Manuel de « référence »

Version hors-ligne du site Spip.net

Tome 3a

Guide du webmestre et du bidouilleur

SPIP pas à pas

Boucles et balises: manuel de référence

24 septembre 2007 Rabillard Fabien

SPIP est un système de publication pour l'Internet qui s'attache particulièrement au fonctionnement collectif, au multilinguisme et à la facilité d'emploi. C'est un logiciel libre, distribué sous la licence GNU/GPL. Il peut ainsi être utilisé pour tout site Internet, qu'il soit associatif ou institutionnel, personnel ou marchand.

SPIP est développé (programmé, documenté, traduit, etc.) et utilisé par une communauté de personnes que chacun est invité à rejoindre (ou simplement à contacter) sur différents sites Web, listes de discussion par email et rencontres (les fameux « Apéros-SPIP »). Le programme est né en 2001 d'une initiative du minirézo, un collectif défendant le Web indépendant et la liberté d'expression sur Internet. Il est actuellement utilisé sur des dizaines de milliers de sites très divers. Ce site est la documentation officielle.

Le document que vous avez entre les mains ou sur votre écran est un copier/coller du site Spip.net effectué entre le 15 et le 24 septembre 2007.
Ce document est destiné a ceux qui n'ont pas de connexion internet ou qui préfèrent avoir une version papier. Mais si vous voulez avoir la dernière version en ligne et à jour rendez vous sur le site Spip.net.

Je n'est fait aucune modification ou correction par rapport au site, j'ai juste revu la mise en page afin d'adapter le site sur « papier ».

Ce document a été réalisé avec OpenOffice.org 2.2.1 et la police « Libération » de RedHat.

Bonne lecture. Fabien.

P.S.: La plupart des liens hypertexte ont été laissés afin d'avoir accès aux articles en lignes.

Guide du webmestre et du bidouilleur

Guide du webmestre et du bidouilleur

Où placer les fichiers de squelettes ?

Dans le dossier squelettes/ :-)

Depuis [SPIP 1.8], les squelettes sont rangés dans un dossier dédié, nommé dist/. Le dossier squelettes/ accueillera vos squelettes personnalisés.

Les avantages de ce rangement sont évidents : meilleure séparation du code de SPIP et de la structure du site, possibilité de changer tout un ensemble de squelettes d'un seul coup, etc.

> *Historique :* Dans les versions antérieures à [SPIP 1.8], les fichiers de squelettes fournis dans la distribution de SPIP étaient placés à la racine. Voir : « Qu'est-ce que les fichiers « dist » ? ».

Les squelettes par défaut : dist/

Depuis [SPIP 1.8], les squelettes de la distribution — c'est-à-dire ceux fournis en standard à l'installation de SPIP — sont regroupés dans le répertoire dist/. Ces fichiers contiennent les informations sur la mise en page par défaut du site et ne doivent pas être modifiés. Vous pouvez examiner le contenu de ce répertoire et partir de ce jeu de squelettes pour adapter la mise en page à vos besoins [1].

Toutefois, **il ne faut surtout pas modifier les fichiers du répertoire** dist/, sinon vous risqueriez de perdre toutes vos modifications à chaque mise à jour de SPIP !
Pour éviter cela, **faites une copie des fichiers que vous souhaitez modifier**, et placez-les dans un autre répertoire, comme indiqué ci-après.

Votre dossier squelettes/

Depuis [SPIP 1.8], les squelettes personnalisés doivent être rangés dans un répertoire nommé squelettes/ (attention au « s » final !), que vous créerez à la racine de votre site SPIP. Que vous souhaitiez installer un jeu complet de squelettes (pris sur SPIP - Contrib ou ailleurs), ou apporter une légère modification aux squelettes par défaut, placez vos squelettes dans ce répertoire.

Ainsi, un utilisateur qui veut créer sa propre mise en page, développera ses propres fichiers article.html, rubrique.html, etc. dans le répertoire squelettes/. Notez bien qu'il n'est pas indispensable de placer un jeu de squelettes complet dans ce répertoire.

Pour afficher les pages du site, SPIP cherche les squelettes prioritairement dans le dossier squelettes/ ; si SPIP n'y trouve pas un fichier .html qui lui est nécessaire, il ira chercher celui de la distribution dans le dossier dist/.
Ainsi, si vous n'avez placé qu'un seul fichier dans le dossier squelettes, par exemple article.html, SPIP utilisera ce squelette pour afficher les articles, et ceux de la dist pour toutes les autres pages du site.

Le dossier squelettes/ **est destiné à recevoir tous les fichiers nécessaires à la mise en page d'un site**. On y rangera donc :

- les squelettes, c'est-à-dire les fichiers .html avec du code SPIP ;
- les fichiers inclus dans les squelettes (ainsi que, pour les versions antérieures à SPIP 1.9, leur fichier php3 correspondant) et les modèles (depuis SPIP 1.9) ;
- les formulaires modifiés, de préférence dans un sous-répertoire formulaires/
- les feuilles de style CSS qui produisent l'habillage graphique ; les personnaliser permet en effet de varier, parfois spectaculairement, l'habillage d'un site sans intervenir dans les squelettes. Voir : « Mettez-y votre style ! » ;
- les images utilisées dans les squelettes ;
- le fichier mes_fonctions.php contenant les filtres et variables de personnalisation propres à ce jeu de

squelettes ;
- les fichiers javascripts ;
- les fichiers de langue personnalisés (cf. : « Internationaliser les squelettes », méthode des fichiers de langues), de préférence dans un sous-répertoire lang/ ;
- etc...

Utiliser un autre dossier de squelettes

Depuis SPIP 1.5 il est possible de ranger les squelettes dans un répertoire portant le nom de votre choix, en le déclarant dans le fichier ecrire/mes_options.php, avec la variable de personnalisation $dossier_squelettes, comme expliqué dans la documentation correspondante. SPIP ira chercher les squelettes en priorité dans le répertoire ainsi déclaré.

Ceci vous permet, par exemple, d'essayer un nouveau jeu de squelettes sans écraser l'ancien, ou de gérer dynamiquement plusieurs jeux de squelettes.

Priorité des dossiers de squelettes

Soyons plus exhaustifs et résumons. Grosso modo, lorsque SPIP doit utiliser un fichier, il le cherche dans différents répertoires dans l'ordre suivant :

1. en premier lieu dans liste de dossiers désignés dans variable $dossier_squelettes, si celle-ci est définie ;
2. ensuite dans le dossier squelettes/ situé à a racine du site ;
3. puis (depuis SPIP 1.9) dans la liste de dossiers de la variable $plugins ;
4. ensuite à la racine du site ;
5. dans le répertoire dist/ ;
6. et enfin dans le répertoire ecrire/.

« Grosso modo », car à cela s'ajoutent quelques subtilités [2], dont un ordre de priorité par fichier de squelette, qui permet des variantes plus fines : par rubrique, par branche ou par langue, comme expliqué dans la documentation correspondante.

> *Remarque :* Depuis SPIP 1.9 : En fait, le mécanisme décrit ci-dessus pour choisir l'emplacement d'un fichier ne s'applique pas seulement aux squelettes, mais aussi à l'ensemble du code de SPIP. Dans le jargon des développeurs on parle de « surcharger du code », l'ordre de choix des dossiers étant dans le « SPIP_PATH ». Cela met en place le cadre et les normes pour le développement des « plug-ins », extensions des fonctionnalités de SPIP que tout un chacun dans la communauté peut apporter.
>
> Ainsi, on peut modifier à souhait n'importe quelle caractéristique du comportement de SPIP sans pour cela se priver à l'avenir des évolutions de la distribution et du support de la communauté. SPIP est devenu modulaire !
>
> *Historique :* Le fait que SPIP recherche des squelettes à la racine est historique, car c'était le premier endroit où ils étaient placés. Cela avait l'avantage de rendre les squelettes « visibles » dans un navigateur, puisque les liens aux .css et autres habillages graphiques étaient forcément saisis « en dur » depuis la racine.
> De plus, jusqu'à SPIP 1.8.3, SPIP cherchait les squelettes à la racine *avant* le dossier squelettes/.

Et les fichiers .php3 (ou .php) dans tout ça ?

Rappelons avant tout qu'à partir de SPIP 1.9, il n'y a plus de fichier .php3 (ou .php) pour les squelettes : SPIP calcule toutes ses pages à partir du script unique spip.php. Tout ce qui suit est donc historique.

> Dans [SPIP 1.8.2] et [SPIP 1.8.3], il existait un fichier **page.php3** qui préfigurait le spip.php et la forme d'inclusion de SPIP 1.9. En effet, **page.php3** permettait, à lui seul, d'appeler n'importe quel squelette en passant en paramètre la variable de fond, c'est-à-dire le fichier squelette .html à

utiliser :

```
www.monsite.org/page.php3?fond=mon_squelette&...
```

Le plus simple était alors d'utiliser un appel de ce type [3] pour tout squelette autre que ceux des objets de base (article, breve, rubrique, sommaire,...) pour lesquels un fichier .php3 existe à la racine, ce qui permettait l'appel habituel de la forme :

```
www.monsite.org/article.php3?...
```

Jusqu'à [SPIP 1.8.1] il fallait créer un fichier .php3, qui soit le pendant de votre .html dans l'ancienne structure des squelettes SPIP, celui-ci devait forcément être placé à la racine du site [4].

Notes

[1] C'est même une méthode vivement conseillée, car ce jeu de squelettes a été pensé pour être aussi modulaire que possible.

[2] Comme une nomenclature des fichiers en fonction de leur rôle, ce qui fait, par exemple, que SPIP ira chercher les fichiers de langue dans un sous répertoire lang/, comme nous l'avons vu plus haut.

[3] Il est à noter que cette méthode fonctionne aussi pour les appels avec la balise <INCLURE(page.php3){fond=inc-entete}>.

[4] Sauf, éventuellement, ceux qu'on n'utilisait que dans un appel <INCLURE(squel.php3)> et pas directement par une URL. Dans ce cas, le fichier squel.php3 pouvait *aussi* être déplacé dans le dossier de squelettes.

Qu’est-ce que les fichiers « dist » ?

Vous avez ...dist ?

Que sont les fichiers se trouvant dans le répertoire **dist** ou ayant -dist.html dans leur nom ? Ce sont les fichiers de la « distribution » de SPIP.

> **Attention :** Cet article est considéré comme une archive historique valable de SPIP 1.3 jusqu’à SPIP 1.7, SPIP 1.7.2 . Pour les versions plus récentes, consultez : « Où placer les fichiers de squelettes ? ».

Comme vous le savez sûrement déjà (sinon, il vous faudra lire le tutorial, puis parcourez le manuel de référence), le système de squelettes repose sur des fichiers .html contenant la présentation graphique du site. Ainsi, « article.html » met en forme les articles du site, et « rubrique.html » met en forme les rubriques, etc.

Or, nous avons remarqué que fréquemment, les utilisateurs qui manipulaient leur site public en modifiant ces fichiers .html fournis avec SPIP rencontraient des problèmes lors des mises à jour, s’ils n’avaient pas pris leurs précautions en sauvegardant les fichiers modifiés.

En effet, en réinstallant tous les nouveaux fichiers livrés avec SPIP, ils écrasaient purement et simplement leurs fichiers modifiés (oubliant de faire une copie de sauvegarde de leurs modifications).

Depuis SPIP 1.3 et jusqu’à SPIP 1.7, SPIP 1.7.2 , les fichiers de squelettes fournis dans la distribution de SPIP sont nommés « article-dist.html », « rubrique-dist.html », et ainsi de suite. Pour personnaliser ces fichiers, il suffit de les renommer d’abord « article.html », « rubrique.html », etc. (sans le -dist final).

> *Attention :* Depuis [SPIP 1.8], les fichiers .html sont mieux rangés. Un répertoire **« dist »** est destiné aux fichiers fournis avec la distribution de spip. Voir : « Où placer les fichiers de squelettes ? ».

Ainsi, à la prochaine mise-à-jour, seuls les fichiers « -dist.html » seront écrasés et le webmestre ne perdra pas ses personnalisations. Une petite amélioration, mais qui évite bien des déboires.

Pour aller plus loin

Voici l’ordre (par priorité décroissante) dans lequel sont utilisés les fichiers de squelettes selon leur nom :

- rubrique=10.html : si ce fichier existe, il ne s’applique qu’à la rubrique numéro 10 ;
- si ce fichier n’existe pas, SPIP regarde si il n’y a pas un fichier rubrique-10.html, si ce fichier existe, la rubrique 10 ainsi que ses sous-rubriques l’utilisent, c’est donc « récursif » ;

Note : pour que ces fichiers soient pris en compte il faut que le fichier par défaut (rubrique.html) se trouve dans le même répertoire.

- si ce fichier n’existe pas, SPIP regarde s’il n’y a pas un fichier rubrique.html, qui s’applique à toutes les rubriques du site qui ne sont pas concernées par les fichiers indiqués ci-dessus ;
- jusqu’à [SPIP 1.7.2] : si ce fichier n’existe pas, SPIP utilise alors le fichier rubrique-dist.html qui est le fichier fourni par défaut. Si vous voulez modifier ce fichier, renommez-le en rubrique.html, de façon à ne pas écraser vos modifications à la prochaine mise à jour de SPIP.

Note : Si les squelettes sont rangés dans un sous-répertoire dédié (depuis SPIP 1.5), SPIP recherche prioritairement les squelettes dont il a besoin dans ce répertoire, sinon à la racine, comme expliqué ci-dessus. L’article sur les variables de personnalisation explique comment procéder pour ranger tous les squelettes du site dans un sous-dossier.

FAQ webmestre

Les bases

1. Comment fais-je pour modifier la mise en page du site public ?

La gestion de la mise en page s'appuie sur des fichiers à l'extension .html appelés *squelettes* de mise en page. Leur rôle correspond grosso modo à ce que d'autres logiciels nomment « modèles » « gabarits », ou en anglais, « *templates* ».

Chaque fichier est associé à un type de page différent : ainsi un *squelette* pour le sommaire, un pour l'affichage des articles, un pour l'affichage des rubriques, etc. Un squelette contient du HTML standard définissant l'habillage de la page, dans lequel on insère des « codes » spécifiques à SPIP afin de définir quelles informations vont venir « habiter » cet habillage.

Le langage des squelettes de SPIP est très souple et permet de réaliser des mises en page très variées : un simple coup d'oeil à uZine, Vacarme, Hacktivist News Service ainsi que les sites enregistrés par leurs créateurs sur cette page saura vous en convaincre. Il est donc dommage de garder la mise en page d'origine, même si celle-ci est très utile pour se mettre le pied à l'étrier.

2. Est-il possible d'écrire ces squelettes soi-même ?

Oui, c'est un des intérêts majeurs de SPIP. Pour cela allez voir :

- le tutorial, pour comprendre les bases de la programmation des squelettes.
- le manuel de référence, qui liste toutes les possibilités de programmation.

3. Je ne sais pas / ne veux pas apprendre à programmer. Peut-on utiliser des mises en pages déjà existantes ?

Oui. En dehors de la mise en page par défaut, d'autres jeux de squelettes sont disponibles sur le site des contributions à SPIP, dans la rubrique « Squelettes ».

Il suffit en général de récupérer l'archive voulue (le fichier au format .zip ou .tar.gz, au choix), de la décompresser chez vous, et de transférer son contenu par FTP à la racine de votre site SPIP. Vous pouvez faire une sauvegarde de vos fichiers .html actuels, au cas où vous voulez revenir en arrière.

4. Il n'y a pas beaucoup de jeux de squelettes disponibles. Pourquoi ?

Ces jeux de squelettes sont alimentés par les webmestres SPIP qui nous fournissent leurs créations. Nous comptons donc sur les webmestres pour compléter cette base de squelettes afin d'encourager l'entraide et la richesse des sites SPIP. (cf. section « Partager » plus bas dans cette FAQ)

Créer ses squelettes

1. Peut-on utiliser un éditeur textuel pour créer et modifier ses squelettes ?

Oui, comme on le ferait pour du HTML classique.

2. Peut-on utiliser un éditeur graphique (WYSIWYG) pour créer et modifier ses squelettes ?

Oui, comme on le ferait pour du HTML classique. Voir cependant la question suivante.

3. J'essaie d'utiliser un éditeur graphique pour créer mes pages, mais il modifie les tags SPIP. Peut-on résoudre ce problème ?

Certains éditeurs graphiques « corrigent » automatiquement les tags qu'ils ne comprennent pas. La plupart ont toutefois une option permettant de désactiver cette fonctionnalité. Nous avons consacré un article spécifique à DreamWeaver, mais la démarche est équivalente pour les autres éditeurs (GoLive...).

Partager vos créations

1. J’ai écrit des squelettes pour mon site. Comment fais-je pour qu’ils soient disponibles à tous ?

N’hésitez pas à vous inscrire sur le site SPIP-Contrib mentionné plus haut, afin de proposer vos squelettes au téléchargement et que d’autres puissent à leur tour s’en inspirer pour créer leur propre site.

Liens utiles

- **SPIP - Contrib**

  L’espace des contributions externes, qui recense l’ensemble des fonctionnalités supplémentaires mises à disposition à la communauté par les utilisateurs de SPIP. À ajouter sous forme de plugins, de scripts, de filtres, de thèmes graphiques, de modèles ou de boucles à inclure dans vos squelettes.

Principe général

Tout le contenu d'un site géré sous SPIP est installé dans une base de données MySQL. Pour présenter ces informations aux visiteurs du site, il faut donc réaliser l'opération qui consiste à lire les informations, à les organiser et à les mettre en page, afin d'afficher une page HTML dans le navigateur Web.

A moins d'utiliser un gestionnaire de contenu avancé comme SPIP, cette opération est assez fastidieuse :

- il faut connaître la programmation PHP et MySQL, et écrire des « routines » relativement complexes ;
- l'intégration de telles routines dans une mise en page HTML élaborée est assez pénible ;
- il faut développer toute une interface pour que les utilisateurs autorisés modifient le contenu de la base de données ;
- il faut prendre en compte des problèmes de performances : le recours systématique à du code MySQL et PHP est gourmand en ressources, ralentit la visite et, dans des cas extrêmes, provoque des plantages du serveur Web.

SPIP propose une solution complète pour contourner ces difficultés :

- la mise en page du site est effectuée au moyen de gabarits au format HTML nommés *squelettes*, contenant des instructions simplifiées permettant d'indiquer où et comment se placent les informations tirées de la base de données dans la page ;
- un système de cache permet de stocker chaque page et ainsi d'éviter de provoquer des appels à la base de données à chaque visite. Non seulement la charge sur le serveur est réduite, la vitesse très largement accélérée, de plus un site sous SPIP reste consultable même lorsque la base MySQL est plantée ;
- un « espace privé » permettant aux administrateurs et aux rédacteurs de gérer l'ensemble des données du site.

Pour chaque type de page, un squelette

L'intérêt (et la limite) d'un système de publication automatisé, est de définir un gabarit pour, par exemple, tous les articles. On indique dans ce gabarit (le squelette) où viendront se placer, par exemple, le titre, le texte, les liens de navigation... de l'article, et le système fabriquera chaque page individuelle d'article en plaçant automatiquement ces éléments tirés de la base de données, dans la mise en page conçue par la/le webmestre.

Ce système automatisé permet donc une présentation cohérente à l'intérieur d'un site... Et c'est aussi sa limite : il ne permet pas de définir une interface différente pour chaque page isolée (on verra plus loin que SPIP autorise cependant une certaine souplesse).

Lorsque vous installez SPIP, un jeu de squelettes est proposé par défaut. Il se trouve dans le répertoire **dist/**, à la racine de votre site. Dès que vous modifiez ces fichiers pour les adapter à vos besoins, ou si vous voulez installer un autre jeu de squelettes, il convient de créer un répertoire nommé **squelettes/** au même niveau. Pour plus de détails sur cette question, lire l'article Où placer ses squelettes .

Lorsqu'un visiteur demande la page http://exemple.org/spip.php?article3437, SPIP va chercher un squelette nommé « article.html ». SPIP se base donc sur l'adresse URL de la page pour déterminer le squelette à utiliser :

L'URL	appellera le squelette
spip.php?article3437	article.html
spip.php?rubrique143	rubrique.html
spip.php?mot12	mot.html
spip.php?auteur5	auteur.html
spip.php?site364	site.html

Avec deux cas particuliers :

- L'URL spip.php appelle le squelette sommaire.html. Il s'agit de la page d'accueil du site.
- L'URL spip.php?page=abcd appelle le squelette abcd.html. En d'autres termes, vous pouvez créer des squelettes qui ne sont pas prévus par le système et les nommer comme vous le souhaitez.

Cette syntaxe sert également pour les pages telles que le plan du site ou les résultats de recherche par exemple : spip.php?page=plan, spip.php?page=recherche&recherche=ecureuil.

Lorsque SPIP appelle un squelette, il lui passe un contexte

Par ailleurs, vous aurez constaté que l'URL fournit d'autres éléments que le nom du squelette. Par exemple, dans spip.php?article3437, le numéro de l'article demandé (3437) ; dans spip.php?page=recherche&recherche=ecureuil, le mot recherché (ecureuil).

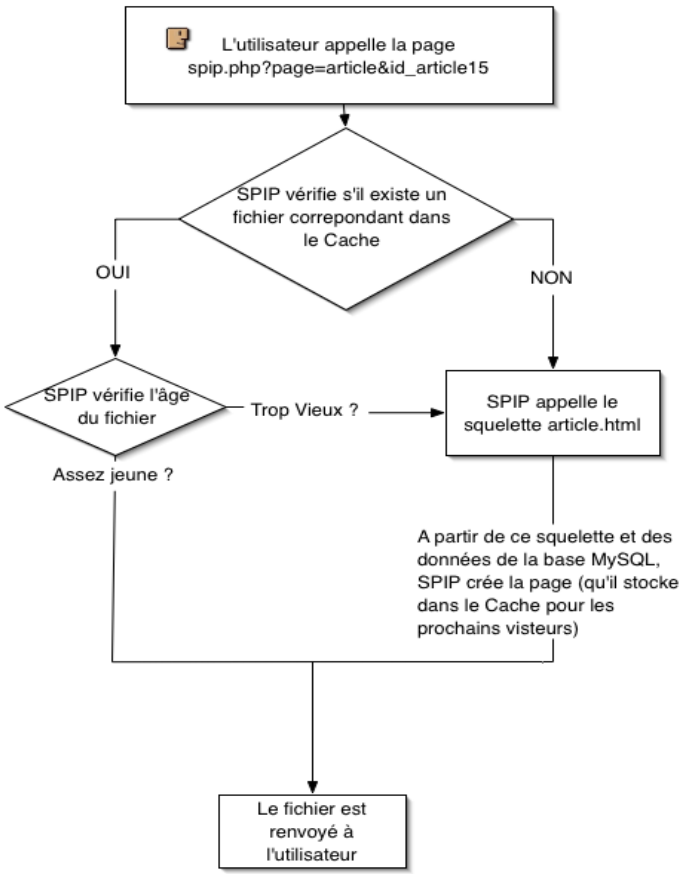
Il s'agit d'un « contexte », c'est-à-dire, une ou plusieurs « variables d'environnement », que SPIP va fournir au squelette pour qu'elles puissent être utilisées dans la composition de la page. En effet, le squelette article.html a besoin de connaître le numéro de l'article demandé pour rechercher son titre, son texte,... dans la base de données. De même, le squelette recherche.html doit connaitre les mots recherchés par le visiteur pour trouver les enregistrements de la base de données qui contiennent ces termes.

Dans toute URL, les variables d'environnement apparaissent après le « ? ». Lorsqu'il y en a plusieurs, elles sont séparées par des « & ».

