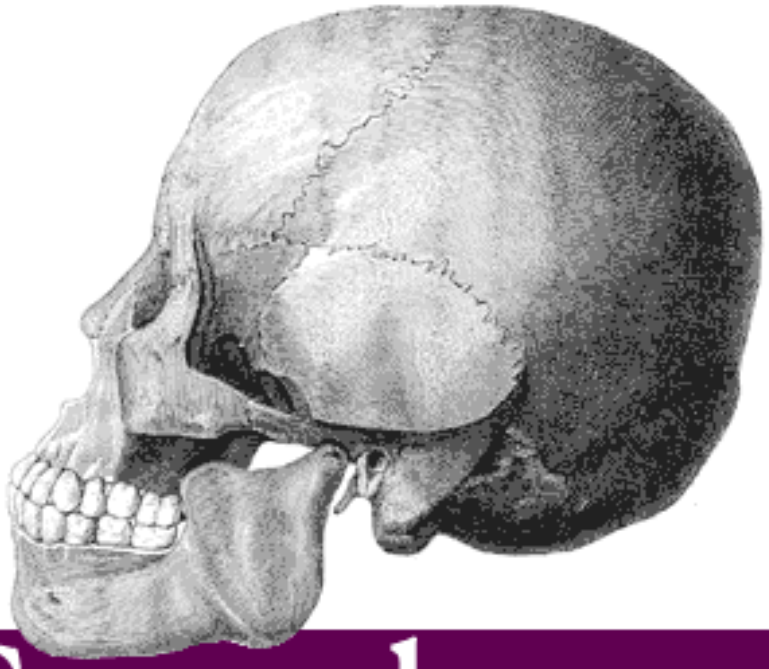
SPIP

Système de Publication pour l'Internet



Squelettes

Guide du webmestre

uZine 3

Avant-propos

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

VERSION 20021217
Compilation du document
à l'aide de PDFLaTeX

Philippe Charlier

[1] Version actuelle : SPIP 1.5
[2] http ://www.uzine.net

Table des matières

Lien utile : nous fournissons sur notre site FTP de nombreux exemples de squelettes[3].

1 FAQ webmestre

2 juillet 2001 par l'équipe de SPIP

1.1 Les bases

1. **Comment fais-je pour modifier la mise en page du site public ?**
   La gestion de la mise en page s'appuie sur des fichiers à l'extension `.html` appelés *squelettes* de mise en page. Leur rôle correspond grosso modo à ce que d'autres logiciels nomment « modèles » (ou en anglais, « templates »).
   Chaque fichier est associé à un type de page différent : ainsi un squelette pour le sommaire, un pour l'affichage des articles, un pour l'affichage des rubriques, etc. Un squelette contient du HTML standard définissant l'habillage de la page, dans lequel on insère des tags (ou balises) spécifiques à SPIP afin de définir quelles informations vont venir « habiter » cet habillage.
   Le langage des squelettes de SPIP est très souple et permet de réaliser des mises en page très variées : un simple coup d'oeil à uZine[4], Vacarme[5], Hacktivist News Service[6] ainsi que les sites enregistrés par leurs créateurs sur cette page[7] saura vous en convaincre. Il est donc dommage de garder la mise en page d'origine.
2. **Est-il possible d'écrire ces squelettes soi-même ?**
   Oui. Pour cela allez voir :
   ▷ le tutorial[8], pour comprendre les bases de la programmation des squelettes.
   ▷ le manuel de référence[9], qui liste toutes les possibilités de programmation.
   Si vous souhaitez un exemple didactique, vous pouvez aussi partir du jeu de squelettes « simples »[10], qui utilise un HTML très basique afin de bien laisser comprendre l'utilisation des tags spécifiques à SPIP.
3. **Je ne sais pas / ne veux pas apprendre à programmer. Peut-on utiliser des mises en pages déjà existantes ?**
   Oui. En dehors de la mise en page par défaut, d'autres jeux de squelettes sont disponibles sur le serveur de téléchargement[11], dans le répertoire « SQUELETTES ».
   Il suffit en général de récupérer l'archive voulue (le fichier au format .zip ou .tar.gz, au choix), de la décompresser chez vous, et de transférer son contenu par FTP à la racine de votre site SPIP. Vous pouvez faire une sauvegarde de vos fichiers .html actuels, au cas où vous voulez revenir en arrière.
4. **Il n'y a pas beaucoup de jeux de squelettes disponibles. Pourquoi ?**
   Ces jeux de squelettes sont alimentés par les webmestres SPIP qui nous fournissent leurs créations. Nous comptons donc sur les webmestres pour compléter cette base de squelettes afin d'encourager l'entraide et la richesse des sites SPIP. (cf. section « Partager » plus bas dans cette FAQ)

