Smiley.NET is an FBML facebook application that uses the Facebook Developer Toolkit. This sample is meant to mimick the Smiley sample produced by Facebook in PHP. The idea behind this application is to show techniques for integrating your application within the Facebook Platform and also ideas for writing an FBML app in ASP.NET. Before we walkthrough the code, I thought it would make sense to link to a couple of places that anyone writing a Facebook application should know.

Useful Links

The platform homepage – This includes links to various articles, tools and documentation

Anatomy of a Facebook Application – This walks through the various concepts that explain how an application fits into the facebook platform. This article actually uses the smiley app as an example for several of the concepts.

How to write an ASP.NET Facebook Application – A great walkthrough explaining the basics of using the facebook developer toolkit.

Facebook Developer Toolkit – Open source project that wraps much of the Facebook platform to make it easier to use from .NET.

Facebook API Doc – Part of the facebook developer wiki where the various API methods are described.

Now let's take a look at the code. This sample is in the following directory in the codeplex source $/FacebookToolkit/Samples/FBMLCanvasSample.

The first thing to do is setup your facebook application at the facebook develop application. This is where you will name your application and also get an API Key and Secret. You really can't do anything without your API Key and Secret. After getting this setup, you can get Smiley.NET to function.

I am going to look at the code first, and second describe the process of setting this application up to run. So, to start making this application I created a new Web Application. In this Web application, I added references to facebook.dll and facebook.web.dll from the toolkit. The facebook assembly has the code that I can use to call the various facebook apis. facebook.web assembly has some controls and code that I can leverage to make it much easier for my app to live in the facebook platform.

Setup.aspx (Application configuration and Template creation)

If you plan to have your application integrated with the feed system and the publisher, you will need to learn about how these work. The links below are great starting points.

http://wiki.developers.facebook.com/index.php/Feed

http://wiki.developers.facebook.com/index.php/New_Design_Publisher

Now that you have the overview, You will need to setup up the feed and publisher urls and create your templates. To do this you can use the tools on the facebook developer site as described in the links below, or you can use the facebook REST api. For this sample, we constructed a page that needs to be invoked one time after the application is deployed. The page is config/setup.aspx. This page has code to set the app properties, and create two feed templates. After these are created, the web.config is updated with the ids that were generated and then we can start publishing.

Here is the code used to set the app properties, this is done in place of the edit settings feature at www.facebook.com/developers.

```
            var dict = new Dictionary<string, string>
            {
                {"application_name","Smiley.NET"},
                {"callback_url",callback},
                {"tab_default_name","Smile.NET"},
                {"profile_tab_url","mysmiles.aspx"},
                {"publish_action","Smile at!"},
                {"publish_url",callback +
"handlers/otherPublishHandler.aspx"},
                {"publish_self_action","Smile!"},
                {"publish_self_url",callback +
"handlers/publishHandler.aspx"},
                {"info_changed_url",callback + "handlers/infoHandler.aspx"},
                {"wide_mode","1"}
            };
            this.API.admin.setAppProperties(dict);
```

Next we setup 2 feed template bundles. Each template bundle contains a 1 line version, a short story version and a long version. An app can have up to 100 templates. In this case, we have one for what is published to the users feed (via New Smiley functionality) and 1 bundle for when a smile is sent to a friend via Send Smiley functionality. I have included the code for creating one of the bundles here.

```
            var one_line_story = new List <string>{"{*actor*} is feeling
{*mood*} today"};
            var short_story = new List<feedTemplate>();
            var short_story_template = new feedTemplate
                                        {
                                            TemplateTitle = "{*actor*} is feeling
so {*mood*} today",
                                            TemplateBody = "{*actor*} just wanted
to let you know that he is so {*mood*} today",
                                            PreferredLayout = "1"
                                        };
            short_story.Add(short_story_template);

            var full_story = new feedTemplate
```

```
                                    {
                                        TemplateTitle = "{*actor*} is feeling
very {*mood*} today",
                                        TemplateBody = "<div style=\"padding:
10px;width : 200px;height : 200px;margin: auto;text-align: center;border:
black 1px;cursor: pointer;border: black solid 2px;background: orange;color:
black;text-decoration: none;\"><div style=\"font-size: 60pt;font-weight:
bold;padding: 40px;\">{*emote*}</div><div style=\"font-size: 20px; font-
weight:bold;\">{*mood*}</div></div>"
                                    };
            if (string.IsNullOrEmpty(t1))
            {

long bundle1id = this.API.feed.registerTemplateBundle(one_line_story, short_s
tory, full_story);
```

