

Evaluation: Facebook API



October 11, 2007

Author(s): Wolfram Koska

Copyright: © 2007 TNG Technology Consulting. All rights reserved, including any reprinting, copying, use or communication of the contents of this document or parts thereof. No part of this document may be reproduced, communicated to third parties, stored in any kind of electronic storage or retrieval system or broadcast without the express written permission of TNG. We reserve the right to update or modify the contents of this document at any time without prior notice.

Disclaimer: The information contained in this document has been obtained from sources deemed reliable and has been professionally screened and analyzed. However, TNG does not warrant that the information is correct, complete or adequate. TNG disclaims any liability for mistakes, omissions or bad judgements regarding the information or its interpretation. The responsibility for the selection, perception and interpretation of the information in this document is completely the readers'. The opinions expressed in this document can change at any time without prior notice.

Contact: TNG Technology Consulting GmbH
Betastraße 13a
85774 Unterföhring
Federal Republic of Germany

Web: <http://www.tngtech.com>
E-Mail: info@tngtech.com
Phone: +49-89-2158996-0
Fax: +49-89-2158996-9

Table of Contents

1 Management Summary.....	4
2 About Facebook.....	5
2.1 Terms of Use, Privacy Policy, etc.....	5
2.2 Developer Terms of Use.....	6
3 Facebook Application.....	7
3.1 Application Types.....	7
3.2 Application Properties.....	7
3.3 An „Internal“ Web Application.....	8
3.3.1 The Canvas.....	8
3.3.2 The Profile.....	9
3.4 External Web and Desktop Applications.....	10
4 Facebook API.....	11
4.1 Documentation.....	11
4.2 API.....	11
4.3 FQL.....	12
4.4 API client libraries.....	12
4.5 FBML.....	13
4.6 JavaScript.....	14
5 Personal Experience and Opinion.....	16
5.1 Usability.....	16
5.2 Internationalization.....	16
5.3 Security.....	16
5.4 Conclusion.....	17
6 Appendix.....	18
6.1 Database Layout of the Example Application.....	18
6.2 appinclude.php.....	18
6.3 appinclude.php.....	19

1 Management Summary

This document contains an evaluation of the Facebook programming API, done by Wolfram Koska from Sept. 19 2007 – Sept. 25 2007. In the process of this evaluation, a small Facebook internal web application was developed. The application can be used to invite other friends to use the application, and to give „credits“ (imaginary points or currency) to friends.

The Facebook platform has a high number of active users from which an application could potentially benefit. The goal of the evaluation was to answer the question if the API can be used to develop high quality applications which integrate into the Facebook platform.

To make the reader familiar with the Facebook platform, it will be described in the next chapter. In the following chapter, a description of the Facebook API and it's documentation is given. Finally, reasons for the conclusion will be listed: The API makes a solid impression, it is comprehensive and there exist many functions assisting the developer in creating a Facebook application. The only disadvantage is the lacking secure http support. Thus, it is safe to recommend, unless secure http is a requirement.

2 About Facebook

When signing up to Facebook, not much information has to be supplied: Name, password, educational status, valid email address, and date of birth. The terms of use and privacy policy have to be agreed upon. Later, a user can enter detailed information about all his interests, previous schools / companies, etc.

Facebook is a social network site: A user can search by name or email for a friend and suggest friendship. When the other user agrees to the friendship, both will be able to access the profile page of the other one and all of the information that he provided about himself. When a user has not agreed on a friendship, it is not possible to access more information than his or her name, the networks he or she is part of and a photo.

A network is a group of people, not necessarily friends. There exist networks for countries, universities, companies etc. New networks can be created by the users. A user can join any number of networks.

Users can upload photos, send each other messages if they are friends, send friendship requests, or „poke“ each other (this might be done for example if the user thinks that he has found a friend, but is not sure). Additionally, friends can send each other application invitations, that means, a request to install a application, so that both are using this application.

Facebook applications are programs using the Facebook API and can be added and removed any time by a user for his or her account. The most common application type is an internal web application, which is accessed directly on the Facebook web site. See chapter 3.1 for more information about application types.

2.1 Terms of Use, Privacy Policy, etc.

A few comments on the terms of use and privacy policy follow (please note that the author of this evaluation is no expert in law).

The terms of use are typical for such sites, requiring the user to provide accurate data, and requiring him to agree that all information that he posts to Facebook does not harm the rights of anyone else and complies to applicable law. The user must not copy or redistribute any of the content available.

Also, the terms of use limit the usage of the site to personal, non-commercial use (except for advertising programs offered by Facebook).

