PARIE 1 EXTENSION DE PRESTASHOP (LES MODULES)

I. FONCTIONNEMENT

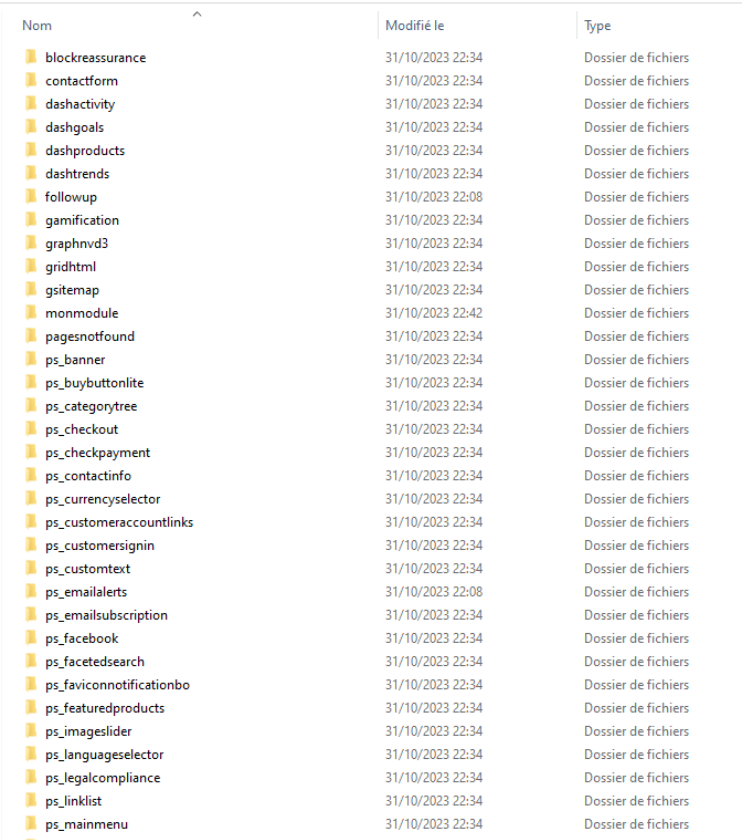
Un module Prestashop se constitue d'un fichier PHP principal avec autant de fichiers PHP que vous jugez nécessaires, des templates (fichiers .tpl) et des ressources (CSS, JavaScript, images, polices). Un module peut s'afficher uniquement pour le marchand dans le back office ou pour le visiteur sur la boutique, ou les deux. Le module peut ensuite interagir avec ce que Prestashop appel les hooks (crochet) qui permet lors de l'appel d'un hook par Prestashop d'exécuter certaines parties du code de votre module. Par exemple lorsque le hook displayHome qui affiche la page d'accueil est appelé, il est possible de lier son module à ce hook et d'afficher des informations ou de réaliser une opération à ce moment-là.

Un module peut :

- Afficher du contenu HTML dynamique en fonction de vos besoins
- Exécuter divers tâches (import / export, modification en masse, etc.)
- Être configuré selon les besoins de la tâche à réaliser. Plus la configuration est complète, plus les possibilités sont grandes pour l'utilisateur
- Ajouter des fonctionnalités à Prestashop sans toucher aux fichiers de base du logiciel, ce qui permet d'éviter les problèmes lors des mises à jour de l'application.

II. EMPLACEMENT

Tous les modules se trouvent dans le dossier /modules à la racine de votre installation de Prestashop. Vous verrez en vous rendant dans ce dossier qu'il existe déjà pas mal de modules, une grande partie d'entre eux sont des modules standards installés par défaut avec Prestashop.



Nom	Modifié le	Type
blockreassurance	31/10/2023 22:34	Dossier de fichiers
contactform	31/10/2023 22:34	Dossier de fichiers
dashactivity	31/10/2023 22:34	Dossier de fichiers
dashgoals	31/10/2023 22:34	Dossier de fichiers
dashproducts	31/10/2023 22:34	Dossier de fichiers
dashtrends	31/10/2023 22:34	Dossier de fichiers
followup	31/10/2023 22:08	Dossier de fichiers
gamification	31/10/2023 22:34	Dossier de fichiers
graphnvd3	31/10/2023 22:34	Dossier de fichiers
gridhtml	31/10/2023 22:34	Dossier de fichiers
gsitemap	31/10/2023 22:34	Dossier de fichiers
monmodule	31/10/2023 22:42	Dossier de fichiers
pagesnotfound	31/10/2023 22:34	Dossier de fichiers
ps_banner	31/10/2023 22:34	Dossier de fichiers
ps_buybuttonlite	31/10/2023 22:34	Dossier de fichiers
ps_categorytree	31/10/2023 22:34	Dossier de fichiers
ps_checkout	31/10/2023 22:34	Dossier de fichiers
ps_checkpayment	31/10/2023 22:34	Dossier de fichiers
ps_contactinfo	31/10/2023 22:34	Dossier de fichiers
ps_currencyselector	31/10/2023 22:34	Dossier de fichiers
ps_customeraccountlinks	31/10/2023 22:34	Dossier de fichiers
ps_customersignin	31/10/2023 22:34	Dossier de fichiers
ps_customtext	31/10/2023 22:34	Dossier de fichiers
ps_emailalerts	31/10/2023 22:08	Dossier de fichiers
ps_emailsubscription	31/10/2023 22:34	Dossier de fichiers
ps_facebook	31/10/2023 22:34	Dossier de fichiers
ps_facetedsearch	31/10/2023 22:34	Dossier de fichiers
ps_faviconnotificationbo	31/10/2023 22:34	Dossier de fichiers
ps_featuredproducts	31/10/2023 22:34	Dossier de fichiers
ps_imageslider	31/10/2023 22:34	Dossier de fichiers
ps_languageselector	31/10/2023 22:34	Dossier de fichiers
ps_legalcompliance	31/10/2023 22:34	Dossier de fichiers
ps_linklist	31/10/2023 22:34	Dossier de fichiers
ps_mainmenu	31/10/2023 22:34	Dossier de fichiers

