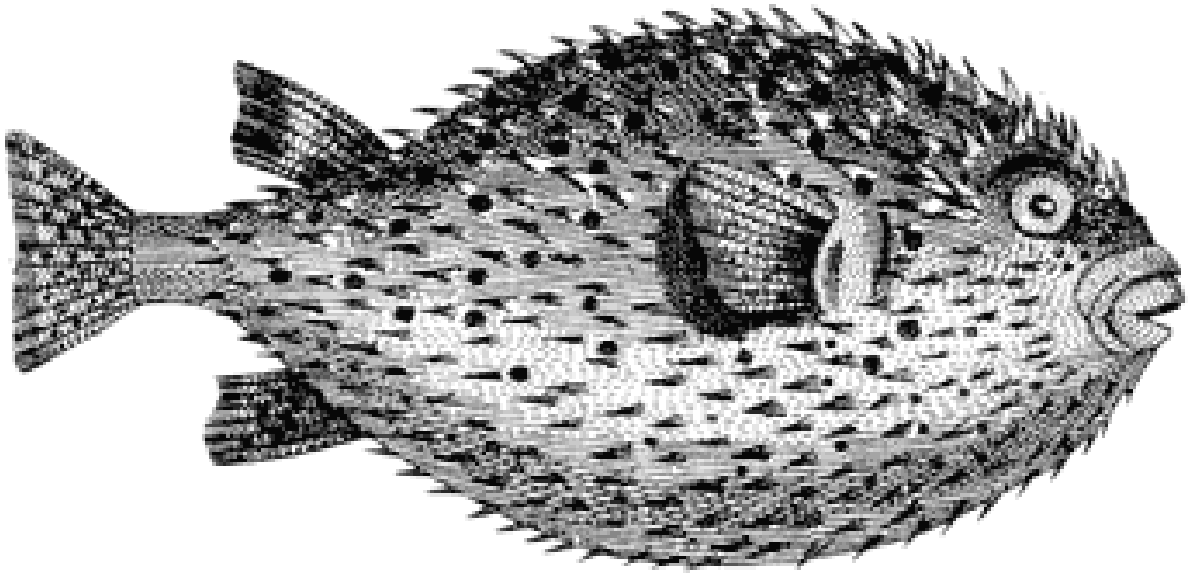
SPIP

Système de Publication pour l'Internet



Trucs et astuces

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 Présenter les résultats d'une recherche par secteurs

11 avril 2002

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.

2 Afficher des éléments par lignes dans un tableau

11 avril 2002

Par exemple, on peut vouloir créer un tableau contenant les titres des articles d'une rubrique agencés sur trois colonnes, 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

L'astuce consiste à jouer à la fois avec les doublons et avec les boucles récursives. On construit une première boucle qui affiche les trois premiers articles de la rubrique *une fois les doublons éliminés*. On voit qu'il suffit ensuite de réafficher cette boucle à chaque fois qu'il reste des articles pour afficher graduellement tous les articles, ceux déjà affichés venant à chaque fois grossir les rangs des doublons. Pour cela, dans le code conditionnel de cette boucle, on ajoute un appel récursif vers la boucle elle-même : elle sera affichée tant qu'elle produit des résultats.

```
<table>
<B_ligne>
  <tr>
    <BOUCLE_ligne (ARTICLES) {id_rubrique} {doublons} {par titre} {0,3}>
      <td width="33%">
        <a href="#URL_ARTICLE">#TITRE</a>
      </td>
    </BOUCLE_ligne>
  </tr>

    <BOUCLE_ligne_suite (BOUCLE_ligne)></BOUCLE_ligne_suite>
```

```
</B_ligne>
</table>
```

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.

3 Afficher le nombre de messages du forum lié à un article

11 avril 2002

C'est un poil acrobatique.

À première vue, il est très simple de connaître le nombre d'éléments d'une boucle : il suffit d'utiliser le code SPIP : `#TOTAL_BOUCLE`. Ce code peut s'utiliser non seulement à l'intérieur de la boucle, mais aussi (c'est le seul 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).

Nous devons créer une boucle de type FORUMS, liée à un article, de façon à compter son nombre de résultats.

