
WEBMASTERS
Les secrets des techniques de référencement
Réussir le référencement sur Google™ de votre site
Micro Application

1ère Édition - Février 2009

**Auteur** - Gilles GREGOIRE



ISBN : 978-2-300-016134

Couverture réalisée par Sébastien Wiegant

MICRO APPLICATION
20-22, rue des Petits-Hôtels
75010 PARIS
Tél. : 01 53 34 20 20
Fax : 01 53 34 20 00

Support technique :
Également disponible sur www.microapp.com
http://www.microapp.com

# Avant-propos

La collection *Webmasters* s'adresse aux personnes initiées au développement de sites web qui souhaitent découvrir et mettre en pratique les nouvelles technologies Internet. Sans négliger les aspects théoriques, nous donnons toujours priorité à la pratique afin que vous puissiez rapidement être autonome.

À travers les différents titres de cette collection vous découvrirez les technologies qui font le web 2.0 et feront ce que certains nomment déjà le web 3.0.

## Conventions typographiques

Afin de faciliter la compréhension des techniques décrites, nous avons adopté les conventions typographiques suivantes :

- **gras** : menu, commande, boîte de dialogue, bouton, onglet.
- *italique* : zone de texte, liste déroulante, case à cocher, bouton radio.
- `Police bâton` : instruction, listing, texte à saisir.
- ➥ : dans les scripts, indique un retour à la ligne volontaire dû aux contraintes de la mise en page.

Il s'agit d'informations complémentaires relatives au sujet traité. Propose conseils et trucs pratiques.

Mise en garde sur un point important à ne pas négliger.

# Sommaire

# Utiliser ce livre à 100 %

Ce livre, dédié au référencement et au positionnement dans les moteurs de recherche et plus particulièrement dans Google, est conçu pour vous apporter des connaissances, des exemples sous différentes formes dont du code source. La complexité va croissante au fil des chapitres, approche pédagogique classique pour permettre au plus grand nombre d'acquérir un maximum de connaissances.

La programmation de sites web dotés d'un référencement automatisé de bonne qualité va entraîner le lecteur dans une situation conflictuelle. En effet, cette programmation offre une palette assez intéressante dans la carrière d'un codeur : programmer pour répondre aux exigences des moteurs de recherche dont Google, le leader actuel, est très souvent contradictoire avec les besoins et exigences de tous les autres intervenants dans la conception, puis la production et enfin la maintenance du site.

Les besoins de mise en page du référenceur (donc les algorithmes de mise en page automatique que vous allez coder) seront plus ou moins en conflit avec les besoins de mise en page en charge de séduire les internautes.

Les besoins de navigation et d'ergonomie du site, pour les internautes, sont souvent contradictoires avec les besoins du référenceur. Il existe donc des conflits dans les demandes faites aux développeurs pour coder les programmes.

Il faudra donc identifier des compromis et les coder en conséquence. Il y a une infinité de cas particuliers. Il est bien sûr impossible de tous les traiter dans ce livre.

Néanmoins, pour répondre à un maximum de cas, nous aborderons les différentes problématiques une par une, avec les explications exposant le contexte. Dans l'exemple de site web, nous traiterons également des cas classiques de conflits entre les intervenants et le codeur/référenceur.

Le but de ce livre est de vous "apprendre à pêcher" et non de vous servir des "poissons prêts à l'emploi". Vous aurez à adapter le contenu de ce livre à vos propres besoins.

Dernier point, cet ouvrage est consacré au référencement et non aux subtilités de la programmation PHP/MySQL. Charge à chaque codeur d'adapter ses Framework, méthodes, habitudes, savoir-faire aux exemples de codes sources utilisés. L'accent de ces codes sources est mis sur la lisibilité et non sur la mise en œuvre de génie logiciel ou d'optimisation dans la gestion des ressources processeur, mémoire ou disques durs.

## PHP, MySQL, XHTML, etc.

Les lecteurs de ce livre ont typiquement la charge de développer, de coder des sites Internet en faisant appel à diverses technologies complémentaires. Ces différents modules collaborent pour permettre le fonctionnement d'un site web dynamique s'adressant à de multiples internautes simultanément :

- système d'exploitation : Linux par exemple ;
- moteur http, par exemple : Apache ;
- langage de codage des pages, par exemple XHTML/CSS ;
- langage de script côté navigateur, JavaScript par exemple ;
- langage de script côté serveur : PHP5 ;
- base de données relationnelles : MySQL par exemple.

Il existe de nombreux autres logiciels capables de réaliser les tâches de ces différents modules : Microsoft Windows Server, Microsoft IIS, ASP, VBScript, etc. On peut même coder des sites Internet dynamiques sur des Mainframes IBM.

Logiciels libres

Dans le présent ouvrage, nous ciblerons le plus grand nombre, ceux qui développent des sites avec des technologies logiciels libres comme Linux, Apache, PHP…

Néanmoins, tous les principes et codes sources évoqués dans ce livre avec XHTML/CSS, PHP/MySQL, Apache sont utilisables avec des technologies propriétaires comme celles de Microsoft ou d'autres.

- Apache vs. IIS ;
- PHP vs. ASP ou .NET ;
- MySQL vs. SQL Server.

> ⚠ Microsoft et autres environnements propriétaires
>
> Les algorithmes indiqués, les règles de fonctionnement et de comportement sont valides et aisés à transposer du logiciel libre vers le monde Microsoft ou vers d'autres environnements propriétaires.


► Fig. 1 : Page d'accueil du site exemple utilisé dans ce livre

## Erreur méthodologique majeure à éviter

### Un site web visible via Google

Un site Internet peu visible n'est guère utile à son possesseur. Il est crucial, pour la majorité des sites web, d'être visibles sur leurs mots-clés dans les **SERP**, *Search Engines Pages Results*, les pages de résultats renvoyées par Google ou autre moteur de recherche à tout internaute envoyant une requête sur ledit moteur.

Quand un codeur reçoit un cahier des charges pour analyser puis coder un site Internet, ce dernier a déjà été analysé fonctionnellement puis conçu par le marketing et les chefs de projets.

Dans l'esprit du "client" qui va déployer et profiter du trafic sur le site, le site devra être indexé et bien positionné dans Google dès que possible.

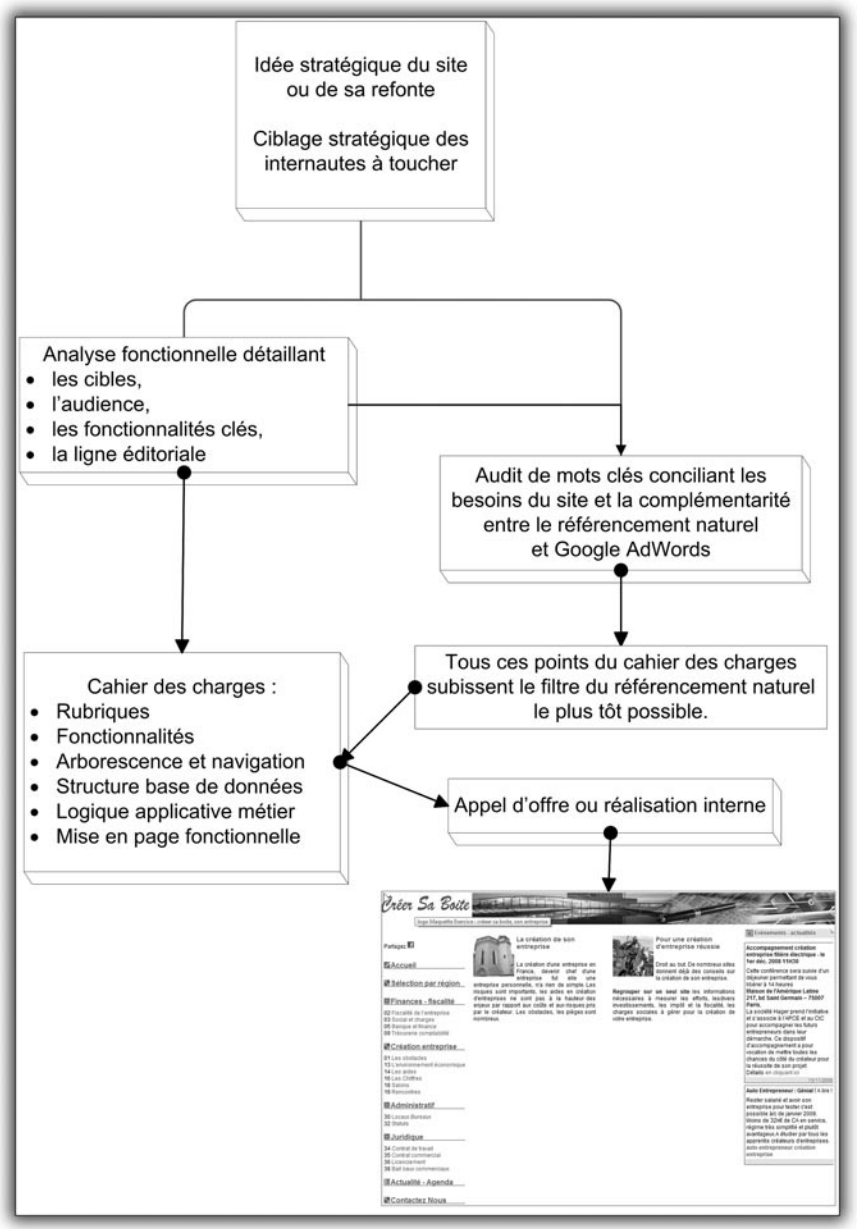
### Le référencement, dernière étape ?

En général, dans l'esprit de nombreux acteurs du marché, on traite le référencement en dernier comme une simple action technique. Rien n'est plus faux. Pour améliorer la qualité du référencement et diminuer délais et coûts en tout genre, il est fondamental d'intégrer le référencement dans les problématiques d'analyse et de conception fonctionnelle du site.

Le présent ouvrage aidera le codeur à pallier partiellement ce type d'erreur méthodologique.

Dans l'objectif d'une démarche qualité, le marketing ou le service en charge du dossier doivent être sensibilisés aux contraintes du référencement et la démarche incluse dès la conception du projet. Réfléchir au référencement et au positionnement d'un site en dernière étape est une erreur méthodologique majeure qui nuira au succès de cette tâche. Le nombre de mots et expressions clés en français ou dans d'autres langues est forcément limité. Plus il y a de sites web en compétition sur une expression clé donnée, plus l'obtention d'un bon rang, d'un bon positionnement dans les SERP de Google devient difficile et exige un travail de qualité. Ce sont les meilleurs qui gagnent.

C'est la meilleure équipe qui gagne. Web marketeur, codeur, chef de projet constituent une équipe qui doit être coordonnée pour être efficiente.




► Fig. 2 : Organisation projet web intégrant le référencement sur Google en amont

## Enjeu stratégique : automatiser le référencement

La plupart des acteurs du web appellent et fusionnent, improprement, le référencement et le positionnement dans les premières pages de résultats de Google : "référencement sur Google".

Il existe bien deux étapes distinctes, faisant appel à des savoir-faire différents :

- Le référencement regroupe les actions techniques sur le site.
- Le positionnement regroupe d'une part les actions web marketing, donc aucunement techniques sur le site et d'autre part la gestion des `tags clouds`, des `nuages de liens` utilisés pour la navigation dans le site ou vers l'extérieur, donc une action technique interne au site.

Nous y reviendrons en détail plus loin.

Les actions techniques sur le site ont toutes un point commun : la répétition d'actions techniques basiques et élémentaires sans oubli ni erreur. La conséquence est évidente : référencer techniquement un site est long et fastidieux pour un être humain. En d'autres termes, un référencement strictement manuel coûtera cher et pourra, malgré tout, contenir des erreurs préjudiciables à la qualité et aux performances.

> ⚠ Meilleure qualité et un coût diminué
>
> En automatisant les multiples tâches fastidieuses du référencement technique au sein du site, on améliorera fortement la qualité du référencement et du positionnement tout en diminuant nettement les coûts.

Il n'y a pas que cet aspect qualité et optimisation des investissements. Un autre aspect stratégique, capable de conférer un avantage compétitif au possesseur du site, intervient dans l'automatisation du référencement d'un site : piloter et adapter la stratégie de référencement depuis une simple table SQL.

> ⚠ Piloter la stratégie de référencement depuis une table
>
> Nous verrons dans ce livre qu'il existe un zone commune à toutes les actions techniques de référencement : les mots-clés. Il faut les disposer, les baliser, les ancrer, les pointer, les lier, etc. Mais on manipule toujours des mots- clés. Administrer la répartition des mots-clés sur les pages d'un site Internet, depuis une simple table MySQL, simplifie et automatise le référencement tout en le rendant plus efficace, en travaillant en "temps réel" pour un coût négligeable. Cela confère un avantage compétitif important.

## Organisation de cet ouvrage

Inutile de rentrer directement dans le codage des codes sources avant d'avoir compris :

- les fondamentaux des moteurs de recherche et de Google en particulier ;
- les problématiques de référencement et autres cadratures de cercles à résoudre ;
- les multiples pièges du référencement.

Les premiers chapitres exposeront, en vagues successives, les principes et le pourquoi du référencement. On pourra ensuite passer à l'algorithmique à mettre en œuvre plus facilement et illustrer ces approches avec des exemples simples de codes sources en PHP/MySQL/XHTML/CSS.

## Objectifs et enjeux de ce livre

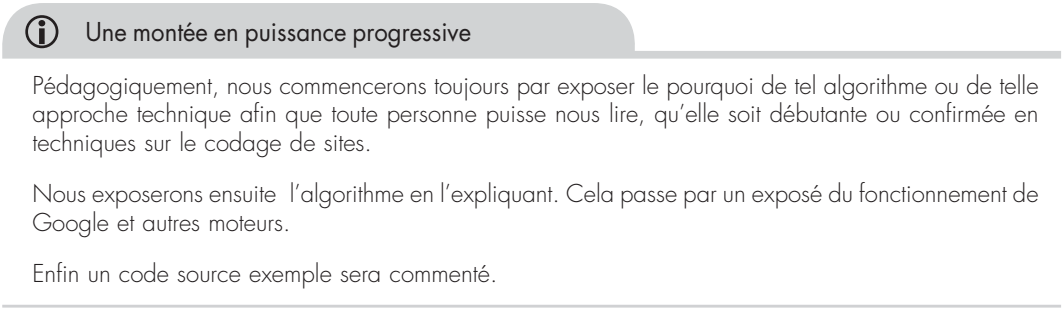
Les enjeux du codage correct en référencement d'un site web portent sur plusieurs points :

- Un site accessible techniquement sera pris en compte par Google. Autrement, il sera ignoré. Coder sans bloquer les bots de Google ou sans déclencher des pénalités infligées par Google suite à des erreurs techniques sur le site est un des devoirs du codeur.
- Un site accessible sémantiquement sera "compris" par Google. Le choix des mots-clés et autres réflexions web marketing n'est pas de son ressort, bien sûr. Mais la partie codage pour mettre en œuvre efficacement les décisions web marketing est du ressort du codeur. Le webmestre aura en charge, quant à lui, la conception de la mise en page adéquate et laissera le codeur l'automatiser.
- Un site codé de manière à pouvoir aisément et efficacement manipuler les bots de Google et les algorithmes d'indexation de Google offre plusieurs avantages. Le site web sera davantage réactif aux instructions du *traffic manager* en charge de booster l'audience : en peu de temps, de nouveaux mots-clés sont ajoutés, retirés, modifiés sur la globalité du site tout en préservant la cohérence. Face à un tel site, Google réagit positivement via un meilleur positionnement aussi rapide que le permettent les freins internes à Google contre une ascension trop rapide dans un index sur un mot-clé précis.
- Un codeur, un chef de projet, un webmestre, un traffic manager maîtrisant les concepts exposés dans les lignes précédentes aura une bien meilleure employabilité

sur le marché du travail, un meilleur salaire, de meilleur bonus et primes qu'un codeur ne se préoccupant pas de la qualité de son code et de ses méthodes face aux exigences de Google. Un site mal codé gênera le positionnement final dans l'index de Google ; il aura donc une influence négative sur les performances commerciales du site.

## Pédagogie et public ciblé par ce livre

Cet ouvrage cible tous les techniciens, du chef de projet au webmestre en passant par les référenceurs, en charge d'analyser, coder, déployer un site Internet et d'en assurer la maintenance.




Comprendre le pourquoi d'un algorithme ou d'une approche implique d'exposer, de manière détaillée, le fonctionnement de Google vu par un référenceur et donc les attentes à respecter pour être "compris" par le moteur et mieux positionné.

- Le **chef de projet** identifiera les questions et approches à mettre en place dans son protocole d'analyse fonctionnelle et dans les échanges avec le client pour rédiger un cahier des charges prenant en compte, dès le début, le référencement et le positionnement à obtenir pour le site.
- Le **webmestre**, débutant ou confirmé, désireux de hisser les pages de son site Internet sur des mots-clés précis dans l'index de Google, comprendra mieux les interactions des différentes optimisations. Le webmestre pourra planifier et réaliser de multiples actions techniques plus cohérentes, mieux codées, plus efficaces plutôt que de travailler sans comprendre toujours ce qu'il fait ou en consacrant son temps précieux à des tâches répétitives et fastidieuses.
- Le **Traffic Manager** désireux de mieux appréhender Google et le référencement naturel comprendra davantage les problématiques techniques que les codeurs et chefs

de projets doivent gérer. Il pilotera plus efficacement ses budgets et ses campagnes de web marketing tout support.

- Le **codeur** informatique ayant en charge de développer ou coder l'automatisation du référencement d'un site Internet trouvera des explications et des exemples en codes sources.

# 1

# Optimiser des pages pour Google

Ce chapitre a pour objectif d'exposer les concepts du référencement naturel et du positionnement. Ces deux approches complémentaires ont pour but de faire apparaître une page précise, la *landing page* ou page d'atterrissage, dans une page de résultats de Google sur une expression clé donnée.

## 1.1 Des concepts et termes techniques à maîtriser

Comme nous l'avons déjà écrit dans la préface, il est inutile de rentrer dans un code source sans connaître et comprendre les raisons des algorithmes à coder.

⚠ Comprendre, connaître les technologies du Web

Les lecteurs du présent ouvrage sont de profil technique ou possédant un solide "vernis" technique. La pédagogie des explications suivra ce principe. Il n'y aura pas d'explication sur les mécanismes de base des technologies du Web. Le lecteur est censé connaître un minimum de HTML et de XHTML, des feuilles de style CSS et du codage en PHP ou en JavaScript.

Nous allons aborder les concepts du référencement et les illustrer au fil des paragraphes puis préciser les parties qui concernent particulièrement le codeur.

