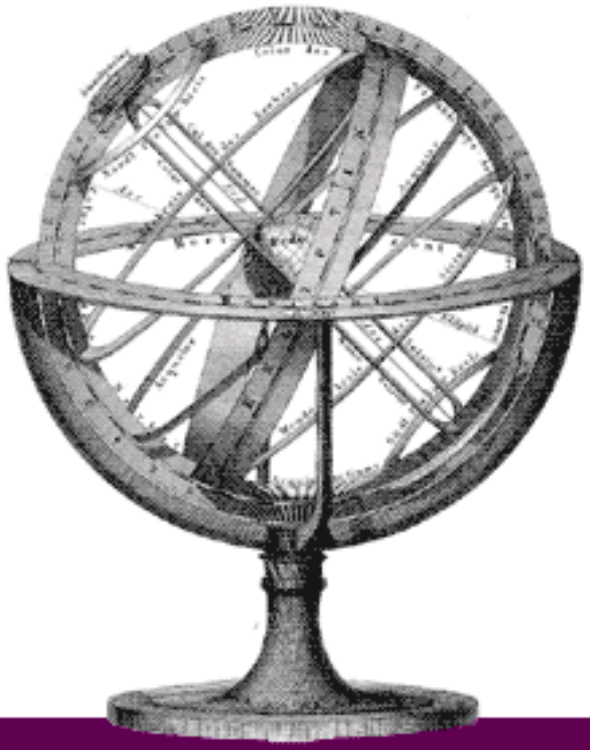
SPIP

Système de Publication pour l'Internet



Le manuel

de référence

uZine 3

Avant-propos

SPIP[1] est le système de publication développé par le minirézo pour la gestion du site uZine[2]. Nous le livrons à chacun, sous licence libre (GPL). Vous pouvez donc l'utiliser *librement* pour votre propre site, qu'il soit personnel, associatif ou marchand.

VERSION 20021214
Compilation du document
à l'aide de PDFLaTeX

Philippe Charlier

[1] Version actuelle : SPIP 1.4.2
[2] http ://www.uzine.net

Table des matières

1 Principe général

2 juin 2001 par l'équipe de SPIP

Tout le contenu d'un site géré sous SPIP est installé dans une base de données mySQL (dans la partie protégée par mot de passe `/ecrire`). Pour présenter ces informations aux visiteurs du site, il faut donc réaliser l'opération qui consiste à récupérer les informations, à les organiser et à les mettre en page, afin de délivrer une page HTML.

Cette opération est traditionnellement assez pénible :

- il faut connaître PHP et mySQL, et effectuer des routines relativement complexes ;
- l'intégration de telles routines dans une mise en page HTML élaborée est assez pénible ;
- le recours systématique à des requêtes mySQL à chaque affichage d'une page est gourmand en ressources, ralentit la visite et, dans des cas extrêmes, provoque des plantages du serveur.

SPIP propose ici 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.

1.1 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, toutes les articles bénéficieront de la même interface, simplement le système placera des informations différentes dans ce graphisme (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.

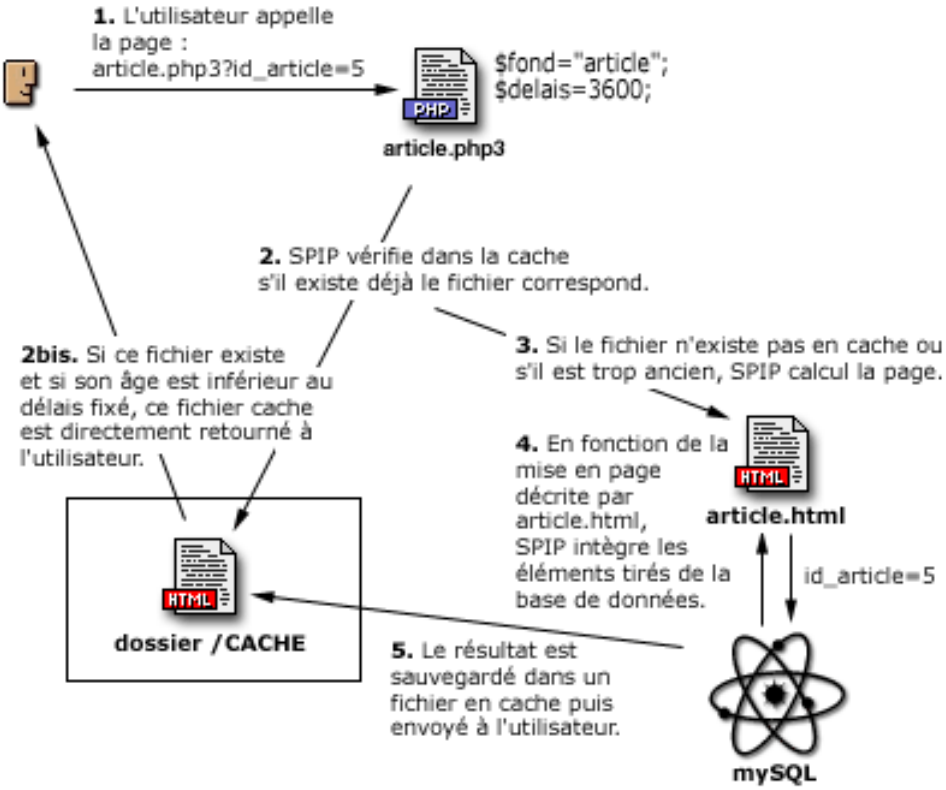
Vous pouvez naturellement modifier ces couples, et en créer d'autres.

1.2 Le principe de fonctionnement du cache

L'appel d'une page spécifique se fait par l'intermédiaire du fichier `.php3`.

Par exemple, pour appeler l'article 5, l'URL correspond est :

```
article.php3?id_article=5
```



1 Le fichier appelé est donc `article.php3`, avec en variable `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, 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 délais 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).

1.3 Le fichier .PHP3

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

```
<?
$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`. (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 `spip-article.html`, et on fixera dans `article.php3` : `$fond="spip-article"`.

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.

1.4 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 d'uZine[3] (visualisez le code source pour obtenir le texte du squelette).

1.5 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-heavy.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 » (tiret suivi d'un numéro de rubrique). Par exemple, si vous créez un fichier : `article-60.html`, tous articles contenus dans la rubrique 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 ce nouveau squelette.

> *Remarque.*
> Dans notre exemple, on aura certainement également intérêt à créer :
> un `rubrique-60.html`, 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.

[3] http ://www.uzine.net/article.html

1.6 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 squelette 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.

2 Des boucles et des balises

1er juin 2001 par l'équipe de SPIP

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 (section 1 page 1), 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[4], plus didactique, et revenir ensuite ici pour une documentation plus précise.

2.1 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 de sous-rubriques dans cette rubrique spécifique ?

[4] http ://www.uzine.net/rubrique144.html

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.

2.2 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 » …

2.3 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 (section 3 page 7) ;
- syntaxe générale des balises de SPIP (section 4 page 12) ;
- et, ensuite, une page spécifique à chaque type de boucles, indiquant quelles balises on peut y utiliser.

3 La syntaxe des boucles

28 mai 2001 par l'équipe de SPIP

3.1 Syntaxe de base

La syntaxe simplifiée d'une boucle est la suivante :

```
<BOUCLEn(TYPE){critère1}{critère2}...{critèrex}>

Code HTML + balises SPIP

</BOUCLEn>
```

On a vu, dans l'explication sur les boucles et les balises (section 2 page 5), 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 :

```
<BOUCLEn(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, au choix, le nom de la boucle, ou le numéro de la boucle. Cet élément est 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 manoeuvre) d'utiliser plusieurs boucles dans un même squelette : leur donner un nom 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)`. Cet élément 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}` ... Ils 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 ».

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

  Les différents critères et leur syntaxe seront explicités dans la suite, pour chaque type de boucle (certains critères fonctionnent pour tous les types de boucles (section 16 page 29), certains sont spécifiques à certaines boucles).

