Manuel AFS Widgets

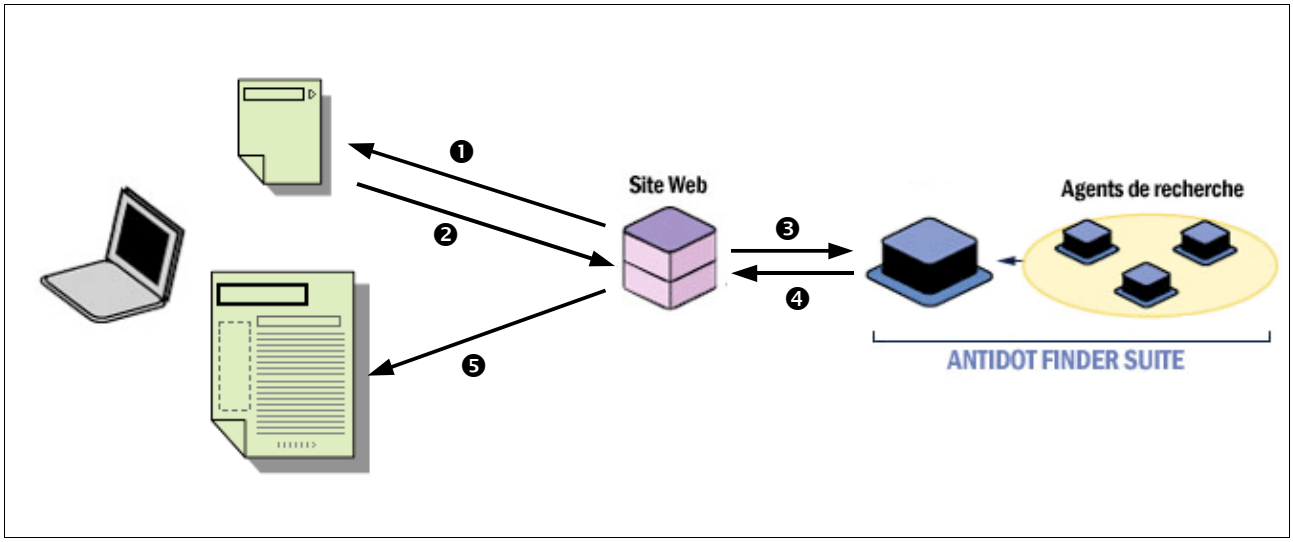
Version AFS Widgets :	1.2
Version de documentation :	6.5-1.2
Date :	2008-04-16
Référence :	AFS/DOC/...

Table des matières

1. Introduction

1.1. Présentation

Le moteur de recherche AFS fournit en natif des réponses au format XML. Ce flux de réponse peut être utilisé directement par des applications, ou transformé selon le besoin en un flux XML de format différent ou en HTML afin d'être affiché dans un navigateur Web.



Cette transformation XML->XML ou XML->HTML est réalisée au niveau du serveur par application d'une transformation XSLT.

Le schéma suivant illustre les différentes étapes de ce processus de génération des pages réponses au format HTML par le serveur AFS :

1. l'utilisateur navigue sur le site et les pages sont délivrées par le serveur Web (flux HTML) ;
2. lorsque l'utilisateur fait une recherche, celle-ci est envoyée au serveur Web ou au frontal AFS (requête HTTP) ;
3. cette requête est transmise au serveur AFS ;
4. AFS génère la réponse au format natif XML et ce flux XML est traduit en HTML par application d'une feuille de style XSL. Il en résulte une page réponse HTML ;
5. cette page réponse HTML est renvoyée au navigateur client qui l'affiche (flux HTML).

Ce mécanisme est extrêmement puissant et performant mais il nécessite des compétences adaptées et le besoin d'intervenir au niveau du serveur dès qu'une feuille de style XSL est modifiée.

C'est pourquoi, afin de faciliter l'intégration du flux de réponses AFS dans le cas courant de l'affichage au sein d'un navigateur Web, une solution alternative est proposée . Elle repose sur l'utilisation de widgets.

Un widget (nom issu de la contraction des mots « Window » et « gadget ») est un composant graphique élémentaire qui permet de réaliser l'affichage d'un élément d'information. L'assemblage de différents widgets permet de composer des pages de contenu.

Les widgets proposés par Antidot permettent d'intégrer rapidement et simplement le moteur de recherche AFS dans des pages HTML existantes, puisqu'aucune transformation n'est mise

en place au niveau du serveur et que le flux de réponse est consommé directement dans son format natif XML au niveau du poste client par le navigateur Web. Le flux est alors transformé en local par les widgets afin de réaliser l'affichage final.

A chaque élément qui compose une page de réponse AFS correspond un widget spécifique :

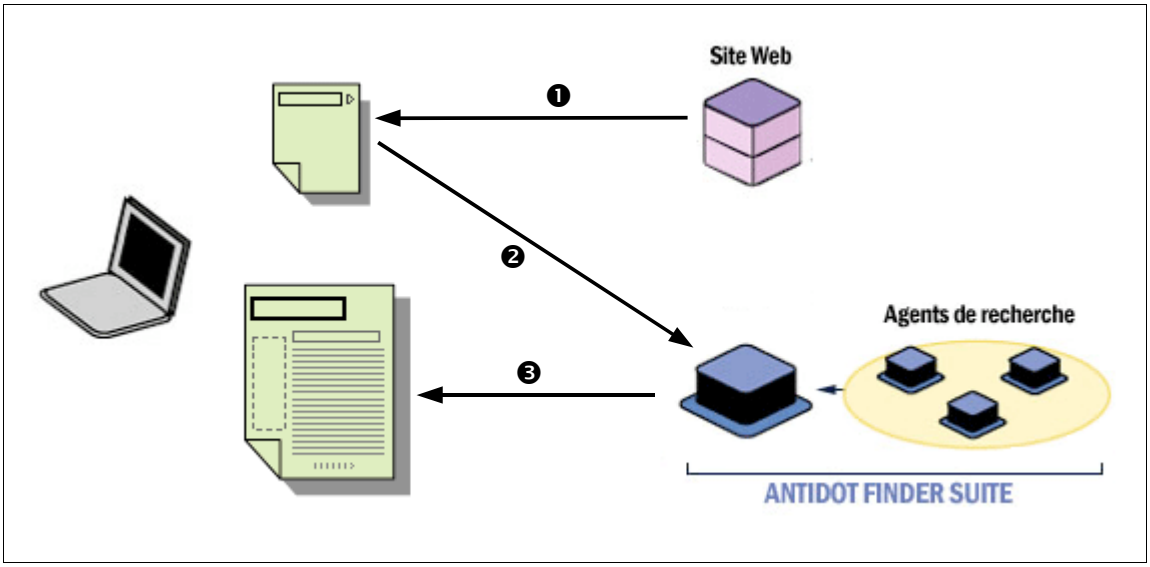
1. zone de rappel de la recherche courante ;
2. zone de saisie des mots clés ;
3. facettes pour le filtrage dynamique des réponses ;
4. affichage du nombre de réponses ;
5. suggestions de recherche contextuelles (RTE) ;
6. liens publicitaires (agent Kword) ;
7. pager de navigation entre les pages de résultat ;
8. zone d'affichage des résultats ;
9. ...

La mise en oeuvre de ces widgets se fait de façon très simple en incorporant des balises AWML (AFS Widget Markup Language) dans la page HTML. Chaque balise correspond à un widget et représente un élément du résultat de la recherche.

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:html="" xml:lang="fr"
   xmlns:awml="lib/AFSWidgets.xsd">
   <head>
      <title>Exemple partiel</title>
   </head>
   <body>
      <awml:widget type="keywords" style="display: none"></awml:widget>
      <awml:widget type="results" style="display: none"></awml:widget>
      <awml:widget type="pager" style="display: none"></awml:widget>
   </body>
</html>
```

1.2. Principe de fonctionnement des Widgets d'AFS

Les Widgets d'AFS sont des composants entièrement écrits en JavaScript (en utilisant la plate-forme de développement Open Source GWT[1]) et peuvent donc s'exécuter sur tous les navigateurs récents. A chaque recherche, ils interrogent directement le moteur de recherche grâce à la technologie AJAX et affichent les résultats, dans la page XHTML ouverte par l'internaute, et ce sans recharger la page.



Le schéma ci-dessus illustre le principe de fonctionnement :

1. l'utilisateur navigue sur le site et les pages sont délivrées par le serveur Web (flux HTML) ;
2. lorsque l'utilisateur fait une recherche, celle-ci est envoyée directement à AFS par l'intermédiaire d'une requête AJAX ;
3. AFS renvoie directement au navigateur le flux de réponse XML qui est alors récupéré par les widgets qui l'utilisent pour générer la page finale.

1 http://code.google.com/webtoolkit/

Ce type d'intégration a plusieurs avantages :

- Intégration rapide dans des pages existantes ;
- Mise en forme totalement libre (grâce au langage CSS) ;
- Aucun traitement côté serveur : le navigateur du client s'occupe de la mise en forme des résultats ;
- Aucune dépendance forte du côté de l'internaute : les widgets utilisant le JavaScript, il n'y a pas d'extension à installer.
- Aucun langage supplémentaire à connaître : le langage AWML est une extension de XHTML.

Par ailleurs, la compatibilité entre les différents navigateurs du marché est assurée par la bibliothèque GWT. Antidot vérifie la compatibilité totale des widgets avec les navigateurs suivants :

- Les navigateurs de type Mozilla (Firefox, Camino...) sur systèmes Windows et Linux ;
- Microsoft Internet Explorer 6 et supérieur ;
- Safari pour MacOS X Leopard.

D'autres navigateurs peuvent également être supportés.

1.3. Pré-requis pour l'intégration

Les widgets peuvent être mis en oeuvre par toute personne ayant des connaissances XHTML et CSS. Ceci inclut les personnes participant à la réalisation des pages d'un site. Le langage AWML est un ensemble de balises qui viennent s'ajouter à celles du XHTML. Leur appellation est étudiée pour une prise en main rapide.

Des connaissances de JavaScript permettent d'aller encore plus loin comme présenté dans la section 4.3.

Comme cela a été montré, aucune technologie particulière n'est nécessaire du côté serveur puisque les widgets interrogent le moteur de recherche d'AFS via une connexion AJAX. Ce type de connexion peut cependant nécessiter l'utilisation d'un proxy comme cela est décrit dans la section 2.7.

1.4. Plan du document

La section 2 de ce document explique le principe général de l'interaction entre les widgets et AFS.

La section 3 présente les widgets disponibles et leur spécification en AWML.

La section 4 propose des mécanismes pour enrichir l'utilisation des widgets.

La section 5 présente le kit mis à la disposition du webmestre.

La section 6 montre comment utiliser FireBug pour obtenir plus d'informations sur l'exécution des widgets.

La section 7 répond à l'ensemble des questions couramment rencontrées.

La dernière section fournit un lexique des termes techniques employés dans cette documentation.

2. Intégration des Widgets

Cette section présente les concepts de base pour l'intégration des Widgets d'AFS dans une page.

2.1. Principe général

Les Widgets d'AFS sont incorporés à une page XHTML par ajout de balises du langage AWML (AFS Widget Markup Language). Ces balises définissent :

- le service de recherche à interroger (balise « awml:service ») ;
- les widgets qui afficheront les résultats de recherche (balises « awml:widget »).

Au chargement de la page, le fichier JavaScript des widgets est interprété. L'interprétation du script lance l'analyse du XHTML pour en extraire les balises AWML et ainsi initialiser les widgets. Chaque balise « awml:widget » est alors remplacée par sa représentation graphique (en XHTML). La balise « awml:service » contient les informations générales sur le service : l'URL du service de recherche, ses paramètres fixes et diverses options liées à ce service.

Lors d'une recherche, une requête est lancée sur le service de recherche via la technologie AJAX supportée depuis plusieurs années par les navigateurs. La réponse du service de recherche est ensuite traitée par les widgets pour en afficher le contenu.

2.2. Fichiers à déployer

Le serveur Web doit héberger plusieurs fichiers (.html, .js, .png et .gif) contenant le JavaScript et les données annexes des widgets. Ces fichiers sont fournis par Antidot sous la forme d'une archive et peuvent être déployés dans n'importe quel dossier du serveur. Il est toutefois important que le nom de domaine d'accès à ces fichiers soit le même que celui d'accès à la page contenant les widgets. C'est une restriction de sécurité des navigateurs.

Pour plus d'informations sur cette restriction de sécurité, voir la section 2.7.

2.3. Préparation de la page

Les Widgets d'AFS exploitent la souplesse du XML (dont est dérivé le XHTML) pour définir leur propre langage AWML (AFS Widget Markup Language). La page hôte des widgets doit donc forcément être de type XHTML. Il n'est pas possible d'intégrer les widgets dans une page HTML (le HTML ne peut être étendu pour inclure de nouvelles balises). Plus précisément, et afin que votre page soit validée par le W3C, la déclaration DOCTYPE de la page doit être « XHTML 1.0 Transitional ».

Toutefois, il est possible d'intégrer à toute page (HTML comme XHTML) une zone de recherche qui ouvre la page de recherche intégrant les widgets tout en lançant une recherche. Pour plus d'informations, se reporter à la section 4.4.

La page XHTML[2] doit donc commencer par :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Ce DOCTYPE permet au navigateur d'interpréter correctement la page comme étant du XHTML. Il ne faut pas ajouter l'entête XML[3] car Microsoft Internet Explorer n'interprète alors plus le document comme du XHTML.

2 La structure globale de la page XHTML se trouve à la fin de cette section.
3 L'entête XML correspond à la balise <?xml...>

La balise « html » doit déclarer l'espace de nommage habituel du XHTML mais également l'espace de nommage « awml »[4] et l'espace de nommage « html »[5]. Ces deux espaces de nommage doivent obligatoirement porter ces noms ; il n'est pas possible d'en changer.

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:html="" xml:lang="fr"
   xmlns:awml="lib/AFSWidgets.xsd">
```

La section « head » de la page doit inclure l'appel au fichier JavaScript d'initialisation des widgets. Sans cet appel, les balises AWML ne sont pas interprétées.

```
<head>
   <script type="text/javascript"
      src='net.antidot.widgets.AFSWidgets.nocache.js'></script>
</head>
```

Enfin, le corps de la page doit inclure une balise « iframe » particulière afin de gérer l'historique de recherche.

```
<body>
   <iframe src="javascript:''" id="__gwt_historyFrame"
      style="width: 0; height: 0; border: 0"></iframe>
</body>
```

Il faut ensuite ajouter une balise « awml:service » comme présenté dans la section suivante.

La page XHTML doit comporter un minimum de code pour assurer le bon fonctionnement de l'application. Au final, la structure[6] de la page XHTML, qu'il est nécessaire de respecter, est donc :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:html="" xml:lang="fr"
   xmlns:awml="lib/AFSWidgets.xsd">
   <head>
      <script type="text/javascript"
         src='net.antidot.widgets.AFSWidgets.nocache.js'></script>
   </head>
   <body>
      <iframe src="javascript:''" id="__gwt_historyFrame"
         style="width: 0; height: 0; border: 0"></iframe>
   </body>
</html>
```

2.4. Le service de recherche

La balise « awml:service » fournit aux widgets diverses informations sur le service de recherche à utiliser :

- l'URL d'appel ;
- les paramètres fixes (ils seront toujours envoyés au moteur, quelle que soit la recherche effectuée) ;
- diverses options d'interprétation des résultats et/ou de comportement global.

4 L'espace de nommage « awml » permet d'inclure du AWML au sein de la page XHTML
5 L'espace de nommage « html » permet d'inclure du XHTML au sein des balises AWML (voir la section 2.8.2)
6 Il faut noter que la page XHTML doit obligatoirement respecter cette structure pour assurer le bon fonctionnement de l'application. De plus, la déclaration du service de recherche et des widgets doit être effectuée au sein de la balise <body>

La **balise « awml:service »** expose les **attributs** suivants :

Nom	***Description***	***Requis***	***Par défaut***
url	L'URL d'appel du moteur de recherche (voir la section 2.7 pour les restrictions de sécurité)	Non	/cgi-bin/findall
onclear	Ces attributs permettent d'associer du JavaScript à divers événements des widgets. Ils sont détaillés dans la section 4.3.	Non	
onerror			
onload			
onsearch			
onendsearch			
oncancelsearch			

La balise « awml:service » supporte également les sous-balises :

- « awml:param » pour définir des paramètres d'appel du service de recherche. L'attribut « name » correspond au nom du paramètre, le contenu de la balise à la valeur. Ainsi, dans l'exemple en fin de cette section, l'URL de recherche commencera toujours par « /rechercher?C=1 » ;
- « awml:option » pour définir des options liées au service de recherche (la liste des options possibles est donnée ci-dessous). L'attribut « name » correspond alors au nom de l'option, sa valeur étant dans le contenu de la balise.

Les options possibles sont :

Nom	**Description**	**Requise**	**Par défaut**
usePageParams	Si « true » et si l'URL d'appel de la page inclut des paramètres de recherche, la recherche se lance automatiquement au chargement de la page à partir des paramètres de l'URL. Sinon doit être mis à « false ».	Non	true
disableValidation	Si « true », désactive la validation du XML issue du moteur de recherche. Une limitation du navigateur Microsoft Internet Explorer nécessite la mise à « true » de cette option. Sinon doit être mis à « false ».	Non	false
scrollOnLoad	Si « true », remonte le focus du navigateur au début de la page à chaque nouvelle recherche. Sinon doit être mis à « false ».	Non	true
timeout	Le délai d'expiration des requêtes AJAX en millisecondes. Passé ce délai, une requête AJAX vers le moteur de recherche est annulée et une erreur est remontée.	Non	30000 (30s)
toolbar	Si « true » active la barre d'outils (voir la section 6.3). Sinon doit être mis à false.	Non	false

Si plusieurs balises « awml:param » portent le même nom de paramètre (même valeur pour l'attribut « name »), le paramètre prendra la valeur de la dernière de ces balises. De même pour les balises « awml:option ».

Voici un exemple qui déclare :

- l'URL de recherche comme étant « /rechercher » ;
- le paramètre « C » qui a pour valeur 1 (c'est le numéro de votre service AFS) ;
- l'option « usePageParams » à « true » pour activer la recherche automatique au chargement de la page.

```
<awml:service url="/rechercher" style="display: none">
   <awml:param name="C">1</awml:param>
   <awml:option name="usePageParams">true</awml:option>
</awml:service>
```

2.4.1. Déclarer les tris

L'utilisation des tris permet de classer les résultats en fonction d'un ou plusieurs critères, chacun dans un ordre ascendant ou descendant. Pour informer le moteur du tri, il faut préciser le paramètre « SORT_ORDER ». En ajoutant une balises « awml:orders » dans la déclaration du service, les Widgets gèrent automatiquement l'ajout de ce paramètre.

Pour plus d'informations sur l'utilisation du paramètre « SORT_ORDER », n'hésitez pas à demander la documentation auprès du support à l'adresse support@antidot.net ou de demander directement à votre contact privilégié chez Antidot.

La balise « awml:orders », à ajouter à l'intérieur d'un balise « awml:service », peut contenir une à plusieurs balises « awml:order » qui déclare, chacune, un tri. Un tri a un nom (attribut « name »), un libellé (balise « awml:labell ») et un à plusieurs critères sur lequel porte le tri (balise « awml:field »).

La **balise « awml:orders »** propose **les attributs** suivants :

Nom	***Description***	***Requis***	***Par défaut***
default	Nom du tri activé par défaut. Ce tri est activé au chargement de la page, si aucune recherche n'est à lancer dès le chargement (voir l'option « usePageParams » du service).	Non	Aucun tri
force	Nom du tri à utiliser pour chaque recherche n'ayant pas de tri.	Non	Aucun tri

La **balise « awml:order »** propose l'**attribut** suivant :

Nom	***Description***	***Requis***
name	Nom du tri. Cet attribut permet de différencier les tris.	Oui

La **balise « awml:order »** contient à son tour **une balise « awml:label »** qui associe un libellé au tri. Il est utiliser par exemple par le Widget « context » (section 3.1.5) ou le Widget « sort » (section 3.1.7). La balise « awml:order » contient également **une à plusieurs balises « awml:field »** qui précisent les critères de tri. Les critères sont appliqués dans l'ordre, c'est-à-dire que le tri à lieu sur le premier critère puis, pour chaque ensemble de réponses ayant la même place pour ce premier critère, le second critère est utilisé pour les trier, et ainsi de suite.

Le contenu de la **balise « awml:label »** définit le libellé associé au tri. Contrairement à la plupart des libellé, elle ne peut pas contenir de HTML. De plus elle ne possède pas d'attribut.

La **balise « awml:field »** propose **l'attribut** suivant :

Nom	*Description*	*Requis*	*Par défaut*
order	Sens du tri. Cette attribut peut prendre la valeur « ASC » pour un tri ascendant et « DESC » pour un tri descendant.	Non	ASC

Le contenu de la balise « awml:field » indique le critère sur lequel est effectué le tri. Les valeurs possibles sont :

- Un champs SHM ;
- Une information associé au résultat par le moteur de recherche : pertinence (« afs:relevance »), le poids (« afs:weight »), distance entre les mots recherchés dans le résultat (« afs:pathlen »)...

La documentation du moteur de recherche donne plus d'informations sur les critères disponibles.

Voici un exemple de déclaration des tris au sein de la balise « awml:service » :

```
<awml:orders default="pertinence">
   <awml:order name="pertinence">
      <awml:label>Tri par pertinence</awml:label>
      <awml:field order="ASC">afs:relevance</awml:field>
      <awml:field order="DESC">Localisation</awml:field>
      <awml:field order="ASC">Type</awml:field>
   </awml:order>
   <awml:order name="localisation">
      <awml:label>Tri par localisation</awml:label>
      <awml:field order="ASC">Localisation</awml:field>
   </awml:order>
      <awml:order name="type">
      <awml:label> Tri par type</awml:label>
      <awml:field order="DESC">Type</awml:field>
   </awml:order>
</awml:orders>
```

Dans cet exemple, trois tris sont proposés :

- le tri nommé « pertinence » qui est activé par défaut. Le paramètre « SORT_ORDER » correspondant est « afs:relevance|ASC-Localisation|DESC-Type|ASC » ;
- le tri « localisation » qui produit un paramètre « SORT_ORDER »à « Localisation|ASC » ;
- le tri « type » ayant pour valeur « Type|DESC ».

Le Widget « sort » permet à l'utilisateur de choisir le tri à utiliser (voir la section 3.1.7).

2.4.2. Définir les clusters

Cette fonctionnalité est détaillée dans la section 3.3.3.

2.4.3. Définir les groupes

Cette fonctionnalité est détaillée la section 3.3.4.

2.5. Les Widgets

La balise « awml:widget » permet de déclarer un widget. Cette balise doit se trouver dans le corps de la page XHTML (ie. dans la balise « body ») et peut être la fille de n'importe quelle balise XHTML. Le widget apparaîtra alors en lieu et place de la balise « awml:widget », ce qui offre une grande souplesse dans l'agencement des widgets.

Cette section expose les principes généraux des widgets. La section 3 présente plus en détail tous les widgets.

2.5.1. Différents types de widget

Les widgets exploitent le flux XML de réponse du moteur de recherche pour générer leur vue. Chaque widget a une fonction propre :

- Afficher les résultats pour le widget de type « results » ;
- Permettre d'accéder aux pages de résultats pour le widget « pager » ;
- Offrir une zone de saisie des mots clés à rechercher pour le widget « keywords » ;
- ...

Le type du widget est déclaré via l'attribut « type ». Ainsi, le widget de saisie des mots clés se déclare de la façon suivante :

```
<awml:widget type="keywords" style="display: none"></awml:widget>
```

Les widgets disponibles sont :

Type	*Description*
autoFacets	Affiche les facettes (filtres d'affinage et de sélection) par détection automatique
clusteredResults	Affiche les résultats de la recherche lorsque des clusters sont définis
clusters	Affiche les clusters définis et permet de passer de l'un à l'autre
context	Affiche les paramètres fixés (mots-clés, groupe, cluster, facettes) pour la recherche en cours et permet de les retirer
facets	Affiche les facettes (filtres d'affinage et de sélection) configurées
groupedResults	Affiche les résultats de la recherche lorsque des groupes sont définis
hint	Affiche les suggestions orthographiques
history	Affiche l'historique des recherches
info	Affiche une ligne d'information concernant la recherche (nombre de réponse, durée...)
keywords	Permet la saisie des mots clés à rechercher
kword	Affiche les promotions
pager	Affiche la liste des pages de résultats et permet de changer de page
results	Affiche les résultats de la recherche
rte	Affiche les expressions proches
sort	Affiche la liste des tris des résultats proposés et permet d'activer un tri parmi la liste

La section 3 présente en détail chaque type de widget.

Une même page peut contenir plusieurs widgets d'un même type. La plupart des widgets peuvent être paramétrés à l'aide d'options. Ces options permettent de modifier le comportement d'un widget. Ainsi, le widget « info » peut être mis deux fois : une fois pour afficher le nombre de résultat et une autre fois pour afficher la durée de la recherche. Ou encore il est possible de mettre deux fois le même « pager » dans la page (un en haut au début de la liste des réponses et un en bas de page)...

2.5.2. Les options

La plupart des widgets disposent d'options qui permettent de modifier leur comportement. Par exemple, le widget « keywords » propose l'option « keepFacets » qui permet de conserver ou pas les filtres déjà positionnés quand un nouveau mot clé est recherché.

Ces options sont modifiables par l'ajout de balises « awml:option » dans le corps de la balise « awml:widget ». L'attribut « name » de ces balises correspond au nom de l'option, leur contenu à la valeur. Pour reprendre l'exemple précédent, voici comment conserver les filtres lors d'une nouvelle recherche :

```
<awml:widget type="keywords" style="display: none">
   <awml:option name="keepFacets">true</awml:option>
<awml:widget>
```

Si plusieurs balises « awml:option » portent le même nom d'option (même valeur pour l'attribut « name ») à l'intérieur d'une balise « awml:widget », c'est la valeur de la dernière déclarée qui sera conservée.

La liste des options supportées par chaque widget est donnée dans la section 3.

2.5.3. Les libellés

Certains widgets offrent la possibilité d'afficher du texte (en plus des informations extraites du flux de réponse XML). Par exemple, le widget affichant les expressions proches (RTE) permet de faire précéder la liste des expressions par un texte.

Ces textes sont appelés libellés. Ils sont modifiables par l'ajout de balises « awml:label » dans le corps de la balise « awml:widget ». L'attribut « name » de ces balises correspond au nom du libellé, et le contenu de la balise correspond à la valeur du libellé, c'est-à-dire au texte à afficher. Quand un libellé n'a pas de nom (c'est le cas pour certains widgets qui n'offrent qu'un seul libellé), l'attribut « name » doit être omis. C'est le cas, par exemple, pour le widget « rte » :

```
<awml:widget type="rte" style="display: none">
   <awml:label>Les RTE</awml:label>
</awml:widget>
```

Il est possible d'inclure dans les libellés, du HTML, ce qui donne plus de souplesse dans la mise en forme. Le HTML ainsi ajouté doit être du XHTML et toutes les balises doivent être précédées de l'espace de nommage « html » (comme expliqué dans la section 2.8.2).

