SPIP, *Système de Publication pour l'Internet*

Journées sur les RESeaux. Marseille, 5 Décembre 2005

Emmanuel Saint-James@lip6.fr

Plan

- Techniques et Technologies sous-jacentes
- Démonstration
- Utilisation avancée

Système de Gestion de Contenu : quoi, pourquoi ?

- C’est un moteur de mise en page de bases de données
- s’appuyant sur un ou plusieurs serveurs de base de données
- et un langage de programmation interfacé à HTTP.
- Il prend en charge la présentation des données
- et offre une interface conviviale pour leur saisie.
- Les données peuvent être :
    - du texte
    - des images fixes ou animées
    - des documents sonores.

  dans tous les formats.

Système de Gestion Contenu : Combien ?

- Le site cms-matrix recense plus de quatre cents CMS
- Comparables selon quelques centaines de critères ... mal définis
- C’est une technologie vieille de bientôt dix ans
- Assez gourmande en temps de calcul
- Donc nécessitant un *cache* soigneusement conçu
- Cas particulier : un *blog* est un CMS alimenté (officiellement...) par une seule personne.

9 bonnes raisons de choisir Spip

- Une prise en main immédiate grâce à une ergonomie optimale ;
- Une interopérabilité *http/html/css* parfaite tant côté serveur que client ;
- Un langage de mise en page puissant et intuitif ;
- Un correcteur *typographique* embellissant les saisies ;
- Une gestion rigoureuse du multi-linguisme (www.spip.net) ;
- Une équipe de développement réactive (spip-dev) ;
- Une communauté d'utilisateurs de même (spip-user) ;
- Dix mille sites installés dans le monde depuis 5 ans (contributions) ;
- Un logiciel libre (licence GNU).

4 bonnes raisons de ne pas choisir Spip

- Pas (encore) d'Interface de Programmation de l'Application ;
- Pas (encore) d'interopérabilité des serveurs de bases de données ;
- Pas (encore) de partage des sources (> 10Mo) ;
- Pas (encore) d'interface graphique pour le graphiste.

3 mauvaises raisons de ne pas choisir Spip

- les pages produites ne respectent pas toujours les normes du W3C ;
  - ... pour la même raison qu'un compilateur ne garantit pas de l'infaillibilité des programmes qu'il compile.
- le système de droits est simpliste
  - ... à l'image des systèmes d'exploitation usuels.

- la saisie est au kilomètre, il faut valider puis (pré-)visualiser
  - ... chaque texte saisi apparaitra à différents endroits de l'espace public, avec différents squelettes de mise en page ; il n'a pas d'apparence en soi.

Spip : vue générale

- un *espace public* pour
  - afficher selon des *squelettes de mise en pages* le contenu de la base de données ;
  - proposer de l'interaction avec les visiteurs par l'intermédiaire des *forums*.
- un *espace privé* pour :
  - alimenter la base en données à afficher ;
  - surveiller la vie du site (statistiques, forums) ;
  - gérer les participants.
- en coulisse : un *compilateur* traduisant les squelettes en scripts PHP/SQL qui alimenteront le *cache* du site.

Spip : structure de la base de données

- Une table des *rubriques*, une rubrique contenant :
  - des articles
  - des pièces jointes
  - des sous-rubriques
  - des brèves (pour les rubriques de premier niveau) ;
- Une table des *auteurs*, distinguant (par droits décroissants) :
  - les administrateurs généraux
  - les administrateurs restreints à une ou plusieurs rubriques
  - les rédacteurs
  - les abonnés des forums
- Les tables d'articles, de brèves, de mot-clés, de statistiques etc.

Le processus de rédaction

- Un *rédacteur* est une personne recensée dans la table des auteurs comme ayant le droit d'enrichir la table des articles mais sans pouvoir les publier ;
- Les rédacteurs sont déclarés
  - soit par un administrateur,
  - soit (si le site a été configuré ainsi) automatiquement via un formulaire ;
- Un article peut être rédigé à plusieurs, pendant plusieurs jours ;
- Les articles en cours de rédaction sont invisibles aux autres auteurs du site ;
- Les rédacteurs peuvent joindre à un article des logos et autres documents ;
- Lorsque les rédacteurs sont satisfaits de leurs textes, il les font passer du statut *en cours de rédaction* , au statut *soumis à publication*.

