Manuel de « référence »

Version hors-ligne du site Spip.net

Tome 3c

Guide du webmestre et du bidouilleur

Tutoriel : utilisation avancée des boucles et des mots-clés

Les plugins pas à pas

30 septembre 2007 Rabillard Fabien

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

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

Le document que vous avez entre les mains ou sur votre écran est un copier/coller du site Spip.net effectué entre le 28 et le 30 septembre 2007.
Ce document est destiné a ceux qui n'ont pas de connexion internet ou qui préfèrent avoir une version papier. Mais si vous voulez avoir la dernière version en ligne et à jour rendez vous sur le site Spip.net.

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

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

Bonne lecture. Fabien.

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

Tutoriel : utilisation avancée des boucles et des mots-clés

Attention : ce tutoriel est ancien et contient des indications qui ne sont pas valables avec l'actuelle version de SPIP !

Ce tutoriel vous explique diverses méthodes destinées à dépasser les limites apparentes de SPIP.
Attention : il s'adresse à des utilisateurs déjà expérimentés.

Introduction

SPIP est un produit limité. Il y a des choses qu'il fait très bien, d'autres qu'il ne peut pas réaliser. Essayez par exemple de créer un site consacré au cinéma, avec des liens sur tous les noms (acteurs, réalisateur, équipe technique, à la façon des fiches de l'Internet Movie Database) vers d'autres films qu'ils ont réalisé, et vous réaliserez que SPIP n'a vraiment pas été conçu pour cela ! À l'inverse, créer et gérer un site de contenu éditorial à la structure simple, à la façon d'uZine, est très simple.

Cependant, entre ces deux extrêmes (des sites irréalisables avec SPIP aux sites pour lesquels SPIP est particulièrement adapté), il existe une multitude d'usages, de besoins, facilement réalisables, mais à priori inaccessibles avec la version standard des squelettes.

Certains webmestres (fort courageux), confrontés aux limites apparentes du produit, se lancent bille en tête dans le code source de SPIP dans le but de l'adapter à leurs besoins spécifiques. Si cette attitude très « *open source* » est louable, en revanche elle prive ces webmestres de la possibilité de suivre les évolutions du logiciel standard, et requiert des connaissances poussées en PHP.

Or, et c'est le but de ce tutorial, il existe de nombreuses possibilités pour dépasser les limites de SPIP, uniquement par une utilisation personnalisée des squelettes et de la structure du site.

À qui s'adresse ce document ?

Le présent tutorial est destiné aux webmestres qui désirent dépasser certaines limites apparentes de SPIP. Il est donc **impératif d'avoir déjà compris le fonctionnement des squelettes et des boucles** qui gèrent l'interface publique.

Si vous débutez, commencez avec le document *SPIP pas à pas*, destiné aux webmestres qui s'initient au fonctionnement des squelettes.

Si vous savez déjà bien utiliser les squelettes, nous vous conseillons d'avoir à portée de la main une version imprimée du *Manuel de référence*.

Quelle version de SPIP ?

Les exemples donnés ici utilisent des fonctionnalités présentes à partir de la version SPIP 1.3. Certaines peuvent être réalisées avec des versions précédentes, mais l'utilisation de la version 1.3 offre plus de souplesse.

Quelles autres connaissances techniques sont nécessaires ?

Ce tutorial est très progessif. Cependant, il présente la construction pas à pas de squelettes complets. Outre la compréhension du mécanisme général des squelettes (et du système de boucles), il convient de comprendre le code HTML. Le HTML utilisé ici sera volontairement rudimentaire mais, si vous n'avez jamais réalisé une page Web autrement qu'avec un logiciel Wysiwyg, vous risquez de souffrir.

Nous n'aborderons pas ici l'utilisation de PHP dans les squelettes ; la connaissance de ce langage est donc ici inutile.

Les impératifs

- Ne pas toucher à structure de SPIP lui-même. À aucun moment nous n'irons modifier le code

source du produit, ni les tables de la base de données utilisées par SPIP. Cela garantira que les fonctionnalités apportées ici resteront compatibles avec les futures évolutions du logiciel, et que l'interface privée conservera sa cohérence.

- Réaliser un site dont l'interface de navigation reste cohérente. Un des buts ici étant de réaliser une navigation plus riche que celle proposée par les squelettes standards, il ne faut pas à l'inverse que l'interface devienne totalement incompréhensible pour le visiteur. (Vous verrez cependant que nous n'avons pas ici développé le graphisme de nos pages ; graphiquement, l'interface obtenue à la fin de ce tutorial sera hideuse... mais en revanche, les liens entre les différents éléments structurels du site seront présents.)

- Réaliser un site dont les mises à jour restent simples. Ceux qui gèrent le site depuis l'espace privé ne doivent pas passer plusieurs heures pour ajouter un nouvel article... Il s'agit bien de profiter des automatismes liés à un site dynamique.

Comment utiliser ce tutorial ?

Ce tutorial est très progressif, et les codes fournis sont complets : vous n'y trouverez pas des « bouts de code » sortis de nulle part à recopier, au contraire le code complet des squelettes se construira au fur et à mesure des articles. La méthode retenue ici est la création de pages de squelette de plus en plus complexes, chaque étape étant expliquée.

Nous vous conseillons donc de suivre ce tutorial dans l'ordre de ses articles, et de créer les fichiers en même temps que les explications, en intégrant à chaque fois les variations données en exemple.

Vous trouverez ces exemples beaucoup plus clairs si vous réalisez vous-même les pages présentées ici. Le code vous semblera plus simple, puisqu'à chaque nouvel élément intégré, vous verrez immédiatement l'impact des modifications.

Tous les squelettes réalisés dans ce tutorial sont regroupés dans nos archives. Vous pouvez les télécharger dès à présent, mais nous vous conseillons néanmoins de réaliser ces squelettes *ab initio* en suivant ce tutorial : si vous travaillez directement à partir des squelettes terminés, vous perdez la construction progressive et vous risquez de ne pas comprendre leur structure.

La progression de ce tutorial

- Les premiers articles (de « Le but du jeu » à « Écrire des articles ») insistent sur l'importance bien penser la structure de son site et de définir la constitution des articles avant de démarrer un site complexe. Techniquement, cette partie est à la portée de tous les utilisateurs de SPIP (on peut donc la lire même si on n'a aucune connaissance du fonctionnement des squelettes).

Cette partie vous semblera peut-être un peu trop simple (il n'y a aucune astuce technique). Elle introduit cependant l'utilisation des mots-clés comme outil de structuration du site. Surtout, elle permet de comprendre que la structure du site est primordiale pendant le démarrage d'un projet de site ambitieux. Cette étape, lorsqu'elle est négligée (et elle l'est souvent), conduit à des sites qui deviennent ingérables et dont l'interface publique est incohérente (ces problèmes se posent rarement lorsque l'on démarre le site, avec une poignée d'articles). Enfin, nous y insistons (longuement...) sur le fait que les choix techniques (programmation des squelettes, utilisation des mots-clés) dépendent directement des choix éditoriaux (cette évidence étant primordiale avec un système de publication).

- Les articles suivants (de « Première version du squelette des articles » à « Le site complet ») proposeront la création pas à pas des squelettes. Il faut à partir de ce moment avoir un bonne connaissance du système de boucles de SPIP. Les difficultés à ce stade seront d'ordre logique, et non informatiques. Volontairement, nous fabriquerons les squelettes en plusieurs étapes successives : au départ des squelettes extrêmement simples, nous y reviendrons, en ajoutant plus de complexité dans la structure des boucles.

Vous trouverez dans cette partie quelques astuces logiques dans la gestion des boucles ; surtout, vous y

verrez l’utilisation des mots-clés comme éléments de structuration de la navigation.

- Nous obtiendrons à la fin de ce tutorial un site complet et exploitable (moyennant la création d’une interface graphique un peu plus élaborée). Cependant, le dernier article (« Le site complet »), vous fournira quelques idées de développements supplémentaires que vous pourrez ajouter à votre site, en exploitant les principes expliqués ici.

Le but du jeu : un site consacré aux jeux vidéo

Les explications qui vont suivre sont basées sur la réalisation d'un site consacré aux jeux vidéo.

Pourquoi les jeux vidéo ?

Une notion à ne jamais perdre de vue lorsqu'on réalise des squelettes avec SPIP, c'est qu'il s'agit de présenter un contenu éditorial (penser en terme de possibilités techniques ou de base de données est le meilleur moyen de créer un site à l'interface incompréhensible). Il faut donc, en permanence, utiliser un vocabulaire et des images directement liées au contenu que l'on veut présenter.

Nous devions donc, pour que ce tutorial soit lisible, nous fixer un objectif *éditorial*, afin de pouvoir présenter chaque bidouille technique comme une réponse à un besoin éditorial.

Un site de jeux vidéo offre, pour notre démonstration, plusieurs avantages :

- le vocabulaire associé aux contenus de ce type de site est connu de tous,
- la navigation sur ces sites est très souple (il existe de nombreuses façons différentes de naviguer sur de tels sites) ;
- c'est l'exemple-type du site impossible à réaliser avec les squelettes standards de SPIP (sauf à se contenter d'un site très en deça de ce que l'on attend de ce genre de contenu) ;
- en revanche, c'est un très bon exemple de ce qu'il est possible de faire pour dépasser les limites de SPIP, tout en conservant la cohérence et la simplicité de son utilisation.

Quelles sont les difficultés que présente un tel site ?

La principale difficulté concerne le **rubriquage** des articles :

- chaque jeu fait l'objet de plusieurs « articles » : un ou plusieurs tests (essayer le produit disponible dans le commerce), une *preview* (décrire un jeu qui n'est pas encore disponible dans le commerce, mais dont on a une version *beta*), des *news* (généralement, avant la sortie, il s'agit de répercuter quelques informations sur le futur jeu), des trucs et astuces (ou *tips*), une solution complète des énigmes du jeu ;

- certains jeux sont disponibles sur différentes machines (PC, Dreamcast, Playstation...) ; certains tests, previews, solutions concernent une seule version du jeu, parfois plusieurs (et cela n'est pas systématique même pour un même jeu : la solution d'un jeu est souvent commune à plusieurs versions, alors que les test doivent être faits pour chaque version) ;

- chaque jeu appartient à une catégorie (jeu d'action/aventure, jeu de réflexion, jeu de plateforme, jeu de simulation sportive, jeu de course, jeu de baston...).

Avec SPIP, il n'est possible de créer qu'une seule structure hiérarchique des rubriques. De manière simple, il faudrait choisir une unique structure :

- une structure sur les types d'articles (une rubrique pour les tests, une rubrique pour les previews, une solution pour les solutions...) ;
- **ou** une structure selon les machines (une rubrique pour PC, une rubrique pour Playstation, une rubrique pour Dreamcast...) ;
- **ou** une structure selon la catégorie de jeu (une rubrique pour l'aventure, une pour la simulation sportive, une pour la baston...).

Dans tous les cas, on voit bien qu'on perd la richesse de navigation que doit offrir un tel site, qui devrait permettre de passer à un autre article concernant un même jeu, mais aussi de naviguer selon une machine, ou de consulter d'autres jeux de la même catégorie...

Une seconde difficulté tient dans les **informations systématiques** que l'on voudrait inclure pour chaque jeu, pour pouvoir les présenter comme des informations « à part » :

- la date de sortie du jeu ;
- une note d'appréciation.

On pourrait évidemment décider d'écrire « à la main » ces informations à l'intérieur des articles, mais alors on ne pourrait plus les présenter sur des pages spécifiques (« les prochaines sorties », « les jeux sortis récemment », « les meilleurs jeux dans cette catégorie... »).

Cette reproduction du sommaire que nous obtiendrons donne une idée de la diversité des informations présentées sur le site, et les possibilités de navigation :

- au centre, les tests ; pour chaque jeu testé, on indique la note attribuée, et la machine (Dreamcast, Playstation...) ;
- en haut, on peut afficher des sommaires consacrés à chaque machine ;
- en haut à gauche, on pourra voir les prochaines sorties de jeux, ainsi que la liste des jeux ayant reçu les meilleures notes ;
- à gauche apparaît la navigation dans le site selon les catégories de jeux (survival-horror, simulation sportive...) ;
- à droite, les previews, les soluces, les astuces, les news...

La structure du site

Si la souplesse de SPIP dans la gestion de la structure des rubriques autorise, généralement, à commencer à rédiger ses articles sans trop se soucier du rubriquage (il est toujours facile, ensuite, de déplacer des articles ou des rubriques), dans le cas qui nous intéresse, il faut au contraire commencer par définir la structure retenue. C'est, en effet, elle qui conditionne les possibilités d'un tel site.

Avec SPIP, il n'est possible de définir qu'une seule et unique structure hiérachique des rubriques. Il va donc falloir décider comment nous allons créer les rubriques de notre site.

Les possibilités

Comme nous l'avons noté dans l'article précédent, chaque article dépend ici de plusieurs critères. On peut donc choisir la structure des rubriques parmi les structures suivantes :

- une structure par machines : une rubrique pour Dreamcast, une rubrique pour Playstation, une rubrique pour PC...
- une structure par jeu : une rubrique pour *Resident Evil*, une rubrique pour *Gran Turismo*, une rubrique pour *Mortal Combat*...
- une structure par catégorie d'articles : une rubrique pour les tests, une rubrique pour les news, une rubrique pour les solutions...
- une structure par genre de jeu : une rubrique pour les jeux de sport, une pour les jeux de plateforme, une pour les jeux de baston...

Si l'on était un peu tordu, on pourrait également adopter les structures suivantes :

- une structure par qualité des jeux (les meilleurs jeux dans une rubrique, les jeux moyens dans une autre...) ;
- une structure par date de rédaction, adaptée notamment à la transposition d'un mensuel papier sur le Web (les articles du numéro de novembre 2001, une rubrique pour décembre 2001, une rubrique pour janvier 2002, etc.).

Les impératifs

Certaines caractéristiques de notre site permettent de choisir une structure plutôt qu'une autre.

- Il est naturel, lorsqu'on navigue sur un site, de pouvoir passer directement du test d'un jeu à sa solution, de la preview aux news... Le visiteur, intéressé par la preview, va vouloir connaître les dernières informations sur le développement du produit (les news). Un joueur, après avoir lu l'opinion exprimée dans un test, voudra trouver rapidement des indications pour terminer son jeu (les solutions, les astuces).

Le plus simple, ici, consiste donc à réunir tous les articles d'un même jeu dans une même rubrique. La rubrique d'un jeu contiendra à la fois la preview, les tests, la solution...

Ainsi, la plus petite rubrique du site sera la rubrique consacrée à tous les articles d'un seul jeu.

- On pourrait décider de créer des grandes rubriques par machine, et d'y placer, en tant que sous-rubriques, chacun des jeux avec leurs différents articles (un jeu existant sur plusieurs machines aurait une rubrique spécifique à chacun des machines). Cependant, il est fréquent que certains articles sur un même jeu soient communs à plusieurs machines. Si le test de la version Dreamcast de *Alone in the dark* est différent de celui de la version Playstation, en revanche la solution des énigmes est commune aux deux supports. Avec un rubriquage par machines, on aurait naturellement le test Dreamcast dans la rubrique Dreamcast, et le test Playstation dans la rubrique

ad hoc ; mais la solution, commune aux deux versions, devrait alors être publiée dans les deux rubriques ; très inélégant et fastidieux.

- Les genres de jeux présentent une structure hiérarchique : une course de rallye est une course de voitures, qui est elle-même une simulation sportive ; un « survival horror » est un jeu d'aventure/action ; un jeu de baston en 2D est un jeu de baston... Ce type de structure est donc particulièrement adapté à SPIP, car on peut imbriquer des sous-rubriques dans des rubriques autant que nécessaire. Une grande rubrique « Sport » contiendra plusieurs sous-rubriques (Course, Football, Glisse, Tennis...), chacune de ces sous-rubriques pouvant à son tour contenir d'autres sous-rubriques.

- À l'inverse, les autres structures ne présentent pas réellement de structure hiérarchique. Les machines ne « dépendent » pas les unes des autres : on pourrait évidemment réunir les consoles de jeu à l'intérieur d'une grande rubrique, et créer une autre grande rubrique pour les PC, mais cela n'offre pas grand intérêt, et n'offrirait pas une grande profondeur hiérachique. De la même façon, une catégorie d'article ne dépend pas d'une autre (les news ne dépendent pas des solutions qui ne dépendent pas des tests...) ; un rubriquage par catégorie d'article ne permettrait donc pas une réelle structure de rubriques/sous-rubriques.

La structure retenue

En réalité, n'importe laquelle des possibilités évoquées précédemment est possible, car les techniques que nous aborderons plus loin permettraient de s'adapter à ces différentes situations. Cependant, il est plus agréable de choisir une structure plus directement adaptée à la logique de SPIP ; dit autrement : un site qui serait déjà très présentable avec une telle structure, même sans utiliser les astuces que nous exposerons plus loin (c'est-à-dire un site que l'on peut déjà visiter avec les squelettes standards). Ainsi, l'utilisation de l'espace privé et la gestion du site resteront cohérents et simples.

Nous allons donc privilégier la structure la plus hiérachique : la structure par genres de jeux. Nous créerons de grandes rubriques thématiques (Action/aventure, Combats, Jeux pédagogiques, Plateforme, Sport, Stratégie...), chacune de ces grandes rubriques ayant plusieurs sous-rubriques.

Dans chacune des sous-rubriques, nous créerons enfin une sous-rubrique pour chaque jeu, regroupant tous les types d'articles du même jeu sur toutes les plateformes de ce jeu (par exemple, tous les articles concernant *Alone in the Dark 4*, que ces articles soient des news, des previews, des tests, des solutions... qu'ils concernent la version PC/Windows, Dreamcast ou Playstation... seront regroupés dans l'unique rubrique consacrée à ce jeu).

Logiquement, le test de la version Dreamcast de *Alone in the Dark* se trouvera dans la rubrique *Alone in the Dark*, elle-même installée dans une rubrique « Survival horror », elle-même dépendant de la rubrique « Action/Aventure ».

Nous verrons plus loin que, si cette structure s'utilise facilement avec SPIP (et qu'un tel site est déjà facile à visiter avec les squelettes standards), en revanche cela pose quelques difficultés de logique de présentation, puisque l'on mélange des rubriques dont le nom est un jeu avec des rubriques qui sont des grandes catégories de jeux.

Que deviennent les machines et les genres d'articles ?

Nous avons donc perdu, avec cette structure, la possibilité d'indiquer dans la navigation, pour chaque article, s'il agit d'un test, d'une solution, de news... ou encore s'il concerne une version PC, Dreamcast, Playstation...

Est-ce qu'il ne serait pas pratique, ici, d'avoir pour chaque article un menu déroulant proposant une liste des machines, et un menu déroulant avec la liste des types d'articles ? On sélectionnerait dans ces menus déroulants et, d'un clic, on pourrait indiquer qu'il s'agit d'un *test*, et que cela concerne la Dreamcast. On pourrait même indiquer que cela concerne la Dreamcast *et* la Playstation pour un même article.

(Vous n’imaginez pas le nombre de webmestres qui, à ce stade, se lancent à corps perdu dans le code source de SPIP pour ajouter les menus déroulants dont il a besoin !)

Ca tombe bien : ce menu déroulant existe déjà. C’est celui des mots-clés.

Les mots-clés, associés aux articles, permettent de créer des navigations *transversales* au sein de la structure hiérarchisée d’un site sous SPIP, c’est-à-dire de passer directement d’un article dans une rubrique à un article situé dans une autre rubrique (cela quelle que soit la position de l’autre rubrique dans la hiérarchie).

Certes, l’usage immédiat des mots-clés consiste à créer des mots-clés purement thématiques ; mais si l’on considère qu’il s’agit d’un moyen de navigation transversale dans le rubriquage, on peut tout autant les utiliser pour donner le nom de la machine utilisée ou le type d’article.

La situation s’y prête particulièrement bien ici : ce que nous devons indiquer ne présente aucune structure hiérarchique, et surtout le nombre d’éléments est relativement fixe. Une fois que l’on a déterminé la liste complète des machines, il n’est plus nécessaire d’en ajouter (l’apparition d’une nouvelle console n’est pas fréquente) ; et les types d’articles sur notre site seront déterminés à l’avance. On fixera donc la liste des mots-clés une bonne fois pour toute avant de commencer (les ajouts seront très rares), et nous aurons notre menu déroulant qui permettra, pour chaque article, d’indiquer la machine concernée et le type d’article.

N.B. Il existe d’autres cas où l’utilisation des mots-clés est un piège. Ce sont notamment les cas où la liste de mots n’est pas fixée à l’avance, et où il faudrait quasiment créer un nouveau mot-clé à chaque nouvel article. Certes, cela fonctionnerait avec SPIP, mais la mise à jour du site serait particulièrement pénible.

Par exemple, un site consacré au cinéma. Nous aurions une structure thématique, selon les grands genres du cinéma. Ensuite, nous voudrions pouvoir passer d’un film de tel réalisateur aux autres films du même réalisateur. Une solution (piège) consisterait donc à créer un mot-clé pour ce réalisateur : lorsqu’on ajoute la fiche d’un film, on lui associerait le mot-clé de son réalisateur. De même pour les acteurs principaux, l’auteur du scénario, etc. La navigation dans le site serait ainsi d’une très grande richesse. Mais la mise à jour avec SPIP serait infernale : une liste de mots-clés interminable, et surtout la nécessité de créer des mots-clés à chaque fois qu’on ajoute un film.

Imaginons encore que nous ayons retenu pour notre site une structure par grande catégorie d’articles : une rubrique pour les tests, une rubrique pour les previews, une rubrique pour les news... Une fois dans le test de *Alone in the dark*, on voudrait afficher des liens vers sa solution, ses news... Le seul moyen d’y arriver serait alors d’utiliser un mot-clé « Alone in the dark ». On voit qu’alors, à chaque nouveau jeu abordé sur le site, on devrait créer un nouveau mot-clé : très fastidieux à gérer, difficile à utiliser (rapidement le menu déroulant deviendrait interminable), interface inutilement surchargée (certains jeux n’étant plus du tout traités quelques mois après leur sortie, leur présence dans la liste des mots-clés affichée à chaque fois devient inutile).

