OVIDENTIA

Guide d'utilisation du langage OVML

Version Documentation	***Version OVIDENTIA / Module***	***Date***
1.0	6.0.5	07/12/2006
1.1	6.0.5	19/02/2007
1.2	6.3.3	29/07/2007
1.3	7.2	29/10/09

Table des matières

1 - Introduction

1.1 - OvML : définition

OvML est l'abréviation de Ovidentia Markup Language, ce qui signifie que c'est un langage informatique utilisant les marqueurs. Il s'inspire du HTML et du XML qui sont deux formats très liés. Cependant, HTML et XML sont deux langages de description : l'un permettant de décrire une page Web et l'autre permettant de classer des données. Ils ne sont en aucun cas des langages de programmation comme Javascript ou PHP. OvML a été créé par CANTICO et est proche du HTML par l'utilisation de balises mais ajoute des fonctionnalités proches des langages de programmation comme la possibilité de gérer des variables, de lancer des fonctions et de faire des conditions.

Ressemblances dans la syntaxe :
Une balise HTML : <TABLE width="100%"> </TABLE>
Une balise XML : <STRING id="Number"> </STRING>
Une balise OVML : <OCArticles topicid="25"> </OCArticles>

OvML donc un langage propre à **OVIDENTIA** car les outils ne concernent que des accès à des informations dans un site basé sur **OVIDENTIA**. Seul **OVIDENTIA** inclut un moteur pour l'interprétation des balises OvML et leurs exécutions.

1.2 - Pourquoi OVML ?

OvML vient accompagner le guide du développeur et devient un outil indispensable pour le webmaster voulant ajouter du dynamisme à ses pages Web.

Protection :
- Afin de garantir la fiabilité d'un site basé sur **OVIDENTIA**, on ne peut en aucun cas modifier ou supprimer de la base de données des informations en passant par OvML. OvML se sert de marqueurs (balises) pour aller chercher l'information : contenus des articles ou renseignements sur un utilisateur peuvent alors être récupérés.
- Les résultats retournés par les containers du langage tiennent compte des droits d'accès sauf à quelques exceptions (présence du paramètre "filter" sur les containers).

1.3 - Où trouver la documentation ?

La documentation OvML est disponible sur le site communautaire http://www.ovidentia.org . Le guide du développeur (Developer's guide) peut aussi être un bon outil pour approfondir les possibilités de personnalisation d'un site.

1.4 - Utilisation de l'OVML

OvML est utilisé couramment dans les skins car c'est eux qui émettent les informations principales visibles à la connexion sur un site. Mais ce n'est pas la seule utilisation de l'OvML dans OVIDENTIA. L'éditeur wysiwyg lancé lors de la création des sections personnalisées ou d'un article est un outil qui gère la possibilité d'ajouter des fichiers OVML. On peut alors créer du contenu dynamique ou ajouter un lien vers une information du site. De même, l'OvML peut être appelé dans les modules.

2 - Apprendre le langage

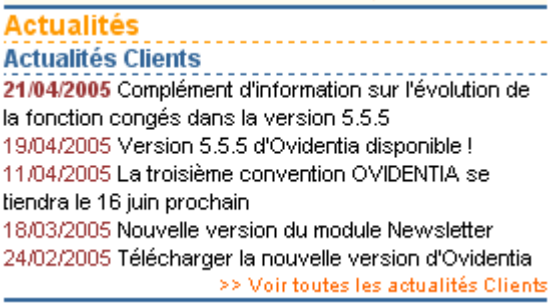
2.1 - La syntaxe

La syntaxe est proche du langage HTML car on utilise uniquement des balises pour écrire. Toutes les requêtes que l'on veut faire seront donc sous la forme :
<nom de la fonction> </nom de la fonction>
où on remarque une balise de début et une balise de fin pour marquer la portée de l'entité dans le code.

Trois types composent le langage : les variables, les fonctions et les containers. Chaque type est identifié par un préfixe.
<OV…> pour les variables
<OF…> pour les fonctions
<OC…> pour les containers
Ex. : le container Articles s'écrit : <OCArticles>
Une variable va permettre de stocker une information ou de l'afficher, une fonction sera un outil pour traiter les données et un container sera un moyen de récupérer le contenu listé de la base de données.

Les paramètres :
Chaque entité peut contenir un ou plusieurs paramètres définis dans la documentation. Un paramètre est un renseignement de plus pour l'entité et s'ajoute à l'intérieur d'une balise de cette manière :
<OVbabSlogan strlen="100">
Ici, on récupère la valeur de la variable globale babSlogan. Le paramètre strlen, dont la valeur est à 100, indique que l'on ne veut que les 100 premiers caractères de cette variable.

