Prestashop - Développement d'un module

David Annebicque

Décembre 2016

1 Objectif de cette troisième partie sur Prestashop

Réalisation d'une page de configuration dans le back-office (B-O) qui doit permettre de configurer les éléments à afficher sur le front-office (F-O). Manipulation de formulaires. Toute la documentation reste à disposition pour tout complément. La documentation de Prestashop 1.7 est encore très limitée. Toutes les fonctionnalités de Prestashop 1.6 ne sont plus forcément d'actualité...

2 Activer la configuration d'un module

Pour activer la page de configuration d'un module dans le B-O cela est assez simple, il suffit d'ajouter dans le fichier php du module (le premier créé) une méthode getContent(). Cette méthode déclenche l'affiche, mais se chargera aussi de sauvegarder les données.

2.1 getContent

Ci-dessous un exemple de méthode getContent. Cette méthode gère à la fois l'affichage du formulaire, mais aussi le traitement des données lorsque le formulaire est soumis.

```
1 <?php
2 public function getContent()
3 {
4  $output = null;
5
6  if (Tools::isSubmit('submit'.$this->name))
```

```
7   {
8    $my_module_name =
     ↪  strval(Tools::getValue('DAMODULE_NAME'));
9
10          if (!$my_module_name
11                  || empty($my_module_name)
12                  ||
                    ↪  !Validate::isGenericName($my_module_name)
13          )
14                  $output .= $this->displayError(
15                  $this->getTranslator()->trans(
16                  'Invalid Configuration value',
17                  array(),
18                  'Modules.DaModule'));
19          else
20          {
21                  Configuration::updateValue('DAMODULE_NAME',
                    ↪           $my_module_name);
22                  $output .= $this->displayConfirmation(
23                  $this->getTranslator()->trans(
24                  'Settings updated',
25                  array(),
26                  'Modules.DaModule'));
27          }
28   }
29
30   return $output.$this->displayForm();
31  }
```

2.2 Quelques helpers

— Configuration : est un objet (singleton) qui contient toute la configuration. On peut ”get(nom)” (récupérer) ou ”updateValue(nom, valeur)”. Il existe d'autres méthodes.

— Tools permet par exemple de récupérer des données d'un formulaire (getValue(champs)), de changer de langue, de créer un lien, ...

La documentation la plus fiable pour les helpers est de lire les méthodes, souvent documentées directement dans les fichiers du répertoire Classes

2.3 Helper message

Il existe également un helper pour afficher des messages d'erreur ou de confirmation.

```php
<?php
$output = $this->displayConfirmation($this->l('Settings updated'));
$output = $this->displayError($this->l('Invalid Configuration value'));
```

2.4 Helper Formulaire

Ci-dessous est donné un exemple de méthode permettant d'afficher un formulaire dans une page de configuration. Pour cela on utilise un helper, qui permet de tout paramètrer pour l'affichage : Le titre, les champs, la langue, les boutons disponible... Les dénominations sont suffisamment explicites pour comprendre l'action.

```php
<?php
public function displayForm()
{
// Get default language
$default_lang = (int)Configuration::get('PS_LANG_DEFAULT');

// Init Fields form array
$fields_form[0]['form'] = array(
'legend' => array(
'title' => $this->l('Settings'),
),
'input' => array(
array(
'type' => 'text',
'label' => $this->l('Configuration value'),
'name' => 'DAMODULE_NAME',
'size' => 20,
'required' => true
)
),
'submit' => array(
'title' => $this->l('Save'),
```

```
23  'class' => 'btn btn-default pull-right'
24  )
25  );
26
27  $helper = new HelperForm();
28
29  // Module, token and currentIndex
30  $helper->module = $this;
31  $helper->name_controller = $this->name;
32  $helper->token = Tools::getAdminTokenLite('AdminModules');
33  $helper->currentIndex =
    ↪ AdminController::$currentIndex.'&configure='.$this->name;
    ↪
34
35  // Title and toolbar
36  $helper->title = $this->displayName;
37  $helper->show_toolbar = true;        // false -> remove
    ↪ toolbar
38  $helper->toolbar_scroll = true;      // yes - > Toolbar is
    ↪ always visible on the top of the screen.
39  $helper->submit_action = 'submit'.$this->name;
40  $helper->toolbar_btn = array(
41  'save' =>
42  array(
43  'desc' => $this->l('Save'),
44  'href' =>
    ↪ AdminController::$currentIndex.'&configure='.$this->name.'&save'.$this-
45  '&token='.Tools::getAdminTokenLite('AdminModules'),
46  ),
47  'back' => array(
48  'href' =>
    ↪ AdminController::$currentIndex.'&token='.Tools::getAdminTokenLite('Admi
49  'desc' => $this->l('Back to list')
50  )
51  );
52
53  // Load current value
54  $helper->fields_value['DAMODULE_NAME'] =
    ↪ Configuration::get('DAMODULE_NAME');
55
56  return $helper->generateForm($fields_form);
```

```
57 }
```

Pour afficher un formulaire il faut écrire :

```
1 $output = $this->displayForm();
```

Cet affiche est cumulable avec les messages d'erreur ou de confirmation. Dans ce cas il faut tout concaténer dans la variable output.

3 Travail demandé

Réaliser une page de configuration qui permettra de modifier le nom du module (à la fois dans la barre latérale et dans la page principale), ainsi que de choisir quelles rubriques afficher (en proposer au moins 3). Vous êtes libre de la forme. L'ensemble des données seront sauvegardées dans la base de données (table configuration), et les répercutions seront appliquées sur le F-O.