```
<awml:widget type="rte" style="display: none">
   <awml:label>Les <html:strong>RTE</html:strong></awml:label>
</awml:widget>
```

Si plusieurs balises « awml:label » portent le même nom de libellé (même valeur pour l'attribut « name ») à l'intérieur d'une balise « awml:widget », le libellé prendra la valeur de la dernière de ces balises.

La liste des libellés supportés par chaque widget est donnée dans la section 3.

2.5.4. Les libellés paramétrés

Certains libellés, définis avec la balise « awml:label », proposent l'injection de valeurs issues du résultat dans le libellé. Cette injection est possible grâce à l'inclusion de séquences de caractères particulières dans le libellé. Le principe est similaire à celui des « printf » en C, C++, PHP... et repose complètement sur celui de la classe java.text.MessageFormat en Java.

Avant l'affichage du libellé, le widget recherche les séquences de caractères commençant par une accolade ouvrante (« {«) et terminant par une accolade fermante (« } »). Chaque séquence doit contenir un numéro correspondant à l'information à injecter dans le libellé et peut également inclure des informations de formatage.

Ce principe est décrit dans la documentation de la classe java.text.MessageFormat de l'API Java (http://java.sun.com/j2se/1.4.2/docs/api/java/text/MessageFormat.html).

Prenons l'exemple du widget « info » qui propose d'afficher un libellé. Ce libellé (sans nom), peut faire référence à 3 informations : les mots-clés (référence 0), le nombre de réponse (référence 1) et la durée (référence 2). Le code suivant illustre l'affichage de ces informations :

```
<awml:widget type="info" style="display: none">
   <awml:label>
      Recherche de <html:em>{0}</html:em>,
      {1,choice,0#aucune réponse|1#1 réponse|2#{1,number} réponses}
      en {2} secondes
   </awml:label>
</awml:widget>
```

La syntaxe des injections est :

- le numéro de la référence ;
- le type de formatage ;
- les options de formatage spécifique au type.

Seule la référence est obligatoire. Les options de formatage ne peuvent être précisées que si le type de formatage est précisé. Enfin, chaque élément est séparé par une virgule.

Les types de formatage disponibles sont :

type	***option***	***L'information est formatée...***
		... automatiquement suivant son contenu.
« number »		... comme un nombre. Cela revient à préciser « integer » comme option
	« integer »	... comme un nombre
	« currency »	... comme une valeur monétaire
	« percent »	... comme un pourcentage
	Chaîne de format	... suivant le formatage de nombre donné.
« date »		... comme une date.
	« short »	... comme une date en utilisant un format compact.
	« medium »	... comme une date en utilisant un format court.
	« long »	... comme une date en utilisant un format long.
	« full »	... comme une date en utilisant un format verbeux.
	Chaîne de format	... suivant le formatage de date donné.
« time »		... comme une heure.
	« short »	... comme une heure en utilisant un format compact.
	« medium »	... comme une heure en utilisant un format court.
	« long »	... comme une heure en utilisant un format long.
	« full »	... comme une heure en utilisant un format verbeux.
	Chaîne de format	... suivant le formatage d'heure donnée.
« choice »	Choix	... comme expliqué plus loin.

Le type « number » propose l'utilisation d'une chaîne de format pour personnaliser pleinement l'affichage des nombres. Cette chaîne est constituée de caractères dont certains ont une signification particulière :

Caractère	***Signification***
0	Un chiffre
#	Un chiffre s'il est différent de 0
.	Le séparateur de décimal
-	Un signe – indiquant un nombre négatif
,	Un séparateur de millier
E	Le séparateur de mantisse/exposant pour les nombres scientifiques.
;	Sépare la représentation des nombres positifs (à gauche) des nombres négatifs (à droite)
%	Affiche le nombre en pourcentage
'	Indique le début et la fin d'une chaîne de caractères. Les caractères situés entre deux ' sont affichés tel quel. Pour afficher un ', il faut en mettre 2.

La documentation de la classe Java DecimalFormat (en anglais) offre plus d'informations sur ce formatage : http://java.sun.com/j2se/1.4.2/docs/api/java/text/DecimalFormat.html.

De même, le type « date » propose les caractères particuliers suivants :

Caractère	***Signification***
y	L'année
M	Le mois
w	La semaine dans l'année
D	Le jour dans l'année
d	Le jour dans le mois
E	Le libellé du jour de la semaine

La répétition d'un même caractère permet d'avoir l'information en version courte ou longue. Ainsi, si le caractère « y » est répété au moins 4 fois, l'année apparaît sur 4 chiffres. La documentation de la classe Java SimpleDateFormat (en anglais) offre plus d'informations sur ce formatage : http://java.sun.com/j2se/1.4.2/docs/api/index.html.

Enfin, le type « time » propose les caractères particuliers suivants :

Caractère	***Signification***
H	L'heure sur 24 heures
K	L'heure sur 12 heures
a	AM ou PM suivant l'heure
m	Les minutes
s	Les secondes
S	Les millisecondes

De même que pour le format « date », la répétition d'un caractère permet d'afficher une représentation plus longue de l'information. Par exemple si l'heure est « 3 » et que la chaîne de format contient « HH », alors il sera affiché « 03 ». La documentation de la classe Java SimpleDateFormat (en anglais) offre plus d'informations sur ce formatage : http://java.sun.com/j2se/1.4.2/docs/api/index.html.

La description des widgets de la section 3 précise, pour chaque widget, les libellés paramétrables et les informations qui peuvent être injectées.

2.5.5. Internationalisation

2.5.5.1. Les libellés

Les widgets permettent de facilement internationaliser la page XHTML de recherche via les balises « awml:label ». Le jeu de caractères supporté par ces balises est exactement le même que celui de la page.

Voici un exemple du widget d'affichage des informations en français :

```
<awml:widget type="info" style="display: none">
   <awml:label>
      Recherche de <html:em>{0}</html:em>,
      {1,choice,0#aucune réponse|1#1 réponse|2#{1,number} réponses}
      en {2} secondes
   </awml:label>
</awml:widget>
```

Recherche de **accueil**, **163** réponses en **0.358** secondes

Voici un exemple du widget d'affichage des informations en italien :

```
<awml:widget type="info" style="display: none">
   <awml:label>
      Ricerca di <html:em>{0}</html:em>,
      {1,choice,0#niente risposta|1#1 risposta|2#{1,number} risposte}
      in {2} secondi
   </awml:label>
</awml:widget>
```

Ricerca di **accueil**, **163** risposte in **0.35** secondi

Voici un exemple du widget d'affichage des informations en anglais :

```
<awml:widget type="info" style="display: none">
   <awml:label>
      Search for <html:em>{0}</html:em>,
      {1,choice,0#no answer|1#1 answer|2#{1,number} answers}
      in {2} seconds
   </awml:label>
</awml:widget>
```

Search for **accueil**, **163** answers in **0.358** seconds

2.5.5.2. Les nombres

Selon le pays, la norme d'affichage des nombres et des dates est différente. Pour forcer l'affichage des nombres selon la norme d'un pays, il faut ajouter la balise « meta » suivante dans la section « head » :

```
<meta name="gwt:property" content="locale=fr_FR">
```

L'attribut « content » indique la langue par défaut du site Web. Par exemple, « locale=fr_FR » correspond à la langue française et « locale=en_US » correspond à la langue anglaise.

Dans cet exemple, l'ajout de la balise « meta » ci-dessus permet de formater le nombre de secondes selon la norme française[7] :

Recherche de **accueil**, **169** réponses en **0,414** secondes

Sans cette balise, le nombre de secondes respecte la norme anglaise[8] :

Recherche de **accueil**, **169** réponses en **0.418** secondes

Les widgets supportent la mise en forme des nombres pour la langue anglaise et la langue française. N'hésitez pas à contacter votre support client si vous souhaitez disposer d'autres formats de nombres.

2.6. Mise en forme

La mise en forme des widgets est entièrement paramétrable à l'aide de feuilles de style. Le XHTML produit par chaque widget est peu structuré et offre de nombreuses classes CSS pour permettre l'intégration totale dans la plupart des chartes graphiques.

2.6.1. Les classes CSS

Chaque widget a au moins une classe CSS : celle permettant de mettre en forme le widget globalement. Il peut ensuite offrir d'autres styles pour affiner la mise en forme.

La section 3 présente les widgets en détail et liste les classes CSS qui s'appliquent à chaque widget.

Les widgets peuvent être insérés dans n'importe quelle balise HTML. Ils n'interfèrent donc pas dans la structuration de la page. De plus, les balises « awml:widget » peuvent inclure un attribut « id » unique. Cet attribut est alors transféré sur la balise « div » qui contient tout le XHTML du widget. Cela permet d'appliquer des styles CSS sur un widget en particulier.

Par exemple, le code :

```
<style type="text/css">
  #MonWidget {
    border: red 3px dotted;
  }
</style>

<awml:widget id="MonWidget" type="info" style="display: none">
   <awml:label>Recherche de <html:em>{0}</html:em></awml:label>
</awml:widget>
```

7 Le séparateur, utilisé dans le format des nombres de la norme française, est la virgule.
8 Le séparateur, utilisé dans le format des nombres de la norme anglaise, est le point.

affiche :

Recherche de *antidot*

2.6.2. Les sous-classes CSS d'état

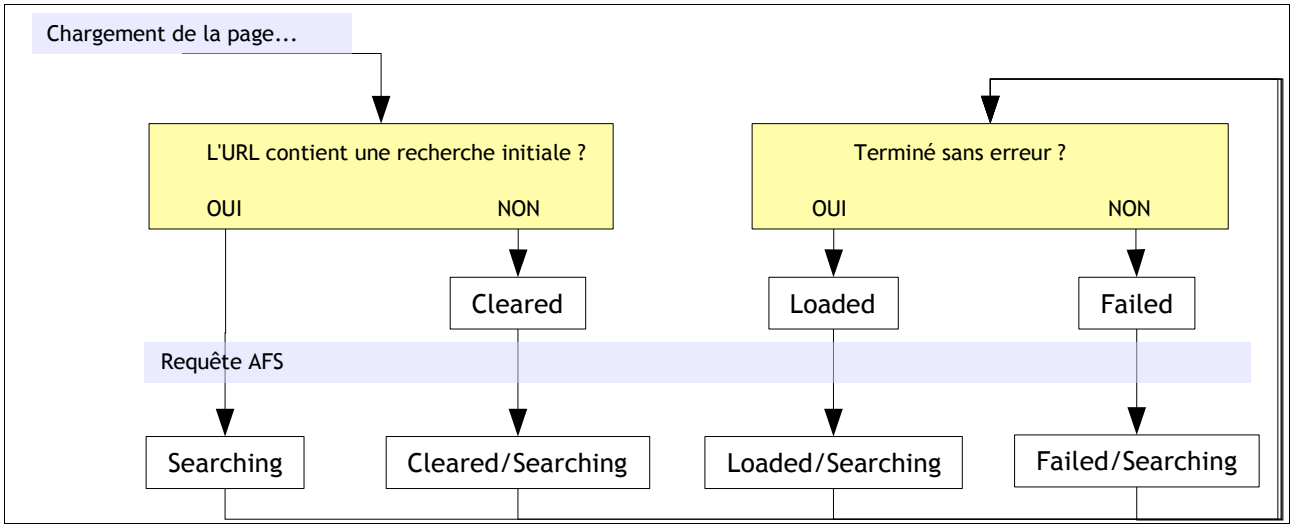
Les widgets proposent un mécanisme particulier, concernant les classes CSS, qui permet la modification d'une classe CSS en fonction de l'état du widget. Ce mécanisme est en fait permis par la bibliothèque GWT et repose sur l'ajout d'un suffixe à la classe CSS.

Par exemple, supposons que la classe CSS appliquée à un élément (widget ou partie d'un widget) soit « MaClasse ». L'élément est alors mis en forme par la classe « MaClasse ». Si on lui ajoute la sous-classe « -Caché », alors l'élément sera mis en forme par les classes « MaClasse » et « MaClasse-Caché ». La sous-classe fonctionne alors comme un suffixe de la classe CSS.

Ces sous-classes CSS d'état peuvent être vues comme des pseudo-classes CSS si ce n'est qu'elles commencent par un tiret (« - »). Elles n'interfèrent pas avec les pseudo-classes ou les sélecteurs CSS mais viennent les enrichir.

Tous les widgets proposent les sous-classes suivantes qui s'appliquent au style global du widget :

Nom	***Description***
-Searching	Une recherche est en cours. Cette sous-classe n'est définie que lorsqu'une recherche est en cours.
-Loaded	Des résultats sont disponibles. Cette sous-classe vient remplacer toute sous-classe -Cleared ou -Failed précédemment définie.
-Cleared	Le widget a été vidé[9]. Cette sous-classe vient remplacer toute sous-classe -Loaded ou -Failed précédemment définie.
-Failed	Une erreur s'est produite lors d'une recherche. Cette sous-classe vient remplacer toute sous-classe -Loaded ou -Cleared précédemment définie.



Les widgets dont l'affichage dépend des réponses du moteur proposent également la sous-classe « -Set ». Suivant les widgets, elle s'applique sur la classe du widget (« hint », « rte »...), sur la classe d'une sous-balise (« results », « clusteredResults »...) ou les deux

9 Cela se produit lors du chargement de la page s'il n'y a pas de recherche initiale

(« facets »). Cette sous-classe vient en complément de la sous classe -Loaded pour indiquer que le widget affiche des résultats.

Par exemple, pour le widget « rte », la sous-classe est définie seulement si des RTE sont présentes dans les résultats. Cela permet de masquer le widget via CSS lorsqu'il est vide[10].

2.6.3. Les widgets repliables

Certains widgets sont repliables : un clic sur leur libellé ajoute une sous-classe CSS particulière au widget, un autre clic la retire. Le contenu du widget étant regroupé dans un élément avec une classe CSS particulière, il devient facile de faire disparaître ce contenu en fonction de la présence de cette sous-classe.

La sous-classe CSS est « -Folded ».

Les widgets proposant cette fonctionnalités sont : les facettes, context, clusters, sort et history.

Ainsi, le widget « context » est découpé en deux partie :

- un bandeau de classe CSS « AFSContextWidget_Header » qui contient le libellé ;
- le reste du widget dans un élément de classe CSS « AFWContextWidget_Content ».

Suivant le nombre de clic sur le libellé, la classe CSS « AFWContextWidget » est suffixé ou non de la sous-class CSS « -Folded ».

L'instruction CSS suivante permet de masquer ou non le contenu suivant la présence de la sous-classe :

```
.AFSContextWidget-Folded .AFSContextWidget_Content {
   display: none;
}
```

Au chargement de la page, la sous-classe n'est pas appliqué. Donc l'exemple ci-dessus affichera le contenu du widget au chargement de la page et le masquera au premier clique. En retirant « -Folded » de l'exemple, le contenu sera masqué au chargement et affiché au premier clic.

2.7. Restrictions de sécurité

Les widgets interrogent le moteur de recherche d'AFS via une connexion AJAX. C'est une technologie disponible sur les navigateurs récents qui permet à un programme JavaScript de dialoguer avec un serveur sans charger complètement une nouvelle page HTML.

Ce type de connexion connaît toutefois une restriction de sécurité : le serveur contacté par le JavaScript doit forcément être le serveur qui a produit la page XHTML contenant le JavaScript. Donc si la page XHTML a pour adresse « http://www.mondomaine.com/page.html » alors le JavaScript ne peut lire que les pages dont l'URL commence par « http://www.mondomaine.com/ ». Ainsi, si le moteur de recherche est accessible grâce à l'URL « http://recherche.mondomaine.com/cgi-bin/findall », il n'est pas possible aux widgets d'interroger directement le moteur.

Pour contourner cette limitation, Antidot propose d'utiliser un proxy sur le serveur de la page XHTML (« www.mondomaine.com »). Antidot met à disposition de ses clients diverses implémentations de proxy :

- Par configuration d'Apache (grâce à l'extension « mod_proxy ») ;
- Par un script PHP ;

10 Un Widget est vide lorsqu'il ne contient pas de données

- Par un Servlet Java ;
- Par un module C# ;
- ...

Les widgets interrogent alors le proxy (par exemple via l'URL « http://www.mondomaine.com/rechercher ») et ce dernier transmet la requête au moteur. Quand le moteur répond, le proxy transmet la réponse aux widgets.

Pour plus d'informations sur les proxy proposés par Antidot, n'hésitez pas à contacter le support à l'adresse support@antidot.net ou de demander directement à votre contact privilégié chez Antidot.

Pour plus d'informations sur la restriction de domaine, n'hésitez pas à consulter :

- http://www.mozilla.org/projects/security/components/same-origin.html ;
- http://en.wikipedia.org/wiki/Same_origin_policy.

2.8. Remarques importantes

Cette section regroupe des remarques qui n'ont pas forcément de lien entre elles mais qu'il est important de prendre en compte pour s'assurer d'une intégration réussie.

2.8.1. Toujours inclure une balise fermante

Le XML (et donc le XHTML) requiert que toute balise soit fermée et permet d'écrire les balises sans contenu sous la forme d'une balise vide (la balise ouvrante se termine alors par le caractère « / » : « <balise/> »). Toutefois certains navigateurs n'interprètent pas correctement les balises vides. Il est donc fortement recommandé de toujours écrire les balises AWML ouvrantes et fermantes, même s'il n'y a pas de contenu. Cela touche également les balises de l'espace de nommage « html » qui permet d'inclure du HTML dans le AWML.

2.8.2. Préfixer le XHTML inclus dans le AWML par « html: »

Certaines balises AWML permettent l'inclusion de XHTML dans leur contenu. Par exemple, les balises « awml:label », qui permettent de définir un libellé à afficher dans un widget, offrent la possibilité que ce libellé inclut du XHTML pour un formatage plus poussé.

Ces balises XHTML ne peuvent être écrites directement car certains navigateurs traitent mal le mélange de XHTML avec des balises d'autres types. C'est pourquoi la balise « html » inclut une déclaration pour l'espace de nommage « html » et que toutes les balises XHTML incluses dans des balises AWML doivent être préfixées de « html: ».

Par exemple :

```
<awml:widget type="rte" style="display: none">
   <awml:label>
      Les <html:strong>expressions proches</html:strong> :
   </awml:label>
</awml:widget>
```

2.8.3. Masquer les balises AWML et leur contenu

Les navigateurs ignorent les balises qu'ils ne connaissent pas mais pas leur contenu. Si du texte est inclus dans une balise AWML (comme dans les balises « awml:label »), alors le navigateur affichera ce texte lors du chargement de la page. Rapidement le JavaScript des widgets est exécuté et ce texte disparaît.

Mais sur certains navigateurs, la page est rendue une première fois avant l'exécution du JavaScript puis une nouvelle fois après (moteur de type Gecko principalement). Ce qui produit un flash sur le texte : il apparaît et disparaît rapidement.

Pour éviter ce problème, il est possible de fixer l'attribut CSS « display » à « none » pour toutes les balises « awml:service » et « awml:widget ».

Voici un exemple :

```
<awml:widget type="rte" style="display: none">
   <awml:label>Les expressions proches :</awml:label>
</awml:widget>
```

Il est à noter que l'attribut « style » des balises AWML n'influence en rien le style CSS des widgets.

3. Détails des widgets

Cette section présente plus en détail les différents widgets. Chaque widget est décrit par son rôle, les classes CSS, les libellés, les options et les balises définies pour ce widget. La spécification des widgets est également accompagnée d'un exemple de code AWML et d'une capture d'écran du résultat.

Cette section se découpe en 4 sections :

- les widgets de navigation (« keywords », « pager », « facets », « autoFacets », « context », « clusters », « sort ») ;
- les widgets d'état (« info ») ;
- les widgets des résultats (« results », « clusteredResults », « groupedResults ») ;
- les autres widgets (« hint », « kword », « rte », « history »)

Il est recommandé de lire la section 2 avant celle-ci.

3.1. Les widgets de navigation

3.1.1. Widget « keywords »

Le widget « keywords » permet à l'utilisateur de saisir un mot clé et de lancer une nouvelle recherche avec ce mot clé. Il propose également la suggestion d'expression à rechercher au cours de la frappe.

Pour plus d'informations sur la suggestion automatique proposés par Antidot, n'hésitez pas à contacter le support à l'adresse support@antidot.net ou de demander directement à votre contact privilégié chez Antidot.

Il supporte les **classes CSS** suivantes :

- *AFSKeywordsWidget* : le widget ;
- *AFSKeywordsWidget_Go* : le bouton de lancement de la recherche ;
- *AFSKeywordsWidget_Suggests* : la liste des suggestions ;
- *AFSKeywordsWidget_Suggests .item* : un élément de la liste des suggestions ;
- *AFSKeywordsWidget_Suggests .item-selected* : l'élément sélectionné de la liste des suggestions.

Exemple :

```
.AFSKeywordsWidget {
   display: inline;
   margin-left: 10pt;
}
.AFSKeywordsWidget_Go {
   border: 1px solid #777777;
   cursor: pointer;
}
.AFSKeywordsWidget_Suggests {
   border: 1px solid #000000;
   background: #EFE9DA;
}
.AFSKeywordsWidget_Suggests .item {
   padding: 2px;
}
```

```
.AFSKeywordsWidget_Suggests .item-selected {
   background: #DED8C8;
}
```

Le **libellé** suivant est mis à disposition :

Nom	*Description*	*Requis*	*Par défaut*
button	HTML du bouton lançant la recherche	Non	<html:button>OK</html:button>

Les **options** suivantes sont disponibles :

Nom	*Description*	*Requis*	*Par défaut*
keepFacets	« true » pour conserver les valeurs des facettes actives lors d'une nouvelle recherche ; « false » sinon.	Non	false
focusAtStartup	« true » pour positionner le curseur dans la zone de saisie du mot clé au chargement de la page dans le navigateur ; « false » sinon.	Non	true
suggestURL	L'URL du serveur de suggestion.		
suggestTimeout	Le délai d'expiration en millisecondes d'une requête vers le serveur de suggestion. Passez ce délai, la requête est abandonnée.	Non	30000 (30s)
suggestMinLength	Le nombre minimum de caractères que la zone de saisie doit contenir pour que des suggestions soient proposées.	Non	2
suggestDelay	Le délai minimum en millisecondes entre la frappe de l'utilisateur et l'affichage des suggestions (ce délai peut être dépassé par le temps de récupération des suggestions).	Non	0
suggestLimit	Nombre maximum de suggestions à afficher parmi celles données par le serveur. 0 pour toutes les suggestions.	Non	0
suggestAJAX	« true » pour utiliser une requête AJAX, « false » pour utiliser le « Cross domain script tag hack » qui permet de contourner les limitations de sécurités des connexions AJAX[11].	Non	true

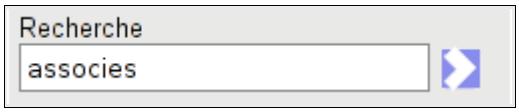
Voici un exemple de son utilisation :

