
# PHPforFB

*A Professional PHP Framework for Facebook®*

Programmer's

Guide & complete Reference

# *Copyright*



Many Thanks to my Friends: Javad Ghodrati, Heiko Ellwang and Josef Schneider

Handbook Version: Februar 2012

# PHPforFB

## A Professional PHP Framework for Facebook®

---

## Introduction

*The PHPforFB framework allows for an easy and very efficient way of programming Facebook® apps and fan page extensions (page apps).*

*The PHPforFB manual is exhaustive and helps programmers to get started swiftly developing their own Facebook® apps in PHP.*

# Table of Contents

# Part I

***PHPforFB Framework***

# 1 PHPforFB Framework

## 1.1 Introduction

Facebook® is the world's largest and most import social network. In Facebook®, there are people gathered from all population groups, from all over the world, and they are online on a daily basis.

Facebook® offers numerous functions which are helpful for its users. Moreover, it offers a programming interface, which every programmer can use to enhance Facebook® with new features in the form of so-called 'apps' (short for applications). As of late, even fan pages can be extended with programs ('fan page extensions', or 'pageapps') written by oneself. Facebook® can be extended in any programming language, because the Facebook® programming interfaces are purely based on HTTP.

PHP is the most widely used programming/scripting language in the online world, and mastered by many programmers. Facebook® is being used by many people. Hence, PHP and Facebook® are a perfect match. Unfortunately, it isn't exactly easy and efficient to develop Facebook® applications in PHP, and even the PHP software development kit (SDK) that is offered by Facebook® can be laborious at times.

The **PHPforFB** PHP framework allows for an easy and very efficient way of programming Facebook® apps and fan page extensions (page apps). The **PHPforFB** manual is exhaustive and helps programmers to get started swiftly developing their own Facebook® apps in PHP. The manual is comprised of several tutorials, a methods and properties reference, as well as some information about practical uses of **PHPforFB**.

What the manual offers:

- A complete reference of all of **PHPforFB**'s methods and variables, including examples

## 1.2 What is PHPforFB?

**PHPforFB** is a PHP framework that helps to simplify the development and operation of Facebook® applications (and page extensions). By the help of this framework, you can write and publish Facebook® applications quickly and efficiently, while only basic PHP and web knowledge is required.

The benefit of **PHPforFB** is that you have only marginally to deal with Facebook® server communication and its respective interfaces. Instead, you can concentrate on the development and functionality of your application.

The **PHPforFB** framework provides the Facebook® functionality and access to its interfaces, which are indispensable for an Facebook® application. Two examples:

- The whole process of prompting the user for permissions is taken care of with a simple method.
- Dashboard entries can be posted with a single method invocation.

If you have ever written a Facebook® application before, you probably know how difficult and tiresome it can be to achieve even simple goals. Much is documented only incompletely, and many important details must be worked out difficultly.

This work is now being undertaken for you by **PHPforFB**!

## 1.3 Why use PHPforFB framework?

**PHPforFB** makes it easy to develop Facebook® applications, and it reduces the development time and effort to the bare application programming.

**The most important advantages of the PHPforFB framework:**

- **PHPforFB** increases your efficiency, lowers the development costs and saves time!
- **PHPforFB** allows for an improved user-friendliness!
- **PHPforFB** increases the performance of your application more than fivefold by caching Facebook® query results internally!
- **PHPforFB** runs on any server, even on inexpensive mass hosters' small web space!
- **PHPforFB** runs with any PHP version since PHP 4!
- Facebook® grows constantly and **PHPforFB** offers its latest features!
- Facebook® develops and changes continuously, and **PHPforFB** adapts to its latest changes!

## 1.4 The benefits of PHPforFB

**PHPforFB** offers many advantages for both beginners and professionals:

### 1.4.1 Cost efficiency and time saving

Thanks to **PHPforFB**, you don't have to deal with Facebook® server communication. No knowledge of GraphAPI details is required, and you can dive right into application development.

With **PHPforFB** you can save a lot of development effort and money.

You can realize more projects and tackle projects which without **PHPforFB** were not viable financially.

### 1.4.2 Focusing on the essentials

By the use of **PHPforFB**, you can concentrate on the bare essentials of application development.

The major part of the tiresome interface communication is being undertaken by **PHPforFB** .

As a result, you will have to program only the effective application.

### 1.4.3 Security with OAuth 2.0

**PHPforFB** was designed from the start with the OAuth 2.0 authentication method exclusively.

As a result, **PHPforFB** offers full compliance with Facebook®'s security guidelines and prevents the loss and theft of data.

### 1.4.4 Increased performance

**PHPforFB** contains an efficient caching mechanism, which helps to reduce the number of queries to Facebook® servers to a minimum.

The caching will result in an amazing performance gain, because every real request to Facebook® servers takes approximately two seconds.

Instead of repeatedly requesting data in each of an application's script files (and by this, keeping the user waiting needlessly), data once obtained can be reused as long as an user session exists.

The **PHPforFB** caching mechanism is available spanning the entire session.

### 1.4.5 More user-friendliness

**PHPforFB** allows to develop Facebook® applications with an attention to detail for improved user-friendliness.

For instance, at any point inside an application, an (additional) permission can be requested from the user. After the permission was granted, the application is continued at exactly the same point.

You are precisely in control of the program flow.

### 1.4.6 Special advantages for beginners

Beginners can save the trouble of programming the Facebook® interfaces.

With **PHPforFB**, the application programmer is shielded from the complicated (but necessary) programming of authentication as well as GraphAPI and FQL communication.

### 1.4.7 Special advantages for professionals

Professionals can optionally use the GraphAPI and FQL interfaces directly, inside and assisted by **PHPforFB**, while they can save the routine of programming the interfaces for authentication (just like beginners).

For professionals, time is usually money, and with **PHPforFB** they can reach their goals more quickly.

### 1.4.8 Facebook changes constantly

Facebook® changes constantly and offers an ever-growing number of interactions, be it the form of new interfaces or the extension of existing ones. It is possible that a once created application stops working all of a sudden, because a particular access method is no longer favored and supported by Facebook®.

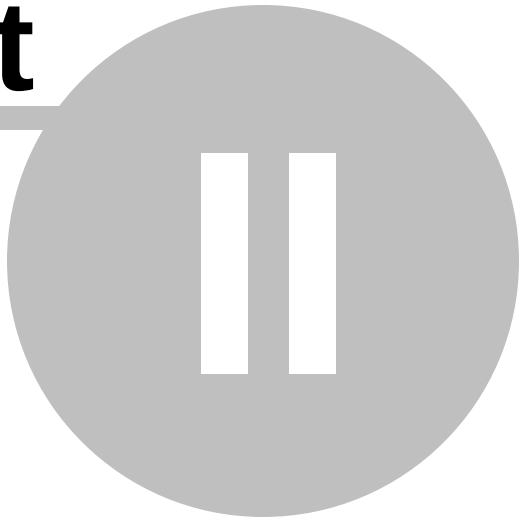
This can even render an entire Facebook® application obsolete altogether. But with **PHPforFB**, adaptations are constantly being undertaken to keep your applications up-to-date; they remain unaffected by these changes.

### 1.4.9 Facebook grows and extends its interfaces

Facebook® is ever-growing in size and functionality, and not only constantly, but sometimes rapidly and with short-term announcements.

There is often no, insufficient, inconvenient, or ambiguous documentation regarding the different interfaces and possibilities available in the internet. If you do not like to constantly research solutions for hours and hours, with **PHPforFB** you can use the most important features of Facebook® easily and conveniently, while you will always have access to the latest extensions and features.

# Part



*Framework Reference*

# 2 Framework Reference

## 2.1 Constructor

The constructor of the **PHPforFB** framework creates an object of the class **PHPforFB**.

The created instance will be readily initialized with existing values from Facebook®, and its properties can be queried immediately after creation (see examples).

The class can save its state in the session space, and restore itself completely upon restoration in the scope of the same PHP project. Therefore, it is important to create an instance of the class in the first line of every PHP script that needs it.

Also, the constructor reestablishes any data which have previously been queried from Facebook® (such as the "userData" property), and makes them available for subsequent use.

If successful, the constructor returns a newly created instance of the class.
If an error occurs during initialization, the lastErrorCode object variable will be greater 0, and lastError contains a description of the error.

