Enabling the Auto-Update

Enabling the Auto-Update

Since PrestaShop 1.5, it is possible to have your module auto-update: once a new version is available on Addons, PrestaShop suggests an "Update it!" button to the user. Clicking this button will trigger a series of methods, each leading closer to the latest version of your module.

In order to bring auto-update support to your module, you need three main things:

- Clearly indicate the module's version number in its constructor method: `$this->version = '1.1';`
- Create an `/upgrade` sub-folder in the module's folder.
- Add an auto-update PHP script for each new version.

For instance:

```
/*
* File: /upgrade/Upgrade-1.1.php
*/
function upgrade_module_1_1($module) {
  // Process Module upgrade to 1.1
  // ....
  return true; // Return true if success.
}
```

...and then:

```
/*
* File: /upgrade/Upgrade-1.2.php
*/
function upgrade_module_1_2($module) {
  // Process Module upgrade to 1.2
  // ....
  return true; // Return true if succes.
}
```

Each method should bring the necessary changes to the module's files and database data in order to reach the latest version.

For instance, here is the `install-1.4.9.php` file from the gamification module:

```
<?php
if (!defined('_PS_VERSION_'))
    exit;
function upgrade_module_1_4_9($object)
{
    return Db::getInstance()->execute(
            'CREATE TABLE IF NOT EXISTS `'._DB_PREFIX_.'tab_advice` (
              `id_tab` int(11) NOT NULL,
              `id_advice` int(11) NOT NULL,
              PRIMARY KEY (`id_tab`, `id_advice`)
            ) ENGINE='._MYSQL_ENGINE_.' DEFAULT CHARSET=utf8;');
}
```

The homeslider module's `install-1.2.1.php` file does even more:

```
<?php
if (!defined('_PS_VERSION_'))
    exit;
function upgrade_module_1_2_1($object)
{
    return Db::getInstance()->execute('
    UPDATE '._DB_PREFIX_.'homeslider_slides_lang SET
        '.homeslider_stripslashes_field('title').',
        '.homeslider_stripslashes_field('description').',
        '.homeslider_stripslashes_field('legend').',
        '.homeslider_stripslashes_field('url')
    );
}
function homeslider_stripslashes_field($field)
{
    $quotes = array('"\\\'"', '"\'"');
    $dquotes = array('\'\\\\"\'', '\'"\'');
    $backslashes = array('"\\\\\\\\"', '"\\\\"');
    return '`'.bqSQL($field).'` = replace(replace(replace(`'.bqSQL($field).'`, '.$quotes[0].', '.$quotes[1].'),
'.$dquotes[0].', '.$dquotes[1].'), '.$backslashes[0].', '.$backslashes[1].')';
}
```

PrestaShop will then parse all of these scripts one after the other, sequentially. It is therefore highly advised to number your module's versions sequentially, and to only use numbers – because the upgrade code uses PHP's version_compare() method.

If the new version of your module adds or update its hooks, you should make sure to update them too ⚠️

Indeed, since the hooks are (usually) defined when the module is installed, PrestaShop will not install the module again in order to include the new hooks' code, so you have to use the upgrade methods:

For instance, here's the `install-1.2.php` file from the blockbestseller module:

```
<?php
if (!defined('_PS_VERSION_'))
    exit;

function upgrade_module_1_2($object)
{
    return ($object->registerHook('addproduct')
      && $object->registerHook('updateproduct')
      && $object->registerHook('deleteproduct')
      && $object->registerHook('actionOrderStatusPostUpdate'));
}
```