Un point intéressant du langage OvML est qu'il peut s'insérer dans du code HTML. Lors de la création des pages d'accueil d'un site, il sera confortable d'ajouter du code OvML pour récupérer dynamiquement des informations dans la base de données et d'un autre côté, HTML permettra d'améliorer l'affichage de ces informations (tableaux, frames…).



exemple d'utilisation d'OvML pour afficher les dernières actualités

2.2 - Les variables

2.2.1 - Les variables et leurs paramètres

Il n'existe pas de variables globales dans OvML sauf des variables propres aux containers. En revanche, il est possible de récupérer les variables globales d' **OVIDENTIA**. Une variable peut aussi récupérer la valeur d'une autre variable.
Il existe plusieurs outils donnés en paramètres d'une variable, afin de formater une date ou encore de couper une chaîne.

Exemple d'utilisation des paramètres :
<OVArticleDate date="%d %j %m %Y">
<OVnbArticles saveas="TotalArticles">
<OVArticleAuthor author="%F %L" strcase="lower">

Liste des paramètres utilisables pour chaque variable
→voir la documentation OvML pour plus de détails
saveas : Permet de sauvegarder une variable dans une autre
strlen : Permet de récupérer les n premiers caractères de la variable
striptags : Supprime les tags HTML de la variable
htmlentities : Convertis les entités HTML
stripslashes : Supprime les slashs doublés du contenu de la variable
urlencode : Encode la variable pour l'utilisation dans une url
jsencode : Encode la variable pour l'utilisation dans un javascript
strcase : Convertis le contenu de la variable en majuscule / minuscule
nlremove : Supprime les retours chariots
trim : Supprime les blancs
nl2br : Transforme les retours chariots en
sprintf : Retourne une chaîne formatée
date : Formate une date
author : Formate l'auteur

Exemples :

- Afficher un résumé des 100 premiers caractères d'une variable 'montexte' :
<OVmontexte strlen="100,,etc…">
',etc…' sera écrit en fin du résumé si la variable contient un nombre de caractères d'au moins 100.

- Supprimer les blancs (espaces) dans une variable et convertir la variable en minuscule :
<OVmontexte trim="all" strcase="lower">

2.2.2 - Création d'une variable

La création d'une variable se fait par l'utilisation de la fonction PutVar comme montrée ci-dessous :
<OFPutVar name=”mavariable” value=”10”>
Ceci créé une variable appelée mavariable avec une valeur initiale à 10.

Pour accéder à la variable par la suite, il suffira d'utiliser <OVmavariable> qui correspond à sa valeur.
Une variable n'a pas de type. Entier ou chaîne, il faut toujours écrire la variable entre guillemets dans le paramètre 'value'.
En revanche, afin que les fonctions d'opérations numériques fonctionnent, il faut qu'un nombre décimal soit écrit avec un point en tant que séparateur :
<OFPutVar name=”mavariable” value=”5.6”> : correct
<OFPutVar name=”mavariable” value=”7,3”> : incorrect

Modifier une variable :
Pour modifier la valeur d'une variable, sans passer par les fonctions arithmétiques, il faut obligatoirement redéclarer cette variable : on utilise la même syntaxe.

2.2.3 - Copie d'une valeur de variable dans une autre

Pour copier une valeur de variable dans une autre, on utilise le paramètre saveas des variables.
<OVvar1 saveas="var2">
Ceci va copier la valeur de "var1" dans une nouvelle variable nommée "var2".

2.2.4 - Récupérer des valeurs de variables globales

Récupérer des valeurs de variables globales correspond à créer une nouvelle variable du même nom. Il faut donc prévoir ces restrictions dans les noms de variables OvML afin qu'aucune nouvelle variable ne puisse être confondue avec une variable globale.

Exemple :
$babSiteName contient le nom du site comme spécifié dans l'administration d'OVIDENTIA. Pour avoir cette valeur, on créé une variable du même nom :
<OFPutVar name="babSiteName">
Pour ensuite afficher cette variable dans un code html :

```
<html>
<head> <title>Titre de la fenêtre<title></head>
<body>
        Le nom du site est : <OVbabSiteName>.
</body>
</html>
```

Si on préfère changer le nom de la variable, il faut dans un premier temps faire la déclaration avec

le nom de la variable globale, puis copier la valeur dans une autre :
<OFPutVar name="babSiteName">
<OVbabSiteName saveas="mavariable">
Ceci va copier la valeur de babSiteName dans une nouvelle variable nommée mavariable.
Remarque :
Les variables globales booléennes comme BAB_SESS_LOGGED ont la valeur nulle si la valeur est fausse (false) et vaut 1 si la valeur est vraie (true).