3.2 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 :

```
<Bn>

  Code HTML optionnel avant

<BOUCLEn(TYPE){critère1}{critère2}...{critèrex}>

  Code HTML + balises SPIP

</BOUCLEn>

  Code HTML optionnel après

</Bn>

  Code HTML alternatif

<//Bn>
```

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

</BOUCLE1>

  </UL>

</B1>

  Cette rubrique ne contient pas d'article.

<//B1>
```

donne les résultats suivants :

▷ **Il y a un seul article :**

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

▷ **Il y a plusieurs articles :**

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

▷ **Il n'y a aucun article :**

```
Cette rubrique ne contient pas d'article.
```

Remarque : La manière dont SPIP interprète les boucles interdit de mettre une boucle entre `<B1>` et `<BOUCLE1>`. Par contre, il est possible de mettre des boucles supplémentaires dans les parties optionnelles situées après la définition `<BOUCLE1...>`. Si vous devez vraiment installer une boucle dans la partie optionnelle *avant*, il faut passer par une commande `<INCLURE()>`, voir ci-dessous...

3.3 Des critères d'environnement en cascade

Chaque boucle effectue la sélection des éléments tirés de la base de données en fonction de critères. 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 :

`rubrique.php3?id_rubrique=15`

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

Clairement, avec le même code de squelette, si on appelle l'adresse :

`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>

      <LI>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 chacun 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.

3.4 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'intérieure 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 les autres.

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

```
<BOUCLE_rubrique(RUBRIQUES){id_rubrique}>

  Titre de la rubrique
    <BOUCLE_articles(ARTICLES){id_rubrique}>
    <LI> Titre de l'article
    </BOUCLE_articles>

    <BOUCLE_sous_rubriques(RUBRIQUES){id_rubrique}>
    <LI> Titre de la sous-rubrique
    </BOUCLE_sous_rubriques>

</BOUCLE_rubrique>

  Il n'y a pas de rubrique à cette adresse.

<//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 rubriques 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.

4 La syntaxe des balises SPIP

26 mai 2001 par l'équipe de 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 constitué d'éléments précis : un titre, une date, un texte, etc. A 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.

4.1 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}>
<LI> #TITRE
</BOUCLE_articles>
```

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

```
<LI> Titre de l'article 1
<LI> Titre de l'article 2
...
<LI> 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.

4.2 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}>
<LI> #SURTITRE<BR>
#TITRE
</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 : « `<LI> <BR>` », c'est-à-dire une petite puce suivie d'une ligne blanche.

Ce que nous devons faire : n'afficher le code « `<BR>` » 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}>
<LI> [(#SURTITRE)<BR>]
#TITRE
</BOUCLE_articles>
```

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

4.3 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|...|filtren) 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*).

> *Remarque.* En réalité, les *filtres* sont des fonctions PHP. Vous pouvez donc utiliser directement les fonctions de PHP (à condition que ces fonctions fonctionnent avec une seule variable), en plus des fonctions livrées avec SPIP.

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 (section 20 page 36) aux différents filtres livrés avec SPIP.

4.4 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|...|filtren) option après ]
```

4.5 Filtres à plusieurs variables

Attention, ce passage s'adresse aux utilisateurs avertis. [SPIP 1.5] autorise le passage de paramètres dans les filtres. La syntaxe est :

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

Le filtre doit être défini de la manière suivante dans `mes_fonctions.php3` :

```
<?php
function filtre($texte, $arg1='valeur par defaut 1', $arg2='valeur par défaut 2')
{
    ....calculs....
    return (une chaîne de caractères);
}
?>
```

5 La boucle ARTICLES

24 mai 2001 par l'équipe de SPIP

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

```
<BOUCLEn(ARTICLES){critères...}>
```

Les éléments contenus dans une telle boucle sont des articles.

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

5.1 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}` retourne l'article dont l'identifiant est `id_article`. Comme l'identifiant de chaque article est unique, ce critère ne retourne qu'une ou zéro réponse.
- `{id_rubrique}` retourne la liste des articles contenus dans la rubrique `id_rubrique`.
- `{id_secteur}` retourne les articles dans ce secteur (un secteur est une rubrique qui ne dépend d'aucune autre rubrique, c'est-à-dire située à la racine du site).
- `{id_auteur}` retourne les articles correspondant à cet identifiant d'auteur (utile pour indiquer la liste des articles écrits par un auteur).
- `{id_mot}` retourne les articles correspondant à cet identifiant de mot-clé (utile pour indiquer la liste des articles traitant d'un sujet donné).
- [SPIP 1.3] `{titre_mot=xxxx}`, ou `{type_mot=yyyy}` retourne les articles liés au mot-clé dont le nom est « xxxx », ou liés à des mots-clés du groupe de mots-clés « yyyy ». Attention, on ne peut pas utiliser plusieurs critères `{titre_mot=xxxx}` ou `{type_mot=yyyy}` dans une même boucle.
- [SPIP 1.4] `{id_groupe=zzzz}` permet de sélectionner les articles liés à un groupe de mots-clés ; principe identique au `{type_mot}` précédent, mais puisque l'on travaille avec un identifiant (numéro du groupe), la syntaxe sera plus « propre ».
- `{recherche}` retourne 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 (section 19 page 36).

5.2 Les critères d'affichage

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

- Les critères communs à toutes les boucles (section 16 page 29) s'appliquent évidemment.
- `{exclus}` permet d'exclure du résultat l'article dans lequel on se trouve déjà (par exemple, lorsque l'on affiche les articles contenus dans la même rubrique, on ne veut pas afficher un lien vers l'article dans lequel on se trouver déjà).
- `{doublons}` ou `{unique}` (ces deux critères sont rigoureusement identiques) permettent d'interdire l'affichage d'articles déjà affichés dans d'autres boucles elles-mêmes marquées `{doublons}`.

5.3 Les balises de cette boucle

Les balises tirées de la base de données

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

- `#ID_ARTICLE` affiche l'identifiant unique de l'article. Utile pour fabriquer des liens hypertextes non prévus (par exemple vers une page « Afficher au format impression ».
- `#SURTITRE` retourne le surtitre.
- `#TITRE` retourne le titre de l'article.
- `#SOUSTITRE` retourne le soustitre.
- `#DESCRIPTIF` retourne le descriptif.
- `#CHAPO` retourne le texte d'introduction (chapeau).
- `#TEXTE` retourne le texte principal de l'article.
- `#PS` retourne le post-scriptum.
- `#DATE` est la date de mise en ligne.(Modifiable après la mise en ligne.)
- `#DATE_REDAC` est la date de première publication.(Modifiable à volonté.)
- `#DATE_MODIF` [SPIP 1.5] est la date de dernière édition de l'article : précisément, il s'agit de la dernière date à laquelle cet article a été ouvert en édition *qu'il ait été modifié ou pas*. Pratique dans de nombreux cas, mais pas d'une rigueur scientifique . . . (Non modifiable, sauf si on veut la fixer à « maintenant » : il suffit alors . . . d'ouvrir l'article en édition.)
- `#ID_RUBRIQUE` est l'identifiant de la rubrique dont dépend l'article.
- `#ID_SECTEUR` est l'identifiant du secteur dont dépend l'article (le secteur étant la rubrique située à la racine du site).
- `#VISITES` est le nombre de visites sur cet article.
- `#POPULARITE` donne le pourcentage de popularité de cet article, voir la documentation : La « popularité » des articles (section 24 page 43).

