Vqmod

Ce qu'il faut savoir

Eric Boutigny - 2013

Ceci est une traduction « libre » de la page :

http://code.google.com/p/vqmod/wiki/Scripting

Note personnelle : *Sans vouloir faire « à tout prix » la promotion de l'utilisation de Vqmod et des fichiers que vous pouvez être amené à utiliser. Je vous engage vivement à utiliser cette fonctionnalité.*

Les raisons à cela :

Lorsque vous souhaitez modifier l'affichage d'une page ou modifier une fonctionnalité, actuellement vous ne pouvez faire autrement que de modifier le code d'une page existante (ajout et/ou suppression d'une partie de code).

Cela implique du temps, mais surtout, lors de la sortie d'une nouvelle version de Opencart, si vous souhaitez pouvoir bénéficier des évolutions de cette nouvelle version, vous devrez reprendre un à un l'ensemble des fichiers que vous aviez modifié, pour y replacer tout le travail que vous aviez mis en place précédemment !

L'utilisation de Vqmod, va vous permettre d'optimiser votre temps, car dans la majorité des cas, les fichiers xml (mod) que vous aller créer, si ils sont bien conçus, seront réutilisable dans la nouvelle version de Opencart, parfois avec quelques modifications légères, qui seront dues à la modification d'une partie du code de cette nouvelle version, mais ce possible temps de modification ne sera jamais aussi important que dans le cas de la « remise en place » de vos adaptations directement dans le code.

Si vos modifications sont faites par l'utilisation de Vqmod, une mise à jour ne prend que peu de temps :

1 désactivation de l'ensemble des fichiers xml (mod)

2 Mise à jour de Opencart

3 souvent réinstallation de Vqmod

4 réactivation de l'ensemble des fichiers xml (mod)

Explications de base :

Un **mod** (réduction de modification), est un fichier xml qui comporte une ou plusieurs modifications dans une fonctionnalité, un affichage, …, il peut concerner une ou plusieurs fichiers.

Un mod ne peut pas travailler sur un fichier « .css ».

Sur Opencart, les fichiers qui pourront être modifiés sont les fichiers « .php » et les fichiers d'affichage « .tpl ».

Pour simplifie la lecture et la compréhension de ce qui suit :

Tout ce qui est noté en vert est « **indispensable** » au fonctionnement d'un mod, pour la grande majorité ce sont les <balises>.

Tout ce qui est noté en bleu est facultatif.

L'ouverture d'une balise se note de cette façon : <balise> et la fermeture de la même balise se note </balise>. Si une balise n'est pas fermé cela provoquera une erreur, généralement visible dans le haut de la page de vqmod_manager, et sera caractérisé par un affichage incomplet de votre mod au sein de vqmod_manager.

Dans les explications suivantes, la notation : *<modification>/<id>*signifie que la balise *<id>* ne peut être ouverte, puis refermée qu'à l'intérieur de la balise *<modification>*, ainsi de suite ; la balise *<modification>* sera refermée par (la balise *</modification>*) à dernière ligne du mod.

La structure de base standard d'un mod (fichier xml pour Vqmod) se présente comme suit :

```
<?xml version="1.0" encoding="UTF-8"?>
<modification>
	<id>Titre du mod</id>
	<version>1.0</version>
	<vqmver>2.X</vqmver>
	<author>nom de l'auteur</author>
	<file name="chemin vers le fichier">
		<operation info="informations sur l'a modification que vous mettez en place">
			<search position="attribut de positionnement"><![CDATA[
			ce que vous recherchez
			]]></search>
			<add><![CDATA[
			ce que vous voulez mettre en place
			]]></add>
		</operation>
	</file>
</modification>
```

Débuter avec Vqmod

Les règles d'or

Premières choses à mettre dans votre esprit ...

La balise <**search**> ne peut être qu'une seule ligne, mais ne doit pas être une ligne entière, comme les correspondances partielles travaillent trop. Vous devez apprendre à utiliser l'attribut offset pour englober correctement toutes les lignes nécessaires, mais cela ne signifie pas que vous devriez inclure de nombreuses lignes. Au lieu d'essayer d'inclure de nombreuses lignes de haut en bas, essayez de démarrer en bas et de trouver des lignes allant vers le haut.

Moins on travaille... Mieux c'est ! C'est aussi valable pour Vqmod, plus le fichier xml sera court et concis plus vite il fera ce pourquoi il est fait !

Chaque fois que cela est possible, essayer de faire correspondre la plus petite partie unique du code. Cela permettra d'améliorer les chances du script d'être plus résistant dans le futur et d'éviter de casser d'autres mods qui peuvent avoir des changements proche de la vôtre.

Faites attention à toutes les options d'attributs pour chaque balise. Il y a beaucoup d'énergie dépensée par ceux qui survolent la syntaxe.

