TT Like & Follow Wordpress Plugin

The ThemeTailors Like & Follow Wordpress Plugin (or WPLF) brings you the chance to **transform your WP multisite into a Social Network**.

It include 2 features:

- Users registered into your WP Multisite can “Like“ the posts of other registered users;
- Registered users can “Follow“ the sites of the multisite’s network.

ATTENTION!

This plugin is meant to work **only** with Multisite Configuration.
Before continuing to the installation, you must have configured your Wordpress site to have enabled the Multisite feature.

Please, refer to this guide for Multisite configuration:

http://codex.wordpress.org/Create_A_Network

Instruction

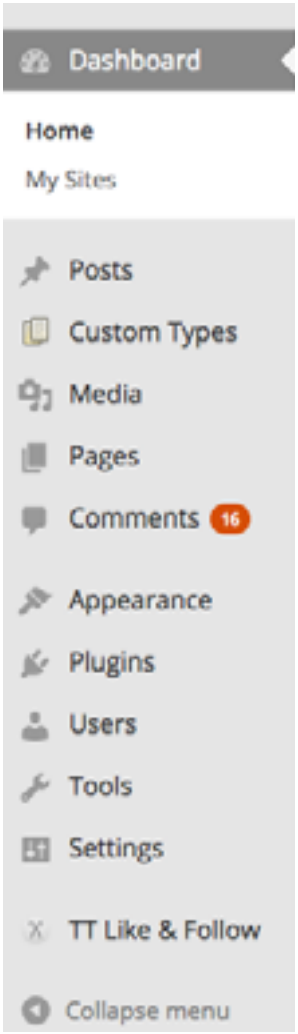
This plugin needs to be activaded on the Network Admin Dashboard, to have it active on all the sites of the network.

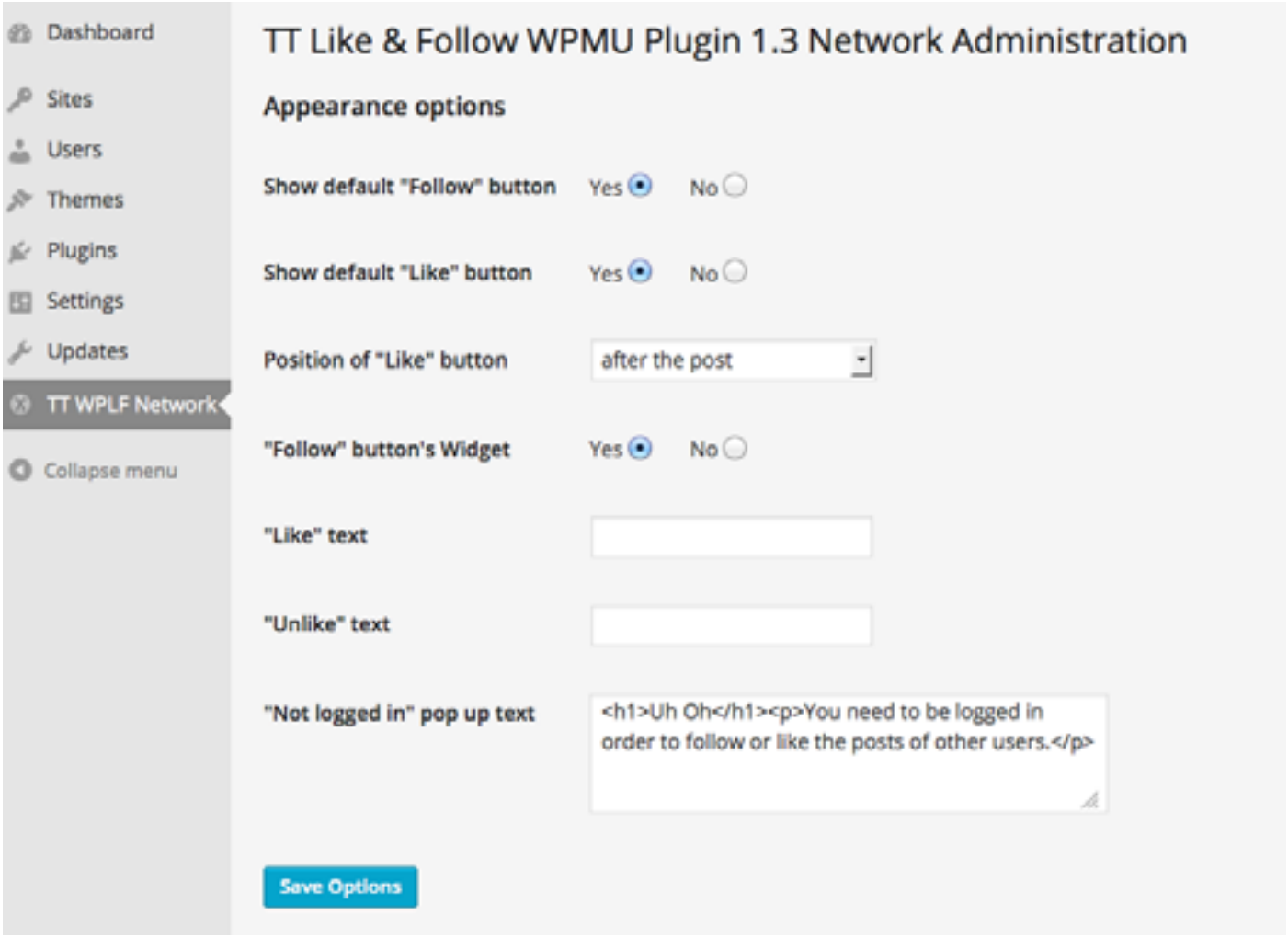
After the activation, a new section will be added in the network admin dashboard and in the site admin dashboard.