Mise en place de la structure

Puisque nous avons déterminé la structure du site dans l'article précédent, rendons-nous dans l'espace privé de SPIP, et entrons les éléments nécessaires.

Tâche toujours ingrate, mais indispensable avant de rédiger le moindre article et même de créer les squelettes. En effet, les éléments que nous allons créer dès maintenant serviront de base à la suite.

La structure des rubriques

Commençons par créer le rubriquage thématique (selon les grandes catégories de jeux).

C'est une question de choix personnels, mais voici une idée du résultat (non exhaustif) :

Action/aventure
Exploration et énigmes
FPS (Quake-like)
RPG (Jeux de rôle)
Survival horror
Combats
2D
3D
Jeux pédagogiques
Lecture
Mathématiques
Physique
Culture
Plateforme
2D
3D
Sports
Course
Formule 1
Rallye
Grand Tourisme
Football
Glisse
Tennis
Stratégie
Casse-têtes
Jeux de cartes
Jeux de plateau
Wargames
Stratégie en temps réel

Cette structure est incomplète, mais c'est là son intérêt : on peut facilement la modifier en fonction des besoins, sans pour autant devoir repenser le principe général ni les squelettes. Quand une rubrique contient trop de jeux, on peut créer des sous-rubriques pour éviter d'obtenir des listes trops longues (par exemple, si l'on a trop de « jeux de cartes », on peut créer des sous-rubriques « Poker », « Bridge », « Belotte » dans cette rubrique et y transférer les jeux).

Nous avons prévu de créer ensuite des rubriques pour chaque jeu (une rubrique « Resident Evil », une rubrique « Alone in the dark »...). Mais comme ces rubriques ne concernent pas la structure « fixe » du

site, nous verrons cela dans l'article suivant.

Les machines

Comme nous l'avons expliqué précédemment, les machines seront indiquées par des mots-clés.

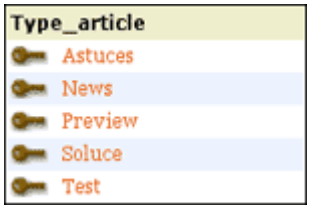
Il faut d'abord créer un « groupe de mots », que nous nommerons « Machines ».

Ensuite, nous allons créer un mot-clé pour chaque machine, chacun de ces mots-clés étant installé dans le groupe « Machines ». Ce qui donne :

Il est fréquent, sur ce genre de site, de signaler la machine concernée par un petit logo. Nous allons donc attribuer à chaque mot-clé un logo lui correspondant (considérons ce logo comme obligatoire ; en effet, dans les squelettes, nous afficherons fréquemment le logo sans indiquer le nom de la machine en clair). Pour assurer la cohérence graphique, ces logos doivent être approximativement de même taille (par exemple environ 25 pixels x 25 pixels) :

Les types d'articles

De la même façon, nous allons maintenant définir une liste des types d'articles. Cela donne :



Nous n'attribuons pas de logo à ces mots-clés (vous pouvez décider de la faire, vous pourrez ainsi créer des éléments de navigation graphiques).

Une nuance (que nous verrons de manière plus évidente par la suite)concernant les mots-clés : en ce qui concerne les types d'article, ces mots seront utilisés nominativement dans les squelettes ; il faut donc choisir des noms et s'y tenir (si, dans le cas précédent, où les mots concernent les machines, « Playstation » pourra être ultérieurement changé en « PS One » sans difficulté, ici on ne pourra plus changer « Soluce » en « Solution » sans devoir modifier les squelettes).

Notez que le groupe de mots se nomme « Type_article », et non « Type d'article » ; en effet, nous appelerons spécifiquement ce titre dans les squelettes, et les noms avec des espaces et des apostrophes sont moins faciles à manipuler.

Écrire des articles

Il est temps désormais de créer les premiers articles. Non seulement cela nous permettra dans ce tutorial d'expliquer la démarche, mais de toute façon cette étape est indispensable avant de pouvoir travailler sur les squelettes.

N.B. Si vous avez déjà manipulé les squelettes de SPIP, vous savez qu'on ne peut réellement travailler que lorsqu'au moins un article est publié. Nous vous conseillons même, dans le cadre de la création de l'interface graphique, de créer le maximum d'articles possibles, avec plusieurs articles par rubriques, plusieurs rubriques... et même des articles de longueurs différentes, avec ou sans surtitre et soustitre, avec ou sans chapeau, descriptif... Plus ces articles seront nombreux et de formats divers, plus vous pourrez créer une interface « souple » (c'est-à-dire adaptée à tous les cas de figure possible). Evidemment, pour les tests d'interface, ces articles peuvent très bien être constitué de « faux texte » (des copier-coller d'articles pris sur le Web, du charabia, du latin de cuisine...).

Une rubrique par jeu

Nous avons décidé que tous les articles concernant un jeu seraient regroupés dans une même rubrique. Avant même d'écrire le premier article au sujet d'un jeu, il faut donc créer une rubrique portant le nom de ce jeu (pour les articles suivants, évidemment on se placera dans la même rubrique).

La rubrique va donc prendre un rôle très important : c'est elle qui donne le nom du jeu, et c'est elle qui fournit le logo, pour tous les articles qu'elle contient. Par exemple, nous allons créer une rubrique « Resident Evil : Code Veronica » et nous lui attribuerons un logo. Ensuite, lorsque nous insérerons un article contenant la solution du jeu dans cette rubrique, cet article n'aura pas besoin de rappeler le nom du jeu (l'article se nommera simplement « Solution complète »), et il ne sera pas obligatoire de lui attribuer un logo.

Plaçons-nous dans la rubrique :

- Action/aventure | Survival horror

et créons une nouvelle sous-rubrique intitulé « Resident Evil : Code Veronica ».

Nous plaçons un « descriptif » pour cette rubrique, présentant la trame général du jeu.

Enfin nous utilisons une copie d'écran pour créer le logo de la rubrique.

N.B. Ces logos de rubrique seront par la suite très utilisés pour obtenir l'interface graphique du site. Nous vous conseillons donc de choisir arbitrairement (en fonction de vos exigences graphiques) un format unique pour tous les logos. Cela facilitera la création d'une interface cohérente. Nous choisissons par exemple que tous les logos de rubriques font 120 x 160 pixels.

Cela donne :

Le test complet du jeu !

Voilà, la rubrique créée, nous pouvons écrire notre premier article concernant ce jeu.

N.B. Soulignons ici un premier inconvénient sur ce site : il faut s'astreindre (et l'expliquer aux autres participants du site) à créer une rubrique avant de pouvoir écrire le premier article concernant un jeu. Or le comportement naturel serait d'écrire l'article directement dans la rubrique « Survival horror », et de lui donner pour titre (ou pour surtitre) le nom du jeu. Cela conviendrait pour un site plus simple (traitant d'une seule machine, ne proposant que des tests ou des soluces...). Mais puisque nous sommes partis sur un site plus ambitieux, cela induit plus de formalisme dans la structuration.

Dans un premier temps, il suffit de créer du texte, d'intégrer des images... rien d'inhabituel.

Attention cependant, il y a deux points à déterminer immédiatement, et s'y tenir systématiquement par la suite :

- le nom du jeu est-il indiqué dans les « cases » de l'article lui-même ? (en titre, surtitre, soustitre...)
- le « descriptif » est-il obligatoire ?

Cela détermine la façon dont nous traiterons les squelettes par la suite. Vous pouvez choisir de façon arbitraire, il suffira de réaliser vos squelettes en conséquence.

Pour les exemples qui seront fournis dans ce tutorial, nous avons décidé que l'article lui-même n'indique pas le nom du jeu (ni dans le titre, ni dans les sur/soustitres) ; ce nom sera récupéré systématiquement depuis la rubrique qui contient l'article. Et dans nos squelettes, nous utiliserons le #DESCRIPTIF ; donc il est conseillé d'en mettre un, sinon l'espace attribué au descriptif sera vide.

Enfin, il n'est pas obligatoire d'attribuer un logo à l'article lui-même, puisque nous récupérerons celui de la rubrique. Mais si on décide de placer un logo à l'article, nous conseillons de respecter le format adopté pour les logos des rubriques (dans notre cas 120 x 160 pixels).

En gros, cela donne :

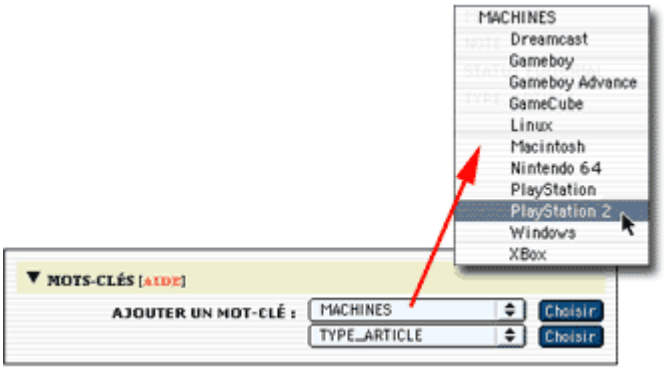
- Le titre de l'article n'est pas le nom du jeu, mais une accroche (par exemple : « Le retour de Claire Redfield ») ;
- il y a un descriptif ;
- nous n'utilisons que la date de publication en ligne (pas de « date de première publication » - nous avons même désactivé cette possibilité dans la « Configuration précise ») ; cette date correspond bien à la date de publication de l'article, donc de mise à jour du site Web (pour la gestion des dates de sortie commerciale du jeu, nous verrons cela par la suite) ;

- dans les auteurs, nous travaillons également de manière habituelle : l'auteur est le rédacteur qui, sur le site, a écrit l'article (les test, les news...). Ne pas essayer ici d'indiquer un « auteur du jeu » ou autre subtilité.

Pour quelle(s) machine(s) ? Quel type d'article ?

Il reste maintenant à indiquer quelle(s) machine(s) cet article concerne, et de quel type d'article il s'agit.

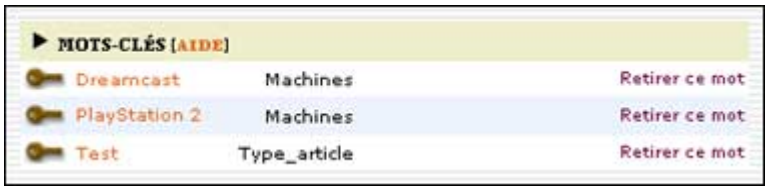
Pour cela, nous allons utiliser les menus déroulants « Ajouter un mot-clé ». Voilà qui est bien pratique...



Pour ce jeu, nous considérerons que le test vaut aussi bien pour la version Dreamcast que pour la version Playstation 2. Si on considère que les deux versions sont suffisamment différentes, on peut tout aussi bien écrire un test différent pour chaque machine, il suffira alors de ne sélectionner qu'une seul machine par article.

Quant au type d'article, il s'agit ici d'un test.

Ce qui nous donne :



Une nouvelle fois, il faut souligner le besoin de formaliser strictement la manière de publier un article. Avant toute publication, il faut donc vérifier la présence des mots-clés nécessaires.

- L'indication de la machine n'est pas réllement obligatoire (l'article apparaîtra tout de même au sommaire) ; mais son absence rendra l'interface incohérente (certains articles indiquant des machines, d'autres non) et l'article ne sera plus référencé dans la navigation « par machine ». Il est possible d'indiquer plusieurs machines pour un même article.

- Le type d'article est indispensable. Nous verrons par la suite que nous faisons un usage intensif de ces types d'articles. En cas d'oubli, l'article risque purement et simplement de ne pas apparaître sur le site public. Il ne faut indiquer qu'un seul type d'article pour un article.

Cela fait, créez d'autres articles, à des formats différents, certains avec plusieurs machines, selon différents types d'articles... En particulier, essayez de créer une rubrique complète (avec tous les types d'articles), et enrichissez les rubriques proches de « Survival horror », de façon à pouvoir travailler sur des squelettes avec des rubriques représentatives de l'activité du site. Pensez de plus à modifier les dates de publication, afin de pouvoir tester les classements des articles et des rubriques.

L'idéal étant, dans cette situation, de créer un maximum d'articles dans tout le site, avec des dates de publication couvrant une période assez large.

Première version du squelette des articles

Nous avons désormais un site très structuré depuis l'espace privé, et de nombreux articles publiés. Il est temps de nous attaquer à la création du premier squelette.

Beaucoup de webmestres commencent cette étape en travaillant à partir d'un squelette standard, qu'ils modifient par petites touches.

De notre côté, non seulement pour les besoins de ce tutorial, mais aussi par habitude, nous construisons les squelettes à partir d'un document totalement vide. Étape par étape, les différentes boucles sont ajoutées, puis finalement l'interface graphique (au départ du code HTML extrêmement simple, la construction graphique avec des tableaux étant ajoutée en dernière étape). Cette méthode présente plusieurs avantages :

- cela permet de bien comprendre sa propre construction intellectuelle lors du développement d'une page : quelle boucle dépend de quelle autre, etc. ;
- initialement débarassé des éléments d'interface graphique, il est plus simple de bien percevoir la hiérachie des informations sur la page ; ainsi l'interface graphique découle-t-elle de cette hiérachie d'une manière plus cohérente ;
- le code HTML généré est souvent plus clair : les redondances sont moins nombreuses (défaut fréquent du copier-coller de code HTML).

Dans ce qui suit, nous utiliserons donc du code HTML très simple (notamment : pas de tableaux). La création d'une interface graphique plus élaborée, en fin de processus, est laissée au soin du lecteur.

Volontairement, nous construirons le code étape par étape. Pour éviter des pages interminables, lorsque nous enrichirons une boucle existante, nous ne reproduirons pas l'intégralité de la page, mais uniquement la partie qui nous intéresse. Pour s'y retrouver, on se référera souvent au nom des boucles déjà existantes.

La page de base

Commençons donc avec un fichier « article.html » vierge.

Extrêmement simple, voici la structure minimale d'un article, sans aucun élément de navigation :

```
<html>
<title>[#NOM_SITE_SPIP]
<BOUCLE_titre(ARTICLES){id_article}>#TITRE</BOUCLE_titre></title>
</head>

<body>
<blockquote>
     <BOUCLE_principale(ARTICLES){id_article}>

     [<h3>(#SURTITRE)</h3]
     <h2>#TITRE</h2>
     [<h3>(#SOUSTITRE)</h3>]

     [(#DATE|affdate)]

     [(#LOGO_ARTICLE_RUBRIQUE|right)]
     [<b>(#CHAPO|justifier)</b>]

     [(#TEXTE|justifier)]

     [<p align="right" align='justify'>(#LESAUTEURS)]

     [<hr>(#PS)]
     [<hr>(#NOTES)]

     </BOUCLE_principale>
</blockquote>
</body>
</html>
```

Facile :

- la BOUCLE_titre permet d'afficher le titre de l'article dans l'entête de la page ;
- la BOUCLE_principale contient d'intégralité de la page ; elle permet de positionner toutes les autres boucles à l'intérieur de l'article sélectionné.

Récupérer les infos sur le jeu

Le squelette précédent serait une bonne base pour un webzine standard (tout l'information nécessaire est dans l'article), mais dans notre cas, une information vitale ne se trouve pas dans l'article : de quel jeu s'agit-il ?

En effet, nous avons décidé que le titre du jeu n'était pas dans l'article, mais dans la rubrique contenant cet article. On pourrait se contenter de la boucle HIERARCHIE, mais nous préférerons ici prendre une habitude qui reviendra systématiquement dans nos exemples : **il faut passer de l'article à sa rubrique pour récupérer le nom du jeu**. (C'est le même principe pour récupérer le logo ; à la différence près que la balise #LOGO_RUBRIQUE_ARTICLE effectue automatiquement cette opération.)

Ce sera donc une constante sur ce site : les boucles ARTICLES contiennent souvent une boucle RUBRIQUES permettant de « remonter d'un cran », c'est-à-dire de récupérer le titre du jeu. Cela sera le cas dans les pages des rubriques, ainsi que sur la page de sommaire.

Reprenons le début du code, tout au début de la BOUCLE_principale :

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

     <BOUCLE_rubrique(RUBRIQUES){id_rubrique}>
     <h1>#TITRE</h1>
     </BOUCLE_rubrique>
...
```

Notez que le #TITRE de la rubrique est affiché avec la taille maximale (h1) ; logique, puisque c'est le titre du jeu. Le titre de la rubrique, qui est d'ordinaire seulement un élément de navigation dans le site, est ici l'information principale de l'article.

Afin d'obtenir une esquisse de navigation, affichons la hiérarchie. Nous allons démarrer cette hiérarchie depuis la rubrique qui contient la sous-rubrique actuelle, puisque ce qui nous intéresse est la liste des catégories de jeux (et nous connaissons déjà le nom de la dernière sous-rubrique - le titre du jeu). Nous nous plaçons donc à l'intérieur de la BOUCLE_rubrique, ajoutons une BOUCLE_rub_parent (qui permet encore de « remonter d'un cran » dans la hiérachie), et à partir de là nous demandons la hiérachie :

```
<BOUCLE_rubrique(RUBRIQUES){id_rubrique}>
     <BOUCLE_rub_parent(RUBRIQUES){id_enfant}>
          <BOUCLE_hierarchie(HIERARCHIE){id_rubrique}{" : "}>
          <a href="#URL_RUBRIQUE">#TITRE</a>
          </BOUCLE_hierarchie>
     </BOUCLE_rub_parent>
     <h1>#TITRE</h1>
     </BOUCLE_rubrique>
```

Laissons cette page « article.html » en l'état. Rien de compliqué pour l'instant. Il manque la gestion des mots-clés, et les liens entre les articles d'un même rubrique (c'est-à-dire traitant du même jeu).

Seule subtilité : il faut remonter à la rubrique qui contient l'article pour obtenir le titre du jeu, et il faut encore « remonter d'un cran » (la rubrique qui contient la rubrique qui contient le jeu) avant de démarrer la hiérarchie.

La page des rubriques

Commençons avec un fichier « rubrique.html » vierge.

La première version de notre fichier sera très simple. Nous intégrons tout de suite la BOUCLE_hierarchie, qui ne présente ici aucune particularité.

```
<html>
<title>[#NOM_SITE_SPIP]
<BOUCLE_titre(RUBRIQUES){id_rubrique}>#TITRE</BOUCLE_titre></title>
</head>

<body>
<blockquote>
     <BOUCLE_principale(RUBRIQUES){id_rubrique}>

  <BOUCLE_hierarchie(HIERARCHIE){" : "}>
  <a href="#URL_RUBRIQUE">#TITRE</a>
  </BOUCLE_hierarchie>

     <h1>#TITRE</h1>

     [(#LOGO_RUBRIQUE|right)]

     [(#TEXTE|justifier)]

     [<hr>(#NOTES)]

     </BOUCLE_principale>
</blockquote>
</body>
</html>
```

Si vous installez cette page sur votre site, ça ne va pas être spectaculaire...

Les articles contenus dans cette rubrique

Ajoutons la liste des articles contenus dans cette rubrique. Plaçons cette BOUCLE_articles à la fin de la BOUCLE_principale (après le texte et ses notes).

```
...
 <B_articles>Les articles :
 <ul>
 <BOUCLE_articles(ARTICLES){id_rubrique}>
  <li> <a href="#URL_ARTICLE">#TITRE</a>
 </BOUCLE_articles>
 </ul>
 </B_articles>

</BOUCLE_principale>
...
```

En réalité, cette boucle ne présente pas grand intérêt : elle affiche les articles sans classement particulier (elle ne tient pas compte des mots-clés de type d'articles). Nous l'installons pour l'instant telle quelle, afin d'avoir une première navigation possible sur le site. Nous reprendrons cette partie un peu plus loin

dans ce tutorial.

Les sous-rubriques

Installons maintenant la boucle des sous-rubriques (après la BOUCLE_articles) :

```
...
 <B_sous_rubriques>Les sous-rubriques :
 <ul>
 <BOUCLE_sous_rubriques(RUBRIQUES){id_parent}{par titre}>
  <li> <a href="">#TITRE</a>
 </BOUCLE_sous_rubriques>
 </ul>
 <B_sous_rubriques>


</BOUCLE_principale>
...
```

Et là, on rencontre le vrai premier problème de notre tutorial : les rubriques définissant des genres de jeux sont mélangées avec des rubriques consacrées directement à des jeux.

Par exemple, dans la structure de démonstration, la rubrique « Action/aventure » contient à la fois, présentés exactement de la même manière :

- deux sous-rubriques thématiques (« Exploration et énigmes » et « Survival horror »),
- et deux rubriques de jeux (« Devil May Cry » et « Soul Reaver 2 »).

L'affichage de la boucle ci-dessus fournit donc la liste :

- Devil May Cry,
- Exploration et énigmes,
- Soul Reaver 2,
- Survival horror.

Or il faut impérativement différencier très nettement les rubriques « catégories de jeux » (utiles essentiellement à la navigation) et les rubriques « jeux » (qui donnent le titre des jeux).

Pour résoudre cette difficulté, nous allons utiliser le système des boucles. Aucune bidouille informatique donc, mais il va faut bien comprendre la logique du site et la logique des boucles.

La logique du site : une rubrique « Jeu » est une rubrique qui contient des articles ; une rubrique de « catégorie de jeux » ne contient pas directement d'articles. C'est la définition de notre site : tout article concernant un jeu est placé dans une rubrique portant le nom du jeu ; ainsi une rubrique de catégorie de jeu ne contient que des sous-rubriques de catégories et des rubriques portant le nom des jeux, mais jamais directement des articles (est-ce bien clair ?). Par exemple, dans notre cas :

- « Soul Reaver 2 » et « Devil May Cry » contiennent les articles concernant ces jeux (tests, previews, etc.) ;
- « Survival horror » et « Exploration et énigmes » ne contiennent par définition pas d'articles. S'il y a des jeux dans « Survival horror », les articles les concernant sont dans des sous-rubriques portant le nom de ces jeux, mais pas directement dans la rubrique « Survival... ».

Le but de la manoeuvre va donc consister à séparer les rubriques *sans* articles des rubriques *avec* des articles. Et le plus simple est de faire deux boucles différentes.

Les rubriques sans articles

Commençons par les sous-rubriques qui ne contiennent pas d'articles (c'est-à-dire les grandes catégories de type « Survival horror », « Exploration »...). Modifions la BOUCLE_sous_rubriques ainsi :

```
<BOUCLE_sous_rubriques(RUBRIQUES){id_parent}{par titre}>

  <BOUCLE_sans_articles(ARTICLES){id_rubrique}>
  </BOUCLE_sans_articles>
  <h3><a href="#URL_RUBRIQUE">#TITRE</a></h3>
  <//B_sans_articles>

 </BOUCLE_sous_rubriques>
```

Le principe de cette astuce est simple :

- à l'intérieur de la BOUCLE_sous_rubriques, on insère une nouvelle BOUCLE_sans_articles, qui va récupérer les articles qu'elle contient. Notez que l'intérieur de la BOUCLE_sans_articles est vide : si la sous-rubriques contient des articles, ces derniers ne sont pas affichés (la boucle ne sert qu'à contrôler leur présence) ;

- le #TITRE de la BOUCLE_sous_rubriques est inséré dans le *texte optionnel « sans »* de la BOUCLE_sans_articles. Juste avant le <//B_sans_articles>. Ainsi, le titre de la rubrique ne s'affiche que si BOUCLE_sans_articles ne contient aucun article (c'est-à-dire si la BOUCLE_sous_rubriques elle-même ne contient aucun article).

La BOUCLE_sous_rubriques, dans l'absolu, récupère bien toutes les rubriques (y compris celle des jeux). Mais le #TITRE (avec le lien hypertexte) n'est affiché que si la BOUCLE_sans_articles est vide.

Les rubriques de jeu (avec des articles)

Nous appliquons le principe exactement inverse à une BOUCLE_jeux (que nous pouvons placer juste avant BOUCLE_sous_rubriques).