Comment faire Scripts vQmod

vQmod utilise actuellement un parseur XML au format par défaut, mais d'autres parseurs peuvent être créés à l'avenir aussi bien.

Un exemple simple de script de remplacement ressemble à ceci:

```
<? xml version = "1.0" encoding = "UTF-8"?>

<modification>

	<id> Remplacer 123 avec ABC </ id>

	<version> 1.0.0 </ version>

	<vqmver> 1.0.9 </ vqmver>

	<author> qphoria </ author>

	<file name="relative/path/myfile.php">

		<opération>

			<search position="replace"> <! [CDATA [

			$ var = '123 ';

			]]> </ search>

			<add> <! [CDATA [

			$ var = 'ABC';

			]]> </ add>

		</ operation>

	</ file>

</ modification>
```

Ce script modifie et remplace simplement la valeur d'une variable de "myfile.php", il recherche la variable $var = 123 pour affecter à cette variable $var la valeur ABC avant de continuer.

La syntaxe des balises et des options obligatoires ou facultatives

vQmod prend en charge un certain nombre de paramètres:

Balise *<modification>*

C'est le plus haut niveau du fichier et ne peut être présent qu'une fois dans le fichier xml.

Balise *<modification>/<id>*

C'est le nom et la description du mod.

Format: Texte libre. (Information), dans l'exemple plus haut : `<id> Remplacer $var=123 par $var=ABC </ id>`

Balise *<modification>/<version>*

Il s'agit de la version du mod.

Format: Nombre décimal et (1.0.0) (Information), dans l'exemple plus haut : `<version> 1.0.0 </ version>`

Balise *<modification>/<vqmver>*

Ceci est la version minimale requise de VirtualQMod nécessaire pour que le script fonctionne.

Format: Nombre décimal et (1.0.0) (Information), dans l'exemple plus haut : `<vqmver> 1.0.9 </ vqmver>`

Balise *<modification>/<author>*

C'est l'auteur du mod.

Format: Texte libre (Information), dans l'exemple plus haut : `<author> qphoria </ author>`

Balise *<modification>/<file>*

L'utilisation de la balise <file> sert à indiquer le nom du fichier à modifier

Cette balise nécessite l'attribut "name" sert à indiquer le chemin relatif par rapport à l'emplacement du fichier index.php **principal** (par exemple catalog/controller/product/ product.php). Il peut être délimité par des virgules, pour s'appliquer à plusieurs fichiers à la fois (2.3.0 +). L'attribut Name prend un caractère générique (*) ou joker, pour la construction du chemin dynamique. Chaque (*) ou joker, est limitée à un seul niveau de répertoire.

- catalog/view/theme / * / template /product/product.tpl

- catalog/view/theme / * / * /product/product.tpl

- Etc

Cette balise s'écrit : `<file name="relative/path/myfile.php">`

Un mod (script xml) peut concerner des modifications dans plusieurs fichiers, il peut donc y avoir plusieurs balises de fichiers. Chaque fichier peut avoir son propre ensemble d'opérations.

Un attribut optionnel "path" peut être utilisée pour préfixer le nom des fichiers, cela permet d'éviter les répétitions. Ceci est simplement ajouté au début des noms de fichiers et aucune validation n'est tenté (2.3.0 +)

L'attribut facultatif «error» configuré par : log|skip|abort

skip : signifie qu'il va simplement ignorer ce fichier.

log : est le même comme capitaine, mais enregistre l'erreur. (Par défaut)

abort : signifie enregistrer l'erreur et annuler les opérations restantes dans le script XML particulier. Il ne revient pas modifier d'autres fichiers déjà réalisés dans ce script et ne s'arrête pas d'autres fichiers XML.

Les chemins génériques vont ignorer l'attribut "error" complètement

Cette balise s'écrit : `<file name="relative/path/myfile.php" error="log|skip|abort">`

Balise *<modification>* / *<file>* /*<opération>*

C'est le compilateur de l'opération proprement dit.

Il peut y avoir plusieurs opérations dans la même balise <file>.

Un attribut optionnel "info" suggéré. Peut être utilisé par vQmod / vQmod Manager pour OC dans le futur, mais à l'heure actuelle n'est pas utilisé. Ce n'est cependant idéal pour faire plus facile de lire des fichiers

L'attribut optionnel "error" configuré par : log|skip|abort

skip : signifie qu'il va simplement appliquer l'ensemble des opérations ce fichier, même si une seule ne le peut pas.

log : fait la même chose que skip, mais enregistre l'erreur. (Par défaut)

abort : enregistre l'erreur et revient au fichier original, sans aucune modification. (par défaut)

Cette balise s'écrit soit simplement : `<opération>`, soit `<opération info= "texte d'infos">`, soit `<opération error="log|skip|abort">`

Balise *<modification>*/*<file>*/*<operation>*/*<ignoreif>*

Cela permet une recherche à faire sur le fichier. Cette recherche fonctionne sur plusieurs lignes, pas seulement sur des lignes simples. Cette balise est facultative et si la recherche est trouvé, l'opération est ignorée

L'attribut optionnel "regex" à true|false Format de données à l'intérieur balise nécessite alors délimiteurs d'expression régulière car elle utilise la méthode de preg_match standard pour php. Par défaut cet attribut est à false.