[3] http ://rezo.net/spip-dev/SQUELETTES/
[4] http ://www.uzine.net/
[5] http ://www.vacarme.eu.org
[6] http ://hns.samizdat.net/
[7] http ://www.uzine.net/article884.html
[8] http ://www.uzine.net/rubrique144.html
[9] http ://www.uzine.net/rubrique143.html
[10] http ://rezo.net/spip-dev/SQUELETTES/simples/
[11] http ://rezo.net/spip-dev/

1.2 Créer ses squelettes

1. **Peut-on utiliser un éditeur textuel pour créer et modifier ses squelettes ?**
   Oui, comme on le ferait pour du HTML classique.
2. **Peut-on utiliser un éditeur graphique (WYSIWYG) pour créer et modifier ses squelettes ?**
   Oui, comme on le ferait pour du HTML classique. Voir cependant la question suivante.
3. **J'essaie d'utiliser un éditeur graphique pour créer mes pages, mais il modifie les tags SPIP. Peut-on résoudre ce problème ?**
   Certains éditeurs graphiques « corrigent » automatiquement les tags qu'ils ne comprennent pas. La plupart ont toutefois une option permettant de désactiver cette fonctionnalité. Nous avons consacré un article spécifique à DreamWeaver, mais la démarche est équivalente pour les autres éditeurs (GoLive…).

1.3 Partager

1. **J'ai écrit des squelettes pour mon site. Comment fais-je pour qu'ils soient disponibles à tous ?**
   N'hésitez pas à les envoyer à l'adresse spip-dev@rezo.net, qui est la mailing-list des développeurs. Pour cela, créez simplement un fichier compressé (.zip, .tar.gz, etc.) contenant les différents fichiers .html. N'oubliez pas d'inclure les fichiers graphiques (images) le cas échéant.

2 Contribuer au développement de SPIP

10 juin 2001 par l'équipe de SPIP

Quelques règles

Si vous voulez contribuer à la programmation de SPIP, l'idée la plus importante à retenir est la suivante : *vous arrivez sur un projet qui est déjà fonctionnel*. Ce projet est muni d'un ensemble de règles qui, toutes arbitraires qu'elles peuvent paraître, assurent sa cohérence. Ces règles n'ont pas besoin d'être énoncées explicitement pour exister : certaines sont clairement visibles après un examen plus ou moins détaillé du code, et les règles tacites doivent être respectées au même titre que les autres.

Il est formellement conseillé de bien suivre ces règles. Ce respect n'a pas à être lié ou non à vos goûts personnels : il permet de garder sa cohérence et son unité au projet, et de le conserver aussi lisible qu'il l'était auparavant. N'oubliez pas que d'autres personnes que vous sont amenées à lire, comprendre, voire modifier votre code.

Par exemple, il est évident que les fonctions SPIP sont écrites sous la forme ma_fonction(). Dans le cadre de ce projet, il serait donc parfaitement déplacé d'ajouter des fonctions en les écrivant MaFonction() - même si dans l'absolu cette forme n'est pas plus critiquable que l'autre.

Tout ceci n'empêche pas évidemment de critiquer une règle et d'en proposer une meilleure, le cas échéant. N'hésitez pas à le faire ; mais il doit y avoir de véritables raisons à cela.

Enfin, tout règle souffre des exceptions. Mais celles-ci doivent avoir une véritable justification, non uniquement la paresse du programmeur ; elles doivent être le plus rares possible. Notamment, garder à l'esprit que le « provisoire » a souvent tendance à devenir définitif quand personne n'a envie de le corriger ; or il est logique et juste que tout programmeur soit responsable de la finition de son propre code, mais non de celui des autres.