```
<BOUCLE_jeux(RUBRIQUES){id_parent}{par titre}>

  <B_avec_articles>
  <h4><a href="#URL_RUBRIQUE">#TITRE</a></h4>
  [(#LOGO_RUBRIQUE|right)]
  <ul>
  <BOUCLE_avec_articles(ARTICLES){id_rubrique}>
  <li><a href="#URL_ARTICLE">#TITRE</a>
  </BOUCLE_avec_articles>
  </ul>
  </B_avec_articles>

 </BOUCLE_jeux>
```

Cette fois, le #TITRE de la rubrique est placé dans le *texte conditionnel « avant »* ; on affiche le #LOGO_RUBRIQUE (évidemment, à terme, on fera de beaux tableaux pour éviter les chevauchements disgracieux). Et au passage, on en profite pour afficher la liste des articles de la BOUCLE_avec_articles.

La BOUCLE_jeux et la BOUCLE_sous_rubriques sont au départ totalement identiques : leurs critères sont les mêmes. À l'intérieur de ces boucles, ce sont donc les mêmes rubriques qui sont traitées. En revanche, les boucles BOUCLE_avec_articles et BOUCLE_sans_articles servent de « filtres » : ils prennent chacune des rubriques de BOUCLE_jeux et BOUCLE_sous_rubriques, et vérifient la présence ou non d'articles.

Est-ce que l'on aurait pu ici utiliser le critère {doublons} ? Non, car la première boucle affichée (BOUCLE_sous_rubriques) traite bien *toutes* les sous-rubriques (seulement, le titre de ces sous-rubriques

est affiché ou non selon les cas). Lui attribuer un critère {doublons} interdirait par la suite de traiter aucune de ces sous-rubriques. Ensuite, la BOUCLE_jeux associée à ce critère de {doublons} tenterait de récupérer toutes les sous-rubriques qui n'ont pas été traitées par la première boucle ; or toutes les sous-rubriques ont bien été traitées. La BOUCLE_jeux ne pourrait donc rigoureusement rien afficher.

Laissons maintenant cette page « rubrique.html », nous y reviendrons plus tard.

Les mots-clés dans les articles

Revenons à nos articles... Nous allons maintenant exploiter les mots-clés pour déterminer la ou les machines concernés, puis gérer « intelligemment » les liens vers les autres articles concernant le même jeu.

Les machines

Indiquer quelles machines sont concernées est très simple : il s'agit d'une simple application de la boucle (MOTS). Nous insérons la boucle suivante dans l'article (par exemple entre le #SOUSTITRE et la #DATE) :

```
<BOUCLE_machines(MOTS){id_article}{type=Machines}>
 [(#LOGO_MOT|left)]
 </BOUCLE_machines>
```

Nous n'affichons que le logo de la machine. (Évidemment, une fois tout le processus de création des boucles terminé, il conviendra de fignoler l'interface graphique...)

Dans la même rubrique, au sujet du même jeu

Afficher les autres articles contenus dans la même rubrique n'est guère plus compliqué.

Commençons par modifier l'appel de la BOUCLE_principale avec un {doublons}, pour éviter ensuite de créer un lien vers l'article déjà affiché (ça n'est pas bien grave, mais ça ne fait pas très propre) :

```
<BOUCLE_principale(ARTICLES){id_article}{doublons}>
```

Nous pourrions désormais nous contenter de récupérer tous les articles de la même rubrique (très simplement, une boucle (ARTICLES){id_rubrique}{doublons}). Cependant, nous voulons ici différencier clairement les types d'articles, non seulement pour rendre l'interface plus claire, mais aussi pour pouvoir réaliser des liens et des présentations différents selon les cas.

Commençons par une boucle vers les autres tests du même jeu. Nous insérons cette boucle toujours à l'intérieur de la BOUCLE_principale ; par exemple juste après #NOTES.

```
<B_tests><p>Autres tests de ce jeu :
 <ul>
 <BOUCLE_tests(ARTICLES){id_rubrique}{titre_mot=Test}{doublons}>
  <li><a href="#URL_ARTICLE">#TITRE</a>
 </BOUCLE_tests>
 </ul>
 </B_tests>
```

(Pour vérifier le fonctionnement de cette boucle, placez plusieurs articles de type « Test » dans la même rubrique. Vous pouvez même leur attribuer des machines différentes - un test de la version PC, un test de la version Dreamcast, un test de la version Playstation...)

Comme vous le constatez, c'est très simple :

- on appelle les (ARTICLES) selon le critère {id_rubrique} (c'est-à-dire les articles appartenant à la même rubrique) ;
- on ajoute la restriction suivante : le titre d'un des mots-clés liés à ces articles doit être « Test » (notez ici l'importance de respecter le choix initial du nom du mot-clé, et de ne plus le

modifier par la suite) ;
- le critère {doublons} interdit d'afficher à nouveau l'article principal.

Les *textes optionnels avant et après* (les <ul>...) ne servent que pour la présentation graphique.

N.B. Cette boucle utilise une fonctionnalité introduite dans le version SPIP 1.3 : dans une boucle (ARTICLES), vous pouvez sélectionner les articles selon le titre d'un mot-clé (avec le critère titre_mot), ou selon un groupe de mot-clé (critère type_mot).

Cependant cette boucle n'est pas suffisante : puisqu'il s'agit de tests du même jeu, mais sur d'autres machines, il faut indiquer de quelle machine il s'agit. Il suffit d'ajouter une boucle de mots (comme ci-dessus) à l'intérieur de cette BOUCLE_tests et d'afficher les logos correspondants. Ajoutons au passage la date de publication de l'article.

Le code devient :

```
<B_tests><p>Les tests de ce jeu :
 <ul>
 <BOUCLE_tests(ARTICLES){id_rubrique}{titre_mot=Test}>
  <li>
  <BOUCLE_tests_machines(MOTS){id_article}{type=Machines}>
   [(#LOGO_MOT)]
  </BOUCLE_tests_machines>
  <a href="#URL_ARTICLE">#TITRE</a> - [(#DATE|affdate)]
 </BOUCLE_tests>
 </ul>
 </B_tests>
```

Même chose pour les previews, les soluces et les astuces :

```
<B_previews><p>Les previews de ce jeu :
 <ul>
 <BOUCLE_previews(ARTICLES){id_rubrique}{titre_mot=Preview}>
  <li>
  <BOUCLE_previews_machines(MOTS){id_article}{type=Machines}>
   [(#LOGO_MOT)]
  </BOUCLE_previews_machines>
  <a href="#URL_ARTICLE">#TITRE</a> - [(#DATE|affdate)]
 </BOUCLE_previews>
 </ul>
 </B_previews>

 <B_soluces><p>Les soluces de ce jeu :
 <ul>
 <BOUCLE_soluces(ARTICLES){id_rubrique}{titre_mot=Soluce}>
  <li>
  <BOUCLE_soluces_machines(MOTS){id_article}{type=Machines}>
   [(#LOGO_MOT)]
  </BOUCLE_soluces_machines>
  <a href="#URL_ARTICLE">#TITRE</a> - [(#DATE|affdate)]
 </BOUCLE_soluces>
 </ul>
 </B_soluces>

 <B_astuces><p>Les trucs et astuces de ce jeu :
 <ul>
 <BOUCLE_astuces(ARTICLES){id_rubrique}{titre_mot=Astuces}>
  <li>
  <BOUCLE_astuces_machines(MOTS){id_article}{type=Machines}>
   [(#LOGO_MOT)]
  </BOUCLE_astuces_machines>
  <a href="#URL_ARTICLE">#TITRE</a> - [(#DATE|affdate)]
 </BOUCLE_astuces>
 </ul>
 </B_astuces>
```

(Ces trois boucles fonctionnent exactement sur le même principe que la BOUCLE_tests.)

Restent les news... Nous allons ici adopter un comportement très différent. Les news étant généralement des informations « courtes » (annonce d'un délais, rumeurs...), on aura avantage à les présenter toutes réunies sur une même page (peut-être pas, mais pour les besoins de ce tutorial, on dira que c'est mieux !).

Le petit morceau de code fera l'affaire :

```
<BOUCLE_news(ARTICLES){id_rubrique}{titre_mot=News}{doublons}{par
date}{inverse}{0,1}> <p><a
href="news_jeu.php3?id_rubrique=#ID_RUBRIQUE">Les news de ce
jeu</a> - dernière mise-à-jour : [(#DATE|affdate)]
</BOUCLE_news>
```

L'astuce qui permet d'obtenir un code aussi court :

- tout d'abord, on ne s'intéresse plus à la machine concernée ; cela supprime une boucle ;
- la BOUCLE_news récupère le dernier article ({par date}{inverse}{0,1}) associé au mot-clé « News ». Un seul article, cela suffit largement pour savoir s'il y a une page consacrée aux news ; et en récupérant la plus récente news, afficher sa date revient à annoncer la « dernière mise-à-jour » de la page des news...

De manière plus élégante, on aurait pu réaliser une BOUCLE_news vide et insérer le lien hypertexte dans le *texte optionnel* de la boucle.

Important : le lien hypertexte pointe vers une page « news_jeu.php3 », avec l'id_rubrique de la rubrique

actuelle. Nous allons devoir, par la suite, créer le squelette correspondant.

Les jeux dans la même catégorie

Nous voulons, pour finir, afficher les jeux de la même catégorie (par exemple, les autres jeux du genre « Survival horror »).

Nous allons d'abord utiliser la méthode qui consiste à « remonter d'un cran » (pour passer de la rubrique de l'article à la rubrique de la catégorie de ce jeu). Puis nous allons récupérer la méthode utilisée dans les rubriques pour afficher séparément les rubriques qui contiennent des articles (c'est-à-dire des rubriques directement consacrées à un jeu).

Le code est :

```
<BOUCLE_rub_act(RUBRIQUES){id_rubrique}{doublons}>
  <BOUCLE_jeux(RUBRIQUES){meme_parent}{doublons}{par date}{inverse}>
   <B_avec_articles>
   <h4><a href="#URL_RUBRIQUE">#TITRE</a></h4>
   <BOUCLE_avec_articles(ARTICLES){id_rubrique}{age < 90}>
   </BOUCLE_avec_articles>
  </BOUCLE_jeux>
 </BOUCLE_rub_act>
```

Les subtilités de ce code :

- la BOUCLE_rub_act (qui renvoit la rubrique actuelle) contient un {doublons} pour éviter ensuite d'afficher le jeu principal ;
- logiquement, la BOUCLE_jeux contient elle aussi un {doublons} ; attention : multiplier les critères doublons peut conduire à des difficultés. Si vous dupliquez ces boucles dans la même page pour un autre usage, prenez soin à éviter ce critère ;
- la BOUCLE_avec_articles ne récupère que les articles agés de moins de 3 mois (90 jours). En effet, nous ne voulons pas ici afficher l'intégralité des jeux de la même catégorie, mais uniquement ceux qui ont été modifiés récemment.

Les mots-clés dans les rubriques

Revenons au fichier « rubrique.html ». Nous l'avions laissé avec une méchante BOUCLE_articles, sans intérêt pour notre site, puisqu'elle ne classait pas les articles par catégories.

Le monde est bien fait : nous venons de programmer une telle fonctionnalité pour les articles !

Il suffit donc de copier-coller le bloc qui nous intéresse, de l'insérer à la place de la BOUCLE_articles (qui disparaît donc), et de voir s'il y a des modifications nécessaires. Ce qui nous donne :

```
<BOUCLE_les_articles(RUBRIQUES){id_rubrique}>
<B_tests><p>Les tests de ce jeu : <ul>
<BOUCLE_tests(ARTICLES){id_rubrique}{titre_mot=Test}> <li>
<BOUCLE_tests_machines(MOTS){id_article}{type=Machines}>
[(#LOGO_MOT)] </BOUCLE_tests_machines> <a
href="#URL_ARTICLE">#TITRE</a> - [(#DATE|affdate)]
</BOUCLE_tests> </ul> </B_tests>
<B_previews><p>Les previews de ce jeu : <ul>
<BOUCLE_previews(ARTICLES){id_rubrique}{titre_mot=Preview}>
<li>
<BOUCLE_previews_machines(MOTS){id_article}{type=Machines}>
[(#LOGO_MOT)] </BOUCLE_previews_machines> <a
href="#URL_ARTICLE">#TITRE</a> - [(#DATE|affdate)]
</BOUCLE_previews> </ul> </B_previews>
<B_soluces><p>Les soluces de ce jeu : <ul>
<BOUCLE_soluces(ARTICLES){id_rubrique}{titre_mot=Soluce}>
<li>
<BOUCLE_soluces_machines(MOTS){id_article}{type=Machines}>
[(#LOGO_MOT)] </BOUCLE_soluces_machines> <a
href="#URL_ARTICLE">#TITRE</a> - [(#DATE|affdate)]
</BOUCLE_soluces> </ul> </B_soluces>
<B_astuces><p>Les trucs et astuces de ce jeu : <ul>
<BOUCLE_astuces(ARTICLES){id_rubrique}{titre_mot=Astuces}>
<li>
<BOUCLE_astuces_machines(MOTS){id_article}{type=Machines}>
[(#LOGO_MOT)] </BOUCLE_astuces_machines> <a
href="#URL_ARTICLE">#TITRE</a> - [(#DATE|affdate)]
</BOUCLE_astuces> </ul> </B_astuces>
<BOUCLE_news(ARTICLES){id_rubrique}{titre_mot=News}{par
date}{inverse}{0,1}> <p><a
href="news_jeu.php3?id_rubrique=#ID_RUBRIQUE">Les news de ce
jeu</a> - dernière mise-à-jour : [(#DATE|affdate)]
</BOUCLE_news> </BOUCLE_les_articles>
```

Les modifications apportées :

- inutile de changer la requête {id_rubrique}, puisque c'était déjà sur cette rubrique que l'on se basait dans les articles ;
- supprimons tous les {doublons}, ils ne sont d'aucune utilité ici (il n'y a pas d'article déjà affiché) ; autant ne pas prendre le risque de conflits avec de futurs développements de notre page s'ils ne servent à rien ;
- l'ensemble est installé dans une grande BOUCLE_les_articles. Que fait cette boucle ? Strictement rien : elle renvoit la rubrique dans laquelle nous nous trouvons déjà. Mais elle nous sera utile ci-après...

L'autre point faible de notre page était la BOUCLE_jeux, dans laquelle la BOUCLE_avec_articles

profitait de son passage pour afficher la liste des articles de la sous-rubrique sans effectuer de tri. Or, trier les articles d'une rubrique, c'est ce que nous venons de faire ci-dessus.

Modifions donc la BOUCLE_jeux ainsi :

```
<BOUCLE_jeux(RUBRIQUES){id_parent}{par titre}>

  <B_avec_articles>
  <h4><a href="#URL_RUBRIQUE">#TITRE</a></h4>
  [(#LOGO_RUBRIQUE|right)]
  <ul>
  <BOUCLE_avec_articles(ARTICLES){id_rubrique}{0,1}>
   <BOUCLE_repeter_articles(boucle_les_articles)></BOUCLE_repeter_articles>
  </BOUCLE_avec_articles>
  </ul>
  </B_avec_articles>

 </BOUCLE_jeux>
```

La BOUCLE_avec_articles est désormais effectuée une seule fois (sur un seul article), ce qui est suffisant pour savoir si elle contient un article. On voit apparaître une BOUCLE_repeter_articles : il s'agit d'un boucle qui reproduit exactement le comportement de la BOUCLE_les_articles, à partir de l'endroit où elle se trouve (c'est-à-dire qu'elle démarre non plus depuis la rubrique principale, mais de la sous-rubrique). L'intérêt désormais de limiter la BOUCLE_avec_articles à un unique élément est alors clair : sans cette limitation, la BOUCLE_repeter_articles serait exécutée autant de fois qu'il y a d'articles dans la sous-rubrique.

Évidemment, le résultat graphique de cet exemple est hideux : trop chargé, peu clair. Disons qu'il était intéressant d'utiliser une « boucle récursive » dans ce tutorial...

L’interface des news

Toutes les news d’un jeu

Nous avons inséré, dans les squelettes des articles et des rubriques, un lien vers une page « news_jeu.php3 », qui doit présenter d’un coup toutes les news concernant un jeu.

Créons donc ce squelette...

Tout d’abord (principe du couple de fichiers pour gérer les squelettes de SPIP), nous allons créer le fichier d’appel « news_jeu.php3 ». Il suffit par exemple de copier le fichier « article.php3 » (fourni avec SPIP) et d’en modifier quelques éléments. Voici le contenu du fichier « news_jeu.php3 » :

```
<?
$fond = "news_jeu";
$delais = 24 * 3600;

include ("inc-public.php3");
?>
```

Le fichier de squelette utilisé sera donc « news_jeu » (« .html », ou une des variantes localisées à une rubrique si nécessaire, telle « news_jeu-52.html »...).

Le squelette sera donc un fichier « news_jeu.html ». Ce fichier est utilisé avec une variable de *rubrique*, puisque c’est bien la rubrique qui contient le nom du jeu et l’intégralité des articles concernant le jeu. Vite fait, bien fait, on va recopier le contenu du fichier « rubrique.html » et supprimer de dont nous n’avons plus besoin :

```
<html>
<title>[#NOM_SITE_SPIP]
<BOUCLE_titre(RUBRIQUES){id_rubrique}>#TITRE</BOUCLE_titre></title>
</head>

<body>
<blockquote>
<BOUCLE_principale(RUBRIQUES){id_rubrique}>

 <BOUCLE_hierarchie(HIERARCHIE){" : "}>
 <a href="#URL_RUBRIQUE">#TITRE</a>
 </BOUCLE_hierarchie>

     <h1><a href="#URL_RUBRIQUE">#TITRE</a></h1>

     [(#LOGO_RUBRIQUE|right)]

</BOUCLE_principale>
</blockquote>
</body>
</html>
```

Cela affiche le titre de la rubrique (c’est-à-dire le nom du jeu), le logo de la rubrique, et la hiérarchie. Seule modification : le titre du jeu est un lien hypertexte pour revenir à la page générale (lors de la finition graphique du site, on pourra évidemment préférer un lien plus discret).

Tout ce qu’il nous reste à faire : insérer une unique boucle qui affichera tous les articles de cette rubrique liés au mot-clé « News ». À l’intérieur de la BOUCLE_principale, sous le #LOGO_RUBRIQUE, il suffit

d’indiquer :

```
<BOUCLE_news(ARTICLES){id_rubrique}{titre_mot=News}{par date}{inverse}>
  <hr><h3>#TITRE</h3>
  [(#DATE|affdate)]

  <BOUCLE_news_machines(MOTS){id_article}{type=Machines}>
   [(#LOGO_MOT|right)]
  </BOUCLE_news_machines>

  [(#TEXTE|justifier)]
  [<p><font size=2>(#PS)</font>]
  [<p>(#NOTES)]

