clever age
conseil en architecture technique

Documentation AGORA

Erwan LE BESCOND
elebescond@clever-age.com
☐+33 1 49 01 28 63

03/11/2003

INTRODUCTION

AGORA propose un large enrichissement de la solution libre de gestion de contenu SPIP[1]. Les avantages de ce socle sont multiples.
Tout d'abord, SPIP est à présent une solution éprouvée, stable et très largement utilisée sur Internet. Son code est donc fiabilisé, très peu de bugs subsistants. De plus, SPIP est très bien sécurisé et rend donc AGORA plus résistant à la plupart des techniques d'attaque (hacking, exploit, …).
AGORA bénéficie donc nativement de toute l'expertise de la large communauté SPIP. Bien sûr SPIP ne peut couvrir l'ensemble des besoins recensés dans le cadre du projet AGORA, et donc certaines fonctionnalités ont été modifiées pour proposer un périmètre fonctionnel plus abouti. De plus, d'autres modules ont été adjoints à SPIP pour couvrir certains besoins qu'il ne gère pas nativement. La gestion des lettres d'information, l'utilisation de moteurs de recherche externes et la gestion des utilisateurs et des groupes d'utilisateurs en sont quelques exemples.
Qui plus est, la volonté de redistribuer AGORA sous forme de logiciel libre lui permettra d'accrocher une grande partie des utilisateurs de SPIP, trouvant qu'à l'heure actuelle certaine fonctionnalité lui font cruellement défaut.

PRINCIPE GENERAL DE SPIP:
Il est conseillé de lire le principe général de SPIP disponible à l'adresse suivante http://www.spip.net/fr_article877.html.

Cette documentation présentera dans un premier temps les nouveautés apportées par le projet AGORA. Ce chapitre permettra aux personnes connaissant SPIP de rapidement prendre connaissance des apports d'AGORA.
La seconde partie de ce document propose une présentation technique des différents modules d'AGORA afin d'apporter les renseignements nécessaire à toutes personnes souhaitant contribuer sur le projet.
La dernière partie est réservée au futur webmestre sous AGORA. Cette partie explique comment installer AGORA et comment créer ses squelettes.

1. Ce qu'apporte AGORA aux SPIPEURS

Souplesse de l'environnement technique

Contrairement à SPIP, AGORA est conçu pour fonctionner sur un grand nombre de plateforme différente. Cependant il est nécessaire que l'environnement technique utilisé propose au minimum un serveur WEB avec le module PHP4 ainsi qu'une base de données relationnelle respectant la norme SQL 92.

La conception modulaire d'AGORA permet d'ajouter le support de nouveaux environnements techniques sans avoir à toucher au code actuel.

Le principe et l'intérêt d'AGORA sont d'utiliser différentes abstractions pour chacun de ses modules. Ces différentes abstractions permettront de ne pas ajouter de code spécifique à SPIP lors des prochaines évolutions. Par exemple le support d'une nouvelle base de données, d'un nouveau moteur d'indexation, d'une nouvelle couche d'identification ne nécessitera pas de toucher au code actuel de SPIP mais d'écrire uniquement un driver spécifique. Le choix d'une l'une ou l'autre des briques techniques se paramètre dans un fichier de configuration.

Le tableau suivant récapitule les environnements techniques supportés **actuellement** par SPIP et AGORA.

	SPIP	**AGORA**
Serveur WEB		
• Apache	Oui	Oui
• IIS	Oui	Oui
Base de données		
• MySql	Oui	Oui
• PostGresql	Non	Oui
• SqlServer	Non	Oui
• Oracle	Non	Oui
Moteur de recherche		
• MnoGoSearch	Non	Oui
Serveur LDAP (authentification uniquement)	Oui	Oui

Les profils utilisateurs

Tout comme SPIP, AGORA fait une distinction entres les utilisateurs :

- Les auteurs accédant à la partie privée du site
- Les visiteurs du site

AGORA propose donc 1 profil visiteur et surtout 4 profils «Contributeurs» :

- Administrateur: Il détient tous les droits sur le site.
- Webmestre: Il détient l'ensemble des droits d'une partie du site.
- Rédacteur en chef: Il a en charge la validation des contenus proposés à la publication.
- Rédacteur: Il peut proposer un article à la validation.

Avec ces nouveaux statuts, AGORA propose une nouvelle définition des droits qui suivent les 2 règles suivantes:

- L'accès ou non à des parties du contenu se fait par l'attribution de rubriques à chaque utilisateurs.
- La liste des autorisations aux fonctionnalités est définie dans les tableaux suivants.

Droits sur les rubriques:

	Administrateur	Webmestre	Rédacteur en chef	Rédacteur
Créer	**oui**	**oui**	**oui**	**non**
Modifier	**oui**	**oui**	**oui**	**non**
Supprimer	**oui**	**oui**	**oui**	**non**
Affecter une rubrique à un auteur	**oui**	**oui**	**oui**	**non**
Désaffecter une rubrique	**oui**	**oui**	**oui**	**non**

Droits sur les articles :

	Administrateur	Webmestre	Rédacteur en chef	Rédacteur
Créer	**oui**	**oui**	**oui**	**non**
Modifier	**oui**	**oui**	**oui**	**non**
Supprimer	**oui**	**oui**	**non**	**non**
Proposer à la publication	**oui**	**oui**	**oui**	**non**
Valider un article	**oui**	**oui**	**oui**	**non**
Publier un article	**oui**	**oui**	**non**	**non**
Archiver un article	**oui**	**oui**	**non**	**non**
Refuser un article	**oui**	**oui**	**oui**	**non**
Mettre à la poubelle un article	**oui**	**oui**	**non**	**non**
Gérer auteur affecté à un article	**oui**	**oui**	**oui**	**non**
Gérer les mots clés affectés à un article	**oui**	**oui**	**oui**	**oui**

Droits sur les brèves :

	Administrateur	Webmestre	Rédacteur en chef	Rédacteur
Créer	**oui**	**oui**	**oui**	**oui**
Modifier	**oui**	**oui**	**oui**	**oui**
Supprimer	**oui**	**oui**	**non**	**non**
Publier	**oui**	**oui**	**oui**	**non**
Refuser	**oui**	**oui**	**oui**	**non**

Droits sur les auteurs :

	Administrateur	Webmestre	Rédacteur en chef	Rédacteur
Créer	**oui**	**oui**	**oui**	**non**
Modifier	**oui**	**oui**	**oui**	**non**
Supprimer	**oui**	**oui**	**oui**	**non**
Gérer les mots clés affectés aux auteurs	**oui**	**oui**	**oui**	**non**

En plus des règles fournies ci-dessus, la gestion des auteurs répond à la contrainte supplémentaire suivante :
Un auteur ayant moins ou autant de droits qu'un autre auteur ne peut effectuer des modifications sur celui-ci.

Droits sur les mots :

	Administrateur	Webmestre	Rédacteur en chef	Rédacteur
Créer un groupe de mots clés	**oui**	**oui**	**oui**	**non**
Créer un mot clé	**oui**	**oui**	**oui**	**non**
Modifier un groupe de mots clés	**oui**	**oui**	**oui**	**non**
Modifier un mot clé	**oui**	**oui**	**oui**	**non**
Supprimer un groupe de mots clés	**oui**	**oui**	**non**	**non**
Supprimer un mot clé	**oui**	**oui**	**non**	**non**

Droits sur les syndications :

	Administrateur	Webmestre	Rédacteur en chef	Rédacteur
Créer	**oui**	**oui**	**oui**	**oui**
Modifier	**oui**	**oui**	**oui**	**oui**
Supprimer	**oui**	**oui**	**non**	**non**

Droits sur les documents :

	Administrateur	Webmestre	Rédacteur en chef	Rédacteur
Créer	**oui**	**oui**	**oui**	**oui**
Modifier	**oui**	**oui**	**oui**	**oui**
Supprimer	**oui**	**oui**	**oui**	**oui**

Droits sur les signatures :

	Administrateur	Webmestre	Rédacteur en chef	Rédacteur
Créer	**oui**	**oui**	**oui**	**oui**
Modifier	**oui**	**oui**	**oui**	**oui**
Supprimer	**oui**	**oui**	**oui**	**oui**
Gérer une pétition	**oui**	**oui**	**non**	**non**

Le workflow

AGORA propose de nouvelles étapes dans la gestion des éléments principaux suivants :

Workflow des articles :

La gestion des articles se fait au travers des statuts suivants :

- **En cours de rédaction:** le ou les auteurs sont en train d'y travailler, il n'apparaît donc pas sur le site public, et son accès est limité sur le site privé.
- **Proposé à la publication :** lorsque l'auteur décide que son article est terminé, il le propose au comité de rédaction (les administrateurs et les autres rédacteurs) afin de décider s'il doit être publié ou non. L'article n'est toujours pas visible publiquement, mais tous les participants à l'espace privé peuvent le voir et son invités à le commenter dans un forum lié à cet article.
- **Validé :** le contenu de l'article a été validé et peut être publié sur le site public.
- **Publié :** l'article est publié sur le site public.
- **Refusé :** l'article n'est pas publié.
- **Archivé :** l'article est archivé.
- **Poubelle :** l'article est mis à la poubelle et sera supprimé automatiquement.

Seuls les articles publiés apparaissent dans le site public. Le statut archive permet d'archiver du contenu et d'y accéder dans la partie public du site en utilisant le critère **{archive}** dans la boucle **(ARTICLES)**.

Workflow des rubriques :

Comme dans SPIP seules les rubriques possédant au moins un élément prêt pour la publication est publié sur le site public. Par contre avec le nouveau statut archive pour les articles, les rubriques fonctionnent dans AGORA de la façon suivante :

Une rubrique ne contenant aucun élément à publier a son statut fixé à «privé ». Les rubriques privées n'apparaissent pas dans le site public.

Une rubrique contenant au moins un élément à publier voit son statut fixé à « publié ».

Une rubrique contenant que des éléments archivés est fixée au statut « archivé ». Il est possible d'accéder à ces rubriques en utilisant le critère **{archive}** dans la boucle **(RUBRIQUES)**.

Ces rubriques apparaissent dans la partie publique du site comme les rubriques avec le statut « publié ».

Une rubrique contenant des archives avec au moins un élément publié est quant à elle fixée au statut « pub_archivé ». Ces rubriques apparaissent dans la partie publique du site comme les rubriques avec le statut « publié ». Il est possible d'utiliser le critère {archives} pour parcourir ces rubriques au double statut.

Amélioration fonctionnelle du forum

Tout comme SPIP, AGORA propose le même module de forum intégré mais de nouvelle fonctionnalité ont été apporté au sein des forums publics.
Les forums de discussion sont toujours directement liés au contenu rédactionnel du site, c'est-à-dire qu'on peut ouvrir un forum indépendant pour chaque article, pour chaque rubrique et pour chaque brève.
Par défaut, les forums de AGORA sont modérés à posteriori. Cela signifie que chaque message envoyé par un utilisateur du site est immédiatement publié.

L'administrateur du site pourra décider de modifier ce comportement par défaut des forums. Il pourra choisir :

- L'absence totale de forums sur son site.
- Des forums modérés à priori : les contributions n'apparaissent qu'une fois validées par un administrateur.
- Des forums modérés à posteriori: les contributions sont automatiquement publiées.
- Des forums sur abonnement modéré à priori: les participants doivent auparavant s'inscrire. Les contributions n'apparaissent qu'une fois validées par un administrateur.
- Des forums sur abonnement modéré à posteriori: les participants doivent auparavant s'inscrire. Les contributions sont automatiquement publiées.

L'interface d'administration des forums publics a été complètement refaite pour faciliter l'administration des messages postés.
→ Une première page permet de visualiser l'ensemble des forums du site et de voir rapidement le nombre de message à valider dans le cas d'une gestion des forums « à priori ».
→ Une seconde page permet de visualiser avec possibilité de filtrage les différents messages postés pour un forum choisi. L'affichage peut se faire sous la forme d'une arborescence ou par ordre chronologique des messages.
→ Une troisième page permet de gérer un message choisi. Les actions possibles sur cette page sont les suivantes :

- Valider un message
- Mettre à la poubelle un message
- Refuser le message
- Répondre au message
- Editer le message (Lors de l'édition d'un message, un texte est ajouté à la fin du message afin d'indiquer qui a modifié ce message et à quelle date)

Les droits de gestion des forums répondent à la grille suivante :

	Administrateur	**Webmestre**	**Rédacteur en chef**	**Rédacteur**
Créer	**oui**	**oui**	**oui**	**oui**
Modifier	**oui**	**oui**	**non**	**non**
Supprimer	**oui**	**oui**	**non**	**non**
Publier	**oui**	**oui**	**oui**	**non**
Refuser	**oui**	**oui**	**oui**	**non**

Dans la partie publique du site, les fonctionnalités suivantes ont fait leur apparition :

- Choix pour l'auteur du message d'être averti d'éventuelles réponses.
- **{auteur}** Ce critère de la boucle **(FORUMS)** permet de n'afficher que les messages d'un auteur choisi.
- **#FORMULAIRE_RECHERCHE_FORUM** permet d'afficher un forum de recherche sur les messages du forum du contexte. Cette recherche d'effectue sur le sujet des messages, le contenu et l'auteur. Pour utiliser la recherche de message dans la boucle **(FORUMS)** il faut utiliser le critère **{recherche_forum}**
- Il est également possible de faire une sélection des messages par date. Il existe 2 critères définis pour la boucle **(FORUMS)** qui sont **{dateMin}** et **{dateMax}**.

{dateMin} retourne les messages dont la date de postage est supérieure à dateMin. Pour utiliser ce critère il est nécessaire de passer en paramètres HTTP les variables suivantes:

- anneeMin
- moisMin
- jourMin

{dateMax} retourne les messages dont la date de postage est inférieure à dateMax. Pour utiliser ce critère il est nécessaire de passer en paramètres HTTP les variables suivantes:

- anneeMax
- moisMax
- jourMax

L’arborescence de mots clés

La gestion avancée de mots clefs proposée par AGORA apporte une très grande souplesse pour simplifier leurs usages et obtenir facilement un comportement très riche.
Comme dans SPIP, les contributeurs peuvent créer des catégories de mots clefs, appelées groupes, et par la suite les remplir. Le paramétrage très complet des groupes de mots clés permet de les utiliser de différentes manières. En effet **AGORA apporte au sein de chaque groupe de mots clés une arborescence permettant d’obtenir une granularité plus ou moins fine.**

Une fois les mots clefs configurés par les administrateurs, les rédacteurs peuvent les sélectionner pour enrichir leurs contenus.
Dans AGORA les mots sont rattachables aux éléments suivants:

- Les articles
- Les brèves
- Les rubriques
- Les sites référencés ou syndiqués
- Les auteurs

Un mot clé permet d’influer notablement sur le comportement d’un élément auquel il est rattaché. Ainsi, le plus souvent, l’attrait majeur des mots clé est de permettre une gestion sur le comportement d’un contenu à la fois au niveau du fond, que de la forme

- **Les mots clés et le fond**

Le choix d’un mot clé dans un groupe de mot clé permet de contextualiser l’élément auquel un mot est rattaché. Par exemple, l'ajout de mots clés aux articles permet de regrouper des articles en catégorie. Une recherche par mot clé devient donc possible.

- **Les mots clés et la forme**

Le mécanisme de squelette d’AGORA ainsi que l’organisation des mots clés en groupe permet également d’utiliser les mots clés pour gérer la forme d’un article. Par exemple, il est tout à fait imaginable d’avoir un groupe de mots clé « Présentation » dans lequel on peut sélectionner des mots comme « colonne droite », « centre de la rubrique » ou encore « news ».

Les mots clés permettent d’agir avec une très grande liberté sur le comportement des éléments auxquels ils sont rattachés. Ils permettent donc d’éliminer de très nombreuses problématiques fonctionnelles, en terme de publication de contenu.
Ainsi, l’ensemble de l’arborescence de mots clefs disponible dans une instance d’AGORA est un outil indispensable mis à la disposition des rédacteurs et des webmestres.

Dans la partie publique du site, les fonctionnalités suivantes ont fait leur apparition :

- La boucle **(HIERARCHIE_MOT)**. Tout comme la boucle (HIERARCHIE) cette boucle effectue une récursivité. Cette boucle prend en critère {id_mot}. Elle renvoie donc la liste des mots clés depuis la racine jusqu’à cet identifiant id_mot.
- **{id_parent}** Ce critère de la boucle **(MOTS)** retourne la liste des mots clés fils sous le mot du contexte de la boucle.

- **{id_enfant}** Ce critère de la boucle **(MOTS)** retourne le mot père du mot courant la boucle.

La personnalisation du contenu

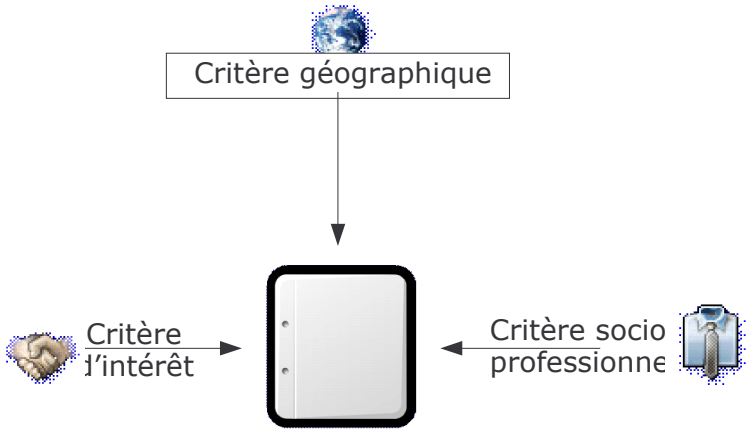
Nous venons de voir dans la partie précédente qu'AGORA propose une arborescence de mots clés mais permet également d'attacher des mots clés aux utilisateurs du site. Avec ces 2 principes AGORA offre à chaque utilisateur du site la possibilité d'avoir un espace personnalisé sur le site.

Chaque visiteur étant identifié par Login/mot de passe, il est possible de fournir une page présentant les derniers articles correspondant à son profil. Le visiteur ne recevra que les articles possédant au moins l'un des mots de son arborescence de mots.