2.1 Règles de présentation et d'écriture

Les règles qui suivent sont communes à un nombre plus ou moins grand de langages de programmation : au minimum tous les langages présentant une syntaxe similaire à PHP (c'est-à-dire, outre PHP lui-même, C, C++, Java…).

Ces règles sont communément acceptées, de façon aussi naturelle que les règles de présentation et de typographie d'un texte en langage naturel ; d'ailleurs elles sont fréquemment similaires.

Présentation

- Le code doit être espacé et indenté de manière à mettre en valeur sa structure et la séparation entre les différents blocs logiques (fonctions notamment). L'espacement et l'indentation doivent être suffisants pour rendre la structure compréhensible dès le premier regard ; ils ne doivent pas être excessifs. On doit y apporter le même soin qu'à la division en paragraphes d'un texte en langage naturel.
- L'indentation sera faite de préférence avec le caractère de tabulation. Cela permet de choisir librement la profondeur d'indentation dans les options de son éditeur de texte, tout en n'imposant pas ce choix aux autres développeurs.
- Tout bloc contenu à l'intérieur d'une paire d'accolades sera indenté d'une et une seule tabulation. De même, récursivement, pour les sous-blocs : ajout d'une et une seule tabulation supplémentaire à chaque entrée dans un niveau de profondeur supplémentaire. Cette règle vaut bien aussi pour la déclaration de fonctions.
- Le code qui ne fait pas partie d'une fonction ne doit pas être indenté.
- L'utilisation des transitions PHP-HTML (`<?php et ?>`) doit être minimisée. L'éviter lorsqu'il s'agit d'afficher uniquement de petits morceaux de HTML. Ne pas oublier qu'un petit morceau de code PHP inséré au milieu d'un océan de HTML est invisible sans un examen très attentionné.

Typographie

- Lors de l'utilisation de parenthèses ou de crochets, il ne faut pas laisser d'espace après la parenthèse ouvrante ni avant la parenthèse fermante.
- Lors de l'utilisation d'opérateurs binaires (`+`, `=`, `*`, `AND`, ...), il faut laisser un espace de part et d'autre de l'opérateur. Exception manifeste dans cette phrase, où les opérateurs sont mentionnés et non utilisés en tant que tels.
- Les opérateurs unaires (!, ...) doivent être collés au paramètre auquel ils s'appliquent.
- Par convention, lors d'un appel de fonction, il n'y a pas d'espace devant la parenthèse ouvrante : « `f($x)` » et non « `f ($x)` ». A contrario, et pour bien distinguer, on laisse un espace devant la parenthèse quand il s'agit d'une structure de contrôle intégrée au langage : « `if (!$x)` » et non « `if(!$x)` ».
- Les virgules et points-virgules sont suivis mais non précédés d'un espace.

2.2 Règles de programmation

Réfléchir

Avant de programmer une nouvelle fonctionnalité, réfléchir...

- méthodes et algorithmes utilisés pour l'implémentation : légèreté, performance, robustesse (ne pas hésiter à faire quelques calculs grossiers pour valider les choix) ;
- adéquation au projet : portabilité, sécurité, souplesse ;
- implications sur les autres fonctionnalités : modifications et ajouts à faire sur les fonctionnalités existantes ;
- place « naturelle » pour cette fonctionnalité dans le projet : en matière d'interface, de fichiers...

Ne pas négliger la factorisation ou mise en commun du code (par des fonctions, notamment dans des fichiers à inclure). Eviter par contre le plus possible les fichiers inclus contenant du code hors fonctions (sauf lorsque c'est « naturel » et voulu).

Nommer

- *Variables et fonctions* :

Quel que soit le projet, le nommage doit rester homogène pour que le code soit facile à lire. Ainsi, sous SPIP, les noms de variables et de fonctions seront en minuscules ; les noms composés, de la forme variable_composee.
D'une manière générale, les noms seront ni trop brefs, ni trop longs ; ils seront suffisamment explicites. Cette règle est particulièrement importante pour les variables globales, qui peuvent être partagées entre plusieurs fichiers et de nombreuses fonctions. Pour les variables locales (i.e. à une fonction), la règle est plus souple. Notamment, on peut employer des variables d'une lettre, par exemple pour obtenir des expressions plus compactes. Remarquer que dans tous les langages de programmation, un certain nombre de lettres sont par tradition associées à certains usages (exemples : `$i`, `$j` pour des compteurs de boucles, `$n` pour un dénombrement, `$t` pour un instant ou une durée en secondes...). Ne pas détourner ces conventions permet à vos lecteurs d'être plus vite dans le bain.