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}>
```

contient uniquement les messages qui répondent à l'article. Habituellement, pour accéder aux réponses à ces messages, on inclut une seconde boucle à l'intérieur de celle-ci…Ici, nous voulons que la boucle sélectionne absolument tous les messages attachés 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}>
```

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 pas le titre des messages, on évitera même de lui faire afficher des espaces ou des retours à la ligne (sinon votre page HTML contiendra des dizaines de lignes vides, inélégantes) ; l'intérieur de la boucle n'affiche donc rigoureusement rien, mais on doit afficher, après la boucle, le nombre de résultats.

Une subtilité à bien comprendre : le texte conditionnel alternatif s'affiche *si la boucle n'affiche rien* ; 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 contributions au forum ») dans ce cas.

```
<BOUCLE_nb_forums(FORUMS){id_article}{plat}></BOUCLE_nb_forums>
  [Il y a (#TOTAL_BOUCLE) contribution(s) au forum.]
<//B_nb_forums>
```

4 Afficher les derniers articles de vos rédacteurs par rubrique

1er mai 2002 par Ecran de Bureau

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. Vous pouvez voir un exemple ici[3].

Comment procéder ?

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

4.1 Création du fichier myauteur.php3

Dans le fichier `myauteur.php3` mettre le code suivant :

```
<?php

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

include ("inc-public.php3");

?>
```

4.2 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>
```

[3] http ://zonewebmaster.membres.jexiste.org/

```
<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>
```

4.3 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* ou sera inscrit les derniers articles écrits par l'auteur.

Le lien devra être écrit de la manière suivante :

```
<a href="myauteur.php3?id_auteur=#ID_AUTEUR">nom du lien</a>
```

5 Liens « Article précédent », « Article suivant »

1er mai 2002

Le but est d'afficher des liens vers l'article précédent, puis vers l'article suivant dans la page d'un article. Les squelettes par défaut présentent déjà la liste des articles présents dans la même rubrique, il s'agit donc ici d'obtenir une présentation plus précise.

Il n'est pas possible d'obtenir cet effet directement avec le code SPIP lui-même. Cependant, il est simple à réaliser en utilisant un peu de PHP.

Nous nous placerons à l'intérieur de la boucle principale de la page, celle qui affiche l'article « principal ». Pour simplifier l'explication, nous résumerons le squelette de notre page d'article au code suivant :

```
<html>
<body>

<BOUCLE_principale(ARTICLES){id_article}>

  <h1>#TITRE</h1>
  #TEXTE

</BOUCLE_principale>

</body>
</html>
```

Nous allons avoir besoin, dans le code PHP qui affiche et sélectionne les articles précédents et suivants, de mémoriser le numéro de l'article principal. Pour cela, ajoutons simplement la définition de la variable PHP `$ze_article` à l'intérieur de la `BOUCLE_principale` :

```
<html>
<body>

<BOUCLE_principale(ARTICLES){id_article}>
  <?
    $ze_article = "#ID_ARTICLE";
  ?>

  <h1>#TITRE</h1>
  #TEXTE

</BOUCLE_principale>

</body>
</html>
```

Nous plaçons l'`#ID_ARTICLE` entre guillements, de façon à éviter une erreur PHP lorsqu'on charge la page sans numéro de rubrique.

Il suffit maintenant d'insérer les boucles affichant l'article précédent et l'article suivant :

```
<html>
<body>

<BOUCLE_principale(ARTICLES){id_article}>
  <?
    $ze_article = "#ID_ARTICLE";
  ?>
```

```
  <BOUCLE_art_prec(ARTICLES){id_rubrique}{par titre}>
    <?
    if ($ze_article == #ID_ARTICLE) {
      echo stripslashes("$art_prec");
    }
    $art_prec = "<div>Article précédent : <a href='#URL_ARTICLE'>
                                          [(#TITRE|addslashes)]</a></div>";
    ?>
  </BOUCLE_art_prec>

  <h1>#TITRE</h1>
  #TEXTE

  <BOUCLE_art_suiv(ARTICLES){id_rubrique}{par titre}>
    <?
    $texte_suiv = "<div align='right'>Article suivant : <a href='#URL_ARTICLE'>
                                          [(#TITRE|addslashes)]</a></div>";
    if ($ze_article == $article_prec) {
      echo stripslashes("$texte_suiv");
    }
    $article_prec = #ID_ARTICLE;
    ?>
  </BOUCLE_art_suiv>