Prenons par exemple une organisation du contenu concernant un article sur le projet de loi sur les retraites de médecins dans l'oise. L'article se voit associer 3 mots clés différents :

- France > Nord > Oise
- Profession libérale > Santé > Médecin
- Législatif > Projet de loi > retraites

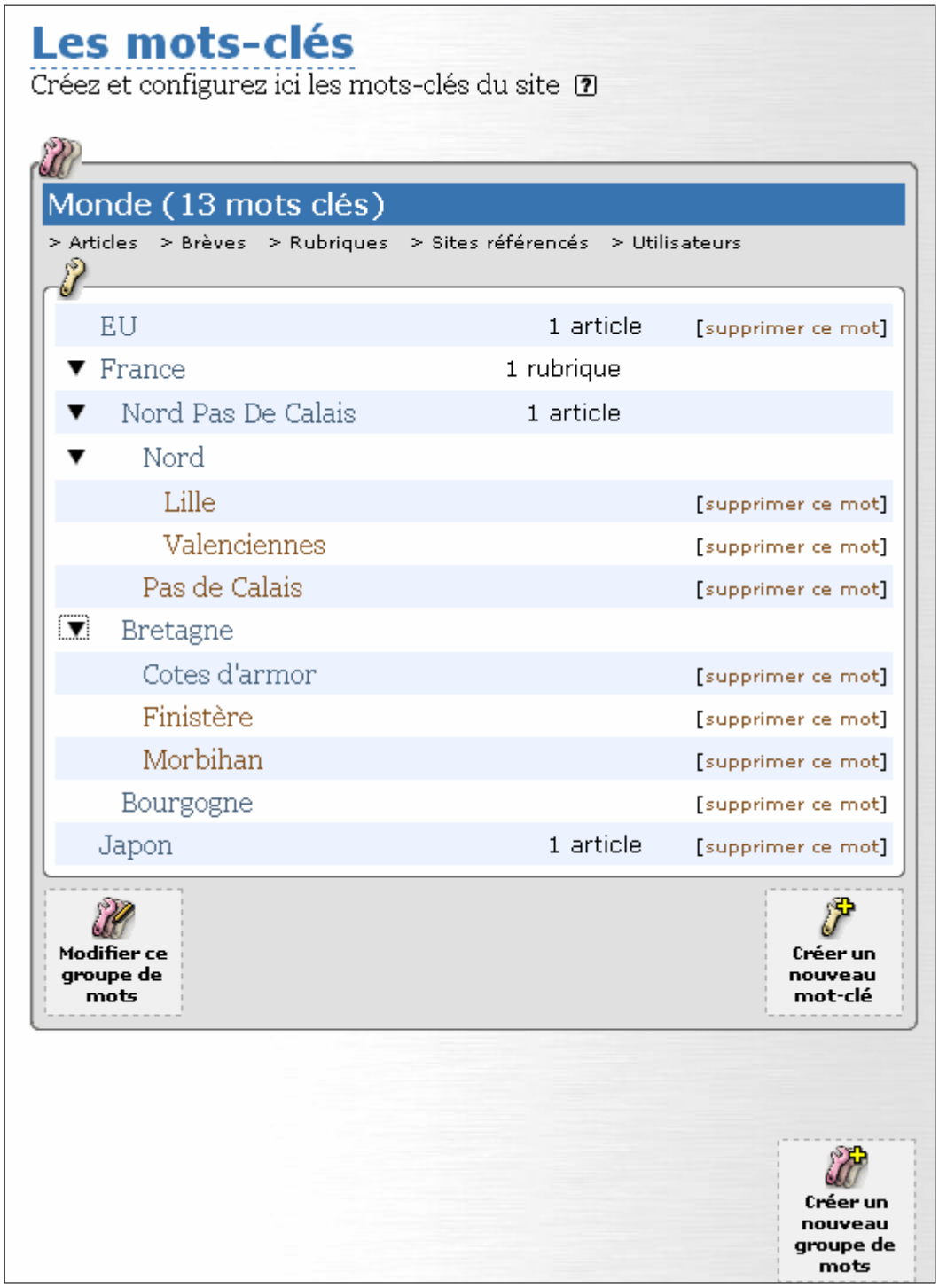
L'article est donc qualifié suivant 3 critères.

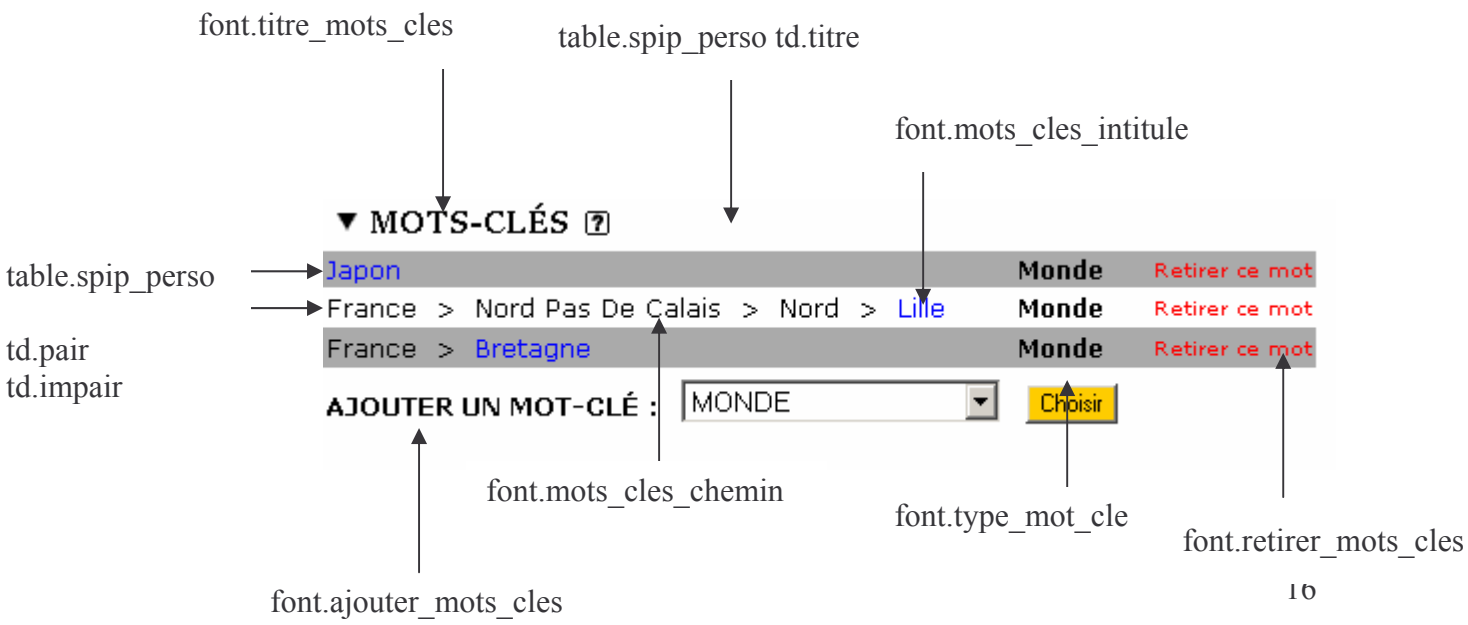


Dès lors on peut considérer que le profil visiteur le plus intéressé par cet article est un médecin exerçant dans l'Oise, intéressé par la retraite.

Le rendu personnalisé du contenu dans les squelettes s'obtient à l'aide du critère **{personnalisation}** ou **{personnalisation=id_groupe1, id_groupe2,…}** dans les boucles **(ARTICLES)**, **(RUBRIQUES)**, **(BREVES), {SITES}.** Si aucun identifiant de groupe n'est précisé avec le critère, l'ensemble des groupes de mots sera utilisé pour effectuer la personnalisation. Dans le cas contraire, seul les groupes de mots passés en paramètre du critère seront pris en compte pour gérer la personnalisation.

Le formulaire de personnalisation du front est accessible au travers du tag **#FORMULAIRE_PERSONNALISATION**. Ce tag affiche un formulaire qui permet de modifier son adresse mail et de choisir les mots clés attachés à son profil.
Le style de ce formulaire est paramétrable dans la feuille de style spip_style.css :


Les mots-clés
Créez et configurez ici les mots-clés du site
Monde (13 mots clés)
> Articles > Brèves > Rubriques > Sites référencés > Utilisateurs
EU 1 article [supprimer ce mot]
France 1 rubrique
Nord Pas De Calais 1 article
Nord
Lille [supprimer ce mot]
Valenciennes [supprimer ce mot]
Pas de Calais [supprimer ce mot]
Bretagne
Cotes d'armor [supprimer ce mot]
Finistère [supprimer ce mot]
Morbihan [supprimer ce mot]
Bourgogne [supprimer ce mot]
Japon 1 article [supprimer ce mot]
Modifier ce groupe de mots
Créer un nouveau mot-clé
Créer un nouveau groupe de mots


font.titre_mots_cles
table.spip_perso td.titre
font.mots_cles_intitule
MOTS-CLÉS
table.spip_perso
Japon Monde Retirer ce mot
France > Nord Pas De Calais > Nord > Lille Monde Retirer ce mot
td.pair
td.impair
France > Bretagne Monde Retirer ce mot
AJOUTER UN MOT-CLÉ :
MONDE
Choisir
font.mots_cles_chemin
font.type_mot_cle
font.retirer_mots_cles
font.ajouter_mots_cles

AGORA apporte une gestion de favoris au niveau du front. Cette fonctionnalité permet à tout visiteur de mémoriser une liste d'article, de brève et de rubrique.
Les tags fournis pour gérer cette fonctionnalité sont les suivants :
#GESTION_FAVORIS : ce tag est utilisable au sein des boucles ARTICLES, BREVES et RUBRIQUES. Il fournit un lien qui permet d'ajouter ou de retirer ce favori.

Un critère de boucle **{favoris}** utilisable dans les boucles ARTICLE, BREVES et RUBRIQUES permet de n'afficher que les éléments mis dans les favoris.

NB : L'utilisation des fonctionnalités de personnalisation nécessite la mise à 0 du CACHE.

Gestion avancée de lettre d'informations

AGORA propose un module de gestion de newsletter très puissant qui permet d'envoyer en quelques clicks une newsletter personnalisé à chacun des utilisateurs d'un site.
Ce module de newsletter est basé sur CLEVERMAIL (http://www.clever-age.org/rubrique.php3?id_rubrique=2) qui a été adapté à AGORA.

La gestion des newsletter dans la partie privé du site est la suivante :

- Création d'un modèle de newsletter avec son squelette « gabarit ».

La création d'une newsletter demande de renseigner les éléments indispensables suivants :

- Titre
- Descriptif rapide
- Type de modération (Non implémenté pour le moment dans cette version d'AGORA)
- Un titre de confirmation d'inscription
- Un texte de bienvenue lors de l'inscription : Dans ce texte de bienvenue, il est nécessaire de fournir le lien vers la confirmation d'inscription et le lien pour se désinscrire de la newsletter. Des tags spécifiques à CleverMail sont disponibles et paramétrables dans l'administration d'AGORA.
  - **#FORMAT# :** le format de réception des newsletters choisi lors de l'inscription.
  - **#ADDRESS# :** l'adresse mail de l'inscrit.
  - **#URL# :** l'url de validation d'inscription à la newsletter.
  - **#UNSUBSCRIBE# :** l'url de désinscription à la newsletter.
- Un titre de confirmation de demande de désinscription.
- Un texte de confirmation de demande de désinscription. Il est également possible d'utiliser les tags spécifiques à ce module.
- Un sujet par défaut à chaque instance de newsletter.
- Le nombre d'article à attacher à cette newsletter.
- La possibilité d'uploader un squelette générant un contenu au format HTML et un autre au format TEXTE.

Ensuite pour chaque gabarit de newsletter, il est possible de définir des groupes de destinataires en leur associant des mots clés afin de pouvoir gérer la personnalisation du contenu en fonction du destinataire même type de personnalisation vue dans la partie précédente.

- Création d'une instance de newsletter.

La création d'une instance de newsletter se fait en 3 étapes :

- Sujet
- Choix des articles à attacher à la newsletter.
- Organisation des articles et validation de la saisie

Pour chaque instance de newsletter, il est possible de désactiver certain groupe de destinataire défini par le modèle. Il est également possible de définir des groupes de destinataires spécifiques pour cette instance de newsletter. Il est également possible de saisir une liste d'adresse email sous format CSV.

- Envoie de la newsletter

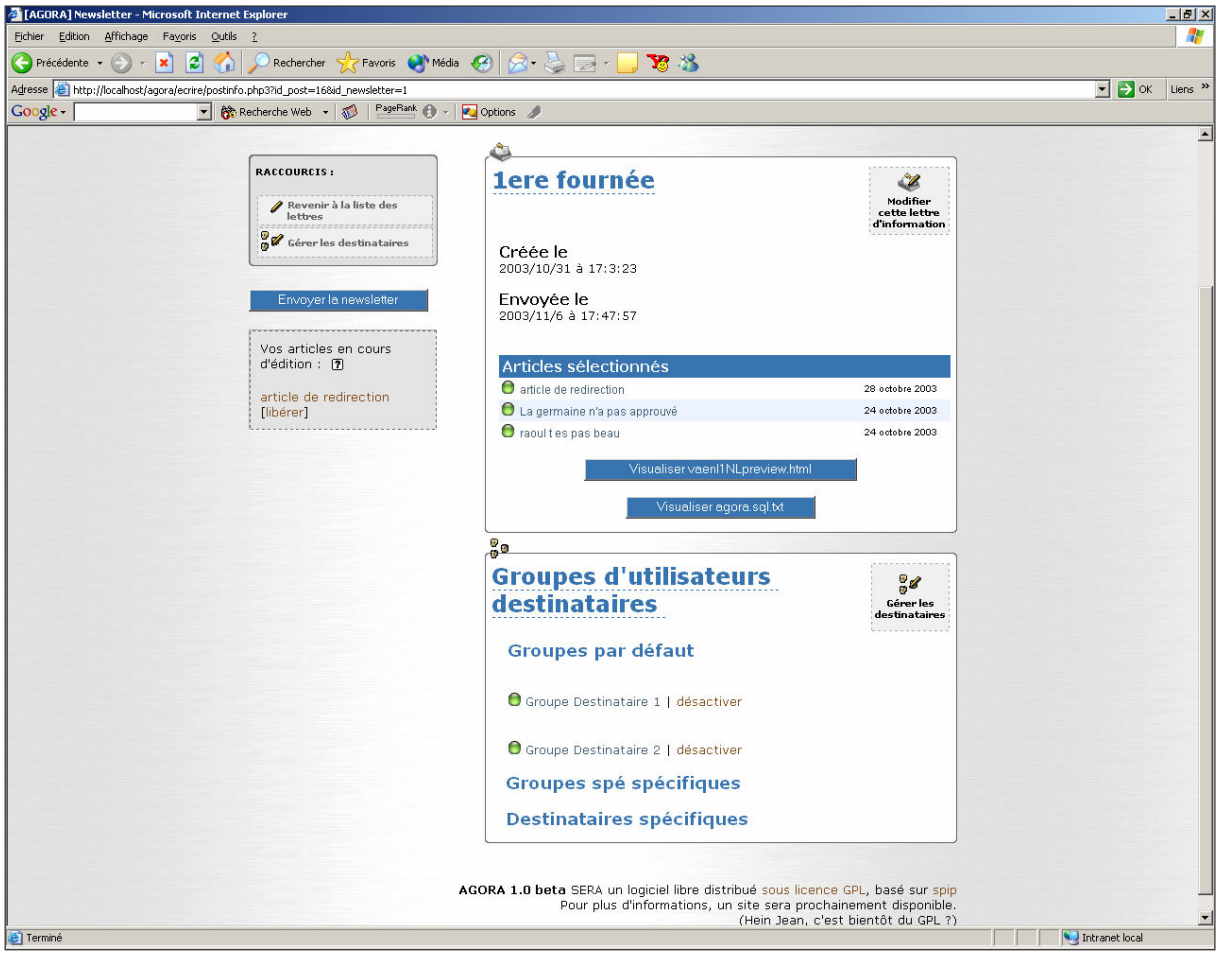
L'envoie de la newsletter est programmé au sein d'une tache CRON. Il faut exécuter la commande (php cm/queue_process.php) L'activation d'un envoi directement à partir de l'administration d'AGORA.

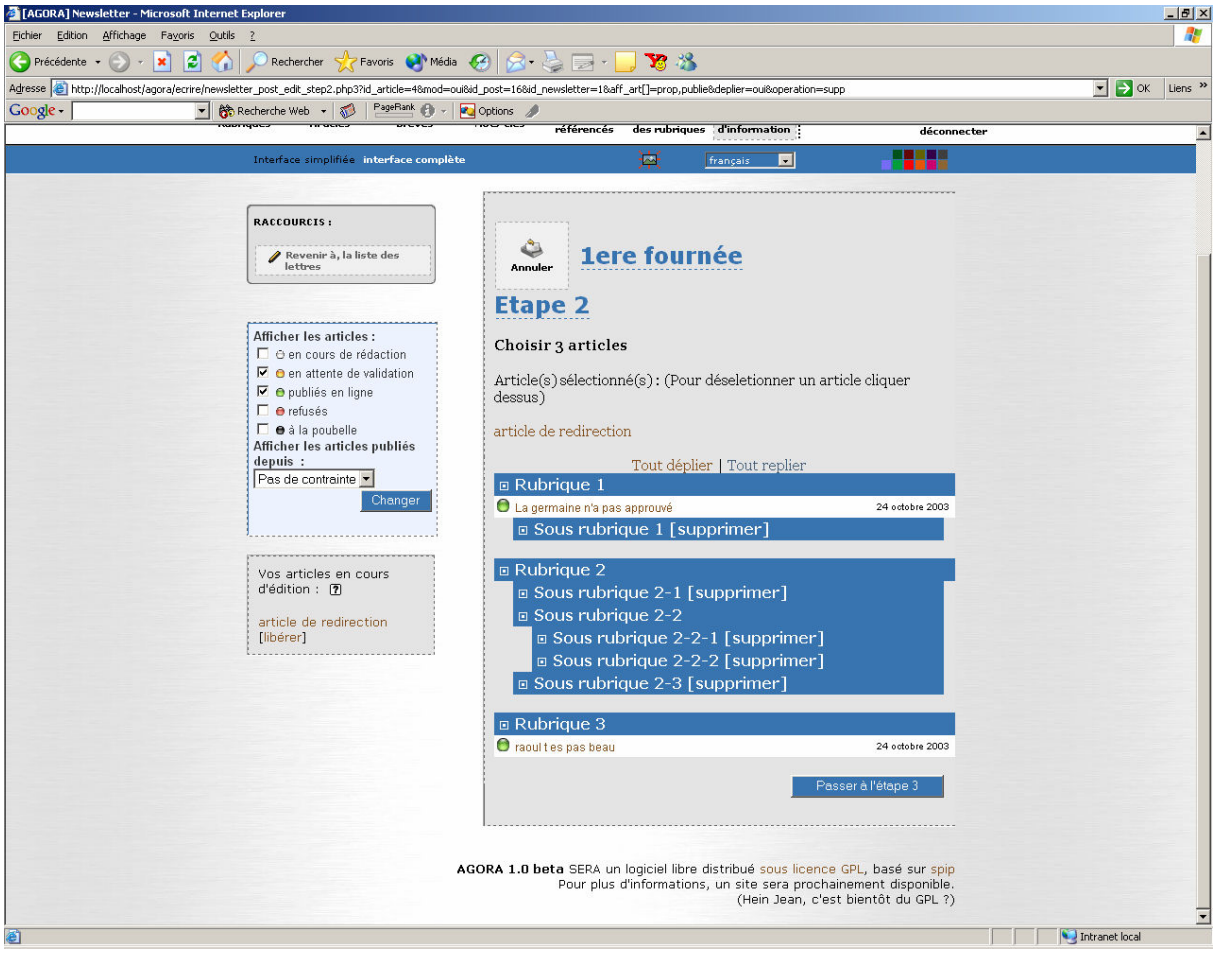
La gestion des newsletter dans la partie public se fait simplement à l'aide du tag **#FORMULAIRE_INSCRIPTION_NEWSLETTER**. Ce tag fabrique le formulaire d'inscription à une newsletter défini sur le site. Ce formulaire ne s'affiche pas si aucun gabarit de newsletter n'a été préalablement défini.

