Manuel de « référence »

Version hors-ligne du site Spip.net

Tome 3b

Guide du webmestre et du bidouilleur

Habillage graphique
Trucs et astuces
Guide des fonctions avancées

29 septembre 2007 Rabillard Fabien

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

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

Le document que vous avez entre les mains ou sur votre écran est un copier/coller du site Spip.net effectué entre le 20 et le 29 septembre 2007.
Ce document est destiné a ceux qui n'ont pas de connexion internet ou qui préfèrent avoir une version papier. Mais si vous voulez avoir la dernière version en ligne et à jour rendez vous sur le site Spip.net.

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

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

Bonne lecture. Fabien.

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

Habillage graphique

Introduction aux feuilles de style

Les feuilles de style permettent de centraliser et de gérer de manière beaucoup plus aisée les indications graphiques que l'on insérait traditionnellement dans le HTML. Elles s'écrivent dans un langage spécifique : le CSS.

Comme vous le savez déjà, SPIP traite séparément le contenu de sa mise en page et son habillage graphique : les squelettes trient et affichent les contenus souhaités en pages HTML, dont l'habillage graphique est réalisé par des feuilles de style . Passez à la vitesse supérieure pour habiller vos squelettes : utilisez les feuilles de style avec SPIP !

Pourquoi les feuilles de style ?

Si vous réalisez des pages Web de manière « traditionnelle », les indications graphiques sont insérées directement dans le code HTML de votre page. Ainsi à chaque fois que vous voulez mettre un texte en rouge, vous écrivez <font color="red">. Pour afficher un tableau avec des bordures épaisses, vous écrivez <table border="2">.

Avec cette méthode et un site statique (où chaque article a une page HTML spécifique), changer la maquette de tout un site est un cauchemar : il faut rechercher dans tous les fichiers HTML, les portions de code à modifier, et effectuer ces modifications une par une (par exemple remplacer <font color="red"> par <b> si l'on décide que les éléments anciennement affichés en rouge seront désormais en gras).

Comme vous le savez déjà, SPIP améliore beaucoup la situation : vous n'avez plus à modifier des centaines de fichiers HTML, juste quelques squelettes ; et votre mise en page est remise à jour automatiquement sur l'ensemble du site.
Cependant le problème n'est pas entièrement résolu. Par exemple, mettons que vous ayez décidé d'employer un certain bleu pastel sur beaucoup d'élements du site, afin de donner une identité graphique à votre site : les liens, les encarts, certains éléments de navigation... sont affichés en bleu pastel. Le jour où vous voudrez remplacer ce bleu pastel par un vert pâle, vous devrez modifier tous les endroits du squelette où ce bleu apparaissait pour le remplacer par le vert pâle. Cela peut être décourageant : il n'est pas aisé dans ces conditions de changer rapidement le rendu des pages, ne serait-ce que pour faire des essais.

La solution réside dans l'utilisation des « **feuilles de style externes** ». Une feuille de style est un fichier où vous définissez un ensemble de propriétés graphiques, et les endroits où elles s'appliquent. On note deux avantages capitaux des feuilles de style :

- **la feuille de style est un fichier unique et centralisé**, que vous pouvez appliquer à autant de fichiers HTML (et de squelettes SPIP) que vous le désirez ;
- **les propriétés graphiques sont définies une seule fois dans la feuille de style**, quel que soit le nombre d'endroits où ces propriétés sont appliquées dans le HTML.

Concrètement

Pour être appliquée à un fichier HTML (qui peut être un squelette SPIP), la feuille de style doit être déclarée dans l'entête de votre page (entre les balises <head>), de la façon suivante :

<link rel="stylesheet" type="text/css" href="mes_styles.css">

- Ici le fichier mes_styles.css contient les propriétés graphiques que vous voulez appliquer à la page HTML (dans la suite de cette rubrique, on supposera que mes_styles.css est le nom que vous avez choisi pour ce fichier).

- Ce fichier porte l'extension « .css ». En effet, est le nom du langage utilisé pour les feuilles de style, de la même manière que HTML est le nom du langage utilisé pour la réalisation de pages Web.

Notez bien que le CSS n’est pas propre à SPIP, il s’agit d’un standard du Web [1].

> Note : une feuille de style peut s’appliquer aussi bien à une page HTML classique (« statique ») qu’à un squelette SPIP (« dynamique »). Cela veut dire que toute astuce CSS valable dans du HTML classique sera aussi utilisable dans un squelette de votre site.

Si vous avez bien lu les paragraphes précédents, vous serez peut-être dubitatifs : oui, il faut apprendre un nouveau langage pour utiliser les feuilles de styles (SPIP n’y est pour rien !). Les CSS n’utilisent pas, en effet, la syntaxe du HTML. Cependant ce langage est très simple, et il suffit de quelques exemples pour mettre le pied à l’étrier. De très nombreuses documentations existent sur ce sujet par ailleurs ; consultez les ressources proposées à cette page : « CSS : pour en savoir plus ».

Notes

[1] La première version de CSS a vu le jour en 1996. C’est un langage de feuille de style, approuvé comme *Recommandation du W3C*.

SPIP et les feuilles de style

Le code généré par SPIP est doté de certains styles qu'il convient de définir.

Dans l'article précédent, nous avons vu quels étaient les avantages des feuilles de style CSS. Voyons maintenant quel usage spécifique SPIP fait des feuilles de style.

Des styles définis par SPIP

Dans SPIP, certains styles jouent un rôle important : ils servent à modifier les propriétés graphiques des éléments qui ne sont *pas* définis dans *votre* HTML (celui de votre squelette), mais dans le code *généré par SPIP*. En effet, **SPIP associe de lui-même plusieurs styles au code qu'il génère**.

Ainsi, lorsque l'on utilise les raccourcis SPIP dans les articles (permettant par exemple de mettre en gras, en italique, créer des liens hypertextes, des intertitres, des tableaux, etc.), SPIP produit les balises HTML nécessaires à ces effets, chacune de ces balises étant alors dotée de sélecteur CSS.

> Par exemple, le raccourci suivant :
>
> Ceci est un [lien->http://www.uzine.net]
>
> est ainsi transformé en code HTML :
>
> Ceci est un lien

Quel est l'intérêt ? Ces balises portent un nom spécifique dans l'attribut class : ce nom définit à quelle « classe » ils appartiennent, c'est-à-dire un ensemble d'éléments HTML qui hériteront des mêmes propriétés graphiques définies dans la feuille de style.
Dans notre exemple, le code HTML est complété par l'appel à un style CSS intitulé « spip_out ». Le webmestre peut donc pousser la personnalisation graphique des liens sortants en modifiant la définition de ce style « spip_out » (couleur différente, fond coloré, police utilisée, etc.).

L'apparence de la plupart des raccourcis SPIP peut ainsi être paramétrée dans les feuilles de style. Cela vaut aussi pour les formulaires automatiques (répondre à un forum, signer une pétition...) et d'autres encore. Certains de ces styles sont très utiles, voire indispensables, d'autres seront réservés aux webmestres qui souhaitent obtenir des effets exotiques.

Où se trouvent ces définitions de style ?

Les propriétés graphiques appliquées aux pages HTML sont regroupées dans les fichiers .css qui accompagnent les squelettes. Depuis SPIP 1.8, SPIP 1.8.1, les squelettes et leurs feuilles de style sont regroupés dans le répertoire dist/. Dans les versions antérieures, ils se trouvent à la racine.

- Les définitions de style propres à SPIP se trouvent dans la « feuille de style externe » nommée spip_style.css. Celle-ci regroupe les définitions des styles associées au code généré par SPIP (passées en revue dans cette rubrique).
- Une autre feuille de style, spip_admin.css (disponible depuis SPIP 1.8, SPIP 1.8.1), permet de contrôler l'apparence des boutons d'administration (« recalculer cette page », etc.).

Vous pouvez les modifier (c'est même conseillé : « Mettez-y votre style ! »), mais notez bien que vous ne pouvez pas les renommer. **Ces styles sont indispensables et doivent nécessairement être définis pour le bon fonctionnement de vos squelettes**.

Historique : Dans les versions antérieures à SPIP 1.9, certaines définitions de styles (concernant les images ou les formulaires) ne sont pas disponibles dans les feuilles de style externes et ne sont donc difficilement personnalisables.

La gestion du cache

Le fait que les styles soient définis dans un fichier séparé a une conséquence importante. En effet, ce fichier, au contraire de vos squelettes n'est pas géré par SPIP (il n'en a pas besoin !). Cela signifie que **si vous modifiez une feuille de style, vous n'avez pas besoin de vider le cache de SPIP : il suffit de recharger la page dans votre navigateur.** Cela rend le réglage de la mise en page encore plus aisé.

Rappelons tout de même que votre feuille de style doit être déclarée dans vos fichiers HTML, et que ceux-ci doivent être recalculés une première fois pour que cette déclaration soit prise en compte.

Mettez-y votre style !

Vous pouvez modifier les styles fournis avec SPIP et ajouter les votres, en créant votre propre feuille de style. Voici comment.

Si vous connaissez le langage CSS (sinon lisez d'abord cet article : « Introduction aux feuilles de style »), vous pouvez très facilement modifier l'apparence de votre site, sans même avoir besoin de connaître le langage des boucles et balises de SPIP.

Créez votre feuille de style

Lors de l'installation de SPIP, les squelettes sont distribués avec plusieurs « feuilles de style externes » qui regroupent les indications produisant l'habillage graphique du site. Vous pouvez modifier ces fichiers et ajouter vos propres définitions de style, mais il est préférable de le faire dans votre propre fichier CSS afin de pas voir vos ajouts « écrasés » lorsque vous installerez une nouvelle version de SPIP.

> **Important :** ne travaillez jamais directement dans les fichiers fournis par défaut, sinon vous risqueriez de perdre toutes vos modifications à chaque mise à jour de SPIP ! Pour éviter cela, faites une copie des fichiers que vous souhaitez modifier.

1. Créez un fichier mes_styles.css (ou tout autre nom que vous avez décidé de lui donner) et rangez-le dans votre dossier « squelettes ». Vous copierez dans ce fichier les définitions de styles que vous souhaitez utiliser et modifier ; mais pour la suite de ce tutorial, nous allons considérer que vous partez d'une feuille vierge.

2. Appelez cette feuille de style dans l'entête de votre squelette, c'est-à-dire entre les balises <head> du fichier HTML (aux côtés du title et autres meta). De la façon suivante :

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

Bien souvent une seule feuille de style suffit pour l'habillage graphique d'un site, mais vous pouvez déclarer de cette façon autant de feuilles de style que nécessaire.

> *Historique :* À partir de SPIP 1.8.2, on utilise :

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

> Depuis SPIP 1.9, la balise #CHEMIN remplace et améliore #DOSSIER_SQUELETTE.

Respectez la « cascade »

Il est important de garder à l'esprit le fonctionnement « **en cascade** » du (*Cascading Style Sheets* signifie littéralement « feuilles de style en cascade ») : lorsque plusieurs définitions de style concernent un même élément, **est appliqué en priorité le style le plus proche de l'élément**. L'ordre dans lequel les styles sont « lus » a donc une importance.

- ***Feuilles de style externes***

Si vous utilisez plusieurs feuilles de style, notez que l'ordre dans lequel celles-ci sont appelées dans l'entête de la page a une importance. Si vous appelez *d'abord* mes_styles.css et *ensuite* spip_style.css : ce sont les

styles de cette dernière, plus *proches*, qui s'appliqueront prioritairement aux votres. Pensez donc à faire l'inverse :

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

- ***Styles définis dans le code HTML***

Si vous ne souhaitez pas toucher aux fichiers CSS, vous pouvez continuer à insérer, par endroits, des indications graphiques directement dans le code HTML de vos squelettes : en définissant quelques styles dans le head, et/ou en plaçant des indications de style directement dans les balises HTML de la page.

Les styles placés directement dans les balises, étant au plus *proches* des éléments concernés, seront prioritaires sur ceux définis dans le head, eux-mêmes prioritaires sur ceux des feuilles de style *externes*.

> *Historique :* Dans les versions antérieures à SPIP 1.9 certains styles sont définis en dur dans le code généré par SPIP, ce qui contrarie leur surcharge.

Des styles qui ont de la « class »

Comment fait-on alors pour changer, par exemple, l'apparence de tous les intertitres SPIP ? C'est très simple. Ouvrez votre fichier mes_styles.css dans un éditeur de texte et ajoutez-y la ligne suivante :

```
h3.spip { color: red; font-size: 18px; }
```

Rechargez la page : tous les intertitres apparaissent comme par magie en rouge ; remarquez de plus que les autres h3 de votre page, s'il y en a, ne sont *pas* affichés en rouge.

Expliquons brièvement la syntaxe de cette règle de mise en page :

- h3.spip juste avant les accolades signifie que la règle ne s'applique qu'aux <h3> dotés d'un attribut class égal à « spip ». Notez bien : ni les <h3> n'ayant pas cet attribut, ni les balises ayant cet attribut sans être des <h3>, ne seront concernés.

Si vous ajoutez vos propres styles, sachez que la valeur donnée à l'attribut class est totalement arbitraire. La seule chose qui compte, est que vous utilisiez bien le même nom dans le code HTML (class="toto") et dans votre feuille de style CSS (.toto { ... }).
Rappelons toutefois que vous ne pouvez pas renommer les class associées au code généré par SPIP (dont les définitions de style sont regroupées dans spip_style.css).

- Les accolades contiennent la liste des propriétés graphiques associées au style ainsi défini. Ici nous voyons que la couleur est réglée à rouge et que la police de caractères doit être affichée avec une taille de 18 pixels.

Notons que toutes les propriétés non définies dans cette liste garderont leur valeur habituelle pour la balise considérée ; dans le cas présent, le h3 générera toujours un texte en gras, car rien dans cette définition de style ne dit le contraire.

Une typographie personnalisée

Après une introduction générale sur les feuilles de style, nous allons maintenant passer en revue quelques-uns de leurs usages les plus courants sous SPIP.

Intéressons-nous ici aux styles créés lorsque des raccourcis typographiques sont insérés dans un texte entré sous SPIP. Que l'on affiche un article, une brève, une rubrique ou autre n'a aucune importance : les styles portent toujours les mêmes noms. Cela ne nous empêchera pas de découvrir comment y appliquer éventuellement des habillages graphiques distincts...

Le texte de base

Même le texte de base sous SPIP, celui que vous entrez dans un des formulaires de l'espace privé, en tapant au kilomètre, génère des tags HTML particuliers. En effet, il est découpé en paragraphes ; à chaque paragraphe correspond un tag <p class="spip">. Nous pouvons donc associer à ces paragraphes un style bien précis, qui ne sera pas appliqué au reste du texte de la page.

Commençons par choisir une police de caractères. Pour cela on utilise la propriété font-family, qui prend pour valeur un ou plusieurs noms de polices de caractères à utiliser. Pour un corps de texte comme celui d'un article il vaut mieux une fonte à empattements ; mettons que vous vous décidiez pour « Bookman Old Style ». Ajoutez donc à votre fichier mes_styles.css la règle suivante :

```
p.spip {
  font-family: "Bookman Old Style";
}
```

(notez qu'un nom de fonte en plusieurs mots doit être entouré de guillemets...) Un problème peut survenir si la police « Bookman Old Style » n'est pas installée sur l'ordinateur de vos visiteurs : chaque ordinateur a une configuration différente, et n'oubliez pas que les fontes « gratuites » de Microsoft sont rarement installées sur Linux et Macintosh... Il est préférable de prévoir ce cas et spécifier une ou plusieurs polices de remplacement successives :

```
p.spip {
  font-family: "Bookman Old Style", "Times New Roman", serif;
}
```

On spécifie ici comme remplaçantes successives de Bookman, la classique « Times New Roman », et en dernier recours « serif ». « serif » n'est pas une police en soi, c'est un code générique qui indique au navigateur de prendre la police à empattements par défaut de l'ordinateur ; de même « sans-serif » spécifie la police sans empattements par défaut (en général Arial ou Helvetica).

> On retiendra cette règle importante : dans la propriété font-family, il convient toujours de proposer plusieurs choix successifs pour s'adapter aux polices de caractères installées sur l'ordinateur du visiteur. Notons que cette règle est aussi valable pour le tag <font face="..."> du HTML traditionnel.

Bien évidemment d'autres propriétés sont à votre disposition. Vous pouvez par exemple régler la taille du texte avec la propriété font-size. Notez cependant que les navigateurs disposent d'un réglage pour configurer la taille du texte par défaut, et le texte principal de vos pages ne devrait pas outrepasser ce réglage, pour des raisons de confort visuel : c'est l'utilisateur qui choisit la taille de base, non le webmestre.

Notez bien que les styles que vous appliquez aux tags <p> s'appliquent à chaque paragraphe en tant qu'objet autonome. Cela autorise certains effets intéressants, comme par exemple d'indenter la première ligne des paragraphes en utilisant la propriété text-indent. Par défaut, cette propriété prend pour valeur zéro, c'est-à-dire qu'il n'y a pas d'indentation. On peut la modifier pour obtenir, sur chaque première ligne, un décalage de soixante pixels à droite :

```
p.spip {
  text-indent: 60px;
}
```

Appliquer un traitement différencié

Plutôt que de nous étendre sur la panoplie de styles générés automatiquement par les raccourcis typographiques de SPIP, et que vous pourrez habiller à votre guise (ils sont passés en revue dans les articles suivants de cette même rubrique), étudions ici le cas où vous voulez appliquer un habillage différent à un même style, selon sa position dans ce squelette. Ce besoin est légitime : on veut par exemple afficher le corps de texte dans une police à empattements avec indentation en début de paragraphe, mais le post-scriptum dans une police plus « lisse » (sans empattements) et plus petite, sans indentation.

Cette opération est en réalité très simple. Il faut d'abord modifier votre squelette afin d'introduire les éléments qui permettront de discriminer le texte et le post-scriptum. Cela prendra par exemple, à l'intérieur de la boucle ARTICLES principale, la forme suivante :

```
<div class="texte">#TEXTE</div>
<div class="ps">#PS</div>
```

Il faut ici recalculer la page (car on a modifié le HTML...). L'affichage du navigateur est toujours le même : normal, nos nouveaux styles ne faisant l'objet d'aucune règle dans la feuille de style, ils sont ignorés par le navigateur. Remédions-y :

```
.texte p.spip {
  font-family: "Times New Roman", serif;
  text-indent: 50px;
}
.ps p.spip {
  font-family: Tahoma, Arial, sans-serif;
  font-size: 90%;
}
```

La grande nouveauté ici ne réside pas dans les propriétés graphiques mais dans la façon dont on les applique au code HTML. En effet, « .texte p.spip » signifie : « cette règle s'applique à tous les tags <p class="spip"> qui sont contenus dans un tag ayant un attribut class égal à "texte" ». On pourrait restreindre un peu cette règle en spécifiant que le tag parent doit en plus être un tag (le début de la règle s'écrirait alors « div.texte p.spip ») ; mais comme nous maîtrisons la structure de nos propres squelettes, il n'est pas utile de rendre la feuille de style très restrictive.

Toujours est-il que cette feuille de style a le résultat voulu : les paragraphes du corps de texte s'affichent avec une indentation, ceux du post-scriptum dans une fonte plus petite et sans indentation. Pour vérifier que ces règles s'appliquent bien à chacun des paragraphes <p class="spip"> et non au englobant, on peut s'amuser à définir un cadre noir :

```
.texte p.spip {
  border: 1px solid black;
}
```

On note que chaque paragraphe du corps de texte (mais pas du post-scriptum) est entouré de son propre cadre noir. Si on avait simplement écrit « .texte » au lieu de « .texte p.spip », c'est le texte tout entier qui serait entouré d'un unique cadre noir englobant. Remarquons en passant l'apparition de la propriété border...

> Note : cette astuce, consistant à tracer un cadre de couleur pour savoir à quels éléments s'applique précisément une règle, peut être très utile quand votre feuille de style s'enrichit. N'hésitez pas à l'utiliser si vous commencez à perdre pied...

Cette méthode est très puissante et se généralise avec profit pour la structuration de votre mise en page.

Styles des raccourcis typographiques de SPIP

Voici les styles CSS associés aux raccourcis typographiques les plus courants et les plus simples de SPIP.

Pour faciliter la mise en forme des textes, SPIP propose un certain nombre de « raccourcis SPIP » qui font l'objet d'un traitement automatisé : ils sont remplacés par le code HTML correspondant, doté d'un style (généralement la classe spip). Pour affiner encore davantage la mise en forme des textes, vous pouvez modifier les styles CSS associés à ces raccourcis typographiques.

Les paragraphes de SPIP

Pour créer des paragraphes dans SPIP, il suffit de laisser une ligne vide. Les paragraphes ainsi générés par SPIP sont dotés de la classe spip :

```
<p class="spip">Voici le texte de mon paragraphe</p>
```

Pour ajuster, par exemple, l'espace entre chaque paragraphe, modifiez la définition de style : p.spip.

> *Remarque :* ceci n'est valable que pour les textes en plusieurs paragraphes ; les textes écrits d'un seul bloc, en un seul paragraphe, n'étant pas dotés par SPIP de balises <p>. Il est donc préférable de ne pas définir ce style.

Gras et italique

Le gras et l'italique sont générés par les raccourcis suivants :

```
Du texte {en italique}, du texte {{en gras}}
```

Leur apparence est contrôlée par les définitions de style respectives : i.spip et strong.spip. Ces styles sont peu utiles, et ne sont d'ailleurs pas définis par défaut.

> *Historique :* dans les versions antèrieures à SPIP 1.8, SPIP 1.8.1, le texte en gras est entouré de la balise <b> et donc stylé par la définition b.spip.

Les intertitres de SPIP

Les intertitres sont créés par le raccourci suivant :

```
{{{Un intertitre}}}
```

Leur apparence est contrôlée par la définition de style h3.spip. Ce style est sans doute l'un des plus importants, car il permet de définir la taille, la police et le positionnement des intertitres dans les articles : vous serez certainement amenés à le modifier en fonction de vos choix graphiques et typographiques.

Notez en particulier les attributs margin et padding qui permettent d'agir sur l'espacement de l'intertitre avec les paragraphes précédent et suivant. Sans ce réglage, l'intertitre serait soit trop « collé » au reste du texte, soit trop espacé.

Le trait de séparation horizontal de SPIP

Le trait de séparation horizontal est généré par ce raccourci SPIP :

```
----
```

Son apparence est contrôlée par la définition de style : hr.spip.

> *Remarque :* La ligne horizontale (<hr />) est un élément délicat à styler, car les styles CSS qui lui sont appliqués sont interprétés differemment selon les navigateurs.

Les notes de bas de page

Les notes de bas de page, sont créées par le raccourci :

Le texte[[Ceci est une note de bas de page.]].

Leur apparence est contrôlée par plusieurs définitions de style :

- a.spip_note permet de différencier visuellement les appels des notes de bas de page des autres liens (c'est-à-dire leurs numéros, tant dans le corps du texte qu'en bas de page) ;
- p.spip_note contrôle l'affichage des notes elles-mêmes ; bien souvent inutile (on préférera en effet définir ce style plus simplement, à partir du style de la balise parente.).

Styles des liens hypertextes

Comment définir l'apparence de ses liens ?

Sur le web, les liens hypertextes sont traditionnellement caractérisés par la couleur bleue et le soulignement. Mais vous avez certainement remarqué que cette présentation varie d'un site à l'autre. Vous pouvez très facilement personnaliser l'apparence de vos liens avec les styles CSS. Exemple très classique :

```
a {
  color: green;
  text-decoration: none;
}
a:hover {
  color: red;
  text-decoration: underline;
}
```

- « a » concerne tous les liens affichés sur votre page web, c'est-à-dire toutes les balises <a>, sans exception, qu'elles aient ou non un attribut class. Dans l'exemple ci-dessus, les liens s'afficheront en vert, sans « décoration » : ils ne sont pas soulignés.

- Plusieurs états sont disponibles pour les liens avec les « pseudo classes » (:hover, :visited, etc.). Ainsi « a:hover » concerne les liens « survolés ». Dans l'exemple ci-dessus, les liens deviennent rouges et soulignés lorsqu'ils sont survolés par le pointeur de la souris.

Notez que la recommandation CSS 2 précise que, pour être pleinement prise en compte, la règle a:hover doit être placée après les autres.

Les liens hypertextes de SPIP

SPIP 1.2, SPIP 1.2.1 va plus loin, en vous permettant de différencier graphiquement les différents types de liens, notamment les liens internes au site et les liens vers d'autres sites. Cela se fait avec quelques définitions de style spécifiques, que nous vous recommandons de personnaliser à votre goût :

- a.spip_in concerne les liens à l'intérieur de votre propre site. Par exemple :

Le raccourci [->article1177] génére un lien interne, vers l'article 1177 de votre site, ainsi : SPIP et les feuilles de style

- a.spip_out concerne les liens vers l'extérieur de votre site. Par exemple :

Le raccourci [uZine->http://www.uzine.net] affiche le lien externe suivant : uZine

- a.spip_url traite les adresses URL transformées en lien hypertexte. Par exemple :

Le raccourci [->http://www.uzine.net] affiche directement l'URL, avec un lien hypertexte vers cette adresse, ainsi : http://www.uzine.net

- a.spip_glossaire concerne les liens vers le glossaire externe (en l'occurrence l'encyclopédie en ligne Wikipédia). Par exemple :

Le raccourci [?SPIP] génére le lien suivant : SPIP

Pour ne pas s'emmêler les liens

Notez bien que, pour un lien donné, plusieurs définitions de style interviennent. Par exemple, si vous avez précisé :

```
a {
  color: green;
  text-decoration: underline;
}
a.spip_in {
  color: orange;
}
```

Les liens internes (dotés de la classe spip_in) sont de couleur orange, mais héritent également du soulignement appliqué à <a>. Les autres liens, y compris ceux qui ne sont pas générés par SPIP, s'afficheront en vert souligné.

> On note ici une propriété fondamentale des feuilles de style : les règles graphiques s'appliquent dans l'ordre allant de la plus générique à la plus spécifique. Cela permet de spécifier un comportement général pour la plupart des éléments, et de modifier ce comportement pour un plus petit sous-ensemble d'éléments. Cette caractéristique fait toute la puissance des feuilles de style.

Exposer le lien activé

Depuis SPIP 1.7.1, on peut mettre en évidence, dans une liste de liens, celui concernant la page où l'on se trouve. Le style que l'on utilisera pour ce faire est : .on. Voir : « Exposer un article dans une liste ».

Styles des citations dans SPIP

Citations d'auteurs, de code informatique ou de poésie, SPIP facilite l'insertion de citations dans vos textes. *Code is poetry !*

Les citations de SPIP

Pour simplifier l'insertion de citations dans vos textes, SPIP 1.7, SPIP 1.7.2 propose le raccourci suivant :

<quote>Ceci est une citation.</quote>

ce qui donne :

> Ceci est une citation.

Son apparence est contrôlée par la définition de style blockquote.spip.

Signalons l'existence d'une autre balise HTML, que vous pouvez également utiliser dans vos textes : à la différence de celle utilisée dans le raccourci de SPIP (<blockquote>) qui « va à la ligne », <q> permet de faire une citation au fil du texte.

Poésie

SPIP 1.7, SPIP 1.7.2 propose de plus une mise en forme particulière pour la poèsie, avec le raccourci suivant :

<poesie>Voici de la poésie.</poesie>

ce qui donne :

> Voici de la poésie.

Son apparence est contrôlée par la définition de style div.spip_poesie.

Le code informatique affiché dans SPIP

Pour citer du code informatique au fil du texte, on utilise le raccourci SPIP suivant :

<code>Du code dans le texte</code>

Le style associé est .spip_code.

Introduit dans SPIP 1.3, le très astucieux raccourci <cadre>...</cadre> permet de présenter du code informatique dans un bloc de formulaire, ce qui facilite le copier-coller du code. Son apparence est contrôlée par la définition de style : .spip_cadre.

Ces deux derniers raccourcis, et leurs définitions de style correspondantes, sont peu utilisés, sauf dans le cas d'une documentation technique (comme celle-ci) où l'on doit citer des morceaux de code informatique, des noms de fichiers ou de répertoires.

Styles des logos, images et documents

Les styles CSS associés par SPIP aux logos, aux images et autres documents permettent de contrôler leur affichage dans la page.

Style des logos

Les logos des éléments (rubriques, articles, brèves, auteurs, sites) sont systématiquement dotés du style .spip_logos, placé sur la balise HTML <img>. Ce style peut être très utile pour, par exemple, déterminer la position du logo.

Ainsi pour aligner à droite les logos placés dans le cartouche, par exemple, tout en ménageant un espace entre l'image et le texte, vous ferez simplement :

```
.cartouche .spip_logos {
  float: right;
  margin-left: 1em;
}
```

Styles des images et documents

Depuis SPIP 1.8, SPIP 1.8.1, l'insertion avec les raccourcis <docXX|left> et <imgXX|right>, de documents et images dans le corps de texte d'un article (ou d'une brève), est contrôlée par des styles CSS.

Trois définitions de style permettent de personnaliser l'affichage des documents, de leurs titres et légendes :

- .spip_documents concerne la boîte qui contient la vignette et les informations du document (<docXX|left>) ou l'image insérée sans titre ni descriptif (<imgXX|right>) ;
- .spip_doc_titre contrôle l'affichage du titre du document ;
- .spip_doc_descriptif contrôle l'affichage du descriptif du document.

Trois styles sont utilisés en complément, indispensables pour définir le positionnement du document ou de l'image dans la page :

- .spip_documents_center quand le document est centré (<docXX|center>) ;
- .spip_documents_left quand le document est aligné à gauche (<docXX|left>) ;
- .spip_documents_right quand le document est aligné à droite (<docXX|right>).

Notez bien que ces styles concernent de la même façon les images (<imgXX>) et les documents (<docXX>).

NB : Avant de modifier radicalement ces définitions de style, notez que celles-ci sont également utilisées, dans les squelettes par défaut (depuis SPIP 1.9), pour styler les éléments du portfolio et des listes de documents joints (aux articles et/ou rubriques).

Images avec ou sans bordure

Depuis SPIP 1.9 il est possible d'appliquer une bordure aux images. Par exemple : .spip_documents img { border: 1px solid grey; } encadrera vos images d'une fine bordure grise.

Inversement, pour ne pas voir vos logos encadrés de cette horrible bordure bleue qui signale traditionnellement les « images cliquables » sur certains navigateurs (comme FireFox), n'oubliez pas de régler leur bordure à zéro : .spip_logos { border: 0; }.

Styles des tableaux de SPIP

Les styles CSS permettent de paramétrer finement l'affichage des tableaux générés par SPIP.

Les tableaux constituent un moyen facile de présenter l'information en rangées et colonnes de cellules. Pour tirer profit de cet article, mieux vaut connaître la syntaxe HTML des tableaux. La balise <table>, qui permet d'insérer un tableau dans la page, est l'une des balises les plus utilisées en HTML.

Les tableaux sont créés dans SPIP de la façon suivante :

```
||Légende du Tableau|Résumé du Tableau||
| {{Nom}} | {{Date de naissance}} | {{Ville}} |
| Jacques | 5/10/1970 | Paris |
| Claire | 12/2/1975 | Belfort |
| Martin | 1/31/1957 | Nice |
| Marie | 23/12/1948 | Perpignan |
```

ce qui produit cet affichage :

Légende du Tableau

Nom	Date de naissance	Ville
Jacques	5/10/1970	Paris
Claire	12/2/1975	Belfort
Martin	1/31/1957	Nice
Marie	23/12/1948	Perpignan

Les styles permettent de paramétrer finement l'affichage des tableaux :

- table.spip permet de modifier le comportement général du tableau, notamment ses dimensions, ses et sa position (calé à gauche, centré, etc.) ;
- table.spip caption concerne la légende (optionnelle) du tableau ;
- table.spip tr.row_first concerne la « première ligne » du tableau (ici en jaune). Pour que la première ligne soit prise en compte comme rangée de titres, il faut que chacun des éléments qu'elle contient soient en gras ;
- table.spip tr.row_odd et table.spip tr.row_even pour les autres lignes. Un des intérêts de ces styles est la possibilité d'appliquer deux couleurs différentes via « row_odd » et « row_even » (ici, gris clair et gris foncé), permettant d'alterner les couleurs d'une ligne à l'autre, ce qui facilite agréablement la lecture du tableau.
- table.spip th et table.spip td concernent les cellules du tableau, et permettent, par exemple, de contrôler leur espacement intérieur (padding), afin d'aérer la présentation.

Ils sont beaux, mes formulaires !

Vous avez personnalisé la mise en page et la typographie de votre site, mais maintenant ce sont les formulaires SPIP qui jurent totalement sur le reste ! Pas de panique, là aussi les feuilles de style remédient au problème.

Différents formulaires sont utilisés dans le site public, pour le moteur de recherche interne, la rédaction des messages des forums, les inscriptions à l'espace privé, etc. Il en va pour ces formulaires SPIP comme pour le reste : leur aspect graphique peut être modifié via CSS afin de s'insérer sans hiatus dans votre design.

Pour tirer profit de cet article, mieux vaut connaître les balises HTML propres aux formulaires.

À chaque formulaire son style

Tous les formulaires SPIP utilisés dans le site public sont contenus dans une div dotée du même style, .formulaire_spip, qui permet d'appliquer facilement une modification devant concerner l'ensemble de ces formulaires.

Par exemple, pour mettre en gras tous les descriptifs des champs de saisie (<label>) de vos formulaires, vous stylerez ainsi :

```
.formulaire_spip label {
  font-weight: bold;
}
```

Depuis SPIP 1.9, chaque formulaire est, de plus, doté d'un style qui lui est propre. Celui-ci permet, inversemment, de modifier l'apparence de certains formulaires en particulier, sans interférer sur les autres. Chacun de ces styles est nommé de même que la balise appelant le formulaire et son squelette. Par exemple, le fichier HTML du formulaire de recherche est formulaire_recherche.html ; celui-ci est inséré dans les squelettes grâce à la balise #FORMULAIRE_RECHERCHE ; et le style qui lui est associé est donc .formulaire_recherche.
Depuis SPIP 1.9.2, les squelettes des formulaires sont déplacés et ainsi renommés : dist/formulaires/recherche.html.