</BOUCLE_principale>

</body>
</html>
```

Les boucles « `art_prec` » et « `art_suiv` » s'effectuent autant de fois qu'il y a d'articles dans la même rubrique que notre article principal. À chaque occurence, elles mémorisent le texte et le lien de l'article, sans l'afficher ; ce lien n'est affiché que lorsque l'identifiant de l'article de l'occurence correspond à l'identifiant de notre article principal.

Pour l'article précédent, c'est très simple :

- ▷ on teste le numéro de l'article, et on affiche un texte éventuel ;
- ▷ puis on mémorise le texte du lien ; ainsi, dans le test, c'est bien une valeur récupérée dans l'occurence précédente (donc l'article précédent) qui est affichée.

Pour l'article suivant, c'est à peine plus compliqué :

- ▷ on mémorise le lien vers l'article de l'occurence ;
- ▷ on teste si le numéro de l'occurence *précédente* correspond à l'article principal ;
- ▷ on mémorise le numéro de l'article en tant que `$article_prec`, pour le test effectué lors de l'occurence suivante.

Notez au passage qu'on mémorise une variable construite sur `[(#TITRE|addslashes)]`. En effet, sans cela, on obtiendrait un message d'erreur PHP pour les articles contenant des guillements. Lorsqu'on affiche ce texte, on le passe donc par stripslashes pour supprimer les caractères indésirables.

Important : pour que votre navigation reste cohérente sur l'ensemble du site, il convient de bien choisir le critère de classement des boucles `BOUCLE_art_prec` et `BOUCLE_art_suiv`. Ici, nous avons indiqué `{par titre}`, car dans notre site nous affichons les listes d'articles classés selon leurs titres. Si vous préférez classer les articles selon leur date, du plus récent au plus ancien, il faut modifier ces boucles avec les critères : `{par date}{inverse}`. (Pour être cohérent, il convient d'adopter les mêmes critères de classement que la boucle « Dans la même rubrique », ou que la boucle d'affichage des articles dans le squelettes des rubriques.)

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

11 avril 2002

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. Le critère de sélection qui nous intéresse ici est : « `age` ».

```
<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 faut utiliser le critère « `age_redac` » au lieu de « `age` ». L'âge est indiqué en nombre de jours.

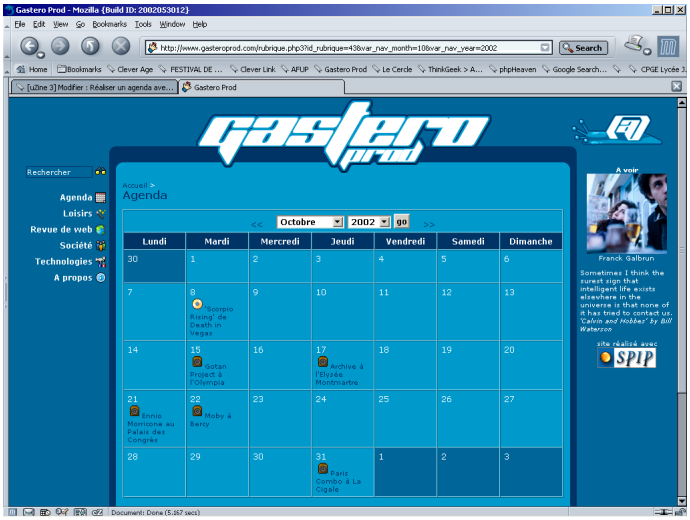
Notons que cette manipulation est possible avec tous les types de données auxquels est associée une date (brèves, forums…).

7 Réaliser un agenda avec SPIP

16 novembre 2002 par Nicolas "Brush" Hoizey

Un joli squelette avec du PHP dedans …

[SPIP 1.4.2] Dans cet article nous allons utiliser SPIP pour présenter un agenda d'événements sous forme de calendrier. Plutôt que d'intégrer un logiciel externe spécialisé, voyons comment réaliser cela avec un squelette mélant boucles SPIP classiques, et code PHP.



Exemple d'agenda
L'agenda de Gastero Prod[4]

[4] http ://www.gasteroprod.com/rubrique.php3 ?id_rubrique=43

7.1 De quelles informations avons-nous besoin ?

La première étape consiste à déterminer comment les informations seront stockées dans SPIP, et donc saisies par les rédacteurs.