Les balises calculées par SPIP

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

- `#NOTES` les notes de bas de page (calculées à partir de l'analyse du texte).
- `#INTRODUCTION` : [SPIP 1.4] si l'article contient un descriptif, c'est celui-ci qui est utilisé ici ; sinon, SPIP affiche les 600 premiers caractères du début de l'article (chapeau puis texte). [SPIP 1.3] Dans les versions précédentes de SPIP, ce sont systématiquement les premiers caractères de l'article (chapeau puis texte) qui sont pris en compte (le descriptif n'est pas utilisé).
- `#LESAUTEURS` les auteurs de cet article. Cela permet d'éviter de créer une boucle `AUTEURS` pour obtenir le même résultat.
- `#URL_ARTICLE` est l'URL de la page de l'article.
- `#FORMULAIRE_FORUM` fabrique l'interface permettant de poster un message répondant à cet article.
- `#FORMULAIRE_SIGNATURE` fabrique l'interface permettant de signer la pétition associée à cet article.
- `#PARAMETRES_FORUM` fabrique la liste des variables exploitées par l'interface du formulaire permettant de répondre à cet article. Par exemple :

```
[<A HREF="forum.php3?(#PARAMETRES_FORUM)">Répondre à cet article</A>]
```

Les logos

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

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

Les logos s'installent de la manière suivante :

```
[(#LOGO_ARTICLE|alignement|adresse)]
```

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

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

Par ailleurs deux balises permettent de récupérer un seul des deux logos :

- `#LOGO_ARTICLE_NORMAL` est le logo sans survol ;
- `#LOGO_ARTICLE_SURVOL` est le logo de survol.

6 La boucle RUBRIQUES

22 mai 2001 par l'équipe de SPIP

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.

6.1 Les critères de sélection

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

- `{tout}` les rubriques sont sélectionnées dans l'intégralité du site.
- `{id_rubrique}` retourne la rubrique dont l'identifiant est `id_rubrique`. Comme l'identifiant de chaque rubrique est unique, ce critère retourne une ou zéro réponse.
- `{id_secteur}` retourne les rubriques de ce secteur.
- `{id_parent}` retourne la liste des rubriques contenues dans une rubrique.
- `{id_enfant}` retourne la rubrique qui contient la rubrique (une seule réponse ; ou zéro réponse si la présente rubrique est située à la racine du site).
- `{meme_parent}` retourne la liste des rubriques dépendant de la même rubrique que la rubrique en cours. Permet d'afficher les rubriques « sœurs » qui se trouvent au même niveau dans la hiérarchie.
- `{recherche}` retourne les rubriques correspondant aux mots indiqués dans l'interface de recherche (moteur de recherche incorporé à SPIP). Voir la page consacrée au moteur de recherche (section 19 page 36).
- 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.

6.2 Les critères d'affichage

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

- ▷ Les critères communs à toutes les boucles (section 16 page 29) s'appliquent évidemment.
- ▷ {exclus} permet d'exclure du résultat la rubrique dans lequel on se trouve déjà (utile avec meme_parent).
- ▷ {doublons} ou {unique} (ces deux critères sont rigoureusement identiques) permettent d'interdire l'affichage de rubriques déjà affichées dans d'autres boucles.

6.3 Les balises de cette boucle

Les balises tirées de la base de données

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

- ▷ #ID_RUBRIQUE affiche l'identifiant unique de la rubrique.
- ▷ #TITRE retourne le titre de la rubrique.
- ▷ #DESCRIPTIF retourne le descriptif.
- ▷ #TEXTE retourne le texte principal de la rubrique.
- ▷ #ID_SECTEUR est l'identifiant du secteur dont dépend la rubrique (le secteur étant la rubrique située à la racine du site).

Les balises calculées par SPIP

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

- ▷ #NOTES les notes de bas de page (calculées à partir de l'analyse du texte).
- ▷ #INTRODUCTION les 600 premiers caractères du texte, les enrichissements typographiques (gras, italique) sont supprimés.
- ▷ #URL_RUBRIQUE est l'URL de la page de la rubrique.
- ▷ [SPIP 1.4] #DATE affiche la date de la dernière publication effectuée dans la rubrique et/ou ses sous-rubriques (articles, brèves ...).
- ▷ #FORMULAIRE_FORUM fabrique l'interface permettant de poster un message répondant à cette rubrique.
- ▷ #PARAMETRES_FORUM fabrique la liste des variables exploitées par l'interface du formulaire permettant de répondre à cette rubrique. Par exemple :

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

- ▷ #FORMULAIRE_SITE [SPIP 1.4] Le #FORMULAIRE_SITE affiche une interface permettant aux visiteurs du site de proposer des référencements de sites. Ces sites apparaîtront comme « proposés » dans l'espace privé, en attendant une validation par les administrateurs.
  Ce formulaire ne s'affiche que si vous avez activé l'option « Gérer un annuaire de sites » dans la *Configuration sur site* dans l'espace privé, et si vous avez réglé « Qui peut proposer des sites référencés » sur « les visiteurs du site public ».

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)]
```

- [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.

7 La boucle BREVES

20 mai 2001 par l'équipe de SPIP

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

```
<BOUCLEn(BREVES){critères...}>
```

7.1 Les critères de sélection

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

- `{tout}` les rubriques sont sélectionnés dans l'intégralité du site.
- `{id_breve}` retourne la brève dont l'identifiant est `id_breve`. Comme l'identifiant de chaque brève est unique, ce critère retourne une ou zéro réponse.
- `{id_rubrique}` retourne toutes les brèves contenues dans la rubrique en cours.
- [SPIP 1.2] `{id_mot}` retourne toutes les brèves liées au mot-clé en cours (à l'intérieur d'une boucle de type `(MOTS)`.
- [SPIP 1.3] `{titre_mot=xxxx}`, ou `{type_mot=yyyy}` retourne les brèves liées au mot-clé dont le nom est « xxxx », ou liées à des mots-clés du groupe de mots-clés « yyyy ». Attention, on ne peut pas utiliser plusieurs critères `{titre_mot=xxxx}` ou `{type_mot=yyyy}` dans une même boucle.
- [SPIP 1.4] `{id_groupe=zzzz}` permet de sélectionner les brèves liées à un groupe de mots-clés ; principe identique au `{type_mot}` précédent, mais puisque l'on travaille avec un identifiant (numéro du groupe), la syntaxe sera plus « propre ».
- `{recherche}` retourne les brèves correspondant aux mots indiqués dans l'interface de recherche (moteur de recherche incorporé à SPIP). Voir la page consacrée au moteur de recherche (section 19 page 36).

7.2 Les critères d'affichage

Les critères communs à toutes les boucles (section 16 page 29) s'appliquent.

7.3 Les balises de cette boucle

Les balises tirées de la base de données

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

- #ID_BREVE affiche l'identifiant unique de la brève.
- #TITRE retourne le titre de la brève.
- #DATE retourne la date de publication de la brève.
- #TEXTE retourne le texte de la brève.
- #NOM_SITE le nom du site indiqué en références.
- #URL_SITE l'adresse (URL) du site indiqué en références.
- #ID_RUBRIQUE l'identifiant de la rubrique dont dépend cette brève.

Les balises calculées par SPIP

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

- #NOTES les notes de bas de page (calculées à partir de l'analyse du texte.
- #INTRODUCTION les 600 premiers caractères du texte, les enrichissements typographiques (gras, italique) sont supprimés.
- #URL_BREVE est l'URL de la page de la brève.
- #FORMULAIRE_FORUM fabrique l'interface permettant de poster un message répondant à cette brève.
- #PARAMETRES_FORUM fabrique la liste des variables exploitées par l'interface du formulaire permettant de répondre à cette brève. Par exemple :

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

Le logo

- #LOGO_BREVE le logo de la brève, éventuellement avec la gestion du survol.
  Le logo s'installe de la manière suivante :

```
LOGO_BREVE|alignement|adresse)]
```

- #LOGO_BREVE_RUBRIQUE affiche, si il existe, le logo de la brève ; si ce logo n'a pas été attribué, SPIP affiche le logo de la rubrique [SPIP 1.4].

8 La boucle AUTEURS

18 mai 2001 par l'équipe de SPIP

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

```
<BOUCLEn(AUTEURS){critères...}>
```

8.1 Les critères de sélection

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

- {tout} les auteurs sont sélectionnés, qu'ils aient écrit un article ou non.
- {id_auteur} retourne l'auteur dont l'identifiant est id_auteur. Comme l'identifiant de chaque auteur est unique, ce critère retourne une ou zéro réponse.
- {id_article} retourne tous les auteurs de cet article.

8.2 Les critères d'affichage

Les critères communs à toutes les boucles s'appliquent (section 16 page 29).

8.3 Les balises de cette boucle

Les balises tirées de la base de données

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

- `#ID_AUTEUR` affiche l'identifiant unique de l'auteur.
- `#NOM` retourne le nom de l'auteur.
- `#BIO` retourne la biographie de l'auteur.
- `#EMAIL` retourne son adresse email.
- `#NOM_SITE` le nom de son site Web.
- `#URL_SITE` l'adresse (URL) de son site.
- `#PGP` sa clé publique pour PGP.
- `#FORMULAIRE_ECRIRE_AUTEUR` [SPIP 1.4] affiche un formulaire permettant d'écrire à l'auteur. Il faut que le serveur hébergeant le site accepte d'envoyer des mails. Ce système permet de ne pas divulguer l'adresse email de l'auteur.

Les balises calculées par SPIP

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

Le logo

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

Le logo s'installe de la manière suivante :

```
[(#LOGO_AUTEUR|alignement|adresse)]
```

9 La boucle FORUMS

16 mai 2001 par l'équipe de SPIP

La boucle `FORUMS` retourne une liste de messages de forums.