```
<awml:widget type="keywords" style="display: none"></awml:widget>
```

11 Le « Cross domain script tag hack » permet de contacter un serveur autre que celui ayant servi la page HTML, ce que ne permet pas les connexions AJAX. Toutefois, certains réglages de sécurité des navigateurs bloquent ce mode.

Voici un exemple insérant une image à la place du bouton défini par défaut :

```
Recherche
<awml:widget type="keywords" style="display: none">
   <awml:label name="button">
      <html:img src="fleche.gif"></html:img>
   </awml:label>
</awml:widget>
```



3.1.2. Widget « pager »

Le widget « pager » affiche la zone qui permet de changer de page de résultats.

Ce widget supporte les **classes CSS** suivantes :

- *AFSPagerWidget* : le widget ;
- *AFSPagerWidget_PreviousPage* : le lien d'accès à la page précédente ;
- *AFSPagerWidget_NextPage* : le lien d'accès à la page suivante ;
- *AFSPagerWidget_PagesBefore* : l'indicateur visible quand le premier numéro de page affiché n'est pas le numéro de la première page de résultat ;
- *AFSPagerWidget_PagesAfter* : l'indicateur visible quand le dernier numéro de page affiché n'est pas le numéro de la dernière page de résultat ;
- *AFSPagerWidget_Page* : le numéro d'une page.

La sous-classe CSS « -Current » est appliquée à la classe « AFSPagerWidget_Page » quand l'élément affiché correspond au numéro de la page actuelle (cf. la section 2.6.2 sur les sous-classes CSS).

Exemple :

```
.AFSPagerWidget {
   margin: 4pt 0;
   text-align: center;
   display: none;
   margin-bottom: 20px;
}
.AFSPagerWidget-Set {
   display: block;
}
.AFSPagerWidget_PreviousPage, .AFSPagerWidget_NextPage,
.AFSPagerWidget_PagesBefore, .AFSPagerWidget_PagesAfter,
.AFSPagerWidget_CurrentPage, .AFSPagerWidget_Page {
   display: inline;
   margin: 0 3pt;
}
.AFSPagerWidget_Page-Current a {
   color: #FF0000;
   font-weight: bold;
}
.AFSPagerWidget_PreviousPage, .AFSPagerWidget_NextPage {
   margin-right: 20pt;
}
```

Le widget expose également les **libellés** suivants :

Nom	*Description*	*Requis*	*Par défaut*
previousPage	Le lien d'accès à la page précédente	Non	<< Previous
nextPage	Le lien d'accès à la page suivante	Non	Next >>
pagesBefore	L'indicateur de pages avant le premier numéro affiché	Non	...
pagesAfter	L'indicateur de pages après le dernier numéro affiché	Non	...

Le widget ne propose aucune **option**.

Voici un exemple :

```
<awml:widget type="pager" style="display: none">
   <awml:label name="previousPage">&lt;&lt; Pr&eacute;c&eacute;dente</awml:label>
   <awml:label name="nextPage">Suivante &gt;&gt;</awml:label>
</awml:widget>
```

<< Précédente 1 2 3 4 5 6 7 8 9 10 ... Suivante >>

3.1.3. Widget « facets »

Le widget « facets » affiche les facettes du résultat de la recherche. Ces facettes permettent d'organiser les résultats de la recherche. Par exemple, les différentes pages de résultats peuvent être organisées selon une taxonomie. Le moteur de recherche d'AFS peut ensuite être appelé avec un filtre particulier afin de restreindre la recherche à une catégorie de la taxonomie.

3.1.3.1. Principe

Pour que le widget affiche les différents filtres, il faut préciser le nom du filtre utilisé dans les résultats du moteur. Cela se fait en ajoutant la balise « awml:facet » dont l'attribut « name » est le nom de la facette. Lorsque le nom du paramètre à utiliser dans l'appel au moteur ne correspond pas au nom de la facette suffixé de « _FILTER », il est également nécessaire de le préciser à l'aide de l'option « param ».

Il peut y avoir plusieurs balises « awml:facet » qui définissent chaque facette à afficher dans le widget. Cependant, lorsqu'il y a plusieurs balises « awml:facet » qui possèdent la même valeur pour l'attribut « name », seule la dernière balise est prise en compte.

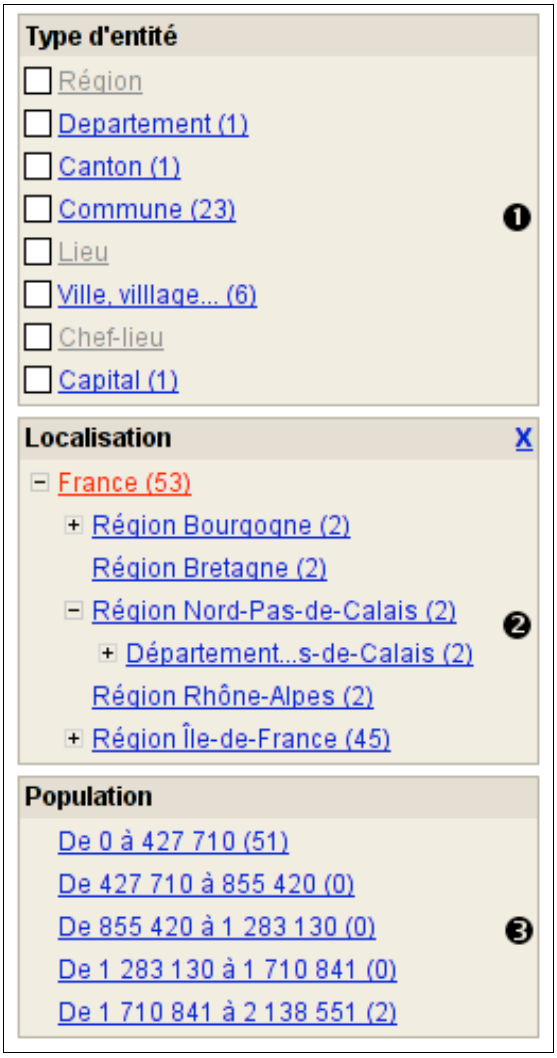
Dans le cas d'un service de recherche proposant des clusters (voir la section 3.3.3.1), il est possible de n'afficher les facettes regroupées dans ce widget que pour certains d'entre-eux. Il faut alors préciser l'option « clusters ».

Ce widget supporte les trois types de facettes qui peuvent apparaître dans le flux XML de réponse :

❶ Les filtres « flat » : ils comportent un ensemble de catégories.

❷ Les filtres « tree » : ils contiennent des catégories qui peuvent elles-mêmes contenir des sous-catégories.

❸ Les filtres « interval » : ils contiennent des valeurs correspondant à des intervalles.



Le type de facette doit être précisé sur la balise « awml:facet » par l'ajout de l'attribut « type ». Les valeurs possibles sont : « flat », « tree » ou « interval ». Le type de facette doit correspondre au type de facette définit lors de la configuration du moteur.

Les options et libellés décrits plus bas peuvent être placés dans la balise « awml:widget » ou dans une des sous-balises « awml:facet ». S'ils sont définis dans la balise « awml:widget », ils ont une portée globale : toutes les facettes les utilisent. S'ils sont définis dans la balise « awml:facet », ils ne s'appliquent qu'à cette facette. Lorsqu'une option ou un libellé est défini aux deux niveaux, c'est celui de la balise « awml:facet » qui est utilisé. Il est alors possible de définir un comportement général et de préciser pour certaines facettes des comportements spécifiques.

3.1.3.2. Les styles visuels (CSS)

Les **classes CSS** suivantes peuvent être utilisées :

- *AFSFacetsWidget* : le widget ;
- *AFSFacetWidget* : une facette ;
- *AFSFacetWidget_Header* : l'entête de la facette ;
- *AFSFacetWidget_Label* : le libellé de la facette dans l'entête ;
- *AFSFacetWidget_Remove* : le bouton de retrait de la facette ;
- *AFSFacetWidget_Values* : la liste des valeurs de la facette ;
- *AFSFacetWidget_Value* : une valeur de la facette ;
- *AFSFacetWidget_ValueLabel* : le libellé de la valeur d'une facette.

Les **sous-classes CSS** suivantes sont également disponibles (voir la section 2.6.2 concernant les sous-classes) :

- *-Flat* : s'applique sur la facette (*AFSFacetWidget*) pour préciser qu'elle est de type « flat » ;
- *-Tree* : s'applique sur la facette (*AFSFacetWidget*) pour préciser qu'elle est de type « tree » ;
- *-Interval* : s'applique sur la facette (*AFSFacetWidget*) pour préciser qu'elle est de type « interval » ;
- *-Multiple* : s'applique sur la liste des valeurs (*AFSFacetWidget_Values*) pour préciser que la facette est à sélection multiple (voir les options ci-dessous) ;
- *-Single* : s'applique sur la liste des valeurs (*AFSFacetWidget_Values*) pour préciser que la facette est à sélection simple (voir les options ci-dessous) ;
- *-Disabled* : s'applique sur une valeur (*AFSFacetWidget_Value*) et son libellé (*AFSFacetWidget_ValueLabel*) pour indiquer qu'elle est désactivée. C'est par exemple le cas lorsque des balises « awml:static » sont déclarées mais que certaines n'ont pas d'entrée correspondante dans le flux de réponse ;
- *-Selected* : s'applique sur une valeur (*AFSFacetWidget_Value*) et son libellé (*AFSFacetWidget_ValueLabel*) pour indiquer qu'elle est sélectionnée.
- *-Folded* : voir la section 2.6.3.

Exemple :

```
.AFSFacetsWidget {
   width: 246px;
}
.AFSFacetWidget {
   border: 1px solid #ACA693;
   background: #EFE9DA;
   margin-bottom: 4pt;
   display: none;
}
.AFSFacetWidget-Set {
   display: block;
}
.AFSFacetWidget_Remove {
   text-align: right;
}
```

```
.AFSFacetWidget_Label,.AFSFacetWidget_Remove {
   font-weight: bold;
   padding: 2pt;
   cursor: pointer;
}
.AFSFacetWidget_Header {
   background: #DED8C8;
   width: 100%;
}
.AFSFacetWidget-Folded .AFSFacetWidget_Values {
   display: none;
}
.AFSFacetWidget-Flat select {
   width: 220px;
   font-size: 10pt;
   border: 1px solid #ACA693;
   background-color: #F6F4EC;
}
.AFSFacetWidget_Values {
   padding: 1pt;
}
.AFSFacetWidget_ValueLabel {
   color: #0000cc;
   text-decoration: underline;
   cursor: pointer;
   margin: 0;
   padding: 0;
}
.AFSFacetWidget_Values-Multiple .AFSFacetWidget_Value {
   background-image: url(UnChecked.gif);
   background-repeat: no-repeat;
   background-position: 0 center;
}
.AFSFacetWidget_Values-Multiple .AFSFacetWidget_Value-Selected {
   background-image: url(Checked.gif);
}
.AFSFacetWidget_ValueLabel-Selected {
   color: #ff0000;
}
.AFSFacetWidget_ValueLabel-Disabled {
   color: gray;
}
```

3.1.3.3. Les libellés et options

Ce widget expose les **libellés** suivants :

Nom	***Description***	***Requis***	***Par défaut***
<aucun>[12]	Libellé de la facette.	Non	Le nom de la facette
remove	Libellé permettant de retirer le filtrage sur cette facette.	Non	<html:button>remove </html:button>

12 Le libellé n'a pas de nom (voir la section 2.5.3).

Pour les facettes de **type « flat »**, les **libellés** suivants sont également disponibles :

Nom	***Description***	***Requis***	***Par défaut***
more	Libellé de la liste déroulante	Non	More choice...
emptyLabel	Libellé venant remplacer celui des valeurs sans libellé ou dont leur libellé est « afs:no_cat ».	Non	Aucune modification

Pour les facettes de **type « tree »**, le **libellé** suivant est également disponible :

Nom	***Description***	***Requis***	***Par défaut***
emptyLabel	Libellé venant remplacer celui des valeurs sans libellé ou dont leur libellé est « afs:no_cat ».	Non	Aucune modification

Pour les facettes de **type « interval »**, le **libellé** suivant est également disponible :

Nom	***Description***	***Requis***	***Par défaut***
item	Libellé des valeurs. Ce libellé ne doit pas inclure de HTML (balise « html: »). Il est paramétré : {0} correspond à la valeur de début de l'intervalle ; {1} à la valeur de fin.	Non	From {0} to {1}

Les **options** suivantes sont disponibles pour toutes les facettes :

Nom	***Description***	***Requis***	***Par défaut***
param	Nom du paramètre de la facette.	Non	(voir « suffix »)
suffix	Si aucune option « param » n'est précisée, le nom du paramètre est le nom du filtre suffixé de cette option.	Non	_FILTER
maxLength	Longueur maximale (en caractères) des libellés (0 pour aucune limite).	Non	0
clusters	Nom des clusters (voir section 3.3.3.1) pour lesquels la facette doit apparaître. La facette ne sera alors visible que lorsque le cluster actif est listé dans cette option. C'est une liste de noms de cluster séparés par des virgules (« , »).	Non	
removePosition	Position du bouton de retrait du filtre (lorsqu'une valeur est sélectionnée pour la facette). Cette option peut prendre pour valeur « header » (faire apparaître le bouton à côté du libellé) ou « bottom » (le faire apparaître sous la liste).	Non	bottom
showCount	« true » pour afficher les décomptes	Non	true
agent	Le nom d'un agent auquel la facette est rattachée (voir ci-dessous).	Non	Tous les agents
unique	Le nom d'un agent devant répondre quand cette facette est active (voir ci-dessous).	Non	Ne pas limiter

Les facettes utilisent un paramètre lors de l'appel au moteur de recherche pour lui indiquer de filtrer les résultats suivant une valeur. Si l'option « param » n'est pas précisée, le paramètre

est le nom de la facette suivi de l'option « suffix ». Par défaut l'option « suffix » est à « _FILTER ».

Quand deux facettes dans le flux de réponse ont le même nom mais sont associées à deux agents différent, l'option « agent »permet de sélectionner la facette d'un agent en particulier. Sans cette option, la facette affichée est la première rencontrée.

Grâce à l'option « unique », il est possible de limiter les réponses produites à un agent quand une facette est sélectionnée. Cette option prend un nom d'agent et lorsque la facette est activé (c'est-à-dire les résultats filtrés par une valeur de la facette), le paramètre « UNIQUE » est ajouté à l'appel du moteur avec comme valeur celle de l'option. Ainsi si l'option est à « user1 », seules les réponses de l'agent « user1 » seront affichées.

Pour les facettes de **type « flat »**, les **options** suivantes sont également disponibles :

Nom	***Description***	***Requis***	***Par défaut***
maxItemCount	Nombre maximum d'entrées visibles, les autres sont placées dans une liste déroulante (0 empêche l'affichage de la liste déroulante).	Non	10
minListItemCount	Nombre minimum d'entrées dans la liste déroulante.	Non	2
multipleSelection	« true » pour permettre la sélection de plusieurs valeurs ; « false » sinon.	Non	false
hideEmptyLabel	« true » pour masquer valeur n'ayant pas de libellé ou dont le libellé est « afs:no_cat » ; « false » sinon	Non	false

Les facettes de type « flat » affichent les valeurs en liste. Si la facette a de nombreuses valeurs, il peut être intéressant de ne pas toutes les afficher. L'option « maxItemCount » permet de limiter la longueur de cette liste ; les autres valeurs sont alors placées dans une liste déroulante en dessous de la liste principale. Une valeur de 0 pour cette option empêche l'affichage de la liste déroulante.

L'option « minListItemCount » vient compléter cette option pour éviter de produire une liste déroulante trop courte (par exemple avec un seul élément). Si le nombre d'éléments dans la liste déroulante est inférieur à la valeur de l'option, d'autres valeurs sont prises de la liste principale pour compléter la liste déroulante. Pour n'avoir qu'une liste déroulante, il faut mettre cette option à une valeur très grande (100000 par exemple).

L'option « multipleSelection » permet la sélection multiple sur une facette de type « flat ». Suivant la configuration du moteur, la sélection multiple se comporte comme un « ou » ou un « et ». Il est intéressant de coupler cette option avec la déclaration de valeurs statiques via des balises « awml:static » comme expliqué plus loin.

Pour les facettes de **type « tree »**, les **options** suivantes sont également disponibles :

Nom	***Description***	***Requis***	***Par défaut***
split	Chaîne de séparation des libellés. Seule la chaîne située après le dernier séparateur est conservée.	Non	
hideEmptyLabel	« true » pour masquer les valeurs n'ayant pas de libellé ou dont le libellé est « afs:no_cat » ; « false » sinon	Non	false
allSelectable	« true » pour que tous les noeuds de l'arborescence soient sélectionnables ; « false » pour que ce ne soit que les feuilles.	Non	true

Une facette de type « tree » est généralement créée à partir de valeurs telles que « <valeur1>/<value2>/<valeur3> ». Le libellé de la catégorie correspondant est alors égal à cette chaîne mais il est préférable d'utiliser seulement la dernière valeur comme libellé. L'option « split » permet cela : en précisant le caractère délimiteur (« / » ici), la facette ne conserve que le dernier libellé dans la valeur du champs : « <valeur3> ».

Pour les facettes de **type « interval »**, les **options** suivantes sont également disponibles :

Nom	***Description***	***Requis***	***Par défaut***
minParam	Nom du paramètre fixant la valeur minimum pour ce filtre dans l'URL de recherche	Non	(voir ci-dessous)
maxParam	Nom du paramètre fixant la valeur maximum pour ce filtre dans l'URL de recherche	Non	

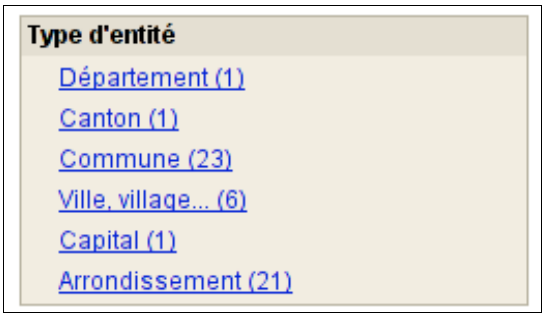
Les facettes de type « interval » utilisent deux paramètres là où les autres facettes n'en utilisent qu'un seul. Ces paramètres indiquent les valeurs minimum et maximum pour filtrer les résultats. Les options « minParam » et « maxParam » viennent alors remplacer l'option « param » et donnent la possibilité de préciser les paramètres à utiliser. En l'absence de ces options, leur valeur est le nom du filtre terminé par « _FILTER » et préfixé de « MIN_ » (pour « minParam ») ou « MAX_ » (pour « maxParam »).

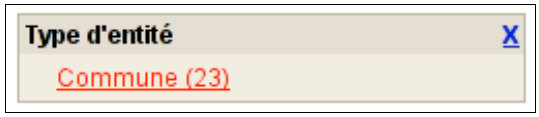
Voici un exemple d'utilisation du widget « facets » sans l'option « multipleSelection »:

```
<awml:widget type="facets" style="display: none">
   <awml:facet name="CG">
      <awml:option name="param">CGFILTER</awml:option>
      <awml:label>Compétences géographiques</awml:label>
   </awml:facet>
   <awml:facet name="REG">
      <awml:label>Région</awml:label>
      <awml:option name="param">REGFILTER</awml:option>
   </awml:facet>
</awml:widget>
```

3.1.3.4. Les balises « awml:static »

Les facettes de **type « flat »** proposent également les **balises « awml:static »**. Si aucune balise « awml:static » n'est présente, les facettes se basent sur la réponse du moteur de recherche pour produire l'affichage (image ci-dessous, à gauche). Quand une facette est active, il n'y a alors plus que des réponses pour la valeur correspondante au filtre placé, ce qui limite l'affichage de la facette à cette valeur (image ci-dessous, à droite).





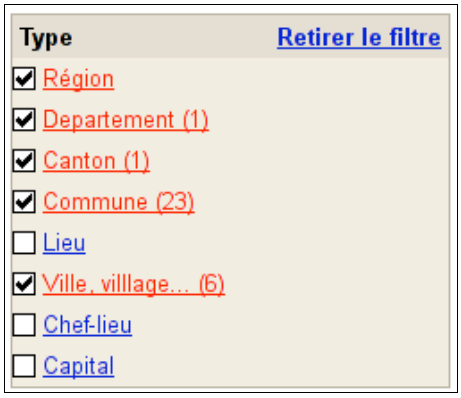
Lorsque des balises « awml:static » sont présentes, la facette ne lit plus la réponse du moteur pour créer son affichage mais utilise ces balises. Ainsi, même si un filtre est placé, toutes les valeurs déclarées pour cette facette sont visibles (image de droite) ; celles non présentes dans la réponse n'ont pas le nombre de réponses entre parenthèses et sont désactivées (sous-classe CSS « -Disabled »).

Une balise « awml:static » comporte un attribut « value » précisant la valeur du filtre lorsque l'élément est choisi et un contenu qui sert de libellé.

Voici un exemple d'utilisation du widget « facets » avec l'option « multipleSelection » activée et des libellés statiques définis :

```
<awml:widget type="facets" style="display:none">
  <awml:label name="remove">Retirer le filtre</awml:label>
  <awml:option name="removePosition">header</awml:option>

  <awml:facet name="Type" type="flat">
    <awml:label>Type</awml:label>
    <awml:static value="0">Région </awml:static>
    <awml:static value="1">Departement</awml:static>
    <awml:static value="2">Canton</awml:static>
    <awml:static value="3">Commune</awml:static>
    <awml:static value="4">Lieu</awml:static>
    <awml:static value="5">Ville, villlage...</awml:static>
    <awml:static value="6">Chef-lieu</awml:static>
    <awml:static value="7">Capital</awml:static>
    <awml:option name="multipleSelection">true</awml:option>
  </awml:facet>
</awml:widget>
```



3.1.4. Widget « autoFacets »

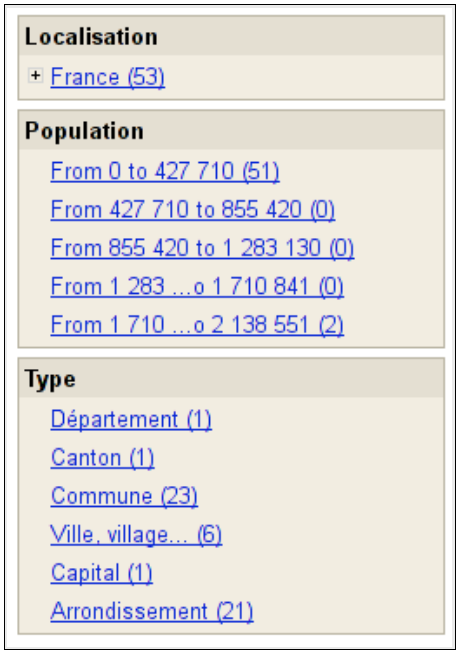
Le widget « autoFacets » est un cas particulier du widget « facets ». Ce widget affiche les filtres de manière automatique c'est-à-dire en fonction des facettes présentes dans la réponse du moteur. Ce widget peut être très utile lors des phases de développement pour avoir rapidement la liste des facettes disponibles. Il est possible d'utiliser les options et libellés supportés par le widget « facets » (voir la section 3.1.3 détaillant le widget « facets ») pour modifier le comportement des facettes.

Cependant, il ne propose pas de balise « awml:facet ». Il n'est donc pas possible de le configurer différemment pour chaque facette.

Les **classes CSS**, les **libellés** et les **options** associés à ce widget sont identiques à celles du widget « facets » (voir la section 3.1.3 détaillant le widget « facets »).

Voici un exemple d'utilisation :

```
<awml:widget type="autoFacets" style="display: none">
   <awml:label name="remove">X</awml:label>
   <awml:option name="maxItemCount">0</awml:option>
   <awml:option name="maxLength">26</awml:option>
   <awml:option name="removePosition">header</awml:option>
</awml:widget>
```



3.1.5. Widget « context »

Le widget « context » offre une zone de visualisation des paramètres de recherche. Il liste les mots-clés recherchés, les facette filtrées et, s'il y a lieu, le groupe ou le cluster actif.

Ce widget supporte les **classes CSS** suivantes :

- *AFSContextWidget* : le widget ;
- *AFSContextWidget*_Label : le libellé ;
- *AFSContextWidget_Item* : un élément affiché (la valeur d'un paramètre : mots clés, facette ou groupe/cluster) ;
- *AFSContextWidget_ItemLabelCell* : la cellule contenant le libellé d'un élément ;
- *AFSContextWidget_ItemLabel* : le libellé d'un élément ;
- *AFSContextWidget_ItemValueCell* : la cellule contenant la valeur d'un élément ;
- *AFSContextWidget_ItemValue* : la valeur d'un élément ;
- *AFSContextWidget_ItemRemoveCell* : la cellule contenant le bouton de retrait d'un élément ;
- *AFSContextWidget_ItemRemove* : le bouton de retrait d'un élément.

Exemple :

