Tech Note

embarcadero

Creating Your First Facebook Application

José León

November 2010

Americas Headquarters
100 California Street, 12th Floor
San Francisco, California 94111

EMEA Headquarters
York House
18 York Road
Maidenhead, Berkshire
SL6 1SF, United Kingdom

Asia-Pacific Headquarters
L7. 313 La Trobe Street
Melbourne VIC 3000
Australia

Table of Contents

INTRODUCTION

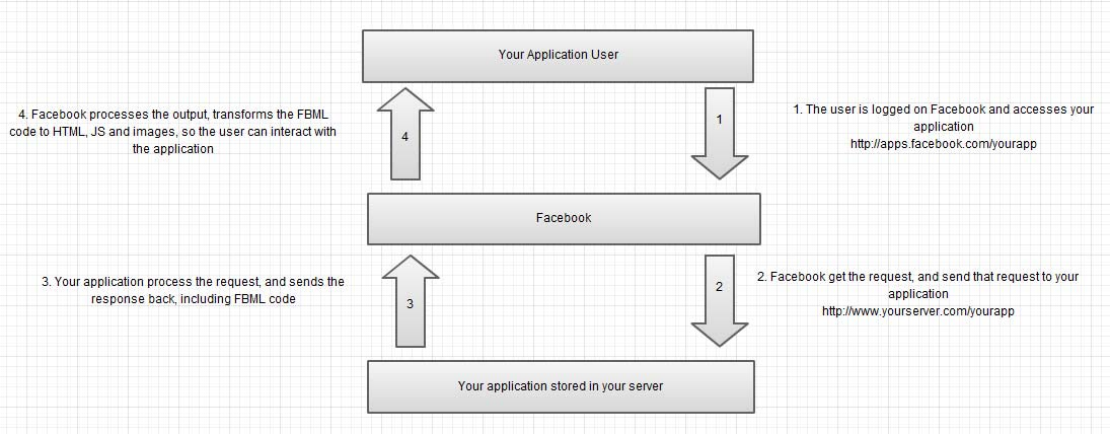
Facebook, the leading social networking site, is increasing being used as a platform for applications of all types - including games, marketing promotions, business applications and utilities. Building applications to run on Facebook opens opportunities for developers to reach an audience of more than 500 million users. According to Facebook, every month more than 70% of Facebook users engage with platform applications. If you haven't tried developing applications for Facebook yet, the process can be easier than you think.

The goal of this technical paper is to show how to develop a simple Facebook application using Embarcadero RadPHP™ XE. We will create a utility application that tracks lending items to Facebook friends. The paper covers everything from the setup of the application in the Facebook network to creating the application and gathering information from users and friends.

BASIC APPLICATION SETUP

First, you have to understand how the Facebook application model works. Facebook acts as a proxy between the users and your application, but your application should be stored on your own server.

Here it's a very simplified flow diagram on how all this works:



CREATING THE RPCL APPLICATION

Using **File | New | Other…** menu option, on the **RadPHP XE Projects** category, there is an item called **Facebook Application**. Select it and click OK, and the IDE will create the basic skeleton for a Facebook Application, which is made up of a regular Form and a DataModule containing a FBApplication component.

The FBApplication component allows you to connect with Facebook and use its API. It also provides the ApplicationID and ApplicationSecret properties to uniquely identify your application into the Facebook network.