```
<BOUCLEn(FORUMS){critères...}>
```

9.1 Les critères de sélection

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

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

- ▷ {id_article} retourne les messages correspondant à cet article.
- ▷ {id_rubrique} retourne les messages correspondant à cette rubrique.
- ▷ {id_breve} retourne les messages correspondant à cette brève.
- ▷ {id_parent} retourne les messages dépendant d'un autre message. Indispensable pour gérer des *threads* dans les forums.
- ▷ {id_enfant} retourne le message dont dépend le message actuel (permet de « remonter » dans la hiérachie des *threads*). (SPIP 1.3)
- ▷ {meme_parent} retourne les autres messages répondant à un même message. (SPIP 1.3)
- ▷ {plat} : par défaut, seuls les messages n'ayant pas de parent (i.e. à la racine d'un *thread*) sont affichés. En ajoutant ce critère, vous pouvez sélectionner tous les messages quelle que soit leur position dans un thread (dans la limite des autres critères, bien sûr). Cela permet d'afficher les messages par ordre strictement chronologique par exemple, ou de compter le nombre total de contributions dans un forum.
- ▷ {id_secteur} retourne les messages correspondant au secteur. A priori, peu utile ; mais cela permet par exemple de faire un grand forum thématique regroupant tous les messages d'un secteur, quel que soit l'endroit où l'on se trouve.
- ▷ 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.

9.2 Les critères d'affichage

Les critères communs à toutes les boucles (section 16 page 29) s'appliquent .

9.3 Les balises de cette boucle

Les balises tirées de la base de données

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

- ▷ #ID_FORUM affiche l'identifiant unique du message.
- ▷ #ID_BREVE affiche l'identifiant de la brève à laquelle ce message est attaché. Attention, cela n'est pas récursif : un message qui répond à un message attaché à une brève ne contient pas lui-même le numéro de la brève.
- ▷ #ID_ARTICLE est l'identifiant de l'article à laquelle le message répond.
- ▷ #ID_RUBRIQUE l'identifiant de la rubrique à laquelle le message répond.
- ▷ #DATE est la date de publication.
- ▷ #TITRE est le titre.
- ▷ #TEXTE est le texte du message.
- ▷ #NOM_SITE le nom du site Web indiqué par l'auteur.
- ▷ #URL_SITE l'adresse (URL) de ce site Web.
- ▷ #NOM est le nom de l'auteur du message.
- ▷ #EMAIL est l'adresse email de l'auteur.
- ▷ #IP est l'adresse IP de l'auteur du message au moment de l'envoi de sa contribution.

Les balises calculées par SPIP

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

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

```
[<a href="forum.php3?(#PARAMETRES_FORUM)">Répondre à ce message</a>]
```

10 La boucle MOTS

14 mai 2001 par l'équipe de SPIP

La boucle MOTS retourne une liste de mots-clés.

```
<BOUCLEn(MOTS){critères...}>
```

10.1 Les critères de sélection

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

▷ {tout} les mots sont sélectionnés dans l'intégralité du site.
▷ {id_mot} retourne le mot-clé dont l'identifiant est id_mot.
▷ {id_groupe} retourne les mots-clés associés au groupe de mots dont le numéro est id_groupe [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).

10.2 Les critères d'affichage

Les critères communs à toutes les boucles (section 16 page 29) s'appliquent.

10.3 Les balises de cette boucle

Les balises tirées de la base de données

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

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

10.4 La boucle (GROUPES_MOTS)

D'une utilisation marginale, la boucle GROUPES_MOTS [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 ;
- #TITRE, le titre du groupe [NB : à l'intérieur de la boucle (MOTS), vous pouvez utiliser #TYPE pour afficher cette valeur].

11 La boucle DOCUMENTS

15 juin 2002 par ARNO*

[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}>
```

11.1 Les critères de sélection

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

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

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

11.2 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).
- ▷ {doublons} prend ici une importance particulière : elle permet non seulement de ne pas réafficher des documents déjà affichés par une autre boucle, mais également de ne pas réafficher les documents déjà intégrés à l'intérieur d'un article. Si l'on oublie ce critère, on affichera *tous* les documents associés à un article, *y compris* ceux qui auraient déjà été affichés à l'intérieur du texte.
- ▷ {extension=...} permet de sélectionner les documents selon leur terminaison (terminaison du fichier multimédia, par exemple « mov », « ra », « avi » ...). Cela peut être utilisé par exemple pour réaliser un *porfolio*, c'est-à-dire une boucle n'affichant que les documents de type image, une seconde boucle ensuite, avec une présentation graphique différente, les autres types de documents :

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

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

11.3 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.
- ▷ #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.
- ▷ #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)]
```

12 La boucle SITES (ou SYNDICATION)

12 mai 2001 par l'équipe de SPIP

La boucle `SITES` (SPIP 1.3) retourne une liste de sites référencés. (Si l'on a syndiqué des sites référencés, cette boucle s'utilise, naturellement, associée à une boucle `SYNDIC_ARTICLES` qui permet de récupérer la liste des article de ces sites.)

```
<BOUCLEn(SITES){critères...}>
```

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

```
<BOUCLEn(SYNDICATION){critères...}>
```

12.1 Les critères de sélection

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

- `{tout}`, tous les sites référencés.
- `{id_syndic}` retourne le site référencés dont l'identifiant est `id_syndic`.
- `{id_rubrique}` retourne les sites référencés dans cette rubrique.
- `{id_secteur}` retourne les sites référencés dans ce secteur.
- [SPIP 1.3] `{id_mot}` retourne toutes les sites liés au mot-clé en cours (à l'intérieur d'une boucle de type `(MOTS)`).
- [SPIP 1.3] `{titre_mot=xxxx}`, ou `{type_mot=yyyy}` retourne les sites liés au mot-clé dont le nom est « xxxx », ou liés à des mots-clés du groupe de mots-clés « yyyy ». Attention, on ne peut pas utiliser plusieurs critères `{titre_mot=xxxx}` ou `{type_mot=yyyy}` dans une même boucle.
- [SPIP 1.4] `{id_groupe=zzzz}` permet de sélectionner les sites liés à un groupe de mots-clés ; principe identique au `{type_mot}` précédent, mais puisque l'on travaille avec un identifiant (numéro du groupe), la syntaxe sera plus « propre ».

12.2 Les critères d'affichage

Les critères communs à toutes les boucles (section 16 page 29) s'appliquent.

- `{moderation=oui}` [SPIP 1.4] affiche les sites syndiqués dont les liens sont bloqués a priori (« modérés ») ; l'inverse de ce critère est `{moderation!=oui}`.
- (SPIP 1.3) `{syndication=oui}`, `{syndication=non}` permet de n'afficher que les sites référencés faisant l'objet d'une syndication, ou les sites non syndiqués.
- (SPIP 1.2) `{doublons}` ou `{unique}` (ces deux critères sont rigoureusement identiques) permettent d'interdire l'affichage de sites référencés déjà affichés dans d'autres boucles.

12.3 Les balises de cette boucle

Les balises tirées de la base de données

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

- `#ID_SYNDIC` affiche l'identifiant unique du site syndiqué.
- `#NOM_SITE` est le nom du site syndiqué.
- `#URL_SITE` est l'adresse (URL) du site syndiqué.
- `#DESCRIPTIF` est le descriptif du site syndiqué.
- `#ID_RUBRIQUE` est le numéro de la rubrique contenant cette syndication.
- `#ID_SECTEUR` est le numéro de la rubrique-secteur (à la racine du site) contenant cette syndication.

Autre balise

- `#LOGO_SITE` affiche le logo attribué au site.

13 La boucle SYNDIC_ARTICLES

10 mai 2001 par l'équipe de SPIP

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

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

> (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}`.

13.1 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.

13.2 Les critères d'affichage

Les critères communs à toutes les boucles (section 16 page 29) s'appliquent.

13.3 Les balises de cette boucle

Les balises tirées de la base de données

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

- #ID_SYNDIC_ARTICLE affiche l'identifiant unique de l'article syndiqué.
- #ID_SYNDIC affiche l'identifiant unique du site syndiqué contenant cet article syndiqué.
- #TITRE est le titre de l'article.
- #URL_ARTICLE est l'adresse (URL) du site syndiqué (sur son site original).
- #DATE est la date de publication de cet article.
- #LESAUTEURS, les auteurs de l'article syndiqué.
- #DESCRIPTIF le descriptif de l'article syndiqué.
- #NOM_SITE est le nom du site syndiqué contenant cet article.
- #URL_SITE est l'adresse (URL) du site

14 La boucle SIGNATURES

7 mai 2001 par l'équipe de SPIP

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