The network admin’s section of the plugin has some basic appearance options:

- **Show default “Follow” button**:
  Here you can choose to show the default “Follow” button on the sites. If you choose “No”, a code will be given to be pasted into the code of the theme.
Default value is “*Yes*”.

- **Show default “Like” button**:
  Here you can choose to show “Like” button on the posts. If you choose “No”, a code will be given to be pasted into the code of the theme. If “Yes” is selected the following option will appear.
Defaut value is “*Yes*”.





- **Position of “Like” button**:
  Here you choose the position of the “Like” button inside the post. You have 3 options:
    - Before the post
    - After the post
    - Before and after the post

Default value is “*after the post*”.

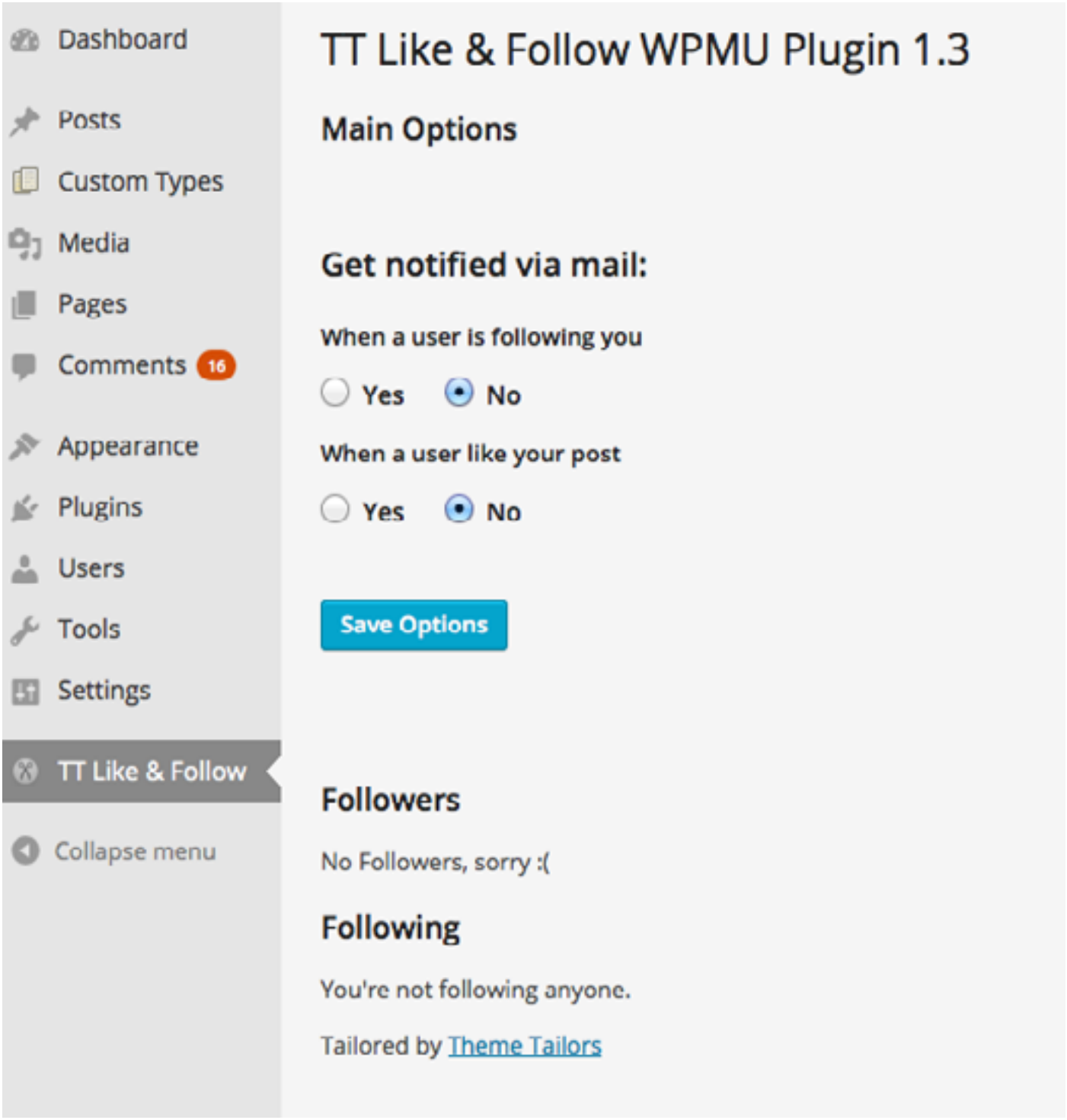
- **“Follow” button’s Widget**:
