UMIL – Unified Mod Install Library

Introduction

UMIL is a simple to use library that makes mod database alterations a pain-free task for both the user and the mod author. It is designed to support the most common database alterations and additions performed by modifications to phpBB3 and along with that supports every database system phpBB3 can be installed or used on with the default package.

UMIL was designed and written by EXreaction along with help from the Modifications team at phpbb.com.

Installation

Installation of UMIL is about as simple as anything can get. All you must do is download and extract the latest stable version of UMIL from http://www.phpbb.com/mods/umil and then upload the files and folders contained in the root directory within the package to your phpBB3 forum's root directory (the one with common.php in it).

Usage

UMIL provides a wide variety of options as to how you can use it, but there are 2 main methods.

- Fully Automatic
    - The fully automatic method of UMIL provides the easiest method of creating and managing database additions or alterations for the mod author.
    - You do not need to write separate install, update, and uninstall instructions as they are all handled from the array of version actions. The only thing you need to worry about is what changes were made from the most recent version and then record those with the up-coming version number in the action list.
    - All one must do in order to use UMIL Auto is setup a few variables containing information such as the name of the mod, the name of the config variable that will hold the currently installed version number, the language file which contains the required language entries for the UMIL Auto system, and an array of versions and actions performed with each version.
    - If UMIL does not provide you with a built in method of doing what your modification requires you can also list in the version actions the name of a custom function to run. It gets sent two items, the current action (install|update|uninstall) and the current version that actions are being executed for. Within that function you may perform any custom actions you desire and when you are done you may return a string with the text you would like outputted to the user as to what just happened (if you would like to). You may also send an array back with the keys 'command' for the name of the command, and 'result' for the result of the command.

- Manual
    - Using the manual method of UMIL provides you with the option to do anything you would like with or without a front-end displayed to the user. For example, you may have built a modification which you would like to install automatically without any actions run from the user, and that can be easily accomplished using this method.
    - To use the manual method you can just create an instance of the class within umil/umil.php (if you do not intend to display the commands and actions performed or you copy the umil core file to some other location you should send true to the constructor to set it to stand alone mode). Then you may call the built in functions as you wish.
    - If you would like to display a front-end to the user that appears the same or similar to how it would be displayed with using the Fully Automatic method you may use the umil/umil_frontend.php file and class. To understand more about how to use the frontend system you may want to look at how things are done from within the umil_auto file. All that the Fully Automatic method does is provide an easy method for mod authors to use the frontend with install, uninstall, and update options without having to actually write the interactions with the frontend class.

UMIL Core Functions

The following list contains some basic information about each of the core UMIL functions and what information you should send to them in order to properly use them.

Cache Functions

- cache_purge($type = '', $style_id = 0)
    - This function is for purging either phpBB3's data cache, authorization cache, or the styles cache.
    - $type – String
        - Leave blank to purge phpBB3's data cache (runs $cache->purge();)
        - Use 'auth' to clear the authorization/permissions system's cache.
        - Use 'imageset' to clear the imageset cache and refresh the imageset. You may enter a specific $style_id if you would like, or leave as 0 to clear them all.
        - Use 'template' to clear the template cache and refresh the template. You may enter a specific $style_id if you would like, or leave as 0 to clear them all.
        - Use 'theme' to clear the theme cache and refresh the theme. You may enter a specific $style_id if you would like, or leave as 0 to clear them all.
    - $style_id – Integer

        - Leave as 0 when submitting either ‘imageset’, ‘template’, or ‘theme’ to clear them all, or submit a specific id to clear only that one.

Config Functions

- config_exists($config_name, $return_result = false)
    - This function is to check to see if a config variable exists or if it does not.
    - $config_name – String
        - The name of the config setting you wish to check for.
    - $return_result – Boolean
        - True if you would like the function to return the value of that config setting if it exists.
        - False if you would like it to only return true/false on if it exists/if it does not.
    - By default it returns true if it exists/false if not or if $return_result is set to true it returns the value of the config variable.
- config_add($config_name, $config_value = '', $is_dynamic = false)
    - This function allows you to add a config setting.
    - $config_name – String
        - The name of the config setting you would like to add
    - $config_value – Mixed
        - The value of the config setting
    - $is_dynamic – Boolean
        - False if the value does not change often
        - True if the value changes very often (for example, Total posts)
- config_update($config_name, $config_value = '', $is_dynamic = false)
    - This function allows you to update an existing config setting.
    - Variable inputs same as config_add
- config_remove($config_name)
    - This function allows you to remove an existing config setting.
    - $config_name – String