```
<BOUCLEn(SIGNATURES){critères...}>
```

14.1 Les critères de sélection

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

- {tout} toutes les signatures sont 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.

14.2 Les critères d'affichage

Les critères communs à toutes les boucles (section 16 page 29) s'appliquent.

Attention. Dans ce type de boucles, certains critères de classement de la boucle 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.

14.3 Les balises de cette boucle

Les balises tirées de la base de données

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

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

15 La boucle HIERARCHIE

6 mai 2001 par l'équipe de SPIP

La boucle HIERARCHIE est basée sur un principe différent des autres boucles, car elle effectue une récursivité (au lieu de simplement tirer une liste de la base de données).

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

```
<BOUCLEn(HIERARCHIE){critères...}>
```

15.1 Les critères de sélection

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

▷ {id_rubrique} retourne la liste des rubriques depuis la racine jusqu'à la rubrique correspondant à cet identifiant.
▷ {id_article} retourne la liste des rubriques depuis la racine jusqu'à l'article correspondant à cet identifiant.

15.2 Les critères d'affichage

Attention : les critères communs à toutes les boucles (section 16 page 29) ne s'appliquent pas tous à ce type de boucle.

Les critères d'affichage utilisables sont : {"inter"} et {a,b}.

15.3 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 (section 6 page 17).

16 Les critères communs à toutes les boucles

5 mai 2001 par l'équipe de SPIP

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.

16.1 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.

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).

16.2 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.

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.

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. (SPIP 1.2) On peut utiliser la notation `{xxx != yyy}`, le `!` correspondant à la négation (opérateur logique `NOT`).

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

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

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

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

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. La 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 connait 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).

Par exemple :

```
<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>
```

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.

16.3 Affichage d'une partie des résultats

▷ `{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 (celle 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 :

```
petition.php3?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 200-ième. Avec l'URL :

```
petition.php3?id_article=13&debut_signatures=300
```

on obtient la liste des 100 signatures à partir de la 300-ième.

▷ `{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 ...

16.4 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}{", "}>
```

17 Les balises propres au site

7 décembre 2001 par l'équipe de SPIP

Certaines balises sont disponibles en dehors des boucles ; leur contenu est défini lors de la configuration de votre site.

- `#URL_SITE_SPIP` est l'adresse du site. Elle ne comprend pas le / final, ainsi vous pouvez créer un lien du type `#URL_SITE_SPIP/sommaire.php3`
- `#NOM_SITE_SPIP` est le nom du site.
- `#EMAIL_WEBMASTER` [SPIP 1.5] est l'adresse du webmestre. Par défaut, SPIP prend l'adresse de celui qui a installé le site (le premier administrateur). (Si vous préférez un formulaire « écrire au webmestre », cf. Les formulaires section 18 page 33).
- `#CHARSET` [SPIP 1.5] est le jeu de caractères utilisé par le site. Sa valeur par défaut est `iso-8859-1`, jeu de caractères dit « iso-latin ». Cf. *Voyage dans la tour de Babel du net*[5] pour une introduction aux charsets, en attendant une documentation plus complète de cette fonctionnalité de SPIP.
- Par ailleurs, pour aller plus vite dans l'écriture des squelettes, on peut faire usage de la balise : `#PUCE` [SPIP 1.5], qui affiche devinez-quoi ;
- et, pour améliorer le placement sur la page des boutons réservés aux admins (« Recalculer cette page », « Modifier cet article »), il existe une balise `#FORMULAIRE_ADMIN` [SPIP 1.5]. Cette balise est optionnelle : si elle est présente, les boutons d'administration s'afficheront à leur emplacement, sinon ils viendront en bas de page.

18 Les formulaires

16 août 2002 par l'équipe de SPIP

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.

18.1 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 (section 19 page 36).

#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` (section 9 page 21).

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

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 (section 23 page 41).

[5] http ://www.uzine.net/article1785.html

#FORMULAIRE_SIGNATURE

Le `#FORMULAIRE_SIGNATURE` autorise la signature des pétitions associées aux articles (ce formulaire se place donc dans une boucle `ARTICLES`.

N.B. 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

[SPIP 1.4] Le `#FORMULAIRE_SITE` affiche une interface permettant aux visiteurs du site de proposer des référencements de sites. Ces sites apparaîtront comme « proposés » dans l'espace privé, en attendant une validation par les administrateurs.

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

Les sites référencés étant, dans SPIP, attachés aux rubriques, on ne peut placer ce `#FORMULAIRE_SITE` qu'à l'intérieur d'une boucle `RUBRIQUES` (section 6 page 17).

#FORMULAIRE_ECRIRE_AUTEUR

[SPIP 1.4] Placé à l'intérieur d'une boucle `AUTEURS` (section 8 page 20), ce formulaire permet d'envoyer un mail à l'auteur (d'un article). Cela permet, en modifiant les squelettes (qui, par défaut, affichent les liens contenant les adresses email des auteurs des articles), de pouvoir écrire aux auteurs sans afficher leur adresse email sur le site public.

18.2 Inscription, authentification . . .

#FORMULAIRE_INSCRIPTION

Sans doute le plus important, le `#FORMULAIRE_INSCRIPTION` gère l'inscription des nouveaux rédacteurs. Il n'affiche une interface d'inscription 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 ; donc ce formulaire ne fonctionne que si votre hébergeur autorise l'envoi de mails par PHP.

#LOGIN_PRIVE

[SPIP 1.4] Tout aussi important (sinon plus), le `#LOGIN_PRIVE` 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_login.php3`, c'est-à-dire en standard par le squelette nommé `login-dist.html` En effet, lors des accès directs à l'adresse « `/ecrire` » de votre site, c'est vers `spip_login.php3` que SPIP va vous rediriger.

#LOGIN_PUBLIC

[SPIP 1.4] D'une utilisation beaucoup plus spécifique, `#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 aura probablement, dans une future version de spip, un usage élargi, mais, pour l'instant, elle ne sert qu'à authentifier les visiteurs pour les sites proposant des forums *modérés sur abonnement*.

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|mapage.php3)]
```

Si votre site offre une inscription automatique à l'espace privé, les données de connexion à l'espace public sont identifiques 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 un URL permettant à un visiteur authentifié de se déconnecter.

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 } ?>
```

18.3 Feuilles de style

On peut notablement modifier l'interface graphique des formulaires par l'intérmédiaire des feuilles de style[6], notamment les classes `forml`, `spip_encadrer` et `spip_bouton`.

[6] http ://www.uzine.net/article1177.html

19 Les boucles de recherche

4 mai 2001 par l'équipe de SPIP

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

19.1 L'interface de recherche

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

```
#FORMULAIRE_RECHERCHE
```

Par défaut, le formulaire enverra les requêtes vers une page `recherche.php3` ; 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|adresse.php3)]
```

où `adresse.php3` est la page vers laquelle vous désirez envoyer l'utilisateur.