Dans l'URL spip.php?page=recherche&recherche=ecureuil, on a donc deux variables : page et recherche, auxquelles on attribue les valeurs respectives « recherche » et « écureuil ».

Dans le cas de spip.php?article3437, SPIP a simplifié l'URL qui correspond en fait à : spip.php?page=article&id_article=3437 (si si, vous pouvez essayer !). On a donc ici aussi deux variables : page a la valeur "article" et id_article a la valeur "3437". Ces variables permettent à SPIP d'utiliser les données de l'article 3437 dans le squelette article.html.

Le principe de fonctionnement du cache, de manière simplifiée



SPIP regarde si une page correspondante à l'URL demandée se situe dans le **CACHE**

1. Si la page existe, SPIP vérifie qu'elle n'est pas trop ancienne.
    1. Si la page est trop ancienne, il la recalcule à partir du squelette et de la base MySQL. Puis il la stocke dans le **CACHE**, avant de l'envoyer à l'utilisateur.
    2. Si la page est assez récente, il la retourne à l'utilisateur.
2. Si la page n'existe pas, il le calcule à partir du squelette et de la base MySQL. Puis il la stocke dans le **CACHE**, avant de l'envoyée à l'utilisateur.

Lors d'une visite suivante, si le délai entre les deux visites est suffisamment court, c'est donc cette nouvelle page stockée dans /CACHE qui est retournée, sans avoir à faire un nouveau calcul à partir de la base de données. En cas de plantage de la base de données, c'est forcément la page en cache qui est retournée, même si elle est « trop âgée ».

> *Remarque.* On voit ici que chaque page du site est mise en cache individuellement, et chaque recalcul est provoqué par les visites du site. Il n'y a pas, en particulier, un recalcul de toutes les pages du site d'un seul coup à échéance régulière (ce genre de « grosse manœuvres » ayant le bon goût de surcharger le serveur et de le faire parfois planter).

Par défaut, une page est considérée comme trop vieille au bout de 3600 secondes. [1]

Le fichier .html

Dans SPIP, nous appelons les fichiers .html les **squelettes**. Ce sont eux qui décrivent l'interface graphique de vos pages.

Ces fichiers sont rédigés directement en HTML, auquel on ajoute des instructions permettant d'indiquer à SPIP où il devra placer les éléments tirés de la base de données (du genre : « placer le titre ici », « indiquer à cet endroit la liste des articles portant sur le même thème »...).

Les instructions de placement des éléments sont rédigées dans un langage spécifique, qui fait l'objet du présent manuel d'utilisation. Ce langage constitue par ailleurs la seule difficulté de SPIP.

« Encore un langage ? » Hé oui, il va vous falloir apprendre un nouveau langage. Il n'est cependant pas très compliqué, et il permet de créer des interfaces complexes très rapidement. Par rapport au couple PHP/MySQL, vous verrez qu'il vous fait gagner un temps fou (surtout : il est beaucoup plus simple). C'est un *markup language*, c'est-à-dire un langage utilisant des balises similaires à celles du HTML.

> *Remarque.* De la même façon que l'on apprend le HTML en s'inspirant du code source des sites que l'on visite, vous pouvez vous inspirer des squelettes utilisés sur d'autres sites fonctionnant sous SPIP. Il suffit d'aller chercher le fichier « .html » correspondant. Par exemple, vous pouvez voir le squelette des articles de ce présent cycle (visualisez le code source pour obtenir le texte du squelette).

Une interface différente dans le même site

En plus de la mise en page par défaut des différents contenus du site (rubrique.html, article.html, etc.), vous pouvez créer des squelettes différents pour des rubriques particulières du site et leurs contenus. Il suffit de créer de nouveaux fichiers .html et de les nommer selon le principe suivant :

> - Une interface différente pour une rubrique et ses contenus (sous-rubriques, articles, brèves, etc.) : il faut compléter le nom du fichier squelette correspondant par « -numéro » (un tiret suivi d'un numéro de rubrique).

Par exemple, si vous créez un fichier rubrique-60.html, il s'appliquera à la rubrique n°60 et à ses sous-rubriques en lieu et place de rubrique.html. Le squelette article-60.html s'appliquera aux articles de la rubrique n°60. Si cette rubrique contient des sous-rubriques, leurs articles adopteront également le nouveau squelette. Idem pour breve-60.html... et ainsi de suite.

> - Une interface pour une seule rubrique. (SPIP 1.3) On peut créer une interface qui s'applique à une rubrique, mais pas à ses sous-rubriques. Pour cela, il faut compléter le nom du fichier

squelette correspondant par « =numéro ».

Par exemple, il faut créer un fichier rubrique=60.html, qui s'appliquera uniquement à la rubrique n°60, mais pas à ses sous-rubriques. Idem pour article=60.html, breve=60.html, etc. Ces squelettes s'appliquent aux contenus de la rubrique n°60 mais ceux de ses sous-rubriques retrouvent l'habillage par défaut.

> Notez bien : le numéro indiqué est celui d'une rubrique. Les squelettes article-60.html ou article=60.html ne concernent donc pas l'article n°60 mais bien tous les articles de la rubrique n °60.

Que peut-on mettre dans un fichier .HTML

Les fichiers .html sont essentiellement des fichiers « texte », complétés d'instructions de placement des éléments de la base de données.

SPIP analyse uniquement les instructions de placement des éléments de la base de données (codées selon le langage spécifique de SPIP) ; il se contrefiche de ce qui est placé dans ce fichier et qui ne correspond pas à ces instructions.

Leur contenu essentiel est donc du HTML. Vous déterminez la mise en page, la version du HTML désiré, etc. Vous pouvez évidemment y inclure des feuilles de style (CSS), mais également du JavaScript, du Flash... en gros : tout ce qu'on place habituellement dans une page Web.

Mais vous pouvez également (tout cela n'est jamais que du texte) créer du XML (par exemple, « backend.html » génère du XML).

Plus original : toutes les pages retournées au visiteur sont tirées du **CACHE** par une page écrite en PHP. Vous pouvez donc inclure dans vos squelettes des instructions en PHP, elles seront exécutées lors de la visite. Utilisée de manière assez fine, cette possibilité apporte une grande souplesse à SPIP, que vous pouvez ainsi compléter (par exemple ajouter un compteur, etc.), ou même faire évoluer certains éléments de mise en page en fonction des informations tirées de la base de données.

P.-S.

Cet article n'est valable qu'à partir de SPIP 1.9. Pour les versions antérieures, lire la version archivée.

Notes

[1] Une fois que vous aurez compris le langage SPIP, vous pourrez modifier ce délai en vous servant de la balise #CACHE.

Principe général

version archivée

Cet article n'est valable que jusqu'à SPIP 1.8.3 et obsolète à partir de SPIP 1.9. Pour les versions récentes de Spip, lire ici.

Sa lecture tel quel n'en est pas pour autant inutile. Il faut retenir ce qui est dit pour les fichiers .html, les fichiers .php3 disparaissent et les URL changent.

Tout le contenu d'un site géré sous SPIP est stocké dans une base de données mySQL. Pour présenter ces informations aux visiteurs du site, il faut donc réaliser l'opération qui consiste à lire les informations, à les organiser et à les mettre en page, afin d'afficher une page HTML dans le navigateur Web.

Cette opération est traditionnellement assez pénible :

- il faut connaître la programmation PHP et MySQL, et écrire des « routines » relativement complexes ;
- l'intégration de telles routines dans une mise en page HTML élaborée est assez pénible ;
- il faut prendre en compte des problèmes de performances : le recours systématique à du code mySQL et PHP est gourmand en ressources, ralentit la visite et, dans des cas extrêmes, provoque des plantages du serveur Web.

SPIP propose une solution complète pour contourner ces difficultés :

- la mise en page du site est effectuée au moyen de pages HTML nommées *squelettes*, contenant des instructions simplifiées permettant d'indiquer où et comment se placent les informations tirées de la base de données dans la page ;
- un système de cache permet de stocker chaque page et ainsi d'éviter de provoquer des appels à la base de données à chaque visite. Non seulement la charge sur le serveur est réduite, la vitesse très largement accélérée, de plus un site sous SPIP reste consultable même lorsque la base mySQL est plantée.

Pour chaque type de document, un couple de fichiers

L'intérêt (et la limite) d'un système de publication automatisé, c'est que l'on ne va pas redéfinir une interface différente en HTML pour chaque page isolée. Par exemple, tous les articles bénéficieront de la même interface ; le système placera simplement des informations différentes dans cette même mise en page (on verra plus loin que SPIP autorise cependant une certaine souplesse).

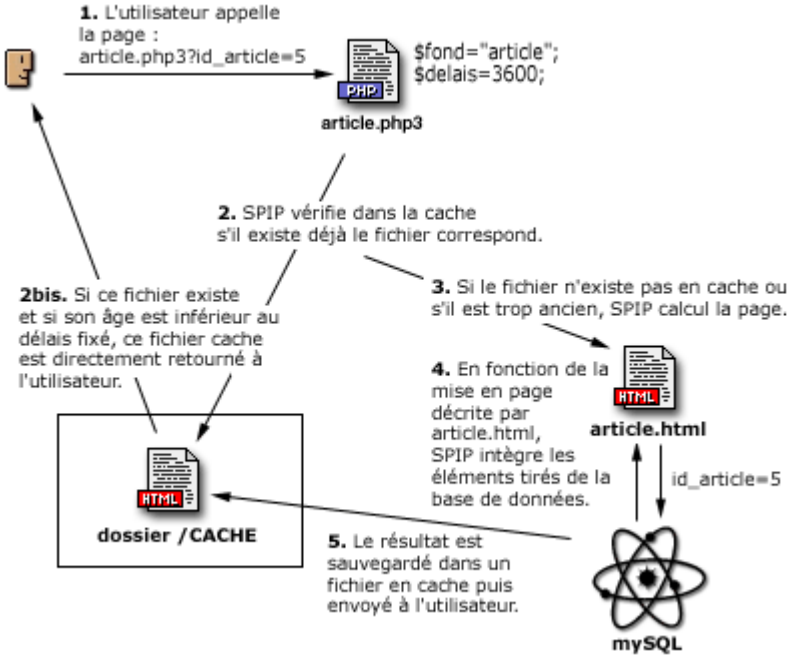
L'avantage de cette manière de procéder est évident : on définit un format-type (squelette) pour, par exemple, tous les articles, et le système fabriquera chaque page individuelle en plaçant automatiquement le titre, le texte, les liens de navigation... de chaque article.

Pour chaque type de document, SPIP vous demande *deux fichiers* : un fichier .php3 et un fichier .html. Lors de l'installation de SPIP, vous trouverez ainsi les couples : « article.php3 / article.html », « rubrique.php3 / rubrique.html », etc.

Vous pouvez naturellement modifier ces couples, et en créer d'autres. Depuis [SPIP 1.8.2], néanmoins, il existe un fichier page.php3 utilisable pour tous les types de documents que l'on pourra créer.

Le principe de fonctionnement du cache

L'appel d'une page spécifique se fait par l'intermédiaire du fichier PHP. Par exemple, pour appeler l'article n°5, l'URL correspondante est : http://www.monsite.net/article.php3?id_article=5



1. Le fichier appelé est donc article.php3, avec en paramètre id_article=5.

2. Le fichier article.php3 est un fichier PHP ; sa première tâche consiste à vérifier dans le dossier /CACHE sur votre serveur, s'il existe déjà un fichier correspondant à cet article.

2bis. Si un tel fichier existe dans /CACHE, article.php3 vérifie sa date de création. Si ce fichier est suffisamment récent, il le retourne directement à l'utilisateur. Le processus de consultation est alors terminé.

3. S'il n'existe pas un tel fichier dans /CACHE (première visite sur cet article, par exemple), ou si son âge est trop ancien, SPIP démarre le calcul de cette page.

4. C'est alors la page article.html qui est chargée et analysée. Cette page contient la mise en page correspondant à ce type de document. Il s'agit de HTML complété d'indications permettant de placer les éléments tirés de la base de données. En fonction des éléments requis par article.html, SPIP va chercher les informations nécessaires tirées de la base de données mySQL et les insérer aux endroits prévus.

5. Un fichier est ainsi fabriqué par article.php3, à partir de la description contenue dans article.html, avec les éléments tirés de la base de données. Ce fichier est alors sauvegardé dans le dossier /CACHE et renvoyé au visiteur.

Lors d'une visite suivante, si le délai entre les deux visites est suffisamment court, c'est donc ce nouveau fichier stocké dans /CACHE qui est retourné, sans avoir à faire un nouveau calcul à partir de la base de données. En cas de plantage de la base de données, c'est forcément le fichier en cache qui est retourné, même s'il est « trop âgé ».

Remarque. On voit ici que chaque page du site est mise en cache individuellement, et chaque recalcul est provoqué par les visites du site. Il n'y a pas, en particulier, un recalcul de toutes les pages du site d'un seul coup à échéance régulière (ce genre de « grosse manoeuvre » ayant le bon goût de surcharger le serveur et de le faire parfois planter).

Le fichier PHP

Le fichier « .php3 » est très simple. Par exemple, article.php3 contient uniquement :

```
<?php
$fond = "article";
$delais = 24 * 3600;

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

Son seul but est donc de fixer deux variables ($fond et $delais) et d'appeler le fichier qui déclenche le fonctionnement de SPIP (inc-public.php3).

La variable $fond est le nom du fichier qui contient la description de la mise en page (le *squelette*). Ici, puisque $fond="article", le fichier de description sera contenu dans article.html [1]. Notez bien que, dans la variable $fond, on n'indique pas la terminaison « .html ».

> *Remarque.* L'intérêt de choisir soi-même le nom du fichier de squelette (que l'on aurait pu déduire automatiquement du nom du fichier .php3) est, si nécessaire, d'utiliser un autre nom. Cela pour ne pas écraser, éventuellement, des fichiers HTML qui subsisteraient d'une ancienne version du site que l'on ne souhaite pas supprimer. S'il existe, d'une ancienne version du site, un fichier article.html que l'on ne souhaite pas effacer, on utilisera par exemple un fichier squelette pour SPIP intitulé article-nouveau.html, et on fixera dans article.php3 : $fond="article-nouveau".

La variable $delais est l'âge maximum pour l'utilisation du fichier stocké en /CACHE. Ce délai est fixé en secondes. Un délai de 3600 correspond donc à une heure ; un délai de 24*3600 est donc de 24 heures.

On jouera sur cette valeur en fonction de la fréquence des ajouts de contenu du site (nouveaux articles, nouvelles brèves...). Un site actualisé plusieurs fois par jour pourra adopter un délai d'une heure ; un site publiant quelques articles par semaine pourra adopter un délai nettement plus long. De même, le contenu des pages est important : si vous insérez la syndication du contenu de sites fréquemment mis à jour, vous souhaiterez sans doute adapter votre propre délais à celui des sites référencés.

> *Remarque.* Certains webmestre succombent à la tentation de fixer des délais dérisoires (quelques secondes), pour que le site corresponde très exactement, à chaque instant, aux dernières modifications de la base de données. Dans ce cas, vous perdez tous les avantages du système de cache : les visites sont nettement ralenties et, à l'extrême, sur des sites très fréquentés, vous pouvez provoquer des plantages de la base de données (ou vous faire virer par votre hébergeur parce que vous monopolisez la puissance de sa machine...).
>
> *Remarque.* Une autre raison qui semble pousser les webmestres à fixer des délais très courts est la présence des forums sur leur site. En effet, pour que les contributions s'affichent dès qu'elles sont postées, ils pensent nécessaire de réduire le délais de recalcul. C'est inutile : SPIP gère cet aspect automatiquement ; lorsqu'une contribution à un forum est postée, la page correspondante est effacée du cache, et recalculée immédiatement, quel que soit le délai fixé pour cette page.

Depuis [SPIP 1.8.2], on peut utiliser un même fichier .php3 pour tous les squelettes. Il s'agit du fichier page.php3 qui est à la racine du site. Il est un peu plus complexe que celui décrit ci-dessus, mais il joue le même rôle. Lorsqu'on l'utilise, les variables $fond et $fond peuvent être passées en paramètres, par exemple : http://www.monsite.org/page.php3?fond=monsquelette&id_rubrique=12

Le fichier HTML

Dans SPIP les fichiers .html sont appelés les **squelettes**. Ce sont eux qui déterminent de la mise en page.

Ces fichiers sont rédigés directement en HTML, auquel on ajoute des instructions permettant d'indiquer à SPIP quels éléments il doit extraire de la base de données et où il devra les placer (du genre : « placer le titre ici », « indiquer à cet endroit la liste des articles portant sur le même thème », etc.).

Les instructions de placement des éléments sont rédigées dans un langage spécifique, qui fait l'objet du

présent manuel d'utilisation. Ce langage constitue par ailleurs la seule difficulté de SPIP.

« Encore un langage ? » Hé oui, il va vous falloir apprendre un nouveau langage. Il n'est cependant pas très compliqué, et il permet de créer des interfaces complexes très rapidement. Par rapport au couple PHP/mySQL, vous verrez qu'il vous fait gagner un temps fou (surtout : il est beaucoup plus simple). C'est un *markup language*, c'est-à-dire un langage utilisant des balises similaires à celles du HTML.

> *Remarque.* De la même façon que l'on apprend le HTML en s'inspirant du code source des sites que l'on visite, vous pouvez vous inspirer des squelettes utilisés sur d'autres sites fonctionnant sous SPIP. Il suffit d'aller chercher le fichier « .html » correspondant. Par exemple, vous pouvez voir le squelette des articles d'uZine (visualisez le code source pour obtenir le texte du squelette).

Une interface différente dans le même site

Tout d'abord, notez qu'il est possible de créer des couples de fichiers pour le même élément logique (articles, rubriques, ...). Par exemple, vous pouvez créer des fichiers pour visiter un même article avec des interfaces différentes : article.php3/html pour le format normal, imprimer.php3/html pour le même article dans un format adapté à l'impression, article-texte.php3/html pour l'article dans un format texte (adapté aux mal-voyants par exemple), article-lourd.php/html avec une interface lourdingue adaptée au haut-débit, etc.

- **Une interface différente selon les rubriques.** Vous pouvez, pour un même type de document, créer des squelettes différents selon les rubriques du site. Il s'agit de créer simplement de nouveaux fichiers .html en fonction des rubriques (inutile, ici, de modifier le fichier .php3, on se contente de jouer sur les noms des fichiers squelettes).

Il suffit de compléter le nom du fichier squelette de « -numéro » (un tiret suivi d'un numéro de rubrique). Par exemple, si vous créez un fichier : article-**60**.html, tous articles contenus dans la rubrique n°60 utiliseront ce squelette (et non plus le squelette par défaut article.html). Notez bien : le numéro indiqué est celui d'une *rubrique*. Si cette rubrique 60 contient des sous-rubriques, les articles contenus dans ces sous-rubriques utiliseront également le nouveau squelette article-60.html.

> *Remarque.* Dans notre exemple, on aura certainement également intérêt à créer un rubrique-60.html, voire un breve-60.html, etc. pour accompagner le changement de mise en page de cette rubrique.

- **Une interface pour une seule rubrique.** (SPIP 1.3) On peut créer une interface qui s'applique à une rubrique, *mais pas à ses sous-rubriques*. Pour cela, il faut créer un fichier : article=60.html, qui s'appliquera uniquement aux articles de la rubrique 60, mais pas à ses sous-rubriques.

Que peut-on mettre dans un fichier HTML

Les fichiers HTML sont essentiellement des fichiers « texte », complétés d'instructions de placement des éléments de la base de données.

SPIP analyse uniquement les instructions de placement des éléments de la base de données (codées selon le langage spécifique de SPIP) ; il se contrefiche de ce qui est placé dans ce fichier et qui ne correspond pas à ces instructions.

Leur contenu essentiel est donc du HTML. Vous déterminez la mise en page, la version du HTML désiré, etc. Vous pouvez évidemment y inclure des feuilles de style (CSS), mais également du JavaScript, du Flash... en gros : tout ce qu'on place habituellement dans une page Web.

Mais vous pouvez également (tout cela n'est jamais que du texte) créer du XML (par exemple, « backend.php3/html » génère du XML).

Plus original : toutes les pages retournées au visiteur sont tirées du /CACHE par une page écrite en PHP. Vous pouvez donc inclure dans vos squelettes des instructions en PHP, elles seront exécutées lors de la visite. Utilisée de manière assez fine, cette possibilité permet une grande souplesse à SPIP, que vous pouvez ainsi compléter (par exemple ajouter un compteur, etc.), ou même faire évoluer certains éléments de mise en page

en fonction des informations tirées de la base de données.

Notes

[1] Si article.html n’existe pas, le fichier « dist » est pris à la place. Lire à ce propos « Qu’est-ce que les fichiers « dist » ? ».

SPIP pas à pas

Mon premier squelette

(je le sors du placard)

Si le système de squelettes peut de prime abord paraître intimidant, c'est que ce qu'on lui demande est suffisamment riche pour l'obliger à être complexe. Mais ! Complexe ne veut pas dire compliqué. Voici un exemple minimal de squelette.

Matériel requis pour ce tutoriel

- Un SPIP installé quelque part. On supposera, pour commencer, que votre base SPIP contient au minimum une rubrique et deux articles *publiés*. Si ce n'est pas le cas, vous pouvez très vite y remédier en copiant-collant les premiers textes qui vous passent sous la main (vérifiez quand même qu'il ne s'agit pas de votre déclaration enflammée au petit ami de votre voisin de bureau).

- Un éditeur de texte pour créer et modifier les fichiers utilisés par SPIP. *Note :* certaines personnes auront le réflexe de vouloir utiliser DreamWeaver (ou autre logiciel graphique) pour modifier les fichiers .html. Cependant pour des exemples simples DreamWeaver compliquera la tâche et risque même de modifier vos fichiers dans votre dos. Il est donc vraiment préférable d'utiliser un éditeur de texte classique (par exemple le bloc-notes sous Windows).

---

1. Dans les versions antérieures à SPIP 1.9, avant d'utiliser un squelette, il faut pouvoir l'appeler. Si vous utilisez une version supérieure de SPIP, passez directement à l'étape suivante, ci-après. Sinon, pour appeler le squelette, créez à la racine de votre site un fichier tutoriel.php3 contenant les lignes suivantes :

```
<?php
$fond = "tutoriel";
$delais = 0;
include "inc-public.php3";
?>
```

Puis testez dans votre navigateur : http://votresite.net/tutoriel.php3. Pas très glorieux, n'est-ce pas ? Le message d'erreur vous informe qu'il manque un fichier. C'est le fameux squelette, que nous allons maintenant créer.

SPIP 1.9 a nettement simplifié la création de squelettes, en vous épargnant cette première étape d'appel du squelette. En effet, à partir de la version 1.9, il n'y a plus de fichier .php3 (ou .php) pour les squelettes, qui sont tous calculés à partir du script unique spip.php.
La suite de ce tutorial reste valable quelque soit la version de SPIP utilisée.

---

1. À la racine du site, déposez un fichier tutoriel.html, qui contient ce qui suit :

```
<BOUCLE_article(ARTICLES){id_article=1}>
#TITRE
</BOUCLE_article>
```

Puis affichez la page http://votresite.net/spip.php?page=tutoriel (ou rechargez la page http://votresite.net/tutoriel.php3, si vous utilisez une version antérieure à SPIP 1.9) :

SPIP est allé chercher le titre de l’article n°1 de votre base, et l’a inscrit à la place de #TITRE.

Si ça ne fonctionne pas, vérifiez que votre article n°1 est bien « publié » (et pas « en attente » ou « en cours de rédaction »).

Puis ajoutez du HTML et d’autres appels de « champs » SPIP, et vous obtenez rapidement votre article n ° 1 :

