Mieux comprendre et utiliser les hooks

Mieux comprendre et utiliser les “hooks”

Cet article a été écrit par Julien Breux, et publié sur le blog de PrestaShop le 5 mai 2011.

Qu’est-ce qu’un « hook » ?

Comme vous l’avez sûrement déjà constaté, PrestaShop est un logiciel qui vous permet de créer des modules ayant pour but d’interagir directement avec l’affichage ou les événements du coeur de la solution.

Les « hooks » ou « crochets » vous permettent de récupérer ces événements ou encore modifier l’ affichage.

Il existe donc en réalité deux types de « hooks » distincts.

- Les « hooks » d’action (permettant par exemple d’envoyer un mail lors de l’inscription d’un client)
- Les « hooks » de vue (permettant par exemple d’afficher un module dans une colonne)

Astuce Parfois les « hooks » de vue peuvent aussi servir de « hooks » d’action, il suffit simplement de ne rien leur faire afficher. Par exemple pour effectuer une tâche récurrente sur la page d’accueil avec le « hook » « home ».

Si nous prenons l’exemple du thème de base de PrestaShop, sur la page d’accueil, la solution utilise les « points d’accroche » suivant :

Nom du « hook »	Description
header	« hook » d’affichage « En-tête »
top	« hook » d’affichage « Haut »
leftColumn	« hook » d’affichage « Colonne de gauche »
home	« hook » d’affichage « Page d’accueil »
rightColumn	« hook » d’affichage « Colonne de droite »
footer	« hook » d’affichage « Pied de page »

Comme vous pouvez le constater, la totalité des « hooks » utilisés sont des « hooks » de vue.

Ce qui signifie que chacun de vos modules peut indépendamment s’accrocher sur ceux-ci et y afficher une information.

Comment les utiliser ?

Tout d’abord pour utiliser correctement les « hooks », il vous faut créer dans la classe de votre module une méthode non statique publique commençant par le mot clé « hook » suivi du nom du « hook » utilisé.
Puis, un seul et unique argument est passé à la méthode, il s’agit du tableau des différentes informations de contexte envoyées au « hook ».

```
public function hookNameOfHook($params)
{
}
```

Ensuite, il est important dans l'installation de votre module de l'accrocher aux différents « crochets » désirés. Pour cela, vous utiliserez la méthode « registerHook » n'acceptant qu'un seul paramètre également, il s'agit du nom du « hook »

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

Astuce

Il est inutile d'utiliser la méthode de « désinstallation » du module pour supprimer le « hook ».

Enfin, il est important de comprendre le fonctionnement d'appel de ces différents « hooks » afin de pouvoir en créer de nouveaux par la suite.

Il existe dans PrestaShop deux appels pour les « hooks » sachant que le second tire parti du premier.

Le premier appel est la méthode directe. Il prend deux arguments : le nom du « hook » et un tableau des différentes informations de contexte.

```
$params = array(
    'param_1' => 'value_1',
    'param_2' => 'value_2',
);
Module::hookExec('NameOfHook', $params);
```

Le second appel est un « raccourci » vers le premier afin d'afficher plus « proprement » l'appel à effectuer. Tous les « raccourcis » sont disponibles dans la classe « Hook ».

```
class HookCore extends ObjectModel
{
    // ...
    static public function updateProduct($product)
    {
        $params = array('product' => $product);
        return Module::hookExec('updateProduct', $params);
    }
    // ...
}
```

L'appel du « hook » nommé « updateProduct » se fera donc comme ci-dessous dans le coeur de PrestaShop.

```
Hook::updateProduct(new Product(/* ... */));
```

Nous avons bien appelé la classe « HookCore » en utilisant « Hook » comme nom de classe. Ceci est dû à l'override que nous aborderons la prochaine fois !

Comment en ajouter de nouveau ?

Si vous avez suivi jusqu'ici le déroulement d'utilisation des « hooks » et après utilisation de PrestaShop, alors vous allez sûrement vous rendre compte qui vous manquera des « hooks » comme « FaireLeCafe », « FaireAManger », ou encore « RepondreAuxClientsAMaPlace ».

Pas de panique !

Pour créer votre petit « hook » à vous, il vous suffit tout simplement d'enregistrer une ligne dans la base de données dans la table « ps_hook » avec le nom de votre « hook », son titre. (Voir 0 et 1 si celui-ci est compatible avec LiveEdit ou non)

```
INSERT INTO `ps_hook` (`name`, `title`, `description`)
  VALUES ('nameOfHook', 'Name Of Hook', 'It is a custom hook !');
```

Enfin, utilisez-le(s) simplement comme nous avons vu dans cet article.