19.2 Le squelette des résultats

Les boucles permettant d'afficher les résultats de la recherche sont, en réalité, des boucles déjà abordées ici : `ARTICLES` (section 5 page 15), `RUBRIQUES` (section 6 page 17), `BREVES` (section 7 page 19). Vous pouvez en effet effectuer des recherches non seulement sur les articles, mais aussi sur les rubriques et les brèves.

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 critères d'affichage et les balises de ces boucles sont inchangées.

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 indique la pertinence des résultats (attention, dans l'absolu cette valeur n'est pas très explicite, elle est surtout utile pour le classement des résultats).

20 Les filtres de SPIP

3 mai 2001 par l'équipe de SPIP

Nous avons vu dans la syntaxe des balises SPIP (section 4 page 12) qu'il était possible de modifier le comportement et l'affichage des balises en leur attribuant des filtres.

```
[ option avant (#BALISE|filtre1|filtre2|...|filtren) option après ]
```

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

20.1 Les filtres de mise en page

- ▷ `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>`).

20.2 Les filtres des dates

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

- ▷ `affdate` affiche la date en français, par exemple « 13 janvier 2001 ».
- ▷ `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 . . .).

20.3 Filtres de logos

- ▷ `fichier` [SPIP 1.4]. Affecté à un logo, ce filtre permet de récupérer directement l'URL du fichier du 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.php3`) ;
  - ⋆ 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`).

20.4 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 fenêtre ; 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'article `{par num titre}` mais ne pas afficher les numéros (car ils ne servent qu'à ordonner les articles).

- `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 `<br>` en sauts de ligne, et les espaces insécables en espaces simples. Utilisation, par exemple, pour faire un descriptif `META` à partir du
  `#DESCRIPTIF` : `[<meta name='description' content='(#DESCRIPTIF|textebrut)'>]`

20.5 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

```
<a href="#URL_ARTICLE" [ title = "(#DESCRIPTIF|supprimer_tags
                                        |attribut_html)" ]>#TITRE</a>
```

20.6 Ajouter ses propres fonctions

Les filtres de SPIP sont des fonctions PHP à une seule variable. Vous pouvez utiliser directement les fonctions habituelles de PHP, mais également créer les vôtres (à la condition qu'elles n'aient qu'une seule variable), 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.php3` : si SPIP repère un fichier ayant ce nom, il l'inclut automatiquement.

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://rezo.net/spip-dev/contrib/ARNO/` ; il suffit de l'ajouter dans votre fichier `mes_fonctions.php3` pour l'utiliser.

Autre exemple, pour faire un filtre qui coupe les textes à une longueur donnée (ex : 50 caractères), on peut créer le filtre personnalisé suivant dans `mes_fonctions.php3` :

```
<?php
    function couper50($texte) {
        return couper($texte, 50);
    }
?>
```

21 Les boucles récursives

2 mai 2001 par l'équipe de SPIP

Les boucles récursives sont une fonction très puissante pour gérer la mise en forme de l'interface. Leur programmation est particulièrement simple, mais leur utilisation demande une bonne maîtrise logique de l'enchaînement des boucles.

L'appel d'une boucle récursive est très simple : il suffit d'indiquer dans le TYPE de la boucle le nom d'une autre boucle :

```
<BOUCLEn(bouclex)></BOUCLEn>
```

Il n'y a ici aucun critère : en réalité, la boucle n correspond à une copie pure et simple de la boucle x. L'ensemble de la boucle 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. (Il faut bien entendu que la boucle x précède la boucle n.)

L'utilisation la plus simple consiste à « dupliquer » une boucle sans avoir à la recopier. Ainsi, toute modification de la boucle d'origine x sera automatiquement dupliquée dans la boucle n.

Tout l'intérêt, en réalité, consiste à placer la boucle n à l'intérieur de la boucle x : on obtient ainsi 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.

Cette technique permet de créer notamment l'affichage des thread des forums. Cela devient très simple : 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}>
<P>#TITRE

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

</BOUCLE_forum>
```

On peut ainsi, en très peu de lignes, provoquer l'affichage de l'intégralité de la structure (rubriques, sous-rubriques . . .) du site.

22 <INCLURE> d'autres squelettes

16 août 2002 par l'équipe de SPIP

[SPIP 1.4] Lorsque l'on a des éléments de texte et des boucles communs à plusieurs fichiers, on peut vouloir extraire ces éléments des pages où ils se trouvent, les installer dans un fichier séparé, et les appeler depuis les autres squelettes. De cette façon, le code commun est regroupé dans un unique fichier, ce qui facilite notamment les modifications qui concernent plusieurs squelettes d'un seul coup.

> Les habitués de PHP connaissent la fonction include, dont le principe est similaire à ce qui est présenté ici.

Dans SPIP, on peut appeler un fichier depuis un autre squelette grâce à la balise `<INCLURE>` (on peut aussi utiliser `<INCLUDE>`, qui est identique). Sa syntaxe complète est :

```
<INCLURE(fichier.php3){paramètre1}{paramètre2}...>
```

Le « `fichier.php3` » est le nom du fichier que l'on veut intégrer dans sa page. Par exemple, imaginons que toutes les pages du site affichent les mêmes informations en bas de page. On regroupe alors le code HTML de ce « pied de page » dans un fichier : « `pied.html` », squelette lui-même appelé par la page « `pied.php3` » (toujours selon le principe des couples de fichiers destinés à appeler des squelettes). Il suffit d'intégrer dans chacun des squelettes voulant appeler ce fichier :

```
<INCLURE(pied.php3)>
```

Certaines inclusions peuvent dépendre d'une variable.
Par exemple, imaginons un squelette « `hierarchie.html/php3` », qui affiche la hiérarchie menant à une rubrique ; on appelerait cette page par une URL de la forme :
`hierarchie.php3?id_rubrique=xxx`.

Dans les squelettes voulant afficher la hiérarchie à partir de la rubrique courante, il faut donc indiquer que le paramètre concerné est `{id_rubrique}` ; si nécessaire, on aura créé une boucle permettant de récupérer le numéro de la rubrique concernée, et on installera le code suivant à l'intérieur de cette boucle :

```
<INCLURE(hierarchie.php3){id_rubrique}>
```

On peut imaginer que, dans certains squelettes, on désire récupérer non pas la hiérarchie en fonction d'une rubrique « variable » (la rubrique dans laquelle se trouve l'appel d'inclusion), mais en fonction d'une rubrique dont on connaît à l'avance le numéro. Pour cela, on peut fixer la valeur du paramètre ainsi :

```
<INCLURE(hierarchie.php3){id_rubrique=5}>
```

N.B. Il est possible d'indiquer plusieurs paramètres dans la balise `<INCLURE>` ; cependant ce cas est rare. En effet, il est déjà rare dans un site sous SPIP de créer volontairement des URL à plusieurs variables, il est donc tout aussi rare d'avoir besoin d'inclusions avec plusieurs paramètres. En tout état de cause ces variables se cumuleraient par un « et logique » comme lorsqu'on les juxtapose dans la définition des boucles.

N.B. Le fichier inclus est lui-même un couple de fichiers de squelette. Ce fichier disposera donc de sa propre valeur de `$delais`. Cela peut s'avérer pratique pour éviter des recalculs d'éléments lourds du site, qui ne changent pas, en y affichant une liste d'éléments qui, eux, sont très régulièrement remis à jour.

23 Les variables de personnalisation

16 août 2002 par l'équipe de SPIP

Certains comportements des pages de votre site peuvent être modifiés au moyen de variables PHP. Ces variables sont normalement définies par SPIP, mais, pour obtenir une personnalisation plus fine du site, le webmestre peut les modifier.

23.1 Où indiquer ces variables ?

Inutile d'entrer dans le code source de SPIP lui-même pour fixer ces variables (ouf !).

Pour l'ensemble du site

Si vous voulez fixer ces variables pour l'intégralité du site, vous pouvez les indiquer, avec une syntaxe un peu différente, dans un fichier intitulé `mes_fonctions.php3`, placé à la racine du site. (Il faudra éventuellement créer ce fichier, et entourer les définitions de vos variables par les marqueurs `<?php et ?>`, voir les exemples ci-dessous.)

Pour chaque type de squelette

[SPIP 1.4] Vous pouvez aussi définir ces variables squelette par squelette. Pour cela, il faut les installer *au début* du fichier PHP appelant le squelette (par exemple `article.php3`, `rubrique.php3` ...). Ils s'insèrent naturellement à côté des variables obligatoires `$fond` et `$delais`.

23.2 Les variables du texte

Ces variables sont utilisées lors du calcul de la mise en page (correction typographique) par SPIP.

- `$debut_intertitre` fixe le code HTML inséré en ouverture des intertitres (par le raccourci `{{{`). En standard, sa valeur est :