La boucle **(ARTICLES)** se voit doter également du critère **{id_post_newsletter}** pour renvoyer les articles liés à l'Id de l'instance de newsletter fournit par le contexte.

Le principe fonctionnel d'une inscription publique à la newsletter est le suivant :

- Inscription à une lettre d'information en saisissant son adresse électronique après avoir choisi son format de réception sur le formulaire en ligne.
- Réception d'un email demandant la confirmation d'inscription. Cette confirmation s'effectue en cliquant sur le lien fourni.
- Possibilité de modifier son format de réception en ressaisissant son adresse mail à partir du formulaire en front.
- Pour se désinscrire d'une newsletter, il suffit de cliquer sur le lien fourni par mail puis de confirmer cette désinscription (comme lors de l'inscription).




[AGORA] Newsletter - Microsoft Internet Explorer
Fichier Edition Affichage Favoris Outils ?
Précédente Rechercher Favoris Média
Adresse http://localhost/agora/ecrire/newsletter_post_edit_step2.php3?id_article=48&mod=oui&id_post=168id_newsletter=1&aff_art[]=prop,publie&deplier=oui&operation=supp
Google Recherche Web PageRank Options
références des rubriques d'information déconnecter
Interface simplifiée interface complète français
RACCOURCIS :
Revenir à, la liste des lettres
Afficher les articles :
en cours de rédaction
en attente de validation
publiés en ligne
refusés
à la poubelle
Afficher les articles publiés depuis :
Pas de contrainte
Changer
Vos articles en cours d'édition :
article de redirection
[libérer]
Annuler
1ere fournée
Etape 2
Choisir 3 articles
Article(s) sélectionné(s) : (Pour déseletionner un article cliquer dessus)
article de redirection
Tout déplier | Tout replier
Rubrique 1
La germaine n'a pas approuvé 24 octobre 2003
Sous rubrique 1 [supprimer]
Rubrique 2
Sous rubrique 2-1 [supprimer]
Sous rubrique 2-2
Sous rubrique 2-2-1 [supprimer]
Sous rubrique 2-2-2 [supprimer]
Sous rubrique 2-3 [supprimer]
Rubrique 3
raoul t es pas beau 24 octobre 2003
Passer à l'étape 3
AGORA 1.0 beta SERA un logiciel libre distribué sous licence GPL, basé sur spip
Pour plus d'informations, un site sera prochainement disponible.
(Hein Jean, c'est bientôt du GPL ?)
Intranet local

Nouveautés apportées au moteur de recherche : Seven Seas

AGORA propose le même moteur de recherche que SPIP mais peut intégrer n'importe quel moteur d'indexation.

Ce moteur de recherche est mis à disposition tant des visiteurs (pour rechercher dans le contenu du site) qu'aux contributeurs (recherche d'un contributeur, d'un visiteur, ou d'un contenu ...).
Cet outil est facilement accessible depuis n'importe quelle interface publique ou d'administration.
Ce moteur de recherche peut indexer des documents de différents types (HTML, XML, Doc, PDF, …).

Concernant l'indexation des documents Word, XML, Excel, etc., l'outil retenu est la brique logicielle libre Seven Seas avec le driver pour utiliser mnoGoSearch.

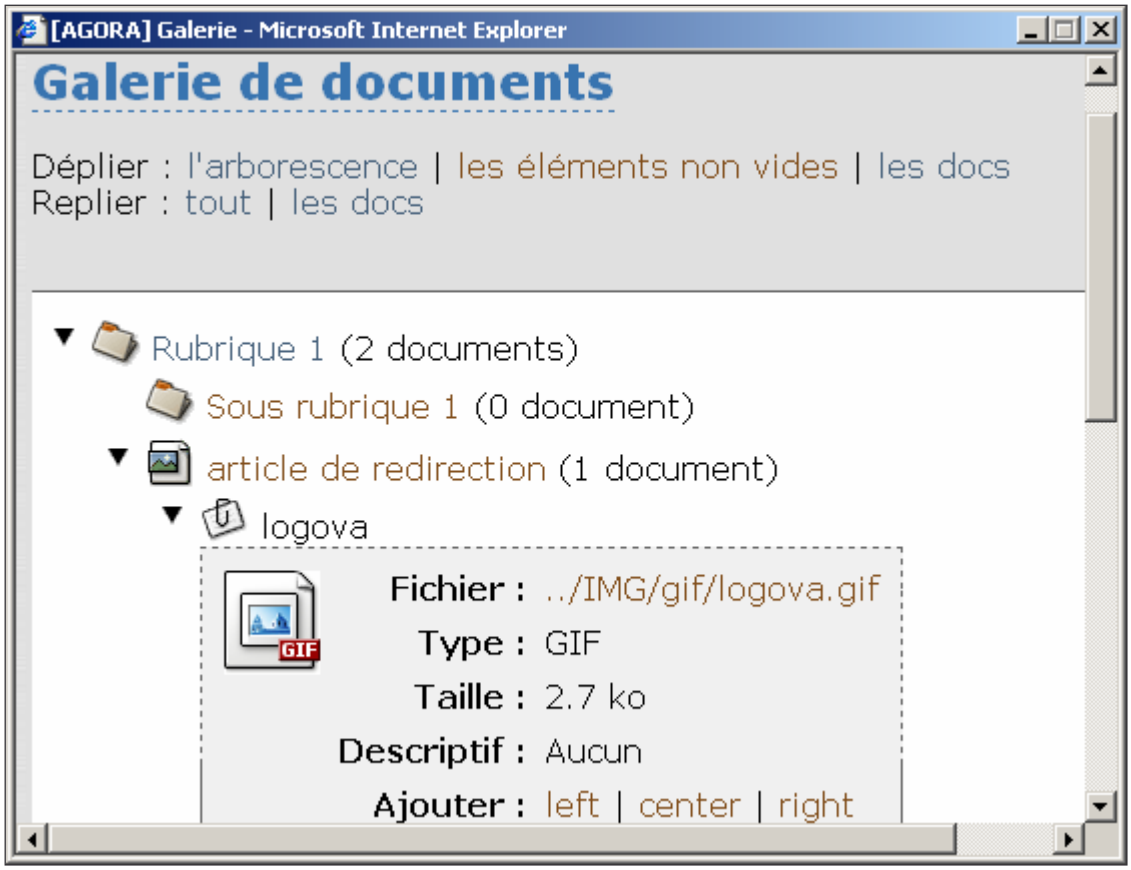
Seven Seas est une couche d'intégration de divers moteurs d'indexation, tels que Microsoft Index Server, MnoGoSearch, ht://dig ou encore Oracle Intermedia.

Un des multiples intérêts de Seven Seas est de permettre une consolidation des résultats d'un moteur existant déjà sur une application au moyen des résultats fournis par des moteurs d'indexation.

En effet SPIP propose nativement un moteur de recherche efficace sur les contenus qu'il gère (Articles, brèves, etc.). Par contre ce moteur n'est pas capable d'effectuer des recherches dans des documents bureautiques (Word, PDF, etc.) éventuellement rattachés à des contenus gérés par SPIP.

La galerie de documents

La galerie de documents permet de naviguer dans l'ensemble de l'arborescence de rubrique défini dans AGORA pour insérer un lien vers un document dans un contenu AGORA.



Gestion de sondage

Ce module permet d'attacher à chaque article un sondage.

Le paramétrage de ce sondage se fait dans la partie administration. Il est nécessaire de saisir au moins 2 réponses pour activer un sondage. Il est également possible de paramétrer l'affichage ou non des résultats d'un sondage dans la partie publique du site.

La boucle de gestion des sondages dans la partie publique du site permet l'utilisation des critères suivants :

#FORMULAIRE_SONDAGE fabrique le formulaire de réponse au sondage.
#TEXTE_REPONSE fournit le texte de la réponse.
#VOTES_NOMBRE fournit le nombre de vote pour une réponse.
#VOTES_TOTAL fournit le nombre de vote total.
#VOTES_POURCENTAGE fournit le pourcentage de réponse pour une question.

Le service Dernières visites

Ce module permet **d'afficher la liste des dernières pages visitées**, pour un utilisateur inscrit en tant qu'auteur.

L'utilisateur doit au préalable se connecter soit en tant que visiteur, soit en tant que rédacteur, webmestre ou administrateur… ce service est à l'origine prévu uniquement pour les membres du site internet. Les inscrits pourront aisément retrouver les 10 dernières pages qu'ils ont au préalable visitées.

Il est possible de configurer le nombre de page à mémoriser en changeant le méta **activer_derniere_page_vue**, configuré à 10 par défaut.

Au niveau du front le tag **#DERNIERES_VISITES** permettra d'afficher la liste des dernières pages visitées. Ces pages s'affichent sous la forme d'une liste, stylée selon la classe '**spip_list_dernier_document'**. l'intitulé des pages est le texte fournit par la balise <title></title> de celle-ci. Si celui ci manque on fixe le titre avec la chaîne de langue **'page_sans_titre'**.

Au niveau du modèle de données la table, spip_dernieres_visites a été ajoutée:

id_pers int
date varchar(19)
url varchar(200)
titre varchar(100)

Les raccourcis typographiques

Fonctionnalités plus complètes

Les raccourcis qui suivent offrent des fonctionnalités plus puissantes et d'un usage plus spécifique. Si cela est votre premier contact avec les raccourcis de SPIP, il est sans doute inutile que vous tentiez de les apprendre par cœur dès maintenant. Il vous suffit de savoir qu'ils existent ; lorsque vous en aurez réellement besoin, revenez sur cette page, il sera alors beaucoup plus facile pour vous de mémoriser des raccourcis dont vous avez réellement l'utilité.

Tableaux

Pour réaliser des tableaux très simples dans SPIP, il suffit de faire des lignes dont les « cases » sont séparées par le symbole « | » (pipe, un trait vertical), lignes commençant et se terminant par des traits verticaux. Il est impératif de laisser des lignes vides avant et après ce tableau.

Par exemple, **le tableau** :

Nom	**Prénom**	**Age**
Marso	Ben	23 ans
Capitaine	non	connu
Philant	Philippe	46 ans
Cadoc	Bébé	4 mois

se code ainsi :

```
| {{Nom}} | {{Pr&eacute;nom}} | {{Age}} |
| Marso | Ben | 23 ans |
| Capitaine | | non connu |
| Philant | Philippe | 46 ans |
| Cadoc | B&eacute;b&eacute; | 4 mois |
```

Remarquez que toutes les entrées de la première ligne sont placées en gras. SPIP identifie ainsi qu'il s'agit d'une page d'entête, et lui attribue une présentation différente des autres lignes (fond de couleur différente). La présence d'une telle ligne n'est pas obligatoire.

Il est également possible de renseigner le titre de tableau (ceci afin de maximiser l'accessibilité à ce dernier). Pour cela, il faut inscrire cette information entre des doubles pipes ("||") avant le tableau.

Par exemple, si **notre tableau décrit la liste des passagers d'un bateau**, il faut écrire :

```
||Passagers à bord||
| {{Nom}} | {{Pr&eacute;nom}} | {{Age}} |
| Marso | Ben | 23 ans |
| Capitaine | | non connu |
| Philant | Philippe | 46 ans |
| Cadoc | B&eacute;b&eacute; | 4 mois |
```

ce qui produira :

Passagers à bord

Nom	**Prénom**	**Age**
Marso	Ben	23 ans
Capitaine	non	connu
Philant	Philippe	46 ans
Cadoc	Bébé	4 mois

Spécifier les titres des liens

Il est possible de spécifier un titre particulier pour un lien.

Voici le code à utiliser :
voici un **[lien|merci de visiter ce site->http://agora.gouv.fr]**

Ceci produira le texte suivant :
→ voici un lien

Utilisation des liens internes (ancres)

Il est possible d'utiliser des liens interne afin de naviguer à l'intérieur d'un article long. Il faut, tout d'abord placer un repère, puis faire un lien vers ce repère (par exemple, au début de l'article : "aller aux conclusions de cet article").

Voici le code à utiliser pour créer un repère : [repère<-]

L'insertion de ce code ne produira rien de visible lors de la visualisation de l'article. Mais on peut maintenant revenir à ce repère en insérant un lien comme ceci :

[retour au repère->#repere]

Ceci produira le texte suivant :

→ retour au repère

Ouvrir des liens dans une nouvelle fenêtre

L'ajout d'une seconde flèche dans le raccourci du lien, provoquera l'ouverture de celui ci dans une nouvelle fenêtre du navigateur :

Note : ce raccourci est compatible avec tous les autres
Par exemple :
si vous êtes perdu, cliquez sur ce [lien->>http://www.perdu.com].
produira :

→ si vous êtes perdu, cliquez sur ce lien. (S'ouvre dans une nouvelle fenêtre vers www.perdu.com)

Acronymes

Un acronyme est un sigle, un ensemble d'initiales servant d'abréviation, qui se prononce comme un mot normal. Il est possible d'inseré un acronyme en renseignant sur sa définition.

Voici le code à utiliser :
Ce site fonctionne sous [AGORA|Texte de l'acronyme]

Ceci produira le texte suivant :
→ Ce site fonctionne sous AGORA
(si vous passez votre souris sur AGORA, le texte de l'acronyme apparaît comme le ALT d'une image Html)

Les champs Extras

Pour utiliser les champs "extra", il faut installer dans le fichier ecrire/mes_options.php3 un tableau définissant les champs en question, pour chaque type d'objet (article, rubrique, brève, auteur ou mot) que l'on veut ainsi étendre ;

Les champs EXTRA dans l'espace public s'utilisent avec la syntaxe suivante:

```
[(#EXTRA|extra{"nom_du_champ"})]
```

Définition de tous les extras possibles

```
$GLOBALS['champs_extra'] = Array (
        'auteurs' => Array (
                        "sexe" => "ligne|brut",
                        "age" => "ligne|propre|&Acirc;ge du capitaine",
                        "biblio" => "bloc|propre|Bibliographie"
                ),

        'articles' => Array (
                        "prix" => "ligne|typo|PRIX",
                        "isbn" => "ligne|typo|ISBN"
                )
        );
```

On peut optionnellement vouloir affiner les extras :
- pour les articles/rubriques/breves en fonction du secteur ou de la rubrique ;
- pour les auteurs en fonction du statut
- pour les mots-clés en fonction du groupe de mots

L'héritage des champs EXTRA n'est possible qu'à partir d'une définition par secteur. Il est par contre possible de redéfinir pour une rubrique donnée des extras particulier. Dans ce cas cette rubrique n'hérite plus des extras définis par le secteur

```
$GLOBALS['champs_extra_proposes'] = Array (
        'auteurs' => Array (
                // tous : par defaut
                'tous' =>  'age|sexe',
                // une biblio pour les admin (statut='0minirezo')
                '0minirezo' => 'age|sexe|biblio'
                ),
```

```
        'articles' => Array (
                // tous : par defaut
                'tous' => '',
                // 1 : id_secteur=1;
                1 => 'prix',
                // 4 : id_rubrique=4;
                4 => 'isbn'
                )
        );

*/
```

Comment faire de la perso dans le back-office?

La personnalisation du back-office se limite Ã pouvoir gérer des champs EXTRA (voir doc. correspondante) et Ã personnaliser certaines chaînes de langue.

Les chaînes de langue personnalisables sont modifiables dans le fichier agora-0_fr.php3 situation dans ecrire/lang/

Ces chaînes de langue correspondent aux textes des pages suivantes:

articles_edit.php3
articles.php3
breve_edit.php3
breve_voir.php3
naviguer.php3
rubrique_edit.php3

Il est également possible de personnaliser les textes par rubrique. Cette personnalisation nécessite de créer un nouveau fichier de langue avec un nom précis.

Ex:
- Pour personnaliser les chaînes de langue en français pour la rubrique 19 et toutes ces sous rubriques il suffit de créer un fichier nommé :
agora-19_fr.php3

- Pour personnaliser les chaînes de langue en espagnol pour la rubrique 5 sans que les sous rubriques en héritent il suffit de créer un fichier nommé :
agora=5_en.php3

NB: Pour pouvoir gérer des chaînes de langue d'autres fichiers, il faut modifier l'appel Ã la fonction _T() en lui passant en parametre la rubrique courante.

Ex:

```
// Nouveau code
_T('art_info_nouvel_article', null, $id_rubrique)

// Ancien code
_T('info_nouvel_article')
```

Contribuer au projet AGORA: Présentation technique

La nécessité d'une couche d'abstraction

Comme nous l'avons vu dans la partie précédente, l'utilisation d'un environnement technique nécessite l'écriture de code spécifique pour son utilisation dans le reste de l'application.

Les conclusions que l'on peut tirer de ce constat sont assez évidentes :

- L'architecture logicielle est rigidifiée par le développement spécifique :
  - Problème de « scalability ». Par exemple, passer de l'utilisation machine commune au serveur d'applications et au moteur d'indexation à l'utilisation d'un serveur dédié à l'indexation remet souvent en cause le code spécifique.
  - Difficulté pour changer d'environnement technique en cas de besoin.

- Le temps de développement de l'application est rallongé par la nécessité de réécrire systématiquement du code spécifique.

Le code produit nécessite systématiquement une phase de correction de bugs, dont la durée est non négligeable.

Pour répondre à cette problématique, l'architecture d'AGORA fait appel à différente couche d'abstraction (abstraction de base de données, abstraction de moteur de recherche, abstraction du moteur d'autorisation …).