Facebook dissociates itself from any third party applications.

The complete terms of use can be accessed here: <https://register.facebook.com/terms.php>

There is also a privacy policy, providing details about what information Facebook collects and what it does (or does not) with this information: <https://register.facebook.com/policy.php>

When installing 3rd party applications, the user agrees to the Facebook Application Terms of Use, which can be found here: http://developers.facebook.com/user_terms.php

For a normal user, there is very much to read: it is probable that only a very small percentage of the users have read all of the required terms and policies.

2.2 Developer Terms of Use

When a user wants to contribute to Facebook with add ins, he has to read a lot of terms. A list can be found here: <http://developers.facebook.com/terms.php>

The author of this evaluation did not read everything. Yet some important things that were encountered follow:

- Facebook limits the information any application is allowed to store to a number of Ids (the user Id, primary network Id, event Id, etc). Note that this is of course a limit to the information which the application gets from Facebook, not to the information the application collects from its users.
- Facebook requires the applications to comply to its standards, which basically require the application to be legal in every way. There are also a number of presentational (e.g., pop ups are not allowed for Facebook applications) and marketing requirements, if one should opt to advertise his application.

There is a huge amount of terms to read, which should be read before development of a commercial Facebook application is started, to ensure that it will comply with all the requirements Facebook provides.

All in all, it looks like the terms' general goal is to keep the Facebook environment clean and secure and keep a professional look.

3 Facebook Application

3.1 Application Types

The Facebook API can be used from external web applications, external desktop applications and internal web applications. Every application using the API has to authenticate itself and the user before information is exchanged between Facebook and the application.

1. External web applications: The user is redirected to Facebook, where he will authenticate himself using his user name and password. From there, he is sent back to the external web application with an authentication token.
2. External desktop applications: The application has to request an authentication token from Facebook and then send the user to a login page in a browser (using the authentication token). As soon as the user successfully logged in, he can close the browser and work with the application.
3. Internal web applications: The user is authenticated on the Facebook web site. The application is displayed from within the Facebook site in the „canvas“ (see section 3.3.1).

3.2 Application Properties

Every user can create an application by using the „Developer“ application. When he does this, he will get an application key and a secret key. These are 32-byte IDs used to authenticate the application to the Facebook server and ensure that the parameters are passed correctly (the parameters and the results of remote calls will always be signed to protect from malicious attacks).

The developer chooses a support email address, callback URL, valid IP addresses of the application's servers, etc. More developers can be added for an application, choosing from other facebook users. A number of additional parameters can be set to configure the application and it's behaviour, for example the application icon, the URL to be called after a user has added the application, and much more.

3.3 An „Internal“ Web Application

This evaluation of the Facebook API was done by creating an internal web application. This means that the application runs completely in the context of the Facebook website. See Illustration 1: Internal Web Application for an example (the green and yellow overlays are not part of the original page). The green part on the left is the „Left Nav“, where a link to the application can be placed by



Illustration 1: Internal Web Application

the user. The yellow part is the „canvas“, where the main application contents are shown. This can be done either as the facebook-specific markup language „FBML“ (see section 4.5) or as an if-rame.

The application itself (and its database) does not run on the Facebook servers (as the name „internal“ might suggest). An application developer will deploy his application on an own web server, and Facebook knows about this location from the „application callback URL“ setting.

3.3.1 The Canvas

In the canvas, the application can dynamically create it's content. For this, the parameters of the web page are sent to the callback URL of the application, as in the following example:

http://apps.facebook.com/canvas_url/userpage.php?parameter1=value1

Here, *canvas_url* is the canvas URL of the application, and *userpage.php?parameter1=value1* will be appended to the callback URL of the application (which could be, for example, <http://www.-myapplication.com/>). Facebook will then call this URL, appending all needed Facebook parameters such as the user id, the application's api key, session key and session expiration time (if a session is already established), etc, signed with the application's secret key. The application can then validate and process the request. The result is a HTTP response, which is preprocessed (see the section about FBML) and then displayed in the canvas part of the application.