  You can choose to add a widget that will be inserted in the sidebar.
Default Value is “*Yes*”.

- **“Like” text**:
  It’s the text that will be displayed on the “Like“ buttons in the posts of your Network.
Default value is “*Like*”.

- **“Unlike” text**:
  It’s the text that will be displayed after the click on the “Like” button.
Defaul value is “*Unlike*”.

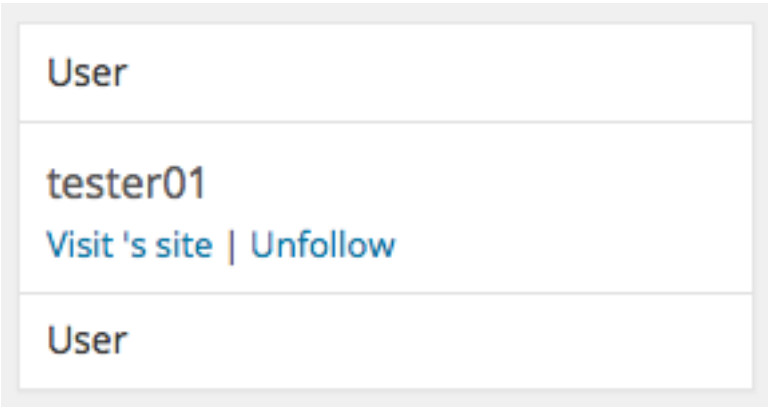
- **“Not logged in” pop up text**:
  That’s the text that will appear if a user clicks on the “Follow” and the “Like” buttons and isn’t logged in. You can use HTML tags to typesetting the text.
Default value is “*<h1>Uh Oh</h1> You need to be logged in order to follow or like the posts of other users. *”

The site admin's section contains email notification's options and two list, one for who's follow you and one for the users you're following.