```
<BOUCLE_article(ARTICLES){id_article=1}>
<h1>#TITRE</h1>
<b>#CHAPO</b>
<div align="justify">#TEXTE</div>
</BOUCLE_article>
```

Ajoutez ensuite les champs manquants pour parfaire l’affichage de l’article : #SURTITRE, #LESAUTEURS, #SOUSTITRE, #NOTES, etc.

Bien !

Un squelette, plusieurs articles...

c'est à ça que ça sert !

La leçon précédente nous a permis d'extraire des données de l'article n°1 de la base et d'en faire une page Web. Généralisons...

Notre squelette est bien inutile s'il ne sert qu'à afficher l'article n°1. **Apprenons-lui à afficher n'importe quel article :**

Pour cela nous allons appeler notre page Web avec un paramètre, du type id_article=2 : dirigez votre navigateur sur l'URL suivante :
« http://votresite.net/spip.php?page=tutoriel&id_article=2 » [*]

S'affiche... toujours l'article 1 (et pas 2). Modifions dans le squelette tutoriel.html la ligne qui définit la « boucle article » :
<BOUCLE_article(ARTICLES){id_article}>
Comme vous le voyez, on remplace simplement {id_article=1} par {id_article} tout court.

Voilà : http://votresite.net/spip.php?page=tutoriel&id_article=2 vous donne maintenant l'article 2. [1]

La BOUCLE_article s'exécute dans un « *contexte* » où id_article est égal à 2 (c'est la valeur qui est passée dans l'URL). Si on lui précise {id_article=1} elle va chercher l'article n° 1, mais si on lui demande juste {id_article}, elle va chercher l'article dont le numéro est indiqué par le contexte (ici l'URL).

Visitez maintenant ces pages :

- http://votresite.net/spip.php?page=tutoriel&id_article=1,
- http://votresite.net/spip.php?page=tutoriel&id_article=2 et
- http://votresite.net/spip.php?page=tutoriel [*].

Voyez-vous la différence ? Les deux premières pages vous donnent les articles n°1 et 2, la troisième n'a pas d'id_article dans son contexte, et génère une erreur.

Bravo ! Votre squelette est maintenant « contextuel ».

Notes

[*] Rappelons que dans les versions antérieures à SPIP 1.9, l'URL pour afficher notre tutorial est : http://votresite.net/tutoriel.php3. Lorsqu'on lui passe un paramètre : http://votresite.net/tutoriel.php3?id_article=2, etc.

[1] Non ? Il devrait...

Une rubrique

ou comment faire des listes du contenu de la base

La leçon précédente nous a appris à afficher des éléments en fonction du contexte. Nous allons ici voir comment ce contexte varie au fur et à mesure des BOUCLES rencontrées.

Modifions notre squelette « tutoriel.html » de la manière suivante :

```
<BOUCLE_article(ARTICLES)>
#TITRE<br />
</BOUCLE_article>
```

Là, on supprime carrément la condition {id_article}. Attention : cette BOUCLE peut générer une page énorme si votre base contient déjà pas mal d'articles : mieux vaut prendre nos précautions et ajouter tout de suite {0,10} pour limiter aux 10 premiers articles...

<BOUCLE_article(ARTICLES){0,10}>

Résultat : en appelant simplement http://votresite.net/spip.php?page=tutoriel [*] (plus besoin d'id_article désormais, puisque cette condition a été supprimée) les titres des 10 premiers articles publiés s'affichent, séparés chacun par un saut de ligne. À partir de là, on voit comment on peut produire le sommaire d'une rubrique : affichons les 10 articles les plus récents appartenant à cette rubrique.

```
<BOUCLE_article(ARTICLES){id_rubrique}{par date}{inverse}{0,10}>
<a href="#URL_ARTICLE">#TITRE</a><br>
</BOUCLE_article>
```

Prenons dans l'ordre :

- id_rubrique : ne prend que les articles appartenant à la rubrique id_rubrique (*cf.* ci-dessous pour que cette variable soit définie dans le contexte de notre BOUCLE_article) ;
- {par date}{inverse} : trie par date dans l'ordre chronologique décroissant...
- {0,10} : ... et prend les 10 premiers résultats.
- Enfin, #TITRE va afficher non seulement le titre de l'article mais en plus créer un lien vers cet article.

Reste à invoquer le squelette, *en lui passant le contexte id_rubrique=1* :

http://votresite.net/spip.php?page=tutoriel&id_rubrique=1 [*]

La magie de SPIP tient dans la combinaison de ce type de fonctionnalités. Si vous êtes arrivé jusqu'ici, c'est gagné !

Notes

[*] Rappelons que dans les versions antérieures à SPIP 1.9, l'URL pour afficher notre tutorial est : http://votresite.net/tutoriel.php3. Lorsqu'on lui passe un paramètre : http://votresite.net/tutoriel.php3?id_rubrique=1, etc.

Boucles en boucles

plusieurs niveaux de lecture

Nous savons générer une liste de titres dans une rubrique. Maintenant, nous allons afficher, sur la même page, les éléments de la rubrique elle-même : son titre, son texte de présentation, etc.

Essayez !

Et voici une solution :

```
<BOUCLE_rubrique(RUBRIQUES){id_rubrique}>
<h1>#TITRE</h1>

<BOUCLE_article(ARTICLES){id_rubrique}{par date}{inverse}{0,10}>
<a href="#URL_ARTICLE">#TITRE</a><br/>
</BOUCLE_article>

[(#TEXTE|justifier)]
</BOUCLE_rubrique>
```

On appelle la page avec http://votresite.net/spip.php?page=tutoriel&id_rubrique=1 [*].
Que s'est-il passé ici ?

Notre boucle ARTICLES est intégrée dans une boucle RUBRIQUES. Le contexte de la boucle ARTICLES est l'id_rubrique donné par la boucle RUBRIQUES, qui elle-même va chercher le contexte donné par l'URL (id_rubrique=1). Donc nous sommes bien, au niveau des ARTICLES, avec l'id_rubrique demandé. De ce point de vue rien ne change.

En revanche, la boucle RUBRIQUES a permis à SPIP de sélectionner les valeurs des champs de la rubrique en question : on peut donc afficher le #TITRE et le #TEXTE de cette rubrique. Notez bien que ce #TEXTE serait celui de la rubrique *même si* on appelait aussi #TEXTE dans la boucle ARTICLES. Le fonctionnement arborescent de SPIP garantit que le #TEXTE d'un article ne déborde pas de la boucle ARTICLES...

Dernière remarque : on a introduit un *filtre* |justifier sur le champ #TEXTE. Ce filtre modifie le contenu du texte avant de l'installer dans la page finale. Ca vous fait saliver ?

Notes

[*] Rappelons que dans les versions antérieures à SPIP 1.9, l'URL pour afficher notre tutorial est : http://votresite.net/tutoriel.php3. Lorsqu'on lui passe un paramètre : http://votresite.net/tutoriel.php3?id_rubrique=1, etc.

Gérer le cache

et éviter de faire ramer le serveur qui n'a pas que ça à faire

Rappelons que ce tutoriel n'est valable que jusqu'à SPIP 1.8.3 et obsolète à partir de SPIP 1.9. Il est en cours de réécriture dans l'espace privé de ce site.

Vous pouvez néanmoins réaliser ce tutoriel en omettant les fichiers .php3 et en localisant les fichiers html dans le dossier squelettes/ **que vous créerez au besoin à la racine.**

Dans les leçons précédentes nous avons commencé à élaborer des squelettes. Le succès de notre site risque d'être fulgurant. Pensons tout de suite aux pauvres neurones de notre ordinateur. Dans cette leçon, rien d'amusant, rien d'essentiel non plus. Les flemmards en profiteront pour roupiller au fond près du radiateur...

À partir de SPIP 1.9, le délai du cache décrit ci-dessous est défini dans le fichier html du squelette, par une balise #CACHE{délais}**, par exemple** #CACHE{3600}

Résumé pour ceux-ci et pour les gens pressés : dans les fichiers d'appel de type tutoriel.php3*, réglez* $delais = 3600 ; *au lieu de* 0.

...

Au moment où une page est demandée à SPIP, celui-ci regarde si, par hasard, il n'aurait pas déjà calculé cette page auparavant. Si l'URL demandée est http://votresite.net/tutoriel.php3?id_article=12, SPIP regarde dans son sous-répertoire CACHE/ si ce fichier existe, et, le cas échéant, compare l'âge du fichier caché aux $delais fixés dans le fichier d'appel tutoriel.php3.

Dans notre exemple nous avions fixé des $delais=0 ; - d'où un recalcul systématique des pages à chaque consultation du site. Passons à $delais=3600 ; (c'est en secondes).

Notre page web n'est donc recalculée que si, lorsqu'un visiteur la demande, sa version cachée date de plus d'une heure (soit 3600 s.). Sinon, SPIP lit simplement le contenu du fichier caché [1], et renvoie le résultat sans se connecter à la base de données (sauf pour y insérer un « hit » dans les statistiques).

Comment fixer ces $delais de manière à optimiser le rapport réactivité/charge du serveur ? Pas de solution miracle, mais n'hésitez pas à fixer un délai d'une journée (i.e. $delais=24*3600 ;) ou plus pour les articles et les rubriques. Les pages de navigation les plus importantes peuvent avoir des $delais plus courts (vingt minutes ou une heure par exemple) si votre site est censé réagir à la validation fréquente de nouvelles brèves et de sites syndiqués... Si vous êtes sur un serveur partagé avec d'autres sites, soyez respectueux des autres et ne prenez pas tout le temps de calcul pour des pages qui changent rarement : ce serait d'autant plus idiot que, sur les gros articles ou sur les sommaires, le calcul des pages peut prendre quelques secondes, ce qui ralentit la consultation de vos pages...

Comment provoquer une mise à jour hors délai ? Nous venons de décider de $delais extrêmement longs, et nous repérons une fôte d'ortografe dans une page. Correction dans l'espace privé... Comment effacer tout de suite cette vilaine cicatrice du site ?

- Depuis l'espace privé, cliquer sur « Voir en ligne » déclenche le recalcul pour les pages correspondant à #URL_ARTICLE ou #URL_RUBRIQUE de l'article ou de la rubrique correspondante. C'est le cas le plus courant. Mais sinon ?

- Dans la partie « Sauvegarde/Restauration » de l'espace privé, un bouton « vider le cache » efface *tous* les fichiers cachés (utile si vous faites plein de modifications et avez un site très complexe, à éviter sinon).

- Toutefois, la solution la plus simple est de demander à SPIP, dans la page d'accueil de l'espace privé, de vous « poser un cookie d'administration ». Ce cookie s'incrustera sur votre navigateur, et

SPIP vous reconnaîtra au moment de vous envoyer la page dans le site public : il vous proposera alors, en bas de page, un bouton « Recalculer cette page ».

Retour au contexte : On revient ici à la notion de contexte. Si le squelette est appelé avec un contexte d'id_article, d'id_rubrique ou encore d'id_breve, un autre bouton vous est proposé quand SPIP détecte le cookie : « Modifier cet article (ou rubrique, ou brève) », qui vous mène directement sur la page correspondante dans le back-office. Merci qui ?

Derniers détails :

- pour des raisons évidentes, le moteur de recherche ne déclenche pas de cache, et les pages avec forum sont réactualisées dès qu'une nouvelle contribution est envoyée.
- le répertoire CACHE/ dans l'arborescence du site est découpé en 16 sous-répertoires numérotés 0, 1, 2... 9, A, B... F, dans lesquels les fichiers cachés se distribuent quasi-aléatoirement ; cela s'appelle « *hasher le cache* » et rien que pour cela mérite bien qu'on le mentionne.
- les fichiers cachés sont exploités même si la base de données est « tombée », ce qui garantit le site contre des pannes transitoires du serveur mySQL.

Notes

[1] Pour les spécialistes, il s'agit en fait d'un *include* PHP du fichier correspondant, permettant d'exécuter du code depuis le cache...

Des filtres

Subtilités squelettiques

Si les BOUCLES permettent de structurer la page de manière logique, reste à présenter les données de manière esthétique. Question dizahïgne SPIP ne peut rien pour vous, mais sachez user de ses philtres...

Une donnée stockée dans la base de données se présente comme un bloc de texte, et on peut avoir envie de manipuler sa valeur avant de l'afficher à l'écran. Les *filtres* sont faits pour ça :

- les filtres les plus utilisés (ils sont appelés automatiquement) sont |typo et |propre ; le premier est un correcteur typographique, dont la mission principale est d'ajouter des espaces insécables où il en faut (*cf.* l'aide en ligne de SPIP) ; le second s'intéresse aux paragraphes, aux raccourcis SPIP (italiques, gras, intertitres, etc.) - il n'est appliqué par défaut qu'aux textes longs (#TEXTE, #CHAPO, etc.)

- d'autres filtres sont très utiles : citons |majuscules (à la fonctionnalité évidente), |justifier ou |aligner_droite (qui définissent l'alignement du texte par rapport aux bords verticaux), ou encore l'ésotérique |saison (qui affiche « été » si la variable est une date comprise entre le 21 juin et le 20 septembre)...

Pour utiliser un filtre il faut entourer la variable de parenthèses et de crochets (on verra plus tard les implications) : [blah blah (#VARIABLE|filtre) bloh bloh]

On peut enchaîner les filtres les uns à la suite des autres [1] : ainsi [(#DATE|saison|majuscules)] affichera-t-il « HIVER ».

Exercice portant sur l'ensemble des leçons précédentes : Afficher en majuscules les titres des 10 articles les plus récents de la rubrique passée en contexte, et mettre en tête de page la saison courante (c'est-à-dire la saison à laquelle a été publié l'article le plus récent de toute la base).

...

Pourquoi ces crochets ? Supposons que votre base contient des articles datés et d'autres non datés. La variable #DATE vaut « 2001-07-01 10-53-01 » (date au format mySQL) dans le premier cas, et « 0000-00-00 00-00-00 » dans le second. Pour afficher la date dans un joli (?) cadre, on va utiliser, dans le squelette, les lignes suivantes :

```
[<table border="1"><tr><td>
(#DATE|affdate)
</td></tr></table>]
```

Ici le filtre |affdate affiche la date en lettres (au format « 1er juillet 2001 »), mais renvoie une chaîne vide si la date est inconnue (égale à « 0000... »). Les crochets délimitent ce qu'il faut afficher autour de la date *si le resultat entre parenthèses n'est pas une chaîne vide*.

Résultat : seuls les articles datés provoquent l'affichage d'un tableau contenant la date. Un squelette bien construit définira précisément ce qu'il faut afficher *ou pas* en fonction du contenu... Les filtres servent aussi à ça.

P.-S.

Notons que certains filtres de présentation peuvent être avantageusement remplacés par des feuilles de style. Ainsi |majuscules est équivalent à l'attribut CSS « text-transform: uppercase », et |justifier à « text-align: justify ».

Lire *SPIP et les feuilles de style* pour plus de détails sur les styles CSS offerts par SPIP.

Notes

[1] On peut appeler ça un « *pipeline* »...

Boucles et balises : manuel de référence

Des boucles et des balises

Tout ce qui suit concerne désormais le langage de description de la mise en page des *squelettes* dans SPIP ; si vous avez bien compris l'explication précédente, vous savez que nous travaillons donc dans les fichiers « .html ».

La présente documentation est volontairement technique (il s'agit ici de références techniques) ; vous pouvez préférer commencer par notre guide Pas à pas, plus didactique, et revenir ensuite ici pour une documentation plus précise.

Des boucles

La notion de base du langage de SPIP est la *boucle*.

- La logique de la boucle

Une base de données, classiquement, c'est une liste d'éléments : ici, une liste des articles, une liste des rubriques, une liste des auteurs, etc. Pour « fabriquer » le site, on va donc extraire de cette liste certains de ses éléments :

- à la base, on veut extraire *un seul élément* d'une liste ; par exemple, afficher l'article désiré ;
- mais il est fréquent d'extraire *plusieurs éléments* d'une liste ; par exemple, dans la page d'une rubrique, on veut afficher *tous* les articles contenus dans cette rubrique, ainsi que *toutes* les sous-rubriques contenues dans cette rubrique ;
- plus subtil : il arrive fréquemment qu'il n'y ait pas d'éléments satisfaisants à tel ou tel endroit ; SPIP doit alors pouvoir gérer l'éventualité de l'absence de ces éléments ; par exemple, le squelette de l'affichage des rubriques demande l'affichage de toutes les sous-rubriques contenues dans une rubrique ; que faire, alors, s'il n'y a pas sous-rubriques dans cette rubrique spécifique ?

Ces trois situations sont traitées par la notion unique de *boucle*, qui permet à la fois de gérer l'affichage d'un seul élément, de plusieurs éléments successifs, ou l'absence d'éléments.

Le système de boucle permet, dans un code unique :

- d'indiquer à quel endroit du code HTML on a besoin de quel type d'élément (à tel endroit on veut récupérer la liste des articles, à tel endroit on veut inclure la liste des sous-rubriques...) ;
- de prévoir l'affichage d'un élément unique ;
- d'indiquer comment est affichée une liste de plusieurs éléments ;
- de déterminer ce qu'on affiche lorsqu'il n'y a aucun élément correspondant.

Analogie avec la programmation en PHP/mySQL

Ceux qui ont déjà programmé des requêtes mySQL en PHP savent que le traitement se déroule en deux temps :

- la construction de la syntaxe de la requête (qui consiste à dire « je veux récupérer la liste des articles contenus dans telle rubrique... ») ;
- l'analyse et l'affichage des résultats au travers d'une boucle.

Ce sont ces deux événements qui sont gérés, dans SPIP, au travers des boucles.

Les balises SPIP

Grâce aux boucles, on a donc récupéré des éléments uniques ou des listes d'éléments : par exemple une liste d'articles ou une liste de rubriques...

Cependant, chaque élément de telles listes est composé de plusieurs éléments précis : par exemple un article se compose d'un titre, d'un surtitre, d'un sous-titre, d'un texte d'introduction (chapeau), d'un texte principal, d'un post-scriptum, etc. Il existe ainsi des balises spécifiques à SPIP, permettant d'indiquer précisément à quel endroit on affiche des éléments : « placer le titre ici », « placer le texte ici »...

Les balises à l’intérieur des boucles

Voici, au travers d’un cas classique, le principe de fonctionnement général d’une boucle accompagnée de ses balises (attention, ça n’est pas du langage SPIP, c’est une description logique) :

BOUCLE : afficher la liste des articles de cette rubrique

- *afficher ici le titre de l'article*
- *afficher le sous-titre*
- *afficher le texte*

Fin de la BOUCLE

Cette boucle, analysée par SPIP, peut donner trois résultats différents.

- **Il n’y a aucun article dans cette rubrique.**

Dans ce cas, bien évidemment, aucun des éléments « afficher ici... (titre, sous-titre...) » n’est utilisé. En revanche, si on l’a prévu, on peut afficher un message du genre « Il n’y a pas d’article ».

- **Il y a un seul article dans cette rubrique.**

Dans ce cas, très simplement, la page HTML est construite sur le modèle de la boucle :

- *Titre de l'article*
- *Sous-titre*
- *Texte de l'article*

- **Il y a plusieurs articles dans cette rubrique.**

La description de la mise en page (« placer ici... ») va alors être calculée successivement pour chacun des articles. Ce qui donne simplement :

- *Titre de l'article 1*
- *Sous-titre de l'article 1*
- *Texte de l'article 1*
- *Titre de l'article 2*
- *Sous-titre de l'article 2*
- *Texte de l'article 2*

...

- *Titre du dernier article*
- *Sous-titre du dernier article*
- *Texte du dernier article*

La suite de ce guide de référence se construira donc de la manière suivante :

- syntaxe générale des boucles ;
- syntaxe générale des balises de SPIP ;
- et, ensuite, une page spécifique à chaque type de boucles, indiquant quelles balises on peut y utiliser.

La syntaxe des boucles

Syntaxe de base

La syntaxe simplifiée d'une boucle est la suivante :

<BOUCLE*n*(TYPE){critère1}{critère2}...{critèrex}>
* Code HTML + balises SPIP
</BOUCLE*n*>

On a vu, dans l'explication sur les boucles et les balises, que le « Code HTML + balises SPIP » se répétait autant de fois que la boucle obtenait d'éléments tirés de la base de données (c'est-à-dire une fois, plusieurs fois, ou zéro fois).

La ligne importante, ici, est :

<BOUCLE*n*(TYPE){critère1}{critère2}...{critèrex}>

- **L'élément** BOUCLE est l'ordre indiquant qu'il s'agit d'une boucle SPIP ; on ne peut donc pas le modifier ; dit autrement, toutes les boucles de SPIP commencent par l'instruction BOUCLE.

- **L'élément *n*** est le nom ou le numéro de la boucle, librement choisi par le webmestre, pour chaque boucle qu'il utilise. On verra plus loin qu'il est possible (c'est même tout l'intérêt de la manœuvre) d'utiliser plusieurs boucles dans un même squelette : les nommer est donc indispensable pour les identifier.

Si vous décidez de numéroter vos boucles, la syntaxe devient par exemple (pour la boucle 5) :
<BOUCLE5...> ... </BOUCLE5>

Si vous décidez de donner un nom à vos boucles (c'est généralement plus pratique, votre code est plus lisible), il faut impérativement faire précéder ce nom par le symbole « _ » (que l'on appelle habituellement *underscore*). Par exemple :
<BOUCLE_sousrubriques...> ... </BOUCLE_sousrubriques>

- **L'élément** (TYPE) est primordial : il indique quel type d'éléments on veut récupérer. La syntaxe est importante : le TYPE est indiqué entre parenthèses (sans espaces), en majuscules, et ce TYPE doit correspondre obligatoirement à l'un des types prévus dans SPIP (qu'on trouvera dans la présente documentation) : ARTICLES, RUBRIQUES, AUTEURS, BREVES, etc.

Pour l'exemple précédent, on aurait donc :

<BOUCLE_sousrubriques(RUBRIQUES)...>
...
</BOUCLE_sousrubriques>

- **Les critères** {critère1}{critère2}...{critèrex} indiquent à la fois selon quels critères on veut sélectionner les éléments de la base de données (afficher les sous-rubriques incluses dans cette rubrique, afficher les autres rubriques installées au même niveau hiérarchique que la présente rubrique...), et la façon dont on va classer ou sélectionner les éléments (classer les articles selon leur date, selon leur titre... afficher uniquement les 3 premiers articles, afficher la moitié des articles...). Comme on peut combiner les critères, on peut très aisément fabriquer des requêtes très puissantes, du genre « afficher la liste des 5 articles les plus récents écrits par cet auteur ».

Les critères sont entre acolades ; ils peuvent être séparés les uns des autres par un espace. Exemple :

<BOUCLE_meme_auteur(ARTICLES) {id_auteur} {par date}{inverse} {0,5}>
...
</BOUCLE_meme_auteur>

Les différents critères et leur syntaxe seront explicités par la suite, pour chaque type de boucle (certains critères fonctionnent pour tous les types de boucles, d'autres sont spécifiques à certaines boucles).

Syntaxe complète

Le syntaxe indiquée précédemment peut être complétée par des éléments conditionnels. En effet, la boucle précédente affiche successivement les éléments contenus *à l'intérieur* de la boucle. SPIP permet de plus d'indiquer ce qu'on affiche avant et après la boucle au cas où elle contient un ou plusieurs résultats, et ce qu'on affiche s'il n'y a aucun élément.

Cela donne :

<B*n***>**
* Code HTML optionnel avant
<BOUCLE*n*(TYPE){critère1}{critère2}...{critèrex}>
* Code HTML + balises SPIP
</BOUCLE*n***>**
* Code HTML optionnel après
</B*n***>**
* Code HTML alternatif
<//B*n***>**

Le *code optionnel avant* (précédé de <Bn>) n'est affiché que si la boucle contient au moins une réponse. Il est affiché avant les résultats de la boucle.

Le *code optionnel après* (terminé par </Bn>) n'est affiché que si la boucle contient au moins une réponse. Il est affiché après les résultats de la boucle.

Le *code alternatif* (terminé par <//Bn>) est affiché à la place de la boucle (et donc également à la place des codes optionnels avant et après) si la boucle n'a trouvé aucune réponse.

Par exemple, le code :