```
$debut_intertitre = "\n <h3 class=\"spip\">\n";
```

- `$fin_intertitre` est le code HTML inséré en fermeture des intertitres (raccourci `}}}`). Sa valeur normale est :

```
$fin_intertitre = "\n</h3><br>\n";
```

  N.B. L'intertitre standard proposé par SPIP peut sembler laid ou inefficace ; il s'explique par l'histoire du développement de ce programme ; sur votre propre site, vous pouvez évidemment choisir de créer un code qui plus « élégant » (en faisant notamment appel à une simple feuille de style).
- `$ouvre_ref` est le code d'ouverture des appels des notes de bas de page ; par défaut, c'est une espace insécable et un crochet ouvrant ;
- `$ferme_ref` est le code de fermeture des appels des notes de bas de page ; par défaut, c'est un crochet fermant.
- `$ouvre_note` est le code d'ouverture de la note de bas de page (telle qu'elle apparaît dans `#NOTES`) ; par défaut, un crochet ouvrant ;
- `$ferme_note` est le code de fermeture des notes de bas de page (un crochet fermant).
  Des choix alternatifs pourront être par exemple d'utiliser des parenthèses ; ou, plus joliment, d'ouvrir avec le tag HTML `<sup>`, et de fermer avec `</sup>`.

▷ Le fichier `puce.gif` et la variable `$puce`. Lorsque vous commencez une nouvelle ligne par un tiret, SPIP le remplace par une petite « puce » graphique. Cette puce est constituée par le fichier `puce.gif` installé à la racine du site ; vous pouvez modifier ce fichier selon vos besoins. Mais vous pouvez aussi décider de fixer vous-même le choix de la puce, au travers de la variable `$puce`. Par exemple pour indiquer un autre fichier graphique :

```
$puce = "<img src='mapuce.gif' alt='-' align='top' border='0'>";
```

ou par un élément HTML non graphique :

```
$puce = "---";
```

23.3 Les variables pour les forums publics

Il existe des variables permettant de fixer le comportement des forums publics *avec des mots-clés*.

N.B. Ces variables ne sont utilisées que lorsque vous créez des forums publics dans lesquels les visiteurs peuvent sélectionner des mots-clés ; leur utilisation est donc extrêmement spécifique (et pas évidente . . .).

▷ `$afficher_texte` (« oui »/« non »). Par défaut, les forums publics sont conçus pour permettre aux visiteurs d'entrer le texte de leur message ; mais lorsque l'on propose le choix de mots-clés dans ces forums, on peut décider qu'aucun message n'est utile, seul la sélection des mots-clés importe. Dans ce cas, on pourra indiquer :

```
$afficher_texte = "non";
```

▷ `$afficher_groupe` permet d'indiquer les différents groupes de mots-clés que l'on souhaite proposer dans tel forum. En effet, tous les forums sur un site ne sont pas forcément identiques, et si, à certains endroits, on peut vouloir afficher une sélection de tous les groupes de mots-clés (ceux que l'ont a rendu accessibles aux visiteurs depuis l'espace privé), à d'autres endroits, on peut vouloir n'utiliser que certains groupes, voire aucune groupe (pas de sélection de mots-clés du tout).
La variable `$afficher_groupe` est un tableau (`array`), et se construit donc de la façon suivante :

```
$afficher_groupe[] = 3;
$afficher_groupe[] = 5;
```

impose l'affichage *uniquement* des groupes 3 et 5.

```
$afficher_groupe[] = 0;
```

interdit l'utiliser des mots-clés dans ces forums (puisqu'il n'existe pas de groupe de mots-clés numéroté 0.
Si l'on n'indique rien (on ne précise pas `$afficher_groupe`), tous les groupes de mots-clés indiqués, dans l'espace privés, comme « proposés aux visiteurs du site public » sont utilisés.

23.4 Le dossier des squelettes

[SPIP 1.5] Si l'on souhaite mettre les squelettes de son site dans un dossier particulier, par exemple pour faire des essais de différents jeux de squelettes trouvés sur Internet, ou parce qu'on aime que les choses soient bien rangées, etc., il est possible de fixer dans `mes_fonctions.php3` la variable `$dossier_squelettes`.

```
<?php
    $GLOBALS['dossier_squelettes'] = 'design';
?>
```

A partir de ce moment-là, aucun des squelettes fournis en standard avec SPIP (les `article-dist.html`) ne fonctionnera plus, pas plus que ceux qui vous aurez installé à la racine du site. Il vous faut alors créer le dossier `design/` à la racine du site, et y installer votre squelette d'article, par exemple, faute de quoi la page `article.php3?id_article=1` vous indiquera obstinément que « *le squelette* `design/article.html` *n'existe pas* ».

Les avantages de ce rangement peuvent sembler 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.) ; l'inconvénient principal est qu'il sera plus difficile de visualiser les squelettes via un simple navigateur (les liens vers les images, notamment, risquent de « casser »).

23.5 Exemples

▷ Codes modifiés globalement, fichier `mes_fonctions.php3` (attention, la syntaxe change un peu, il faut ajouter la mention `$GLOBALS['xxx']` pour chacune des variables à personnaliser.

```
<?php
   $GLOBALS['debut_intertitre'] = "<h3 class='mon_style_h3'>";
   $GLOBALS['fin_intertitre'] = "</h3>";

   $GLOBALS['ouvre_ref'] = ' (';
   $GLOBALS['ferme_ref'] = ')';
   $GLOBALS['ouvre_note'] = '(';
   $GLOBALS['ferme_note'] = ') ';

   $GLOBALS['espace_logos'] = 0;
?>
```

▷ Codes modifiés pour les seules pages de rubrique, fichier `rubrique.php3` :

```
<?php
   $fond = "rubrique";
   $delais = 2 * 3600;
   $espace_logos = 20;
   include ("inc-public.php3");
?>
```

24 La « popularité » des articles

31 juillet 2002 par l'équipe de SPIP

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

24.1 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 :

- 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 ...

24.2 Balises

Des balises permettent de récupérer et d'afficher ces valeurs. 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>
```

NB : 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. »

24.3 Critères

Enfin, un critère de tri peu 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}>
...
</BOUCLE_hitparade>
```

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

25 Utiliser des URLs personnalisées

1er mai 2001 par l'équipe de SPIP

Par défaut, les pages générées par SPIP utilisent des adresses relatives ressemblant à `article.php3?id_article=123`, donnant des URLs du type `http://www.minirezo.net/article.php3?id_article=123`. Ce type de syntaxe, courant chez les sites « dynamiques », n'est cependant pas très joli ni très évocateur. Il y a possibilité d'avoir des adresses plus à votre goût - par exemple `article123.html` - , et SPIP vous aide en partie dans cette tâche.

Cette fonctionnalité fait appel à la distinction entre deux types d'URLs :

- *l'URL apparente* d'une page, c'est-à-dire telle qu'elle est tapée et/ou affichée dans la barre d'adresse du navigateur. Par exemple `http://www.uzine.net/article765.html`. Ce sont ces URLs qu'on cherche à rendre plus « jolies » ou plus « signifiantes » ;
- *l'URL réelle* de la page, c'est-à-dire l'URL qui est « vue » par SPIP lorsque la page est calculée sur le serveur. Par exemple `http://www.uzine.net/article.php3?id_article=765` ; en général, cette URL peut aussi être tapée directement dans le navigateur (vous pouvez vérifier).

25.1 Choisir le type d'URLs apparentes

Le fichier `inc-urls.php3` à la racine de SPIP, contient la déclaration d'une variable PHP contenant le type d'URLs à utiliser. Par défaut :