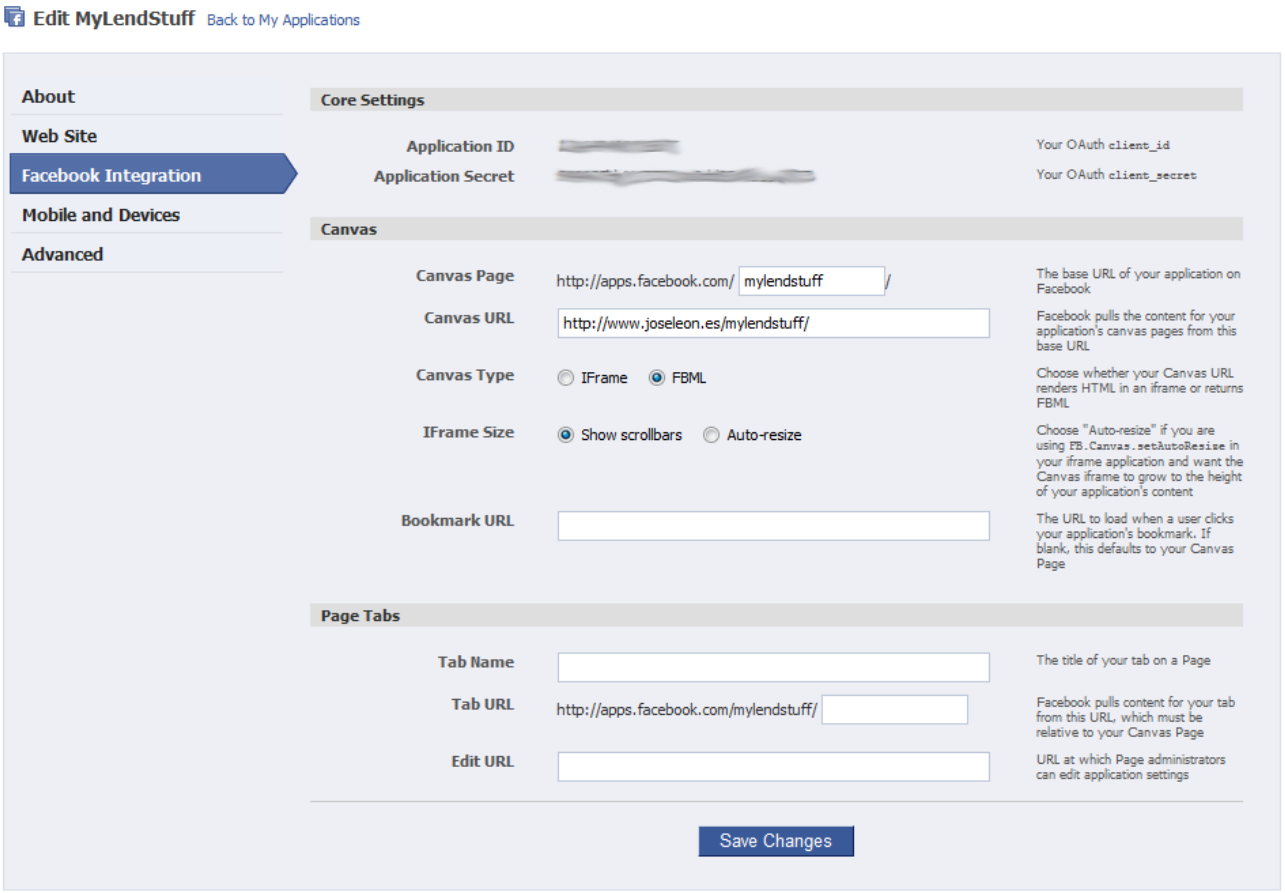
SETTING UP THE APPLICATION IN THE FACEBOOK NETWORK

To allow Facebook users to use your application, you need to add it to the Facebook network, this can be done by pointing your browser to:

http://www.facebook.com/developers

You will be asked for an Application Name and a captcha verification, your application will be created, and you will be forwarded to editing the application parameters. There are five categories for settings (About, Web Site, Facebook Integration, Mobile and Devices, Advanced), but for this sample, we are interested only on the Facebook Integration settings. From top to bottom, the interesting settings are:

- Application ID
- Application Secret
- Canvas Page
- Canvas URL
- Canvas Type
- IFrame Size



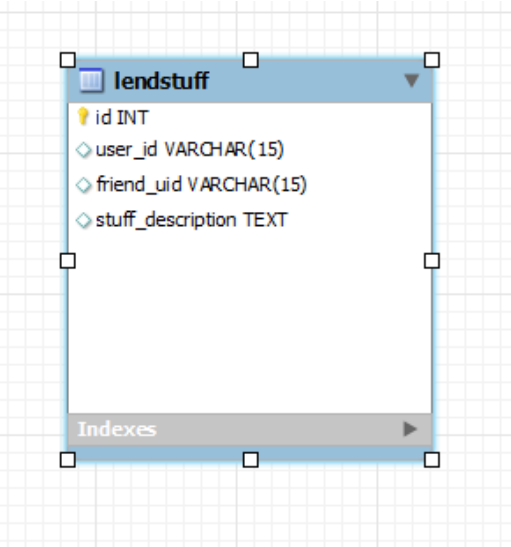
GETTING YOUR SERVER READY TO EXECUTE THE APPLICATION

First, you have to prepare your server to execute an RPCL application. While there is a Deployment Wizard on the IDE that helps you with the process, my preference is to upload the full RPCL library, so you don't have to worry about missing any files.

