GETTING STARTED WITH THE SILVERPOP API

June 5, 2015

OVERVIEW

1. Background

This is an article in a series that can be found at: http://c-l-associates.com/Home/SilverPop. These articles should be followed in the following order:

1. http://c-l-associates.com/Home/Getting_Started_With_the_SilverPop_API
2. http://c-l-associates.com/Home/Login_and_Logout_With_the_SilverPop_API
3. http://c-l-associates.com/Home/Setting_Up_a_SFTP_Client
4. http://c-l-associates.com/Home/SilverPop_API_ImportList
5. http://c-l-associates.com/Home/SilverPop_API_SaveMailing
6. http://c-l-associates.com/Home/SilverPop_API_GetLists
7. http://c-l-associates.com/Home/SilverPop_API_ScheduleMailing

This article explains how to get started using the IBM SilverPop API. This article will focus on doing API development in Microsoft C#. SilverPop provides very few code samples for API development. In addition, their code samples are in PHP, Curl and Java. If you are using any other language, you are pretty much on your own. Don't expect SilverPop support to look over your C# code and tell you what is wrong.

When doing SilverPop API development, you will need to get familiar with 4 things provided by SilverPop. You will struggle until you get familiar with the following:

1. The XML API guide. There is another API called the Transact API; we will not be discussing that. The Transact API has rather limited functionality for doing mass mailings. If you are say, sending out 1,000 emails, you will not want to use the Transact API. It's use is in doing a dozen emails, not sending in mass.
2. The XML test harness. This is a small web page where you can test XML snippets to see how the API works. It is an essential tool once you get underway.
3. The Engage portal. Everything happens in the Engage portal. As you are posting requests to the API server, things will happen in the Engage portal. It is essential to understanding how SilverPop Engage works. It is also a great debugging tool.
4. The SilverPop support portal. The SilverPop support staff is very good. You will need them at some point. You will need to gain separate access to the support portal apart from the Engage portal.

PROJECT OVERVIEW

1. What We Will Be Coding

We will be building a solution for interacting with the SilverPop API. The solution will contain 2 projects:

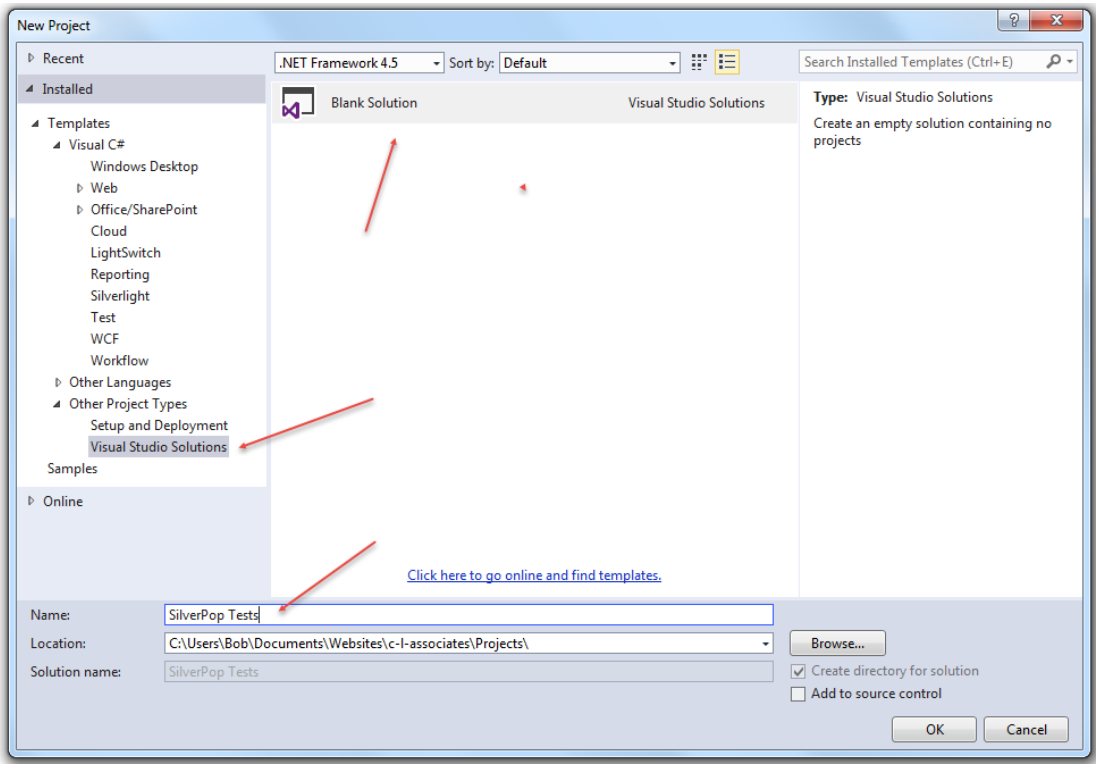
1. A SilverPop API service. This will be a regular C# class library where we will isolate the code necessary to interact with SilverPop.
2. A unit test client project where we will test the SilverPop API.