Styles des champs de saisie

Le style .forml est appliqué aux champs de saisie des formulaires. Il permet de définir la couleur du fond et la largeur des champs de saisie, mais aussi leur taille et police de caractère. Par exemple :

```
.forml {
  width: 99%;
  padding: 1px;
  border: 1px solid #666;
  font-family: Verdana;
  font-size: 11px;
}
```

Ce style est particulièrement utile pour homogénéiser l'ensemble des champs de saisie, quelques soient les balises qui le reçoivent, telles que <input> et <textarea>, toutes deux présentes dans le formulaire de forum.

Depuis SPIP 1.9, chaque champ de saisie est étiqueté d'un terme explicatif, enveloppé dans une balise HTML <label>. On modifiera l'apparence de ces étiquettes par cette définition de style : .formulaire_spip label.

Les styles CSS permettent non seulement de changer les couleurs et les polices de caractères, mais aussi de gérer le positionnement relatif des objets dans la page. Ils permettent même de définir précisément la disposition des éléments entre eux (par exemple : <input>, <textarea> et <label>), sans utiliser de <table>

pour la mise en page.

Des boutons à vos couleurs

Une nouvelle qui ravira les débutant-e-s en CSS : on peut changer la couleur des champs mais aussi des boutons des formulaires [1]. Le style .spip_bouton est celui utilisé pour les boutons des formulaires SPIP.

Par exemple, pour que les boutons aient un fond bleu clair, et une bordure suffisemment visible (épaisse, en relief et bleue foncée), modifiez la règle suivante dans votre feuille de style :

```
.spip_bouton {
  background-color: #b0d0FF;
  border: 2px outset #000060;
  color: black;
}
```

Depuis SPIP 1.7, SPIP 1.7.2 , le formulaire de forum propose une barre de raccourcis typographiques. L'apparence de celle-ci est contrôlée par le style .spip_barre. Ainsi, pour modifier l'apparence des icônes qui la composent, et, par exemple, distinguer celles-ci au survol de la souris, vous définirez les styles pour .spip_barre a img puis .spip_barre a:hover img.

Organiser ces éléments de façon visiblement logique

Depuis SPIP 1.9, les différents éléments d'un formulaire sont plus rigoureusement regroupés en « blocs logiques » grâce aux balises HTML dédiées <fieldset> et <legend>. Leur apparence est donc contrôlée par .formulaire_spip fieldset et .formulaire_spip legend, tout simplement (ceci était autrefois réalisé avec .spip_encadrer). Utile pour encadrer chaque partie d'une bordure, et aérer vos formulaires en espaçant ces blocs les uns des autres, par exemple.

Quelques styles complémentaires permettent d'affiner encore la présentation de vos formulaires :

- L'apparence des messages d'erreur et autres réponses renvoyées par les formulaires peut être personnalisée par le style .reponse_formulaire.
- Certains formulaires permettent de prévisualiser les informations saisies avant de les envoyer. Depuis SPIP 1.9, l'apparence de cette prévisualisation est contrôlée par le style .previsu.
- Enfin, ce dernier style n'est pas utilisé dans les formulaires, mais est lié au formulaire de recherche interne au site : .spip_surligne permet de mettre en évidence les termes recherchés dans les pages de résultats.

P.-S.

Cet article de Pompage.net introduit au style des formulaires : « CSS : on reprend tout à zéro ! (13ème épisode) ».

Notes

[1] Notons cependant que certains navigateurs imposent leurs propres boutons stylisés et ne vous laisseront pas en changer l'aspect.

CSS : pour en savoir plus

Le CSS n'est pas propre à SPIP. C'est un langage normalisé par le W3C et très utilisé sur le Web pour l'habillage graphique. Vous êtes libres de vous contenter du niveau abordé dans ces pages, mais nous vous invitons à consulter la très nombreuse documentation existant sur le CSS.

Citons quelques ressources intéressantes.

Ressources francophones sur le CSS

- **OpenWeb** est un site offrant à la fois un regard expert sur le Web et des exemples concrets d'utilisation sur les CSS et autres standards du Web.
- un cours « CSS débutant », de Mammouthland ;
- un excellent tutorial : **« CSS : on reprend tout à zéro ! »**, par pompage.net, qui permet de débuter en douceur et en toute beauté ;
- une collection de recettes pour une utilisation efficace des CSS, et un forum CSS et standards W3C pour s'entre-aider, chez Alsacreations ;
- un pense-bête, pour ne pas se perdre dans le feu de l'action ;
- et surtout, la traduction française de la documentation officielle, assez aride mais incontournable : les **spécifications CSS2** du W3C ;

En anglais

- Listes d'éléments : des exemples d'effets graphiques à foison ;
- de très bons tutoriaux sur le positionnement d'objets avec les CSS ;
- le W3C a ses propres *tips'n'tricks*...
- et pour ceux qui ont le goût du risque, l'intégrale des spécifications originales du W3C !

Petits outils

- Si vous utilisez l'excellent navigateur Firefox, des extensions telles que *CSS Viewer*, le plug-in *EditCSS*, ou la barre d'outils *Web Developer* vous rendront des services appréciables en matière de CSS.

Trucs et astuces

Classer selon la date ou selon un ordre imposé

Nous souhaitons trier les articles de façon différenciée, de cette façon :

- dans certaines rubriques, les articles sont publiés les uns à la suite des autres ; on veut donc les présenter selon l'ordre chronologique : les plus récents en début de liste, les plus anciens en fin de liste ;
- dans d'autres rubriques, on souhaite afficher les articles dans un ordre précis, en les numérotant ; sur le site public, on veut donc les présenter selon cet ordre indiqué par la numérotation ;
- enfin, dans les autres rubriques, les articles doivent être classés dans l'ordre chronologique, *sauf certains* que l'on souhaite placer en tête de liste dans un ordre précis.

Il y a plusieurs méthodes pour réaliser ce classement. Notez bien que les deux méthodes présentées ci-après sont **valables tant pour ordonner les articles que les autres objets de SPIP** (rubriques, brèves, etc.).

Méthode simple

Rappel : Pour classer les articles selon un ordre imposé, on numérote leurs titres dans l'espace privé, avec un numéro suivi d'un point et d'un espace : « 1. Premier article », « 2. Deuxième article », etc.
Dans les squelettes, pour ne pas afficher ces numéros sur le site public, on passe le filtre |supprimer_numero sur la balise #TITRE, et on utilise le critère {par num titre} sur la boucle ARTICLES pour ordonner selon les numéros indiqués.

Pour classer selon un ordre imposé ET/OU selon la date, on écrira donc ceci (à l'intérieur d'une boucle RUBRIQUES) :

```
<BOUCLE_articles(ARTICLES) {id_rubrique} {par num titre}{!par date}>
 [(#TITRE|supprimer_numero)]
</BOUCLE_articles>
```

Ainsi les articles de la rubrique seront tout d'abord classés par numéros, puis ceux portant un numéro identique seront classés par date inverse.

Remarque importante : Cela est tout à fait satisfaisant tant que, dans une rubrique donnée, tous les articles sont numérotés ou qu'aucun ne l'est. Par contre, si une rubrique contient des articles non numérotés, sauf un par exemple, celui-ci s'affichera en dernier. En effet, **non numérotés, les articles sont considérés comme ayant zéro pour numéro**. Pour éviter ce classement, on emploie alors la méthode suivante.

Méthode fine

À l'intérieur d'une boucle RUBRIQUES, nous allons effectuer le test suivant : est-ce qu'il existe, dans cette rubrique, au moins un article dont le titre commence par un numéro suivi d'un point ?

```
<BOUCLE_test_numero(ARTICLES) {id_rubrique} {titre==^[0-9]+\.} {0,1}>
 Il existe un article numéroté dans cette rubrique
</BOUCLE_test_numero>
 Il n'y a pas d'article numéroté
<//B_test_numero>
```

Le critère intéressant ici est : {titre==^[0-9]+\.}

Il s'agit d'une sélection sur le titre, selon une *expression régulière* (« == » indique une sélection selon une expression régulière) dont la syntaxe est : au début du titre (« ^ » indique le début de la chaîne testée), il y a un ou plusieurs (« + » indique « au moins un des caractères précédents ») caractères compris entre 0 et 9 (« [0-9] » signifie « caractères compris entre 0 et 9 inclus »), suivis du caractère « point » (« \. »).

Notez enfin qu'on ne sélectionne qu'un seul article ainsi numéroté ({0,1}) ; de cette façon, l'intérieur de la boucle ne sera effectué qu'une seule fois. De plus, il suffit qu'il existe un seul article numéroté pour provoquer l'affichage d'une liste par ordre « numéroté ».

Cette boucle affiche ainsi « Il existe un article numéroté dans cette rubrique » s'il y a au moins un article dont le titre est précédé d'un numéro dans la rubrique, et « Il n'y a pas d'article numéroté » sinon.

Il suffit maintenant d'installer à la place de ces mentions des boucles d'affichage des articles selon l'ordre de présentation désiré :

```
<ul>
<BOUCLE_test_numero(ARTICLES) {id_rubrique} {titre==^[0-9]+\.} {0,1}>
 <BOUCLE_ordre_numeros(ARTICLES) {id_rubrique} {par num titre}>
  <li>[(#TITRE|supprimer_numero)]</li>
 </BOUCLE_ordre_numeros>
</BOUCLE_test_numero>
 <BOUCLE_ordre_date(ARTICLES) {id_rubrique} {par date}{inverse}>
  <li>#TITRE<li>
 </BOUCLE_ordre_date>
<//B_test_numero>
</ul>
```

Ainsi les articles numérotés de la rubrique sont affichés en premier, classés par numéros, suivis de ceux ne portant pas de numéro, classés du plus récent au plus ancien.

Trier par ordre alphabétique, sauf un article qu'il faut afficher en premier

Mettons que vous voulez présenter une série d'articles par ordre alphabétique, sauf le prologue que vous voulez légitimement afficher au début de la liste...

Il existe une astuce pour « duper » le tri effectué par SPIP : il suffit d'ajouter un espace au début du titre de l'article (par exemple, transformer « Prologue » en « Prologue »). Le tri alphabétique pensera alors que ce titre doit « passer avant les autres », et l'affichera en premier. Cependant l'espace supplémentaire du titre sera ignoré par le navigateur Web, et ne provoquera pas de décalage disgrâcieux dans la mise en page.

Notez bien : l'astuce ne fonctionnera que si vous utilisez un classement alphabétique sur le titre des articles (c'est-à-dire le critère {par titre}). Tout autre classement ne fonctionnera pas.

Remarque : cela s'applique également aux rubriques, brèves, sites référencés, etc.

Plusieurs logos pour un article

Il est fréquent, pour rythmer la navigation sur son site, de vouloir utiliser des logos différents (notamment de tailles différentes) pour un même article en fonction de l'endroit où il apparaît.

Par exemple, utiliser un « gros » logo sur la page d'accueil du site qui permette de bien mettre en valeur l'article principal du moment, et un « petit » logo pour la navigation générale du site.

Rappelons que depuis SPIP 1.7, SPIP 1.7.2 , les webmestres disposent d'une fonction image_reduire qui permet de créer différentes versions (de différentes tailles) d'un *même* logo d'article. Mais il faut une autre méthode pour utiliser des logos *différents* pour un même article.

Les webmestres ont créé des méthodes personnelles basées sur l'utilisation différenciée du logo « normal » et du logo « pour survol ». Par exemple : le logo « normal » utilisé comme « petit logo » (appelé par la balise #LOGO_ARTICLE_NORMAL), et sur le sommaire, le logo « pour survol » (appelé par la balise #LOGO_ARTICLE_SURVOL) pour afficher la « grande » version du logo. Cette méthode complique souvent le code des squelettes, et interdit l'utilisation habituelle des logos « avec survol » que SPIP fournit automatiquement. Elle est de plus d'une souplesse très limitée.

Depuis SPIP 1.4 , il est possible de joindre des *documents* aux articles (et, accessoirement, aux rubriques). Nous allons expliquer ci-dessous comment utiliser ces documents joints pour créer plusieurs logos pour un même article.

Principe général

- Nous continuerons à utiliser les deux logos de l'article pour afficher les logos « normaux » (ceux qui apparaissent dans les liens de navigation les plus fréquents, par exemple sur les pages des rubriques), ce qui permet de conserver la simplicité de gestion des logos avec SPIP et la gestion automatique du survol (on revient à l'utilisation évidente de la balise #LOGO_ARTICLE, ou de #LOGO_ARTICLE_RUBRIQUE).

- Nous déciderons de joindre aux articles un document (généralement une image aux formats GIF, JPEG ou PNG) auquel nous donnerons systématiquement le même nom. Il nous suffira d'afficher ce document (en l'appelant par son nom) à la place du logo « normal » lorsque nous le désirerons.

- Cette méthode permet ainsi de créer autant de logos différents que nécessaire pour un même article (pas seulement un grand logo et un petit logo, mais pourquoi pas une image pixelisée avec un travail typographique élaboré pour afficher le titre, etc.).

- Nous verrons de plus que, grâce aux boucles de SPIP, on pourra très facilement dans les squelettes déterminer si un tel « grand » logo (document portant le nom choisi par nous) est présent, et agir en conséquence (afficher à la place le logo « normal », du texte spécifique, ou carrément un autre élément graphique).

- Miracle de la technologie moderne, des formats propriétaires et de l'accès par haut-débit, de telles versions spécifiques des logos, étant des documents joints, pourront être d'un autre format que des images. On pourra ainsi afficher, en tant que « grands » logos, des animations Flash ou Shockwave, des animations vidéo...

Mise en place des documents et choix des noms

- Nous décidons (arbitrairement, mais faites selon vos besoins) que les documents joints utilisés en tant que « gros » logo seront tous intitulés « spip_logo » ; ce document « spip_logo » sera affiché sur la page du sommaire de notre site à la place du logo normal.

Nous utiliserons d'autres noms dans la suite de cet exemple pour créer des effets plus fins : décidons immédiatement qu'ils auront tous des noms commençant par « spip_... ». (Cela nous permettra, dans l'affichage habituel des documents joints à un article d'exclure tous les documents dont le nom commence par « spip_... ». De cette façon, l'utilisation de documents en tant que logos alternatifs n'interfèrera pas avec l'affichage, par exemple, d'un portfolio.)

- Sur un article publié en ligne (de façon à pouvoir bidouiller nos squelettes en les testant), nous installons simplement :

  - un logo normal ; nous pouvons, si nous le voulons, installer une version de l'image « pour survol » pour la gestion automatique du changement d'image lors du survol avec la souris ;

Le logo « normal »

Le logo est installé classiquement. A priori, il s'agit d'une image de taille modeste.

- un « document joint » (par le pavé « JOINDRE UN DOCUMENT ») ; pour faire simple, installons une image (JPEG, GIF, PNG) ; une fois ce document installé (« uploadé »), nous lui donnons pour titre « spip_logo ». Voilà, c'est la seule manip nécessaire... SPIP affiche ce document en bas de la page de l'article dans l'espace privé, en donnant son titre (« spip_logo ») et en indiquant ses dimensions en pixels.

Le document « spip_logo »

Le seul impératif est de donner à ce document le titre « spip_logo ». Il est inutile d'installer une vignette de prévisualisation.

Dans le cas d'un document multimédia (Flash, Shockwave...), il faut indiquer à la main ses dimensions en pixels.

- Nous décidons de l'usage de ce document intitulé « spip_logo » : il sera affiché sur la page d'accueil du site à la place du logo normal du dernier article publié. De cette façon, la page de Une du site peut afficher une « grande » image pour mettre en valeur l'article en vedette.

Afficher « spip_logo » en Une du site

- Commençons par insérer une boucle toute simple pour afficher le dernier article publié sur le site et son logo « normal ». (Dans tous les exemples qui suivent, le code HTML est réduit à son strict minimum ; à vous d'enrober cela avec la mise-en-pages graphique qui vous convient.)

```
<BOUCLE_article_vedette(ARTICLES){doublons}{par date}{inverse}{0,1}>
 #LOGO_ARTICLE
 <h1>#TITRE</h1>
</BOUCLE_article_vedette>
```

Cette boucle très simple affiche le premier article ({0,1}) parmi tous les ARTICLES, sélectionnés par date de publication ({par date}) du plus récent au plus ancien ({inverse}). On affiche donc bien le dernier article publié sur le site. À l'intérieur de la boucle, on affiche le logo de l'article suivi du titre de l'article.

- Nous avons dit que nous voulions afficher, à la place du logo normal, le document joint à cet article dont le titre est « spip_logo ». Le code devient :

```
<BOUCLE_article_vedette(ARTICLES){doublons}{par date}{inverse}{0,1}>
 <BOUCLE_logo_article_vedette(DOCUMENTS){id_article}{titre=spip_logo}>
  [(#EMBED_DOCUMENT)]
 </BOUCLE_logo_article_vedette>

 <h1>#TITRE</h1>
</BOUCLE_article_vedette>
```

La BOUCLE_logo_article_vedette sélectionne parmi les documents joints à cet article ({id_article}) celui dont le titre est « spip_logo » ({titre=spip_logo}). À l'intérieur de la boucle, on demande l'affichage de ce document joint (#EMBED_DOCUMENT).

L'usage de #EMBED_DOCUMENT dans les squelettes permet d'insérer, via le système de boucles, directement le document à l'intérieur de la page. SPIP se charge de créer le code correspondant à des images ou à des fichiers multimédia.

– Inconvénient : si l'article n'a pas de document joint intitulé « spip_logo », le code précédent n'affiche que le titre. On va donc effectuer une nouvelle modification, qui permet d'afficher le logo « normal » de l'article s'il n'existe pas de document joint pour cet usage. *Notez bien :* une fois cette méthode comprise, il n'y aura plus d'autres subtilités pour réaliser tous les effets suivants...

```
<BOUCLE_article_vedette(ARTICLES){doublons}{par date}{inverse}{0,1}>
 <BOUCLE_logo_article_vedette(DOCUMENTS){id_article}{titre=spip_logo}>
  [(#EMBED_DOCUMENT)]
 </BOUCLE_logo_article_vedette>
  #LOGO_ARTICLE
 <//B_logo_article_vedette>
```

```
<h1>#TITRE</h1>
</BOUCLE_article_vedette>
```

Nous avons tout simplement ajouté l'appel au logo « normal » (#LOGO_ARTICLE) en texte alternatif (ce qui se trouve avant <//B...> d'une boucle s'affichant si la boucle ne fournit pas de résultat - ici, s'il n'y a pas de document joint à l'article portant le titre « spip_logo »).

Nous avons obtenu le résultat désiré :

- s'il existe un document joint associé à l'article auquel nous avons donné le titre « spip_logo », il est directement affiché ;
- *sinon*, c'est le logo « normal » qui est affiché.

Exclure ces documents spécifiques de l'affichage normal des documents joints

Dans le squelette des articles, on affiche les documents joints grâce à la BOUCLE_documents_joints, dont les critères essentiels sont :

<BOUCLE_documents_joints(DOCUMENTS){id_article}{mode=document}{doublons}>

On appelle les DOCUMENTS liés à cet article ({id_article}), qui sont bien des documents joints et non des images ({mode=document}) et qu'on n'a pas déjà affichés à l'intérieur du texte de l'article en utilisant le raccourci <EMBxx> ({doublons}).

Modifions ce code pour interdire l'affichage, dans cette boucle (qui est une sorte de « portfolio »), des documents dont le nom commence par « spip_... » (on ne veut pas afficher ici le « gros » logo utilisé en page de Une du site) :

```
<BOUCLE_documents_joints(DOCUMENTS){id_article}{titre!==^spip\_}{mode=document}{doublons}>
```

Le critère {titre!==^spip_} est une expression régulière, dont la syntaxe est très codifiée. On sélectionne les documents dont le titre n'est pas formé ainsi (le !== signifie « qui ne correspond pas à l'expression régulière ») : les premiers caractères (le symbole ^ indique le début de la chaîne de caractères) sont « spip » suivi de « _ » (dans la syntaxe des expressions régulières, « _ » indique le caractère « _ », de la même façon que « \. » indique le caractère « . »).

Ce critère sélectionne donc les documents joints dont le titre *ne commence pas* par « spip_ ».

L'exposé du principe général est terminé, vous avez largement de quoi vous amuser avec ça sur votre propre site. Les exemples suivants n'en sont que des variations.

Afficher toujours un gros logo en haut de page

Je décide, toujours de manière arbitraire, qu'il doit toujours y avoir une grosse image en haut de page de mes articles. Il s'agit d'un choix graphique de ma part : pour assurer l'unité graphique de mon site, j'affiche en haut de page une version de grand format liée à l'article (une variation du principe du « grand » logo) et, à défaut, une image stockée ailleurs sur mon site.

- Toujours le même principe : je joins à mon article un document dont je fixe le titre à « spip_haut ». (Pour éviter que ce document ne s'affiche dans le « portfolio » de la BOUCLE_documents_joints précédente, je fais commencer son titre par « spip_... ».)

Dans mon squelette des articles, j'affiche simplement en haut de page ce document :

```
<BOUCLE_doc_haut(DOCUMENTS){id_article}{titre=spip_haut}>
 #EMBED_DOCUMENT
</BOUCLE_doc_haut>
```

Comme dans l'exemple précédent, j'affiche le document, lié à l'article de cette page, et dont le titre est « spip_haut ». Fastoche.

- Comme dans le premier exemple, je pourrais décider d'afficher le logo de l'article si ce document n'existe pas :

```
<BOUCLE_doc_haut(DOCUMENTS){id_article}{titre=spip_haut}>
 #EMBED_DOCUMENT
</BOUCLE_doc_haut>
 #LOGO_ARTICLE
<//B_doc_haut>
```

- Mais ça n'est pas le résultat désiré. Je veux, pour des impératifs graphiques, toujours afficher une grande image aux dimensions prédéterminées.

Je vais donc (toujours un choix arbitraire de ma part) créer des images de substitution, utilisées « par défaut », au cas où un article n'aurait pas d'image en propre. Ces images répondent à mes impératifs graphiques (par exemple, elles ont toutes les mêmes dimensions que les documents que j'utilise d'habitude en « spip_haut »).

Sur mon site, je créée une rubrique pour accueillir « en vrac » ces documents de substitution. J'active les documents associés aux rubriques. (Je peux aussi créer un article qui accueillerait tous ces documents, et ainsi ne pas activer les documents joints aux rubriques. Le code serait à peine différent.) Admettons que cette rubrique porte le numéro 18 (c'est SPIP qui va fixer automatiquement le numéro de la rubrique lors de sa création). J'installe tous mes documents de substitution à l'intérieur de cette rubrique numéro 18. (Il est inutile de leur donner un titre.)

Pour appeler, au hasard, un document installé dans cette rubrique, il me suffit d'invoquer les critères suivants :

```
<BOUCLE_doc_substitution(DOCUMENTS){id_rubrique=18}{0,1}{par hasard}>
 #EMBED_DOCUMENT
</BOUCLE_doc_substitution>
```

(Notez bien : le critère {par hasard} ne signifie pas que l'image sera différente à chaque visite de la page, mais qu'elle sera sélectionnée au hasard à chaque recalcul de la page.)

On prendra soin, dans la navigation du site, d'interdire l'affichage de la rubrique 18, qui n'a pas besoin d'être présentée aux visiteurs. Le critère {id_rubrique!=18} *fera l'affaire.*

- Pour terminer la mise en place du dispositif, il nous suffit d'insérer cette boucle affichant un document de substitution pris au hasard dans la rubrique 18 en tant que texte alternatif à notre BOUCLE_doc_haut (à la place du #LOGO_ARTICLE) :

```
<BOUCLE_doc_haut(DOCUMENTS){id_article}{titre=spip_haut}>
 #EMBED_DOCUMENT
</BOUCLE_doc_haut>
 <BOUCLE_doc_substitution(DOCUMENTS){id_rubrique=18}{0,1}{par hasard}>
  #EMBED_DOCUMENT
 </BOUCLE_doc_substitution>
<//B_doc_haut>
```

Afficher un titre graphique

Toujours sur le même principe, nous allons afficher une version graphique du titre de l'article. Il s'agit d'une image réalisée avec un logiciel de dessin, où apparaît , avec une typographie particulièrement soignée (effets de relief, de dégradés de couleurs, avec des polices de caractère exotiques...) le titre de l'article.

- Décrétons qu'il s'agira d'un document joint associé à l'article, que nous titrerons « spip_titre ».
- Pour appeler ce document, à l'endroit où doit être affiché le #TITRE de l'article, installons la

boucle désormais connue :

```
<BOUCLE_doc_titre(DOCUMENTS){id_article}{titre=spip_titre}>
 #EMBED_DOCUMENT
</BOUCLE_doc_titre>
```

Notez à nouveau que cette méthode permet non seulement d'utiliser une image pour afficher le titre, mais aussi une animation Flash, un film... Dans ces cas, il vous faudra indiquer à la main pour votre document joint quelles sont ses dimensions en pixels.

- Complétons le dispositif : s'il n'existe pas de document joint portant le titre « spip_titre », il faut afficher en tant que texte HTML classique les informations nécessaires :

```
<BOUCLE_doc_titre(DOCUMENTS){id_article}{titre=spip_titre}>
 #EMBED_DOCUMENT
</BOUCLE_doc_titre>
 [<div>(#SURTITRE|majuscules)</div>]
 <div><font size=6>#TITRE</font></div>
 [<div>(#SOUSTITRE|majuscules)</div>]
 <br>[(#DATE|nom_jour)] [(#DATE|affdate)]</p>
<//B_doc_titre>
```

*** * ***

Signalons un dernier avantage à cette méthode...

Elle permet de faire par la suite encore évoluer radicalement l'interface graphique de votre site. Sans avoir à supprimer un par un tous les documents intitulés « spip_haut », « spip_titre »..., il vous suffit de créer de nouveaux squelettes qui ne les appellent tout simplement pas.

Par exemple, si les documents « spip_haut » étaient précédemment tous conçus pour une largeur de 450 pixels, et que la nouvelle interface graphique requiert des images d'une largeur de 600 pixels, vous n'aurez pas besoin de modifier un par un tous vos fichiers « spip_haut ». Il vous suffit, dans les squelettes, de ne plus faire appel aux documents intitulés « spip_haut », mais d'utiliser un nouveau nom (par exemple « spip_large ») et d'installer au fur et à mesure vos nouvelles versions de documents en les titrant « spip_large ». Pendant la transition, il n'y aura pas d'incohérences graphiques.

Les plus jetés d'entre vous peuvent même imaginer toutes sortes de tests sur le type de document ({extension=...}) ou sur leur taille ({largeur=...}, {hauteur=...}) (aucun PHP nécessaire) pour réaliser des interfaces selon ces tests (par exemple, une certaine interface graphique si « spip_haut » fait 450 pixels de largeur, et une autre s'il fait 600 pixels de largeur...).

Afficher les derniers articles de vos rédacteurs par rubrique

Par défaut SPIP vous propose une page auteur qui vous permet de montrer la liste des auteurs/rédacteurs participant à votre site, ainsi que leurs dernières contributions.

Mais un problème vient à se poser quand vous avez plusieurs rédacteurs et que ceux-ci participent activement à votre site. Cela finit par être une page à rallonge.

Cependant il existe un moyen de montrer les dernières contributions de vos auteurs/redacteurs et ce pour chacun d'eux.

Comment procéder ?

Tout d'abord, on va créer deux fichiers : un fichier myauteur.php3 et un fichier myauteur.html

Création du fichier myauteur.php3

Dans le fichier myauteur.php3 mettre le code suivant :

```
<?php

$fond = "myauteur";
$delais = 24*3600;

include ("inc-public.php3");

?>
```

Création du fichier myauteur.html

Dans le fichier myauteur.php3 mettre les codes suivants :

- Juste après la balise <body>, mettre

```
<BOUCLE_principale(AUTEURS){id_auteur}{unique}>
```

- Juste avant la balise </body>, mettre

```
</BOUCLE_principale>
```

- Dans le corps de la page HTML, voici le code à installer (on ne peut déterminer une rubrique car par défaut l'auteur n'est pas associé à une rubrique mais à un article, le code peut parait biscornu mais on va donc retrouver la rubrique par rapport à l'article) :

Code pour le dernier article

```
<B_appel_article>

Dernier article écrit par <BOUCLE_nom_auteur(AUTEURS){id_auteur}>[(#NOM)]</BOUCLE_nom_auteur><br>

<BOUCLE_appel_article(ARTICLES){id_auteur}>
 <BOUCLE_appel_rubrique_article(RUBRIQUES){id_rubrique}{par titre}{doublons}>
  [(#TITRE|majuscules)]
  <ul>
  <BOUCLE_rappel_article(ARTICLES){id_rubrique}{par date}{inverse}{doublons}{0,15}>
      <li><a href="#URL_ARTICLE">[(#TITRE)<br></a>]
  </BOUCLE_rappel_article>
  </ul>
 </BOUCLE_appel_rubrique_article>
</BOUCLE_appel_article>
</B_appel_article>

Cette auteur n'a pour l'instant écrit aucun article

<//B appel article>
```

Code pour article choisi au hasard

```
<B_appel_article>

Dernier article écrit par <BOUCLE_nom_auteur(AUTEURS){id_auteur}>[(#NOM)]</BOUCLE_nom_auteur><br>

<BOUCLE_appel_article(ARTICLES){id_auteur}>
 <BOUCLE_appel_rubrique_article(RUBRIQUES){id_rubrique}{par titre}{doublons}>
  [(#TITRE|majuscules)]
  <ul>
  <BOUCLE_rappel_article(ARTICLES){id_rubrique}{par hasard}{doublons}{0,15}>
      <li><a href="#URL_ARTICLE">[(#TITRE)<br></a>]
  </BOUCLE_rappel_article>
  </ul>
 </BOUCLE_appel_rubrique_article>
</BOUCLE_appel_article>
</B_appel_article>

Cette auteur n'a pour l'instant écrit aucun article

<//B appel article>
```

Et enfin

Maintenant, il faut configurer votre page auteur (page où vous énumérez vos différents auteurs) pour que, en cliquant sur le lien auteur, celui-ci, dirigera vers la page *myauteur* où sera inscrit les derniers articles écrits par l'auteur.

Le lien devra être écrit de la manière suivante :

nom du lien

Afficher des éléments par lignes dans un tableau

Soit à créer un tableau de trois colonnes contenant les titres des articles d'une rubrique, le nombre de lignes dépendant du nombre total d'articles ; sur le principe :

article 1	article 2	article 3
article 4	article 5	article 6
article 7	article 8	article 9

Il faut évidemment utiliser une boucle Articles, mais la difficulté réside dans le placement de balises tr ouvrante et fermante tous les trois passages dans la boucle. L'astuce consiste à utiliser la balise #COMPTEUR_BOUCLE et le filtre alterner. Cette balise indique le nombre de passages déjà effectués dans la boucle : si le reste de sa division par 3 vaut 0 il faut produire une balise tr ouvrante au début du code, s'il vaut 2, il faut produire une balise tr fermante à la fin.

Pour cela, point n'est besoin d'opérations arithmétiques dans le texte du squelette. Il suffit d'utiliser le filtre alterner, qui accepte un nombre quelconque d'arguments, disons *n*, le premier devant être un entier, appelons-le i. Ce filtre calcule le reste *k* de la division de *n-1* par *i* et retourne alors son *k*+unième argument. Il suffit donc de lui donner 3 arguments vides sauf un pour ne produire les balises tr qu'aux passages appropriés.

Si l'on veut un code HTML irréprochable, il faut produire une séquence de <td></td> après la boucle pour compléter la dernière ligne du tableau lorsque le nombre total d'articles n'est pas divisible par 3. Il suffit à nouveau d'utiliser le filtre alterner, mais appliqué cette fois à la balise #TOTAL_BOUCLE.

Ce qui donne :

```
<B_ligne>
<table>
<BOUCLE_ligne (ARTICLES) {id_rubrique} {par titre}>[
(#COMPTEUR_BOUCLE|alterner{'<tr>',","})]
 <td width="33%">
 <a href="#URL_ARTICLE">#TITRE</a>
 </td>[
(#COMPTEUR_BOUCLE|alterner{",",'</tr>'})
]</BOUCLE_ligne>
[(#TOTAL_BOUCLE|alterner{
    '<td></td><td></td></tr>','<td></td></tr>', "})]
</table>
</B_ligne>
```

Pour améliorer la lisibilité, on peut colorer différemment les lignes paires et impaires en mettant un attribut style dans la balise tr ouvrante. Il faut alors remplacer '<tr>' ci-dessus par une nouvelle utilisation de #COMPTEUR_BOUCLE|alterner, l'argument d'un filtre pouvant lui-même utiliser des balises et des filtres. Le squelette suivant donnera ainsi le tableau visualisé au début de cet article :

```
<B_ligne>
<table>
<BOUCLE_ligne (ARTICLES) {id_rubrique} {par titre}{0,7}>[
(#COMPTEUR_BOUCLE|
 alterner{[(#COMPTEUR_BOUCLE|
  alterner{'<tr style="background: #eee;">',
    '<tr style="background: #ddd;">'})],
     ","})]
 <td width="33%">
 <a href="#URL_ARTICLE">#TITRE</a>
 </td>[
(#COMPTEUR_BOUCLE|alterner{",",'</tr>'})
]</BOUCLE_ligne>
[(#TOTAL_BOUCLE|alterner{'<td></td><td></td></tr>','<td></td></tr>', "})]
</table>
</B_ligne>
```

Le même type de boucle, en remplaçant l'appel du titre par le logo (avec la balise #LOGO_ARTICLE), permet d'afficher une galerie où chaque logo d'article donne un aperçu (dont la taille sera de préférence fixée afin d'avoir une belle mise en page), et le texte de l'article contient la ou les œuvres exposées.

P.-S.

Pour obtenir ce résultat, la version initiale de cet article utilisait les boucles récursives de manière détournée. Ce détournement ne sera plus possible à terme dans SPIP, et est en outre moins efficace que la méthode ci-dessus qui n'effectue qu'une seule requête au serveur SQL.