```
.AFSContextWidget {
   display: none;
   background: #EFE9DA;
   border: 1px solid #ACA693;
   margin: auto;
   width: 244px;
   margin-bottom: 4pt;
}
.AFSContextWidget-Set {
   display: block;
}
.AFSContextWidget_Label {
   background: #DED8C8;
   font-weight: bold;
   padding: 3pt;
}
.AFSContextWidget_Item {
   border-top: solid 1px #DED8C8;
   padding: 1pt;
   width: 100%;
}
.AFSContextWidget_ItemValueCell {
   text-align: center;
   font-style: italic;
}
.AFSContextWidget_ItemRemoveCell {
   text-align: right;
}
.AFSContextWidget_ItemRemove {
   text-decoration: underline;
   color: #0000CC;
   font-weight: bold;
   cursor: pointer;
}
```

Il expose les **libellés** suivants :

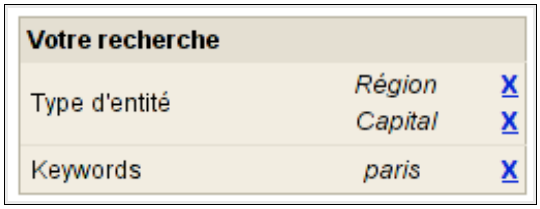
Nom	Description	Requis	Par défaut
<aucun>[13]	Libellé d'entête.	Non	
keywords	Libellé pour les mots clés de la recherche.	Non	Keywords
cluster	Libellé pour le cluster actif.	Non	Cluster
group	Libellé pour le groupe actif	Non	Group
order	Libellé pour le tri actif	Non	Order
remove	Libellé pour le bouton de retrait d'un des paramètres de la recherche	Non	<html:button>remove </html:button>

L'**option** suivante est disponible :

Nom	Description	Requis	Par défaut
resetFacetsOnClusterClear	« true » pour retirer les filtres sélectionnés lorsque le cluster en cours est désactivé ; « false » sinon.	Non	true

Voici un exemple d'utilisation du widget :

```
<awml:widget type="context" style="display:none">
   <awml:label>Votre recherche</awml:label>
   <awml:label name="keywords">Keywords</awml:label>
   <awml:label name="cluster">Groupe</awml:label>
   <awml:label name="remove">X</awml:label>
   <awml:option name="resetFacetsOnClusterClear">false</awml:option>
</awml:widget>
```



3.1.6. Widget « clusters »

Le widget « clusters » affiche la liste des clusters déclarés pour le service (voir la section 3.3.3.1). Ce widget permet de changer rapidement de cluster. Son comportement est proche du widget « facets » dans la mesure où il permet d'organiser les résultats de la recherche et de filtrer les résultats en fonction des flux du moteur de recherche.

Les classes et sous-classes CSS sont similaires à celles du widget « facets ».

Les **classes CSS** suivantes sont disponibles :

- *AFSClustersWidget* : le widget ;
- *AFSClustersWidget_Header* : l'entête ;
- *AFSClustersWidget_Label* : le libellé dans l'entête ;
- *AFSClustersWidget_Remove* : le bouton de retrait du cluster actif ;
- *AFSClustersWidget_Values* : la liste des clusters ;
- *AFSClustersWidget_Value* : un cluster ;

13 Le libellé n'a pas de nom (voir la section 2.5.3).

– *AFSClustersWidget_ValueLabel* : le libellé du cluster.

Les **sous-classes CSS** suivantes sont également disponibles (voir la section 2.6.2) :

– *-Disabled* : s'applique sur un cluster (*AFSClustersWidget_Value*) et son libellé (*AFSCLustersWidget_ValueLabel*) pour indiquer qu'il est désactivé (exemple : le cluster n'a pas de réponse) ;

– *-Selected* : s'applique sur un cluster (*AFSClustersWidget_Value*) et son libellé (*AFSClustersWidget_ValueLabel*) pour indiquer qu'il est sélectionné.

Exemple :

```
.AFSClustersWidget {
   width: 246px;
   border: 1px solid #ACA693;
   background: #EFE9DA;
   margin-bottom: 4pt;
   display: none;
}
.AFSClustersWidget_Header {
   background: #DED8C8;
   width: 100%;
}
.AFSClustersWidget_Label,.AFSClustersWidget_Remove {
   font-weight: bold;
   padding: 2pt;
   cursor: pointer;
}
.AFSClustersWidget-Folded .AFSClustersWidget_Values {
   display: none;
}
.AFSClustersWidget _Remove {
   text-align: right;
}
.AFSClustersWidget _Values {
   padding: 1pt;
}
.AFSClustersWidget _ValueLabel {
   color: #0000cc;
   text-decoration: underline;
   cursor: pointer;
   margin: 0;
   padding: 0;
}
.AFSClustersWidget _ValueLabel-Selected {
   color: #ff0000;
}
.AFSClustersWidget _ValueLabel-Disabled {
   color: gray;
}
```

Ce widget expose les **libellés** suivants :

Nom	***Description***	***Requis***	***Par défaut***
<aucun>[14]	Libellé d'entête.	Non	
remove	Libellé permettant de retirer le cluster actif de l'appel au moteur.	Non	<html:button>remove </html:button>

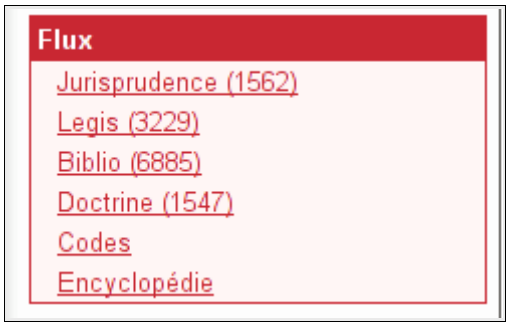
14 Le libellé n'a pas de nom (voir la section 2.5.3).

Les **options** suivantes sont disponibles :

Nom	*Description*	*Requis*	*Par défaut*
maxLength	Longueur maximale (en caractères) des libellés de cluster (0 pour aucune limite)	Non	0
showCount	« true » pour afficher le nombre de réponses pour chaque cluster entre parenthèses après le libellé du cluster. Pour certains services de recherche, il n'est pas possible d'extraire le nombre de réponses, dans ce cas, aucun nombre n'est affiché. « false » sinon.	Non	false
removePosition	Position du bouton de retrait du cluster en cours. Cette option peut prendre pour valeur « header » pour faire apparaître le bouton à côté du libellé du filtre ou « bottom » pour le faire apparaître sous la liste des valeurs.	Non	bottom

Voici un exemple :

```
<awml:widget type="clusters" style="display: none">
   <awml:label name="remove">X</awml:label>
   <awml:label>Flux</awml:label>
   <awml:option name="removePosition">header</awml:option>
   <awml:option name="showCount">true</awml:option>
</awml:widget>
```



3.1.7. Widget « sort »

Le widget « sort » permet de trier les résultats selon un champ ou plusieurs champs de l'index avec un ordre ascendant ou descendant (voir la section 2.4.1 pour plus de précisions sur les tris).

Ce widget supporte les **classes CSS** suivantes :

- *AFSSortWidget* : le widget ;
- *AFSSortWidget_Header* : l'entête de la liste des tris ;
- *AFSSortWidget_Label* : le libellé de la liste des tris ;
- *AFSSortWidget_Remove* : le bouton de retrait du tri actif;
- *AFSSortWidget_Content* : le conteneur de la liste des tris ;
- *AFSSortWidget_Items* : la liste des tris ;
- *AFSSortWidget_Item* : un élément de la liste des tris ;
- *AFSSortWidget_ItemLabel* : le libellé d'un élément de la liste des tris.

La **sous-classe CSS** suivante est également disponible (voir la section 2.6.2 concernant les sous-classes) :

- *-Selected* : s'applique sur un élément de la liste des tris (*AFSSortWidget_Item* et *AFSSortWidget_ItemLabel*) pour indiquer qu'il est sélectionné.

Exemple :

```
.AFSSortWidget {
   border: 1px solid #ACA693;
   background: #EFE9DA;
   width: 244px;
   margin: auto;
   margin-bottom: 4pt;
   display: none;
}
.AFSSortWidget-Set {
   display: block;
}
.AFSSortWidget-Folded .AFSSortWidget_Content {
   display: none;
}
.AFSSortWidget_Label {
   background: #DED8C8;
   font-weight: bold;
   padding: 3pt;
}
.AFSSortWidget_Header {
   background: #DED8C8;
   width: 100%;
}
.AFSSortWidget_Label,.AFSSortWidget_Remove {
   font-weight: bold;
   padding: 2pt;
   cursor: pointer;
}
.AFSSortWidget_Remove {
   text-align: right;
}
.AFSSortWidget_ItemLabel {
   color: #0000cc;
   text-decoration: underline;
   cursor: pointer;
   margin: 0;
   padding: 0;
}
.AFSSortWidget_ItemLabel-Selected {
   color: #FF0000;
}
```

Il expose les **libellés** suivants :

Nom	Description	Requis	Par défaut
<aucun>[15]	Libellé à placer juste avant la liste des tris proposés. Il permet d'introduire la liste.	Non	
remove	Libellé permettant de retirer le tri actif	Non	<html:button> remove <html:button>

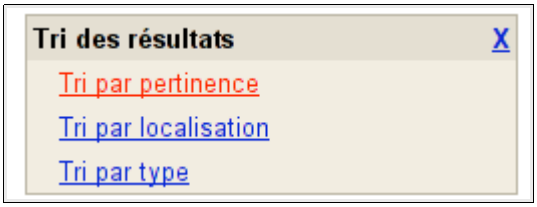
Les **options** suivantes sont disponibles :

Nom	Description	Requis	Par défaut
removePosition	Position du bouton de retrait du tri (lorsqu'une valeur est sélectionnée pour le tri). Cette option peut prendre pour valeur « header » pour faire apparaître le bouton à côté du libellé du tri ou « bottom » pour le faire apparaître sous la liste des tris proposés.	Non	bottom

Les tris doivent être déclarés pour pouvoir être activés au niveau du moteur de recherche (voir la section 2.4.1 pour la déclaration des tris).

Voici un exemple d'utilisation du widget « sort » utilisant la déclaration des tris spécifiée dans l'exemple de la section 2.4.1:

```
<awml:widget type="sort" style="display:none">
   <awml:label>Tri des r&eacute;sultats</awml:label>
   <awml:label name="remove">X</awml:label>
   <awml:option name="removePosition">header</awml:option>
</awml:widget>
```



3.2. Les widgets d'état

3.2.1. Widget « info »

Le widget « info » permet d'afficher, lorsque des résultats d'une recherche sont chargés, un libellé pouvant contenir le nombre de réponses, le temps nécessaire à la recherche et les mots clés recherchés.

Ce widget supporte la **classe CSS** suivante :

– *AFSInfoWidget* : le widget.

Exemple :

```
.AFSInfoWidget-Set {
   display: block;
}
```

15 Le libellé n'a pas de nom (voir la section 2.5.3).

```
.AFSInfoWidget {
   margin: 7pt 0pt 4pt 30pt;
   background: #EFE9DA;
   padding: 4pt 10px;
   margin-right: 3pt;
   display: none;
}
```

Le **libellé** suivant est mis à disposition :

Nom	*Description*	*Requis*	*Par défaut*
<aucun>[16]	Libellé à afficher	Non	

Ce libellé est un libellé paramétré (voir la section 2.5.4) qui supporte les références suivantes :

{0} Les mots-clés de la recherche ;

{1} Le nombre de résultats obtenus ;

{2} Le temps de réponse du moteur de recherche.

Le widget « info » propose **l'option** :

Nom	*Description*	*Requis*	*Par défaut*
agent	Le paramètre {1} (nombre de résultats) correspond alors au nombre de réponses de l'agent spécifié.	Non	

Voici un exemple de son utilisation :

```
<awml:widget type="info" style="display: none">
   <awml:label>
      Recherche de <html:em>{0}</html:em>,
      {1,choice,0#aucune réponse|1#1 réponse|2#{1,number} réponses}
      en {2} secondes
   </awml:label>
</awml:widget>
```

Recherche de *antidot*, 167 réponses en 0.408 secondes

3.3. Les widgets de résultats

3.3.1. Le formatage des résultats

Quel que soit le widget d'affichage des résultats (« results », « clusteredResults » ou « groupedResults »), les résultats sont affichés par l'intermédiaire d'un formateur. Celui-ci décide des informations à afficher et de leur agencement. Il existe 3 formateurs, chacun ayant ses avantages :

- le formateur « default » : utilise la présentation habituelle des résultats de recherche (titre, description, URL) ;
- le formateur « custom » : offre une grande liberté dans l'affichage en donnant accès directement à la réponse XML du moteur de recherche ;
- le formateur « none » : n'affiche pas de résultat.

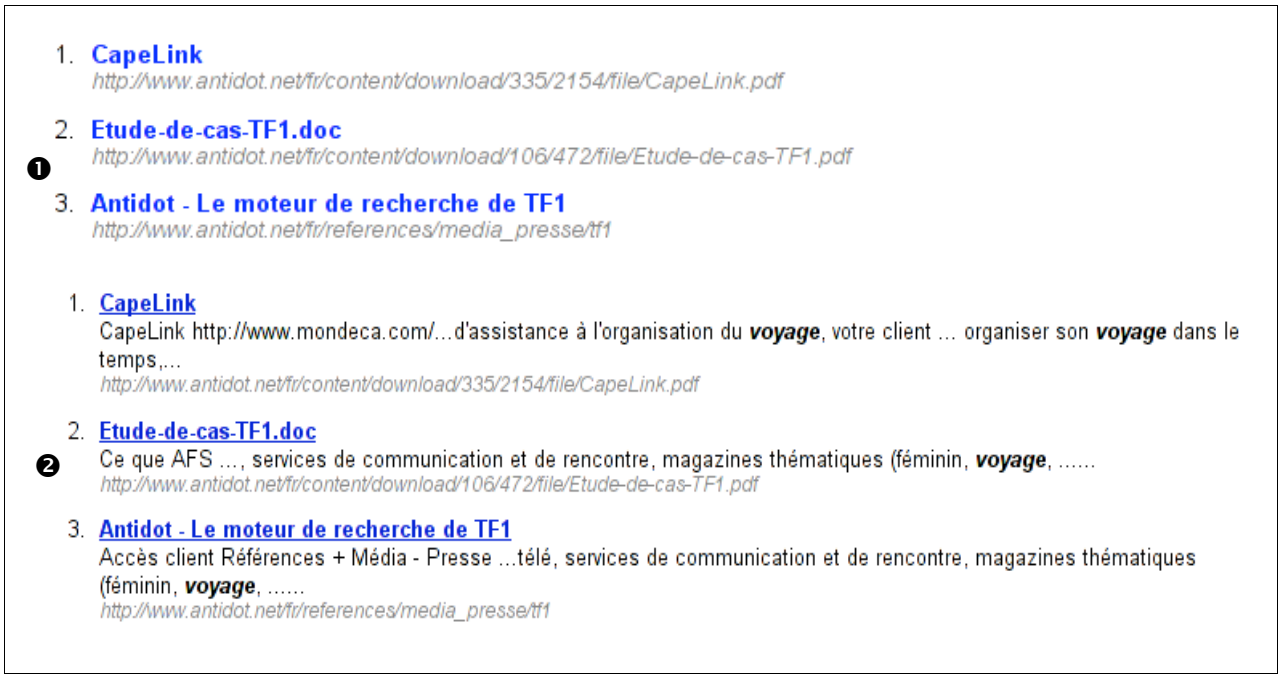
16 Le libellé n'a pas de nom (voir la section 2.5.3).

Le choix d'un formateur de résultat peut se faire au niveau de l'agent qui a produit la réponse. Cela donne la possibilité de présenter différemment les réponses issues de flux différents. Le choix d'un formateur pour un agent est réalisé grâce à la balise « awml:agent ».

Voici un exemple de configuration du widget :

```
<awml:widget type="results" style="display: none">
   <awml:agent names="user1" format="custom">                    ❶
      <html:div class="MonTitre">{XML_Agent/ROOT/PAGE/@title}</html:div>

      <html:div class="MonUrl">{XML_Agent/ROOT/PAGE/URI/text()}</html:div>
   </awml:agent>
   <awml:agent names="antibot,engine1,engine2" format="default">  ❷
   </awml:agent>
</awml:widget>
```



3.3.1.1. Choisir un formateur (balise « awml:agent »)

Une balise « awml:agent » peut être mise dans n'importe quel widget de résultats afin de préciser le formateur à utiliser pour les réponses. Il peut y avoir plusieurs balises pour un même widget et si ces balises déclarent un formateur plusieurs fois pour un même agent, c'est la dernière qui est conservée.

La balise « awml:agent » propose les **attributs** suivants :

Nom	*Description*	*Requis*	*Par défaut*
names	Les agents auxquels s'appliquent ce formateur. C'est une liste de noms d'agents séparés par des virgules (« , »)	Non	Tous les agents sauf hint, promotion, parametric, rte.
format	Le type de formateur	Non	default

Lorsque l'attribut « names » n'est pas précisé, le formateur s'applique à tous les agents sauf aux agents « hint », « parametric », « promotion » et « rte ». Ces agents ont en effet des widgets particuliers pour mettre en forme leurs résultats. Il est toutefois possible d'activer un

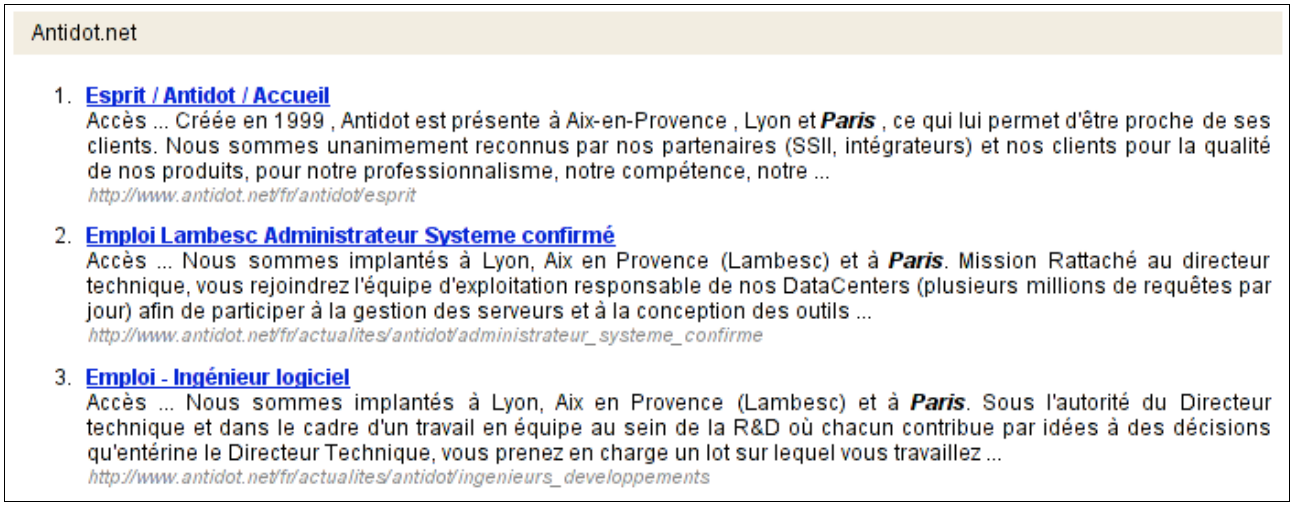
formateur pour ces agents en ajoutant une balise « awml:agent » dont l'attribut « names » contient le nom de ces agents.

Le contenu de la balise dépend ensuite du formateur.

Si aucun formateur n'est précisé dans une balise de résultat, tous les agents sauf « hint », « parametric », « promotion » et « rte » sont traités avec le formateur « default ».

3.3.1.2. Le formateur « default »

Cet affichage correspond à l'affichage standardisé des moteurs de recherche Internet. Il permet d'afficher le titre et/ou la description et/ou l'URL du résultat. Le format KWIC (*KeyWord In Context*) est utilisé afin de mettre en évidence les mots clés recherchés qui se trouvent dans le titre ou la description de chaque résultat.



Ce formateur définit les **classes CSS** suivantes :

- *AFSResultsWidget_Title* : le titre d'un résultat ;
- *AFSResultsWidget_Description* : la description d'un résultat ;
- *AFSResultsWidget_URL* : l'URL d'un résultat ;
- *AFSResultsWidget_WordMatch* : le KWIC (les mots recherchés présents dans le titre ou la description).

Exemple :

```
.AFSResultsWidget_Title {
   font-weight: bold;
}
.AFSResultsWidget_Description {
   padding-right: 10px;
}
.AFSResultsWidget_URL a {
   font-style: italic;
   font-size: 9pt;
   text-decoration: none;
   color: #777777;
}
.AFSResultsWidget_WordMatch {
   font-style: italic;
   font-weight: bold;
}
```

Il expose le **libellé** suivant :

Nom	*Description*	*Requis*	*Par défaut*
url	Libellé de l'URL à afficher. La séquence « {0} » est remplacée par la valeur de l'URL dans le flux de réponse.	Non	L'option « urlPattern »

Il supporte également les **options** suivantes :

Nom	*Description*	*Obligatoire*	*Par défaut*
showTitle	« true » pour afficher le titre de chaque résultat ; « false » sinon	Non	true
showImage	« true » pour afficher l'image de chaque résultat ; « false » sinon.	Non	true
showDescription	« true » pour afficher la description de chaque résultat ; « false » sinon.	Non	true
showURL	« true » pour afficher l'URL de chaque résultat ; « false » sinon.	Non	true
urlPattern	La cible de l'URL pour les résultats. La séquence « {0} » est remplacée par la valeur de l'URL dans le flux de réponse.	Non	{0}
urlMaxLength	Longueur maximale (en caractères) de l'URL lors de son affichage. Si elle est trop longue, l'URL est coupée en son milieu (et « ... » est ajouté).	Non	80
target	Précise la cible de l'URL lorsque l'internaute clique dessus. Les valeurs possibles sont les mêmes que l'attribut « target » de la balise HTML « a ».	Non	_current

Si une option n'est pas déclarée au niveau de la balise « awml:agent », elle est recherchée au niveau du widget.

3.3.1.3. Le formateur « custom »

L'affichage personnalisé exploite la souplesse du XHTML tout en proposant la sélection d'information dans le flux de résultat par du XPath.

Quand une balise« awml:agent » contient un attribut « format » à la valeur « custom », le widget exploite le contenu de la balise pour générer l'affichage d'une réponse. Le contenu de la balise correspond à un langage mélangeant du XHTML pour la mise en forme, du XPath pour injecter des données depuis la réponse XML et des balises particulières AWML.

Le XHTML est recopié tel quel pour chaque réponse. La seule obligation est celle de préfixer les balises XHTML de « html: » (voir la section 2.8.2).

Les XPath doivent être entourés d'accolades et se trouver dans les attributs ou le contenu des balises XHTML. Lors de la génération d'une réponse, les XPath sont évalués et le résultat est injecté en lieu et place des XPath (et de leurs accolades). Les XPath ont pour contexte le noeud XML du résultat. Un XPath erroné est simplement ignoré (remplacé par une chaîne vide).

Les XPath doivent obligatoirement retourner un noeud ou un ensemble de noeuds. Les XPath de type « count(//SearchAgentReply) » ou « ./position() » ne sont donc pas autorisés.

Enfin les balises AWML offrent diverses structures de contrôle ou permettent d'extraire des informations. Les sous-sections suivantes détaillent chaque balise.

Par exemple, voici un extrait des résultats du moteur :

```
...
   <XML_Agent>
      <ROOT title="Mon titre" image="http://www.mondomaine.com/images/monimage.jpg">
         <entry name="entree1">Texte de l'entrée 1</entry>
         <entry name="entree2">Texte de l'entrée 2/entry>
      </ROOT>
   </XML_Agent>
...
```

Il est alors possible de configurer le formateur « custom » ainsi :

```
<awml:agent names="user1" format="custom">
   {XML_Agent/ROOT/@title/text()}
   <awml:if test="XML_Agent/ROOT/@image">
      <html:img src="{XML_Agent/ROOT/@image/text()}" alt=""></html:img>
   </awml:if>
   <html:ul>
      <awml:for nodes="XML_Agent/ROOT/entry">
         <html:li>{@name} = {text()}</html:li>
      </awml:for>
   </html:ul>
</awml:agent>
```

Il produira alors le HTML suivant pour la réponse en question :