Voici quelques concepts ou définitions à comprendre afin de vous permettre de mieux lire et utiliser ce livre.

### Mots, expressions clés, requête, SERP

Une *expression clé* est composée de *mots-clés*. Une *expression clé* est utilisée par les internautes pour rechercher un élément plus ou moins précis via une *requête* saisie dans un moteur de recherche. Exemple : l'expression clé `"TV plasma FULL HD"` est composée de quatre mots-clés.

Un moteur de recherche renvoie les résultats d'une requête vers l'internaute sous la forme d'une suite de 1 à 100 pages maximum avec 10 résultats par page. On appelle ces pages des *SERP* (*Search Engine Results Page(s)*), acronyme anglais signifiant : page(s) des résultats retournée(s) par le moteur de recherche à l'internaute lui ayant soumis une requête.

### Google et les landing pages

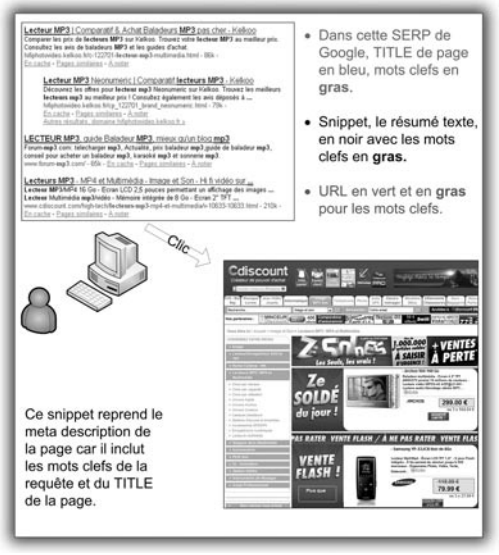
Google référence des pages HTML et non des sites Internet. Nous verrons que de nombreuses conséquences en découlent et que cela impacte fortement la structure du site, la stratégie de référencement globale du site et le travail d'optimisation de chaque page.

Une première conséquence du référencement de pages HTML et non d'un site va influencer le travail du codeur de site. Le web marketing ou le référenceur exposera la nécessité de prévoir des pages d'atterrissage (*landing pages*) pour les expressions clés stratégiques du site.

Quelques explications s'imposent pour que le codeur comprenne ce que l'on va attendre de lui. On peut classer les expressions clés d'un site en trois grands groupes :

- *Stratégiques* : elles sont à l'origine des "meilleurs" prospects et clients, d'un grand pourcentage du trafic du site, etc.
- *Importantes* : elles contribuent fortement au trafic et au "business" du site.
- *Longues traînes* : les déclinaisons multiples de ces expressions clés génèrent une longue traîne : de multiples petits ruisseaux d'internautes arrivent sur le site via une infinité de combinaisons de mots-clés. Et les petits ruisseaux, à force d'être assemblés, forment des rivières intéressantes en volume de trafic.

Il ne suffit pas de bien positionner la page d'accueil du site sur une expression clé dans l'index de Google. Il faut que l'internaute ait envie de cliquer sur le résumé présenté dans la SERP de Google. Et s'il clique sur le lien depuis la SERP de Google, il faut qu'il soit intéressé par ce qu'il voit dans la page du site qui s'affiche. Autrement, il rebondira en zappant sans même lire les informations de la page qui lui sont destinées.




► Fig. 1.1 :
De la SERP Google via le snippet incitatif au clic jusqu'à la landing page.

Pour améliorer le taux de clics dans la SERP, il faut concevoir la page afin de maîtriser ce que Google affichera. S'il y a trop de mots ou d'expressions clés à gérer pour une page, cas classique de la page d'accueil du site, il sera impossible de réaliser correctement ce travail. Il faut donc spécialiser la page sur quelques mots-clés.

⚠ 3 à 7 mots-clés maximum à référencer sur une page

Nous verrons plus loin pourquoi, mais il est très difficile de dépasser la limite de 5 à 7 mots-clés à référencer par page. De plus, arriver à prévoir le résumé de texte que Google affichera dans la SERP devient d'une complexité exponentielle, à mesure que l'on ajoute des mots-clés à prendre en compte dans une page.

Une autre raison justifie la mise en place d'une landing page spécialisée sur quelques mots-clés. L'internaute, séduit par le résumé de la page affiché dans la SERP de Google, clique et visualise alors la page. Si cette page ne le séduit pas en 2 secondes environ, il rebondit (zapper) hors du site. Il est très difficile de concevoir une page d'accueil capable de séduire des internautes très divers requêtant sur de multiples mots-clés. Il est plus facile de concevoir une telle page si elle est spécialisée pour chaque expression clé stratégique.

Le résumé de la page affiché par Google dans une SERP est composé de 2 parties distinctes :

- le *title* de la page, limité à une dizaine de mots ;
- de un à deux extraits de quelques mots, chacun contenant la première occurrence de chaque mot-clé composant l'expression clé saisie en requête.

Ce résumé est très important pour le web marketing. Il conditionne le taux de clics sur le résumé de la page pour emmener l'internaute intéressé vers la page elle-même.

Réussir à caser plus de 5 mots-clés en conservant une lisibilité correcte dans un title limité à 10 mots maximum relève de l'exploit difficile à réitérer sur toutes les pages du site. Il est donc déraisonnable de tenter de dépasser cette limite sur les landing pages.

⚠ Perfectionner le title et les mots-clés

L'internaute va lire dans la SERP de Google un résumé de la page proposée. Ce résumé est composé du title et d'un extrait de la page contenant les mots-clés de sa requête. Soigner particulièrement cette rédaction dans chaque landing page augmentera le taux de clics depuis les SERP de Google. Celui-ci mesure ces clics vis-à-vis des mots-clés de la requête. Plus il y a de clics, plus il a tendance à hisser la page en positionnement sur ces mots-clés.

Il est concevable, pour un codeur, de créer un *backoffice* de site Internet en charge d'automatiser la gestion de la partie répétitive de ces problématiques.

## Architecture très simplifiée d'un moteur

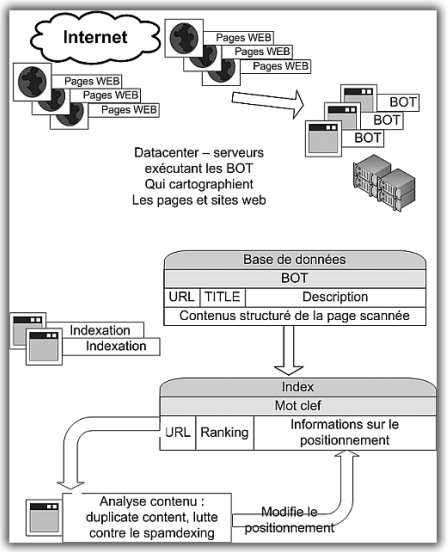
Le but de ce premier paragraphe est d'expliquer le minimum à connaître pour un codeur sur le principe de fonctionnement d'un moteur de recherche.

Un moteur de recherche accède aux pages des sites sur Internet via une multitude de petits programmes, les *bots*, qui suivent les liens lisibles de chaque page de tout site web.

> ⚠ **Bot**
>
> Ce petit programme informatique est lancé à des millions d'exemplaires sur des milliers de serveurs de Google. Un *bot simule un internaute mal voyant* capable uniquement de "lire" les données de type texte. L'impact sur les techniques d'optimisation est très important. Un bot suit uniquement les liens en dur. Il ne peut interpréter les liens "programmés" en JavaScript, en Flash ou autre langage. Il télécharge chaque page sur laquelle il passe via les liens qui interconnectent ces pages entre elles. Ces pages sont ensuite cartographiées par d'autres programmes de Google.

Google est composés d'algorithmes, de robots logiciels, lancés séquentiellement pour réaliser une mission unique jamais finalisée : indexer sur leurs mots-clés les pages de tout l'Internet techniquement accessible.

Ces programmes de Google n'ont pas la finesse de l'esprit humain, ils sont plutôt "basiques" même s'ils sont capables de traiter un volume énorme d'informations. Pour améliorer son référencement, il est crucial de se mettre au niveau de ce type de programme et des attentes sous-jacentes.




► Fig. 1.2 : Principe fonctionnel de l'indexation.

Un bot de Google cartographie tout ce qui est accessible en données des pages d'un site. Les données cartographiées sont stockées sur un serveur. Puis d'autres programmes Google analysent les données cartographiées et ils les indexent. Nous verrons plus loin les principes de l'indexation pleine page. Une fois les pages de ce site indexées, un algorithme va les positionner parmi les pages déjà indexées sur chaque mot-clé de chaque page de ce site.

ⓘ L'enjeu du référencement

L'enjeu du référencement puis du positionnement est d'influencer le moteur pour avoir, en haut de l'index, la landing page du mot-clé ciblé. Le rôle du codeur est fondamental et peut faire la différence entre des sites concurrents sur la même expression clé stratégique.

## Mots-clés et Google writing

Le *Google writing* désigne une technique rédactionnelle adaptée à l'optimisation de la rédaction de texte pour le moteur Google (et les autres aussi). Ce travail, qui est du contenu, est à la charge du rédacteur. Le rédacteur utilisera au maximum les mots-clés qu'on lui a désignés.

Mais la mise en page associée au Google writing est à développer par le codeur : conteneur XHTML, feuille de style avec des entrées doubles (même résultats visuels de 2 classes distinctes codées différemment) et autre optimisations.

Le codeur verra dans le présent ouvrage pourquoi et comment automatiser une mise en page capable de mieux optimiser une page pour Google.

## Cookies, sessions et URL

Un *cookie* est un code de marquage envoyé par un site Internet sur votre disque dur. Ce code est unique. Si vous revenez sur le site Internet auteur dudit *cookie*, seul ce site, son auteur, peut y accéder et le lire pour vous identifier et ainsi appliquer vos attentes en affichages, en données préférées et pour de multiples autres usages comme le contrôle d'accès, par exemple.

Le codeur de site doit savoir qu'un bot de Google n'accepte pas les cookies. Les pages verrouillées par ce procédé lui seront illisibles si le cookie est obligatoire pour accéder au contenu. Nous verrons qu'une mauvaise gestion des cookies peut entraîner des pénalités par Google. En effet, le moteur peut prendre des maladresses de codage pour du *spamdexing* et donc réagir en sanctionnant plus ou moins fort la page HTML générée par votre site dynamique.


► Fig. 1.3 : URL surchargée de amazon.fr sur le livre Saga Fascination, Tome 4 : Révélation

Le web marketing est en général conscient que dans 10 à 15 % des cas, un cookie est impossible à poser ou à gérer. Il n'est pas fiable de baser des applications web sur la gestion de cookies, sauf à accepter les internautes rétifs aux cookies ou à un taux d'erreur de 10 à 15 %.

Une page HTML possède une seule URL. Et à une URL correspond une seule page HTML.

Cette notion pour Google d'une URL unique pour une page à contenu unique a un énorme impact sur la publication des contenus d'un site et sur la gestion des liens internes dans un site. Le codeur devra gérer de multiples conflits susceptibles de passer pour du spamdexing auprès de Google.

### ⓘ Un même article peut appartenir à plusieurs thèmes

Un exemple parmi d'autres, juste pour illustrer le propos : soit un site publiant des articles sur la (bonne) gestion des entreprises. Un article sur les charges sociales de la rémunération d'un dirigeant a un contenu précis mais intéressant plusieurs thématiques : la fiscalité, les charges sociales, la gestion financière. On risque donc dans un site de trouver :

**www.bonne-gestion.fr/fiscalite-entreprise/charges-remuneration-dirigeant.html** ;

**www.bonne-gestion.fr/charges-sociales/charges-remuneration-dirigeant.html** ;

**www.bonne-gestion.fr/gestion-financiere/charges-remuneration-dirigeant.html**.

Soit, trois URL distinctes pour une même page. C'est à éviter. Nous verrons plus loin une technique pour coder en évitant la visibilité par Google de ce type de conflit néfaste au bon positionnement dans l'index du moteur.

Une *session* se caractérise par l'ajout dans l'URL d'un ou plusieurs codes permettant de personnaliser les échanges avec un internaute précis ou ayant un profil précis.

Par rapport aux cookies, cette méthode offre l'avantage d'être 100 % fiable. L'inconvénient, c'est que Google n'apprécie pas trop. En effet, on trouve de tout dans les URL à session : des jetons d'authentification, des codes permettant à la logique des scripts PHP de réagir, etc.


► Fig. 1.4 : Dans Google, URL amazon.fr sur le même contenu : livre Saga Fascination, Tome 4 : Révélation

Au final, il existe de multiples URL différentes pointant vers une seule page physique. Les sessions introduisent donc de massifs *duplicate content* dans le site. Google a l'habitude et il se contente de plus ou moins bien désindexer les URL surnuméraires. Là, le web marketing (le référenceur) se heurte à un problème que le codeur va devoir résoudre : "Comment connaître l'URL qui va survivre à la purge de Google ?". La réponse est importante, car les mots-clés composant une URL participent au référencement. Si Google élimine l'URL en charge de véhiculer des mots-clés et prend en compte une URL truffée de codes de session comme valide, le site perdra en qualité de référencement et donc en positionnement.

## Lien et backlinks

Les pages HTML d'un même site web ou de différents sites sont reliées par des liens.

### ⓘ Lien en dur

Quand un lien est directement accessible, c'est-à-dire qu'il n'est pas enfoui dans un codage informatique de type JavaScript, on dit que c'est un lien "en dur". Un lien en dur apparaît en bas du navigateur quand la souris le survole.

Les *liens en dur* sont visibles et accessibles par tout type de navigateur, dont ceux conçus pour les mal voyants et les périphériques Braille avec voix de synthèse associée. Les bots des moteurs de recherche accèdent uniquement aux liens en dur. Ils peuvent les suivre pour cartographier les contenus des pages ainsi reliées. Les liens enfouis dans des codes informatiques sont invisibles aux bots.

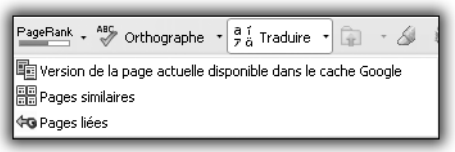
Un *backlink* désigne un lien en dur entrant sur une page. Ce lien peut être issu :

- d'une page du même site web (le nom de domaine est identique pour les deux URL des deux pages interconnectées par ce lien) ;
- d'un autre site.

La nuance entre *lien en dur* et *backlink* tient à la position d'où on s'exprime. Lorsqu'on fait référence à une page et que l'on souhaite désigner les liens en dur pointant vers ladite page et utilisés pour travailler le positionnement dans l'index de Google sur un ou des mots-clés, on parlera de backlinks. C'est plus court, plus concis et plus clair que de parler de liens en général. Quand on veut évoquer un lien en général, donc un lien en dur, codé dans un Flash ou en JavaScript, on emploie la dénomination générique de lien.

## Première approche du Page rank

Le *Google Page rank* désigne l'indice de popularité d'une page HTML calculé par des algorithmes de Google. Nous considérerons que cet indice représente grossièrement le volume en backlinks pointant vers ladite page. Cela correspond à une forme de popularité. Plus une page reçoit de liens, plus cela indique que des internautes la trouvent intéressantes car ils l'écrivent dans des sites, des blogs, des forums, etc. Plus il y aura de backlinks posés par ces internautes, plus la popularité de la page ciblée par ces backlinks augmentera. Le Page rank est affiché dans la barre d'outils Google pour Firefox ou pour IE.




► Fig. 1.5 :
Barre d'outils Google : Page Rank affiché.

Le *Page rank* (acronyme PR) agrège de nombreuses autres informations. Dans un post sur le blog officiel de Google, Matt Cutts, le responsable de l'antispam team de Google, répond à une question d'un webmestre en indiquant que le Page rank est utilisé pour "calculer" qui est à l'origine d'un texte ; sous entendu, les pages ayant ce même texte seront considérées comme des *duplicates content*, des copies qui seront ignorées par Google.

### Le Page rank est très important

De nombreux posts dans des forums et des blogs dénigrent l'intérêt du Page rank. C'est une erreur. Le PR sert de levier à de nombreuses actions de Google. En avoir augmente potentiellement l'efficacité du référencement et du positionnement, à condition de l'utiliser. Il faut l'employer correctement, bien sûr.

## Structure de site : hiérarchiser

Un site web est destiné à publier des contenus. L'architecture du web nécessite de structurer ces contenus en rubriques, sous-rubriques, pages reliées par des liens groupés en menus.

La structure de publication est exprimée par l'ergonomie des liens, des menus, de la navigation. La qualité de cette structure a un énorme impact sur le positionnement. Plus la structure est "cartésienne", clairement hiérarchisée, mieux le moteur comprend les contenus du site et donc mieux il le positionnera sur les mots-clés.

Le référenceur doit faire comprendre au moteur le contenu de chaque page. Il dispose de nombreuses techniques complémentaires pour y arriver.

L'une d'elle consiste à imposer aux bots de Google un parcours strictement hiérarchisé entre page d'accueil, têtes de rubriques, sous-rubriques, pages filles.

Pour faciliter cette hiérarchie, chaque page traite d'un sujet précis. La page est optimisée pour quelques mots-clés que l'on trouve dans l'URL de cette page ainsi que dans les différents critères la composant : title de page *meta*, balise *Hn*, etc. Nous y reviendrons plus loin.

En imposant une telle vision du site à Google, tout en laissant d'autres principes de navigation en service sur le site mais invisibles pour Google, le codeur améliore fortement le positionnement des landing pages de son site dans l'index de Google. De même, si pour des raisons de lisibilité pour les humains, un rédacteur décide de faire des jeux de mots dans un titre ou de mélanger des thèmes dans une page, il sera efficace de limiter la lecture de Google sur ces éléments, voire de la lui interdire.

## Le référencement : un savoir-faire difficile à acquérir

Google est une entreprise américaine donnant très peu d'indications sur son fonctionnement technique. Il suffit de lire les interviews de responsables techniques de Google pour s'en convaincre. Voici quelques exemples :

À partir de quel seuil le moteur réagira-t-il ainsi ? Pas de réponse. Quels sont les critères de classement dans l'index ? Vague réponse : contentez-vous de mettre votre contenu en valeur.

Les techniques de classement dans l'index de Google sont secrètes afin de laisser échapper le moins de savoir-faire possible vers les concurrents et les référenceurs :

- Pour protéger son avance, Google ne publie pas de réelles informations sur le fonctionnement de ses algorithmes. Les référenceurs sont donc obligés de chercher

par eux-mêmes, ce qui coûte cher. Cela incite chacun d'entre eux à protéger la partie "pointue" de son savoir-faire. D'où une pénurie d'informations de qualité pour référencer et positionner efficacement son site.

