Introduction to Facebook Platform Development

Excerpted from

Unlocking Facebook
EARLY ACCESS EDITION
A Developer's Guide to the Facebook Platform
Timothy J. Stevens

MEAP Release: May 2008
Softbound print: March 2009 (est.) | 275 pages
ISBN: 1933988878

Facebook is one of the most popular social networking sites on the web. It has over 70 million users registered with thousands and thousands more signing on every day, all linked together in a tangled web of relationships that spans the globe. Many Facebook users hit the site dozens and dozens of times every day, some never leaving it so long as their computer is on. It’s an essential communication tool for many, and a huge opportunity for others.

In May of 2007 Facebook released the so-called “Platform,” a web-based programming interface enabling developers to write little applications that run within users’ profiles or that can be installed on the desktop. Within days hundreds and hundreds of applications were released, filtering through the site as users passing along the ones they thought were interesting, sharing them with friends.
Your goal of course should be to write an application that your users will want to pass on to others. It’s up to you to come up with the right idea, but with the info you learn here you can put that idea into motion. This green paper will give you the knowledge you need to get started, starting with which development environment and client is best for you, then working up getting your first Facebook application running, both on the desktop and on the web.

But, first, there are some key concepts that will need to be understood before diving in.

**Key concepts**

The true beauty of the Platform is the ability to create simple applications from scratch in no time; applications that can be up and online for sharing with users in minutes. But, before you can start coding, you need to understand a few basics.

**Profiles**

Everyone who signs up for Facebook, whether for fun or to re-connect with classmates or for any other reason, starts by creating a profile. A profile is effectively just a collection of information about the user in question, whether it be details about schools, employment, or just books. A full listing of the information included within (and available from) a user’s profile is included in Appendix B, but the best way to see what exactly is there is to take a few minutes and to go through the profile creation process. Even if you already have a profile it can be a worthwhile refresher to walk through the steps of making a new one, or just explore all the tabs in the edit profile section of the site, the first of which is shown in figure 1 below.

![Figure 1 An empty profile in Facebook](image-url)
Application

Of course you wouldn’t be reading this if you didn’t at least have some idea of what a Facebook Application is. But, it’s important to understand the two primary types, and the differences between them.

Web-based

Far and away the most popular type of application is the web-based app, something that the user doesn’t need to install or configure; they just click on a link and there it is. Web-based apps are typically written in PHP or similar web scripting languages and tend to be light and simple. However, within the web category there are different types as well: apps that run within the Facebook site (called the “Canvas”) and those that run outside of it, distinctions that will be covered later.

Desktop-based

Less popular but potentially more functional is the desktop-based application. These apps generally need to be downloaded and installed onto a user’s computer before they can run them. This means you have some greater system resources and capabilities available to you, but that step of installing can be a real deterrent for many.

The Platform (A.K.A. Facebook API)

Facebook’s term for its programmer interface is simply the Facebook Platform. It’s a development framework or API that offers a number of operations that are naturally focused on querying for information about users. It’s functional, but it’s not exactly easy to interact with.

Platform Client (A.K.A. API Wrapper)

You can make direct calls to the API posting data to the Facebook Platform site and providing operation parameters as HTTP POST parameters. But, this requires a lot of manual work to encode the outgoing parameters and to parse the results, typically delivered as XML. A Platform client or wrapper hides all those details through libraries available for a number of different development languages, meaning you can focus on interacting with Facebook -- not parsing blocks of XML.

Registering your application with Facebook

Before you even start coding you should start by registering your application with the site. This is true even if you don’t know exactly what you want your app to do, because you can’t start testing anything without a registered app. Registration gives you both an API Key and a Secret Key, two unique application identifiers that must be provided by your code with every call to the Facebook Platform. Make a call without either of those and you won't get anything back other than an error message.

To register your app, head to Facebook’s developers page, at http://www.facebook.com/developers/. There, click the “Set Up New Application” button to bring up a page like that in figure 2:
The creation process is quick, just requiring a name, but after that there is a whole suite of options to configure starting with the name and extending all the way down to the credits given to the developers. The next section runs through them all.