2.2.5 - Gestion des dates

La date du jour est une donnée que l'on peut récupérer par OvML grâce aux variables globales. Toutes les dates peuvent être générées selon plusieurs formats. A l'origine, lorsque l'on récupère une date dans la base de données (date d'un article, date d'un événement) le format est en timestamps : chiffre correspondant au nombre de secondes qui se sont écoulées depuis le 1er janvier 1970 jusqu'à la date présente, très utilisé dans les langages de programmation.

Récupération de la date du jour :
On utilise la variable globale babCurrentDate :
<OVbabCurrentDate">

Attention : cette variable n'étant pas une variable globale du portail, il ne faut pas utiliser la syntaxe <OFPutVar> contrairement aux autres variables globales.

Changer le format d'affichage :
On utilise le paramètre "date" des variables.

Valeurs possibles du paramètre "date" :
S : Format court de la date défini au niveau site ou au niveau utilisateur
L : Format long de la date défini au niveau site ou au niveau utilisateur
T : Format de l'heure défini au niveau site ou au niveau utilisateur

Pour personnaliser chaque donnée (ajouter le caractère % devant chaque lettre) :
d : Trois premières lettres du jour de la semaine Dim pour Dimanche
D : Jour de la semaine Dimanche par exemple
j : Jour du mois, sur deux chiffres (éventuellement avec un zéro) : "01" à "31"
m : Mois, en trois lettres : par exemple "Avr" (pour Avril)
M : Mois en lettres : Avril par exemple
n : Mois; i.e. "01" à "12"
Y : Année, 4 chiffres; i.e. "2003"
y : Année, 2 chiffres; i.e. "03"
H : heure, au format 24h, "00"
→ Tous les caractères peuvent être ajoutés dans la valeur du paramètre, on peut donc afficher une date entre crochets ou avec les chiffres séparés par des tirets.

Exemples :
<OVmadate date="S"> : formate la date dans un format court
<OVmadate date="%j %n %Y"> : formate la date en chiffres
exemple de résultat : 05 11 2005
<OVmadate date="%D - %j %M (%y)"> : formate la date en lettres
exemple de résultat : Mardi - 13 Juin (2005)

2.2.6 - Tester l'existence d'une variable

On utilise la fonction IfNotIsSet. Ceci n'est pas une fonction de condition, si la variable n'existe pas, elle sera créée.

Syntaxe :
<OFIfNotIsSet name="mavariable" value="1">
Dans ce cas, si la variable mavariable n'est pas définie, elle sera créée avec une valeur initiale de 1.

2.3 - Les containers

2.3.1 - Introduction

Les containers permettent de lister des données ou de faire des conditions (voir les containers de conditions). Un container a toujours deux balises qui lui correspondent : une d'ouverture et une de fermeture afin de délimiter son champ d'action.
Il est possible d'imbriquer les containers sauf s'ils ont le même nom.

Exemple d'imbrication :

```
<OCRecentArticles>
        <OCArticles>
        </OCArticles>
</OCRecentArticles>
```

Dans le cas où imbriquer deux containers identiques est indispensable, il existe l'astuce de rajouter un paramètre de nom différent sans valeur dans chaque balise :

```
<OCRecentArticles ext1>
        <OCRecentArticles ext2>
        </OCRecentArticles ext2>
</OCRecentArticles ext1>
```

Les paramètres :
Tout comme les variables, les containers possèdent des paramètres mais qui ne sont pas communs à tous. Les paramètres sont des moyens de trier, de filtrer ou de modifier l'information qui sera récupérée par les variables. Attention : certains paramètres sont obligatoires.
Exemple :

```
<OCRecentFiles folderid="1" last="10">
</OCRecentFiles>
```

La syntaxe Container/Variables :
Si un container arrive à aller chercher une information dans la base de données, sans variables, il ne sert à rien. Un container permet donc de nommer ce que l'on recherche et les variables précisent l'information voulue. Les variables doivent être écrites à l'intérieur des containers qui

leurs correspondent.
Lorsqu'un container est affiché par le moteur d'ovml, il est parcouru plusieurs fois afin d'afficher une entité à la fois. Si on demande la liste des dernières contributions avec leurs noms et leurs introductions, le moteur va afficher la première contribution avec son nom et son introduction puis la deuxième, et ainsi de suite…
Exemple :

```
<OCRecentFiles folderid="1" last="10">
        <OVFileName>
        <OVFileAuthor>
</OCRecentFiles>
```

Ici, le container RecentFiles liste les derniers fichiers chargés. Le paramètre folderid permet de spécifier le répertoire des fichiers et last indique le nombre maximum de fichiers à lister.
Les variables FileName et FileAuthor à l'intérieur du container récupèrent respectivement le nom des fichiers et leurs auteurs.