- Les internautes ont fait Google. Seuls ces derniers pourront le défaire. Moins les concurrents possèdent d'informations, moins les référenceurs peuvent tenter de polluer l'index de Google en imposant leurs pages au détriment de pages concurrentes. Il est alors plus facile pour Google de rester en tête et même d'accroître son avance.
- Tant que l'usage de Google est fortement répandu parmi les internautes du monde entier, les entreprises souhaitant être vues sur le Net devront passer par Google, via ses liens sponsorisés AdWords ou/et le référencement naturel. Cela représente une énorme sources de revenus.
- Google va donc soigneusement protéger son succès auprès des internautes en luttant contre toute pollution de son index. La masse des utilisateurs fidélisés par la qualité des réponses de Google établit ce moteur comme leader sur le marché de la recherche sur le Net. Il n'est pas question pour Google de laisser des actions de référencement trop agressives polluer son index et compromettre, à terme, le succès de son *business model*.

Le codeur doit comprendre le savoir-faire d'un référenceur afin de mieux appréhender les exigences du référencement et leurs impacts sur ses codes PHP, ASP, .NET, etc.

## Codage web2.0 et réseaux sociaux

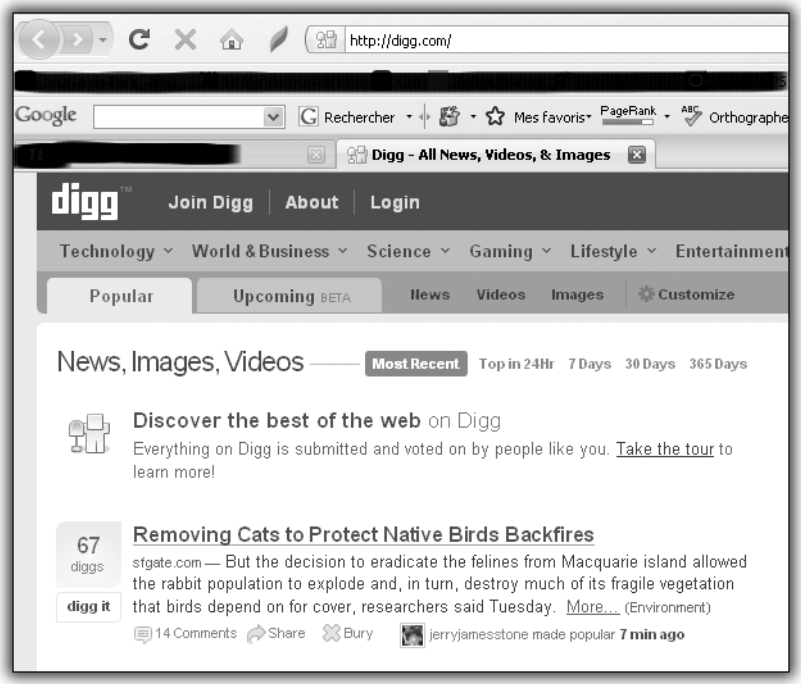
L'usage *réseau social* (YouTube, Facebook et des dizaines d'autres) intéresse fortement Google.

Cet usage réseau social n'est pas une mode mais une vraie révolution des relations sociales, du fonctionnement des communautés, dans le monde. Elle en est à ses débuts.

Google et quelques autres l'ont bien compris. Réseau social et mobilité vont profondément remodeler la société dans les prochaines années.

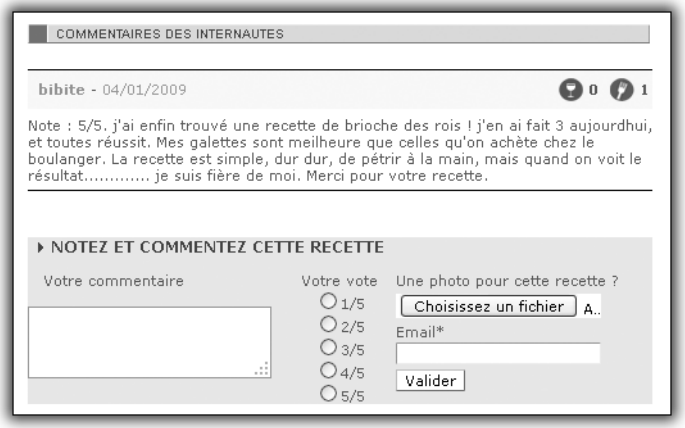
Les réseaux sociaux regorgent de backlinks déposés par des internautes : mes *sites préférés*, *partage de favoris*, etc. Google exploite les backlinks déposés dans des réseaux sociaux accessibles à ses bots.

Faciliter la pose de backlinks par des internautes abonnés à des réseaux sociaux est techniquement aisé à réaliser.




► Fig. 1.6 : Page Rank sur digg.com et backlinks sur les pages du site

Une autre approche *web2.0* consiste à laisser les internautes interagir avec les pages de votre site. Plus il y a d'échanges autour de votre page – ajout de textes dans le site lui-même, réactions autour de vos articles –, plus le volume d'intégration des textes dans la page augmente, "lestant" la page en textes et donc améliorant son positionnement.




► Fig. 1.7 : Sur 750g.com, un commentaire qui renforce le référencement de la page : mots clefs et cohérence sont présents.

Les bots de Google cartographient chaque page. Ils peuvent tracer la fréquence de mise à jour et ainsi, au vu de fréquentes mises à jour via l'ajout de posts et autres commentaires, en déduire une signature type d'une page *web 2.0*. Il est indiscutable, tests à l'appui, que ses algorithmes en tiennent compte et qu'ils favorisent le positionnement de ces pages *web 2.0* au détriment de pages *web 1.0*.

**Installer des fonctions web2.0**

Si vos pages ont un contenu justifiant des échanges avec les internautes, profitez-en. Toute page ayant du succès en interactivité avec les internautes se voit octroyer des avantages dus aux backlinks ainsi générés et au texte ajouté à la page : commentaires, avis, notes, etc. Google favorise les pages *web2.0* actives et utilisées dans son index.

Un codeur pourra développer puis installer des modules PHP/MySQL destinés à :

- l'interactivité de toute page avec des internautes : avis, vote, notes, questions/réponses, etc. ;
- la modération des participations écrites a priori (validation avant tout affichage) ou a posteriori (retrait ou amendement de la participation écrite si elle ne convient pas au webmestre).

## Principes du référencement et du positionnement naturel

**Le référencement naturel d'un site Internet**

Ce sont les actions techniques et rédactionnelles destinées à optimiser les pages du site pour tenter de les faire afficher dans les SERP d'un moteur sur un ou des mots-clés précis. Ces actions techniques sont localisées dans les pages du site et jamais en dehors.

*Optimiser une page HTML* signifie mettre en œuvre des techniques Internet et rédactionnelles sur une page pour faciliter la compréhension des contenus de cette page par le moteur de recherche.

L'objectif est de le pousser à prendre en compte les mots-clés dont le site estime avoir besoin en priorité sur cette page. Quand on définit que telle page sera la landing page, la page d'atterrissage sur telle expression clé, il faut tout mettre en œuvre pour pousser en avant cette page vers le top5 dans les SERP de Google :

- Optimisation de la page, objet du présent chapitre.

- Optimisation des autres pages pour mettre celle-ci en avant via la technique des nuages de liens. Le principe de cette approche est décrit un peu plus loin dans le chapitre consacré aux backlinks.
- *Netlinking*. Cet aspect web marketing est détaillé dans un autre livre chez MicroApplication : *Le Guide complet du référencement sur Google*. Dans le chapitre consacré aux backlinks du présent ouvrage, nous effleurerons le sujet afin de nous consacrer à l'aspect codage destiné à faciliter le *netlinking* et sa variante interne au site, les nuages de liens (*tags clouds*)

ⓘ Positionnement d'un site Internet

Ce sont les actions extérieures au site lui-même qui vont hisser les pages d'un site dont le référencement est correct, vers les premières positions dans une page de résultats de moteur (une SERP) sur les mots-clés voulus par le webmestre ou le traffic manager.

Sur chaque page de tout site web accessible à Google, différents critères, que nous détaillerons plus loin, sont évalués par Google. Ils lui permettent de trier dans son index les millions de pages ayant un mot-clé en commun.

Ensuite, Google évalue les backlinks pointant vers chaque page d'un site. Cette évaluation finalise pour quelques heures le classement dans l'index. Ce classement est régulièrement recalculé. Sauf pour l'applications de pénalités, le fonctionnement de ces algorithmes d'évaluation lisse les montées ou descentes de pages au sein de l'index.

*Réussir un positionnement* des landing pages de son site dans les SERP de Google exige de réaliser un travail séquentiel :

- un référencement de qualité, c'est-à-dire les travaux d'optimisation des pages du site respectant une cohérence vis-à-vis des mots-clés utilisés ;
- puis un travail de netlinking interne (nuages de liens) ;
- enfin un netlinking externe vers les landing pages puis un netlinking vers d'autres pages.

## Contenu et mots-clés

Pour avoir un site positionné en bonne place, il faut que ce dernier dispose de réels contenus : textes, visuels, vidéo, etc. Ces derniers créent de la valeur ajoutée pour l'internaute.

Les volumes, en nombre de mots par page et en nombre de pages, sont importants. Une fois optimisés, ces pages et ces textes influeront fortement sur le référencement du site et sur son positionnement dans l'index du moteur.

Le web marketing aura à sa charge de sélectionner les mots-clés et la répartition de ceux-ci sur les pages du site.

**Clics dans les SERP et positionnement mobile**

Le taux de clics sur les pages du site a une influence sur le positionnement. Plus il y a d'internautes saisissant une requête avec l'expression clé `gite équestre Normandie`, qui cliquent vers une même landing page, plus cette landing page se hissera vers de meilleures positions sur ces mots-clés composant la requête. Les internautes votent via Google pour une page associée à des mots-clés.

## Accessibilité des mots-clés

L'accessibilité se décompose en deux types :

- *Technique*. Le robot doit pouvoir accéder à la page et à son contenu. En effet, de nombreux sites ont des JavaScripts et autres gadgets techniques qui bloquent plus ou moins les bots de Google.
- *Sémantique*. Rendre accessible au moteur les mots et expressions clés retenus lors de l'étape précédente revient à optimiser les pages du site. Pour cela, le travail est découpé en plusieurs étapes :
  - répartir les mots-clés et les combinaisons de mots et expressions clés sur les pages du site ;
  - considérer un maximum de 5 mots-clés par page (nous verrons le pourquoi de cette limitation plus loin) ;
  - optimiser en conséquence chaque page du site autour des mots-clés qui lui sont affectés.

Sauf erreur de votre part, le moteur prendra en compte vos 3 à 5 mots-clés stratégiques pour chaque page. *L'accessibilité sémantique* a ainsi pour objectif d'orienter le choix de Google sur vos mots-clés pour chacune des pages de votre site web, les landing pages comme les pages en charge de pousser les landing pages en avant.

## Backlinks et nuages de liens

Un site correctement référencé pourra recevoir des actions de positionnement pour disposer de pages visibles dans les deux premières SERP sur les mots-clés.

Le netlinking (*linkbuilding*) est une des deux seules actions permettant d'avoir un contrôle sur le positionnement des pages. Il peut être interne (gestion des liens entre pages du même site) ou externe (les liens venant de pages externes au site). On pourra jouer sur plusieurs caractéristiques :

- la qualité de chaque lien : la qualité de l'environnement de départ, le texte du libellé du lien et de la page visée par l'URL du lien ;
- le volume des backlinks ;
- les pages visées : on pourra favoriser certaines pages plutôt que d'autres et ainsi influencer le positionnement de certaines pages au détriment d'autres pages ; c'est un des intérêts de disposer de nombreuses pages hiérarchisées via des menus.

### Une stratégie de référencement

Le marketing décide d'une stratégie de référencement : mots-clés, pages d'atterrissage, etc.

Puis au fil de l'acquisition de statistiques, de retours sur l'utilisation du site, des changements sont apportés à cette stratégie :

- de nouveaux mots ou expressions clés sont à prendre en compte ;
- de nouvelles pages sont ajoutées, de nouvelles rubriques sont créées ;
- de nouvelles landing pages sont définies pour les nouveaux mots et expressions clés.

Il est du ressort du codeur de fournir un site pouvant aisément mettre à jour sa stratégie de mots-clés et de pages d'atterrissage. Nous étudierons cela plus loin.

## 1.2 La formule magique du référencement

Nous avons vu plusieurs éléments distincts qui comptent dans le référencement.

Pour faciliter la compréhension de ces mécanismes complémentaires mais indissociables, voici une formule qui devrait aider à mieux les appréhender.

Afin de faciliter la compréhension de la formule, nous utiliserons l'exemple d'une page *P* à positionner sur l'expression clé `"formation web"`.

Le positionnement d'une page HTML sur une expression clé est fonction de :

- La présence dans les différents critères de cette page de l'expression clé : `title` de page, `Hn`, texte, balises `ALT` et `TITLE` d'image, ancres (départ de liens).
- La densité de cette présence pour chaque type de critère. Exemple : on trouve 4 fois l'expression `formation web` en `H1` sur un total de 40 mots. La densité est de 2 fois 4/20, 20 % pour chaque mot-clé composant notre expressions clé `formation web`. Plus la densité est élevée, mieux c'est, jusqu'à atteindre un seuil limite. À partir de ce seuil, on risque de passer du coté du *spamdexing*.

- Du volume de critères de poids. Une expression clé présente uniquement en `H1` pèse davantage pour un bon positionnement que cette même expression clé présente uniquement en `H2`. La présence du même mot-clé, en `H1` et en `H2`, s'additionne et elle octroie un bonus de cohérence à notre page.
- Du Page rank de la page. Il fait office de levier sur les mots-clés déposés dans les ancres. Un mot-clé en ancre de lien, elle-même dans un `h1`, pèsera davantage pour le positionnement dans une page à PR3 que la même page en PR1. `<h1> <a href="formation-web-marketing.html"> Formations en web marketing</a></h1>` cumule le levier du `H1`, de l'ancre sur les mots-clés formation/web/marketing multiplié par le Page rank pour notre page et pour la page visée par l'URL du lien. Ces trois mots-clés se partagent la poussée.
- Du volume de backlinks pointant vers notre page et ayant notre expression clé dans le libellé du lien.
- De la qualité de la page de départ des backlinks abordés précédemment. La poussée transférée par chaque lien vers les bonnes positions dans l'index sera fonction du Page rank de la page de départ, du volume de mots dans chaque libellé, de la cohérence de ce mot entre la page de départ et la page d'arrivée.
- Si `formation` est présent dans différents critères de la page de départ et de la page d'arrivée, la cohérence ainsi réussie pousse notre page davantage vers les bonnes positions dans l'index de Google. Moins la cohérence est bonne, moins il y a de poussée. À partir d'un certain seuil, la poussée globale est tout simplement ramenée à `0`.

## 1.3 Tricher ou non en référencement

Il y a peu de bonnes positions dans les SERP de Google : TOP5, page 1, page 2 ou page 3, face à des centaines de milliers ou des millions de pages candidates aux mêmes expressions clés que votre site.

### Spamdexing

Toutes les actions polluant l'index de Google constituent des attaques contre la qualité des SERP du moteur. Elles sont considérées comme du spamdexing. Nous avons réparti en différents groupes les actions de spamdexing capables de polluer l'index de Google.

Google doit s'assurer d'un index propre pour conserver son public d'internautes. C'est une question de survie et de revenus pour Google. Le moteur a édicté de multiples niveaux de pénalités contre les actions de spamdexing pour protéger ses intérêts.

Mais avoir absolument un bon positionnement peut être une question de survie ou de rentabilité pour l'entreprise via son site web. L'entreprise aura tendance à lancer des campagnes de netlinking. Celles-ci incluent une forme ou une autre de *paidlinks* prohibés par Google (chaque paidlink floue l'algorithme de Page rank et lui prend des revenus issus de *AdWords*).

⚠ Comprendre les enjeux du spamdexing

C'est au responsable du projet ou de l'entreprise d'arbitrer et d'assumer les décisions en toute connaissance de cause pour l'emploi ou non de techniques de spamdexing.

## Spamdexing : les différentes techniques de triche

Nous avons regroupé les techniques de triche (le spamdexing) en différentes catégories pour en favoriser la compréhension. Un codeur maladroit peut produire à son insu du spamdexing et ainsi précipiter le site web dans des pénalités Google plus ou moins fortes.

Le codeur doit comprendre les différentes techniques de spamdexing pour éviter d'en coder à son insu.

### Spamdexing sur la page

Il regroupe les techniques de triche et erreurs humaines portant sur le contenu d'une page accessible aussi bien par un internaute humain que par un bot de Google.

Quelques exemples : écrire une partie des textes d'une page en police de couleur blanche sur un fond blanc, dissimuler des textes dans la page au moyen de faux calques DHTML, multiplier les titres de niveau 1 (textes entre des balises `h1`), etc. constituent des actions de spamdexing au niveau de la page.

Le principe consiste à faire prendre en compte, dans une page par le moteur, des textes ou une mise en page invisibles aux internautes. Le bot prend en compte ces textes cachés et les mots significatifs qui les composent ainsi que la mise en page avec l'optimisation sous-jacente. Le référencement de la page en est donc amélioré jusqu'à la découverte plus ou moins rapide des textes cachés par les algorithmes anti-fraude de Google.

Cette technique de triche ne présente aucun intérêt. On peut faire aussi performant sans tricher, donc sans risque.

⚠ Un codage maladroit peut générer du spamdexing

Par exemple, un site change son affichage et sa charte graphique en fonction des quatre saisons : été, hiver, automne, printemps. Un test oublié, un correctif CSS tardif peuvent entraîner la réalisation accidentelle d'un texte dans une couleur de police identique à celle du fond. Cette forme de spamdexing est pénalisée par Google.

## Le cloaking

⚠ Le cloaking

Cette technique porte sur des astuces en programmation aboutissant à rendre invisible à des internautes humains une page visible uniquement par un bot de moteur, et à rendre invisible au bot une page destinée exclusivement à des humains. Les deux pages ont exactement la même URL.

Ce type de triche peut être facilement éventé par Google.

Les gains obtenus par la technique de *cloaking* sont faibles par rapport à un site correctement optimisé sans triche. Son intérêt est nul. On peut trouver d'autres approches pour éviter le cloaking.

Exemple : un site d'automobiles de luxe allemandes utilise des descriptions de leurs produits avec des mots qui n'ont aucun rapport avec la réelle fonction de l'objet : palace roulant, sérénité, ambiance feutrée et autres qualificatifs qu'aucun internaute ne saisit dans une requête concernant le monde automobile.