Convention

Avant de s'attaquer à AGORA, il est nécessaire dans un premier temps de savoir qu'AGORA a le désir de suivre l'évolution de SPIP. C'est-à-dire que chaque nouvelle fonctionnalité apportée par SPIP doit pouvoir être intégrée rapidement dans AGORA en faisant appel aux différentes classes métiers. Pour satisfaire à cette contrainte, le principe suivant a été adopté :
Toute modification d'un code SPIP par du code AGORA doit être marquée de la façon suivante :

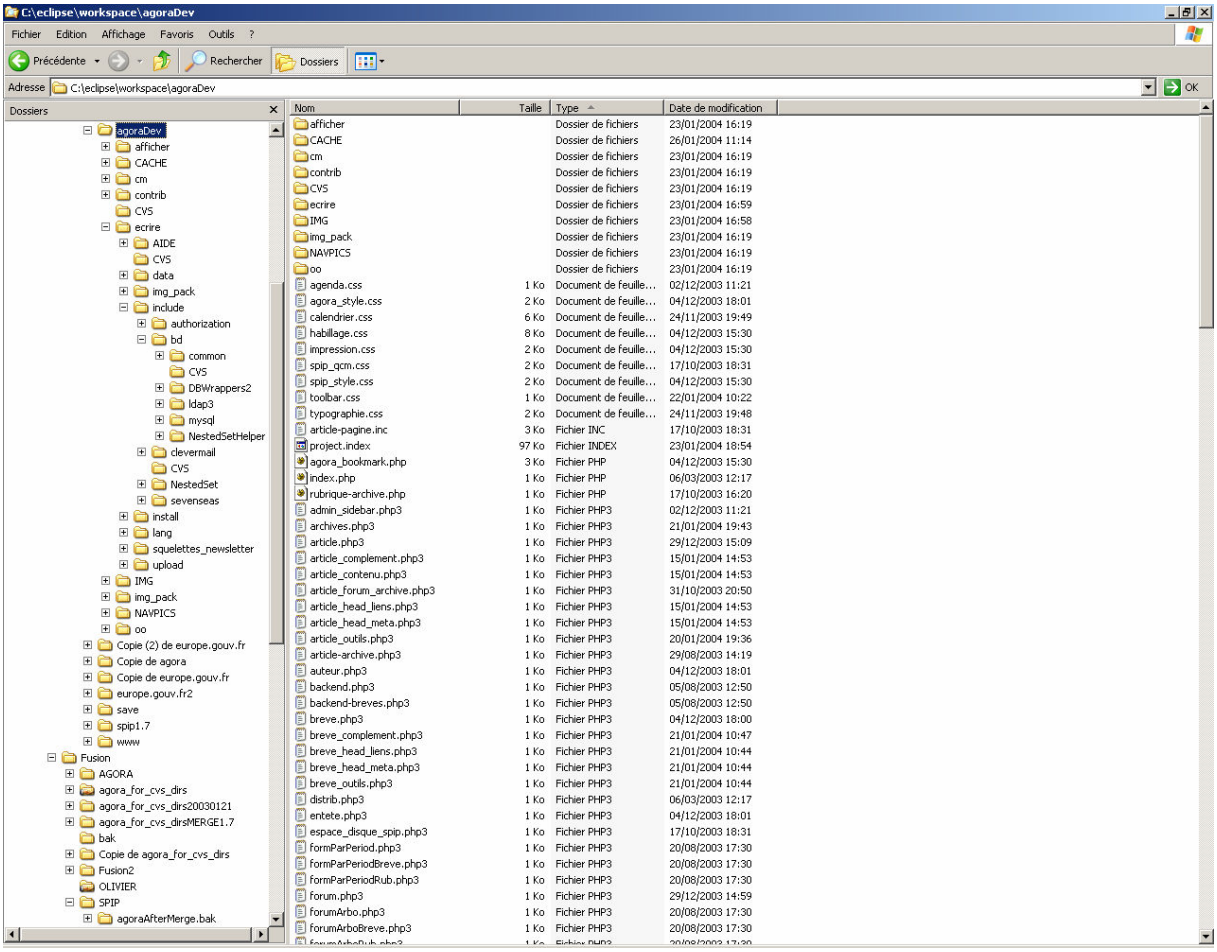
```
//********** Modification elebescond@clever-age.com **************************
LE NOUVEAU CODE AGORA
/*
L'ANCIEN CODE DE SPIP
*/
//************************** Fin Modification ***************************************
```

En gardant en commentaire le code original de SPIP, la procédure d'UPDATE CVS depuis le repository de SPIP permettra de fusionner facilement les sources des 2 projets.

La convention de codage du projet AGORA est la même que celle utilisé dans le projet PEAR. http://pear.php.net/manual/fr/standards.php
La principale caractéristique à suivre est d'utiliser une indentation de 4 espaces, sans tabulation.

Présentation de l'arborescence de fichiers d'AGORA

L'arborescence d'AGORA est la même que celle de SPIP avec de nouveaux répertoires.



Voici un descriptif des répertoires principaux d'AGORA:

/afficher : Ce répertoire contient les squelettes de base d'uns distribution d'AGORA.

/cm : Ce répertoire contient la base de clevermail utilisé par le module de newsletter d'AGORA

/ecrire: Ce répertoire contient le back office d'AGORA

/ecrire/include/authorization : Ce répertoire contient le module de gestion des droits utilisateur d'AGORA.

/ecrire/include/bd : Ce répertoire contient des classes métier d'AGORA

/ecrire/include/bd/common : Ce répertoire correspond au drivers standard utilisé par n'importe quelle base de données respectant la norme SQL 92.

/ecrire/include/bd/mysql : Ce répertoire correspond à une implémentation spécifique des classes métiers d'AGORA pour MySql.

/ecrire/include/bd/DBWrapper2 : Ce répertoire regroupe un ensemble de classe utilisé lors du parsing des squelettes.

/ecrire/include/bd/ldap3 : Ce répertoire fournit une implémentation des méthodes métiers de gestion des auteurs stockés sous LDAP.

/ecrire/include/bd/NestedSetHelper : Ce répertoire fournit une classe d'aide à la gestion de l'arborescence de mot clé d'AGORA.

/ecrire/include/NestedSet : ce répertoire contient une réimplémentation du package PEAR::DB_NestedSet afin de fournir une gestion de l'arborescence de mot clé respectant la norme SQL 92.

/ecrire/include/sevenseas : ce répertoire contient l'abstraction de moteur d'indexation externe d'AGORA.

/ecrire/lang : ce répertoire gère le multilinguisme d'AGORA. (Seule la langue française est définie pour le moment).

/ecrire/squelettes_newsletter : Ce répertoire stocke les squelettes uploadés pour les différentes newsletters définies dans votre AGORA.

/sql : Ce répertoire regroupe la définition du modèle de données pour les différentes bases de données supportées par AGORA.

Le modèle de données

→ spip_actions
Définition des droits nécessaires pour accéder à une action.

Clé primaire action

→ spip_articles
Stockage des articles.

Clé primaire id_article.
id_secteur (ref rubriques.id_rubrique) indique le secteur correspondant à la rubrique susmentionnée.
id_trad (ref articles.id_article) l'article traduit.

→ spip_auteurs
Stockage des auteurs.

Clé primaire id_auteur.

→ spip_auteurs_articles
Liaison entres les articles et leurs auteurs.
Id_auteur (ref auteurs.id_auteur).
Id_article (ref articles.id_article).

→ spip_auteurs_messages
Liaison entres les messages et leurs auteurs.

Id_auteur (ref auteurs.id_auteur).
Id_message (ref messages.id_message).

→ spip_auteurs_rubriques
Définit les rubriques accessibles par les auteurs.

Id_auteur (ref auteurs.id_auteur).
Id_rubrique (ref rubriques.id_rubrique).

→ spip_bookmarks
Stockage des favoris.

Clé primaire (id_auteur, id_objet, type_bookmark)
Id_auteur (ref auteurs.id_auteur).

→ spip_breves
Stockage des brèves.

Clé primaire id_breve
Id_rubrique (ref rubriques.id_rubrique)

→ spip_cm_lists
Stockage des gabarits de newsletters.

Clé primaire lst_id.

→ spip_cm_lists_subscribers
Liaison des inscriptions publiques aux newsletters.

Clé primaire (lst_id, sud_id).
Lst_id (ref spip_cm_lists.lst_id).
Sub_id (ref spip_cm_subscriber.sub_id).

→ spip_cm_pending
Stockage des actions utilisateurs à la newsletter (confirmation des inscriptions/désinscriptions).

Clé primaire (lst_id, sub_id).
Lst_id (ref spip_cm_list.lst_id).
Sub_id (ref spip_cm_subscriber.sub_id).

→ spip_cm_posts
Stockages des instances de newsletter.

Clé primaire (pst_id).
Lst_id (ref spip_cm_list.lst_id).

→ spip_cm_posts_articles
Liaison des articles aux instances de newsletter.

Article_id (ref articles.id_article).
Pst_id (ref spip_cm_post.pst_id).

→ spip_cm_posts_done
Stockage des envoies effectués

Clé primaire (pst_id, email).

Pst_id (ref spip_cm_post.pst_id).

→ spip_cm_posts_links
Non utilise (Voir TODOLIST).

→ spip_cm_posts_queued
Gestion de la file des envoies.

Clé primaire (pst_id, email).
Pst_id (ref spip_cm_post.pst_id).

→ **spip_cm_subscribers**
Stockage des inscriptions.

Clé primaire sub_id.

→ **spip_cm_user_groups**
Stockage des groupes d'utilisateurs.

Clé primaire id_groupe.
Lst_id (ref spip_cm_list.lst_id).
Pst_id (ref spip_cm_post.pst_id).

→ **spip_documents**
Stockage des documents uploadés.

Clé primaire id_document.
id_type (ref spip_type_document.id_type).

→ **spip_documents_articles**
Liaison entres les documents et les articles.

id_document (ref spip_documents.id_document).
id_article (ref spip_articles.id_article).

→ **spip_documents_breves**
Liaison entres les documents et les brèves.

id_document (ref spip_documents.id_document).
id_breve (ref spip_breves.id_breve).

→ **spip_documents_rubriques**
Liaison entres les documents et les rubriques/

id_document (ref spip_documents.id_document).
id_rubrique (ref spip_articles.id_rubrique).

→ **spip_forum**
Stockage des messages des forums

Clé primaire id_forum.

Id_parent (ref spip_forum.id_forum).
Id_rubrique (ref spip_rubriques.id_rubrique).
Id_article (ref spip_articles.id_article).
Id_breve (ref spip_breves.id_breve).
Id_syndic (ref spip_syndics.id_syndic).

→ **spip_forum_cache.**

Utilisé afin d'adapter le système de cache à l'instantanéité des forums. Pour chaque fichier du cache ayant donné lieu à une requête sur la table spip_forum, la table spip_forum_cache stocke les paramètres de la requête (article, rubrique, brève et éventuel message parent du forum). Lorsqu'un message est posté, les paramètres du message sont comparés avec ceux présents dans spip_forum_cache, et pour chaque correspondance trouvée le fichier cache indiqué dans la table est effacé. Ainsi les messages n'attendent pas le recalcul régulier de la page dans laquelle ils s'insèrent pour apparaître dans l'espace public.

Clé_primaire (id_forum, id_rubrique, id_article, id_breve, id_syndic, fichier)
Id_forum (ref spip_forum.id_forum).
Id_rubrique (ref spip_rubriques.id_rubrique).
Id_article (ref spip_articles.id_article).
Id_breve (ref spip_breves.id_breve).
Id_syndic (ref spip_syndics.id_syndic).

→ spip_groupes_mots.
Stockage des groupes de mots clés

Clé primaire id_groupe.

→ spip_index_articles.
Indexation des articles

hash (ref spip_index_dico.hash).
Id_article (ref spip_articles.id_article).

→ spip_index_auteurs.
Indexation des auteurs.

hash (ref spip_index_dico.hash).
Id_auteur (ref spip_auteurs.id_auteur).

→ spip_index_breves.
Indexation des brèves.

hash (ref spip_index_dico.hash).
Id_breve (ref spip_breves.id_breve).

→ spip_index_dico.
Le dictionnaire d'indexation.

Clé primaire dico.

→ spip_index_mots.
Indexation des mots clés.

hash (ref spip_index_dico.hash).
Id_mot (ref spip_mots.id_mot).

→ **spip_index_rubriques.**
Indexation des rubriques.

hash (ref spip_index_dico.hash).
Id_rubrique (ref spip_rubriques.id_rubrique).

→ **spip_index_syndic.**
Indexation des sites.

hash (ref spip_index_dico.hash).
Id_syndic (ref spip_syndics.id_syndic).

→ **spip_liens.**
Stockage des liens entres les articles

Clé primaire (contenant, lien_interne, lien_externe).

→ **spip_lock_mots.**
Locks pour utilisation des NestedSet sur les mots clés.

Clé primaire lockId.

→ **spip_messages.**
Stockage des messages.

Clé primaire id_message.

Id_auteur (ref spip_auteurs.id_auteur).

→ **spip_meta.**
Stockage des valeurs de configuration de l'instance d'Agora.

Clé primaire nom.

→ **spip_mots.**
Stockage des mots clés.

Clé primaire id_mot.
Id_groupe (spip_groupe_mot.id_groupe).

→ **spip_mots_articles.**
Liaison des mots clés aux articles.

Id_mot (ref spip_mots.id_mot).
Id_article (ref spip_articles.id_article).

→ **spip_mots_auteurs**
Liaison des mots clés aux auteurs.

Id_mot (ref spip_mots.id_mot).

Id_auteur (ref spip_auteurs.id_auteur).

→ spip_mots_breves
Liaison des mots clés aux brèves.

Id_mot (ref spip_mots.id_mot).
Id_breve (ref spip_breves.id_breve).

→ spip_mots_cm
Liaison des mots clés aux groupes de destinataire de la newsletter.

Id_mot (ref spip_mots.id_mot).
Id_groupe (ref spip_cm_groups.id_groupe).

→ spip_mots_forum
Liaison des mots clés aux messages du forum.

Id_mot (ref spip_mots.id_mot).
Id_forum (ref spip_forums.id_forum).

→ spip_mots_rubriques
Liaison des mots clés aux rubriques.

Id_mot (ref spip_mots.id_mot).
Id_rubrique (ref spip_rubriques.id_rubique).

→ spip_mots_syndic
Liaison des mots clés aux syndications.

Id_mot (ref spip_mots.id_mot).
Id_syndic (ref spip_syndics.id_syndic).

→ spip_petitions
Stockage des petitions.

Clé primaire Id_article.

→ spip_profils
Stockage des profils utilisateurs.

Clé profil poids.

→ spip_referers
Stockage des referers.

Clé primaire referer_md5.

→ spip_referers_articles
Stockage des referers pour les articles.

Clé primaire (id_article, referer_md5).

Id_article (ref spip_articles.id_article).

→ **spip_referers_temp**
Stockage temporaire des referers.

Clé primaire (ip, referer_md5, type_referer, id_objet).

→ **spip_rubriques**
Stockage des rubriques.

Clé primaire id_rubrique.
id_parent (spip_rubriques.id_rubrique).
id_secteur (spip_rubriques.id_rubrique).

→ **spip_signatures**
Stockage des signatures.

Clé primaire id_signature.

Id_article (ref spip_articles.id_article).

→ **spip_sondages**
Stockage des sondages.

Clé primaire id_article.

→ **spip_sondages_reponses**
Stockage des réponses proposées par chaque sondage.

Clé primaire (id_article, id_reponse).

Id_article (ref spip_sondages.id_article).

→ **spip_sondages_votes**
Stockage des votes pour chaque sondage.

Clé primaire (id_article, email_votant).

Id_article (spip_sondages.id_article).

→ **spip_syndic**
Stockage des syndications.

Clé primaire id_syndic.
Id_rubrique (ref spip_rubriques.id_rubrique).
Id_secteur (ref spip_rubriques.id_rubrique).

→ **spip_syndic_articles**
Stockage des articles syndiqués.

Clé primaire id_syndic_article.
Id_syndic (ref spip_syndic.id_syndic).

→ **spip_types_documents**
Stockage des différents types de document géré par AGORA

Clé primaire id_type.

→ **spip_visites**
Stockage des visites

Clé primaire date_heure.

→ **spip_visites_articles**
Stockage des visites pour chaque article.

Clé primaire (date_heure, id_article).
Id_article (ref spip_article.id_article).

→ **spip_visites_temp**
Stockage temporaire des visites.

Clé primaire (ip, type, id_objet).

→ **spip_*_seq**
Table de séquence pour l'élément *. (http://pear.php.net/manual/fr/package.database.db.intro-sequences.php)

Les classes métiers

La possibilité d'instancier AGORA avec différentes bases de données a nécessité le développement d'une abstraction de base de données.

1) Le Principe général :

L'abstraction de bases de données repose principalement sur l'utilisation de la norme SQL 92, définissant les standards en matière de base de données relationnelles.
Idéalement, toute base de données vérifiant cette norme doit pouvoir être utilisée avec AGORA. Dans la pratique cette vérité n'en est pas une, puisque la plupart des bases de données commerciales ou libres n'implémentent qu'un sous ensemble de cette norme.
Le principe général de cette couche d'abstraction est donc d'utiliser un composant central générique, vérifiant parfaitement la norme SQL 92. L'utilisation d'une base de données ne vérifiant pas un sous ensemble convenable de la norme nécessaire au fonctionnement d'AGORA nécessite donc l'utilisation d'un driver spécifique.
Dès qu'un composant d'AGORA nécessite une action sur une source de données, il délègue cette tâche à la couche d'abstraction. Il s'agit donc d'un composant fondamental du framework, étant donné qu'il a la lourde responsabilité d'assurer l'ensemble des échanges de données entre le cœur applicatif d'AGORA et les différentes sources de données qu'AGORA est susceptible d'utiliser.

Cette couche d'abstraction se décompose elle-même en deux parties :

- Abstraction technique de base de données
- Abstraction métier

- **Abstraction technique**

L'abstraction technique a pour but de permettre la connexion de l'application à n'importe quelle base de données, en utilisant une API standardisée. Elle permet donc un premier niveau d'abstraction. Cependant, cette couche seule ne suffit pas, puisqu'elle ne gère que les échanges d'informations entre le tiers données et le tiers applicatif. Elle ne permet pas en effet de transformer des requêtes SQL en requêtes compréhensibles par n'importe quelle base de données. Par exemple, une requête utilisant des fonctionnalités spécifiques MySQL ne sauraient être traduites par cette couche pour fonctionner sur n'importe quelle autre base.
C'est pour cela qu'au dessus de cette couche purement technique, se trouve une couche métier, dont le rôle est principalement de standardiser les requêtes SQL, en les abstrayant derrière une API unifiée.
La solution logicielle employée pour cette couche d'abstraction technique est PEAR::DB, qui présente l'avantage de fonctionner avec un très grand nombre de bases de données différentes, et de limiter pour certains cas précis l'utilisation de commandes SQL spécifiques.

- **Abstraction métier**

La couche d'abstraction métier est la couche appelée depuis le cœur applicatif d'AGORA lorsqu'il est nécessaire d'effectuer une requête sur la base de données utilisée. On y retrouve ainsi une modélisation de tous les objets métiers utilisés par AGORA, parmi lesquels on peut citer les articles, brèves, rubriques, auteurs, etc.

C’est au sein de cette couche que la notion de « driver » fait sens. En effet, c’est dans cette couche qu’est regroupé l’ensemble des requêtes SQL. Donc, lorsqu’une requête standard n’est pas supportée par une base de données, il s’agit de redéfinir une requête qui, elle, fonctionne correctement dans ce contexte.
L’utilisation d’une méthodologie objet dans la réalisation de cette couche permet de faciliter l’écriture de ces drivers. En effet, les mécanismes d’héritage et de polymorphisme permettent à l’auteur d’un driver de n’avoir à écrire uniquement les quelques méthodes contenant des requêtes non supportées.