L'objectif principal étant qu'un rédacteur puisse définir une date à laquelle un événement donné doit avoir lieu, nous avons besoin des éléments suivants :

- au moins un champ de type texte
- un champ de type date

Pour le champ texte, c'est simple, et nous allons même pouvoir nous régaler. SPIP nous en propose au moins 3 très utiles et systématiquement présents quel que soit le paramétrage du site : le titre, le descriptif et le texte.

7.2 La date de publication antérieure ou la *vraie* date de publication ?

La date de publication d'un article est déterminée automatiquement lors de sa validation, ce qui signifie que l'auteur ne peut pas la préciser à l'avance. Elle peut en revanche être modifiée par le valideur, mais elle n'est donc pas vraiment utile pour indiquer une date d'événement - surtout si on a configuré son site pour que SPIP ne publie pas les articles post-datés.

Heureusement, SPIP propose aussi une *date de publication antérieure*, normalement utilisée pour indiquer à quelle date a été publié un article repris ultérieurement dans SPIP, et qui peut être définie librement par le rédacteur[5].

C'est cette date que nous allons utiliser pour définir les événements de l'agenda.

Utilisation de la date de publication antérieure

7.3 Organiser les événements

Si l'agenda doit contenir beaucoup d'informations de natures diverses, il peut s'avérer utile de les qualifier (concert, film, réunion ?) pour les mettre en évidence et éventuellement ensuite les afficher de manière sélective.

Dans SPIP deux méthodes de classement viennent immédiatement à l'esprit :

- classement par rubrique (et éventuellement sous-rubriques) ;
- classement par mot-clé.

Dans cet exemple, nous allons utiliser les rubriques, ce qui nous permettra de constituer toute une arborescence pour stocker les articles événement ; cela va nous permettre de filtrer les types d'événements avec une notion de granularité progressive (c'est pas chic ça ?).

Dans l'agenda de Gastero Prod[6], le rubricage n'est pas très développé, mais il est déjà possible de sélectionner uniquement les événements musicaux[7], et éventuellement de filtrer encore plus en ne considérant que les concerts[8].

[5] Vérifiez que vous avez configuré votre site pour gérer ce type de date supplémentaire.
[6] http ://www.gasteroprod.com/rubrique.php3 ?id_rubrique=43
[7] http ://www.gasteroprod.com/rubrique.php3 ?id_rubrique=46
[8] http ://www.gasteroprod.com/rubrique.php3 ?id_rubrique=44

Une hiérarchie de rubriques pour les thèmes

Voilà, les événements sont créés dans une arborescence de rubriques thématiques, sont positionnés à une date, passons aux choses sérieuses :

7.4 L'affichage de l'agenda, PHP à la rescousse

Nous souhaitons afficher l'agenda sous la forme standard d'un tableau mensuel dont chaque colonne représente un jour de la semaine. Nous souhaitons aussi pouvoir naviguer d'un mois à l'autre, voire même choisir directement un mois arbitraire.

Tout cela n'est pas possible directement si l'on utilise exclusivement les boucles de SPIP, d'où l'usage de PHP.

Voici une explication, pas à pas, de tous les éléments du squelette (vous trouverez en fin d'article le squelette complet à télécharger) :

Préparation des informations à traiter

Définissons déjà deux tableaux contenant les noms des 12 mois de l'année et des 7 jours de la semaine.

```
$months = array('', 'Janvier', 'Février', 'Mars', 'Avril', 'Mai', 'Juin', 'Juillet', 'Août',
                                                'Septembre', 'Octobre', 'Novembre', 'Décembre');
$days = array('Dimanche', 'Lundi', 'Mardi', 'Mercredi', 'Jeudi', 'Vendredi', 'Samedi');
```

Voici maintenant une petite fonction toute bête permettant de simplifier la génération d'une date au format *timestamp* pour les plus fainéants d'entre nous qui ne veulent pas avoir besoin de préciser que l'heure, les minutes et les secondes sont nuls ... ;)

```
function mkdate($month, $day, $year)
{
  return mktime(0, 0, 0, $month, $day, $year);
}
```