The easiest way to do this is to compress the RPCL folder in a .tar.gz file, and upload it to the server. After uploading, access the server using SSH and decompress this file. Finally, delete the .tar.gz file and you are done.

DATABASE DESIGN

For this sample, we need a simple table in a database. We want to store the ID of the user lending out items, the IDs of the user receiving items, and a description of what the items ("stuff") are.



And here is the SQL to create such structure:

```
1. DROP TABLE IF EXISTS `lendstuff`;
2. CREATE TABLE IF NOT EXISTS `lendstuff` (
3.   `id` INT(11) NOT NULL AUTO_INCREMENT,
4.   `user_id` VARCHAR(15) NOT NULL,
5.   `friend_uid` VARCHAR(15) DEFAULT NULL,
6.   `stuff_description` text,
7.   PRIMARY KEY  (`id`)
8. ) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=10 ;
```

SETTING UP THE DATAMODULE

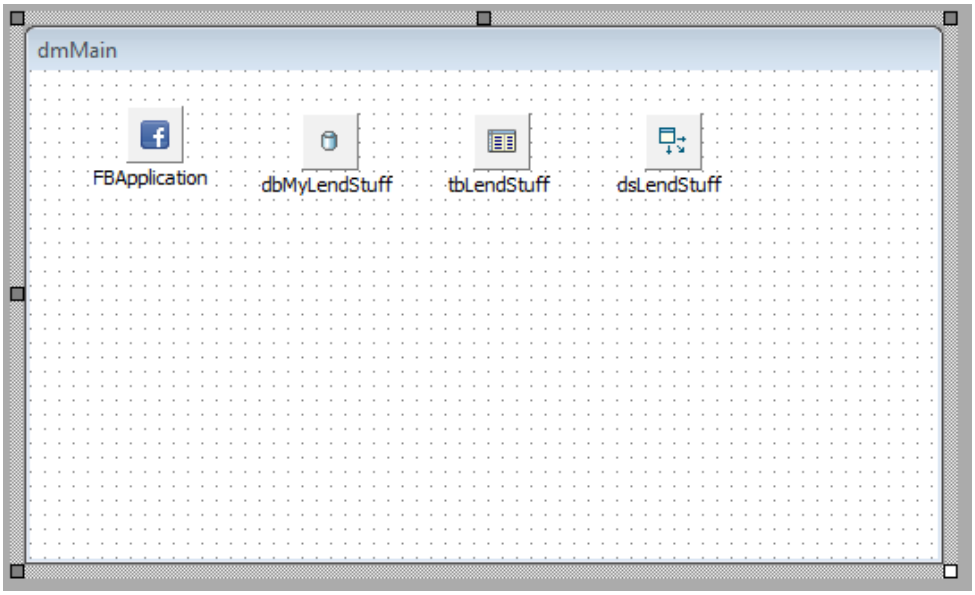
To access the database table, we need to use three components: Database, Table and Datasource. Drop these components into your DataModule, which we will call dmMain.

The Database component provides the connectivity with the database itself. You need to setup the properties DatabaseName, DriverName, Host, UserName and UserPassword with the correct values.

The Table component should be linked to the Database component through the Database property, then set the TableName property to the name of the table we want to access, in this case, "lendstuff".

The Datasource component provides connectivity to data-aware controls to any dataset. To do so, set the DataSet property of the component to the table we want to access.

This is how your DataModule should look:



REQUIRING THE USER TO BE LOGGED IN

In this application, we need the ID of the user that is accessing the application, and to do that, we need to ensure that the user is logged in. To do this, we use the requireLoggedUser() method of the FBApplication component.