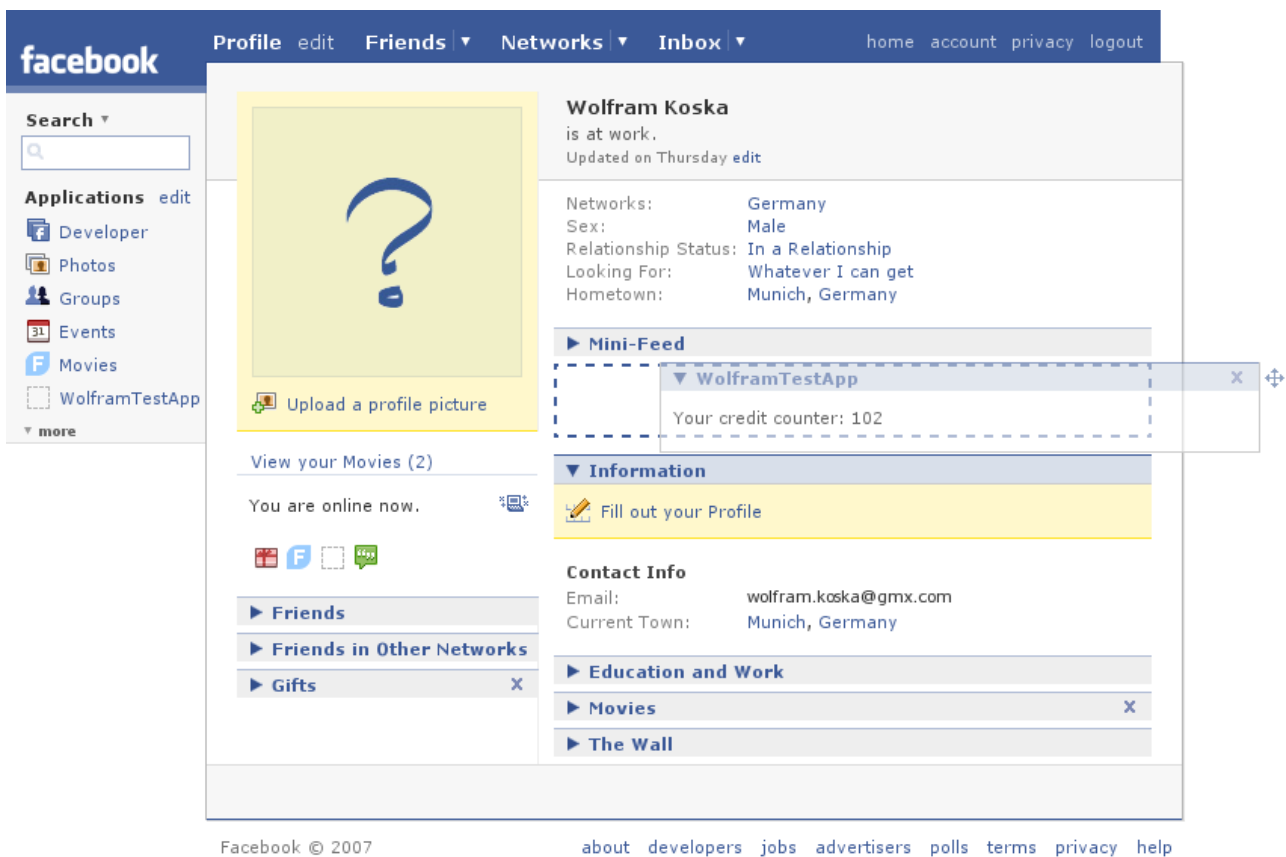


Illustration 2: The Profile Page

3.3.2 The Profile

Another part of Facebook where an application can manifest itself (besides the canvas) is the profile page.

The profile page of a user can be accessed by his friends, too. It consists of a small left and a larger right column, in which small, moveable windows called „profile boxes“ reside. Usually, for each application installed by the user, there is one profile box in the profile page (but the user can disable the application to display anything in the profile page, and an application does not have to provide a profile box).

Illustration 2: The Profile Page shows an example profile page. Most profile boxes are minimized. The „WolframTestApp“ Profile box is in the process of being moved under the „Mini-Feed“ profile box.

The information that is displayed in the profile box is static: It is generated once, cached and displayed until the cached content is renewed. There are, though, methods to be able to dynamically respond to user input even in the profile boxes, called „MockAJAX“ in the Facebook documentati-

on: Special FBML tags enable the profile box to send information to the application and replace the contents of a <div>-tag with the returned result (without reloading the whole page, thus the AJAX in the name).

3.4 External Web and Desktop Applications

For this evaluation, the no external web or desktop application was created. The differences to the internal web application lie mainly in the authentication process and the presentation: They are not displayed inside the Facebook web page context. Their development effort might be higher because of the lacking FBML and Facebook JavaScript objects, which prove to be quite useful to the developer.