Masterpage

Now, I create my masterpage. In this case, I needed to pick between using a masterpage to handle the facebook authentication handshake and a basepage. The masterpage is a little cleaner in my opinion and really fits exactly how I want this application to work. In this project the master page is called FBMLMaster.master. Let's look at what this contains.

It defines some placeholder labels for fbml content I want to write from the codebehind, it also defines the dashboard and tabs that will be the container for my application, lastly it defines a content area where each "page" will display its content.

```
<asp:label runat="server" id="css" />
<asp:label runat="server" id="js" />
<asp:label runat="server" id="header" >
<fb:dashboard/>
<fb:tabs>
<fb:tab-item title="Public"  href="default.aspx"
selected="<%=Convert.ToInt32(selected=="default") %>"/>
<fb:tab-item title="Home"  href="home.aspx"
selected="<%=Convert.ToInt32(selected=="home") %>" />
<fb:tab-item title="My Smiles"  href="mysmiles.aspx"
selected="<%=Convert.ToInt32(selected=="mysmiles") %>"/>
<fb:tab-item title="New Smiley"  href="newsmiley.aspx"
selected="<%=Convert.ToInt32(selected=="newsmiley") %>"/>
<fb:tab-item title="Send Smiley"  href="sendsmiley.aspx"
selected="<%=Convert.ToInt32(selected=="sendsmiley") %>"/>
</fb:tabs>
</asp:label>
<div id="main_body">
    <asp:ContentPlaceHolder ID="body" runat="server">
    </asp:ContentPlaceHolder>
</div>
```

The code-behind for the master page has a couple of critical things going on. First it derives from facebook.web.CanvasFBMLMasterPage. This is the base MasterPage that will handle the authentication handshake with facebook.

The other 2 things that you will find here are. 1) the SetSelectedTab function. This a function that can be called from each "page" such that we can set the selected property in the fb:tabs markup. 2) Insertion of the CSS and Javascript. In the load of the master page, I am checking if the current page is being shown in a profile tab or if it is being shown in the application canvas. When shown in the profile, I need to inject the CSS inline and hide the tabs, when in the canvas, I can just add a link reference that the fbml rendering engine at facebook can handle.

```
        protected void Page_Load(object sender, EventArgs e)
        {

if (!string.IsNullOrEmpty(Request.Params["fb_sig_in_profile_tab"]))
            {
                header.Visible = false;

css.Text = FBMLControlRenderer.RenderFBML("~/controls/FBMLCSS.ascx");
            }
            else
            {
                css.Text = string.Format("<link rel=\"stylesheet\"
type=\"text/css\" href=\"{0}css/page.css?id={1}\" />", callback, cssVersion);

js.Text = FBMLControlRenderer.RenderFBML("~/controls/FBMLJS.ascx");
            }
        }
```

The one thing you might notice is the call here to FBMLControlRenderer.RenderFBML. The FBMLControlRenderer is a class defined in facebook.web. This will just output raw FBML given a particular usercontrol. I find writing fbml directly in the user control and using this utility as a very clean way to format and write my fbml (much better than trying to piece it together in a code-behind). If you look at any of the UserControls in the controls directory of the Sample application you will see this in action.   There is another concept supported by the FBMLControlRenderer which relates to dynamic content. But, we will look at that with the ProfileBox.ascx.

Now that we have MasterPage that is linking our CSS and JS and controlling the authentication flow, we now need to implement the pages that are associated with the tab described above. The first page is Default.aspx. This page is included to demonstrate one thing, and that is a page that is accessible without logging into facebook.