→Voir la documentation pour la liste de tous les containers

2.3.2 - OVCIndex et OVCCount

Deux variables sont à disposition du développeur afin de connaître la valeur de l'entrée courante (compteur) et le total des entrées au moment du parcours de n'importe quel container. Ainsi, il est possible d'utiliser ces variables pour réaliser des conditions ou pour afficher un classement chiffré.
CIndex : index de l'entrée courante de 0 à n
CCount : nombre total d'entrées dans le container

La variable CIndex va donc s'incrémenter à chaque nouvelle entité d'un container.
Exemple :

```
<OCForums>
        <OVCIndex><OVForumName>
        <OVForumDescription>
</OCForums>
```

Ici, on affiche la liste des forums existants : leurs noms et leurs descriptions. La variable CIndex affichera un numéro devant chaque nom. Ce numéro correspondra à l'ordre d'affichage dans le parcours du container. Attention, le premier chiffre sera 0.

2.3.3 - Les containers de conditions

Les containers de conditions sont comme les boucles "if" dans les langages de programmation. Ces containers permettent d'effectuer des comparaisons entre deux valeurs : variables OvML alphanumériques. Une condition affichera un code ou un autre selon le résultat. Tout le code compris entre les deux balises ne sera exécuté que si la condition est vraie.
Syntaxe générale :
<OCxxxx expr1="valeur" expr2="valeur"> ….. </OCxxxx>
Les variables à comparer s'écrivent sous forme de paramètres. La première expression (expr1) est toujours comparée à la deuxième (expr2).

Liste des containers :
IfEqual : Vrai si expr1 est égal à expr2

IfNotEqual : Vrai si expr1 est différent expr2
IfLessThan : Vrai si expr1 plus petit strictement que expr2
IfLessThanOrEqual : Vrai si expr1 plus petit ou égal que expr2
IfGreaterThan : Vrai si expr1 plus grande strictement que expr2
IfGreaterThanOrEqual : Vrai si expr1 plus grande ou égal que expr2

Exemple :
<OCIfEqual expr1="<OVCategoryId>" expr2="<OVcat>">
</OCIfEqual>

2.4 - Les fonctions

Les fonctions sont des outils pour le calcul numérique ou la sauvegarde d'une variable. Leurs résultats peuvent être écrits dans une variable.
Syntaxe :
<OFxxxx param1="value" param2="value" …>

Liste des fonctions :
Translate : Permet de traduire une chaîne
PutVar : Permet de sauvegarder une variable
IfNotIsSet : Permet de sauvegarder une variable, si elle n'est pas déjà définie
UrlContent : Permet de récupérer le contenu d'une url
AOAddition : Permet d'additionner
AOSubtraction : Permet de soustraire
AOMultiplication : Permet de multiplier
AODivision : Permet de diviser
AOModulus : Permet de récupérer le reste d'une division (modulo)
Header : Permet d'envoyer un header HTTP pour le fichier ovml

→Voir la documentation pour plus de détails

Exemples :
- Traduire une chaîne :
<OFTranslate text="Bonjour tout le monde" lang="nl-be">
Ceci traduit la chaîne dans le langage dont le préfixe est nl-be. La traduction se rapporte aux fichiers xml de langues : la chaîne est inchangée s'il n'existe pas de traduction.
L'attribut lang est optionnel. Dans ce cas, **OVIDENTIA** utilisera la langue choisie par l'utilisateur courant.