4 Facebook API

4.1 Documentation

There are two places to look for documentation: The Facebook documentation site¹ and the Facebook documentation wiki². For many parts, the information on the web site and the wiki is identical, but the documentation on the wiki is more complete and up to date. For example, the FBML documentation can only be found in the wiki (although it is linked from the web page).

The documentation of the Facebook API is not excessive, but for what was needed to create the small application for this evaluation, complete. Looking through the documentation, there are images lacking here and there and maybe some details, but overall, the documentation, especially the wiki, makes a good impression. A large number of code examples complete the appearance, although sometimes they could be commented better. A running example application including at least part of those examples would be nice to have.

4.2 API

The Facebook API provides calls for authentication, to get information about users, user groups, friends, notifications, events and feeds. There are also API calls to update and retrieve the information contained in the profile box.

For each API call, there is a detailed description of the parameters and the returned XML.

The available API calls seem quite comprehensive, and it is possible to retrieve most (if not all) of the information the user has access to (within the allowed bounds – in a session of one user, an application cannot retrieve detailed information about other users which are not his friends, for example).

Facebook provides an online API test console³, where the developer can enter API calls and check the results that will be generated.

There is also a „Data Storage API“, which is currently in beta status and was not used or tested during this evaluation.

1 <http://developers.facebook.com/documentation.php>

2 http://wiki.developers.facebook.com/index.php/Main_Page

3 <http://developers.facebook.com/tools.php?api>

4.3 FQL

A recent addition to the Facebook API was the „Facebook Query Language“, which has an SQL like syntax. An example FQL statement would be:

```
SELECT name FROM group WHERE gid IN (SELECT gid FROM group_member WHERE uid = u1)
```

This query gets the names of the groups user *u1* is a member of. Using the traditional Facebook API calls, this would have included more than one API call and retrieved much more than the desired information. Thus, FQL allows for more efficient information retrieval. According to the documentation, most traditional API calls are internally mapped to FQL queries nevertheless.

There are some limits to the FQL statements, for example, there can only exist one argument to the FROM clause, and there must always be a WHERE clause present, including at least one of a group of special identifiers (for example a user id, user name, group id, etc) to limit the number of results returned.

There is detailed information available on the web site on what columns from which tables can be retrieved on <http://developers.facebook.com/documentation.php?v=1.0&doc=fql>

4.4 API client libraries

Facebook provides client libraries for its API. Officially supported are those written in Java and PHP. Unofficial versions exist for a number of other programming languages, including but not limited to .NET, Ruby, Perl and Python. I have worked with and examined the PHP client library.

The library itself is not very bulky. It's a bit more than 800 lines (including comments) and provides:

- Session handling
- Facebook parameter processing and validation
- „Low level“ methods converting and signing parameters and sending them in a CURL (or basic, if CURL is not available) HTTP call to the server, as well as converting the XML response into a PHP array
- PHP function calls wrapping the API calls to use the low level methods
- Some utility methods provided for convenience
- Constants

Unfortunately, the author of this evaluation did not find any documentation of the PHP API client library, so he had to look into himself to see what is available, what it does, how it does its work and how to call the methods wrapping the API calls. Due to the small size of the library, this was not a big problem, though.

The example application gives a good starting point, although there could be more provided, especially when it comes to passing own parameters to the application, as the application has to take care of sanity and validity checks itself. It would be nice to have that handled by the library too, to ensure a certain level of security of the applications. It could probably be done using the lower level library calls, or at least a modified version of them, but there is no documentation about this.

4.5 FBML

FBML (FaceBook Markup Language) is Facebook's approach on providing the user with a number of custom tags in addition to the normal html tags, and removing some of the normal html tags. The naming is not consistent: some HTML tags were given additional parameters, but the tags still have the same name (e.g. <form>), where other FBML tags have their own namespace (e.g. <fb:name>). The result is a mixture of normal HTML with an addition of some attributes and completely new FBML tags.

Still, the FBML tags provide a valuable assistance for the developer. With the help of just a single tag for example, an input box can be created where the user can enter the name of one of his friends, which will provide a drop down list suggesting names matching the already entered prefix of the name (just like google suggest – one of the most prominent web 2.0 examples). When submitting the form, the user id is passed as the parameter instead of the entered name. Of course, this does not take the burden of parameter validation away from the developer, but it definitely makes life easier.

There exist quite a number of such tags for choosing one or more friends, for user and group content, embedded media, an „editor“ (which is a nicely formatted form, including a number of input fields), page navigation, notifications and (add application) requests, some miscellaneous tags for time, redirects and more, etc. A complete list can be found on the Facebook developer wiki⁴.