Default.aspx (publically available canvas page)

The default page just includes some text within the content placeholder and then sets the RequireLogin property of the login and sets the Selected Tab.

```
        protected void Page_PreInit(object sender, EventArgs e)
        {
            Master.RequireLogin = false;
            Master.SetSelectedTab("default");
        }
```

A couple of important notes, this page is using the MasterType attribute so that we get a strongly typed instance of the master page in the code-behind without needing to cast.

```
<%@ MasterType VirtualPath="~/FBMLMaster.Master" %>
```

The other is that interaction with the master page needs to occur in Page_PreInit.

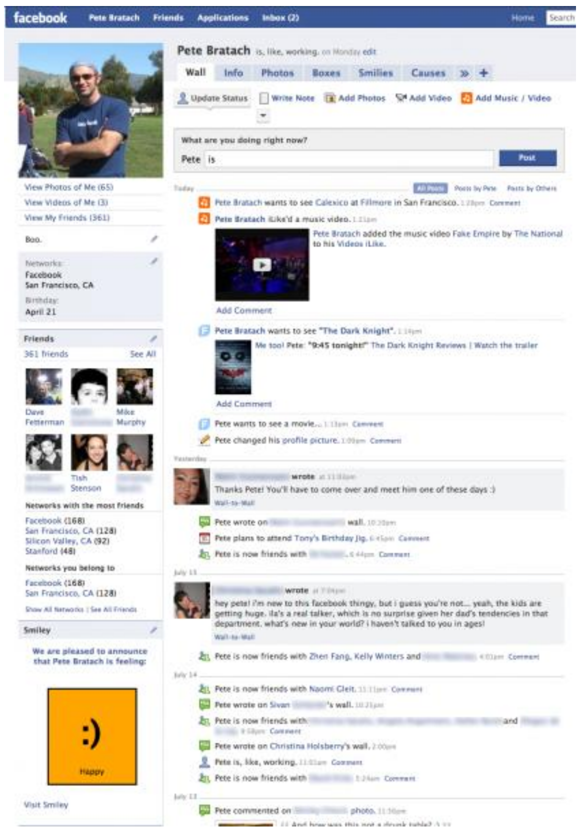
Home.aspx (Profile box, Application Info Sections and Extended Permissions)

This page demonstrates various things a canvas application can do to integrate with a user's profile. The first thing we are doing is in the page_load we are storing a variable on facebook to indicate whether we have setup this user before. If we have we don't need to do it again. The other thing we are doing is creating the info section for this app and the profile box. This is not a recommended approach, instead you should do this based on user action rather than just at load, but this just shows how to do it in the code-behind. We will also see how to add a button to your app for accomplishing the same things.

```
        protected void Page_Load(object sender, EventArgs e)
        {
            // You need to set info or profile box in order for the buttons 
on the page to show up.
            // Don't set them every time.
            var pref = this.Master.API.data.getUserPreference(1);
            if (pref != "set")
            {
                this.Master.API.profile.setInfo("My 
Smiles", 5, getSampleInfo(), this.Master.API.uid);

this.Master.API.profile.setFBML(this.Master.API.uid, null, getUserProfileBox(
), null);
                this.Master.API.data.setUserPreference(1, "set");
            }
        }
```

The profile box and info tab section are described in the anatomy of a facebook app. They are shown below in this context

Profile Box



Pete Bratach is, like, working. on Monday edit

What are you doing right now?

Pete is

View Photos of Me (65)
View Videos of Me (3)
View My Friends (361)

Boo.

Networks:
Facebook
San Francisco, CA

Birthday:
April 21

Friends
361 friends See All

Dave Fetterman
[redacted]
Mike Murphy
[redacted]
Tish Stenson
[redacted]

Networks with the most friends
Facebook (168)
San Francisco, CA (128)
Silicon Valley, CA (92)
Stanford (48)

Networks you belong to
Facebook (168)
San Francisco, CA (128)

Show All Networks | See All Friends

Smiley

We are pleased to announce that Pete Bratach is feeling:

:)
Happy

Visit Smiley

Today

Pete Bratach wants to see Calexico at Fillmore in San Francisco. 1:28pm Comment

Pete Bratach iLike'd a music video. 1:21pm

Pete Bratach added the music video Fake Empire by The National to his Videos iLike.