Ce site de voitures luxueuses décide de la nécessité de disposer de pages et contenus pour les internautes d'un côté, et de ces mêmes pages (même URL) avec des contenus optimisés en mots-clés "utiles" pour Google.

Principe de fonctionnement : un JavaScript ou un script PHP sur chaque page "bricolée" détecte la provenance de l'internaute. Si le visiteur vient d'une adresse IP appartenant à Google, Yahoo, ou Microsoft, on lui présente une page optimisée "moteur" ; autrement, la même URL correspond à un contenu totalement différent orienté vers un internaute humain à séduire. Il existe de multiples variantes de ces scripts, toutes orientées dans la présentation de contenus optimisés visibles exclusivement par les bots des moteurs pendant que les mêmes URL présentent d'autres contenus aux humains.

⚠ Une autre formulation du cloaking

Une même URL présente au même moment deux contenus totalement différents selon que le visiteur est un bot ou un internaute.

Le moteur indexe la page optimisée, car il a été abusé dans un premier temps. Quand l'internaute clique sur un lien menant à cette page depuis un SERP, il arrive sur un tout autre contenu de cette même page (il a été détecté comme humain par le script).

Une erreur humaine en codage du site peut aboutir au même résultat. Un bogue ou un test sur une identification, un cookie par exemple, mène à une page ou à une autre (mais avec la même URL). Quelquefois, le site affiche des textes différents sur une même page selon la situation rencontrée.

Pour Google, cela ressemble à une URL unique ayant au moins deux contenus plus ou moins différents au même moment selon le profil de l'internaute : un clic ici donne toujours cela, mais ce même clic avec une configuration de navigateur ou une provenance différente donne, sur la même URL, un contenu différent. Si les contenus sont totalement différents, cela peut occasionner une interprétation d'action polluante, de spamdexing, contre l'index du moteur et une réaction de celui-ci sous forme de sanction. Si les écarts de contenu sont faibles, Google ignorera le problème. Au codeur de faire en sorte d'éviter ces situations.

⚠ Un moteur de publication

Pour éviter les erreurs, nous recommandons d'utiliser autant que faire se peut un moteur de publication qui, une fois au point, permet de publier sans risque d'erreur de programmation des contenus sans *duplicate content* interne et autres erreurs susceptible de pénaliser le référencement du site sur Google. Nous verrons le code source d'une version simple d'un tel logiciel dans ce livre. Il servira à publier les informations de notre site web exemple à référencer automatiquement.

## Duplicate content

Des contenus identiques existant sur différentes URL sont du *duplicate content*.

⚠ Une URL, une page unique en contenu

Selon les règles des moteurs, à une page unique (une URL unique) doit correspondre un contenu unique afin de ne pas polluer l'index du moteur avec de multiples pages à contenus identiques aux URL différentes.

En fait, cette règle de Google est plus souple ; il y a de nombreux cas où un petit contenu identique est publié par de nombreuses pages de sites différents : synopsis d'un film, communiqués de presse, descriptions de produits vendus par de nombreux sites, nouvelles d'actualité, etc.

Néanmoins, passés certains seuils en nombre de caractères et quelques autres critères que nous n'aborderons pas encore, Google peut juger que deux pages possèdent des contenus trop proches. Il sanctionne une des deux pages en la désindexant, partiellement ou totalement selon le volume de duplicate content.

Le duplicate content est une erreur très courante, qu'il est très facile de provoquer accidentellement manuellement ou via des erreurs de codage :

- Un copier-coller maladroit. Exemple : depuis la page du site français, le webmestre copie colle des données vers la page correspondante dans le site belge francophone.
- Une page trop légère en contenu unique et parsemée de textes/menus/images répétés sur d'autres pages du même site. Par exemple, la page *Nos coordonnées* contient tous les menus du site et juste une poignées de caractères destinés à l'adresse postale et au numéro de téléphone. Une telle page sera probablement évaluée comme ayant trop de duplicate content interne au site ; elle sera donc désindexée.
- De multiples sites publient la même information, vendent le même produit avec le même descriptif, etc.
- Une faute en hébergement : des substitutions aléatoires du nom de domaine par un nom générique de l'hébergeur qui font correspondre pour Google plusieurs URL/pages ayant exactement le même contenu.
- Une erreur de codage en *URL rewriting* faisant correspondre plusieurs URL distinctes vers la même page physique.
- Un bogue dans la gestion informatique des contenus aboutissant à avoir des contenus identiques dispersés sur plusieurs pages différentes.
- Vos contenus ont été dupliqués par un autre site, volontairement ou non.
- Des cumuls de plusieurs petites erreurs aboutissent à des contenus identiques partiels dans des pages trop légères en volume de contenu.

## Spam de liens, les paidlinks

Cela porte sur "les liens non naturels versus les liens naturels". C'est la technique qui provoque le plus de polémiques pour de multiples raisons.

On vient de voir que les techniques de triche ne possèdent aucune efficacité. Elles sont aisées à détecter.

Les backlinks sont primordiaux et uniques pour obtenir un bon positionnement. Il est tentant d'agir via du spam de liens pour obtenir des backlinks.

Les règles distinguant les liens naturels des liens non naturels édictées par Google sont floues, potentiellement contraires à la législation fiscale des pays occidentaux dans certaines interprétations, et difficiles à bien contrôler par le moteur. C'est donc la porte ouverte pour tout site web à investir dans de nombreuses formes de campagnes de backlinks (on appelle cela *netlinking*). Les pages capables d'inciter un internaute à poser un lien vers elles appartiennent au monde du *linkbaiten*, technique non réprouvée consistant à disposer d'une qualité de contenu incitant les internautes à en parler et à poser des liens depuis leurs blogs, posts de forum, etc.

Le *paidlink*, comme l'indique son nom, consiste à "*payer*" le poseur de lien et le site hébergeur dudit lien pour avoir un lien en dur.

Les paid links, technique de web marketing en positionnement n'ayant rien à voir avec la programmation, sont abordés en détail dans le *Guide complet du référencement sur Google*, paru chez MicroApplication.

## 1.4 Résumé de ce chapitre

Nous avons abordé dans ce chapitre les termes techniques, concepts et autres informations nécessaires pour comprendre les contraintes du référencement et du positionnement vis-à-vis d'un site et de sa programmation.

De nombreuses tâches répétitives et fastidieuses comme optimiser page HTML par page HTML, et une qualité sans défauts de ces tâches caractérisent le travail de base.

Les codeurs percevront la nécessité :

- d'automatiser les URL ;
- d'éviter de générer accidentellement du *spamdexing* ;
- d'éviter de publier ou afficher des informations perturbant la logique de Google ;
- de faciliter un minimum le travail du référenceur et du web marketing via un back office simple leur permettant d'administrer les mots-clés du site.

Deux autres aspects ont normalement attiré l'attention du codeur :

- utiliser la programmation pour des fonctions *web2.0* pour améliorer référencement et positionnement ;
- automatiser les backlinks internes : les tags clouds (nuages de liens).

Ces tâches attendant le codeur sont rapides à décrire, mais vous aider à leurs mises en place demandera quelques chapitres dans ce livre.

# 2

# Automatiser le référencement avec PHP et MySQL

Dans le chapitre précédent, vous avez acquis un premier niveau en notions de référencement sur Google. L'objectif des chapitres à venir, incluant celui-ci, est de vous exposer les algorithmes et des codes sources liés à l'optimisation de chaque page HTML puis la coordination de ces optimisations individuelles pour mettre en place une stratégie globale de référencement et de positionnement dans l'index du moteur de recherche Google.

## 2.1 Introduction

Comme pour la construction d'une maison, il va falloir commencer par les fondations pour construire le référencement naturel d'un site web.

Les fondations d'un site pouvant être référencé sur Google sans aucun frein consiste en un logiciel de publication de pages avec lequel il sera possible de tout faire.

En effet, les *Joomla* et autres logiciels CMS (*Content, Management System*), sont des logiciels permettant d'organiser puis de publier des pages. Ils présentent quelques défauts que nous voulons éviter dans notre propre logiciel de publication dont :

- La liberté pour le codeur ou le webmestre d'optimiser une page ou ses pages à sa totale convenance. On ne fait pas ce que l'on veut avec un logiciel de CMS. Par exemple, *Joomla 1.5* possède encore quelques solides défauts nuisibles à un référencement de très bonne qualité.
- La simplicité de mise en œuvre de ces optimisations. Il faut un laps de temps non négligeable pour maîtriser *Joomla 1.5* ou un autre logiciel de CMS. Ne pas pratiquer revient à oublier. Les coûts de référencement s'en trouvent augmentés d'autant.
- La rapidité pour ajouter, modifier des contenus. *Joomla 1.5* est lent. Le temps passé à attendre, la multiplicité des onglets à cliquer, ralentissent le référenceur. Cela augmente donc le coût du référencement.

Le présent chapitre commence par exposer et expliquer les codes sources d'un logiciel simple de publication de pages HTML.

## 2.2 Codes sources – site web exemple du livre

Tout au long de ce livre, nous allons illustrer son contenu avec des codes sources issus d'un exemple de site pouvant être référencé sur Google.

Ce logiciel de publication est écrit en PHP, il utilise des données gérées sous MySQL et génère du XHTML, CSS2.

Vous pourrez reprendre les codes sources de cet ouvrage et les décliner vers vos besoins propres.

### Un logiciel de publication adapté

La problématique de base d'un site web est schématiquement toujours la même : lister des résumés et permettre, d'un clic, de zoomer sur un article dont le résumé a intéressé l'internaute. Le logiciel de publication que nous allons regarder s'appuie sur ce principe.

Vous pourrez donc l'adapter à de nombreux sujets et le compléter avec vos besoins fonctionnels.

Ce site web exemple présente aux internautes des articles consacrés à la création d'entreprises et à des idées, des conseils destinés aux futurs chefs d'entreprise.

Le propos de ce site web exemple à référencer et à positionner est donc de publier ce type d'informations. Pour cela, il s'appuie sur notre "moteur" de publication très simple type *Drill Down* :

- Affichage d'une liste de résultats, des résumés d'articles, sur une recherche par clic, par exemple tous les articles sur la `fiscalité d'entreprise`.
- Un clic sur un de ces articles affichés lance la page publiant la totalité de l'article.
- Un article peut appartenir à plusieurs thèmes, rubriques, ce qui va poser des problèmes face à Google.
- On peut afficher les articles propres à une région de France ou à toute la France. Là aussi, nous rencontrerons des problèmes face à Google.

Dans cet exemple, on rencontre donc des cas qui peuvent perturber le référencement si on ne code pas en prenant en compte les exigences des moteurs de recherche.

En clair, malgré son apparente simplicité, plusieurs pièges classiques dans lesquels nombres de codeurs tombent, seront à résoudre sur ce site exemple afin de réussir son référencement puis son positionnement.

⚠ Site exemple

Ce site web exemple est destiné exclusivement à étudier le codage de site à référencer

En aucun cas, les sources de ce site ne sont destinées à enseigner la programmation en PHP/MySQL. Il existe des auteurs spécialisés sur le sujet. De même, dans cet exemple, on n'utilisera pas de Framework PHP qui aurait alourdi l'exemple. Nous ferons simple pour favoriser la pédagogie de nos objectifs : apprendre à automatiser au maximum un référencement de qualité.

## 2.3 Coder une optimisation de page pour Google

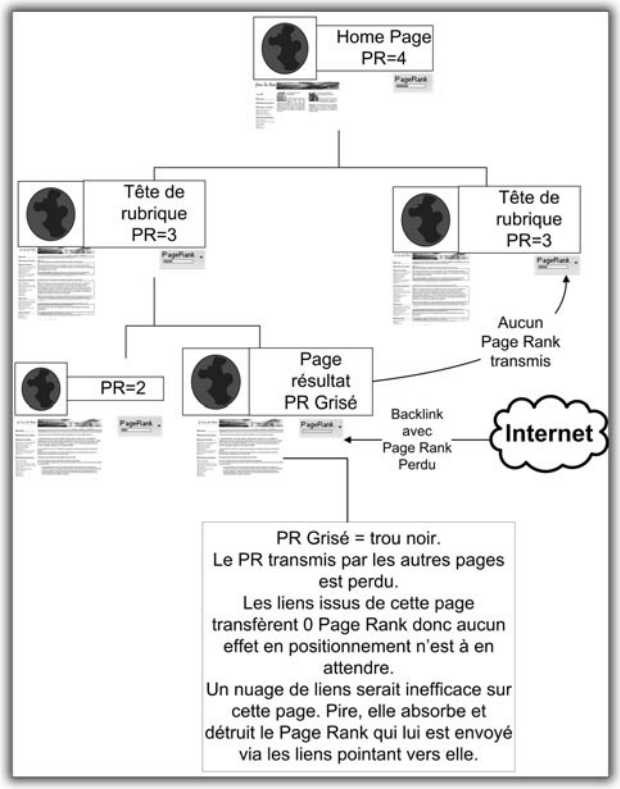
Google analyse différents critères dans une page :

- son `title` ;
- le `meta description` (ce critère n'est pas utilisé pour le calcul du positionnement d'après une interview de Matt Cutts, mais ce critère est cependant utilisé pour un affichage en snippet ). ;
- les textes entre balises `Hn`, de `h1` à `h8` ;
- les mises en exergue ;
- les textes des ancres de liens ;

- les libellés des `backlinks` entrant vers la page ;
- les textes en balises `ALT` et `TITLE` de visuel ;
- les textes ;
- les mots composant l'URL ;
- la densité d'un mot précis face aux autres mots-clés ;
- les libellés des ancres de liens quittant la page vers une autre page.

Un mot-clé stratégique se doit d'apparaître dans *tous* ces critères, impérativement.

Peut on réaliser ce travail manuellement ? Oui, mais nous laissons le lecteur imaginer ce que cela lui coûterait en temps sur un volume de quelques centaines ou même quelques dizaines de pages. Non seulement c'est une tâche chronophage et fastidieuse, mais en plus, avec la fatigue, les référenceurs manuels font des erreurs capables de diminuer le trafic. Ce qui est, bien sûr, totalement contraire à nos objectifs.




► Fig. 2.1 : Toute page désindexée détruit le Page Rank qui lui arrive et ne transmet aucun PR vers les autres pages.

La première idée qui vient à l'esprit en voyant un tel volume de travail est la suivante : "Optimisons juste quelques pages".

En effet, c'est une première piste. Mais elle présente de gros inconvénients. Chaque page peut participer de manière cohérente et synchrone au positionnement des autres pages. En clair, plus il y a de pages référencées dans un site, plus il est facile de positionner les landing pages. Nous verrons dans un autre chapitre, celui dédié aux backlinks, les raisons techniques.

⚠ Influence sur le positionnement

Le volume de pages HTML correctement référencées influe sur le positionnement

Plus il y a de pages correctement référencées dans un site, plus il est facile de positionner les *landing pages* sur leurs mots-clés respectifs dans les SERP de Google. On a donc tout intérêt à automatiser par des scripts PHP/MySQL le référencement de son site afin de référencer correctement toutes les pages du site.

Il y a de nombreuses informations à manipuler. Une base de données est nécessaire pour administrer ces données et utiliser les bonnes données sur la bonne page, au bon moment.

## 2.4 Données, mots-clés, référencement et méthodologie

⚠ Données et mots-clés se traitent différemment

Il est impératif de fixer une règle, une méthodologie sur la gestion des mots-clés au sein des structures de pages.

Plusieurs approches existent. En voici deux simples parmi de multiples possibilités :

- Tous les mots-clés stratégiques de la page sont dans le `TITLE` de page et les autres mots-clés apparaissent dans les autres critères de la page : `Hn`, `ALT` de visuels etc. Partant de là, la logique codée en PHP/MySQL exploitera cette règle.
- Dans les tables décrivant les données, on prévoit des champs supplémentaires : expression clé n°1, expression clé n°2, etc. La logique codée en PHP exploitera cette règle pour automatiser le référencement de chaque page dynamique du site.

Une règle fixant où sont les mots-clés d'une page dynamique dans une table ou une structure est nécessaire. En effet, le codeur va implémenter via des scripts PHP une logique allant chercher des informations stratégiques au référencement dans la base de données afin de les afficher dans les endroits adéquats et efficaces de la page à afficher automatiquement.

Les personnes en charge de rédiger, d'importer les données en tables MySQL devront respecter cette règle, autrement le référencement sera raté, voire contreproductif.

Nous utiliserons les 2 approches simultanément dans le site afin de les illustrer. Il appartiendra ensuite à chaque lecteur de choisir ou d'inventer la méthode qui convient à son site et à ses besoins propres.

## La structure de page en MySQL

Pour automatiser l'optimisation des pages HTML, il va falloir les construire à la volée depuis une base de données.

La structure de chaque gabarit de page doit donc être codé en XHTML/CSS mais ce codage doit être transposé en objet PHP5/table MySQL.

Commençons simplement. Une page en charge d'afficher un article est composée :

- D'un `title` de page.
- D'un `meta description`.
- D'un titre `H1`.
- D'un petit paragraphe que nous appellerons chapeau. Il sera en charge d'afficher un résumé de l'article, un chapeau donc dans le jargon des rédacteurs.
- D'un grand paragraphe que nous appellerons texte principal.
- D'une image dotée d'une balise `ALT` et d'une balise `TITLE`.
- D'un gabarit général de présentation : Logo en haut, bandeau graphique de présentation, définition des balises de mise en page en feuille de style `<p>`, `<h1>`, etc.

On va aller chercher les informations sur les données à afficher dans des tables MySQL.

### Table article

Voici la structure de la table *article* de notre site exemple à référencer telle que exportée depuis MySQL :

```
--
-- Structure de la table 'articles'
--

CREATE TABLE 'articles' (
  'id' int(5) NOT NULL auto_increment,
  'titre' varchar(255) NOT NULL,
  'url' varchar(255) NOT NULL,
```

```
  'chapeau' text NOT NULL,
  'texte' text NOT NULL,
  'date-si' date NOT NULL,
  'date-online' date NOT NULL,
  'date-obsolete' date NOT NULL,
  'contributeur' int(5) NOT NULL default '1',
  'source' varchar(255) NOT NULL,
  'poids' int(11) NOT NULL default '0',
  'region' int(5) NOT NULL default '1',
  'title' varchar(255) default NULL
         COMMENT 'title de la page article zoomé',
  'description' varchar(255) default NULL
         COMMENT 'meta description page zoom',
  'expression_clef' varchar(120) NOT NULL,
  PRIMARY KEY ('id'),
  KEY 'expression_clef' ('expression_clef')
) ENGINE=MyISAM
         DEFAULT CHARSET=latin1 AUTO_INCREMENT=409;
```

Nous y retrouvons nos champs importants et quelques autres utiles pour la logique du site exemple (poids, date-si, etc.)