```
<B1>
 Cette rubrique contient les éléments suivants:
 <ul>
  <BOUCLE1(ARTICLES){id_rubrique}>
  <li>#TITRE</li>
  </BOUCLE1>
 </ul>
</B1>
 Cette rubrique ne contient pas d'article.
<//B1>
```

donne les résultats suivants :

- **s'il y a un seul article :**

```
Cette rubrique contient les éléments suivants:
<ul>
  <li>Titre de l'article</li>
</ul>
```

- **s'il y a plusieurs articles :**

```
Cette rubrique contient les éléments suivants:
<ul>
  <li>Titre de l'article 1</li>
  <li>Titre de l'article 2</li>
  ...
  <li>Titre du dernier article</li>
</ul>
```

- **s'il n'y a aucun article :**

Cette rubrique ne contient pas d'article.

Historique : Jusqu'à [SPIP 1.7.2], La manière dont SPIP interprétait les boucles interdisait de mettre une boucle entre <Bn> et <BOUCLEn>. Par contre, il restait possible de mettre des boucles supplémentaires dans les parties optionnelles situées *après* la définition <BOUCLEn...>. Si vous deviez vraiment installer une boucle dans la partie optionnelle *avant*, il fallait passer par une commande <INCLURE()>.

Des critères d'environnement en cascade

Chaque boucle effectue la sélection des éléments tirés de la base de données en fonction de critères. Certains de ces critères correspondent à l'environnement dans lequel se trouve la boucle.

Par exemple : si on prévoit une boucle du genre « Afficher les articles inclus dans cette rubrique », il faut savoir de quelle rubrique il s'agit. C'est ce que l'on nomme l'environnement.

- **L'environnement fourni par l'URL**

Lorsque l'on visite une page d'un site SPIP, son adresse contient généralement une variable. Par exemple : spip.php?rubrique15 (avant SPIP 1.9, l'url s'écrit : rubrique.php3?id_rubrique=15)

Cette variable définit donc un premier environnement : la boucle « Afficher les articles inclus dans cette rubrique » doit alors être comprise comme « Afficher les articles de la rubrique 15 ».

Clairement, avec le même code de squelette, si on appelle l'adresse : spip.php?rubrique7 (avant SPIP 1.9, l'url s'écrit : rubrique.php3?id_rubrique=7) ; l'interprétation de cette boucle deviendra « Afficher les articles de la rubrique 7 ».

- **L'environnement fourni par les autres boucles**

À l'intérieur d'une boucle, l'environnement est modifié par chaque élément de la boucle. En plaçant des boucles les unes à l'intérieur des autres, on hérite ainsi d'environnements imbriqués les uns dans les autres.

Ainsi, dans la structure suivante :

```
<BOUCLE_articles: afficher les articles de cette rubrique>
 Afficher le titre de l'article
 <BOUCLE_auteurs: afficher les auteurs de cet article>
  Nom de l'auteur
 </BOUCLE_auteurs>
</BOUCLE_articles>
```

On doit comprendre que :

- la première boucle (BOUCLE_articles) affiche les articles en fonction de la rubrique, selon l'environnement fournit par l'URL (id_rubrique=15 par exemple) ;
- dans cette boucle, on obtient un ou plusieurs articles ;
- « à l'intérieur » de chacun de ces articles, on a un environnement différent (celui de l'article, c'est-à-dire, par exemple, id_article=199) ;
- la seconde boucle (BOUCLE_auteurs), qui est installée *à l'intérieur* de la première boucle, dépend pour chacune de ses exécutions successives (elle est exécutée pour chaque article de la première boucle) : « afficher les auteurs de cet article » devient successivement « afficher les auteurs du premier article », « du deuxième article » et ainsi de suite.

On voit que, par l'imbrication de boucles successives, on obtient différentes boucles, incluses les unes dans les autres, qui dépendent du résultat des boucles dans lesquelles elles sont situées. Et finalement, la toute première boucle (celle qui contient toutes les autres) dépend d'un paramètre fixé dans l'adresse de la page.

Boucles incluses et boucles successives

Si l'on peut inclure des boucles les unes à l'intérieur des autres (chaque boucle incluse dépendant alors du

résultat de la boucle à l'intérieur de laquelle elle est installée), on peut tout aussi bien installer des boucles les unes à la suite des autres ; des boucles successives n'influent pas les unes sur les autres.

Par exemple, la page d'une rubrique est typiquement constituée des éléments suivants :

```
<BOUCLE_rubrique(RUBRIQUES){id_rubrique}>
 <ul>Titre de la rubrique
 <BOUCLE_articles(ARTICLES){id_rubrique}>
   <li> Titre de l'article</li>
 </BOUCLE_articles>
 <BOUCLE_sous_rubriques(RUBRIQUES){id_rubrique}>
   <li> Titre de la sous-rubrique </li>
 </BOUCLE_sous_rubriques>
 </ul>
</BOUCLE_rubrique>
 <ul>Il n'y a pas de rubrique à cette adresse.</ul>
<//B_rubrique>
```

La première boucle (BOUCLE_rubrique) dépend de la variable passée dans l'URL de la page (id_rubrique=15 par exemple).

Les boucles suivantes (BOUCLE_articles et BOUCLE_sous_rubriques) sont installées *à l'intérieur* de la première boucle. Ainsi, s'il n'existe pas de rubrique 15, la première boucle ne donne aucun résultat (le code alternatif « Il n'y a pas de rubrique... » est affiché), et donc les deux boucles incluses sont totalement ignorées. Mais s'il existe une rubrique 15, ces deux sous-boucles seront analysées.

On constate également que ces deux boucles se présentent *l'une après l'autre*. Ainsi, elles fonctionnent en fonction de la première boucle, mais indépendamment l'une de l'autre. S'il n'y a pas d'articles dans la rubrique 15 (BOUCLE_articles), on affichera tout de même la liste des sous-rubriques de la rubrique 15 (BOUCLE_sous_rubriques) ; et inversement.

Compteurs

Deux balises permettent de compter les résultats dans les boucles.

- **#TOTAL_BOUCLE** retourne le nombre total de résultats affichés par la boucle. On peut l'utiliser dans la boucle, dans ses parties *optionnelles* — avant et après — ou même dans la partie *alternative* après la boucle.

Par exemple, pour afficher le nombre de documents associés à un article :

```
<BOUCLE_art(ARTICLES){id_article}>
  <BOUCLE_doc(DOCUMENTS) {id_article}></BOUCLE_doc>
    [il y a (#TOTAL_BOUCLE) document(s).]
  <//B_doc>
</BOUCLE_art>
```

Attention : si la partie centrale de la boucle ne retourne rien (c'est le cas avec la boucle <BOUCLE_doc> ci-dessus, qui ne sert qu'à compter le nombre de résultats), le #TOTAL_BOUCLE ne pourra être affiché que dans la partie alternative *après* de la boucle (<//B_doc>).

- **#COMPTEUR_BOUCLE** retourne le numéro de l'itération actuelle de la boucle. On peut par exemple l'utiliser pour numéroter des résultats :

```
<BOUCLE_art(ARTICLES) {par date} {inverse} {0,10}>
#COMPTEUR_BOUCLE - #TITRE<br>
</BOUCLE_art>
```

La syntaxe des balises SPIP

Chaque type de boucle permet de sélectionner des éléments de la base de données de SPIP : des articles, des rubriques, des brèves, etc. Chacun de ces éléments est lui-même constitué d'éléments précis : un titre, une date, un texte, etc. À l'intérieur d'une boucle, il faut donc pouvoir indiquer à quel endroit du code HTML on place tel ou tel de ces éléments précis.

Pour cela, on va utiliser des balises SPIP.

Fonctionnement simplifié

Une balise SPIP se place à l'intérieur d'une boucle (puisqu'il faut savoir si l'on veut récupérer un élément d'un article, d'une rubrique, etc.). Le nom de ces balises est généralement simple, et nous fournirons, pour chaque type de boucle, la liste complète des balises que l'on peut utiliser.

Une balise est toujours précédée du signe dièse (#).

Par exemple, affichons une liste de noms d'articles :

```
<BOUCLE_articles(ARTICLES){id_rubrique}>
#TITRE<br />
</BOUCLE_articles>
```

Lorsque la boucle sera exécutée, la balise SPIP #TITRE sera à chaque fois remplacée par le titre de l'article en question :

Titre de l'article 1
Titre de l'article 2
...
Titre du dernier article

Rien de bien compliqué : on se contente d'indiquer à l'intérieur du code HTML le nom de l'élément désiré, et celui-ci est remplacé par le contenu tiré de la base de données.

Codes optionnels

Dans la pratique, un élément de contenu est souvent accompagné de code HTML *qui ne doit s'afficher que si cet élément existe*, faute de quoi la mise en page devient imprécise.

Par exemple : il existe une balise SPIP pour indiquer le surtitre d'un article. Or de nombreux articles n'ont pas de surtitre.

Complétons l'exemple précédent :

```
<BOUCLE_articles(ARTICLES){id_rubrique}>
#SURTITRE<br />
#TITRE<br />
</BOUCLE_articles>
```

qui, classiquement, nous donne une liste d'articles, avec désormais l'indication du titre et du surtitre de chaque article. Mais que se passe-t-il si l'article n'a pas de surtitre ? On obtient le code : «
 », c'est-à-dire une ligne blanche (un retour chariot).

Ce que nous devons faire : n'afficher le code «
 » que si un surtitre existe pour l'article.

La syntaxe de la balise SPIP devient alors :

[*texte optionnel avant* **(#BALISE)** *texte optionnel après* **]**

La balise qui détermine l'option est placée entre parenthèses, et l'ensemble du texte conditionnel entre crochets. Le *texte optionnel avant* et le *texte optionnel après* ne s'affichent que s'il existe, dans la base de données, un élément correspondant à cette balise.

Notre exemple devient :

```
<BOUCLE_articles(ARTICLES){id_rubrique}>
[(#SURTITRE)<br />]
#TITRE<br />
</BOUCLE_articles>
```

On obtient alors le résultat recherché : s'il existe un surtitre pour cet article, il est affiché et suivi du
 ; s'il n'existe pas de surtitre, même le
 est occulté.

Utilisations avancées

A partir de [SPIP 1.8] on peut imbriquer des balises étendues les unes dans les autres. Ainsi, si dans notre exemple on voulait n'afficher le logo de l'article que si le surtitre est défini, on pourrait écrire :

```
<BOUCLE_articles(ARTICLES){id_rubrique}>
[[(#LOGO_ARTICLE)<br />](#SURTITRE)<br />]
</BOUCLE_articles>
```

Note:On ne peut jamais mettre une boucle dans le code optionnel d'une balise. Mais si on veut faire n'afficher une boucle qu'en fonction d'une certaine balise, on peut utiliser <INCLURE{...}> à l'intérieur d'un code optionnel.

Balises non ambiguës

Quand on imbrique des boucles les unes dans les autres, il peut arriver que deux boucles aient des balises homonymes.

Par exemple, dans le code suivant :

```
<BOUCLE_rubriques(RUBRIQUES){id_rubrique}>
     <BOUCLE_articles(ARTICLES){id_rubrique}>
          #TITRE
     </BOUCLE_articles>
</BOUCLE_rubriques>
```

la balise **#TITRE** désigne le titre d'un article. Ainsi, si on voulait afficher le titre de la rubrique à l'intérieur de la boucle **_articles**, on ne pourrait pas utiliser #TITRE.

Depuis [SPIP 1.8], on peut appeler une balise homonyme de l'une des boucles englobantes en explicitant le nom de la boucle à laquelle la balise appartient. Il faut alors spécifier le nom de la boucle entre le **#** et le nom de la balise.

On écrira alors la balise #BALISE de la boucle _boucle [1] de la façon suivante : **#*_boucle*:BALISE**. Par exemple :

```
<BOUCLE_rubriques(RUBRIQUES){id_rubrique}>
     <BOUCLE_articles(ARTICLES){id_rubrique}>
          #_rubriques:TITRE > #TITRE
     </BOUCLE_articles>
</BOUCLE_rubriques>
```

affichera le titre de la rubrique, puis le titre de l'article : la balise #TITRE pour la boucle *_rubriques* devient #_rubriques:TITRE pour ne pas être confondue avec la balise #TITRE de la boucle *_articles*.

Filtrer les résultats

Il est fréquent de vouloir modifier un élément tiré de la base de données, soit pour obtenir un affichage différent (par exemple, afficher le titre entièrement en majuscules), ou pour récupérer une valeur découlant de cet élément (par exemple, afficher le jour de la semaine correspondant à une date).

Dans SPIP, on peut directement appliquer des *filtres* aux éléments récupérés de la base de données, en les indiquant dans la syntaxe des balises SPIP, qui devient :

[*option avant* **(#BALISE**|filtre1|filtre2|...|filtre*n***)** *option après* **]**

La syntaxe est donc de faire suivre le nom de la balise, entre les parenthèses, par les filtres succesifs, séparés par une barre verticale (nommée habituellement *pipe*).

Voici quelques filtres fournis par SPIP :

- *majuscules*, passe le texte en majuscules (plus puissant que la fonction de PHP correspondante, qui ne fonctionne pas correctement avec les caractères accentués) ; par exemple :

[(#TITRE|majuscules)]

- *justifier*, affiche le texte en justification totale (c'est-à-dire <P align=justify>) ; par exemple :

[(#TEXTE|justifier)]

La présente documentation consacre un article aux différents filtres livrés avec SPIP.

Court-circuiter le traitement par SPIP

SPIP applique un traitement typographique à tous les textes tirés de la base de données. En particulier, il place des espaces insécables avant certains symboles (point-virgule, point d'interrogation, etc.), et analyse des raccourcis de mise en page.

Dans certains cas, vous pouvez avoir besoin de court-circuiter ce traitement, afin de récupérer directement le texte brut tel qu'il est placé dans la base de données. Pour cela, il suffit d'ajouter une astérisque (*) à la suite de la balise SPIP. Ce qui donne :

[*option avant* **(#BALISE***|filtre1|filtre2|...|filtre*n***)** *option après* **]**

Les paramètres des balises

Depuis [SPIP 1.8], certaines balises [2] acceptent des paramètres. On passera alors une liste de paramètres entre accolade «{» et «}» avec des virgules « , » pour séparer chaque paramètre. Par exemple : #ENV{lang,fr}.

Un paramètre peut être une constante ou une autre balise. Seulement les balises de forme simple peuvent être passées en paramètres (i.e. pas de code optionnel ou de filtres). On peut mettre les paramètres entre guillemets simples « '...' » si l'on ne veut pas qu'ils soient interprétés par SPIP.

Notes

[1] *Précision* : n'oubliez pas le cas échéant, l'underscore "_" initial dans le nom de la boucle si celui ci ne commence pas par un numéro.

[2] **#ENV** et **#EXPOSER**

La boucle ARTICLES

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

<BOUCLEn(ARTICLES){critères...}>

Les éléments contenus dans une telle boucle sont des articles.

Les critères de sélection

On utilisera l'un ou 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} sélectionne 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} sélectionne les articles contenus dans la rubrique dont l'identifiant est id_rubrique.

- {id_secteur} sélectionne 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} (depuis SPIP 1.4) sélectionne 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} sélectionne les articles correspondant à cet identifiant d'auteur (utile pour indiquer la liste des articles écrits par un auteur).

- {id_mot} sélectionne 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} (depuis SPIP 1.3) sélectionne respectivement 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} (depuis SPIP 1.4) 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 SPIP) cumulable avec le précédent {type_mot=yyyy}]

- {lang} (depuis SPIP 1.7.1) sélectionne les articles de la langue demandée dans l'adresse de la page.

- Les critères {date} (ou {date=...} ou {date==...}) permettent de sélectionner un article en fonction de la date passée dans l'URL (depuis SPIP 1.7.2).

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

Le statut de l'article

Comme toutes les boucles de SPIP, une boucle ARTICLES ne retourne que des articles *publiés* ; dans le cas où le site est réglé de manière à ne pas publier les articles « post-datés », un autre test est fait sur la date de l'article. Jusqu'à SPIP 1.8.2 il n'existait aucun moyen de débrayer ce système et d'afficher les articles « en cours de rédaction », « proposés à la publication » ou « refusés ». C'est désormais possible grâce au critère {statut} :

- **{statut=***prop/prepa/publie/refuse/poubelle***}** (depuis SPIP 1.8.2) sélectionne les articles en fonction de leur statut de publication :

  - {statut=prepa} sélectionne les articles en cours de rédaction dans l'espace privé ;
  - {statut=prop} sélectionne les articles proposés à la publication ;
  - {statut=publie} sélectionne les articles publiés sur le site, y compris les articles « post-datés » ;
  - {statut=refuse} sélectionne les articles qui ont été refusés à la publication ;
  - {statut=poubelle} sélectionne les articles qui ont été mis à la poubelle.

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.

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 affiche le surtitre.
- #TITRE affiche le titre de l'article.
- #SOUSTITRE affiche le soustitre.
- #DESCRIPTIF affiche le descriptif.
- #CHAPO affiche le texte d'introduction (chapeau).
- #TEXTE affiche le texte principal de l'article.
- #PS affiche le post-scriptum.
- Les balises de dates : #DATE, #DATE_REDAC, #DATE_MODIF sont explicitées dans la documentation sur « La gestion des dates ».
- #ID_RUBRIQUE affiche l'identifiant de la rubrique dont dépend l'article.
- #ID_SECTEUR affiche l'identifiant du secteur dont dépend l'article (le secteur étant la rubrique parente située à la racine du site).
- #NOM_SITE et #URL_SITE affichent le nom et l'url du « lien hypertexte » de l'article (si vous avez activé cette option).
- #VISITES affiche le nombre total de visites sur cet article.
- #POPULARITE affiche le pourcentage de popularité de cet article ; voir la documentation « La « popularité » des articles ».
- #LANG affiche la langue de cet article.

Les balises calculées par SPIP

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

- #URL_ARTICLE affiche l'URL de la page de l'article.
- #NOTES affiche les notes de bas de page (calculées à partir de l'analyse du texte).
- #INTRODUCTION (depuis SPIP 1.4) affiche le descriptif de l'article, sinon affiche les 600 premiers caractères du début de l'article (chapeau puis texte). Dans les versions antérieures à SPIP 1.3, ce sont systématiquement les premiers caractères de l'article (chapeau puis texte) qui sont pris en compte (le descriptif n'est pas utilisé).
- #LESAUTEURS affiche les auteurs de cet article, avec lien vers leur propre page publique (afin de pouvoir directement leur écrire ou de consulter la liste des articles qu'ils ont publié). Cela évite de créer une boucle AUTEURS pour obtenir le même résultat. Dans les versions antérieures à SPIP 1.9, cette balise affiche les auteurs de l'article avec lien vers leur adresse e-mail.
- #PETITION affiche le texte de la pétition si elle existe. Si elle existe mais que le texte est vide, retourne un espace (une chaîne non vide sans incidence dans une page html).
- #FORMULAIRE_SIGNATURE fabrique et affiche le formulaire permettant de signer la pétition associée à cet article.
- #FORMULAIRE_FORUM fabrique et affiche le formulaire permettant de poster un message répondant à cet article. Pour en savoir plus, voir aussi « Les formulaires ».
- #PARAMETRES_FORUM fabrique et affiche la liste des variables exploitées par le formulaire permettant de répondre à cet article. Par exemple :

```
[<a href="spip.php?page=forum&(#PARAMETRES_FORUM)">Répondre à cet article</a>]
```

Depuis SPIP 1.8.2 on peut lui passer un paramètre spécifiant l'adresse de retour après avoir posté le message. Par exemple : Répondre à cet article renverra le visiteur sur la page actuelle une fois que le message a été validé.

Historique : Dans les versions antérieures à SPIP 1.9 il aurait fallu écrire forum.php3? et non spip.php?page=forum&
De façon générale jusqu'à SPIP 1.9, les urls des pages générées par SPIP étaient de la forme http://monsite.net/xxx.php3 et non pas http://monsite.net/spip.php?page=xxx.

Les logos

- #LOGO_ARTICLE affiche le logo de l'article, éventuellement avec la gestion du survol.
- #LOGO_RUBRIQUE affiche le logo de la rubrique de l'article.
- #LOGO_ARTICLE_RUBRIQUE affiche le logo de l'article, éventuellement remplacé par le logo de la rubrique s'il n'existe pas de logo spécifique à 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 affiche le logo sans survol ;
- #LOGO_ARTICLE_SURVOL affiche le logo de survol.

La boucle RUBRIQUES

La boucle RUBRIQUES retourne une liste de... rubriques (étonnant, non ?)

<BOUCLEn(RUBRIQUES){critères...}>

Remarque. Une boucle RUBRIQUES n'affiche que des rubriques « actives », c'est-à-dire contenant des articles publiés, des documents joints (à partir de SPIP 1.4), 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. À partir de la version SPIP 1.7.1, il est possible de forcer l'affichage des rubriques vides (voir ci-dessous, le critère {tout}).

Les critères de sélection

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

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

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

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

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

- À partir de la version SPIP 1.4 , les rubriques peuvent 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*.

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

- {lang} (depuis SPIP 1.7.1) sélectionne les rubriques de la langue demandée dans l'adresse de la page.

- {tout} (depuis SPIP 1.7.1) sélectionne *toutes* les rubriques, c'est-à-dire : les rubriques vides *en plus* des rubriques contenant des éléments publiés.
On réservera ce choix à des besoins très spécifiques ; en effet, par défaut, SPIP n'affiche pas sur le site public les rubriques qui ne contiennent aucun élément actif, afin de garantir que le site ne propose pas de « culs de sac » (navigation vers des pages ne proposant aucun contenu).

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 la rubrique dans laquelle on se trouve déjà (utile avec meme_parent).

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 affiche le titre de la rubrique.

- #DESCRIPTIF affiche le descriptif.

- #TEXTE affiche le texte principal de la rubrique.

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

- #LANG affiche la langue de cette rubrique.

Les balises calculées par SPIP

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

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

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

- #URL_RUBRIQUE affiche l'URL de la page de la rubrique.

- #DATE (depuis SPIP 1.4) affiche la date de la dernière publication effectuée dans la rubrique et/ou ses sous-rubriques (articles, brèves...).

- #FORMULAIRE_FORUM fabrique et affiche le formulaire permettant de poster un message répondant à cette rubrique. Pour en savoir plus, voir aussi « Les formulaires ».

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

```
[<a href="spip.php?page=forum&(#PARAMETRES_FORUM)">Répondre à cette rubrique</a>]
```

Depuis SPIP 1.8.2 on peut lui passer un paramètre spécifiant l'adresse de retour après avoir posté le message. Par exemple : Répondre à cette rubrique renverra le visiteur sur la page actuelle une fois que le message a été validé.

Historique : Dans les versions antérieures à SPIP 1.9 il aurait fallu écrire forum.php3? et non spip.php?page=forum&
De façon générale jusqu'à SPIP 1.9, les urls des pages générées par SPIP étaient de la forme http://monsite.net/xxx.php3 et non pas http://monsite.net/spip.php?page=xxx.

- #FORMULAIRE_SITE (depuis SPIP 1.4) fabrique et affiche un formulaire 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 ».

Le logo

- #LOGO_RUBRIQUE le logo de la rubrique, éventuellement avec la gestion du survol. S'il n'y a pas de logo pour cette rubrique, SPIP 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***)]

Où :

- *alignement* est l'alignement du logo (left ou right)
- *adresse* est une url si on veut ajouter un lien directement sur le logo (par exemple #URL_RUBRIQUE).

Depuis SPIP 1.4 : #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 BREVES

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

<BOUCLEn(BREVES){critères...}>

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} sélectionne 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} sélectionne toutes les brèves contenues dans la rubrique en cours.
- {id_mot} (depuis SPIP 1.2) sélectionne 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} (depuis SPIP 1.3) sélectionne 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} (depuis SPIP 1.4) 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 ».
- {lang} sélectionne les brèves de la langue demandée dans l'adresse de la page.
- {recherche} sélectionne les brèves correspondant aux mots indiqués dans l'interface de recherche (moteur de recherche incorporé à SPIP). Voir la page consacrée au moteur de recherche.

Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

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 affiche le titre de la brève.
- #DATE affiche la date de publication de la brève.
- #TEXTE affiche le texte de la brève.
- #NOM_SITE affiche le nom du site indiqué en références.
- #URL_SITE affiche l'adresse (URL) du site indiqué en références.
- #ID_RUBRIQUE affiche l'identifiant de la rubrique dont dépend cette brève.
- #LANG affiche la langue de cette brève. Par défaut, la langue d'une brève est la langue du secteur dans lequel elle se trouve.

Les balises calculées par SPIP

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

- #NOTES affiche les notes de bas de page (calculées à partir de l'analyse du texte).
- #INTRODUCTION affiche les 600 premiers caractères du texte, les enrichissements typographiques (gras, italique) sont supprimés.
- #URL_BREVE affiche l'URL de la page de la brève.
- #FORMULAIRE_FORUM fabrique et affiche le formulaire permettant de poster un message répondant à cette brève. Pour en savoir plus, voir aussi « Les formulaires ».
- #PARAMETRES_FORUM fabrique la liste des variables exploitées par l'interface du formulaire permettant de répondre à cette brève.