Add Comment

Pete Bratach wants to see "The Dark Knight". 1:14pm

Me too! Pete: "9:45 tonight!" The Dark Knight Reviews | Watch the trailer

Add Comment

Pete wants to see a movie... 1:13pm Comment

Pete changed his profile picture. 1:09pm Comment

Yesterday

[redacted] wrote at 11:31am

Thanks Pete! You'll have to come over and meet him one of these days :)

Wall-to-Wall

Pete wrote on [redacted] wall. 10:20pm

Pete plans to attend Tony's Birthday Jig. 6:45pm Comment

Pete is now friends with [redacted]. 6:44pm Comment

July 15

[redacted] wrote at 7:04pm

hey pete! i'm new to this facebook thingy, but i guess you're not... yeah, the kids are getting huge. ila's a real talker, which is no surprise given her dad's tendencies in that department. what's new in your world? i haven't talked to you in ages!

Wall-to-Wall

Pete is now friends with Zhen Fang, Kelly Winters and [redacted] 4:01pm Comment

July 14

Pete is now friends with Naomi Gleit. 11:11pm Comment

Pete wrote on Sivan [redacted]'s wall. 10:21pm

Pete is now friends with [redacted] and [redacted] 9:58pm Comment

Pete wrote on Christina Holsberry's wall. 2:00pm

Pete is, like, working. 11:01am Comment

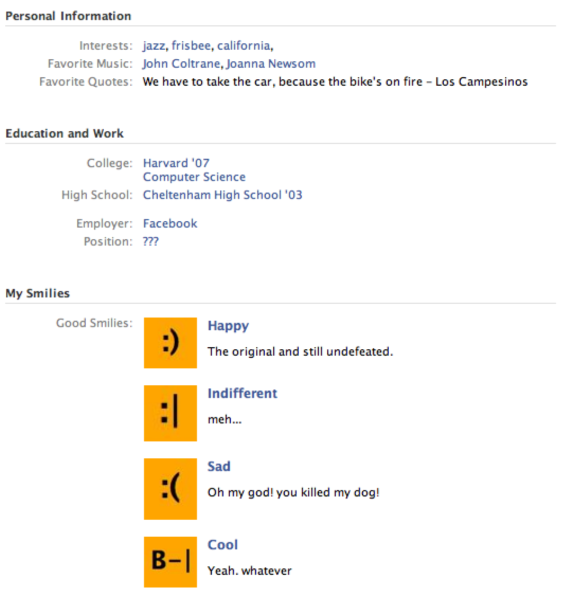
Pete is now friends with [redacted] 1:24am Comment

July 13

Pete commented on [redacted] photo. 11:30pm

And how was this not a drunk table? :) [illegible]

Info Sections



Personal Information

Interests:	jazz, frisbee, california,
Favorite Music:	John Coltrane, Joanna Newsom
Favorite Quotes:	We have to take the car, because the bike's on fire – Los Campesinos

Education and Work

College:	Harvard '07 Computer Science
High School:	Cheltenham High School '03
Employer:	Facebook
Position:	???

My Smilies

Good Smilies:

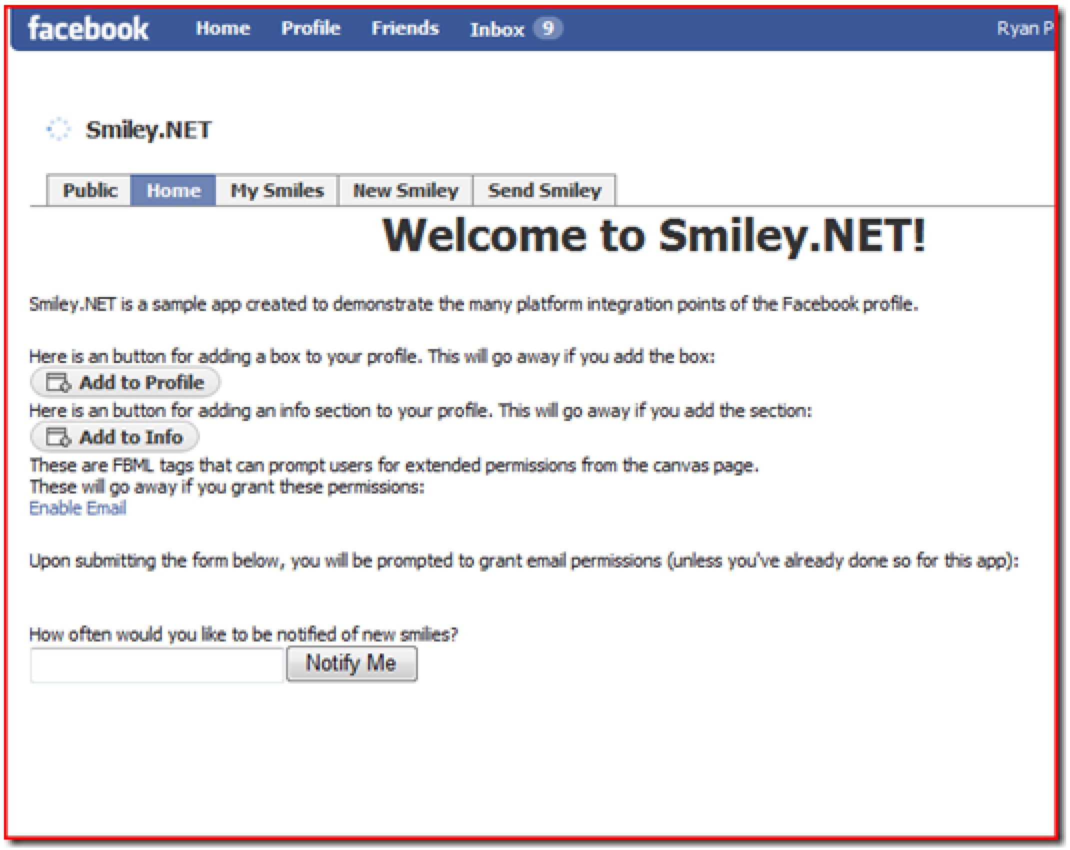
:) **Happy**
The original and still undefeated.

:| **Indifferent**
meh...

:(**Sad**
Oh my god! you killed my dog!

B-| **Cool**
Yeah. whatever

Next will look at the actual content of Home.aspx. This page demonstrates several more ways to integrate your application. Here is a view of the page, we will then examine each section (how it is done and what it does)



Welcome to Smiley.NET!

Smiley.NET is a sample app created to demonstrate the many platform integration points of the Facebook profile.

Here is an button for adding a box to your profile. This will go away if you add the box:

Here is an button for adding an info section to your profile. This will go away if you add the section:

These are FBML tags that can prompt users for extended permissions from the canvas page.
These will go away if you grant these permissions:

Upon submitting the form below, you will be prompted to grant email permissions (unless you've already done so for this app):

How often would you like to be notified of new smilies?

The first section is a button for adding a profile box (as shown above) for this application. The fbml for this button is

```
<!-- Profile box -->
Here is an button for adding a box to your profile. This will go away if you
add the box:
<div class="section_button"><fb:add-section-button section="profile"/></div>
```

The way that the add-section-button will work for the profile is that facebook will callback to the publisher that you have specified as the **Self-Publish Callback URL**. You need to implement a handler that supports the publisher interface to tell facebook what to display in the profile box. The publisher interaction is among the most complicated parts of building a facebook application. I am not going to go into the details here, but we can revisit in a future blog post if people are interested. In the meantime, there is a pretty good walkthrough out on the wiki. http://wiki.developers.facebook.com/index.php/CSharp_InLine_Publisher. In this case the Publisher for the profile box is publishHandler.aspx in the handlers folder.