```
$type_urls = "standard";
```

Cette variable détermine le nom du fichier PHP qui est appelé pour gérer les URLs. Avec la déclaration par défaut ci-dessus, c'est `inc-urls-standard.php3`. Vous remarquerez qu'il y a aussi un fichier `inc-urls-html.php3`. Il permet de traiter des adresses du type de celles que nous avons prises comme exemple (« `article123.html` »). Vous pouvez donc décider d'utiliser plutôt ce fichier que le fichier « standard » en remplaçant la ligne précitée par la suivante :

```
$type_urls = "html";
```

Si vous voulez plutôt utiliser vos propres adresses (ce pour quoi vous devez savoir programmer en PHP), il est fortement conseillé de partir d'un des fichiers existants et de le recopier sous le nom que vous aurez choisi : `inc-urls-XXX.php3`.

Voyons maintenant les différents types de modifications à apporter.

25.2 Programmer la traduction des adresses apparentes en adresses réelles

Pour que l'adresse `article123.html` appelle bien en réalité le fichier PHP `article.php3` avec comme paramètre `id_article=123`, il va falloir configurer le serveur Web qui héberge votre site, soit dans un fichier `.htaccess` (ça ne marche pas toujours), soit dans le fichier de configuration centrale du serveur si vous y avez accès. Cela utilise, sous le serveur Apache[7] (le plus utilisé), ce qu'on appelle des *Rewrite Rules* : des règles de réécriture d'adresses Web.

Savoir écrire ces règles n'est pas simple pour les non-programmeurs, et nous ne pouvons pas vous donner de solutions infaillibles car cela dépend de votre configuration : cette partie est entièrement entre vos mains (ou celles de votre hébergeur).

Néanmoins et à titre d'exemple, voici les règles utilisées sur le site d'uZine :

[7] http ://httpd.apache.org/

```
RewriteEngine on

# urls spip
RewriteRule ^/rubrique([0-9]+)\.html$   /rubrique.php3?id_rubrique=$1 [QSA,L]
RewriteRule ^/article([0-9]+)\.html$    /article.php3?id_article=$1 [QSA,L]
RewriteRule ^/breve([0-9]+)\.html$      /breve.php3?id_breve=$1 [QSA,L]
RewriteRule ^/secteur([0-9]+)\.html$    /secteur.php3?id_rubrique=$1 [QSA,L]
```

(note si vous écrivez vos propres règles : utilisez toujours l'option `[QSA]` car l'adresse peut présenter des paramètres supplémentaires - comme lorsque vous appuyez sur le bouton « recalcul », affiché si vous placez le cookie d'administration ; l'option `[L]` signifie, quant à elle, que la réécriture s'arrête là si la règle est appliquée : `Last`)

Il est conseillé de tester la validité de ces adresses (en les essayant depuis votre navigateur) dès que vous avez mis en place la configuration correspondante, car sinon la suite ne servirait à rien.

25.3 Générer les URLs apparentes dans les pages SPIP

SPIP incorpore une fonctionnalité permettant de générer automatiquement les URLs du type que vous aurez choisi, à l'intérieur même des pages calculées par SPIP. Ainsi, les liens vers des articles seront par exemple générés sous la forme `article123.html`.

Les fichiers `inc-urls-XXX.php3` sont là à cet effet. Si vous avez décidé de créer le vôtre propre, il va falloir programmer vous-mêmes la génération des URLs. Sinon, c'est déjà fait. Vous découvrirez qu'un certain nombre de fonctions PHP sont définies, dans le fichier, sous le nom `generer_url_type` d'objet (par exemple `generer_url_article`). Le rôle de ces fonctions devrait vous être évident pour peu que vous compreniez un minimum le PHP. A vous de les modifier à votre guise. A titre d'exemple, vous pouvez vous plonger dans le fichier `inc-urls` *du Monde diplo*[8].

Dans tous les cas, n'oubliez pas d'aller modifier le type choisi à l'intérieur de `inc-urls.php3`.

Enfin, dans vos squelettes, vérifiez que pour calculer les liens à l'intérieur des boucles, vous utilisez toujours les raccourcis `#URL_ARTICLE`, `#URL_RUBRIQUE`, `#URL_BREVE`, etc. Ce sont en effet eux qui déclenchent l'utilisation des fonctions évoquées ci-dessus, permettant la génération des « bonnes » URLs.

26 Le support LDAP

11 décembre 2001 par l'équipe de SPIP

Attention, cet article est vraiment destiné à des utilisateurs avancés, qui maîtrisent l'usage de LDAP et souhaitent appuyer SPIP sur un annuaire LDAP existant.

LDAP (Lightweight Directory Access Protocol) est un protocole permettant d'interroger un annuaire contenant des infos d'utilisateurs (nom, login, authentification...). Depuis la version [SPIP 1.5] il est possible de vérifier si un rédacteur est dans la base LDAP avant de lui donner accès à l'espace privé.

A l'installation, SPIP détecte si PHP a été compilé avec le support LDAP. Si oui, à la cinquième étape (créer un accès), un bouton permet d'ajouter un annuaire LDAP à la configuration SPIP. La configuration qui suit est relativement simple, elle essaie de deviner les paramètres au maximum. Notamment, elle permet de choisir le statut par défaut des auteurs venant de l'annuaire.

Note : si on ne s'en sert pas, PHP n'est généralement pas compilé avec le support LDAP, donc aucune indication supplémentaire ne viendra perturber l'installation sur les configurations habituelles.

[8] http ://rezo.net/spip-dev/contrib/Fil/inc-urls-diplo.php3

Les infos de connexion au serveur LDAP sont écrites dans `inc_connect.php3`. Corollaire : il faut supprimer ce fichier et relancer l'installation pour activer LDAP sur un site SPIP existant.

Dans la table `spip_auteurs`, est ajouté un champ « source » qui indique d'où viennent les infos sur l'auteur. Par défaut, c'est « spip », mais ça peut aussi prendre la valeur « ldap ». Ca permet de savoir notamment quels champs ne doivent pas être changés : en particulier, on ne doit pas autoriser la modification du login, car sinon il y a une perte de synchronisation entre SPIP et LDAP.

A l'authentification, les deux méthodes sont testées à la suite : SPIP puis LDAP. En fait un auteur LDAP ne pourra pas être authentifié par la méthode SPIP (méthode standard avec challenge md5) car le pass est laissé vide dans la table `spip_auteurs`. Un auteur SPIP, quant à lui, sera authentifié directement depuis la table `spip_auteurs`. D'autre part, si le login entré ne vient pas de SPIP, le mot de passe est transmis en clair.

Quant un auteur LDAP se connecte pour la première fois, son entrée est ajoutée dans la table `spip_auteurs`. Les champs remplis sont : nom, login et email qui viennent de LDAP (champs 'cn', 'uid' et 'mail' respectivement) et le statut dont la valeur par défaut a été définie à l'installation (rédacteur, admin ou visiteur). Important : on peut modifier le statut par la suite, afin de choisir ses admins à la main par exemple.

Une fois un auteur connecté, il est authentifié par la voie classique, c'est-à-dire simplement avec le cookie de session. Ainsi on ne se connecte à LDAP que lors du login (`spip_cookie.php3`). De même, les infos prises en compte dans l'affichage et les boucles sont celles de `spip_auteurs`, pas celles de l'annuaire.

Pour les auteurs SPIP, rien ne change. On peut les créer et les modifier comme à l'habitude. D'ailleurs, le premier compte admin sera toujours un compte SPIP (rien n'est prévu pour créer le premier compte admin depuis LDAP lors de l'install). C'est préférable pour éviter d'être bloqué en cas de panne du serveur LDAP.

Index