- `id` : la clé unique de chaque enregistrement de cette table. Classique.
- `titre` : le titre de l'article (ne pas confondre avec le `title` de la page). Il est destiné à être affiché entre balises `H1` si ce titre contient des mots-clés. *Titre* est destiné à un affichage pour les internautes. Il doit être en ligne, cohérent avec le chapeau et le texte du même article.
- `url` : le nom de cette article, composé de mots-clés impérativement et avec une règle d'écriture pour le contenu du champ : pas de majuscule, remplacement du caractère espace par le caractère tiret (le – de la touche 6), pas de caractère accentué. On utilise uniquement les caractères admis dans les URL. Deux possibilités pour obtenir le respect de cette règle : on fait attention lors de la saisie ou un script parcourt en batch cette colonne et convertit en bons caractères tout ce qui sort de cette règle. PHP offre de multiples fonctions permettant de manipuler des strings afin de les rendre compatibles avec Internet : passer de *upper* à *lower case*, éliminer tout *slash antislash*, *quote* et guillemet, etc. Voir les manuels PHP.

  `URL` doit être unique. Tout caractère non admis conduira cette page en erreur 404.
- `chapeau` : résumé de texte. Le texte du chapeau est destiné à être affiché dans les pages présentant des résumés. Ainsi on évite le *duplicate content* avec la page affichant le texte complet. Si le chapeau était juste le début du texte, on aurait des duplicate contents multiples et préjudiciables au positionnement du site.
- `texte` : le texte complet de l'article.

- `date-xxx` : les différentes dates ; importation dans le SI (système d'informations), mise en ligne, date de retrait automatique. Ces champs sont présents pour la logique métier de ce site exemple. Ils n'offrent aucun intérêt en référencement.
- `contributeur, source, poids` : Ces champs sont présents pour la logique métier de ce site exemple.
- `region` : un article peut être propre à une région de France ou à toute la France. A priori, cela ressemble à un champ présent pour la logique métier. Oui, mais son existence va poser des problèmes de référencement qu'il nous faudra résoudre.
- `title` : title de cette page. Rappel : il doit contenir des mots-clés. La logique de référencement va s'appuyer sur ce champ.
- `description` : meta description de cette page. Un automate peut dupliquer le contenu du title dans ce champ si vous ne disposez pas du temps et des ressources pour le gérer dans un premier temps.
- `expression_clef` : contient une expression composée de un à quelques mots-clés. Cette expression clé doit être la plus stratégique de cette page. Lors de la conception de la mise en page codée en PHP, cette donnée pourra servir à plusieurs reprises pour optimiser la page face à Google.

## Table rubrique

La table *rubrique* liste les rubriques du site :

```
--
-- Structure de la table 'rubrique'
--

CREATE TABLE 'rubrique' (
  'id' int(5) NOT NULL,
  'nom' varchar(255) NOT NULL,
  'url' varchar(255) NOT NULL,
  'title' varchar(255) default NULL
        COMMENT 'title de la page article.php',
  'description' varchar(255) default NULL
        COMMENT 'meta description de la page article.php',
  PRIMARY KEY ('id')
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

- `id` : la clé unique de chaque enregistrement de cette table. Classique.
- `nom` : le nom de la rubrique. Ce nom pourrait servir à générer le menu de navigation. Dans notre exemple, nous nous contenterons d'un menu codé à main et inséré dans un script PHP. L'objectif de ce livre n'est pas un cours sur l'optimisation du codage PHP d'un site web et sa maintenance de contenu.

- `url` : le nom de cette rubrique avec une règle d'écriture pour le contenu du champ : pas de majuscule, remplacement du caractère espace par le caractère tiret (le – de la touche 6), pas de caractère accentué. On utilise uniquement les caractères admis dans les URL. Deux possibilités pour obtenir le respect de cette règle : on fait attention lors de la saisie ou un script parcourt en batch cette colonne et convertit en bons caractères tout ce qui sort de cette règle. PHP offre de multiples fonctions permettant de manipuler des strings afin de les rendre compatibles avec Internet : passer de *upper* à *lower case*, éliminer tout slash, antislash, quote et guillemet, etc. Voir les manuels PHP.
- `URL` doit être unique. Tout caractère non admis entraînera une erreur 404 si on utilise ce champ.
- `title` : décrit très brièvement les contenus types des enregistrements de cette table. Ce contenu est utilisable pour construire le title d'une page listant des articles d'une même rubrique.
- `description` : description plus longue des contenus types des enregistrements de cette table. Ce contenu est utilisable pour construire une meta description d'une page listant des articles d'une même rubrique.

## Table article_rubrique

| Champ | Type | Interclassement | Attributs | Null | Défaut | Extra | Action |
|---|---|---|---|---|---|---|---|
| id | int(5) | | | Non | | auto_increment | |
| titre | varchar(255) | latin1_swedish_ci | | Non | | | |
| url | varchar(255) | latin1_swedish_ci | | Non | | | |
| chapeau | text | latin1_swedish_ci | | Non | | | |
| texte | text | latin1_swedish_ci | | Non | | | |
| date-si | date | | | Non | | | |
| date-online | date | | | Non | | | |
| date-obsolete | date | | | Non | | | |
| contributeur | int(5) | | | Non | 1 | | |
| source | varchar(255) | latin1_swedish_ci | | Non | | | |
| poids | int(11) | | | Non | 0 | | |
| region | int(5) | | | Non | 1 | | |
| title | varchar(255) | latin1_swedish_ci | | Oui | *NULL* | | |
| description | varchar(255) | latin1_swedish_ci | | Oui | *NULL* | | |
| expression_clef | varchar(120) | latin1_swedish_ci | | Non | | | |

► Fig. 2.2 : PhpMyAdmin de la table articles

Un article peut appartenir à plusieurs rubriques. La table *article_rubrique* liste les couples (`id` article , `id` rubrique). Cela va nous entraîner dans des *duplicate contents* si on ne fait pas attention lors du codage :

```
--
-- Structure de la table 'article_rubrique'
--

CREATE TABLE 'article_rubrique' (
  'id_article' int(5) NOT NULL,
  'id_rubrique' int(5) NOT NULL,
  'primaire' tinyint(4) NOT NULL default '0'
          COMMENT 'UN : rubrique principale',
  PRIMARY KEY ('id_article','id_rubrique')
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

- `id_article` : l'identifiant de l'article.
- `id_rubrique` : l'identifiant de la rubrique. Le même article peut appartenir à autant de rubriques que nécessaire.
- `primaire` : ce champ par défaut à 0 indique qui est la rubrique principale dont dépend un article. Il existe une seule rubrique primaire par article. Ce champ sera crucial pour les algorithmes en charge :
  - D'éviter les duplicate contents.
  - De gérer les fils de navigation sans erreur. Nous n'abordons pas la gestion d'un fil de navigation dans ce livre. C'est hors sujet et ceci ne présente aucun intérêt en référencement.

### Table region

La table *region* liste les codes et régions de France :

```
--
-- Structure de la table 'region'
--

CREATE TABLE 'region' (
  'id' int(4) NOT NULL auto_increment,
  'nom' varchar(255) default NULL,
  'acronyme' varchar(255) default NULL,
  PRIMARY KEY ('id')
) ENGINE=MyISAM DEFAULT CHARSET=latin1
    COMMENT='liste des régions françaises' AUTO_INCREMENT=89;

--
-- Contenu de la table 'region'
--
```

```
INSERT INTO 'region' ('id', 'nom', 'acronyme') VALUES
(1, 'France', 'France'),
(2, 'Alsace', 'Alsace'),
(3, 'Aquitaine', 'Aquitaine'),
(4, 'Auvergne', 'Auvergne'),
(5, 'Basse Normandie', 'B Normandie'),
(6, 'Bourgogne', 'Bourgogne'),
(7, 'Bretagne', 'Bretagne'),
(8, 'Centre', 'Centre'),
(9, 'Champagne-Ardenne', 'Champagne'),
(10, 'Corse', 'Corse'),
(11, 'Départements d''Outre-Mer', 'DOM'),
(12, 'Franche-Comté', 'Franche-Comté'),
(13, 'Haute-Normandie', 'H Normandie'),
(14, 'Ile-de-France', 'IdF'),
(15, 'Languedoc-Roussillon', 'Languedoc'),
(16, 'Limousin', 'Limousin'),
(17, 'Lorraine', 'Lorraine'),
(18, 'Midi-Pyrénées', 'Mi-Pyrénées'),
(19, 'Nord-Pas-de-Calais', 'Nord-PdC'),
(20, 'Pays de la Loire', 'Loire'),
(21, 'Picardie', 'Picardie'),
(22, 'Poitou-Charentes', 'Poitou-Ch'),
(23, 'Provence-Alpes-Côte-d''Azur', 'PACA'),
(24, 'Rhône-Alpes', 'Rhône-Alpes'),
(25, 'Territoires d''Outre-Mer', 'TOM');
```

- `id` : la clé unique de chaque enregistrement de cette table. Classique.
- `nom` : le nom de la région.
- `acronyme` : la version simplifiée du nom de la région.

## Importer mots-clés et données

Il faut disposer de données dans les différentes tables.

Ces données doivent respecter la procédure d'identification des mots-clés décidée pour ce site exemple :

- Le `title` de page contient un descriptif de 10 mots et doit être composé avec les mots et expressions clés stratégiques affectés à cette page.
- `Expression_clef` contient l'expression clé stratégique de cette page. Cette expression peut être composée de quelques mots-clés. Elle sera placée automatiquement entre balises `H1` quelque part sur la page.

Ces données pourront être saisies manuellement via *PhpMyAdmin* ou via une importation CSV vers chaque table.

### Résumé et étape suivante

Nous venons de détailler la base de données MySQL, les tables et leurs structures.

Ces tables seront utilisées pour générer des pages HTML dynamiques avec une optimisation native en référencement sur Google.

Nous avons les données et les structures. Pour passer aux algorithmes, il nous faut d'abord revisiter les principes du référencement naturel. Ayez en mémoire les termes et concepts du chapitre 1, *Optimiser des pages pour Google*, cela sera plus facile.

## 2.5 Référencement, positionnement naturel : principes

⚠ Le référencement naturel d'un site Internet

Ce sont les actions techniques et rédactionnelles pour optimiser les pages du site afin de tenter de les faire afficher dans les SERP d'un moteur sur un ou des mots-clés précis.

*Optimiser une page HTML* signifie mettre en œuvre des techniques Internet et rédactionnelles sur cette page afin d'en faciliter la compréhension des contenus par le moteur de recherche.

L'objectif est de le pousser à prendre en compte les mots-clés dont le site estime avoir besoin en priorité sur cette page. Quand on définit que telle page sera la landing page, la page d'atterrissage, sur telle expression clé, il faut tout mettre en œuvre pour pousser en avant cette page afin de la *positionner* naturellement vers le top5 dans les SERP de Google :

- Optimisation de la page, objet du présent chapitre.
- Optimisation des autres pages pour mettre celle-ci en avant via la technique des nuages de liens. Le principe de cette approche est décrit un peu plus loin dans le chapitre consacré aux backlinks.
- *Netlinking*. Cet aspect web marketing est détaillé dans un autre livre, chez MicroApplication : *Le guide complet du référencement sur Google*. Dans le chapitre consacré aux backlinks du présent ouvrage, nous effleurerons le sujet afin de nous consacrer à l'aspect codage destiné à faciliter le netlinking et sa variante interne au site, les nuages de liens (tags clouds).

⚠ Positionnement d'un site Internet

Ce sont les actions qui vont hisser les pages d'un site dont le référencement est correct, vers les premières positions dans une page de résultats de moteur (une SERP) sur les mots-clés voulus par le webmestre ou le traffic manager.

Sur chaque page de tout site web accessible à Google, différents critères, que nous détaillerons plus loin, sont évalués par Google. Ils lui permettent de trier dans son index les millions de pages présentant un mot-clé en commun.

Ensuite Google évalue les backlinks pointant vers chaque page d'un site. Cette évaluation finalise pour quelques heures le classement dans l'index. Ce classement est régulièrement recalculé. Le fonctionnement des algorithmes lisse les montées ou descentes de pages au sein de l'index.

*Réussir un positionnement* des *landing pages* de son site dans les SERP de Google exige de réaliser un travail séquentiel :

- Un référencement de qualité, c'est-à-dire les travaux d'optimisation de chaque page du site respectant une cohérence vis-à-vis des mots-clés utilisés et un travail d'optimisation globale de toutes les pages entre elles.
- Ensuite et seulement ensuite, un travail de `netlinking` interne (nuages de liens) et externe pourra être mis en œuvre.

## Contenu et mots-clés

Pour avoir une page de site positionnée en bonne place, il faut que les pages de ce site disposent de réels contenus : textes, visuels, vidéo, etc. créant de la valeur ajoutée vis-à-vis de l'internaute lecteur.

Les volumes, en nombre de mots par page et en nombre de pages, sont importants pour la qualité du référencement. Une fois optimisés, une fois liés ensemble par des nuages de liens, ces pages et ces textes influeront fortement sur le référencement naturel du site et sur son positionnement dans l'index du moteur.

Le web marketing aura à sa charge de sélectionner les mots-clés et la répartition de ceux-ci sur les pages du site.

## Accessibilité des mots-clés

L'accessibilité d'un mot-clé se décompose en deux types :

- *technique* : Le robot doit pouvoir accéder à la page et à son contenu. En effet, de nombreux sites ont des JavaScripts et autres gadgets techniques (flash) qui bloquent plus ou moins les bots de Google.
- *sémantique* : Rendre accessible au moteur les mots et expressions clés retenus lors de l'étape précédente revient à optimiser les pages du site. Pour cela, le travail est découpé en plusieurs étapes :
  - répartir les mots-clés et les combinaisons de mots et expressions clés sur les pages du site ;
  - considérer un maximum de 5 mots-clés par page (nous verrons le pourquoi de cette limitation plus loin) ;
  - optimiser en conséquence chaque page du site autour des mots-clés qui lui sont affectés.

Sauf erreur de votre part, le moteur prendra en compte vos 3 à 5 mots-clés stratégiques pour chaque page. L'accessibilité sémantique a ainsi pour objectif d'orienter le choix de Google sur votre choix de mots-clés pour chacune des pages de votre site web, les landing pages comme les pages en charge de pousser les landing pages en avant. En cas d'erreur d'optimisation, Google choisira vos mots-clés.

### Une stratégie de référencement

Le marketing décide d'une stratégie de référencement : mots-clés, pages d'atterrissage…

Puis au fil de l'acquisition de statistiques, de retours sur l'utilisation du site, des changements sont apportés à cette stratégie :

- De nouveaux mots ou expressions clés sont à prendre en compte.
- De nouvelles pages sont ajoutées, de nouvelles rubriques sont créées.
- De nouvelles landing pages sont définies pour les nouveaux mots et expressions clés.

Il est du ressort du codeur de fournir un site pouvant aisément mettre à jour sa stratégie de mots-clés et de pages d'atterrissage. Nous étudierons cela plus loin.

## 2.6 Publication de pages dynamiques

Pour programmer une optimisation de page, encore faut-il publier les pages de manière dynamique.

Nous devons disposer d'un "moteur" de publication. Nous l'appellerons *logiciel de publication* pour éviter les confusions avec le terme moteur (de recherche). Ce noyau de code est chargé d'afficher des pages avec des informations en base de données :

- sur clic de l'internaute sur un lien en dur (sinon un bot de Google ne pourra pas suivre le lien) ;
- dans des gabarits d'affichage incluant une navigation par menu.

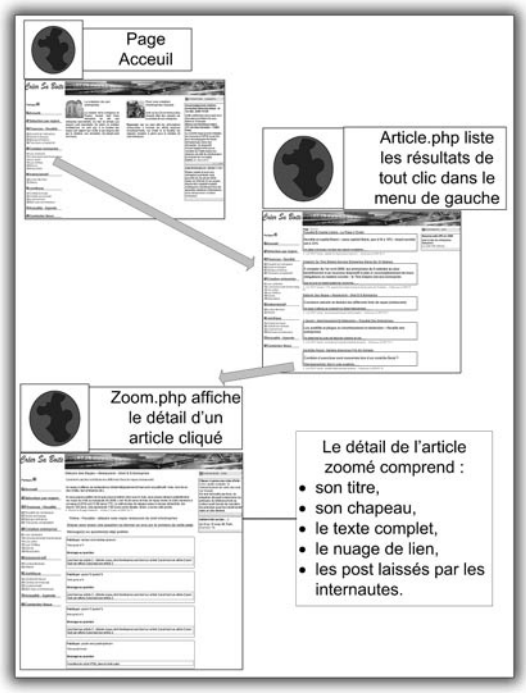



Fig. 2.3 : Logique du logiciel de publication de contenus référençables sur Google en natif.

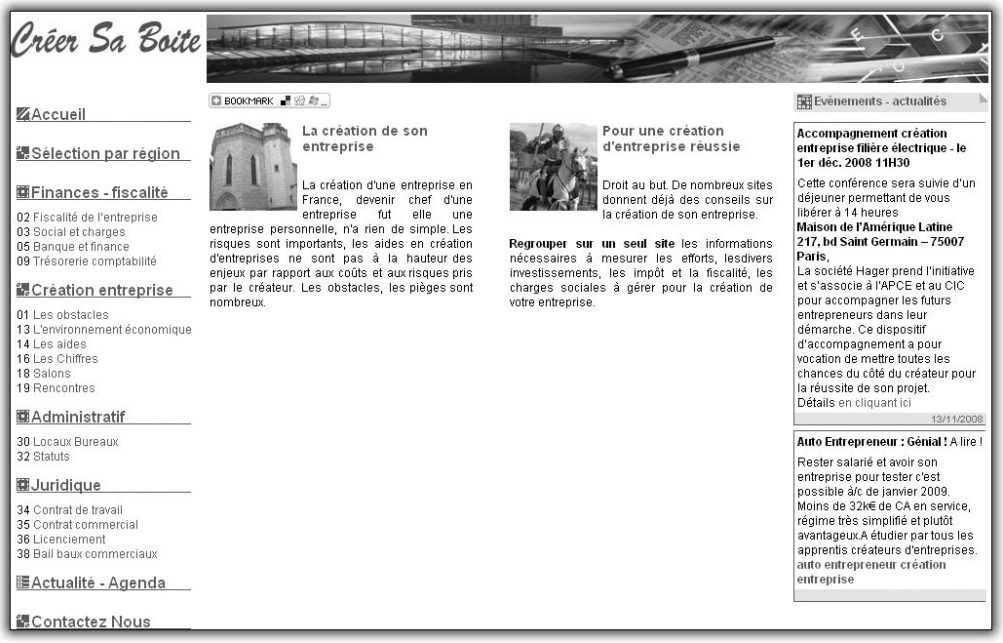
Ce livre ne porte pas sur la programmation PHP/MySQL du meilleur *logiciel de publication* du monde. Le code source utilisé dans notre exemple a donc été fortement simplifié pour permettre au lecteur de mieux appréhender les techniques de référencement naturel automatisées en PHP dans ce petit logiciel.