For example, creating a named user link (with many formatting options) is as easy as this:

```
<fb:name uid="12345" firstnameonly="true" possessive="true" />
```

(This would create a link to a user's profile, let's name him John Doe, displayed as „John's“.)

4 <http://wiki.developers.facebook.com/index.php/FBML>

The FBML tags are a very valuable addition to the Facebook API, enabling the developer to quickly create standard tags, forms, etc. This also helps to keep the look and feel of Facebook applications consistent, so that they will fit nicely into the whole and users will still feel that they are „at Facebook“.

Facebook provides an online FBML test console⁵ (like the API test console), where the user can enter any FBML code and view the resulting HTML code.

4.6 JavaScript

Using JavaScript is allowed, even when not in an iframe context. Facebook will scan the JavaScript code and prepend the application id to all function definitions, calls and variable names. Also, Facebook has implemented its own DOM objects, while using the default DOM objects is not allowed. Thus, the syntax is different, but they claim to have most functionality covered.

Facebook supplies the developer with an own AJAX object to support the developers writing dynamic content, as well as a Dialog object to be able to hook into Facebook's base dialog abstraction.

All in all, Facebook's JavaScript approach (called „FBJS“ by Facebook) does not look bad: it provides for a „sandbox“ like operation mode, so that different applications do not interfere with each other on the profile page, and gives access to some convenient methods to add Web 2.0 content to the application.

It is hard to tell how difficult doing forbidden things would be; there is, for example, not a word about JavaScript's eval() function in the documentation. No testing was done in conjunction with this evaluation, though.

5 <http://developers.facebook.com/tools.php?fbml>

5 Personal Experience and Opinion

5.1 Usability

No bugs or any other weird behaviour was encountered while using Facebook and developing the small example application. The Facebook site itself is well done and clear, only the profile page can become quite overloaded depending on what applications immortalize themselves there.

Facebook checks for every request if the user is a developer of the application, and if that is the case, it will include a bit more information in the resulting HTML (or JavaScript code), usually as comments, which is useful for debugging. Also, some tips for debugging JavaScript are given in the developer wiki.

For development, the application can be restricted to be added only by the application's developers.

5.2 Internationalization

The internationalization can be criticized: Facebook seems to have problems dealing with special characters such as the German umlauts and other characters appearing in non-US character sets. The communication with the API is required to be completely in UTF-8, which is a good thing. But the special character issue is not raised in the documentation at all, and no help is provided to overcome PHP's lacking unicode support, resulting in weird signs even in Facebook's own pages where special characters like for example an „ü“ as in „München“ should appear. This should be less of an issue using Java.

5.3 Security

The author of this evaluation thinks that the session and authentication handling mechanism of Facebook is sufficiently secure. Signing the Facebook parameters with the private api key should provide for a security level that is better than what is found in many other web applications.

Unfortunately, the Facebook API is lacking in secure http support: The only place where it is used is the login process. This means that the data and the parameters are sent unencrypted most of the time, and if a user would like to make a secure payment, for example, he would have to be sent to an external site to process the payment request.

5.4 Conclusion

The Facebook API looks very usable, and especially the FBML and JavaScript objects could enable a developer to create Facebook applications fast, providing a look and feel that matches the original Facebook platform.

The Facebook API is not a complete framework, though: The developer receives some help, but still has to care about many things himself, such as parameter validation and sanitisation.

The lack of secure http support though is a real drawback, as there is no way to safely transport critical information (other than the login/authorization information) between Facebook and the Facebook application.

Still, it is probably very easy to get a big user base for a good, new application fast.

6 Appendix

6.1 Database Layout of the Example Application

Field	Type	Null	Key	Default	Extra
uid	int(11)	NO	PRI		
credits	int(11)	NO			

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
sender	int(11)	NO	MUL		
receiver	int(11)	NO	MUL		
amount	int(11)	NO			
time	int(11)	NO	MUL		