```
<img src="http://www.mondomaine.com/images/monimage.jpg"/>
<ul>
   <li>entree1 = Texte de l'entrée 1</li>
   <li>entree2 = Texte de l'entrée 2</li>
</ul>
```

Ce processus se répète ensuite pour chaque réponse.

Le formateur « custom » offre donc une grande souplesse dans la mise en forme d'une réponse. Il trouve particulièrement sa place pour les flux de type XML ou base de données.

3.3.1.3.1. Les XPath

Les XPath présents dans les attributs des balises XHTML et dans le texte sont remplacés par les valeurs qu'ils pointent lorsqu'ils sont entourés d'accolades (ie. « {Node1/Node2/Node3} »). Ces XPath doivent toujours pointer sur un noeud ou un ensemble de noeuds (la fonction « text() » est également admise car retournant un noeud « text »). Il n'est pas possible de retourner le résultat de fonctions telles que « count » ou « position() ».

A l'évaluation, ces XPath sont remplacés par leur valeur pointée, ce qui inclut les balises XML qui ont pu être rencontrées. Par exemple, en reprenant le XML précédent :

```
...
   <XML_Agent>
      <ROOT title="Mon titre" image="http://www.mondomaine.com/images/monimage.jpg">
         <entry name="entree1">Texte de l'entrée 1</entry>
         <entry name="entree2">Texte de l'entrée 2</entry>
      </ROOT>
   </XML_Agent>
...
```

Le XPath *{XML_Agent/ROOT}* injectera dans le HTML :

```
<ROOT title="Mon titre" image="http://www.mondomaine.com/images/monimage.jpg">
   <entry name="entree1">Texte de l'entrée 1</entry>
   <entry name="entree2">Texte de l'entrée 2</entry>
</ROOT>
```

La section CDATA n'apparaîtra pas (voir la section).

Le XPath *{XML_Agent/ROOT/entry}* :

```
<entry name="entree1">Texte de l'entrée 1</entry>
<entry name="entree2">Texte de l'entrée 2<entry>
```

Le XPath *{XML_Agent/ROOT/entry[position() = 1]}* :

```
<entry name="entree1">Texte de l'entrée 1</entry>
```

Le XPath *{XML_Agent/ROOT/entry[position() = 1]/text()}* :

```
Texte de l'entrée 1
```

Le XPath *{XML_Agent/ROOT/@image}* :

```
image="http://www.mondomaine.com/images/monimage.jpg"
```

Le XPath *{XML_Agent/ROOT/@image/text()}* :

```
http://www.mondomaine.com/images/monimage.jpg
```

Enfin le XPath *{XML_Agent/ROOT/text()}* ne donnera aucun résultat car la balise ROOT n'a pas de texte comme fils direct.

Les balises « awml:text » et les balises « awml:children » permettent des injections plus complexes.

3.3.1.3.2. Les balises « awml:text » et « awml:children »

La balise « awml:text » permet d'extraire le texte contenu dans un noeud de la réponse quel que soit sa profondeur dans l'arborescence XML. La balise doit avoir comme contenu le XPath. Par exemple :

```
<awml:text>XML_Agent/ROOT</awml:text>
```

produit :

```
Texte de l'entrée 1
Texte de l'entrée 2
```

La balise « awml:children » permet d'injecter tout le XML contenu dans un noeud (sans inclure les balises d'ouverture et fermeture du noeud). Par exemple :

```
<awml:children>XML_Agent/ROOT</awml:children>
```

produit :

```
<entry name="entree1">Texte de l'entrée 1</entry>
<entry name="entree2">Texte de l'entrée 2</entry>
```

3.3.1.3.3. Les sections CDATA

Les navigateurs supportés par les Widgets d'AFS traitent de façon particulièrement les sections CDATA. Si elles sont présentes dans la page HTML ou injectées dans celle-ci, les sections CDATA sont ignorées (non affichées). Par contre, la fonction XPath « text() » permet d'accéder aux sections CDATA du XML de réponses.

L'utilisation d'injections (XPath entouré par des accolades) revient à copier le XML de réponse dans la page HTML. Si le XPath sélectionne un ou plusieurs noeuds (ex. « /root ») contenant des sections CDATA, elles ne seront pas visibles. Par contre, si ce XPath sélectionne le texte de ce XPath (ex. « /root//text() »), alors les sections CDATA sont transformées en texte et affichées.

Par exemple, en modifiant l'exemple ci-dessus :

```
...
   <XML_Agent>
      <ROOT title="Mon titre" image="http://www.mondomaine.com/images/monimage.jpg">
         <entry name="entree1">Texte de l'entrée 1</entry>
         <entry name="entree2"><![CDATA[Texte de l'entrée 2]]>/entry>
      </ROOT>
   </XML_Agent>
...
```

Le XPath *{XML_Agent/ROOT/entry}, tout comme l'instruction :*

```
<awml:children>XML_Agent/ROOT</awml:children>
```

produisent :

```
<entry name="entree1">Texte de l'entrée 1</entry>
<entry name="entree2"><![CDATA[Texte de l'entrée 2]]>/entry>
```

mais les navigateurs ignorent la section CDATA et ne prennent en compte que :

```
<entry name="entree1">Texte de l'entrée 1</entry>
<entry name="entree2"><entry>
```

Par contre le XPath *{XML_Agent/ROOT//text()}* et l'instruction :

```
<awml:text>XML_Agent/ROOT</awml:text>
```

produiset :

```
Texte de l'entrée 1
Texte de l'entrée 2
```

car la section CDATA est transformée en texte.

Ce comportement s'applique à tous les navigateurs supportés par les Widgets d'AFS.

3.3.1.3.4. La balise « awml:for »

La balise « awml:for » permet d'itérer une mise en forme sur un ensemble de noeuds XML. Son attribut « nodes » doit contenir un XPath valide qui retourne 0 à plusieurs noeuds. Le corps de la balise « awml:for » est évalué pour chaque noeud et injecté en lieu et place de la balise « awml:for ». Les XPath présents dans ce corps sont relatifs au noeud en cours.

3.3.1.3.5. La balise « awml:if »

La balise « awml:if » permet de n'évaluer un bloc que si un noeud est présent. L'attribut « test » doit contenir un XPath pointant sur 0 ou plusieurs noeuds. Si au moins 1 noeud satisfait ce XPath, alors le corps de la balise est évalué et son résultat est injecté en lieu et place de la balise « awml:if ».

3.3.1.3.6. La balise « awml:skip »

La balise « awml:skip » permet d'ignorer une réponse. Si cette balise est rencontrée, le traitement est interrompu et la réponse en cours n'est pas affichée dans le widget de résultats.

3.3.1.3.7. Les balises « awml:choose », « awml:when », « awml:otherwise »

La balise « awml:choose » permet de créer des conditions plus complexes que la balise « awml:if ». Elle ne propose pas d'attribut mais utilise ses sous-balises « awml:when » et « awml:otherwise », les seules permises.

Une balise « awml:when » permet de poser une condition qui, si elle est vérifiée, mène à l'évaluation du contenu de la balise « awml:when ». L'attribut « test » doit contenir un XPath pointant sur 0 ou plusieurs noeuds. Si au moins 1 noeud satisfait ce XPath, le corps de la balise est évalué et remplace la balise « awml:choose ». Le reste de la balise « awml:choose » est alors ignoré. Si la condition n'est pas satisfaite, la balise « awml:choose » examine la balise suivante.

Après toutes les balises « awml:when », il est possible d'ajouter une balise « awml:otherwise » qui sera traitée si aucune balise « awml:when » n'a pu être satisfaite.

3.3.1.3.8. La balise « awml:using »

La balise « awml:using » permet de définir un noeud de base via son attribut « node ». Celui-ci doit contenir un XPath pointant sur un noeud existant. Le corps de la balise est évalué et son résultat vient remplacer celle-ci. Tous les XPath rencontrés dans le corps ont alors pour base le noeud pointé par l'attribut « node ».

Il devient alors possible de réduire les XPath pointant sur des données très loin dans l'arborescence XML de la réponse.

3.3.1.3.9. Le traitement du KWIC

Quand une partie de la réponse XML est injectée dans le HTML, son contenu est traité pour remplacer automatiquement les balises du KWIC.

Ainsi les balises « Ew » sont remplacées par une balise « span » ayant une classe CSS « AFSResultsWidget_WordMatch ». Les balises « Etc » sont remplacées par « ... ».

3.3.1.4. Le formateur « none »

Ce type n'affiche aucun résultat. Il permet de désactiver l'affichage des résultats pour certains agents.

3.3.2. L'affichage simple (widget « results »)

Le widget « results » permet d'afficher les résultats du moteur de recherche. Il ne prend pas en compte les clusters et les groupes (voir les widgets « clusteredResults » et « groupedResults ») mais offre toutes les options de mise en forme des formateurs. Le corps de la balise peut contenir des options, des libellés et des balises « awml:agent ».

Il offre également la possibilité d'afficher un message lorsqu'une erreur survient (le serveur ne répond pas, la connexion internet est perdue, une erreur inattendue s'est produite...). Ce message est spécifié grâce au libellé « error » et vient en lieu et place des résultats (voir la section 4.1).

Enfin, lorsqu'il n'y a pas de résultats, le libellé « empty » permet d'afficher un message personnalisé.

Le widget supporte les **classes CSS** suivantes :

- *AFSResultsWidget* : le widget ;
- *AFSResultsWidget_Results* : le conteneur de la liste des résultats ;
- *AFSResultsWidget_Label* : le libellé de la liste ;
- *AFSResultsWidget_Items* : la liste des résultats ;
- *AFSResultsWidget_Item* : un élément de la liste des résultats ;
- *AFSResultsWidget_Result* : un résultat ;
- *AFSResultsWidget_Empty* : le libellé affiché quand il n'y a pas de résultats ;
- *AFSResultsWidget_Error* : le libellé affiché quand il y a une erreur ;

Il supporte également la **sous-classe CSS** suivante :

- *-Empty* : s'applique à la liste des résultats (*AFSResultsWidget_Results*) lorsqu'il n'y a pas de résultats.

Exemple :

```
.AFSResultsWidget_Results {
   margin: 0;
   padding: 0;
   padding-left: 30pt;
   display: none;
   text-align: justify;
}
.AFSResultsWidget_Results-Set,.AFSResultsWidget_Results-Empty {
   display: block;
}

.AFSResultsWidget_Label {
   background: #EFE9DA;
   padding: 4pt 10px;
   margin-right: 3pt;
}
.AFSResultsWidget_Results-Empty {
   text-align: center;
   font-weight: bold;
   padding: 20px;
}
.AFSResultsWidget_Results-Empty .AFSResultsWidget_Label {
   display: none;
}
```

Il expose les **libellés** suivants :

Nom	Description	Requis	Par défaut
<aucun>[17]	Libellé d'entête. Si ce libellé n'est pas défini, l'emplacement prévu disparaît.	Non	
error	Message d'erreur affiché lorsqu'une erreur se produit.	Non	
empty	Message affiché lorsqu'il n'y a pas de réponses. C'est un libellé paramétré : {0} correspond aux mots-clés.	Non	

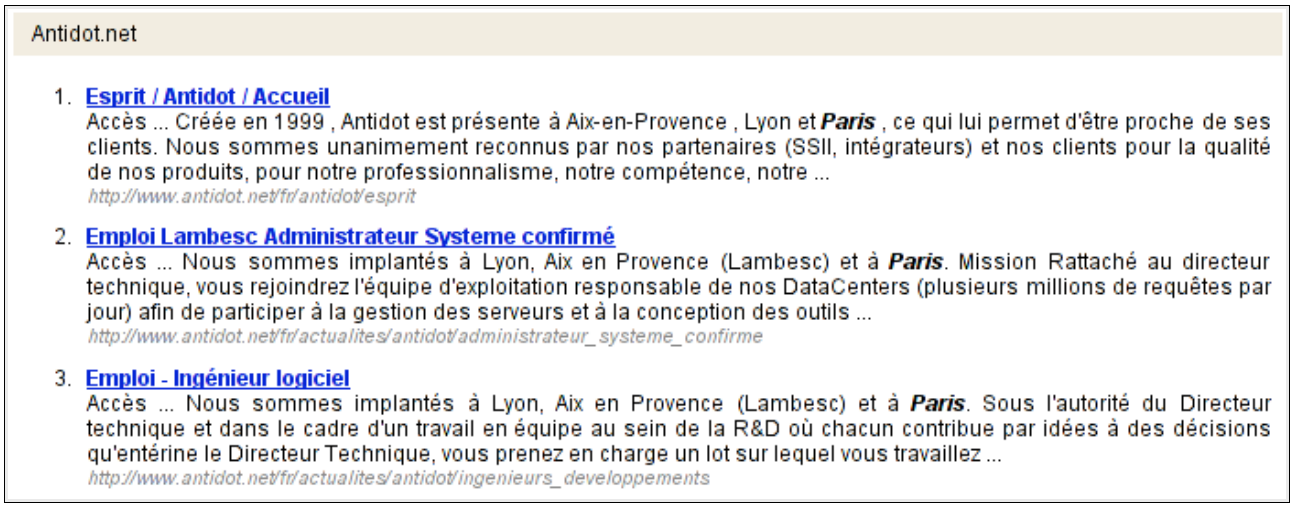
Le libellé « empty » est un libellé paramétré : la séquence « {0} » est remplacée par les mots-clés recherchés lors de son affichage.

Il propose l'option suivante :

Nom	Description	Requis	Par défaut
limit	Limite le nombre de réponses affichées au nombre indiqué par cette option.	Non	Aucune limite

Voici un exemple où les réponses de l'agent « antibot » sont mises en forme avec le formateur « default » :

```
<awml:widget type="results" style="display: none">
   <awml:label>Antidot.net</awml:label>
   <awml:agent names="antibot" format="default"></awml:agent>
</awml:widget>
```



3.3.3. Résultats en clusters (widget « clusteredResults »)

3.3.3.1. Qu'est ce qu'un cluster ?

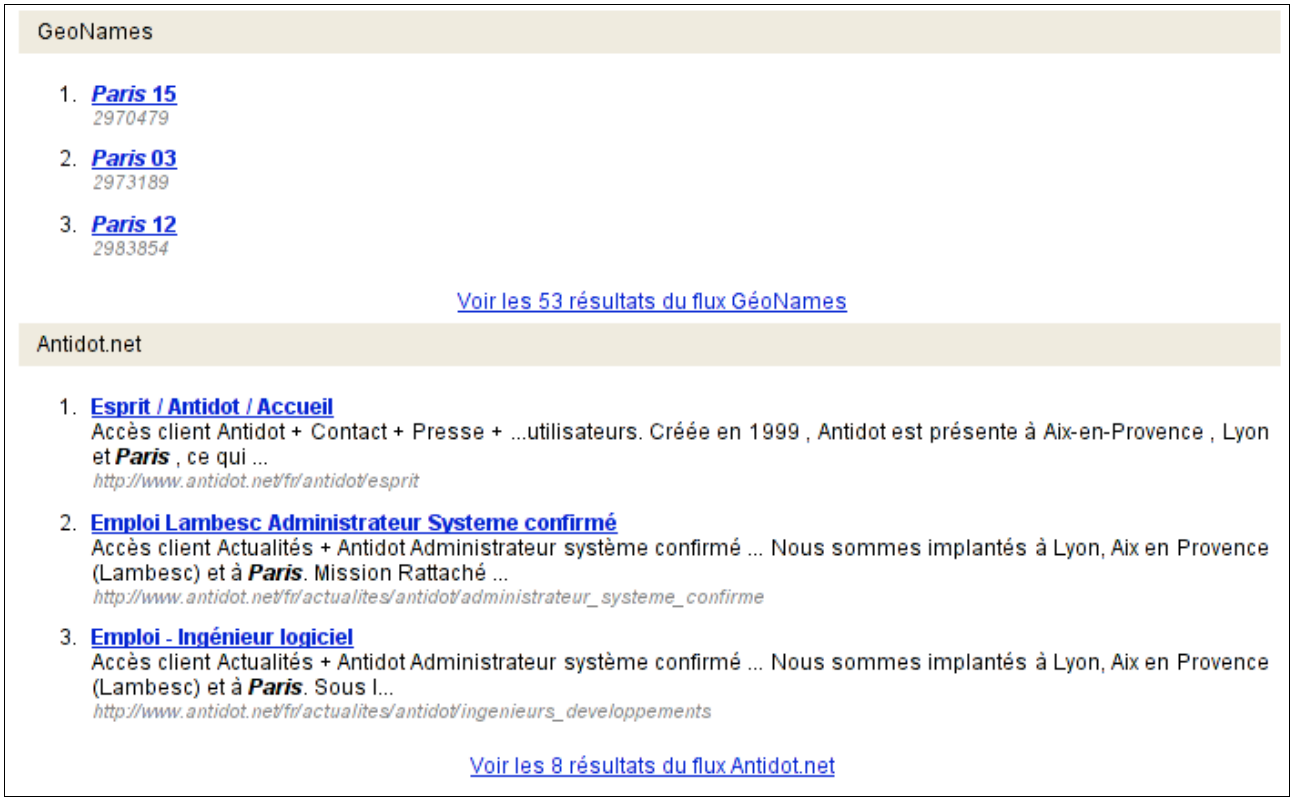
Un cluster peut être vu comme une facette qui agit non pas au niveau réponses, comme le font les facettes, mais au niveau flux de réponse. Un même service de réponse peut contenir plusieurs flux correspondant à des données venant de plusieurs sources (par exemple l'indexation d'un site internet d'un côté et une base de données de l'autre).

Grâce aux options de configuration « dispatch » du moteur de recherche il est possible de limiter les flux à produire suivant la valeur d'un paramètre (généralement « CAT »). Ainsi, il

17 Le libellé n'a pas de nom (voir la section 2.5.3).

est possible de configurer le moteur pour n'obtenir que 10 réponses du flux « antibot » lorsque le paramètre est à « Web » ou 10 réponses du flux « user1 » lorsque le paramètre est à « Database ». Enfin, lorsque le paramètre n'est pas présent, il est possible de configurer le moteur pour qu'il retourne un échantillon de chaque flux : par exemple 3 réponses de chaque.

Les widgets associent un **cluster** à chacune des valeurs possibles du paramètre. Au départ aucun cluster n'est présent et un échantillon de chaque flux est donné par le moteur. L'affichage est alors divisé par paquet : un ensemble de réponses par flux avec un lien permettant d'activer le cluster correspondant à l'ensemble de réponses (lien « Voir les ... réponses... » sur l'image suivante).



Lorsqu'un cluster est actif, seules les réponses de ce flux sont alors affichées. Cela revient à l'affichage simple (widget « results »).

Pour plus d'informations sur les options « dispatch » et les clusters, n'hésitez pas à consulter la documentation du moteur AFS, à contacter le support à l'adresse support@antidot.net ou à demander directement à votre contact privilégié chez Antidot.

3.3.3.2. Déclarer les clusters

Les clusters doivent être déclarés pour que les widgets les connaissent. Cette déclaration est faite dans la balise « awml:service » par l'ajout d'une balise « awml:clusters » et, à l'intérieure, d'une balise « awml:cluster » par cluster.

La **balise « awml:clusters »** propose **l'attribut** suivant :

Nom	***Description***	***Requis***
param	Nom du paramètre permettant d'activer un cluster	Oui

La **balise « awml:cluster »** propose les **attributs** suivants :

Nom	*Description*	*Requis*	*Par défaut*
value	Valeur du paramètre pour activer ce cluster	Oui	
agent	Nom de l'agent associé au cluster (voir ci-dessous)	Non	Aucun

L'attribut « value » permet d'identifier un cluster. Il est utilisé par exemple dans l'option « clusters » du widget « facets ».

L'attribut « agent » permet d'associer un agent au cluster. Lors de l'activation du cluster, l'appel au moteur se fait alors avec le paramètre « UNIQUE » affecté du nom de l'agent. Cet appel permet de forcer l'exclusion des réponses des autres agents et peut être utile dans certaines configurations. Cette information offre également des optimisations au niveau de certains widgets, il est donc intéressant d'ajouter cet attribut.

Enfin, le **contenu** de la balise « awml:cluster » correspond au **libellé**. Il est utilisé par exemple dans le widget « clusters ».

3.3.3.3. Widget « clusteredResults »

Le widget « clusteredResults » permet d'afficher des résultats lorsque le service de recherche propose des clusters. Lorsqu'un cluster est actif, seules les réponses pour ce cluster sont visibles dans le résultat du moteur de recherche. Le widget a alors le même fonctionnement que le widget « results » (dont il est issu).

Lorsqu'aucun cluster n'est actif, le moteur fournit quelques réponses pour chaque cluster. Le widget affiche ces réponses regroupées par cluster : c'est le mode « sommaire ». Pour chaque cluster, et quand il y a plus de réponses que celles actuellement visibles, le widget ajoute un lien d'exploration permettant d'activer ce cluster.

Ce widget étend les fonctionnalités du widget « results » par l'ajout de balises « awml:cluster ». Ces balises permettent de définir les options, les libellés et la mise en forme des résultats pour un cluster. Seuls les clusters ayant une balise « awml:cluster » sont traités.

Les **balises « awml:cluster »** proposent **l'attribut** suivant :

Nom	*Description*	*Requis*
names	La liste des clusters associés à cette balise. Les clusters sont identifiés par leur valeur et séparés par des virgules (« , »).	Oui

Ces balises peuvent contenir des balises « awml:option », « awml:label » et « awml:agent ». Les options et libellés non déclarés dans une balise « awml:cluster » sont recherchés au niveau du widget (et si aucune déclaration n'est trouvée alors la valeur par défaut est utilisée). Enfin, les déclarations de formateurs (balise « awml:agent ») au niveau du widget sont prises en compte si aucune déclaration spécifique au cluster n'existe.

Les balises « awml:cluster » peuvent donc être vues comme un widget « results » dans le widget « clusteredResults ». En mode « sommaire » toutes les balises sont traitées, mais quand un cluster est actif, seule la balise correspondante au cluster actif l'est.

Les **classes CSS, options et libellés** sont les mêmes que pour le widget « results » et s'appliquent à chaque cluster. Seule le libellé « error » s'applique globalement : en cas d'erreur aucun cluster n'est affiché et tout le widget est remplacé par le libellé d'erreur.

Les déclarations de cluster proposent également le **libellé** suivant :

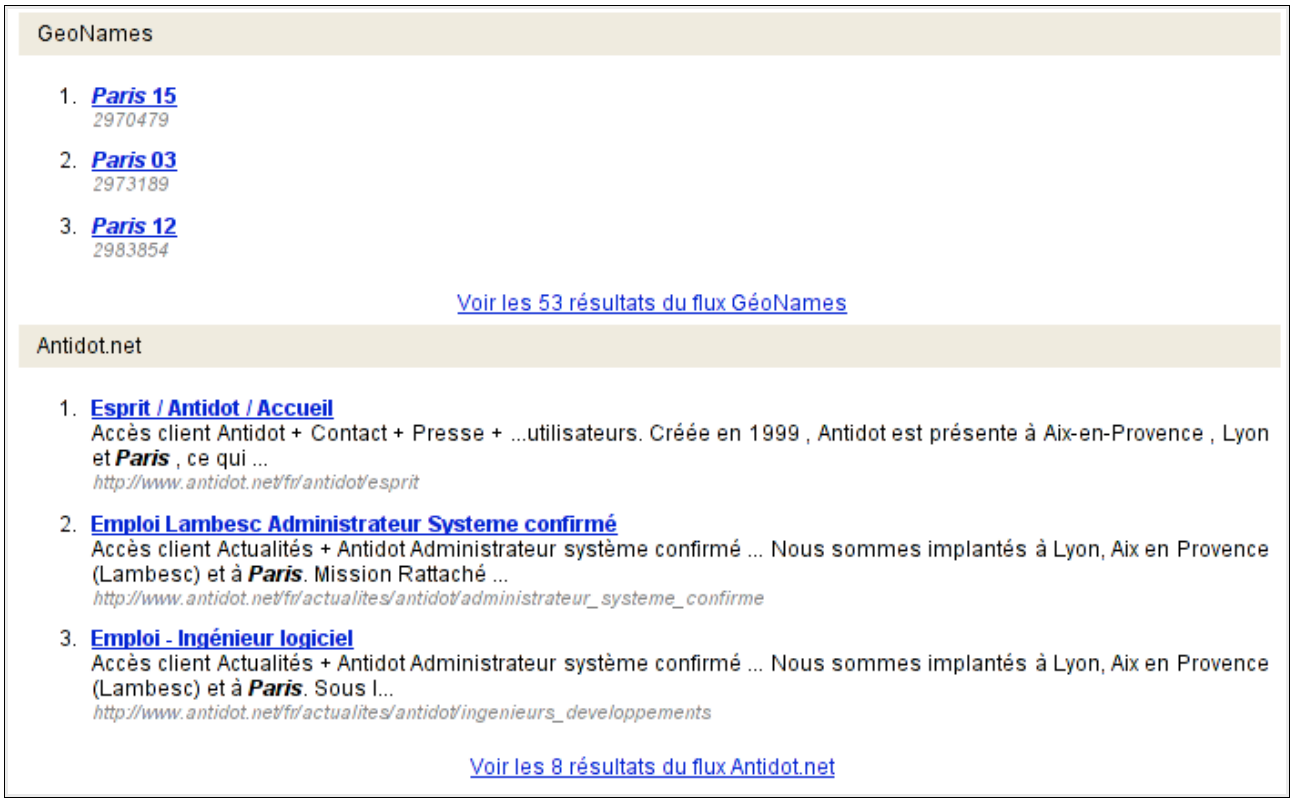
Nom	*Description*	*Requis*	*Par défaut*
more	Libellé du lien d'exploration affiché en mode « sommaire ». Le libellé est un libellé paramétré : {0} correspond au nombre de réponses, {1} à la valeur du paramètre pour le cluster et {2} au libellé du cluster.	Non	Show {0} results for {2} >>

Les déclarations de cluster proposent également **l'option** suivante :

Nom	*Description*	*Requis*	*Par défaut*
summaryLimit	Nombre de réponses à afficher en mode « sommaire » (0 pour toutes celles disponibles). Le lien d'exploration n'est alors visible que si le moteur a retourné plus de réponses que cette limite.	Non	0

Voici un exemple reposant sur un service avec deux clusters :

