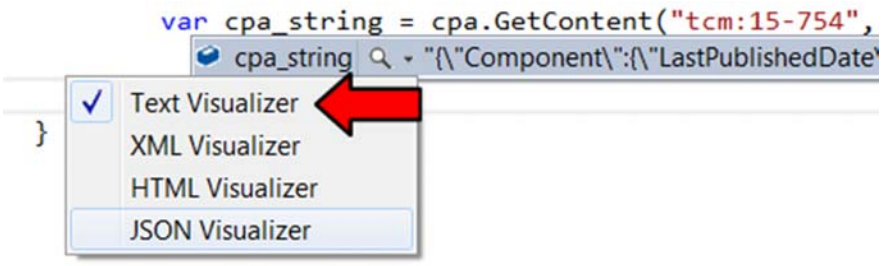
```
protected void Page_Load(object sender, EventArgs e)
 {
      GetData();
 }
 public static void GetData()
{
      //Create Component Presentation object and grab a News Article that has been
      Published to the Broker
      //Note!  This Component is a Promotional Item also used for SmartTarget
      ComponentPresentationAssembler cpa = new ComponentPresentationAssembler();
      var cpa_string = cpa.GetContent("tcm:15-754", "tcm:15-718");

}
```

2. You now need to create a class so you can store the response from SDL Web in an Object. You only need to do this once and the class can then be reused for all subsequent responses of similar type from Content Interaction Service.

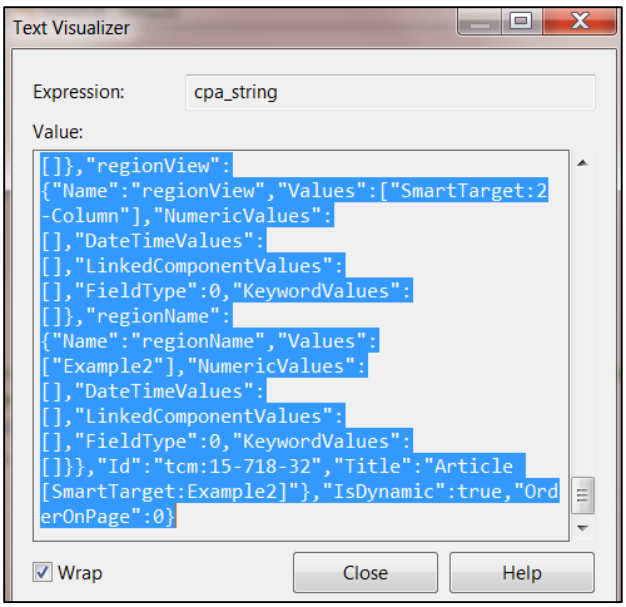
   a. In your default.aspx.cs file set a break point as shown:



   b. Now run the Project until it hits the break.
   c. Step into the code until you hit the closing method curly brace '}' and then select cpa_string, click on the down arrow and select 'Text Visualizer' as shown:



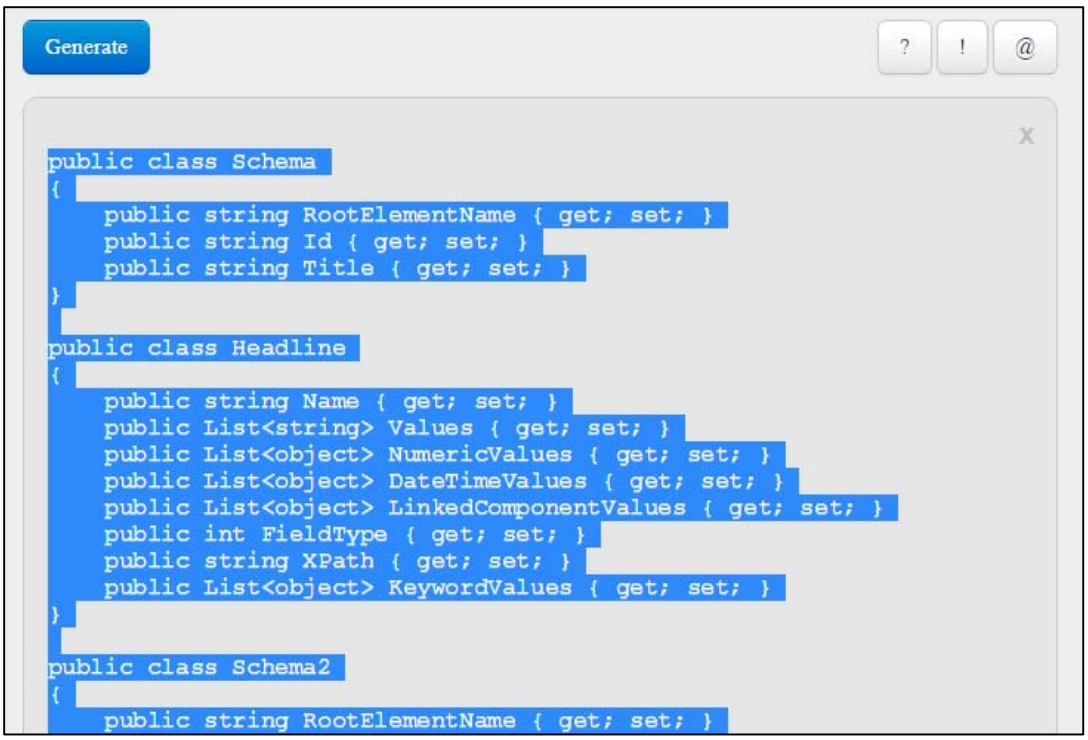
d. From the window that pops up, select the JSON response in the Value pane and copy it.