6.2 appinclude.php

```
<?php
require_once 'facebook-platform/client/facebook.php';

/**
 * This is a wrapper for the facebook client API
 */
class FBWrapper {
    private $appapikey = '4d0745235b54b3c01512474d005d96f7';
    private $appsecret = '33e778b25600d6ec6904fbe2e1853a8c';
    private $appcallbackurl = 'http://gardan.hopto.org/pqowieur/';
    private $appname = 'WolframTestApp';
    private $appdir = 'pqowieur';
    private $Facebook;
    private $uid;
    private $params;
    private $friends = null;
    private $app_friends = null;

    public function __construct() {
        $this->Facebook = new Facebook($this->appapikey, $this->appsecret);
        $this->uid = $this->Facebook->require_login();

        //catch the exception that gets thrown if the cookie has an invalid session_key in it
        try {
            if (!$this->Facebook->api_client->users_isAppAdded()) {
                $this->Facebook->redirect($this->Facebook->get_add_url());
            }
        } catch (Exception $ex) {
            //this will clear cookies for your application and redirect them to a login prompt
            $this->Facebook->set_user(null, null);
            $this->Facebook->redirect($this->appcallbackurl);
        }
    }

    public function getFBParams() {
        return $this->Facebook->fb_params;
    }
}
```

```

public function getAddUrl() {
    return $this->Facebook->get_add_url();
}

public function getUrl($params = null) {
    $paramstr = '';
    if( is_array($params) ) {
        $paramstr = '?';
        $i = 0;
        foreach( $params as $k => $v ) {
            $paramstr .= ($i++ == 0 ? '' : '&').$k.'='.$v;
        }
    }
    return $this->Facebook->get_facebook_url('apps').'/'.$this->appid.'/'.$paramstr;
}

public function setProfile($fbml) {
    $this->Facebook->api_client->profile_setFBML($fbml);
}

public function getUid() {
    return $this->uid;
}

public function getFriends() {
    if( is_null($this->friends) ) {
        $this->friends = $this->Facebook->api_client->friends_get();
        if( $this->friends === '' )
            $this->friends = array();
        foreach( $this->friends as $k => $v )
            $this->friends[$k] = (int)$v;
    }
    return $this->friends;
}

public function getFriendsWithApp() {
    if( is_null($this->app_friends) ) {
        $this->app_friends = $this->Facebook->api_client->friends_getAppUsers();
        if( $this->app_friends === '' )
            $this->app_friends = array();
        foreach( $this->app_friends as $k => $v )
            $this->app_friends[$k] = (int)$v;
    }
    return $this->app_friends;
}

public function getFriendsWithoutApp() {
    return array_diff($this->getFriends(), $this->getFriendsWithApp());
}
}
?>

```

6.3 appinclude.php

```

<?php
require_once 'appinclude.php';

/**
 * Database abstraction
 */
class MyDatabase {
    private $dbhost = 'localhost';
    private $dbuser = 'facebook';
}

```

```

private $dbpass = 'facebook';
private $dbname = 'facebook';
private $db = null;
private $fbw;
private $uid;

public function __construct($fbw) {
    $this->fbw = $fbw;
    $this->uid = (int)$fbw->getUid();
    try {
        $this->db = new PDO('mysql:host='.$this->dbhost.';dbname='.$this->dbname,
            $this->dbuser, $this->dbpass, array(PDO::ATTR_PERSISTENT => true));
        $this->db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    } catch (PDOException $e) {
        $error = 'Application unavailable: Database error.';
        die(0);
    }
}

public function beginTransaction() {
    $this->db->beginTransaction();
}

public function commit() {
    $this->db->commit();
}

public function rollBack() {
    $this->db->rollBack();
}

/**
 * Give a new user his initial credits
 */
public function initialCredits($uid = null) {
    if( is_null($uid) )
        $uid = $this->uid;
    $value = 100;
    $stmt = $this->db->prepare('INSERT INTO credits (uid, credits) VALUES (?, ?)');
    $stmt->bindValue(1, $uid, PDO::PARAM_INT);
    $stmt->bindValue(2, $value, PDO::PARAM_INT);
    if( !$stmt->execute() ) {
        throw new Exception(0, 'Database error');
    } else {
        $this->fbw->setProfile('Your credit counter: '.$value);
        return $value;
    }
}

/**
 * Get a user's credit counter
 */
public function getCredits($uid = null) {
    if( is_null($uid) )
        $uid = $this->uid;
    $result = null;
    $stmt = $this->db->prepare('SELECT credits FROM credits WHERE uid = ?');
    $stmt->bindValue(1, $uid, PDO::PARAM_INT);
    $stmt->execute();
    $stmt->bindColumn(1, $result);
    $stmt->fetch(PDO::FETCH_BOUND);
    if( is_null($result) ) {
        $result = $this->initialCredits($uid);
    }
    return $result;
}