2) Les types de données

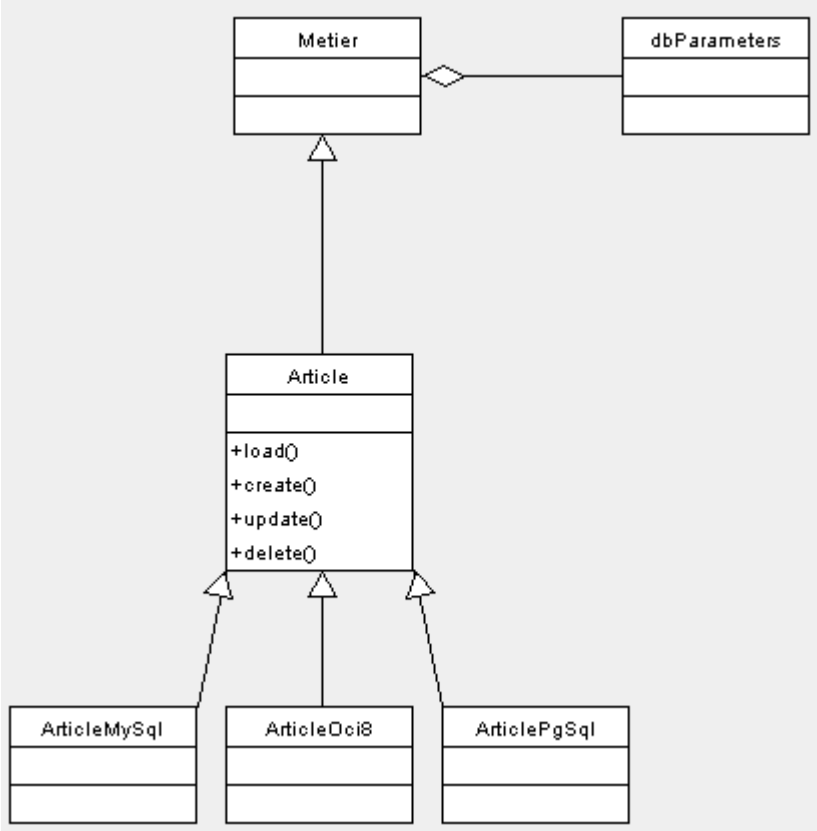
Pour avoir des requêtes respectant la norme SQL 92, il a été nécessaire d'adapter les types de données.

MySql	SQL Server	Oracle	PgSql
bigint, int, smallint	bigint, int	number	
varchar	varchar	varchar	varchar
Blob, longblob	text	clob	
Double	float	float	

La gestion des dates contrairement à SPIP qui utilise le type date de MySql, AGORA utilise le format de date ISO (« 0000-00-00 00:00:00 ») soit une chaîne de 19 caractères. Pour la manipulation des dates, nous utilisons la classe PEAR::Date.

3) Réalisation :

- **Gestion des éléments de base d'AGORA**



Cet extrait de diagramme illustre les propos de l'abstraction métier. En effet pour chaque objet métier Agora (un article sur ce schéma) un driver spécifique à chaque base de données (ArticleMySql, ArticleOci8, ArticlePgSql ...) est à écrire en plus de la classe de base (Article) implémenter un driver standard SQL 92. L'instanciation du bon objet métier dans l'applicatif d'AGORA se fait par l'intermédiaire d'une Factory (sous la forme d'une fonction PHP défini dans le fichier inc_article_factory.php) qui instancie le bon objet en fonction des paramètres de l'application.

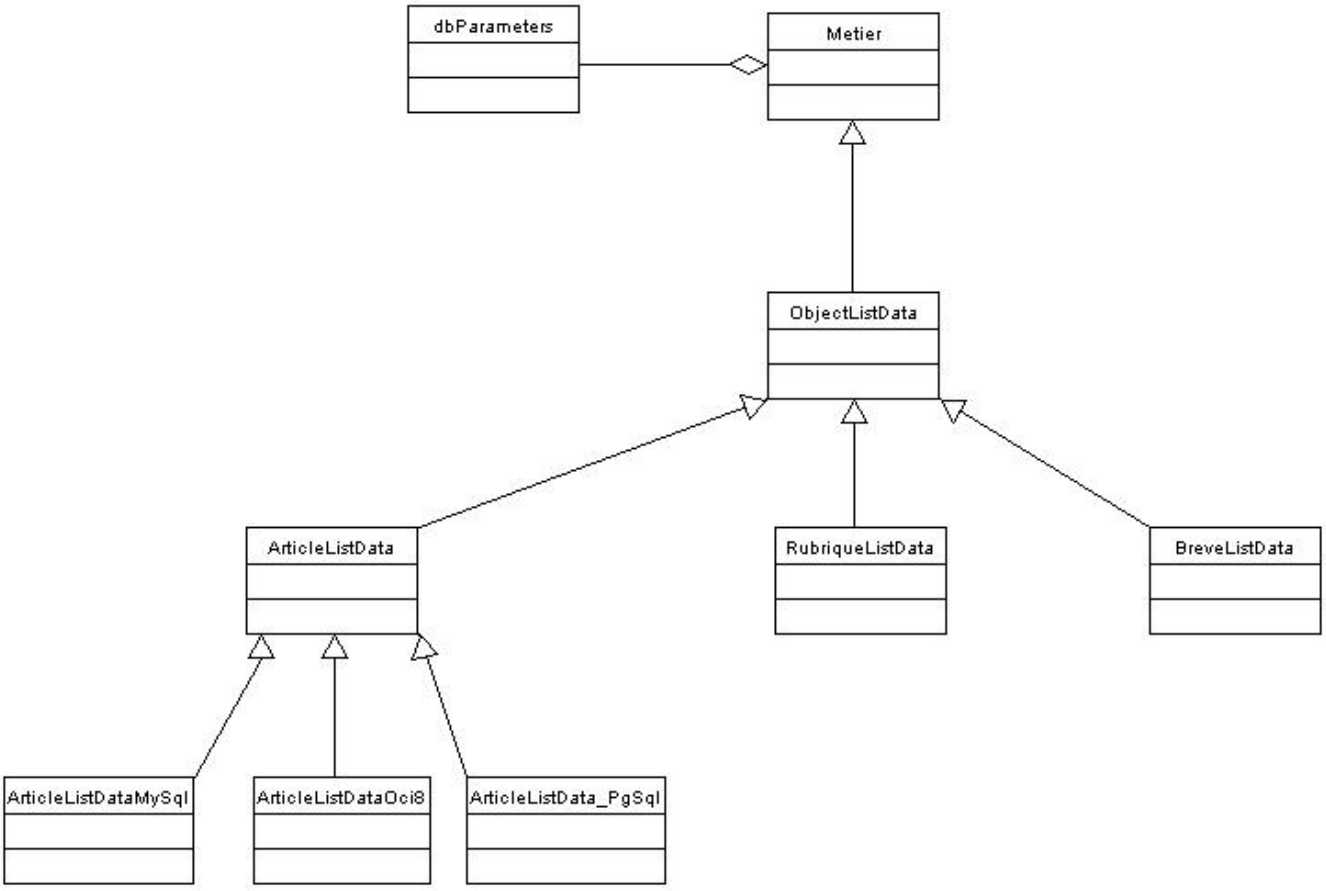
Il a donc été nécessaire de déporter l'ensemble des appels à la base de données SPIP vers nos différentes classes métiers.

***Exemple*:**

```
/* ANCIEN CODE SPIP */
spip_query("UPDATE spip_articles SET date_modif=NOW(),
auteur_modif=$connect_id_auteur WHERE id_article=$id_article");

/* NOUVEAU CODE AGORA */
$articleMetier = &recuperer_instance_article();
$loadOK = $articleMetier->load($id_article);
if(PEAR::isError($loadOK)) {
      die($loadOK->getMessage());
}
```

```
$maDate = new Date();
$articleMetier->setModificationDate($maDate->getDate());
$articleMetier->setAuteurModif($connect_id_auteur);
$updateOK = $articleMetier->update();
if(PEAR::isError($updateOK)) {
      die($updateOK->getMessage());
}
```

- **Gestion des listes d'élément d'AGORA**



Cet extrait de diagramme illustre les propos de l'abstraction métier au niveau de la gestion du listing d'élément dans AGORA. Le principe de l'écriture de drivers commun et de drivers spécifiques est repris. Par contre le principe d'utilisation de la liste d'élément est de :

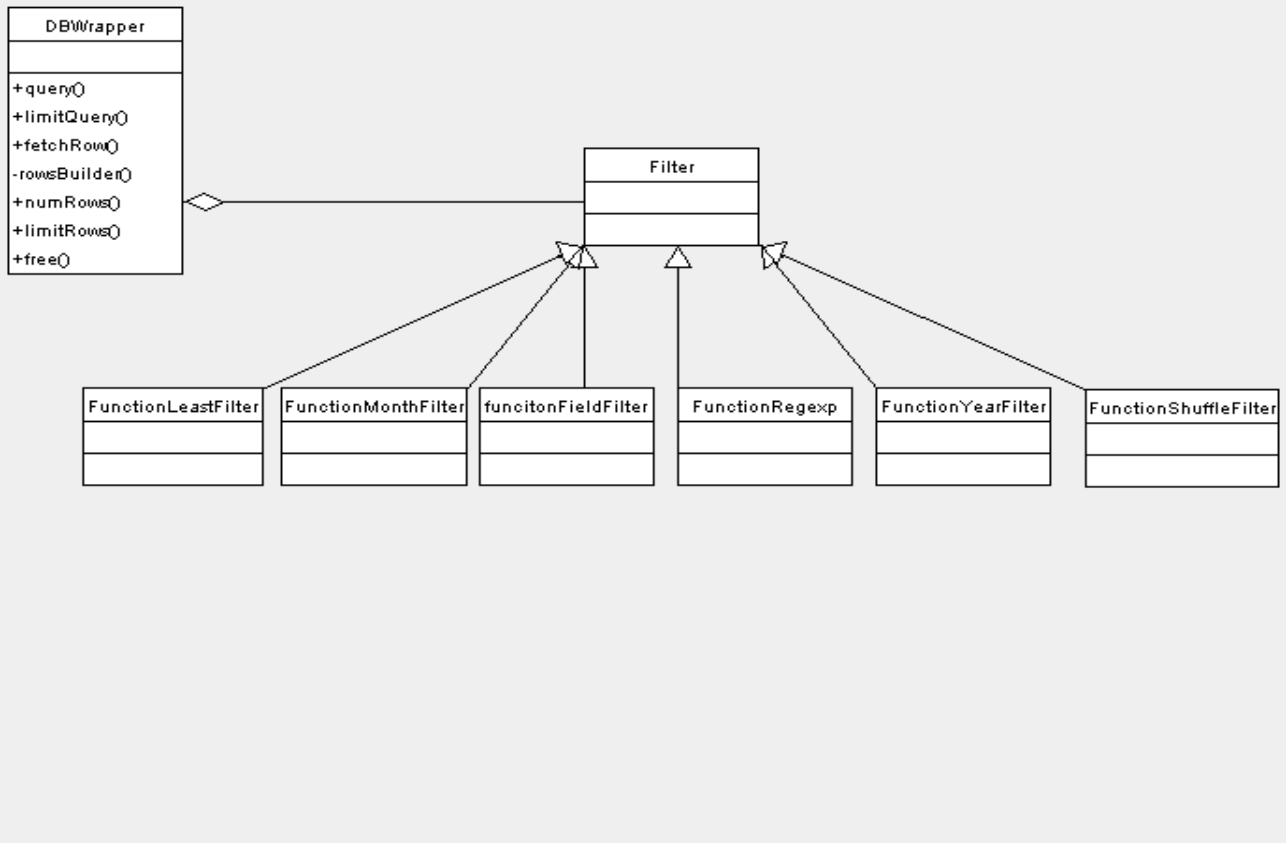
Définir une liste de requêtes dans un tableau associatif ($_queries). A chaque requête, il est nécessaire d'associer une requête retournant le nombre de résultat afin de dimensionner la zone de listing à afficher ($_queriesCount). Il est possible de mettre dans ces requêtes des tags prédéfinis qui seront traités par la méthode getQuery () de objectListData avant l'exécution de celles-ci.
Les tags supportés :

- <!spipNOW_DATE_FORMAT_ISOspip!> afin de remplacer la fonction non standard « now() » de MySql.
- <!spipDATE_SUBTRACT_ONE_DAYspip!> afin de retirer un jour à la date courante.

Une fois la requête correctement formatée la méthode displayHTMLList s'occupe d'afficher correctement la liste d'éléments.

Le besoin d'AGORA de satisfaire le support de toutes bases de données satisfaisant la norme SQL 92 a nécessité de rendre la compilation des squelettes conforme à cette norme.

Le choix qui a été fait a été fait de nous appuyer sur la méthode de compilation des squelettes actuelle de SPIP en y rajoutant une couche supplémentaire transformant et exécutant les requêtes générées. Le résultat renvoyé par cette requête peut également subir quelques transformations le cas échéant.



Le principe est le suivant: Pendant le parsing du squelette, les requêtes SQL sont générées tout en étant marquées par une liste de filtres.
Chaque requête est envoyée avec sa liste de filtre à un objet Wrapper de DB se chargeant de manipuler la requête, de l'exécuter, de traiter le résultat puis de renvoyer celui-ci.

Ce diagramme montre les filtres indispensables au bon fonctionnement des squelettes AGORA.

- FunctionLeastFilter : Ce filtre se charge de simuler la fonction LEAST de MySql.
- FunctionMonthFilter : Ce filtre se charge de simuler la fonction MONTH de MySql.
- FunctionFieldFilter : Ce filtre se charge de simuler la fonction FIELD de MySql.
- FunctionRegExp : Ce filtre se charge de simuler la fonction RegExp de MySql.
- FunctionYearFilter : ce filtre se charge de simuler la fonction Year de MySql.
- FunctionShuffleFilter : Ce filtre se charge de simuler la fonction Shuffle de MySql.

4) Comment écrire un driver spécifique ?

Si vous souhaiter écrire un driver pour les articles supportant la base de données « mySGBD » (il faut que celle-ci soit supporté par PEAR::DB) vous devez simplement créer dans le répertoire **ecrire/include/bd/mySGBD/** le fichier article_mySGBD.php en implémentant dans une classe BD_article_mySGBD étendant la classe BD_article les méthodes métiers correspondant aux articles.

5) Comment « AGORATISER » une contribution SPIP

Prenons l'exemple de la contribution Admin SideBar (http://www.uzine.net/spip_contrib/article.php3?id_article=202).

L'installation de la contribution se fait de la même façon que sous SPIP :

- Créez un fichier admin_sidebar.php3 et un fichier admin-sidebar.html à la racine de votre site.
- Copiez collé le code ci-dessous dans admin-sidebar.php3

```
<?
$fond = "admin_sidebar";
$delais = 0;

include ("inc-public.php3");
?>
```

- Copiez collé le code ci-dessous dans admin-sidebar.html

```
<?php
//Ici votre configuration
$url_site="http://maison.noplay.net/noplay2/"; //l'url de votre site web
(avec / à la fin)
$titre="Noplay.net Admin SideBar"; // le titre de votre sidebar


// Ne pas modifier aprés cette ligne
include("ecrire/inc_connect.php3");
$url=$url_site."admin_sidebar.php3?mode=read";
?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="author" content="Noplay">
<meta http-equiv="Refresh" content="1200; URL=<?php echo $url;?>">
<title><?php echo $titre;?></title>
<?php
if ($_GET['mode']!="read") {
      print "
      <script language=\"JavaScript\">
function addSidebar() {
if ((typeof window.sidebar == \"object\") && (typeof
window.sidebar.addPanel == \"function\")) {
window.sidebar.addPanel (\"".$titre."\",
\"".$url."\",\"\");
```

```
} else {
var rv = window.confirm (\"Cette fonctionnalité est uniquement disponible
sur les \"
+ \"navigateur supportant les sidebar\");
if (rv)
document.location.href = \"http://frenchmozilla.sourceforge.net/\";
}
}
</script>
      ";
}
?>
</head>
<body>
<b><?php echo $titre;?></b>
<br>
<br>
<?php
if ($_GET['mode']!="read") {
      print "<a href=\"javascript:addSidebar();\">Installer la
sidebar</a><br><br>";
}
?>
<?php
$aumoinsun=0;
$query  = "SELECT * FROM spip_forum WHERE `statut`='prop'";
$result = spip_query($query);
while ($row = spip_fetch_array($result)) {
                if (!$aumoinsun)                {
                        $aumoinsun=1;
                        echo "Forums: <br><i>";
                };
       echo '- <a href="ecrire/controle_forum.php3"
target="_content">'.$row['titre']."</a><br>";
}
if ($aumoinsun) echo "</i><br><br>";
?>
Dernières bréves:
<br>
<i>
<?php
$query  = "SELECT * FROM spip_breves WHERE statut='prop'";
$result = spip_query($query);
while ($row = spip_fetch_array($result)) {
      echo "- <a href=ecrire/breves.php3?id_breve=".$row['id_breve']."
target=_content>".$row['titre']."</a><br>";
}
?>
</i>
<br>
Derniers articles:
<br>
<i>
<?php
$query  = "SELECT * FROM spip_articles WHERE statut='prop'";
$result = spip_query($query);
while ($row = spip_fetch_array($result)) {
      echo "- <a
href=ecrire/articles.php3?id_article=".$row['id_article']."
target=_content>".$row['titre']."</a><br>";
}
```

```
?>
</i>
<br>
Derniers sites:
<br>
<i>
<?php
$query  = "SELECT * FROM spip_syndic WHERE statut='prop'";
$result = spip_query($query);
while ($row = spip_fetch_array($result)) {
      echo "- <a href=ecrire/sites.php3?id_syndic=".$row['id_syndic']."
target=_content>".$row['nom_site']."</a><br>";
}
?>
</i>
<br>
<br>
<small><a href="http://www.noplay.net/rubrique29.html">Une création de
Noplay.net</a></small>
</body>
</html>
```