Ne pas afficher les articles publiés depuis plus d'un an

Cela s'effectue avec le critère « *age* », qui est l'âge de l'article (calculé depuis sa date de mise en ligne dans l'espace public) en nombre de jours.

Ainsi pour conserver tous les articles de moins d'un an dans la rubrique courante, nous utiliserons le critère `{age < 365}` :

```
<B_articles_recents>
 <ul>
  <BOUCLE_articles_recents(ARTICLES) {id_rubrique} {age < 365} {par titre}>
  <li>#TITRE</li>
  </BOUCLE_articles_recents>
 </ul>
</B_articles_recents>
```

Pour prendre en compte l'âge vis-à-vis de la date de première publication au lieu de la date de mise en ligne, il faudra utiliser le critère « *age_redac* » au lieu de « *age* ». L'âge est indiqué en nombre de jours. Voir : « La gestion des dates ».

Notons que cette manipulation est possible avec tous les types de données auxquels est associée une date (brèves, forums...).

Présenter les résultats d’une recherche par secteurs

Il suffit d’inclure la boucle de recherche dans une boucle de type rubriques sélectionnant les rubriques de premier niveau ; dans la boucle de recherche, on ajoute alors le critère « *id_secteur* » pour se limiter au secteur courant.

```
<BOUCLE_secteurs(RUBRIQUES){racine}>

<B_recherche>
 <b>#TITRE</b>
 <ul>
<BOUCLE_recherche(ARTICLES){recherche}{id_secteur}{par points}{inverse}{0,5}>
 <li><a href="#URL_ARTICLE">#TITRE </a>
</BOUCLE_recherche>
 </ul><hr>
</B_recherche>

</BOUCLE_secteurs>
```

On remarquera que le titre du secteur n’est affiché que si la recherche a donné des résultats pour ce secteur. D’autre part, pour chaque secteur on n’affiche que les cinq articles les mieux classés, par ordre décroissant de pertinence.

Attention cependant, comme la recherche est effectuée autant de fois qu’il y a de secteurs, le calcul risque d’être ralenti.

Afficher le nombre de messages répondant à un article

Pour afficher le nombre de messages du forum lié à un article, nous devons créer une boucle de type FORUMS, liée à un article, de façon à compter son nombre de résultats.

À première vue, il est très simple de connaître le nombre d'éléments d'une boucle : il suffit d'utiliser la balise : #TOTAL_BOUCLE. Mais son usage est un poil acrobatique.

Bien sélectionner *tous* les messages

Première subtilité : nous voulons *tous* les messages des forums liés à l'article, en comptant les réponses aux messages, les réponses aux réponses...

Une simple boucle de type : <BOUCLE_forum(FORUMS) {id_article}> sélectionne uniquement les messages qui répondent directement à l'article. Pour accéder aux réponses à ces messages, on inclut habituellement une seconde boucle à l'intérieur de celle-ci, mais ce n'est pas ce que nous souhaitons faire ici.

Pour pouvoir tous les compter, nous voulons que la boucle sélectionne absolument tous les messages du forum lié à l'article, sans tenir compte de leur hiérarchie. Pour cela, il faut spécifier le critère {plat}, qui comme son nom l'indique sert à afficher un forum à plat. Ce qui donne : <BOUCLE_forum(FORUMS) {id_article} {plat}>

N'afficher *que* le total

Voyons maintenant comment compter les éléments qu'elle contient. La difficulté, ici, c'est que justement cette boucle ne doit rien afficher ! Elle n'affiche ni les messages ni leur titre, et on évitera même de lui faire afficher des espaces ou des retours à la ligne (sinon votre code HTML final contiendra des dizaines de lignes vides, inélégantes), en ne laissant pas même un espace entre les deux tags ouvrants et fermants de la boucle : <BOUCLE_forum(FORUMS) {id_article} {plat}></BOUCLE_forum>.

L'intérieur de la boucle n'affiche donc rigoureusement rien, mais on doit afficher, après la boucle, le nombre de résultats. La balise #TOTAL_BOUCLE peut s'utiliser non seulement à l'intérieur de la boucle, mais aussi (c'est la seule dans ce cas) dans le *texte conditionnel après* (le texte qui s'affiche après la boucle si elle contient des éléments) et le *texte conditionnel alternatif* (le texte qui s'affiche si la boucle est vide).

Une subtilité à bien comprendre : le texte conditionnel alternatif s'affiche *si la boucle ne trouve aucune réponse* ; il est donc affiché même si la boucle sélectionne des éléments (ici des messages de forum) mais qu'elle ne contient aucun affichage.

Nous devons donc placer #TOTAL_BOUCLE dans le texte conditionnel alternatif. S'il n'y aucun message de forum attaché à l'article, #TOTAL_BOUCLE sera vide, il ne faut donc pas afficher le texte englobant (« il y a N message(s) de forum ») dans ce cas.

```
<BOUCLE_combien(FORUMS) {id_article} {plat}></BOUCLE_combien>
[Il y a (#TOTAL_BOUCLE) message(s) de forum.]
<//B_combien>
```

Un message, des messages...

Nous affichons désormais le nombre de messages, s'il en existe. Mais nous voulons faire mieux : nous souhaitons que le terme « message » soit automatiquement accordé au singulier lorsque la boucle ne trouve qu'un seul résultat, et accordé au pluriel lorsqu'elle en trouve plusieurs.

Pour cela, nous allons utiliser les filtres de test |=={...} et |?{...,...} (introduits avec SPIP 1.8, SPIP 1.8.1) pour accorder « message » selon la valeur de #TOTAL_BOUCLE : [(#TOTAL_BOUCLE|=={1}|?{message,messages})]

Ce qui nous donne donc :

```
<BOUCLE_combien(FORUMS) {id_article} {plat}></BOUCLE_combien>
[Il y a (#TOTAL_BOUCLE) [(#TOTAL_BOUCLE|=={1}|?{message,messages})] de forum.]
<//B_combien>
```

Un menu déroulant pour présenter une liste d'articles

On souhaite réaliser un menu déroulant en utilisant les commandes HTML adaptées à la création de formulaire ; de plus on veut que ce menu serve à aller à l'URL de l'article sélectionné. Si l'URL des articles est du type `article.php3?id_article=123`, le bout de code suivant conviendra :

```
<FORM ACTION="article.php3" METHOD="get">

 <SELECT NAME="id_article">
  <BOUCLE_menu_articles(ARTICLES) {id_rubrique} {par titre}>
  <OPTION VALUE="#ID_ARTICLE">#TITRE</OPTION>
  </BOUCLE_menu_articles>
 </SELECT>

 <INPUT TYPE="submit" NAME="Valider" VALUE="Afficher l'article">
</FORM>
```

Les critères de la boucle articles (ici : les articles de la rubrique courante, triés par titre) seront modifiés selon vos besoins. Ce type de construction marche bien sûr aussi pour les brèves, rubriques...

Selon le même principe, il est tout aussi facile de présenter une liste de rubriques, de brèves... ou même l'intégralité de la structure du site.

Remplir les meta-tags HTML des pages d'article

Le but de cet exemple est d'installer dans les méta-tags de notre page, la liste des mots-clés associés à l'article ainsi que le nom des auteurs.

Si l'on veut optimiser le référencement du site par les moteurs de recherche, on peut par exemple mentionner le descriptif de l'article, les mots-clés associés, ainsi que le nom du ou des auteurs.

```
<head>
<BOUCLE_head(ARTICLES){id_article}>
<title>#TITRE</title>
<meta name="Description" content="#DESCRIPTIF">
<meta name="Keywords" content="<BOUCLE_keywords(MOTS){id_article}{","}>#TITRE
</BOUCLE_keywords>">
<meta name="Author" content="<BOUCLE_author(AUTEURS){id_article}{","}>#NOM
</BOUCLE_author>">
</BOUCLE_head>
</head>
```

On remarquera que pour les mots-clés et l'auteur, on utilise une boucle imbriquée pour aller chercher ces informations à partir de l'*id_article* courant. De plus, on spécifie une virgule comme séparateur afin que le contenu du meta-tag soit compréhensible (y compris par un moteur de recherche).

Attention ! le code donné ci-dessus à titre d'exemple est un peu « naïf » : si le #NOM d'un auteur ou le #DESCRIPTIF d'un article peuvent contenir des tags html (mise en italiques, saut de paragraphe...) la page qui en résultera sera en effet pleine d'erreurs. Pour éviter cela, il faut penser à passer un filtre comme |supprimer_tags sur le champ en question (remplacer #DESCRIPTIF par [(#DESCRIPTIF|supprimer_tags)])...

Guide des fonctions avancées

Au-delà du manuel de référence, vous trouverez ici une description détaillée des fonctions plus avancées à la disposition du webmestre.

Mutualisation du noyau SPIP

La mutualisation du noyau de SPIP est possible depuis SPIP 1.9. Il s'agit de pouvoir gérer plusieurs sites SPIP sur un seul serveur ou hébergement en n'utilisant qu'une seule fois les fichiers du noyau de SPIP.

Cela permet un gain d'espace disque important, ainsi qu'une possibilité de mise à jour de SPIP simple de l'ensemble des sites en ne mettant à jour que le noyau.

Les évolutions de SPIP 1.9.1 simplifiaient un peu la procédure, mais c'est avec SPIP 1.9.2 et ses améliorations [1] que la mutualisation devient plus robuste permettant la mise en place d'un partage du noyau de SPIP.

Cet article explique la procédure pour SPIP 1.9.2, sur des serveurs Apache [2] autorisant la réécriture d'url (url_rewriting).

Il y a plusieurs méthodes pour arriver aux mêmes résultats, selon que l'on souhaite configurer directement la mutualisation depuis un hébergement ou depuis un serveur.

Le concept...

Depuis SPIP 1.9.2 les dossiers nécessaires au fonctionnement du noyau SPIP (ecrire, dist, oo), et ceux marquant l'activité d'un site (config, IMG, tmp, local) sont clairement identifiables et séparés. C'est cette séparation qui permet d'avoir plusieurs sites SPIP autonomes pour un même noyau de SPIP.

Cette autonomie repose sur l'existence d'un quatuor de répertoires par site, dans lesquels Spip va écrire les données résultant de l'activité du site. Ces répertoires sont au nombre de quatre, car on distingue les données temporaires et permanentes d'une part, et les données accessibles par http et celles qui ne le sont pas d'autre part, d'où quatre types de données. Les deux répertoires inaccessibles par http seront protégées par un .htaccess installé automatiquement par Spip (les administrateurs du serveur pourront même mettre ces répertoires en dehors de l'arborescence servie par http).

Avant Spip 1.9.2, ces quatre types de données n'étaient pas clairement distingués, et se retrouvaient dans les répertoires IMG, CACHE, ecrire et ecrire/data. Avec Spip 1.9.2, ces quatre répertoires sont distincts et nommés par les constantes PHP suivantes :

```
define('_NOM_TEMPORAIRES_INACCESSIBLES', "tmp/");
define('_NOM_TEMPORAIRES_ACCESSIBLES', "local/");
define('_NOM_PERMANENTS_INACCESSIBLES', "config/");
define('_NOM_PERMANENTS_ACCESSIBLES', "IMG/");
```

Dans une installation de Spip non mutualisée, ces répertoires sont créés à la racine du site, lors de l'application de la fonction spip_initialisation sur les quatre valeurs ci-dessus. Pour obtenir la mutualisation des sources de Spip, il suffit de savoir associer une URL spécifique à son quatuor de répertoires spécifiques, en appelant la fonction spip_initialisation sur quatre autres valeurs, déduites de l'URL.

Créer les bons répertoires

Pour démarrer la mutualisation, il faut tout d'abord partir d'un site fonctionnel. Pour les exemples, nous partirons d'un site appelé par l'url http://example.org/ stocké physiquement dans /home/toto/public_html/.

Il faut créer un répertoire (par exemple nommé 'sites') qui va contenir les répertoires des sites mutualisés, à la racine de la distribution (au même niveau que /ecrire).

- mutualisation d'un répertoire :
  Si l'on souhaite que les adresses http://example.org/premier_site/ et http://example.org/deuxieme_site/ appellent chacun un site spip, il faut créer alors les sous-

répertoires /premier_site et /deuxieme_site dans le répertoire /sites, puis dans chacun d'eux créer les répertoires /config, /IMG, /tmp, /local. Ces quatre répertoires doivent être accessibles en écriture. Il sera possible par la suite d'ajouter un répertoire /squelettes aussi, à côté de ces répertoires.

- mutualisation sous-domaine et domaine (quelques idées) :
Si l'on souhaite que l'adresse http://toto.example.org/, http://exemple.tld/ et http://utilisateur.example.com/ appellent chacun un site spip, on peut envisager de créer dans /sites les répertoires /toto, /exemple.tld, et /example.com/utilisateur

Remarque : Toutes les url doivent pointer à la racine de la distribution SPIP, c'est-à-dire dans /home/toto/public_html/. C'est le rôle que vont remplir soit un fichier .htaccess soit la configuration du serveur Apache expliqués plus loin.

Des redirections bien faites !

Pour que SPIP reconnaisse qu'un site est mutualisé, il faut que le script spip.php (appelé par index.php) soit executé. Celui-ci cherchera, grâce à un code ajouté dans /config/mes_options.php si l'URL appelée correspond à un site mutualisé ou non. Pour cela, il faut qu'une adresse http://example.org/premier_site/ ou http://example.org/deuxieme_site/ soit redirigée vers http://example.org/ pour executer alors index.php...

C'est le rôle attribué au fichier .htaccess (ou directement dans la configuration du serveur Apache)

Il faut copier et renommer le fichier htaccess.txt à la racine de la distribution en .htaccess, puis le modifier :

Pour autoriser la réécriture d'URL (rien ne change normalement) : RewriteEngine On

Si la distribution SPIP est dans un sous-répertoire, modifier rewritebase. Ici, le site est à la racine donc : RewriteBase /

Enfin, dans réglages personnalisés, ajouter le code suivant pour que les répertoires /premier_site , /deuxieme_site et /troisieme_site soient traités depuis la racine de la distribution :

```
#Mutualisation
RewriteRule ^(premier_site|deuxieme_site|troisieme_site)$ /$1/ [R,L]
RewriteRule ^(premier_site|deuxieme_site|troisieme_site)/(.*) /$2 [QSA,L]
```

Le premier rewriterule redirige les adresses http://example.org/premier_site vers http://example.org/premier_site/. Le deuxième redirige tout ce qu'il y a derrière /premier_site/ à la racine, par exemple : http://example.org/premier_site/article112 est redirigé vers http://example.org/article112. (Cependant, cette redirection est transparente, le nom de l'url ne change pas, il est toujours http://example.org/premier_site/article112, même et surtout pour php !)

Et si SPIP est dans un sous répertoire ?

Dans ce cas là, les fichiers de spip se trouvent dans /home/toto/public_html/spip/ , SPIP est appelé par http://example.org/spip/, les sites mutualisés par http://example.org/spip/premier_site/ ou http://example.org/spip/deuxieme_site/.

Il faut alors mettre dans le .htaccess :

```
RewriteEngine On
RewriteBase /spip/

#Mutualisation
RewriteRule ^(premier_site|deuxieme_site|troisieme_site)$ /spip/$1/ [R,L]
RewriteRule ^(premier_site|deuxieme_site|troisieme_site)/(.*) /spip/$2 [QSA,L]
```

Et pour rediriger tous les répertoires comme des sites mutualisés ?

Il vous est possible **à la place** des rewriterules ci-dessus d'utiliser un code générique, utilisable quelque-soit le nom du répertoire du site mutualisé :

- à la racine

```
RewriteCond %{REQUEST_URI} !^/(config|dist|ecrire|IMG|oo|plugins|sites|squelettes|tmp|local)/(.*)
RewriteRule ^[^/]+/(.*) /$1 [QSA,L]
```

- ou dans un dossier /spip :

```
RewriteCond %{REQUEST_URI} !^/spip/(config|dist|ecrire|IMG|oo|plugins|sites|squelettes|tmp|local)/(.*)
RewriteRule ^[^/]+/(.*) /spip/$1 [QSA,L]
```

Et pour les domaines et sous domaines ?

Par un heureux hasard, il n'y a rien à faire ici puisqu'ils pointent normalement déjà vers la racine de la distribution SPIP...

Mutualiser selon l'URL grâce à mes_options.php

C'est le fichier /config/mes_options.php à la racine de la distribution qui va réaliser la majeure partie du travail : il doit chercher si une URL est à mutualiser ou non, et initialiser SPIP en fonction.

De nombreux cas peuvent se présenter, entre les répertoires, les domaines et sous domaines. PHP peut utiliser deux variables pour tester les URLs qui ont appelé le script :

```
$_SERVER['REQUEST_URI']; // contient tout ce qu'il y a derrière le nom de domaine : /premier_site/article112 par exemple...
$_SERVER['SERVER_NAME']; // contient le nom du domaine et sous domaine : utilisateur.example.org par exemple
```

Ce sont ces deux variables qui vont être comparées à une expression régulière pour extraire le nom du répertoire qui contient la mutualisation.

Un moyen simple de mutualiser tous les répertoires est de copier le code suivant :

```
<?php
if ( preg_match(',/([a-zA-Z0-9_-]+)/?,',$_SERVER['REQUEST_URI'],$r)) {

  if (is_dir($e = _DIR_RACINE . 'sites/' . $r[1]. '/')) {

     $cookie_prefix = $table_prefix = $r[1];

     define('_SPIP_PATH',
        $e . ':' .
        _DIR_RACINE .':' .
        _DIR_RACINE .'dist/:' .
        _DIR_RESTREINT);

     spip_initialisation(
        ($e . _NOM_PERMANENTS_INACCESSIBLES),
        ($e . _NOM_PERMANENTS_ACCESSIBLES),
        ($e . _NOM_TEMPORAIRES_INACCESSIBLES),
        ($e . _NOM_TEMPORAIRES_ACCESSIBLES)
        );

    $GLOBALS['dossier_squelettes'] = $e.'squelettes';

     if (is_readable($f = $e._NOM_PERMANENTS_INACCESSIBLES._NOM_CONFIG.'.php')) include($f);
  }
}
?>
```

La ligne preg_match récupère le nom d'un dossier dans l'arborescence de l'URL, par exemple 'premier_site' dans http://example.org/premier_site/ ... Puis le script tente une mutualisation si le répertoire /sites/premier_site/ existe.

Si SPIP est dans un répertoire /spip, il faut modifier la première ligne par : `if (preg_match(',/spip/([a-zA-Z0-9_-]+)/?,', $_SERVER['REQUEST_URI'], $r)) {`

Il faut dire dire à SPIP quel est le préfixe de table de base de données utilisé, ce que fait `$cookie_prefix = $table_prefix = $r[1];`
Cette ligne va créer des tables MySQL avec des préfixes ayant le nom du répertoire contenant le site : mon_site_article à la place de spip_article par exemple. Cela permet d'héberger tous les sites sur une seule et même base de données. Vous pouvez garder le préfixe par défaut (spip), mais il faut penser à avoir plusieurs bases de données différentes pour chaque site.
Pour cela, mettre à la place :

```
$cookie_prefix = $r[1];
$table_prefix='spip';
```

La fonction spip_initialisation admet quatre paramètres qui sont l'adresse de chacun des répertoires nécessaires au fonctionnement du site mutualisé.

`$GLOBALS['dossier_squelettes'] = $e.'squelettes';` définit l'emplacement du dossier squelettes du site

mutualisé.

Enfin les dernières lignes chargent un éventuel sites/premier_site/config/mes_options.php.

Information :

Toute modification du fichier config/mes_options.php du noyau SPIP affectera les options de tous les sites hébergés.

Par exemple, mettre dans ce fichier : $type_urls = 'propres' ;

Donnera par défaut à tous les sites ce type d'url... Mais chaque site peut le changer dans son propre /sites/mon_site/config/mes_options.php.

Configurer apache pour les domaines et sous-domaines

La mutualisation côté serveur, pour ce qui concerne la gestion des sous-domaines ou des domaines reste simple, mais nécessite de créer quelques redirections d'URL dans la configuration du serveur Apache pour tenir compte de ces sites.

Voici un exemple de configuration pour un serveur nommé 'exemple.tld' (ici un SPIP mutualisé) utilisant des sous-domaines (SPIP appelé par http://utilisateur.example.org/spip/).

Le noyau SPIP est dans '/home/toto/public_html/spip/'.

Il faut alors créer les répertoires /sites/exemple.tld/ , /sites/exemple.tld/utilisteur/.

Le fichier de configuration se situe dans Apache 2/linux dans /etc/apache2/sites_availables/default (vous pouvez aussi créer un nouveau fichier dans le dossier sites_availables et l'activer).

```
# SERVEUR exemple.tld
# SPIP par sous domaine...
<VirtualHost *>
      ServerName exemple.tld
       ServerAlias *.exemple.tld

      # Redirection vers le SPIP noyau
       DocumentRoot "/home/toto/public_html"
       <Directory "/home/toto/public_html/">
            AllowOverride All
            Order allow,deny
            Allow from all
       </Directory>

       # Seule l'adresse http://utilisateur.exemple.tld/spip/* doit être redirigée

       # (utilisateur.exemple.tld/spip/* -> /home/toto/public_html/*)
       RewriteCond %{SERVER_NAME} (www\.)?([^.]+)\.example\.net$
       RewriteRule ^/spip/(.*) /home/toto/public_html/spip/$1 [QSA,L]

       # (utilisateur.exemple.tld/* -> /home/toto/public_html/sites/exemple.tld/utilisateur/*)
       RewriteCond %{SERVER_NAME} (www\.)?([^.]+)\.example\.net$
       RewriteRule (.*) /home/toto/public_html/sites/exemple.tld/%1/$1 [QSA,L]

</VirtualHost>
```

Il est par contre nécessaire de tester ces mutualisations possibles dans /config/mes_options.php :

```
<?php
# pour utilisateur.exemple.tld/spip/
if ( preg_match(',(.*)\.exemple\.tld/spip/?,',$_SERVER['SERVER_NAME'].$_SERVER['REQUEST_URI'],$r)) {

  if (is_dir($e = _DIR_RACINE . 'sites/exemple.tld/' . $r[1]. '/')) {

    $cookie_prefix = $table_prefix = $r[1];

    define('_SPIP_PATH',
        $e . ':' .
        _DIR_RACINE .':' .
        _DIR_RACINE .'dist/:' .
        _DIR_RESTREINT);

     spip_initialisation(
        ($e . _NOM_PERMANENTS_INACCESSIBLES),
        ($e . _NOM_PERMANENTS_ACCESSIBLES),
        ($e . _NOM_TEMPORAIRES_INACCESSIBLES),
        ($e . _NOM_TEMPORAIRES_ACCESSIBLES)
```

```
        );

    $GLOBALS['dossier_squelettes'] = $e.'squelettes';

    if (is_readable($f = $e._NOM_PERMANENTS_INACCESSIBLES._NOM_CONFIG.'.php')) include($f);
  }
}
?>
```

Note sur les sauvegardes et les restaurations

Chaque site copie ses fichiers de sauvegardes dans le répertoire /sites/premier_site/tmp/dump (ou /sites/premier_site/tmp/upload/*login* pour les sauvegardes d'un administrateur restreint). Les restaurations se font par le même dossier.

Attention :

A l'heure actuelle, SPIP enregistre dans la base de données, pour ce qui concerne le dossier /IMG des sites mutualisés une adresse sites/premier_site/IMG/* au lieu de IMG/* comme pour un site SPIP seul.

Une restauration ne fonctionnera donc que si elle s'effectue dans le site qui a créé la sauvegarde.

Une astuce pour restaurer le site ailleurs consiste à éditer le fichier dump.xml et à remplacer toutes les occurrences (à l'exception de celles des déclarations dir_img et dir_logo dans l'ouverture de la balise spip) :

- sites/premier_site/IMG/
    - par (spip seul) : IMG/
    - ou (spip mutualisé) : sites/mon_nouveau_site/IMG/

Inversement, pour passer un SPIP seul dans un répertoire mutualisé, il faut remplacer toutes les occurrences de :

- IMG/
- par : sites/mon_site/IMG/

(Cette difficulté sera corrigée par le passage à SPIP 1.9.3.)

Notes

[1] Nouvelle distribution des répertoires pour clarifier la mutualisation, correction des problèmes de logins et de noms de sites qui se mélangeaient parfois entre les sites mutualisés, fonction d'initialisation d'un site à mutualiser plus claire

[2] Pour information : ces procédures ont été testées avec Apache 2.0.55 et PHP 5.1.2 sur serveur Ubuntu Dapper Drake ainsi que sur PHP 5.1.6 sur serveur Ubuntu Edgy Eft

Le validateur XML intégré

Depuis que le World Wide Web Consortium a lancé la Web Accessibility Initiative, les problématiques de validation XML et d'accessibilité du Web aux déficients visuels ont convergé. SPIP s'est très tôt intéressé aux problèmes d'accessibilité, avec la version 1.5 en 2002. En revanche, il a longtemps récusé XML eu égard à la rareté des pages conformes : l'abondance de pages HTML non XHTML fait que les navigateurs ont leur propre analyseur, et les langages basés sur XML, comme SVG, MathML et XQuery, ont connu un démarrage très lent. Toutefois, la convergence des problématiques d'accessibilité et de validation d'une part, et l'implémentation en natif de SVG dans plusieurs navigateurs de l'autre, a amené SPIP, dans sa version 1.8.1, à offrir une interface avec des outils de validation comme Tidy ou le validateur officiel du W3C. Ces outils, comme l'indiquent sans détour les pages d'accueil de leur site, souffrent de quelques limitations, révèlent des divergences dans leurs messages d'erreur, et ne sont pas installables par l'internaute moyen [1] Ils sont de plus inopérants face aux nouvelles technologies comme Ajax, qui modifient incrémentalement le contenu d'une page Web. Plus grave encore, les *Document Type Definition* du W3C ont elles aussi des limitations, y compris la DTD XHTML 1.0 dite *stricte* : alors que la spécification officielle interdit l'emboîtement des balises a, label ou form, la définition formelle de la grammaire décrite considère comme valides les constructions <label for='x'><label... ou <form action='.'> <form... par exemple [2].

Face à cette situation, SPIP 1.9.2 innove radicalement en proposant un *validateur extensible, intégré, incrémental et optionnel*, fondé sur le *Simple Analyzer for XML* fourni en standard par PHP. Cette pluralité d'aspects le destine autant aux webmestres qu'aux graphistes et aux développeurs d'extensions de SPIP avec évidemment des manipulations différentes.

Validation pour les webmestres

Pour les webmestres, mettre simplement dans le fichier mes_options.php l'instruction $xhtml = 'sax'; provoquera une remise en page du code HTML produit par un squelette, chaque ligne comportant une seule balise ouvrante et au plus un attribut, avec renfoncement de la marge gauche proportionnellement au nombre de balises ouvrantes non fermées. Dans le cas où la page se révèle invalide, ce travail sera finalement ignoré, et le texte initial sera envoyé au client HTTP. Dans les ceux cas, la mise en cache éventuelle a évidemment lieu après l'action du validateur, qui se contente donc ici d'une simple mise en page.

Quelques précisions en ce qui concerne le traitement des attributs. Leur valeur sera systématiquement entourée d'une apostrophe, sauf si elle contient une apostrophe auquel cas elle sera entourée de guillemets (et si elle contient aussi des guillemets, ils seront écrits "). En raison d'une erreur de conception de SAX [3], les entités XML seront préalablement converties dans le *charset* du site à l'exception de & < > et " qui sont correctement traitées par SAX (car c'est indispensable). La liste des entités et leur valeur seront déduites du DOCTYPE indiqué par le squelette ; à défaut SPIP prendra un ensemble prédéfini équivalent à la DTD du latin1 enrichie de €, œ et Œ définis dans la DTD des symboles spéciaux.

Validation pour les graphistes

Pour les créateurs de squelettes, le validateur est disponible à travers le débusqueur introduit en SPIP 1.8. Cet outil n'est visible que des administrateurs du site, car ils disposent des boutons d'administration lorsqu'ils visitent les pages de l'espace public. L'un de ces boutons se nomme *Analyse XML* et déclenche explicitement la demande de validation de la page visitée. Cette analyse est d'abord confiée à SAX qui, même en l'absence de DOCTYPE, repère :

- les mauvais emboîtements de balises ouvrantes et fermantes ;
- les attributs sans valeur ;
- les attributs sans délimiteur ;
- les attributs sans espace avant le suivant ;

- les entités XML mal écrites (notamment un & littéral, alors que XML impose d'écrire &).

À la première erreur rencontrée, le débusqueur de SPIP essaiera immédiatement de retrouver de quel squelette provient l'erreur (il y en a plusieurs en cas d'inclusion) et à quelle ligne, en donnant des liens vers les fichiers sources (cette détermination ne peut pas toujours aboutir, une marge d'erreur est à prendre en compte).

Si cette première analyse est correcte, cette nouvelle version de SPIP va plus loin en prenant en compte le DOCTYPE de la page. Il peut être de type PUBLIC ou SYSTEM, et dans le premier cas la DTD sera mise en cache pour accélérer les analyses ultérieures. Chaque DTD peut en inclure d'autres, qui seront chargées pareillement. Afin d'éviter les redondances de calcul tout en tenant compte du système d'inclusion conditionnelle des DTD, SPIP met également en cache une structure de données spécifique qu'il déduit de la lecture de toutes les définitions d'éléments, d'attributs et d'entités issus du DOCTYPE indiqué. A l'aide de cette structure, SPIP *valide* la page, autrement dit vérifie que :

- tous les noms d'attributs utilisés dans une balise sont acceptés par la DTD ;
- tous les attributs obligatoires d'une balise sont présents ;
- toutes les valeurs d'attributs sont conformes au *motif* indiqué éventuellement dans la DTD ;
- tous les attributs de type ID ont une valeur composée de lettres, chiffres, souligné, tiret et deux-points ;
- tous les attributs de type IDREF ou IDREFS référencent des ID effectivement présents dans la page ;
- toutes les entités XML utilisées sont définies dans la DTD ;
- tous les noms de balises utilisées sont définis dans la DTD ;
- toute balise non vide est autorisée à l'être par la DTD (par exemple un img non vide sera dénoncé) ;
- toute balise vide est autorisée à l'être par la DTD (par exemple un tr vide sera dénoncé) ;
- toute balise utilisée est fille d'une balise l'admettant comme telle dans la DTD ;
- toute balise devant apparaître avant une de ces sœurs apparaît effectivement ainsi (par exemple head avant body)
- toute balise devant apparaître un nombre fixe de fois apparait effectivement ainsi (par exemple title une unique fois dans head).

Si des erreurs sont détectées, le validateur affichera un tableau de toutes les erreurs, avec leur nombre d'occurrences, des liens vers les lignes incriminées et des suggestions de corrections déduites automatiquement des constructions autorisées par la DTD. En l'absence d'erreur, le débusqueur affichera le code selon la mise en page décrite précédemment.

Validation pour les développeurs

Les pages Web en accès restreint ont particulièrement besoin d'un validateur intégré pour être mises au point, puisque les outils de validation externes n'ont par définition pas accès à ces pages. En ce qui concerne les scripts, standards ou venant d'extension, de l'espace de rédaction de SPIP on leur applique la validateur XML en plaçant $GLOBALS['transformer_xml'] = 'valider_xml'; dans le fichier mes_options.php. L'espace privé de SPIP 1.9.2 a été rendu conforme XHTML 1.0 grâce à ce mécanisme. Affecter cette variable globale à 'indenter_xml' provoquera l'indentation du source HTML si celui-ci est conforme XML, sans chercher à le valider.

Il est également possible de lancer, avec la souris, l'analyse XML du résultat d'un script Ajax activé dans l'espace de rédaction. Un tel script ne retournant pas une page HTML complète mais seulement un fragment, le validateur intégré fabriquera une page avec le DOCTYPE courant, un en-tête réduit à la balise Title, et une balise Body composée de ce fragment. Une fenêtre s'ouvrira avec le résultat de l'analyse, comme pour une page normale, pendant que la fenêtre initiale recevra le résultat du script Ajax comme d'habitude. En raison d'une spécification du W3C inutilisable quant au modèle d'événements [4], ce lancement du validateur n'est pas provoqué par un bouton spécifique de la souris, mais par un clic pendant lequel une au moins des deux touches Alt ou Meta est enfoncée.

Par ailleurs, le validateur de SPIP peut être appliqué à n'importe page présente sur le Web. Tout site SPIP

installé à l'URL http://*u* contient la page http://*u*ecrire/valider_xml que les auteurs du site peuvent invoquer explicitement pour valider des pages non limitées à celle du site. Ce validateur ne s'applique cependant qu'à des documents XML ; en l'absence de DOCTYPE, la DTD XHTML1.0 transitionnal sera prise.

Comme le suggèrent les lignes ci-dessus, le validateur s'applique à n'importe quel DOCTYPE, y compris ceux référençant des DTD résidantes sur le site. Rendre accessibles des pages Web, c'est être plus rigoureux dans l'utilisation des attributs et des balises, on peut donc facilement se construire un outil de validation d'accessibilité en écrivant des DTD moins laxistes que la XHTML 1.0 dite stricte. Remplacer #IMPLIED par #REQUIRED dans les attributs jugés indispensables est immédiat. Forcer input, select, textarea et button a être exclusivement des filles de la balise label est à peine plus difficile. En outre, le validateur accepte n'importe quel motif (au sens de PCRE) comme type d'attribut, il sera alors appliqué à chaque occurrence de cet attribut dans la page analysée.

Enfin, il est possible de définir ses propres règles de validation associées à un attribut. Les types usuels ID, IDREF et IDREFS sont implémentés par les fonctions validerAttribut_ID, validerAttribut_IDREF, validerAttribut_IDREFS. Il suffit d'introduire de nouveau type S1 ... Sn dans la DTD, et de définir les fonctions associées dans le fichier mes_options pour provoquer des contrôles personnalisés. En fin d'analyse, la fonction surchargeable inc_valider_passe2 est appelée, afin d'effectuer les contrôles rétrospectifs (c'est ici que les attributs IDREF sont vérifiés comme référençant des attributs effectivement présents). Cette interface de programmation est encore un peu frustre et sera revisée après expérimentation. Mais elle permet dès à présent de mettre sur pied très rapidement de nouvelles spécifications d'accessibilité.