```
<awml:widget type="clusteredResults" style="display: none">
   <awml:label name="empty">Aucun r&eacute;sultat</awml:label>
   <awml:label name="more">Voir les {0} r&eacute;sultats du flux {2}</awml:label>
   <awml:cluster names="GeoNames">
      <awml:label>GeoNames</awml:label>
      <awml:agent names="user1"></awml:agent>
   </awml:cluster>
   <awml:cluster names="Antidot">
      <awml:label>Antidot.net</awml:label>
      <awml:agent names="antibot"></awml:agent>
   </awml:cluster>
</awml:widget>
```



3.3.4. Résultats groupés (widget « groupedResults »)

3.3.4.1. Qu'est ce qu'un groupement ?

Il est possible de demander au moteur de produire des résultats groupés par rapport à la valeur d'une facette. Il y a alors autant de groupes qu'il y a de valeurs possibles pour la facette et le moteur propose quelques résultats pour chaque groupe. Explorer ce groupe revient ensuite à filtrer les résultats suivant la valeur de la facette pour le groupe à explorer (lien « Voir les ... » dans l'image ci-dessous).

Un groupement a lieu quand le moteur est appelé avec le paramètre « GROUP_BY ». Il est possible de préciser le nombre de résultats à afficher pour chaque groupe, voire paginer les groupes quand il y en a beaucoup (comme sur l'image ci-dessus où un pager apparaît).

Pour plus d'informations sur le paramètre « GROUP_BY » et les groupements, n'hésitez pas à consulter la documentation du moteur AFS, à contacter le support à l'adresse support@antidot.net ou à demander directement à votre contact privilégié chez Antidot.

3.3.4.2. Déclarer le groupement

Le groupement à utiliser doit être déclaré pour que les widgets le connaissent. Cette déclaration est faite dans la balise « awml:service » par l'ajout d'une balise « awml:categorizations » et, à l'intérieur, d'une balise « awml:categorization »[18].

18 Cette structure a été choisie pour permettre dans le futur la déclaration de plusieurs groupements. Actuellement, une seule est supportée.

La **balise « awml:categorizations »** propose **l'attribut** suivant :

Nom	***Description***	***Requis***	***Par défaut***
default	La valeur du groupement à activer au chargement des widgets.	Non	Aucun

La **balise « awml:categorization »** propose les **attributs** suivants :

Nom	***Description***	***Requis***	***Par défaut***
value	Valeur du paramètre « GROUP_BY » pour activer ce groupement.	Oui	
agent	Nom de l'agent associé au groupement (voir ci-dessous)	Non	Aucun
filter	Nom du filtre à utiliser lorsque l'utilisateur explore un groupe	Non	Le nom du filtre suivi de « _FILTER »
nbResults	Nombre de résultats à afficher dans les groupes	Non	3
nbGroups	Nombre de groupes par page	Non	10

L'attribut « value » permet d'identifier le groupement. Il est utilisé, par exemple, pour l'attribut « default » de la balise « awml:categorizations » pour activer un groupement au chargement des widgets.

L'attribut « agent » permet d'associer un agent au groupement. Lors de l'activation du groupement, l'appel au moteur se fait alors avec le paramètre « UNIQUE » affecté du nom de l'agent. Cet appel permet de forcer l'exclusion des réponses des autres agents et peut être utile dans certaines configurations.

Enfin, le **contenu** de la balise « awml:categorization » correspond au **libellé**.

3.3.4.3. Widget « groupedResults »

Le widget « groupedResults » prend en charge l'affichage des résultats lorsqu'un groupement est actif (mode « sommaire). Dans ce cas, il produit une liste de réponses par groupe, précédée d'un libellé. Quand un groupe a plus de réponses disponibles que visibles, il ajoute un lien d'exploration. Ce lien rappelle le moteur en plaçant un filtre sur la facette associée au groupement et dont la valeur est celle du groupe.

Quand aucun groupement n'est actif, le widget se comporte comme le widget « results », affichant les résultats disponibles.

La déclaration du widget est similaire à celle du widget « results ». La différence est qu'un ensemble de réponses est généré pour chaque groupe en mode « sommaire » ; ce qui peut être vu comme une évaluation répétée du widget.

Les **classes CSS, options et libellés** sont les mêmes que pour le widget « results ».

Toutefois le widget propose également les **libellés** suivants :

Nom	***Description***	***Requis***	***Par défaut***
more	Libellé du lien d'exploration affiché en mode « sommaire ». Le libellé est un libellé paramétré : {0} correspond au nombre de réponses, {1} à la valeur du paramètre pour le cluster et {2} au libellé du cluster.	Non	Show {0} results for {2} >>
group	Libellé précédent les résultats d'un groupe. Ce libellé est paramétré : {0} correspond au nombre total de réponses, {1} au libellé du groupe tel que lu dans la réponse du moteur. Si aucun libellé n'est précisé, la zone prévue à cet effet est masquée.	Non	

Le libellé « group » remplace le libellé sans nom en mode « sommaire ».

Voici un exemple d'utilisation :

```
<awml:widget type="groupedResults" style="display: none">
   <awml:label>Résultats</awml:label>
   <awml:label name="group">Résultats pour {1} ({0})</awml:label>
   <awml:label name="more">Voir les {0} r&eacute;sultats du groupe {1}</awml:label>
   <awml:label name="empty">Aucun résultat</awml:label>
</awml:widget>
```

3.4. Les autres widgets

3.4.1. Widget « hint »

Le widget « hint » affiche les suggestions orthographiques liées aux mots-clés recherchés.

Il supporte les **classes CSS** suivantes :

- *AFSHintWidget* : le widget ;
- *AFSHintWidget_Label* : le libellé précédent la suggestion orthographique ;
- *AFSHintWidget_Suggest* : les mots suggérés.

Il propose **la sous-classe CSS** suivante :

- *-NoResults* : appliquée au widget quand il y a des suggestions et que le moteur n'a retourné aucune réponse pour les mots-clés en court. Cela permet de n'afficher les suggestions orthographiques que lorsqu'ils n'y a pas de réponses.

Exemple :

```
.AFSHintWidget {
   background: #EFE9DA;
   padding: 4pt 30pt;
   margin: 8pt 0;
   display: none;
}
.AFSHintWidget-Set {
   display: block;
}
.AFSHintWidget div {
   display: inline;
}
.AFSHintWidget_Suggest a {
   font-style: italic;
}
```

Le **libellé** suivant est mis à disposition :

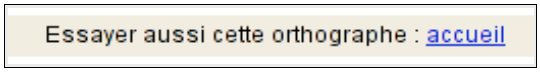
Nom	*Description*	*Requis*	*Par défaut*
<aucun>[19]	Libellé à afficher avant la suggestion orthographique	Non	Hint:

L'**option** suivante est disponible :

Nom	*Description*	*Requis*	*Par défaut*
keepFacets	« true » pour conserver les valeurs des facettes actives lors d'une nouvelle recherche ; « false » sinon.	Non	false

Voici un exemple de son utilisation :

```
<awml:widget type="hint" style="display: none">
   <awml:label>Essayer aussi cette orthographe : </awml:label>
</awml:widget>
```



19 Le libellé n'a pas de nom (voir la section 2.5.3).

3.4.2. Widget « kword »

Le widget « kword » affiche les réponses provenant de l'agent promotion présent dans le flux de réponse. Cet affichage peut inclure un titre et/ou une image et/ou une description et/ou une URL : les éléments à afficher sont dictés par les options spécifiées.

Ce widget offre une mise en forme standardisée des Kword. Pour des mises en formes plus évoluées, il suffit d'utiliser le widget results avec un formateur « custom » (les Kword étant les résultats de recherche de l'agent « promotion » ou « commercial »). Par exemple :

```
<awml:widget type="results" style="display:none">
   <awml:agent names="promotion" format="custom">
      <html:a href="{OptionReply/Page/URI/text()}">{Title/text()}</html:a>
      <html:div>{OptionReply/@ReplyLabel}</html:div>
   </awml:agent>
</awml:widget>
```

Ce widget supporte les **classes CSS** suivantes :

- *AFSKwordWidget : le widget* ;
- *AFSKwordWidget_Label* : le libellé précédant la liste des promotions ;
- *AFSKwordWidget_Kwords* :la liste des promotions ;
- *AFSKwordWidget_KwordItem* : une élément de la liste des promotions ;
- *AFSKwordWidget_Kword* :une promotion ;
- *AFSKwordWidget_Title* : le titre d'une promotion ;
- *AFSKwordWidget_Image* : l'image d'une promotion ;
- *AFSKwordWidget_URL* : l'URL d'une promotion ;
- *AFSKwordWidget_Description* : la description d'une promotion.

Exemple :

```
.AFSKwordWidget {
   margin: 5pt 0pt;
   margin-left: 10pt;
   margin-top: 20pt;
   margin-right: 120pt;
   margin-left: 20pt;
   display: none;
   border: 1px dotted #DED8C8;
}
.AFSKwordWidget-Set {
   display: block;
}
.AFSKwordWidget_Image {
   float: left;
   margin: 2px;
   margin-right: 4px;
}
.AFSKwordWidget_Title {
   font-weight: bold;
}
.AFSKwordWidget_URL a {
   font-style: italic;
   font-size: 9pt;
   text-decoration: none;
   color: #777777;
}
```

Il expose les **libellés** suivants :

Nom	*Description*	*Requis*	*Par défaut*
<aucun>[20]	Libellé à placer juste avant la liste des promotions. Il permet d'introduire la liste. Si ce libellé n'est pas défini, l'emplacement prévu disparaît.	Non	
url	Libellé de l'URL à afficher. La séquence « {0} » est remplacée par la valeur de l'URL dans le flux de réponse. Ce libellé permet notamment de distinguer l'URL à afficher et la cible de l'URL.	Non	La valeur de l'option « urlPattern »

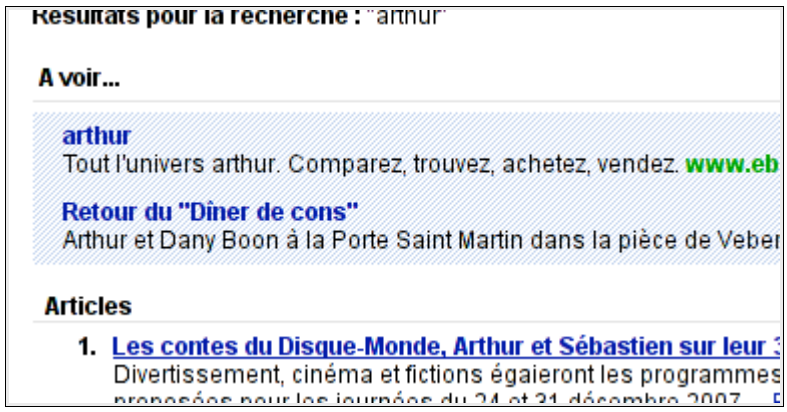
Les **options** suivantes sont disponibles :

Nom	*Description*	*Requis*	*Par défaut*
showTitle	« true » pour afficher le titre de chaque promotion ; « false » sinon	Non	true
showImage	« true » pour afficher l'image de chaque promotion ; « false » sinon.	Non	true
showDescription	« true » pour afficher la description de chaque promotion ; « false » sinon.	Non	true
showURL	« true » pour afficher l'URL de chaque promotion ; « false » sinon.	Non	true
urlPattern	La cible de l'URL pour les promotions. La séquence « {0} » est remplacée par la valeur de l'URL dans le flux de réponse. Cette option permet notamment de distinguer le libellé et la cible de l'URL.	Non	{0}
urlMaxLength	Longueur maximale (en caractères) de l'URL lors de son affichage. Si elle est trop longue, l'URL est coupée en son milieu (et « ... » est ajouté) pour limiter sa longueur à la valeur indiquée.	Non	80
target	Précise la cible de l'URL lorsque l'internaute clique dessus. Les valeurs possibles sont les mêmes que l'attribut « target » de la balise HTML « a ».	Non	_current

20 Le libellé n'a pas de nom (voir la section 2.5.3).

Voici un exemple :

```
<awml:widget type="kword" style="display: none">
   <awml:option name="showURL">false</awml:option>
   <awml:option
      name="urlPattern">http://www.mondomain.com/page.php?id={0}</awml:option>
   <awml:option name="target">_blank</awml:option>
</awml:widget>
```



3.4.3. Widget « rte »

Le widget « rte » affiche les expressions proches (RTE, *Related Topic Expression*). Ces expressions suggèrent à l'internaute soit une recherche parallèle, en relançant la recherche sur l'expression donnée, soit une recherche plus approfondie en ajoutant l'expression suggérée aux mots-clés actuels.

Le widget « rte » propose toujours la recherche parallèle via la liste des expressions suggérées. La recherche approfondie est proposée si le libellé « append » est défini ; dans ce cas, chaque expression est suivie de ce libellé qui permet de lancer la recherche approfondie.

Ce widget supporte les **classes CSS** suivantes :

- *AFSRTEWidget* : le widget ;
- *AFSRTEWidget_Label* : le libellé précédent la liste des expressions ;
- *AFSRTEWidget_Append* : le libellé « append » lorsqu'il est affiché ;
- *AFSRTEWidget_RteWords* : les expressions suggérées ;
- *AFSRTEWidget_Items* : la liste des expressions ;
- *AFSRTEWidget_Item* : une expression ;

Exemple :

```
.AFSRTEWidget {
   background: #EFE9DA;
   float: right;
   margin: 3pt;
   display: none;
   border: 1px dotted #DED8C8;
   margin-top: 48pt;
}
.AFSRTEWidget-Set {
   display: block;
}
.AFSRTEWidget_Append {
   display: inline;
}
```

```
.AFSRTEWidget_RteWords {
   display: inline;
}
.AFSRTEWidget_Label {
   font-weight: bold;
   background: #DED8C8;
   padding: 4px;
}
```

Il expose les **libellés** suivants :

Nom	*Description*	*Requis*	*Par défaut*
<aucun>[21]	Libellé à placer juste avant la liste des expressions. Il permet d'introduire la liste. Si ce libellé n'est pas défini, l'entête disparaît.	Non	
append	Libellé affiché après chaque expression pour permettre à l'internaute de lancer une recherche approfondie avec l'expression suggérée. Si ce libellé est vide, la recherche approfondie n'est pas proposée.	Non	

L'**option** suivante est disponible :

Nom	*Description*	*Requis*	*Par défaut*
keepFacets	« true » pour conserver les facettes courantes pour la nouvelle recherche ; « false » sinon.	Non	false

Voici un exemple d'utilisation du widget :

```
<awml:widget type="rte" style="display: none">
   <awml:label><html:strong>Expressions connexes :</html:strong></awml:label>
</awml:widget>
```

3.4.4. Widget « history »

Le widget « history » propose un historique des recherches effectuées. Il permet d'accéder aux recherches les plus récentes.

Ce widget supporte les **classes CSS** suivantes :

- *AFSHistoryWidget* : le widget ;
- *AFSHistoryWidget_Header* : l'entête de la liste ;
- *AFSHistoryWidget_Label* : le libellé de la liste ;
- *AFSHistoryWidget_Content* : le conteneur de la liste ;
- *AFSHistoryWidget_Items* : la liste ;
- *AFSHistoryWidget_Item* : un élément de la liste ;

21 Le libellé n'a pas de nom (voir la section 2.5.3).

– *AFSHistoryWidget_ItemLabel* : le libellé d'un élément de la liste.

Exemple :

```
.AFSHistoryWidget {
   border: 1px solid #ACA693;
   background: #EFE9DA;
   width: 244px;
   margin: auto;
   margin-bottom: 4pt;
   display: none;
}
.AFSHistoryWidget-Set {
   display: block;
}
.AFSHistoryWidget_Label {
   background: #DED8C8;
   font-weight: bold;
   padding: 3pt;
}
.AFSHistoryWidget_Item {
   color: #0000cc;
   text-decoration: underline;
   cursor: pointer;
   margin: 0;
   padding: 0;
}
```

Il expose le **libellé** suivant :

Nom	***Description***	***Requis***	***Par défaut***
<aucun>	Libellé à placer juste avant la liste des recherches effectuées. Il permet d'introduire la liste. Si ce libellé n'est pas défini, l'emplacement prévu disparaît.	Non	

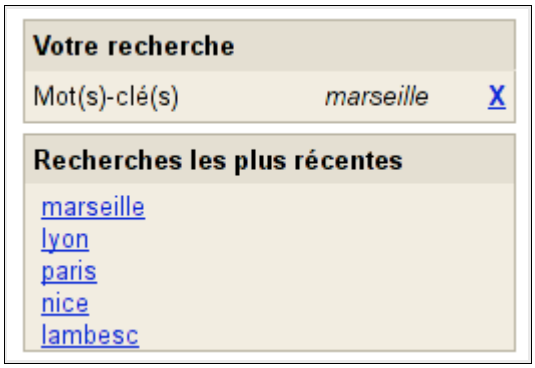
Les **options** suivantes sont disponibles :

Nom	***Description***	***Requis***	***Par défaut***
order	Ordre d'affichage des recherches effectuées. Cette option peut prendre pour valeur « oldestFirst » pour afficher l'historique des recherches de la moins récente à la plus récente et « newestFirst » pour afficher les recherches de la la plus récente à la moins récente.	Non	newestFirst
maximum	Nombre maximum de recherches à afficher	Non	10
cookie	Nom du cookie utilisé pour mémoriser les recherches effectuées.	Non	afsHistory
expire	Délai d'expiration du cookie en minutes.	Non	0

Si l'option « expire » est à 0, l'historique n'est conservé que pendant la session du navigateur (c'est-à-dire jusqu'à la fermeture du navigateur). Le cookie ne conserver que les entrées nécessaires à l'affichage demandé (ie. La valeur de l'option « maximum »).

Voici un exemple d'utilisation du widget :

```
<awml:widget type="history" style="display:none">
   <awml:label>Recherches les plus r&eacute;centes</awml:label>
   <awml:option name="order">newestFirst</awml:option>
   <awml:option name="maximum">5</awml:option>
   <awml:option name="cookie">AFSHistoryCookie</awml:option>
<awml:option name="expire">30</awml:option>
</awml:widget>
```



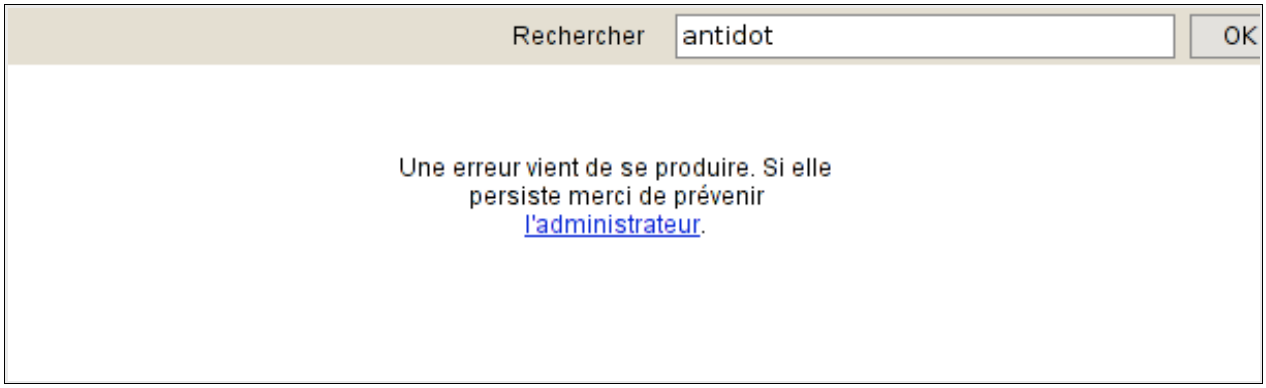
4. Aller plus loin

Cette section aborde des concepts plus avancés qui permettent de rendre l'intégration des widgets dans la page plus complète.

4.1. Afficher un message d'erreur

Lors d'une recherche il peut se produire des erreurs. Par exemple, une erreur réseau peut survenir, le moteur de recherche peut être soudainement surchargé, il ne peut plus être contacté... Les widgets « results », « clusteredResults » et « groupedResults » proposent, par le biais d'un libellé appelé « error », l'affichage d'un message d'erreur personnalisable lorsqu'une erreur se produit. Le message d'erreur apparaît alors en lieu et place des résultats de la recherche.