 </BOUCLE_news>
```

La BOUCLE_news fait tout le travail : elle affiche le titre de chaque news, son texte, et si nécessaire le post-scriptum et des notes de bas de page.

Comme à l’habitude, la BOUCLE_news_machines affiche le logo des machines concernées par la news.

Considérons notre page des news terminée. On pourra évidemment y ajouter les liens vers les articles de la même rubrique (les tests, les previews, les solutions...).

Et encore d’autres moyens de se compliquer la vie !

Noter les jeux, annoncer les dates de sortie

Notre site utilise désormais ce qui était prévu à l’origine :

- le rubriquage thématique (par genre de jeux) ;
- l’indication des machines concernées par chaque article ;
- la gestion des types d’articles (tests, news, previews...).

L’étape suivante consistera à créer des navigations parallèles (naviguer selon une unique machine, ou selon un type d’articles).

Mais avant de nous lancer dans le développement de ces navigations transversales, compliquons-nous un peu l’affaire, en ajoutant deux autres éléments indispensables aux sites de jeux vidéo :

- la note attribuée à un jeu ;
- la date de sortie officielle du jeu.

Ces deux éléments devraient être, désormais, simples à installer avec le système des mots-clés. Ils permettront eux-même de proposer des navigations transversales (notamment une page des jeux ayant les meilleures notes, et une page d’annonce des prochaines sorties).

L’attribution d’une note à un jeu

Il est logique d’attribuer une note à la fin d’un test :

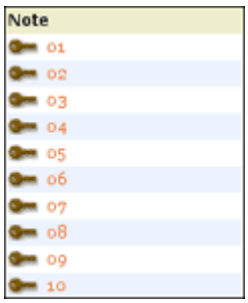
- pour les previews, c’est trop tôt, puisqu’il ne s’agit pas d’une version définitive ; pour les autres types d’articles (news, soluces...), ça n’a tout simplement rien à voir ;
- il est impossible d’attribuer une note à la rubrique-jeu, puisque cette rubrique peut présenter différentes versions d’un jeu.

Donc, nous décidons que, dans le fonctionnement de notre site, la note est liée à un article de test. Dans une même rubrique d’un jeu, puisqu’il peut y avoir plusieurs tests différents (selon les critiques, selon les machines...), il y aura donc plusieurs notes, associées à ces tests.

Arbitrairement, nous choisissons une notation sur 10 (de 1 pour nul, à 10 pour génial).

Retour à la page de gestion des mots-clés :

- nous créons un groupe de mots intitulé « Note » ;
- dans ce groupe de mots, nous créons 10 mots-clés, dont les noms sont successivement « 01 », « 02 », ..., « 09 », « 10 ». Ce qui donne :



On peut choisir, comme pour les machines, d’attribuer un logo différent à chaque note (par exemple des étoiles), et ensuite dans les squelettes utiliser les logos plutôt que le nombre indiqué par le mot-clé en toutes lettres. Par pure fainéantise, nous nous contenterons ici d’utiliser le texte des mots-clés.

Désormais, avant la publication d’un test, en plus de sélectionner les mots-clés correspondant au type de l’article (« Test », donc) et ceux des machines concernées, il faut penser à choisir une note. Si l’on veut un fonctionnement cohérent, il est évident qu’on ne doit sélectionner qu’une seule note par article.

Intégrer la note dans les squelettes

La boucle permettant d'afficher la note d'un article est très simple.

- Dans « article.html », plaçons la note de l'article sous la signature de l'auteur (avant le #PS et les #NOTES) :

```
<BOUCLE_note(MOTS){id_article}{inverse}{type=Note}>
  [<p align="right"><b>NOTE : (#TITRE)/10</b>]
 </BOUCLE_note>
```

Ajoutons également la mention de la note dans les « autres tests de ce jeu ». La BOUCLE_tests est ainsi modifiée :

```
<B_tests><p>Les tests de ce jeu :
 <ul>
 <BOUCLE_tests(ARTICLES){id_rubrique}{titre_mot=Test}>
  <li>
  <BOUCLE_tests_machines(MOTS){id_article}{type=Machines}>
   [(#LOGO_MOT)]
  </BOUCLE_tests_machines>
  <a href="#URL_ARTICLE">#TITRE</a>
  <BOUCLE_autre_note(MOTS){id_article}{inverse}{type=Note}>
   [ - NOTE : (#TITRE)/10]
  </BOUCLE_autre_note>
  - [(#DATE|affdate)]
  </BOUCLE_tests>
 </ul>
 </B_tests>
```

- Dans « rubrique.html », cette même BOUCLE_tests est à remplacer directement.

Voilà, l'intégration d'une note de test n'a pas pris cinq minutes ! Tout cela étant trop facile, nous verrons par la suite comment exploiter ailleurs cette note attribuée aux jeux.

La date de sortie officielle des jeux

L'affichage de la date de sortie officielle des jeux est plus problématique. En effet, une date ne peut pas être elle-même un mot-clé (nous avons déjà souligné que les mots-clés devaient être des éléments stables de la structure : on ne va donc pas créer des mots-clés de dates !).

Il existe une possibilité : utiliser la date de première publication de certains articles. Problème : quels articles ?

- Les dates de sortie sont connues (de manière plus ou moins approximative) longtemps à l'avance, donc on ne peut pas attendre qu'il y ait des articles consacrés au jeu pour pouvoir l'annoncer.

- Les articles d'un jeu peuvent traiter d'une plateforme ou de plusieurs. Or les dates de sorties selon les plateformes est très variables. On peut donc difficilement utiliser des articles communs à plusieurs plateformes pour annoncer les dates. Par exemple, *Alone in the Dark : the new nightmare*, est très similaire entre ses versions Playstation 2 et Dreamcast, il est donc logique de consacrer un test unique aux deux versions ; en revanche, les dates de sorties sont différentes sur ces plateformes.

- Tant que le jeu n'est pas commercialisé, les dates annoncées doivent être corrigées régulièrement (retards, plus grande précision...). Le webmestre doit donc facilement trouver où il doit modifier ces informations.

Dans la structure de notre site, on peut donc considérer que la date de sortie d'un jeu est une information

indépendante des autres informations (tests, previews, etc.) ; nous ne pouvons donc pas utiliser la « date de première publication » d'un autre article pour indiquer cette date de sortie. Nous allons procéder de la façon suivante (il en existe certainement d'autres) :

- nous créons un nouveau type d'article (en plus des tests, previews, etc.) ; installé dans le groupe « Type_article », nous le nommerons « Date_sortie » ;

- dans la rubrique du jeu, nous créons autant d'articles nécessaires qu'il y a de versions du jeu avec des dates de sortie différentes. Ces articles sont *vides* : ils n'ont qu'un titre et une date de « publication » (fixée manuellement à la date de sortie du jeu). Son titre lui-même n'a aucune importance, nous ne l'afficherons pas sur le site public ; en revanche, ce titre facilitera la navigation dans le site privé ;

- à chaque article associé à « Date_sortie », on attribue également le mot-clé de la machine concernée par cette date.

Voici par exemple comment procéder avec *Resident Evil : Code Veronica* :

- créer un nouvel article dans la rubrique « Code Veronica » ;
- titrer cet article « Sortie Veronica Dreamcast » (titre de convenance, sans autre intérêt que d'identifier ces articles dans l'espace privé) ;
- attribuer les mots-clés « Dreamcast » et « Date_sortie » à cet article ;
- « publier » l'article, et modifier sa date en « mai 2000 » (si l'on ne connait pas le jour exact, on peut sélectionner « n.c. » dans le menu déroulant du jour) ;

- recommencer l'opération avec un nouveau article « Sortie Veronica PS2 » ;
- lui attribuer les mots-clés « Playstation 2 » et « Date_sortie » ;
- publier et modifier la date (13 septembre 2001).

Pour le premier article, cela donne :

L'auteur de ces articles est totalement indifférent (inutile de perdre du temps à le supprimer, de toute façon il ne sera pas utilisé).

Notez l'intérêt de notre façon d'afficher les articles d'une rubrique *en fonction de certains types d'articles* (tests, previews...) : les articles ayant pour mot-clé « Date_sortie » ne sont pas affichés (avec les squelettes standards de SPIP, ces « articles » pourtant vides de tout contenu seraient affichés comme les autres).

Afficher la date dans les articles

Lorsque l'on se trouve dans un article (d'un jeu), la (les) date(s) de sortie sont contenus dans des articles associés au mot « Date_sortie », situés dans la même rubrique. Pour connaître la date de sortie d'un jeu depuis un article de test (par exemple), il faut donc récupérer le (ou les) article(s) installés dans la même rubrique et ayant pour mot-clé « Date_sortie ».

Dans le squelette « article.html », insérons (par exemple juste après le titre) :

```
<BOUCLE_sortie(ARTICLES){id_rubrique}{titre_mot=Date_sortie}>
  <li> #TITRE :
  [<b>(#DATE|affdate)</b>]
  </BOUCLE_sortie>
```

Cependant, nous avions décidé de ne pas utiliser le titre de l'article contenant la date. Nous souhaitons en effet unifier la présentation de cette information, sans pour autant nous obliger à saisir le titre toujours de la même façon. Nous n'allons donc pas afficher le #TITRE de cet article-bidon (ni même le titre de la rubrique, nous connaissons déjà le titre du jeu, inutile de le rappeler ici). En revanche, pour différencier (éventuellement) les différentes dates de sorties en fonction des machines, nous allons insérer :

```
<BOUCLE_sortie(ARTICLES){id_rubrique}{titre_mot=Date_sortie}>
        <li> Date de sortie
                <BOUCLE_machine_sortie(MOTS){id_article}{type=Machines}{",
"}>#TITRE</BOUCLE_machine_sortie> :
        [<b>(#DATE|affdate)</b>]
        </BOUCLE_sortie>
```

Ce qui lors de la visite donnera :

- Date de sortie Playstation : 22 août 2001
- Date de sortie Dreamcast : mai 2001

Cela ne nous satisfait pas encore (c'est un tutorial, nous avons le droit d'être exigeants !) : la BOUCLE_sortie affiche les dates de sortie sur toutes les machines ; or, notre article peut ne pas traiter de toutes ces machines. Nous voulons donc n'afficher que les dates de sortie correspondant aux machines traitées par l'article.

Attention, ça devient coton, et cela utilise une subtilité des boucles :

```
<BOUCLE_mac2(MOTS){id_article}{type=Machines}>
        <BOUCLE_sortie(ARTICLES){id_rubrique}{titre_mot=Date_sortie}>
        <BOUCLE_verifier_mot(ARTICLES){id_article}{id_mot}>
        <li> Date de sortie
                <BOUCLE_machine_sortie(MOTS){id_article}{type=Machines}{",
"}>#TITRE</BOUCLE_machine_sortie> :
        [<b>(#DATE|affdate)</b>]
        </BOUCLE_verifier_mot>
        </BOUCLE_sortie>
</BOUCLE_mac2>
```

Vous reconnaissez les BOUCLE_sortie et BOUCLE_machine_sortie, que nous n'avons pas modifées. En revanche, deux boucles apparaissent.

(1) L'ensemble est placé dans une BOUCLE_mac2, qui va récupérer les mots-clés des machines associées à cet article. À l'intérieur de cette boucle, nous avons donc une certaine valeur pour « id_mot » (l'identifiant de chaque machine concernée par l'article).

(2) La BOUCLE_sortie, inchangée, récupère tous les articles de la rubrique associés au mot-clé « Date_sortie ». Notez bien : cette boucle n'utilise pas « id_mot », donc les articles de cette boucle sont à la fois des boucles associés au mot-clé dont « id_mot » et qui n'y sont pas associés. Second point important, c'est une boucle (ARTICLES), cette boucle ne renvoit aucune information sur les mots-clés ; donc la valeur « id_mot » fournie par BOUCLE_mac2 reste inchangée à l'intérieur de cette boucle.

(3) La BOUCLE_verif_mot est une subtilité intéressante : c'est une boucle (ARTICLES) sélectionnant

l’article dont l’identifiant est « id_article » ; or nous sommes déjà dans la BOUCLE_sortie, qui a déjà retourné un article. Donc la BOUCLE_verif_mot retourne exactement le même résultat que la boucle précédente ! À cette différence que la BOUCLE_verif_mot instaure un critère supplémentaire, {id_mot}, l’article doit donc être associé au mot-clé « id_mot ». Résultat : si l’article retourné par BOUCLE_sortie est associé au mot-clé « id_mot », on continue ; si l’article de BOUCLE_sortie n’est pas associé, on s’arrête et on passe au suivant.

- On peut ainsi dire que la BOUCLE_verif_mot est un filtre : elle prend l’article et vérifie s’il correspond au critère du mot-clé.

- On peut le présenter autrement : BOUCLE_verif_mot est seulement un critère de sélection supplémentaire pour la BOUCLE_sortie. Le plus simple en effet aurait été de ne pas utiliser de BOUCLE_verif_mot, mais d’ajouter un critère à BOUCLE_sortie, ainsi :

```
<BOUCLE_sortie(ARTICLES){id_rubrique}{id_mot}{titre_mot=Date_sortie}>
```

C’est-à-dire : récupérer les articles de la même rubrique associés à la machine « id_mot » et au mot-clé « Date_sortie ». Cela aurait été plus direct, malheureusement SPIP n’accepte pas ce genre de constructions (il faut un seul critère portant sur les mots-clés dans une boucle). Une telle boucle est donc refusée par SPIP.

Vous remarquerez que nous avons empilé quatre boucles successives pour afficher une information aussi peu importante que la date de sortie d’un jeu ! On peut le déplorer, mais on peut préférer penser que, si l’on comprend bien la logique des boucles, les possibilités de SPIP sont beaucoup plus étendues que ce que les squelettes standards laissent supposer.

N.B. En utilisant un peu de code PHP très simple, on aurait obtenu le même résultat avec une seule boucle. Dans le développement d’un site, on aura certainement avantage à gagner du temps en préférant la version de raccourcie utilisant un peu de PHP. Mais nous tenions à montrer que des certains résultats peuvent très bien se passer de PHP.

Quelques sommaires alternatifs

Déjà une tripotée d'articles dans ce tutorial, et nous n'avons pas encore réalisé le sommaire !

Tenez-vous bien : nous n'allons pas encore nous y attaquer. Le sommaire général du site, c'est en effet la partie la plus difficile à réaliser graphiquement, et l'interface doit présenter les informations les plus importantes de manière cohérente (ni trop, ni pas assez...). Avant de créer le sommaire, il faut donc déjà connaître tous les types d'information présents sur le site, et les différentes structures de navigation. Le sommaire général, c'est donc pour la fin...

Nous allons donc commencer à réellement exploiter les informations de notre site pour créer des « sommaires » alternatifs. Les sites de jeux vidéo regorgent en effet de ces pages récapitulatives présentant les jeux selon différents critères : le calendrier des sorties, les meilleures notes...

Le tableau des meilleures notes

Créons un couple de fichiers pour afficher la liste des meilleurs jeux.

La page d'appel sera « notes.php3 », dont le contenu est :

```
<?php
$fond = "notes";
$delais = 24 * 3600;

include ("inc-public.php3");
?>
```

Nous pouvons maintenant commencer le squelette, « notes.html » :

```
<html>
<title>[#NOM_SITE_SPIP] Les meilleurs jeux</title>
</head>

<body>
<blockquote>

<h2>Les meilleurs jeux</h2>

<BOUCLE_notes(MOTS){type=Note}{par titre}{inverse}>
<h1>#TITRE/10</h1>

</BOUCLE_notes>

</blockquote>
</body>
</html>
```

Nous avons installé une première BOUCLE_notes, qui affiche tous les mots-clés de type « Note » (c'est-à-dire « 01 », « 02 », ..., « 10 ») par ordre inversé (« 10 » en haut de liste).

Affichons le titre des articles correspondant à chaque note (c'est-à-dire les articles associés au mot-clé renvoyé par la BOUCLE_notes) :

```
<BOUCLE_notes(MOTS){type=Note}{par titre}{inverse}>

 <B_articles>
 <h1>#TITRE/10</h1>

 <BOUCLE_articles(ARTICLES){id_mot}{par date}{inverse}>
  <li>#TITRE
 </BOUCLE_articles>

</BOUCLE_notes>
```

La seule subtilité : le mot (la note) est désormais affiché dans le *texte conditionnel avant* de la BOUCLE_articles. De ce façon, seules les notes qui ont été réellement attribuées à des articles sont affichées.

Nous affichons ici le titre de l'article. Ce n'est pas ce qui nous intéresse. Nous voulons afficher :

- le titre du jeu, c'est-à-dire le titre de la rubrique qui contient l'article ;
- la (les) machine(s) sur lesquelles tournent ces jeux ;
- la (les) date(s) de sortie du jeu.

Rien de bien compliqué, nous avons déjà réalisé ces opérations sur d'autres pages (pour la date de sortie, nous recopions directement ce que nous avons développé pour « article.html ») :

```
<BOUCLE_notes(MOTS){type=Note}{par titre}{inverse}>

        <B_articles>
        <h1>#TITRE/10</h1>
        <ul>

        <BOUCLE_articles(ARTICLES){id_mot}{par date}{inverse}>
                <BOUCLE_rubrique_jeu(RUBRIQUES){id_rubrique}>
                <li><b><a href="#URL_ARTICLE">#TITRE</a></b>
                </BOUCLE_rubrique_jeu>
                <BOUCLE_machines_jeu(MOTS){id_article}>
                #LOGO_MOT
                </BOUCLE_machines_jeu>
                <BOUCLE_mac2(MOTS){id_article}{type=Machines}>
                        <BOUCLE_sortie(ARTICLES){id_rubrique}{titre_mot=Date_sortie}>
                        <BOUCLE_verifier_mot(ARTICLES){id_article}{id_mot}>
                        <br> Date de sortie
                                <BOUCLE_machine_sortie(MOTS){id_article}{type=Machines}{",
"}>#TITRE</BOUCLE_machine_sortie> :
                        [<b>(#DATE|affdate)</b>]
                        </BOUCLE_verifier_mot>
                        </BOUCLE_sortie>
                </BOUCLE_mac2>

        </BOUCLE_articles>
        </ul>
        </B_articles>

</BOUCLE_notes>
```

Seule petite subtilité : la balise #URL_ARTICLE est utilisée dans une boucle de type RUBRIQUES, de

façon en réaliser le lien vers la page du test plutôt que sur la page générale de la rubrique du jeu. Cette utilisation d'un élément propre aux ARTICLES est possible ici, car l'#URL_ARTICLE est réalisé à partir de la variable « id_article ». Cet id_article est fourni par la BOUCLE_articles, puis la BOUCLE_rubrique_jeu ne la modifie pas (puisqu'il s'agit d'une boucle de RUBRIQUES). C'est le même principe qui permet d'utiliser « id_mot » dans la BOUCLE_verifier_mot.

Dernier problème à régler : cette page fonctionne bien au lancement du site et pendant les essais, avec une dizaine d'articles, mais quand votre site aura un an et plusieurs milliers de jeux tests (au minimum !), la liste sera trop longue. De toute façon, afficher indifféremment des jeux sortis il y a une semaine et des jeux sortis depuis plusieurs mois n'a pas grand intérêt : un jeu noté « 9/10 » il y a un an serait certainement moins bien noté aujourd'hui. Introduisons donc une limitation de temps en modifiant la BOUCLE_articles :

```
<BOUCLE_articles(ARTICLES){id_mot}{age<90}{par date}{inverse}>
```

{age<90}, c'est-à-dire les articles publiés depuis moins de 3 mois.

Le calendrier des sorties

Incontournable sur les pages de jeux vidéo, la page indiquant les sorties récentes et les prochaines sorties.

La page d'appel sera « sorties.php3 » :

```
<?
$fond = "sorties";
$delais = 24 * 3600;

include ("inc-public.php3");
?>
```

Commençons le fichier « sorties.html » ainsi :

```
<html>
<title>[#NOM_SITE_SPIP] Les sorties</title>
</head>

<body>
<blockquote>

<h2>Les prochaines sorties</h2>

<BOUCLE_sorties(ARTICLES){titre_mot=Date_sortie}{par date}{age < 0}>
<p> #TITRE - [(#DATE|affdate)]
</BOUCLE_sorties>

</blockquote>
</body>
</html>
```

Le #TITRE est celui de l'article contenant la date de sortie ; c'est un titre bidon utilisé par commodité dans l'espace privé. Il faut donc le remplacer par le titre du jeu, c'est-à-dire le titre de la rubrique qui contient cet article. Toujours le même processus. Nous voulons de plus afficher le logo des machines concernées par cette sortie. Là non plus rien de compliqué.