2. Getting Ready to Access the SilverPop API

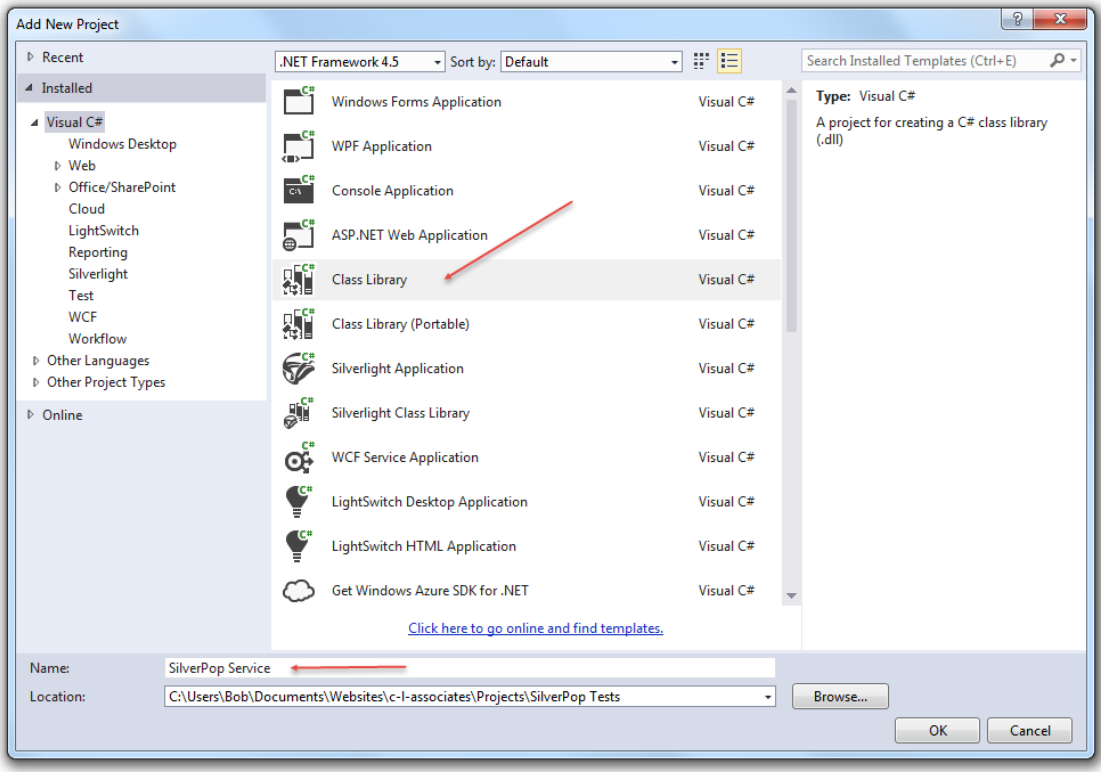
1. First, you need to become a SilverPop subscriber. This involves getting your organization setup with SilverPop.
2. Once, you are a subscriber, you need to gain access to the Engage portal. In this document I will not document using the Engage portal. I assume you will gain that knowledge elsewhere. I will only document things about the Engage portal that help in debugging and viewing the results of the API.
3. Setup an API user for SilverPop. You will need to do two things:
    a. Indicate that this user will be an API user.
    b. In the firewall settings, you will need to put in your public IP address. This is the IP address for your development machine. It is best to have a static IP address, or the SilverPop API will begin failing once your DHCP address changes.
4. Get a copy of the latest XML API PDF document.
5. With the XML SDK you should also get a copy of the HTML test harness.
6. Gain access to the support portal. This will be a different login from your Engage login.