Selecting a name

Here you type in the name that your users will see when being asked whether they want to authorize your application to access their profile. If you’re just doing a dummy application to get a feel for the Platform name obviously isn’t too important. However, when it comes to your final application, the one you hope will be downloaded by hundreds or thousands, it’s important to choose a good name.

Some apps, like Zoosk, have found success with completely nonsensical names. Most popular applications, though, have names that in some way indicate what they offer. Keep in mind that potential users will see little more than the name of your app before deciding whether to check it out, so it’s useful to include some indicator of what your app offers in the name. Scrabulous is a popular example of an application that features a cute and catchy name but still lets you know what it offers (Scrabble-like gameplay).

Names don’t have to be unique, but to avoid confusion you should take the time to search the Facebook Application Directory (http://www.facebook.com/applications/) and be sure that your app’s name isn’t already taken.

Don’t worry, though; if you can’t come up with a catchy name right away you can always just put in something temporary and change it later!

NOTE

While the Facebook Terms of Service (TOS) isn’t particularly strict when it comes to naming, there is one restriction that some may find somewhat constricting: you’re not
allowed to use the word “Face” anywhere in the name of your application. This unfortunately removes thousands of potentially clever app names from the mix, but also ensures that any time a user sees the word “Face” on the site they know it’s official.

**Application vitals**
The name is perhaps the most important decision to make when registering your application with Facebook, but it’s far from the only one. There’s a full form of information to provide after that, info that is largely optional – but only if you like the defaults. Figure 3 shows these fields.

<table>
<thead>
<tr>
<th><strong>Optional fields</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Developer Contact E-mail</strong></td>
</tr>
<tr>
<td>Limit 100 characters</td>
</tr>
<tr>
<td>For Facebook correspondence only. We will contact you at this address if there are any problems or important updates.</td>
</tr>
<tr>
<td><strong>User Support E-mail</strong></td>
</tr>
<tr>
<td>Limit 100 characters</td>
</tr>
<tr>
<td>Messages sent from users on your application’s help page will be sent to this address.</td>
</tr>
<tr>
<td><strong>Callback URL</strong></td>
</tr>
<tr>
<td>Limit 100 characters</td>
</tr>
<tr>
<td>After logging into Facebook, users are redirected to the callback URL. Include authorization overview for more details.</td>
</tr>
<tr>
<td><strong>Canvas Page URL</strong></td>
</tr>
<tr>
<td><a href="http://apps.facebook.com/">http://apps.facebook.com/</a></td>
</tr>
<tr>
<td>Your application will be removable in the Facebook navigation at this URL. Either as <a href="http://www.domain.com">www.domain.com</a> or a Facebook canvas page with the {app_name}.top. Most things you will want to do will be easier and safer with <a href="http://www.URL">www.URL</a>.</td>
</tr>
<tr>
<td><strong>Application Type</strong></td>
</tr>
<tr>
<td>Websites</td>
</tr>
<tr>
<td>My application uses the mobile platform.</td>
</tr>
<tr>
<td>Checking this will enable some additional RMS functionality for applications.</td>
</tr>
<tr>
<td><strong>IP addresses of Servers Making Requests</strong></td>
</tr>
<tr>
<td>(comma-separated)</td>
</tr>
<tr>
<td>If you supply this information (e.g., 192.120.1.4, 192.120.3), requests from addresses other than these listed will be rejected.</td>
</tr>
<tr>
<td><strong>Can your application be added on Facebook?</strong></td>
</tr>
<tr>
<td>Yes</td>
</tr>
<tr>
<td>Select: Yes if your application can be added to a Facebook account.</td>
</tr>
<tr>
<td><strong>TOS URL</strong></td>
</tr>
<tr>
<td>The URL pointing to your application’s Terms of Service, which the user must accept.</td>
</tr>
<tr>
<td><strong>Developers</strong></td>
</tr>
<tr>
<td>Type: Stevens</td>
</tr>
<tr>
<td>Type a friend’s name.</td>
</tr>
<tr>
<td><strong>Default Frame Canvas Size Option</strong></td>
</tr>
<tr>
<td>Smart size</td>
</tr>
<tr>
<td>See wiki page for more details.</td>
</tr>
</tbody>
</table>

Figure 3 Providing optional application details
While many of the above fields are self-explanatory, most could use some further explanation.

