

PARIE II CONFUGURATION DE MODULE

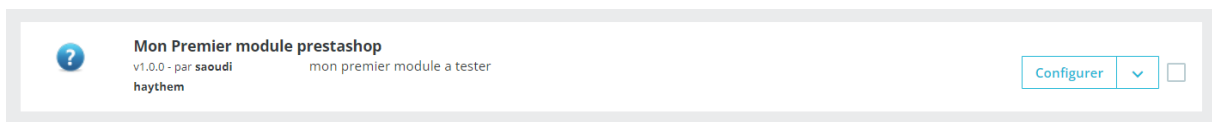
1- Créer une vue de configuration

Notre premier module a été bien créé, maintenant il faut commencer à lui donner une utilité. Pour se faire nous allons commencer par créer un formulaire dans le back office pour gérer ses réglages (configuration).

a- Afficher une page de configuration

Pour que Prestashop sache que votre module dispose d'une page de configuration, il faut utiliser la méthode ***getContent()*** qui retournera un formulaire accessible par l'utilisateur.

Si cette méthode existe, un bouton **Configurer** sera présent à droite du module, même si cette méthode ne retourne rien !



Voici le contenu de cette méthode :

```
public function getContent(){  
    $message=null;  
    if(Tools::getValue("courserating"))  
    {  
        Configuration::updateValue('COURSE_RATING',Tools::getValue("courserating"));  
        $message ="Form saved correctly";  
    }  
    $courserating=Configuration::get('COURSE_RATING');  
    $this->context->smarty->assign(['courserating'=>$courserating,  
    'message'=>$message  
    ]);  
    return $this->display(__FILE__, 'views/templates/admin/configuration.tpl');
```

Tools::getValue() est une méthode spécifique à PrestaShop qui récupère le paramètre spécifié depuis POST ou GET. Dans ce cas, nous récupérerons la valeur de du champ input nommé « courserating » puis la stockons dans la variable \$courserating.

b- La classe Configuration

Les méthodes de la classe Configuration permettent d'accéder à la table **ps_configuration** dans laquelle la plupart des modules inscrivent leurs réglages sans écrire des requêtes SQL .

- *Configuration::get(key)* : Récupère la valeur donnée en paramètre
- *Configuration::update(key, value)* : Modifie le réglage avec la clé donnée en paramètre, à la valeur spécifiée. Si le réglage n'existe pas il est automatiquement créé
- *Configuration::deleteByName(key)* : Supprime le réglage de la base de données

Notez que lorsque vous enregistrez un réglage, la valeur de celui-ci peut-être une chaîne de caractère, un nombre entier, un tableau, etc.

2- Créer un formulaire de configuration

Ceci peut être réaliser avec deux façons :

- Avec un un fichier « .tpl » placer dans le dossier **/views/templates/admin/**
- Avec une méthode créée dans la classe principale « MonModule » et invoquée dans la méthode getContent(). Cette méthode utilisera la classe **HelperForm** qui met à disposition toute une série de méthodes pour créer des formulaires dans le back office

a- Première façon : avec un fichier .tpl

- ❖ Créer un fichier nommé «configuration.tpl» dans le répertoire **/views/templates/admin/**. La méthode **display** permet d'accéder et d'afficher cette vue.

```
return $this->display(__FILE__, 'views/templates/admin/configuration.tpl');
```

Voici le contenu du fichier « *configuration.tpl* »

```

{if $message!=null}
<div class="alert alert-success" role="alert">
  <p class="alert-text">{$message}</p>
</div>
{else}

{/if}
<form action="" method="post">
<div class="form-group">
<label class="form-control-label" for="input1">Normal input</label>
<input type="text" class="form-control" name="courserating" value="{$courserating}" id="input1" />

</div>
<div class="form-group">
<button type="submit" class="btn btn-primary">Submit</button>

</div>
</form>

```

b- Deuxième façon : HelperForm

Dans cette partie, nous allons gérer l’affichage du formulaire dans une méthode à part de la classe MyModule appelée displayForm. Cette méthode sera invoqué dans la méthode getContent().

La méthode displayForm utilisera la classe HelperForm qui met à disposition toute une série de méthodes pour créer et des formulaires dans le **back office**.

Voici à quoi ressemblera cette méthode :

```

public function displayForm() {

    $form=[
        'form'=>[
            'legend'=>['title'=>'Paramètre du cours'],
            'input'=>[
                [
                    'type'=>'text',
                    'label'=>'Evaluation du cours',
                    'name'=>'courserating',
                    'size'=>20,
                    'required'=>true
                ]
            ],
            'submit'=>[
                'title'=>'Enregistrer La configuration',
                'class'=>'btn btn-success pull-right'
            ]
        ]
    ];

    $helper=new HelperForm();
    $helper->module=$this;
    $helper->name_controller = $this->name;
    $helper->token=Tools::getAdminTokenLite('AdminModules');
    $helper->currentIndex=AdminController::$currentIndex.'&configure='.$this->name;
    $helper->submit_action='submit'.$this->name;
    $helper->fields_value['courserating']=Configuration::get('COURSE_RATING');
    return $helper->generateForm(array($form));

}
}

```

- ✓ On crée tout d'abord les différents éléments qui composeront notre formulaire : titre (legend), champ (input) et bouton de validation (submit).
- ✓ Ensuite on va créer un nouvel objet HelperForm avec `$helper = new HelperForm()` et il recevra les champs définis en paramètre lors de la génération.

voici les champs définis :

- `$helper->module = $this`: L'objet HelperForm fait référence à l'instance actuelle du module, ce qui permet d'accéder aux méthodes et propriétés du module.
- `$helper->name_controller = $this->name`: Renseigne le nom du controller, ici le nom du module
- `$helper->token=Tools::getAdminTokenLite('AdminModules')`: Utilise `Tools::getAdminTokenLite()` pour générer un jeton de sécurité, évitant les attaques CSRF
- `$helper->currentIndex = AdminController::$currentIndex.'&configure='.$this->name`: Définit l'URL vers laquelle le formulaire sera soumis, avec le paramètre `configure` pour savoir quel module est configuré.
- `$helper->submit_action='submit'.$this->name`: Détermine l'action qui sera déclenchée lors de la soumission du formulaire.

- `$helper->fields_value['courserating']= Configuration::get('COURSE_RATING') :` récupère la valeur actuelle de notre champ dans la base pour l'afficher

Dernière chose, génération du formulaire avec, en paramètre, la liste des champs à créer à l'aide de la ligne :

- `return $helper->generateForm(array($form))`

Pour obtenir le résultat suivant :

Maintenant on va réécrire le code de la méthode `getContent()` comme suit :

```
public function getContent() {
    $output="";
    if(Tools::isSubmit('submit'.$this->name)){
        $courserating=Tools::getValue('courserating');
        if($courserating && !empty($courserating))
        {
            Configuration::updateValue('COURSE_RATING',$courserating);
            $output.=$this->displayConfirmation('Enregistrement avec succès');
        }
        else
        {
            $output.=$this->displayError('erreur d enregistrement');
        }
    }
    return $output.$this->displayForm();
}
```