- *Fichiers* :
  Pour des raisons historiques, les fichiers à inclure dans l'espace public seront appelés `inc-fichier.php3`.
  Dans l'espace privé, ce sera `ecrire/inc_fichier.php3` (noter le tiret bas à la place du tiret normal).
  Les fichiers de l'espace public appelés par redirection HTTP depuis l'espace privé sont appelés `spip_fichier.php3`.
  Tous les autres fichiers ont un nom qui ne commence ni par « `inc` », ni par « `spip` ».
- *Tester* :
  Une fois une modification importante apportée, il est bon de la tester soi-même, sans attendre que quelqu'un d'autre le fasse à sa place. Dans le cadre de SPIP, cela veut dire vérifier que le programme marche de manière correcte sur un certain nombre d'hébergeurs (par exemple : Altern, Free...) et de configurations (par exemple : différentes versions de PHP, de MySQL, restriction plus ou moins grande des droits d'accès aux répertoires...) ; mais aussi qu'un certain nombre de situations parmi les plus courantes (dans le cas d'une interface graphique notamment) sont traitées correctement.

2.3 Partager vos modifications

Une fois que vous êtes satisfait de votre modification du code, il est grand temps d'en parler avec les autres développeurs de SPIP, et de voir s'il mérite d'être intégré à la distribution officielle de SPIP...Rendez-vous sur la liste de diffusion spip-dev[12].

A bientôt !

3 La structure de la base de données

12 juin 2001 par l'équipe de SPIP

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 !

[12] http ://listes.rezo.net/mailman/listinfo/spip-dev

3.1 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.

3.2 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.

3.3 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.

3.4 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.

3.5 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 (section 4 page 10).

4 Le moteur de recherche

15 juin 2001 par l'équipe de SPIP

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.

4.1 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. C'est pourquoi le moteur est désactivé par défaut : il risque de ralentir quelque peu votre site (voir plus loin au chapitre performances pour des indications plus précises), et il sera avantageusement remplacé par un vrai moteur si vous avez la possibilité d'en installer un (si oui, vous pouvez par exemple utiliser ht ://Dig[13], un moteur en logiciel libre).

4.2 L'indexation

L'indexation est réalisée selon deux modalités différentes :

1. Si un contenu (article, brève...) est publié ou modifié depuis l'espace privé, alors il est réindexé immédiatement afin de tenir les données à jour.
2. Lors de la visite d'un contenu dans le site public, ce contenu est réindexé uniquement s'il n'a pas déjà été indexé. Pour être plus précis (mais ça ne change rien au principe général), SPIP attend qu'une page soit affichée en utilisant le cache, 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.

Le deuxième cas est là pour assurer que, si vous activez le moteur de recherche sur un site déjà rempli, les anciens contenus seront également pris en compte par le moteur ; de même si vous restaurez une sauvegarde (les données d'indexation ne sont pas incluses dans les sauvegardes pour des raisons d'espace disque).

L'indexation traite un à un 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'occurences. 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 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 (en cas de clavier inadapté...). 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`.

4.3 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, et conversion des caractères

[13] http ://htdig.org/

accentués. 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, c'est-à-dire de pertinence (mais cela est laissé à la volonté de la personne qui écrit les squelettes de mise en page[14]).

4.4 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'uZine[15] comporte, à la date d'écriture de ce texte, environ 400 articles et 400 brèves publiés, 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 (sauf en cas de machine très lente).

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.

Quant à la recherche, elle est simplement additive : il n'y a pas d'opérateurs booléens, l'opérateur implicite étant grosso modo un « OU » logique. D'autre part, le fait que les mots de trois lettres ou moins soient ignorés peut vous gêner si vous avez des besoins spécifiques (beaucoup de sigles, d'abbréviations…).

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.

[14] http ://www.uzine.net/article903.html
[15] http ://uzine.net/

5 Rapidité du site public

16 juin 2001 par l'équipe de SPIP

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…

5.1 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.