We need to filter the Table component so it only retrieves records for the current logged users. In the OnBeforeOpen event of the Table, we can require the user to be logged in, in order to get the User ID.

```
1. function tbLendStuffBeforeOpen($sender, $params)
2. {
3.   $this->FBApplication->requireLoggedUser();
4.   $this->_loggeduserid=$this->FBApplication->UserID();
5.   $this->tbLendStuff->Filter='user_id='.$this->_loggeduserid;
6. }
```

We are storing the User ID in a DataModule property, so we can reuse it later if required. This is the code for such property:

```
1. protected $_loggeduserid='';
2.
3. function readLoggedUserID() { return $this->_loggeduserid; }
4. function writeLoggedUserID($value) { $this->_loggeduserid=$value; }
5. function defaultLoggedUserID() { return ''; }
```

DEVELOPING THE INDEX PAGE

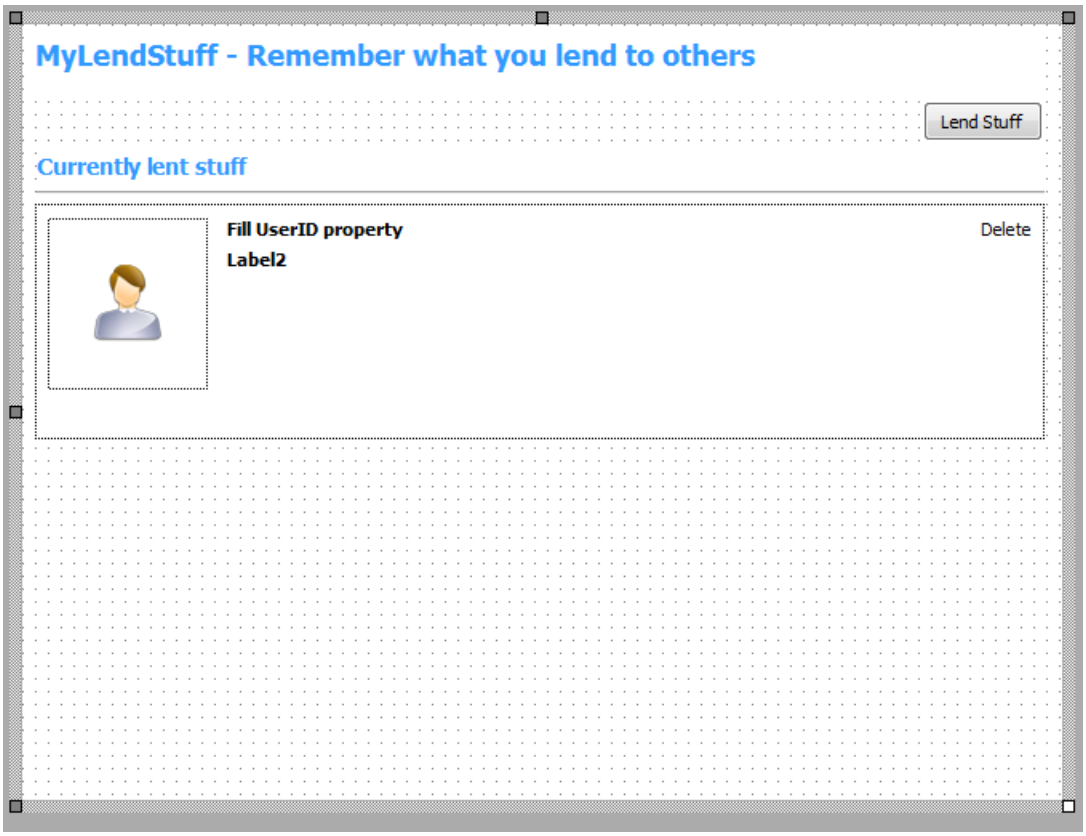
The main page is going to show what stuff have you lent and to whom.

First, let's place a Label component at the top, specifying the purpose of the application (in this case, MyLendStuff – Remember what you lend to others). After that, let's place another Label with the current page contents (Currently lent stuff), to show a list of the current stuff you have lent.

To show the stuff, we need to iterate through a database table and show the profile image for the friend who borrowed the stuff from us, the name for it, and the stuff description.

For that, we are going to use a DBRepeater component. This component is attached to a Datasource and iterates through it, generating, on each iteration, all components it contains.