Liste des préfixes par défaut :
fr : français
en : anglais
nl : néerlandais
be : belge
de : allemand

- Faire une division d'une variable avec 50, sauvegarder son résultat et en afficher son reste entier :
<OFAODivision expr1="<var>" expr2="50" saveas="varResultat">
<OFAOModulus expr1="<var>" expr2="50">

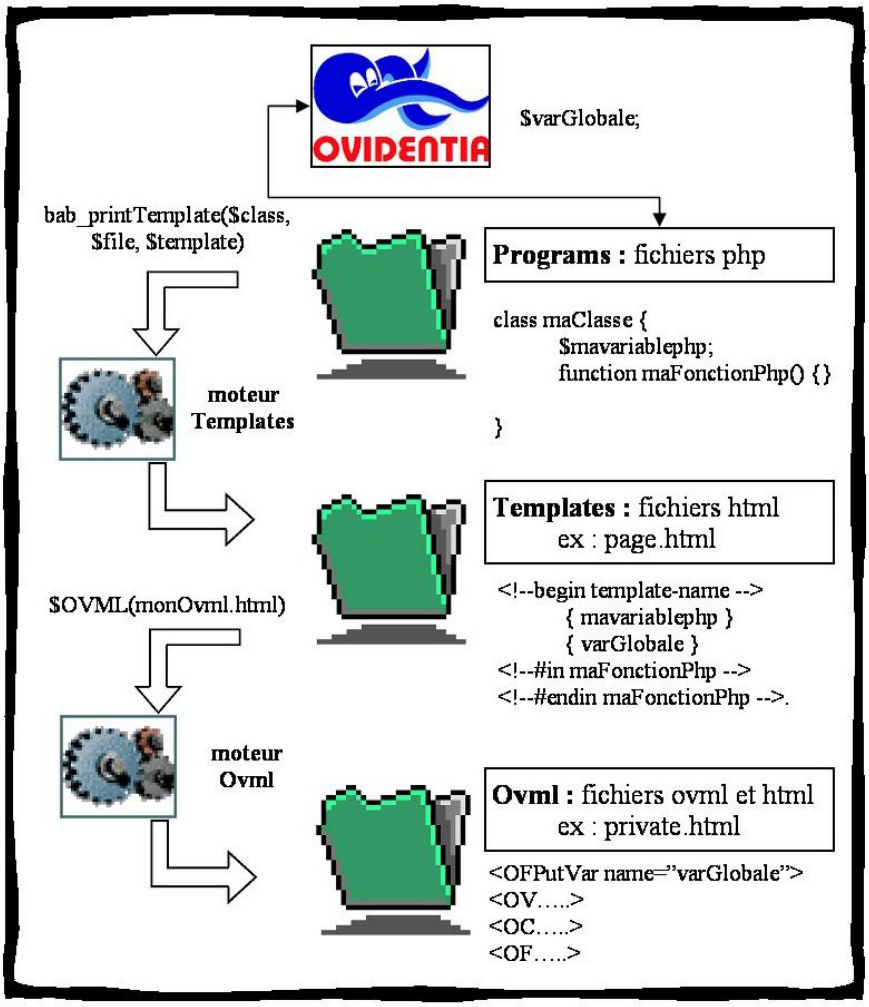
3 - Réalisations

3.1 - Utilisation d'un code OvML dans OVIDENTIA

Un fichier OvML est un fichier texte simple. Il peut avoir n'importe quelle extension mais doit absolument se retrouver dans le dossier OvML du skin courant (les sous-répertoires sont autorisés). Il n'y a aucune restriction sur l'affichage de sortie : on peut lancer un fichier dans une page vierge, une popup... Certaines fonctions d'OVIDENTIA permettent aussi l'ajout de code OvML (voir éditeur wysiwyg).

3.1.1 - Les liens du noyau avec OvML : php→templates→ovml

Un fichier OvML est utilisé dans le cadre de la personnalisation d'un skin ou dans un développement spécifique. Il permet avant toute chose de faire des requêtes dans la base de données sans avoir à développer et devient donc un outil précieux pour les clients qui veulent ajouter du contenu dynamique sur leurs sites.
L'implémentation d'un fichier OvML dans OVIDENTIA est soumise à des règles. Dans le développement, on distingue un fichier php d'un fichier template et d'un fichier OvML. Ceci dans le but de séparer le code pour une meilleure visualisation et dans l'objectif de faciliter les modifications futures. En clair, lors du développement, on créé des fichiers php qui vont appeler des fichiers templates (généralement à l'extension html), afin de séparer le code html du code php. On doit alors utiliser le moteur d'OVIDENTIA pour la gestion des variables et fonctions. De même, on créé des fichiers OvML qui seront appelés par url ou bien depuis les fichiers templates (voir le schéma ci-dessous).



Pour lancer un fichier OvML, il doit donc être placé absolument dans le dossier OvML de la skin qui est créée afin que le moteur OvML puisse fonctionner et interpréter le code.