```

```

/**
 * Add/subtract credits to/from a user's credit account.
 * @param int $value Positive value to add, negative value to subtract credits.
 */
public function addCredits($value, $uid = null) {
    if( is_null($uid) )
        $uid = $this->uid;
    $credits = $this->getCredits($uid, true) + $value;
    $stmt = $this->db->prepare('UPDATE credits SET credits = ? WHERE uid = ?');
    $stmt->bindValue(1, $credits, PDO::PARAM_INT);
    $stmt->bindValue(2, $uid, PDO::PARAM_INT);
    $stmt->execute();
    if( !$stmt->execute() ) {
        $result = $this->initialCredits($uid);
        $this->fbw->setProfile('Your credit counter: -error-');
        throw new Exception(0, 'Database error');
    } else {
        $this->fbw->setProfile('Your credit counter: '.$credits);
    }
}

/**
 * Create a history entry for a transaction.
 */
public function recordHistory($sender, $receiver, $amount) {
    $stmt = $this->db->prepare('INSERT INTO history (sender, receiver, amount, time) VALUES (?, ?, ?, ?)');
    $stmt->bindValue(1, $sender, PDO::PARAM_INT);
    $stmt->bindValue(2, $receiver, PDO::PARAM_INT);
    $stmt->bindValue(3, $amount, PDO::PARAM_INT);
    $stmt->bindValue(4, time(), PDO::PARAM_INT);
    if( !$stmt->execute() ) {
        throw new Exception(0, 'Database error');
    }
}

/**
 * Get transaction history records for a user.
 */
public function getHistory($uid = null) {
    if( is_null($uid) )
        $uid = $this->uid;
    $stmt = $this->db->prepare('SELECT * FROM history WHERE (sender = ? OR receiver = ?) ORDER BY time DESC');
    $stmt->bindValue(1, $uid, PDO::PARAM_INT);
    $stmt->bindValue(2, $uid, PDO::PARAM_INT);
    if( !$stmt->execute() ) {
        throw new Exception(0, 'Database error');
    }
    $result = array();
    while( $row = $stmt->fetch() ) {
        $result[] = $row;
    }
    $stmt->closeCursor();
    return $result;
}

}

/**
 * "Template" class
 */
class MyDisplay {
    /**
     * Create fbml code for the transaction history of a user
     */
    public static function transfer_history($hist) {
        $res = '';
        if( sizeof($hist) > 0 ) {
            $res = '<fb:fbml><br/><h1>Transfer history</h1>';
        }
    }
}

```

```

    foreach( $hist as $h ) {
        $res .= '<p>';
        $res .= '<fb:name uid="'. $h['sender'].'" useyou="true" capitalize="true"/>';
        $res .= ' transferred '. $h['amount']. ' to ';
        $res .= '<fb:name uid="'. $h['receiver'].'" useyou="true" capitalize="false"/>';
        $res .= ' on '. date(DATE_RFC2822, $h['time']). '.';
        $res .= '</p>';
    }
    $res .= '</fb:fbml>';
}
return $res;
}

/**
 * Default page canvas template
 */
public static function canvas_default($error, $msg, $credits, $uid, $url) {
    if( $error == 'database' )
        return '<p>A Database error has occurred. Please try again later.</p>';
    return <<<CANVAS_DEFAULT
    <fb:fbml>
    <fb:header>WolframTestApp</fb:header>
    $msg
    <p>You have $credits credits!</p>
    <fb:editor action="?do=transfer" labelwidth="80">
        <fb:editor-custom label="Choose a friend">
            <fb:friend-selector uid="$uid()" idname="receiver"/>
        </fb:editor-custom>
        <fb:editor-text label="Amount" name="amount" value=""/>
        <fb:editor-buttonset>
            <fb:editor-button value="Give away!"/>
        </fb:editor-buttonset>
    </fb:editor>
    <p><a href="$url">Your friend is not in the list? Invite new friends to
        WolframTestApp!</a></p>
    </fb:fbml>
CANVAS_DEFAULT;
}

/**
 * Invite friends page canvas template
 */
public static function canvas_invite($uid, $url, $friendswithapp) {
    $invtext = <<<INVTEXT
    You have been invited to join the credit system!
    <fb:name uid="$uid" useyou="false" capitalize="true"/>
    wants You to add WolframTestApp so that you can trade credits with <fb:pronoun possessive="true"
uid="$uid"/>.
    <fb:req-choice url="$url" label="Add it!"/>
INVTEXT;
    $invtext = htmlentities($invtext);
    return <<<CANVAS_INVITE
    <fb:fbml>
    <fb:header>WolframTestApp</fb:header>
    <p>Here, you can invite friends to use WolframTestApp too (only your friends that don't use Wolf-
ramTestApp yet are listed).</p>
    <fb:request-form action="index.php" method="POST"
        invite="true" type="WolframTestApp" action="" content="$invtext">
        <fb:multi-friend-selector showborder="false" actiontext="Invite your friends to use WolframTestApp."
max="20" exclude_ids="$friendswithapp">
    </fb:request-form>
    </fb:fbml>
CANVAS_INVITE;
}