This is how your index page should look:



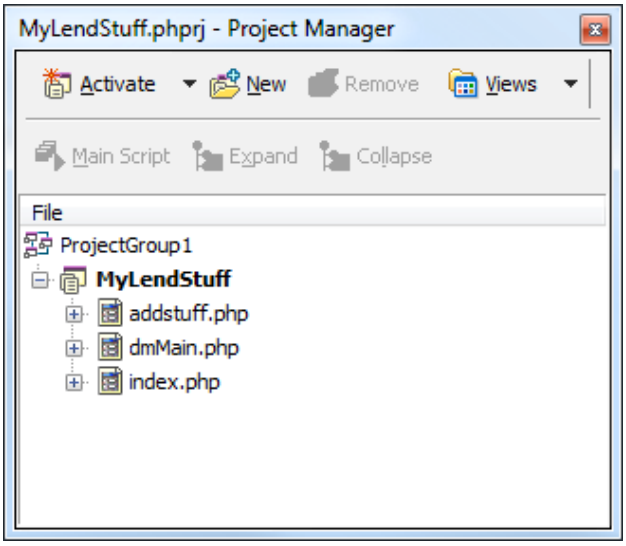
As you can see at the top right, there is a Button called "Lend Stuff", that is the action is going to allow the user to enter a new record on the database. To do that, simply double-

click on the Button and use the RTL function redirect() to point the browser to another page, in this case, addstuff.php:

```
1.      function btnLendStuffClick($sender, $params)
2.      {
3.        redirect('addstuff.php');
4.      }
```

ADDING NEW STUFF

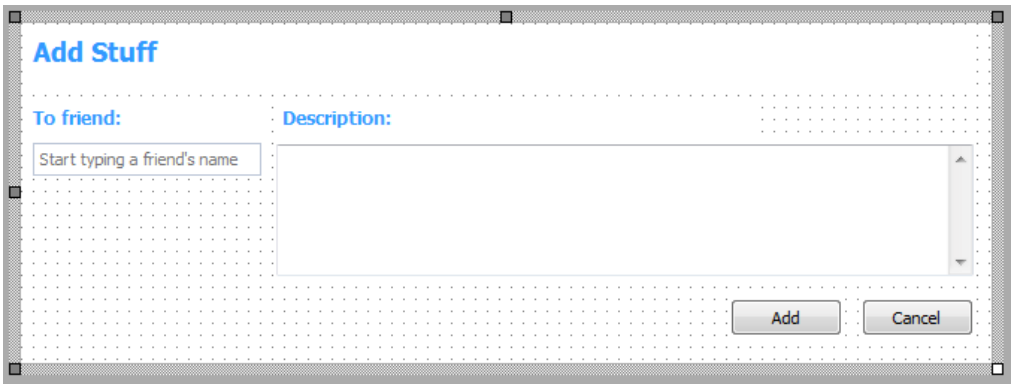
To add new stuff, we need a new page. Let's use **File | New | Form – PHP** to create one, and name it "addstuff.php". At this moment, all the files we need for this project are created, and this is how your Project Manager should look:



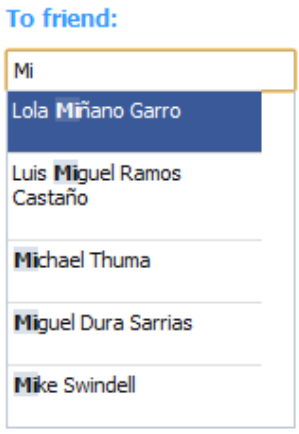
Following the general design we used on the index page, place a top label with an "Add Stuff" caption, then add a FBFriendSelector component and a Memo component below that. The FBFriendSelector is like a regular Edit control, but it allows the user to choose a friend just by typing some characters. We are going to use this control to let the user select to which friend to lend the stuff. The stuff to lend is going to be entered in the Memo component.

Also, two Buttons at the bottom: one to cancel the operation and return back to the index page, and one to actually add the stuff to the database.

This is how your form should look like:



If you run your application on Facebook, this is how the FBFriendSelector component works:



Now, let's make the page work by implementing the code for the Add button:

```
1. function btnAddClick($sender, $params)
2. {
3.   global $dmMain;
4.
5.   $dmMain->addLendStuff($this->FBFriendSelector1->SelectedUserID,
   $this->Memo1->Text);
6.
7.   redirect('index.php');
8. }
```

We are using a method of the DataModule called addLendStuff(). It takes, as first argument, the ID of the friend we want to lend the stuff to, and, as second argument, the stuff description.