        - The name of the config setting you would like to remove

Module Functions

- module exists($class, $parent, $module)
    - Check to see of a module exists
    - $class – String
        - The class to check (‘acp’, ‘mcp’, or ‘ucp’)
    - $parent – String, Integer, or Boolean False
        - Send either Boolean False to ignore parents and check in the entire class.
        - Send an Integer with the parent module_id to search under it
        - Send a String with the parent module_langname to search under it
    - $module – String
        - The module_langname to search for.
    - Returns true if the module exists with the sent requirements or false if it does not
- module_add($class, $parent = 0, $data = array(), $include_path = false)
    - Add a module
    - $class – String
        - The class to check (‘acp’, ‘mcp’, or ‘ucp’)
    - $parent – String or Integer
        - Send an Integer with the parent module_id to put it under (0 for a main category)
        - Send a String with the parent module_langname to put it under
    - $data – Mixed
        - There are two main ways to send data to add a module and a way to add a category.
        - To add a new category, simply send a string for the $data. Like: 'ACP_CAT_TEST_MOD'
        - The first way (the easy way) to add modules is to send an array with the module base name and an array of modes you would like to add. This method makes the system check the corresponding info_$class_ file and then adds the modes submitted if they exist (you may also not set modes and it will add all of them) using the information given in that file. Using that method, you would send

something like the following (for the ACP):

```
array(
  ‘module_basename’ => ‘board’,
   ‘modes’ => array(‘settings’, ‘features’),
)
```

This would add the settings and features modes listed in the includes/acp/info/info_acp_board.php file.

If you were to send this instead:

```
array(
  ‘module_basename’ => ‘board’,
)
```

That would add all of the modes listed in the includes/acp/info/info_acp_board.php file.

        - The alternate way is to submit the full information required for the module manually.

          I will not explain how to use this one because you probably shouldn’t use it unless you can figure it out, but the data sent may look something like this (except with the required information filled out):

```
array(
  'module_enabled' => 1,
  'module_display' => 1,
  'module_basename' => '',
  'module_class'  => $class,
  'parent_id' => (int) $parent,
  'module_langname' => '',
  'module_mode' => '',
  'module_auth'   => '',
)
```

    - $include_path – Boolean | String
        - If you would like to use a custom include path, specify that here, otherwise leave it as false

- module_remove($class, $parent = 0, $module = '', $include_path = false)
    - Remove a module
    - $class – String
        - The class to check (‘acp’, ‘mcp’, or ‘ucp’)
    - $parent – String or Integer

        - Send an Integer with the parent module_id to put it under (0 for a main category)
        - Send a String with the parent module_langname to put it under
    - $module – Mixed
        - This supports a number of methods as well, but mainly you would send a string with the module_id or module_langname of the module you would like to remove.
        - You can send the exact same information you would have used to add a module as well (from the $data array) and it'll remove the ones it would have added using the same array for module_add.
    - $include_path – Boolean | String
        - If you would like to use a custom include path, specify that here, otherwise leave it as false

Permission (auth) Functions

- permission_exists($auth_option, $global = true)
    - Check if a permission (auth) setting exists.
    - $auth_option – String
        - The name of the permission (auth) option
    - $global – Boolean
        - True if it is a global permission that we are searching for
        - False for a local (forum based) permission
    - Returns true if it exists or false if it does not exist.
- permission_add($auth_option, $global = true)
    - Add a permission (auth) option
    - $auth_option – String
        - The name of the permission (auth) option
    - $global – Boolean
        - True if it is a global permission that we are adding
        - False for a local (forum based) permission
- permission_remove($auth_option, $global = true)

    - Remove a permission (auth) option
    - $auth_option – String
        - The name of the permission (auth) option
    - $global – Boolean
        - True if it is a global permission that we are removing
        - False for a local (forum based) permission
- permission_set($name, $auth_option = array(), $type = 'role', $has_permission = true)
    - Allows you to set permissions for a certain group/role
    - $name – String
        - The name of the role/group
    - $auth_option – Mixed
        - The auth_option or array of auth options you would like to set
        - Example: ‘u_attach’ or array(‘u_attach’, ‘u_chgavatar’)
    - $type – String
        - The type - ‘role’ or ‘group’
    - $has_permission – Boolean
        - True to set the permission to ‘Yes’
        - False to set the permission to ‘Never’
- permission_unset($name, $auth_option = array(), $type = 'role')
    - Allows you to remove permissions from a certain group/role (basically setting their permission to ‘No’)
    - $name – String
        - The name of the role/group
    - $auth_option – Mixed
        - The auth_option or array of auth options you would like to set
        - Example: ‘u_attach’ or array(‘u_attach’, ‘u_chgavatar’)
    - $type – String
        - The type - ‘role’ or ‘group’.