**DEVELOPER E-MAIL**
This is the address that Facebook admins will use to contact you with notifications like changes to the Facebook Platform and confirmation of application registrations. It’s important that this be an e-mail address that you check often, lest you miss some important update vital to the continued health of your app.

**SUPPORT E-MAIL**
This is the e-mail address where questions and complaints from users will be sent. It will not be made public to users, but if your application gets popular it could see quite a bit of traffic. In most cases it’s suitable to use the same address in both fields, but if you find yourself receiving a lot of questions it would be advisable to create a separate address just for contact from users.

**CALLBACK URL**
If your application is web-based, the Callback URL is the address where users will be redirected to after logging in to your application. This URL can be anything you like and can even be empty if your application runs exclusively within the Facebook canvas, but if you provide a URL it has to be an address where you have a page running that can receive the parameters provided by the login call, session information that you will need to maintain for the duration of your user’s session.

**CANVAS PAGE URL**
This is the URL that your web-based application can be accessed from directly through Facebook, starting at the root http://apps.facebook.com/. Ideally the portion of the URL after that root would simply be your application name but, though the name does not have to be unique across all of Facebook, this URL must be.

Additionally here you can specify whether your web-based app will use FBML or just straight HTML in an IFrame. FBML is a limited subset of HTML tags with a variety of Facebook-specific tags added, tags that make certain commonly-used profile attributes very easily accessible.

**APPLICATION TYPE**
This is where you indicate whether your application will be accessible exclusively through a web browser, or whether it’s downloaded and installed onto a user’s desktop computer. Unfortunately your application can’t be both.

**MOBILE INTEGRATION**
This checkbox indicates that your application is intended for use on or will interact with mobile devices, specifically cell phones. That includes the ability to display content specifically when a user is viewing their profile on a mobile device and also enables your application to send and receive SMS text messages.
IP ADDRESSES
If your application is web-based and is not running within the Facebook canvas, you can specify one or more IP addresses that represent the server (or servers) hosting your application. If Facebook receives any request from any address other than those listed using the identifiers associated with your application, that request will be rejected.

ADDING APP TO FACEBOOK
This set of radio buttons indicates whether your application can be “added,” which is Facebook terminology for installing a web-based app into your profile. Select “No” if you’re writing a web-based app that doesn’t run within the Facebook canvas, or if you’re writing a desktop app. Select “Yes” if you’re working on an app that will appear within the user’s profile, in which case a slew of additional options appears.

TOS URL
Facebook itself has a lengthy terms of service (http://developers.facebook.com/terms.php) which you must accept before you can post your application. If you’d like to add your own terms which your users will be asked to accept, host it online and provide the URL here. Facebook will add a prompt for users to accept it using your app.

DEVELOPERS
Here you list out who wrote the application. You’re adding by default, but you can search among your list of friends and add them as well.

IFRAME SIZE OPTION
This option again applies to apps running within the Facebook canvas. This option indicates whether the Iframe housing your app’s display automatically sizes itself (“smart fsize”) or whether you explicitly control its size (“resizable”).

Choosing a Platform Client
If you like, you can write either a web-based or a desktop application that makes direct calls to the Facebook Platform, providing operation call parameters to HTTP POSTs and parsing the resulting data from blocks of XML. But, the overhead of doing all that can be a significant hindrance for someone looking to get up to speed quickly and dig into the Platform.

