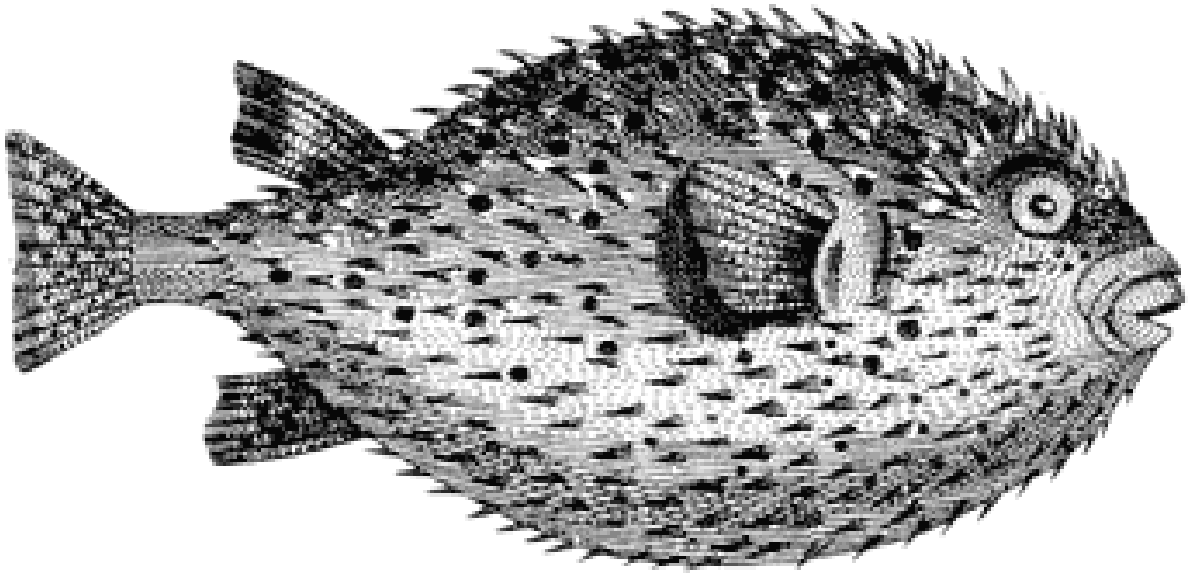
SPIP

Système de Publication pour l'Internet



SPIP pas à pas

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

1 Mon premier squelette

12 juin 2001 par l'équipe de SPIP

(je le sors du placard)

Comment écrire un premier squelette qui marchouille

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

*On supposera, pour commencer, que votre base SPIP contient au minimum une rubrique et deux articles **publiés**.*

Avant d'utiliser un squelette, il faut pouvoir l'appeler : créez à la racine de votre site un fichier `tutoriel.php3` contenant les lignes suivantes

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

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

A la racine du site, déposez un fichier « `tutoriel.html` », qui contient ce qui suit :

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

Puis rechargez la page `http://votresite.net/tutoriel.php3`. C'est mieux, n'est-ce pas ? SPIP est allé chercher le titre de l'article n° 1 de votre base, et l'a inscrit à la place de `#TITRE`.

Si ça ne fonctionne pas, vérifiez :

- ▷ que votre article n°1 est bien « publié » (et pas « en attente » ou « en cours de rédaction »).
- ▷ que la `<BOUCLE...>` ne commence pas sur le premier caractère du squelette (bug de SPIP jusqu'à la version 1.0beta24.)

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

```
<BOUCLE_article(ARTICLES){id_article=1}>
<H1>#TITRE</H1>
<B>#CHAPO</B>
<P align=justify>#TEXTE
</BOUCLE_article>
```

Ajoutez ensuite les champs manquants : `#SURTITRE`, `#LESAUTEURS`, `#SOUSTITRE`, `#NOTES`, etc.

Bien !

2 Un squelette, plusieurs articles

9 juin 2001 par l'équipe de SPIP

c'est à ça que ça sert . . .

Et voici le premier contexte.

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

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

Pour cela nous allons appeler notre page Web avec une variable `id_article=2` : pointez le navigateur sur l'URL `http://votresite.net/tutoriel.php3?id_article=2`.

S'affiche . . . l'article 1. Modifions dans le squelette `tutoriel.html` la ligne qui définit la « boucle article » :

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

(On remplace `{id_article=1}` par `{id_article}`.)

Voilà : `http://votresite.net/tutoriel.php3?id_article=2` vous donne l'article 2. Non ? Il devrait . . .

La `BOUCLE_article` s'exécute dans un contexte où `id_article=2`. Si on lui précise `{id_article=1}` elle va chercher l'article n° 1, mais si on lui demande `{id_article}`, elle va chercher l'article dont le numéro est précisé dans le contexte.

Cliquez maintenant sur :

- `http://votresite.net/tutoriel.php3?id_article=1`,
- `http://votresite.net/tutoriel.php3?id_article=2` et
- `http://votresite.net/tutoriel.php3`.

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

Bravo ! Votre squelette est maintenant « contextuel ».