Notes

[1] Le validateur du W3C se présente sous la forme d'un archive de plus de 1Mo, comportant essentiellement un script CGI en Perl, nécessitant plusieurs modules de ce langage et son interface avec le serveur HTTP. En outre, l'analyse syntaxique et la validation ne sont pas effectuées par ce programme, mais déléguées à l'analyseur *onsgmls* programmé en C++ et nécessitant donc d'être compilé après chargement des sources du projet OpenSP de 1Mo également (certains systèmes d'exploitation incluent cependant déjà l'exécutable de 128Ko résultant). Cette difficulté explique sans doute la rareté de ses installations, qui du coup sont souvent saturées.

[2] Cette carence est justifiée par ce passage de la recommandation XHTML qui affirme : *The HTML 4 Strict DTD forbids the nesting of an 'a' element within another 'a' element to any descendant depth. It is not possible to spell out such prohibitions in XML.* Cette affirmation n'est etayée par aucune démonstration ou référence scientifique. Les théorèmes établissant les pouvoirs de l'analyse syntaxique automatique ont été énoncés et démontrés dès les années 1950, et contredisent une telle affirmation. W3C, go to school ?

[3] À la rencontre de ce qu'il considère comme un lexème, SAX appelle un *preneur d'événements*. Faute de considérer les attributs comme des lexèmes, il provoquera la même séquence d'appel pour les 3 textes suivants :
&eolig;&EOlig;<a ...
&eolig;<a title='&EOlig;' ...
<a title='&eolig;' href='&EOlig;' ....
En conséquence, le preneur d'événements *Entité XML* ne saura pas si l'entité provoquant son appel fait partie de l'élément de texte précédant la prochaine balise, ou si elle fait partie de l'un des attributs de cette balise, et lequel. L'ambiguïté ne peut même pas être levée à l'aide des numéros de ligne et de colonne (ou de caractère du flux), qui indiquent dans tous les cas l'emplacement précédant la balise.

[4] Alors que le système d'exploitation le plus répandu dans le monde suivait pour une fois sagement les leçons d'Unix et plus précisément de X-Window, en décrivant un bouton de souris par un masque de bits, le W3C a cru bon de produire une spécification incompatible et mal conçue, que ne suit pratiquement aucun navigateur.

Sécurité : SPIP et IIS

Sécurité par défaut de SPIP

Il existe deux dossiers « sensibles » dans SPIP, ce sont CACHE et ecrire/data. Le premier comporte tous les fichiers qu'utilise votre cache pour accélérer l'affichage des pages, il est donc moyennement sensible, mais le deuxième stocke les journaux d'activité de spip (les spip.log) et vous permet notamment de créer dump.xml, le fichier de sauvegarde de la base de données.

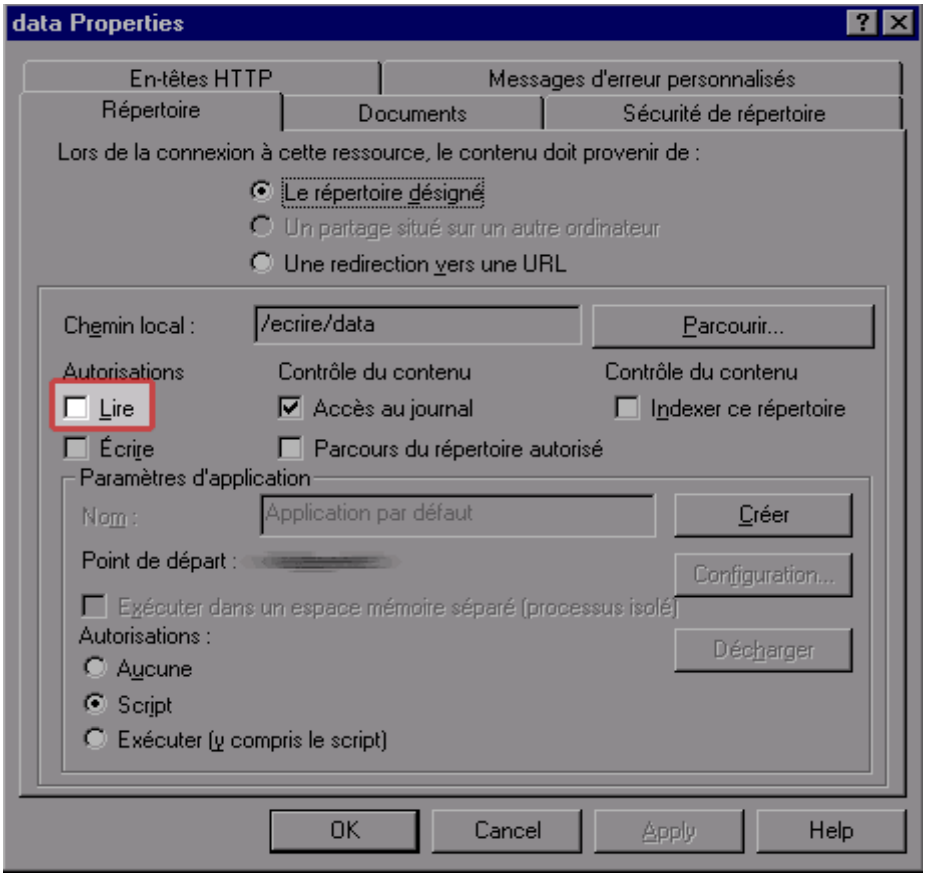
Or le fichier dump.xml contient des données très sensibles : en particulier on peut y voir *tous* les articles, même s'ils ne sont pas rendus publics sur le site, sans compter qu'il liste également les identifiants et les mots de passe [1] des rédacteurs et administrateurs du site.

La sécurité de tous ces fichiers est assurée traditionnellement par des fichiers de configuration d'accès nommés .htaccess. SPIP génère automatiquement ces fichiers pour empêcher l'accès aux données sensibles stockées sur le serveur : vous pouvez vérifier que CACHE et ecrire/data contiennent chacun un fichier .htaccess. Hélas, ces fichiers fonctionnent sous Apache (le serveur Web libre faisant tourner la majorité des sites Web de l'Internet) mais pas sous IIS (Internet Information Services, le serveur Web de Microsoft).

Protéger ses données sous IIS : une étape de plus

Si votre site SPIP est installé sur un IIS, n'importe qui peut donc voir les dossiers censément sécurisés via .htaccess : il faut donc les protéger.

Pour protéger un dossier sur votre site : allez dans le panneau d'administration de votre serveur Web, faites un clic droit sur le dossier concerné, cliquez sur « propriétés », et dans l'onglet « Répertoire » décochez la case « Lire ».



Le panneau de propriétés du dossier /ecrire/data/

Décocher la case "Lire" suffit à protéger le dossier exactement comme le fait Apache avec les fichiers .htaccess

Faites cette opération pour chacun des deux dossiers CACHE et ecrire/data. Si la manipulation est bonne, vous ne devriez plus pouvoir accéder aux fichiers de ces dossiers à travers le serveur web. Testez votre configuration en essayant d'afficher http://www.votresite.com/ecrire/data/spip.log avec votre navigateur. Vous devriez obtenir un message du type « Accès refusé ».

Notes

[1] Les mots de passe sont chiffrés par SPIP, mais gardez bien à l'esprit qu'aucune protection n'est inviolable.

Rapidité du site public

Contrairement à la plupart des systèmes de publication gratuits, SPIP intègre un système de cache permettant d'accélérer l'affichage du site public. Quelques pistes pour comprendre ce qui influe sur la rapidité de votre site...

Optimiser un site

Si vous vous inquiétez pour la rapidité de votre site, il est bon de vous intéresser aux pistes suivantes :

- Votre hébergement Web offre-t-il des performances de bonne qualité ? Evidemment, c'est subjectif. L'expression « mauvaise qualité » recouvre à coup sûr la plupart des hébergeurs gratuits (notamment Free). « Bonne qualité » inclut forcément une machine dédiée (i.e. qui ne sert qu'à votre site) de fabrication récente, mais aussi des hébergeurs commerciaux pas trop au rabais. Entre les deux, ça devient très subjectif, en fonction de vos exigences, de la taille de votre site....

- Si la qualité de votre hébergement laisse à désirer, vous aurez intérêt à ne pas créer de squelettes trop complexes, i.e. qui demandent à SPIP d'afficher trop d'informations différentes. Cela vaut pour tout type d'informations : tout ce qui, dans les squelettes, est susceptible d'être transformé par SPIP en données affichables. Notez, en particulier, que les squelettes fournis par défaut démontrent au maximum les possibilités de SPIP, et par conséquent génèrent des pages assez lourdes.

- N'oubliez pas non plus de régler les délais d'expiration des différents types de pages. Ainsi, si votre site contient un grand nombre d'articles en archives, vous avez peut-être intérêt à augmenter la durée d'expiration des articles, sinon les articles consultés peu souvent ne bénéficieraient pas du système de cache.

L'influence du cache

La présence du cache change quelque peu la donne en matière de rapidité. Ce n'est pas tant le nombre de visites de votre site qui sera le point critique, que la capacité de votre serveur à recalculer les pages dans le temps imparti au script PHP (en effet, sur la plupart des serveurs, une limite de durée d'exécution par appel de script est fixée afin d'éviter les abus et les erreurs de programmation). Par contre, si la page demandée est dans le cache et n'a pas expiré, la réponse du serveur devrait être quasi-instantanée (dans le cas contraire, votre serveur est vraiment très chargé).

La qualité des performances devient ainsi objectivement mesurable si, lors du recalcul d'une page du site, on obtient un « *timeout* », c'est-à-dire que le serveur a dépassé le temps maximal d'exécution d'un script PHP. Alors il faut soit changer d'hébergement, soit se résoudre à afficher des pages plus simples : pour cela, modifier les squelettes pour afficher moins d'informations sur une même page.

Sur une machine dédiée

Si vous utilisez votre propre machine, il faut vous assurer qu'elle pourra tenir la charge. N'importe quelle machine pas trop vieille (moins de trois ans environ) devrait en être capable.

Par contre, l'utilisation de SPIP, par rapport à d'autres systèmes de publication, permet de mutualiser les ressources techniques entre plusieurs sites. En effet, tant que le cache est utilisé, la machine est peu sollicitée, donc plusieurs sites peuvent cohabiter sans problème (sauf s'il y a vraiment un très grand nombre de visites). Le problème est donc surtout de prévenir qu'il y ait trop de passagers à bord, c'est-à-dire qu'un trop grand nombre de « services » hébergés (sites Web, boîtes à e-mail...) mette en péril la qualité du service.

<INCLURE> d'autres squelettes

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.

Syntaxe d'inclusion

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

Depuis SPIP 1.4 , on peut appeler un squelette depuis un autre squelette grâce à la balise <INCLURE> (on peut aussi utiliser <INCLUDE>, qui est identique). Sa syntaxe générale est :

<INCLURE(fichier.php3){paramètre}...>

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 SPIP et HTML de ce « pied de page » dans un fichier « pied.html », squelette lui-même appelé par le fichier « pied.php3 » (toujours selon le principe des couples de fichiers destinés à appeler des squelettes). Il suffit d'ajouter la ligne suivante, à l'endroit voulu, dans chacun des squelettes voulant afficher le bas de page :

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

Depuis SPIP 1.8.2 la distribution inclut un fichier page.php3 qui permet d'appeler tout squelette en passant en paramètre le « fond ». Ainsi on pourra s'éviter le fichier pied.php3 et remplacer avantageusement l'appel ci-dessus par :

```
<INCLURE(page.php3){fond=pied}>
```

Cette syntaxe est encore simplifiée depuis SPIP 1.9, puisqu'on ne précise plus désormais que le nom du squelette à inclure, sous la forme :

```
<INCLURE{fond=pied}>
```

Inclusions dépendant du contexte

Certaines inclusions peuvent dépendre du contexte. Par exemple, imaginons un squelette « hierarchie », qui affiche le chemin menant à une rubrique depuis la racine du site ; on appelerait cette page par une URL de la forme : « spip.php?page=hierarchie&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 placera le code suivant à l'intérieur de cette boucle :

```
<INCLURE{fond=hierarchie}{id_rubrique}>
```

Note : dans ce cas, le squelette hierarchie.html commencera certainement par une boucle rubriques avec le critère {id_rubrique}...

On peut imaginer que, dans certains squelettes, on désire récupérer non pas la hiérarchie en fonction d'une rubrique « variable » (au gré du contexte, par exemple le paramètre passé dans l'URL), 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{fond=hierarchie}{id_rubrique=5}>
```

N.B. Il est possible d'indiquer plusieurs paramètres dans la balise <INCLURE> ; cependant ce cas est très rare en pratique. Evitez d'ajouter des paramètres inutiles, qui rendront le cache moins efficace et votre site plus lent.

N.B. Le fichier inclus étant lui-même un squelette, il disposera donc de sa propre valeur de $delais [1]. Cela peut s'avérer pratique pour séparer des éléments lourds du site, que l'on recalculera peu souvent, et quelques éléments dynamiques nécessitant une mise à jour fréquente (par exemple, syndication).

Dans un contexte multilingue

Si le multilinguisme de SPIP est activé, depuis SPIP 1.7, SPIP 1.7.2 , il est possible de définir la langue de l'environnement d'un squelette inclus en utilisant le paramètre {lang}.

- S'il n'y a pas de paramètre de langue utilisé, c'est-à-dire sous la forme <INCLURE{fond=pied}>, le squelette inclus est appelé en utilisant la langue par défaut du site ;

- <INCLURE{fond=pied}{lang=es}> appelle le squelette en espagnol. Bien sûr, vous pouvez remplacer « es » par le code ISO de la langue souhaitée : *en* pour l'anglais, *fr* pour le français, *vi* pour le vietnamien, etc. (voir : « Internationaliser les squelettes ») ;

- <INCLURE{fond=pied}{lang}> appelle le squelette dans la langue courante du contexte d'inclusion.

Il convient de noter que cela rend possible l'utilisation des codes de fichiers de langue dans les squelettes inclus (voir : « Internationaliser les squelettes »).

Les squelettes inclus supportent les mêmes mécanismes de sélection par langue que les squelettes de « premier niveau ». En d'autres termes les squelettes inclus (ici pied.html) peuvent être déterminés par rapport à une certaine langue (pied.es.html, par exemple) de la même manière que tout autre squelette. Encore une fois, voir « Internationaliser les squelettes » pour plus de détails.

Notes

[1] Rappelons que la variable $delais définit la périodicité de mise à jour du cache. Voir la section « Le fichier .php3 » dans « Principe général - version archivée ».

Réaliser un site multilingue

La modification la plus importante qu'apporte SPIP 1.7 est sa gestion « naturelle » des sites multilingues. La totalité de cet article de documentation concerne SPIP 1.7.

Préalable : qu'est-ce qu'un site multilingue ?

Il n'est pas question dans cet article de rédiger un tutoriel complet sur les sites multilingues : d'une part, il y a certainement plusieurs « visions » de ce qu'on appelle « multilinguisme » ; d'autre part, nous manquons toujours de recul pour pouvoir définir « la meilleure méthode ».

Vous trouverez donc ci-dessous une revue des différents outils que SPIP propose pour gérer des sites multilingues ; à vous de les utiliser, et discutons-en dans les espaces prévus à cet effet (wiki, listes de discussion, etc.)

Mais avant de lire, oubliez un peu votre projet du jour, et pensez aux situations suivantes :

- un site de poésies, classées par thèmes (rubriques) ;
- un site de documentation pour, par exemple, un logiciel comme SPIP ;
- un site institutionnel en 25 langues ;
- un site corporate bilingue ;
- le site d'une association bulgare avec quelques pages en anglais.

Le site de poésie choisira plutôt ses langues article par article ; la documentation de SPIP, pour sa part, les ventile par « secteurs » (rubriques de premier niveau), et affiche les traductions disponibles pour chaque article, quand ces traductions sont disponibles. Le site institutionnel en 25 langues ne pourra sans doute pas fournir les 25 traductions en même temps, mais cherchera tout de même à conserver des arborescences parallèles ; le site corporate bilingue aura obligatoirement une traduction en face de chacun des articles, et une arborescence de rubriques en deux parties, la partie anglaise étant « clonée » sur la partie en auvergnat ; l'association bulgare affectera l'anglais à un secteur particulier de son site, le reste des secteurs étant tous en bulgare (par défaut).

Principe

Pour pouvoir autoriser ces situations différentes, et d'autres encore, le modèle mis en place dans SPIP consiste à déterminer une langue pour chaque article, chaque rubrique et chaque brève. Dans l'espace public comme dans l'espace privé, cette langue détermine le mode de correction typographique qui est appliqué aux textes ; dans l'espace public cela détermine également la langue des élements insérés par SPIP autour de ces « objets » : dates et formulaires principalement.

Pour créer un site multilingue avec SPIP, il faut d'abord configurer le site en conséquence : dans la configuration du site, à la section « langues » bien entendu. Là vous pourrez activer la gestion du multilingue, et choisir les langues que vous utiliserez sur votre site.

Configurer l'espace privé

Pour gérer plus facilement le site, on peut choisir dans la configuration du site avec quelle précision s'effectuera le réglage des langues, ce qui permet de masquer l'interface là où elle n'est pas nécessaire et de limiter les risques d'erreurs [1]. SPIP propose trois niveaux d'interface différents pour choisir les langues affectées aux articles (et brèves, etc.) ; par ordre croissant de complexité :

- **Par secteur** (rubrique de premier niveau) : à chaque secteur du site correspond une langue modifiable par les administrateurs, qui concerne toutes ses sous-rubriques ainsi que les articles et les brèves qui y sont publiés ; **ce réglage devrait satisfaire les besoins de la plupart des sites**

multilingues tout en conservant une structure et une interface simples.

- **Par rubrique** : de manière plus fine, avec ce réglage, on peut changer la langue pour chacune des rubriques du site, pas seulement celles de premier niveau.

- **Par article** : la langue peut être modifiée au niveau de chaque article ; ce choix est compatible avec les précédents (on peut par exemple choisir la langue par rubrique mais appliquer des exceptions de-ci de-là à certains articles) et permet toutes les finesses imaginables, mais attention à ne pas produire un site à la structure incompréhensible...

Blocs multilingues

[SPIP 1.7.2] Certains objets, comme les auteurs ou les mots-clés, peuvent s'orthographier différemment selon qu'ils sont affectés à un article dans une langue ou dans une autre. Cependant, il serait absurde de concevoir des « traduction d'un mot-clé » ou « traduction d'un auteur », car c'est bien le même auteur qui signe les deux articles, ou le même mot-clé (même « concept ») qu'on leur attache. Ces objets n'ont donc pas de langue au sens de SPIP, mais il est tout de même possible, à l'aide des « blocs multi », de les faire s'afficher dans la langue du contexte dans lequel ils sont invoqués (pour le dire plus simplement : faire que le mot-clé « Irak » s'affiche « Iraq » quand il est affecté à un article en anglais).

Le « bloc multi » est un raccourci SPIP, dont la structure est relativement intuitive : <multi>chaîne 1 [xx] chaîne 2 [yy] chaîne 3 ...</multi>

Pour reprendre l'exemple du mot-clé, son titre serait entré sous cette forme dans l'espace privé :

```
<multi>[fr]Irak [en]Iraq</multi>
```

Si un bloc multi est appelé à s'afficher dans une langue qui n'est pas prévue, c'est toujours la première partie du bloc qui s'affiche (« chaîne 1 » dans le premier exemple, « Irak » dans le second). Cela, afin de ne jamais avoir d'affichage vide [2].

NB : les blocs multi peuvent aussi être utilisés, selon la même structure, dans les squelettes, *cf.* Internationaliser les squelettes.

Boucles et balises : comment faire

Une fois l'espace privé réglé aux petits oignons, passons maintenant au site public. Hé oui, même si chaque article dispose maintenant de sa propre langue judicieusement choisie (selon le mécanisme expliqué plus haut), les squelettes doivent bien pouvoir en tenir compte dans l'affichage du site.

1. Une bonne nouvelle pour commencer : **le multilinguisme des squelettes est pour la plus grande part totalement naturel ; il n'est pas nécessaire de faire des squelettes différents pour afficher des articles de langues différentes. Un même squelette adapte *automatiquement* son affichage à la lange courante.**

Ainsi tous les éléments affichés autour et dans un article d'une langue donnée, seront affichés dans cette langue. Cela concerne aussi bien la date de publication de l'article que les formulaires de réponse au forum, de signature d'une pétition, etc. Plus généralement : toute balise SPIP incluse dans une boucle ARTICLES sera affichée dans la langue de l'article (de même pour les rubriques et les brèves).

Exemple : si votre page d'accueil contient un sommaire affichant les dix derniers articles publiés ainsi que leur date de publication, la date des articles en vietnamien s'affichera en vietnamien, celle des articles en créole de la Réunion s'afficheront en créole de la Réunion, etc.

> *Note : ce fonctionnement suppose que la langue de l'article fait l'objet d'une traduction dans SPIP. Ainsi, si un article est écrit en volapück mais que votre version de SPIP n'est pas encore traduite en volapück (nous vous invitons bien évidemment à corriger cette lacune en participant à l'effort de traduction), la date de l'article s'affichera en toutes lettres certes, mais dans une langue par défaut - le français probablement.*

2. Le sens de l'écriture

Si votre site contient des langues s'écrivant de gauche à droite (la plupart des langues) mais aussi des

langues s'écrivant de droite à gauche (notamment l'arabe, l'hébreu ou le farsi), il faudra de petits compléments au code HTML pour que l'affichage se fasse sans accroc [3].

SPIP offre à cet effet une balise spécifique : #LANG_DIR, qui définit le sens d'écriture de la langue courante. Cette balise est utilisable comme valeur de l'attribut dir dans la plupart des tags HTML (cela donne donc « ltr » pour les langues s'écrivant de gauche à droite, et « rtl » pour les autres [4]).

Une boucle d'affichage du sommaire devient donc :

```
<ul>
<BOUCLE_sommaire(ARTICLES){par date}{inverse}{0,10}>
  <li dir="#LANG_DIR">[(#DATE|affdate)]:
  <a href="#URL_ARTICLE">#TITRE</a></li>
</BOUCLE_sommaire>
</ul>
```

Si la mise en page repose sur des éléments alignés à droite ou à gauche, ceux-ci devront être inversés pour les langues écrites de la droite vers la gauche : on peut tout de suite penser à remplacer *tous* [5] les éléments du squelette marqués left ou right par les balises #LANG_LEFT et #LANG_RIGHT. Pour ce qui est de la définition de la page elle-même, il est alors judicieux de commencer par indiquer la langue de l'élément demandé, et la direction générale de la page :

```
<html dir="#LANG_DIR" lang="#LANG">
<head>
...
</head>
<body>
...
</body>
</html>
```

3. Les liens de traduction

SPIP propose un système de traduction entre articles : on peut spécifier quelles sont les différentes traductions d'un article (note : ces traductions sont elles-mêmes des articles à part entière). Le critère {traduction} permet alors, dans une boucle ARTICLES, de récupérer toutes les versions d'un même article.

Par exemple, pour afficher toutes les traductions de l'article courant :

```
<BOUCLE_traductions(ARTICLES){traduction}{exclus}>
[<a href="#URL_ARTICLE" rel="alternate" hreflang="#LANG">(#LANG|traduire_nom_langue)</a>]
</BOUCLE_traductions>
```

Notons le critère {exclus}, qui permet de ne pas afficher la version courante, et le filtre |traduire_nom_langue qui fournit le nom véritable de la langue à partir de son code informatique (cela permet d'afficher « français » au lieu de « fr », « English » au lieu de « en », etc.).

- Un critère complémentaire {origine_traduction} (pour les plus acharnés) permet de sélectionner uniquement la « version originale » de l'article courant.

Une page du wiki de spip-contrib rassemble des exemples de boucles utilisant ces critères : http://www.spip-contrib.net/spikini...

4. Éléments supplémentaires

[SPIP 1.7.2] introduit d'autres éléments permettant de fabriquer des sites multilingues :

— le critère {lang_select} sert à forcer la sélection de la langue pour la boucle (AUTEURS), qui normalement ne le fait pas (à l'inverse, le critère {lang_select=non} permet de dire aux boucles (ARTICLES), (RUBRIQUES) ou (BREVES) de ne pas sélectionner la langue).

— la variable de personnalisation $forcer_lang indique à SPIP qu'il doit vérifier si le visiteur dispose d'un cookie de langue, et si oui le renvoyer vers la page correspondante. C'est ce que fait la page de connexion à l'espace privé livrée en standard avec SPIP.

— les balises #MENU_LANG (et #MENU_LANG_ECRIRE) affichent un menu de langue qui permet au visiteur de choisir « cette page en ... ». La première balise affiche la liste des langues du site ; la seconde la liste des langues de l'espace privé (elle est utilisée sur la page de connexion à l'espace privé).

— enfin, les critères optionnels (*cf.* SPIP 1.7, SPIP 1.7.2) permettent d'utiliser une même boucle (en fait, un même squelette) pour afficher soit tous les articles du site dans toutes les langues, soit seulement les articles dans la langue passée dans l'URL. Ça peut être utile dans les backend, par exemple, ou dans les boucles de recherche :

```
<BOUCLE_recents(ARTICLES){lang?}{par date}{inverse}{0,10}>
```

```
<BOUCLE_recherche(ARTICLES){lang?}{recherche}{par points}{inverse}{0,10}>
```

Des squelettes internationaux pour un site massivement multilingue

Ce qui précède nous a permis de rendre multilingue la partie proprement SPIP de notre squelette : tout ce qui est issu des boucles s'affiche dans le bon sens, avec la bonne typographie, et les éléments produits par SPIP (formulaires, dates...) sont dans la langue demandée.

Pour un site présentant un nombre modeste de langues (bilingue par exemple), ou pour lequel il existe une langue principale et quelques langues annexes, on pourrait en rester là. Les textes figés présents dans les squelettes, c'est-à-dire les mentions écrites directement dans le HTML comme « Plan du site », « Espace de rédaction », « Répondre à ce message »... peuvent dans certains cas rester dans une seule langue ; ou alors, un site bilingue pourra utiliser des squelettes séparés pour chacune des deux langues.

Cependant, si vous voulez réaliser et gérer efficacement un site présentant beaucoup de langues à part entière, il devient illusoire de maintenir des squelettes séparés, ou d'imposer une navigation dans une langue unique (même en anglais ou en espéranto...). Pour réaliser un jeu de squelettes unique fonctionnant dans toutes les langues, il faut internationaliser les squelettes afin de modifier les textes indépendamment du code HTML qui les contient (qui reste, lui, figé d'une langue à l'autre). Cette tâche nécessite de mettre un peu « les mains dans le cambouis » et fait l'objet d'un article séparé.

Détails annexes

- Les raccourcis typographiques <code> et <cadre> produisent toujours un texte écrit de gauche à droite, même si la langue de l'article s'écrit normalement de droite à gauche. En effet ces deux raccourcis sont destinés à afficher du code ou des données informatiques, qui sont à peu près toujours écrits de gauche à droite (et, la plupart du temps, en caractères occidentaux).

- Toujours en ce qui concerne le sens d'écriture, notons que les attributs left et right du HTML sont aussi souvent présents dans les feuilles de style. Cela veut dire que vous devrez peut-être inclure la partie correspondante de la feuille de style dans vos squelettes (pour utiliser les balises #LANG_LEFT et #LANG_RIGHT) plutôt que de la placer dans un fichier séparé.

Voici un récapitulatif du comportement des balises SPIP liées au sens de l'écriture :

Langue	#LANG_LEFT	#LANG_RIGHT	#LANG_DIR
langues écrites de gauche à droite	left	right	ltr
arabe,farsi,hébreu...	right	left	rtl

- Un filtre |direction_css permet d'« inverser » un fichier CSS pour les langues s'écrivant de droite à gauche. Si la feuille de style CSS à inverser s'appelle par exemple *style.css*, ce filtre utilise (dans le cas où la langue courante s'écrit de droite à gauche) une éventuelle feuille *style_rtl.css* ; si celle-ci

n’existe pas, il crée automatiquement une feuille RTL en remplaçant toutes les occurrences de *left* par *right* et vice-versa (et la stocke dans le répertoire IMG/cache-css/). Il s’applique le plus souvent sur une balise #CHEMIN, de cette façon : [(#CHEMIN*style.css*|direction_css)].

Notes

[1] On précise ici que dans la configuration livrée d’origine, SPIP reste monolingue, afin de ne pas compliquer du tout l’interface.

[2] Si l’on veut au contraire un affichage vide, il faut créer explicitement une première partie vide avec un nom de langue quelconque.

[3] Théoriquement le HTML devrait régler tous ces détails automatiquement, mais le résultat n’est pas toujours à la hauteur, surtout lors des passages à la ligne ou si l’on mélange des langues utilisant un sens d’écriture différent.

[4] Malheureusement, les instances de normalisation semblent pour l’instant ignorer le boustrophédon, ce qui empêche son utilisation en HTML.

[5] Tous, ou presque tous, nous vous laissons découvrir si votre mise en page présente des cas particuliers...

Internationaliser les squelettes

L'internationalisation des squelettes, abordée dans cet article est disponible à partir de SPIP 1.7.

Pourquoi créer des squelettes multilingues ?

Dès la mise en ligne de documents, SPIP adapte certaines informations « automatiques » dans la langue désirée. Notamment les dates sont affichées dans la langue du site, ou d'un article, ou d'une rubrique, les formulaires sont affichés dans la langue correspondante (par exemple l'interface pour poster des messages de forums)... Tout cela est d'ores et déjà traduit par SPIP.

Ce n'est cependant pas suffisant : les webmestres insèrent dans leurs squelettes un certain nombre d'informations, décrivant notamment les principes de navigation dans le site. Il est nécessaire, par exemple, d'afficher des textes du style « Plan du site », « Répondre à cet article », « Articles du même auteur », « Dans la même rubrique »... Pour un site dans une seule langue, ces différents éléments sont faciles à insérer : on les insère tels quels dans le code HTML des squelettes. Le problème apparaît lorsque le site est multilingue : sous un article en français, on veut afficher *« Répondre à cet article »*, mais sous un article en anglais, on a besoin d'afficher un autre texte (*« Comment on this article »*).

[SPIP 1.7.2] propose trois méthodes pour gérer ces éléments de texte différents selon les langues :

— (1) une méthode consistant à stocker les éléments de texte des squelettes dans des *fichiers de langue* (un fichier différent par langue utilisée sur le site), séparés des squelettes ; un squelette *unique* (par exemple article.html appelant, selon des codes définis par le webmestre, ces éléments de texte en fonction de la langue utilisée ; de cette façon, un même squelette article.html affichera automatiquement le texte « Répondre à cet article » ou « Comment on this document » en fonction de la langue de l'article. Cette méthode est vivement conseillée, elle offre le plus de souplesse, elle facilite les mises à jour du site (on travaille avec un squelette unique qui gère automatiquement plusieurs langues), et à terme des outils seront ajoutés à SPIP facilitant le travail collectif de traduction de l'interface de votre site (plusieurs administrateurs, parlant chacun une langue différente, pourront *traduire* l'interface d'un même site depuis l'espace privé, sans avoir besoin d'intervenir sur les fichiers des squelettes) ;

— (2) une méthode plus rapidement accessible, techniquement très simple, reposant sur la création de fichiers de squelettes différents pour chaque langue. Dans cette méthode, on fabrique un fichier article.html pour gérer les articles en français, et un fichier article.en.html pour les articles en anglais (note : article.html gère en réalité toutes les langues sauf l'anglais). Cette méthode est déconseillée si le site utilise de nombreuses langues et/ou si on utilise des squelettes distincts selon les rubriques. Par ailleurs, elle est dans tous les cas avantageusement remplacée par la méthode 3 décrite ci-dessous :

— (3) la méthode des « blocs multilingues », introduite par [SPIP 1.7.2], fonctionne aussi bien dans les contenus que dans les squelettes. Il suffit de mettre dans le squelette la construction suivante :

```
<multi>
[fr] Répondre à cet article
[en] Comment on this article
</multi>
```

et la phrase s'affichera dans la langue voulue. Si ce système est très souple, il atteint toutefois ses limites dès que le nombre de langues est important, et que l'on souhaite que différents traducteurs interviennent dans le site (il faut en effet qu'ils puissent modifier le squelette, ce que la méthode des fichiers de langue permet d'éviter).

1. Méthode des fichiers de langue

Le principe des fichiers de langue consiste à insérer dans un squelette unique un *code*, lequel correspondra dans chaque langue à un élément de texte (une « chaîne ») ; entre les différentes langues le code ne varie pas, mais le texte est traduit.

Par exemple, nous pouvons décider que le *code* telechargement correspond :

— en français, à la chaîne *« télécharger ce fichier »*,
— en anglais, à la chaîne *« download this file »*,
— en espagnol, à la chaîne *« descargar este archivo »*,
— etc.

Dans le fichier de squelette des articles (un seul fichier gérera toutes les langues), article.html, il suffit d'insérer le code (notez la syntaxe) :

```
<:telechargement:>
```

Lors de l'affichage d'un article, ce *code* sera remplacé par sa traduction dans la langue de l'article.

Par exemple, dans le squelette article.html, nous insérons dans la boucle affichant les documents associés à l'article, le code suivant :