For this reason many developers turn to one of the many clients available. Platform clients trivialize many of the complexities of interacting with such web APIs and let you get straight to the job at hand: writing your application. Wrappers are available for nearly every popular programming language out there, including:

- PHP
- Java
- .Net
- JavaScript
- Ruby
The top three languages in the list above (or multiple languages in the case of .Net) are covered by libraries either written by Facebook devs or licensed by them for official distribution to developers. This means there’s some guarantee for functionality and consistency with the Platform itself, making them a good choice. That said, the many other wrappers out there are quite good and reliable as well, so it’s really down to you choosing the language that you prefer.

For web-based apps here we’ll be working with the PHP client library, as it’s the most popular. On the desktop side, for applications that must be installed, we’ll work with C# and the .Net library, since Visual Studio’s excellent design tools make throwing together a windows app quick and easy. Also, Facebook’s partnership with Microsoft means the .Net library is the one that’s likely to get the most attention in the future.

However, if you’re so inclined, the PHP library can also be used for writing desktop applications and likewise the .Net library can also be used for created web-based applications, so again, choose the language you prefer.

First example – authentication and saying “Hello”
Authentication is the one step that every Facebook application must complete, and in many ways it is the most important step. In many cases it’s also the most complicated thing your application will do. Authenticating a user through when making direct calls to the Platform requires multiple steps and the calculation of a secret key. But, the use of a Platform Client makes the process much, much easier.

The first example applications we’ll show what authentication looks like when logging in the user for a desktop-based application. Then we’ll do the same thing for a web-based one. As you’ll soon see, the process is subtly different between the two.

Desktop example
First, start by designing a simple form in Visual Studio, like that shown in figure 4.
You'll need to open the source for your form and add this using declaration to get access to the Facebook classes you'll need:

```csharp
using Facebook.Components;
```

This is a reference to the core namespace containing the classes and utilities provided by the .Net Facebook client. This namespace is included in Facebook.dll, part of the client download. You'll need to add a reference to that as well.

Next you'll need an instance of FacebookService to handle the heavy lifting for you. Make it global so that you can access it from the various methods that you'll be creating, something like the code in listing 1.

### Listing 1 Start for Chapter02 desktop example

```csharp
namespace Chapter02
{
    public partial class SecurityLogin : Form
    {
        private FacebookService service = null;
        public SecurityLogin()
```
Before you can use it you need to instantiate that `FacebookService` instance, of course, which can be done in the `Form_Load` method. For this example we called the form `SecurityLogin`, as listing 2:

### Listing 2 Form initialization

```
private void SecurityLogin_Load(object sender, EventArgs e)
{
    service = new FacebookService();
}
```

That’s simple enough, what with the complete lack of parameters. Naturally, though, you’ll need to actually need to populate the thing with your API and secret keys before you can start making any calls. That’s what happens when the button labeled “Login” is clicked, as shown in listing 3:

### Listing 3 Clicking the login button

```
private void button1_Click(object sender, EventArgs e)
{
    service.ApplicationKey = textBoxAPIKey.Text; #1
    service.Secret = textBoxSecretKey.Text; #2
    try
    {
        service.ConnectToFacebook(); #3
        labelLoggedIn.Text = "True, Hello World!";
    }
    catch (Facebook.Exceptions.FacebookException fbe) #4
    {
        labelLoggedIn.Text = fbe.Message;
    }
}
```

#1 Here the API key from the UI is applied to the service.
#2 Here the secret key is plugged in.
#3 The heavy lifting is applied here.
#4 A little minimalist error handling.

That’s really all there is to it. You might think you need to somehow handle receiving the user’s username and password and providing that to Facebook, but actually the login process is still handled through the Facebook webpage. Even if you’re running a desktop application, the user must be directed to log in through the site, something that’s handled automatically through that `ConnectToFacebook` call above, which we’ll see in action in just a moment.