There exist three different modes of operation:

"**app**"/"**page**" : The class is used for an application or a fan page extension.

"**callback**" : The class is registered with a callback URL at Facebook®

"**direct_use**" : Direct access to Facebook® data from the outside (without registration, authentication or similar)

**Invocation:**

```
__construct ( ARRAY $structInitData)
```

**Parameters:**

| | |
|---|---|
| ARRAY $structInitData | Array of initialization values.<br><br>Depending on the mode, certain parameters are required e.g. for authentication with Facebook®. In addition to these, optional parameters can be used for some basic configuration.<br><br>Required arguments in Array $structInitData:<br><br>• *"mode" = "app"* or "*page*"<br><br>"app_id"<br>"app_name" (name of canvas page) |

| | |
|---|---|
| | "sec_key"<br><br>These three parameters are obtained when an application is registered with Facebook®. **app_name** is the last part of the canvas page name, which must be given when an application is created; e.g. "friends-optimizer" when the name of the canvas page is " http://apps.facebook.com/friends-optimizer".<br><br>• *"mode" = "callback"*<br>No mandatory parameters<br><br>• *"mode" = "direct_use"*<br>No mandatory parameters<br><br>Optionally, the following parameters can be specified:<br>• logdir<br>• logLevel<br>• logging<br>• caching<br>• cacheExpires<br>• runOutofIframe |
| | |

**Returns:**

| | |
|---|---|
| Object instance | **Note:**<br>On error, lastErrorCode is greater than 0 and lastError contains an error description |

**Variables available after initialization:**

| | |
|---|---|
| "mode" = "app"/"page" | userID(*), accessToken(*), userUsedLanguage, userActualCountry, userIsAdult, userLoggedIn, userAuthenticated, callFromFacebook, isMobileDevice, mobileDevice, callAsPage(**), pageID(**), isPageAdmin(**), userLikesPage(**)<br><br>(*) Only if the user has consented to at least basic permissions<br>(**) Only in "page" mode |
| "mode" = "callback" | userID, userUsedLanguage, userActualCountry |
| "mode" = "direct_use" | *none* |

See the following example.

**Important:** This should take place on every page of the application, or in a file that is included from every script in the project (such as a file named "config.php").

```php
$structInit = array(
   'app_id' => APP_ID,
   'app_name' => APP_NAME,
   'sec_key' => APP_SECKEY
);

$FacebookAPP = new PHPforFB($structInit);
if($FacebookAPP->lastErrorCode>0){
   #An error occurred during creation.
   echo "PHPforFB Error: ".$FacebookAPP->lastErrorCode." -> ".$FacebookAPP->lastError;
   exit;
}
else{
   #The actual application code follows here
}
```

## 2.2 Variables / Properties

### 2.2.1 accessToken

Contains the "access_token" value, which is assigned by Facebook® after authentication.

The access token permits the access to certain data via the Graph API. The Graph API returns a different number of values depending on the granted permissions and the presence of the access token.

This variable will be set after a successful authentication and is valid during the lifetime of the application.

The PHPforFB framework takes care of the management, creation of and queries to the access token.

### 2.2.2 appFBURL

Invocation address of the application in Facebook®.

The format is always http://apps.facebook.com/CANVAS_NAME

### 2.2.3 appID

Facebook® Id of the application.

The application Id is assigned by Facebook® upon Registration of one's own application and can be looked up at Facebook®.

### 2.2.4 appName

Facebook® canvas name of the application.

**app_name** is the last part of the canvas page name, which must be given when an

application is created; e.g. "friends-optimizer" when the name of the canvas page is " http://apps.facebook.com/friends-optimizer".

The canvas name is chosen by the application creator when Registering one's application.

### 2.2.5 appServerURL

Server address / URL of the script (starting page), which is used by the PHPforFB framework.

### 2.2.6 cacheExpires

Specifies the time (in seconds) for cache expiration, that is, the time after which query results from Facebook® will be renewed and updated.

PHPforFB buffers query results from Facebook® in a cache. This helps to accelerate applications considerably.

This variable is used only if caching is set to TRUE. By default, caching is active.

Initially, this value is 600, which equals to 10 minutes for a session per user.

**Values:**

| | |
|---|---|
| INT | Default = 600 |

### 2.2.7 caching

Returns the status and activates / deactivates caching.

The variable determines whether PHPforFB should buffer Facebook® contents in a cache, or if contents should be fetched from Facebook's servers on each access. If caching is active, the number of requests sent to Facebook is minimized. Because each query can take up to three seconds, caching can drastically improve the reactivity of an application.

The cache relates to every session of a user.

The cache's expiration time can be adjusted with cacheExpires. By default, this value is 600 seconds = 10 minutes.

The initial value is TRUE, i.e. caching is by default activated.

**Werte:**

| | |
|---|---|
| TRUE | Caching active (Default) |
| FALSE | Caching deactivated |

### 2.2.8 callAsPage

Reflects the invocation type of the site.

This variable returns TRUE if the current call is a fan page invocation, i.e. if the application is called from a page.

**Werte:**

| | |
|---|---|
| TRUE | Yes, it is a page call |
| FALSE | No (Default), it is a regular application call |

### 2.2.9 callFrom

Specifies from which Facebook® section the application was called.

A Facebook® user can invoke an application by a click to a link shown at various places, e.g. in a friend's list of applications.

With this variable you can determine from where an user has called the application. This can be interesting, for example, for usage statistics.

Empty means that the origin was unknown or could not be determined.

**Values:**

| | |
|---|---|
| STRING | Possible values (excerpt):<br>• "nf" : Newsfeed (starting page, dashboard)<br>• "appd_friends_apps" : List of a friend's applications<br>• "appd_my_recent" : List of own applications<br>• "bookmarks" : Bookmark in one's own profile<br>• "" = Unknown / indeterminable |

### 2.2.10 callFromFacebook

Specifies whether the current invocation was made from Facebook®.

PHP scripts using PHPforFB can work without being called from Facebook®. With this variable, the application can show useful contents even without being integrated into Facebook® and present functionality and contents to a non-Facebook® user.

**Values:**

| | |
|---|---|
| TRUE | Yes, it was called from Facebook®. |
| FALSE | No, the application was called from outside Facebook®<br>(see appServerURL) |

### 2.2.11 isMobileDevice

Specifies whether the application was called from a mobile device.

By use of this variable, the application can present a layout optimized for mobile devices.

If this value is FALSE, then no mobile devices was detected.

The type of a mobile device can be found in the mobileDevice variable.

**Values:**

| | |
|---|---|
| TRUE | Yes, the application was called from a known mobile device. The device type can be found in mobileDevice. |
| FALSE | No mobile device was detected. |

### 2.2.12 isPageAdmin

Specifies if the user is also the owner of a fan page.

If the current invocation is a fan page call (callAsPage = TRUE), then this variable specifies if it has been requested by the page owner.

**Note:** If a page has several administrators, then this variable cannot distinguish them from the owner. If called by an administrator, isPageAdmin is always TRUE.

**Values:**

| | |
|---|---|
| TRUE | Yes, the user calling is a page administrator. |
| FALSE | No, the user is not an administrator of the page. |

### 2.2.13 lastError

Contains an text describing the last error.

PHPforFB methods return FALSE in case of an error. The explanation of the error can be found in this variable.

See also "Error codes and messages"

### 2.2.14 lastErrorCode

Contains the last error's numerical error code.

PHPforFB methods return FALSE in case of an error. The error number can be found in this variable.

The following errors are possible: "Error codes and messages"

### 2.2.15 logDir

Absolute path to the directory for log files.

For debugging purposes, PHPforFB can report its activities in a protocol log. The verbosity of the log can be adjusted with the logLevel variable.

If no logging directory is specified, then logging is disabled completely.

Log files are generated on a daily basis.

### 2.2.16 logging

Activates / deactivates logging.

The initial value is 0, that is, no events will be logged.

**Werte:**

| | |
|---|---|
| 0 | deactivated (Default) |
| 1 | activated |

### 2.2.17 logLevel

Specfiies the verbosity of PHPforFB's protocol log.

**Values:**

| | |
|---|---|
| 0 | Only errors (Default) |
| 1 | + important notices |
| 2 | + additional notices |
| 3 | + debug notices |

### 2.2.18 mobileDevice

Specifies the type of a mobile device.