```
[<a href="spip.php?page=forum&(#PARAMETRES_FORUM)">Répondre à cette bréve</a>]
```

Depuis SPIP 1.8.2 on peut lui passer un paramètre spécifiant l'adresse de retour après avoir posté le message. Par exemple : Répondre à cette bréve renverra le visiteur sur la page actuelle une fois que le message a été validé.

Historique : Dans les versions antérieures à SPIP 1.9 il aurait fallu écrire forum.php3? et non spip.php?page=forum&
De façon générale jusqu'à SPIP 1.9, les urls des pages générées par SPIP étaient de la forme http://monsite.net/xxx.php3 et non pas http://monsite.net/spip.php?page=xxx.

Le logo

- #LOGO_BREVE affiche 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 (depuis SPIP 1.4) affiche, si il existe, le logo de la brève ; si ce logo n'a pas été attribué, SPIP affiche le logo de la rubrique.

La boucle AUTEURS

La boucle AUTEURS, comme son nom l'indique, retourne une liste d'auteurs.

<BOUCLEn(AUTEURS){critères...}>

Si l'on ne précise pas de critère de sélection, la boucle retournera tous les auteurs *ayant un article publié*.

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.
- {lang} sélectionne les auteurs qui ont choisi, dans l'espace privé, la langue demandée dans l'adresse de la page. Si un auteur ne s'est jamais connecté dans l'espace privé, il ne sera pas trouvé par ce critère.
- {lang_select} Par défaut, une boucle AUTEURS affiche les balises et les chaînes localisées dans la langue du contexte (de la boucle englobante ou de l'url). Si on utilise ce critère, ces informations seront localisées dans la langue choisie par l'auteur.

Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

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 affiche le nom de l'auteur.
- #BIO affiche la biographie de l'auteur.
- #EMAIL affiche son adresse email.
- #NOM_SITE affiche le nom de son site Web.
- #URL_SITE affiche l'adresse (URL) de son site.
- #PGP affiche sa clé publique pour PGP.
- #LANG affiche la langue de l'auteur (c'est-à-dire celle qu'il a choisie dans l'espace privé).

Les balises calculées par SPIP

- #FORMULAIRE_ECRIRE_AUTEUR (depuis SPIP 1.4) fabrique et 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.
- #NOTES affiche les notes de bas de page (calculées à partir de l'analyse du texte).

- #URL_AUTEUR affiche l'adresse de la page spip.php?auteurxxx.

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 (depuis SPIP 1.6) 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...}>

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. Attention, il ne s'agit pas de messages des articles de cette rubrique, mais bien des messages de cette rubrique. En effet, il est possible d'activer dans l'espace privé des forums pour chaque rubrique.

- **{id_breve}** retourne les messages correspondant à cette brève.

- **{id_syndic}** retourne les messages correspondant à ce site.

- **{id_thread}** introduit dans [SPIP 1.8], retourne les messages appartenant à ce fil de discussion. *Note :* id_thread n'est rien d'autre que l'identifiant id_forum du message qui démarre le fil de discussion (aussi appelé « pied » de la discussion).

- **{id_parent}** retourne les messages dépendant d'un autre message. Indispensable pour gérer des fils de discussion (« *threads* ») dans les forums.

- **{id_enfant}** retourne le message dont dépend le message actuel (permet de « remonter » dans la hiérachie des fils de discussion). (SPIP 1.3)

- **{meme_parent}** retourne les autres messages répondant à un même message. (SPIP 1.3)

- **{plat}** ou **{tout}** : affiche tous les messages de forum sans prendre en compte leur hiérarchie : avec 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 par exemple d'afficher les messages par ordre strictement chronologique par exemple, ou de compter le nombre total de contributions dans un forum.

N.B. En l'absence de critère {id_forum} ou {id_parent}, lorsque {plat} n'est pas utilisé, seuls les messages n'ayant pas de parent (i.e. à la racine d'un thread) sont affichés.

- **{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.

- À partir de la version SPIP 1.4, les messages des forums peuvent être liées à des mots-clés. Les critères de mots-clés peuvent donc être désormais 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*.

Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

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_THREAD** introduit dans [SPIP 1.8], affiche l'identifiant du fil de discussion auquel appartient ce message. (Il s'agit de l'id_forum du *pied* de la discussion.)
- **#URL_FORUM** donne, depuis [SPIP 1.8], l'adresse canonique de la page qui affiche le message de forum (par exemple, avec les URLs normales de SPIP, article.php3?id_article=8#forum15 pour le message 15 associé à l'article 8).
- **#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** est l'identifiant de la rubrique à laquelle le message répond.
- **#ID_SYNDIC** est l'identifiant du site auquel 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.

Les balises calculées par SPIP

- **#FORMULAIRE_FORUM** fabrique l'interface permettant de poster un message de réponse. Pour en savoir plus, voir aussi « Les formulaires ».
- **#PARAMETRES_FORUM** fabrique la liste des variables exploitées par l'interface du formulaire permettant de répondre à ce message. Par exemple :

```
[<a href="spip.php?page=forum&(#PARAMETRES_FORUM)">Répondre à ce message</a>]
```

Depuis [SPIP 1.8.2] on peut lui passer un paramètre spécifiant l'adresse de retour après avoir posté le message. Par exemple : Répondre à ce message renverra le visiteur sur la page actuelle une fois que le message a été validé.

Historique : Dans les versions antérieures à [SPIP 1.9] il aurait fallu écrire forum.php3? et non spip.php?page=forum&

De façon générale jusqu'à [SPIP 1.9], les urls des pages générées par spip étaient de la forme http://monsite.net/xxx.php3 et non pas http://monsite.net/spip.php?page=xxx.

La boucle MOTS

La boucle MOTS retourne une liste de mots-clés.

<BOUCLEn(MOTS){critères...}>

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* SPIP 1.4 .
- {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 SPIP 1.4 .
- {id_breve} retourne les mots associés à une brève SPIP 1.4 .
- {id_syndic} retourne les mots associés à un site référencé SPIP 1.4 .
- {id_forum} retourne les mots associés à un message de forum SPIP 1.4 (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).

Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

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 affiche le titre (le mot-clé lui-même).
- #DESCRIPTIF affiche le descriptif du mot.
- #TEXTE affiche le texte associé au mot.
- #TYPE affiche la catégorie dans laquelle est installé ce mot-clé (par exemple, le mot-clé « France » pourrait être associé à la catégorie « Pays »).
- #LOGO_MOT SPIP 1.4 affiche le logo associé au mot-clé.
- #URL_MOT affiche l'adresse de ce mot

La boucle (GROUPES_MOTS)

D'une utilisation marginale, la boucle GROUPES_MOTS, introduite avec SPIP 1.5, mérite d'être citée ici :

elle permet, si vous avez plusieurs groupes de mots-clés, de sélectionner ces groupes, et d'organiser par exemple une page récapitulative de tous les mots-clés classés par groupe, puis par ordre alphabétique à l'intérieur de chaque groupe, par exemple via le code suivant :

```
<BOUCLE_groupes(GROUPES_MOTS){par titre}>
<h1>#TITRE</H1>
<BOUCLE_mots(MOTS){id_groupe}{par titre}{" - "}>
#TITRE
</BOUCLE_mots>
</BOUCLE_groupes>
```

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 DOCUMENTS

Introduite avec SPIP 1.4 , 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* (depuis la version 1.4, les images sont gérées, au niveau du programme, comme un genre spécifique de documents), 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_document} sélectionne le document dont l'identifiant est id_document. Comme l'identifiant de chaque document est unique, ce critère ne retourne qu'une ou zéro réponse.
- {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 »).

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*).

N.B. Dans les sites SPIP existant avant la version 1.4, l'habitude a été prise de ne pas pouvoir afficher les images qui ne sont pas insérées *à l'intérieur* du texte de l'article. De fait, si vous ajoutez un boucle DOCUMENTS en *mode=image* sur un site déjà existant, vous risquez de voir réapparaître dans cette boucle des images qui n'étaient pas destinées à être publiées sur le site public. Donc, n'utilisez une telle boucle que sur un site créé avec la version 1.4, ou bien procédez avec beaucoup de précautions (vérifiez les anciens articles pour éviter la publication d'images parasites).

- {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 ».

- {distant} permet, depuis SPIP 1.8.2, de sélectionner les documents selon qu'ils soient distant ou non. C'est à dire stockés sur un autre site ou téléchargés dans l'espace web du site. On précisera {distant=oui} et {distant=non} respectivement. (voire SPIP 1.8.2)

- {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 [1].

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, SPIP 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.

- #FICHIER [SPIP 1.8.2] affiche le nom de fichier du document. Une utilisation intéressante de cette balise est combinée avec le filtre image_reduire, dans le cadre d'un portfolio pour afficher une réduction de l'image plutôt que de son logo ; par exemple en utilisant :

```
[<a
 href="#URL_DOCUMENT">(#FICHIER|image_reduire{500}) </a>]
```

- #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.

- #MIME_TYPE affiche le type MIME du fichier — par exemple : image/jpeg —, *cf.* Type de média internet.

- #DATE est la date de mise en ligne du document. (Modifiable après l'ajout). Voir « La gestion des dates » pour des détails sur l'utilisation du champ #DATE.

- #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)]
```

- #DISTANT est une balise qui affiche « oui » ou « non » selon que le document est distant (référencé par une url) ou pas.

Notes

[1] Si on utilise un critère avec un nom ({doublons unnom}), celui ci n'exclura pas les documents intégrés dans le texte de l'article.

La boucle SITES (ou SYNDICATION)

La boucle SITES retourne une liste de sites référencés.

<BOUCLEn(SITES){critères...}>

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.

> *Historique :* Avant SPIP 1.3, cette boucle était nommée SYNDICATION, car seuls des sites syndiqués pouvaient être référencés.
> <BOUCLEn(SYNDICATION){critères...}>
> Les deux dénominations sont rigoureusement équivalentes (mais « SITES » correspond mieux au fait que, depuis la version 1.3 de SPIP, il s'agit d'un système de *référencement* de sites, la syndication étant une option).

Les critères de sélection

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

- {tout} sélectionne *tous* les sites référencés.
- {id_syndic} sélectionne le site référencé dont l'identifiant est *id_syndic*.
- {id_rubrique} sélectionne les sites référencés dans cette rubrique.
- {id_secteur} sélectionne les sites référencés dans ce secteur.
- {id_mot} (depuis SPIP 1.3) sélectionne 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} (depuis SPIP 1.3) sélectionne 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} (depuis SPIP 1.4) 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 ».

Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

- {syndication=oui}, ou {syndication=non} (depuis SPIP 1.3) permet de n'afficher que les sites référencés faisant l'objet d'une syndication, ou les sites non syndiqués.
- {moderation=oui} (depuis SPIP 1.4) affiche les sites syndiqués dont les liens sont bloqués a priori (« modérés ») ; l'inverse de ce critère est {moderation!=oui}.

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_site}).

- #ID_SYNDIC affiche l'identifiant unique du site référencé. Par exemple pour renvoyer vers la page décrivant le site *(site.html sur la /dist)* avec le code suivant :

```
<BOUCLE_sites(SITES) {id_rubrique} {par nom_site}>
<li><a href="[(#ID_SYNDIC|generer_url_site)]">#NOM_SITE</a>
</BOUCLE_sites>
```

- #NOM_SITE affiche le nom du site référencé.
- #URL_SITE affiche l'adresse (URL) du site référencé.
- #DESCRIPTIF affiche le descriptif du site référencé.
- #ID_RUBRIQUE affiche le numéro de la rubrique contenant ce site.
- #ID_SECTEUR affiche le numéro de la rubrique-secteur (à la racine du site) contenant ce site.

Autres balises

- #LOGO_SITE affiche le logo attribué au site.
- #URL_SYNDIC affiche l'adresse (URL) du fichier de syndication de ce site.
- #FORMULAIRE_FORUM fabrique et affiche le formulaire permettant de poster un message de forum à propos de ce site. Pour en savoir plus, voir aussi « Les formulaires ».
- #PARAMETRES_FORUM fabrique la liste des variables exploitées par l'interface du formulaire permettant de poster un message de forum à propos de ce site. Par exemple : [Répondre à ce forum]

Depuis [SPIP 1.8.2] on peut lui passer un paramètre spécifiant l'adresse de retour après avoir posté le message. Par exemple : Répondre à ce message renverra le visiteur sur la page actuelle une fois que le message a été validé.

Historique : Dans les versions antérieures à SPIP 1.9 il aurait fallu écrire forum.php3? et non spip.php?page=forum&

De façon générale jusqu'à SPIP 1.9, les urls des pages générées par SPIP étaient de la forme http://monsite.net/xxx.php3 et non pas http://monsite.net/spip.php?page=xxx.

La boucle SYNDIC_ARTICLES

La boucle SYNDIC_ARTICLES retourne une liste des articles des sites syndiqués.

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

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).

> (SPIP 1.3) À partir de la version 1.3 de SPIP, la boucle SITES (ou SYNDICATION) n'affiche plus uniquement des sites syndiqués, mais plus généralement des sites référencés (la syndication de certains sites référencés étant une option). On pourra donc, pour obtenir une présentation graphique plus précise, utiliser une boucle SYNDIC_ARTICLES uniquement à l'intérieur d'une boucle SITES utilisant le critère {syndication=oui}.

Les critères de sélection

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

- {tout}, 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.

Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

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.
- #TITRE affiche 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 affiche l'adresse (URL) de l'article syndiqué (sur son site original).
- #DATE affiche la date de publication de cet article.
- #LESAUTEURS affiche les auteurs de l'article syndiqué.

- #DESCRIPTIF affiche le descriptif de l’article syndiqué.
- **#NOM_SITE** affiche le nom du site syndiqué contenant cet article.
- **#URL_SITE** affiche l’adresse (URL) du site

La boucle SIGNATURES

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

<BOUCLEn(SIGNATURES){critères...}>

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 sélectionnés dans l'intégralité du site.
- {id_signature}, la signature correspondant à l'identifiant courant.
- {id_article} retourne les signatures de la pétition de cet article.

Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent.

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

- {par nom_email} classe les résultats selon le #NOM du signataire ;
- {par ad_email} classe selon l'#EMAIL du signataire.

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 affiche l'identifiant de l'article pour cette pétition.
- #DATE affiche la date de publication.
- #MESSAGE affiche le texte du message.
- #NOM affiche le nom de l'auteur du message.
- #EMAIL affiche l'adresse email de l'auteur.
- #NOM_SITE affiche le nom du site Web indiqué par l'auteur.
- **#URL_SITE** affiche l'adresse (URL) de ce site Web.

La boucle HIERARCHIE

La boucle HIERARCHIE retourne la liste des RUBRIQUES qui mènent de la racine du site à la rubrique ou à l'article en cours.

<BOUCLEn(HIERARCHIE){critères...}>

Les critères de sélection

On utilisera obligatoirement l'un des deux critères suivants pour indiquer comment on sélectionne les éléments :

- {id_article} retourne la liste des rubriques depuis la racine jusqu'à la rubrique contenant l'article correspondant à cet identifiant.
- {id_rubrique} retourne la liste des rubriques depuis la racine jusqu'à la rubrique correspondant à cet identifiant (exclue).

Note : Depuis [SPIP 1.8], {tout} permet d'obtenir **aussi** la rubrique correspondant à l'identifiant spécifié.

Les critères {id_article} ou {id_rubrique} ne peuvent pas être utilisés avec une comparaison. Par exemple, <BOUCLE_hi(HIERARCHIE) {id_article=12}> retournera une erreur.

> *Attention :* cette boucle sera obligatoirement placée à l'intérieur d'une boucle ARTICLES ou RUBRIQUES — elle ne va pas par elle-même « chercher » l'id_article ou id_rubrique indiquée dans l'URL. (Le même principe vaut pour les boucles HIERARCHIE des squelettes inclus par la commande <INCLURE{fond=xxx}> ou <INCLURE(xxx.php3)> pour les version antérieure à 1.9)

Les critères d'affichage

Depuis [SPIP 1.8], tous les critères de La boucle RUBRIQUES peuvent être utilisés avec cette boucle, y compris les critères de tri (il devient possible par exemple de trier une <BOUCLE_x(HIERARCHIE) {id_article}{par hasard}>).

> *Historique :* Jusqu'à la version [SPIP 1.7.2], Les critères communs à toutes les boucles ne s'appliquent pas tous à ce type de boucle. Seuls les critères {"inter"} et {a,b} étaient utilisables.

Les balises de cette boucle

Les éléments obtenus avec une boucle HIERARCHIE sont des rubriques. On peut donc utiliser toutes les balises proposées pour les boucles RUBRIQUES.

Les critères communs à toutes les boucles

Certains critères s'appliquent à (presque) tous les types de boucles. Ce sont des critères destinés à restreindre le nombre de résultats affichés ou à indiquer l'ordre d'affichage. On peut sans difficulté combiner plusieurs de ces critères de sélection.

Classer les résultats

{par *critère_de_classement***}** indique l'ordre de présentation des résultats. Ce critère de classement correspond à l'une des balises tirées de la base de données pour chaque type de boucle. Par exemple, on pourra classer les articles {par date}, {par date_redac} ou {par titre}. (Notez que, si les balises sont en majuscules, les critères de classement sont en minuscules.)

Cas particulier : {par hasard} permet d'obtenir une liste présentée dans un ordre aléatoire.

Inverser le classement. De plus, {inverse} provoque l'affichage du classement inversé. Par exemple {par date} commence par les articles les plus anciens ; avec {par date}{inverse} on commence la liste avec les articles les plus récents.

Depuis SPIP 1.9, le critère *inverse* peut prendre en paramètre n'importe quelle balise pour varier dynamiquement le sens du tri. Par exemple, il est possible d'écrire : <BOUCLE_exemple(ARTICLES){par #ENV{tri}}{inverse #ENV{senstri}}>, ce qui permet de choisir la colonne de tri et le sens du tri par l'url (&senstri=1 ou &senstri=0)

Classer par numéro. [SPIP 1.3] Lorsqu'on réalise le classement selon un élément de texte (par exemple le *titre*), le classement est réalisé par ordre *alphabétique*. Cependant, pour forcer un ordre d'affichage, on peut indiquer un numéro devant le titre, par exemple : « 1. Mon premier article », « 2. Deuxième article », « 3. Troisième... », etc ; avec un classement alphabétique, le classement de ces éléments donnerait la série « 1, 10, 11, 2, 3... ». Pour rétablir le classement selon les numéros, on peut utiliser le critère :

{par num critère**}**

Par exemple :

```
<BOUCLE_articles(ARTICLES){id_rubrique}{par date}{inverse}>
```

affiche les articles d'une rubrique classés selon l'ordre chronologique inversé (les plus récents au début, les plus anciens à la fin), et :

```
<BOUCLE_articles(ARTICLES){id_rubrique}{par titre}>
```

les affiche selon l'ordre alphabétique de leur titre ; enfin :

```
<BOUCLE_articles(ARTICLES){id_rubrique}{par num titre}>
```

les affiche selon l'ordre du numéro de leur titre (remarque : l'option {par num titre} ne fonctionne pas pour les plus anciennes versions de MySQL, antérieures à la version 3.23).

```
<BOUCLE_articles(ARTICLES){id_rubrique}{par multi titre}>
```

Dans le cadre d'un site multilingue le critère **{par multi** critère**}** permet de trier par ordre alphabétique dans

chaque langue. Sans l'ajout de "multi" la boucle renvoie le même classement pour chaque langue.

Classer selon plusieurs critères A partir de SPIP 1.8, SPIP 1.8.1, on peut classer selon plusieurs critères : **{par** critère1**,** critère2**}**. On indique ainsi des ordres de classement consécutifs. Les résultats seront d'abord triés selon le *critère1*, puis le *critère2* pour les résultats ayant le même *critère1*. On peut spécifier autant de critères que nescessaire.

Par exemple {par date, titre} triera les résultats par *date* puis les résultats ayant la même *date* seront triés par *titre*.

Avec [SPIP 1.8.2] on peut spécifier plusieurs critères **{par** ...**}** pour une boucle pour arriver au même résultat. Par exemple : {par date} {par titre} est équivalent à l'exemple précédent.

> *Remarque :* Quand on utilise plusieurs critères de tri, le critère **{inverse}** ne s'applique qu'au critère de tri placé juste avant.
>
> C'est pourquoi [SPIP 1.8.2] introduit la notation **{!par ...}** qui inverse un critère de tri en particulier. Par exemple : {!par date} {par num titre} tri par *date* décroissantes puis par numéros croissants dans le *titre* pour les résultats ayant la même *date*.

Comparaisons, égalités

{*critère* < *valeur***}** Comparaison avec une valeur fixée (on peut utiliser « > », « < », « = », « >= », « <= ». Tous les *critères de classement* (tels que tirés de la base de données) peuvent également être utilisés pour limiter le nombre de résultats.

La valeur à droite de l'opérateur peut être :

- Une valeur constante fixée dans le squelette. Par exemple :

```
<BOUCLE_art(ARTICLES){id_article=5}>
```

affiche l'article dont le numéro est 5 (utile pour mettre en vedette un article précis sur la page d'accueil).

```
<BOUCLE_art(ARTICLES){id_secteur=2}>
```

affiche les articles du secteur numéro 2.