Le processus de publication

- Un *administrateur* affecte chaque texte des rédacteurs à une rubrique et le publie, soit en spécifiant une date ultérieure, soit immédiatement ;
- Lors des prochaines visites du site, le squelette de mise en page s'appliquera aux nouvelles entrées dans la base, et placera ses résultats en *cache* ;
- Le cache est réglable et réinitialisable, à différents niveaux de granularité ; il est constitué de fichiers PHP (en fait du HTML dans la plupart des cas) ;
- La publication d'un article entrainera l'indexation de son contenu, exploité par le moteur de recherche interne du système ;
- Tout article ou brève publiés peuvent être ensuite modifiés voire retirés ;
- Le texte d'un article peut être vide : c'est le cas limite de la galerie de photos.

Autour de la publication I : les mots-clés

- Pour gérer l'information transversale à l'arborescence des rubriques, existe une table des *groupes de mot-clés* , référençant une table des *mots-clés* ;
- Seuls les administrateurs généraux peuvent créer des groupes de mot-clés et les remplir ;
- L'attachement de mot-clé à un article ou une rubrique peut être accordé à chaque participant, groupe de mots par groupe de mots.
- Les mots-clés peuvent servir d'aide à la navigation dans l'espace public, mais certains peuvent aussi n'avoir de sens que dans l'espace privé, par exemple pour gérer plus finement les droits ou associer un squelette de mise en page.

Autour de la publication II : les visiteurs

- A chaque article, peut être associé un *forum de discussion* ouvert aux visiteurs selon plusieurs modes au choix :
  - *sur abonnement* : un visiteur donne son mail et y reçoit un mot de passe lui permettant de poster des messages aussitôt publiés ;
  - *modéré a priori* : les messages arrivent dans l'espace privé, et les administrateurs décident ou non de leur publication ;
  - *modéré a posteriori* : les messages sont publiés immédiatement et anonymement
- A chaque article peut également être associé une *pétition* avec collecte de signature (distinguées par le mail).
- Enfin, un article peut référencer (par *syndication*) d'autres sites, ou être une redirection vers une autre page Web.

Autour de la publication III : le suivi

Pour mesurer l'impact d'une publication, Spip offre :

- une collecte des visites quotidiennes, évitant de comptabiliser les visites rapprochées à partir d'une même adresse IP ;
- un graphique de *popularité* fondé sur ces visites ;
- la liste des liens entrants, permettant de trouver les pages référençant le site ;
- une notification des interventions sur les forums, envoyée :
  - par mail,
  - par flux RSS,
  - ou par abonnement webcal (format ICS).

La vie quotidienne des administrateurs

Pour gérer l'activité éditoriale, les administrateurs disposent :

- de leur page d'accueil signalant tous les articles et brèves soumis à publications ;
- de forums internes pour chaque article et pour l'équipe éditoriale ;

- d'un système de messagerie interne pour diffuser des annonces générales, fixer des rendez-vous, mémoriser des tâches ;
- d'un calendrier synthétisant toute ces informations à travers diverses *vues* ;
- de flux RSS et d'un abonnement *webcal/ICS* exportant ces informations.

Reconfigurer Spip

Spip est livré prêt à l'emploi mais il est possible :

- d'utiliser les *panneaux de configuration* pour régler certaines utilisations ;
- d'écrire ses propres *squelettes de mise en pages* à l'aide :
  - des *filtres* de mise en page,
  - des *boucles*, *critères* et *balises* (ou *champs*) interrogeant la base de données,
  - des *scripts* PHP standards ;
- de référencer dans les squelettes des tables non gérées par Spip, voire non gérées par son serveur SQL ;
- d'étendre ou de modifier, en programmant en PHP, les entités ci-dessus.

Les panneaux de configuration

Les espaces public et privé sont présentés en fonction d'options de configuration :

- le jeu de caractères ;
- le multilinguisme (gérant le bidirectionnel) ;
- le degré d'interactivité accordé aux visiteurs (syndication, mot-clé, ...) ;
- le degré d'intervention laissé aux rédacteurs (mot-clé, pièces jointes, complexité des articles et des brèves) ;
- et également l'activation personnelle de l'option *interface complète* , inactive lors de la prise en main.

L’appel de squelettes

Pour invoquer un squelette, utiliser le script page.php et le paramètre fond.