Par exemple, nous avons, entre autres, supprimé de ce logiciel de publication la gestion du fil de navigation, nous avons simplifié la mise en page au maximum et avons utilisé une charte graphique des plus simples. Ces éléments n'apportent rien à la compréhension de ce livre et alourdissaient trop les codes sources.

## Présentation du site publié

Ce mini site utilise le logiciel de publication.

Pour commencer, une page d'accueil est nécessaire : *index.php*.




► Fig. 2.4 : Page d'accueil du mini site utilisant le logiciel de publication

La page *index.php* est codée spécifiquement et à part. Elle n'est pas affichée par le logiciel de publication. Elle possède un menu, installé à gauche de la fenêtre dans un souci de simplicité :

```
<div id='menu_vertical'>
 <ul>
   <li class='menutitrered materiel'>
       <a href='index.php'>Accueil</a>
   </li>
 </ul>
 <ul>
   <li class='menutitrered societe'
       title='Filtre des résultats par région'>
         <a href='region.php' rel='nofollow'>
            Sélection par région
         </a>
   </li>
 </ul>
 <ul>
   <li class='menutitrered logiciel'>
     <a href='#'>Finances - fiscalité</a>
   </li>
```

```
    <li><span>02 </span>
     <a
    href='finances-fiscalite-entreprise-diminuer-deduction-exoneration
    ➥ -impots-taxes/rubrique/2/1.html'
        title='02 - Fiscalité de l'entreprise'>
          Fiscalité de l'entreprise
      </a>
    </li>
    <li><span>03 </span>
     <a href='Charges-sociales-diminuer-deduction-exoneration-entreprise/
    ➥ rubrique/3/1.html'
        title='03 - Social et charges'>Social et charges
     </a>
   </li>
   <li><span>05 </span>
     <a href='Banques-finances-creation-entreprise-expert-comptable-
    ➥ comptabilite/rubrique/5/1.html'
        title='05 - Banque finance'>Banque et finance</a>
   </li>
   <li><span>09 </span>
     <a href='Tresorerie-gestion-entreprise-comptable-comptabilite-expert/
    ➥ rubrique/9/1.html'
       title='09 - Trésorerie comptabilité entreprise'>
          Trésorerie comptabilité
     </a>
   </li>
  </ul>
  <ul>
   <li class='menutitrered societe'>
     <a href='#'>Création entreprise</a>
   </li>
   <li><span>01 </span>
     <a href='obstacles-a-la-creation-entreprise-societe/rubrique
    ➥ /1/1.html'
        title='01 - Les obstacles'>Les obstacles</a>
   </li>
   <li><span>13 </span>
    <a href='Environnement-economique-entreprise-economie-creation-
   ➥ entreprises/rubrique/13/1.html'>L'environnement économique</a>
   </li>
   <li><span>14 </span>
    <a href='aides-financieres-creation-entreprises/rubrique/14/1
   ➥ .html'>Les aides</a>
   </li>
   <li><span>16 </span>
     <a href='Les-chiffres-economie-creation-reprise-entreprise/
    ➥ rubrique/16/1.html'>Les Chiffres</a>
   </li>
```

```
    <li><span>18 </span>
      <a href='Salons-expositions-creation-reprise-entreprises
    ➥ /rubrique/18/1.html'>Salons</a>
    </li>
    <li><span>19 </span>
      <a href='Rencontre-createurs-entreprises-idees-echanges-forum
    ➥ -coaching-partage-experience/rubrique/19/1.html'>Rencontres</a>
    </li>
  </ul>
  <ul>
    <li class='menutitrered logiciel'>
      <a href='#'>Administratif</a>
    </li>
    <li><span>30 </span>
      <a href='Locaux-bureaux-hebergement-domiciliation-entreprise
    ➥ -societe-loyer-bail-immobilier/rubrique/30/1.html' >Locaux
    ➥ Bureaux</a>
    </li>
    <li><span>32 </span>
      <a href='Statut-juridique-risques-entreprise-legal/rubrique
    ➥ /32/1.html'>Statuts</a>
    </li>
  </ul>
  <ul>
    <li class='menutitrered logiciel'>
     <a href='#'>Juridique</a>
    </li>
    <li><span>34 </span>
      <a href='Contrat-de-travail-entreprise-salarie-collaborateur
    ➥ -rupture-clause/rubrique/34/1.html' >Contrat de travail</a>
    </li>
    <li><span>35 </span>
      <a href='Contrat-commercial-devis-obligations-factures
    ➥ -facturation-reception-travaux-recette-different-vice-cache
    ➥ -garantie-avenant/rubrique/35/1.html' >Contrat commercial</a>
    </li>
    <li><span>36 </span><a
  ➥ href='Licenciement-salarie-collaborateur-contrat-travail-entreprise
  ➥ /rubrique/36/1.html' >Licenciement</a>
    </li>
    <li><span>38 </span><a
   href='Bail-baux-entreprise-commercial-domiciliation-location-bureaux
  ➥ -immobilier-loyer/rubrique/38/1.html' >Bail baux commerciaux</a>
    </li>
  </ul>
  <ul>
    <li class='menutitrered temoignage'>
      <a href='Actualites-agenda-createurs-creation-reprise-entreprises
    ➥ /rubrique/99/1.html'>Actualité - Agenda</a>
```

```
    </li>
  </ul>
  <ul>
    <li class='menutitrered societe'>
      <a href='contact-dvv2.php'
         rel='nofollow'>Contactez Nous</a>
    </li>
  </ul>

    </div> <!-- fin menu vertical -->
```

Ce menu est présenté en code source HTML associé à une feuille de style. Il est très simple à comprendre.

`<div id='menu_vertical'>` indique le conteneur décrit en feuille de style. Ce conteneur définit une colonne à gauche de l'écran où le contenu du menu est affiché.

`<li class='menutitrered materiel'><a href='index.php'>Accueil</a></li>` affiche une entrée principale du menu en l'associant à un pictogramme, `materiel`. Tout ceci est défini en feuille de style. Voici l'extrait de la feuille de style impliquée dans cette gestion de pictogramme :

```
#menu_vertical ul{
  padding:15px 0 0 0;
  margin:0;
  margin-left:4px ;
  list-style:none;
}
#menu_vertical li a{
padding:0;
margin:0;
color:#2C6A93;
background-color:inherit;
text-decoration:none;
}
#menu_vertical li a:hover{
font-weight:bold;
text-decoration:underline;
color:#ffbe33;
background-color:inherit;
}
.menutitrered a{
display: block;
background-repeat:no-repeat;
height:25px;
width:180px;
```

```
background-position:left top;
font-weight:bold;
font-size:16px;
text-indent:15px;
padding:5px 0 0 0;
color:#2C6A93!important;
background-color:#FFFFFF;
line-height:0.9em
}
.menutitrered a:hover{
text-decoration:none !important;
color:#ffbe33;
background-color:#FFFFFF;
}
.solutions a{
background-image:url(../artpackage/solutions1.gif);
background-repeat:no-repeat;
}
.societe a{ background-image: url(../artpackage/societe1.gif);
background-repeat:no-repeat;
}
.temoignage a{
background-image:url(../artpackage/temoignage1.gif);
background-repeat:no-repeat;
}
.logiciel a{background-image:url(../artpackage/logiciel1.gif);
background-repeat:no-repeat;
}
.materiel a{background-image:url(../artpackage/materiel1.gif);
background-repeat:no-repeat;
}
.services a{background-image:url(../artpackage/services1.gif);
background-repeat:no-repeat;
}
.solutions a:hover {
background-image:url(../artpackage/solutions2.gif);
background-repeat:no-repeat;
color:#ffbe33!important;
background-color:#FFFFFF;
}
.societe a:hover {
background-image: url(../artpackage/societe2.gif);
background-repeat:no-repeat;
color:#ffbe33!important;
background-color:#FFFFFF;
}
.temoignage a:hover {
```

```
background-image:url(../artpackage/temoignage2.gif);
background-repeat:no-repeat;
color:#ffbe33!important;
background-color:#FFFFFF;
}
.logiciel a:hover {
background-image:url(../artpackage/logiciel2.gif);
background-repeat:no-repeat;
color:#ffbe33!important;
background-color:#FFFFFF;
}
.materiel a:hover {
background-image:url(../artpackage/materiel2.gif);
background-repeat:no-repeat;
color:#ffbe33!important;
background-color:#FFFFFF;
}
.services a:hover {
background-image:url(../artpackage/services2.gif);
background-repeat:no-repeat;
color:#ffbe33!important;
background-color:#FFFFFF;
}
```

Si on clique sur cette entrée *Accueil*, le lien activé est : *index.php*, la page d'accueil. Aucune importance si vous ne comprenez pas ce codage. Vous pourrez le réécrire ou l'adapter à vos habitudes. Pour information, ce code HTML appelle une classe stockée en feuille de style, `menutirered`, qui prend un argument, `materiel`. Il y a alors affichage du pictogramme appelé `materiel` suivi du texte `Accueil` avec un lien vers *index.php*. Rien de bien intéressant pour le moment.

```
<li><span>02 </span><a href=' finances-fiscalite-entreprise-diminuer-deduction
➥ -exoneration-impots- taxes/rubrique/2/1.html' title='02 - Fiscalité de
➥ l'entreprise'>Fiscalité de l'entreprise</a></li>.
```

cette ligne de code affiche :

- `02` – c'est le n° de rubrique : pratique pour déboguer.
- Un lien HTML associé à un title de lien.
- Un libellé de lien : `Fiscalité de l'entreprise`.

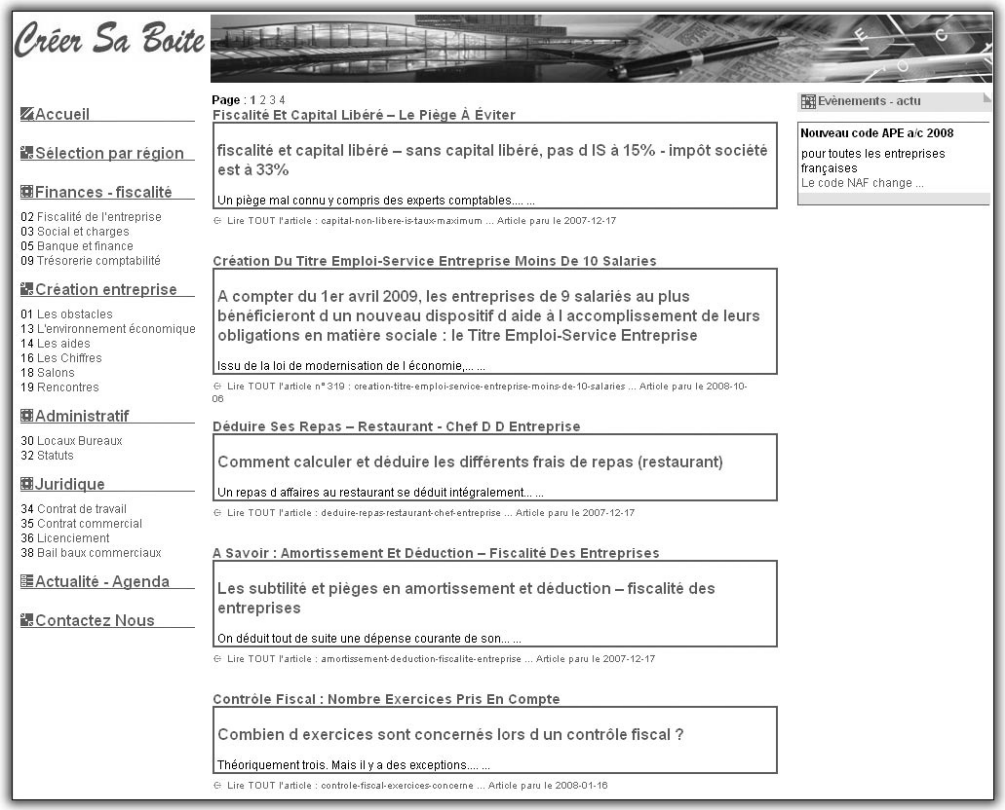
Une zone centrale et une colonne de droite sont présentes.

- La zone centrale affiche deux conteneurs de texte avec une vignette photo pour chacun.

- La colonne de droite affiche 2 conteneurs publiant des actualités ou nouveautés.

Nous venons de parcourir rapidement la page *Index.php*.

Un clic sur 02 `Fiscalité de l'entreprise` dans le menu, en haut à gauche, lance un lien qui affiche une page listant tous les articles ayant un rapport avec la fiscalité de l'entreprise.

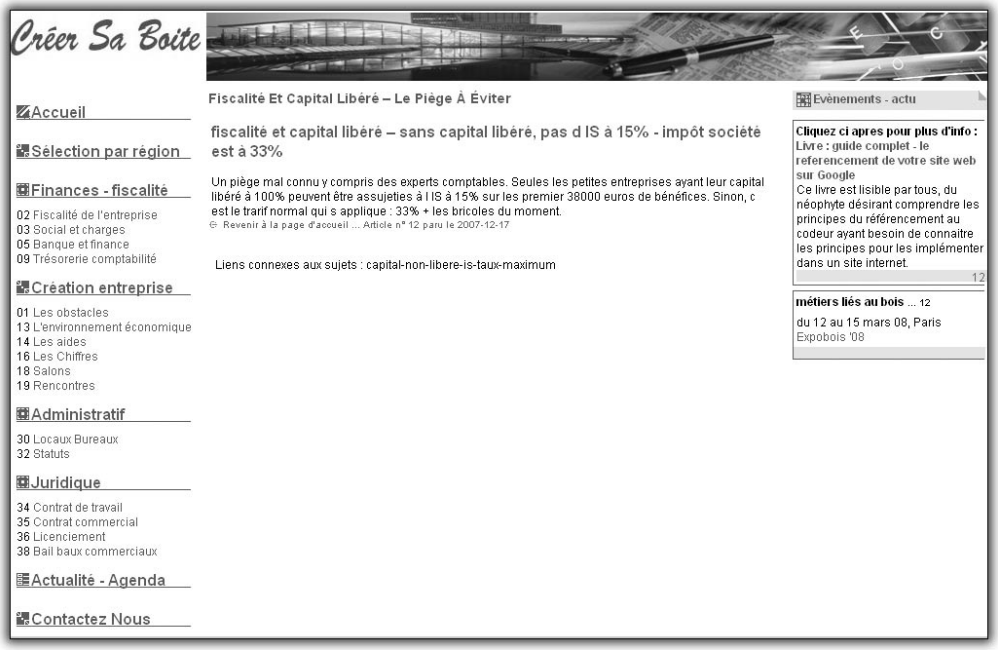



► Fig. 2.5 : Page article.php affichant tous les articles d'un même thème.

Cette page affiche, comme la page *index.php*, un gabarit d'affichage, logo+bandeau graphique+menu du haut+une colonne droite, et la liste des articles répondant à ce thème : `fiscalité de l'entreprise`.

Ces articles sont affichés 5 par 5. Il y a 4 pages de résultats. On peut naviguer dans ces pages directement via la barre de navigation avec les numéros de pages.

On peut avoir le détail complet d'un article en cliquant sur un des deux liens disponibles pour cela dans le titre ou en bas de chaque bloc article `Lire tout l'article...`




► Fig. 2.6 : Page zoom.php affichant le détail d'un article

### Logiciel de publication adapté aux besoins de Google

Ce mini site est publié par trois scripts PHP, quelques tables MySQL, quelques données, un fichier *.htaccess* pilotant l'*URL rewriting*. Nous allons ajouter de nouveaux scripts et de nouvelles tables au fil des paragraphes et de la montée en puissance du lecteur dans le domaine de l'automatisation du référencement sur Google.

Ces trois scripts sont le cœur de notre logiciel de publication :

- *index.php* est la page d'accueil du site.
- *article.php* affiche tous les articles d'une même rubrique.
- *zoom.php* affiche le contenu détaillé d'un article précis.

## Génération de pages dynamiques – article.php

Étudions le HREF du lien cliqué dans notre exemple, `02 Fiscalité de l'entreprise`, dans le menu, en haut à gauche :

```
href='finances-fiscalite-entreprise-diminuer-deduction-exoneration-
➥ impots-taxes/rubrique/2/1.html'
```

Existe-t-il une telle page sur le site ? Non. Elle est générée par le logiciel de publication.

Décomposons, en simplifiant, la réalité technique d'Internet, afin de rester lisible :

- Un lien comme celui-ci est envoyé au serveur http.
- Le serveur http, Apache, compare cette URL au contenu d'un fichier, *.htaccess*.
- Si cette URL rencontre une structure d'URL identique dans ce fichier, il y a une égalité et donc Apache déclenche une réécriture de l'URL. Le fichier *.htaccess* contient bien sûr l'URL à mettre à la place. Et là, cette URL existe réellement sur le disque du serveur http.

Notre URL est transformée par Apache en *article.php* suivi des arguments passant à ce programme PHP la demande du lien : afficher les 5 premiers résultats de tous les articles appartenant à la rubrique n°2, *finances et fiscalité des entreprises*.

Ce mécanisme, fort pratique en référencement, s'appelle l'*URL rewriting* ou en français, *réécriture à la volée des URL*. Nous reviendrons en détail sur ce mécanisme dans le paragraphe qui lui est dédié dans le présent chapitre.

Toutes les entrées du menu font appel à ce mécanisme. Nous disposons ainsi de mots-clés pertinents dans l'URL alors que la réalité est constituée de l'appel au même script PHP avec des codes passés en arguments lui indiquant quoi faire comme traitement.

Depuis la liste de tous les articles d'une rubrique, on peut, pour chaque résumé d'article, cliquer sur un lien en haut de paragraphe ou dans le bas de celui-ci.

Par exemple, pour

```
http://localhost/BookN2/V1-00/article/controle-fiscal-exercices
➥ -concerne.html&src=Fiscalite-entreprise
```

**http://localhost/BookN2/V1-00/** : représente l'URL de ma machine de développement sous WAMP. Dans un site en ligne, on obtiendrait quelque chose du genre : **http://www.monsite.com/** .