- A partir de [SPIP 1.8], une balise disponible dans le contexte de la boucle. Par exemple :

```
<BOUCLE_art(ARTICLES){id_article=5}>
<BOUCLE_titre(ARTICLES) {titre=#TITRE}>
...
</BOUCLE_titre>
</BOUCLE_art>
```

sert à trouver les articles qui ont le même titre que l'article 5.

> *Attention :* On ne peut utiliser q'une balise simple. Il n'est pas permis de la filtrer ou de mettre du code optionnel.
>
> Spécialement, si on veut utiliser la balise **#ENV** — ou tout autre balise prenant des paramètres —, on doit utiliser la notation : {titre = #ENV{titre}} et **pas** : {titre = [(#ENV{titre})]}.

Expressions régulières :

Très puissant (mais nettement plus complexe à manipuler), le terme de comparaison « == » introduit une comparaison selon une expression régulière. Par exemple :

```
<BOUCLE_art(ARTICLES){titre==^[aA]}>
```

sélectionne les articles dont le titre commence par « a » ou « A ».

Négation :

A partir de [SPIP 1.2] On peut utiliser la notation {xxx != yyy} et {xxx !== yyy}, le ! correspondant à la négation (opérateur logique NOT).

```
<BOUCLE_art(ARTICLES){id_secteur != 2}>
```

sélectionne les articles qui n'appartiennent pas au secteur numéro 2.

```
<BOUCLE_art(ARTICLES){titre!==^[aA]}>
```

sélectionne les articles dont le titre ne commence pas par « a » ou « A ».

Affichage en fonction de la date

Pour faciliter l'utilisation des comparaisons sur les dates, on a ajouté des critères :

- age et age_redac correspondent respectivement à l'ancienneté de la publication et de la première publication d'un article, en jours : {age<30} sélectionne les éléménts publiés depuis un mois ;
- les critères mois, mois_redac, annee, annee_redac permettent de comparer avec des valeurs fixes ({annee<=2000} pour les éléments publiés avant la fin de l'année 2000).

On peut combiner plusieurs de ces critères pour effectuer des sélections très précises. Par exemple :

```
<BOUCLE_art(ARTICLES){id_secteur=2}{id_rubrique!=3}{age<30}>
```

affiche les articles du secteur 2, à l'exclusion de ceux de la rubrique 3, et publiés depuis moins de 30 jours.

Astuce. Le critère age est très pratique pour afficher les articles ou les brèves dont la date est située « dans le futur », avec des valeurs négatives (à condition d'avoir sélectionné, dans la Configuration précise du site, l'option « Publier les articles post-datés »). Par exemple, ce critère permet de mettre en valeur des événements futurs. {age<0} sélectionne les articles ou les brèves dont la date est située dans le futur (« après » aujourd'hui)...

[SPIP 1.3] *Âge par rapport à une date fixée.* Le critère age est calculé par rapport à la date d'aujourd'hui (ainsi {age<30} correspond aux articles publiés depuis moins d'un mois par rapport à aujourd'hui). Le critère **age_relatif** compare la date d'un article ou d'une brève à une date « courante » ; par exemple, à l'intérieur d'une boucle ARTICLES, on connaît déjà une date pour chaque résultat de la boucle, on peut donc sélectionner par rapport à cette date (et non plus par rapport à aujourd'hui).

```
<BOUCLE_article_principal(ARTICLES){id_article}>

 <h1>#TITRE</h1>

 <BOUCLE_suivant(ARTICLES){id_rubrique}{age_relatif<=0}{exclus}{par date}{0,1}>
 Article suivant: #TITRE
 </BOUCLE_suivant>

</BOUCLE_article_principal>
```

Par exemple :

la BOUCLE_suivant affiche un seul article de la même rubrique, classé par date, dont la date de publication est inférieure ou égale à la date de l'« article_principal » ; c'est-à-dire l'article de la même rubrique publié après l'article principal.

De plus amples informations sur l'utilisation des dates se trouvent dans l'article sur « La gestion des dates ».

Affichage d'une partie des résultats

- {branche} A partir de [SPIP 1.8.2], limite les résultats — pour des boucles ayant un #ID_RUBRIQUE — à la branche actuelle (la rubrique actuelle et ses sous-rubriques). Par exemple :
<BOUCLE_articles(ARTICLES) {branche}> retournera tous les articles de la rubrique actuelle et de ces sous-rubriques,
<BOUCLE_articles(ARTICLES) {!branche}> retournera tous les articles qui ne sont pas dans la rubrique actuelle ou ses sous-rubriques,

  On peut utiliser le critère **{branche?}** *optionnel* pour ne l'appliquer que si une rubrique est sélectionnée dans le contexte (une boucle englobante ou l'url fournie un id_rubrique). Par exemple :
<BOUCLE_articles(ARTICLES) {branche?}> retournera tous les articles de la rubrique actuelle et de ces sous-rubriques si il y a un id_rubrique dans le contexte, sinon, tous les articles du site.

- {doublons} ou {unique} (ces deux critères sont rigoureusement identiques) permettent d'interdire l'affichage des résultats déjà affichés dans d'autres boucles utilisant ce critère.

historique : A partir de [SPIP 1.2] et jusqu'à [SPIP 1.7.2], seules les boucles ARTICLES, RUBRIQUES, DOCUMENTS et SITES acceptaient ce critère.

- {doublons *xxxx*} à partir de [SPIP 1.8], on peut avoir plusieurs jeux de critères {doublons} indépendants. Les boucles ayant {doublons rouge} n'auront aucune incidence sur les boucles ayant {doublons bleu} comme critère.

- {exclus} permet d'exclure du résultat l'élément (article, brève, rubrique, etc.) 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 trouve déjà.

- **{***xxxx* **IN** *a,b,c,d***}** à partir de [SPIP 1.8], limite l'affichage aux résultats ayant le critère *xxxx* égal à *a*, *b*, *c* **ou** *d*. Les résultats sont triés dans l'ordre indiqué (sauf demande explicite d'un autre critère de tri). Il est aussi possible de sélectionner des chaînes de caractères, par exemple avec {titre IN 'Chine', 'Japon'}.

Avec SPIP 1.9, les balises sont admises dans les arguments de IN, et notamment la balise ENV, à laquelle sont appliqués les filtres d'analyse pour assurer que la requête SQL sera bien écrite. De manière dérogatoire, SPIP testera si l'argument de ENV désigne un tableau (venant par exemple de saisies de formulaire dont l'attribut name se termine par []). Si c'est le cas, et si les filtres d'analyse ont été désactivés en suffixant cette balise par une double étoile, alors chaque élément du tableau sera considéré comme argument de IN, SPIP appliquant les filtres de sécurité sur chacun d'eux.

Le squelette standard formulaire_forum_previsu fournit un exemple d'utilisation avec une boucle MOTS ayant le critère {id_mot IN #ENV**{ajouter_mot}} : cette boucle sélectionne seulement les mots-clés appartenant à un ensemble indiqué dynamiquement. Ici, cet ensemble aura été construit par le formulaire du squelette standard choix_mots, qui utilise des attributs name=ajouter_mot[].

- **{*a*,*b*}** où *a* et *b* sont des chiffres. Ce critère permet de limiter le nombre de résultats. *a* indique le résultat à partir duquel on commence l'affichage (attention, le premier résultat est numéroté 0 - zéro) ; *b* indique le nombre de résultats affichés.

Par exemple {0,10} affiche les dix premiers résultats ; {4,2} affiche les deux résultats à partir du cinquième (inclus).

- **{debut_*xxx*,*b*}** est une variante très élaborée de la précédente. Elle permet de faire commencer la limitation des résultats par une variable passée dans l'URL (cette variable remplace ainsi le *a* que l'on indiquait précédemment). C'est un fonctionnement un peu compliqué, que fort heureusement on n'a pas besoin d'utiliser trop souvent.

La variable passée dans l'URL commence forcément par debut_*xxx* (où *xxx* est un mot choisi par le webmestre) . Ainsi, pour une page dont l'URL est :

spip.php?page=petition&id_article=13&debut_signatures=200

avec un squelette (petition.html) contenant par exemple :

```
<BOUCLE_signatures(SIGNATURES){id_article}{debut_signatures,100}>
```

on obtiendra la liste des 100 signatures à partir de la 201-ième [rappel]. Avec l'URL :

spip.php?page=petition&id_article=13&debut_signatures=300

on obtient la liste des 100 signatures à partir de la 301-ième [rappel].

- **{*a*,n-*b*}** à partir de [SPIP 1.8], est une variante de {a,b} qui limite l'affichage en fonction du nombre de résultats dans la boucle. *a* est le résultat à partir duquel commencer à faire l'affichage ; *b* indique le nombre de résultats à **ne pas afficher** à la fin de la boucle.

  {0,n-10} affichera tous les résultats de la boucle sauf les 10 derniers.

- **{n-*a*,*b*}** à partir de [SPIP 1.8], est le pendant de {a, n-b}. On limite à *b* résultats en commençant l'affichage au a^e résultat avant la fin de la boucle.

Par exemple : {n-20,10} affichera au 10 résultats en partant du 20^e résultat avant la fin de la boucle.

- **{*a*/*b*}** où *a* et *b* sont des chiffres. Ce critère permet d'afficher une partie *a* (proportionnellement) des résultats en fonction d'un nombre de « tranches » *b*.

Par exemple : {1/3} affiche le premier tiers des résultats. Ce critère est surtout utile pour présenter des listes sur plusieurs colonnes. Pour obtenir un affichage sur deux colonnes, il suffit de créer une première boucle, affichée dans une case de tableau, avec le critère {1/2} (la première moitié des résultats), puis une seconde boucle dans une seconde case, avec le critère {2/2} (la seconde moitié des résultats).

Attention. L'utilisation du critère {doublons} avec ce critère est périlleuse. Par exemple :

```
<BOUCLE_prem(ARTICLES){id_rubrique}{1/2}{doublons}>
 <li> #TITRE
</BOUCLE_prem>
<BOUCLE_deux(ARTICLES){id_rubrique}{2/2}{doublons}>
 <li> #TITRE
</BOUCLE_deux>
```

n'affichera pas tous les articles de la rubrique ! Imaginons par exemple qu'il y ait au total 20 articles dans notre rubrique. La BOUCLE_prem va afficher la première moitié des articles, c'est-à-dire les 10 premiers, et interdire (à cause de {doublons}) de les réutiliser. La BOUCLE_deux, elle, va récupérer la deuxième moitié des articles de cette rubrique *qui n'ont pas encore été affichés* par la BOUCLE_prem ; donc, la moitié des 10 articles suivants, c'est-à-dire les 5 derniers articles de la rubrique. Vous avez donc « perdu » 5 articles dans l'opération...

Affichage *entre* les résultats

{"*inter*"**}** permet d'indiquer un code HTML (ici, *inter*) inséré *entre* les résultats de la boucle. Par exemple, pour séparer une liste d'auteurs par une virgule, on indiquera :

```
<BOUCLE_auteurs(AUTEURS){id_article}{", "}>
```

Divers

{logo} permet de ne sélectionner que les articles (ou rubriques, etc) qui disposent d'un logo. Il fonctionne aussi dans la boucle (HIERARCHIE). Le critère inverse {!logo} liste les objets qui n'ont pas de logo.

Notes

[rappel] le premier résultat est numéroté 0, donc le 200^{e} résultat représente réellement la 201 ^{e} signature

Les balises propres au site

Les balises suivantes sont disponibles à n'importe quel endroit du squelette, même en dehors d'une boucle (hors « contexte »).

Balises définies à la configuration

Le contenu de ces balises est défini dans l'espace privé, lors de la configuration de votre site.

- #NOM_SITE_SPIP affiche le nom du site.

- #URL_SITE_SPIP affiche l'adresse du site. Elle ne comprend pas le / final, ainsi vous pouvez créer un lien du type #URL_SITE_SPIP/sommaire.php3

- #DESCRIPTIF_SITE_SPIP (depuis SPIP 1.9) affiche, comme son nom l'indique, le descriptif du site, que l'on renseigne dans la page de configuration générale du site.

- #EMAIL_WEBMASTER (depuis SPIP 1.5) affiche l'adresse du webmestre. Par défaut, SPIP prend l'adresse de celui qui a installé le site (le premier administrateur).

- #LOGO_SITE_SPIP (depuis SPIP 1.8, SPIP 1.8.1) affiche le logo du site. Depuis SPIP 1.9 cette balise renvoie le logo du site 0. Il ne faut pas confondre avec le logo de la racine, aussi désigné sous le nom de logo standard des rubriques, c'est-à-dire celui de la rubrique 0.

- #CHARSET (depuis SPIP 1.5) affiche le jeu de caractères utilisé par le site. Sa valeur par défaut est iso-8859-1, jeu de caractères dit « iso-latin » [1].

- #LANG (depuis SPIP 1.7, SPIP 1.7.2) : utilisée en dehors des boucles ARTICLES, RUBRIQUES, BREVES et AUTEURS, cette balise affiche la langue principale du site.

- #LANG_DIR, #LANG_LEFT, #LANG_RIGHT (depuis SPIP 1.7, SPIP 1.7.2) : ces balises définissent le sens d'écriture de la langue du contexte actuel (par exemple, de l'article qu'on est en train d'afficher). Voir l'article « Réaliser un site multilingue » pour plus d'information.

- #MENU_LANG (et #MENU_LANG_ECRIRE) (depuis SPIP 1.7, SPIP 1.7.2) : ces balises fabriquent et affichent un menu de langues permettant au visiteur d'obtenir la page en cours dans la langue choisie. La première balise affiche la liste des langues du site ; la seconde la liste des langues de l'espace privé (elle est utilisée sur la page de connexion à l'espace privé).

Balises de mise en page

- SPIP 1.8.2 introduit la balise **#DOSSIER_SQUELETTE** pour pouvoir développer des squelettes facilement transportables et échangeables. Elle permet d'obtenir le chemin du dossier dans lequel est installé le squelette utilisé.

  On peut ainsi placer les fichiers « accessoires » (feuille de style, javascript, etc...) au squelette dans le répertoire du squelette et donc simplement distribuer ce dossier pour échanger ses squelettes. On écrira donc, par exemple, pour inclure une feuille de style du répertoire squelette :

```
<link rel="stylesheet" href="#DOSSIER_SQUELETTE/mon_style.css" type="text/css" />
```

  Depuis SPIP 1.9, la balise #CHEMIN remplace et améliore #DOSSIER_SQUELETTE. #CHEMIN{xxx} donnera le chemin complet vers le fichier xxx, qu'il se trouve à la racine, dans le dossier des squelettes, dans dist/ etc.

```
<link rel="stylesheet" href="#CHEMIN{mon_style.css}" type="text/css" />
```

- #PUCE (depuis SPIP 1.5) qui affiche devinez-quoi...

- #FORMULAIRE_ADMIN (depuis SPIP 1.5) est une balise optionnelle qui permet de placer les boutons d'administration (« recalculer cette page », etc.) dans ses squelettes. Lorsqu'un administrateur parcours le site public, si cette balise est présente, elle sera remplacée par les boutons d'administration, sinon, les boutons seront placés à la fin de la page.

Depuis SPIP 1.8, SPIP 1.8.1, on peut aussi modifier la feuille de style *spip_admin.css* pour contrôler la position des boutons.

- #DEBUT_SURLIGNE, #FIN_SURLIGNE sont deux balises qui indiquent à SPIP dans quelle partie de la page colorer les mots clefs recherchés.

Quand l'utilisateur arrive sur une page depuis une page de recherche SPIP, les mots clefs cherchés sont colorés automatiquement dans la page trouvée. SPIP ne distingue pas, par exemple, entre un menu, du code javascript ou le texte de l'article, il colore les mots partout, ce qui peut être moche ou même créer des problèmes dans les scripts. On peut utiliser ces deux balises pour limiter la coloration à une partie de la page. Par exemple :

```
<BOUCLE_article(ARTICLES) {id_article}>
<INCLURE(menu.php3) {id_rubrique}>
#DEBUT_SURLIGNE
<h1>#TITRE</h1>
#CHAPO
#TEXTE
#NOTES
#FIN_SURLIGNE
</BOUCLE_article>
```

- La balise #INSERT_HEAD (depuis SPIP 1.9.1) doit se situer entre les balises <head> et </head> de vos squelettes. Elle permet à SPIP, ainsi qu'aux plugins éventuels, d'ajouter du contenu entre ces deux balises html.

Balises techniques

Attention, ces balises s'adressent à des utilisateurs avertis de SPIP.

- La balise #REM ne produit aucun affichage : elle permet de commenter le code des squelettes, de cette façon : [(#REM) Ceci est un commentaire.]. Ces commentaires n'apparaissent pas dans le code généré pour le site public.

- #SELF (depuis SPIP 1.8, SPIP 1.8.1) retourne l'URL de la page appelée, nettoyée des variables propres à l'exécution de SPIP. Par exemple, pour une page avec l'url : article.php3?id_article=25&var_mode=recalcul la balise #SELF retournera : article.php3?id_article=25

Par exemple pour faire un formulaire :

```
<form action="#SELF" method="get">
```

Remarque : la balise #SELF représentant l'adresse de la page, elle n'est pas compatible avec les <INCLURE()> (sauf si le $delais de l'inclusion est mis à 0).

- #URL_PAGE (depuis SPIP 1.9) retourne une url de type « page » (cf. les urls de spip), vers la page passée en paramètre et qui pourra être utilisée dans un lien. Par exemple, pour accéder à la page générée par le squelette toto.html, située dans votre dossier-squelette, #URL_PAGE{toto} généra automatiquement l'url spip.php?page=toto. Un second paramètre est autorisé pour ajouter des paramètres à l'url. Exemple #URL_PAGE{toto,id_article=#ID_ARTICLE} génèrera l'url spip.php?page=toto&id_article=XXX.

- [(#ENV{*xxxx*,*zzzz*})] (depuis SPIP 1.8, SPIP 1.8.1) permet d'accéder à la variable de nom *xxxx*

passée par la requête HTTP. *zzzz* est une partie optionnelle qui permet de retourner une valeur même si la variable *xxxx* n'existe pas.

Par défaut, la balise #ENV est filtrée par htmlspecialchars. Si on veut avoir le résultat brut, l'étoile « * » peut être utilisée comme pour les autres balises : [(#ENV*{xxxx})].

Par exemple pour limiter la liste d'auteurs affichés :

```
<BOUCLE_auteurs(AUTEURS) {nom == #ENV{lettre,^A}}>
```

Retourne la liste d'auteur ayant le nom correspondant à l'expression régulière passé dans l'url par la variable lettre (liste_auteur.php3?lettre=^Z) ou les auteurs qui ont un nom commençant par un 'A' s'il n'y a pas de variable dans l'url.

- La balise #SPIP_CRON (depuis SPIP 1.8, SPIP 1.8.1) est liée à la gestion par SPIP des calculs qu'il doit faire périodiquement (statistiques, indexation pour le moteur de recherche, syndication de sites etc.).

Si cette balise n'est pas présente sur le site, le moteur de SPIP effectue ses calculs, en temps utile, après avoir envoyé une page à un visiteur ; malheureusement php ne permet pas de fermer la connexion à la fin de la page, et dans certains cas cela peut conduire certains visiteurs malchanceux (ceux dont le passage déclenche une procédure un peu longue, notamment la syndication) à constater une certaine lenteur dans l'affichage de la page demandée.

La balise #SPIP_CRON permet de contourner ce problème : son rôle est de générer un marqueur invisible dont la propriété « background » pointe sur le script spip_background.php3 ; ce script à son tour effectue les calculs nécessaires « en tâche de fond », et renvoie une image transparente de 1×1 pixel. Cette astuce permet donc d'éviter tout sentiment de « ralentissement » en déportant les éventuelles lenteurs sur un script annexe.

A noter : cette balise n'est pas stratégique, et sa présence ou son absence ne modifient en rien la régularité du calcul des tâches périodiques du site.

- La balise #SET{*variable*,*valeur*} et son pendant #GET{*variable*} ont été introduites par SPIP 1.9.1. La balise #SET{xxx,yyy} affecte une valeur yyy à une variable xxx **propre au squelette calculé**. Cette valeur peut être récupérée par la balise #GET{xxx}. Les variables créées ainsi ne sont pas transmises au squelette inclus.

Attention ! Si l'on affecte une valeur à une variable dans la partie facultative avant d'une boucle, il ne sera pas possible de récupérer cette valeur dans la boucle. Cela tient à la manière dont Spip calcule les squelettes.

- La balise #HTTP_HEADER{*argument*} (depuis SPIP 1.9) permet de modifier l'entête HTTP de la page retournée par SPIP. Exemple : #HTTP_HEADER{Content-Type: text/css}. **Attention !** Le fait d'utiliser cette balise supprime les boutons d'administration. Cette balise ne peut pas être utilisée dans des squelettes inclus via la syntaxe <INCLURE>.

- La balise #EVAL{*argument*} (depuis SPIP 1.9) évalue l'expression PHP mise en accolade. Par exemple #EVAL{1+1} affichera 2, #EVAL{_DIR_IMG_PACK} affichera ainsi le chemin vers le répertoire ecrire/img_pack/. Attention, il est fortement conseillé de s'en servir avec modération.

- La balise #CACHE{*temps*} permet de déterminer le délai au bout duquel le squelette est réinterprété. Le temps est exprimé en secondes. Il peut se mettre sous forme de calcul. Par exemple : #CACHE{24*3600}.

Notes

[1] Cf. www.uzine.net/article1785.html pour une introduction aux charsets, en attendant une documentation plus complète de cette fonctionnalité de SPIP.

Les formulaires

SPIP permet une grande interaction du site avec les visiteurs ; pour cela, il propose de nombreux formulaires sur le site public, permettant tantôt de gérer les accès à l'espace privé, tantôt d'autoriser l'ajout de messages et signatures.

Les formulaires s'insèrent dans les squelettes par une simple balise ; SPIP se charge ensuite de gérer le comportement (souvent complexe) de ces formulaires en fonction de l'environnement et des configurations effectuées dans l'espace privé.

Fonctions interactives

- #FORMULAIRE_RECHERCHE

Il s'agit du formulaire du moteur de recherche intégré à SPIP. Il est présenté dans l'article sur les boucles de recherche.

- #FORMULAIRE_FORUM

Le #FORMULAIRE_FORUM gère l'interface permettant de poster des messages dans les forums publics. Il concerne donc en premier chef la boucle FORUMS mais peut être utilisé dans toutes les boucles acceptant un forum :

- La boucle ARTICLES,
- La boucle RUBRIQUES,
- La boucle BREVES,
- La boucle SITES (ou SYNDICATION).

Le formulaire dépend évidemment du choix des forums modérés a posteriori, a priori ou sur abonnement.

[SPIP 1.8.2] Par défaut, une fois le message posté, le visiteur est renvoyé vers la page de l'élément [1] auquel il a répondu. On peut décider de renvoyer le visiteur vers une autre page en passant une url en paramètre à cette balise. Par exemple :

- [(#FORMULAIRE_FORUM{'spip.php?page=merci'})] renverra vers la page *spip ?page=merci*.

  Historique : Pour les versions antérieures à [SPIP 1.9], il aurait fallu écrire merci.php3.

  Jusqu'à [SPIP 1.9] , les fichiers de spip avaient une extension en .php3 et non en .php.

- [(#FORMULAIRE_FORUM{#SELF})] renverra vers la page où le formulaire de forum est placé (voir la balise #SELF).

Dans le cas (très spécifique) où l'on a autorisé la présence de mots-clés dans les forums publics, on peut affiner le comportement de ce formulaire avec des variables de personnalisation.

- #FORMULAIRE_SIGNATURE

La balise #FORMULAIRE_SIGNATURE affiche un formulaire permettant aux visiteurs du site de signer les pétitions associées aux articles. Cette balise se place donc dans une boucle ARTICLES.