Le squelette standard rubrique.html, par exemple, peut s’appeler ainsi :

```
http://mon_site/page.php?fond=rubrique&id_rubrique=numero
```

Si l’on souhaite régler la durée en cache de la page produite, il faut écrire un script spécifique affectant la variable delais et appelant le mécanisme *calcul et mise en cache éventuels / envoi du cache* . Voici l’intégralité du script standard *rubrique.php* :

```
$fond = "rubrique" ;
$delais = 2 * 3600 ;
include ("inc-public.php3") ;
```

Comment déclarer un squelette

Un squelette est un fichier :

- dont le contenu est celui d'un MIME-TYPE avec éventuellement des interpolations de PHP ;
- dont le nom se termine par .html (raison historique, mais sans incidence sur le contenu) ;
- dont l'emplacement est la racine du site ou, afin d'organiser des *thèmes* , un répertoire indiqué dans SPIP_PATH (inspiré du PATH en Shell) lui-même défini dans le fichier ecrire/mes_options.php.

Ses filtres non standards peuvent être placés dans le fichier mes_fonctions.php ou *S*_fonctions.php, où *S* est le nom du squelette.

Principe des squelettes de mise en page

Un squelette de mise en page est une page HTML enrichie des entités suivantes :

- Des couples de balises ouvrante/fermante spécifiant une *boucle* :
  - dans la balise ouvrante, l'équivalent d'une requête SQL *select*,
  - entre les deux balises, le texte HTML devant être produit pour chacune des réponses de la requête ;
- Des *champs SQL* rendus accessibles par les requêtes ci-dessus, éventuellement transmis à des *filtres de mise en page* écrit en PHP ;
- D'autres balises (pour structuration) et champs (pour accéder à l'environnement global) ;
- Les boucles peuvent s'imbriquer, voire s'appeler récursivement.

(Une) Syntaxe des champs

Un champ est repéré par :

- Le signe # suivi d'une majuscule ou d'un souligné, et d'autres majuscules, soulignés ou chiffre : #TITRE
- Pour lever d'éventuelles ambiguïtés, le nom d'un champ peut commencer par le nom de la boucle qu'il référence, suivi d'un deux-points : #_main :TITRE ;
- Un champ peut être suivi d'une étoile pour désactiver le correcteur typographique et l'anti-XSS (*cross-site scripting*), à vos risque et périls : <a title='#URL_ARTICLE*'> ;
- Un champ non vide, peut être *étendu* à des textes HTML ou d'autre champs à émettre, extension délimitée par des crochets et des parenthèses : [(#SOUSTITRE)] (la *structure conditionnelle* de Spip).

(Une) Syntaxe des filtres

Un filtre de mise en page est une fonction PHP :

- s'appliquant à un champ étendu ou au résultat d'un autre filtre, les filtres étant séparés par un tube :
  [(#TITRE | trim | strtolower)]
- il peut recevoir plus d'un argument, en utilisant des accolades :
  [title='(#TITRE | substr{0,30})'] .
- comme les champs étendus, les filtres peuvent être imbriqués :
  [(#TITRE | substr{0,[(#TITRE | strpos{" "})]}].

Taxonomie des filtres

Spip offre plus d'une centaine de filtres qu'on peut classer ainsi :

- des transcodeurs pour éviter les mauvais choix d'encodage de caractères ou d'entités HTML, respecter les normes du W3C, prévenir les trous de sécurités XSS ;
- des correcteurs typographiques pour respecter les règles d'usage ;
- des ajusteurs d'images, de logo et de vignettes ;
- des visualiseurs de date et d'agenda, véritable système dans le système.

D’autres champs

En plus des champs SQL définis par boucles, Spip offre plusieurs champs spéciaux :

- Des champs définis lors de la configuration du site : #CHARSET, #LANG, #URL_SITE_SPIP, #NOM_SITE_SPIP... ;
- Des champs définis lors de l’éxécution du squelette : #TOTAL_BOUCLE, #COMPTEUR_BOUCLE, #DOSSIER_SQUELETTE... ;
- Des champs définis lors de la requête HTTP, notamment #ENV `{` *paramètre, défaut* `}` qui donne la valeur d’un paramètre passé dans l’URL ;
- Des champs regroupant plusieurs champs, notamment #PARAMETRES_FORUM construisant une QUERY_STRING nécessaire aux formulaires de réponse.