Gestion de la navigation

Les paramètres de navigation sont passés dans l'URL au travers des variables `var_nav_month` et `var_nav_year`[9]. La première chose à faire en début de script est donc de déterminer quel mois il faut afficher, le mois courant étant pris par défaut si aucun n'est précisé :

```
if(isset($GLOBALS['var_nav_month'])) {
  $cal_day = mkdate($GLOBALS['var_nav_month'], 1, $GLOBALS['var_nav_year']);
} else {
  $cal_day = time();
}

$D = intval(date('d', $cal_day));
$M = intval(date('m', $cal_day));
$Y = intval(date('Y', $cal_day));
```

Récupération de la liste des événements

Nous allons commencer par placer dans une variable PHP de type tableau nommée `$events` la liste des événements susceptibles d'être affichés dans le calendrier du mois demandé, en fonction bien sûr de la restriction thématique correspondant à la rubrique courante :

```
$events = array();
?>

<BOUCLE_principale(RUBRIQUES){id_rubrique}{doublons}>

<p>
<span class="small">
<a href="#URL_SITE_SPIP">Accueil</a> >
<BOUCLE_hier1(HIERARCHIE){doublons}>
  <a href="#URL_RUBRIQUE">#TITRE</a> >
</BOUCLE_hier1>
</span>
<br />
[<span class="title">(#TITRE)</span>]
</p>

<BOUCLE_evenements(ARTICLES){branche}>
  <?php
  $date = ereg_replace("^([0-9]{4})-([0-9]{2})-([0-9]{2}).*$", "\\1\\2\\3", '#DATE_REDAC');
  if ($date > date("Ymd", mkdate($M, $D - 31, $Y)) &&
                                          $date < date("Ymd", mkdate($M, $D + 31, $Y))) {
    if (!isset($events[$date])) {
      $events[$date] = array();
    }
    $events[$date][] = array('link' => '#URL_ARTICLE', 'title' => '[(#TITRE|texte_script)]',
                                                          'logo' => "#LOGO_ARTICLE_RUBRIQUE");
  }
  ?>
</BOUCLE_evenements>
```

[9] On utilise la notation var_xxx pour indiquer à SPIP de ne pas prendre en compte ces variables pour le calcul du cache de la page, ce qui permet à la fois de limiter l'impact sur le cache (un seul fichier cache traitera toutes les pages de l'agenda) et d'accélérer l'affichage (puisque SPIP n'aura besoin d'aller chercher les données dans la base qu'une seule fois pour toutes ces pages).

La sélection sur les dates est en fait réalisée en PHP sur la liste complète des articles renvoyés par SPIP sans contrôle de date.

Affichage de la navigation

Nous utilisons ici des liens vers les mois précédent et suivant, et un petit formulaire avec les listes des 12 mois et des années de 1995 à 2010.

```
<table width="100%" cellpadding="5" cellspacing="0" align="center" border="1" class="small">
<form method="get">
<tr>
  <th colspan="7">
    <a href="/rubrique.php3?id_rubrique=#ID_RUBRIQUE&var_nav_month=<?=($M - 1)?>
                                                &var_nav_year=<?=$Y?>"><b>&lt;&lt;</b></a>
       
    <input type="hidden" name="id_rubrique" value="#ID_RUBRIQUE" />
    <?php
    echo '<select name="var_nav_month">';
    for($i = 1; $i < 13; $i++) {
      echo '<option value="'.$i.'"'.($i == $M ? ' selected="selected"' : '').'>'.
                                                                    $months[$i].'</option>';
    }
    echo '</select>';
    echo ' <select name="var_nav_year">';
    for($i = 1995; $i < 2011; $i++) {
      echo '<option value="'.$i.'"'.($i == $Y ? ' selected="selected"' : '').'>'.$i.'</option>';
    }
    echo '</select> ';
    ?>
    <input type="submit" value="go" />
       
    <a href="/rubrique.php3?id_rubrique=#ID_RUBRIQUE&var_nav_month=<?=($M + 1)?>
                                                &var_nav_year=<?=$Y?>"><b>&gt;&gt;</b></a>
  </th>
</tr>
</form>
```

Affichage des noms de jours en en-tête de colonnes

```
<tr>
  <?php
  for($i = 1; $i < 8; $i++) {
    echo '<th width="14%" class="calendar_head">'.$days[$i%7].'</th>';
  }
```