Voici le fichier admin-sidebar.html « agoratisé » avec les commentaires expliquant les modifications apportées (en rouge).

```
<?php
//Ici votre configuration
$url_site="http://maison.noplay.net/noplay2/"; //l'url de votre site web
(avec / à la fin)
$titre="Noplay.net Admin SideBar"; // le titre de votre sidebar


// Ne pas modifier aprés cette ligne
include("ecrire/inc_connect.php3");
$url=$url_site."admin_sidebar.php3?mode=read";
?>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="author" content="Noplay">
<meta http-equiv="Refresh" content="1200; URL=<?php echo $url;?>">
<title><?php echo $titre;?></title>
<?php
if ($_GET['mode']!="read") {
      print "
      <script language=\"JavaScript\">
function addSidebar() {
if ((typeof window.sidebar == \"object\") && (typeof
window.sidebar.addPanel == \"function\")) {
window.sidebar.addPanel (\"".$titre."\",
\"".$url."\",\"\");
} else {
var rv = window.confirm (\"Cette fonctionnalité est uniquement disponible
sur les \"
+ \"navigateur supportant les sidebar\");
if (rv)
document.location.href = \"http://frenchmozilla.sourceforge.net/\";
}
}
</script>
      ";
```

```
}
?>
</head>
<body>
<b><?php echo $titre;?></b>
<br>
<br>
<?php
if ($_GET['mode']!="read") {
      print "<a href=\"javascript:addSidebar();\">Installer la
sidebar</a><br><br>";
}
?>
<?php
$aumoinsun=0;
/*
la première étape consiste à mettre en commentaire le code SPIP à
« agoratiser ».
Ensuite il est nécessaire de bien comprendre ce que fait le code mis en
commentaire. Ce code consiste à exécuter une requête permettant la
récupération de l'ensemble des messages de la table spip_forum étant au
statut 'prop'. Ensuite une boucle while parcoure le tableau de résultat
$result pour afficher un lien vers les messages proposés.
Pour faire appel à la couche métier d'AGORA il faut :
- déterminer quel type de classe utilisé : dans notre cas il s'agit de la
classe forum
- déterminer quelle méthode permet d'obtenir la liste des messages avec le
statut 'prop' à partir de la PHPDoc : il s'agit de la méthode
getAllForStatut($statut). Cette méthode renvoie un tableau d'objet forum.
*/
/********* Modification elebescond@clever-age.com ***************/
/*
Récupération d'une instance de forum a partir de la faction
recuperer_instance_forum() qui renvoie le bonne objet en fonction de la
configuration de la base de données.
*/
$forumMetier = recuperer_instance_forum();
/*
Récupération du tableau de message correspondant au statut 'prop'
*/
$forums = $forumMetier->getAllForStatut('prop');
/*
Utilisation de la méthode static isError() de PEAR pour vérifier qu'il n'y
pas d'erreurs PEAR généré lors de l'exécution de la méthode avec affichage
du message d'erreur si c'est le cas.
*/
if(PEAR ::isError($forums)) {
      die($forums->getMessage()) ;
}
/*
Parcours du tableau de messages pour afficher les liens vers les messages
proposés
*/
while ( list(, $monForum) = each($forums) ) {
    if (!$aumoinsun) {
        $aumoinsun=1;
        echo "Forums: <br><i>";
    };
/*
Appel de la méthode getTitre() pour récupérer le titre du message en cours
*/
```

```
    echo'- <a href="ecrire/controle_forum.php3" target="_content">'.$forum-
>getTitre()."</a><br>";
}
/*
$query  = "SELECT * FROM spip_forum WHERE `statut`='prop'";
$result = spip_query($query);
while ($row = spip_fetch_array($result)) {
                if (!$aumoinsun)                {
                        $aumoinsun=1;
                        echo "Forums: <br><i>";
                };
       echo '- <a href="ecrire/controle_forum.php3"
target="_content">'.$row['titre']."</a><br>";
}
*/
/********* Fin Modification elebescond@clever-age.com *************/
if ($aumoinsun) echo "</i><br><br>";
?>
Dernières brêves:
<br>
<i>
<?php
/*
la première étape consiste à mettre en commentaire le code SPIP à
« agoratiser ».
Ensuite il est nécessaire de bien comprendre ce que fait le code mis en
commentaire. Ce code consiste à exécuter une requête permettant la
récupération de l'ensemble des brêves de la table spip_breves étant au
statut 'prop'. Ensuite une boucle parcoure le tableau de résultat $result
pour afficher un lien vers les brêves proposées.
Pour faire appel à la couche métier d'AGORA il faut :
- déterminer quel type de classe utilisé : dans notre cas il s'agit de la
classe breve
- déterminer quelle méthode permet d'obtenir la liste des messages avec le
statut 'prop' à partir de la PHPDoc : Il n'existe pas de méthode permettant
de réaliser cette opération. Il va falloir créer la méthode
getAllForStatut($statut). Cette méthode renverra un tableau d'objet de
brêve.
*/
/********* Modification elebescond@clever-age.com ***************/
/*
Récupération d'une instance de brêve a partir de la faction
recuperer_instance_breve() qui renvoie le bonne objet en fonction de la
configuration de la base de données.
*/
$breveMetier = recuperer_instance_breve();
/*
Récupération du tableau de message correspondant au statut 'prop'
*/
$breves = $breveMetier->getAllForStatut('prop');
/*
Utilisation de la méthode static isError() de PEAR pour verifier qu'il n'y
pas d'erreurs PEAR généré lors de l'exécution de la méthode avec affichage
du message d'erreur si c'est le cas.
*/
if(PEAR ::isError($breves)) {
      die($breves->getMessage()) ;
}
/*
Parcours du tableau de messages pour afficher les liens vers les messages
proposés
```

```
*/
while ( list(, $maBreve) = each($breves) ) {
/*
Appel des méthodes :
      getBreveId() pour récupérer l'id de la brêve en cours
      getTitre() pour récupérer le titre de la brêve en cours
*/
    echo "- <a href=ecrire/breves.php3?id_breve=".$maBreve->getBreveId()."
target=_content>".$maBreve->getTitre()."</a><br>";
}
/*
$query  = "SELECT * FROM spip_breves WHERE statut='prop'";
$result = spip_query($query);
while ($row = spip_fetch_array($result)) {
      echo "- <a href=ecrire/breves.php3?id_breve=".$row['id_breve']."
target=_content>".$row['titre']."</a><br>";
}
*/
/********* Fin Modification elebescond@clever-age.com *************/
?>
</i>
<br>
Derniers articles:
<br>
<i>
<?php
/********* Modification elebescond@clever-age.com *************/
$articleMetier = recuperer_instance_article();

$articles = $articleMetier->getAllForStatut('prop');

if(PEAR ::isError($articles)) {
      die($articles->getMessage()) ;
}

while ( list(, $monArticle) = each($articles) ) {
      echo "- <a href=ecrire/articles.php3?id_article=".$monArticle-
>getArticleId()." target=_content>".$monArticle->getTitre()."</a><br>";

}
/*
$query  = "SELECT * FROM spip_articles WHERE statut='prop'";
$result = spip_query($query);
while ($row = spip_fetch_array($result)) {
      echo "- <a
href=ecrire/articles.php3?id_article=".$row['id_article']."
target=_content>".$row['titre']."</a><br>";
}
*/
/********* Fin Modification elebescond@clever-age.com *************/
?>
</i>
<br>
Derniers sites:
<br>
<i>
<?php
/********* Modification elebescond@clever-age.com *************/
$syndicMetier = &recuperer_instance_syndic();
$sites = $syndicMetier->getAllForStatut('prop');
```

```
if(PEAR ::isError($sites)) {
      die($sites->getMessage()) ;
}

while ( list(, $monSite) = each($sites) ) {
      echo "- <a href=ecrire/sites.php3?id_syndic=".$monSite-
>getSyndicId()." target=_content>".$monSite->getNomSite()."</a><br>";
}
/*
$query  = "SELECT * FROM spip_syndic WHERE statut='prop'";
$result = spip_query($query);
while ($row = spip_fetch_array($result)) {
      echo "- <a href=ecrire/sites.php3?id_syndic=".$row['id_syndic']."
target=_content>".$row['nom_site']."</a><br>";
}
*/
/********* Fin Modification elebescond@clever-age.com *************/
?>
</i>
<br>
<br>
<small><a href="http://www.noplay.net/rubrique29.html">Une création de
Noplay.net</a></small>
</body>
</html>
```

Création de la méthode métier getAllForStatut($statut) dans la classe brêve

```
    // {{{ getAllForStatut($statut)

    /**
     * Returns an array of Breve.
     *
     * @return Array of Breve
     * @param $statut
     * @access public
     */

    function &getAllForStatut($statut) {
        $forums = array();
        //Recuperation d'une connection a la base de données
        $db = &$this->_getDB();

        //Si la récupération ne se passe pas bien c'est à dire que l'on a
//recupere une DBError, on retourne une PEARError personnalisée.
        if (DB::isError($db)) {
            return PEAR::raiseError("[".get_class($this)." DB_breve :
getAllForStatut()] ".$db->getMessage()."", null,
                null, null, null, null, false);
        }
        //Ecriture de la requête
        $query = "SELECT * FROM spip_breves WHERE statut='$statut'";

        //echo "<br><br>$query<br><br>";
        //Exécution de la requête
        $queryResult = $db->query($query);
        //Si l'exécution se passe mal, on retourne une PEARError.
        if (DB::isError($queryResult)) {
            return PEAR::raiseError("[".get_class($this)." DB_breve :
getAllForStatut()] ".$queryResult->getMessage()."", null,
```

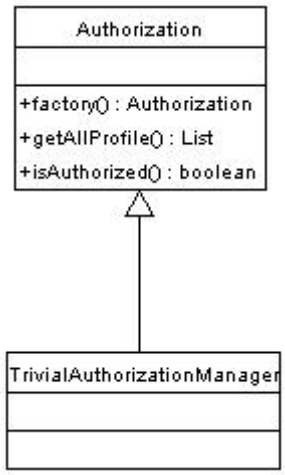
```
                null, null, null, null, false);
        }

        //Parcours du tableau de résultat
        while ($row = $queryResult->fetchRow()) {
             //Recuperation d'une instance de brêve
            $resultBreve = &BD_breve::factory($this->getDbParameters(),
$this->getDbOptions());
             //Mise au bon format du champ maj
            $maDate = new Date($row['maj']);
            $row['maj']= $maDate->getDate(DATE_FORMAT_TIMESTAMP);
             //initialisation de la brève
            $resultBreve->_fetchData($row);
             //Stockage de la brève dans le tableau de breve
            $breves[] = &$resultBreve;
        }
        $queryResult->free();
        return $breves;
    }
    // }}}
```

Le module d'autorisation

AGORA propose par défaut 5 profils utilisateur avec pour chacun d'eux un poids lui étant associé:

Profil	Poids
texte_statut_poubelle	0
Item_non_abonne	1
item_redacteur	2
item_redacteur_en_chef	4
item_webmestre	8
item_administrateur	16

Les fichiers gérant les autorisations sont placés dans le répertoire ecrire/include/authorization/



L'implémentation de ce module d'autorisation est faite par la classe TrivialAuthorizationManager dans le fichier trivial_authorization.php. Le stockage des informations d'autorisation a pour support la table spip_actions.

La méthode isAuthorized () effectue le test d'autorisation en comparant dans un premier temps si le profil de l'utilisateur correspond bien au profil minimum requis (en comparant les poids). Ensuite divers tests sont effectués suivant les actions.

L'exemple suivant montre comment réaliser un test d'autorisation dans le code de présentation d'AGORA sur l'action 'modifierArticle':

```
$authorization = &recuperer_instance_authorization();
if (!$authorization->isAuthorized($GLOBALS['connect_id_auteur'],
'modifierArticle', array('id_article' => $id_article))) {
      echo _T('avis_acces_interdit');
      exit;
}
```

La méthode isAuthorized () va donc dans un premier comparer le profil de l'auteur avec celui requis pour faire l'action. Si la personne est autorisée à exécuter l'action, un second test est fait en prenant en compte l'Id de l'article passé en paramètres.

L'implémentation de ce second test consiste à vérifier que l'utilisateur à bien accès à la rubrique à laquelle appartient l'article.

Le module de personnalisation

Comme nous l'avons vu dans la partie précédente, AGORA permet de gérer une personnalisation du contenu basé sur l'attribution de mots clés aux utilisateurs du site.

La problématique de cette personnalisation des pages consiste dans le stockage des pages HTML générées et stockées dans le CACHE d'AGORA. Il ne faut pas qu 'un visiteur X tombe sur la page personnalisée d'un visiteur Y. Il est donc nécessaire de placer le CACHE à 0 pour les squelettes de personnalisation.

Le module d'authentification

Les modules d'authentification MySql et LDAP sont semblables fonctionnellement à ceux de SPIP. En revanche afin de respecter la couche métier technique d'AGORA, la gestion de l'authentification a été migré pour assurer la conformité. Ainsi la configuration du SGBD et LDAP se fait au travers du fichier ecrire/include/bd/inc_config_metier.php .

Le module de recherche

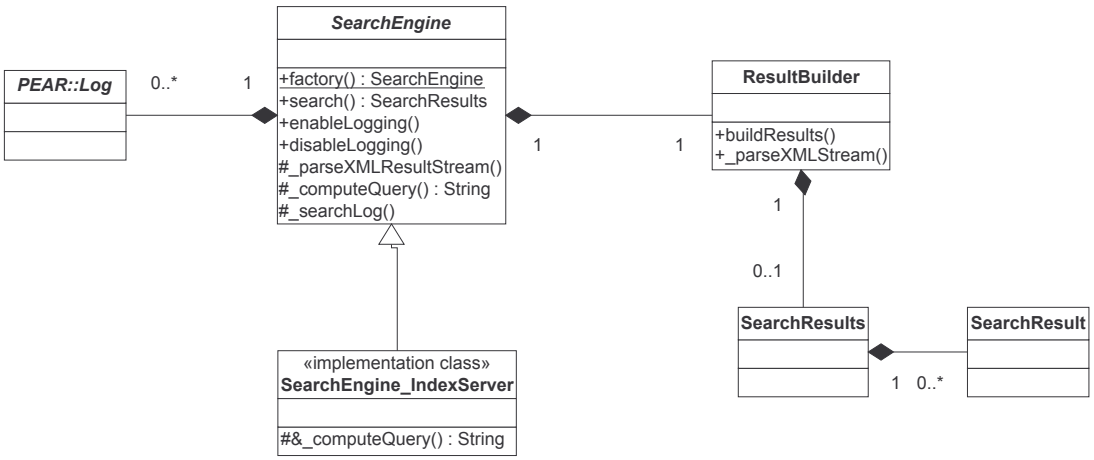
Comme nous l'avons vu dans la première partie de ce document, le module de recherche d'AGORA se voir enrichit par l'intégration d'un moteur d'indexation MnoGoSearch au travers de Seven Seas.

Les principaux objectifs de Seven Seas sont :

- Permettre une utilisation transparente de différents moteurs, sur serveur dédié ou non.
- Assurer l'indépendance entre le code de l'application et l'utilisation d'un moteur spécifique.
- Etre facilement et simplement extensible à d'autres moteurs.

Dans cette optique, Seven Seas permet d'utiliser un jeu d'interfaces unifiées pour l'accès à n'importe quel moteur d'indexation, qu'il fonctionne sur un serveur distant ou non. Cette solution permet de remédier à tous les problèmes évoqués précédemment.

Architecture



On retrouve donc l'interface principale pour effectuer des requêtes sur un moteur d'indexation dans la classe SearchEngine. Les « drivers » pour les différents moteurs sont des classes qui héritent de cette classe abstraite. Ainsi, le driver pour Index Server a été représenté pour illustrer ce mécanisme.
La classe ResultBuilder est une implémentation par défaut permettant de construire un arbre de résultat depuis le flux XML reçu du moteur d'indexation. L'utilisation d'une classe personnalisée est tout à fait envisageable, en écrivant une classe héritant de ResultBuilder.
Les classes SearchResult et SearchResults ne sont que des conteneurs : SearchResults contient des SearchResult, eux-mêmes contenant les informations nécessaire à l'exploitation d'un résultat précis.

Configuration de MNOGOSEARCH :

Tout d'abord il faut configurer MnoGoSearch pour que celui-ci indexe le répertoire où AGORA dépose les documents uploadés c'est à dire le répertoire "monAgora/IMG/".
Cette configuration est à réaliser dans le fichier indexer.conf.

Server http://documents/IMG/
Alias http://documents/IMG/ file:/votreChemin/monAgora /IMG
CheckOnly NoMatch */
HrefOnly */

Une fois cette configuration réalisée, il est nécessaire de définir le template de résultat suivant dans le fichier monMnoGoSearch/etc/search.htm

```
<!--restop-->
<?xml version="1.0" encoding="ISO-8859-1"?>
<results xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSc
hemaLocation="indexresults.xsd">
   <numberofresults>$t</numberofresults>
   <firstresultindex>$f</firstresultindex>
   <lastresultindex>$l</lastresultindex>
<!--/restop-->
<!--res-->
   <result>
      <title>$DX</title>
      <pertinence>$DR</pertinence>
      <vpath>NA</vpath>
      <path>$DT</path>
      <filename><![CDATA[$DU]]></filename>
      <abstract><![CDATA[$DE]]></abstract>
   </result>
<!--/res-->
<!--resbot-->
</results>
<!--/resbot-->
```

Le module de newsletters

AGORA propose un module de gestion de lettre d'information basé sur la solution CLEVERMAIL.

L'ensemble des fonctionnalités d'administration de CLEVERMAIL a été directement intégré dans le back office d'AGORA. Seul les fichiers suivants ont encore leur utilité.

Les fichiers suivants gardent tout de même leur utilité :
→ do.php : Ce fichier prend un Id d'action et effectue les traitements nécessaire (Inscription, désinscription, changement du format de réception des newsletter).
→ interface.php : Ce fichier contient le formulaire front d'inscription à la newsletter.
→ rm.php : Ce fichier permet de générer le mail de désinscription d'un abonné.
→ subscribe.php : Ce fichier permet de générer le mail d'inscriptions à une newsletter.

2. Installer AGORA

Configuration nécessaire

- Serveur WEB avec module PHP4.
- Le système de composant PEAR (http://pear.php.net).
- 1 Base de données respectant la norme SQL 92 (la version actuelle d'AGORA a été validée avec MySql, PostGresql, Oracle 8i et SQL Server).

Installation automatique

Récupérer l'archive du projet AGORA et décompacter celle-ci dans le répertoire d'installation souhaitée.
Créer votre base de données.
Se rendre dans le répertoire ecrire et suivre les instructions.

Migration SPIP 1.7 vers AGORA ()

La migration d'un SPIP vers un AGORA n'est possible que depuis la version 1.7 de SPIP. Il est nécessaire de mettre à jour votre version de SPIP. Il suffit de se laisser guider ensuite par l'installation automatique en suivant les points suivants :

La migration doit s'effectuer sur une instance d'AGORA vierge.

- Déployer une instance d'AGORA sur la machine. Cette dernière doit pouvoir accéder à la base de données de SPIP.
- Faire un dump de la base originale de SPIP (En cas de problème).
- Copier les différents fichiers uploadés de votre SPIP: copier le répertoire MONSPIP/IMG vers MONAGORA/IMG

AGORA doit alors en effet utiliser la base de données migrée, qui est l'ancienne base de données SPIP. Vous pouvez donc en faire un dump pour créer une autre base, à paramétrer alors dans le inc_config_metier.php, et remettre votre DUMP de SPIP si vous désirez continuer à l'utiliser.