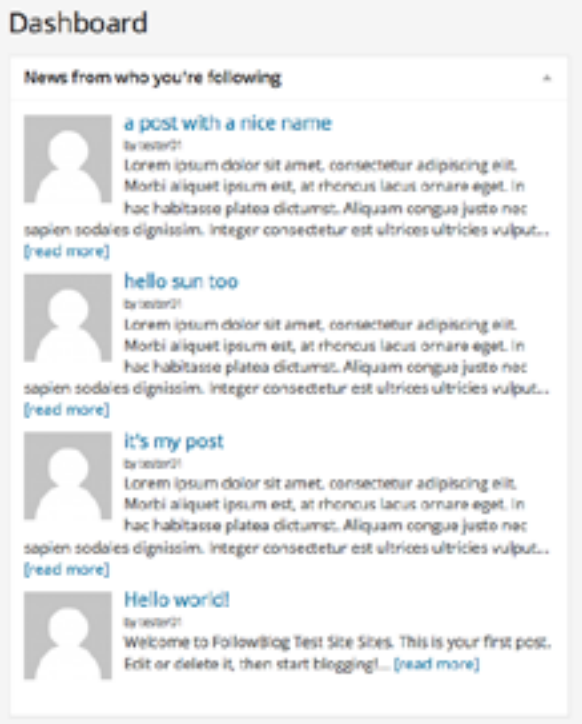


The lists shows the name of the user, a link to the user's site and the possibility to follow (if is one of your follower) or unfollow (if it's in the following list).

After you click on "Follow/Unfollow" the user will remains on the list until the refreshing of the browser.



In your admin site's dashboard you can see **the latest posts** from the people that you're following.



The panel will display the latest 4 posts, from every followed user, in chronological order.

Every post will show the linked title of the post, the author's name, an excerpt of the post and the author's avatar.

Using TT WPLF in your own theme

Like & Follow Wordpress Plugin comes with a handful sets of functions that you can insert in your theme.

These functions are called by a class that can be extended and improved.

To create the instance of the class:

```
$foo = new WPLF( userID , blogID );
```

userID (integer)
The ID of a desired user. If the value is null, the current user ID will be used with the help of the WP function `get_current_user_id()`.

Default value: *null*

blogID (integer)
The ID of a desired blog, If the value in null, the current blog ID will be used with the help of the global variable $blog_id.

Default value: *null*

Here's a list of the functions inside the WPLF class:

followButton

As its name says, this function creates the "Follow" button.
You can call it everywhere in the code like this:

```
echo $foo->followButton( class );
```

class (string)
custom classnames that you can insert in the button to modify its default style.

Default value: *null*

Return value: a string that contains the formatted HTML "Follow" button.

likeButton

this function creates the “Like” button.

```
    echo $foo->likeButton( args );
```

args (mixed values)
The function uses the same pattern used in Wordpress to define the multiple variables inside. You can use arrays or a string to set the following variables:

blogID (integer)
The ID of the blog you want to target the button.

Default value: *the blogID set when the class’s instance is created*

postID (integer)
The ID of the post you want to target.

Default value: *$post->ID from global variable $post.* ***Remember****, if this value is set to default, you must use the function inside the loop.*

class (string)
The custom classnames that you can insert in the button to modify its style.

Defaul value: *null*

table (string)
It refers to the table located in the WP database. It targets a specified table that contain the target post.
If you’re using a custom type post, you don’t need to modify this parameter.

Default value: *‘posts’*

Return value: a string that contains the formatted HTML “Like” button.

insertFollower

This function insert the selected user into the “following” list of a target user. It’s the function that is called when clicking on the “Follow” button.

```
    $foo->insertFollower( follower );
```

follower (integer)
The follower’s ID. There’s no default value, this variable is mandatory.

Return value: an associative array that contains the following values:
‘text’ = “Unfollow”
‘check’ = if the insertion fails, the value will be *false*
‘fn’ = “unfollow”

insertLike

Insert a target post into the like list. It's the function called after you press on the "Like" button.

```
$foo->insertLike( comingString );
```

comingString (string)
The string that's coming from the "Like" button. It's formed as follows:

```
blog_id-post_id-table
```

The string is split by the function and processed.