```
<a href="#URL_DOCUMENT"><:telechargement:></a>
```

Si l'article en question est en français, cela produira :

télécharger ce fichier

si l'article est en anglais :

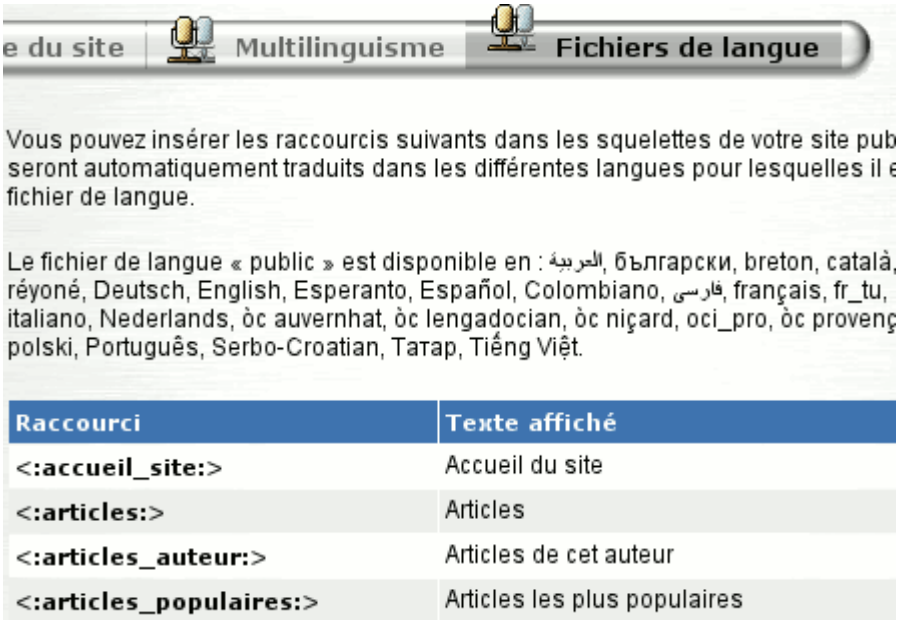
download this file

et ainsi de suite. Un unique squelette, contenant un unique code, affiche un texte traduit dans toutes les langues utilisées sur le site.

- **Utiliser des textes déjà traduits**

Pour faciliter le travail des webmestres, SPIP fournit un ensemble de chaînes déjà traduites (par les traducteurs de SPIP). En utilisant ces chaînes, correspondant à des éléments de texte fréquemment utilisés sur des sites Web, le webmestre peut rapidement réaliser une interface fonctionnant dans différentes langues, mêmes celles qu'il ne parle pas lui-même.

Vous pouvez lister les chaînes disponibles depuis l'espace privé : allez dans la section « Gestion des langues » de la partie « Administration du site », puis cliquez sur l'onglet « Fichiers de langues ». Vous n'avez plus qu'à y piocher les codes de votre choix pour la réalisation de vos squelettes.



Les fichiers de langue dans l'espace privé

Exemple : un webmestre veut réaliser l'interface pour un site en français, en espagnol et en arabe, mais il ne parle pas lui-même l'espagnol et l'arabe. En insérant dans ses squelettes les codes livrés avec SPIP, il n'a pas à se soucier d'obtenir des traductions en espagnol et en arabe, car le travail de traduction a déjà été effectué en amont par les traducteurs de SPIP ; ainsi, en mettant au point l'interface en français avec les codes fournis par SPIP, il sait que ses pages s'afficheront immédiatement en espagnol et en arabe.

Si par la suite on ajoute des articles en polonais, ils seront immédiatement affichés avec les éléments de texte traduits en polonais, sans que le webmestre ait à intervenir à nouveau.

> Autre avantage de cette méthode : elle facilite la création de squelettes « à distribuer » immédiatement multilingues. Des squelettes réalisés selon cette méthode seront immédiatement utilisables dans toutes les langues dans lesquelles sera traduit SPIP.

D'un point de vue technique, les éléments de texte fournis en standard avec SPIP sont stockés dans les fichiers de langue « public » :

— /ecrire/lang/public_fr.php contient les chaînes en français,
— /ecrire/lang/public_en.php en anglais
— etc.

- Créer ses propres codes

Il est de plus possible de créer ses propres codes, correspondant à des chaînes que l'on désire ajouter soi-même.

Il s'agit alors de créer des fichiers de langue personnels, sur le modèle des fichiers public.... Pour créer ses propres fichiers, on installera, dans votre répertoire de squelette ou dans le répertoire /ecrire/lang :

— local_fr.php3 pour définir les chaînes en français,
— local_en.php3 en anglais,
— ...

> *historique :* Dans les versions antérieures à [SPIP 1.8], les fichiers de langue personnels se plaçaient seulement dans le répertoire /ecrire/lang.

Par exemple, on pourra créer les chaînes suivantes :

— telechargement pour afficher « Télécharger la dernière version »,
— quoideneuf pour afficher « Modifications récentes ».

Selon cette méthode, on insère dans les squelettes les codes <:telechargement:> et <:quoideneuf:>, ils seront ensuite affichés avec les traductions correspondantes, telles qu'elles sont définies dans les fichiers

local_...php3.

Notons que les codes sont arbitraires : c'est vous qui les choisissez. Nous recommandons bien évidemment de choisir des codes qui vous permettent de les retenir facilement (plutôt que des numéros par exemple). Comme souvent avec les codes informatiques, il est préférable de n'utiliser que des lettres de l'alphabet latin, et sans accents...

Les *fichiers de langue* contiennent les différentes traductions des codes que vous utiliserez ; ce sont des fichiers PHP contenant chacun un tableau associant aux codes les chaînes correspondantes dans chaque langue.

Ils contiendront par exemple :

- *Version française :*

```
<?php
$GLOBALS[$GLOBALS['idx_lang']] = array(
 'telechargement' => 'T&eacute;l&eacute;charger la derni&egrave;re version',
 'quoideneuf' => 'Modifications r&eacute;centes'
);
?>
```

- *Version catalane :*

```
<?php
$GLOBALS[$GLOBALS['idx_lang']] = array(
 'telechargement' => 'Descarregar la darrera versi&oacute;',
 'quoideneuf' => 'Modificacions recents'
);
?>
```

La construction est la suivante :

— au début du fichier :

```
<?php
$GLOBALS[$GLOBALS['idx_lang']] = array(
```

— à la fin du fichier :

```
);
?>
```

— la partie qu'il faut enrichir soit-même consiste en plusieurs lignes de *définitions*, sur le modèle :

```
'code' => 'La chaîne à afficher',
```

N.B. Chaque ligne de définition se termine par une virgule, sauf la dernière ligne.

N.B.2. Le texte de la chaîne à traduire doit être convertie en codes HTML (les caractères accentués, par exemple, sont convertis en leur équivalent HTML, du type é).

Les apostrophes à l'intérieur de la chaîne doivent être *échappées*, c'est-à-dire précédées d'un antislash. Par exemple, la chaîne « sur l'internet » doit être inscrit : sur l\'internet.

Note : à terme, il est prévu d'inclure un outil permettant de gérer et créer ses propres fichiers de langue sans avoir à modifier « à la main » des fichiers PHP. Un tel outil facilitera de plus l'utilisation de caractères « spéciaux » (caractères accentués, caractères dans des alphabets non occidentaux, échappement des apostrophes...), ainsi que la collaboration de plusieurs personnes au processus de traduction de l'interface du site public.

Pour l'instant, l'outil permettant de gérer la traduction des chaînes de texte n'est pas livré directement avec SPIP, et son utilisation très généraliste (nous l'utilisons pour traduire toute l'interface de SPIP, et pas seulement des fichiers de langue de type local...php3) le rend un peu complexe par rapport à cette tâche. Ce programme, **trad-lang**, qui nous sert à traduire le logiciel SPIP, le site spip.net, etc., est lui-même disponible sous licence GNU/GPL, mais il n'est pas intégré en standard à SPIP. Vous pourrez le télécharger, pour l'utiliser pour votre site ou pour d'autres projets de logiciels. Si vous l'améliorez, ou avez des idées pour le transformer, venez en discuter sur la liste des traducteurs de SPIP, spip-trad.

2. Des squelettes séparés pour chaque langue

La seconde méthode, plus accessible aux webmestres débutants, consiste à créer des squelettes différents pour chaque langue. Un peu sur le même principe qui consiste à créer des squelettes spécifiques pour différentes rubriques pour obtenir des interfaces graphiques différentes.

Nous voulons réaliser un site en français (langue par défaut), en anglais et en espagnol. Nous réalisons trois fichiers de squelette différents :

— article.html pour le français (en réalité, pour toutes les langues qui n'ont pas de fichier de langue spécifique ci-après),
— article.en.html pour l'anglais,
— article.es.html pour l'espagnol.

(Note : si on publie un article en allemand, alors qu'il n'existe pas de squelette article.de.html sur notre site, c'est le squelette article.html qui sera utilisé.)

Important : pour que les squelettes « par langue », définis par l'ajout d'un .lang à la fin du nom, soient pris en compte, il faut obligatoirement qu'il existe la version « par défaut » sur le site.

Ici, si article.html n'existe pas (on aurait préféré nommer directement un article.fr.html), les fichiers anglais et espagnol ne seront pas pris en compte.

On peut combiner cela avec le nommage « par rubriques », et l'ordre suivant sera pris en compte :

— article=8.es.html (le squelette pour les articles en espagnol de la rubrique 8, mais pas ses sous-rubriques),
— article=8.html (le squelette pour les articles de la rubrique 8, mais pas ses sous-rubriques),
— article-2.es.html (le squelette pour les articles en espagnol de la rubrique 2 et ses sous-rubriques),
— article-2.html (le squelette pour les articles de la rubrique 2 et ses sous-rubriques),
— article.es.html (le squelette pour les articles en espagnol),
— article.html (le squelette pour les articles),
— article-dist.html (le squelette pour les articles livré avec SPIP).

Note : sauf pour quelques exceptions, il faut utiliser ici les codes de langues à deux lettres normalisés par l'ISO, comme « es ». On en trouvera notamment la liste sur le site de la Bibliothèque du Congrès des Etats-Unis (eh oui !). Pour s'assurer d'un maximum de compatibilité, il faut utiliser en priorité les codes ISO à deux lettres (« iso 639-1 »), quand ils existent, et pour une désignation plus spécialisée d'une langue dans sa famille linguistique les codes ISO à trois lettres (« iso 639-2 T »). Par exemple, l'allemand sera désigné comme « de ». Mais l'ISO ne prend en compte que 182 langues sur les 5 000 à 7 000 langues parlées dans le monde ; pour les langues non encore répertoriées, on peut s'inspirer du répertoire proposé par le site ethnologue.com ; pour en savoir plus, rendez-vous sur la liste spip-trad.

Se simplifier la vie. Une des méthodes de structuration très simple qu'autorise SPIP pour gérer un site multilingue consiste à associer directement les langues aux rubriques (et non article par article). Ainsi, dans le cas où les articles d'une même langue sont regroupés dans une même rubrique (voire par secteurs), on peut se contenter de créer les squelettes spécifiques *par rubrique*, sans utiliser alors les noms de fichiers par langues.

De cette façon, si, tous les articles en espagnol sont regroupés dans la rubrique 8 (et ses sous-rubriques), on peut se contenter de nommer le fichier adapté à l'espagnol rubrique-8.html plutôt que rubrique.es.html.

On devine les problèmes qui peuvent se poser avec cette méthode :

— le fichier article-2.html *doit* exister si on veut que article-2.es.html puisse être sélectionné ;
— on ne peut pas décider, à l'inverse, que article.es.html doit être utilisé à la place d'article-2.html si article-2.es.html n'existe pas : la sélection par rubriques a toujours la priorité sur la sélection par langues ;
— une correction de mise en page dans un squelette implique de faire la même correction dans les autres versions,
— on doit pouvoir insérer soi-même des éléments de texte dans les fichiers des squelettes, alors qu'on ne comprend pas forcément la langue (imaginez-vous créer ainsi les squelettes en arabe alors que vous parlez un peu le français de l'ouest et assez mal l'occitan du nord...).

Cette méthode est donc destinée à travailler rapidement sur des sites peu complexes (peu ou pas de squelettes spécifiques aux rubriques) et/ou comportant peu de langues différentes. Dès que votre projet multilingue devient un peu plus ambitieux, il est vivement conseillé de travailler avec la méthode des fichiers de langue.

3. Les blocs multilingues

Les blocs multi définis dans l'article Réaliser un site multilingue fonctionnent aussi bien dans le texte des auteurs ou mots-clés que dans les squelettes. Attention, ces blocs ne gèrent *que* du texte, et pas des boucles SPIP !

Pour gérer rapidement un site multilingue, c'est sans doute, dans un premier temps, la meilleure méthode, à la fois accessible et efficace ; ensuite, une fois votre site stabilisé, si vous avez besoin d'affiner vos squelettes (par exemple, pour les ouvrir à plus de langues ; ou pour les distribuer sous forme de contrib ; ou encore pour les « professionnaliser »), il faudra transformer vos blocs multi en fichiers de langue.

Utiliser des URLs personnalisées

Après l'installation, les pages générées par SPIP utilisent des adresses relatives ressemblant à spip.php?article765, donnant des URLs du type http://www.spip.net/spip.php?article765.

Il y a possibilité d'avoir des adresses plus à votre goût — par exemple article123.html ou Titre-de-l-article.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.spip.net/fr_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.spip.net/spip.php?article765 ; en général, cette URL peut aussi être tapée directement dans le navigateur (vous pouvez vérifier).

Choisir le type d'URLs apparentes

Dans le fichier ecrire/mes_options.php [1] (à créer le cas échéant), vous pouvez déclarer une variable PHP contenant le type d'URLs à utiliser. En l'absence de ce réglage, SPIP utilisera :

```
$type_urls = "page";
```

La variable $type_urls détermine le nom du fichier PHP qui est appelé pour gérer les URLs. Avec la déclaration par défaut ci-dessus, c'est le premier urls/page.php trouvé dans SPIP_PATH.

Vous remarquerez que SPIP propose aussi les fichiers urls/standard.php, urls/html.php, urls/propres.php, urls/propres-qs.php et urls/propres2.php dans le répertoire ecrire/

- Le fichier urls/html.php permet de traiter des adresses du type (« article123.html »). Vous pouvez décider d'utiliser les « URLs "html" » en mettant dans ecrire/mes_options.php la ligne :
  $type_urls = "html";

- Le fichier urls/propres.php permet de traiter des adresses du type (« Titre-de-l-article »). Il faut alors ajouter :
  $type_urls = "propres";

- Le fichier urls/propres2.php est une variation du précédent, qui donne des adresses du type (« Titre-de-l-article.html »). Il faut alors ajouter :
  $type_urls = "propres2";

- Le fichier urls/propres-qs.php est une variation du précédent, qui donne des adresses du type (« ./?Titre-de-l-article »). Il faut alors ajouter :
  $type_urls = "propres-qs";. Ce dernier est à utiliser si votre hébergeur ne vous permet pas d'utiliser le module de réecriture d'urls d'apache (cf. mod_rewrite)

- Enfin, le fichier urls/standard.php permet aux nostalgiques des versions précédentes de spip de donner des adresses du type (« article.php3?id_article=765 »). Il faut alors ajouter :
  $type_urls = "standard";

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 : urls/XXX.php. Il est par exemple très aisé de modifier la fonction _generer_url_propre() dans urls/propres.php pour obtenir des variations très intéressantes ; si vous faites cela, merci de partager vos modifications sur le site SPIP Contrib'.

Programmer la traduction des adresses apparentes en adresses réelles

Pour que l'adresse article123.html appelle bien en réalité le fichier PHP spip.php 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 (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, depuis SPIP 1.8, SPIP 1.8.1 est fournit un fichier htaccess.txt à titre d'exemple, qui fonctionne sur la plupart des hébergeurs avec les types d'URLs cités précédemment (« standard », « html », « propres » et « propres2 »). Pour l'activer il faut le recopier à la racine du site sous le nom .htaccess. Il est fortement conseillé de l'ouvrir au préalable pour vérifier quelques aspects de configuration.

Vous devrez ensuite tester la validité de ces adresses, en appelant la page « Voir en ligne » sur un article, un auteur, une brève, une rubrique, etc.

Générer les URLs apparentes dans les pages SPIP

Afin d'afficher partout les URLs du type choisi, utilisez dans vos squelettes les balises #URL_ARTICLE, #URL_RUBRIQUE, #URL_BREVE, etc.

Transition d'un type d'URLs à l'autre

Depuis SPIP 1.8, SPIP 1.8.1, tout est prévu pour que la transition d'un type d'adresses à l'autre se fasse en douceur : installez le fichier htaccess.txt, et vous pouvez ensuite librement basculer des adresses « standard » aux adresses « propres2 », « propres » ou « html », et vice-versa, sans jamais provoquer d'erreur 404 pour les visiteurs (ou les moteurs de recherche) qui auraient mémorisé les anciennes adresses.

Dernier détail pour faciliter la transition, si vous choisissez les URLs propres ou propres2, les visites des pages portant les anciennes adresses (standard ou html) sont redirigées automatiquement vers les nouvelles adresses.

Notes

[1] Remarque : les versions précédentes de SPIP incluaient le fichier inc-urls.php3 à la racine du site s'il était présent ; cette méthode est encore valable mais est considérée comme obsolète...

Le moteur de recherche

SPIP intègre un moteur de recherche, désactivé par défaut. Ce moteur, lorsqu'il est activé par un administrateur dans la page de configuration, permet d'effectuer des recherches sur différents types d'informations présentes dans la base de données : les articles, les rubriques, les brèves, les mots-clés et les auteurs. Depuis SPIP 1.7.1 les fils de discussion des forums (threads) et les signatures de pétitions sont également indexés.

Principe

Il y a deux grandes façons de faire un moteur de recherche. La première est de chercher tout bêtement dans le type de stockage existant (fichiers HTML, base de données... selon le type de site). Cette méthode est très lente car le type de stockage n'est pas prévu à cet effet.

La seconde méthode, qui a été choisie pour SPIP (et qui est aussi celle de tous les moteurs professionnels), est d'établir un mode de stockage spécifique aux besoins de la recherche. Par exemple, le score de chaque mots d'un article peut être stocké directement afin d'être facilement retrouvé, et d'avoir le score total d'une recherche par une simple addition. L'avantage est que la recherche est très rapide : presque aussi rapide que n'importe quel calcul de page. L'inconvénient est qu'il faut une phase de construction du dit stockage des informations : cela s'appelle l'indexation. L'indexation a un coût en termes de ressources (temps de calcul et espace disque), et elle introduit également un léger décalage temporel entre l'ajout ou la modification d'un contenu, et la répercussion de cet ajout ou de cette modification sur les résultats de recherche.

D'autre part, dans le cas de SPIP, nous sommes obligés d'utiliser PHP et MySQL comme pour le reste du logiciel, ce qui ne permet pas de réaliser un moteur très performant, en termes de rapidité, mais aussi de pertinence ou d'enrichissements divers (indexation de documents extérieurs au site, création de champs sémantiques permettant de proposer des recherches plus fines, etc.).

L'avantage du moteur interne, cependant, c'est qu'il permet de gérer l'affichage des résultats à travers les mêmes méthodes (squelettes) que le reste des pages de SPIP, et à l'intérieur du même environnement visuel.

L'indexation

L'indexation est réalisée lors des visites du site public. Afin d'éviter que le cumul d'une indexation et d'un calcul de page ne mène à un *timeout* sur les serveurs particulièrement lents, SPIP attend qu'une page soit affichée en utilisant le cache [1].

L'indexation traite une à une les différentes données textuelles d'un contenu donné : par exemple, pour un article, le chapo, le descriptif, le titre, le texte... Pour chaque donnée textuelle, le score de chaque mot est calculé en comptant simplement le nombre d'occurrences. A cet effet, les mots de trois caractères ou moins sont ignorés (ils sont, pour la plupart, non significatifs, et alourdiraient la base de données) ; d'autre part, les caractères accentués sont translittérés (convertis en leurs équivalents non-accentués), pour éviter les problèmes de jeux de caractères et aussi pour permettre d'effectuer des recherches en version non accentuée.

Ensuite, les scores de chaque mot sont cumulés, de façon pondérée, entre les différentes données textuelles du contenu indexé. La pondération permet, par exemple, de donner plus de poids aux mots présents dans le titre d'un article que dans le corps du texte ou le post-scriptum...

Les fonctions d'indexation peuvent être étudiées au sein du fichier ecrire/inc_index.php3. Pour mieux visualiser la dynamique d'indexation du site, vous pouvez ouvrir le fichier ecrire/data/spip.log, ou encore regarder la page ecrire/admin_index.php3 (nota : cette page, encore expérimentale, n'est pas livrée avec toutes les versions de SPIP, et n'existe qu'en français. Elle a été retirée de [SPIP 1.9] et est désormais disponible dans le plugin recherche_etendue avec d'autres fonctions des gestion de l'indexation).

Dans la version [SPIP 1.6], d'importantes modifications ont été apportées au comportement du moteur :

- meilleur comportement dans un environnement multilingue ;
- le tiret bas (*underscore*) n'est plus considéré comme un séparateur de mot, mais comme un caractère alphabétique (utile pour de la documentation informatique) ;
- les mots de deux lettres (et plus) ne contenant que des majuscules et des chiffres sont considérés comme des sigles, et sont indexés, ce qui supprime l'un des principaux inconvénients de la limitation de l'indexation aux mots de plus de 3 lettres (G8, CNT, ONU sont désormais indexés).

La recherche

La recherche s'effectue simplement en séparant le texte de recherche en ses différents mots ; le même filtre est appliqué que lors de l'indexation : suppression des mots de trois lettres ou moins (sauf sigles), et translittération.

Pour chaque contenu recherché, le score des différents mots est ensuite récupéré puis additionné afin d'obtenir le score total. Enfin, les résultats sont en général affichés par ordre décroissant de score ({par points}{inverse}), c'est-à-dire de pertinence (mais cela est laissé à la volonté de la personne qui écrit les squelettes de mise en page).

Performances

Rapidité

Sur un serveur récent et pas trop chargé, l'indexation d'un texte long (plusieurs dizaines de milliers de caractères) prendra entre une et deux secondes : l'attente est presque imperceptible, comparée aux délais de chargement via le réseau. Les contenus courts sont indexés de façon quasi-instantanée. Bien sûr, ces affirmations doivent être modulées selon la taille du site. Un site vraiment très gros risque de voir les temps d'indexation s'allonger légèrement ; pour relativiser, signalons qu'un site comme Le Courrier des Balkans comporte, à la date d'écriture de ce texte, environ 3 800 articles publiés, et plus de 7500 messages de forum, et que le moteur de recherche de SPIP ne donne aucun signe de faiblesse.

Par ailleurs, statistiquement, on peut considérer de façon approximative que chaque contenu ne sera indexé qu'une seule fois : compte tenu qu'il y a en général beaucoup plus de visites sur un site que de mises à jour de contenus, le surcroît de charge du serveur apparaît négligeable.

Qualité

La qualité de l'indexation est plus faible que sous des moteurs de recherche professionnels. PHP étant un langage plutôt lent, la phase d'extraction des mots a dû être simplifiée au maximum pour que les temps d'indexation restent minimes. Par conséquent, les données d'indexation comportent quelques « déchets », c'est-à-dire des morceaux de texte qui ne correspondent pas à de « vrais » mots, mais ont été indexés comme tels (il s'agit souvent de contenus techniques comme des noms de fichiers, ou de passages à la ponctuation malmenée). L'exemple d'uZine, où l'on constate environ 2 % de tels « déchets », nous laisse cependant penser que ces données sont quantité négligeable, d'autant qu'il y a peu de chance qu'elles déclenchent un résultat positif lors d'une recherche.

La recherche n'offre pas d'opérateurs booléens, l'opérateur implicite étant grosso modo un « OU » logique. Cependant, depuis SPIP 1.7.1, les articles trouvés s'affichent dans un ordre qui privilégie les résultats contenant le plus de mots orthographiés précisément selon la requête. Ainsi, une requête sur « la main rouge » mettra en évidence les articles contenant « main » et « rouge », loin devant les articles ne contenant que « maintenance » ou « rouget » - ceux-ci apparaîtront, mais plus loin dans le classement.

Espace disque

MySQL n'étant pas spécialement conçu pour le stockage de données d'indexation, l'utilisation du moteur de recherche a tendance à faire beaucoup grossir l'espace disque utilisé par la base de données. Pour donner quelque précision, disons qu'un contenu génère des données d'indexation de taille comprise entre la taille du contenu et le double de celle-ci. Donc, si l'on fait abstraction des données ne donnant pas lieu à indexation (les forums par exemple), l'indexation fait entre doubler et tripler la place prise par la base de données. Cela peut être gênant si la place vous est très comptée.

Si jamais vous désactivez le moteur de recherche afin d'économiser de l'espace disque, n'oubliez pas ensuite d'effacer les données d'indexation (dans la page de sauvegarde/restauration de la base de données) afin de réellement libérer l'espace disque occupé par ces données.

Notes

[1] Si, donc, vous avez mis tous les $delais à zéro, ou si votre site n'est pas visité (site de test), l'indexation n'aura pas lieu.

Tidy : validation XHTML 1.0

[SPIP 1.8.1] permet aux webmestres qui le désirent d'utiliser sur leurs pages l'outil Tidy, introduisant ainsi une fonction de validation XHTML sur leur site.

Principe général

Tidy est un outil (externe à SPIP) qui permet de transformer du code HTML 4 *relativement propre* en code XHTML 1.0 transitional valide. SPIP exploite cet outil pour permettre aux webmestres de proposer des sites conformes aux recommandations du XHTML 1.0 transitional.

Important : Tidy n'est pas un outil « magique » : il est incapable de transformer du code « très sale » en code conforme. Face à certaines « erreurs » de code, il refuse purement et simplement de fonctionner. L'outil est donc intégré dans SPIP pour, à la fois, créer du code conforme, et permettre de « traquer » les erreurs de code dans vos pages.

- **Étape 1 :** il convient, avant tout, de créer du code aussi propre et conforme que possible, avant même le passage par Tidy. Cela se fait à plusieurs niveaux :
  — tout d'abord, SPIP produit, dans ses traitements typographiques, du code propre, et à chaque version plus conforme (« *compliant* ») ; notez bien : SPIP vise la conformité *HTML 4.01 transitional* ;
  — les squelettes du site public doivent eux-mêmes être aussi conformes que possible (pour rester cohérent, on visera la conformité HTML 4.01).

- **Étape 2 :** si Tidy est présent sur le serveur, et si le webmestre active cette option (voir plus loin), alors SPIP fait passer les pages produites par Tidy, qui va alors tenter de nettoyer les pages et de les transformer en pages conformes au XHTML 1.0 transitional.

- **Étape 3 :** si le traitement a bien fonctionné (Tidy n'a pas rencontré d'erreur « bloquante »), alors la page affichée est bien du XHTML 1.0, dont on peut d'ailleurs faire valider la conformité par le W3C validator ; en revanche, si Tidy n'a pas fonctionné (voir plus loin), alors c'est la page d'origine qui est affichée — dans ce cas (c'est un outil important à prendre en compte), SPIP affiche un bouton d'administration signalant l'« erreur Tidy », et créé un fichier récapitulant les différentes pages ayant rencontré des erreurs.

Encore une fois, afin de ne pas préter à l'outil des possibilités « magiques » qu'il n'a pas, et à ses développeurs des intentions qui ne sont pas les leurs, il est important de noter que SPIP ne se contente pas de se reposer sur un outil (Tidy, dont les limites sont connues), mais de l'intégrer dans une logique plus large de mise en conformité :
— d'abord en améliorant le code produit par SPIP,
— en utilisant ensuite Tidy à la fois comme outil de « nettoyage » du code, mais aussi *en proposant une indication des erreurs* pour permettre au webmestre d'améliorer son code.

La mise en conformité du code ne peut reposer sur une solution purement technique, mais sur un travail personnel pour lequel, avec Tidy, SPIP offre un outil de suivi.

Installer Tidy

- **Tidy, extension de PHP**

Tidy existe en tant qu'extension de PHP. C'est la façon la plus simple de l'utiliser, l'outil étant alors directement utilisable par le webmestre. Pour déterminer sa présence, on pourra consulter la page /ecrire/info.php3 de son site pour obtenir la configuration de son serveur et la liste des extensions de PHP disponibles.

N.B. : À l'heure actuelle, l'utilisation de Tidy en tant qu'extension de PHP n'a pas pu être testée en situation de production dans SPIP. *En théorie*, cela fonctionne, mais tout retour d'expérience de la part de webmestres intéresse les développeurs — sur la liste spip-dev. (Nous avons besoin de retours d'expérience sur les versions 1 et 2 de Tidy, c'est-à-dire sur des sites en PHP 4, en PHP 5, mais également des installations via PEAR.)

- **Tidy comme programme indépendant**

Il est par ailleurs possible d'utiliser Tidy en ligne de commande (c'est-à-dire en tant que programme indépendant de PHP s'exécutant directement sur le serveur).

Cette version est particulièrement pratique, puisque :

— il existe des versions de Tidy déjà compilées pour la plupart des systèmes d'exploitation,
— il est souvent possible et simple d'installer ces versions de Tidy sur un hébergement sans avoir d'accès *root*,
— certains administrateurs de sites ont subi des incompatibilités lors de l'installation de Tidy en extension PHP (avec, semble-t-il, ImageMagick) ; la version en ligne de commande ne provoque pas ce genre de problème.

Avant toute chose, vérifiez que Tidy n'est pas déjà présent sur votre serveur. Pour cela, installez dans le fichier /ecrire/mes_options.php3 les lignes suivantes :

```
define('_TIDY_COMMAND', 'tidy');
$xhtml = true;
```

Et vérifiez sur votre site public si les pages sont modifiées (soit transformées en XHTML, soit affichage du message « Erreur tidy »). Si cela ne fonctionne pas, pensez à supprimer ces deux lignes, et essayez d'installer Tidy selon la méthode suivante (ou demandez à la personne responsable de votre hébergement de le faire).

Vous pouvez installer une version déjà compilée de Tidy correspondant à votre système.

— On trouvera ces versions sur le site officiel de Tidy ; il en existe pour Linux, divers *BSD, MacOS X, etc.
— Décompressez l'archive téléchargée, et installez le fichier « tidy » sur votre site.
— Vérifiez les droits d'exécution de ce fichier sur le serveur (si nécessaire, passez les droits en « 777 »). (Si vous avez un accès SSH à votre serveur, vous pouvez tester le programme directement depuis le terminal. Si vous n'avez pas un tel accès, rien de grave, passez aux étapes suivantes, sachant que vous serez plus démuni si cela ne fonctionne pas du premier coup.)
— Configurez l'accès à ce fichier en indiquant le chemin d'accès en le définissant ainsi :

```
define('_TIDY_COMMAND', '/usr/bin/tidy');
$xhtml = true;
```

Si le chemin indiqué dans _TIDY_COMMAND est correct, alors Tidy sera déclenché lors des affichages des pages de votre site public.

Important. La définition de _TIDY_COMMAND doit se trouver dans /ecrire/mes_options.php3 et non à la racine du site dans /mes_fonctions.php3. Cela est dû au fonctionnement assez spécifique du système (post-traitement des fichiers tirés du cache de SPIP).

La partie $xhtml = true;, en revanche, fonctionne comme une « variable de personnalisation » ; vous pouvez, si vous souhaitez faire des essais ou restreindre le fonctionnement à une partie du site, définir cette variable au niveau du fichier d'appel, par exemple article.php3 si vous ne voulez passer tidy que sur les articles.

Nettoyez votre code...

Encore une fois, il faut bien comprendre que Tidy n'est capable de rendre conforme que du code qui est déjà, à l'origine, très propre. Avec les squelettes livrés avec SPIP (eux-mêmes déjà conformes HTML 4) et le code produit par défaut par SPIP, cela ne pose aucun problème : le code est très proche du HTML 4 conforme (« *compliant* »), aussi Tidy n'a aucun mal à en faire du XHTML 1.0 transitional parfaitement conforme.

Lorsque Tidy a bien fonctionné, vous constatez dans le code source de vos pages :

— que le « DOCTYPE » de votre page est devenu « XHTML... »,
— que le code est joliment indenté,
— que l'ensemble passe la validation W3C sans difficulté.

Si le DOCTYPE n'est pas modifié, c'est que Tidy a renoncé à corriger la page, dans laquelle il a rencontré des erreurs impossibles (pour lui) à corriger.

Les erreurs peuvent avoir deux sources : les squelettes, et le texte des articles.

- **Vos squelettes ne sont eux-mêmes pas conformes ;** dans ce cas, il faut les corriger. C'est le cas le plus fréquent.

Vous pouvez, ici, commencer par désactiver Tidy (passer la variable $xhtml à false), et passer vos pages au Validator en visant le conformité HTML 4.01 transitional (SPIP visant, dans ses traitements typographiques, cette conformité, autant rester cohérent). Le W3C Validator est un outil très pratique pour nettoyer son code.

Une fois vos squelettes aussi proches que possible du HTML 4, Tidy n'aura pas de difficulté à produire du XHTML très compliant. Si ces pages sont complètement conformes, c'est encore mieux ! (Et pas impossible : les squelettes livrés avec SPIP sont déjà conformes).

- **Certaines articles contiennent des codes erronés**

SPIP laissant les rédacteurs travailler en « code source », ceux-ci peuvent insérer du code non conforme à l'intérieur de leurs articles. (Par exemple, dans la documentation de www.spip.net, on trouve à certains endroits l'utilisation de balises HTML <tt>...</tt> que Tidy considère comme inacceptables.)

Aussi, une fois les squelettes nettoyés, on pourra chercher à corriger les textes de certains articles (cela concerne, donc, les insertions de HTML directement dans les articles ; encore une fois, le code produit par SPIP est essentiellement conforme, et ne provoque pas de « blocage » de Tidy).

Pour cela, outre l'apparition, sur les pages concernées, d'une bouton d'aministration intitulé « Erreur Tidy », SPIP tient à jour en permanence un fichier /ecrire/data/w3c-go-home.txt qui contient la liste des pages impossibles à valider [1]. Une fois vos squelettes rendus propres (paragraphe précédent), le nombre de pages défaillantes devrait être limité aux articles contenant du code HTML non accepté par Tidy.

> Il est assez difficile de définir précisément ce que Tidy considère comme une « erreur insurmontable ». Par exemple, les balises mal fermées ne sont pas réellement considérées comme impossible à corriger (par exemple, passer en italique dans un paragraphe et fermer l'italique dans un autre paragraphe fabrique du code HTML non conforme, que Tidy parvient cependant à bien corriger). Le plus souvent, il s'agit de balises insérées à la main totalement inexistantes (par exemple : taper <bt> à la place de
), ou de balises HTML considérées comme obsolètes dans le HTML 4 (telles que <tt> ou <blink>), que Tidy refusera purement et simplement de traiter.

Conclusion

Encore une fois, l'outil Tidy ne doit surtout pas être considéré comme un produit « miracle » : il ne transforme pas du code sale en code conforme. Son intégration dans SPIP suit donc une logique d'accompagnement d'un processus de nettoyage :

— SPIP lui-même continue à produire du code de plus en plus propre ;

— Tidy sert alors à mettre la dernière touche de nettoyage sur du code déjà très « compliant » (au passage, en résolvant quelques incompatibilités difficiles à gérer avec un code unique entre le HTML et le XHTML, telles que certaines balises auto-fermantes en XHTML, et non fermées en HTML, telles que
) ;

— Tidy est ensuite utilisé pour identifier les erreurs de codage dans le code source des articles eux-mêmes (lors de l'insertion, assez fréquent, de code HTML « à la main » dans le corps des articles).

Encore une fois, ces fonctionnalités sont toutes récentes, et demandent sans doute des tests supplémentaires. N'hésitez pas à faire part de votre expérience sur spip-dev.

Notes

[1] Ce fichier titre son nom de l'article W3C go home!, publié sur sur uZine, qui critiquait l'acharnement des prophètes de la *compliance* à emm... les webmestres qui se contentent de faire des pages Web ; l'essentiel, rappelons-le, est de publier sans se casser la tête. Si en plus on peut le faire de manière conforme, tant mieux, et c'est l'objet de cette passerelle SPIP-Tidy.

Les variables de personnalisation

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.

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 comme globales, avec une syntaxe un peu différente, dans un fichier intitulé mes_fonctions.php (mes_fonctions.php3 dans les versions antérieurs à SPIP 1.9), placé à la racine du site, ou, **de préférence**, dans votre dossier squelettes/ (cf. Où placer les fichiers de squelettes ?).
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...). Elles s'insèrent naturellement à côté des variables obligatoires $fond et $delais. Voir les exemples.

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 = "</h3>\n";
```

- $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 ^{, et de fermer avec}.