To access that method, we need to add the global declaration to access the datamodule object.

After adding the stuff, redirect to the index.php page.

But wait, we have not create the addLendStuff() method on the datamodule yet. Here is the code for it:

```
1. function addLendStuff($touserid, $stuff)
2. {
3.   $this->tbLendStuff->append();
4.   $this->tbLendStuff->user_id=$this->_loggeduserid;
5.   $this->tbLendStuff->friend_uid=$touserid;
6.   $this->tbLendStuff->stuff_description=$stuff;
7.   $this->tbLendStuff->post();
8. }
```

It simply appends a new record to the table, set the fields with the new values, and post it to the server.

DELETING STUFF

To delete stuff, we need to add that option in each detail of the DBRepeater control. To do that, add a Label control inside the repeater control, change the caption to "Delete" and generate the OnBeforeShow event. Now type in this code:

```
1. function lbDeleteBeforeShow($sender, $params)
2. {
3.   $this->lbDelete->Link='index.php?action=delete&id='.
4.   $this->DBRepeater1->DataSource->DataSet->id;
5. }
```

This code renders the Delete label as a link, with an URL specifying the action to be performed and the ID of the record to be deleted.

To actually execute the delete action, on the OnBeforeShow event of the index page, we should check if there is an action parameter to be executed and perform the operation.

Generate the OnBeforeShow event of the index page and type this code:

```
1.     function Page32BeforeShow($sender, $params)
2.     {
3.       global $input;
4.       global $dmMain;
5.
6.       $action=$input->action;
7.       if (is_object($action))
8.       {
9.         $action=$action->asString();
10.         if ($action=='delete')
11.         {
12.           $dmMain->deleteLendStuff();
13.           redirect('index.php');
14.         }
15.       }
16.     }
```

This code is executed before the page is shown. We can check for an action parameter on input, and if that action is 'delete', then execute the deleteLendStuff() method of the dmMain datamodule.

Let's check the source code of that method:

```
1. function deleteLendStuff()
2. {
3.   $this->tbLendStuff->delete();
4. }
```

This is just a call to the delete() method of the Table component, so how this component knows the record to be delete?

Before opening the table, we have navigated to the correct record using the 'id' parameter on the request with the following code:

```
1.   function tbLendStuffBeforeOpen($sender, $params)
2.   {
3.      $this->FBApplication->requireLoggedUser();
4.      $this->_loggeduserid=$this->FBApplication->UserID();
5.      $this->tbLendStuff->Filter='user_id='.$this->_loggeduserid;
6.
7.      global $input;
8.      $id=$input->id;
9.      if (is_object($id))
10.     {
11.       $id=$id->asString();
12.       if ($id!='') $this->tbLendStuff->Filter.=' and id='.$id;
13.     }
14.   }
```

If there is an 'id' parameter, we are setting the Filter property to the right record, so a call to delete() will delete it.

CONCLUSION

In this paper, we've looked at how to create a Facebook application using RadPHP. We saw how to create an application and database, adding different functionality including user logins and lookups, and deploying the application on the Facebook network. Hopefully this example application shows you how the visual development environment in RadPHP can save you time and effort as you create PHP and Facebook applications. That's the idea behind "Rapid Application Development", the "Rad" in RadPHP.

ABOUT RADPHP XE

Embarcadero® RadPHP™ XE revolutionizes PHP web development with a completely integrated, rapid visual development approach and component framework. RadPHP XE provides a powerful editor, debugger, visual development tools and connectivity with leading databases. The integrated reusable class library includes components for everything from UI design to building applications for Facebook. Visit the Embarcadero web site to learn more about RadPHP XE and download a free trial.

Embarcadero Technologies, Inc. is the leading provider of software tools that empower application developers and data management professionals to design, build, and run applications and databases more efficiently in heterogeneous IT environments. Over 90 of the Fortune 100 and an active community of more than three million users worldwide rely on Embarcadero's award-winning products to optimize costs, streamline compliance, and accelerate development and innovation. Founded in 1993, Embarcadero is headquartered in San Francisco with offices located around the world. Embarcadero is online at www.embarcadero.com.