3.1.2 - Les diverses façons d'exécuter un code OvML

Il existe deux possibilités d'exécuter un fichier OvML : depuis un lien (url) ou dans un fichier template (on verra par la suite qu'il est possible d'exécuter un fichier depuis les articles aussi).

- Lancer un fichier OvML depuis une url du site :
Syntaxe :
« index.php?tg=oml&file=monovml.html »
« index.php?tg=oml&file=mondossier/monovml.html» si le fichier se trouve dans le sous-dossier mondossier.

- Lancer un fichier OvML dans une popup :
Syntaxe :
« index.php?tg=oml&file=monovml.html&echo=1 »
Le paramètre echo est une variante pour lancer le fichier OvML sans l'interface **OVIDENTIA**, le résultat se retrouve seul dans la fenêtre au lieu de se retrouver au centre avec les sections et

l'interface.

- Lancer un fichier OvML depuis un fichier template :
Il faut écrire « { $OVML(monovml.html) } » dans le code.
« { $OVML(mondossier/monovml.html) } » si le fichier se trouve dans le sous-dossier mondossier.

3.1.3 - Envoyer des paramètres à un fichier OvML

Si le paramètre a pour nom monparametre, on pourra le réutiliser avec la syntaxe <OVmonparametre> dans le fichier OvML.
On ajoute simplement le paramètre dans l'URL ainsi que sa valeur. On peut passer plusieurs paramètres à la fois.
Syntaxe :
« index.php?tg=oml&file=monovml.html&monparametre=valeur »

Il est aussi possible depuis la version cliente 5.5.7 d'OVIDENTIA de passer des paramètres avec la syntaxe $OVML(monovml.html). Voir ci-dessous.

Remarque : le fichier qui va recevoir le paramètre ne doit pas utiliser la syntaxe <OFPutVar> pour récupérer la valeur.

3.1.4 - Lancer un fichier OvML depuis du contenu OVIDENTIA

Il est possible d'ajouter un contenu OvML dans un article, une faq ou une section. Pour cela, on utilise l'icône d'ajout de liens OvML dans l'éditeur Wysiwyg ou on écrit simplement « $OVML(monovml.html) » dans le texte (nul besoin d'être en mode HTML).
Depuis la version cliente 5.5.7 d'OVIDENTIA, il est possible de passer des paramètres avec la syntaxe $OVML(monovml.html). Il suffit d'ajouter le paramètre avec sa valeur de cette manière : $OVML(monovml.html,monparametre=1).

Remarque : il est aussi possible de passer en paramètre une variable template à un fichier OVML :
{ $OVML(monovml.html,artid="articleid") }

3.1.5 - Optimiser le temps d'exécution d'un script OVML

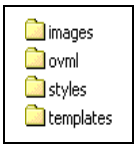
Fonction disponible à partir de la version 7.2 d'Ovidentia.
Si votre script n'a pas besoin d'être réactualisé à chaque exécution, vous pouvez décider d'utiliser le cache via la syntaxe $OVMLCACHE.
$OVMLCACHE s'utilise dans les mêmes conditions que la syntaxe $OVML.
$OVMLCACHE peut recevoir un paramètre supplémentaire : _ovml_cache_duration=86400. Il permet d'indiquer le temps maximum pendant lequel le script ne sera pas réactualisé. Valeur en secondes.
Si ovml_cache_duration n'est pas renseigné, le script sera réactualisé toutes les 3600 secondes (1 heure).
Remarque : le cache est enregistré dans la session de l'utilisateur courant. Donc le script est systématiquement réactualisé si l'utilisateur se déconnecte.

Exemple :
{ $OVMLCACHE(monFichier.html,param1=3,_ovml_cache_duration=86400) }

Attention : pensez bien aux conséquences d'utilisation du cache OVML avant de l'appliquer sur vos scripts. Pendant 3600 secondes, vous pouvez modifier vos scripts et des données d'Ovidentia sans pouvoir le vérifier à l'affichage !

3.2 - Création d'une page OVML pour un skin

3.2.1 - Fonctionnement : arborescence d'un dossier skin



Un skin est régi selon les mêmes règles que le développement de modules. On retrouve un dossier templates et un dossier OvML. Le dossier styles contient les fichiers css nécessaires et le dossier images contient toutes les images référencées dans le code html.