```
<BOUCLE_sorties(ARTICLES){titre_mot=Date_sortie}{par date}{age < 0}>
<p>
 <BOUCLE_rubrique_jeu(RUBRIQUES){id_rubrique}>
  <b><a href="#URL_RUBRIQUE">#TITRE</a></b>
 </BOUCLE_rubrique_jeu>

 <BOUCLE_machines_jeu(MOTS){id_article}>
 #LOGO_MOT
 </BOUCLE_machines_jeu>

  - [(#DATE|affdate)]
</BOUCLE_sorties>
```

Il y a ici un défaut de navigation important : il y a systématiquement un lien vers la rubrique du jeu. Or, et surtout pour les jeux qui ne sont pas encore sortis, il est très possible qu'il n'y ait aucune information publiée sur ce jeu, en dehors de sa date de sortie. On risque alors de pointer vers une rubrique qui ne présente aucune information.

Modifions donc encore ce code, pour afficher non plus un lien vers la rubrique elle-même, mais vers chaque type d'articles.

Afin d'obtenir une présentation un peu plus condensée, commençons par créer une interface sous forme de lignes et de colonnes (et au passage, nous supprimons le lien hypertexte fautif) :

```
<B_sorties>
<table cellpadding="4">
<BOUCLE_sorties(ARTICLES){titre_mot=Date_sortie}{par date}{age < 0}>
<tr>

 <BOUCLE_rubrique_jeu(RUBRIQUES){id_rubrique}>
  <td><b>#TITRE</b></td>
 </BOUCLE_rubrique_jeu>

 <td align="center">
 <BOUCLE_machines_jeu(MOTS){id_article}>
 #LOGO_MOT
 </BOUCLE_machines_jeu>
  
 <//B_machines_jeu>
 </td>

  <td>[(#DATE|affdate)]</td>
</tr>
</BOUCLE_sorties>
</table>
</B_sorties>
```

Ce code ne présente pas de grosse difficulté. Expliquons rapidement le positionnement des balises du tableau.

- <table> et </table> sont placés en *texte optionnel* de la BOUCLE_sorties ; de cette façon, si la liste des jeux à venir est vide, on évite d'afficher un <table></table> peu élégant.

- Les <tr> et </tr> sont à l'intérieur de la BOUCLE_sorties, et englobent l'intégralité de son contenu ; il s'agit ici de créer les lignes du tableau.

- Dans la BOUCLE_rubrique_jeu, nous plaçons les <td> et </td> directement dans la boucle, autour du #TITRE. Nous savons en effet que, l'article étant forcément dans une rubrique, il y aura forcément une (et une seule) réponse dans BOUCLE_rubrique_jeu (inutile de placer ces codes

dans du *texte optionnel* ; et comme il n'y aura qu'une seule réponse, on peut se placer à l'intérieur de la boucle).

- C'est différent pour la BOUCLE_machines_jeu : les <td align="center"> et </td> sont à l'extérieur de la boucle, et ne sont même pas placés en *code optionne*. En effet : (1) ils sont à l'extérieur de la boucle, car il peut y avoir plusieurs machines concernées par cette date de sortie (c'est-à-dire plusieurs mots-clés de machines associés à l'article) ; si on avait placé ces balises à l'intérieur de la boucle, comme précédemment, le tableau perdrait son alignement, puisqu'on créerait des « cases » supplémentaires à chaque nouveau logo de mot-clé ; (2) ils ne sont pas placés dans du texte optionnel, car ce ne sont pas des balises optionnelles ; si on a oublié d'indiquer une machine pour la date de sortie, il faut tout de même créer cette « case » du tableau, certes vide, pour conserver l'alignement. Enfin, nous avons placé un   en *texte optionnel alternatif* : s'il n'y a pas de mot-clé de machine associé à la date de sortie, on affiche cet espace insécable pour que la « case » ait un contenu (dans de nombreux butineurs, une case <td></td> n'est pas affichée de la même façon qu'une case <td> </td>.

Nous allons maintenant afficher pour chaque jeu la liste des Tests qui concernent ce jeu (insérons cela juste avant le </tr> final de la ligne) :

```
<td>
 <B_tests_jeu>
 <b>Test :</b>
 <BOUCLE_tests_jeu(ARTICLES){id_rubrique}{titre_mot=test}>
  <br><a href="#URL_ARTICLE">#TITRE</a>
 </BOUCLE_tests_jeu>
  
 <//B_tests_jeu>
 </td>
```

Cette BOUCLE_tests_jeu utilise des méthodes déjà expliquées.

Cependant, cela ne nous convient pas encore : nous affichons ici tous les tests, pour toutes les machines, alors que notre date de sortie ne concerne pas forcément toutes les versions du jeu (on peut ici afficher la date de sortie de la versions Dreamcast, alors qu'on a déjà publié des tests pour les versions Playstation, Dreamcast, Window...).

On ne veut donc afficher que les tests qui concernent les versions concernées par cette date de sortie. Nous avons déjà réalisé un processus similaire dans la page « article.html » pour n'afficher que les dates de sortie sur les mêmes machines que l'article affiché (BOUCLE_mac2). Cela donne :

```
<td>
 <BOUCLE_mac2(MOTS){id_article}{type=Machines}>

  <BOUCLE_tests_jeu(ARTICLES){id_rubrique}{titre_mot=test}>
  <BOUCLE_verifier_mot(ARTICLES){id_article}{id_mot}>
   <li><a href="#URL_ARTICLE">#TITRE</a>
  </BOUCLE_verifier_mot>
  </BOUCLE_tests_jeu>

 </BOUCLE_mac2>
  
 </td>
```

La BOUCLE_mac2 récupère la liste des machines de la date de sortie. La BOUCLE_tests_jeu récupère l'ensemble des tests de jeu de la rubrique (toutes machines confondues). La BOUCLE_verifier_mot vérifie pour chaque test récupéré dans BOUCLE_tests_jeu qu'il est associé à la machine récupérée par BOUCLE_mac2.

Nous avons pour l'instant supprimé l'affichage de « Tests :» en texte conditionnel pour des raisons de

lisibilié. Voici maintenant le code, exactement sur le même principe, qui permet d'afficher cette indication (ainsi que le) :

```
<td>
	<B_mac2>
	<b>Tests :</b>
		<BOUCLE_mac2(MOTS){id_article}
{type=Machines}><BOUCLE_tests_jeu(ARTICLES){id_rubrique}
{titre_mot=test}><BOUCLE_verifier_mot(ARTICLES){id_article}{id_mot}>
			<li><a href="#URL_ARTICLE">#TITRE</a>
		</BOUCLE_verifier_mot></BOUCLE_tests_jeu></BOUCLE_mac2>
	 
	<//B_mac2>
	</td>
```

Notez bien : les BOUCLE_mac2, BOUCLE_tests_jeu et BOUCLE_verifier_mot sont « collées », c'est-à-dire qu'elles ne sont plus séparées par des retours à la ligne ni par des espaces. En effet, nous testons le « contenu » de la BOUCLE_mac2 pour savoir si nous affichons l'indication « Tests :» ou le   (les textes conditionnels). Si nous avions conservé les retours à la ligne ou des espaces blancs entre les boucles, quel que soit le résultat de BOUCLE_tests_jeu et de BOUCLE_verifier_mot, la BOUCLE_mac2 aurait été considérée comme ayant un contenu à cause de l'affichage de ces retours-chariot.

Pour afficher une liste des previews, dupliquons simplement ce code, en modifiant le nom des boucles et en remplaçant titre_mot=test par titre_mot=preview :

```
<td>
	<B_mac3>
	<b>Previews :</b>
	<BOUCLE_mac3(MOTS){id_article}{type=Machines}><BOUCLE_preview_jeu(ARTICLES)
{id_rubrique}{titre_mot=preview}><BOUCLE_verifier_mot3(ARTICLES){id_article}{id_mot}>
			<li><a href="#URL_ARTICLE">#TITRE</a>
		</BOUCLE_verifier_mot3></BOUCLE_preview_jeu></BOUCLE_mac3>
	 
	<//B_mac3>
	</td>
```

Ajoutons enfin une boucle pour les « news » concernant ce jeu. Ici nous ne nous préoccuperons pas des machines, puisque toutes les news sont affichées sur une unique page. Nous n'afficherons donc pas de #TITRE, mais directement la mention « Les news » ; le lien hypertexte pointe vers la page spécifique que nous avons créée pour ce jeu. Notez enfin la restriction {0,1} qui permet de ne récupérer qu'une seule « news » pour créer un unique lien.

```
<td>
	<BOUCLE_news_jeu(ARTICLES){id_rubrique}{titre_mot=news}{0,1}>
		<br><a href="news_jeu.php3?id_rubrique=#ID_RUBRIQUE">Les news</a>
	</BOUCLE_news_jeu>
	 
	<//B_news_jeu>
	</td>
```

Pour être exhaustif, il resterait à recommencer l'opération avec les Soluces et les Astuces. Mais puisqu'il s'agit ici d'afficher une liste de sorties, considérons que ces informations ne sont pas pertinentes ici. Même si, techniquement, on peut les afficher facilement, faisons un choix éditorial : l'information n'est pas intéressante ici et surchargerait l'interface, donc nous préférons ne pas l'indiquer. (Mais si cela vous amuse, vous pouvez vous entraîner en les intégrant vous-même.)

Une première version du sommaire

Nous commençons à avoir une idée plus précise de la navigation de notre site. Réalisons une première version du sommaire.

```
<html>
<head>
<title>[#NOM_SITE_SPIP] Sommaire</title>
</head>
<body>

<center>
<a href="sorties.php3">Les prochaines sorties</a>
| <a href="notes.php3">Les meilleurs jeux du moment</a>
</center>

 <BOUCLE_secteurs(RUBRIQUES){id_parent=0}{par titre}>
  <p><b><a href="#URL_RUBRIQUE">#TITRE</a></b>

  <B_sous_rubriques>
  <ul>
  <BOUCLE_sous_rubriques(RUBRIQUES){id_parent}>
   <li><a href="#URL_RUBRIQUE">#TITRE</a>
  </BOUCLE_sous_rubriques>
  </ul>
  </B_sous_rubriques>

 </BOUCLE_secteurs>

</body>
</html>
```

Avant de placer les boucles qui permettront d'afficher les nouveautés du site, nous nous contentons de créer (à la main) les liens vers les deux sommaires alternatifs, et d'afficher la structure du site (uniquement les grandes rubriques et leurs sous-rubriques).

Installons la liste des cinq plus récents tests publiés sur le site. Pour chacun, nous afficherons (en plus du logo, du titre du test, et son descriptif) :

- une boucle (RUBRIQUES) pour récupérer le titre du jeu ;
- une boucle (MOTS) avec pour type de mot « note » pour afficher la note ;
- une seconde boucle (MOTS) avec pour type « machines » pour afficher le logo des machines concernées.

Rien que nous ne sachions déjà faire.

```
<BOUCLE_tests(ARTICLES){titre_mot=test}{par date}{inverse}{0,5}>
 <p><div style="border:1px solid black">
 [(#LOGO_ARTICLE_RUBRIQUE|left|#URL_ARTICLE)]

 <BOUCLE_rub_tests(RUBRIQUES){id_rubrique}>
 <h3>#TITRE</h3>
 </BOUCLE_rub_tests>

 <h4>#TITRE</h4>

 <BOUCLE_note_tests(MOTS){id_article}{type=Note}>
 <b>NOTE : #TITRE/10</b><p>
 </BOUCLE_note_tests>

 <BOUCLE_mac_tests(MOTS){id_article}{type=Machines}>
 [(#LOGO_MOT|left)]
 </BOUCLE_mac_tests>

 [(#DESCRIPTIF)]
 <p align="right"><a href="#URL_ARTICLE">Lire ce test...</a>
 </div>
</BOUCLE_tests>
```

Ajoutons maintenant les cinq dernières previews, cinq soluces et cinq astuces. Les présentations sont différentes à chaque fois, mais le principe est rigoureusement similaire à la boucle précédente (dans des versions simplifiées, puisque nous voulons afficher moins d'informations) :

```
<B_previews>
<p><b>Previews :</b>
<ul>
<BOUCLE_previews(ARTICLES){titre_mot=preview}{par date}{inverse}{0,5}>

 <BOUCLE_rub_previews(RUBRIQUES){id_rubrique}>
 <li><b>#TITRE</b> /
 </BOUCLE_rub_previews>

 <a href="#URL_ARTICLE">#TITRE</a>

 <BOUCLE_mac_previews(MOTS){id_article}{type=Machines}>
 [(#LOGO_MOT)]
 </BOUCLE_mac_previews>

</BOUCLE_previews>
</ul>
</B_previews>

<B_soluces>
<p><b>Soluces :</b>
<ul>
<BOUCLE_soluces(ARTICLES){titre_mot=soluce}{par date}{inverse}{0,5}>

 <BOUCLE_rub_soluces(RUBRIQUES){id_rubrique}>
 <li><b><a href="#URL_ARTICLE">#TITRE</a></b>
 </BOUCLE_rub_soluces>

 <BOUCLE_mac_soluces(MOTS){id_article}{type=Machines}>
 (#TITRE)
 </BOUCLE_mac_soluces>

</BOUCLE_soluces>
</ul>
</B_soluces>


<B_astuces>
<p><b>Astuces :</b>
<ul>
<BOUCLE_astuces(ARTICLES){titre_mot=astuces}{par date}{inverse}{0,5}>

 <BOUCLE_rub_astuces(RUBRIQUES){id_rubrique}>
 <li><b><a href="#URL_ARTICLE">#TITRE</a></b>
 </BOUCLE_rub_astuces>

 <BOUCLE_mac_astuces(MOTS){id_article}{type=Machines}>
 (#TITRE)
 </BOUCLE_mac_astuces>

</BOUCLE_astuces>
</ul>
</B_astuces>
```

Enfin, la liste des news. Toujours le même principe ; cette fois, le lien hypertexte pointe vers la page commune aux news d'un jeu. Le lien se fait sur le titre du jeu (c'est-à-dire le titre de la rubrique), et nous affichons le titre de la news.

```
<B_news>
<p><b>News :</b>
<ul>
<BOUCLE_news(ARTICLES){titre_mot=news}{par date}{inverse}{0,5}>

 <BOUCLE_rub_news(RUBRIQUES){id_rubrique}>
 <li><b><a href="news_jeu.php3?id_rubrique=#ID_RUBRIQUE">#TITRE</a>:</b>
 </BOUCLE_rub_news>
 #TITRE

</BOUCLE_news>
</ul>
</B_news>
```

Il nous reste désormais à indiquer la structure thématique du site (par grand genre de jeux : Action/aventure, Plateforme, Sport...). Là encore rien de bien compliqué :

```
<BOUCLE_secteurs(RUBRIQUES){id_parent=0}{par titre}>
  <p><b><a href="#URL_RUBRIQUE">#TITRE</a></b>

  <B_sous_rubriques>
  <ul>
  <BOUCLE_sous_rubriques(RUBRIQUES){id_parent}>
   <BOUCLE_art_sous(ARTICLES){id_rubrique}>
   </BOUCLE_art_sous>
   <li><a href="#URL_RUBRIQUE">#TITRE</a>
   <//B_art_sous>
  </BOUCLE_sous_rubriques>
  </ul>
  </B_sous_rubriques>

 </BOUCLE_secteurs>
```

Notons tout de même cette subtilité : dans les sous-rubriques, la BOUCLE_art_sous vérifie la présence d'articles (cette boucle n'affiche rien). L'affichage du titre et du lien et du titre de la sous-rubrique n'est effectif que s'il n'y pas de rubrique dans la sous-rubrique, puisque cet affichage est placé dans le *texte optionnel alternatif*. En effet, il ne faudrait pas afficher les rubriques de jeu placés directement dans une rubrique.

Notre sommaire est complet. Pas terminé pour autant (dans les articles suivants, nous étendrons encore ses possibilités), mais complet :

- la structure thématique du site par grande rubriques de type de jeux est présentée ;
- les tests récents (dont nous considérons - et c'est un choix éditorial - qu'ils constituent l'élément le plus important du site) sont affichés de manière très visible ;
- nous exploitons l'indication des machines ;
- nous exploitons les « types d'articles » (les news, previews, tests, soluces, astuces sont présentés séparément) ;
- la note des jeux est exploitée, ainsi que les dates des sorties...

Tout cela, notez bien, sans utiliser la moindre ligne de code PHP, et sans modifier la structure de la base de données. Nous n'avons même pas détourné certaines indications des articles pour y arriver (on aurait pu par exemple décider que le surtitre des articles serait utilisé pour indiquer la machine, et le soustitre pour indiquer la note ; nous aurions ainsi pu afficher ces informations).

Un sommaire pour chaque machine

Les amateurs de jeux vidéo possèdent généralement une seule machine (deux pour les accros, trois pour les collectionneurs...). Une page de sommaire général, si elle se justifie à l'arrivée sur le site (le visiteur indique l'URL du site, on ne sait donc pas à priori ce qu'il utilise comme machine), contient trop d'informations superflues pour le propriétaire d'une seule machine : le possesseur d'une Playstation 2 n'est pas concerné par le dernier jeu sorti sur Dreamcast.

Donc, il faut pouvoir présenter un sommaire différent pour chaque machine. Un sommaire qui n'affiche que les jeux de cette machine.

Nous allons donc créer un squelette dont le contenu change en fonction de la machine. Nous nommerons ces fichiers : « sommaire_machine.php3 » et « sommaire_machine.html ». La machine étant indiquée, dans notre structure, par un mot-clé, ces pages dépendront donc d'un « id_mot ». Chaque machine différente sera donc appelée par une URL du type :

- sommaire_machine.php3 ?id_mot=23 (23 étant le numéro du mot-clé de la machine).

Revenons à notre sommaire général : nous allons y inclure les liens vers ces sommaires spécialisés ; la création automatique de ces liens (avec les « id_mot » correspondants) nous évitera de devoir effectuer les tests en cherchant les numéros (id_mot) de chaque machine. Dans « sommaire.html », insérons la boucle suivante :

```
<table cellpadding=5><tr>
<BOUCLE_somm_mac(MOTS){type=Machines}{par titre}>
<td><a href="sommaire_machine.php3?id_mot=#ID_MOT">#LOGO_MOT</a></td>
</BOUCLE_somm_mac>
</tr></table>
```

Attention : si l'on a réalisé des logos avec survol, le lien ne fonctionnera pas correctement, car SPIP ajoutera une commande javascript. Le code ci-dessus ne fonctionne qu'à la condition que le logo de chaque mot ne soit constitué que d'un unique fichier (pas d'image de survol).

Le moyen le plus simple et le plus efficace d'éviter un tel désagrément consiste à ne pas utiliser directement #LOGO_MOT, qui construit l'intégralité de la balise HTML de l'image, éventuellement avec un petit code javascript pour gérer le survol, mais à la place l'appel d'une image dont le fichier est : [(#LOGO_MOT|fichier)] (ce qui retourne uniquement le nom du fichier de l'image).

La boucle devient :

```
<table cellpadding=5><tr>
<BOUCLE_somm_mac(MOTS){type=Machines}{par titre}>
<td><a href="sommaire_machine.php3?id_mot=#ID_MOT"><img src='IMG/[(#LOGO_MOT|fichier)]'
border=0></a>
</td>
</BOUCLE_somm_mac>
</tr></table>
```

Le fichier « sommaire_machine.php3 »

Comme toujours :

```
<?php
$fond = "sommaire_machine";
$delais = 24 * 3600;

include ("inc-public.php3");
?>
```

Le squelette du sommaire par machine

Inutile de nous compliquer la vie : dupliquons le fichier « sommaire.html », et renommons-le « sommaire_machine.html ».

Puisque cette page dépend d'un « id_mot », nous commençons par ajouter une BOUCLE_principale qui va contenir l'intégralité des autres boucles. Il suffit donc d'ajouter l'appel de cette boucle juste après <body> et de la refermer juste avant </body> (tout en fin de fichier) :

```
<body>
<BOUCLE_principale(MOTS){id_mot}>

     ...

</BOUCLE_principale>
</body>
```

Le titre de la page, qui jusqu'à présent ne contenait que : <h1>#NOM_SITE_SPIP</h1, est modifié pour afficher le nom et le logo de la machine (c'est-à-dire du mot-clé correspondant à « id_mot ») :

```
<h1>#NOM_SITE_SPIP / #LOGO_MOT #TITRE</h1>
```

La BOUCLE_somm_mac, qui affiche tous les logos des machines pour créer les liens vers les sommaires spécifiques, ne doit plus contenir le logo de la machine principale (nous sommes déjà sur cette page, inutile donc de créer un lien vers elle-même). Il suffit d'insérer le critère {exclus} dans la définition de cette boucle :

```
<BOUCLE_somm_mac(MOTS){type=Machines}{exclus}{par titre}>
...
</BOUCLE_somm_mac>
```

La BOUCLE_tests, qui affiche les derniers tests publiés, ne doit désormais afficher que les tests qui concernent des jeux sur la machine sélectionnée. Pour cela, nous allons ajouter à l'intérieur de cette boucle une BOUCLE_verif_test, qui va vérifier pour chaque article qu'il concerne bien la machine choisie.

Inconvénient désormais : la BOUCLE_tests affichait les cinq derniers tests. Désormais, sur ces cinq articles, nous ne sélectionnons que ceux qui concernent notre machine. Il est fort probable que cela donne moins de cinq articles, et dans certains cas extrêmes, cela n'affichera plus aucun article. Modifions le critère, et décidons d'afficher non pas les cinq derniers tests ({0,5}), mais tous les tests publiés depuis moins de trois mois ({age<90}) ; on pourra modifier cette valeur en fonction de l'activité réelle du site (si le site est très actif, on réduira le délai de cette boucle, si le site publie peu d'articles, on pourra le rallonger).

Au passage, supprimons la petite boucle qui affichait le logo des machines concernées : cela devient inutile, puisque cette page n'affiche que les informations d'une unique machine.

La boucle devient :

```
<BOUCLE_tests(ARTICLES){titre_mot=test}{par date}{inverse}{age < 90}>
<BOUCLE_verif_test(ARTICLES){id_article}{id_mot}>
 <p><div style="border:1px solid black">

 [(#LOGO_ARTICLE_RUBRIQUE|left|#URL_ARTICLE)]

 <BOUCLE_rub_tests(RUBRIQUES){id_rubrique}>
 <h3>#TITRE</h3>
 </BOUCLE_rub_tests>

 <h4>#TITRE</h4>

 <BOUCLE_note_tests(MOTS){id_article}{type=Note}>
 <b>NOTE : #TITRE/10</b><p>
 </BOUCLE_note_tests>

 [(#DESCRIPTIF)]
 <p align="right"><a href="#URL_ARTICLE">Lire ce test...</a>
 </div>
</BOUCLE_verif_test>
</BOUCLE_tests>
```

Le principe est identique pour les previews, soluces, astuces, news : on ajoute une boucle de vérification, et on supprime la boucle qui affiche la liste des machines concernées :

```
<B_previews>
<p><b>Previews :</b>
<ul>
<BOUCLE_previews(ARTICLES){titre_mot=preview}{par date}{inverse}{age
< 90}><BOUCLE_verif_previews(ARTICLES){id_article}{id_mot}>
<BOUCLE_rub_previews(RUBRIQUES){id_rubrique}>
<li><b>#TITRE</b> / </BOUCLE_rub_previews>
<a href="#URL_ARTICLE">#TITRE</a>
</BOUCLE_verif_previews></BOUCLE_previews>
</ul>
</B_previews>
<B_soluces>
<p><b>Soluces :</b>
<ul>
<BOUCLE_soluces(ARTICLES){titre_mot=soluce}{par date}{inverse}{age
< 90}><BOUCLE_verif_soluces(ARTICLES){id_article}{id_mot}>
<BOUCLE_rub_soluces(RUBRIQUES){id_rubrique}>
<li><b><a
href="#URL_ARTICLE">#TITRE</a></b>
</BOUCLE_rub_soluces>
</BOUCLE_verif_soluces></BOUCLE_soluces>
</ul>
</B_soluces>
<B_astuces>
<p><b>Astuces :</b>
<ul>
<BOUCLE_astuces(ARTICLES){titre_mot=astuces}{par
date}{inverse}{0,5}><BOUCLE_verif_astuces(ARTICLES){id_article}{id_mot}>
<BOUCLE_rub_astuces(RUBRIQUES){id_rubrique}>
<li><b><a
href="#URL_ARTICLE">#TITRE</a></b>
</BOUCLE_rub_astuces>
</BOUCLE_verif_astuces></BOUCLE_astuces>
</ul>
</B_astuces>
<B_news>
<p><b>News :</b>
<ul>
<BOUCLE_news(ARTICLES){titre_mot=news}{par
date}{inverse}{0,5}><BOUCLE_verif_news(ARTICLES){id_article}{id_mot}>
<BOUCLE_rub_news(RUBRIQUES){id_rubrique}>
<li><b><a
href="news_jeu.php3?id_rubrique=#ID_RUBRIQUE">#TITRE</a>:</b>
</BOUCLE_rub_news> #TITRE
</BOUCLE_verif_news></BOUCLE_news>
</ul>
</B_news>
```

Important. Vous noterez ci-dessus que la boucle de vérification est « collée » à la première boucle (par exemple, entre <BOUCLE_previews(ARTICLE)...> et <BOUCLE_verif_previews(ARTICLE)...>, il n'y a ni espace ni retour à la ligne). En effet, le *texte conditionnel avant* de la première boucle (ici, la mention « Previews ») est activé si la boucle contient au moins un caractère. S'il y a un espace ou un retour chariot avant l'appel de la boucle de vérification, alors même si la boucle n'affiche aucun article (car, depuis trois mois, on n'a publié aucun article de ce type pour cette machine), cet espace parasite activerait l'affichage du code conditionnel.

Voyons maintenant l'affichage des prochaines sorties. On peut décider de créer une nouvelle page des sorties, concernant uniquement la machine sélectionnée, mais nous préférerons ici afficher directement ces sorties sur notre sommaire spécifique (les sorties n'étant pas si nombreuses par machines). En appliquant les mêmes méthodes que précédemment, on remplace le lien « Les prochains sorties » par :