Return value: an associative array with the following values:
'text' = the value set in the admin backend
'check' = if the insertion fails, the value will be *false*
'fn' = "unlike"

deleteFollower

Remove a selected user from the "following" list from the target user. It's called when you press the "Unfollow" button.

```
$foo->deleteFollower( follower );
```

follower (string)
The user's slug name that will be removed from the list. It's a mandatory variable.

Return value: an associative array with the following values:
'text' = "Follow [USERNAME]"
'check' = if the deletion fails, the value will be *false*
'fn' = "follow"

deleteLike

Remove a selected post (or a custom type) from the "Like" list of the target user. It's the function called when you press on the "Unlike" button.

```
$foo->deleteLike( deleteString );
```

deleteString (string)
The string that's coming from the "Unlike" button. Like the **insertLike** function, it follows the same pattern:

```
blog_id-post_id-table
```

Return value: an associative array with the following values:
'text' = the value set in the admin backend
'check' = if the deletion fails, the value will be *false*
'fn' = "follow"

checkFollow

This function checks if the user is already followed.

```
    $foo->checkFollow( follower_id );
```

follower_id (integer)
The ID of the target user. This variable is mandatory

Return value: (boolean) if the user is already followed, the function will return *true*

checkLike

Checks if the post is already liked by a target user.

```
    $foo->checkLike( likeString );
```

likeString (string)
String that will be processed by the function. Like **insertLike** and **deleteLike** functions, it uses the same pattern:

```
    blog_id-post_id-table
```

Return value: (boolean) if the post is altready liked, the function will return *true*

getFollowers

Return an array that contains all the users that follows the target user.

```
    $foo->getFollowers( args );
```

args (mixed values)
the arguments used for the pagination and to define the output type. You can insert the values as an associative array or as a query string, just like the function get_results in the wpdb class.

page (integer)
The current page defined fro the pagination.

Default value: *1*

limit (integer)
The number of results that will be displayed per page.

Default value: *false*

output_type (string)
Sets the type of the resulting array. To set a different type of array, check the wpdb class on the Wordpress Codex Page.

Default value: *OBJECT*

Return value: An array with the result of the query.

getLikes

Return an array that contains the likes got from the target user.

```
    $foo->getLikes( args );
```

args (mixed values)
The arguments used to filter and paginate the results. You can insert the values as an associative array or as a query string, just like the function get_results in the wpdb class.

blogID (integer)
The ID of a target blog in the network. if the value is false the function will return all the likes of the target user.

Default value: *false*

postID (integer)
The ID of the post. To target a post, you must also enter the blogID.

Default value: *false*

page (integer)
The current page defined for pagination.

Default value: *1*

limit (integer)
The number of results that will be displayed per page.

Default value: *false*

table (string)
It refers to the table located in the WP database. It targets a specified table that contain the target post.

Default value: *'posts'*

output_type (string)
Sets the type of the resulting array. To set a different type of array, check the wpdb class on the Wordpress Codex Page.

Default value: *'OBJECT'*

exclude (boolean)
You can exclude the target user from the results. Setting *true* this parameter, the query will return only the likes that other user gave to the selected post.

Default value: *false*

Return value: an array with the result of the query.

getFollowing

Return the users that the target user is following.

```
    $foo->getFollowing( args );
```

args (mixed values)
Like the “getFollowers” function, the args used in this function are for the pagination and to define the output type on the resulting query.

page (integer)
The current page defined fro the pagination.

Default value: *1*

limit (integer)
The number of results that will be displayed per page.

Default value: *false*

output_type (string)
Sets the type of the resulting array. To set a different type of array, check the wpdb class on the Wordpress Codex Page.

Default value: *OBJECT*

Return value: an array with the result of the query.

getLiked

Return the post liked from the target user.

```
    $foo->getLiked( args );
```

args (mixed values)
You can insert the values as an associative array or as a query string, just like the function get_results in the wpdb class.

blogID (integer)
The ID of a target blog in the network. if the value is false the function will return all the likes of the target user.

Default value: *false*

page (integer)
The current page defined for pagination.

Default value: *1*

limit (integer)
The number of results that will be displayed per page.

Default value: *false*

table (string)
It refers to the table located in the WP database. It targets a specified table that contain the target post.

