SpotiApp framework

Developer manual. Revision 1, 2010-05-26 13:20

Important advicory

NOTE: SpotiApps has no official link with the Spotify application. It's a third party platform/app which are provided "as-is" and with no gurance. You're installing and using this application on your own liability and responsibility. Please do not use the application in any way which imply and rights to Spotify.

We will NOT accept any creation of "SpotiApps" which infrage any rights belonging to Spotify or their partners. Making AddOns whose purpose is to rip or do other unauthorized copying of content anvailable on Spotify will not be tolerated.

SpotiApp will be runned on a ad-hoc basis and will in no way change the application or its default data. It will run alongside the application and the user have to run it simulantely to get it work.

SpotiApp is only anvailable for the windows platform. There is currently no plans to port it to any other platforms.

License

SpotiApps is released under a propierty license belonging to Alexander Forselius.

The spotiApp framework consists of a overlay application which adds a versatile extension system for Spotify. By SpotiApp running, the users can access 3rd party plugins for Spotify following the Spotify gui system ”inside” the spotify application.

The application is intended for playlist management, so third party spotify sites can integrate their service directly into the Spotify window.

Developing SpotiApp tutorial

Download the Spofity Library from spotiapps.krakelin.com and place it to a apporiate folder.

Start Visual Studio/Express. Select **New > Project.** Choose ”Class Library”. Add a referens to the dll you downloaded from the site.

Then you modify main class in this way. Let the names be untouched. Our SpotiApp is assembled as ”Test”.

Then you need to add some important references. This references must be added in addition to the SpotiApp .exe assembly:

```
System.Windows.Forms
System.Drawing
System.Drawing.Drawing2D
Spofity;
```

WARNING! There is a issue with Visual Studio 2010 causing the studio to crash if you try to open the class with double click, because the derived form consues the visual studio form editor.

SpotiApp.cs

```
using System;
using System.Collections.Generic;

using System.Text;
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Design;
using Spofity;
namespace Test
{
    public class MainForm : Form1
    {


        protected override void OnLoad(EventArgs e)
        {



        }
    }
}
```

Then right-click on the project, select Add | New.. and choose ”User Control”. Give a name ”Home” to the User Control and double click it.

You'll see that the development goes exactly as you have done before in C#. SpotiApps are simple C# application that runs ”inside” Spotify...

You have access to the most common controls that Spotify uses for their native pages. The controls is even customized to get a so native experience as possible. Here is a explaination of some:

SpotifyLabel

A preconfigured label control to fit the Spotify interface.

SpotifySmallLabel

A smaller version SpotifyLabel

CDCover

This control can be used to create a cover image like those in artist view.

SpotiEntry

SpotiEntry is a control that works similar to list entries in Spotify. SpotiApp framework automatically binds them together so they behave as real selections.

Setting up the runtime

When you're ready with your form, go back to the main form and write

```
Home MyHome;
public override void OnLoad(EventArgs e){
     MyHome = new Home();
     Home = AddSection(”Home”,MyHome);
     SetActiveSection(”Home”);
}
```

The view will be bound to a ”Home” section.

Testing

To test the runtime, convert the project to a Windows application and create a new class ”Program.cs”. Then write this:

Program.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Runtime.CompilerServices;
using System.Reflection;
using System.Xml;
namespace Test
{
    static class Program
    {
        [STAThread]
        static void Main(string[] args)
        {

            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);


            MainForm D = new MainForm();
            D.FormBorderStyle = FormBorderStyle.Sizable;
            D.TestMode = true;

            Application.Run(D);


        }
    }
}
```

Notable functionality of the Spofity form:

- OpenURI(URI) – Opens a spotify URI, for example a such a playlist. The view will automatically be reverted back to the original spotify view, but be able to view it again by pressing back on Spotify

Feeds

To add dynamic content, such playlists from search queries, you cann add and update a set of elements by the 'AddFeed' function. The function is specified as this:

AddFeed(string Section,string ElementType,string URI,int Limit,Point Location, string nspace,EventHandler Enter)

```
 <param name="Section">The section of the SpotiApp the feed will be applied to</param>
        /// <param name="elementType">Type of elements that can be created.
Valid values is:

sp:entry = A list entry, automatically integrates with each other
sp:button = A button
</param>
        /// <param name="URI">The URI to the feed (must be RSS 2.0)</param>
        /// <param name="limit">The limit of amount elements to add</param>
        /// <param name="Location">Location of the feed</param>
        /// <param name="nspace">Space of the feed</param>
        /// <param name="Enter">Event that occur when the user activates the feed
elements,</param>
```

Handling URI data from a list view

The most common usage of this control is to gather list of playlists from other sections. Those playlists are bounded to their Uris. The HREF tag links to the relative page of the feed entry.

For example, let say you generate playlists from a feed here, and then want spotify to open the playlist instantly when the user choose the entries:

```
void Entry_Onclick(object sender, EventArgs e)
{
     Element Sender = (Element)((Control)sender).Tag;
     this.OpenSpotifyLink(GetURIFromPage(new Uri(Sender.GetAttribute("href"))));
}
```

Then associate a new feed by this:

```
AddFeed("Home","sp:entry","<MyFeedURI>",5,new Point(42,270),”MyFeed”,new
EventHandler(Entry_Onclick));
```

Replace <MyFeedURI> with a proper URI. Associate it with a button onclick and the program will fix the most work for you.

Issues when running live

The runtime is under development. Issues I'm aware of is described here:

- Menus goes under the SpotiApp section
- SpotiApp section flicker during resize/move of the Spotify window.
- Some slowness in some areas due to window sensors.