Liste des modules installés par défaut dans le dossier /modules de Prestashop

Les développeurs ont pris le parti de réaliser des modules séparés pour certaines fonctionnalités afin de les extraire du cœur de l'application et de faciliter leur gestion de manière indépendante.

III. STRUCTURE D'UN MODULE

Dans Prestashop 1.7 voici un exemple d'architecture pour un dossier de module :

```
/mymodule
|
├── mymodule.php
├── config.xml
├── logo.png
├── controllers/
|   └── front/
|   └── admin/
├── views/
|   ├── templates/
|   |   ├── front/
|   |   └── admin/
|   ├── css/
|   └── js/
└── translations/
└── upgrade/
```

mymodule.php	**Rôle** : C'est le fichier principal du module. Il contient la classe du module, qui doit étendre la classe Module de PrestaShop. **Contenu** : Déclare les propriétés du module ($name, $version, $author, etc.), les méthodes d'installation, de désinstallation, de mise à jour, ainsi que les hooks (points d'accroche).
config.xml	**Rôle** : Configure certains paramètres ou dépendances du module. **Contenu** : Paramètres de configuration XML utilisés par le module ou par PrestaShop lors de l'installation.
logo.png	**Rôle** : Image utilisée pour représenter le module dans le back-office de PrestaShop. **Taille recommandée** : 32x32 pixels.
Dossiers controllers/front/ et controllers/admin/	**Rôle** : Contiennent les contrôleurs pour le front-office (visiteurs) et le back-office (administrateurs). **Fonctionnalité** : Chaque contrôleur permet de définir des pages spécifiques du module.
Dossier views/	Ce dossier contient tous les fichiers destinés à être affichés, comme les fichiers de templates, CSS, et JavaScript. • **views/templates/** : **Rôle** : Contient les fichiers de templates .tpl pour l'interface utilisateur du module, qui peuvent être séparés pour le front (front/) et le back-office (admin/). • **views/css/** : **Rôle** : Dossier pour les fichiers CSS personnalisés du module. • **views/js/** : **Rôle** : Dossier pour les fichiers JavaScript du module.
Dossier translations/	**Rôle** : Contient les fichiers de traduction pour le module. **Fonctionnement** : PrestaShop utilise ce dossier pour stocker les traductions des chaînes de caractères.
Dossier upgrade/ (optionnel)	**Rôle** : Contient des scripts de mise à jour pour les nouvelles versions du module.

IV. CREER SON PREMIER MODULE

Pour créer votre premier module, rendez-vous dans le dossier /modules et là, créez un dossier avec le nom de votre module.

/modules/monmodule

Ensuite le dossier doit contenir le fichier principal du module pour être valide. Ce fichier doit être nommé comme le dossier, c'est à dire :

monmodule.php

1. **Fichier principal du module**

Dans ce fichier monmodule.php nous allons gérer le fonctionnement de base du module :

- Son constructeur
- Méthode d'installation / désinstallation
- Inscription et fonctionnement des hooks
- Gestion des vues dans le back office

2. **Test initial**

Pour commencer, la première étape au début de chaque module est de vérifier que la constante PS_VERSION est définie :

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

Cette ligne de code permet de vérifier qu'une constante qui est toujours définie dans Prestashop existe et est accessible. Cela indique que l'environnement est bien chargé. Ce code permet d'éviter à un individu malveillant d'accéder et d'exécuter ce fichier seul.

3. **Classe principale**

Ensuite on définit la classe du module, le fichier principal doit définir la classe principale et celle-ci doit être nommée de la même manière que le fichier de module et son dossier, en version **Camel Case**. C'est à dire en passant les premières lettres de chaque mot en majuscule. Donc le nom de module monmodule devient lorsqu'on instancie la classe :