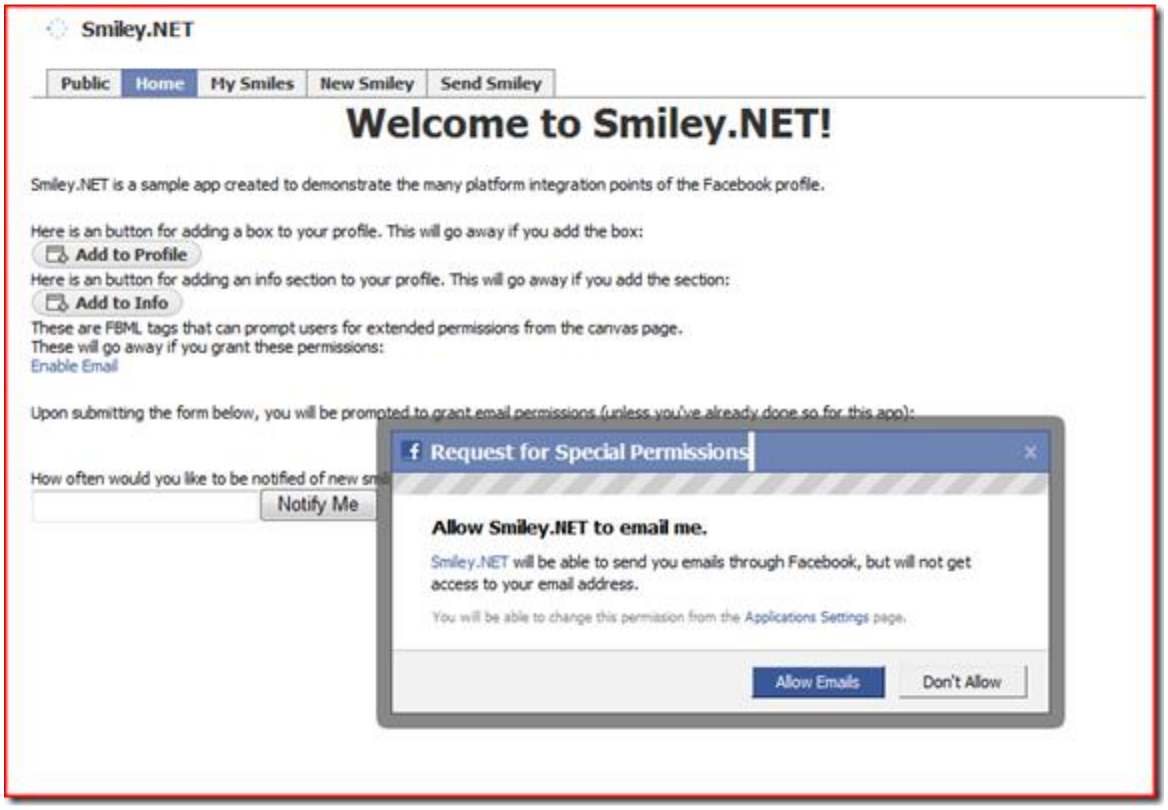
The next section is the similar to the first. Except this time it is for an info section and the publisher for this is infoHandler.aspx.

```
<!-- Info section -->
Here is an button for adding an info section to your profile. This will go
away if you add the section:
<div class="section_button"><fb:add-section-button section="info" /></div>
```

The next section demonstrates how to prompt the user for application specific permission. In the screen above, we see a link to enable email. This link disappears after the user grants the permission. The fbml here is:

```
<!-- Permissions -->
These are FBML tags that can prompt users for extended permissions from the
canvas page.<br />These will go away if you grant these permissions:<br />
<fb:prompt-permission perms="email">Enable Email</fb:prompt-permission>
<br />
<fb:prompt-permission perms="infinite_session">Enable Permanent
Login</fb:prompt-permission>
```

When the user clicks this link, they are prompted to allow the application to send emails, as shown here.



Welcome to Smiley.NET!

Smiley.NET is a sample app created to demonstrate the many platform integration points of the Facebook profile.

Here is an button for adding a box to your profile. This will go away if you add the box:

Add to Profile

Here is an button for adding an info section to your profile. This will go away if you add the section:

Add to Info

These are FBML tags that can prompt users for extended permissions from the canvas page.
These will go away if you grant these permissions:
Enable Email

Upon submitting the form below, you will be prompted to grant email permissions (unless you've already done so for this app):

How often would you like to be notified of new sm

Notify Me

Request for Special Permissions

Allow Smiley.NET to email me.

Smiley.NET will be able to send you emails through Facebook, but will not get access to your email address.