- 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 = "<li>";
```

- $nombre_surligne représente le nombre maximum de fois où un mot recherché sera surligné dans le texte. Par défaut, cette valeur est définie à **4**.

- $ligne_horizontale est le code de remplacement du raccourci typographique ---- (quatre tirets) ou ____ (quatre caractères de soulignement) qui permet d'insérer une ligne horizontale dans le texte. Par défaut, c'est le code <hr class="spip" />.

- $url_glossaire_externe est l'adresse utilisé pour le raccourcis automatiques [?SPIP] vers un glossaire. Par défaut, le glossaire externe renvoie vers l'encyclopédie libre wikipedia.org.

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 privé, comme « proposés aux visiteurs du site public » sont utilisés.

Interdire l'affichage des boutons d'admin

Toutes les pages de squelette se voient ajouter des « boutons d'admin » (notamment : « recalculer cette page ») lorsqu'on est administrateur et qu'on a activé le cookie de correspondance. Cette fonctionnalité, très pratique pour gérer le site, peut s'avérer malpratique dans certains cas ; par exemple pour des fichiers XML, que l'on ne veut en aucun cas voir perturbés par de tels ajouts.

[SPIP 1.7] La variable flag_preserver permet d'interdire ces affichages.

```
$flag_preserver = true;
```

On verra par exemple l'utilisation de cette variable dans backend.php3.

Le dossier des squelettes

Depuis 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 la variable $dossier_squelettes, dans le fichier mes_fonctions.php placé à la racine du site, ou **de préférence**, dans ecrire/mes_options.php.

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

À partir de ce moment-là, SPIP ira chercher en priorité les squelettes présents dans le dossier design/ (que vous aurez créé à la racine du site). Si, de plus, vous utilisez <INCLURE{fond=xxx}>, SPIP ira chercher le fichier xxx.html d'abord dans design/, puis, s'il n'y figure pas, à la racine du site.

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. En effet, même s'ils sont situés dans ce sous-dossier, l'HTML contenu dans ces fichiers de squelette doit être conçu comme s'ils étaient à la racine. Ainsi, les liens vers les images ou les CSS, notamment, risquent de « casser ».

Cette variable permet d'indiquer un ou plusieurs dossiers où SPIP cherchera les squelettes en priorité :

```
<?php
   $GLOBALS['dossier_squelettes'] = 'mes_skel1:mes_skel2';
```

Ceci ouvre différentes possibilités, comme :

- essayer un nouveau jeu de squelettes sans écraser l'ancien,
- organiser vos squelettes en une arborescence de sous-répertoires : mes_skel:mes_skel/rss:mes_skel/formulaires:mes_skel/mailing,
- gérer dynamiquement plusieurs jeux de squelettes grâce à des *plug-ins* comme le switcher,
- etc.

Exemples

- Pour modifier des variables uniquement pour un certain type de squelettes (par exemples pour les pages de rubriques), il suffit de les définir dans le fichier d'appel de ces squelettes. Par exemple, pour les rubriques, on peut fixer des valeurs directement dans rubrique.php3 :

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

Ici, on a modifié la valeur de l'espace autour des logos.

- Pour modifier des valeurs de variables pour l'ensemble du site, on peut les définir dans le fichier mes_fonctions.php.

Attention, lorsqu'on définit des valeurs dans ce fichier, il faut impérativement utiliser la syntaxe $GLOBALS['xxx'] pour chacune des variables à personnaliser. Par exemple, pour définir la valeur de $debut_intertitre, on utilise la syntaxe $GLOBALS['debut_intertitre'].

L’utilisation de cette syntaxe est imposée par des impératifs de sécurité des sites.

```
<?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;
?>
```

Il existe d’autres variables permettant de modifier le comportement de SPIP au niveau technique. Ces variables sont décrites sur le site de documentation technique.

Le support LDAP

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 informations 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 : ceux-ci peuvent être rédacteurs (conseillé), administrateurs ou simples visiteurs.

> *Note : par défaut, l'extension LDAP de PHP n'est généralement pas activée, donc SPIP n'affichera pas le formulaire correspondant lors de l'installation. Pensez à activer l'extension LDAP dans votre installation de PHP si vous voulez utiliser LDAP avec SPIP.*

Si SPIP est déjà installé et que vous voulez configurer l'annuaire LDAP, il faudra reprendre l'installation en effaçant le fichier ecrire/inc_connect.php3.

Une fois la configuration correctement effectuée, tous les utilisateurs de l'annuaire LDAP seront identifiés en tapant leur login (ou nom) dans l'annuaire LDAP, puis leur mot de passe. Notez que cela n'empêche pas de créer directement des auteurs dans SPIP ; ces auteurs ne seront pas recopiés dans l'annuaire mais gérés directement par SPIP. D'autre part les informations personnelles (biographie, clé PGP...) des auteurs authentifiés depuis LDAP ne seront pas non plus recopiées dans l'annuaire. Ainsi SPIP n'a besoin que d'un accès *en lecture seule* à l'annuaire LDAP.

Important : créez toujours un premier administrateur « normal » (non LDAP) lors de l'installation de SPIP. C'est préférable pour éviter d'être bloqué en cas de panne du serveur LDAP.

Pour en savoir plus

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.

Quand 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.

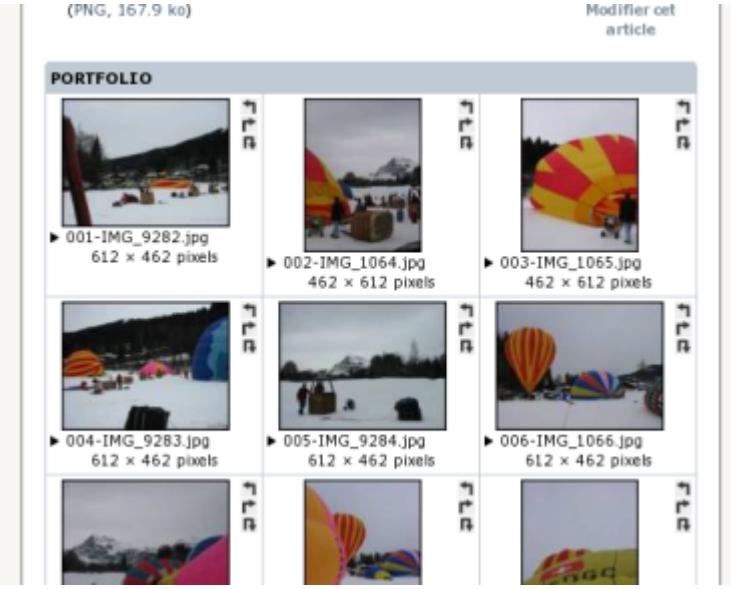
Le traitement des images

via GD, GD2 ou Imagemagick

SPIP permet d'utiliser les systèmes de traitement d'images installés sur votre site d'hébergement. Introduites dans la version 1.7, ces fonctions de SPIP sont complétées et plus puissantes.

SPIP utilise le traitement d'images de trois manières différentes :

— la création de vignettes de prévisualition pour les images installées comme « documents joints » ; cela était déjà présent dans la version 1.7 ; [SPIP 1.8] permet de plus, dans ces « portfolios », de faire subir à chaque image une rotation à 90° (cela est particulièrement intéressant lorsqu'on installe une série de photographies depuis un appareil photo numérique) ;



— à de nombreux endroits dans l'espace privé, l'affichage de « vignettes » destinées à illustrer la navigation, à partir des logos des articles, des rubriques et même des auteurs (par exemple, si l'on dote les participants à un site de logos d'auteurs, des vignettes de ces logos accompagneront tous les messages de ces auteurs dans les forums de l'espace privé) ;

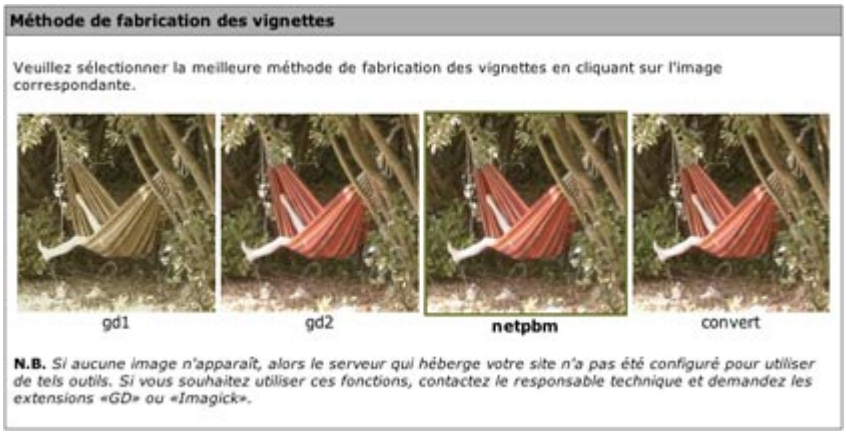
— dans les squelettes, les webmestres disposent d'une fonction image_reduire particulièrement utile pour contrôler sa mise en page, et créer différentes versions (de différentes tailles) d'une même image. Nous ne pouvons qu'encourager les webmestres à « jouer » avec cette fonction pour enrichir et contrôler leur interface graphique ; on en tirera avantage pour obtenir :

- des alignements d'images parfaits (par exemple : toutes les images de la même largeur), sans s'obliger à installer des images de dimensions prédéfinies ;
- la garantie de ne pas faire « exploser » sa mise en page lorsqu'une image trop grande est

installée par un rédacteur,
- des effets de survol et d'animation réalisés simplement en utilisant la même image à des tailles différentes (sans devoir jouer avec les « logos de survol »),
- des interfaces de portfolios (galeries de photos) épatantes...

Choix du système de traitement d'images

Mais, pour réaliser ces opérations de traitement d'images, SPIP fait appel à des systèmes qui ne peuvent pas être installés automatiquement avec SPIP, mais doivent être présents sur le serveur qui héberge votre site. Il faut donc que ces systèmes soient présents *en plus de SPIP* (dit autrement : il ne suffit pas que SPIP soit installé pour que les fonctions de traitement d'images soient disponibles ; il faut en fait que ces fonctions soient présentes par ailleurs).



Le choix d'un système de traitement d'images se fait dans la partie « Configuration » (configuration avancée) de l'espace privé. SPIP permet de choisir parmi 5 méthodes différentes de traitement des images.

Imagemagick

Imagemagick en tant qu'extension de PHP (php-imagick) est le choix privilégié par SPIP. SPIP est capable de déterminer seul sa présence. Si Imagemagick est présent sur votre serveur, alors SPIP l'utilisera automatiquement.

Si Imagemagick n'est pas présent sur votre serveur, alors SPIP vous proposera de choisir parmi d'autres méthodes. Ces méthodes n'étant pas détectables par SPIP (et, en tout cas, pas parfaitement), une vignette vous est proposée pour chaque méthode ou, éventuellement, pas de vignette si la méthode ne fonctionne pas sur votre site. Vous êtes alors invité à sélectionner votre méthode préférée (parfois : sélectionner la seule réellement disponible !).

GD, GD2

GD (et sa version 2, nettement plus puissante) est une extension de PHP désormais fréquemment présente sur les serveurs, y compris les hébergeurs mutualisés.

Si GD2 est présente, vous pouvez l'utiliser, elle donne des résultats de bonne qualité.

En revanche, GD (comprendre : « version 1 de GD ») est proposée comme pis-aller : le traitement des images se fait en 256 couleurs, et introduit de fortes dégradation des images ; elle n'est donc à sélectionner que *si aucune autre méthode ne fonctionne sur votre site*.

Imagemagick par convert

Convert est le logiciel en ligne de commande de Imagemagick. La qualité est absolument épatante, cependant son installation est relativement complexe.

Une fois *convert* installé sur votre site, vous devez configurer le chemin d'accès dans mes_options.php3 (il s'agit d'un appel en ligne de commande) par la variable suivante :

```
$convert_command = '/bin/convert';
```

Il convient ici d'indiquer le chemin complet d'accès au programme. Sous Linux, ce chemin est souvent

```
$convert_command = '/bin/convert';
```

sous MacOS X, s'il est installé avec Fink :

```
$convert_command = '/sw/bin/convert';
```

(ces valeurs sont fournies à titre indicatif ; comme tout programme, il peut être installé quasiment n'importe où...).

NetPBM

Cette méthode consiste en trois programmes, déjà anciens, qui permettent de réaliser le redimensionnement de l'image. L'avantage de cette méthode est que ces programmes peuvent être installés sans accès *root* sur la plupart des hébergements.

On trouvera, sur le site du logiciel *gallery*, une explication claire et des versions précompilées de NetPBM.

Dans SPIP, on configure le chemin d'accès à pnmscale (l'un seulement des trois programmes installés - les deux autres chemins s'en déduiront, puisque les programmes sont installés dans le même répertoire) par la variable suivante :

```
$pnmscale_command = '/bin/pnmscale';
```

(encore une fois, c'est-à-vous de déterminer le chemin d'accès réel de votre installation).

*** ***

Pour rappel, vous pouvez obtenir nombre d'informations utiles sur votre système via la page /ecrire/info.php3, notamment :

— le système utilisé (utile pour installer NetPBM précompilé) ;
— la version de PHP ;
— la présence éventuelle des extensions GD, GD2 et Imagemagick.

Enfin, en cas de difficulté, la meilleure solution consiste à contacter votre hébergeur pour qu'il installe les extensions nécessaires si aucune n'est présente. La présence d'au moins une extension graphique de PHP est désormais une norme chez les hébergeurs, n'hésitez pas à demander leur installation pour en bénéficier sur votre site.

La gestion des pages 404

Depuis [Spip 1.8] il est possible de créer facilement un squelette pour la page d'erreurs 404 qui sera affichée si l'internaute demande une page qui n'existe pas.

Page d'erreur 404 , quesaco ?

Lorsqu'une personne navigue dans le Web, il peut lui arriver d'appeler une page web qui n'existe pas. Lorsqu'un serveur web reçoit la demande pour cette page, il répond alors par une page web spécifique (pour signaler que la page demandée n'existe pas), définie par le propriétaire du site (ou par défaut celle fournie avec le serveur), et qui contient généralement un message d'explication, et est accompagnée d'un code de réponse « 404 » du protocole http (qui, seul, ne serait pas très explicite), d'où le nom de « page 404 » parfois donné à ce genre de page.

Et SPIP dans tout ça ?

Il peut également arriver qu'un utilisateur demande un fichier qui existe, mais qui, faute d'information, n'apporte pas de contenu. Pour prendre un exemple avec SPIP, si quelqu'un tape dans son navigateur un adresse comme comme http://adresse-d-un-site-spip/spip.php?article520, et que l'article 520 n'existe pas encore — ou n'est pas publié en ligne —, il serait logique que SPIP retourne un message d'erreur de la même façon que la procédure précédemment décrite. On pourrait qualifier cela de « pseudo erreur 404 ».

C'est effectivement ce que fait SPIP. Il retourne une page 404 construite à partir d'un squelette. Pour la personnaliser, il suffit donc de créer son propre fichier 404.html, comme n'importe quel squelette SPIP. Ce fichier 404.html sera donc enregistré avec les autres squelettes que vous avez créés (voir à ce sujet l'article « Où placer les fichiers de squelettes ? »).

Réglage de SPIP

SPIP ne peut pas savoir tout seul quand retourner cette « pseudo erreur 404 », il faut donc lui expliciter dans chacun des squelettes qui pourrait la générer.

Par convention, SPIP va retourner la page d'erreur 404 seulement quand le contenu retourné par les boucles du squelette est complètement vide.

Le principe est assez simple, par exemple :

- le squelette de recherche, s'il ne trouve pas de résultats, affichera la page de recherche, probablement avec un message informant l'utilisateur de l'échec de la recherche. Mais il ne retournera pas une « pseudo erreur 404 ».
- par contre, un squelette d'article, si on lui demande un article qui n'existe pas ne doit pas afficher de page article, mais la « pseudo erreur 404 ».

Pour reprendre l'exemple de l'article 520 non publié ou inexistant et de l'appel d'une l'url http://adresse-d-un-site-spip/spip.php?article520, il faudrait concevoir le squelette article de la manière suivante :

```
<BOUCLE_principale(ARTICLES) {id_article}>
code html, y compris entete
</BOUCLE_principale>
```

La boucle _principale ne retourne rien si l'on demande un article inexistant ou non publié, le résultat produit sera alors une page vide, et SPIP engendrera un message d'erreur et retournera une « pseudo erreur 404 ».

Réglage du serveur Web

Cette procédure est nécessaire pour les pages d'erreurs 404 « normales », c'est à dire dues à l'absence du fichier demandé par l'utilisateur.

La méthode la plus simple est sans doute de se servir du fichier .htaccess fournit par SPIP, même si vous ne vous servez pas des URLs propres. Pour cela, renommez le fichier htaccess.txt livré en standard en .htaccess. [1]. Et c'est tout !

Attention ! il peut arriver que votre hébergeur désactive l'usage du fichier .htaccess, auquel cas il faudra le contacter pour résoudre ce problème.

Notes

[1] il est possible que vous ne puissiez pas appeler ainsi votre fichier sur votre ordinateur. Auquel cas nommer le htacces.txt, et renommer le .htaccess lorsque vous l'aurez mis en ligne.

Insérer des formules mathématiques en LaTeX

[SPIP 1.8] introduit une puissante fonctionnalité permettant d'insérer dans les textes des formules de mathématique complexes, en utilisant la syntaxe de TeX/LaTeX.

Cette fonctionnalité permet par exemple d'afficher une formule comme celle-ci :

$$z = \left(\frac{e^{i\theta} + e^{-i\theta}}{2}\right)^2 + \left(\frac{e^{i\theta} - e^{-i\theta}}{2i}\right)^2$$

en la codant directement dans le texte, comme on peut le faire avec TeX.

> **Notez bien :** l'utilisation de cette méthode nécessite, évidemment, de connaître la syntaxe des formules dans TeX. Syntaxe qui n'est pas simple...

Des images dans le texte

Le principe technique de cette méthode consiste à transformer chacune des formules en *image*, image qui est ensuite affichée dans le texte. C'est actuellement la méthode la plus simple et efficace pour afficher des formules mathématiques complexes sur une page Web.

> Pour l'heure, l'affichage de formules mathématiques sur des pages Web grâce au standard MathML n'est absolument pas viable, l'implémentation de MathML dans les navigateurs étant totalement erratique. La solution retenue par SPIP (l'intégration d'images représentant les formules) est actuellement la seule qui garantisse que *tous* les visiteurs d'un site verront correctement les formules mathématiques.

Il est important de comprendre que, dans SPIP, *seules les formules* sont transformées. Il n'est pas question, en particulier, d'utiliser les macro-fonctions de TeX pour réaliser la mise en forme du document. Il s'agit ici d'un outil destiné à intégrer des formules mathématiques à l'intérieur d'un document codé selon les habitudes de SPIP.

Syntaxe dans SPIP

La syntaxe dans SPIP consiste à placer la partie de texte concernée par le traitement entre les pseudo-tags suivants :

```
<math>
  ...
  Ici  on extrait les formules mathématiques...
  ...
</math>
```

> Puisque seules les formules mathématiques sont traitées, on peut en réalité ajouter $...$ de manière très large (en clair : on peut ajouter $tout au début du texte, et$ tout à la fin...).
>
> La seule incompatibilité sera le cas où l'on souhaite afficher le symbole « dollar » ($) dans le texte, ce symbole étant utilisé pour délimiter les formules. (C'est la raison, en réalité, de l'existence des codes $$.)

À l'intérieur de ces pseudo-tags, on code ensuite les formules de mathématiques selon les normes de TEX, en les encadrant de dollars ($) ou de double-dollars ($$) pour les formules centrées.

Voici un exemple :

On peut insérer des matrices : $\begin{pmatrix} 1 & 0 & 0 \\ 0 & a & 0 \\ 0 & 0 & b \end{pmatrix}$; on peut placer des fractions, telles que $\frac{1}{z}$, $\frac{1}{\displaystyle 1+\frac{1}{x}}$; utiliser des lettres grecques α, β, γ, Γ, φ ; centrer des formules complexes :

$$\left|\frac{1}{N}\sum_{n=1}^{N}\gamma(u_n) - \frac{1}{2\pi}\int_0^{2\pi}\gamma(t)\mathrm{d}t\right| \le \frac{\varepsilon}{3}.$$

que l'on pourra coder ainsi :

```
On peut insérer des matrices:
$\pmatrix{1&0&0\cr 0&a&0\cr 0&0&b\cr}$;
on peut placer des fractions, telles que ${1\over z}$,
${1\over\displaystyle 1+{1\over x}}$;
utiliser des lettres grecques $\alpha$, $\beta$,
$\gamma$, $\Gamma$, $\varphi$;
centrer des formules complexes:
$$\left|{1\over N}\sum_{n=1}^N \gamma(u_n)
-{1\over 2\pi}\int_0^{2\pi}\gamma(t){\rm d}t\right|
\le {\varepsilon\over 3}.$$
```

Le système est limité à l'affichage de formules mathématiques. De ce fait, toutes les autres fonctions de T_EX sont désactivées. Parmi la plus importante interdiction : *il n'est pas possible de définir ses propres macros* (\def...{...} est désactivé) *et les macros utilisées en dehors des formules mathématiques ne seront pas reconnues*. À l'usage, on trouvera d'autres limitations, en se souvenant toujours que le but est d'intégrer des formules mathématiques dans ses textes, et rien de plus...

Pour les webmestres

Les équations sont traitées en mode client-serveur : les formules sont envoyées à un serveur, qui retourne à votre site les fichiers graphiques de ces équations. (Évidemment, les fichiers sont sauvegardés sur votre système, et l'échange n'a lieu qu'une seule fois par équation.)

Pour plus d'informations sur le système utilisé, vous pouvez consulter la page du Wiki de SPIP. Vous y trouverez notamment les explications pour monter votre propre serveur d'équations, afin de ne plus dépendre de notre serveur central.

Le calendrier de [SPIP 1.8.2]

_ SPIP 1.8.2 permet de visualiser dans l'espace public des calendriers avec le même affichage que celui de l'espace privé, et plus généralement de construire des agendas quelconques bénéficiant des outils de mises en pages de ces calendriers. Cette possibilité est fournie par un nouveau critère de boucle et trois nouveaux filtres.

Le critère agenda possède un nombre variable d'arguments et peut donc s'écrire de deux façons :

- {agenda *datetime*, *type*, *AAAA*, *MM*, *JJ*}
- {agenda *datetime*, periode, *AAAA*, *MM*, *JJ*, *AAAA2*, *MM2*, *JJ2*}

Les deux premiers arguments sont :

1. un nom de champ SQL de type *datetime* (par exemple date ou date_redac pour la table Articles et date_time pour les Breves) ; cet argument est obligatoire.
2. un *type* de calendrier (jour, semaine, mois ou periode) ; la valeur par défaut est mois.

Ces deux arguments doivent être littéraux (autrement dit ils ne peuvent être calculés dynamiquement par #ENV ou toute autre balise).

Les trois prochains arguments sont optionnels et peuvent être indiqués par des balises (avec et sans filtre) :

1. *YYYY* une chaîne d'exactement 4 chiffres indiquant une année ;
2. *MM* une chaîne d'exactement 2 chiffres indiquant un mois ;
3. *JJ* une chaîne d'exactement 2 chiffres indiquant un jour.

Si ces valeurs sont omises ou nulles, elles sont remplacées par celle de la date courante. Ces paramètres représentent la date d'un jour dans la période choisie :

- si le type est jour, on affichera les éléments dont la date correspond au jour spécifié,
- si le type est semaine, on affichera les éléments dont la date est dans la semaine qui contient le jour spécifié,
- si le type est mois, on affichera les éléments dont la date est dans le mois qui contient le mois spécifié (dans ce cas, le paramètre de jour *JJ* est facultatif).

Par exemple :

```
<B_semaine>
<ul>
<BOUCLE_semaine(ARTICLES) {agenda date, semaine} {par date}>
<li>#TITRE</li>
</BOUCLE_semaine>
</ul>
</B_semaine>
```

affiche une liste des articles publiés dans la semaine actuelle.

```
<BOUCLE_art_principale(ARTICLES) {id_article}>
<B_mememois>
<ul>
<BOUCLE_mememois(ARTICLES) {agenda date, mois, (#DATE|annee), (#DATE|mois)} {par date}>
<li>#TITRE</li>
</BOUCLE_mememois>
</ul>
</B_mememois>
</BOUCLE_art_principale>
```

affiche une liste des articles publiés le même mois que l'article actuel.

Dans le cas où l'argument *type* (le deuxième argument) vaut periode, trois autres arguments peuvent être spécifiés à la fin du critère. Alors :

- *YYYY*, *MM*, *JJ* correspondront à la date de début de la période,
- *YYYY2*, *MM2*, *JJ2* correspondront à la date de fin de la période.

Si le deuxième trio spécifiant la date de fin est omis, la date courante sera prise comme date de fin, et si le premier trio est également absent, la période de sélection couvrira toute la vie du site (pour les sites avec beaucoup d'articles, le temps d'exécution risque d'être excessif si d'autres critères n'en limitent pas le nombre).

Par exemple :

```
<BOUCLE_art_principale(ARTICLES) {id_article}>
<B_apres>
<ul>
<BOUCLE_apres(ARTICLES) {agenda date, periode, (#DATE|annee), (#DATE|mois), (#DATE|jour)} {par date}>
<li>#TITRE</li>
</BOUCLE_apres>
</ul>
</B_apres>
</BOUCLE_art_principale>
```

liste les articles qui ont été publiés après l'article actuel.

Pour mettre en page les éléments sélectionnés par une boucle (en particulier une boucle avec un critère **agenda**) sous forme de calendrier, SPIP fournit trois filtres. Ces filtres fournissent un affichage similaire au calendrier de l'espace privé avec le même système de navigation.

- Le filtre agenda_memo s'applique sur une balise de date (par exemple #DATE ou #DATE_MODIF) et prend quatre paramètres :

  1. un descriptif
  2. un titre
  3. une URL représentant l'élément ayant ce titre et ce descriptif (par exemple #URL_ARTICLE)
  4. un nom de *classe* CSS

Si la balise sur laquelle agenda_memo s'applique n'est pas nulle, il se contente de mémoriser la date et les trois premiers arguments dans un tableau indexé par le dernier argument (le nom de classe CSS) et ne retourne rien (aucun affichage).

L'utilisation du dernier argument comme index pour l'élément à mémoriser permet d'avoir plusieurs calendriers par page. De plus, la classe spécifiée ici sera attribuée à cet élément dans l'affichage en calendrier fourni par **agenda_affiche**. Ainsi, on peut donner des styles différents aux éléments. La feuille *calendrier.css* fournit 28 styles différents qui donnent un exemple de différents styles de calendrier.

- Le filtre agenda_affiche s'applique sur une balise retournant le nombre d'éléments à afficher (en général #TOTAL_BOUCLE) et prend trois paramètres :

  1. un texte qui sera affiché si la balise filtrée ne retourne rien (0 élément à afficher) ;
  2. un type de calendrier (jour, semaine, mois ou periode) ;
  3. des noms de *classes* CSS utilisées dans l'appel du filtre précédent qui permettent de filtrer les éléments à afficher.

Si la balise filtrée est nulle, ce filtre retourne son premier argument. Sinon il retourne les éléments mémorisés par le filtre **agenda_memo** mis en page dans un calendrier du type demandé.

Seul les éléments indexés par **agenda_memo** avec une des classes CSS indiquées dans le dernier argument

seront affichés (ou alors tous les éléments si ce paramètre est omis). Ainsi on peut filtrer les éléments pour les répartir dans plusieurs calendriers sur la même page.

Le type periode restreindra l'affichage à la période comprise entre le plus vieil élément et le plus récent effectivement trouvés, ce qui permet de donner dans le critère une période très large sans récupérer un affichage trop long.

Exemple :

```
<BOUCLE_memorise(ARTICLES) {agenda date, semaine}{par date}>[
(#DATE|agenda_memo{#DESCRIPTIF,#TITRE,#URL_ARTICLE})
]</BOUCLE_memorise>
[(#TOTAL_BOUCLE|agenda_affiche{<:aucun_article:>,semaine})]
<//B_memorise>
```

affiche les articles publiés dans la semaine actuelle sous forme de calendrier.

- Enfin, le filtre agenda_connu teste si son argument est l'un des quatre types de calendriers connus (jour, semaine, mois ou periode).

Ce critère et ces filtres sont utilisés par les nouveaux squelettes agenda_jour.html, agenda_semaine.html, agenda_mois.html et agenda_periode.html, appelés à partir du squelette agenda.html qui indique dans son en-tête les feuilles de style et fonctions javascript nécessaires (mais remplaçables à volonté). Ces squelettes fournissent donc un exemple représentatif d'utilisation.

Images typographiques

Titres graphiques avec la police de son choix

[SPIP 1.9, GD2 et Freetype]

Si GD2 et Freetype sont installés sur votre site [1], SPIP peut fabriquer, de lui-même, des images à partir de titres (ou tout élément de la base de donnée) en utilisant une police de caractères de votre choix.

La fonction qui réalise cet effet est image_typo. La présente page va vous présenter les différentes variables que l'on peut utiliser avec cette fonction.

```
[(#TITRE|image_typo)]
```

Si vous ne précisez aucune variable, image_typo va utiliser la police par défaut fournie avec SPIP : **Dustismo,** une police GPL de Dustin Norlander.

police

Vous pouvez préciser le nom d'une autre police, que vous aurez pris soin d'installer sur votre site.

```
[(#TITRE|image_typo{police=dustismo_bold.ttf})]
```

(**Dustismo-bold** est également livré avec SPIP.)

En théorie, vous pouvez utiliser de nombreux formats de police : TrueType, PostScript Type 1, OpenType... Selon la configuration de votre site, il est possible que certains formats ne soient pas acceptés.

Les polices doivent être installées dans un sous-dossier /polices du dossier /ecrire, ou du dossier contenant vos squelettes.

Si nous installons, par exemple, un fichier TrueType ainsi :
/ecrire/polices/stencil.ttf
il est possible d'utiliser cette nouvelle police [2].

```
[(#TITRE|image_typo{police=stencil.ttf})]
```

taille

On peut préciser la taille d'affichage de la police. Cela s'utilise avec la variable taille.

```
[(#TITRE|image_typo{police=stencil.ttf,taille=36})]
```

Note : on ne précise pas « 36pt », on indique seulement « 36 », sans indication de l'unité.

couleur

Cette variable permet d'indiquer la couleur. Par défaut, le rendu est noir. Cette variable est une couleur RVB hexadécimale, toujours de la forme « 3399BB ». Notez : on omet le « # » qui précède habituellement ce type de code couleur.

```
[(#TITRE|image_typo{police=stencil.ttf,taille=36,couleur=4433bb})]
```

largeur

La variable largeur permet de fixer la largeur *maximale* de l'image. Notez bien : c'est une valeur maximale ; l'image réelle est « recadrée » automatiquement, ensuite, pour adopter les dimensions du texte réellement composé.

Le premier pavé ci-dessous est composé avec une largeur maximale de 300 pixels, le second avec une largeur de 400 pixels.

```
[(#TITRE|image_typo{police=stencil.ttf,largeur=300})]
[(#TITRE|image_typo{police=stencil.ttf,largeur=400})]
```

align

La variable align permet de forcer l'alignement de plusieurs lignes de texte (lorsque c'est le cas) à gauche, droite, ou au centre. Exceptionnellement, on utilise ici une syntaxe anglaise, proche de ce qui se fait pour les feuilles de style.

```
[(#TITRE|image_typo{police=stencil.ttf,align=left})]
[(#TITRE|image_typo{police=stencil.ttf,align=center})]
[(#TITRE|image_typo{police=stencil.ttf,align=right})]
```

hauteur_ligne

hauteur_ligne permet de fixer la hauteur entre chaque ligne de texte (dans le cas où l'image comporte plusieurs lignes).

```
[(#TITRE|image_typo{police=stencil.ttf,taille=36,hauteur_ligne=80})]
```

padding

Certaines polices « dépassent » de leur boîte de rendu, et on obtient un effet désastreux (polices « coupées »). La variable padding permet, exceptionnellement, de forcer un espace supplémentaire *autour* du rendu typographique.