Fonctionnement :
Un skin original existe dans le noyau (dossier skins du noyau). Il contient l'arborescence des dossiers ci-dessus. A l'intérieur du dossier templates, on peut retrouver plus d'une centaine de fichiers html.
Le skin original ne doit jamais être modifié. Si on veut le personnaliser, il suffit de créer un dossier dans le dossier skins à la racine du serveur et non pas dans celui du noyau.
En plus de créer l'arborescence commune à chaque skin (dossiers images, OvML, styles et templates), on ajoute les fichiers images, styles ou templates dont on aura besoin.
→Le moteur d'OVIDENTIA va exécuter un skin en regardant d'abord si les fichiers nécessaires à son exécution sont réunis dans le dossier skins à la racine du serveur. Dans le cas contraire, le moteur exécutera le fichier du skin original du noyau.

Créer un skin revient donc à créer un dossier dans le dossier skins à la racine du serveur, dont le nom sera le nom du skin réalisé. On y place alors les fichiers que l'on veut modifier par rapport au skin original (il faut garder les emplacements originaux).

→Attention, seuls quelques fichiers templates pourront être modifiés

*Fichiers modifiables par l'administrateur dans le dossier templates de chaque skin :
adminsection.html
articlestemplate.html
config.html
forumssection.html
mailtemplate.html
montha.html
page.html
sectiontemplate.html
topcatsection.html
topicsdisplay.html

topicssection.html
usersection.html
warning.html

Fichiers modifiables par l'administrateur dans le dossier OvML de chaque skin :
Tous, car les fichiers OvML sont entièrement personnalisables.

3.2.2 - Les fichiers private.html et public.html

Ces deux fichiers sont particuliers car ce sont les seuls fichiers OvML connus du noyau. Si on place deux fichiers avec ces noms dans le dossier OvML d'un skin, ils seront automatiquement lancés pour apparaître comme étant les pages d'accueil publiques et privées dans OVIDENTIA. Pour la skin ovidentia_fx, on peut voir un exemple de cette utilisation.

3.2.3 - Exemple de code différent selon le profil de l'utilisateur

Un premier test sur l'utilisateur peut se faire grâce à la variable globale BAB_SESS_LOGGED qui renvoie 1 ou 0 selon que l'utilisateur est connecté ou anonyme sur le site.
Un deuxième test possible est l'appartenance de l'utilisateur aux groupes grâce au container ovml IfUserMemberOfGroups.

Exemple :
Ici on teste si l'utilisateur est connecté ou pas. Puis s'il est connecté, on teste son appartenance à 2 groupes.

```
<OFPutVar name="BAB_SESS_LOGGED">
<OCIfEqual expr1="<OVBAB_SESS_LOGGED>" expr2="1">
        Vous êtes connecté.
        <OCIfUserMemberOfGroups groupid="1">
                Vous appartenez au groupe 1
        </OCIfUserMemberOfGroups>
        <OCIfUserMemberOfGroups groupid="2">
                Vous appartenez au groupe 2
        </OCIfUserMemberOfGroups>
</OCIfEqual>
<OCIfEqual expr1="<OVBAB_SESS_LOGGED>" expr2="">
        Vous n'êtes pas connecté.
</OCIfEqual>
```

3.3 - Création d'une section personnalisée

3.3.1 - Le fonctionnement

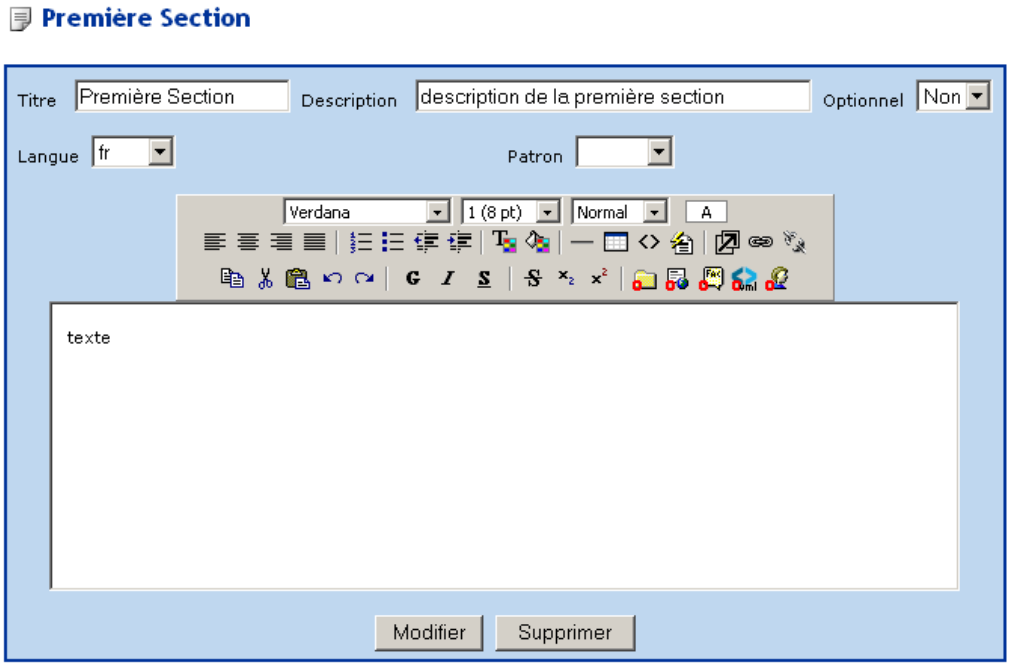
Il faut bien distinguer les sections des catégories. Physiquement, il est difficile de distinguer les deux. Une section est une sorte de menu que l'on peut placer à gauche ou à droite de la fenêtre. Dans chacune des sections, on peut y placer ce que l'on veut, le contenu est personnalisable. En revanche, une catégorie est prévue pour la hiérarchisation des articles. Même si on peut gérer les