3. Les boucles dans AGORA

Cette partie de la documentation a pour but de définir les différentes boucles possibles dans AGORA.
NB : toutes les fonctionnalités de multilinguisme sont documentées à l'adresse suivante :
http://www.spip.net/fr_article2124.html
http://www.spip.net/fr_article2128.html

La boucle (RUBRIQUES)

La boucle RUBRIQUES retourne une liste de rubriques.

<BOUCLEn(RUBRIQUES){critères...}>

Remarque. Une boucle RUBRIQUES n'affiche que des rubriques « actives », c'est-à-dire contenant des articles publiés, des documents joints, des sites publiés - ou des sous rubriques elles-mêmes actives. De cette façon, on évite de se trouver dans des rubriques « culs de sac » n'offrant aucun élément de navigation.

1) Les critères de sélection

On utilisera l'un ou autre des critères suivants pour indiquer comment on sélectionne les éléments.

{tout} les rubriques sont sélectionnées dans l'intégralité du site.

{id_rubrique} retourne la rubrique dont l'identifiant est id_rubrique. Comme l'identifiant de chaque rubrique est unique, ce critère retourne une ou zéro réponse.

{id_secteur} retourne les rubriques de ce secteur. (On peut également, par extension, utiliser le critère {branche} décrit dans La boucle ARTICLES).

{id_parent} retourne la liste des rubriques contenues dans une rubrique.

{racine} retourne la liste des secteurs (rigoureusement identique à {id_parent=0}).

{id_enfant} retourne la rubrique qui contient la rubrique (une seule réponse ; ou zéro réponse si la présente rubrique est située à la racine du site).

{meme_parent} retourne la liste des rubriques dépendant de la même rubrique que la rubrique en cours. Permet d'afficher les rubriques « soeurs » qui se trouvent au même niveau dans la hiérarchie.

{recherche} retourne les rubriques correspondant aux mots indiqués dans l'interface de recherche (moteur de recherche incorporé à AGORA). Voir la page consacrée au moteur de recherche.

Les rubriques pouvant être liées à des mots-clés. Les critères de mots-clés peuvent donc être désormais utilisés dans les boucles (RUBRIQUES) :

{id_mot}, **{titre_mot=xxx}** récupèrent les rubriques liées au mot dont le numéro est id_mot ou dont le titre est titre_mot.
{id_groupe}, **{type_mot=yyyy}** récupèrent les rubriques liées à des mots du groupe id_groupe, ou du groupe dont le titre est type_mot.

Le statut 'archive' apporté par AGORA s'utilise de la façon suivante:

{archives} récupère les rubriques possédant des articles archivés et au moins un autre objet d'un autre statut.
{archive} récupère les rubriques ne possédant que des articles archivés et aucun élément publié.
{personnalisation} récupère uniquement les rubriques ciblant un utilisateur.
{personnalisation=x,y,z} récupère uniquement les rubriques x,y,z ciblant un utilisateur.

2) Les critères d'affichage:

Une fois fixée l'un des critères ci-dessus, on pourra ajouter les critères suivants pour restreindre le nombre d'éléments affichés.

Les critères communs à toutes les boucles s'appliquent évidemment.

{exclus} permet d'exclure du résultat la rubrique dans lequel on se trouve déjà (utile avec meme_parent).

{doublons} ou **{unique}** (ces deux critères sont rigoureusement identiques) permettent d'interdire l'affichage de rubriques déjà affichées dans d'autres boucles.

3) Les balises de cette boucle

Les balises tirées de la base de données

Les balises suivantes correspondent aux éléments directement tirés de la base de données. Vous pouvez les utiliser également en tant que critère de classement (généralement : {par titre}).

#ID_RUBRIQUE affiche l'identifiant unique de la rubrique.

#TITRE retourne le titre de la rubrique.

#DESCRIPTIF retourne le descriptif.

#TEXTE retourne le texte principal de la rubrique.

#ID_SECTEUR est l'identifiant du secteur dont dépend la rubrique (le secteur étant la rubrique située à la racine du site).

Les balises calculées par AGORA

Les éléments suivants sont calculés par AGORA. (Ils ne peuvent pas être utilisés comme critère de classement.)

#NOTES les notes de bas de page (calculées à partir de l'analyse du texte).
#INTRODUCTION les 600 premiers caractères du texte, les enrichissements typographiques (gras, italique) sont supprimés.

#URL_RUBRIQUE est l'URL de la page de la rubrique.

#DATE affiche la date de la dernière publication effectuée dans la rubrique et/ou ses sous rubriques (articles, brèves...).

#FORMULAIRE_FORUM fabrique l'interface permettant de poster un message répondant à cette rubrique.

#PARAMETRES_FORUM fabrique la liste des variables exploitées par l'interface du formulaire permettant de répondre à cette rubrique. Par exemple :

[<A HREF="forum.php3?(#PARAMETRES_FORUM)">Répondre à cette rubrique</A>]

#FORMULAIRE_SITE Le #FORMULAIRE_SITE affiche une interface permettant aux visiteurs du site de proposer des référencements de sites. Ces sites apparaîtront comme « proposés » dans l'espace privé, en attendant une validation par les administrateurs.

Ce formulaire ne s'affiche que si vous avez activé l'option « Gérer un annuaire de sites » dans la Configuration sur site dans l'espace privé, et si vous avez réglé « Qui peut proposer des sites référencés » sur « les visiteurs du site public ».

Les logos

#LOGO_RUBRIQUE le logo de la rubrique, éventuellement avec la gestion du survol. S'il n'y a pas de logo pour cette rubrique, AGORA va automatiquement chercher s'il existe un logo pour la rubrique dont elle dépend, et ainsi de suite de manière récursive.

Le logo s'installe de la manière suivante :

[(#LOGO_RUBRIQUE|alignement|adresse)]

#LOGO_RUBRIQUE_NORMAL affiche le logo « sans survol » ;

#LOGO_RUBRIQUE_SURVOL affiche le logo de survol : ces deux balises permettent par exemple, quand on est dans une rubrique, de gérer un logo « avec survol » pour les liens vers les autres rubriques, et de laisser le logo de survol seul dans la rubrique active.

La boucle (ARTICLES)

Une boucle d'articles se code en plaçant ARTICLES (avec un « s ») entre parenthèses :

<BOUCLEn(ARTICLES){critères...}>

Les éléments contenus dans une telle boucle sont des articles.

Remarque. Une boucle ARTICLES ne retourne que des articles publiés ou archivés. (Il n'existe aucun moyen d'afficher les articles « en cours de rédaction », « proposés à la publication » ou « refusés ».)

1) Les critères de sélection

On utilisera l'un ou l'autre des critères suivants pour indiquer comment on sélectionne les éléments.

{tout} les articles sont sélectionnés dans l'intégralité du site (dans toutes les rubriques). Utile notamment pour afficher les articles les plus récents (dans l'intégralité du site) sur la page d'accueil. [En réalité, le critère « tout » n'est pas traité de manière informatique : c'est un aide-mémoire pour le webmestre ; on obtient le même résultat en n'indiquant aucun des critères suivants.]

{id_article} retourne l'article dont l'identifiant est id_article. Comme l'identifiant de chaque article est unique, ce critère ne retourne qu'une ou zéro réponse.

{id_rubrique} retourne la liste des articles contenus dans la rubrique id_rubrique.

{id_secteur} retourne les articles dans ce secteur (un secteur est une rubrique qui ne dépend d'aucune autre rubrique, c'est-à-dire située à la racine du site).

{branche} : retourne l'ensemble des articles de la rubrique ET de ses sous rubriques. (C'est une sorte d'extension du critère {id_secteur}. Toutefois, à l'inverse de {id_secteur=2}, il n'est pas possible d'appeler directement une branche en faisant par exemple {branche=2} : techniquement parlant, il faut que la rubrique en question figure dans le contexte courant. Ce critère est à utiliser avec parcimonie : si votre site est bien structuré, vous ne devriez pas en avoir besoin, sauf dans des cas très particuliers.)

{id_auteur} retourne les articles correspondant à cet identifiant d'auteur (utile pour indiquer la liste des articles écrits par un auteur).

{id_mot} retourne les articles correspondant à cet identifiant de mot-clé (utile pour indiquer la liste des articles traitant d'un sujet donné).

{titre_mot=xxxx}, ou **{type_mot=yyyy}** retourne les articles liés au mot-clé dont le nom est « xxxx », ou liés à des mots-clés du groupe de mots-clés « yyyy ». Attention, on ne peut pas utiliser plusieurs critères {titre_mot=xxxx} ou {type_mot=yyyy} dans une même boucle.

{id_groupe=zzzz} permet de sélectionner les articles liés à un groupe de mots-clés ; principe identique au {type_mot} précédent, mais puisque l'on travaille avec un identifiant (numéro du

groupe), la syntaxe sera plus « propre ». [Nota : Ce critère n'est pas (en l'état actuel du développement de AGORA) cumulable avec le précédent {type_mot=yyyy}]

{recherche} retourne les articles correspondant aux mots indiqués dans l'interface de recherche (moteur de recherche incorporé à AGORA). Voir la page consacrée au moteur de recherche.

{branche_mot} retourne les articles présents dans l'arborescence du mot courant dans le contexte.

{personnalisation} récupère uniquement les articles ciblant un utilisateur.
{personnalisation=x,y,z} récupère uniquement les articles x,y,z ciblant un utilisateur.

2) Les critères d'affichage

Une fois fixé l'un des critères ci-dessus, on pourra ajouter les critères suivants pour restreindre le nombre d'éléments affichés.

Les critères communs à toutes les boucles s'appliquent évidemment.

{exclus} permet d'exclure du résultat l'article dans lequel on se trouve déjà (par exemple, lorsque l'on affiche les articles contenus dans la même rubrique, on ne veut pas afficher un lien vers l'article dans lequel on se trouver déjà).

{doublons} ou **{unique}** (ces deux critères sont rigoureusement identiques) permettent d'interdire l'affichage d'articles déjà affichés dans d'autres boucles elles-mêmes marquées {doublons}.

3) Les balises de cette boucle

Les balises tirées de la base de données

Les balises suivantes correspondent aux éléments directement tirés de la base de données. Vous pouvez les utiliser également en tant que critère de classement (par exemple : {par date} ou {par titre}).

#ID_ARTICLE affiche l'identifiant unique de l'article. Utile pour fabriquer des liens hypertextes non prévus (par exemple vers une page « Afficher au format impression ».
#SURTITRE retourne le surtitre.

#TITRE retourne le titre de l'article.

#SOUSTITRE retourne le sous titre.

#DESCRIPTIF retourne le descriptif.

#CHAPO retourne le texte d'introduction (chapeau).

#TEXTE retourne le texte principal de l'article.

#PS retourne le post-scriptum.

Les dates : **#DATE**, **#DATE_REDAC**, **#DATE_MODIF** sont explicitées dans la documentation sur « La gestion des dates ».

#ID_RUBRIQUE est l'identifiant de la rubrique dont dépend l'article.

#ID_SECTEUR est l'identifiant du secteur dont dépend l'article (le secteur étant la rubrique située à la racine du site).

#VISITES est le nombre de visites sur cet article.

#POPULARITE donne le pourcentage de popularité de cet article, voir la documentation La « popularité » des articles.

Les balises calculées par AGORA

Les éléments suivants sont calculés par AGORA. (Ils ne peuvent pas être utilisés comme critère de classement.)

#NOTES les notes de bas de page (calculées à partir de l'analyse du texte).

#INTRODUCTION : si l'article contient un descriptif, c'est celui-ci qui est utilisé ici ; sinon, AGORA affiche les 600 premiers caractères du début de l'article (chapeau puis texte).

#LESAUTEURS les auteurs de cet article. Cela permet d'éviter de créer une boucle AUTEURS pour obtenir le même résultat.

#URL_ARTICLE est l'URL de la page de l'article.

#FORMULAIRE_FORUM fabrique l'interface permettant de poster un message répondant à cet article.

#FORMULAIRE_SIGNATURE fabrique l'interface permettant de signer la pétition associée à cet article.

#PARAMETRES_FORUM fabrique la liste des variables exploitées par l'interface du formulaire permettant de répondre à cet article. Par exemple :
[<A HREF="forum.php3?(#PARAMETRES_FORUM)">Répondre à cet article</A>]

Les logos

#LOGO_ARTICLE le logo de l'article, éventuellement avec la gestion du survol.

#LOGO_ARTICLE_RUBRIQUE le logo de l'article, éventuellement remplacé par le logo de la rubrique s'il n'existe pas de logo spécifique à l'article.

#LOGO_RUBRIQUE le logo de la rubrique de l'article.

Les logos s'installent de la manière suivante :

[(#LOGO_ARTICLE|alignement|adresse)]

L'alignement peut être **left** ou **right**. L'adresse est l'URL de destination du lien de ce logo (par exemple **#URL_ARTICLE**). Si l'on n'indique pas d'adresse, le bouton n'est pas cliquable.

Si l'on veut récupérer directement le nom du fichier du logo (alors que les balises précédentes fabriquent le code HTML complet pour insérer l'image dans la page), par exemple pour afficher une image en fond de tableau, on utilisera le filtre |fichier comme suit : **[(#LOGO_ARTICLE|fichier)]**

Par ailleurs deux balises permettent de récupérer un seul des deux logos :
#LOGO_ARTICLE_NORMAL est le logo sans survol ;
#LOGO_ARTICLE_SURVOL est le logo de survol.

La boucle (BREVES)

La boucle BREVES, comme son nom l'indique, retourne une liste de brèves.

<BOUCLEn(BREVES){critères...}>

1) Les critères de sélection

On utilisera l'un ou autre des critères suivants pour indiquer comment on sélectionne les éléments.

{tout} les brèves sont sélectionnées dans l'intégralité du site.

{id_breve} retourne la brève dont l'identifiant est id_breve. Comme l'identifiant de chaque brève est unique, ce critère retourne une ou zéro réponse.

{id_rubrique} retourne toutes les brèves contenues dans la rubrique en cours.

{id_mot} retourne toutes les brèves liées au mot-clé en cours (à l'intérieur d'une boucle de type MOTS).

{titre_mot=xxxx}, ou **{type_mot=yyyy}** retourne les brèves liées au mot-clé dont le nom est « xxxx », ou liées à des mots-clés du groupe de mots-clés « yyyy ». Attention, on ne peut pas utiliser plusieurs critères {titre_mot=xxxx} ou {type_mot=yyyy} dans une même boucle.

{id_groupe=zzzz} permet de sélectionner les brèves liées à un groupe de mots-clés. principe identique au {type_mot} précédent, mais puisque l'on travaille avec un identifiant (numéro du groupe), la syntaxe sera plus « propre ».

{recherche} retourne les brèves correspondant aux mots indiqués dans l'interface de recherche (moteur de recherche incorporé à AGORA). Voir la page consacrée au moteur de recherche.

{personnalisation} récupère uniquement les articles ciblant un utilisateur.
{personnalisation=x,y,z} récupère uniquement les articles x,y,z ciblant un utilisateur.

{branche_mot} retourne les brèves présentes dans l'arborescence du mot courant dans le contexte.

2) Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

3) Les balises de cette boucle

Les balises tirées de la base de données

Les balises suivantes correspondent aux éléments directement tirés de la base de données. Vous pouvez les utiliser également en tant que critère de classement (généralement : {par titre}).
#ID_BREVE affiche l'identifiant unique de la brève.

#TITRE retourne le titre de la brève.

#DATE retourne la date de publication de la brève.

#TEXTE retourne le texte de la brève.

#NOM_SITE le nom du site indiqué en références.

#URL_SITE l'adresse (URL) du site indiqué en références.

#ID_RUBRIQUE l'identifiant de la rubrique dont dépend cette brève.

Les balises calculées par SPIP

Les éléments suivants sont calculés par AGORA. (Ils ne peuvent pas être utilisés comme critère de classement.)

#NOTES les notes de bas de page (calculées à partir de l'analyse du texte).

#INTRODUCTION les 600 premiers caractères du texte, les enrichissements typographiques (gras, italique) sont supprimés.

#URL_BREVE est l'URL de la page de la brève.

#FORMULAIRE_FORUM fabrique l'interface permettant de poster un message répondant à cette brève.

#PARAMETRES_FORUM fabrique la liste des variables exploitées par l'interface du formulaire permettant de répondre à cette brève. Par exemple :
[<A HREF="forum.php3?(#PARAMETRES_FORUM)">Répondre à cette brève</A>]

Le logo

#LOGO_BREVE le logo de la brève, éventuellement avec la gestion du survol.

Le logo s'installe de la manière suivante :

LOGO_BREVE|alignement|adresse)]

#LOGO_BREVE_RUBRIQUE affiche, si il existe, le logo de la brève ; si ce logo n'a pas été attribué, AGORA affiche le logo de la rubrique.

La boucle (AUTEURS)
La boucle AUTEURS, comme son nom l'indique, retourne une liste d'auteurs. Si l'on ne précise pas de critère de sélection, la boucle retournera tous les auteurs ayant un article publié.

<BOUCLEn(AUTEURS){critères...}>

1) Les critères de sélection

On utilisera l'un ou autre des critères suivants pour indiquer comment on sélectionne les éléments.

{tout} les auteurs sont sélectionnés, qu'ils aient écrit un article ou non.

{id_auteur} retourne l'auteur dont l'identifiant est id_auteur. Comme l'identifiant de chaque auteur est unique, ce critère retourne une ou zéro réponse.

{id_article} retourne tous les auteurs de cet article.

2) Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

3) Les balises de cette boucle

Les balises tirées de la base de données

Les balises suivantes correspondent aux éléments directement tirés de la base de données. Vous pouvez les utiliser également en tant que critère de classement (généralement : {par nom}).

#ID_AUTEUR affiche l'identifiant unique de l'auteur.

#NOM retourne le nom de l'auteur.

#BIO retourne la biographie de l'auteur.

#EMAIL retourne son adresse email.

#NOM_SITE le nom de son site Web.

#URL_SITE l'adresse (URL) de son site.

#PGP sa clé publique pour PGP.

#FORMULAIRE_ECRIRE_AUTEUR affiche un formulaire permettant d'écrire à l'auteur. Il faut que le serveur hébergeant le site accepte d'envoyer des mails. Ce système permet de ne pas divulguer l'adresse email de l'auteur.

Les balises calculées par SPIP

#NOTES les notes de bas de page (calculées à partir de l'analyse du texte.

Le logo

#LOGO_AUTEUR le logo de l'auteur, éventuellement avec la gestion du survol.

Le logo s'installe de la manière suivante :

[(#LOGO_AUTEUR|alignement|adresse)]

Les variantes **#LOGO_AUTEUR_NORMAL** et **#LOGO_AUTEUR_SURVOL** permettent un affichage plus fin de ces deux variantes du logo.

La boucle (FORUMS)

La boucle FORUMS retourne une liste de messages de forums.