If the application has been called from a mobile device (isMobileDevice = TRUE), then this class variable contains the type of the mobile device.

If the type is indeterminable, then this variable contains an empty string.

**Values:**

| | |
|---|---|
| STRING | • "wap" = Classic internet-enabled cellphone<br>• "touch" = Smart phone with touchscreen |

| | |
|---|---|
| | • "" = unknown / indeterminable |

### 2.2.19 pageData

Contains information previously obtained regarding the calling fan page (see pageID).

**Warning:** For this variable to contain information, the method PHPforFB->GetPageInfo() must have been called earlier during the session. For this, a single call is sufficient, the data will then be available in subsequent calls in the same session.

**Values:**

| | |
|---|---|
| ARRAY | Various fields, depending on which fields the page fills in and Facebook® offers publicly.<br><br>*Can be displayed using var_dump($INSTANCE->pageData) after calling $INSTANCE->GetPageInfo()* |

### 2.2.20 pageID

Specifies which fan page has caused the current call to the application.

If the current call is a fan page invocation, the Id of the calling fan can be found in this variable.

By use of the page Id, additional information regarding the fan page can be queried from Facebook® (see PHPforFB->GetPageInfo()), or fan page specific information can be displayed.

**Values:**

| | |
|---|---|
| STRING | PageID of the calling fan page. |

### 2.2.21 runOutofIframe

This value must be TRUE if the application should run in a window of its own.

This implies that an application displays its information without the Facebook® border that normally surrounds it, and in a browser window or popup, not in an Iframe.

Running inside the Facebook® border is not mandatory. Set this variable to TRUE on your starting page if the application should not be embedded into Facebook®.

Although every application is first called in a Facebook® border, it can be taken to a window or popup later, after displaying a message and a button in the regular Iframe. PHPforFB supports this and also offers its authentication features in this scenario.

If an application was called from a mobile device, it can be useful to disable the Facebook ® border in order to reserve more space for the application (see isMobileDevice).

By reading the variable runOutOfIframe, the application can detect if it is running outside the Facebook® border at any time.

Initially the value is FALSE, i.e. all applications have the Facebook® border by default.

**Werte:**

| | |
|---|---|
| TRUE | Yes, application is running outside the Facebook® border |
| FALSE | No (default), application has Facebook® border |

### 2.2.22 userActualCountry

Contains the country in which the user resides when the application is called.

This country is determined by the use of a Geo-IP algorithm.

This has nothing to do (but may coincide) with the user's language setting ( userUsedLanguage) in the browser or at Facebook®.

**Werte:**

| | |
|---|---|
| STRING | Two-letter acronym of the country in which the user is currently present |

### 2.2.23 userAlbums

Contains information ("id", "name", "count", "link", "privacy", etc.) about the user's albums, which has been queried previously.

**Warning:** For this variable to contain useful data, the method Methode PHPforFB-> GetAlbums() has to be called earlier during the session. For this, a single call is sufficient, the data will then be available in subsequent calls in the same session.

**Values:**

| | |
|---|---|
| ARRAY | Various fields, depending on which fields are offered publicly by Facebook®<br><br>*Can be displayed using var_dump($INSTANCE->userAlbums) after calling $INSTANCE->GetAlbums()* |

### 2.2.24 userAuthenticated

Specifies whether the user is authenticated.

An authenticated user's Facebook® userID is always known, and the user has granted at least basic permissions.

If the value of this variable is TRUE, then the application has the ability to query the basic user information (e.g. with PHPforFB->GetUserInfo() or PHPforFB->GetUserFriendsInfo()) at the minimum.

If the value of this variable is FALSE, then the user is yet unknown to the application, and must grant permissions first for the application to be able to query information about her or her friends.

Refer to the chapter "Access Rights" for more information about granting permissions.

**Values:**

| | |
|---|---|
| TRUE | userID is known and the user has granted basic permissions at least |
| FALSE | User unknown |

### 2.2.25 userData

Contains information about the current user (see userID), which have been queried previously.

**Warning:** For this variable to contain useful data, the method PHPforFB->GetUserInfo() has to be called ealier during the session. For this, a single call is sufficient, the data will then be available in subsequent calls in the same session.

**Values:**

| | |
|---|---|
| ARRAY | • Field: ["picture_is_symbol"]<br>TRUE, if the profile picture is just an avatar, otherwise FALSE (if the user has uploaded a picture)<br><br>• Field: ["picture"]<br>STRING with an URL of the user's profile picture<br><br>• Field: ["pictures"]<br>An array of URLs of a profile picture's different sizes. The array contains the following fields: ["small"], ["normal"], ["big"]<br><br>• More fields ...<br>... depending on which fields the user has filled in, the user's privacy settings, and which permissions the user has granted.<br><br>*Can be displayed using var_dump($INSTANCE->userData) after invocation of $INSTANCE->GetUserInfo()* |

### 2.2.26 userFriendsData

Contains previously obtained information about the current user's friends.

**Warning:** For this variable to contain useful data, the method PHPforFB->GetFriendsInfo() has to be called earlier during the session. For this, a single call is sufficient, the data will then be available in subsequent calls in the same session.

userFriendsData is separated into two array parts. The first array contains a list of the friends' UserIDs, and the second array contains information about the respective friends.

**Values:**

| | |
|---|---|
| ARRAY | • Field ["friends"]<br>An array of all friends' Facebook® UserIDs<br><br>• Fields [ID_OF_FRIEND]<br>An array of all available(*) information about the friend with the given UserID<br><br>see:<br>http://developers.facebook.com/docs/reference/rest/users.getInfo/<br><br>(*) Depending on which fields the user has filled in, the user's privacy settings and which rights the user has granted.<br><br>*Can be displayed using var_dump($INSTANCE->userFriendsData ["ID_OF_FRIEND"]) after calling $INSTANCE->GetFriendsInfo()* |

### 2.2.27 userID

Contains the current user's Facebook® UserID.

This variable is set only if the user has at least granted basic permissions.

Refer to the chapter "Access Rights" for more information about granting permissions.

**Values:**

| | |
|---|---|
| STRING | Current user's Facebook® UserID. |

### 2.2.28 userIsAdult

Specifies if the user has adult status.

**Values:**

| | |
|---|---|
| TRUE | Yes, the user is at least 21 years old |
| FALSE | The user is not at least 21 years old, or his majority cannot be determined. |

### 2.2.29 userLikes

Contains previously obtained information about the current user's 'likes', i.e. all contents/ objects which the user expressed appreciation for by clicking the 'like' button.

**Warning:** For this variable to contain useful data, the method PHPforFB->GetLikes() has to be called earlier during the session. For this, a single call is sufficient, the data will then be available in subsequent calls in the same session.

**Values:**

| | |
|---|---|
| ARRAY | With the following fields:<br>"category" (e.g. "Application", "Movie", "Musician/band", etc.)<br>"name"<br>"created_time"<br>"id" (Facebook® ObjectID)<br><br>*Can be displayed using var_dump($INSTANCE->userLikes) after calling $INSTANCE->GetLikes()* |

### 2.2.30 userLikesPage

If the current call to the script is a page invocation, then this variable determines whether the user has already liked the page.

**Values:**

| | |
|---|---|
| TRUE | Yes, user has already liked the page |
| FALSE | No, user has not (yet) liked the page |

### 2.2.31 userLoggedIn

Queries whether the user is logged in to Facebook®, or not.

Many Facebook® contents are available without having to be logged in to Facebook®. userLoggedIn can inform the application if the user is logged in, and by this allow it to implement a variant program behavior.

**Werte:**

| | |
|---|---|
| TRUE | Yes, the user is logged in to Facebook® |
| FALSE | No, the user is not logged in to Facebook® (and may have followed a search engine link, for example) |

### 2.2.32 userUsedLanguage

Contains the user's currently preferred language in Facebook®.

An example for the English language is the value "en_EN", or for German "de_DE".

**Values:**

| | |
|---|---|
| STRING | Country/language abbreviation |

## 2.3 Methods

### 2.3.1 AddPermission

This method adds a permission to the list of permissions that is to be retrieved with the next call to the PHPforFB->GetAuthenticationURL() method.

This variable adds the permission to an internal variable. By calling this method repeatedly, several permissions can be collected and requested at the same time.