```
[(#TITRE|image_typo{police=stencil.ttf,padding=5})]
```

Filtrer l'image

Le résultat de image_typo étant une image, il est tout à fait possible de lui appliquer des filtres d'images. Par exemple, ci-après, on rend l'image semi-transparente, ou on lui applique une texture.

```
[(#TITRE|image_typo{police=stencil.ttf,couleur=aa2244}|image_alpha{60})]
[(#TITRE|image_typo{police=stencil.ttf,couleur=aa2244}|image_masque{carre-mur.png})]
```

P.-S.

N.B.1. L'image créée par image_typo est au format PNG 24 avec une couche alpha pour réaliser la transparence. Pour forcer Microsoft Explorer à afficher correctement cette transparence, SPIP utilise une classe de feuille de style spécifique, format_png, définie dans spip_style.css ; celle-ci appelle un « comportement » (*behavior*) rendant l'affichage possible sous MSIE. On a donc, encore une fois, tout intérêt à intégrer le spip_style.css standard dans ses propres squelettes, quitte à le surcharger avec ses propres styles.

N.B.2. L'affichage de certaines polices (notamment les anglaises et certaines italiques) est problématique. Les techniques de rendu typographique dans GD2 sont, visiblement, encore en développement (nous rencontrons bugs sur bugs de ce côté). Espérons que les fonctions GD2 progresseront rapidement.

N.B.3. De l'arabe, du farsi, de l'hébreu ? Malheureusement : non ! Nous rencontrons pour l'heure deux difficultés qui ne permettent pas de proposer de solution « propre » pour l'affichage de l'hébreu et de l'arabe.

> — Tout d'abord, la gestion de l'affichage bidirectionnel n'est pas assuré par GD2 ; il n'est donc pas possible pour l'instant de créer des images typographiques pour des chaînes s'écrivant de droite à gauche.
> — Pour l'arabe (et le farsi), les ligatures ne sont pas gérées. En particulier : les ligatures d'OpenType sont purement et simplement ignorées.

Notes

[1] GD2 est une extension *graphique* de PHP, qui permet de nombreuses manipulations d'images. Freetype, habituellement installé avec GD2, est l'extension qui insère du texte dans une image à partir d'un fichier de police, Truetype ou Postscript. En cas de doute, demandez à votre hébergeur si GD2 est installé.

[2] Attention : si vous ne protégez pas ce dossier (avec un htaccess par exemple), votre fichier de police sera accessible par le Web. Si vous utilisez des polices commerciales, faites attention à ne pas vous retrouver, ainsi, à diffuser des polices pour lesquelles cela n'est pas autorisé.

Couleurs automatiques

SPIP 1.9

[SPIP 1.9 et GD2]

SPIP permet d'extraire automatiquement une couleur d'une image, afin de l'appliquer à d'autres éléments d'interface.

Par exemple, à côté du logo d'un article, nous allons afficher le titre du même article *dans une couleur* tirée de ce logo. De cette façon, tout en rendant l'affichage plus varié (d'un article à l'autre, la couleur utilisée change en fonction du logo), le fait que la couleur soit extraite de l'image assure une certaine cohérence graphique.

Cette fonction consistant à récupérer une couleur dans une image est complétée par toute une série de fonctions permettant de manipuler cette couleur, principalement éclaircir et foncer la couleur. La liste de fonctions est longue, de façon à permettre un nombre très important d'effets.

couleur_extraire

À partir d'une image (logo d'article, logo de rubrique..., mais aussi images de portfolio), on demande à SPIP de tirer une couleur.

```
[(#LOGO_RUBRIQUE||couleur_extraire)]
```

Attention : il ne s'agit pas pour SPIP de déterminer la couleur *dominante* de l'image, mais d'extraire une couleur de l'image. Pour que cette couleur soit réellement « représentative », l'image est réduite à une taille de 20 pixels maximum ; ainsi les différentes couleurs de l'image sont relativement « moyennées ». Cette valeur de 20 pixels est expérimentale : elle est suffisamment basse pour éviter d'extraire une couleur très peu présente dans l'image ; elle est suffisamment élevée pour éviter que la couleur soit systématiquement grisâtre.

Utilisée sans paramètres, la fonction couleur_extraire retourne une couleur située légèrement au-dessus du centre de l'image. Il est possible d'indiquer un point préféré pour le sondage, en passant deux valeurs (*x* et *y*) comprises entre 0 et 20 (conseil : entre 1 et 19 pour éviter les effets de marges).

Par exemple :

```
[(#LOGO_RUBRIQUE||couleur_extraire{15,5})]
```

retourne une couleur située en haut à droite du centre de l'image.

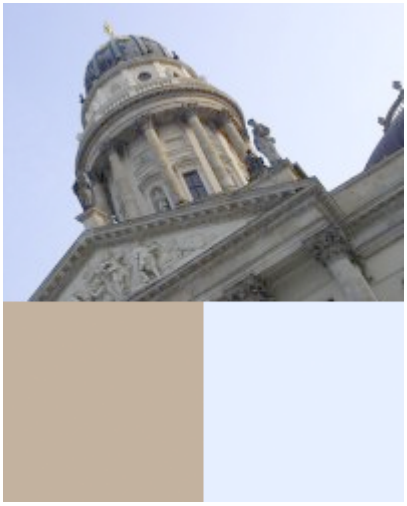
Pour bien comprendre le principe, appliquons ce filtre sur un logo de couleur uniforme :

Le résultat est : ff9200.

Notez bien : les valeurs retournées sont systématiquement en codage RVB hexadécimal, en omettant le « # » qui précède habituellement ces codes. On pensera donc à insérer ce dièse quand on utilise ces valeurs.

On peut, par exemple, appliquer cette couleur au fond d'un pavé :

```
<div style="background-color: #[(#LOGO_RUBRIQUE||couleur_extraire)]; width: 100px; height: 100px;"></div>
```



Appliquons ce filtre à une photographie :

En bas à gauche, la couleur extraire sans paramètres, c'est-à-dire présente un peu au dessus du centre de l'image (marron clair de la pierre du bâtiment). En bas à droite, on force une couleur située en haut à droite de l'image (bleu clair du ciel).

```
[(#LOGO_RUBRIQUE||couleur_extraire)]
[(#LOGO_RUBRIQUE||couleur_extraire{15,5})]
```

L'utilisation de ce filtre est, techniquement, très simple. En revanche, créativité et inventivité seront nécessaires pour l'exploiter... Voici quelques utilisations :

— couleur de texte ;
— couleur de fond d'un pavé ;
— couleur d'une image typographique ;
— modifier la couleur d'une image (image_sepia)...

Modifier la couleur

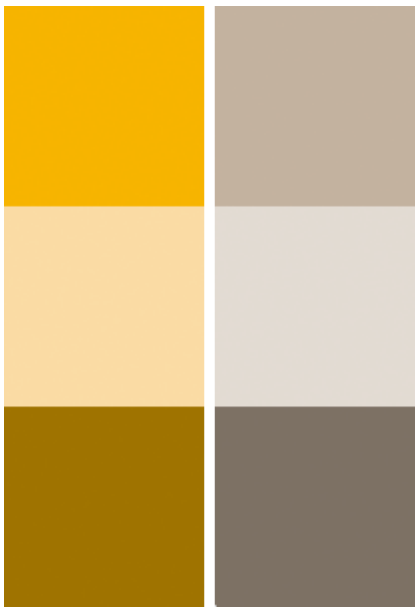
Une fois la couleur extraite, il est utile de la manipuler, afin de jouer avec différentes variantes de la couleur, tout en respectant la cohérence graphique.

- **couleur_foncer**, **couleur_eclaircir**

À partir de la couleur extraite d'une image, nous souhaitons afficher des couleurs plus foncées et plus claires.

```
[(#LOGO_RUBRIQUE||couleur_extraire)]
[(#LOGO_RUBRIQUE||couleur_extraire|couleur_foncer)]
[(#LOGO_RUBRIQUE||couleur_extraire|couleur_eclaircir)]
```

Appliqué aux couleurs extraites des exemples précédents, cela donne :



On constate qu'on a ainsi des camaïeux de couleurs faciles à obtenir, l'ensemble étant très cohérent.

- **couleur_foncer_si_claire**, **couleur_eclaircir_si_foncee**

Si nous appliquons la couleur extraite au fond d'un pavé de texte, il faut déterminer dans quelle couleur nous voulons écrire ce texte (par exemple : noir sur orange ou blanc sur orange ?) ; c'est ce que nous verrons avec les fonctions suivantes.

Pour l'instant, nous décidons que le texte sera d'une certaine couleur. Nous voulons par exemple que le texte soit noir. Il faut donc choisir la couleur du fond en fonction de ce texte noir : il faut que la couleur du fond soit claire (donc : texte noir sur fond clair).

Si nous appliquons le filtre couleur_eclaircir à notre couleur extraite, nous avons deux cas :
— si la couleur est foncée, alors elle est éclaircie et nous obtenons l'effet voulu ;
— si la couleur est déjà claire, alors nous l'éclaircissons encore, et nous obtenons un fond qui peut devenir quasiment blanc. Or, la couleur étant déjà claire , nous aurions voulu l'utiliser telle quelle.

C'est ici que nous appliquons le filtre couleur_eclaircir_si_foncee :
— si la couleur est foncée, nous l'éclaircissons ;
— si la couleur est claire, nous l'utilisons telle quelle.

Le filtre couleur_foncer_si_claire a la logique exactement inverse. Il est très utile, par exemple, pour écrire en blanc sur un fond systématiquement foncé, mais en évitant de rendre ce fond quasiment noir quand la couleur d'origine est déjà foncée.

— **couleur_extreme**, **couleur_inverser**

Le filtre couleur_extreme passe une couleur foncée en noir, et une couleur claire en blanc. Cela est utile pour écrire en noir ou blanc sur un fond coloré.

En réalité, récupérer la couleur « extrême » est habituellement utilisé avec couleur_inverser, il inverse la couleur RVB. Elle transforme notamment du noir en blanc, et du blanc en noir.

En pratique, cela permet d'assurer un bon contraste, quelle que soit la couleur du fond du bloc (alors que, dans l'exemple précédent, nous choisissions la couleur du fond du bloc en fonction d'une couleur de texte).

Appliquons, en couleur de fond, la couleur extraite de l'image :

```
<div style="background-color: #[(#LOGO_ARTICLE||couleur_extraire)];"> ... </div>
```

On obtient donc, selon le logo de l’article, soit un fond foncé, soit un fond clair.

Appliquons, pour la couleur du texte, la couleur extraite, rendue « extrême » :

```
[(#LOGO_ARTICLE||couleur_extraire|couleur_extreme)]
```

— Si la couleur est foncée, la couleur extrême est noire ; nous écrivons en noir sur fond foncé.
— Si la couleur est claire, la couleur extrême est blanche ; nous écrivons en blanc sur fond clair.

Dans les deux cas, c’est peu lisible. On pourra utiliser cette couleur pour un autre effet (par exemple : une bordure autour du div).

Il nous reste à inverser cette couleur pour l’appliquer au texte ;

```
<div
 style="color: #[(#LOGO_ARTICLE||couleur_extraire|couleur_extreme|couleur_inverser)];
  background-color: #[(#LOGO_ARTICLE||couleur_extraire)];">
   ...
</div>
```

— Si la couleur extraite est foncée, la couleur extrême est noire, et l’inverse est alors blanche. On écrit en blanc sur fond foncé.
— Si la couleur extraite est claire, la couleur extrême est blanche, et l’inverse est noire. On écrit en blanc sur fond clair.

Dans les deux cas, le contraste assure une bonne lisibilité.

Traitement automatisé des images

[SPIP 1.9 et GD2]

SPIP permet de faire subir aux images des effets automatisés. Ces effets ont deux vocations :

— tout simplement assurer la cohérence graphique du site, en fabriquant automatiquement des éléments de navigation qui seront toujours réalisés selon les désirs du graphiste ;

— créer des effets relativement spectaculaires, sans pour autant demander aux auteurs des articles de traiter les images eux-mêmes, et sans non plus interdire les évolutions graphiques du site par la suite.

Par exemple : on veut, dans l'interface graphique du site public, que les logos de navigation des articles aient deux aspects :

— dans tous les cas, ils sont « posés sur le sol », avec un reflet sous eux ;

— au repos, ils sont en noir et blanc, assez foncés ; survolés, ils sont en couleur.

Sans les automatismes qui suivent, les webmestres ont pris l'habitude de créer, à la main, deux versions de ces images, et d'installer deux logos sur le site. Deux inconvénients (particulièrement gênants) :

— la manipulation est longue ; par ailleurs elle ne peut pas être confiée à un tiers, qui serait de toute façon incapable de la réaliser correctement ;

— lorsqu'on voudra faire évoluer l'interface graphique du site, on sera bloqué avec ces logos très typés.

Avec les automatismes qui suivent, on peut travailler autrement : les auteurs installent un simple logo d'article (par exemple, une photographie), sans aucun traitement spécifique ; et, automatiquement, les squelettes de SPIP fabriquent :

— une vignette à la bonne taille ;

— la même vignette en noir et blanc, légèrement foncée ;

— les reflets de l'image sur le sol.

Certains des filtres présentés utilisent les fonctions de redimensionnement des images. Il est impératif, après l'installation, de se rendre dans l'espace privé, « Configuration », puis dans l'onglet « Fonctions avancées » : là, sélectionnez « GD2 » pour la « Méthode de fabrication des vignettes ».

Avertissement lenteur

Avant de commencer, signalons que ces fonctions sont *lourdes*. Voire très lourdes si vous utilisez de grosses images. Le calcul d'une image est relativement long, et s'il faut calculer, dans vos squelettes, plusieurs images, voire plusieurs effets pour chaque image, vous risquez d'obtenir des erreurs de timeout (temps maximum d'exécution des scripts dépassé).

Cela dit, il est important de noter que les filtres de traitement des images ont leur propre système de cache : *une fois calculée, une image « filtrée » est sauvegardée, et ne sera plus recalculée*. La charge sur le serveur est donc limitée au premier calcul.

Cet avertissement concerne, en priorité, les hébergeurs mutualisés.

La page « Vider le cache » de l'espace privé affiche la taille utilisée par SPIP pour stocker ces images calculées. On peut ainsi vider ce cache indépendamment du cache des squelettes et des pages HTML.

Transparences

Dans [SPIP 1.9], si l'on utilise GD2, outre les fonctions exposées dans cet article, vous constaterez que les réductions d'image (image_reduire) respectent la transparence des fichiers GIF et PNG 24 (transparence par couche alpha).

Dans la configuration du site, pensez à sélectionner GD2 comme méthode de réduction d'image.

L'image d'origine

Toute image gérée par SPIP peut se voir appliquer les filtres suivants. Sont donc concernés les logos (d'articles, de rubriques...), mais aussi les images de portfolio (images en tant que fichiers joints aux articles), sans oublier les nouvelles images typographiques.

Voici, pour nos exemples, une image à traiter.

Réduire les dimensions d’une image

La fonction reduire_image devient image_reduire, pour adopter la même logique de noms que les nouvelles fonctions graphiques. L’ancienne dénomination est toujours fonctionnelle.

Elle est désormais complétée par une image_reduire_par, qui permet de réduire une image selon une certaine échelle. Là où reduire_image réduit une image à des dimensions fixées à l’avance, image_reduire_par réduit l’image proportionnellement.

Par exemple :

```
[(#TITRE|image_typo{taille=24}|image_reduire_par{2})]
```

réduit l’image typographique d’un facteur 2 (l’image devient deux fois plus petite).

Recadrer une image

La fonction image_recadre{largeur,hauteur,position} permet de recadrer une image (équivalent du crop des logiciels de traitement d’image) avec les combinaisons de left/center/right et top/center/bottom pour la position (ex ’left center’).

Par exemple :

```
[(#FICHIER|image_recadre{90,90,center})]
```

recadre l’image originale en un carré de 90 px de largeur et hauteur ne gardant en se basant sur le centre de l’image

Supprimer la transparence et forcer le format de l’image

La fonction image_aplatir réalise deux opérations :

— elle sauvegarde une image dans un format prédéfini (par exemple, transformer une image PNG en une image GIF) ;
— elle supprime les informations de transparence, et remplace les zones transparentes par une couleur.

Par exemple :

```
[(#TITRE
    |image_typo{police=stencil.ttf,couleur=000000,taille=40}
    |image_aplatir{gif,ff0000})]
```

Le titre transformé en image typographique est un fichier PNG avec des zones transparentes. En passant cette image par le filtre image_aplatir, on la transforme en GIF, en remplaçant les zones transparentes par du rouge (ff0000).

image_nb

Le filtre image_nb passe une image en niveaux de gris (ce qu’on appelle « noir et blanc » lorsqu’on évoque des photographies).

Sans paramètres (image de gauche), le filtre calcule les niveaux de gris en pondérant les composantes de l'image d'origine ainsi :
luminosité = 0,299 × rouge + 0,587 × vert + 0,114 × bleu.

On peut forcer la pondération des trois composantes RVB en passant les valeurs en pour-mille. Par exemple (image de droite) :

```
[(#FICHIER|image_nb{330,330,330})]
```

On a pris chaque composante R, V et B à niveau égal.

image_sepia

Le filtre image_sepia applique un filtre « Sépia ». Appliqué à une photographie, ce genre d'effet donne une tonalité de vieille photographie.

Sans paramètres (image de gauche), la valeur sépia est, par défaut, « 896f5e » (en RVB hexadécimal). On peut passer la valeur de la couleur de sépia en paramètre. Par exemple (image de droite) :

```
[(#FICHIER|image_sepia{ff0033})]
```

image_gamma

Le filtre image_gamma change la luminosité d'une image. Il rend une image plus claire ou plus foncée. Son paramètre est compris entre -254 et 254. Les valeurs supérieures à zéro rendent l'image plus claire (254 rend toute image entièrement blanche) ; les valeurs négatives rendent l'image plus foncée (-254 rend l'image complètement noire).

```
[(#FICHIER|image_gamma{70})]
[(#FICHIER|image_gamma{150})]
[(#FICHIER|image_gamma{254})]
[(#FICHIER|image_gamma{-70})]
[(#FICHIER|image_gamma{-150})]
[(#FICHIER|image_gamma{-254})]
```

image_alpha

Le filtre image_alpha rend l'image semi-transparente, en PNG 24 avec couche alpha. Si l'image était déjà semi-transparente, les deux informations sont mélangées.


```
[(#FICHIER|image_alpha{50})]
[(#FICHIER|image_alpha{90})]
```

Le paramètre est une valeur entre 0 et 127 : 0 laisse l'image inchangée (aucune transparence), 127 rend l'image complètement transparente.

image_flou

Le filtre image_flou rend l'image... floue. On peut lui passer en paramètre un nombre compris entre 1 et 11, définissant l'intensité du floutage (de 1 pixel de floutage à 11 pixels de floutage).

```
[(#FICHIER|image_flou)]
[(#FICHIER|image_flou{6})]
```

Sans paramètre, la valeur de floutage est 3.

Attention : ce filtre est particulièrement lourd (c'est-à-dire nécessite beaucoup de puissance). Plutôt que de tenter un floutage important, on peut préférer flouter plusieurs fois avec des valeurs faibles. Par exemple, remplacer :

```
[(#FICHIER|image_flou{6})]
```

par :

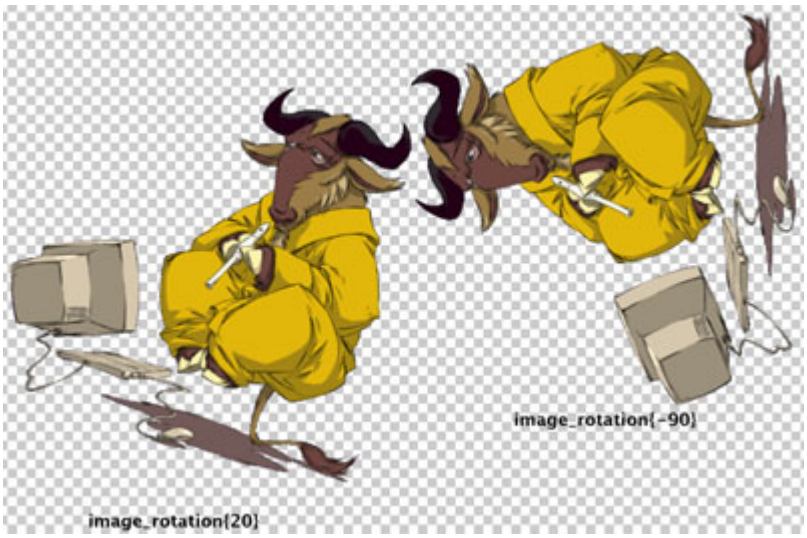
```
[(#FICHIER|image_flou|image_flou)]
```

Au pire, le calcul se fera en deux « recalcul » de squelette, le premier floutage étant sauvegardé en cache.

Attention (2) : ce filtre agrandit l'image, en ajoutant tout autour de l'image une « marge » équivalente à la valeur de floutage. Ainsi, avec le paramètre « 3 » (par défaut), on ajoute 3 pixels de chaque côté de l'image, et le résultat aura donc 6 pixels de large et de haut de plus que l'image d'origine.

image_rotation

Le filtre image_rotation fait tourner l'image d'un angle égal au paramètre passé. Les valeurs positives sont dans le sens des aiguilles d'une montre.




```
[(#FICHIER|image_rotation{20})]
[(#FICHIER|image_rotation{-90})]
```

Sauf pour les rotations à angle droit, la rotation provoque un effet d'escalier. Nous avons tenté de le limiter, mais il reste toujours présent. Une solution pour réduire cet effet consiste à réduire l'image après avoir appliqué la rotation.

Attention : ce filtre est relativement lourd !

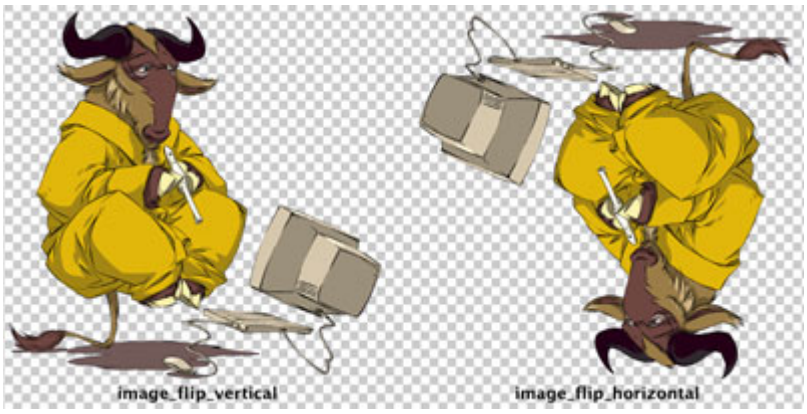
Attention (2) : ce filtre modifie les dimensions de l'image.

image_flip_vertical et image_flip_horizontal

Le filtre image_flip_vertical applique un effet de « miroir » selon un axe vertical ; image_flip_horizontal selon un axe horizontal.

Très simple d'utilisation, il n'y a pas de paramètre.

```
[(#FICHIER|image_flip_vertical)]
[(#FICHIER|image_flip_horizontal)]
```



image_masque

image_masque est le filtre le plus puissant de cette série. De fait, sa logique est nettement plus complexe que les autres filtres. Il permet, à partir d'un fichier PNG 24 en niveaux de gris et avec couche de transparence alpha, de modifier :

— le cadrage de l'image ;
— la transparence d'une image ;
— la luminosité d'une image.

- **Dimensions de l'image**

Si l'image d'origine est plus grande que le fichier masque, alors l'image d'origine est réduite et découpée au format du masque, puis on applique les informations de transparence et de luminosité du masque. Utile pour créer les vignettes de navigation.

Si l'image d'origine est plus petite que le masque, alors on ne recadre pas, on applique simplement les informations de luminosité et de transparence du masque (lui-même non redimensionné).

Voici notre image d'origine :

- **Masque de transparence**

Les informations de transparence du masque sont directement appliquées à l'image d'origine. Un pixel transparent du masque rend le pixel correspondant de l'image d'origine transparent, selon la même valeur de transparence. (Si l'image d'origine est déjà transparente, les informations sont mélangées de façon à conserver les deux infos de transparence.)

Si on a le fichier masque suivant, nommé « decoupe1.png » :

qu'on applique ainsi :

```
[(#FICHIER|image_masque{decoupe1.png})]
```

on obtient l'image suivante :

L'image d'origine a été redimensionnée aux dimensions de « decoupe1.png », et les zones transparentes du masque sont devenues les zones transparentes du résultat.

Attention ! Le filtre utilise la réduction d'image. Pour qu'il fonctionne correctement, il est

impératif de choisir la méthode de réduction GD2 dans la configuration du site, onglet « Fonctions avancées ».

- **Masque de luminosité**

Dans l'exemple ci-dessus, l'image masque est entièrement en gris à 50%. Les couleurs de l'image d'origine sont alors laissées inchangées (on s'est contenté de « découper » l'image.

En faisant varier les couleurs du masque, on va appliquer ces différences de luminosité à l'image traitée. Lorsqu'un pixel du masque est plus clair, alors le fichier résultant est éclairci ; si le pixel du masque est foncé, alors on fonce le fichier résultant.

Par exemple, si notre masque est « decoupe2.png » :

```
[(#FICHIER|image_masque{decoupe2.png})]
```

on obtient l'image suivante :

Dans ces deux exemples, le masque est plus petit que l'image d'origine, on obtient donc une sorte de vignette de l'image. Si le masque est plus grand que l'image d'origine, on l'applique à l'image non redimensionnée. Cela est pratique pour « texturer » une image. On peut ainsi réaliser l'effet suivant : en appliquant un masque en niveau de gris, masque que nous avons créé plus grand que l'image d'origine.

Attention : l'impact sur la luminosité est plus important sur l'image finale que dans le fichier masque.

Attention (2) : en réalité, le masque de luminosité est un masque de coloration. Si l'image masque est colorée, alors on modifiera non seulement la luminosité, mais aussi les couleurs de l'image. Mais cet effet est particulièrement difficile à maîtriser, notamment en partant d'images en couleur.

P.-S.

Sur son site Paris—Beyrouth, ARNO* donne de nombreux exemples d'utilisation de ces filtres graphiques, et explique notamment la philosophie qui les sous-tend : « Pourquoi utiliser les filtres graphiques de SPIP 1.9 ? ».

La syndication de contenus

Avec SPIP 1.9, le système de syndication (voir l'article « Les fichiers backend ») s'enrichit : il est désormais possible d'échanger l'adresse des documents joints aux articles (*podcasting*), de transporter d'un site à l'autre les mots-clés (*tags*) associés aux articles ainsi que leur rubrique (ou *catégorie*). On peut aussi, si on le désire, syndiquer le *contenu intégral* des articles.

Dans tout ce qui suit, on considère que le flux de syndication offert par le site source est suffisamment riche pour avoir prévu toutes les possibilités qu'offre notamment le squelette *dist/backend.html* de SPIP.

Référencement rapide du site

On repère d'abord sur le site source l'URL de son flux de syndication au format (ou Atom). Selon les cas, cette adresse est indiquée directement sur la page, et/ou est « découverte » automatiquement par le navigateur, qui affiche alors une icône caractéristique. Si les squelettes du site source le prévoient, SPIP 1.9 peut aussi découvrir cette adresse de syndication, et il suffit d'indiquer l'adresse du site pour se voir proposer de le syndiquer.

Une fois la syndication activée, les articles présents dans le flux de syndication du site source sont repris sur le site récepteur.

Décider ce que l'on veut émettre

Le webmestre du site source peut décider de ce qu'il met dans son flux RSS : ce choix peut bien évidemment se faire en modifiant le squelette *dist/backend.html*, ou en s'en inspirant pour créer un nouveau flux.

Mais la page de configuration dans l'espace privé propose une option très importante pour la syndication : elle permet de décider si le flux RSS du site comportera l'intégralité du contenu des articles, au format HTML, ou seulement un petit résumé (au format texte). Dans le premier cas (qui est la configuration par défaut du système), les articles récents du site sont entièrement lisibles avec un lecteur RSS : ils sont aussi entièrement « recopiables » d'un site à l'autre. Cela permet par exemple de réaliser automatiquement des sites miroirs, ou des sites composés d'un contenu produit sur d'autres sites.

Décider ce que l'on veut récupérer

Si le site source ne diffuse pas son contenu intégral, il est bien évident que la syndication ne pourra pas en récupérer plus. Mais dans le cas d'une diffusion intégrale, SPIP peut récupérer ce contenu HTML et l'afficher, images comprises.

Site par site, on peut choisir ce que l'on veut récupérer par syndication : le contenu HTML des articles (si disponible) ou un simple résumé au format texte.

On peut également décider de ce que l'on fait des articles qui disparaissent des flux (qui sont en général limités aux 15 articles les plus récents du site) : on peut décider que SPIP les élimine de la base de données (après une période de deux mois), et/ou les passe immédiatement en mode « refusé ». Ces dernières options permettent par exemple de gérer un portail sur des flux très rapides (agences de presse, tags très populaires de sites de photographie, etc) ; ou encore de maintenir un miroir fidèle d'un site (en éliminant les articles qui seraient dépubliés).

Décider ce que l'on veut afficher

- **la *source* :**

De manière habituelle, #NOM_SITE représente le nom du site syndiqué, qui est donc la « source » de l'article. Cependant avec le développement des agrégateurs de contenu (les portails par exemple), la véritable source de l'article peut être un autre site. Le format RSS a prévu ce cas en définissant un champ <source> indiquant la véritable source de l'article. Le moteur de syndication de SPIP récupère ces données quand elles sont présentes, et les balises #SOURCE et #URL_SOURCE permettent d'afficher respectivement le nom et l'adresse de la source.

- **les *tags* :**

Si les mots-clés affectés aux articles sont correctement renseignés dans le flux RSS, ils sont récupérés par le site récepteur ; ils n'arrivent pas individuellement dans la table des mots-clés, mais sont stockés en vrac dans le champ tags de la table spip_syndic_articles.

Si le flux de syndication utilise la notation standard <dc:subject>Tag</dc:subject>, le tag est noté tel quel. SPIP pousse cette notion un peu plus loin en transmettant aussi l'adresse de la page du mot-clé, grâce aux microformats. Le tag est alors récupéré avec son lien, sous la forme : Mot-clé.

Sur le site destinataire, ces tags sont affichés dans l'espace privé sous le descriptif de l'article, et sont affichables sur le site public grâce à la balise #TAGS (on verra plus loin comment la filtrer pour faire un affichage sélectif des tags).

Note : SPIP prévoit une gestion spécifique des tags en provenance des sites del.icio.us (bookmarks collectifs), flickr (photographie) et connotea (annotation d'articles scientifiques), de manière à pouvoir leur attribuer un URL (qui n'est pas prévu dans leurs flux RSS respectifs).

- **la rubrique :**

Dans de nombreuses applications (systèmes de blogs, répertoire de liens...) la catégorie (ou encore *directory*) d'un article est l'équivalent de ce que SPIP appelle la *rubrique*. Il était donc naturel d'utiliser la notion RSS standard de <category>...</category> pour faire connaître l'appartenance de notre article à telle ou telle rubrique.

De même qu'avec les tags, SPIP récupère cette information et l'affiche via #TAGS.

- **les documents joints :**

Les documents qui, dans l'espace privé, figurent en bas de la page de visualisation d'un article, soit dans la section « Documents » soit, pour les images, dans la section « Portfolio », sont eux aussi transmis dans le flux RSS, en utilisant la notion <enclosure ... />. C'est ce qu'on appelle le *podcasting*, désormais une fonction standard de SPIP [1].

L'information sur les *enclosures* est elle aussi récupérée dans la balise #TAGS, mais elle est affichée de façon différenciée dans l'espace privée, sous forme d'un petit trombone pour chaque fichier.

Utiliser la balise #TAGS

Comme on l'a vu ci-dessus, la balise #TAGS affiche en vrac les liens vers les mots-clés, la rubrique et les documents joints. Mais par la grâce des microformats, les liens vers chacun de ces concepts sont « marqués » de la façon suivante :

- <a rel="tag" ...> pour les tags/mots-clés
- <a rel="directory" ...> pour la rubrique/catégorie
- <a rel="enclosure" ...> pour les documents joints/podcast

Si l'on veut n'afficher que l'un de ces types de tags, on utilisera le filtre afficher_tags en précisant le type souhaité en argument :

```
[(#TAGS|afficher_tags{directory})]
```

```
[(#TAGS|afficher_tags{tag})]
```

```
[(#TAGS|afficher_tags{enclosure})]
```

(Par défaut [(#TAGS|afficher_tags)] est équivalent à [(#TAGS|afficher_tags{'tag,directory'})].)

Pour les documents joints, le filtre spécifique afficher_enclosures permet d'afficher les trombones plutôt que des liens classiques :

```
[(#TAGS|afficher_enclosures)]
```

* * *

Manipuler le contenu HTML des articles syndiqués

Cas pratique : notre site syndique un photoblog, qui diffuse systématiquement un petit commentaire suivi de la photographie. Cette dernière se présente sous la forme d'une balise HTML . Une fois ce photoblog syndiqué en version HTML complète dans notre site, nous pouvons décider de n'afficher que la photo, sans le commentaire. Il nous faut alors extraire la balise <img /> ; cela peut se réaliser grâce au filtre extraire_balise{xxx}, qui récupère la première balise HTML <xxx /> d'un contenu.

A partir de là tout est possible :

- [(#DESCRIPTIF|extraire_balise{img})] affiche la photo ;
- [(#DESCRIPTIF|extraire_balise{img}|extraire_attribut{src})] son URL ;
- [(#DESCRIPTIF|extraire_balise{img}|extraire_attribut{width})] sa largeur ;
- on peut même en modifier le style avec, par exemple :

```
[(#DESCRIPTIF|extraire_balise{img}|
    inserer_attribut{style,'border: double red 4px;'})]
```

> Remarque : les contenus HTML provenant d'un site extérieur sont par définition considérés comme « non contrôlés », et donc potentiellement problématiques s'ils contiennent des balises mal fermées, ou du javascript ; SPIP leur applique donc systématiquement le filtre safehtml avant affichage.

Autres filtres liés à la syndication

Ces filtres permettent de convertir les tags de syndication d'un format à un autre, afin de pouvoir par exemple remettre au format RSS des « tags » récupérés par syndication, et enregistrés dans notre base de données sous forme de microformats : tags2dcsubject, enclosure2microformat, microformat2enclosure.

Références

- RSS 2.0
- microformats
- rel=tag
- rel=enclosure
- rel=directory

Notes

[1] Attention les fichiers eux-mêmes ne sont pas recopiés : seules leur URL et les données associées (titre, taille, format) sont récupérées. Par ailleurs il faut signaler, pour les puristes, qu'on prend ici quelque liberté avec la norme RSS, qui interdit normalement d'avoir plusieurs *enclosures* dans un même article.

Le système de pagination

Lorsqu'une boucle renvoie plusieurs dizaines d'articles (ou, pour une pétition, plusieurs milliers de signatures), il n'est pas souhaitable, voire impossible, de tout afficher sur une seule page.

On préférera alors répartir les résultats en plusieurs pages, avec un système de navigation page à page. S'il est possible de réaliser cela avec des boucles SPIP ordinaires, c'est tout de même relativement complexe.

Aussi SPIP 1.9 introduit-il un système simplifié de pagination des résultats d'une boucle.

Exemple

Au plus simple, ce système est composé d'un critère et d'une balise :

- le critère {pagination} s'ajoute sur la boucle à paginer ;
- la balise #PAGINATION, placée dans une des parties optionnelles (« avant » ou « après ») de la boucle, affiche la « pagination ».