5.2 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.

5.3 Sur une machine dédiée

Si vous utilisez votre propre machine, il faut vous assurer qu'elle pourra tenir la charge. N'importe quelle machine récente devrait en être capable ; pour donner des chiffres très approximatifs, mettons qu'il faut au minimum un processeur à 200 MHz, et 128 Mo de mémoire.

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).

Version 1.2 - Le moteur de spip a été entièrement refondu dans la version 1.2. Sous le nom de code « Pantagruel » il comporte désormais deux phases de « cache » : la première lors de la lecture initiale des squelettes (elle crée

les fichiers `CACHE/skel_nomdusquelette.php3`) ; la seconde lors de la lecture des données dans la base mySQL. Chacune des étapes du calcul des pages a été revue et optimisée. Le gain total permet d'obtenir une génération de page en moins de la moitié du temps nécessaire pour la version 1.0.

6 Spip et les feuilles de style

23 septembre 2001 par l'équipe de SPIP

SPIP 1.2 Lorsque l'on utilise les raccourcis typographiques dans les articles dans SPIP (permettant par exemple de mettre en gras, en italique, créer des liens hypertextes, des intertitres, etc.), SPIP produit les balises HTML nécessaires à ces effets, chacune de ces balises étant alors associée à une classe de style CSS.

Par exemple,

```
Ceci est un [lien->http://www.uzine.net]
```

est transformé en code HTML ainsi :

```
Ceci est un <a href="http://www.uzine.net" class="spip_out">lien</a>
```

Le code HTML est ainsi complété par l'appel à un style CSS intitulé « spip_out ». L'utilisateur peut donc pousser la personnalisation de son interface graphique en définissant ce style « spip_out » (couleur différente, fond coloré, police utilisée…).

La plupart des raccourcis typographiques de SPIP peuvent ainsi être paramétrés avec des feuilles de style ; certains sont très utiles, d'autres seront réservés aux webmestres qui souhaitent obtenir des effets exotiques…

6.1 Où se trouve la définition de ces feuilles de style ?

Lors de l'installation de SPIP, avec les squelettes fournis en standard, la définition des feuilles de style se trouve dans le fichier :

▷ spip_style.css

Vous pouvez le modifier (c'est même conseillé), définir votre propre fichier de feuilles de style, ou intégrer directement des définitions dans vos squelettes.

Notez bien : la notion de feuille de style, ou cascading style sheets, n'est pas une norme propre à SPIP, il s'agit d'un standard du Web. De très nombreuses documentations existent sur ce sujet par ailleurs ; consulter par exemple la page du W3C[16] *à ce sujet.*

Afin de suivre la suite de la présente explication, il est vivement conseillé d'ouvrir le fichier « spip_style.css » dans un éditeur de textes.

6.2 Les liens hypertextes

Les deux premières définitions permettent de modifier le comportement de « a » et « a :hover » ; très classiques, elles concernent tous les liens affichés sur votre page Web (afficher les liens sans soulignement, et régler le « survol » des liens hypertextes).

Viennent ensuite trois définitions propres aux raccourcis typographiques de SPIP : « a.spip_in », « a.spip_out », « a.spip_url ».

[16] http ://www.w3.org/Style/CSS/

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

```
Ceci est un [lien interne->article1177]
```

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

```
Ceci est un [lien externe->http://www.uzine.net]
```

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

```
[->http://www.uzine.net]
```

(ce raccourci affiche directement l'URL, avec un lien hypertexte vers cette adresse, ainsi : http ://www.uzine.net).

Le principal intérêt de ces trois styles différents est de permettre de différencier graphiquement les liens internes au site et les liens vers un autre site.

6.3 Les intertitres

Les intertitres, créés par le raccourci suivant :

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

peuvent être définis par le style h3.spip. Ce style est sans doute l'un des plus utiles, car il permet de définir la taille, la police et le positionnement des intertitres dans les articles.

Par défaut, la définition en est :

```
h3.spip {
    font-family: Verdana,Arial,Helvetica,sans-serif;
    font-weight: bold;
    font-size: 120%;
    text-align: center;
}
```

6.4 Code et cadre

Les éléments de code, définis par le raccourci :

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

sont paramétrés par le style **.spip_code**. Peu utilisé.