Balise *<modification>*/*<file>*/*<operation>*/<search>

C'est la première étape obligatoire de l'opération de modification.

Vous ne pouvez rechercher que des lignes simples à la fois, entièrement ou partiellement. Mais compensés par l'utilisation des attributs d'index qui vous aideront.

Automatise l'équilibre des espaces et des sauts de ligne

Un par balise *<operation>*

Il est vivement recommandé d'utiliser des balises CDATA pour envelopper le code recherché.

L'attribut "position" à before|after|replace|top|bottom|all.

Cet attribut de position va permettre de positionner les données que vous placerez dans la balise <add> par rapport aux données de la balise <search>.

before : permet d'insérer les données de la balise <add> avant les données de la balise <search>.

after : permet d'insérer les données de la balise <add> après les données de la balise <search>.

top : va insérer les données de la balise <add> en haut du fichier. Les données de <search> sont ignorées.

bottom : va insérer les données de la balise <add> à la fin du fichier. Les données de <search> sont ignorées.

all : remplace tout le code dans le fichier par les données de la balise <add> . Les données de <search> sont ignorées.

L'attribut optionnel "offset" permet de travailler sur la position

Si la position de recherche est before et offset 3, il mettra les données <add> avant la ligne, 3 lignes au-dessus de la ligne recherchée.

Si la position de recherche est after et offset 3, il mettra les données <add> après la ligne, 3 lignes en dessous du seuil recherché

Si la position de la recherche est de replace et offset 3, il va supprimer le code de la ligne de recherche et les 3 lignes suivantes et le remplacer par les données <add>

Si la position de recherche est top et le offset 3, ile code sera placé avant la ligne, 3 lignes en dessous du haut du fichier

Si la position de recherche est bottom et le offset 3, le code sera placé après la ligne, trois lignes au-dessus du bas du fichier.

L'attribut optionnel "index" pour spécifier le cas d'une balise de recherche devrait être agi sur

Si la chaîne <*search*> est "echo", qu'il y a 5 échos dans le fichier, mais que vous ne souhaitez par remplacer les 1er et 3ème, utilisez comme suit : index = "1,3"

Séparées par des virgules pour indiquer plusieurs occurrences en commençant par "1"

Laissez de côté ou mettre à false pour remplacer toutes les occurrences. (**par défaut**)

L'attribut optionnel "regex" pour spécifier si oui ou non il y a recherche d'une expression rationnelle.

Si c'est true, les données de recherche devraient être un motif d'expression rationnelle valide

Laissez de côté ou mettre à false pour utiliser la recherche de chaîne normale (**par défaut: true**)

L'attribut optionnel "trim" à true|false

true va rogner les espaces et les sauts de ligne.

Laisser de côté ou la valeur true pour rogner. (**par défaut: true**)

<modification>/<file>/<opération>/<add>

Il s'agit de la deuxième étape obligatoire de l'opération.

Elle peut contenir plusieurs lignes

il n'y a qu'un seul <add> par opération.

L'ajout de ces données dépend de l'attribut de position de la balise <search>.

Utilisez les balises CDATA pour envelopper le code.

L'attribut optionnel "trim" à true|false

true va rogner les espaces et les sauts de ligne.

laisser de côté ou mettre à false pour ne pas couper. (**Par défaut: false**)

<! [CDATA [votre code à ajouter]]>

Elles sont appelées balises CDATA et sont utilisés par xml pour préciser que les données incluses ne doivent pas être évaluées. Utilisée avec <search>, cette fonction recherchera **strictement** la chaîne de caractères incluse dans la balise CDATA.

Il est **recommandé de toujours utiliser cette balise** pour entrer la recherche : <search> et l'ajout : <add> des balises de fonctionnement.

Vous avez besoin d'exemples ? Allez donc regarder la page :

http://code.google.com/p/vqmod/wiki/Examples

Rappel :

Installer Vqmod : http://forum.opencart-france.fr/vqmod/virtualqmod-le-module-le-plus-abouti-pour-les-modifications-t438

Installer Vqmod_Manager : http://opencart-france.fr/index.php?route=extension/extension/info&extension_id=20

Pour toutes questions, aides, … :

http://forum.opencart-france.fr/vqmod