catégories, les thèmes et les articles, seuls les sections peuvent être entièrement personnalisables.

Exemple : une section peut être créée pour simplement ajouter des liens vers d'autres sites sur internet. On peut aussi créer une section qui va faire le lien vers des articles du site grâce à un fichier OvML.

Ceci est possible grâce à l'éditeur qui accepte le HTML. Lors d'une création d'une section, on a 2 choix qui s'opèrent : html ou script. Si on créé une section avec l'option HTML, on a l'accès à l'éditeur wysiwyg pour mettre en forme le texte. On peut pousser la personnalisation de la section en ajoutant du code HTML. Attention : si le choix script existe, c'est pour avoir la possibilité d'intégrer du code Javascript. En effet, l'éditeur wysiwyg est pratique pour mettre en forme, mais il supprime les scripts Javascript.

L'onglet Ordre dans la catégorie Sections permet de modifier l'ordre d'affichage (de haut en bas) des différentes sections et permet de choisir si on veut ajouter la section à droite ou à gauche de l'écran.



Il est possible de rendre optionnel ou de désactiver les sections dans la liste des sections.

L'accès aux sections se trouve aussi dans la liste, on peut donc y spécifier les groupes qui y auront accès.

Dans le cadre de l'OvML, les sections sont un outil qui peut afficher de l'OvML ou un lien vers un fichier dans le site. Voir les exemples ci-dessous.

3.3.2 - Exemples de liens internes et popups

Tous les containers pour les articles, événements, faq ou fichiers, ont une variable contenant leur url d'accès à l'intérieur du site. Afin d'afficher le contenu sans la structure du site (sections, barre

de navigation), il faut rajouter le paramètre 'echo=1' dans l'url.

Lien interne vers une contribution d'un forum :
```
<OCPost postid="1">
        <a href="<OVPostUrl>&echo=1"> Cliquez ici </a>
</OCPost>
```

Affichage dans une popup :
Les variables donnant l'url d'un contenu en vue d'être affichées dans une popup tiennent compte de la désactivation des sections et barre de navigation. La fonction bab_popup d'OVIDENTIA peut être utilisée :

```
<OCArticle articleid="1">
        <a href="<OVArticlePopupUrl>" onclick="bab_popup( this.href ) ; return false;"> Cliquez pour lancer la popup </a>
</OCArticle>
```

3.3.3 - Exemple d'arborescence avec la fonction OFRecurse

Nous prenons l'exemple des catégories et des thèmes d'articles. Il peut être bien de créer une vue contenant l'arborescence complète des entités catégories et thème du site. La fonction OFRecurse permet de relancer un container autant de fois que désiré grâce à la récursivité.

Exemple avec une liste :
```
<ul>
  <OCArticleCategories>
  <li><OVCategoryName>
          <ul>
              <OFRecurse parentid="<OVCategoryId>">
              <OCArticleTopics categoryid="<OVCategoryId>">
              <li><OVTopicName></li>
              </OCArticleTopics>
          </ul>
  </li>
  </OCArticleCategories>
</ul>
```

3.4 - Exemples externes

- Module formulaires : récupération de données contenues dans les formulaires
- Module toolbox : création d'un diaporama d'images
- Module sitemap : réalisation d'une arborescence de navigation
- OVIDENTIA : modification de l'affichage des fiches d'annuaires
- Exemples sur Cantico.fr : vue planning des agendas de ressources, menu dynamique vertical, vue arborescente du gestionnaire de fichiers...