(Une) Syntaxe des boucles

Une *boucle* est indiquée par une balise commençant par la chaîne BOUCLE suivie :

- d'une suite de caractères alphanumériques indiquant le *nom de la boucle* ;
- puis d'une paire de parenthèses contenant le *type de la boucle* (en fait souvent le nom d'une table SQL) ;
- puis d'une conjonction de *critères* dénotés par des paires d'accolades ;

Une balise de boucle se présente donc ainsi :
<BOUCLE*nom*(*type*) `{critère 1}{critère 2}...{critère n}` >

Spip offre une trentaine de critères, souvent inspirés des clauses *where*, *limit* et *order* de SQL, avec jointures sur d'autres tables au besoin.

Exemples de boucle

- ```
  <BOUCLE1(AUTEURS)>
     <a href='mailto:#EMAIL'>#NOM</a><br />
  </BOUCLE1>
  ```

  liste, ligne par ligne, des liens de mail vers les auteurs ;

- ```
  <BOUCLE2(ARTICLES){par date}{inverse}{0,5}>
    <a href='#URL_ARTICLE'>#TITRE</a><br />
  </BOUCLE2>
  ```

  liste les titres des 5 derniers articles parus, avec un lien vers leur page ;

- ```
  <BOUCLE3(ARTICLES){id_rubrique}>
    <a href='#URL_ARTICLE'>#TITRE</a><br />
  </BOUCLE3>
  ```

  même chose, pour tous les articles d'une rubrique spécifiée par le contexte (notamment un éventuel id_rubrique mentionné dans l'URL).
  ```

(Une) Syntaxe complète des boucles

SPIP permet d'indiquer ce qu'on affiche avant et après la boucle au cas où elle contient au moins un résultat, et ce qu'on affiche sinon, avec la syntaxe :

```
<Bn> Avant
<BOUCLEn(TYPE){critère1}{critère2}...{critèrex}> Dedans
</BOUCLEn> Après
</Bn> Alternatif
<//Bn>
```

Avant est affiché avant les résultats de la boucle, et seulement s'il y en a.

Après est affiché après les résultats de la boucle, et seulement s'il y en a.

Alternatif est affiché seul, si et seulement si la boucle n'a pas de résultat.

Exemple de boucle complète

Le code Spip suivant :

```
<B4>
<ol>
  <BOUCLE4(ARTICLES){titre_mot=#ENV{cherche}}>
    <li><a href='#URL_ARTICLE'>#TITRE</a></li>
  </BOUCLE4>
</ol>
</B4>
Aucun article associé au mot-clé  "[<b>(#ENV{cherche})</b>]".
<//B4>
```

affichera une liste numérotée de tous les articles associés au mot-clé indiqué dans la variable d'URL cherche s'il y en a, ou le message *Aucun article associé au mot-clé "X"* s'il n'y en a pas.

Exemple de Squelette non HTML : ICS

Le squelette suivant :

```
<BOUCLE4(ARTICLES){statut=prop}>
BEGIN:VEVENT
SUMMARY:[(#TITRE|filtrer_ical)]
UID:article#ID_ARTICLE @ [(#URL_SITE_SPIP|filtrer_ical)]
DTSTAMP:[(#DATE|date_ical)]
DTSTART:[(#DATE|date_ical)]
URL:#URL_SITE_SPIP/articles.php3 ?id_article=#URL_ARTICLE
END:VEVENT
</BOUCLE4>
```

produit l’agenda des articles soumis à publication.

Boucles successives et imbriquées

En cas de boucles imbriquées, les plus internes héritent du contexte des englobantes.

Cet exemple liste toutes les rubriques de premier niveau, avec le titre de leurs brèves et celui de leurs sous-rubriques de niveau 2 précédé du rappel du titre de la rubrique mère :

```
<BOUCLE1(RUBRIQUES){racine}>
 #TITRE<br />
 <BOUCLE11(BREVES){id_rubrique}>#TITRE<br />
 </BOUCLE11>
 <BOUCLE12(RUBRIQUES){id_rubrique}>#1:TITRE/#TITRE<br />
</BOUCLE12>
</BOUCLE1>
```

Boucle récursive

Le type d'une boucle peut désigner non pas une table SQL mais une boucle englobante, qui est alors appelée récursivement. La récursion s'arrête grâce au critère *id_parent* qui compare le champ *id_parent* de la boucle courante avec la *clé primaire* de la boucle englobante.