```

```

/**
 * Default page canvas template
 */
public static function canvas_transfer($error, $amount, $receiver) {
    if( $error == 'amount' )
        return '<p>You have to choose an amount!</p>';
    if( $error == 'toomuch' )
        return '<p>You cannot give away more than you have!</p>';
    if( $error == 'receiver' )
        return '<p>You have to choose a friend of yours!</p>';
    if( $error == 'noappuser' )
        return '<p>Whoops! Your friend is not using WolframTestapp! You have to get him to install this great
tool first. Use the link at the bottom of the page.</p>';
    if( $error == 'database' )
        return '<p>A Database error has ocured. Please try again later.</p>';
    return <<<CANVAS_TRANSFER
        You have transferred $amount credits to <fb:name uid="$receiver"/>.
CANVAS_TRANSFER;
}
}

/**
 * Business logic is here
 */
class MyController {
    private $fbw;
    private $db;
    private $params;

    public function __construct() {
        $this->params = array(
            'do' => $this->validateParam('do'),
            'amount' => (int)$this->validateParam('amount'),
            'receiver' => (int)$this->validateParam('receiver'),
        );
    }

    /**
     * Simplest possible parameter validation
     */
    public function validateParam($p) {
        if( isset($_REQUEST[$p]) )
            return $_REQUEST[$p];
        return null;
    }

    /**
     * Create the default page canvas
     */
    public function canvas_default($msg = '') {
        $error = '';
        $credits = 0;
        try {
            $this->db->beginTransaction();
            $credits = $this->db->getCredits();
            $hist = $this->db->getHistory();
            $this->db->commit();
        } catch(Exception $e) {
            echo $e->getMessage();
            $this->db->rollBack();
            $error = 'database';
        }
        $res = MyDisplay::canvas_default(
            $error,

```

```

        $msg,
        $credits,
        $this->fbw->getUid(),
        $this->fbw->getUrl(array('do'=>'invite'))
    );
    return $res . MyDisplay::transfer_history($hist);
}

/**
 * Create the invite friends page canvas
 */
public function canvas_invite() {
    return MyDisplay::canvas_invite(
        $this->fbw->getUid(),
        $this->fbw->getAddUrl(),
        implode(',', $this->fbw->getFriendsWithApp())
    );
}

/**
 * Display info if a user transferred credits
 */
public function canvas_transfer() {
    $error = '';
    $amount = 0;
    $receiver = 0;
    $credits = 0;
    try {
        $this->db->beginTransaction();
        $credits = $this->db->getCredits();
        $this->db->commit();
    } catch(Exception $e) {
        $this->db->rollBack();
        $error = 'database';
    }

    if( !($this->params['amount'] > 0) )
        $error = 'amount';
    if( $credits < $this->params['amount'] )
        $error = 'toomuch';
    if( !($this->params['receiver'] > 0) )
        $error = 'receiver';
    if( !in_array($this->params['receiver'], $this->fbw->getFriendsWithApp()) )
        $error = 'noappuser';

    if( $error == '' ) {
        try {
            $amount = $this->params['amount'];
            $receiver = $this->params['receiver'];
            $this->db->beginTransaction();
            $this->db->addCredits($amount, $receiver);
            $this->db->addCredits(-$amount);
            $this->db->recordHistory($this->fbw->getUid(), $receiver, $amount);
            $this->db->commit();
        } catch(Exception $e) {
            $this->db->rollBack();
            $error = 'database';
        }
    }

    return MyDisplay::canvas_transfer($error, $amount, $receiver);
}

/**
 * Choose what to display

```

```

*/
public function work() {
    try {
        $this->fbw = new FBWrapper();
        $this->db = new MyDatabase($this->fbw);

        $output = '';
        switch( $this->params['do'] ) {
            case 'invite':
                $output = $this->canvas_invite();
                break;
            case 'transfer':
                $output = $this->canvas_transfer();
                $output = $this->canvas_default($output);
                break;
            default:
                $output = $this->canvas_default();
        }
        echo $output;
    } catch( Exception $e) {
        echo "<p>Error: ".$e->getMessage()."</p>";
        exit(0);
    }
}

}

// Create controller and start business logic
$ctrl = new MyController();
$ctrl->work();

?>

```