**Important:** Only permissions which are currently known to PHPforFB can be obtained with this method.

**Note:** A very simple and efficient way to request multiple permissions at once is provided with the method PHPforFB->ForcePermissions().

For more information and examples see the chapter "List of Access Rights".

**Invocation:**

```
AddPermission ( STRING $permission )
```

**Parameters:**

| | |
|---|---|
| STRING $permission | Wanted permission, e.g.: 'basic' |

**Returns:**

| | |
|---|---|
| TRUE | If the wanted permission was registered successfully |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

### 2.3.2 AddAppAsPage

This method adds the application to a fan page as a menu item (tab).

This way, a fan page can be extended with the application's functionality.

**Invocation:**

```
AddAppAsPage ( STRING $app_apikey )
```

**Parameters:**

| | |
|---|---|
| STRING $app_apikey | API key of the application (not to be confused with the API security key!)<br><br>The API key can be found on the Facebook® developer page:<br><br>https://www.facebook.com/developers/apps.php?app_id=APP_ID |

| | |
|---|---|
| | *(APP_ID is to be replaced by the application Id)* |

**Results:**

| | |
|---|---|
| STRING PageID | If the application was successfully added to an existing fan page, the return value is the page Id of that fan page. |
| STRING "" | If aborted by the user, the return value is an empty string |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

Here is a short example to illustrate usage of this method:

```
$structInit = array(
   'app_id' => APP_ID,
   'app_name' => APP_NAME,
   'sec_key' => APP_SECKEY,
);
$FacebookAPP = new PHPforFB($structInit);
if($FacebookAPP->lastErrorCode>0){
   #an error occurred during creation => output and exit program
   echo "PHPforFB Error: ".$FacebookAPP->lastErrorCode." -> ".$FacebookAPP->lastError;
   exit;
}else{
   //APP_APIKEY must be set
   if(($res = $FacebookAPP->AddAppAsPage(APP_APIKEY)) === FALSE){
      //Error
      echo "PHP4FB Error: ".$FacebookAPP->lastErrorCode." -> ".$FacebookAPP->lastError;
      echo 'APP_APIKEY must be set to the API key from
            <a href="https://www.facebook.com/developers/apps.php?app_id='.APP_ID.'">
            here</a>';
      exit;
   }else{
      if($res==''){
         //No page found or cancelled
         echo '<h3>ERROR</h3>Aborted.';
      }else{
         //Successfully added application to a page
         $pageID = $res;

         $page = $FacebookAPP->getPageInfo($pageID);//Fetch page info
         if($page === FALSE){
            echo "PHPFacebook Error: ".$FacebookAPP->lastErrorCode;
            echo " -> ".$FacebookAPP->lastError;
            exit;
         }
         //Now call the page to change into page mode
```

```
         //Has the page link parameters already?
         if(strpos($page['link'],'?') === false)
            $page['link'] = $page['link'].'?sk=app_'.APP_ID;
         else
            $page['link'] = $page['link'].'&sk=app_'.APP_ID;
         echo '<script type="text/javascript">
               window.open(\''.$page['link'].'\',\'_top\');
               </script>';//Now redirect using Javascript
         //and via page open tab in which the application resides
         exit;
      }
   }
}
```

### 2.3.3 CreateAlbum

Creates a new photo album for the user.

With this function, a new album of the specified name is added to an user's existing albums. Thereafter, pictures can be uploaded to this album using the PHPforFB->PostToAlbum() method.

This method requires an authenticated user who has granted the respective permissions ("user_photos", "friend_photos").

**Invocation:**

CreateAlbum (STRING $name, STRING $caption [, STRING $userID] )

**Parameters:**

| | |
|---|---|
| STRING $name | Name of the album |
| STRING $caption | Album description |
| STRING $userid (optional) | Facebook UserID<br>(Default = the current user) |

**Returns:**

| | |
|---|---|
| STRING | AlbumID of the newly created album |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

**See also:**

http://developers.facebook.com/docs/reference/api/album/

Example to illustrate the creation of a new album for the current user:

```
$alb_name = $FacebookAPP->appName;
$alb_caption = 'Meine '.$FacebookAPP->appName.' Bilder';
$erg = $FacebookAPP->CreateAlbum($alb_name, $alb_caption);
if($erg === FALSE){
   #An error occurred
   echo "Error: ".$FacebookAPP->lastErrorCode." -> ".$FacebookAPP->lastError;
   exit;
}else{
   #Album created successfully
   $alb_id = $erg;
}
```

### 2.3.4 EnableXFBML

This method generates the initialization code needed for the use of XFBML.

XFBML offers simple Facebook® elements and dialogues, which can be used in applications.

For being able to use XFBML tags in an application, an initialization is required, and the needed code is generated by this method. The code must be integrated into a page's body section.

**Invocation:**

```
EnableXFBML ( [BOOL $setlanguage] [, BOOL $output] )
```

**Parameters:**

| | |
|---|---|
| BOOL $setlanguage (optional) | TRUE : Take the language settings for the XFBML generation from the userUsedLanguage field<br><br>FALSE (default): use English as the default language for XFBML generation |
| BOOL $output (optional) | TRUE (default): Generate XFBML initialization code directly, by inserting it into the HTML output<br><br>FALSE : return the initialization code (you must output it manually somewhere in the script) |

**Results:**

| | |
|---|---|
| STRING | empty string if $output = TRUE<br><br>XFBML initialization code if $output = FALSE |

**See also:**

http://developers.facebook.com/docs/reference/javascript/FB.XFBML.parse/

With the following code, which should appear in the first line of your HTML code, XFBML functionality is activated:

```
...
<!--Body begins -->
```

```
<body>
   <?php $FacebookAPP->EnableXFBML(); ?>
   <fb:like href="" send="true" width="450" show_faces="true" font=""></fb:like>
...
</body>
```

### 2.3.5 ForcePermissions

This method is the easiest and most effective way for prompting the user to grant some permissions.

Using this method, several permissions can be obtained at the same time, and it allows to acquire additional permissions amidst an application.

By calling this method, Facebook® starts a dialogue for the user to confirm the requested permissions. Afterwards, the user is taken to the very point in the application at which the dialogue was initiated. The return value is 1 if the user has granted the wanted permissions, or 0 if he rejected. Based on the return value, the application can decide what course should be taken and what the user is next presented with.

**Important**: Please consider the following limitations when using this function:
- This method may be used only once per PHP script file! (Multiple calls to ForcePermissions must be distributed over different scripts.)
- ForcePermissions should be at the beginning of a PHP script file, ideally right after creation of the PHPforFB instance. (Consider that, by calling this method, the same script file can be executed multiple times!)
- After calling the method, previously passed GET or POST arguments are no longer available! (To avoid mistakes due to this constraint, place the required parameters in session variables and read them from the session later, when needed.)

**Note:** Depending on the user's device type (desktop, mobile), ForcePermissions will automatically select a suitable kind of dialogue.

For more information and examples see the chapter "List of Access Rights".

**Invocation:**

```
ForcePermissions ( STRING | ARRAY $permissions )
```

**Parameters:**

| | |
|---|---|
| STRING \| ARRAY<br>$permissions | STRING : A single permission<br>(e.g. PHPforFB->ForcePermissions('user_videos') )<br><br>ARRAY : multiple permissions (e.g. PHPforFB->ForcePermissions( array('basic','publish_stream') ) ) |

**Results:**

| | |
|---|---|
| 1 | If the user has granted all requested permissions |
| 0 | If the user has declined(*) |

| | |
|---|---|
| | (*) Facebook® has established a new behaviour for permission requests recently. If the user denies an application's permission request, then if any permission is requested subsequently by the same application, the user is not offered the possibility to deny the request again, but only to accept it or to exit the application altogether. |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

Example usage of this method:

```
if(($res = $FacebookAPP->ForcePermissions(array('basic','public_stream'))) === FALSE){
   #Here an error has occurred
   echo "PHPforFB Error: ".$FacebookAPP->lastErrorCode." -> ".$FacebookAPP->lastError;
}else{
   if($res==0){
      #The user declined, code WITHOUT permissions follows
      #...
   }else{
      #The user accepted, code WITH permissions follows
      #...
   }
}
```

### 2.3.6 FQL

This method performs an FQL query to Facebook®.