```
<BOUCLE_forum(FORUMS){id_article}>#TITRE
  <B_reponses><ul>
  <BOUCLE_reponses(FORUMS){id_parent}><li>#TITRE
     <BOUCLE_R(boucle_reponses)></BOUCLE_R></li>
  </BOUCLE_reponses>
  </ul></B_reponses>
</BOUCLE_forum>
```

Déroulé d’une boucle récursive

Dans la boucle précédente, si l’on suppose que l’article a suscité 2 réactions, la première ayant elle-même provoqué une réaction, celle-ci à nouveau une autre et celle-ci à nouveau une autre, le code HTML produit aura la structure bien connue des navigateurs :

```
<ul><li> Réponse
  <ul><li>Réponse à la Réponse
    <ul><li>Réponse à la Réponse de la Réponse
        <ul><li>Réponse à la Réponse de la Réponse de la Réponse
          </li></ul>
      </li></ul>
   </li></ul>
</li><li>Réponse 2</li></ul>
```

Encore d’autres boucles

Il existe également une boucle de type HIERARCHIE qui remonte (ou redescend avec le critère inverse) la hiéarchie à partir d’une rubrique donnée.

Depuis Spip 1.8, il est possible de définir de nouvelles boucles, en décrivant dans les fichiers de configuration la déclaration SQL des tables.

Avec Spip 1.9, actuellement en version Alpha, cette description n’est même plus nécessaire, Spip effectuant une requête *showtable* lorsqu’un type de boucle lui est inconnu.

D’autres balises : multilinguisme et inclusion

Il existe encore d’autres entités en Spip :

- Les balises destinées à insérer du texte dans la langue principale du site, qui s’écrivent < :*intitulé* :>, l’ *intitulé* étant un index dans un tableau défini dans les fichiers ecrire/lang/*.php mais surchargeable par un fichier local.php dans son SPIP_PATH ;
- Les balises gérant le multilinguisme à l’intérieur de la page (en fonction des indications fournies par le navigateur), qui s’écrivent <multi>[fr]bonjour[de]Guten Tag</multi> ;
- Une balise d’inclusion pour insérer un squelette dans un autre squelette, ce qui permet de partager du code et de régler de manière différente la durée de vie dans le cache (en-tête longue et contenu bref typiquement) : <INCLURE(*nom*)*paramètres* >.

Problématique des balises de formulaires

Ce sont des champs dont la compilation est délicate car il faut à la fois :

- fournir un squelette purement HTML aux graphistes ;
- tenir compte à chaque requête des valeurs changeantes des paramètres HTTP ...
- ... tout en mettant en cache ce qui en est indépendant.

Le premier point est d'autant plus important que l'accessibilité des formulaires Web aux déficients visuels (voir http://www.w3.org/WAI et http://www.accessiweb.org) demande un travail délicat qui ne peut interférer avec celui du programmeur.

Conception des balises de formulaires

Une balise de formulaire *F* se déploit en 3 étapes :

1. A la compilation du squelette comportant *F*, production de l'appel de la fonction `F_stat` d'arguments les champs listés dans le tableau `F_collecte`
2. A l'exécution du squelette, l'application de `F_stat` renvoie :
   - une chaîne : insertion telle quelle ;
   - un tableau : insertion de <?php *F*_dyn(*tableau*) ?>
3. A la consultation du cache, l'application de `F_dyn` renvoie :
   - une chaîne : insertion telle quelle ;
   - un tableau *squelette, délai, contexte* : calcul du squelette avec `FORM` .

Les balises de formulaires standards

Spip offre une dizaine de formulaires réalisés selon cette technologie :

#FORMULAIRE_ADMIN
#FORMULAIRE_ECRIRE_AUTEUR
#FORMULAIRE_FORUM
#FORMULAIRE_FORUM_PREVISU
#FORMULAIRE_INSCRIPTION
#FORMULAIRE_LOGIN
#FORMULAIRE_LOGIN_FORUM
#FORMULAIRE_MENU_LANG
#FORMULAIRE_OUBLI
#FORMULAIRE_RECHERCHE
#FORMULAIRE_SIGNATURE
#FORMULAIRE_SITE

Conclusion

- Ce que Spip était hier : un générateur de *Webzine* ;
- Ce que Spip est aujourd'hui : un metteur en forme de tables SQL distantes ;
- Ce que Spip sera demain : un atelier de création de sites multi-fonctions.