Table Functions

- table_exists($table_name)
    - Allows you to check to see if a table exists
    - $table_name – String
        - The name of the table
        - Example: ‘phpbb_forums’ or FORUMS_TABLE (‘phpbb_’ is automatically replaced with the boards specified table prefix)
    - Returns true if the table exists or false if it does not
- table_add($table_name, $table_data = array())
    - Add a table using the create_schema_files table layout.
    - $table_name – String
        - The name of the table
        - Example: ‘phpbb_forums’ or FORUMS_TABLE (‘phpbb_’ is automatically replaced with the boards specified table prefix)
    - $table_data
        - The schema instructions for the table data.
        - For more examples of creating the data you should look at the develop/create_schema_files.php file in the get_schema_struct function or either of the example files for UMIL.
        - Should be sent something like:

          ```
          array(
            ‘COLUMNS’ => array(*column data*),
            ‘PRIMARY_KEY’ => array(*primary key(s)*),
            ‘KEYS’ => array(*keys data*),
          )
          ```
- table_remove($table_name)
    - Remove a table
    - $table_name – String
        - The name of the table
        - Example: ‘phpbb_forums’ or FORUMS_TABLE (‘phpbb_’ is automatically replaced with the boards specified table prefix)

Table Column Functions

- table_column_exists($table_name, $column_name)
    - Check to see if a column exists on a table
    - $table_name – String
        - The name of the table
        - Example: ‘phpbb_forums’ or FORUMS_TABLE (‘phpbb_’ is automatically replaced with the boards specified table prefix)
    - $column_name – String
        - The name of the column to check for
    - Returns true if the column exists, false if not.
- table_column_add($table_name, $column_name = '', $column_data = array())
    - Add a column to a table
    - $table_name – String
        - The name of the table
        - Example: ‘phpbb_forums’ or FORUMS_TABLE (‘phpbb_’ is automatically replaced with the boards specified table prefix)
    - $column_name – String
        - The name of the column to add
    - $column_data
        - The data for the column, same as how you would set the data for the column using the table_add function
        - For examples you should check the umil example files.
        - Example (for an auto-increment field):
          array('UINT', NULL, 'auto_increment')
- table_column_update($table_name, $column_name = '', $column_data = array())
    - Alter/change an existing column in a table (does not allow you to rename a column)
    - $table_name – String
        - The name of the table

        - Example: ‘phpbb_forums’ or FORUMS_TABLE (‘phpbb_’ is automatically replaced with the boards specified table prefix)
    - $column_name – String
        - The name of the column to change
    - $column_data
        - The data for the column, same as how you would set the data for the column using the table_add function
        - For examples you should check the umil example files.
        - Example (for an auto-increment field): array('UINT', NULL, 'auto_increment')
- table_column_remove($table_name, $column_name = '')
    - Remove an existing column in a table
    - $table_name – String
        - The name of the table
        - Example: ‘phpbb_forums’ or FORUMS_TABLE (‘phpbb_’ is automatically replaced with the boards specified table prefix)
    - $column_name – String
        - The name of the column to remove

Table Index/Key Functions

- table_index_exists($table_name, $index_name)
    - Check if a table index/key exists (does not check primary/unique)
    - $table_name – String
        - The name of the table
        - Example: ‘phpbb_forums’ or FORUMS_TABLE (‘phpbb_’ is automatically replaced with the boards specified table prefix)
    - $index_name – String
        - The name of the index/key to check for
    - Returns true if the index/key does exist, false if not.
- table_index_add($table_name, $index_name = '', $column = array())

  - Add an index/key to a table
  - $table_name – String
    - The name of the table
    - Example: ‘phpbb_forums’ or FORUMS_TABLE (‘phpbb_’ is automatically replaced with the boards specified table prefix)
  - $index_name – String
    - The name of the index/key to add
  - $column – Mixed
    - The name of the column (or array of column names)
- table_index_remove($table_name, $index_name = '')
  - Remove an index/key from a table
  - $table_name – String
    - The name of the table
    - Example: ‘phpbb_forums’ or FORUMS_TABLE (‘phpbb_’ is automatically replaced with the boards specified table prefix)
  - $index_name – String
    - The name of the index/key to remove

Miscellaneous Functions

- version_check($url, $path, $file)
  - Allows you to download a file for version checking. The file access should use UNIX end lines.
  - $url – String
    - The url to access (ex: www.phpbb.com)
  - $path – String
    - The path to access (ex: /updatecheck)
  - $file – String
    - The name of the file to access (ex: 30x.txt)
  - Returns False if an error occurred or an array, one key for each line in the file it accessed.