The last thing of course is giving the user the option to log off, which is just as easy as logging them in as in listing 4:

### Listing 4 Clicking the logoff button

```
private void button2_Click(object sender, EventArgs e)
{
    service.LogOff();
    labelLoggedIn.Text = "Not Logged In";
```


The running app looks like figure 5:

![Image of Security Demo](image)

**Figure 5** The running application

Simple, as you can see. But, populate the two keys and click "Login" and a lightweight browser instance is brought up automatically showing something like figure 6:
Figure 6. The Facebook login page for a desktop app

Once the user logs in and authorizes the application, the browser window closes itself and the user is directed back to your application, as shown in figure 7.
Authentication for web-based apps

Just like on the desktop with the .Net C# API used above, things are much, much simpler when writing web apps using the PHP client provided by Facebook than when making direct API calls. However there’s a major catch here: almost zero documentation. So, we’ll help you through the high points with an application that logs the user in then displays a quick welcome, indicating that authentication was successful.

Start by installing the PHP client somewhere on your web server, by simply extracting the PHP client to a directory of your choosing, typically called "client". From there create a separate dir to hold your test application and start coding, starting with an include referencing "facebook.php" in the client dir you created above. That file includes the functionality you’ll need to sign a user in.

The code in listing 5 will do just that:

Listing 5 A simple example using the PHP API wrapper

```php
<?php
include_once '../client/facebook.php'; #1

$api_key = '123ABC';
$secret_key = '789XYZ';

$facebook = new Facebook($api_key, $secret_key);
$facebook->require_frame();
$user = $facebook->require_login();
?>

<div style="padding: 10px;">
  <h2>Hello <fb:name firstnameonly="true" uid="<?=$user?>" useyou="false"/></h2>
  <br/>
</div>

#1 This include provides access to all that the PHP library has to offer.

Yes, that’s really all there is to it, but there are some details we should go over with the individual lines here. Let’s start with the instantiation of the Facebook instance:

```php
$facebook = new Facebook($api_key, $secret_key);
```

This is what you will use to access the functionality provided by Facebook. We’re not doing much with it here, except for specifying some options for the way this little app will behave.

```php
$facebook->require_frame();
```

The above line is the equivalent of passing the canvas flag to the login.php page. As described above, this forces the application to be displayed within the Facebook canvas.

```php
$user = $facebook->require_login();
```

This line of code simply indicates that the user has to be actively logged in to use this code, meaning this app can’t run as part of some autonomous background process. It doesn’t, however, require them to explicitly login again if they already have the Facebook page open.
Additionally, the `require_login()` call gets us the instance of a user that we can use to figure out who is logged in. Since our primary focus is just getting authenticated, all we'll do is use a little bit of FBML to display the first name of the user, with the following:

```html
<fb:name firstnameonly="true" uid="<?=$user?>" useyou="false"/>
```

And that’s it. Running the above code gets you a resulting page something like figure 8:

![Figure 8](image.png)

**Figure 8** The resulting display from the sample web authentication example app

You may have noticed that in this app, unlike the desktop app above, we didn’t provide code to enable the user to explicitly log the user out. There are two reasons for that. The first is that, again, the PHP library doesn’t directly support that functionality; there’s no easy `logoff()` function that can be called to kill the user’s session. That’s probably because if you’re running your app within the confines of the Facebook menus (as most do) there’s a logout button already provided for you right there at the top, as you can see in figure 8 above.

Secondly, though, and more importantly, individual apps running within the canvas shouldn’t really take control of the user’s session in that way. A user running your app inside of Facebook would intuitively expect it to simply inherit the behaviors of the Facebook page itself. In other words: if they want to log out they should be directed to click Facebook’s “logout” button at the top, not yours.

But, if you’re really inclined to explicitly log the user out from within your application, you can just redirect the user to this URL: `http://www.facebook.com/logout.php?confirm=1`.

**Summary**

Now you know the basics of creating an app in Facebook. Starting with a good name and an appropriate configuration for your application is vitally important, and properly authenticating your users through the Facebook site is of course a necessity if you want your application to do anything.

What exactly anything entails is, of course, entirely up to you, but once your user is logged in and your app is up and running all that’s holding you back now is your own creativity.