With this method, a query to Facebook® servers can be performed using the Facebook Query Language (FQL). FQL is an interface similar to SQL. It allows for queries which are not supported by other interfaces, such as the Graph API.

The method takes care of all the details of communicating with the Facebook® servers, which can save a lot of work.

If at the time of the query there is an authenticated user with granted permissions, these will be taken into account and used for communication automatically.

**Invocation:**

```
FQL (STRING $query)
```

**Parameters:**

| | |
|---|---|
| STRING $query | string containing the FQL query |

**Returns:**

| | |
|---|---|
| ARRAY | values received from Facebook®, if the request was successful |

| | |
|---|---|
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

**See also:**

http://developers.facebook.com/docs/reference/fql/

### 2.3.7 GetAlbums

This method returns all existing information about a user's albums.

With this method, it is possible to query all photo albums, or to look up a specific one.

If you intend to upload a photo to an album, this method can be used to check for the existence of the album, and the album can then be created using PHPforFB->CreateAlbum (), should the need arise.

**Invocation:**

```
GetAlbums ( [STRING $album_name] [, STRING $album_id] [, STRING $from_id] [, STRING $userid] [, BOOLEAN $from_cache] )
```

**Parameters:**

| | |
|---|---|
| STRING $album_name (optional) | Constrains the request to an album of this name (Default = "" (all)) |
| STRING $album_id (optional) | Constrains the request to an album of this Id (Default = "" (all)) |
| STRING $from_id (optional) | Constrains the request to albums created by this user or application Id (Default = "" (all)) |
| STRING $userid (optional) | Query albums from an user with this Facebook® UserID (Default = current user) |
| BOOLEAN $from_cache (optional) | TRUE (Default) : Takes the result, if it exists, from the internal cache, otherwise sends the request to Facebook® unconditionally<br><br>FALSE : Ignore the cache, always query Facebook® servers |

**Returns:**

| | |
|---|---|
| ARRAY | Album information |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

**See also:**

http://developers.facebook.com/docs/reference/api/album/

A short example illustrating the use of this method:

```
$erg = $FacebookAPP->GetAlbums($FacebookAPP->appName);
if(!empty($erg)){
   //There exists an album for the current user, created by the application
   $alb_id = $erg[0]['id'];
}else{
   //Album does not exist, create it
   $alb_name = $FacebookAPP->appName;
   $alb_caption = 'all '.$FacebookAPP->appName.' pictures';
   $erg = $FacebookAPP->CreateAlbum($alb_name, $alb_caption);
   if($erg === FALSE){
      echo "Error: ".$FacebookAPP->lastErrorCode." -> ".$FacebookAPP->lastError;
      exit;
   }else{
      $alb_id = $erg;
      //Now upload pictures using $FacebookAPP->PostToAlbum() to album $alb_id
   }
}
```

### 2.3.8 GetAuthenticationURL

This method returns the URL needed for authentication.

To authenticate a user and to prompt her for granting certain permissions, an URL is needed that must be requested from Facebook®. This method returns this URL, which then may be integrated into a page, either for displaying a button, or for an automatic Javascript redirect.

**Important:** If the user clicks on the link that is generated from the result of this method, she will be redirected to Facebook®. This redirection must always take place on top of the browser window. If the user has granted the access permissions already, she will be directed to PHPforFB->okURL, or if she declines, to PHPforFB->cancelURL.

For more information and examples see the chapter "List of Access Rights".

**Invocation:**

GetAuthenticationURL ( [BOOLEAN $target] )

**Parameters:**

| | |
|---|---|
| BOOLEAN $target (optional) | Specifies if the link target information should be returned also (Default = TRUE) |

**Returns:**

| | |
|---|---|
| STRING | (if $target = FALSE) the authentication URL |

| | |
|---|---|
| ARRAY | (if $target = TRUE)<br>an array of three elements:<br>ARRAY[0] : Authentication URL<br>ARRAY[1] : Link target name<br>ARRAY[2] : Link target value |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

A short example illustrating the use of this method:

```
//Set the desired permission
$FacebookAPP->AddPermission('user_likes');
//Which page to call if the user accepts
$FacebookAPP->okURL = 'get_permission_ok.php';
//Which page to call if the user declines -
//should be directed to the application's profile page
$FacebookAPP->cancelURL = 'http://www.facebook.com/apps/application.php?id='
                          .$FacebookAPP->appID;
//fetch URL
$authURL = $FacebookAPP->GetAuthenticationURL();
//Output the link
echo '<a href="'.$authURL[0].'" target="'.$authURL[1].
     '">Now prompt for permissions and continue</a>';
```

### 2.3.9 GetClassState

This method serializes the current session's class state into a string.

By calling GetClassState, a copy of the current PHPforFB object with all its internal states, variables, and caches such as userData und userFriendsData can be obtained. The resulting string is base64 encoded. In addition to that, if the server supports it and has GZIP compression enabled, the resulting string will be compressed.

With this method it is possible to save the current class state in session space, and to restore the session later by calling PHPforFB->SetClassState().

This method is particularly useful if the user has granted the application the "offline_access" permission. In this case, the current class state can be saved in a database, a cron job can restore the class at scheduled intervals, and all permissions are readily available e.g. for posting to the user's dashboard using PHPforFB->PostToDashboard().

**Invocation:**

```
GetClassState()
```

**Returns:**

| | |
|---|---|
| STRING | The current class state, base64 encoded and possibly gz compressed |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

### 2.3.10 GetDashboard

This method returns the current user's dashboard entries.

This method always returns the newest entries. For this method to be allowed to read the dashboard, userAuthenticated must be TRUE and an userID must be present.

This method is about querying the user's dashboard ("UserFeed"), which is not to be confused with entries on his starting page. The entries on the starting page can be obtained with PHPforFB->GetHome().

**Invocation:**

```
GetDashboard ( [INT $count] )
```

**Parameters:**

| | |
|---|---|
| INT $count (optional) | Number of entries to fetch<br>(Default = 25) |

**Returns:**

| | |
|---|---|
| ARRAY | User's dashboard entries |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

### 2.3.11 GetFriendsInfo

This method returns information about the current user's friends.

This method allows to query friends of the current user. If desired and permitted individually by these friends, all the available information about these friends can be retrieved.

The only argument to this method is optional and can be used to limit the query to a user, or to a set of users. It is recommended that, if information about just a particular friend is needed, the $uids parameter is used.

By calling this function, the class variable userFriendsData is filled and its contents will subsequently be available everywhere in the scope of the current session.

**Invocation:**

```
GetFriendsInfo ( [STRING | ARRAY $uids] )
```

**Parameters:**

| | |
|---|---|
| STRING \| ARRAY $uids (optional) | STRING: friend's Facebook® UserID<br>ARRAY: A list of friends' Facebook® UserIDs<br>(Default = "" (all)) |

**Returns:**

| | |
|---|---|
| ARRAY | Associative array with all available(*) information about the queried friends, with the following format:<br><br>[FRIEND_USER_ID_1] => DATA,<br>[FRIEND_USER_ID_2] => DATA,<br>...<br><br>(*) Depending on what fields the users have filled in, their privacy settings and the available permissions |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

**See also:**

http://developers.facebook.com/docs/reference/rest/users.getInfo/

### 2.3.12 GetHome

This method returns the home entries (starting page contents) of the current user.

This method always returns the newest entries. For this method to be allowed to read the starting page contents, userAuthenticated must be TRUE and an userID be present.

This method is about querying the user's starting page ("UserHome"), which is not to be confused with entries on his feed / dashboard. The dashboard entries can be obtained with PHPforFB->GetDashboard().

**Invocation:**

```
GetHome ( [INT $count] )
```

**Parameters:**

| | |
|---|---|
| INT $count (optional) | Number of entries to fetch<br>(Default = 25) |

**Returns:**

| | |
|---|---|
| ARRAY | User's home entries |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

### 2.3.13 GetLikes

This method returns a user's 'likes', i.e. the entries which the user has expressed appreciation for by clicking the 'like' button.

This method returns all entries that the user has marked with the 'like' button, as well as entries shown under "Info" in his profile.

The likes returned can be limited to a certain category ("application" or "movie", for instance), or to a single object or name.

By calling this function, the class variable userLikes is filled and its contents will subsequently be available everywhere in the scope of the current session.