- run_actions($action, $versions, $version_config_name, $version_select = '')
    - This function is what is used to install/uninstall/update from the version array method mainly described for use with the UMIL Auto method.
    - $action – String
        - The action (‘install’, ‘update’, ‘uninstall’) to perform using the given instructions.
    - $versions
        - The array of versions (as described for the UMIL Auto method).
    - $version_config_name – String
        - The name of the config setting which holds/will hold the currently installed version
    - $version select – String
        - Added for the UMIL Auto system to allow you to select the version you want to install/update/uninstall to.

Multicall

Nearly all functions for the UMIL Core were designed to accept what I’ve dubbed “Multicall”. The functions which do not accept Multicall are the _exists functions, version_check, and run_actions.

Multicall was designed to allow you to run one function call and have it run multiple times with an array of actions. To use Multicall (for this example we will use config_add) you just send an array with the data for the first parameter, then for each send an array with the variables in order as they would have to be sent otherwise.

For example, the following two sets of code will perform the same thing:

```
$umil->config_add('config_name', 'config_value');
$umil->config_add('config_name1', 'config_value1');
$umil->config_add('config_name2', 'config_value2');
$umil->config_add('config_name3', 'config_value3', true);
```

```
$umil->config_add(array(
  array('config_name', 'config_value'),
  array('config_name1', 'config_value1'),
  array('config_name2', 'config_value2'),
  array('config_name3', 'config_value3', true),
 );
```

Note that you do not need to send all the variables if the default settings for those is correct (in this case, the first three are not dynamic and the last one is, so we only need to send true for the last one because false is default).

UMIL Frontend Functions

- umil_frontend($title = '', $auto_display_results = false, $force_display_results = false)
    - The constructor.
    - $title – String
        - The title of the mod which will be displayed.
    - $auto_display_results – Boolean
        - False to require the calling of the function display_results after each UMIL Core function call to display the results.
        - True if you would like the results displayed after execution of the UMIL Core functions.
    - $force_display_results – Boolean
        - Allows you to force the results to display (otherwise results will only be shown if an error occurred)
- display_stages($stages, $selected = 1)
    - This is used to display the box of stages shown on the left hand side of the UI.
    - $stages
        - The array of stages, for example:
          ```
          $stages = array(
            'CONFIGURE'          => array('url' => *url for the stage if you would like it
          clickable*),
            'CONFIRM',
            'ACTION',
          );
          ```
    - $selected – Integer
        - The stage the user is currently at.
- confirm_box($check, $title = '', $hidden = '')
    - Used to display an inline confirm box that fits in with the UMIL Frontend design instead of breaking out and using the normal phpBB3 one. Should be used instead of the normal confirm_box function for consistency.

        - $check – Boolean
            - True for checking if confirmed (without any additional parameters) and false for displaying the confirm box
        - $title – String
            - Title/Message used for confirm box.
              Message text is _CONFIRM appended to title.
              If title cannot be found in user->lang a default one is displayed
              If title_CONFIRM cannot be found in user->lang the text given is used.
        - $hidden – String or Array
            - The string or array of hidden variables that will be saved during the confirm box output.
- display_options($options)
    - This function is used to output a display similar as to how data is outputted from acp_board.
    - $options
        - Format this exactly as you would if you were outputting something on acp_board except only put in what you would for the 'vars' part.
- display_results($command = '', $result = '')
    - This is used to display the results of a command.
    - $command – String
        - The command to display
    - $result – String
        - The result of the command run.
        - 'SUCCESS' or $user->lang['SUCCESS'] or it will be assumed that an error occurred.
- done()
    - Call this when you are done displaying anything with the UMIL Frontend.

Stand-Alone Mode

UMIL has an option to run in stand-alone mode. This allows you to run the functions from the main UMIL class in umil/umil.php without outputting any information to the user on what is happening. Running in stand-alone mode gives you great flexibility as to what you can do with UMIL because you can use it at any time if required with a mod (say, for example, your mod stores data in a table and you need to be able to add/remove columns to and from it from an ACP panel to accept more information). You can do that using the stand-alone system.

The stand-alone system also makes it simple to have a fully automatic installation and update system for your mod. Doing that you can set it up so your modification is installed without the end user having to go to a special install page and click through to install or update it. This is mostly useful for large modifications that, once installed, require new tables or columns to exist or will cause errors to appear until an installation script is otherwise run. Doing that will make sure that your mod is installed before any error ever would occur from missing database sections.

To use the stand-alone mode, just include the umil.php file and, when you initiate the UMIL class, send true during the initiation (example: $umil = new umil(true);). When using this method you are also free to copy or move the umil/umil.php file to any location you would like for its use (which is not possible without stand-alone mode). It does not require any extra language files or check to make sure it is fully updated as the other modes do.