`/article` : simple signalisation. Elle indique ici que ce qui suit est un article précis. C'est totalement inutile en codage. Mais c'est fort pratique pour travailler son référencement, notamment quand on utilise les commandes Google `inurl:`.

Par exemple : Un site possède plusieurs entrées majeures : forum, annuaire, et le site lui-même.

La commande suivante, saisie dans Google, liste toutes les pages indexées du site quel que soit le répertoire où elles se trouvent :

```
site:www.monsite.com
```

On aura toutes les pages de l'annuaire **www.monsite.com/annuaire/xxxx.html**, toutes les pages du forum, **www.monsite.com/forum/xxxx.html**, etc.

Si vous désirez comptabiliser uniquement les pages offrant un seul article, il vous suffira de saisir dans Google :

```
site:www.monsite.com inurl:article
```

Vous obtiendrez la liste de tous les articles. Il n'y aura pas de mélange des genres avec forum, annuaire et d'autres fonctionnalités du site. Cette astuce est pratique quand on débogue ou qu'on étudie les réactions de Google. Cela permet aussi d'éviter des accidents, même peu probables, comme d'avoir le nommage de 2 fichiers différents avec un même nom :

- **monsite.com/forum/charges-sociales-gerant-sarl.html** : un post sur ce sujet par un internaute

versus

- **monsite.com/article/charges-sociales-gerant-sarl.html** : un article sur ce sujet par un contributeur.

Sans la stricte séparation en 2 répertoires virtuels différents, *article* et *forum*, on aurait eu quelques soucis d'écrasement de fichiers différents offrant un même nom.

**/controle-fiscal-exercices-concerne.html** : désigne l'article précis.

`&src=Fiscalite-entreprise` : `&` est un caractère de séparation dans les URL. `src=` désigne le nom de la rubrique d'où cet article a été cliqué. Cela permettra ainsi de calculer un fil de navigation représentant le réel chemin parcouru par l'internaute.

## Initiation rapide à HTACCESS

Nous avons le principe du logiciel de publication. Un clic sur un lien lisible composé de mots-clés appelle des données précises affichées dans une page HTML.

La page HTML appelée n'existe pas. Il faut la construire depuis les informations contenues dans le lien lui-même qui donnent les clés d'accès aux informations à afficher, informations stockées en base de données.

La première étape est donc très importante : la *réécriture d'URL* à la volée (*URL rewriting*) qui va permettre de transformer cette URL humaine en un appel d'un programme PHP avec des arguments en ligne. Cette étape sert d'interface, de passerelle entre une URL humaine et une URL machine.

Le pilotage de cette réécriture se déroule dans le fichier *.htaccess* (point suivi de `htaccess`) qui contient les instructions pilotant Apache dans la gestion des URL.

Ce fichier *.htaccess* utilise un ancien langage informatique appelé "expressions régulières".

### Fonctionnement simplifié de Apache

Quand un internaute clique sur un lien dans un site, ce lien est envoyé au serveur http Apache pour qu'il renvoie la page associée à cette URL à l'internaute demandeur.

Avant toute chose, pour différentes raisons de sécurité informatique et de diverses autres fonctionnalités dont l'URL rewriting, Apache compare cette URL à des instructions stockées dans ce fichier *.htaccess*.

⚠ Le nom de domaine ne compte jamais

Le nom de domaine n'est jamais concerné par une URL rewriting. La réécriture d'URL démarre toujours après le slash séparant le TLD de la suite. Un TLD est le *.com* ou *.fr* d'un nom de domaine.

*Monsite.com/*`xxxxxx` : `xxxxx` est l'URL soumise à réécriture par Apache via *.htaccess*.

### Première réécriture d'URL à la volée

Voici un premier code source codé dans le fichier *.htaccess* en `expressions régulières`:

```
RewriteEngine on
RewriteRule
   ^([-_a-zA-Z0-9]+)/
    ([-_a-zA-Z0-9]+)/([-_a-zA-Z0-9]+)/([0-9]+).html$
    article.php?referencement=$1&$2=$3&page=$4
```

`RewriteEngine on` : la réécriture d'URL est à utiliser si une instruction qui suit celle-ci le demande.

`RewriteRule` : c'est parti… Cela indique une règle de réécriture d'URL.

`RewriteRule terme1 terme2` : si l'URL soumise à Apache correspond au `terme 1`, l'URL réécrite du `terme 2` est utilisée par Apache en lieu et place de l'URL soumise.

`^ xxxxxx $` : tout ce qui est entre le `^` et le `$` est une expression à évaluer, le `terme 1` de la règle de réécriture. Si elle est juste, la réécriture d'URL sera lancée par Apache. Sinon, il passe à la ligne suivante à traiter dans le fichier *.htaccess*.

`([-_a-zA-Z0-9]+)` : toute chaîne de caractères dont chacun de ceux-ci appartient à un des intervalles suivants :

- de a à z minuscule ;
- de A à Z majuscule ;
- de 0 à 9 ;
- soit – (le signe moins ou tiret), soit _ (le caractère underscore, souligné).

L'expression (ce qui est entre crochets `[` et `]` ) est encadrée par une parenthèse ouvrante `(` et par un plus `+` suivi d'une parenthèse fermante `+)` . Cela indique à Apache que la chaîne de caractères peut avoir n'importe quelle longueur.

Si un caractère n'appartient à aucun de ces intervalles (par exemple le caractère point **.**), l'évaluation de l'expression régulière est fausse et Apache passe à la ligne suivant du fichier *.htaccess*.

`^ ([-_a-zA-Z0-9]+)/` : notre expression régulière dont le comportement est décrit précédemment est encadrée par un chapeau `^` et par un slash **/**.

Le chapeau `^` désigne comme déjà indiqué précédemment, le début du terme 1 (tout ce qui est encadré par `^` et `$`) de notre `rewriterule`.

Le slash désigne le caractère **/** utilisé pour séparer les répertoires dans les URL : **www.monsite.com/vins-de-France/champagnes/**, etc.

Apache va comparer l'URL soumise par tout internaute lors d'un clic à cette sous-expression régulière. Est-ce que le début de l'URL soumise est composée de tout caractère des intervalles `[-_a-zA-Z0-9]` , peu importe le nombre de caractères (ce que désigne le `+` ) et qu'un caractère slash `/` marque cette première partie d'URL soumise ?

Si oui, Apache continue d'évaluer le premier terme de la règle de réécriture.

Revoyons ce premier terme de la règle de réécriture maintenant que nous avons détaillé une première expression régulière :

```
^([-_a-zA-Z0-9]+)/([-_a-zA-Z0-9]+)/([-_a-zA-Z0-9]+)/([0-9]+).html$
```

Apache cherche la correspondance entre toute URL soumise et ce terme composé d'expressions régulières :

Toute URL découpée par trois / avant un *.html*, ce qui nous donne quatre tronçons composant la totalité de l'URL soumise.


► Fig. 2.7 : Une URL en .html découpée par trois / compatible avec la condition du .htaccess

Chaque tronçon doit être composé de caractères appartenant à un des intervalles `[-_a-zA-Z0-9]`.

Si l'URL soumise correspond à ces critères, le `terme 2` est utilisé comme URL en lieu et place de l'URL soumise.

Examinons le `terme 2` de notre *règle de réécriture* :

```
article.php?referencement=$1&$2=$3&page=$4
```

*article.php* : le script PHP appelé par Apache. Autant la page désignée dans l'URL soumise n'existe pas, autant *article.php* est présent dans le répertoire de notre site Internet géré par Apache. C'est un script PHP, programme informatique qui accepte des arguments en entrée, des passages de paramètres essentiels à son bon fonctionnement.

`referencement=$1` : la variable `referencement` reçoit le premier tronçon de notre `terme 1`.

`$_GET["referencement"]` dans un script PHP nous permettra de récupérer la valeur de ce paramètre.

Par exemple :

```
$referencement = $_GET["referencement"];
```

`&$2=$3` : `$2` a une valeur, disons que ce tronçon n°2 sur 4 de l'URL soumise est */rubrique/*.

Apache crée une variable ayant pour nom le texte de notre deuxième tronçon d'URL. Dans notre exemple, ce sera `rubrique`. La variable `rubrique` reçoit le troisième tronçon de notre `terme 1`.

`&page=$4` : la variable `page` reçoit le quatrième tronçon de notre `terme 1`.

Exemple complet :

**http://www.monsite.com/fiscalite-entreprise-SARL/rubrique/1/1.html** : URL soumise à Apache.

Le nom de domaine n'est jamais concerné par une réécriture d'URL. on dispose donc de ceci :

**fiscalite-entreprise-SARL/rubrique/1/1.html** : Il n'y a aucun slash / au début de cette chaîne de caractères.

```
$1 = fiscalite-entreprise-SARL
$2 = rubrique
$3 = 1
$4 = 1
```

Comparaisons des 4 tronçons positives: réécriture d'URL à faire. On aura :

**article.php?referencement=texte-referencement1&rubrique=1&page=1**

## Deuxième réécriture d'URL à la volée

La première règle de réécriture d'URL nous permet d'afficher toute liste d'articles appartenant à une même rubrique.

La deuxième règle de réécriture d'URL va nous permettre d'afficher le détail d'un article précis.

Voici la règle de `réécriture d'URL` en charge de cela dans notre fichier *.htaccess*.

```
RewriteRule
 ^([-_a-zA-Z0-9]+)/([-_a-zA-Z0-9]+).html&src=([-_a-zA-Z0-9]+)$
  zoom.php?referencement=$1&url=$2&src=$3
```

Grâce à l'expérience acquise lors de l'examen détaillé de notre première règle de réécriture, nous allons pouvoir aller plus vite.


► Fig. 2.8 : Une URL en .html&src= découpée par un / compatible avec la 2ème condition du .htaccess

Si l'URL soumise à Apache est de la forme `chaîne de caractères1/chaîne de caractères2.html&src=chaîne de caractères3`, l'URL utilisée par Apache sera :

```
Zoom.php?referencement=chaîne de caractères1&url= chaîne de
➥ caractères2&src=chaîne de caractères3
```

### Résumé sur URL rewriting

Vous venez de voir une règle de réécriture d'URL avec quelques points intéressants :

- Comment on passe d'une URL humaine, visible par les internautes dont Google, à une autre, inadéquate pour le référencement mais totalement adaptée aux besoins des codeurs de sites web dynamiques via la programmation dans le fichier *.htaccess*.
- On peut générer une variable, dans notre exemple, `$2=$3`, génération de la variable `rubrique`. On peut initialiser cette variable juste créée, ici on l'initialise à `1`, et la faire prendre en compte comme argument passé à un script PHP. Ici *article .php* prendra en compte `rubrique=1` et la logique interne de l'application réagira en conséquence en affichant tous les articles dépendant de cette rubrique.

Ce paragraphe ne se veut pas une formation sur *.htaccess* de Apache et sur la programmation des expressions régulières. Il existe de nombreux tutoriaux qui détaillent *.htaccess* bien plus que ce livre. L'objectif de ce paragraphe est de vous initier afin que vous puissiez comprendre et utiliser notre logiciel de publication de pages 100 % accessible aux bots de Google.

## Lister tous les articles d'une rubrique

Le script PHP *article.php* a pour mission d'afficher tous les articles d'une même rubrique. Les URL "humaines" et contenant des mots-clés lisibles par les bots de Google sont de la forme que nous venons de voir dans l'initiation à *.htaccess* et à sa programmation pour l'URL rewriting :

**http://www.monsite.com/fiscalite-entreprise-SARL/rubrique/1/1.html**

Rappel : **monsite.com** n'est jamais pris en compte dans la réécriture d'URL.

Quelques contraintes doivent être prises en compte :

- La pagination des résultats. Dans notre exemple, on affiche les articles cinq par cinq.
- Les URL générées par ce script *articles;php* doivent être compatibles avec les règles de réécriture mises en place dans *.htaccess*, que ce soit pour appeler d'autres listes d'articles ou faire un zoom sur le détail d'un article précis.

Voyons le début du code source de *article.php* :

```
<?php
require_once("php_inc/MiseEnPage.inc.php");
require_once("php_inc/init.inc.php");
require_once("php_inc/biblio_inc.php");

// ----------------------------------
// Mémorisation du texte de référencement
// pour réutiliser dans les URLs
// ----------------------------------
$referencement = "referencement";
if (isset($_GET["referencement"]))
   $referencement = $_GET["referencement"];
```

Classiquement, il y a un `require` sur les diverses bibliothèques de fonctions PHP dont on va avoir besoin dans ce script *articles.php*.

- `MiseEnPage` : les fonctions liées à la mise en page.
- `Init` : l'initialisation des accès aux tables et base de données.
- `Biblio` : bibliothèque contenant des fonctions diverses et variées que nous allons utiliser au fil des pages.
- `$referencement = "referencement";` : vieille habitude, nous préférons toujours instancier une variable à une valeur avant de la manipuler avec des tests. Ainsi, en mode Debug, on aura un référentiel : à l'origine, cette valeur valait "ceci".
- `if (isset($_GET["referencement"]))` : Si l'argument `referencement` existe et qu'il contienne une valeur, on le récupère dans la ligne de code suivante.
- `$referencement = $_GET["referencement"];` : on a la chaîne de caractères `referencement` issue de l'URL rewriting.

Voyons la suite du code source de *article.php*.

Ce bloc de code a pour objectif de construire la requête SQL qui va récupérer la liste de tous les articles correspondant au critère de tri indiqué dans le lien. Dans cette maquette de site, nous avons deux possibilités :

- `rubrique`, critère que nous avons déjà évoqué auparavant ;
- `region` que nous évoquerons plus loin.

```
// ----------------------------------
// On recupere l'information de tri en parametre
// Les attributs de tri sont :
// rubrique, region
// ----------------------------------
```

```
// en sortie de cette sequence de if :
// on doit avoir $sql qui content la requete construite
// pour lister les articles correspondants au(x) critere(s)


if (isset($_GET["rubrique"]))
   {
   $parametre = "rubrique";
   $valeur = $_GET["rubrique"];
   $rubrique = $valeur ;
   $sql = "SELECT *
      FROM 'articles', 'article_rubrique'
      WHERE article_rubrique.id_rubrique='$valeur'
           AND article_rubrique.id_article=articles.id" . "
      ORDER BY poids DESC ";

      // on aura besoin de tout le détail de la rubrique.
   $sql_rubrique= "SELECT *
                     FROM 'rubrique'
                     WHERE rubrique.id='$valeur'" ;
   }
   elseif (isset($_GET["region"]))
      {
      $parametre = "region";
      $valeur = $_GET["region"];
      $sql = "SELECT *
                FROM 'articles'
                WHERE $parametre='$valeur'";
      }
```

- `if (isset($_GET["rubrique"]))` : si rubrique existe et contient quelque chose : `article.php?rubrique=1` correspond positivement à ce test conditionnel.
- `$parametre = "rubrique"` : La chaîne `"rubrique"` est sauvegardée dans une variable `$parametre`.
- `$valeur = $_GET["rubrique"]; $rubrique = $valeur` : Le contenu valeur est stocké dans les variables `$rubrique` et `$valeur` :

```
$sql = "SELECT *
         FROM 'articles', 'article_rubrique'
         WHERE article_rubrique.id_rubrique='$valeur'
           AND article_rubrique.id_article=articles.id" . "
         ORDER BY poids DESC ";
```

Dans cette requête `$sql`, on cherche tous les articles appartenant à la rubrique ciblée.

On aura besoin plus loin de toutes les informations sur la rubrique ciblée par le script *article.php*.

```
 // on aura besoin de tout le détail de la rubrique.
$sql_rubrique= "SELECT *
                FROM 'rubrique'
                WHERE rubrique.id='$valeur'" ;
```

- `$sql_rubrique` recherche toutes les informations sur la rubrique visée.

Continuons notre source *article.php* :

```
// -----------------------
// Si le numero de la page n est pas renseigne par defaut
// on considere que c est la premiere
// -----------------------
$page = ( isset($_GET["page"]) ? $_GET["page"] : 1);

if (isset($_GET["rubrique"])) //arg. de article.php = rubrique
  {
  if ($request = mysql_query ($sql_rubrique) )
   {
   $line_rubrique=mysql_fetch_array($request,MYSQL_BOTH);
// la page affichée aura un title de page et
// un meta description issu des infos de rubrique.
   $title_page = $line_rubrique["title"] ;
   $meta_description_content = $line_rubrique["description"];
   }
  else
   {
   // probleme
   echo ' debug $sql_rubrique - erreur SQL sur :' .
        $sql_rubrique . ' ' . mysql_error() . '<br />' ;
   }
  }
else
  {
  if (isset($_GET["region"]))
    {
    // inutile de mettre un title et
    // un meta desciption pour le moment
    }
  else // probleme - ce n'est pas un argument valide
    {
    echo 'debug pas d\'argument valide - article.php : <br />' ;
    }
}
```

Si on a les articles d'une rubrique à afficher, on récupère toutes les informations sur la rubrique.

Cela permet d'initialiser les futurs title et meta description de la page à afficher avec les title et description de la rubrique. C'est un début qu'il faudra réviser et améliorer ; il y a trop de défauts dans cette approche.

Si on a les articles d'une *region* à afficher, on ne fait rien.

## L'erreur à éviter ou à gérer

Le script *article.php* ne gère qu'un seul critère : `rubrique`. La tentation est grande pour le codeur d'en ajouter d'autres, à choisir parmi les champs de la table MySQL articles. Par exemple :

- Contributeur ;
- URL ;
- Source.

De plus, ce serait très facile et très court à coder. Ce site offrirait à l'internaute de multiples manières de recherches d'articles. Un vrai plus pour le site, les même articles. Sans précaution, un tel site proposerait à Google plusieurs fois le même article sous différentes URL : nous aurions alors des duplicate contents. Cela constitue une des erreurs classiques à éviter en codage web informatique. Google n'indexera qu'une URL avec la possibilité que les mots-clés de cette URL ne soient pas les plus adéquats aux besoins en référencement du site.

Nous verrons un peu plus loin comment gérer les N chemins (les N URL) possibles d'une page de contenu.

Un title de page et un meta description doivent être uniques pour une efficacité maximale. Uniques, mais leurs mots-clés doivent être cohérents avec le contenu de la page sous peine de fortes pertes en positionnement.

Nous avons mis le title de la rubrique comme title de page. Il est très court et répétitif avec toutes les pages listant les articles de cette rubrique cinq par cinq. Il faut donc compléter et personnaliser ce title de page sous peine d'avoir des erreurs signalées par Google comme des répétitions de title ou de meta description (voir votre interface **Web Mastering Tools** dans votre compte Google).