<BOUCLEn(FORUMS){critères...}>

1) Les critères de sélection

On utilisera l'un ou autre des critères suivants pour indiquer comment on sélectionne les éléments.

{id_forum} retourne le message dont l'identifiant est id_forum. Comme l'identifiant de chaque message est unique, ce critère retourne une ou zéro réponse.

{id_article} retourne les messages correspondant à cet article.

{id_rubrique} retourne les messages correspondant à cette rubrique.

{id_breve} retourne les messages correspondant à cette brève.

{id_parent} retourne les messages dépendant d'un autre message. Indispensable pour gérer des threads dans les forums.

{id_enfant} retourne le message dont dépend le message actuel (permet de « remonter » dans la hiérarchie des threads).

{meme_parent} retourne les autres messages répondant à un même message.

{plat} par défaut, seuls les messages n'ayant pas de parent (i.e. à la racine d'un thread) sont affichés. En ajoutant ce critère, vous pouvez sélectionner tous les messages quelle que soit leur position dans un thread (dans la limite des autres critères, bien sûr). Cela permet d'afficher les messages par ordre strictement chronologique par exemple, ou de compter le nombre total de contributions dans un forum.

{id_secteur} retourne les messages correspondant au secteur. A priori, peu utile ; mais cela permet par exemple de faire un grand forum thématique regroupant tous les messages d'un secteur, quel que soit l'endroit où l'on se trouve.

Les messages des forums peuvent être liées à des mots-clés. Les critères de mots-clés peuvent donc être utilisés dans les boucles (FORUMS) :

{id_mot}, **{titre_mot=xxx}** récupèrent les messages liés au mot dont le numéro est id_mot ou dont le titre est titre_mot ;
{id_groupe}, **{type_mot=yyyy}** récupèrent les messages liés à des mots du groupe id_groupe, ou du groupe dont le titre est type_mot.

{branche_mot} retourne les messages présents dans l'arborescence du mot courant dans le contexte.

{recherche_forum} retourne les messages (titre, texte et auteur) contenant indiqués dans l'interface de recherche (moteur de recherche incorporé à AGORA).

{dateMin} retourne les messages dont la date de postage est supérieure à dateMin. Pour utiliser ce critère il est nécessaire de passer en paramètres HTTP les variables suivantes:
anneeMin
moisMin
jourMin
{dateMax} retourne les messages dont la date de postage est inférieure à dateMax. Pour utiliser ce critère il est nécessaire de passer en paramètres HTTP les variables suivantes:
anneeMax
moisMax
jourMax

{auteur} retourne les messages écrits par cet auteur. Il est possible de passer dans le contexte l'auteur recherché.

2) Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

3) Les balises de cette boucle

Les balises tirées de la base de données
Les balises suivantes correspondent aux éléments directement tirés de la base de données. Vous pouvez les utiliser également en tant que critère de classement (généralement : {par titre}).

#ID_FORUM affiche l'identifiant unique du message.

#ID_BREVE affiche l'identifiant de la brève à laquelle ce message est attaché. Attention, cela n'est pas récursif : un message qui répond à un message attaché à une brève ne contient pas lui-même le numéro de la brève.

#ID_ARTICLE est l'identifiant de l'article auquel répond le message.

#ID_RUBRIQUE l'identifiant de la rubrique à laquelle le message répond.

#DATE est la date de publication.

#TITRE est le titre.

#TEXTE est le texte du message.

#NOM_SITE le nom du site Web indiqué par l'auteur.

#URL_SITE l'adresse (URL) de ce site Web.

#NOM est le nom de l'auteur du message.

#EMAIL est l'adresse email de l'auteur.

#IP est l'adresse IP de l'auteur du message au moment de l'envoi de sa contribution.

#PROFIL_AUTEUR est le profil de l'auteur du message.
Les profils possibles sont les suivants:
« texte_statut_poubelle »
« item_redacteur »
« item_redacteur_en_chef »
« item_webmestre »
« item_administrateur »
« item_non_abonne »

Les balises calculées par AGORA

#FORMULAIRE_FORUM fabrique l'interface permettant de poster un message de réponse.

#PARAMETRES_FORUM fabrique la liste des variables exploitées par l'interface du formulaire permettant de répondre à ce message. Par exemple :
[Répondre à ce message]

#FORMULAIRE_RECHERCHE_FORUM fabrique l'interface permettant d'effectuer une recherche sur les messages (titre, texte et auteur).

La boucle (MOTS)

La boucle MOTS retourne une liste de mots-clés.

<BOUCLEn(MOTS){critères...}>

1) Les critères de sélection

On utilisera l'un ou autre des critères suivants pour indiquer comment on sélectionne les éléments.

{tout} les mots sont sélectionnés dans l'intégralité du site.

{id_mot} retourne le mot-clé dont l'identifiant est id_mot.

{id_groupe} retourne les mots-clés associés au groupe de mots dont le numéro est id_groupe.

{id_article} retourne les mots-clés associés à cet article (c'est l'utilisation la plus courante de cette boucle).

{id_rubrique} retourne les mots-clés associés à une rubrique.

{id_breve} retourne les mots associés à une brève.

{id_syndic} retourne les mots associés à un site référencé.

{id_forum} retourne les mots associés à un message de forum (attention, utilisation très spécifique).

{titre=france} retourne le mot-clé intitulé france (par exemple).

{type=pays} retourne les mots-clés du groupe de mots-clés intitulé pays (par exemple).

{racine} retourne la liste des mots de premier niveau dans l'arborescence de mots clés.

{id_parent} retourne la liste des mots fils d'un mot.

{id_enfant} retourne le mot père d'un mot (une seule réponse ; ou zéro réponse si la présente rubrique est située à la racine du site).

{meme_parent} retourne la liste des rubriques dépendant de la même rubrique que la rubrique en cours. Permet d'afficher les rubriques « soeurs » qui se trouvent au même niveau dans la hiérarchie.

2) Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

3) Les balises de cette boucle

Les balises tirées de la base de données

Les balises suivantes correspondent aux éléments directement tirés de la base de données. Vous pouvez les utiliser également en tant que critère de classement (généralement : {par titre}).

#ID_MOT affiche l'identifiant unique du mot.

#TITRE est le titre (le mot-clé lui-même).

#DESCRIPTIF est le descriptif du mot.

#TEXTE est le texte associé au mot.

#TYPE est la catégorie dans laquelle est installé ce mot-clé (par exemple, le mot-clé « France » pourrait être associé à la catégorie « Pays »).

#LEVEL est le niveau du mot dans l'arborescence

#LOGO_MOT [SPIP 1.4] affiche le logo associé au mot-clé.

#URL_MOT donne l'adresse de ce mot

La boucle (GROUPES_MOTS)

<BOUCLEn(GROUPES_MOTS){critères...}>
Les balises et critères associés à cette boucle sont :

#ID_GROUPE, l'identifiant du groupe de mots [également disponible dans la boucle(MOTS)];
#TITRE, le titre du groupe [à l'intérieur de la boucle(MOTS), vous pouvez utiliser **#TYPE** pour afficher cette valeur].

La boucle (SITES)

La boucle SITES retourne une liste de sites référencés. (Si l'on a syndiqué des sites référencés, cette boucle s'utilise, naturellement, associée à une boucle SYNDIC_ARTICLES qui permet de récupérer la liste des articles de ces sites.)

<BOUCLEn(SITES){critères...}>

1) Les critères de sélection

On utilisera l'un ou autre des critères suivants pour indiquer comment on sélectionne les éléments.

{tout} retourne tous les sites référencés.

{id_syndic} retourne le site référencé dont l'identifiant est id_syndic.

{id_rubrique} retourne les sites référencés dans cette rubrique.

{id_secteur} retourne les sites référencés dans ce secteur.

{id_mot} retourne toutes les sites liés au mot-clé en cours (à l'intérieur d'une boucle de type (MOTS)).

{titre_mot=xxxx}, ou **{type_mot=yyyy}** retourne les sites liés au mot-clé dont le nom est « xxxx », ou liés à des mots-clés du groupe de mots-clés « yyyy ». Attention, on ne peut pas utiliser plusieurs critères {titre_mot=xxxx} ou {type_mot=yyyy} dans une même boucle.

{id_groupe=zzzz} permet de sélectionner les sites liés à un groupe de mots-clés ; principe identique au {type_mot} précédent, mais puisque l'on travaille avec un identifiant (numéro du groupe), la syntaxe sera plus « propre ».

{branche_mot} retourne les sites présents dans la branche de arborescence du mot courant dans le contexte.

2) Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

{moderation=oui} affiche les sites syndiqués dont les liens sont bloqués a priori (« modérés ») ; l'inverse de ce critère est {moderation!=oui}.

{syndication=oui}, **{syndication=non}** permet de n'afficher que les sites référencés faisant l'objet d'une syndication, ou les sites non syndiqués.

{doublons} ou **{unique}** (ces deux critères sont rigoureusement identiques) permettent d'interdire l'affichage de sites référencés déjà affichés dans d'autres boucles.

{personnalisation} récupère uniquement les sites ciblant un utilisateur.
{personnalisation=x,y,z} récupère uniquement les sites x,y,z ciblant un utilisateur.

3) Les balises de cette boucle

Les balises tirées de la base de données

Les balises suivantes correspondent aux éléments directement tirés de la base de données. Vous pouvez les utiliser également en tant que critère de classement (généralement : {par titre}).

#ID_SYNDIC affiche l'identifiant unique du site syndiqué.

#NOM_SITE est le nom du site syndiqué.

#URL_SITE est l'adresse (URL) du site syndiqué.

#DESCRIPTIF est le descriptif du site syndiqué.

#ID_RUBRIQUE est le numéro de la rubrique contenant cette syndication.

#ID_SECTEUR est le numéro de la rubrique secteur (à la racine du site) contenant cette syndication.

Autres balises

#LOGO_SITE affiche le logo attribué au site.

#URL_SYNDIC affiche l'adresse (URL) du fichier de syndication de ce site.

La boucle SYNDIC_ARTICLES retourne une liste des articles des sites syndiqués. On peut soit l'utiliser à l'intérieur d'une boucle SITES (cette dernière récupère une liste de sites référencés, ensuite on récupère chaque article de ces sites), soit directement à l'intérieur d'une rubrique (on récupère directement tous les articles syndiqués dans une rubrique, en court-circuitant le passage par la liste des sites).

<BOUCLEn(SYNDIC_ARTICLES){critères...}>

1) Les critères de sélection

On utilisera l'un ou autre des critères suivants pour indiquer comment on sélectionne les éléments.

{tout} récupère tous les sites syndiqués.

{id_syndic_article} retourne l'article syndiqué dont l'identifiant est id_syndic_article. (Dans la pratique, il y très peu d'intérêt à fabriquer une page pour un article syndiqué, puisqu'on préférera renvoyer directement vers l'article en question.)

{id_syndic} retourne la liste des articles du site syndiqué dont l'identifiant est id_syndic.

{id_rubrique} retourne la liste des articles syndiqués dans cette rubrique.

{id_secteur} retourne la liste des articles syndiqués dans ce secteur.

2) Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

3) Les balises de cette boucle

Les balises tirées de la base de données

Les balises suivantes correspondent aux éléments directement tirés de la base de données. Vous pouvez les utiliser également en tant que critère de classement (généralement : {par titre}).
#ID_SYNDIC_ARTICLE affiche l'identifiant unique de l'article syndiqué.

#ID_SYNDIC affiche l'identifiant unique du site syndiqué contenant cet article. syndiqué.

#TITRE est le titre de l'article.
Remarque : il est préférable d'utiliser ici le titre « brut » de l'article syndiqué - via le code [(#TITRE*)] -, pour éviter le moteur typographique. En effet les titres sont censés être déjà « typographiquement corrects » dans les backends, et on ne souhaite pas passer la correction typographique sur des titres en anglais ou sur des titres comprenant des expressions du genre « Les fichiers ~/.tcshrc ».

#URL_ARTICLE est l'adresse (URL) de l'article syndiqué (sur son site original).

#DATE est la date de publication de cet article.

#LESAUTEURS, les auteurs de l'article syndiqué.

#DESCRIPTIF le descriptif de l'article syndiqué.

#NOM_SITE est le nom du site syndiqué contenant cet article.

#URL_SITE est l'adresse (URL) du site.

La boucle (DOCUMENTS)

La boucle DOCUMENTS retourne une liste de documents multimédia associés (à un article, à une rubrique, éventuellement les images incluses dans une brève).
<BOUCLEn(DOCUMENTS){critères...}>

Cette boucle gère non seulement les documents joints non installés dans le texte d'un article, mais peut aussi accéder aux images, aux vignettes de prévisualisation et aux documents déjà insérés dans le corps de l'article.

Pour mémoire, on utilisera donc le plus fréquemment (utilisation courante) la boucle DOCUMENTS avec, au minimum, les critères suivants (explications ci-après) :
<BOUCLEn(DOCUMENTS){mode=document}{doublons}>

Les critères de sélection
Une boucle DOCUMENTS s'utilise en général à l'intérieur d'un article ou d'une rubrique (éventuellement dans une brève, mais ici l'utilisation sera réservée à la récupération d'images, ce qui sera très spécifique).

{id_article} retourne les documents de l'article dont l'identifiant est id_article.

{id_rubrique} retourne les documents de la rubrique id_rubrique.

{id_breve} retourne les documents de la brève id_breve (il n'est pas possible d'associer des documents multimédia à une brève, seulement des images ; l'utilisation d'une boucle DOCUMENTS dans ce cadre sera donc très spécifique).

Notez bien : il n'est pas possible d'utiliser ici le critère {id_secteur} ; les documents sont conçus pour être intimement liés aux articles et aux rubriques, et non à être appelés seuls sans ces éléments (on parle dans SPIP de « documents joints »).

1) Les critères d'affichage

{mode=document} ou **{mode=image}** permet d'indiquer si l'on veut appeler les documents multimédia, ou les images (en effet, désormais les images associées à l'article et éventuellement insérées dans l'article sont traités comme des documents en mode=image).

{doublons} prend ici une importance particulière : elle permet non seulement de ne pas réafficher des documents déjà affichés par une autre boucle, mais également de ne pas réafficher les documents déjà intégrés à l'intérieur d'un article. Si l'on oublie ce critère, on affichera tous les documents associés à un article, y compris ceux qui auraient déjà été affichés à l'intérieur du texte.

{extension=...} permet de sélectionner les documents selon leur terminaison (terminaison du fichier multimédia, par exemple « mov », « ra », « avi »...). Cela peut être utilisé par exemple pour réaliser un porfolio, c'est-à-dire une boucle n'affichant que les documents de type image, une seconde boucle ensuite, avec une présentation graphique différente, les autres types de documents :

<BOUCLE_portfolio(DOCUMENTS){id_article}{extension==jpg|png|gif}{mode=document}{doublons}>

Cette BOUCLE_portfolio récupère les documents joints à un article, non déjà affichés dans le texte de l'article, et donc les extensions des fichiers peuvent être « jpg », « png » ou « gif ».

2) Les balises

#LOGO_DOCUMENT affiche le logo (vignette de prévisualisation) associé à cet article ; si une vignette personnalisée n'a pas été installée manuellement par l'auteur de l'article, AGORA utilise une vignette standard selon le type du fichier.

#URL_DOCUMENT est l'URL du fichier multimédia. Pour afficher une vignette cliquable pointant vers le document multimédia, on utilisera donc le code suivant :
[(#LOGO_DOCUMENT|#URL_DOCUMENT)]

#TITRE affiche le titre du document.

#DESCRIPTIF affiche le descriptif du document.

#TYPE_DOCUMENT affiche le type (fichier Quicktime, fichier Real...) du document multimédia.

#TAILLE affiche la taille du fichier multimédia. Ce chiffre est fourni en octets. Pour de gros fichiers, cette valeur devient rapidement inutilisable ; on pourra donc lui appliquer le filtre taille_en_octets, qui affichera successivement en octets, en kilooctets, ou même en mégaoctets :
[(#TAILLE|taille_en_octets)]

#LARGEUR et **#HAUTEUR** fournissent les dimensions en pixels.

#ID_DOCUMENT affiche le numéro du document.

#EMBED_DOCUMENT est une balise à l'utilisation très spécifique : elle permet d'inclure directement les fichiers de formats autorisés (vidéo, sons) directement dans la page Web ; il faut éviter d'utiliser systématiquement cette balise, car il est déconseillé d'insérer systématiquement les documents dans les pages sans un contrôle strict (sauf à faire exploser la durée de chargement de vos pages Web...). La balise peut être complétée de paramètres propres aux formats utilisés (encore une fois : utilisation très spécifique), par exemple :
[(#EMBED_DOCUMENT|autostart=true)]

La boucle (SIGNATURES)

La boucle SIGNATURES retourne une liste de signataires d'une pétition associée à un article.

<BOUCLEn(SIGNATURES){critères...}>

1) Les critères de sélection

On utilisera l'un ou autre des critères suivants pour indiquer comment on sélectionne les éléments.
{tout} toutes les signatures sont sélectionnées dans l'intégralité du site.

{id_signature} retourne la signature correspondant à l'identifiant courant.

{id_article} retourne les signatures de la pétition de cet article.

2) Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

Attention. Dans ce type de boucles, certains critères de classement de la boucle ne sont pas identiques aux balises AGORA indiquées ci-dessous :

{par nom_email} classe les résultats selon le #NOM du signataire.

{par ad_email} classe selon l'#EMAIL du signataire.

3) Les balises de cette boucle

Les balises tirées de la base de données

Les balises suivantes correspondent aux éléments directement tirés de la base de données. Vous pouvez les utiliser également en tant que critère de classement (généralement : {par titre}).
#ID_SIGNATURE affiche l'identifiant unique du message.

#ID_ARTICLE est l'identifiant de l'article pour cette pétition.

#DATE est la date de publication.

#MESSAGE est le texte du message.

#NOM est le nom de l'auteur du message.

#EMAIL est l'adresse email de l'auteur.

#NOM_SITE le nom du site Web indiqué par l'auteur.

#URL_SITE l'adresse (URL) de ce site Web.

La boucle (SONDAGES)

La boucle SONDAGES retourne une liste des sondages attachés à un article.

<BOUCLEn(SIGNATURES){critères...}>

1) Les critères de sélection

On utilisera l'un ou autre des critères suivants pour indiquer comment on sélectionne les éléments.

{id_article} retourne le sondage associé à un article.

2) Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

{par vote} classe les résultats par vote.

3) Les balises de cette boucle

Les balises tirées de la base de données

Les balises suivantes correspondent aux éléments directement tirés de la base de données.

#FORMULAIRE_SONDAGE fabrique le formulaire de réponse au sondage.

#TEXTE_REPONSE fournit le texte de la réponse.

#VOTES_NOMBRE fournit le nombre de vote pour une réponse.

#VOTES_TOTAL fournit le nombre de vote total.

#VOTES_POURCENTAGE fournit le pourcentage de réponse pour une question.