Affichage des derniers jours du mois précédent

Dans un calendrier mensuel présenté sous forme de tableau, les derniers jours du mois précédent et les premiers du mois suivant sont en général présentés pour que les semaines soient complètes.

```
$TempD = 1;
  if(date('w', mkdate($M, 1, $Y)) != 1) {
```

```
    echo '</tr><tr>';
    $tmp = '';
    while(date('w', mkdate($M, $TempD, $Y)) != 1) {
      $TempD--;
      $case = '<td width="14%" height="50" valign="top" class="calendar_not_this_month">';
      $case .= date('j', mkdate($M, $TempD, $Y));
      $date = date('Ymd', mkdate($M, $TempD, $Y));
      if (isset($events[$date])) {
        while (list(, $event) = each($events[$date])) {
          $case .= '<br />'.$event['logo'].'<a href="'.$event['link'].'" class="small">'.
                                                                    $event['title'].'</a>';
        }
      }
      $case .= '</td>';
      $tmp = $case.$tmp;
    }
    echo $tmp;
  }
```

Affichage du reste du calendrier

```
$TempD = 1;
  while((date('m', mkdate($M, $TempD, $Y)) == $M) || (date('w', mkdate($M, $TempD, $Y)) != 1)) {
    if(date('w', mkdate($M, $TempD, $Y)) == 1) {
      echo '</tr><tr>';
    }
    echo '<td width="14%" height="50" valign="top" class="calendar_'.
     (date('m', mkdate($M, $TempD, $Y)) != $M ? 'not_' : '').'this_'.
     (date('Ymd', mkdate($M, $TempD, $Y)) == date('Ymd') ? 'day' : 'month').'">';
    echo date('j', mkdate($M, $TempD, $Y));
    $date = date('Ymd', mkdate($M, $TempD, $Y));
    if (isset($events[$date])) {
      while (list(, $event) = each($events[$date])) {
        echo '<br />'.$event['logo'].'<a href="'.$event['link'].'" class="small">'.
                                                          $event['title'].'</a>';
      }
    }
    echo '</td>';
    $TempD++;
  }
  ?>
</tr>
</table>
```

Affichage de l'arborescence thématique de l'agenda

```
<?php
function depth($depth, $type)
{
  for($i = 0; $i < $depth; $i++) {
    echo '<img src="/images/dot.gif" width="15" />';
  }
}
```

```
$depth = 1;
?>

<p class="big">Accès direct aux thèmes, et légende des icônes :</p>
<BOUCLE_sous_rubriques1(RUBRIQUES){id_parent=43}{par titre}>
  <p>
  #LOGO_RUBRIQUE
  <a href="#URL_RUBRIQUE">#TITRE</a>
  <B_sous_rubriques>
    <BOUCLE_sous_rubriques(RUBRIQUES){id_parent}{par titre}>
      <br />
      <?php depth($depth, 'rub'); ?>
      #LOGO_RUBRIQUE
      <a href="#URL_RUBRIQUE">#TITRE</a>
      <B_sous_rubriques_2>
        <?php $depth++; ?>
        <BOUCLE_sous_rubriques_2(BOUCLE_sous_rubriques)>
        </BOUCLE_sous_rubriques_2>
        <?php $depth--; ?>
      </B_sous_rubriques_2>
    </BOUCLE_sous_rubriques>
  </B_sous_rubriques>
  </p>
</BOUCLE_sous_rubriques1>
```

Et bien sûr, fermeture finale de la boucle principale :

```
</BOUCLE_principale>
```

7.5 Un peu de personnalisation visuelle

Comme vous pouvez le voir dans le code proposé, la présentation est personnalisable via quelques styles.

Voici l'extrait de feuille de style de Gastero Prod correspondant à l'agenda :

```
.calendar_head
{
  background-color: #003366;
}

.calendar_this_day
{
  background-color: #00bbee;
}

.calendar_this_month
{
  background-color: #0099cc;
}

.calendar_not_this_month
{
  background: #006699;
}
```

7.6 Le code complet du squelette

Et voilà donc le code complet du squelette, à recopier et diffuser sans retenue !

agenda.html
Le fichier HTML du squelette[10]
(Zip, 1.6 ko)

8 Remplir les meta-tags HTML des pages d'article

11 avril 2002

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)]
```

…

[10] http ://www.uzine.net/IMG/zip/doc-710.zip

9 Un menu déroulant pour présenter une liste d'articles

11 avril 2002

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.

Index