**Invocation:**

```
GetLikes ( [STRING $category] [, STRING $name] [, BOOLEAN $from_cache] )
```

**Parameters:**

| | |
|---|---|
| STRING $category (optional) | Constrains the query to a certain category ("application", "movie", "musician/band", "school", etc.) (Default = "" (all)) |
| STRING $name (optional) | Constrains the query to a certain object name (e.g. "Blade Runner" or "PHPforPages") (Default = "" (all)) |
| BOOLEAN $from_cache (optional) | TRUE (Default) : Takes the result, if it exists, from the internal cache, otherwise sends the request to Facebook®<br><br>FALSE : Ignores the cache, always queries Facebook® servers (slower execution) |

**Returns:**

| | |
|---|---|
| ARRAY | Elements of the type ARRAY with the following fields:<br>"category" (e.g. "Application", "Movie" etc.)<br>"name"<br>"created_time"<br>"id" (Facebook® Object ID) |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

This is how you can determine if a user has 'liked' your application already:

```
$erg = $FacebookAPP->GetLikes('Application',$FacebookAPP->appName);
if(empty($erg)){
   #User did not like the application yet
}
else{
   #User liked the application
}
```

### 2.3.14 GetLocationInfo

This method queries information about a particular location registered at Facebook®.

This function requires a valid Facebook® Location Id. This Id can be obtained e.g. by calling PHPforFB->GetUserInfo("","location").

**Invocation:**

```
GetLocationInfo ( STRING $locationID )
```

**Parameters:**

| | |
|---|---|
| STRING $locationID | The Facebook® Location Id of the wanted location |

**Returns:**

| | |
|---|---|
| ARRAY | All available information about the specified location |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

### 2.3.15 GetObjectInfo

This method returns information about any Facebook® Id.

Everything in Facebook® is an object and denoted by an Id. Different sets of fields are provided in the result, depending on the type of the given object.

**Invocation:**

```
GetObjectInfo ( STRING $objectID )
```

**Parameters:**

| | |
|---|---|
| STRING $objectID | A valid Facebook® Id |

**Returns:**

| | |
|---|---|
| ARRAY | All available information about the specified object |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

Example usage:

```php
//after a successful authentication
$likes_var = $FacebookAPP->GetLikes('Restaurant/cafe','');
//query all restaurants that the user likes
if(!empty($likes_var)){
   //Query additional information about the first restaurant
   $likes_var[0]["additional"] = $FacebookAPP->GetObjectInfo( $likes_var[0]["id"] );
}
```

### 2.3.16 GetPageInfo

This method returns all available information about a Facebook® fan page. The parameter $page must denote a valid page Id.

**Invocation:**

```
GetPageInfo ( STRING $pageID )
```

**Parameters:**

| | |
|---|---|
| STRING $pageID | Facebook® page Id |

**Returns:**

| | |
|---|---|
| ARRAY | All available information about the specified page |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

### 2.3.17 GetPermissions

This method queries all permissions that the user has granted.

PHPforFB allows to acquire permissions stepwise, and amidst an application. Also, it is possible that after several visits to an application, the user is prompted to grant additional permissions. With this method, the current state of granted permissions can be determined.

There are two variants. The first variant (Parameter $globally = FALSE) returns the already granted permissions in the current session, but it does not return the permissions granted during earlier visits to the application.

The second variant (Parameter $globally = TRUE) returns all permissions that the user has granted during any visit to the application, and which are still valid.

**Note:** The first variant (permissions granted in the current session) is executed very quickly. The second variant can take up to two seconds and slow down the application considerably.

For more information and examples see the chapter "List of Access Rights".

**Invocation:**

```
GetPermissions ( BOOLEAN $globally )
```

**Parameters:**

| | |
|---|---|
| BOOLEAN $globally | TRUE : return all permissions granted during any visit to this application |

| | |
|---|---|
| | FALSE (Default) : return only permissions granted during this session |

**Returns:**

| | |
|---|---|
| ARRAY | Denominators of the respective permissions |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

### 2.3.18 GetSignedRequest

This method converts a Facebook® "signed_request" (which is part of the data communication protocol) into an associative array.

**Invocation:**

```
GetSignedRequest ( [STRING $signed_request] )
```

**Parameters:**

| | |
|---|---|
| STRING $signed_request (optional) | Content of the "signed_request" parameter sent by Facebook®. If omitted, the most recent "signed_request" is used. |

**Returns:**

| | |
|---|---|
| ARRAY | Keys and associated values in signed_request |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

### 2.3.19 GetUserInfo

This method returns information about an user.

With this method it is possible to query all available information about a user. The set of fields returned depends on the permissions granted to the application by the user (see "Access Rights").

By calling this method, the class variable userData is getting filled (if $userid="" oder $userid = PHPforFB->userID), and its contents will subsequently be available everywhere in the scope of the current session.

**Invocation:**

```
GetUserInfo ( [STRING $userid] [, ARRAY $params] [, BOOLEAN $from_cache] )
```

**Parameters:**

| | |
|---|---|
| STRING $userid (optional) | Facebook® UserID |

| | |
|---|---|
| | (Default = "" (current user)) |
| STRING \| ARRAY $params (optional) | String or array containing the wanted fields, e.g. "name" or "birthdate"<br>(Default = "" (all)) |
| BOOLEAN $from_cache (optional) | TRUE (Default) : Takes the result, if it exists, from the internal cache, otherwise sends the request to Facebook® unconditionally<br><br>FALSE : Ignore cache, always query Facebook® servers (slower execution) |

**Returns:**

| | |
|---|---|
| ARRAY | • Field: ["picture_is_symbol"]<br>TRUE, if the profile picture is just an avatar, otherwise FALSE (if the user has uploaded a picture)<br><br>• Field: ["picture"]<br>STRING with an URL of the user's profile picture<br><br>• Field: ["pictures"]<br>An array of URLs of a profile picture's different sizes. The array contains the following fields: ["small"], ["normal"], ["big"]<br><br>• More fields ...<br>... depending on which fields the user has filled in, the user's privacy settings, and which permissions the user has granted. |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

**See also:**

http://developers.facebook.com/docs/reference/api/user/

A short example illustrating the use of this method:

```
if(empty($FacebookAPP->userData)){
   $userD = $FacebookAPP->GetUserInfo('',array('name','gender'));
}
$userGender = $userD['name'];
$userGender = $userD['gender'];
//[...]
$userLink = $FacebookAPP->userData['link'];
```

### 2.3.20 GraphAPI

This method sends a Graph API request to Facebook®.

Using this method, Facebook®'s Graph API can be accessed directly. The Graph API is the

main programming interface, and it is being extended continuously.

The method takes care of the complete communication with Facebook® servers, which can save a lot of work.

If at the time of the query there is an authenticated user with granted permissions, these will be taken into account and used for communication via the Graph API.

**Invocation:**

```
GraphAPI ( STRING $api_string [, STRING $parameters] [, BOOLEAN $header] [, BOOLEAN $post] )
```

**Parameters:**

| | |
|---|---|
| STRING $api_string | Graph API request |
| STRING \| ARRAY $data (optional) | Additional arguments to be passed to the Facebook® Graph API<br><br>$post = FALSE: STRING of an additional argument, e.g. "limit=25"<br><br>$post = TRUE: ARRAY of fields to be submitted via POST<br><br>(Default = "") |
| BOOLEAN $header (optional) | FALSE : The request should return all available data<br><br>TRUE : Only the request's HTTP header should be returned<br><br>(Default = FALSE) |
| BOOLEAN $post (optional) | FALSE : Data is to be retrieved via Facebook®'s Graph API<br><br>TRUE : Data is to be sent to the Graph API via POST<br><br>(Default = FALSE) |

**Returns:**

| | |
|---|---|
| ARRAY | $header = FALSE<br>The result of the query is an associative array with all fields received by the Graph API |
| STRING | $header = TRUE<br>HTTP header resulting from the query |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

**See also:**

http://developers.facebook.com/docs/reference/api/

### 2.3.21 KillIframeBorder

This method turns off the Facebook® border of an application or fan page.

All applications are embedded into Facebook® using an Iframe. But sometimes it may be more desirable to evade that border and to display the application in a separate window. One possibility is to display a button on the first page of the application that, if clicked, opens the new window. This can be achieved with **PHPforFB**'s runOutOfIframe variable.