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

 class MonModule extends Module{

 }
```

Notez que la classe extends la classe Module, c'est à dire qu'elle hérite de celle-ci. La classe Module est la classe de base pour les modules.

4. Constructeur

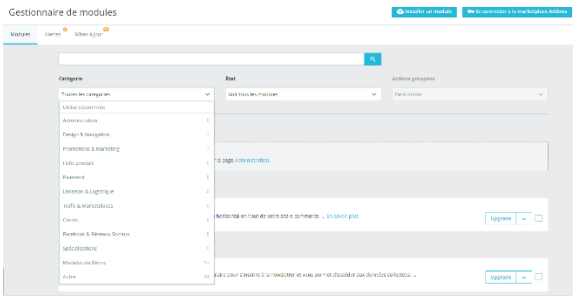
Chaque classe dispose d'un constructeur. Qu'elle hérite de celui de sa classe parente ou qu'il soit clairement défini.

Cette méthode est appelée lorsqu'une classe de votre module est instancié avec le mot clé new. Lorsque le module est chargée par Prestashop, celui-ci créé une nouvelle instance de votre module.

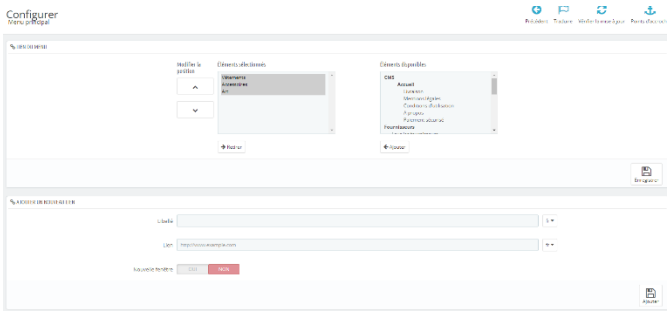
```php
class MonModule extends Module{

    public function __construct(){
      $this->name = 'monmodule';
      $this->tab= 'front_office_features';
      $this->version= '1.0.0';
      $this->author= 'saoudi haythem';

      $this->ps_versions_compliancy=[
          'min' =>'1.7',
          'max' =>_PS_VERSION_
      ];
      $this->bootstrap=true;
      parent::__construct();
      $this->displayName=$this->l('Mon Premier module prestashop');
      $this->description=$this->l('mon premier module a tester');


}
```

Voici en détail ce que fait la méthode __construct :

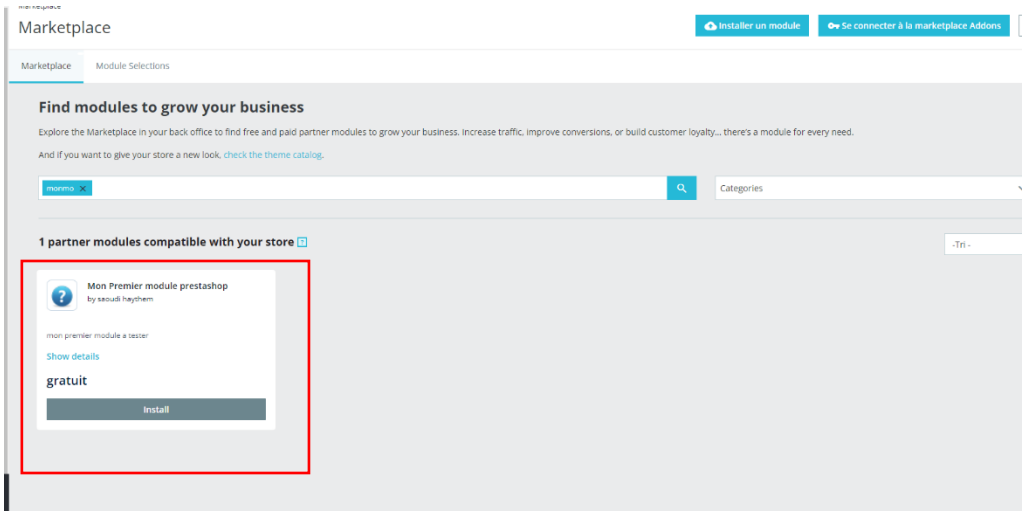
- $this->name = 'monmodule'; permet de définir le nom du module. Il s'agit d'un identifiant interne qui doit porter le même nom que le dossier du module
- $this->tab défini à quel onglet appartient ce module dans la liste des modules



- `$this->version` défini la version du module
- `$this->author` spécifie l'auteur du module
- `$this->ps_versions_compliancy` défini avec quelle version de Prestashop le module est compatible. Ici on indique de la version 1.7 à la version actuelle de Prestashop
- `$this->bootstrap = true` indique qu'on utilisera le système de rendu de `Bootstrap` pour ce module.



  - `parent::__construct()` appel le constructeur de la classe parente, à savoir `Module` pour exécuter la méthode constructeur de base
  - `$this->displayName` défini le nom affiché dans la liste des modules
  - `$this->description` défini la description affichée dans la liste des modules



Installation / désinstallation

Par défaut, les méthodes install et uninstall appellent celles de la classe parente, ici donc si ces méthodes ne sont pas spécifiées Prestashop appel celle de la classe Module.

Voici comment appeler ces méthodes le plus simplement possible :

```
public function install() {

    return parent::install();
}
public function uninstall() {

    return parent::uninstall();
```

Sachez que ne pas définir ces méthodes ou les appeler de cette manière aura exactement le même comportement.

Prestashop les appellent simplement en renvoyant le message (true si c'est OK et false en cas d'erreur) de confirmation d'installation et désinstallation.