3. Creating the Initial Solution

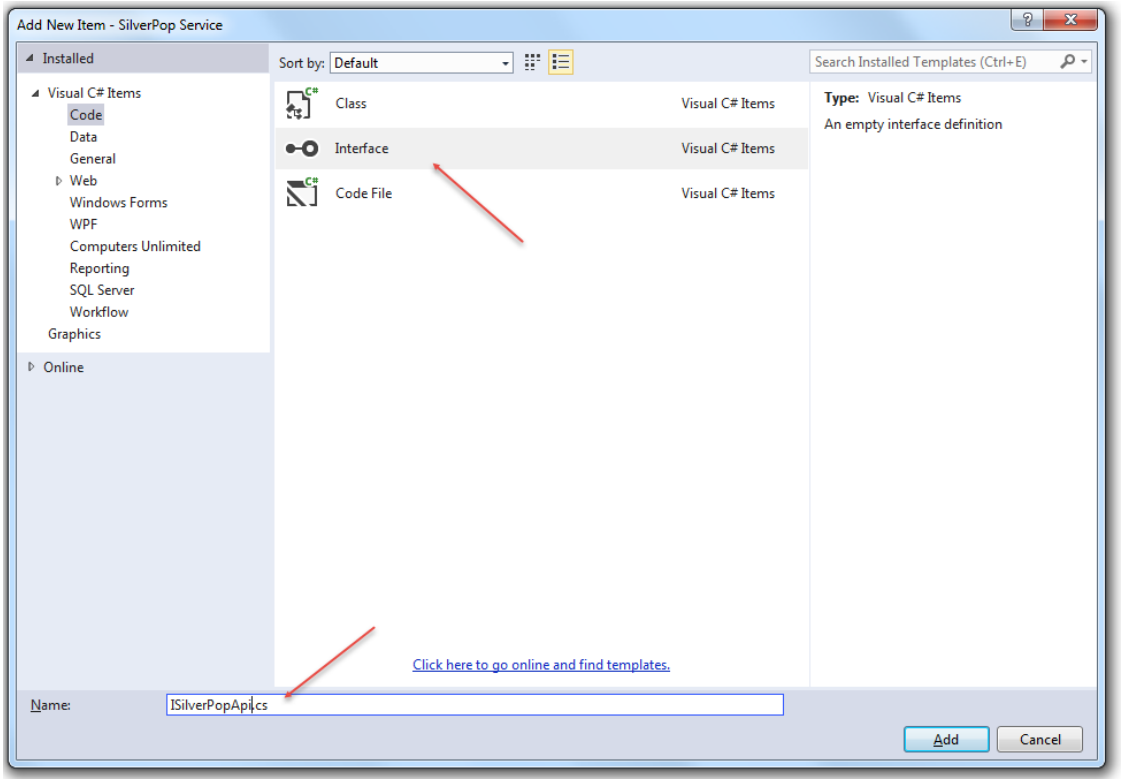
1. Fire up Visual Studio.
2. Create a new solution using the blank solution template. Call it SilverPop Tests:



3. Create a new class library project, called SilverPop Service:



4. Delete the default Class1.cs file in the new project.
5. Add a directory called Interfaces.
6. Add a new interface called ISilverPopApi under the Interfaces directory:



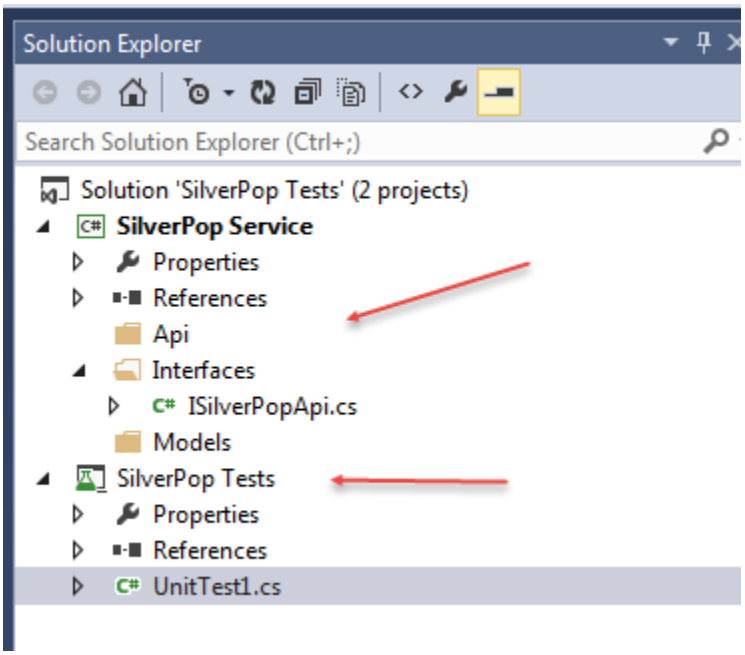
7. Your code should look like this. We will need the enums later:

```
namespace SilverPop_Service.Interfaces
{
        public enum Visibility
        {
                Private = 0,
                Shared = 1
        };

        public enum ListType
        {
                Database = 2,
                Suppression = 13
        }

        public interface ISilverPopApi
        {
        }
}
```

8. Add two more folders under the SilverPop service: Models and Api.
9. Add a new unit test project to the solution called: Silver Pop Tests. Your solution should look like this:



10. That is the end of our initial setup. The next article in this series will show how to log in and log out of the SilverPop API. The next article in this series can be found at: http://c-l-associates.com/Home/SilverPop.