```
<awml:widget type="results" style="display: none">
   <awml:label name="error">
      Une erreur vient de se produire.
      Si elle persiste merci de prévenir
      <html:a href="mailto:admin@mondomaine.com">l'administrateur</html:a>.
   </awml:label>
</awml:widget>
```



L'événement JavaScript « onerror » peut également être écouté pour être averti des erreurs. Pour plus d'informations consulter la section 4.3.

4.2. Afficher un message d'attente

Entre le moment où l'internaute agit sur les widgets et le moment où ceux-ci affichent de nouveaux résultats, il s'écoule un temps d'attente qui est fonction de la charge du serveur, des aléas du réseau, du temps de chargement des résultats par le navigateur du client...

Certains widgets proposent, par le biais d'un libellé nommé « wait », d'afficher un message d'attente. Ce message est visible dès qu'une nouvelle recherche est lancée et jusqu'à ce qu'elle aboutisse, qu'elle soit annulée ou qu'une erreur se produise. Le libellé est affiché par dessus le contenu actuel du widget, dans un calque HTML qui vient recouvrir la totalité du widget tout en restant transparent. Cela permet de continuer à afficher les résultats de la recherche précédente tout en empêchant l'internaute d'agir dessus (par exemple en choisissant une nouvelle valeur pour une facette).

Ce calque a pour classe CSS « AFSWidget_Shadow » et le libellé peut contenir n'importe quel élément HTML. Il est donc possible d'afficher une animation (qui peut être lancée et arrêtée grâce aux événements JavaScript, voir la section 4.3), d'appliquer un calque de couleur avec une légère transparence pour atténuer le widget...

Les widgets qui supportent ce libellé sont :« hint », « info », « pager », « facets», « autoFacets », « kword », « results », « clusteredResults », « groupedResults » et « rte ».

Si le libellé n'est pas défini, aucun message d'attente n'apparaît et le widget reste opérationnel pendant qu'une recherche a lieu.

Voici un exemple :

```
<style type="text/css">
   .Attente1 {
      position: absolute;
      width: 100%; height: 100%;
      background: silver;
      filter:alpha(opacity=80);
      opacity: 0.8;
      z-index: 1;
   }
   .Attente2 {
      width: 100%; height: 100%;
      position: absolute;
      z-index: 2;
   }
   .Attente2 td {
      text-align: center;
      vertical-align: middle;
      font-weight:bold;
   }
</style>

<awml:widget type="results" style="display: none">
   <awml:label name="wait">
      <html:div class="Attente1"></html:div>
      <html:table class="Attente2"><html:tr><html:td>
         Recherche en cours...
      </html:td></html:tr></html:table>
   </awml:label>
</awml:widget>
```

Le style « Attente1 » produit un cache gris semi-transparent au dessus des résultats de la recherche ; le style « Attente2 » place le texte au centre du widget. Durant la recherche, l'affichage devient alors :

4.3. Interaction via le JavaScript

Les widgets offrent des événements JavaScript permettant d'être informé des changements d'état de la recherche (interrogation du moteur, chargement de résultats, erreur...). A cet effet, la balise « awml:service » déclarant le service propose les attributs suivants :

- « onsearch » : une recherche vient d'être lancée ;
- « onendsearch » : la dernière recherche lancée est terminée ;
- « onload » : la réponse du moteur vient d'être chargée ;
- « onerror » : une erreur s'est produite [22] ;
- « onclear » : la dernière recherche a été effacée[23] ;
- « oncancelsearch » : une recherche en cours a été annulée. Cela arrive lorsque, par exemple, l'internaute lance deux recherches d'affilé : la première est annulée par la seconde et l'événement « oncancelsearch » a lieu juste avant que l'événement « onsearch » de la seconde recherche ait lieu.

Lorsqu'un événement a lieu, tous les widgets sont mis à jour avant.

Ces événements permettent par exemple le lancement d'une animation d'attente quand une recherche est lancée comme cela est fait dans le kit webmestre fourni.

Voici un exemple :

```
<awml:service url="/AFSWidgets/prod/" style="display: none"
   onload="alert('Chargé');">
   <awml:param name="C">1</awml:param>
   <awml:param name="afs:output">xml</awml:param>
</awml:service>
```

4.4. Recherche dès le chargement

Il est toujours intéressant de pouvoir lancer une recherche sur un site depuis n'importe quelle page du site. Mais il serait fastidieux d'intégrer les Widgets d'AFS dans chaque page de ce site. Il est préférable d'intégrer les widgets dans une page du site : la page de recherche. Les autres pages ont alors un formulaire qui ouvre la page de recherche en lançant une recherche immédiatement.

Pour que cela fonctionne il faut tout d'abord mettre l'option « usePageParams » du service (balise « awml:service ») à « true ». Ainsi, les widgets utilisent les paramètres de la page dès leur chargement.

Les widgets utilisent comme paramètre la référence de la page (ce qui suit le caractère « # » dans l'adresse d'une page). Cette référence doit avoir un format particulier : elle doit commencer par « search: » suivit des mots-clés à rechercher (ces mots-clés doivent être encodés au format URL). La suite de la référence contient les options et les filtres qui sont séparés les uns des autres (et des mots clés) par le caractère « / ».

Les options commencent par « option: » suivi du nom de l'option (encodée au format URL), du caractère « = » et de sa valeur (encodée au format URL). Les filtres utilisent la même syntaxe si ce n'est qu'ils commencent par « filter: »

Par exemple, la recherche du mot-clé « antidot » donne l'URL :

```
recherche.html#search:antidot
```

22 Erreur du moteur ou du réseau

23 Cet événement a également lieu au chargement de la page XHTML dans le navigateur s'il n'y a pas de recherche automatiquement lancée

De même, la recherche du mot-clé « antidot » avec le filtre « Type » à 3 tout en demandant la page 4 :

```
recherche.html#search:antidot/option:page=4/filter:Type_FILTER=3
```

Sur certains navigateurs la référence peut être encodée au format URL et mener à un double encodage des mots-clés, des options ou des filtres. L'exemple précédent sur le navigateur Firefox devient :

```
recherche.html#search%3Aantidot%2Foption%3Apage%3D4%2Ffilter%3AType_FILTER%3D3
```

L'ajout d'une zone de recherche dans l'ensemble des pages d'un site Web peut être faite comme dans l'exemple suivant :

```
<html>
   <head>
      <script type="text/javascript">
      function onLoad(form) {
         var keywords = form.keywords.value;
         var url = form.action;
         document.location = url + "#search:" + escape(keywords);
         return false ;
      }
      </script>
   </head>

   <body>
      <div>Zone de recherche
         <form onsubmit="return onLoad(this);"
            action="http://localhost/index.html">
            <input type="text" value="" name="keywords"/>
            <input type="submit" value="Rechercher"/>
         </form>
      </div>
   </body>
</html>
```

La définition du formulaire ci-dessus doit être adaptée au contexte du site Web à concevoir. L'utilisation de ce type de zone de recherche est illustrée dans le kit webmestre par le fichier « index.html » du répertoire samples.

5. Kit webmestre

Le kit destiné aux webmestres contient la bibliothèque JavaScript des Widgets d'AFS et des pages d'exemple. Ces exemples sont prévus pour fonctionner avec 3 services de recherche représentatifs des principales fonctionnalités d'AFS.

5.1. La bibliothèque JavaScript

Le dossier « lib » du kit contient tous les fichiers nécessaires au fonctionnement des widgets : des scripts JavaScript, des pages HTML, des fichiers XML, des images... D'une version à l'autre du produit, le nom des fichiers peut changer mais il y a toujours deux fichiers :

- *AFSWidgets.xsd* : le schéma XML des balises AWML. Il permet de valider une page XHTML contenant des balises AWML ;
- *net.antidot.widgets.AFSWidgets.nocache.js* : le script JavaScript principal qui doit être chargé dans la page HTML où les widgets doivent être insérés.

Les autres fichiers sont automatiquement chargés. Tous ces fichiers doivent se trouver sur le même serveur Web. Ils peuvent toutefois se trouver à n'importe quelle profondeur et dans un dossier différent de la page.

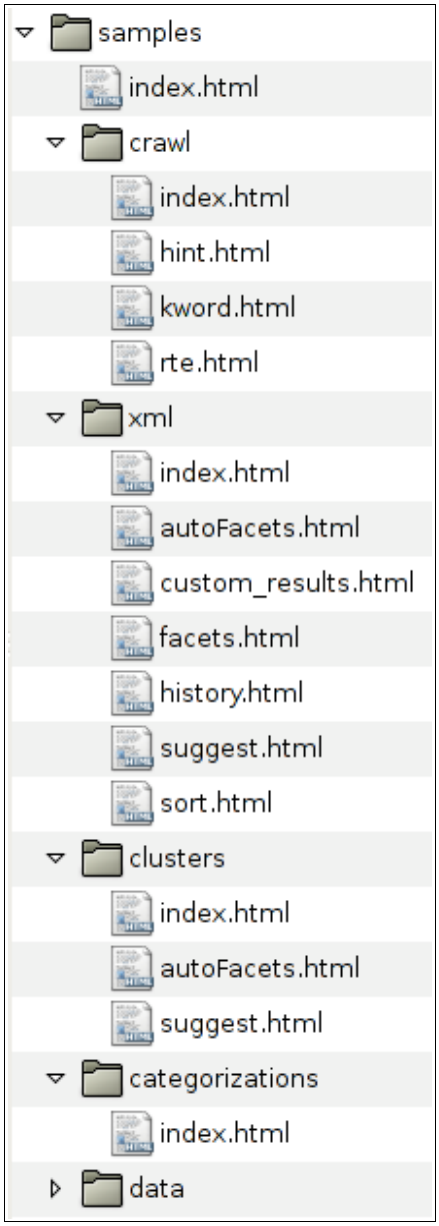
5.2. Les exemples

Le dossier « samples » du kit contient plusieurs pages illustrant le fonctionnement des widgets. Elles sont basées sur 3 services de démonstration d'Antidot :

- Le service 1 : l'indexation du site internet d'Antidot (http://www.antidot.net/) c'est-à-dire un « crawl » (la fraîcheur des pages n'est pas garantie, il ne s'agit que d'une démonstration) ;
- Le service 3 : l'indexation d'une partie de la base de données GéoNames (http://www.geonames.org/). Ce service permet d'utiliser la fonctionnalité de groupement d'AFS comme expliqué dans la section 3.3.4.1 ;
- Le service 4 : un service mêlant les deux services de recherche précédents pour illustrer les clusters (voir la section 3.3.3.1).

Ces services de recherche ne sont pas inclus dans le kit webmestre. Ils sont disponibles sous la forme d'une machine virtuelle prête à l'emploi sur simple demande. N'hésitez pas à contacter le support à l'adresse support@antidot.net ou de demander directement à votre contact privilégié chez Antidot.

5.2.1. Structure des fichiers



La structure des fichiers est découpée selon les principales configurations d'un service de recherche.

La page « index.html », située dans le répertoire « samples », offre un accès rapide vers les principales pages du kit. Dans chaque sous-dossier la page « index.html » contient tous les widgets nécessaires pour créer une page de recherche complète ; les autres pages HTML présentent séparément chaque widget.

Les exemples du kit sont organisés selon les sous-dossiers suivants :

- **« crawl »** : l'utilisation des widgets pour un service de recherche indexant un site internet (service 1) ;
- **« xml »** : l'utilisation des widgets pour un service de recherche indexant des fiches XML ou une base de données (service 3) ;
- **« clusters »** : l'utilisation des widgets avec un service de recherche utilisant des clusters (service 4) ;
- **« categorizations »** : l'utilisation des widgets sur un service permettant le groupement des résultats (service 3).

Le répertoire « data » contient les fichiers annexes (images, feuille de styles CSS...). Il y a par exemple le fichier JavaScript de la barre de progression affichée pendant les requêtes au moteur.

Enfin, toutes les pages utilisent la bibliothèque JavaScript du dossier « lib » à la racine du kit.

5.2.2. Exemples mis à disposition

Tous les exemples de pages HTML ont en commun le widget « keywords », « info » et « pager ». De plus, toutes ces pages HTML sont basées sur la même charte graphique.

Chaque page HTML illustre également le widget d'affichage des résultats avec des utilisations qui peuvent varier selon la configuration du service utilisé. Par exemple, le widget « clusteredResults » est utilisé dans le cas du service configuré en clusters (voir la section 3.3.3.1), le widget « groupedResults » est utilisé dans le cas du service proposant le groupement des réponses (voir la section 3.3.4.1) et le widget « results » est utilisé pour les configuration plus simples.

Toutes les pages intitulées « index.html » incluent l'ensemble des widgets disponibles pour la configuration concernée. Par exemple, dans le cas d'un service configuré avec des clusters, il est intéressant d'utiliser le widget « clusters » pour naviguer rapidement d'un cluster à l'autre.

Les autres pages HTML, situées au sein d'un même répertoire, illustrent un widget en particulier. Par exemple, le fichier « custom_results.html » met en valeur l'utilisation du widget « results » avec le formateur « custom » qui offre une grande souplesse dans la présentation des réponses (voir la section 3.3.1.3).

5.2.2.1. Index global du kit

La page « index.html » du répertoire « samples » permet d'accéder aux exemples du kit. Plus précisément, elle offre la possibilité de lancer une recherche dans une des pages « index.html », contenue dans les sous-dossiers. Elle exploite pour cela la technique présentée en section 4.4 où une zone de recherche, insérée dans n'importe qu'elle page du site, offre la page des widgets et lance immédiatement la recherche.

Chacune de ces zones de saisie des mots-clés permet d'interroger le moteur de recherche associé en chargeant la page HTML contenant la déclaration des widgets correspondante.

5.2.2.2. Moteur de recherche d'un service crawl

Le service de recherche utilisé pour ces exemples est le service 1 correspondant à l'indexation du site internet d'Antidot (http://www.antidot.net/).

La page « index.html » du sous-dossier « crawl » présente une configuration type des widgets pour un service de recherche crawl. Elle comporte les widgets :

- de navigation « keywords », « pager » et « context » ;
- d'état « info » ;
- de résultats « results » avec le formateur « default » ;
- « rte » (le moteur produit des RTE pour le mot-clé « accueil ») ;
- « kword » (le moteur produit des Kwords pour le mot-clé « widget ») ;
- « hint » (le moteur produit des Hints pour le mot-clé « accuil ») ;

De plus, ces 3 derniers widgets sont également illustrés par, respectivement, les pages « rte.html », « hint.html » et « kword.html ».

L'affichage obtenu pour la page principale est :

5.2.2.3. Moteur de recherche d'un service XML

Le service de recherche utilisé pour ces exemples est le service 3 correspondant à une indexation de la base de données GéoNames (http://www.geonames.org/).

La page « index.html » du sous-dossier « xml » présente une configuration type des widgets pour un service de recherche XML ou base de données. Elle comporte les widgets :

- de navigation « keywords », « pager », « facets », « context » et « sort » ;
- d'état « info » ;
- de résultats « results » avec le formateur « default » ;
- d'affichage de l'historique des recherches « history ».

La page « suggest.html » montre comment utiliser le widget « keywords » avec un serveur de suggestion (voir la section 3.1.1).

La page « history.html » permet d'afficher l'historique des recherches à l'aide du widget « history ».

La page « sort.html » propose plusieurs tris des résultats par l'intermédiaire du widget « sort ».

La page « facets.html » se focalise sur l'affichage des filtres (image ci-contre). Elle illustre notamment l'utilisation de l'option « multipleSelection » pour les filtres de type « flat » (voir la section 3.1.3.3) et des balises « awml:static » (voir la section 3.1.3.4).

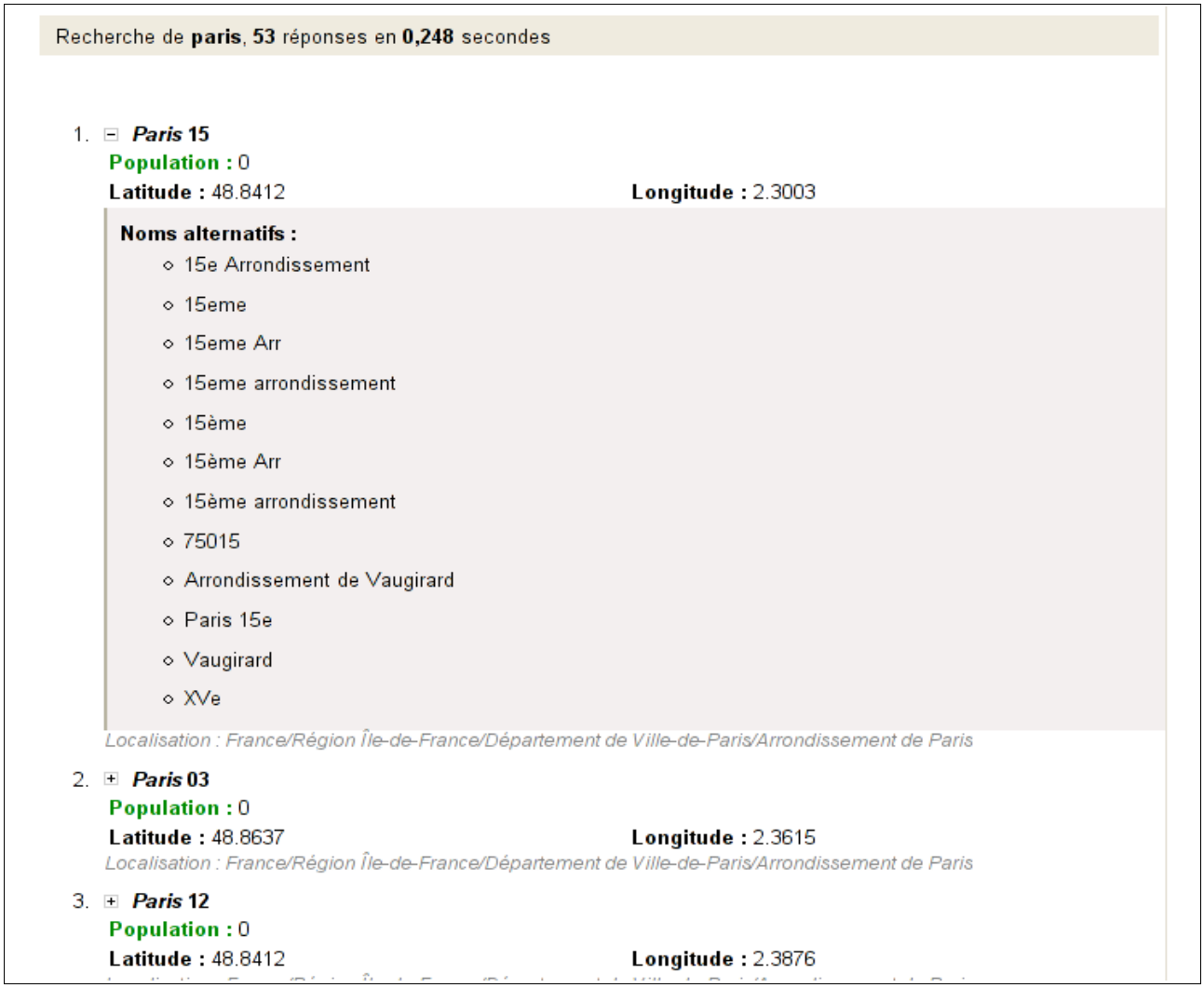
La page « autoFacets.html » utilise le widget « autoFacets » pour générer l'affichage des facettes. Il est intéressant de comparer le résultat de cette page avec la précédente. Le widget « autoFacets » permet de rapidement avoir une vue des facettes disponibles

(voir la section 3.1.4) ; le widget « facets » offre un contrôle total sur l'affichage de ces mêmes facettes (voir la section 3.1.3).

Enfin, la page « custom_results.html » utilise le formateur « custom » pour rendre les réponses du service de recherche dans un format particulier aux données indexées. Ainsi, l'utilisation de la balise « awml:if » permet de n'afficher les données que si elles sont disponibles dans le flux. De même l'utilisation de la balise « awml:for » sur les noms alternatifs des entités produit une liste de ceux-ci. Ainsi pour une réponse XML contenant :