It may be more elegant, however, to disable the border immediately at startup of the application. This can be useful, for instance, if the user is accessing the application from a mobile device. In this case, more space on the display can be retained for the application (see example below).

It is also possible to disable the border on actions of the user inside the application.

This method allows to disable the Facebook® border at any point in the application, and to jump to a specific URL.

**Invocation:**

KillIframeBorder ( [STRING $target] )

**Parameters:**

| | |
|---|---|
| STRING $target (optional) | Absolute or relative URL to jump to<br><br>"" : if no $target is specified, the current URL will be requested again, without the Facebook® border. GET parameters will be preserved, but POST parameters will be lost.<br><br>(Default = "") |

**Returns:**

| | |
|---|---|
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

This example illustrates the use of this method:

```
//user is currently using a mobile device and the application has a border?
if($FacebookAPP->isMobileDevice === TRUE && $FacebookAPP->runOutofIframe === FALSE){
   //yes! deactivate border and reload the page
   if($FacebookAPP->KillIframeBorder() === FALSE)
      echo "PHPforFB Error: ".$FacebookAPP->lastErrorCode;
      echo " -> ".$FacebookAPP->lastError;
   exit;
}else{
   //no, continue
   //...
```

```
}
```

### 2.3.22 PermissionAvailable

This method checks if the current user has already granted a certain Facebook® permission.

PHPforFB allows to acquire permissions not only all at once, but also stepwise, and amidst an application. Also, it is possible that after several visits to an application the user is prompted to grant additional permissions. This method allows to check for the presence of a certain permission, so that it can be requested using e.g. PHPforFB->ForcePermissions() if need be.

This method returns 1 if the user has granted the specified permission, otherwise 0.

**Note:** PermissionAvailable can only check for permissions that have been requested and granted during the running session. For a global examination of a user's permissions, PHPforFB->GetPermissions(TRUE) must be called prior to this function.

For more information and examples see the chapter "List of Access Rights".

**Invocation:**

```
PermissionAvailable ( STRING $permissionName )
```

**Parameters:**

| | |
|---|---|
| STRING $permissionName | Denominates the wanted permission, e.g. "email" |

**Returns:**

| | |
|---|---|
| 1 | If the user has granted the permission |
| 0 | If the user has not yet granted the permission |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

### 2.3.23 PostLinkToDashboard

This method sends a link to the user's dashboard.

**Invocation:**

```
PostLinkToDashboard ( STRING $link [, ARRAY $action] [, STRING $privacy] )
```

**Parameters:**

| | |
|---|---|
| STRING $link | Link to be posted (starting with "http://") |
| ARRAY $action (optional) | Optional link below the entry, next to "comment" (see |

| | |
|---|---|
| | Illustration, element no. 6)<br><br>An associative array with the following elements:<br>"name" : Label of the link (Linktext)<br>"link": URL being pointed to by the link<br><br>(Default = "" (no link)) |
| STRING $privacy<br>(optional) | Specifies who shall be able to see the link.<br>Possible values are "EVERYONE", "ALL_FRIENDS", "NETWORKS_FRIENDS", "FRIENDS_OF_FRIENDS", "CUSTOM", etc.<br><br>(Default = "" ("EVERYONE")) |

**Returns:**

| | |
|---|---|
| STRING | Object Id of the newly created entry |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

**See also:**

http://developers.facebook.com/docs/reference/api/post/

### 2.3.24 PostToAlbum

This method loads and stores a pictures in a photo album of the current user.

With this method, an application can post any picture into a particular album.

An authenticated user and the permission "publish_stream" is required.

**Note:** Using the method PHPforFB->GetAlbums(), all Ids of a user's albums can be retrieved if the picture is to be stored in an existing album. With PHPforFB->CreateAlbum() a new album can be created in the current user's gallery, for posting a picture into it using PostToAlbum.

**Invocation:**

```
PostToAlbum (ARRAY $pictureData [, INT $noDashboard] )
```

**Parameters:**

| | |
|---|---|
| ARRAY $pictureData | An associative array with the following values:<br><br>• "picfile" : **Absolute path** or **URL** of the picture<br>• "caption" : Picture title / caption<br>• "album_id" : Album Id (optional)<br>(Default = APP-Album) |

| | |
|---|---|
| INT $noDashboard (optional) | 0: no, the posted picture should generate a notice or an entry on the user's dashboard<br><br>1: yes, the posted picture should not generate an entry on the dashboard<br><br>(Default = 0) |

**Returns:**

| | |
|---|---|
| STRING | Object Id of the newly created picture |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

**See also:**

http://developers.facebook.com/docs/reference/api/photo/

This example illustrates the use of this method:

```
$erg = $FacebookAPP->PostToAlbum(array(
   'album_id' => $alb_id,
   'caption' => 'My '.$FacebookAPP->appName.' picture from ('.Date('d.m.Y',time()).')',
   'picfile' => 'http://www.app_todo.de/images/image.jpg',
));
if($erg === FALSE){
   # error occurred
   echo "Error: ".$FacebookAPP->lastErrorCode." -> ".$FacebookAPP->lastError;
   exit;
}
else{
   # Done successfully here
}
```

## 2.3.25 PostToDashboard

This method creates an entry on the current user's dashboard.

Dashboard entries can be very important and informative for an application. With this method, entries of any complexity can be generated.

**Note:** An application can post entries to a dashboard only if the "publish_stream" permission is available.

**Warning:** If special characters exist in the $arrPosting fields, then they must be converted using the PHP function HtmlEntities()!

**Invocation:**

```
PostToDashboard ( ARRAY $arrPosting )
```

**Parameters:**

| | |
|---|---|
| ARRAY $arrPosting | An associative array with the entry's contents.<br>The following elements are possible (see illustration below):<br>• "message"<br>• "link"*<br>• "name"*<br>• "description"*<br>• "properties"<br>• "picture"<br>• "actions"<br>• "privacy"<br>(*required elements) |

**Results:**

| | |
|---|---|
| STRING | Object Id of the newly created entry |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

**Description:**


Elements of a dashboard post and in the $arrPosting array:

1. The name of the user of your application, this cannot be changed

2. "message", the message of this entry

3. "name", name of the application, automatically supplemented with a link to the URL supplied with the "link" parameter. The link argument may also contain GET arguments.

4. "caption", additional short text for the link/name above (see 3.).

5. "description", here you can supply a descriptive text (max. 1000 characters), of which the first 300 characters will be displayed.

6. "action", this section will be added to each dashboard entry automatically. Additionally, a link supplied with the "action" parameter will be appended there, with the name taken from the "name" element and the link taken from the "link" element. As can be seen, the name should be short, as it is may get truncated.

7. "picture", the absolute URL of the entry's picture. Facebook fetches this picture

in realtime from its proxy and scales it accordingly. The picture pointed to by this URL should always be available, otherwise it may appear invisible in the dashboard entry. Only pictures in JPEG format are accepted. It should be optimized to 90x90 pixels.

8. "properties", additional links and information related to the entry. This is intended for listing additional properties. It must be an associative array of this form:

```
"properties" => array(
   "Release Date" => "21.8.2011",
   "More Info" => array(
      "text" => "Click here",
      "href" => "http://www.phpforfb.com"
      )
)
```

9. In addition to that, there is the array element "privacy". With this you can decide who is able to see the respective entry. Possible values are "EVERYONE", "ALL_FRIENDS", "NETWORKS_FRIENDS", "FRIENDS_OF_FRIENDS", "CUSTOM"

**See also:**

http://developers.facebook.com/docs/reference/api/post/

A short example illustrating the use of this method:

```
//Create array for the entry
$eintrag = array(
   'message' => 'This is the message sent',
   'link' => $FacebookAPP->appFBURL.'/dashboard.php?p=parameter',
   'name' => APP_TITLE .' - name',
   'caption' => APP_TITLE.' - caption',
   'picture' => 'http://app_todo.de/images/image.jpg',
   'description' => 'description',
   'actions' => array('name' => APP_TITLE.' - action',
                      'link' => $FacebookAPP->appFBURL.'?action=1'),
   'privacy' => array('value'=>'EVERYONE')
);
$erg = $FacebookAPP->PostToDashboard($eintrag);
if($erg === FALSE){
   echo "PHPforFB Error: ".$FacebookAPP->lastErrorCode." -> ".$FacebookAPP->lastError;
   exit;
}
else{
   //entry created successfully, continue application
   //...
}
```

## 2.3.26 RenderHTMLHEADInformation

This method creates Facebook® specific HTML meta tag header lines for the desired URL.