Default value: *'posts'*

output_type (string)
Sets the type of the resulting array. To set a different type of array, check the wpdb class on the Wordpress Codex Page.

Default value: *'OBJECT'*

Return value: an array with the result of the query.

getFollowingPosts

Return the posts from the users followed from the target user.
This function is compatible with the WPMUDEV Post Indexer plugin. If this plugin is installed, the function will use its class to get the posts from the index generated.

```
    $foo->getFollowingPosts( page , limit );
```

page (integer)
The current page defined for pagination.

Default value: *1*

limit (integer)
The number of results that will be displayed per page.

Default value: *10*

Return value: an array filled with stdClass object arrays. The stdClass object contains the following data:

BLOG_ID
ID
post_author
post_author_name
post_date
post_date_gmt
post_content
post_title
post_excerpt
post_status
comment_status
ping_status
post_password
post_name
to_ping
pinged
post_modified
post_modified_gmt
post_content_filtered
post_parent
guid
permalink
menu_order
post_type
post_mime_type
comment_count

getActivity

Returns the follows and the likes that the target user receive. The results are sorted in chronological order.
This function uses **getLikes** and **getFollowers** to gather the data.

```
    $foo->getActivity( page , limit );
```

page (integer)
The current page defined for pagination.

Default value: *1*

limit (integer)
The number of results that will be displayed per page.

Default value: *false*

Return value: an object array.

saveUserOptions

Saves wplf-related usermeta values.

```
    $foo->saveUserOptions( optionName , optionValue );
```

optionName (string)
The name of the target option. NOTE: this function only accepts option's names that are previously declared in the $strictVals variable, inside the WPLF class.

The options accepted are:

email-follow-notification
email-like-notification

No default value. this variable is mandatory.

optionValue (mixed values)
The value that will be saved.

No default value.

Return value: (boolean) if the saving is processed, the function will return *true*

getUserOptions

Gets wplf-related usermeta values.

```
    $foo->getUserOptions( optionName );
```

optionName (string)
The name of the target option. NOTE: this function only accepts option's names that are previously declared in the $strictVals variable, inside the WPLF class.

The options accepted are:

email-follow-notification
email-like-notification

No default value. this variable is mandatory.

Return value: the value from the usermeta selected.

Private functions

There are also 3 private functions that you can use only inside the wplf class.
They are developed to ease the work inside the other functions.

backToObj

Transform nested arrays into objects. This is developed to merge the results used by **getLikes** and **getFollowers** in the **getActivity** function NOTE: works only with first level of array

Instead of other functions in the class, backToObj doesn't need to be declared inside a variable.

```
    $array = array();
    $foo->backToObj( $array );
    //now $array is an object stdClass
```

Return value: No return value. This function modifies the target array.

notify

Sends an email to the target user and to the user that's followed or have his post liked. This function is used by **insertFollower** and **insertLike**. For now, it's developed to work only with these 2 public functions.

```
    $foo->notify( userID , type );
```

userID (integer)
The ID of the user that is followed or have his post liked by the target user. From this ID, the Display name and the email will be used to send the email to him.

No default value. This variable is mandatory.

type (string)
Define the genre of the email that will be sent. For now, it has only 2 types:

```
follow
like
```

By selecting one of these type, the content of the email will change.

No default value. This variable is mandatory.
Return value: Because notify uses wp_mail to send the email, the function will return *true* if the mail is sent.

wplfPage

Used to ease the pagination. It's used by **getFollowers**, **getLikes**, **getFollowing**, **getLiked** and **getFollowingPosts.**

```
$foo->wplfPage( page , limit );
```

page (integer)
The current page defined for pagination.

Default value: *1*

limit (integer)
The number of results that will be displayed per page.

Default value: *10*

Return value: The function will return the right point to gather the data from the query.

Third-part components inside the WPLF Plugin

Wordpress Like & Follow uses **Colorbox jQuery plugin** to handle the lightbox on the front end. The plugin's code is registered and loaded up in the "wp_enqueue_scripts" action hook, with the following codenames:

`wplfColorboxStyle` (for the css style)
`wplfColorbox` (for the js code)

To exploit the full potential of the plugin, please refer to the official site:

http://www.jacklmoore.com/colorbox