You will be able to change this permission from the Applications Settings page.

Allow Emails | Don't Allow

The last section shows how to have an input form that requests a specific permissions and submits some data.

```
<p>Upon submitting the form below, you will be prompted to grant email
permissions (unless you've already done so for this app):</p>
<form promptpermission="email"><br />How often would you like to be notified
of new smilies?<br /><input type="text" name="frequency"><input type="submit"
value="Notify Me"></form></p>
```

MySmiles.aspx (canvas page with dynamic content)

This page is a simple content page, that formats some fbml to show what we have stored about the user. This page shows a call to the data store api to get what we have stored. This application stores all historical smiles in the UserPreference bucket of the datastore api. There really isn't anything else going on. The interesting thing about this page, is that it doubles as the profile tab (if the user adds our app as the profile tab). It doesn't even have any code to handle this. This was all done in the master page, so this page can just function and it will work whether it is in our application canvas or a profile tab.

NewSmiley.aspx (Feeds and Publisher)

This page shows how to publish to the user profile using a feed form, also it show a little bit about using FBJS and AJAX. This page builds a grid of smiley images that when clicked will store your selected smile using the data store and then prompt to publish the selection to your wall.

First is the code used to store the selected Smile. It is not obvious how this works. First you must look at the BuildEmoticonGrid function in the BasePage. This builds the grid and sets the javascript events to call our javascript functions. In this case, we are using the EmoticonGrid that utilizes the javascript function called final which we will look at shortly. In other places, this same grid is used without this function. When we don't use the function, we are demonstrating used the feed form instead of the javascript showFeedDialog function that is used in the final function. More on that here, http://wiki.developers.facebook.com/index.php/Facebook.showFeedDialog. The main difference between the versions of BuildEmoticonGrid are what javascript function is tied to onclick. In this page, it is final.

```
public string BuildEmoticonGrid(Dictionary<string, string> moods, string call
back, string suffix, bool useFinalFunction)
        {
            var ret = new StringBuilder();
            ret.Append("<div class=\"table\"><div class=\"row\">");
            for (int i = 0; i < moods.Count; i++)
            {

var js = string.Format("final('{0}','{1}','{2}','{3}','{4}','{5}','{6}')", Fe
edTemplate1, callback + "images", "http://apps.facebook.com/"
+ suffix, callback, moods.ElementAt(i).Key, moods.ElementAt(i).Value, i);
```

```
                if (!useFinalFunction)
                {

js = string.Format("select('{0}','{1}','{2}')", moods.ElementAt(i).Key, moods
.ElementAt(i).Value, i);
                }
                if (i > 0 && i % 3 == 0)
                {
                    ret.Append("</div><div class=\"row\">");
                }
                ret.Append(string.Format("<div onclick=\"{0}\"
onmouseover=\"over('{3}')\" onmouseout=\"out('{3}')\" class=\"box\"
id=\"sm_{3}\"><div class=\"smiley\">{2}</div><div id=\"smt_{3}\"
class=\"title\">{1}</div></div>", js, moods.ElementAt(i).Key, moods.ElementAt
(i).Value, i));
            }
            ret.Append("</div></div>");
            return ret.ToString();
        }
```

Examining this code, the real key is that the js variable is assigned to the onclick property of each image. Now, let's look at the javascript. If you remember in the master page, the javascript is injected into the FBML from FBMLJS.ascx. Looking in this usercontrol, we find where the action is happening.

```
function final(template_id, image_src, base, callback, title, emote, id) {
  select(title, emote, id);
  var image = image_src + '/smile' + id + ".jpg";
  var template_data = { 'mood': title,
      'emote': emote,
      'mood_src': image,
      'images': [{ 'href': base, 'src': image}]
  };

  var ajax = new Ajax();
  ajax.responseType = Ajax.RAW;
  ajax.post(callback+'handlers/jsFeed.aspx', {'picked':id});

  Facebook.showFeedDialog(template_id, template_data, '', [],
                          function() {document.setLocation(base +
'/mysmiles.aspx');});
}
```