```
<AlternateNames _namespace="alternateNames">
   <AlternateName> 15e Arrondissement </AlternateName>
   <AlternateName> 15eme </AlternateName>
   <AlternateName> 15eme Arr </AlternateName>
   <AlternateName> 15eme arrondissement </AlternateName>
   <AlternateName> 15ème </AlternateName>
   <AlternateName> 15Ã¨me Arr </AlternateName>
   <AlternateName> 15Ã¨me arrondissement </AlternateName>
   <AlternateName> 75015 </AlternateName>
   <AlternateName> Arrondissement de Vaugirard </AlternateName>
   <AlternateName> Paris 15e </AlternateName>
   <AlternateName> Vaugirard </AlternateName>
   <AlternateName> XVe </AlternateName>
</AlternateNames>
```

... produit l'affichage :



5.2.2.4. Moteur de recherche d'un service avec clusters

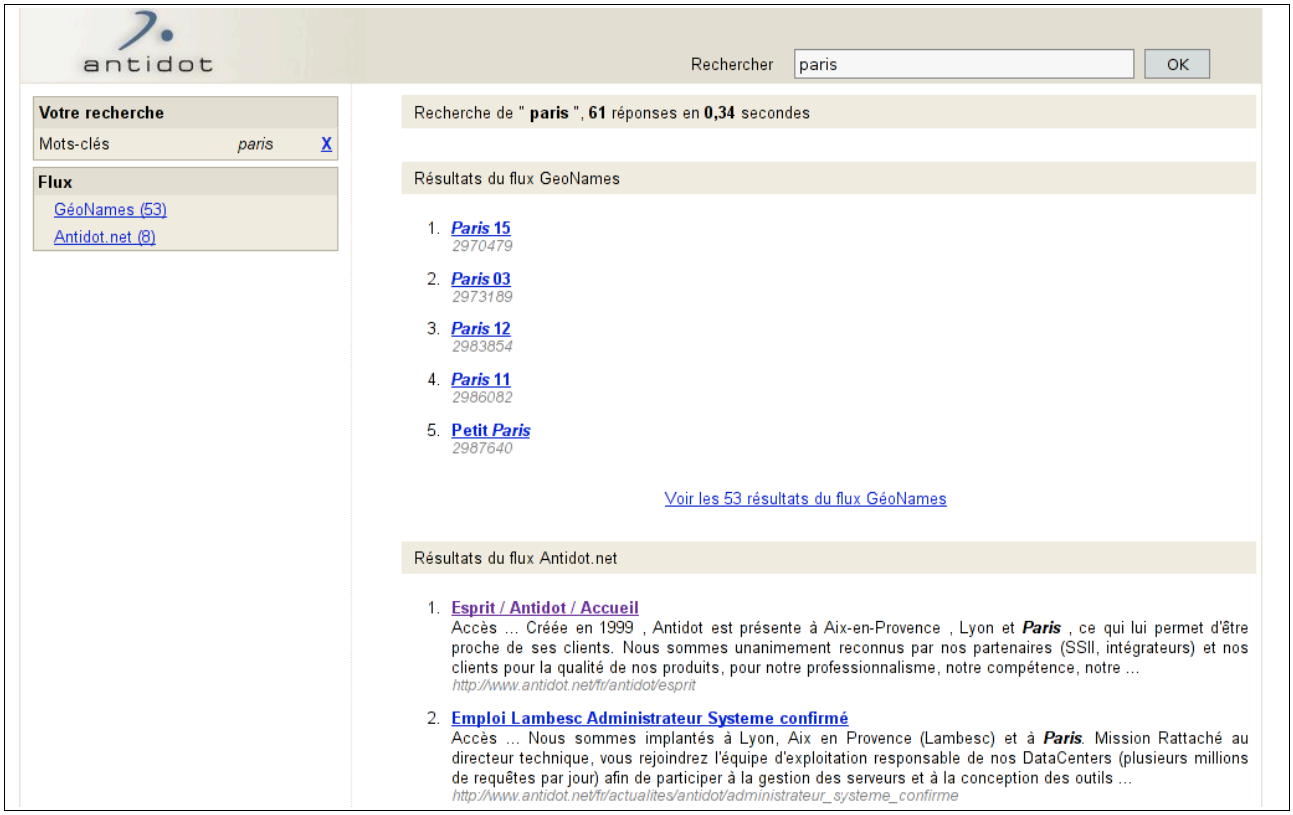
Le service de recherche utilisé pour ces exemples est le service 4 configuré avec des clusters. Il utilise les bases de réponses des services 1 et 3.

La page « index.html » du sous-dossier « clusters » présente une configuration type des widgets pour un service de recherche avec clusters. Elle comporte les widgets :

- de navigation « keywords », « pager », « facets », « context » et « clusters » ;
- d'état « info » ;
- de résultats « clusteredResults » avec le formateur « default » ;

Les deux clusters disponibles sont « Antidot », qui retourne les réponses de la base de réponses du service 1, et « GeoNames » qui retourne les réponses de la base de réponses du service 3.

Lorsque le flux comporte plusieurs clusters, l'utilisation du widget « clusters » est conseillée. Il permet d'une part, d'indiquer l'ensemble des clusters disponibles, et d'autre part, de naviguer entre les clusters. De plus, l'affichage des résultats doit être effectué avec le widget « clusteredResults » afin de regrouper les réponses par cluster (voir la section 3.3.3.3).



Enfin, les pages « autoFacets.html » et « suggest.html » illustrent respectivement le widget « autoFacets » et « suggest ».

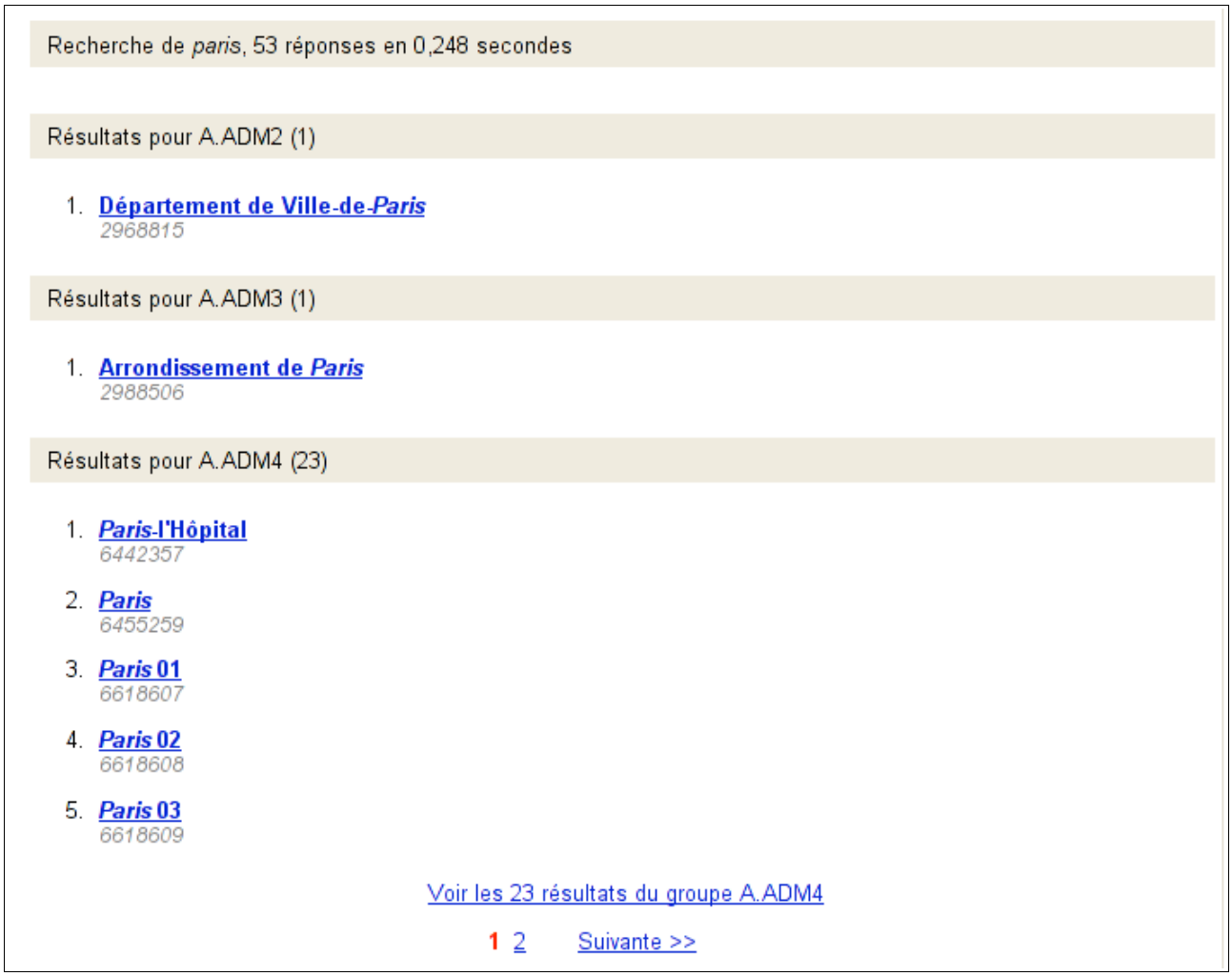
5.2.3. Moteur de recherche d'un service avec groupements

Le service de recherche utilisé pour ces exemples est le service 3 pour lequel il est possible de créer un groupement sur la facette « type ».

La page « index.html » du sous-dossier « categorizations » présente une configuration type des widgets pour un service de recherche avec clusters. Elle comporte les widgets :

- de navigation « keywords », « pager », « autoFacets » et « context » ;
- d'état « info » ;
- de résultats « groupedResults » avec le formateur « default » ;

L'utilisation du widget « groupedResults » permet d'afficher les groupes de réponses quand un groupement est actif.



6. La console d'erreur et la barre d'outils

Durant la phase d'intégration, les widgets peuvent sembler assez opaque sur leur exécution. C'est le comportement par défaut prévu pour les internautes.

Couplés avec l'extension FireBug des navigateurs Mozilla ou avec le script JavaScript FireBug Lite, il devient possible de suivre leur exécution et de mieux comprendre les erreurs.

FireBug et FireBug Lite sont des produits n'ayant aucun lien avec AFS et Antidot. Ils sont disponibles sur le site http://www.getfirebug.com/. Les widgets exploitent ces produits comme console d'erreur lorsqu'ils sont disponibles, restant muets lorsqu'ils ne le sont pas.

6.1. Les niveaux de messages

La console d'erreur permet d'afficher différents niveaux de message en utilisant un code couleur. On distingue trois niveaux de message :

- « error » : signale les erreurs (icône). Elles bloquent le fonctionnement d'un widget voire de tous les widgets ;.
- « warning » : signale les avertissements (icône). Bien qu'ils n'empêchent pas les widgets de fonctionner, les avertissements indiquent un problème dans la déclaration des widgets. L'affichage peut alors différer de celui attendu ;
- « info » : signale les informations (icône) sur le déroulement de l'exécution des widgets.

Les causes possibles des erreurs peuvent être, par exemple :

- un code JavaScript erroné dans les attributs « onload », « onsearch », « onendsearch », « onerror », « oncancel » ou « oncancelsearch » de la déclaration du service (voir la section 4.3) ;
- un problème réseau, du moteur de recherche ou dans la réponse de celui-ci ;
- des attributs manquant au sein des balises AWML.

Une cause habituelle des avertissements vient d'une erreur de XPath dans la déclaration d'un formateur « custom » (voir la section 3.1.3.3).

6.2. Mise en place de la console

L'utilisation de la console d'erreur nécessite l'ajout d'informations dans la page HTML contenant les widgets et de disposer de FireBug ou FireBug Lite.

6.2.1. Préparation de la page

Il est nécessaire de préciser aux widgets quels niveaux de message afficher. Par défaut ils sont configurer pour afficher les niveaux « error » et « warning » (niveau de rapport à « warning »).

Les niveaux de rapport possibles sont :

- « info » : afficher les messages de niveau « info », « warning » et « error » ;
- « warning » : afficher les messages de niveau « warning » et « error » ;
- « error » : afficher les messages de niveau « error » uniquement ;
- « none » : n'afficher aucun message.

Pour choisir le niveau de rapport, il faut placer une balise « meta » particulière dans la balise « head » de la page HTML. L'attribut « name » de la balise doit être « awml:log_level » et l'attribut « content » doit être affecté du niveau désiré.

Par exemple, pour activer le niveau info :

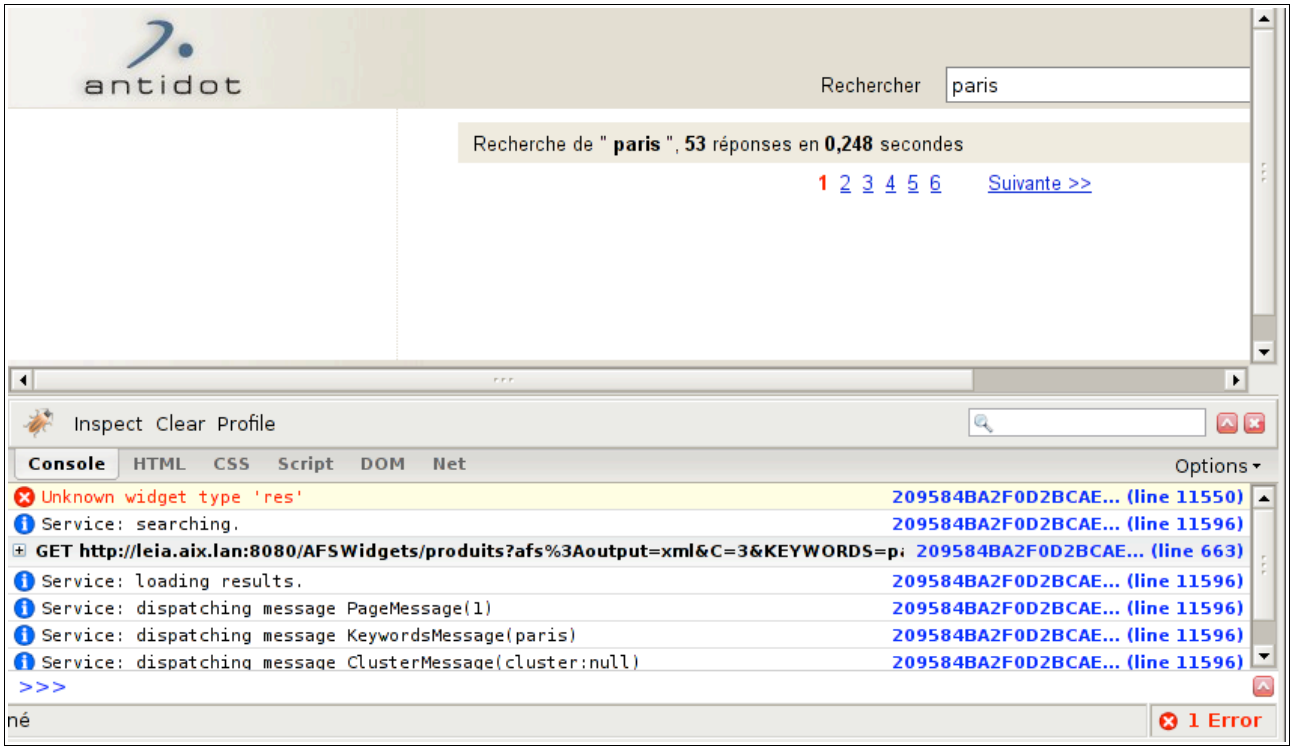
```
<meta name="awml:log_level" content="info" />
```

6.2.2. *FireBug*

FireBug offre de nombreuses fonctionnalités intéressantes pour un développeur Web : changement des styles CSS à la volée, exploration du DOM, console d'exécution/d'erreur de JavaScript, débogage de JavaScript, exploration de variables JavaScript... Il nécessite un navigateur de type Mozilla (tel que FireFox) et l'installation d'une extension

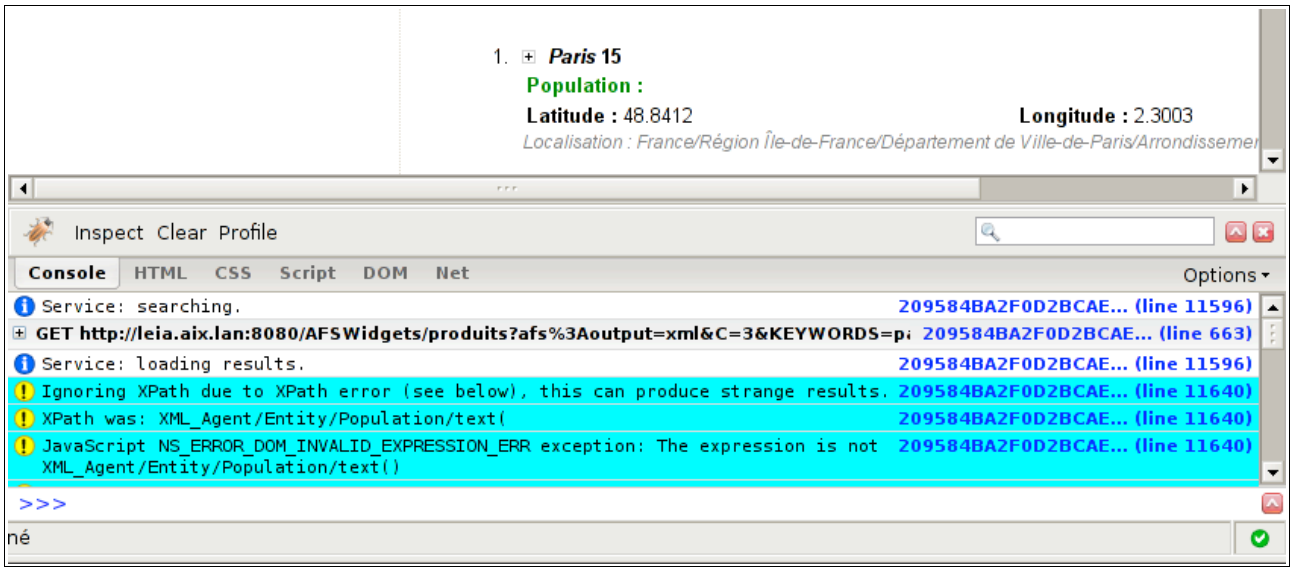
Il est disponible sur le site http://www.getfirebug.com/. Une fois installé, il est peut être lancé avec la touche « F12 », dans le menu « Outils » > « FireBug » ou via une icône affichée en bas à droite de la fenêtre.

Quand une erreur a lieu, un message en rouge apparaît. Dans l'exemple ci-dessous, l'attribut « type » d'une balise « awml:widget » n'est pas valide.



Quand un avertissement a lieu, il est écrit sur fond cyan.

Dans l'exemple ci-dessous, lié à l'utilisation du formateur « custom » (voir section 3.3.1), le XPath saisi pour sélectionner la valeur du champs « Population » dans le XML de la réponse est erroné. La première ligne de l'avertissement indique l'erreur, la seconde le XPath en cause et la troisième un message particulier au navigateur. Le XPath est alors ignoré et aucune « population » n'est affichée pour la réponse comme le montre le libellé vert de l'image.



6.2.3. *FireBug Lite*

FireBug Lite est une version allégée de FireBug ne reprenant que la console d'exécution/d'erreur JavaScript . Entièrement écrit en JavaScript, il peut être utilisé sur n'importe quel navigateur.

Il peut être téléchargé sur la page http://www.getfirebug.com/lite.html.

Il faut alors placer les fichiers de FireBug Lite sur le serveur Web et charger le fichier « firebug.js » dans la page HTML :

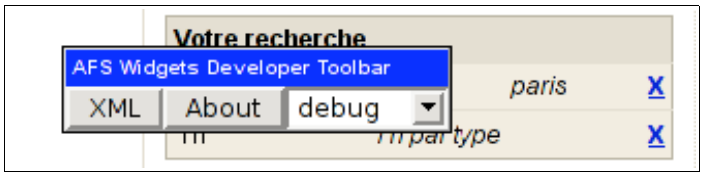
```
<script type="text/javascript" src="firebug.js"></script>
```

La page comporte alors un calque particulier contenant la console FireBug Lite. Elle peut être affichée en utilisant la touche « F12 ».

6.3. La barre d'outils du développeur

Les Widgets d'AFS incluent une barre d'outils à destination des développeurs pour faciliter l'intégration des Widgets. Cette barre d'outils est activée via l'option « toolbar » de la balise « awml:service » (doit être mise à « true »), puis affichée en appuyant sur les touches « Alt » et « F11 » simultanément.

La barre d'outils n'est pas prévue pour être utilisée en production. Il peut exister des défauts d'affichage sur certains navigateurs de même que l'affichage des autres widgets peut être légèrement modifié. C'est pourquoi l'option « toolbar » doit toujours être mise à « false » en production.



Cette barre d'outils propose 3 fonctionnalitées :

- D'afficher le flux XML retourné par le moteur ainsi que l'URL d'appel ;
- Des informations utiles pour les rapports de bogues (version, navigateur...) ;
- Paramètre le niveau de rapport dans la console d'erreur.

Le bouton « XML » ouvre une sous-fenêtre comme visible sur la capture suivante. Le flux XML est affiché nativement par le navigateur. L'affichage peut donc varier d'un navigateur à l'autre. La capture présente l'affichage du XML dans FireFox 2.

7. Foire aux questions

Le temps de réponse affiché par le Widget « info » ne correspond pas à la durée de la requête AJAX telle que donnée par mon outil de débogage Javascript (ex : FireBug).

Le temps affiché par le Widget « info » correspond au temps donné par le moteur de recherche. C'est le temps qui s'est écoulé entre le moment où le moteur a reçu la requête et le moment où il retourne les réponses. Tous les moteurs de recherche utilisent ce principe pour calculer la durée d'une recherche.

Pour l'internaute, la recherche peut sembler durer plus longtemps mais cela n'est pas imputable au moteur. La différence peut être due :

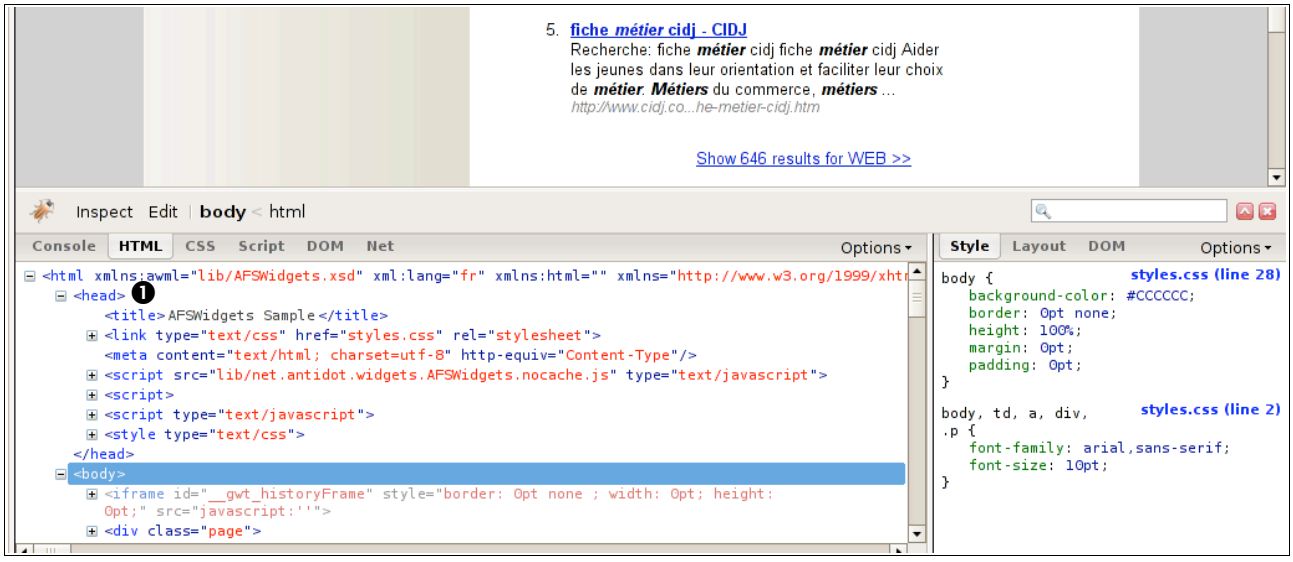
- Aux aléas du réseau : il peut y avoir un délai non négligeable entre le moment où le navigateur de l'internaute envoie une requête au moteur et le moment où celui-ci la reçoit et entre le moment où le moteur de recherche envoie la réponse au navigateur de l'internaute et celui où le navigateur reçoit réellement la réponse. Cela peut venir d'un très grand nombre d'ordinateurs entre le moteur de recherche et l'internaute et/ou une connexion lente et/ou un engorgement sur Internet...
- Au temps de traitement de la réponse par le navigateur : suivant la puissance et la charge de l'ordinateur de l'internaute, suivant les extensions activées sur son navigateur, suivant la présence de divers programmes de filtrage (pare-feu, antivirus...), le navigateur peut mettre plus ou moins de temps pour traiter la réponse et donc les widgets pour l'afficher.

Chaque délai individuellement peut paraître négligeable mais leur somme peut peser lourd dans la différence entre le temps affiché et le temps réel de réponse.

Je souhaite appliquer des styles CSS sur les Widgets mais j'aurais besoin de connaître le XHTML produit par les Widgets.

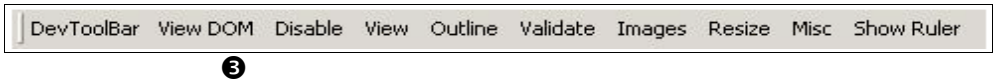
La meilleure façon d'obtenir le XHTML produit par les widgets est de regarder le code source de la page XHTML ***une fois*** les widgets en action. Pour ce faire, il est possible d'utiliser diverses extensions :

- FireBug pour Mozilla/Firefox : ouvrir FireBug, l'activer si ce n'est déjà fait, cliquer sur l'onglet « HTML » ❶, l'arborescence du DOM apparaît ;

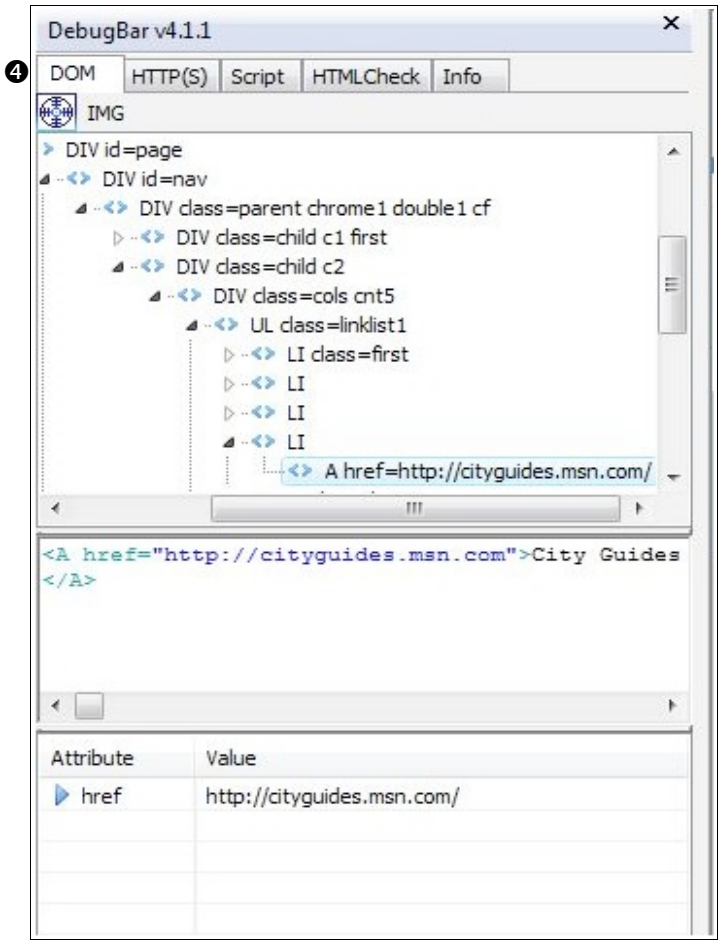


- Web Developer Toolbar pour Mozilla/Firefox : afficher la barre d'outils, cliquer sur « View Source » ❷, « View Generated Source », le code source XHTML de la page apparaît ;

- Microsoft Developer Toolbar sous Internet Explorer : afficher la barre d'outils, cliquer sur « View DOM » ❸, l'arborescence du DOM apparaît ;



- Debug Bar sous Internet Explorer : afficher le panneau Debug Bar, l'arborescence du DOM apparaît dans l'onglet « DOM » ❹. En choisissant le sous-élément « HTML » de « Document », le code source XHTML apparaît juste en dessous.



Je souhaite empêcher l'indexation de la page HTML, incluant les widgets, par Google pour éviter d'indexer le code des Widgets.

Google ne peut pas indexer le code des Widgets mais il peut indexer le texte au sein de la déclaration des widgets. Du coups la valeurs des options, libellés... peuvent être indexés. Il est préférable d'empêcher les moteurs de recherche d'indexer la page contenant les widgets. Pour ce faire il est possible d'ajouter la balise suivante dans la balise « head » de la page :

```
<meta name="ROBOTS" content="NOINDEX, NOFOLLOW" />
```

La page peut également être bloquée via le fichier « robots.txt ».

Je crois avoir rencontré un bogue, comment le rapporter à Antidot ?

Si vous avez un accès au Mantis Antidot, vous pouvez utiliser celui-ci pour nous informer de votre problème. Sinon, vous pouvez envoyer un courriel à support@antidot.net.

Dans tous les cas, plusieurs informations importantes doivent être fournis à Antidot :

- Une description claire de ce que vous souhaitez faire ou obtenir ;
- Une description claire de ce que vous obtenez réellement avec capture d'écran si possible. Lorsque le comportement rapporté est lié à plusieurs actions successives, il est important de lister les étapes pour y arriver ;
- La version des Widgets d'AFS, le système d'exploitation, le navigateur... pour ce faire, la barre d'outils du développeur propose un bouton « About » affichant toutes ces informations. Il suffit donc de copier/coller les informations ;
- Le code source HTML contenant les balises AWML ;
- Toute information qui vous semble importantes.

Le support d'Antidot vous répondra dans les meilleurs délais.

8. Lexique

AJAX (Asynchronous JavaScript And XML)	Méthode de développement d'applications Web combinant plusieurs langages (XHTML, CSS, JavaScript, XML, XHTML, CSS...). Cette méthode permet d'assurer des échanges d'informations rapides entre le navigateur et le serveur Web, en ne modifiant que les parties d'une page Web qui nécessitent d'être mises à jour, ce qui évite de recharger entièrement une page lors de la navigation.
AWML (AFS Widget Markup Language)	Dialecte XML, créé par Antidot, afin de proposer une manière simple et flexible de paramétrer les widgets dans une page XHTML.
CSS (Cascading Style Sheet)	Langage permettant de mettre en forme un site Web.
Doctype	Indique la version HTML ou XHTML utilisée.
GWT (Google Web Toolkit)	Plate-forme de développement Open Source d'application AJAX avec le langage de programmation Java. Sa particularité est de permettre de générer du code JavaScript et des requêtes AJAX à partir d'un code Java sans se soucier des problèmes d'interprétation du JavaScript par les différents navigateurs existants (Internet Explorer, Mozilla, Safari...)
Java	Langage de programmation orienté objet.
JavaScript	Langage de scripts qui peut être incorporé dans un document HTML afin de rendre les pages Web interactives. Ce langage permet d'associer des actions à des événements en réalisant, par exemple, des animations d'images au chargement d'une page ou en affichant un message lorsqu'une erreur se produit au cours du chargement de celle-ci.
KWIC (KeyWord In Context)	Le KWIC correspond au mot clé recherché et présent dans le titre et la description des résultats obtenus à l'issue d'une recherche. Ce mot clé est mis en évidence chaque fois qu'il est présent dans le titre ou la description des résultats.
RTE (Related Topic Expression)	Expression proche suggérée à l'internaute pour lui proposer une recherche transversale
XML (eXtensible Markup Language)	Langage de balisage permettant de représenter des documents et des données de manière structurée. Un document XML est considéré comme une arborescence de différents types de noeuds (élément, attribut, texte..)
XHTML (eXtensible HyperText Markup Language)	Langage de balisage permettant de concevoir des pages Web

XPath	Langage de requêtes permettant de rechercher et d'extraire des informations au sein d'un document XML
XSLT (eXtensible Stylesheet Language Transformations)	Langage de transformation d'un document XML vers un autre type de document (XHTML, HTML...). Le langage XSLT est un dialecte XML qui s'appuie sur XPath pour sélectionner des données au sein d'un document XML