3 Une rubrique

7 juin 2001 par l'équipe de SPIP

ou comment faire des listes du contenu de la base

Faire des listes avec une boucle SPIP

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

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

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

Là, on supprime carrément la condition `{id_article=1}`. Attention : cette BOUCLE va générer une page énorme si votre base contient déjà pas mal d'articles : mieux vaut prendre nos précautions et ajouter tout de suite 0,10 pour limiter aux 10 premiers articles ...

```
<BOUCLE_article(ARTICLES){0,10}>
```

Résultat : les titres des 10 premiers articles de votre base s'affichent, séparés par un saut de ligne. A partir de là, on voit comment on peut produire le sommaire d'une rubrique : affichons les 10 articles les plus récents appartenant à cette rubrique.

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

Prenons dans l'ordre :

- `{id_rubrique}` : ne prend que les articles appartenant à la rubrique `id_rubrique` (cf. ci-dessous pour que cette variable soit définie dans le contexte de notre `BOUCLE_article`).
- `{par date}{inverse}` : tri par date dans l'ordre décroissant ...
- `{0,10}` : ... et prend les 10 premiers résultats.
- Enfin, `<a href=#URL_ARTICLE>#TITRE</A>` va écrire non seulement le titre de l'article mais en plus créer un lien vers cet article.

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

`http://votresite.net/tutoriel.php3?id_rubrique=1`.

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

4 Boucles en boucles

5 juin 2001 par l'équipe de SPIP

plusieurs niveaux de lecture

Affichons sur une même page des éléments en provenance de plusieurs endroits.

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

Essayez !

Et voici une solution :

```
<BOUCLE_rubrique(RUBRIQUES){id_rubrique}>
<H1>#TITRE</H1>

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

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

On appelle la page avec `http://votresite.net/tutoriel.php3?id_rubrique=1`. Que s'est-il passé ici ?

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

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

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

5 Gérer le cache

3 juin 2001 par l'équipe de SPIP

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

Le cache, ou comment faire un site dynamique qui ne bouge pas trop.

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

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

◇ ◇ ◇

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

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

Notre page web n'est donc recalculée que si, lorsqu'un visiteur la demande, sa version cachée date de plus d'une heure (soit 3600 s.). Sinon, SPIP lit simplement le contenu du fichier caché[3], et renvoie le résultat sans se connecter à la base de données (sauf pour y insérer un « hit » dans les statistiques).

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

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

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

- Depuis le back-office, cliquer sur « Voir en ligne » déclenche le recalcul pour les pages correspondant à `#URL_ARTICLE` ou `#URL_RUBRIQUE` de l'article ou de la rubrique correspondante. C'est le cas le plus courant. Mais sinon ?
- Dans la partie « Sauvegarde/Restauration » du back-office, un bouton « vider le cache » efface tous les fichiers cachés (utile si vous faites plein de modifications et avez un site très complexe, à éviter sinon).
- Toutefois, la solution la plus simple est de demander à SPIP, dans la page d'accueil du back-office, de vous « poser un cookie ». Ce cookie s'incrustera sur votre navigateur, et SPIP vous reconnaîtra au moment de vous envoyer la page dans le site public : il vous proposera alors, en bas de page, un bouton « Recalculer cette page ».

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

Derniers détails :

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

6 Des filtres

1er juin 2001 par l'équipe de SPIP

Subtilités squelettiques

Les filtres transforment le contenu de la base de données en code HTML présentable.

Si les `BOUCLES` *permettent de structurer la page de manière logique, reste à présenter les données de manière esthétique. Question dizahïgne SPIP ne peut rien pour vous, mais sachez user des ces filtres* …

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

- les filtres les plus utilisés (ils sont appelés systématiquement) sont `|typo` et `|propre` ; le premier est un correcteur typographique, dont la mission principale est d'ajouter des espaces insécables où il en faut (*cf.* aide en ligne de SPIP) ; le second s'intéresse aux paragraphes, aux raccourcis SPIP (italiques, gras, intertitres, notes de bas de page, etc.) - il n'est appliqué par défaut qu'aux champs longs (`#TEXTE`, `#CHAPO`, etc.)
- d'autres filtres sont très utiles : citons `|majuscules` (à la fonctionnalité évidente), `|justifier` ou `|aligner_droite` (qui définissent l'alignement du texte), ou encore l'ésotérique `|saison` (qui affiche « été » si la variable est une date comprise entre le 21 juin et le 20 septembre) …

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

On peut enchaîner les filtres dans un pipeline : ainsi `[(#DATE_REDAC|saison|majuscules)]` affichera-t-il « HIVER ».

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

◇ ◇ ◇

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

```
[<TABLE border=1><TR><TD>
(#DATE|affdate)
</TABLE>]
```

Ici le filtre `|affdate` affiche la date au format « `1er juillet 2001` », et renvoie une chaîne vide si la date est « `0000...` ». Les crochets délimitent ce qu'il faut afficher *si le resultat entre parenthèses n'est pas une chaîne vide*.

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