```
<B_sorties>
<b>Les sorties #TITRE du prochain mois:</b>
<ul>
<BOUCLE_sorties(ARTICLES){titre_mot=Date_sortie}{par date}{age <
0}><BOUCLE_sorties_verif(ARTICLES){id_article}{id_mot}>
<li>[(#DATE|affdate)]:
<BOUCLE_rub_sorties(RUBRIQUES){id_rubrique}> #TITRE
</BOUCLE_rub_sorties>
</BOUCLE_sorties_verif></BOUCLE_sorties>
</ul>
</B_sorties>
```

Sur le même principe, affichons la liste des jeux préférés pour cette machine depuis six mois :

```
<B_notes>
<b>Nos jeux préférés depuis 6 mois:</b>
<ul>
<BOUCLE_notes(ARTICLES){type_mot=note}{par
titre_mot}{inverse}{age<180}><BOUCLE_notes_verif(ARTICLES){id_article}{id_mot}>
<BOUCLE_rub_notes(RUBRIQUES){id_rubrique}> <li> #TITRE
</BOUCLE_rub_notes>
<BOUCLE_lanote(MOTS){id_article}{type=note}> (#TITRE/10)
</BOUCLE_lanote> </BOUCLE_notes_verif></BOUCLE_notes>
</ul>
</B_notes>
```

La seule subtilité pour l'instant est le classement des articles associés aux mots-clés de type « note » selon le critère {par titre_mot}{inverse} (fonction apparue dans le version 1.3 de SPIP). En effet, les mots de type « note » sont de la forme « 01 », « 02 »..., « 10 » ; on peut donc demander à les classer de 10 à 1.

Cependant, cette présentation ne nous convient pas encore : nous prétendons afficher nos « jeux préférés », alors qu'il s'agit seulement d'un classement par note. Si un jeu est sorti depuis moins de six mois et s'est ramassé une note catastrophique, il sera dans les « jeux préférés » (en bas de liste, mais présent tout de même) !

Nous devons donc trouver un moyen d'afficher uniquement les jeux ayant eu une note supérieure à « 07 » (choix arbitraire). Si vous pensez utiliser pour cela la BOUCLE_notes, en y ajoutant simplement la mention {titre_mot>07}, cela ne fonctionnera pas : SPIP n'accepte qu'un seul critère basé sur les mots-clés dans les boucles autres que les boucles de type (MOTS).

Nous allons utiliser la BOUCLE_lanote (qui affiche la note), et lui demander de ne sélectionner que les notes supérieures à 7 (puisqu'il s'agit ici d'une boucle de type (MOTS), on peut lui fournir plusieurs contraites sur les mots-clés). La BOUCLE_rub_notes, qui affiche le titre du jeu, est alors placé en *texte optionnel après* de la BOUCLE_lanote :

- si la note est supérieure à 7, la BOUCLE_rub_notes est effectuée ;
- sinon rien n'est affiché.

Attention : il est toujours délicat de placer une boucle dans le *texte optionnel* d'une autre boucle (SPIP peut se mélanger facilement les pinceaux). Pour une raison étrange, cela est possible dans le *texte optionnel après*, mais pas dans le *texte optionnel avant*.

Sachant que nous ne récupérerons que les articles notés « 08 », « 09 », « 10 », classés du meilleur au moins bon, nous considérons désormais inutile de rappeler la note elle-même. La BOUCLE_lanote est donc vide, elle ne sert plus qu'à activer ou désactiver la BOUCLE_rub_notes.

Enfin, puisque la BOUCLE_lanote devient un nouveau « filtre » appliqué à la sélection d'articles, il faut supprimer les blancs et les retours chariot autour de cette boucle, de façon à ne réellement activer l'expression « Nos jeux préférés » que si au moins un article est vraiment affiché.

Cela nous donne :

```
<B_notes>
<b>Nos jeux préférés depuis 6 mois:</b>
<ul>
<BOUCLE_notes(ARTICLES){type_mot=note}{par titre_mot}{inverse}
{age<180}><BOUCLE_notes_verif(ARTICLES){id_article}{id_mot}><BOUCLE_lanote(MOTS)
{id_article}{titre>07}{type=note}>
        </BOUCLE_lanote>
        <BOUCLE_rub_notes(RUBRIQUES){id_rubrique}>
        <li> <a href="#URL_RUBRIQUE">#TITRE</a>
        </BOUCLE_rub_notes>
        </B_lanote></BOUCLE_notes_verif></BOUCLE_notes>
</ul>
</B_notes>
```

Tout cela peut sembler un peu indigeste, mais le tout est que cela fonctionne, et toujours sans la moindre fonction PHP !

Maintenant que nos pages de sommaires spécifiques sont presque terminées, nous constatons, en passant de l'une à l'autre, que certaines pages sont vides. En effet, nous n'affichons ici que des informations de moins de trois mois (pour les tests, les news...). Si une machine n'a été concernée par aucun article depuis trois mois, ou même si une machine n'a encore aucun article du tout (même si cela est prévu par un mot-clé, il est très possible que personne sur votre site n'ait encore écrit au sujet des jeux Linux).

Il faut donc modifier la BOUCLE_somm_mac (celle qui affiche les logos des différentes machines) pour que seules les machines ayant réellement des articles publiés depuis trois mois soient affichés. Il suffit pour cela d'ajouter une boucle vérifiant la présence d'article datés de moins de trois mois associés à ce mot-clé. Ce qui nous donne :

```
<BOUCLE_somm_mac(MOTS){type=Machines}{exclus}{par titre}>
<BOUCLE_verif_somm(ARTICLES){id_mot}{0,1}{age < 90}>
        <td>
        <a href="sommaire_machine.php3?id_mot=#ID_MOT"><img src='IMG/[(#LOGO_MOT|
fichier)]' border=0></a>
        </td>
</BOUCLE_verif_somm>
</BOUCLE_somm_mac>
```

On pensera à effectuer le même remplacement dans la page « sommaire.html » du sommaire général (penser à supprimer le critère {exclus} dans ce cas).

Référencer des articles sur le Web

Monter son site consacré aux jeux vidéo, c'est bien ; mais d'autres y ont pensé avant vous, et une tripotée de sites existe déjà. Pour enrichir encore l'information présentée sur notre site, nous allons donc indiquer des articles consacrés aux jeux que nous présentons.

Cela se fait, dans l'espace privé, par le lien « Référencer un site Web » accessible depuis chaque rubrique. Notre structure, qui regroupe tous les articles consacrés à un jeu dans une rubrique dédiée, facilite l'opération ; il suffit de « référencer un site Web » dans la rubrique qui traite du jeu concerné par le lien.

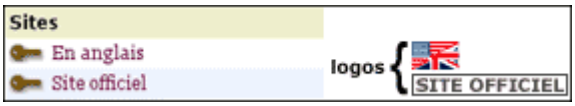
Tirons à nouveau profit de nos mots-clés. Pour chaque page référencée sur le Web, nous indiquerons :

- la machine concernée par cet article (la page référencée concerne une version Playstation, Xbox...) ;
- le type d'article (test, preview...) ; si la page n'entre dans aucune catégorie, nous n'indiquerons pas de type d'article ;
- éventuellement, la note attribuée au jeu sur ce site extérieur.

Afin d'enrichir encore l'information, nous souhaitons indiquer, le cas échéant :

- qu'il s'agit du site officiel du jeu ;
- que l'article est en anglais (vous pouvez prévoir d'autres langues si vous le souhaitez). En effet, certaines solutions de jeu circulent sur le Net en anglais bien avant d'apparaître en français.

Pour cela, créons un nouveau groupe de mots-clés, intitulé « Sites », et deux mots : « En anglais » et « Site officiel » (on pourra aussi créer « En japonais », « En italien »...). Nous attribuons un petit logo pour chacun de ces mots-clés (ce qui permettra de créer une interface plus compacte). Ce qui nous donne :



Reprenons notre fichier « rubrique.html ». Nous ajoutons la boucle suivante à la fin de la BOUCLE_les_articles :

```
<B_sites><p>SUR LE WEB :
  <ul>
  <BOUCLE_sites(SYNDICATION){id_rubrique}{doublons}>
   <li>[(#LOGO_SITE|#URL_SITE)] <a href="#URL_SITE">#NOM_SITE</a>

   <BOUCLE_note_sites(MOTS){type=Note}{id_syndic}>
   #TITRE/10
   </BOUCLE_note_sites>
   <BOUCLE_type_sites(MOTS){type=Sites}{id_syndic}>
   #LOGO_MOT
   </BOUCLE_type_sites>
   <BOUCLE_mac_sites(MOTS){type=Machines}{id_syndic}>
   #LOGO_MOT
   </BOUCLE_mac_sites>
  </BOUCLE_sites>
  </ul>
  </B_sites>
```

La BOUCLE_sites affiche le logo et le nom des sites référencés. Pour chaque site, les trois boucles incluses affichent successivement :

- BOUCLE_note_sites : la note attribuée au jeu par cet article ;

- BOUCLE_type_sites : s'il s'agit du site officiel, ou si l'article est en anglais ;
- BOUCLE_mac_sites : la machine concernée par le site.

Vous noterez le critère {doublons} dans la BOUCLE_sites. Il interdit d'afficher des site référencés qui auraient déjà été affichés. Pour l'instant, c'est inutile, puisque c'est le seul endroit de ce squelette où nous récupérons des sites référencés.

Complétons cependant notre processus. Puisque nous nous autorisons, si cela est pertinent, à indiquer pour chaque site référencé s'il s'agit d'un test, d'une preview, d'une soluce..., exploitons cette information. Nous allons intégrer, à la suite de nos propres tests, soluces... les liens vers les pages référencées de ce type.

Juste après la BOUCLE_tests, nous allons recopier la boucle précédente (en renommant les boucles), et légèrement la modifier, pour n'afficher que les sites référencés auxquels nous avons attribué le type « Test » :

```
<B_sites_tests><p>Des tests sur le Web :
  <ul>
<BOUCLE_sites_tests(SYNDICATION){id_rubrique}{titre_mot=Test}{doublons}>
   <li>[(#LOGO_SITE|#URL_SITE)] <a href="#URL_SITE">#NOM_SITE</a>

   <BOUCLE_note_sites_tests(MOTS){type=Note}{id_syndic}>
   #TITRE/10
   </BOUCLE_note_sites_tests>
   <BOUCLE_type_sites_tests(MOTS){type=Sites}{id_syndic}>
   #LOGO_MOT
   </BOUCLE_type_sites_tests>
   <BOUCLE_mac_sites_tests(MOTS){type=Machines}{id_syndic}>
   #LOGO_MOT
   </BOUCLE_mac_sites_tests>
  </BOUCLE_sites_tests>
  </ul>
  </B_sites_tests>
```

Nous reproduisons cela pour les previews, les soluces et les astuces.

Le critère {doublons} devient utile : chaque type de sites étant affiché successivement (d'abord les tests, puis les previews...), notre toute dernière boucle (BOUCLE_sites) n'affiche plus que les sites référencés qui ne sont d'aucun type. Par exemple, le site officiel d'un jeu, ou le site d'un fan du jeu, ne peut pas être décrit comme uniquement un test ou une solution ; ils seront donc affichés par la BOUCLE_sites.

Pour terminer l'exploitation des sites référencés, recopions ces nouvelles boucles dans « article.html » (où nous affichons déjà les autres articles concernant le même jeu).

Un forum pour chaque jeu

Notre site n'autorise pas, pour l'instant, les visiteurs du site à participer. Or ce genre de site se prête particulièrement bien aux forums ouverts aux visiteurs :

- d'abord parce que sur ce genre de sujet, les utilisateurs aiment à donner leur avis ; les forums de jeux vidéo sont très fournis...
- parce que c'est un élément indispensable au contenu éditorial du site : il est en effet impossible de prévoir à l'avance toutes les questions concernant un jeu (trucs, astuces, bloqué à tel niveau, comment reconfigurer la carte graphique RadeTI 2525 de son PC...) ; les forums sont le lieu indispensable pour ce genre de questions/réponses.

Avant d'intégrer des forums sur notre site, il faut d'abord définir où et comment les intégrer.

- **Première question : où ?**

Avec les squelettes fournis en standard avec SPIP, on a pris l'habitude d'installer les forums directement sous chaque *article*. Or, ici, notre structure éditoriale est totalement différente : tous les articles concernant un jeu sont regroupés dans une *rubrique*. Nous pourrions certes continuer à installer les forums sous chaque article, mais cela serait sans doute peu efficace : plus qu'une réaction à un article précis, le visiteur voudra certainement discuter du jeu « en général ».

Nous décidons (encore un choix éditorial arbitraire !) que les forums ne concerneront pas chaque article (chaque test, chaque preview, etc.), mais le jeu dans son ensemble. C'est-à-dire la rubrique du jeu. Dans chaque article, nous ferons un lien vers une page commune affichant le forum de la rubrique.

- **Deuxième question : comment ?**

La présentation du forum est importante : faut-il tout afficher d'un seul coup (c'est toujours très pratique), ou bien afficher d'abord une liste des titres des messages, et une page spécifique pour chaque message.

Sur un site de jeux vidéo, nous pouvons prévoir des forums très actifs, avec de nombreux messages, certains très longs (certains visiteurs vont certainement poster la critique complète d'un jeu, une soluce interminable...). Impossible dès lors d'afficher tout le contenu de tous les messages sur une seule page.

Nous choisissons donc de présenter d'abord la liste des messages, avec uniquement leur titre ; puis nous réaliserons une seconde page affichant chaque message spécifique.

La page forum_jeu.php3

La page qui affichera la liste complète des messages d'un forum sera donc la page « forum_jeu.php3 ». Cette page est appelée avec un numéro de rubrique (puisque c'est la rubrique qui contient tous les articles du jeu).

Commençons par intégrer le lien vers cette page dans la navigation de notre site.

Sur la page « article.html », ajoutons le lien vers le forum du jeu. Insérons (par exemple après l'affichage des #NOTES) :

```
<p><b><a href="forum_jeu.php3?id_rubrique=#ID_RUBRIQUE">Le forum de ce jeu</a></b>
```

Contentons-nous pour l'instant de ce seul lien de navigation. En effet, pour réaliser un affichage plus complet (nombre de messages notamment), nous avons besoin de créer des messages de forum afin de tester notre interface, et pour l'instant ces forums sont totalement vides.

Créons donc notre squelette des forums...

Commençons, comme d'habitude, par le fichier d'appel : « forum_jeu.php3 » :

```
<?
$fond = "forum_jeu";
$delais = 24 * 3600;

include ("inc-public.php3");
?>
```

Le squelette forum_jeu.html

Créons maintenant le fichier « forum_jeu.html ». Pour l'instant très simple :

```
<html>
<title>[#NOM_SITE_SPIP]
<BOUCLE_titre(RUBRIQUES){id_rubrique}>#TITRE</BOUCLE_titre></title>
</head>

<body>

<a href="index.php3">#NOM_SITE_SPIP</a>

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

 <BOUCLE_hierarchie(HIERARCHIE){" : "}>
 <a href="#URL_RUBRIQUE">#TITRE</a>
 </BOUCLE_hierarchie>

     <h1>FORUM: <a href="#URL_RUBRIQUE">#TITRE</a></h1>

</BOUCLE_principale>
</blockquote>
</body>
</html>
```

Ce premier squelette très simple affiche :

- le retour à la page d'accueil du site ;
- la hiérarchie menant à ce jeu ;
- le titre du jeu.

Grâce à la BOUCLE_principale, nous sommes bien dans la *rubrique* qui concerne ce jeu.

Pour l'instant, notre forum est totalement vide. Avant de créer l'affichage prévu (afficher uniquement la liste des messages), créons une interface permettant de créer un forum complet. Grâce à lui, nous pourrons « alimenter » notre forum, et ainsi créer l'interface ensuite.

Insérons le lien qui nous permettra de poster notre premier message :

```
<h3>[<a href="forum.php3?(#PARAMETRES_FORUM)">Poster un message</a>]</h3>
```

Automatiquement, #PARAMETRES_FORUM indiquera qu'il s'agit de messages liés à une rubrique, puisque notre BOUCLE_principale est bien de type (RUBRIQUES).

Affichons le titre de ces premiers messages :

```
<B_thread>
 <ul>
 <BOUCLE_thread(FORUMS){id_rubrique}{par date}{inverse}>
<li>#TITRE
  </BOUCLE_thread>
 </ul>
 </B_thread>
```

Ajoutons la possibilité de répondre à ces messages :

```
<B_thread>
 <ul>
 <BOUCLE_thread(FORUMS){id_rubrique}{par date}{inverse}>
 <p><li>#TITRE
 <p align="right"><a href="forum.php3?#PARAMETRES_FORUM">répondre à ce message</a>
 </BOUCLE_thread>
 </ul>
 </B_thread>
```

De cette façon, nous pouvons répondre aux messages de « premier » niveau (ce qu'on nomme les « *threads* », c'est-à-dire les messages qui lancent une nouvelle discussion), mais nous ne les affichons pas.

Affichons les réponses :

```
<B_thread>
 <ul>
 <BOUCLE_thread(FORUMS){id_rubrique}{par date}{inverse}>
 <p><li>#TITRE

 <p align="right"><a href="forum.php3?#PARAMETRES_FORUM">répondre à ce message</a>

 <B_reponses>
 <ul>
 <BOUCLE_reponses(FORUMS){id_parent}{par date}>
  <p><li>#TITRE
  <a href="forum.php3?#PARAMETRES_FORUM">répondre à ce message</a>
 </BOUCLE_reponses>
 </ul>
 </B_reponses>

 </BOUCLE_thread>
 </ul>
 </B_thread>
```

Nous affichons désormais les réponses aux threads, il est possible de répondre à ces réponses, mais sans affichage. Pour afficher ces réponses, nous allons désormais ajouter une boucle *récursive*, c'est-à-dire que la BOUCLE_reponses va s'appeler elle-même, ce qui nous permettra d'afficher d'un seul coup l'intégralité des messages.

Modifions la BOUCLE_reponses ainsi :

```
<B_reponses>
 <ul>
 <BOUCLE_reponses(FORUMS){id_parent}{par date}>
  <p><li>#TITRE
  <a href="forum.php3?#PARAMETRES_FORUM">répondre à ce message</a>

  <BOUCLE_rep_messages(boucle_reponses)></BOUCLE_rep_messages>


 </BOUCLE_reponses>
 </ul>
 </B_reponses>
```

Nous avons ajouté *à l'intérieur* de la BOUCLE_reponses une BOUCLE_rep_messages, qui appelle la BOUCLE_reponses. Ce qui permet d'afficher les réponses aux messages de la BOUCLE_reponses, de manière récursive (on exécute cette boucle jusqu'à ce qu'il n'y ait plus aucune réponse).

Arrivé à ce stade, pour les besoins de notre développement, il faut faire une pause, et consacrer un peu de temps à créer de nombreux messages dans un forum, avec plusieurs fils de discussion (*threads*), des réponses à des réponses... de façon à simuler un forum complet. Cela nous permettra de créer plus logiquement notre interface graphique.

On recommence...

Après cette étape, pendant laquelle nous avons alimenté notre forum avec une tripotée de messages, nous allons recommencer notre mise en page, cette fois dans l'optique de ce que nous avions prévu au départ : la page « forum_jeu » n'affichera que les titres des messages, et une autre page affichera chaque message.

Comme nos boucles précédentes n'affichaient pas d'informations, les modifications sont peu importantes :

- il faut afficher le #TITRE en lien hypertexte vers la page de ce message (nous décidons que nous utiliserons une page intitulée « message.php3 » pour cela) ;
- il faut afficher le #NOM de l'auteur du message, et la #DATE d'envoi du message ;
- il faut supprimer la mention « répondre à ce message ».

Nous en profitons pour modifier la présentation graphique. Les <ul> et <li> suffisent pour présenter la structure logique, mais graphiquement ça n'est pas très joli. Nous les remplaçons donc par quelques feuilles de style très simples.

Finalement, notre code devient :