Introduit dans SPIP 1.3, la balise `<cadre>...</cadre>` permet de présenter du code source dans un tableau (élément de formulaire) dans lequel il est facile copier-coller le texte. La feuille de style associée est : **.spip_cadre**, définie ainsi par défaut :

```
.spip_cadre {
        width : 100%;
        background-color: #FFFFFF;
        padding: 5px;
}
```

6.5 Les notes de bas de page

Les notes de bas de page, définies par le raccourci :

```
Le texte[[Une note de bas de page]]
```

sont paramétrés par le style **p.spip_note**. Souvent inutile, puisque les notes peuvent être modifiées directement en HTML lors de l'emploi de la balise `#NOTES` dans vos squelettes.

6.6 Les tableaux

Les tableaux sont définis dans SPIP de la façon suivante :

```
| {{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 donne :

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 feuilles de style permettent de paramétrer finement l'affichage de tels tableaux :

```
table.spip {
}

table.spip tr.row_first {
        background-color: #FCF4D0;
}

table.spip tr.row_odd {
        background-color: #C0C0C0;
}

table.spip tr.row_even {
        background-color: #F0F0F0;
}

table.spip td {
        padding: 1px;
        text-align: left;
        vertical-align: center;
}
```

▹ **table.spip** permet de modifier le comportement général du tableau (notamment sa position, à gauche, centré…) ;

- **table.spip tr.row_first** définit le comportement de la « première ligne » du tableau (ici en jaune) (pour que la « première ligne » soit prise en compte, il faut que les éléments qu'elle contient soient placés en gras) ;
- **table.spip tr.row_odd** pour les lignes paires ;
- **table.spip re.row_even** pour les lignes impaires ;
- **table.spip td** permet de modifier le comportement des cases du tableau.

Un des intérêts repose sur le choix de couleurs différentes pour « row_odd » et « row_even », permettant de faire une présentation de couleurs alternées pour les lignes d'un tableau (ici, gris clair et gris foncé).

6.7 Ligne de séparation horizontale

Une ligne de séparation horizontale, définie par :

```
----
```

peut être modifiée par : **hr.spip**.

6.8 Gras et italique

Le gras et l'italique sont définis par les raccourcis :

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

Ils peuvent être modifiés par les styles : **b.spip** et **i.spip**. Styles peu utiles.

6.9 Les paragaphes

Les paragraphes créés par SPIP (en laissant des lignes vides entre les paragraphes) peuvent être modifiés par le style : **p.spip**.

A priori, peu utile, car on peut directement paramétrer le comportement des éléments de texte en HTML.

6.10 Les formulaires

Dans l'espace public, différents formulaires sont utilisés pour le moteur de recherche interne, l'interface de rédaction des messages des forums, les inscriptions à l'espace privé…

Les feuilles de style sont : **.forml**, **.spip_encadrer**, **spip_bouton**, **.formrecherche**.

Par défaut, ils sont définis ainsi :

```
.forml { width: 100\% ; background-color: #FFDDAA;}
.spip_encadrer { background-color: #EEEEEE; }
.spip_bouton { background-color: #FFCC00;}
.formrecherche { width: 100\% ; background-color: #FFDDAA;}
```

- **.forml** définit les « cases » de texte des formulaires ; utile pour définir la largeur de ces cases, et la couleur du fond ;
- **.spip_encadrer** ; lorsqu'un formulaire propose différentes « parties », la séparation entre ces différentes parties peut être paramétrées avec ce style (par exemple, encadrer chaque partie, créer un espace avant ou après…) ;
- **.spip_bouton** modifie l'aspect du bouton de validation du formulaire ;
- **.formrecherche** modifie l'aspect de la case « Rechercher » du moteur de recherche.

6.11 Conclusion

Vous remarquerez que, par défaut, certaines feuille de style ne sont pas définies. Elles peuvent être considérées comme très accessoires (réservées aux webmestres voulant obtenir des effets graphiques très spécifiques).

En règle générale, les styles qui provoquent des modifications graphiques spectaculaires sur un site, par ailleurs simples à paramétrer, sont celles qui concernent :

- les liens de l'ensemble de la page, **a** et **a :hover**,
- le comportement des intertitres, **h3.spip**,
- les formulaires.