Ensure you copy all of the JSON text!

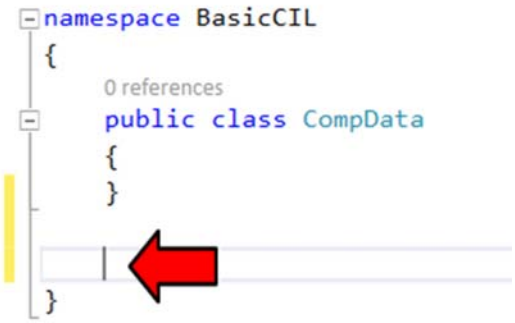
e. Now, open a browser window and navigate to http://json2csharp.com/
f. Enter your JSON response that you copied from the Text Visualizer previously in to the box and click ‘Generate’.

g. This will create a skeleton of all the classes you need to represent the JSON response for the News Article requested in your code.
h. Copy the class list as shown:



Ensure you copy all of the classes!

i. Back in your Project in Visual Studio, right-click on your Project node in Solution Explorer and select 'Add' – 'Class'. Name the Class 'CompData'.
j. Paste the skeleton Class list, copied above, in to the new Class below the class method as shown by the arrow below:



k. Scroll to the bottom of the class and you will see a ‘RootObject’ Class. Copy the methods from the class, highlighted by the red box below:

```
public class RootObject
{
    0 references
    public Component Component { get; set; }
    0 references
    public ComponentTemplate ComponentTemplate { get; set; }
    0 references
    public bool IsDynamic { get; set; }
    0 references
    public int OrderOnPage { get; set; }
}
```

l. Navigate to the top of the Class and paste the methods in the class’ main method as shown:

```
0 references
public class CompData
{
    0 references
    public Component Component { get; set; }
    0 references
    public ComponentTemplate ComponentTemplate { get; set; }
    0 references
    public bool IsDynamic { get; set; }
    0 references
    public int OrderOnPage { get; set; }
}
```

m. Now scroll back to the ‘RootObject’ Class and delete it.

You may get a conflict reported by Visual Studio about having a ‘Name’ method in the ‘Name’ Class. You can fix this by changing the method name to ‘Names’ as shown:

```
public class Name
{
    0 references
    public string Names { get; set; }
    0 references
```

n. Save and close the class.

Deserialize the JSON Component response from the Content Service.

1. Create a Class to store the deserialized response. This will allow you to work with the content later!

    a. Create a new class in your Project named 'SimpleComp'. Add the following code:

```
public class SimpleComp
    {
        //Simple class to hold some of the Component elements
        public string title { get; set; }
        public string headline { get; set; }
        public string imageId { get; set; }
        public string imageUrl { get; set; }

    }
```

    b. Save and close the new class.

2. Add the code to make this all meaningful.

    a. Add the following code to the GetData() method in your default.aspx.cs.

```
public static SimpleComp GetData()
        {
            //Componnet object variables
            String title;
            String headline;
            String imageUrl;

            //Create SimpleComp object
            SimpleComp scomp = new SimpleComp();

            //Create Component Presentation object and grab a News Article that has been
            Published to the Broker
            //Note!  This Component is a Promotional Item also used for SmartTarget
            ComponentPresentationAssembler cpa = new ComponentPresentationAssembler();
            var cpa_string = cpa.GetContent("tcm:15-754", "tcm:15-718");

            DataContractJsonSerializer serializer = new
DataContractJsonSerializer(typeof(CompData));
            using (var ms = new MemoryStream(Encoding.Unicode.GetBytes(cpa_string)))
            {
                //Pass Component object to CompData class so we can break it down
                var componentData = (CompData)serializer.ReadObject(ms);
```

```
                //Grab some Component elements and pass to variables to use in html
                headline = componentData.Component.Fields.headline.Values[0];
                imageUrl =
componentData.Component.Fields.image.LinkedComponentValues[0].Multimedia.Url;
                title = componentData.Component.Title;

                //Popuate the SimpleComponent object
                scomp.title = title;
                scomp.headline = headline;
                scomp.imageUrl = imageUrl;
            }
            //Return SimpleComponent object to calling method
            return scomp;
        }
```

b. Now update the PageLoad method to call the GetData() method and populate the HTML elements (created earlier) in the default.aspx file.

```
protected void Page_Load(object sender, EventArgs e)
        {
            String baseUrl = "http://sdl-training.com/";

            SimpleComp myComp = GetData();
            if (myComp != null)
            {
                titleLbl.Text = myComp.title;
                headlineLbl.Text = myComp.headline;
                image.ImageUrl = baseUrl + myComp.imageUrl;
            }
            else
            {
                titleLbl.Text = "No Component Loaded";
            }
        }
```

c. You can run your Project (without break points) and you should see something like the following:

Winter Wonderland

Get in the spirit!