```
<BOUCLE_thread(FORUMS){id_rubrique}{par
date}{inverse}> <p> <div style="border: solid black 1px;
background-color: #CCCCCC; padding: 3px; width:100%"> <a
href="message.php3?id_forum=#ID_FORUM">#TITRE</a> [| (#NOM)] |
[(#DATE|affdate)] </div> <B_reponses> <div
style="margin-left: 15px">
<BOUCLE_reponses(FORUMS){id_parent}{par date}> <div
style="border-left: solid black 1px; border-bottom: solid black 1px;
border-right: solid black 1px; padding: 3px; width:100%"> <a
href="message.php3?id_forum=#ID_FORUM">#TITRE</a> [| (#NOM)] |
[(#DATE|affdate)] </div>
<BOUCLE_rep_messages(boucle_reponses)></BOUCLE_rep_messages>
</BOUCLE_reponses> </div> </B_reponses>
</BOUCLE_thread>
```

Remarque sur les styles. Nous avons inséré ici les styles directement dans la déclaration des . On pourra préférer les regrouper dans des feuilles de style en début de fichier. Cependant, pendant le développement de la page, il est toujours pratique de pouvoir les modifier directement à l'endroit où se trouve l'information. Par ailleurs, l'usage des styles pose des difficultés de compatibilité avec Netscape 4.7 : non seulement celui-ci n'utilise pas l'intégralité des styles, mais en plus certains styles provoquent des bugs d'affichage. Dans l'exemple ci-dessus, nous avons choisi des styles qui ne provoquent pas ces bugs.

Notre page d'affichage de tous les messages d'un forum est terminée. Dans le cadre d'un véritable site, on gagnerait à afficher de plus les liens vers les articles de cette rubrique (les tests, les previews, un rappel des dates de sortie) ; comme cela a déjà été réalisé par ailleurs, nous laissons ce soin au lecteur.

Afficher chaque message : message.php/message.html

Il nous faut désormais afficher le texte de chaque message, avec la possibilité d'y répondre. Comme nous avons créé les liens sur la page précédente, nous savons que cela se fera avec un couple de fichiers : « message.php3 » et « message.html », appelés par un « id_forum » (id_forum étant le numéro de chaque message du forum).

Le fichier « message.php3 » :

```
<?
$fond = "message";
$delais = 24 * 3600;

include ("inc-public.php3");
?>
```

Le fichier « message.html » contient le squelette :

```
<html>
<title>[#NOM_SITE_SPIP]
<BOUCLE_titre(FORUMS){id_forum}>#TITRE</BOUCLE_titre></title>
</head>

<body>

<a href="index.php3">#NOM_SITE_SPIP</a>

<blockquote>
<BOUCLE_principale(FORUMS){id_forum}>

 <h3>#TITRE</h3>
 [(#DATE|affdate)][, par <A HREF="mailto:#EMAIL">(#NOM)</A>]
 [<BR><A HREF="#URL_SITE">(#NOM_SITE)</A>]

 <p>#TEXTE

</BOUCLE_principale>
</blockquote>
</body>
</html>
```

Cette première version affiche :

- le titre de la page (dans <title>...</title>) ;
- le lien vers la page d'accueil du site ;
- le titre, le texte, la date, l'auteur et un site Web de ce message.

Du côté de la navigation dans le site, tout reste à faire... Nous allons afficher la mention du jeu concerné

par ce message. Dans la BOUCLE_principale, juste avant le #TITRE du message, ajoutons l'indication du titre du jeu (avec un lien vers la rubrique du jeu) et un lien vers la page affichant tous les messages :

```
<BOUCLE_larubrique(RUBRIQUES){id_rubrique}>
  <h3>forum du jeu: <a href="#URL_RUBRIQUE">#TITRE</a></h3>
  afficher <a href="forum_jeu.php3?id_rubrique=#ID_RUBRIQUE">tous les messages</a>
 </BOUCLE_larubrique>
```

Nous recopions notre traditionnelle BOUCLE_hierarchie permettant d'indiquer la hiérarchie des styles de jeux :

```
<BOUCLE_larubrique(RUBRIQUES){id_rubrique}>
  <BOUCLE_hierarchie(HIERARCHIE){" : "}>
  <a href="#URL_RUBRIQUE">#TITRE</a>
  </BOUCLE_hierarchie>
  <h3>forum du jeu: <a href="#URL_RUBRIQUE">#TITRE</a></h3>
  afficher <a href="forum_jeu.php3?id_rubrique=#ID_RUBRIQUE">tous les messages</a>
 </BOUCLE_larubrique>
```

Nous désirons maintenant afficher la hiérarchie des messages qui mènent à notre message (s'il s'agit d'une réponse à un autre message, nous voulons afficher le lien vers ce message « parent »). Nous insérons (après le lien « afficher tous les messages », et avant le #TITRE du message actuel) :

```
<BOUCLE_parents(FORUMS){id_enfant}>
   <br><a href="message.php3?id_forum=#ID_FORUM">#TITRE</a>
  [| (#NOM)] | [(#DATE|affdate)]
 </BOUCLE_parents>
```

Note : le critère {id_enfant} appliqué aux boucles (FORUMS) a été introduit dans la version 1.3 de SPIP. Ce critère permet d'appeler le message auquel le message actuel répond. (Un critère similaire existe pour les boucles (RUBRIQUES), nous l'utilisons d'ailleurs dans « article.html » pour « remonter d'un cran » dans la hiérarchie des rubriques. Curieusement, nous avions oublié ce critère dans les boucles des forums...)

Nous affichons ici le message auquel le message de la BOUCLE_principale répond. Cela ne nous suffit pas, nous voulons afficher toute la succession des messages jusqu'à notre message principal (si le message affiché grâce à la BOUCLE_parents est lui-même une réponse, nous voulons encore remonter d'un cran, et ainsi de suite jusqu'au premier message du *thread*).

Pour cela nous utilisons tout simplement une boucle récursive, comme nous l'avons fait avec la BOUCLE_reponses de « forum_jeu.html ». Nous inversons cependant la logique, puisqu'au lieu de « descendre » la hiérarchie des messages, nous la « remontons ». Pour que la présentation soit cohérente, nous plaçons l'appel de la boucle récursive *avant* l'affichage du #TITRE du message (les message « parents » doivent s'afficher avant le titre de leur réponse) :

```
<BOUCLE_parents(FORUMS){id_enfant}>
  <BOUCLE_par_parents(boucle_parents)></BOUCLE_par_parents>
  <br><a href="message.php3?id_forum=#ID_FORUM">#TITRE</a>
  [| (#NOM)] | [(#DATE|affdate)]
 </BOUCLE_parents>
```

Cette boucle suffit à afficher la navigation et la structure logique dans les messages. Cependant, graphiquement cela ne nous convient pas. En effet, pour bien marquer le fait qu'il s'agit d'un enchaînement de réponses successives, nous voulons rétablir le petit décalage vers la droite à chaque réponse (comme sur la page « forum_jeu.html »).

Nous allons donc utiliser un style provoquant, à chaque appel de la boucle récursive, un décalage vers la gauche (de 15 points, puisque c'est le décalage utilisé dans la page « forum_jeu.html »). Nous en

profitons pour afficher la #TITRE du message dans la même style que sur cette page (liseret à gauche et en dessous) :

```
<B_parents>
<div style="margin-left:-15px">
<BOUCLE_parents(FORUMS){id_enfant}>
<BOUCLE_par_parents(boucle_parents)></BOUCLE_par_parents>
<div style="border-left: solid black 1px; border-bottom: solid black
1px; border-right: solid black 1px; padding: 3px; width:100%"> <a
href="message.php3?id_forum=#ID_FORUM">#TITRE</a> [| (#NOM)] |
[(#DATE|affdate)] </div> </BOUCLE_parents> </div>
</B_parents>
```

L'affichage devient plus cohérent, avec le décalage vers la gauche et les liserets, mais cela n'est pas encore parfait : ce décalage rentre dans la marge gauche de la page, puisque l'on décale vers la gauche, sans avoir auparavant décalé l'ensemble de la page vers la droite (si l'on considère une valeur de marge, on peut ainsi considérer que la marge gauche devient « négative » : au départ une marge égale à 0, puis -15, puis -30...).

Nous allons donc devoir décaler l'ensemble vers la droite. Nous ne pouvons pas le faire à l'intérieur de la BOUCLE_parents, puisque nous décalerions, à chaque récursivité, une fois vers la gauche, et immédiatement une fois vers la droite ; les décalages d'annuleraient à chaque fois.

Nous allons donc créer une nouvelle boucle, avant la boucle d'affichage des parents, dont le seul but sera d'afficher des destinés à provoquer l'affichage vers la droite.

L'ensemble de la page sera décalé vers la droite (s'il y a trois messages avant d'arriver au message principal, tout sera décalé de 45 points vers la droite). Les messages parents partiront donc de cette valeur pour se décaler vers la gauche : le premier message parent sera décalé vers la gauche, avec donc une marge global de 30 points vers la droite (45 - 15), le second sera à 15 points, puis 0 points.

Notre code devient :

```
<BOUCLE_decaler_droite(FORUMS){id_enfant}>
<div style="margin-left: 15px">
<BOUCLE_rec_decaler_droite(boucle_decaler_droite)></BOUCLE_rec_decaler_droite>
</BOUCLE_decaler_droite> <B_parents> <div
style="margin-left:-15px"> <BOUCLE_parents(FORUMS){id_enfant}>
<BOUCLE_par_parents(boucle_parents)></BOUCLE_par_parents>
<div style="border-left: solid black 1px; border-bottom: solid black
1px; border-right: solid black 1px; padding: 3px; background-color:
#CCCCCC; width:100%"> <a
href="message.php3?id_forum=#ID_FORUM">#TITRE</a> [| (#NOM)] |
[(#DATE|affdate)] </div> </BOUCLE_parents> </div>
</B_parents>
```

La BOUCLE_decaler_droite est encore une boucle récursive, similaire à celle destinée à afficher les messages parents. La différence est ici dans le contenu : à chaque message parent, au lieu d'afficher le titre du message, on affiche uniquement un provoquant le décalage vers la droite.

Puisque nous avons provoqué un décalage vers la droite, il faut ensuite réaliser l'opération inverse en fermant tous ces <div...>. Cela se fait par la même boucle, contenant cette fois les balises . Mais où placer cette boucle ? Il nous semble logique d'afficher le texte du message principal avec un décalage vers la droite : il apparaîtra ainsi très clairement comme une réponse au dernier message affiché par la

BOUCLE_parents.

C'est donc juste après la #TEXTE du message principal que nous plaçons le code suivant :

```
<BOUCLE_decaler_gauche(FORUMS){id_enfant}>
 </div>
  <BOUCLE_rec_decaler_gauche(boucle_decaler_gauche)></BOUCLE_rec_decaler_gauche>
 </BOUCLE_decaler_gauche>
```

Nous voulons maintenant afficher les réponses au message principal. De cette façon, nous pourrons nous seulement « remonter » la hiérarchie (avec la BOUCLE_parents), mais aussi la « descendre ».

Pour cela, il nous suffit de recopier la BOUCLE_reponses récursive de la page « forum_jeu.html ». Nous insérons ce code juste après le #TEXTE du message principal, et avant le décalage vers la gauche (il est logique que ces réponses soient décalées à droite du message principal) :

```
<B_reponses>
<div style="margin-left: 15px">
<BOUCLE_reponses(FORUMS){id_parent}{par date}> <div
style="border-left: solid black 1px; border-bottom: solid black 1px;
border-right: solid black 1px; padding: 3px; width:100%"> <a
href="message.php3?id_forum=#ID_FORUM">#TITRE</a> [| (#NOM)] |
[(#DATE|affdate)] </div>
<BOUCLE_rep_messages(boucle_reponses)></BOUCLE_rep_messages>
</BOUCLE_reponses> </div> </B_reponses>
```

Enfin il nous reste à afficher les réponses situées au « même niveau » hiérachique que le message principal (c'est-à-dire : si notre message principal est une réponse à un message, nous voulons afficher les liens vers les autres réponses à ce message).

Affichons toutes les autres réponses, en insérant ce code juste avant le #TITRE de notre message principal :

```
<BOUCLE_niveau(FORUMS){meme_parent}{par
date}{exclus}> <div style="border-left: solid black 1px;
border-bottom: solid black 1px; border-right: solid black 1px; padding:
3px; width:100%"> <a
href="message.php3?id_forum=#ID_FORUM">#TITRE</a> [| (#NOM)] |
[(#DATE|affdate)] </div> </BOUCLE_niveau>
```

Avec le décalage que nous avons créé précédemment, tout cela est du plus bel effet...

Améliorons encore notre présentation : nous allons afficher au dessus de notre message principal uniquement les messages postés *avant* ce message. Pour cela, nous ajoutons un critère à notre BOUCLE_niveau :

```
<BOUCLE_niveau(FORUMS){meme_parent}{par date}{age_relatif>0}{exclus}>
```

Le critère {age_relatif} (apparu dans SPIP 1.3) est similaire au critère habituel {age}. Mais, là où {age} compare la date de l'élément (article, message de forum, etc.) à la date actuelle (aujourd'hui), {age_relatif} compare cette date à la date de l'élément courant : dans notre cas, puisque nous sommes dans la BOUCLE_principale qui affiche le message principal, nous sélectionnions les autres messages en comparant leur date à la date du message principal. Avec {age_relatif>0}, nous récupérons donc uniquement les messages plus anciens que le message principal.

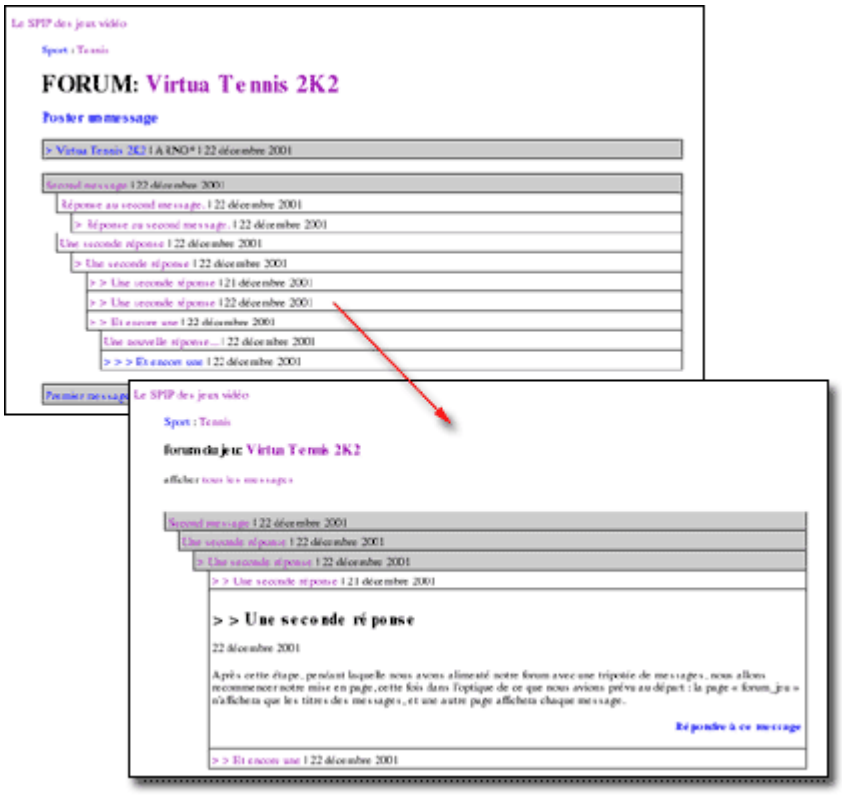
Fort logiquement, nous créons une boucle avec le critère d'age relatif opposé, après les messages de

BOUCLE_reponses.

```
<BOUCLE_niveau_apres(FORUMS){meme_parent}{par
date}{age_relatif<=0}{exclus}> <div style="border-left: solid
black 1px; border-bottom: solid black 1px; border-right: solid black
1px; padding: 3px; width:100%"> <a
href="message.php3?id_forum=#ID_FORUM">#TITRE</a> [| (#NOM)] |
[(#DATE|affdate)] </div> </BOUCLE_niveau_apres>
```

Il ne nous reste plus qu'à retravailler un peu la présentation du message principal (nous le plaçons lui aussi à l'intérieur d'un liseret noir), et à ajouter le lien « Répondre à ce message » :

```
<div
style="border-left: solid black 1px; border-bottom: solid black 1px;
border-right: solid black 1px; padding: 3px; width:100%">
<h2>#TITRE</h2> [(#DATE|affdate)][, par <A
HREF="mailto:#EMAIL">(#NOM)</A>] [<BR><A
HREF="#URL_SITE">(#NOM_SITE)</A>] <p>#TEXTE <p
align="right"><b>[<a
href="forum.php3?(#PARAMETRES_FORUM)">Répondre à ce
message</a>]</b> </div>
```



Nos deux squelettes permettant de gérer les forums d'un jeu sont terminés. Tout cela peut paraître un peu compliqué, mais il faut bien considérer que nous avons décidé de fignoler la présentation de la structure. Nous aurions pu nous contenter d'une interface de navigation beaucoup plus spartiate.

Les liens depuis les autres pages

Reprenons le référencement de ce forum dans les autres pages du site.

Dans la page « article.html », modifions notre lien et ajoutons une boucle :

```
<p><b><a href="forum_jeu.php3?id_rubrique=#ID_RUBRIQUE">Le forum de ce jeu</a></b>
 <BOUCLE_forum(FORUMS){id_rubrique}{plat}>
 </BOUCLE_forum>
 (#TOTAL_BOUCLE messages)</B_forum>
```

La BOUCLE_forum n'affiche rien, nous n'utiliserons que le *texte conditionnel après*. Le critère {plat} de cette boucle de forum indique que nous voulons récupérer *tous* les messages du forum (sans ce critère, nous ne récupérerions que les *threads* ; ici nous récupérons même les messages qui sont des réponses à d'autres messages). Cette boucle (qui n'affiche rien) effectuée, nous pouvons afficher en texte optionnel le nombre total de résultats, grâce à #TOTAL_BOUCLE, c'est-à-dire le nombre de messages dans le forum.

Dans « rubrique.html », cela devient un peu plus compliqué. En effet, de la même manière que nous avions établit une différence de présentation entre les rubriques de navigation (par grandes catégories de jeux) et les rubriques de jeux (contenant des articles), nous ne voulons afficher un lien vers un forum de discussion que s'il s'agit bien de la rubrique d'un jeu. Pour cela, nous créons une boucle qui va tester la présence d'au moins un article dans la rubrique, et notre lien vers le forum ne sera affiché que si cette boucle (BOUCLE_test_jeu) contient au moins un élément. Ce qui nous donne, inséré après la BOUCLE_sites et juste avant la fin de la BOUCLE_les_articles :