La signature des pétitions réclame obligatoirement une validation des signataires par email. Ce formulaire n'a donc d'intérêt que si votre hébergeur autorise l'envoi de mails par PHP.

- #FORMULAIRE_SITE

Introduite dans [SPIP 1.4], la balise #FORMULAIRE_SITE affiche un formulaire 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 sites référencés étant, dans SPIP, attachés aux rubriques, on ne peut placer ce #FORMULAIRE_SITE qu'à l'intérieur d'une boucle RUBRIQUES.

- #FORMULAIRE_ECRIRE_AUTEUR

[SPIP 1.4] Placée à l'intérieur d'une boucle AUTEURS, cette balise affiche le formulaire qui permet d'envoyer un mail à l'auteur. Cela permet d'écrire aux auteurs sans divulguer leur adresse email sur le site public.

[SPIP 1.8.2] Placé dans une boucle ARTICLES, ce formulaire permet d'envoyer un mail à tous les auteurs de cet article.

[SPIP 1.8.2] Placé dans une boucle FORUMS, ce formulaire permet d'envoyer un mail directement à l'auteur du message si l'auteur est enregistré sur le site.

Inscription, authentification...

- #FORMULAIRE_INSCRIPTION

Sans doute le plus importante, la balise #FORMULAIRE_INSCRIPTION affiche le formulaire permettant l'inscription de nouveaux rédacteurs. Celui-ci ne s'affiche que si vous avez autorisé l'inscription automatique depuis le site public (sinon, cette balise n'affiche rigoureusement rien).

L'inscription nécessite l'envoi des informations de connexion (login et mot de passe) par email ; ce formulaire ne fonctionne donc que si votre hébergeur autorise l'envoi de mails par PHP.

- [(#FORMULAIRE_INSCRIPTION{forum})]

Est l'équivalente de la précédente, pour l'inscription des visiteurs, appelés à écrire dans les forums (réservés aux visiteurs enregistrés), option qui se détermine dans la partie privée configuration/interactivité/Mode de fonctionnement par défaut des forums publics.

Après la validation, un message avertit le visiteur : "Votre nouvel identifiant vient de vous être envoyé par email."

- #LOGIN_PRIVE

_ SPIP 1.4 Tout aussi importante (sinon plus), cette balise affiche le formulaire d'accès à l'espace privé (la partie « /ecrire » du site).

Important : cette balise doit impérativement être présente dans le squelette appelé par la page spip.php?page=login, c'est-à-dire dans le squelette nommé login.html. En effet, lors des accès directs à l'adresse « /ecrire » de votre site, c'est vers spip.php?page=login que SPIP va vous rediriger.

> *Historique :* Pour les versions antérieures à SPIP 1.9, il s'agit du squelette appelé par la page spip_login.php3, c'est-à-dire en standard par le squelette nommé spip_login.html
>
> De façon générale jusqu'à SPIP 1.9, les urls des pages générées par SPIP étaient de la forme http://monsite.net/xxx.php3 et non pas http://monsite.net/spip.php?page=xxx.

- #LOGIN_PUBLIC

_ SPIP 1.4 D'une utilisation beaucoup plus spécifique, la balise #LOGIN_PUBLIC affiche un formulaire permettant à vos utilisateurs de s'identifier tout en restant sur le site public (sans entrer dans l'espace privé). Cette balise sert notamment à authentifier les visiteurs pour les sites proposant des forums *modérés sur abonnement*. Elle peut aussi servir de brique de base pour restreindre l'accès à certains contenus sur le site public : mais cela reste d'un maniement complexe, et nécessitera encore des développements et la rédaction

de tutoriels complets avant d'être facilement utilisable par tous ; néanmoins, un exemple d'utilisation avancée est donné plus bas.

Le #LOGIN_PUBLIC, par défaut, « boucle sur lui-même », c'est-à-dire que le formulaire revient sur la page où il se trouve. On peut cependant indiquer une page vers laquelle le formulaire mènera, sous la forme :

```
[(#LOGIN_PUBLIC|spip.php?page=mapage)]
```

> *Historique :* Dans les versions antérieures à SPIP 1.9 il aurait fallu écrire mapage.php3 et non spip.php?page=mapage
>
> De façon générale jusqu'à SPIP 1.9, les urls des pages générées par SPIP étaient de la forme http://monsite.net/xxx.php3 et non pas http://monsite.net/spip.php?page=xxx .

Si votre site offre une inscription automatique à l'espace privé, les données de connexion à l'espace public sont identiques à celles de l'espace privé ; c'est-à-dire que les données envoyées à l'utilisateur pour s'identifier à l'espace public lui permettent également d'accéder à l'espace privé. Si, au contraire, vous avez interdit l'inscription automatique à l'espace privé, *il faut impérativement avoir au moins un article dont les forums seront réglés en mode « sur abonnement »* pour activer cette balise ; dès lors, SPIP pourra fournir des informations de connexion pour le site public sans accès à l'espace privé.

- #URL_LOGOUT [SPIP 1.5] est le pendant de #LOGIN_PUBLIC ; il donne une URL permettant à un visiteur authentifié de se déconnecter.

[SPIP 1.8.2] On peut passer un paramètre à cette balise pour spécifier l'adresse de retour après la déconnection. Par exemple [(#URL_LOGOUT{spip.php?page=sommaire})] renverra vers la page de sommaire.

> *Historique :* Dans les versions antérieur à SPIP 1.9, il aurait fallu écrire [(#URL_LOGOUT{sommaire.php3})] et non pas [(#URL_LOGOUT{spip.php?page=sommaire})]
>
> De façon générale jusqu'à SPIP 1.9, les urls des pages générées par SPIP étaient de la forme http://monsite.net/xxx.php3 et non pas http://monsite.net/spip.php?page=xxx.

Voici un exemple simple, mais complet, d'utilisation de ces deux balises. Il faut passer par un peu de php pour tester la variable $auteur_session, qui indique qu'un auteur est identifié ou non. Si c'est le cas, on peut récupérer (voire tester) son statut, son login, etc., via $auteur_session['statut']....

Notez bien que le contenu n'est « sécurisé » que sur ce squelette. Si votre squelette « imprimer cet article », par exemple, ne vérifie par $auteur_session, tout le monde (y compris les moteurs de recherche !) pourra avoir accès à ce fameux contenu que vous souhaitez protéger.

```
<?php if ($auteur_session) { ?>

Vous êtes authentifié, <a href='#URL_LOGOUT'>cliquez ici pour vous déconnecter</a>

... ici le contenu en accès restreint....

<?php } else { ?>

<h2>Cette partie est en accès restreint</h2>
#LOGIN_PUBLIC

<?php } ?>
```

Styles

On peut sensiblement modifier l'interface graphique des formulaires par l'intermédiaire des feuilles de style. Voir : « Ils sont beaux, mes formulaires ! ».

Notes

[1] article, rubrique, brève, site ou forum

Les boucles de recherche

SPIP dispose d'un moteur de recherche intégré. Il faut donc prévoir une page permettant d'afficher les résultats des recherches.

Le formulaire de recherche

Pour afficher le formulaire de recherche, il suffit d'insérer la balise :

#FORMULAIRE_RECHERCHE

Le formulaire renvoit par défaut vers spip.php?page=recherche ; vous devez donc réaliser un squelette recherche.html permettant d'afficher les résultats.

Vous pouvez décider d'utiliser une autre page d'affichage des résultats. Pour cela, il faut utiliser la balise de la manière suivante :

[(#FORMULAIRE_RECHERCHE|spip.php?page=xxx)]

où spip.php?page=xxx est la page vers laquelle vous désirez envoyer l'utilisateur, et xxx.html est son squelette.

> *Historique :* Dans les versions antérieures à SPIP 1.9 il faut écrire xxx.php3 et non spip.php?page=xxx
>
> De façon générale jusqu'à SPIP 1.9, les urls des pages générées par spip étaient de la forme http://monsite.net/xxx.php3 et non pas http://monsite.net/spip.php?page=xxx.

Le squelette des résultats

Les boucles permettant d'afficher les résultats de la recherche sont, en réalité, des boucles déjà abordées ici : ARTICLES, RUBRIQUES, BREVES et depuis [SPIP 1.8.2] FORUMS. Vous pouvez en effet effectuer des recherches non seulement sur les articles, mais aussi sur les rubriques, les brèves et les forums.

La seule différence, par rapport à ce qui est documenté sur les pages de ces boucles, est le choix du critère de sélection, qui doit être {recherche}. Les autres critères d'affichage et les balises de ces boucles restent ici valables.

Cependant, afin de classer les résultats par pertinence, on utilisera de préférence ce nouveau critère d'affichage : {par points}.

Enfin, on pourra utiliser la balise #POINTS, qui affiche la pertinence des résultats (attention, dans l'absolu cette valeur n'est pas très explicite, elle est surtout utile pour le classement des résultats).

Introduite par [SPIP 1.5.1], la balise #RECHERCHE affiche la requête formulée par le visiteur.

Les filtres de SPIP

Nous avons vu dans la syntaxe des balises SPIP qu'il était possible de modifier le comportement et l'affichage des balises en leur attribuant des filtres.

[option avant **(#BALISE**|filtre1|filtre2|...|filtre*n***)** option après **]**

Les filtres 1, 2, ..., n sont appliqués successivement à la #BALISE.

Les filtres de mise en page

Les filtres de mise en page suivants (majuscules, justifier...) ne sont plus conseillés. Il est recommandé de leur préférer, désormais, l'utilisation des styles CSS correspondants.

- **majuscules** fait passer le texte en majuscules. Par rapport à la fonction de PHP, *majuscules* s'applique également aux lettres accentuées.
- **justifier** fait passer le texte en justification totale (<P align=justify>).
- **aligner_droite** fait passer le texte en justification à droite (<P align=right>).
- **aligner_gauche** fait passer le texte en justification à gauche (<P align=left>).
- **centrer** centre le texte (<P align=center>).

Les filtres des dates

Les filtres suivants s'appliquent aux dates ([(#DATE|affdate)] par exemple).

- **affdate** affiche la date sous forme de texte, par exemple « 13 janvier 2001 ».

[SPIP 1.8] étend la notation de ce filtre. On peut lui passer un paramètre de formatage de la date, correspondant à un format spip (« 'saison' », etc.) ou à un format de la commande php date (« 'Y-m-d' »). Par exemple :

- [(#DATE|affdate{'Y-m'})] affichera numériquement l'année et le mois de la date filtrée séparés par un tiret,
- la notation [(#DATE|affdate{'saison'})] est totalement équivalente à : [(#DATE|saison)].

- Il existe aussi des variantes de *affdate* qui fournissent des raccourcis :

affdate_jourcourt affiche le nom du mois et la valeur numérique du jour, e.g. « 19 Avril ». Si la date n'est pas dans l'année actuelle, alors l'année est aussi affichée : « 1 Novembre 2004 ».

affdate_court affiche le nom du mois et le numéros du jour, e.g. « 19 Avril ». Si la date n'est pas dans l'année actuelle, alors on affiche seulement le mois et l'année sans le numéros du jour : « Novembre 2004 ».

affdate_mois_annee affiche seulement le mois et l'année : « Avril 2005 », « Novembre 2003 ».

- **jour** affiche le jour (en nombre).
- **mois** affiche le mois (en nombre).
- **annee** affiche l'année.
- [SPIP 1.0.2] **heures** affiche les heures d'une date (les dates fournies par SPIP contiennent non seulement le jour, mais également les horaires).
- [SPIP 1.0.2] **minutes** affiche les minutes d'une date.
- [SPIP 1.0.2] **secondes** affiche les secondes.
- **nom_jour** affiche le nom du jour (lundi, mardi...).

- **nom_mois** affiche le nom du mois (janvier, février...).
- **saison** affiche la saison (hiver, été...).
- [SPIP 1.8] introduit le filtre **unique** qui retourne la valeur de l'élément filtré seulement si c'est la première fois qu'elle est rencontrée. Ce filtre n'est pas limité aux dates mais est intéressant pour, par exemple, afficher une liste d'articles par date :

```
<BOUCLE_blog(ARTICLES){par date}{inverse}{"<br>"}>
[<hr /><h1>(#DATE|affdate_mois_annee|unique)</h1>]
#TITRE ...
</BOUCLE_blog>
```

cette balise n'affichera la date qu'à chaque changement de mois.

Voici un autre exemple :

```
<BOUCLE_blog2(ARTICLES){par date}{inverse}>
    [<hr /><h1>(#DATE|annee|unique)</h1>]
        [<h2>(#DATE|affdate{'Y-m'}|unique|nom_mois)</h2>]
            <a href="#URL_ARTICLE">#TITRE</a><br />
</BOUCLE_blog2>
```

affichera une liste ressemblant à :

2005
mars
article de mars
autre article de mars
février
article de février
2004
décembre
un article

On utilise la notation affdate{'Y-m'} pour afficher le nom du mois à chaque année. En effet :

- si l'on ne faisait que #DATE|nom_mois|unique, les noms de mois ne seraient affichés que la première année.
- si le filtrage était : #DATE|unique|nom_mois, on afficherait toutes les dates. En effet, #DATE retourne une date complète qui contient aussi l'heure. Il y a donc une grande chance que les dates complètes de deux articles publiés le même jours soient différentes.

C'est pourquoi on garde juste le mois et l'année de la date avant de la passer au filtre *unique*.

On peut passer un argument optionnel à ce filtre pour différencier deux utilisations indépendantes du filtre. Par exemple : [(#DATE|affdate_mois_annee|unique{ici})] n'aura pas d'incidence sur [(#DATE|affdate_mois_annee|unique{la})].

Filtres de texte

La plupart de ces filtres ont été introduits dans la version [SPIP 1.4]

- **liens_ouvrants** transforme les liens SPIP qui donnent vers des sites extérieurs en liens de type « popup », qui ouvrent dans une nouvelle fenetre ; c'est l'équivalent du *target=blank* du HTML.

N.B. : les développeurs de SPIP estiment qu'il s'agit en général d'une impolitesse, car les internautes savent très bien s'ils ont envie ou pas d'ouvrir une nouvelle fenêtre - or ce système le leur impose. Mais la demande était trop forte, et nous avons craqué ;-)

- **supprimer_numero** sert à éliminer le numéro d'un titre, si par exemple on veut faire des tris

d'articles {par num titre} mais ne pas afficher les numéros (car ils ne servent qu'à ordonner les articles). Le format des préfixes numerotés est « XX. titre », XX étant un nombre à n chiffres (illimité).

- **PtoBR** transforme les sauts de paragraphe en simples passages a la ligne, ce qui permet de « resserrer » une mise en page, par exemple à l'intérieur d'un sommaire

- **taille_en_octets** permet de transformer un nombre d'octets (25678906) en une chaîne de caractères plus explicite (« 24.4 Mo »).

- **supprimer_tags** est une suppression basique et brutale de tous les <...>

- **textebrut** s'apparente au filtre supprimer_tags, mais il agit de manière un peu plus subtile, transformant notamment les paragraphes et
 en sauts de ligne, et les espaces insécables en espaces simples. Il s'utilise, par exemple, pour faire un descriptif META : [<meta name="description" content="(#DESCRIPTIF|textebrut)">]

- **texte_backend** peut être utilisé pour transformer un texte et le rendre compatible avec des flux XML. Ce filtre est utilisé, par exemple, dans le squelette *backend.html* qui génére le fil RSS du site.

- **couper** coupe un texte après un certain nombre de caractères. Il essaie de ne pas couper les mots et enlève le formatage du texte. Si le texte est trop long, alors « (...) » est ajouté à la fin. Ce filtre coupe par défaut à 50 caractères, mais on peut spécifier une autre longueur en passant un paramètre au filtre, par exemple : [(#TEXTE|couper{80})].

- **lignes_longues**, introduit par SPIP 1.9, coupe les mots « trops longs » (utile si l'on a, par exemple, des urls à afficher dans une colone étroite). Ce filtre coupe par défaut à 70 caractères, mais on peut spécifier une autre longueur en passant un paramètre au filtre, par exemple : [(#URL_SITE|lignes_longues{40})].

- **match** utilise une expression régulière (cf. *preg_match()*) pour extraire un motif dans le texte si il est présent, ne retourne rien sinon. Par exemple pour récupérer le premier mot du titre [(#TITRE|match{^\w+})]. Ce peut être un texte simple, afficher "toto" si dans le titre : [(#TITRE|match{toto})]

- **replace** utilise aussi une expression régulière (cf. *preg_replace()*) pour supprimer ou remplacer toutes les occurences d'un motif dans le texte. Avec un seul parametre, une expression régulière, le motif sera remplacé par une chaîne, c'est à dire supprimé. Par exemple pour supprimer tous les "notaXX" du texte [(#TEXTE|replace{nota\d*})]. Lorsqu'un deuxiéme paramètre est fourni, les occurences du motif seront remplacées par cette valeur. Par exemple pour remplacer tous les 2005 ou 2006 du texte en 2007 [(#TEXTE|replace{200[56],2007})]. Ce peut être des textes simples, remplacer tous les "au temps" par "autant" : [(#TEXTE|replace{au temps,autant})]

Filtres de test

- [SPIP 1.6] introduit le filtre |sinon, qui indique ce qu'il faut afficher si l'élément « filtré » est vide : ainsi [(#TEXTE|sinon{"pas de texte"})] affiche le texte ; si celui-ci est vide, affiche *« pas de texte »*.

- [SPIP 1.8] introduit le filtre **|?{***sioui***,***sinon***}** qui est une version évoluée de |sinon. Il prend un ou deux paramètres :

  - *sioui* est la valeur à afficher à la place de l'élément filtré si celui-ci est non vide.
  - *sinon* est optionnel. C'est la valeur à afficher si l'élément filtré est vide. [(#TEXTE|?{#TEXTE,"pas de texte"})] est équivalent à l'exemple donné pour le filtre |sinon.

- [SPIP 1.8] introduit un jeu de filtres pour faire des comparaisons avec des valeurs :

  - **|=={***valeur***}** et **|!={***valeur***}** permettent de vérifier, respectivement, l'égalité ou l'inégalité entre l'élément filtré et *valeur*. Par exemple : <li [(#TITRE|=={édito}|?{'id="edito"',"})]>#TITRE
  - **|>{***valeur***}**, **|>={***valeur***}**, **|<{***valeur***}** et **|<={***valeur***}** comparent l'élément filtré (qui doit être numérique) avec une valeur numérique.
    Par exemple :

```
[(#TOTAL_BOUCLE) [(#TOTAL_BOUCLE|>{1}|?{'articles','article'})] dans cette rubrique.]
```

Remarque : De manière générale, tous les opérateurs de comparaison de php peuvent être utilisés comme filtres dans [SPIP 1.8].

Filtres de logos

- **fichier** [SPIP 1.4]. Affecté à un logo, ce filtre permet de récupérer directement le nom de fichier correspondant au logo.

- ||**autres filtres** Contrairement aux versions précédentes, [SPIP 1.4] permet de passer des filtres « maison » sur les logos : la logique est un peu tordue, car il fallait respecter la compatibilité avec SPIP 1.3. L'analyse se déroule comme suit :

  - si le premier « filtre » n'est pas un alignement, SPIP considère qu'il s'agit d'un URL et fait un lien du logo vers cette adresse ;
  - si le premier filtre est un alignement, SPIP considère que le deuxième « filtre » est un URL ;
  - les filtres suivants sont de vrais filtres au sens habituel (y compris des filtres « maison » déclarés dans mes_fonctions.php ;
  - pour appliquer un filtre quelconque sans mettre d'URL, il faut mettre deux barres. Par exemple : <?php $logo = '[(#LOGO_RUBRIQUE||texte_script)]'; ?> permet de récupérer le logo dans la variable php $logo, pour traitement ultérieur (voir ci-dessous pour la signification de |texte_script).

- [SPIP 1.8] introduit les filtres hauteur et largeur qui retournent les informations sur la taille (i.e. Hauteur et Largeur) de l'élément filtré si c'est une image.

Ces filtres n'ont qu'un intérêt moyen à être appliqués directement à un logo de document puisqu'il y a déjà #HAUTEUR et #LARGEUR à disposition pour les documents. Par contre, on peut les appliquer après le filtre image_reduire pour connaître la taille exacte de l'image réduite.

Plus généralement, on peut les appliquer sur n'importe quelles balises (ou filtre) retournant un balise HTML <img ...>.

- |image_reduire{largeur,hauteur} permet de forcer une taille maximale d'affichage des images et des logos.

Ce filtre s'utilise par exemple sur un logo d'article de la façon suivante :

```
[(#LOGO_ARTICLE|right||image_reduire{130})]
```

Dans cet exemple, logo de l'article apparaît aligné à droite, à une taille maximale de 130 pixels.

Depuis [SPIP 1.8.2], ce filtre peut prendre deux arguments : *largeur* et *hauteur*. Si l'un de ces deux arguments est égal à 0, SPIP ne tient compte que de l'autre et calcule cette dimension en conservant les proportions de l'image. De plus, ce filtre s'applique aussi à la balise #TEXTE et ce sont alors toutes les images que le rédacteur introduit dans le texte grâce aux raccourcis SPIP qui sont alors réduites.

Ainsi, par exemple,

```
[(#TEXTE|image_reduire{600,0})]
```

affiche toutes les images insérées dans le fil du texte à une largeur maximale de 600 pixels. Cela permet de préserver la mise en page même sans que le rédacteur ait à se soucier de la taille des images qu'il télécharge sur le site.

NB. Si l'option « création de vignettes » est activée dans la configuration du site, ces logos réduits seront

des fichiers d'images spécifiques calculés automatiquement par le serveur (idéalement, avec l'extension GD2 installée sur le serveur), pour les formats acceptés par le serveur (avec GD2, habituellement, les formats JPG et PNG). Sinon, c'est une version complète de l'image qui est affichée, mais avec une taille d'affichage fixée directement en HTML.

Historique : SPIP 1.7.1 introduit le filtre |reduire_image. Avec SPIP 1.9, celui-ci devient |image_reduire (les filtres de traitement d'image commencent ainsi tous par |image_).

Les filtres mathématiques

SPIP 1.9 introduit une série de filtres d'opérations mathématiques.

- |plus{xx}, |moins{xx} et |mult{xx} correspondent respectivement à l'addition, la soustraction et la multiplication.
- |div{xx} correspond à la division *non-euclidienne* ("après la virgule").
- |modulo{xx} correspond au reste de la division euclidienne par xx d'un nombre.

Par exemple [(#COMPTEUR_BOUCLE|modulo{5})] compte de 0 à 4 puis revient à 0 etc

De plus l'ensemble des fonctions mathématiques PHP peuvent être utilisées comme filtres.

Autres Filtres

- **traduire_nom_langue** s'applique à la balise #LANG et retourne un traduction du code de langue qu'elle retourne (fr, en, it, etc.) dans cette langue.

Remarque : Les traductions des codes sont faites dans la langue que représente ce code et suivent les conventions d'écriture de cette langue.

Ainsi « fr » sera traduit « français » en minuscule, alors que « es » sera traduit « Español » avec une majuscule.