There are two things going on. First is an ajax call that is used to store the selected smile in the user's data store. This is done with a callback to jsFeed.aspx. This class takes in the selected index and updated the user's data store.

```
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!string.IsNullOrEmpty(Request.Params["picked"]))
            {
                var picked = int.Parse(Request.Params["picked"]);

var moodList = JSONHelper.ConvertFromJSONArray(this.API.data.getUserPreferenc
e(0));
```

```
                moodList.Insert(0, picked.ToString());

this.API.data.setUserPreference(0, JSONHelper.ConvertToJSONArray(moodList));
                var oldCount = 0;

if (!string.IsNullOrEmpty(this.API.data.getUserPreference(2)))
                {
                    oldCount = int.Parse(this.API.data.getUserPreference(2));
                }

this.API.data.setUserPreference(2, (oldCount + 1).ToString());
            }
        }
```

After this is done, the final function used some FBJS to pop a publisher dialog to allow the user to publish data. This is using passed in template id and the Facebook.showFeedDialog javascription function. We pass mysmiles as the return page after the publisher dialog is complete.

SendSmiley.aspx (Multifeed Story Form)

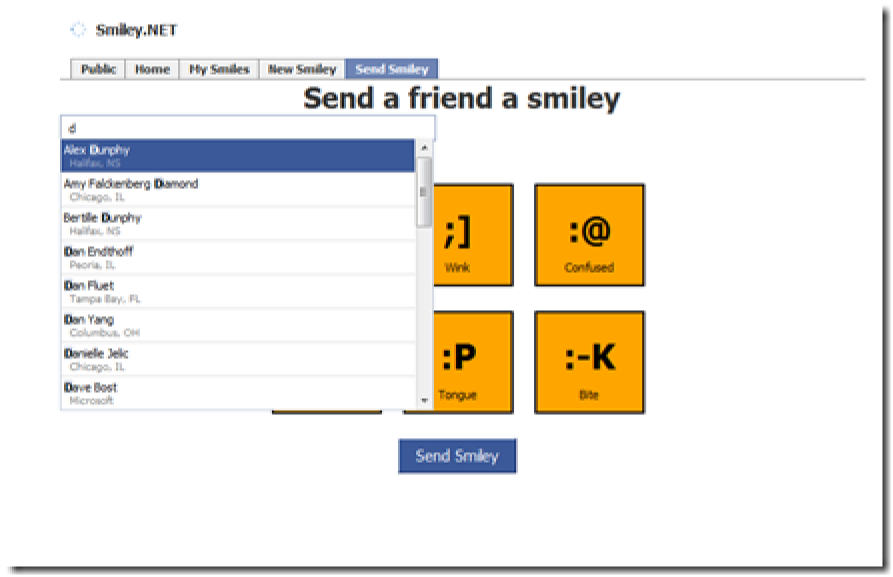
This page is a demostration of how to publish stories to the wall of one of the user's friends. This will use the other publisher that you have specified in your application properties. In this case, our other publisher (that handles publishing to friend's walls is in multiFeedHandler.aspx). To learn more about the multi feed form, read the following wiki page, http://wiki.developers.facebook.com/index.php/MultiFeedStory_form.

In this case, the send smiley page is very simple. Set the action of the multifeed form to the multiFeedHandler and display the Emoticon grid that is not using the final javascript function. In this case, the onclick of the image will just set the style to show which is clicked, and the multi feed form will submit it to the selected friends.

```
protected void Page_Load(object sender, EventArgs e)
        {

multiFeedHandler = string.Format("{0}handlers/multiFeedHandler.aspx", Master.
callback);

grid.Text = BuildEmoticonGrid(getOtherMoods(), Master.callback, Master.suffix
,false);

        }
```

The only other thing to note, is that a multi friend selector is included in this page to allow the user to select the friends they want to target the smiley.

```
<fb:multi-friend-input />
```



Conclusion

Creating an fbml application that really leverages all of the avenues of integration with facebook is not trivial. Hopefully, this sample application and the source code avaiable at codeplex will help people overcome the learning curve. I am going to do my best to support the toolkit better and provide some more articles in the upcoming weeks. In the next article, I want to examine leveraging facebook connect with the toolkit. I imagine that it will lead to some updates to the toolkit.