



## **The Handbook**

### **Codex**

[http://codex.wordpress.org/Creating Tables with Plugins](http://codex.wordpress.org/Creating_Tables_with_Plugins)

### **Version Date**

6 August 2005

# Creating Tables with Plugins

Some [plugins](#) require their own table in the [WordPress database](#). This often requires users to run an *install* script, or even worse, execute an SQL query on their own, using something like [phpMyAdmin](#). It is possible for Plugins to create their own tables in the database.

To make this work, we need it to be efficient. The last thing we want is for the plugin to be trying to create a table every time WordPress loads. This article explains how to let your Plugin create tables in the WordPress database with little effort and a lot of efficiency.

## *Writing an Installation Function*

The first thing to do is to write a function that will install the necessary tables and options in the WordPress database. It will need to:

1. Create a new table
2. Update past versions of the table
3. Install default options
4. Make sure only the administrator can use the function

## Getting Started

We'll call three globals, `$table_prefix`, `$wpdb`, and `$user_level`. `$table_prefix` is the prefix that users defined in their `wp-config.php` file. By default, the prefix is "wp\_", but it **is not** okay to assume that it is. `$wpdb` is the object that allows us to connect to the database. And lastly, `$user_level` is for the getting the [user level](#) of the user who is accessing the page. Add this to your plugin:

```
function jal_install () {  
    global $table_prefix, $wpdb, $user_level;  
  
    $table_name = $table_prefix . "liveshoutbox";
```

The `$table_name` variable will be used later when we connect to the database. If the prefix is "wp\_", the `$table_name` would be `wp_liveshoutbox`.

## Checking the User Level

Next we'll check to make sure the user is properly logged in and has a user level of 8 or higher (WordPress requires that to manage plugins). If they are not, the functions stops running and the page loading is continued.

```
function jal_install () {  
    global $table_prefix, $wpdb, $user_level;  
  
    $table_name = $table_prefix . "liveshoutbox";
```

```
get_currentuserinfo();
if ($user_level < 8) { return; }
```

## Is the Table Already Installed?

Now we want to detect if the plugin has already been installed. We'll get to why later. The code below returns an array of the names of all of the tables in the database, then checks if our table is already there.

```
$first_install = false;

$result = mysql_list_tables(DB_NAME);
$tables = array();

while ($row = mysql_fetch_row($result)) { $tables[] = $row[0]; }

if (!in_array($table_name, $tables)) {
    $first_install = true;
}
```

## Creating and Updating the Table

We still haven't actually *installed* anything. Instead of using a normal `$wpdb` query to make our table, there is a special function in WordPress that we'll use that can create a table if it isn't there, or change the structure if it doesn't match the current table. This will come in very handy if we ever need to make changes to the MySQL structure in the future. This function is called `dbDelta()`, and it is not normally available in WordPress, so we'll have to include its parent file (`/wp-admin/upgrade-functions.php`). It requires strict MySQL structure. You can refer to `/wp-admin/upgrade-schema.php` for more examples that use `dbDelta()`.

```
$sql = "CREATE TABLE ".$table_name." (
    id mediumint(9) NOT NULL AUTO_INCREMENT,
    time bigint(11) DEFAULT '0' NOT NULL,
    name tinytext NOT NULL,
    text text NOT NULL,
    url VARCHAR(55) NOT NULL,
    UNIQUE KEY id (id)
);";

require_once(ABSPATH . 'wp-admin/upgrade-functions.php');
dbDelta($sql);
```

## First Time Installation Only

Every time this function is used, the above code will execute. But there may be some things that we don't want to install every time the function runs. This is why we added the `$first_install` variable in the beginning.

```
if ($first_install == true) {
    $welcome_name = "Mr. Wordpress";
    $welcome_text = "Congratulations, you just completed the installation!";
}
```

```

$insert = "INSERT INTO ".$table_name.
        " (time,name,text) ".
        "VALUES ('".time()."','".$welcome_name."','".$welcome_text."')";

$results = $wpdb->query( $insert );

update_option('shoutbox_fade_from', "666666");
update_option('shoutbox_fade_to', "FFFFFF");
update_option('shoutbox_update_seconds', 4000);
}

```

## The Whole Function

This function is done. Let's see it all in one piece:

```

function jal_install () {
    global $table_prefix, $wpdb, $user_level;

    $table_name = $table_prefix . "liveshoutbox";

    get_currentuserinfo();
    if ($user_level < 8) { return; }

    $result = mysql_list_tables(DB_NAME);
    $tables = array();

    while ($row = mysql_fetch_row($result)) { $tables[] = $row[0]; }

    $first_install = false;

    if (!in_array($table_name, $tables)) {
        $first_install = true;
    }

    $sql = "CREATE TABLE ".$table_name." (
        id mediumint(9) NOT NULL AUTO_INCREMENT,
        time bigint(11) DEFAULT '0' NOT NULL,
        name tinytext NOT NULL,
        text text NOT NULL,
        url VARCHAR(55) NOT NULL,
        UNIQUE KEY id (id)
    );";

    require_once(ABSPATH . 'wp-admin/upgrade-functions.php');
    dbDelta($sql);

    if ($first_install == true) {

        $welcome_name = "Mr. Wordpress";
        $welcome_text = "Congratulations, you just completed the installation!";

        $insert = "INSERT INTO ".$table_name.
                " (time,name,text) ".
                "VALUES ('".time()."','".$welcome_name."','".$welcome_text."')";

        $results = $wpdb->query( $insert );

        update_option('shoutbox_fade_from', "666666");
        update_option('shoutbox_fade_to', "FFFFFF");
        update_option('shoutbox_update_seconds', 4000);
    }
}

```

```
}
```

## ***Calling the function***

When a user activates a plugin, they send a GET request. For example, it sends a `http://example.com/wp-admin/plugins.php?activate=true` to the server. We can use this activation request in our plugin to check if the table we want to create already exists.

```
if (isset($_GET['activate']) && $_GET['activate'] == 'true')
```

This will return `true` every time WordPress receives a GET request with `activate=true`. To protect against troublemakers, we've made sure to check in the function if the person has permission to make plugin changes (WordPress requires [user level](#) 8 or greater). So let's use this conditional to call the installation function.

```
if (isset($_GET['activate']) && $_GET['activate'] == 'true') {  
    add_action('init', 'jal_install');  
}
```

The `add_action` function is for plugins that hook into various features of WordPress. We will be using the 'init' hook, which is used when WordPress initializes.

## ***Resources***

- [WordPress Coding Standards](#)
- [Writing a Plugin](#)
- [Plugin API](#)
- [WordPress Hackers Mailing List: Answer to Plugin Requires Additional Tables](#) (<http://comox.textdrive.com/pipermail/wp-hackers/2005-May/000940.html>)
- [Adding Administration Menus to Plugins](#)