- **alterner{*a,b,c,...*}** [SPIP 1.8.2] s'applique à une balise numérique (en général #COMPTEUR_BOUCLE ou #TOTAL_BOUCLE) et affiche l'argument correspondant à la valeur de cette balise . On peut ainsi alterner un affichage dans une boucle. Par exemple, [(#COMPTEUR_BOUCLE|alterner{'white','grey'})] affichera « white » à la première itération de la boucle, « grey » à la deuxième, « white » à la troisième, « grey » à la quatrième, etc. Ainsi, on peut faire une liste d'article qui utilise une couleur différente pour les lignes paires et impaires :

```
<B_lesarticles>
   <ul>
<BOUCLE_lesarticles(ARTICLES) {par titre}>
   <li style="background: [(#COMPTEUR_BOUCLE|alterner{'white','grey'})]">#TITRE</li>
</BOUCLE_lesarticles>
   </ul>
</B_lesarticles>
```

- **inserer_attribut{*attribut,valeur*}** [SPIP 1.8.2] permet d'ajouter un attribut html dans une balise html générée par SPIP. Par exemple : [(#LOGO_DOCUMENT| inserer_attribut{'alt',#TITRE})] va ajouter un attribut « alt » avec le titre du document dans la balise « img » du logo.
- **extraire_attribut{*attribut*}** [SPIP 1.8.2] est l'inverse du filtre précédent. Il permet de récupérer un attribut d'une balise html générée par SPIP. Par exemple, on peut trouver le chemin de la vignette générée par le filtre *image_reduire* : <div style="background: url([(#LOGO_ARTICLE|| image_reduire{90}|extraire_attribut{src})]) left;"]>#TEXTE
- **parametre_url{*parametre,valeur*}** [SPIP 1.8.2] est un filtre qui permet d'ajouter des paramètres dans une url générée par une balise SPIP. Si *valeur* vaut " le filtre supprime un paramètre actuellement dans l'url. Par exemple, la balise #SELF retourne l'url de la page actuelle, donc :

- [(#SELF|parametre_url{'id_article','12'})] placera une variable id_article égale à 12 dans l'url,
- [(#SELF|parametre_url{'id_article',''})] effacera l'id_article actuellement dans l'url.

On peut par exemple l'utiliser pour faire des boutons pour naviguer parmi les documents sur une page :

```
<BOUCLE_actuel(DOCUMENTS) {id_document}>
  #LOGO_DOCUMENT
  <ul>
  <BOUCLE_precede(DOCUMENTS) {par date} {age_relatif <= 0} {0,1} {exclus}>
  <li>
    <a href="[(#SELF|parametre_url{'id_document',#ID_DOCUMENT})]" title="précédent">
      [(#LOGO_DOCUMENT|image_reduire{70})]
    </a>
  </li>
  </BOUCLE_precede>
  <BOUCLE_suivant(DOCUMENTS) {par date} {age_relatif > 0} {0,1}>
  <li>
    <a href="[(#SELF|parametre_url{'id_document',#ID_DOCUMENT})]" title="suivant">
      [(#LOGO_DOCUMENT|image_reduire{70})]
    </a>
  </li>
  </BOUCLE_suivant>
  </ul>
</BOUCLE_actuel>
```

Filtres techniques

Ces filtres ont été introduits par [SPIP 1.4].

- **entites_html** transforme un texte en entités HTML, que l'on peut donc implanter dans un formulaire, exemple : [<textarea>(#DESCRIPTIF|entites_html)</textarea>]

- **texte_script** transforme n'importe quel champ en une chaîne utilisable en PHP ou Javascript en toute securité, exemple : <?php $x = '[(#TEXTE|texte_script)]'; ?>. Attention : utilisez bien le caractère ' et non " : en effet, dans le second cas, si votre texte contient le symbole $, le résultat peut être catastrophique (affichage partiel, affichage d'autre chose, plantage php, etc.).

- **attribut_html** rend une chaîne utilisable sans dommage comme attribut HTML ; par exemple, si l'on veut ajouter un texte de survol au lien normal vers un article, on utilisera

#TITRE.

- **liens_absolus** [SPIP 1.8.2] s'applique sur une balise de texte et transforme tous les liens que celui-ci contient en liens absolus (avec l'url complète du site). Ce filtre est particulièrement utile dans des squelettes de fil rss par exemple.

- **url_absolue** [SPIP 1.8.2] marche de la même façon que le filtre précédent, mais s'applique à une balise qui retourne une url (par exemple #URL_ARTICLE).

- **abs_url** [SPIP 1.8.2] combine les deux balises précédentes et peut donc s'appliquer à un texte ou à une balise d'url.

- **form_hidden** SPIP 1.9 Si on fait un formulaire qui utilise comme action un lien comprenant des arguments (par exemple, quand on utilise la balise #SELF avec le type d'url par defaut), il faut remettre ces valeurs dans des champs *hidden* ; cette fonction calcule les champs en question. A utiliser par exemple :

```
<form action="#SELF">
[(#SELF|form_hidden)]
...
</form>
```

- Le filtre |**compacte** permet de réduire la taille d'un css ou d'un javascript en supprimant tout les

commentaires. Le filtre prend en entrée le nom du fichier, et produit un nouveau fichier dont il renvoie le nom <link rel="stylesheet" href="[(#CHEMIN{spip_style.css}|compacte)]" type="text/css" media="all" />. (Filtre ajouté dans SPIP 1.9.2)

- Un nombre d'autres filtres techniques sont référencés ici.

Ajouter ses propres fonctions

Les filtres de SPIP sont des fonctions PHP qui reçoivent la balise sur laquelle ils sont appliqués en premier paramètre et retournent le texte à afficher. Vous pouvez utiliser directement les fonctions habituelles de PHP, mais également créer les vôtres, sur le modèle :

```
<?php
function mon_filtre($texte){
  $texte = (bidouillages en PHP) ...;
  return $texte;
}
?>
```

Afin de ne pas avoir à modifier des fichiers de SPIP (qui risqueraient d'être écrasés lors d'une prochaine mise à jour), vous pouvez installer vos fonctions personnelles dans un fichier mes_fonctions.php : si SPIP repère un fichier ayant ce nom, il l'inclut automatiquement.

Historique : Dans les versions antérieures à [SPIP 1.9] ce fichier doit s'appeler mes_fonctions.php3

De façon générale jusqu'à [SPIP 1.9] , les fichiers de spip avaient une extension en .php3 et non en .php.

Depuis [SPIP 1.8] il peut se situer :

- dans le dossier où sont stockés vos squelettes
- à la racine du site Mais **jamais** dans les deux à la fois.

Par exemple, ARNO* a développé le filtre enlettres, qui n'est pas inclus dans la distribution standard de SPIP. Ce filtre écrit un nombre en toutes lettres ([(#DATE|annee|enlettres)] = « deux mille deux ») ; ce filtre peut être téléchargé sur http://www.uzine.net/spip_contrib/a... ; il suffit de l'ajouter dans votre fichier mes_fonctions.php pour l'utiliser.

Filtres avec des paramètres

Depuis [SPIP 1.5], il est possible de passer des paramètres dans les filtres. La syntaxe est :

```
[(#BALISE|filtre{arg1, arg2}|...)]
```

Le filtre doit être défini de la manière suivante dans mes_fonctions.php :

```
function filtre($texte, $arg1='valeur par defaut1', $arg2='valeur par défaut 2')
{
    ....calculs....
    return (une chaine de caractères);
}
```

On peut ainsi appeler n'importe quelle fonction php, ou s'appuyer sur des fonctions définies dans SPIP ou dans mes_fonctions.php, pour peu qu'elles respectent l'ordre des arguments (le texte à traiter doit être impérativement le premier argument). Par exemple, pour enlever les points à la fin d'un texte, on pourra faire : [(#TEXTE|rtrim{'.?!'})].

Depuis [SPIP 1.8], les arguments des filtres peuvent être des balises (sans codes optionnels ni filtres). Par exemple :
[(#TOTAL_BOUCLE|=={#COMPTEUR_BOUCLE}|?{'Fin.',''})]
Depuis [SPIP 1.8.2] on peut mettre un balise avec notation étendue en paramètre. Par exemple :
[(#DESCRIPTIF|sinon{[(#CHAPO|sinon{#TEXTE}|couper{300})]})]

Les boucles récursives

Les boucles récursives offrent une fonctionnalité très puissante de mise en page de structure hiérarchique. Leur écriture est concise, mais leur utilisation demande une bonne maîtrise logique de l'enchaînement des boucles.

Pour construire une boucle récursive, il suffit d'indiquer dans son TYPE le nom d'une boucle contenant celle qu'on écrit :

<BOUCLEx ....>
....
<BOUCLEn(BOUCLEx)></BOUCLEn>
....
</BOUCLEx>

La boucle *n* fonctionne comme si l'on avait recopié l'intégralité de la boucle *x* (toutes les balises et le code HTML, ainsi que les textes conditionnels avant, après et alternatif) à l'endroit où l'on insère la boucle *n*. La boucle *n* étant à l'intérieur de la boucle *x*, on obtient un comportement récursif : la boucle *x* contient une boucle *n*, qui elle-même reproduit la boucle *x* qui contient la boucle *n*, et ainsi de suite, jusqu'à ce que la boucle *x* ne donne plus aucun résultat. Aucun critère ne figure dans la boucle *n*, le changement de contexte à chaque appel de la boucle *x* devant conduire les critères de celle-ci à ne plus trouver aucun élément.

Cette technique permet de créer notamment l'affichage des threads des forums. Une première boucle « fabrique » l'entrée des threads (les messages qui répondent directement à un article), une seconde boucle affiche les réponses à ces messages, et une boucle récursive provoque la récursivité sur cette seconde boucle :

```
<BOUCLE_forum(FORUMS){id_article}>
   #TITRE
   <B_reponses>
   <UL>
   <BOUCLE_reponses(FORUMS){id_parent}>
      <LI>#TITRE
      <BOUCLE_recursive(BOUCLE_reponses)>
      </BOUCLE_recursive>
      </LI>
   </BOUCLE_reponses>
   </UL>
   </B_reponses>
</BOUCLE_forum>
```

Plus généralement, cette fonctionnalité provoque un affichage graphiquement très clair de structures arborescentes, en particulier la hiérarchie des rubriques de votre site.

Remarque 1 : Le nouveau compilateur introduit par SPIP 1.8 considère que la notation <BOUCLEn(BOUCLEx)> à l'extérieur de la boucle *x* est vide de sens, ce qui lui permet d'atteindre l'optimalité du nombre de champs des requêtes SQL qu'il produit, pour tout type de boucles.
Cet emploi n'est pas une récursion mais une inclusion, fonctionnalité à laquelle répond la balise INCLURE, nettement préférable.

Remarque 2 : Dans l'état actuel (SPIP 1.9) du compilateur de SPIP, la séquence <BOUCLEn(BOUCLEx)></BOUCLEn> doit figurer au premier niveau de la boucle *x*, autrement dit la boucle *n* doit être immédiatement englobée par la boucle *x*, non par une autre boucle elle-même à l'intérieur de la boucle *x*.
La levée de cette restriction est à l'étude.

La « popularité » des articles

La notion de *popularité*, exposée ci-dessous, apparaît dans SPIP 1.4.

Comment décompter des visites

Des centaines de méthodes statistiques existent pour décompter des visites sur un site donné. La plupart donnent des courbes horaires, ou par jour, qui permettent de savoir si son site « monte » ou « descend », et de vérifier qu'il y a plus de gens sur le net en fin d'après-midi et dans la semaine, que le week-end ou la nuit...

Notre objectif est un peu différent : il s'agit d'attribuer à chaque article une valeur de « popularité » reflétant assez rapidement une tendance, et permettant de comparer l'activité de différents articles, soit de manière globale sur tout le site (hit-parade), soit à l'intérieur d'une rubrique, soit parmi les articles d'un même auteur, etc.

La méthode retenue est la suivante (rassurez-vous, vous pouvez sauter cette explication si vous n'êtes pas à l'aise en maths) :

- chaque visite sur un article ajoute un certain nombre de points à cet article ; 1 point si c'est un article que l'on consulte depuis le site lui-même en suivant un lien, et 2 points si c'est une « entrée directe » depuis un site extérieur (moteur de recherche, lien hypertexte, syndication...)
- toutes les 10 minutes, le score obtenu est multiplié par un petit facteur d'escompte, qui fait qu'un point attribué par une visite à 10h12 le mercredi ne vaut plus, le lendemain à la même heure, qu'un demi-point, et, le vendredi à 10h12, un quart de point... ;
- le tout est calculé de manière à ce que, dans l'hypothèse où l'article reçoit toujours le même nombre *x* de visites par unité de temps, son score se stabilise sur cette valeur *x*. Autrement dit, si la fréquentation de l'article est stationnaire, sa popularité finira par refléter exactement son nombre de visites par jour (modulo le score 2 donné pour les entrées directes) ;
- cette popularité s'exprime de deux manières : l'une, la popularité_absolue, exprime le score en question (évaluation de la fréquentation quotidienne de l'article) ; l'autre, la popularité_relative, un pourcentage relatif à l'article du site ayant la plus forte popularité (popularité_max) ;
- enfin, la somme de toutes ces valeurs (absolues) sur le site donne la popularité_site, qui permet de comparer la fréquentation de deux sites sous spip...

Boucles et balises

Des balises permettent de récupérer et d'afficher ces valeurs dans vos squelettes. La boucle ci-dessous résume l'ensemble de ces balises :

```
<BOUCLE_pop(ARTICLES){id_article}{popularite>0}>
<h5>Popularité</h5>
Cet article a une popularité absolue égale à #POPULARITE_ABSOLUE, soit
#POPULARITE % de #POPULARITE_MAX. Au total, ce site fait environ
#POPULARITE_SITE visites par jour.
</BOUCLE_pop>
```

La balise la plus utile est #POPULARITE puisqu'elle donne un pourcentage représentant la popularité de l'article relativement à l'article le plus populaire du site. Cela permet ainsi de réaliser facilement des classements compréhensibles par tous (avec des valeurs allant de 0 à 100). Les autres balises donnent des valeurs absolues, plus difficiles à interpréter par les visiteurs du site.

> *Note :* bien que les données soient représentées, dans la base de spip, sous forme de nombres réels, le rendu de toutes ces balises est toujours donné sous la forme d'un nombre entier, ce qui

donnera, sur des sites *très* peu fréquentés (sites de tests, notamment), des choses amusantes du genre :

« Cet article a une popularité absolue égale à 1, soit 17 % de 2. Au total, ce site fait environ 5 visites par jour. »

Enfin, un critère de tri peut se révéler utile : {par popularite}, que l'on utilisera par exemple de la manière suivante pour afficher la liste des 10 articles les plus populaires de la rubrique courante :

```
<BOUCLE_hitparade(ARTICLES){id_rubrique}{par popularite}{inverse}{0,10}>
<li>#TITRE (popularité : #POPULARITE %)</li>
</BOUCLE_hitparade>
```

(On enlèvera {id_rubrique} pour afficher un hit-parade du site entier.)

La gestion des dates

SPIP 1.6 introduit une série de critères et de balises pour mieux gérer les dates des articles. En voici une liste.

Afficher les dates

- #DATE est la date de mise en ligne. (Modifiable après la mise en ligne de l'article, de la brève, etc. La date d'une rubrique est celle de son élément le plus récent.)
- #DATE_REDAC est la date de première publication. (Modifiable à volonté, disponible sur les articles seulement.)
- #DATE_MODIF Apparue avec [SPIP 1.5], cette balise désigne la date de dernière modification de l'article.
- #DATE_NOUVEAUTES [SPIP 1.6] permet d'afficher la date du dernier envoi du mail présentant les nouveautés.

Formater les dates

Si les balises #DATE... sont utilisées sans filtre, alors toutes les informations de date sont affichées dans un format numérique (au format MySQL) : « 2001-12-01 03:25:02 ».

Les filtres |annee, |mois, |jour, |heures, |minutes, |secondes, mais aussi |affdate, |nom_mois, |nom_jour, |saison, etc. s'appliquent pour permettre tous les affichages habituels sous divers formats. Une liste complète des filtres pouvant être appliqués aux dates est fournie dans l'article Les filtres de SPIP.

Contexte de date

[SPIP 1.6] fournit à toutes les boucles un contexte de date. Si l'on se trouve à l'intérieur d'une boucle (ARTICLES), (BREVES) ou (RUBRIQUES), la date en question est la date de publication de l'article, de la brève ou la date de dernière modification de la rubrique.

Si en revanche on se trouve au premier niveau du squelette (c'est-à-dire en-dehors de toute boucle), la date considérée est la date du jour - à moins qu'on ait passé une date dans l'URL de la page (voir l'exemple plus bas).

Dans ce dernier cas, et pour les versions de php supérieures à 3.0.12, la date passée dans l'URL est analysée avec la fonction strtotime : ainsi ?date=2003, ?date=2003/01 fonctionneront, mais aussi date=-1year (il y a un an), ?date=1march1970 (articles publiés le 1er mars 1970), etc.

Critère de date, d'âge, et d'âge relatif

Le critère {age} permet de sélectionner les articles en fonction de la durée qui sépare leur date de publication en ligne avec la date courante. Ainsi {age<30} permettra de ne pas afficher les articles âgés de plus de 30 jours.

L'{age_relatif} permet de comparer les dates de publication de deux articles : si l'on vient de sélectionner un article dans une boucle, une seconde boucle placée à l'intérieur de la première pourra demander les articles publiés dans la semaine qui précède celui-ci, via {age_relatif<=7}{age_relatif>=0}, etc.

Les critères {age} et {age_relatif} permettent de distinguer deux articles publiés le même jour (ce n'était pas le cas avant [SPIP 1.6]). On peut donc désormais programmer des boucles pour obtenir l'article « précédent » ou le « suivant » :

```
<BOUCLE_art(ARTICLES){id_article}>
<BOUCLE_precedent(ARTICLES){age_relatif>=0}{par date}{inverse}{1,1}>
précédent : <a href='#URL_ARTICLE'>#TITRE</a> #DATE
</BOUCLE_precedent>
<br />
<b>#TITRE</b> - #DATE
<br />
<BOUCLE_suivant(ARTICLES){age_relatif<0}{par date}{0,1}>
suivant : <a href='#URL_ARTICLE'>#TITRE</a> #DATE
</BOUCLE_suivant>
</BOUCLE_art>
```

Attention ! Malgré les apparences les comparaisons de date sont d'un maniement délicat : en effet, à cause des « dates floues » (un article publié un mois donné, sans que le jour soit précisé), le calcul de l'age_relatif peut donner la valeur zéro dans un sens, et pas dans l'autre ! D'où la dissymétrie des boucles présentées ci-dessus : dans un sens on cherche le « second plus récent » des articles {age_relatif>=0} (car le plus récent, avec la comparaison non-stricte, ne peut être que l'article lui-même) ; dans l'autre le plus âgé des articles publiés strictement plus tard.

Les critères {jour_relatif}, {mois_relatif} et {annee_relatif} fonctionnent comme l'*age_relatif*, mais prennent en compte des dates arrondies au jour, au mois et à l'année respectivement ; par exemple, si l'URL comporte la variable ?date=2003-01-01, la boucle suivante donnera « tous les les articles du mois de mars 2003 »

```
<h3>Articles de [(#DATE|nom_mois)] [(#DATE|annee)] :</h3>
<BOUCLE_blog(ARTICLES){mois_relatif=0}{par date}{"<br />"}>
<a href='#URL_ARTICLE'>#TITRE</a> ([(#DATE|jour)]/[(#DATE|nom_mois)]
</BOUCLE_blog>
```

La date de rédaction antérieure

Si vous avez activé l'utilisation des dates de publication antérieure, la plupart des critères présentés ci-dessus fonctionnent : il suffit d'ajouter _redac au critère. Ainsi {age_redac>365} affichera les articles dont la date de publication antérieure remonte à plus d'un an.

Si une boucle sélectionne un article dont la date_redac est définie, une boucle interne comportant le critère {annee_relatif_redac=0} ira chercher les articles dont la date de publication antérieure appartient à la même année.

Un exemple de sommaire de site trié par date

A titre d'exemple, voici comment on peut afficher tous les articles d'un site, triés par mois de publication :

```
<BOUCLE_articlem(ARTICLES){par date}{inverse}>
<BOUCLE_premierdumois(ARTICLES){id_article}{doublons}>
<br /><ul><b> [(#DATE|nom_mois|majuscules)] [(#DATE|annee)] </b>
   <li><a href="#URL_ARTICLE">[(#TITRE|couper{50})]</a> -
 [(#DATE|jour)]/[(#DATE|mois)]</li>
</BOUCLE_premierdumois>
   <BOUCLE_MOIS(ARTICLES) {mois_relatif=0}{doublons}{par date}{inverse} >
       <li><a href="#URL_ARTICLE">[(#TITRE|couper{50})]</a>
 - [(#DATE|jour)]/[(#DATE|mois)]</li>
   </BOUCLE_MOIS>
</ul>
</BOUCLE_articlem>
```

Exposer un article dans une liste

La balise #EXPOSE permet de mettre en évidence, dans un menu ou dans une liste, l'objet principal de la page où l'on se trouve.

L'objet qui est « exposé » est l'article (ou la brève, la rubrique, le mot-clé ou l'auteur) qui appartient au « contexte » courant. Dans le cas des rubriques, la traversée de la hiérarchie est gérée, ce qui permet d'« exposer » l'arborescence des rubriques qui contient l'article affiché.

Par défaut, SPIP remplace la balise #EXPOSE par « **on** » si l'objet correspond au contexte ; sinon la balise est simplement ignorée.

Toutefois la balise #EXPOSE accepte un ou deux arguments, qui permettent de préciser ce qui doit s'afficher pour l'objet exposé, et ce qui doit s'afficher pour les autres objets. Ainsi [(#EXPOSE{oui,non})] affichera « oui » pour l'objet exposé, et « non » pour les autres.

Afficher différemment l'article exposé

Utilisée simplement, la balise #EXPOSE permet, par exemple, dans un menu de navigation, de changer l'apparence du lien vers l'article que l'on est précisemment en train de consulter. Pour modifier le style des liens de ce menu, on placera la balise #EXPOSE de la manière suivante dans le squelette article.html :

```
<BOUCLE_principale (ARTICLES) {id_article}>

	<B_menu>
	<ul>
		<BOUCLE_menu (ARTICLES) {id_rubrique}>
		<li>
			<a href="#URL_ARTICLE"[ class="(#EXPOSE)"]>
			#TITRE
			<a>
		</li>
		</BOUCLE_menu>
	</ul>
	</B_menu>

	#TEXTE

</BOUCLE_principale>
```

avec les styles suivants :

```
a { color: blue; }
a.on { color: red; font-weight: bold; }
```

L'article ainsi exposé se distinguera dans la liste par un affichage visuellement différent.

Désactiver le lien exposé

Avec un peu d'astuce il est possible de désactiver le lien vers l'article exposé et dans le même temps de choisir le style à appliquer :

```
<B_menu>
<ul>
    <BOUCLE_menu(ARTICLES){id_rubrique}>
    <li>
        <#EXPOSE{span,a href="#URL_ARTICLE"}[ class="(#EXPOSE)"]>
        #TITRE
        </#EXPOSE{span,a}>
    </li>
    </BOUCLE_menu>
</ul>
</B_menu>
```

créera le code HTML suivant, où les balises <a> sont remplacées par des :

```
<ul>
<li><a href="article1.html">Tout sur ma soeur</a></li>
<li><span class="on">Tout sur moi</span></li>
<li><a href="article3.html">Tout sur mon frère</a></li>
</ul>
```

ce qui s'affiche ainsi :

- Tout sur ma soeur
- Tout sur moi
- Tout sur mon frère

P.-S.

Historique :

Cette fonctionnalité a été introduite par SPIP 1.7.1 avec la balise #EXPOSER dont la syntaxe complète était la suivante : [(#EXPOSER|oui,non)]

Depuis SPIP 1.8.2, celle-ci est désormais obsolète. Il est donc conseillé d'utiliser #EXPOSE, dont la syntaxe complète, ci-dessus présentée, est plus conforme au modèle général des balises de SPIP.