With this method it is possible to integrate one's website contents into Facebook®'s Open Graph Protocol. This means that, if the site's URL is posted or inserted somewhere in Facebook®, Facebook® will display these meta data.

In doing so, a homepage can be associated with a recommendation or a dashboard posting. It is also possible to supply the application with additional information and a thumbnail.

**Invocation:**

RenderHTMLHEADInformation ( ARRAY $metaData )

**Parameters:**

| | |
|---|---|
| ARRAY $metaData | An associative array with the following values:<br>• "title"<br>• "type"<br>• "url"<br>• "image"<br>• "site_name"<br>• "admins"<br>• "description"<br><br>(all elements are optional) |

**Returns:**

| | |
|---|---|
| STRING | HTML meta tags |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

**See also:**

http://developers.facebook.com/docs/opengraph/

Here is a short example:

```
<html>
   <head>
      <title><?=$FacebookAPP->appName;?></title>
<?php
echo $FacebookAPP->RenderHTMLHEADInformation(
array(
   "title" => $FacebookAPP->appName." - title addition",
   "url" => $FacebookAPP->appFBURL,
   "image" => 'http://www.myapp_todo.de/images/image.jpg',
   "description" => APP_TITLE.' : A short description',
   "type" => "website",
));
?>
   </head>
....
```

## 2.3.27 SearchSite

This method allows to search Facebook®.

All public elements are searched for the given term. The found entries are returned in an array.

**Invocation:**

```
SearchSite ( STRING $keyword, STRING $type [, INT $count] [, INT $mode] )
```

**Parameters:**

| | |
|---|---|
| STRING $keyword | Search term or words (separated by spaces), or a word fragment (min. length is two characters) |
| STRING $type | Type of objects to be searched, possible values:<br><br>• "post"<br>• "application"<br>• "page"<br>• "event"<br>• "group"<br>• "user" (requires authentication)<br>• "checkin" (requires authentication) |
| INT $count (optional) | Number of entries to fetch<br>(Default = 25) |
| INT $mode (optional) | 0 : all contents, not just the ones by the current user<br>1 : only the current user's contents (userAuthenticated = TRUE required)<br><br>(Default = 0) |

**Returns:**

| | |
|---|---|
| ARRAY | Search result as an associative array with all fields received |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

## 2.3.28 SearchUserFeed

With this method, the current user's dashboard can be searched.

This method searches all entries for the given search term and returns the result in an array.

**Note:** This method requires an authenticated user und the granted permission "read_stream".

**Invocation:**

```
SearchUserFeed ( STRING $keyword, [, INT $count] )
```

**Parameters:**

| | |
|---|---|
| STRING $keyword | Search term or words (separated by spaces), or a word fragment (min. length is two characters) |
| INT $count (optional) | Number of entries to fetch<br>(Default = 25) |

**Returns:**

| | |
|---|---|
| ARRAY | Search result as an associative array with all fields received |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

### 2.3.29 SetClassState

This method restores a previously stored class object.

This method allows to restore a class instance, which had been serialized with PHPforFB->GetClassState(), and was possibly stored in (and has just been retrieved from) a database. The restored object has the same state as of the point in time when it was serialized, including all variables and cached data, such as userData, userFriendsData.

This method is particularly useful if the application was granted the "offline_access" permission by the user. In this case, the current class state can be saved in a database, a cron job can restore the class at scheduled intervals, and all permissions are readily available for posting to the user's dashboard using PHPforFB->PostToDashboard(), for example.

**Invocation:**

```
SetClassState ( STRING classstate_string )
```

**Parameters:**

| | |
|---|---|
| STRING $classstate_string | Class state returned by PHPforFB->GetClassState() |

**Results:**

| | |
|---|---|
| TRUE | The class instance has been restored successfully and is ready for subsequent use |
| FALSE | If an error occurred; the error is specified by the variables lastError and lastErrorCode |

# Part III

*Appendix*

# 3 Appendix

## 3.1 PHPforFB Conventions

To ease the work with the framework, the following naming conventions apply:

**1. Variables / Properties**

Names of properties and variables begins with a lowercase letter, and each following word begins with an uppercase letter.

E.g. **PHPforFB**->lastErrorCode

**2. Methods**

All **PHPforFB** method names begin with an uppercase letter, and the same applies to the following words.

E.g. **PHPforPage**->GetObjectInfo()

## 3.2 Error codes and messages

All methods and functions of the **PHPforFB** framework return FALSE in the case of an error.

**PHPforFB** errors consist of an error code and a descriptive message.
The error code denotes the type of the error. The message describes the error in more detail and informs about the cause of the problem.

After a method has returned FALSE, the error code and message can be found in the object variables lastErrorCode und lastError.

```
if($PHPforFB->METHODE() === FALSE){
  echo 'Error code: '.$PHPforFB->lastErrorCode.'<p>';
  echo 'Error message: '.$PHPforFB->lastError.'<p>';
}
```

**Note:** These variables always contain the error that occurred last. If another error occurs during the program flow that follows, these variables are being overwritten and the previous error is no longer available.

**Important:** The constructor is an exception to this rule. It does not necessarily return FALSE if an error occurs during creation. After creation with "new", the lastErrorCode must be checked against greater 0 to recognize an error.

```
$FacebookAPP = new PHPforFB($structInit);
if($FacebookAPP->lastErrorCode>0){
   # An error occurred during creation
   echo "PHPforFB Error: ".$FacebookAPP->lastErrorCode." -> ".$FacebookAPP->lastError;
   exit;
}
```

## 3.3 Access rights - list of known permissions

See below a list of named permissions, which are currently supported by the **PHPforFB** framework:

| | |
|---|---|
| "basic" | Access to basic permissions, UserID, UserName, etc. |
| "user_about_me"<br>"friends_about_me" | Access to the "Info" text in the profile (user / user's friends) |
| "user_activities"<br>"friends_activities" | Access to activities and interest in the profile (user / user's friends) |
| "user_birthday"<br>"friends_birthday" | Access to the user's birthdate<br><br>The format is the American date, i.e. month/day/year. Conversion into the German date format:<br>`$strg = $FacebookAPP->userData['birthdate'];`<br>`$german_date = Date('d.m.Y',$strg);`<br><br>**Note:** If the user grants this permission, the application can access the birthday regardless of the user's respective privacy setting |
| "user_education_history"<br>"friends_education_history" | Access to education, occupation in the profile (user / user's friends) |
| "user_hometown"<br>"friends_hometown" | Access to the user's residence in the profile (user / user's friends) |
| "user_likes"<br>"friends_likes" | Access to the user's liked objects (user / user's friends) |
| "user_religion_politics"<br>"friends_religion_politics" | Access to the user's religious and political views (user / user's friends) |
| "user_status"<br>"friends_status" | Access to the user's last status message (user / user's friends) |
| "user_photos"<br>"friends_photos" | Access to the user's photos (user / user's friends) |
| "user_videos"<br>"friends_videos" | Access to the videos uploaded by the user (user / user's friends) |
| "user_website"<br>"friends_website" | Access to the user's website (user / user's friends) |
| "email" | Access to the user's email address |
| "read_friendlists" | Allows reading the user's list of friends |
| "read_stream" | Allows reading the user's dashboard entries |

| | |
|---|---|
| "user_checkins" | Access to the user's last location check-in (siehe http://developers.facebook.com/docs/reference/api/checkin/) |
| "publish_stream" | Permits the application to post links, dashboard entries, etc. |
| "offline_access" | Allows access to user data even if the user is currently offline - needed e.g. for regular dashboard entries |
| "publish_checkins" | Allows an application to login in the name of the user |

**Warning**: The **PHPforFB** framework does not support all of Facebook®'s existing permissions. If you want to use the unsupported (or just recently added) permissions, you should add these manually, so that they are available to the **PHPforFB** methods.

**Listing of all existing Facebook® permissions:**

http://developers.facebook.com/docs/reference/api/permissions/

**PHPforFB**
**A Professional PHP Framework for Facebook®**

**http://www.PHPforFB.com**



**Writen by: Rezar Behzad**
**Co-Programmer: Javad Ghodrati**