Managing Hooks

Managing Hooks

Hooks are a way to associate your code to some specific PrestaShop events.

Most of the time, they are used to insert content in a page.

For instance, the PrestaShop default theme's home page has the following hooks:

Hook name	Description
displayHeader	Displays the content in the page's header area.
displayTop	Displays the content in the page's top area.
displayLeftColumn	Displays the content in the page's left column.
displayHome	Displays the content in the page's central area.
displayRightColumn	Displays the content in the page's right column.
displayFooter	Displays the content in the page's footer area.

Hooks can also be used to perform specific actions under certain circumstances (i.e. sending an e-mail to the client).

You can get a full list of the hooks available in PrestaShop 1.6 in the "Hooks in PrestaShop 1.6" chapter of the Developer Guide.

Using hooks

...in a controller

It is easy to call a hook from within a controller: you simply have to use its name with the `hookExec()` method: `Module::hookExec('NameOfHook');`

For instance:

```
$this->context->smarty->assign('HOOK_LEFT_COLUMN', Module::hookExec('displayLeftColumn'));
```

...in a module

In order to attach your code to a hook, you must create a non-static public method, starting with the "`hook`" keyword followed by either "`display`" or "`action`", and the name of the hook you want to use.

This method receives one (and only one) argument: an array of the contextual information sent to the hook.

```
public function hookDisplayNameOfHook($params)
{
    // Your code.
}
```

In order for a module to respond to a hook call, the hook must be registered within PrestaShop. Hook registration is done using the `registerHook()` method. Registration is usually done during the module's installation.

```
public function install()
{
    return parent::install() && $this->registerHook('NameOfHook');
}
```

...in a theme

It is easy to call a hook from within a template file (`.tpl`): you simply have to use its name with the `hook` function. You can add the name of a module that you want the hook execute.

For instance:

```
{hook h='displayLeftColumn' mod='blockcart'}
```

Creating your own hook

You can create new PrestaShop hooks by adding a new record in the `ps_hook` table in your MySQL database. You could do it the hard way:

```
INSERT INTO `ps_hook` (`name`, `title`, `description`) VALUES ('nameOfHook', 'The name of your hook', 'This is a custom hook!');
```

...but PrestaShop enables you to do it the easy way:

```
$this->registerHook('NameOfHook');
```

If the hook "NameOfHook" doesn't exist, PrestaShop will create it for you. No need to do the SQL query anymore.