```
<BOUCLE_test_jeu(ARTICLES){id_rubrique}{0,1}>
  </BOUCLE_test_jeu>
   <p><b><a href="forum_jeu.php3?id_rubrique=#ID_RUBRIQUE">Le forum de ce jeu</a></b>
  </B_test_jeu>
```

Nous supprimons ici l'affichage du nombre de messages : nous sommes ici à l'intérieur du texte optionnel de la BOUCLE_test_jeu, et l'insertion d'une nouvelle boucle à cet endroit est toujours déconseillée (notamment, la valeur #TOTAL_BOUCLE serait celle de la boucle BOUCLE_test_jeu, et non plus le nombre de messages dans le forum).

Evidemment, on pourra compléter ce processus en ajoutant ces liens vers les forums sur d'autres pages. Par exemple le tableau des annonces des sorties de jeux.

Le site complet

Il y aurait encore beaucoup à faire pour enrichir la navigation sur notre site, uniquement en utilisant les boucles de SPIP. Mais ce tutorial est déjà bien assez long... nous en resterons donc là.

L'intégralité des fichiers réalisés dans ce tutorial sont disponibles sur le site spip_contrib. Ces fichiers contiennent quelques petits éléments de mise en page supplémentaires (des tableaux pour les sommaires), mais rien de fondamentalement complexe.

Si vous voulez bidouiller à partir des exemples fournis ici, n'hésitez pas, il reste de nombreuses possibilités de navigation inexploitées dans notre site :

- vous pouvez créer des pages de sommaire par type d'article : toutes les dernières previews, tous les derniers tests, toutes les soluces...
- vous pouvez vouloir créer des « événements », c'est-à-dire des articles que vous voulez mettre particulièrement en valeur sur la page d'accueil (ces articles n'étant finalement remplacés que par un « événement » suivant). Pour cela, vous pouvez créer un nouveau mot-clé, et effectuer des sélections supplémentaires (notamment sur les pages de sommaire) en fonction de ce mot-clé ;
- vous pouvez créer de nouveaux types d'articles (par exemple les « produits dérivés » tels que les figurines, les films adaptés des films...) ;
- vous pouvez créer de nouvelles grandes rubriques, destinées à recevoir un contenu éditorial différent, tel que des analyses générales sur les évolutions des machines (« faut-il encore acheter une Dreamcast ? », « la Xbox est-elle une bonne machine ? »...), des comparatifs... là encore l'utilisation des mots-clés permettra d'enrichir la structure éditoriale de votre site ; ils suffira de créer des squelettes localisés à ces nouvelles rubriques pour obtenir des interfaces adaptées à ces nouveaux besoins... il sera de plus intéressant d'intégrer l'affichage de ces nouveaux types d'articles dans votre sommaire, en exploitant ou non les mots-clés des machines (par exemple, un article « faut-il acheter une Dreamcast ? » serait affiché dans le sommaire spécifique de cette machine) ;
- sur la page d'un article, vous pouvez améliorer l'affichage des jeux de même genre (les autres jeux de survival horror, les autres jeux de plateforme) : vous pouvez limiter cet affichage qu'aux jeux ayant reçu une bonne note, ou sortis depuis moins de six mois ;
- vous pouvez réaliser une navigation de rubrique en rubrique en n'affichant que ce qui concerne une unique machine (attention, cela devient particulièrement complexe à réaliser).

Comme vous pouvez le constater, il y a beaucoup à faire avec les squelettes de SPIP. C'est même le point important à retenir de ce tutorial : *pour exploiter pleinement les possibilités de SPIP, il faut réaliser ses propres squelettes*. On peut dès lors considérer les squelettes fournis en standard avec SPIP comme un pis-aller : afin que chacun puisse commencer à exploiter immédiatement un site avec SPIP, nous fournissons des squelettes ; mais l'intérêt de SPIP, justement, réside dans la réalisation de ses propres squelettes, adaptés à la structure éditoriale dont on a besoin.

Les plugins pas à pas

Installer un plugin

Depuis SPIP 1.9, il est possible d'intégrer des plugins, qui rajoutent des fonctionnalités à SPIP.

Un *plugin* (également désigné par le nom de greffon) est un logiciel permettant d'améliorer ou d'ajouter des fonctionnalités dans SPIP. Il est écrit spécifiquement pour SPIP et respecte un formalisme (une API) qui leur permet d'interagir. Le plugin tire son nom de l'anglais *to plug* (brancher) car il doit être très facile de le *brancher* sur SPIP, mais aussi de le *débrancher*. L'existence des plugins répond principalement à la nécessité d'éviter l'hypertrophie du « noyau » de SPIP — notamment pour des raisons de maintenance — tout en facilitant grandement les possibilités de personnalisation poussée de son fonctionnement.

Récupérer et installer le plugin

Chaque plugin se présente sous la forme d'un dossier à son nom, contenant un ensemble de fichiers.

Installer ce dossier dans le répertoire ***/plugins*** à la racine du site. Ce répertoire n'existe pas par défaut lors de l'installation de SPIP. Vous devez donc le créer vous-même. Vous pouvez par ailleurs y ajouter des sous-répertoires pour classer les plugins que vous allez utiliser par familles.

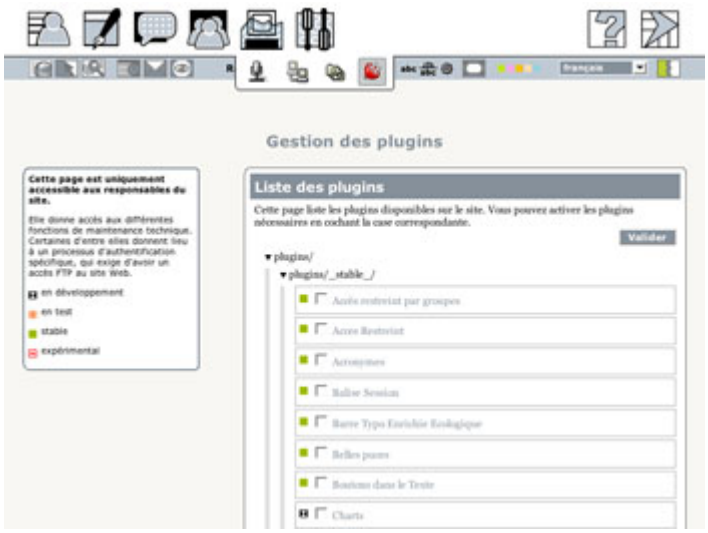
Activer le plugin

Dans l'espace privé, le sous menu « gestion des plugins » du menu « configuration » permet d'activer les plugins.

Pour accéder à ce menu, il est nécessaire :

- d'être administrateur
- d'être en interface complète

Si ce menu n'apparait pas c'est que le repertoire ***/plugins*** est mal installé.



Vous verrez apparaître la liste des plugins. Cochez ceux que vous souhaitez activer, décochez ceux que vous souhaitez désactiver. En cliquant sur la petite flèche à côté de chaque plugin, vous pouvez voir :

- la version et l'état de développement. Les différents états sont :
  - stable : le plugin a déjà été testé, et ne devrait pas poser des problèmes ;
  - test : le plugin est en phase de test. Il peut encore rester quelque bogues ;

        - expérimental : il s’agit d’un essai. Rien n’est donc assuré ;
        - dev : version de développement : à ne pas installer sur un site en production.
    - le ou les auteur(e)s
    - une description du plugin
    - un lien vers une page donnant plus d’information.

Avertissement

ATTENTION, comme pour tout logiciel, il est sérieusement recommandé d’être attentif à ne pas installer n’importe quoi sur votre serveur. Un plugin peut être à un stade de son développement qui n’autorise pas à l’utiliser sans risques. Un plugin présenté comme stable peut aussi contenir des erreurs et ouvrir de ce fait des failles de sécurité dans votre site.

Notez aussi qu’il est possible dans certains cas que deux plugins ne soient pas compatibles entre eux.

Si vous utilisez des plugins et constatez ce qui vous semble être un bug de SPIP, vous **devez impérativement vérifier** qu’il est reproductible sans aucun plugin activé avant de le signaler.
Si le défaut constaté disparait en l’absence de plugins, recherchez le plugin fautif en activant vos plugins un par un afin de préciser la configuration dans laquelle il est constaté. Faute de ces précautions, la résolution de votre problème par la communauté sera des plus aléatoire.

Pourquoi réaliser un plugin ?

Adapter SPIP à ses propres besoins avant la 1.9

Les squelettes, les balises, les boucles, et les filtres

SPIP est personalisable de bien des manières. Les premiers besoins de personnalisation sont vite comblés par la gestion des squelettes, fichiers html agrémentés de code SPIP qu'on stocke via un système de répertoires.

On peut également réaliser ses propres squelettes en s'appuyant sur les élements standards tels que les balises et les boucles, éléments que l'on peut créer et personnaliser à souhait.

Au besoin, on peut transformer l'affichage d'une balise à l'aide de filtres, fonctions codées en php dans un fichiers nommé mes_fonctions.php.

Cependant, il se peut aussi que nos besoins dépassent le cadre classique d'un site sous SPIP, comme publier des articles sur Internet sur un mode collaboratif entre rédacteurs. Il peut alors devenir nécessaire de transformer légèrement ou en profondeur le comportement de SPIP, voire d'ajouter de nouveaux éléments éditoriaux, de les associer et de les gérer à travers une interface personnalisée ou plus ergonomique.

Modifier le coeur de SPIP

Pour obtenir ces personnalisations, il était nécessaire d'entreprendre une démarche complexe de réécriture des fichiers du noyau, c'est-à-dire modifier le code distribué de SPIP. Certes c'est parfaitement légitime [1], mais alors suivre les évolutions de SPIP devenait un travail pénible et lourd.

Adapter SPIP à ses propres besoins après la 1.9

Depuis la version 1.9, SPIP met en place tout un formalisme pour permettre une modification et une extension harmonieuse du code. La principale avancée se situe dans la possibilité de "greffer" (et de retirer la greffe quand on le souhaite sans avoir touché le code de notre SPIP tout neuf) des améliorations et des modifications aussi diverses que variées. Les possibilités sont tellement vastes que toute la question est maintenant de savoir comment faciliter l'installation, tant pour soi-même que pour une distribution publique, de toutes ces personnalisations.

Techniquement, l'introduction des plugins a permis d'ouvrir les possibilités suivantes :

- tous les fichiers du noyau sont surchargeables [2] et toutes les fonctions appelables systématiquement [3],
- une interface d'application (API) est maintenue par la définition d'un certain nombre de points d'entrée dans le code.

On a besoin de réaliser un plugin dans quatre cas :

- **Fonctions et options** : réaliser son premier plugin, migrer et rendre portables ses fonctions et ses options pour soi ou pour les autres.
- **Les points d'entrée** : injecter, le temps d'activation d'un plugin, du code au coeur de SPIP et modifier profondément son fonctionnement.
- **Modifier les fichiers natifs** : en l'absence de point d'entrée, modifier des parties du code de SPIP sans intervenir concrétement sur le noyau.
- **Réecrire son propre code** : inventer son propre script que l'on greffera sur SPIP.

Si vous n'avez aucun de ces objectifs en tête, ne vous bilez pas avec les pages suivantes :) Cependant,

vous serez probablement intéresséE par ce que peuvent vous fournir des plugins distribuées librement.

Notes

[1] Rappelons que SPIP est en GPL : liberté sans restriction de l'utiliser, d'en étudier le fonctionnement et de le modifier pour ses propres usages, de le redistribuer, de redistribuer ses modifications.

[2] Surcharger un fichier consiste à multiplier la lecture d'un fichier afin de lui faire prendre la dernière valeur lue. De façon très shématique, si vous surcharger un fichier qui a pour valeur "Bonjour" avec un fichier qui a pour valeur "Bonsoir", l'écran affichera "Bonsoir" et non "Bonjour".

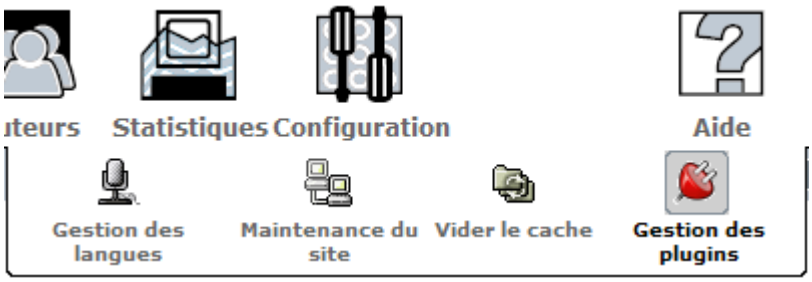
[3] Vous pouvez appeler à partir de n'importe quel fichier une fonction déjà existante, pour autant que vous ayez inclu en début de fichier le fichier qui contient la fonction que vous voulez utiliser. Ceci est particuliérement puissant, car les concepteurs de SPIP ont réalisé beaucoup de fonctions qui sont autant de routines que vous n'aurez pas à reprogrammer. On peut citer en exemple *lire_fichier* (qui lit un fichier), *ecrire_fichier* (qui écrit un fichier), *preg_files* (qui cherche un fichier), toutes les requetes sql et des dizaines de fonctions qui sont simplifiées au possible grâce à des routines contenues dans le code source de SPIP. Voir la doc du code pour en savoir plus.

Réaliser un premier plugin

On crée un plugin au même endroit qu'on installe ceux qu'on peut récupérer et déjà prêts à fonctionner (Voir « Installer un plugin »).

Le système de plugin s'appuie sur la recherche de chemins connus dans une liste bien précise. Les technophiles appelleront ça spip_path (cf. Où placer les fichiers de squelettes ?). L'activation, ou non, d'un plugin décide de l'ajout de son répertoire associé dans cette liste. On devra donc, pour réaliser son premier plugin, créer :

- Un répertoire plugins/ à la racine du site. La conséquence direct de cette action est d'activer l'interface de gestion des plugins. Le premier symptôme étant l'apparition du bouton « Gestion des plugins » dans le menu « Configuration » de l'interface privée (en mode interface complète uniquement)



- Un sous-répertoire mon_premier_plugin/ pour le plugin à réaliser.

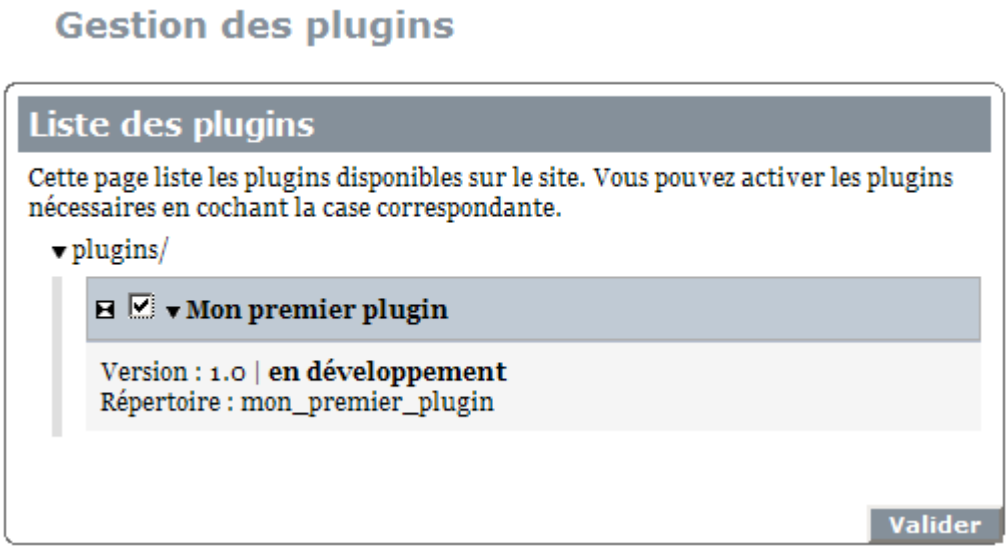
Pour indiquer à SPIP ce qu'est censé faire le plugin, celui-ci est décrit par un fichier, qui porte le nom plugin.xml, qu'on créé dans le répertoire même du plugin. Il s'agit d'un fichier à la syntaxe stricte mais relativement simple que voici :

```
<plugin>
 <nom>Mon premier plugin</nom>
 <version>1.0</version>
 <prefix>demo</prefix>
</plugin>
```

Il s'agît d'un fichier contenant les seules balises obligatoires.

- La balise nom peut être internationalisée grâce à la balise multi (exemple : <nom><multi>My first plugin[fr]Mon premier plugin</multi></nom>). C'est le titre du plugin.
- La balise version est purement informative. Il n'y a pas de règle précise pour gérer les numéros de version. Il appartient au développeur de les gérer lui-même. Dans l'avenir, il est toutefois possible que cette valeur soit utiles pour gérer d'éventuelles dépendances et un minimum de cohérence entre spip et les plugins...
- La balise prefix est cruciale. Il est tout d'abord important de s'assurer de l'unicité de cette valeur parmi tous les plugins que vous installez. Il définit ensuite le préfixe des fonctions que vous allez programmer en php et qui seront les éléments moteurs du plugin. Ainsi, si plusieurs plugins cohabitent avec des fonctions aux traitements similaires, il sera impossible qu'elles provoquent des erreurs à cause de leurs noms. C'est ce qu'on appelle aussi un « espace de noms ».

Bien que ne produisant aucun effet pour le moment, ce minimum permet déjà d'activer le plugin dans l'interface :



Pour ce faire, cochez la case en face du nom du plugin et cliquez sur le bouton « valider » en bas. La zone du titre est grisée.

Amusez-vous à retirer l'une ou l'autre de ses balises pour constater que SPIP peut vous indiquer ce qui manque ou tout simplement ne pas fonctionner.

_*_

Il est possible de placer d'autres éléments dans un fichier plugin.xml :

- La balise <etat> : 4 valeurs possible. En l'absence de cette balise, ou si elle est présente mais vide, le plugin est considéré comme "en développement". Le fonctionnement du plugin n'est aucunement influencé par la valeur de cette balise. À noter qu'une puce colorée permet de reconnaître l'état d'un coup d'œil.

etat	couleur de la puce	signification
dev	noir	En cours de développement. Nombreux bugs possibles. Ne fonctionne peut-être pas.
test	orange	La fonction principale est opérationnelle et est à essayer. Les développeurs comptent sur nous pour remonter des commentaires sur d'éventuelles erreurs ou des remarques sur l'ergonomie d'une interface graphique, par exemple.
stable	vert	Complètement opérationnel. Sans bug, en théorie.
experimental	rouge	C'est l'aventure !

- Les balises <auteur> et <description> sont facultatives et fonctionne comme les textes d'un article (tous raccourcis autorisés) et servent à fournir des informations succintes sur le plugin. Ainsi le code :

```
<plugin>
<nom>Mon premier plugin</nom>
<version>1.0</version> <prefix>demo</prefix>
<etat>dev</etat> <auteur>Jean Dupont
[contact->mailto:jdupont@monsite.net] _ [mon
site->http://www.monsite.net]</auteur> <description>Ce
plugin est une d&eacute;monstration. Il est distribu&eacute;
sous licence GNU/GPL</description>
</plugin>
```

aura pour conséquence d'afficher :

Notez qu'il est nécessaire de coder les caractères accentués avec des entités html. (é pour é, par exemple)

_*_

Si vous placez au même niveau que plugin.xml un fichier squelette.html, votre plugin une fois activé, permet l'utilisation dudit squelette qui sera utilisable en l'appelant dans votre navigateur avec la notation « usuelle » : votresite.net/spip.php?page=squelette. Il s'agît là d'un simpliste mais premier exemple de surcharge de fichier. Ce principe sera approfondi dans un autre tutoriel.

_*_

Pour illustrer cette démonstration, nous allons réaliser un plugin dont le but est de colorer d'une manière particulière, chaque occurance du mot « spip » dans un texte. Nous allons pour cela, placer dans le répertoire du plugin les 2 fichiers utiles et informer spip de leurs noms via quelques balises supplémentaires dans le fichier xml :

```
<plugin>
<nom>Mon dernier plugin</nom>
<version>1.0</version> <prefix>demo</prefix>
<etat>experimental</etat> <auteur>Jean Dupont
[contact->mailto:jdupont@monsite.net] _ [mon
site->http://www.monsite.net]</auteur> <description>Ce
plugin est une d&eacute;monstration. Il est distribu&eacute;
sous licence GNU/GPL</description>
<fonctions>exemple_fonctions.php</fonctions>
<options>exemple_options.php</options>
</plugin>
```

Voici la description des 2 balises ajoutées

- <fonctions> : contient le nom d'un fichier qui sera chargé à chaque recalcul. C'est l'équivalent pour chaque plugin, du fichier mes_fonctions.php. Il n'est donc utilisé que pour le site public, puisque qu'il n'est appelé qu'en cas de calcul du cache. On y mettra typiquement des filtres ou des définition de balises, de critères.
- <options> : contient le nom d'un fichier qui sera chargé à chaque appel de page . C'est l'équivalent pour chaque plugin, du fichier mes_options.php. Il est aussi utilisé dans l'interface privée à chaque appel de page.

Créez les 2 fichiers avec le contenu ci-dessous :

exemple_fonctions.php :

```
<?php

function colore_spip($texte) {
 global $couleur;
 return preg_replace('/([^(class=")])(spip)/i',
  '$1<span style="color: '.$couleur.';">$2</span>',
  $texte);
}

?>
```

exemple_options.php :

```
<?php

$couleur = '#ff017d';

?>
```

Attention ! Il y a des noms de fichier qu'il est préférable ne pas utiliser dans le cas des plugins : nous sommes dans la cadre d'une liste de répertoires connus et spip s'arrête toujours au premier fichier trouvé. Si le fichier de la balise <fonctions>, s'appelle mes_fonctions.php, il y aura confusion avec le fichier de votre dossier squelettes, s'il existe et surtout, si plusieurs plugins sont programmés avec ce nom de fichiers, le système rique de se perdre... idem pour mes_options.php, qu'il faut réservé au répertoire ecrire/ ainsi que pour les fichiers de langues local_xx.php auxquels on préfèrera une autre méthode décrite dans un tutoriel consacré à la surcharge de code.

Donc : **évitez les noms « communs » tels que** mes_fonctions.php **et** mes_options.php.

Nous verrons dans un autre article comment surcharger les répertoires de spip. Sachez donc que des noms de répertoires tels que « modeles », « formulaires », « inc », « action » etc... sont à utiliser à bon escient.

Enfin, prenez garde à être très exact dans la saisie des noms de fichier dans plugin.xml, l'interface de

gestion est très capricieuse pour l'ajout des dernières balises (messages d'erreur qui plantent le serveur...)

Bref, voilà, ce plugin apporte un nouveau filtre pour vos squelettes. Essayez [(#TEXTE|colore_spip)] dans article.html par exemple.

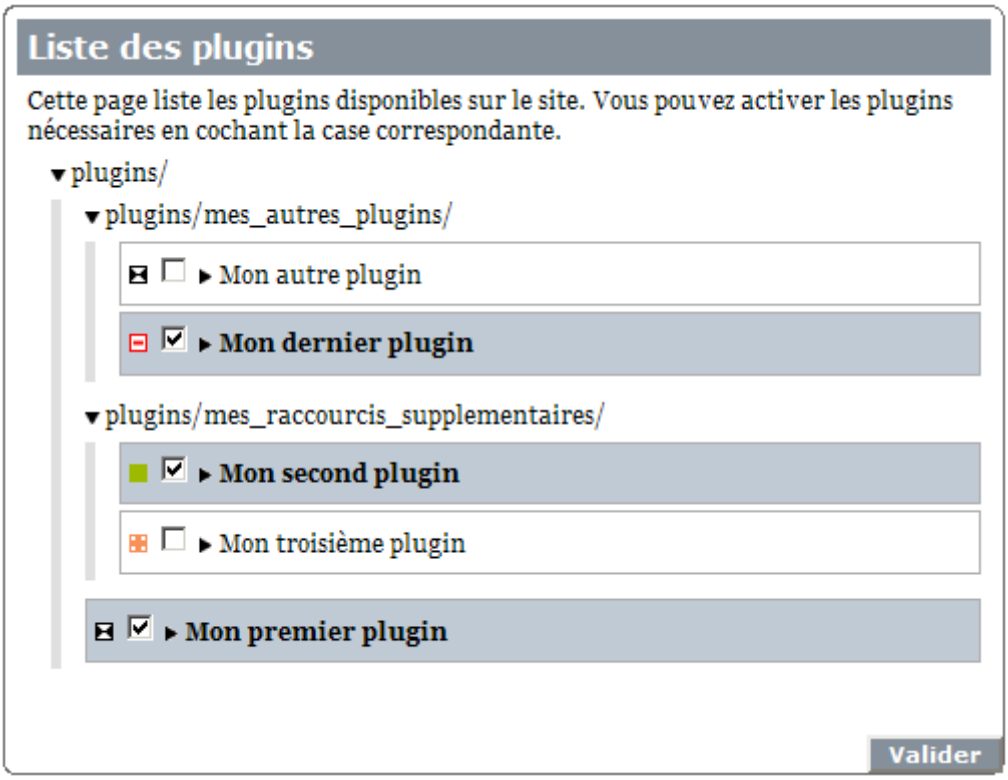
_*_

On installe tous les plugins dans le répertoire plugins/, mais on peut créer des sous-répertoires pour catégoriser les plugins qu'on installe ou développe.

Ainsi, pour l'arborescence suivante :

```
plugins/
  |
  |-- mon_premier_plugin/
  |-- mes_raccourcis_supplementaires/
        |
        |-- raccourcis_2/
        |-- raccourcis_3/
  |-- mes_autres_plugins/
        |
        |-- raccourcis_4/
        |-- raccourcis_5/
```

l'interface de gestion aura l'allure suivante :



Aller plus loin

Une fois lancéE sur cette voie, des interrogations techniques, liées à des problématiques de programmation en php, par exemple, se posent, c'est légitime.

Le site de documentation technique est conçu pour répondre à ces questions. Concernant les plugins, vous pouvez poursuivre à cette adresse pour une première approche sur les points d'entrée de spip (ou « pipeline »).