```
<B_page>
  #PAGINATION
  <ul>
<BOUCLE_page(ARTICLES) {par date} {pagination}>
  <li>#TITRE</li>
</BOUCLE_page>
  </ul>
</B_page>
```

Si le site comporte 90 articles publiés, cette boucle affichera la liste des dix plus anciens articles, surplombée de liens conduisant vers la page qui affiche les dix suivants, les dix d'après, etc. Ces liens sont numérotés comme suit :

0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | ...

Le numéro à partir duquel les résultats sont affichés est passé dans l'url via un paramètre {debut_page=x} portant le même nom (ici, « page ») que la boucle concernée. (Ce paramètre est exploitable dans une autre boucle via le critère classique {debut_page,10}.)

A noter : le nombre total de liens affichés est limité ; des points de suspension permettent, le cas échéant, d'aller directement à la toute fin de la liste, ou de revenir au tout-début.

Ancres de pagination

La balise #PAGINATION comporte une ancre html qui permet au navigateur d'afficher directement la partie de la page qui est paginée ; toutefois si l'on veut mettre les liens de pagination *en dessous* de la liste des articles, il faut pouvoir placer l'ancre *au-dessus* de la liste.

C'est à cela que sert la balise #ANCRE_PAGINATION, qui retourne l'ancre en question, et interdit à la balise #PAGINATION suivante d'afficher son ancre.

```
<B_page>
#ANCRE_PAGINATION
  <ul>
<BOUCLE_page(ARTICLES) {par date} {pagination}>
  <li>#TITRE</li>
```

```
</BOUCLE_page>
  </ul>
#PAGINATION
</B_page>
```

Le nombre total de résultats

Dans une boucle avec le critère {pagination}, #TOTAL_BOUCLE affiche le nombre d'éléments effectivement retournés, c'est-à-dire 10 sur les pages pleines, et 10 ou moins sur la dernière page de résultats.

Pour afficher le nombre d'éléments qui **auraient été retournés** si le critère {pagination} n'avait pas été là, utilisez la balise #GRAND_TOTAL.

```
<B_pagination>
#ANCRE_PAGINATION
<BOUCLE_pagination (ARTICLES) {pagination}>
#TITRE <br />
</BOUCLE_pagination>
Il y a au total #GRAND_TOTAL articles, cette page en affiche #TOTAL_BOUCLE
</B_pagination>
```

indiquera : « Il y a au total 1578 articles, cette page en affiche 10. »

Changer le pas de la {pagination}

Le nombre standard de 10 éléments par page peut être modifié par un paramètre supplémentaire dans le critère.

Ainsi

```
<BOUCLE_page (ARTICLES) {pagination 5}>
#TITRE <br />
</BOUCLE_page>
```

retournera les titres de cinq articles à partir de *debut_page*.

Le paramètre en question peut lui-même être composé comme on le souhaite à partir d'autres balises, notamment #ENV{xx}, ce qui permet de faire un affichage à la demande très complet.

La pagination dans les squelettes inclus

Si votre pagination doit fonctionner dans un squelette inclus, vous **devez** passer en paramètre de la commande INCLURE la formulation {self=#SELF} ; ceci permet au squelette inclus de se calculer avec la bonne valeur du paramètre debut_xxx, et se justifie qui plus est par un besoin de sécurité (la balise #PAGINATION est en effet calculée à partir de l'URL complète de la page).

Styles de la pagination

La pagination est constituée d'une série de liens, et d'un numéro de page correspondant à la page actuelle doté de la class « .on » : on définira donc les styles pour a et .on pour en personnaliser l'apparence.

Choisir le modèle de pagination

Depuis SPIP 1.9.1, la balise #PAGINATION accepte un paramètre {modele}, qui permet de modifier le résultat de la balise.

Ainsi #PAGINATION{precedent_suivant} affichera des liens vers les pages précédentes et suivantes. Les

liens seront les suivants

> page précédente | page suivante

#PAGINATION{page} affichera quelque chose de la forme suivante

> 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ...

#PAGINATION{page_precedent_suivant} affichera quelquechose comme

> ≤ 1 | 2 | 3 | 4 | 5 | 6 | ≥

Il est possible de définir d'autres modèles de pagination, qui devront s'appeler pagination_modele. Pour plus d'info, lire la documentation sur les modèles.

P.-S.

> **Attention :** beaucoup plus simple, ce mécanisme de pagination est aussi *incompatible* avec celui SPIP-Contrib', qui lui a servi de matrice. Si vous utilisiez la contrib, il vous faudra revoir les squelettes concernés.

Les variantes de squelette

par langue, par rubrique ou par branche

SPIP permet de gérer des variantes de squelettes par rubrique, par branche ou par langue.

Comme mentionné au début du manuel de réference et dans la documentation sur le multilinguisme, SPIP permet de gérer des *variantes des squelettes*, par langue et pour certaines rubriques uniquement.

Des mises en page différentes

On peut souhaiter, par exemple, que tous les articles d'une rubrique aient une mise en page différente : couleur de fond et taille de texte différente, informations relatives aux mots clés mises en évidence, etc. Ou encore que le contenu d'une rubrique donnée soit présentée différement parce qu'il correspond à un type de données différents : par exemple en listant tous les articles par numéro, y compris le contenu qui serait court, plutôt que les derniers en dates suivis d'une pagination de tous les articles, avec des liens vers les pages d'articles. On pourra aussi vouloir que l'interface du site soit différente selon la langue de l'article ou de la rubrique.

Les variantes de squelettes sont une manière simple — mais du coup, pas forcément très souple — permettent à SPIP de faire cela. Pour lui indiquer d'utiliser des mises en pages différentes, il suffit de réaliser des squelettes différents auxquels on donnera des noms de fichier qui indiquent quand il faut les utiliser :

- rubrique=22.html : squelette spécial pour la rubrique numéro 22,
- article-2.html : squelette pour tous les articles contenus dans la rubrique 2 et ses sous- rubriques,
- article.en.html : squelette pour les articles anglais,
- rubrique.html : squelette par défaut, s'appliquant à toutes les rubriques n'ayant pas de squelette particulier, **mais dont la présence est obligatoire dans un répertoire pour que les variantes de squelettes soient prises en compte**.

Ordre exhaustif des variantes de squelette

Prenons l'exemple des squelettes d'article, avec des valeurs données de langue et de rubriques (mais l'explication ci-dessous reste valable pour les squelettes de rubriques ou les brèves, et bien sûr, quelles que soient les langues et les numéros de rubriques).

Rappellons avant tout que SPIP recherche *d'abord* le répertoire où il prendra le squelette, comme détaillé dans « Où placer les fichiers de squelettes ? », en cherchant s'il existe un fichier article.html. Il ne faut donc pas créer de variante de squelette (par exemple un fichier article.cs.html) sans créer *dans le même répertoire* un fichier article.html, au risque de voir notre variante ignorée au profit du squelette générique d'un répertoire de priorité moindre.

Si ce fichier existe, SPIP utilise les fichiers de squelettes selon leur nom en commençant par « les plus précis », avec précédence du rubricage par rapport à la langue. Voici donc l'ordre (par priorité décroissante) dans lequel sont utilisés les fichiers de squelettes selon leur nom :

- article=8.cs.html : si ce fichier existe, il ne s'applique qu'aux articles en langue tchèque de la rubrique numéro 8 (mais pas aux articles contenus par ses sous-rubriques).
- article=8.html : si ce fichier existe, il s'applique aux articles de la rubrique 8 (sauf aux articles concernés par un fichier spécifiant aussi la langue, tel que précedemment).
- article-2.es.html : si ce fichier existe, il s'applique aux articles en espagnol contenus dans la rubrique 2 et ses sous-rubriques. Si la rubrique 8 est une sous-rubrique de la 2, si les fichiers ci- dessus existent, ils prévaudront.

- article-2.html : si ce fichier existe, il s'applique aux articles contenus dans la rubrique 2 et ses sous-rubriques (sauf aux articles concernés par les fichiers indiqués ci-dessus).

- article.vi.html : si ce fichier existe, il s'applique aux articles en vietnamien, dans toutes les rubriques (sauf aux articles concernés par les fichiers indiqués ci-dessus).

- Enfin, le fichier article.html s'applique à tous les articles qui ne sont pas concernés par les fichiers indiqués ci-dessus. Et, répétons-le, il est quoi qu'il en soit nécessaire que ce fichier existe.

Historique : Jusqu'à SPIP 1.7, SPIP 1.7.2 , si le fichier article.html n'existe pas, SPIP utilise alors le fichier article-dist.html qui est le fichier fourni par défaut. Voir « Qu'est-ce que les fichiers « dist » ? ».

Utiliser les modèles

SPIP 1.9.1 introduit la notion de *modèle*.

Qu'est-ce qu'un modèle ? Il s'agit d'un petit squelette SPIP qui décrit un fragment de HTML facile à insérer dans un autre squelette ou — et c'est là la principale nouveauté — dans le texte d'un article.

Inspiré des modèles de Wikipédia, le système des *modèles* offre de nouvelles capacités aux webmestres et aux rédacteurs.

Les modèles sont une extension des classiques raccourcis <img1> et <doc1>. Ceux-ci correspondent désormais aux modèles dist/modeles/img.html et dist/modeles/doc.html.

Leur syntaxe a été étendue, et il est possible de spécifier, outre l'alignement (<img1|left>, <img1|right> ou <img1|center>), une classe plus générique, qui soit correspond à un squelette précis, s'il existe, soit à une classe CSS (<img1|classe> ira chercher le modele modeles/img_classe.html, si ce fichier existe, sinon utilisera le modèle modeles/img.html, mais avec un paramètre class="classe").

Mais les modèles ne se limitent pas aux images et documents ; il est ainsi possible de créer de nouveaux raccourcis de la forme <modele1>, simplement en ajoutant un squelette dans le répertoire modeles/ de son dossier de squelettes !

> En l'absence de tout modèle correspondant au raccourci indiqué (par exemple, <breve1>), le gestionnaire de modèle de SPIP regarde s'il connaît l'objet demandé (ici, breve), et si ce dernier a une URL.
>
> 1 Dans ce cas (vérifié ici, car les brèves sont connues du système et disposent d'une fonction d'URL), SPIP remplace le raccourci <breve1> par un petit encadré flottant, avec un lien vers la brève, et l'affichage de son titre, comme si l'on avait indiqué [->breve1].
>
> Si l'objet n'est pas connu, SPIP laisse le raccourci intact, pour qu'il soit éventuellement traité par la suite (par un plugin ou un filtre supplémentaire), ou tout simplement ignoré.

Il est par ailleurs possible de passer des paramètres supplémentaires aux modèles (comme on le faisait pour les documents flash « embed », avec le raccourci <emb1|autostart=true>). La syntaxe en est généralisée et accepte même du HTML, comme dans l'exemple :

```
<son19|couleur=#ff0000
   |legende=Le grand <i>Count Basie</i>
   |photo=12>
```

qui pourrait appeler un modèle modeles/son.html, lequel exploiterait les paramètres pour afficher la photo numéro 12, dans un cadre de couleur #ff0000, avec une légende comportant des mots en italique.

Des usages multiples

Nous sommes loin d'avoir exploré toutes les applications possibles des modèles. En voici quelques unes auxquelles nous avons pensé. Si vous en trouvez d'autres, n'hésitez pas à la partager en proposant vos modèles sur SPIP Zone/SPIP Contrib', qui disposent désormais d'une rubrique dédiée.

- **Changer l'aspect des raccourcis de documents.** Souvent demandée, cette fonctionnalité était jusqu'ici difficile à mettre en place, puisqu'il fallait éditer du code php dans les fichiers du noyau de SPIP. Désormais, il suffit de recopier dist/modeles/img.html dans un sous-répertoire modeles/ de son répertoire de squelettes, et de modifier ce fichier. Idem pour les raccourcis <doc1> et <emb1>, bien que ce dernier soit d'une complexité qui peut rebuter...

Attention : ne vous lancez pas dans la modification des modèles pour des modifications mineures du rendu de ces raccourcis — il est souvent plus facile de modifier les styles spip_documents_xx à partir des fichiers CSS de votre site.

- **Jouer un son avec un player flash.** Un modèle modeles/son_player.html pourrait donner un raccourci <son12|player>.

- **Associer un site à un article.** En ajoutant dans ses squelettes un modèle modeles/site_box.html, on crée immédiatement un nouveau raccourci <site1|box>. On écrit alors le modèle (avec des boucles classiques) de manière à lui faire afficher le nom du site, suivi de liens vers les 3 derniers articles syndiqués, dans une boîte flottant sur la droite, et voici une infobox facile à placer dans un article. Un paramètre pourrait indiquer le nombre d'articles à afficher, la présence ou non de résumés, etc.

- **Afficher une image timbrée.** Une fois qu'on sait faire un squelette qui affiche une photo sous forme de timbre-poste (voir Un site dûment timbré), il suffit de le nommer modeles/timbre.html pour créer le raccourci <timbre12>. Avec, pourquoi pas, des paramètres de taille, de couleur, de choix du tampon, etc [1].

Bien entendu on peut imaginer de la même façon des modèles affichant les images en sépia, en version réduite, etc. Gageons que tout cela sera rapidement disponible sous forme de plugins.

- **Créer un article composite.** Supposons qu'on ait besoin d'un article composé du « chapo » de l'article 1 et du texte de l'article 2. Rien de plus simple : on crée deux modèles, qui renvoient tout simplement, pour l'un, le chapo de l'article demandé, pour l'autre son texte, et dans notre article composite on indique, dans le champ chapo : <article1|chapo>, et dans le champ texte : <article2|texte>. On peut y ajouter filtres, balises et bidouilles à volonté...

- **Installer un article dans plusieurs rubriques.** En programmant des modèles <article1|chapo>, <article1|texte> etc, il devient envisageable de mettre une copie d'un article dans un autre article, sans dupliquer les données. On obtient ainsi un article « fantôme » qu'on peut installer dans une nouvelle rubrique (avec, en plus, la possibilité de le titrer différement ou de lui ajouter des éléments).

- **Faire un sondage.** Le plugin *Forms*, qui permet de créer des formulaires et de les exploiter dans des articles avec le raccourci <form1>, a été réécrit à partir des modèles.

- **Afficher une citation aléatoire.** Si on a mis des citations dans ses brèves, un raccourci <citation|aleatoire> peut en extraire une au hasard (critère {par hasard}{0,1} sur une boucle de brèves), et l'afficher dans un encadré flottant à côté du paragraphe courant.

- **Insérer un document dans une autre langue** que la langue de l'article. Les modèles fonctionnant comme des inclusions, le paramètre lang=xx y est toujours disponible. Si l'un de vos documents est bilingue (par exemple avec un « bloc multi » dans le descriptif), vous pouvez afficher le descriptif en espéranto, dans un article en japonais, en appelant <doc1|left|lang=eo>. Si le squelette du modèle contient des chaînes de langue, elles seront interprêtées, le cas échéant, dans la langue passée en paramètre.

- **Afficher un graphique.** On passe une table de données au modèle, qui se débrouille ensuite pour créer le graphique et l'insérer dans le flux de texte.

- **Un intertitre sous forme d'image.** Pourquoi ne pas envisager un raccourci <imagetypo|texte=Mon intertitre> ?

Des paramètres à foison

La syntaxe des raccourcis de modèles est de la forme <modele12>, <modele|parametre1=truc|parametre2=chose> ou <modele12|alignement|parametre1=etc>. Les paramètres peuvent être composés de HTML et de raccourcis SPIP (à condition, bien sûr, que les modèles appelés aient prévu de les traiter).

On notera que, pour éviter toute collision avec les balises HTML, un modèle ne peut pas être appelé par un raccourci comme <modele>, qui ne contient ni chiffre ni le symbole |.

Les paramètres peuvent s'étendre sur plusieurs lignes, ce qui permet l'écriture relativement aérée :

```
<modele 10
 |pays=Allemagne
 |population=82000000
 |superficie=357027
 |classement=63
 |hymne=<i>Das Lied der Deutschen</i>
 |url=http://fr.wikipedia.org/wiki/Allemagne
>
```

Le squelette récupère tous ces paramètres dans la balise #ENV, ainsi #ENV{population} vaut 82000000. Cette balise étant sécurisée contre toute injection de javascript, si on veut permettre du HTML dans un paramètre, il convient d'utiliser la notation #ENV*{hymne} ; si l'on veut en plus appliquer la typographie de SPIP, on peut employer [(#ENV*{hymne}|typo)] ou [(#ENV*{hymne}|propre)].

Le paramètre principal (ici, 10), est passé sous deux formes : #ENV{id}=10, et #ENV{id_modele}=10. Ce qui permet d'accéder, pour un modèle appelé par <article3|chapo>, au chapo de l'article 3 en faisant : <BOUCLE_a(ARTICLES){id_article}>#CHAPO</BOUCLE_a>

ou bien aux brèves liées au mot-clé numéro 3, par ; <BOUCLE_b(BREVES) {id_mot=#ENV{id}}>#TITRE</BOUCLE_b>

Comme on le voit, chaque modèle devra avoir sa propre documentation, car le raccourci n'indique rien en lui-même sur l'exploitation faite des éléments passés par le raccourci.

Un emploi possible dans les squelettes

Les modèles ne sont pas limités à des raccourcis dans les articles. Il est possible de les appeler depuis un squelette, en utilisant la balise #MODELE{modele} ou [(#MODELE{modele}{p1=truc,p2=chose}{p3=etc}|filtre...)] ; cela est toutefois moins nouveau, car c'est équivalent à une inclusion (statique) de squelette (également permise par la nouvelle balise #INCLURE).

Les modèles par défaut

SPIP est livré avec les modèles suivants :

- img, doc, emb,
- article_mots et article_traductions, qui donnent respectivement la liste des mots-clés associés à un article, et de ses traductions (raccourcis : <article1|mots> et <article1|traductions>, mais ces modèles sont aussi appelés par le squelette par dist/article.html) ;
- lesauteurs, qui définit le produit de la balise #LESAUTEURS, mais ne peut pas être appelé comme un raccourci ;
- et une série de modèles de pagination (voir Le système de pagination).

Quelques conseils pour écrire un modèle

Il est conseillé de commencer par réfléchir à la syntaxe que l'on veut adopter : ce modèle est-il lié à un article précis ? Si oui, on partira sur un <article1|xxx>.

Une fois cette syntaxe établie, on crée le fichier modeles/article_xxx.html, et on y entre simplement la balise suivante : #ENV. Puis, dans un article de test, on tape notre raccourci <article1|xxx|param=x...> ; on voit alors (sous une forme un peu cryptique, il s'agit d'un tableau sérialisé) l'environnement tel qu'il est passé au modèle. On peut par exemple y distinguer notre identifiant d'article (sous le nom #ENV{id} et #ENV{id_article}.

Ensuite on peut commencer à entrer le squelette de notre fragment de page : <BOUCLE_x(ARTICLES){id_article}> ou <BOUCLE_x(ARTICLES){id_article=#ENV{id}}>. Et c'est parti...

Pour que notre modèle soit complet, il faut essayer les syntaxes <article1|xxx|left> et [<article1|xxx>->www.spip.net].

Dans le premier cas, c'est le paramètre align=left qui est passé au modèle (et il est souhaitable que le modèle s'aligne alors du côté demandé).

Dans le second cas, le lien est passé dans un paramètre lien=http://www.spip.net, et la classe du lien dans lien_classe=spip_out. Il est recommandé de prendre en compte l'url demandée, en la transformant en lien, quelque part dans le modèle (par exemple sur le titre ou l'icone) ; dans ce cas, il *faut* ajouter dans la première balise HTML du modèle une class="spip_lien_ok", qui signalera au gestionnaire de modèle que le lien a été pris en compte (faute de quoi le gestionnaire ajoutera un ... autour du modèle produit.

En ce qui concerne les paramètres, la balise #ENV{x} a été conçue de manière à éviter toute injection de HTML ou de javascript non désirée. Dans un modèle, on peut souhaiter autoriser le HTML dans les paramètres : il faut alors utiliser #ENV*{x} pour récupérer les données, et éventuellement les filtrer par |propre ou |typo, selon le type de données (texte pouvant comporter plusieurs paragraphes, ou simple ligne de texte).

Sur le plan de la programmation, il est recommandé de concevoir ses modèles tout en boucles, sans aucun code php ni balise dynamique. A noter toutefois : dans un modèle comportant du php, c'est le résultat du calcul qui est mis en cache, et non le script lui-même (comme c'est le cas avec les squelettes).

Notes

[1] Si l'on préfère que le raccourci s'appelle <img12|timbre>, on nommera le modèle modeles/img_timbre.html.

La structure de la base de données

La structure de la base de données est assez simple. Certaines conventions ont été utilisées, que vous repérerez assez facilement au cours de ce document. Par exemple, la plupart des objets sont indexés par un entier autoincrémenté dont le nom est du type id_objet, et qui est déclaré comme clé primaire dans la table appropriée.

NB : cet article commence à dater, et personne n'a encore pris la peine d'en faire la mise à jour. Il faut le lire comme un élément permettant de comprendre le fonctionnement de SPIP, mais plus comme un outil de référence. Si vous souhaitez contribuer à la documentation en refondant cet article, surtout n'hésitez pas !

Contenu rédactionnel

Les rubriques : spip_rubriques

id_rubrique	bigint(21)
id_parent	bigint(21)
titre	text
descriptif	text
texte	longblob
id_secteur	bigint(21)
maj	timestamp(14)
export	varchar(10)
id_import	bigint(20)

Chaque rubrique est identifiée par son **id_rubrique**.
- **id_parent** est l'*id_rubrique* de la rubrique qui contient cette rubrique (zéro si la rubrique se trouve à la racine du site).
- **titre**, **descriptif**, **texte** parlent d'eux-mêmes.
- **id_secteur** est l'*id_rubrique* de la rubrique en tête de la hiérarchie contenant cette rubrique. Une rubrique dépend d'une rubrique qui dépend d'une rubrique... jusqu'à une rubrique placée à la racine du site ; c'est cette dernière rubrique qui détermine l'*id_secteur*. Cette valeur précalculée permet d'accélérer certains calculs de l'espace public (en effet, les brèves sont classées par secteur uniquement, et non selon toute la hiérarchie).
- **maj** est un champ technique mis à jour automatiquement par MySQL, qui contient la date de la dernière modification de l'entrée dans la table.
- **export**, **id_import** sont des champs réservés pour des fonctionnalités futures.

Les articles : spip_articles

id_article	bigint(21)
surtitre	text
titre	text
soustitre	text
id_rubrique	bigint(21)
descriptif	text
chapo	mediumtext
texte	longblob
ps	mediumtext
date	datetime
statut	varchar(10)
id_secteur	bigint(21)
maj	timestamp(14)
export	varchar(10)
images	text
date_redac	datetime
visites	int(11)
referers	blob
accepter_forum	char(3)

Chaque article est identifié par son **id_article**.
- **id_rubrique** indique dans quelle rubrique est rangé l'article.
- **id_secteur** indique le secteur correspondant à la rubrique susmentionnée (voir le paragraphe précédent pour l'explication de la différence entre les deux).
- **titre**, **surtitre**, **soustitre**, **descriptif**, **chapo**, **texte**, **ps** parlent d'eux-mêmes.
- **date** est la date de publication de l'article (si l'article n'a pas encore été publié, c'est la date de création).
- **date_redac** est la date de publication antérieure si vous réglez cette valeur, sinon elle est égale à « 0000-00-00 ».
- **statut** est le statut actuel de l'article : prepa (en cours de rédaction), prop (proposé à la publication), publie (publié), refuse (refusé), poubelle (à la poubelle).
- **accepter_forum** : permet de régler manuellement si l'article accepte des forums (par défaut, oui).
- **maj** : même signification que dans la table des rubriques.
- **export** est un champ réservé pour des fonctionnalités futures.
- **images** est un champ contenant la liste des images utilisées par l'article, dans un format particulier. Ce champ est généré par spip_image.php3.
- **visites** et **referers** sont utilisés pour les statistiques sur les articles. Le premier est le nombre de chargements de l'article dans l'espace public ; le deuxième contient un extrait de hash des différents referers, afin de connaître le nombre de referers distincts. Voir inc-stats.php3.

Les auteurs : spip_auteurs

id_auteur	bigint(21)
nom	text
bio	text
email	tinytext
nom_site	tinytext
url_site	text
login	tinytext
pass	tinyblob
statut	tinytext
maj	timestamp(14)
pgp	blob
htpass	tinyblob

- Chaque auteur est identifié par son **id_auteur**.
- **nom**, **bio**, **nom_site**, **url_site**, **pgp** sont respectivement le nom de l'auteur, sa courte biographie, son adresse e-mail, le nom et l'URL de son site Web, sa clé PGP. Informations modifiables librement par l'auteur.
- **email**, **login** sont son e-mail d'inscription et son login. Ils ne sont modifiables que par un administrateur.
- **pass** est le hash MD5 du mot de passe.
- **htpass** est la valeur cryptée (i.e. générée par crypt()) du mot de passe pour le .htpasswd.
- **statut** est le statut de l'auteur : 0minirezo (administrateur), 1comite (rédacteur), 5poubelle (à la poubelle), 6forum (abonné aux forums, lorque ceux-ci sont réglés en mode « par abonnement »).
- **maj** a la même signification que dans les autres tables.

Les brèves : spip_breves

id_breve	bigint(21)
date_heure	datetime
titre	text
texte	longblob
lien_titre	text
lien_url	text
statut	varchar(6)
id_rubrique	bigint(21)
maj	timestamp(14)

- Chaque brève est identifiée par son **id_breve**.
- **id_rubrique** est la rubrique (en fait, le secteur) dans laquelle est classée la brève.
- **titre**, **texte**, **lien_titre**, **lien_url** sont le titre, le texte, le nom et l'adresse du lien associé à la brève.
- **date_heure** est la date de la brève.
- **statut** est le statut de la brève : prop (proposée à la publication), publie (publiée), refuse (refusée).
- **maj** : idem que dans les autres tables.

Les mots-clés : spip_mots

id_mot	bigint(21)
type	varchar(100)
titre	text
descriptif	text
texte	longblob
maj	timestamp(14)

- Chaque mot-clé est identifié par son **id_mot**.
- Le **type** du mot-clé est le type, ou groupe, choisi pour le mot-clé. En définissant plusieurs types, on définit plusieurs classifications indépendantes (par exemple « sujet », « époque », « pays »...).
- **titre**, **descriptif**, **texte** parlent d'eux-mêmes.
- **maj** : idem que dans les autres tables.

Les sites syndiqués : spip_syndic

id_syndic	bigint(20)
id_rubrique	bigint(20)
id_secteur	bigint(20)
nom_site	blob
url_site	blob
url_syndic	blob
descriptif	blob

- Chaque site syndiqué est identifié par son **id_syndic**.
- **id_rubrique** et **id_secteur** définissent l'endroit dans la hiérarchie du site où viennent s'insérer les contenus syndiqués.
- **nom_site**, **url_site**, **descriptif** sont le nom, l'adresse et le descriptif du site syndiqué.
- **url_syndic** est l'adresse du fichier dynamique utilisé pour récupérer les contenus syndiqués (souvent il s'agit de *url_site* suivi de backend.php3).

Les articles syndiqués : spip_syndic_articles

id_syndic_article	bigint(20)
id_syndic	bigint(20)
titre	text
url	text
date	datetime
lesauteurs	text

Chaque article syndiqué est identifié par son **id_syndic_article**.
- **id_syndic** réfère au site syndiqué d'où est tiré l'article.
- **titre**, **url**, **date**, **lesauteurs** parlent d'eux-mêmes.

Eléments interactifs

Les messages de forums : spip_forum

id_forum	bigint(21)
id_parent	bigint(21)
id_rubrique	bigint(21)
id_article	bigint(21)
id_breve	bigint(21)
date_heure	datetime
titre	text
texte	mediumtext
auteur	text
email_auteur	text
nom_site	text
url_site	text
statut	varchar(8)
ip	varchar(16)
maj	timestamp(14)
id_auteur	bigint(20)
id_message	bigint(21)

- Chaque message de forum est identifié par son **id_forum**.
- L'objet auquel est attaché le forum est identifié par son **id_rubrique**, **id_article** ou **id_breve**. Par défaut, ces valeurs sont égales à zéro.
- Le message parent (c'est-à-dire le message auquel répond ce message) est identifié par **id_parent**. Si le message ne répond à aucun autre message, cette valeur est égale à zéro.
- **titre**, **texte**, **nom_site**, **url_site** sont le titre et le texte du message, le nom et l'adresse du lien y attaché.
- **auteur** et **email_auteur** sont le nom et l'e-mail déclarés par l'auteur. Dans le cas des forums par abonnement, ils ne sont pas forcément identiques aux données enregistrées dans la fiche de l'auteur (i.e. dans la table spip_auteurs).
- **id_auteur** identifie l'auteur du message dans le cas de forums par abonnement.
- **statut** est le statut du message : publie (lisible dans l'espace public), prive (écrit en réaction à un article dans l'espace privé), privrac (écrit dans le forum interne dans l'espace privé), off (supprimé ou à valider, selon la modération des forums - a priori ou a posteriori).
- **ip** est l'adresse IP de l'auteur, dans les forums publics.
- **maj** a la même signification que dans les autres tables.

Les pétitions : spip_petitions

id_article	bigint(21)
email_unique	char(3)
site_obli	char(3)
site_unique	char(3)
message	char(3)
texte	longblob
maj	timestamp(14)

- **id_article** identifie l'article auquel est associée la pétition (une seule pétition par article).
- **email_unique**, **site_obli**, **site_unique**, **message** définissent la configuration de la pétition : l'adresse e-mail des signataires doit-elle être unique dans les signatures, l'adresse Web est-elle obligatoire, est-elle unique, un message attenant aux signatures est-il autorisé (oui ou non).
- **texte** est le texte de la pétition.
- **maj** : pareil que dans les autres tables.

Les signatures de pétitions : spip_signatures

id_signature	bigint(21)
id_article	bigint(21)
date_time	datetime
nom_email	text
ad_email	text
nom_site	text
url_site	text
message	mediumtext
statut	varchar(10)
maj	timestamp(14)

– Chaque signature est identifiée par son **id_signature**.
- **id_article** identifie l'article, donc la pétition sur laquelle est apposée la signature.
- **nom_email**, **ad_email**, **nom_site**, **url_site** sont le nom, l'adresse e-mail, ainsi que le site Web déclarés par le signataire.
- **message** est le message éventuellement entré par le signataire.
- **statut** est le statut de la signature : publie (acceptée), poubelle (supprimée) ; toute autre valeur donne la valeur de la clé de validation utilisée pour la confirmation par e-mail.
- **maj** a la même signification que dans les autres tables.

Les relations entre objets

Ces tables ne gèrent aucun contenu, simplement une relation entre les objets présents dans d'autres tables. Ainsi :

- **spip_auteurs_articles** spécifie la relation entre auteurs et articles. Si un *id_auteur* y est associé à un *id_article*, cela veut dire que l'auteur en question a écrit ou co-écrit l'article (il peut y avoir plusieurs auteurs par article, et vice-versa bien sûr).
- **spip_mots_articles** définit de même la relation de référencement des articles par des mots-clés.

Gestion du site

La table **spip_meta** est primordiale. Elle contient des couples (nom, valeur) indexés par le nom (clé primaire) ; ces couples permettent de stocker différentes informations telles que la configuration du site, ou la version installée de SPIP.

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

Indexation (moteur de recherche)

Six tables sont utilisées par le moteur de recherche. Elles se divisent en deux catégories.

Le dictionnaire d'indexation : spip_index_dico

Chaque mot rencontré au cours de l'indexation est stocké dans cette table, ainsi que les 64 premiers bits de son hash MD5. C'est le mot qui sert de clé primaire, permettant ainsi d'effectuer très rapidement des requêtes sur un début de mot ; on récupère alors le(s) hash satisfaisant à la requête, afin d'effectuer la recherche proprement dite dans les tables d'indexation.

Les tables d'indexation : spip_index_*

Ces tables, au nombre de cinq, gèrent chacune l'indexation d'un type d'objet : articles, rubriques, brèves, auteurs, mots-clés. Une entrée par mot et par objet est stockée. Chaque entrée contient le hash du mot (en fait les 64 bits de poids fort du hash MD5, cf. ci-dessus), l'identifiant de l'objet indexé (par exemple l'*id_article* pour un article), et le nombre de points associé à l'indexation du mot dans l'objet. Ce nombre de points est calculé en fonction du nombre d'occurences du mot, pondéré par le champ où ont lieu les occurences : une occurence dans le titre d'un article génère plus de points qu'une occurence dans le corps de l'article.

Le mécanisme d'indexation est expliqué plus en détail ici.