La même problématique se pose pour le meta description :

```
$title_page = $title_page . '**' ;
// on separe par ** les données agrégées.
```

```
// Attention : ** est utilisé par la fonction meta_article()
$meta_description_content = $meta_description_content .'**' ;
```

Nous allons agréger des informations venant des articles eux-mêmes. Ces informations seront séparées par un tiret (-) afin de faciliter la mise au point et la détection d'incohérence.

### ⚠ Le snippet Google.

Le snippet Google est le petit descriptif d'une URL affiché par Google dans une SERP. Dans de nombreux cas, il est prélevé par Google dans le `meta description` ou dans le texte de la page en fonction des mots clefs saisis par l'internaute.

Afin de couvrir le cas où l'internaute saisit uniquement les mots clefs stratégiques ciblés par la page, cas théoriquement le plus courant, il est important, si c'est possible, de faire figurer ces mots clés stratégiques dans le `meta description` afin de maîtriser le résumé affiché en snippet.


► Fig. 2-9 : Un snippet Google détaillé

Ainsi un *title* de page sera constitué :

- du *title* de la rubrique (voir table rubrique) ;
- de quelques mots pris au début de chaque title d'article listé dans la page.

Le code source réalise cette opération sur le title et sur le meta description de la page de 5 résultats à afficher via le script *article.php* :

```
// FAIRE ce traitement uniquement si on est sur une recherche
// par rubrique, sinon on ne fait RIEN
```

```
// on veut personnaliser la page article en cours
// aux ($resultats_par_page = 5) articles affichés.
//

if (isset($_GET["rubrique"]))
{
  $sql_calcul_meta = $sql ;

  if ($request = mysql_query($sql))
    $nombre_de_resultats = mysql_num_rows($request);
  else
    $nombre_de_resultats = 1;
// -----------------------
// On definit le nombre maximal de resultats à afficher / page
// -----------------------
  $resultats_par_page = 5;

  // -----------------------
  // On calcule le nombre de pages nécessaires pour
  // afficher tous les articles de cette rubrique
  // -----------------------
  $nombre_de_pages =
         ceil($nombre_de_resultats / $resultats_par_page);

  $debut = ($page - 1)*$resultats_par_page;
// avec $page et $debut, on positionne la requete SQL
// juste sur les 5 articles de la page demandée en affichage
  $sql_calcul_meta .= " LIMIT "
                      . $debut .", "
                      . $resultats_par_page .";";
  // on a construit la requete qui va nous permettre
  // d'acceder aux titles / description de chaque
  // article qui va être affiché
  // on pourra personnaliser ainsi la page article en cours
  // aux articles affichés.
  if ($request = mysql_query($sql_calcul_meta))
    {
    while ($line = mysql_fetch_array($request))
    // on recupere chaque title / description de chacun
    // des 5 articles possibles de la page article.php
      {
      $title_page = $title_page . $line["title"] . '**' ;
      // on a nos 1 + jusque 5 title (1 rubrique, 5 article)
      $meta_description_content = $meta_description_content
                                  . $line["description"]
                                  . '**' ;
    // on a nos 1 + jusque 5 meta description
```

```
      }
   // on prend 2 ou + mots par title ;
   // 4 ou plus par meta description
     $title_page = meta_article($title_page, "title") ;
     $meta_description_content =
           meta_article($meta_description_content , "description") ;}
} // du if (isset($_GET["rubrique"]))
```

Pour des raisons que nous détaillerons plus loin, il est inutile de faire ce traitement sur des pages autres que les rubriques. Ces raisons sont liées aux approches pour éviter les duplicate contents que nous avons évoqués précédemment pour cause d'affichage par *article.php* de même articles avec différentes URL, une par critère de tri.

Ce traitement :

- Simule les calculs pour l'affichage des articles cinq par cinq en gérant la pagination.
- Calcule les (5) articles à afficher dans la page demandée.
- Récupère des informations sur chacun des 5 articles de la page à afficher pour construire un title et un meta description unique et cohérent avec la page.

Commentons le code source :

`if (isset($_GET["rubrique"]))`. On fait ce traitement uniquement si l'on affiche une page de 5 articles avec une rubrique renseignée comme argument passé dans l'URL.

{ $sql_calcul_meta = $sql ; pour éviter les bogues, présents ou surtout à venir, lors de futures évolutions, on sépare les 2 requêtes même si elles sont identiques.

`if ($request = mysql_query($sql))`

`$nombre_de_resultats = mysql_num_rows($request);` on récupère le nombre d'articles à afficher

`Else`

`$nombre_de_resultats = 1;` pour éviter une division par 0 plus loin.

`$resultats_par_page = 5`. Pour tout l'exercice et pour différentes raisons, *on bloquera à cinq (5) cette valeur*. Vous pourrez augmenter cette valeur quand, en ayant pratiqué le reste du livre, vous comprendrez l'impact majeur, positif ou désastreux, sur le référencement de tout changement de cette valeur. Surtout, ne l'augmentez pas. Vous pouvez la diminuer sans crainte en revanche.

`$nombre_de_pages = ceil($nombre_de_resultats / $resultats_par_page)`. On calcule le nombre de pages nécessaires à afficher tous les articles de cette rubrique. Ce calcul nous est utile pour savoir quels seront les 5 articles affichés.

```
$debut = ($page - 1)*$resultats_par_page;
$sql_calcul_meta .= « LIMIT ». $debut .", ". $resultats_par_page .";";
```

Avec `$page` et `$debut`, on positionne la requête SQL juste sur les 5 articles de la page demandée en affichage.

`$page` a été initialisé avec l'argument page lors de l'appel à *articles.php*.

`$sql_calcul_meta` est concaténé avec le calcul SQL permettant d'extraire uniquement les 5 articles qui nous intéressent : ceux qui seront affichés.

Rappel : on limite à cinq enregistrements maximum par page affichée. Il peut y en avoir moins, bien sûr, soit parce que la rubrique commence à se remplir, soit parce que le nombre d'articles d'une rubrique ne correspond pas à un multiples de cinq. La dernière page d'une rubrique aura donc de un à cinq chapeaux d'articles à afficher.

Ces un à cinq enregistrements contiennent pour chacun (voir la table *article*) de précieux champs d'informations pour le référencement : `TITLE` et `META DESCRIPTION` pour notre exemple à ce niveau du livre. Nous allons prélever les informations de ces champs pour créer ainsi ce que le référencement sur Google exige : un `title` et un `meta description` uniques pour chaque page mais cohérents entre eux et avec le contenu de la page (qui affiche les cinq chapeaux issus de notre requête sélective, donc obligatoirement une cohérence entre le chapeau d'un enregistrement et les informations de chaque paire de `title` et `meta description`). La cohérence globale sera donc assurée.

⚠ Contenu du `META DESCRIPTION`

Le `META DESCRIPTION` n'a officiellement aucune importance dans le positionnement d'après Google et Matt Cutts. Mais il a une énorme importance dans le taux de clic depuis la SERP quand il est utilisé comme snippet. Une rédaction adroite du `META DESCRIPTION` permet de favoriser son utilisation par Google comme snippet, typiquement en y regroupant un texte si possible cohérent et lisible avec les mots clés stratégiques de la page. Dans le cas de *article.php*, nous ciblons des requêtes utilisant des combinaisons peu usitées de mots clés (un effet longue traine), ce qui fait que la qualité de construction de `META DESCRIPTION` n'est guère importante, contrairement à celle de *zoom.php*.

`if ($request = mysql_query($sql_calcul_meta))` si on récupère de 1 à 5 enregistrements, on va récupérer les `title` et `meta description`. Sinon, on laisse tel que. Rappel : il y a le `title` et le `meta description` de la rubrique en début de chacun de ces `meta` de la page à afficher.

`{ while ($line = mysql_fetch_array($request))` : on récupère chaque title et chaque description de chacun des 1 à 5 articles possibles.

{ $title_page = $title_page . $line["title"] . '**'. On concatène title_page avec les 1 à 5 titles de page. Chaque ajout est séparé par la séquence ** qui sert de délimiteur obligatoire.

$meta_description_content = $meta_description_content . $line["description"] . '**' ; même chose pour le meta description.
}

Dans $title_page nous avons le title de la rubrique et les 1 à 5 title de pages de nos 1 à 5 enregistrements possibles. Le meta description a été traité de la même manière.

C'est beaucoup trop long. Nous avons trop de mots dans le title. En effet en concaténant les mots de chaque title on obtient un title composé de dizaines de mots. Il faut réduire le nombre de mots. Pour faire simple à ce niveau du livre, nous allons simplement conserver uniquement les premiers mots de chaque title de page. C'est l'objectif des 2 lignes de codes suivantes :

```
$title_page = meta_article($title_page, "title") ; // on prend 2 ou + mots par title ; 4 ou plus par meta description
$meta_description_content = meta_article($meta_description_content , "description") ;
}
```

} accolade fermante du if (isset($_GET["rubrique"])) de la séquence de code étudiée.

Nous devons étudier la fonction meta_article($argument) pour comprendre ce qui se passe :

```
function meta_article($page, $type)
{
  $meta = '' ;
  $long = strlen($page) ;
  $nb_articles = mb_substr_count($page, "**") ;
// mb_substr_count($page, "**") renvoie le nombre
// d'occurences de "**" dans la string $page
// 1 ** par title. 6, le maximum, correspond à
// un title rubrique + 5 enregistrements de la table article.
  switch ($nb_articles) {
  case 0:
    echo "Probleme : 0 occurence de ** dans meta_article <br />";
    break;
  case 1:
    $pas=12;
    break;
  case 2:
    $pas=6;
```

```
    break;
  case 3:
    $pas=4;
    break;
  case 4:
    $pas=3;
    break;
  case 5:
    $pas=3;
    break;
  case 6:
    $pas=2;
    break;
  }

  if ($type == "description" )
   $pas = $pas * 2 ; // 2 fois plus de mots pour meta descrip
  while ($long > 0) // tant qu'on est pas au bout de la chaine
   {
    // on recupere le premier bloc de mots jusque **
    $i = stripos($page , "**") ;
    // on le copie dans $textfull du car n°0 au $i ème
    $textfull = substr($page, 0, $i);
    // on enlève le text déjà traité,
    $page = substr_replace($page, '', 0, $i+2) ;
// on ajoute 2 pour eliminer les 2 * devenus inutiles
    $long = $long - ($i+2) ;
    $tab = str_word_count($textfull,1,'àçéè0123456789ù&;ëâê%_');
    for ($a = 0;$a<$pas;$a++) // on prend les $pas premiers mots
      {
        $meta = $meta . ' ' . $tab[$a] ; // on les concatène
        }
    }
  return ($meta) ;
// le meta, title ou description,
// est à la bonne taille en nombre de mots
}
```

`$page` contient une string : à ce stade du livre, c'est soit un title de page, soit un meta description.

`$type` indique le type de meta à traiter : title (donc limite de mots à 10/12) ou meta description, le double de mots par rapport à title).

Par convention, cette fonction php4 `meta_article()` stockée dans */php_inc/ biblio_inc .php* exige :

- `title` ou `meta_description` de chaque enregistrement de la table `article` et de la table `rubrique` doivent avoir les 3 mots-clés les plus stratégiques en début de string.
- Les différentes concaténations doivent être séparées par le substring ** pour permettre le bon fonctionnement de la fonction `meta_article()`.

Un principe simple est appliqué :

- Nous avons ajouté de 1 à 6 champs issus des tables *rubrique* et *article* dans `$page`.
- À chaque ajout correspond un ajout de ** comme délimiteur.
- Nous comptons le nombre d'ajouts.
- Le meta final ne doit pas dépasser 10 à 12 mots pour un `title` et 20 à 24 mots environ pour un `meta description`.
- La fonction renvoie un `title` ou un `meta description` de la bonne taille. Il sera affiché dans la page issue de *articple.php*.

```
$meta = '' ;
$long = strlen($page) ;
$nb_articles = mb_substr_count($page, "**") ;
// 1 ** par title. 6, le maximum, correspond à
// un title rubrique + 5 enregistrements de la table article.
```

La fonction `mb_substr_count($page, "**")` renvoie le nombre d'occurrences de ** dans la string `$page`. Nous avons donc le nombre d'ajouts issus des tables `rubrique` ou `article` : de 1 à 6.

```
    switch ($nb_articles) {
```

La fonction `switch` nous permet d'initialiser `$pas` à la valeur la plus adéquate pour avoir un maximum de 10/12 mots pour un title de page.

```
    if ($type == "description" )
       $pas = $pas * 2 ;
```

Pour un `meta` de type description, on double la valeur du pas pour avoir un maximum de 20 à 24 mots.

Le reste du code de cette fonction est simple :

- On extrait chaque ajout, une *substring* donc, délimité par **.
- On élimine les ** devenus inutiles.
- On range les mots de cette substring dans un tableau.
- On transfère et on concatène les `$pas` premiers mots de ce tableau dans la string `$meta`.

- On renvoie le résultat qui sera de 10 à 12 mots pour une balise `TITLE` de page et de 20 à 24 mots pour une *meta description* de page.

Cette partie du code source de *article.php* que nous venons de voir, préparait le travail d'affichage.

Passons à l'affichage de notre page :

```
MiseEnPage::haut($title_page,$meta_description_content,$rep);
?>
  <div id="sousban">  </div>
  <div id="pagegauchetete">
  <?php
  MiseEnPage::MenuGauche() ;
 ?>
  </div>   <!-- fin conteneur : page gauche tete -->

<!-- conteneur centre et col. droite -->
  <div id="pagecentretete">
    <div id="pagedroite"> <!-- COLONNE DROITE -->

      <?php
            MiseEnPage::MenuDroite($rubrique) ;
 ?>
  </div>   <!-- fin page droite -->
```

### ⚠ Nomenclature pour les champs

Nous venons de voir que les premiers mots-clés des champs *title* et *description* de ces tables sont utilisés. Il en ressort qu'une procédure doit être appliquée lors de la saisie de ces champs :

`Title` : les 2 à 4 mots-clés fondamentaux doivent être en début de string.

`Meta description` : les 4 à 8 premiers mots doivent contenir les mots-clés fondamentaux et compatibles avec le snippet Google.

Pour `Title`, la lisibilité humaine ne doit en aucun cas être altérée.

Il faut être conscient que les pages affichées par *article.php* ne sont pas appropriées pour êtres des landing pages mais plutôt pour être des pages captant des combinaisons de mots-clés peu usitées.

La fonction PHP5 `haut()` de la classe `MiseEnPage` (dans *php_inc/MiseEnPage.inc.php*) affiche la partie haute de notre gabarit de page : logo, bannière, title de page, meta description, etc.

La mise en page gauche puis centrale et droite est gérée par les fonctions `MenuGauche()` et `MenuDroite()` via les conteneurs `pagegauchetete`, `pagecentretete` et `pagedroite` décrits en feuille de style.

C'est du XHTML de base sur lequel nous ne passerons pas plus de temps.

`MenuGauche()` affiche le menu de gauche de notre site web exemple. Chaque lien possède une URL conforme à notre règle de réécriture (voir *.htaccess* dans le présent chapitre).

Pour rappel :

```
<li><span>02 </span>
   <a href='finances-fiscalite-entreprise-diminuer-deduction
  ➥ -exoneration- impots-taxes/rubrique/2/1.html'
      title='02 - Fiscalité de l'entreprise'>
      Fiscalité de l'entreprise
   </a>
</li>
```

*finances-fiscalite-entreprise-diminuer-deduction-exoneration-impots-taxes/rubrique/2/1.html*

Le premier tronçon de notre URL ne sert qu'au référencement. Il n'a aucun impact sur la logique applicative de notre logiciel de publication. Sa seule mission est de pousser les mots-clés de sa rubrique vers Google.

Le menu et les liens ainsi déposés, manuellement dans une fonction PHP5, vont surprendre nombre de codeurs avertis qui se demanderont "*pourquoi une approche technique aussi basique*". C'est volontaire. En effet, ajouter au présent ouvrage une fonction qui prend un champ dans la table rubrique et génère directement un menu sans intervention manuelle par un codeur l'aurait alourdit inutilement. De telles types d'optimisations strictement PHP ne sont pas dans l'objectif de ce livre.

On fait donc simple, rudimentaire et clair dans ce site web exemple afin de rester concentré sur la partie référencement.

*/rubrique/2/1.html*. Comme déjà discuté, cette URL demande l'affichage de la page 1 des résultats de la rubrique n°2 via l'URL rewriting qui va transformer cette partie de l'URL en paramètres passés au script PHP.

Ci-après, une séquence que nous avons déjà vue : la pagination des résultats. Ici, on ne se contente pas de la simuler pour savoir quelles informations prendre afin de personnaliser les title et meta description de la page. On pagine pour de vrai :

```
<?php
if ($request = mysql_query($sql))
  $nombre_de_resultats = mysql_num_rows($request);
else
  $nombre_de_resultats = 1;

// -----------------------
// On definit le nombre maximal de resultats
// à afficher par page
// -----------------------
$resultats_par_page = 5;

// -----------------------
// On calcule le nombre de pages
// -----------------------
$nombre_de_pages=ceil($nombre_de_resultats/$resultats_par_page);

$debut = ($page - 1)*$resultats_par_page;
$sql .= " LIMIT ". $debut .", ". $resultats_par_page .";";
if ($request = mysql_query($sql))
```

`$request` contient les enregistrements de la table *article* dont il faut afficher les chapeaux dans la page :

```
{
  echo '<div id="pagetext">';
  // on affiche la liste des numéros de page
  // au dessus du futur affichage des 5 / 5 resultats
  echo "<b>Page</b> : ";
  for ($i=1 ; $i<=$nombre_de_pages ; $i++)
    {
    echo ( ($i==$page) ? "<b>" : "");
    echo "
      <a href=\"".$referencement
              ."/".$parametre."/".$valeur
              ."/".$i.".html\">";
    echo $i;
    echo "</a> ";
    echo ( ($i==$page) ? "</b>" : "");
    }
  if ($nombre_de_pages==0)
      echo "<em>aucun résultat</em>";
      echo "<br />";
```

On affiche la gestion des pages en premier sous la forme d'une ligne de nombres cliquables, de 1 à `n`.

Si le nombre de page est de 0, aucun résultat ne correspond à cette demande (rubrique vide).

## Problème de duplicate content


► Fig. 2.10 : Ce chapeau d'article dépend de cette 1ère URL / rubrique primaire.

Nous allons devoir gérer notre premier problème de duplicate content. Si nous affichons un chapeau d'un article appartenant à plusieurs rubriques, nous aurons plusieurs URL distinctes contenant un même article. Il faut donc sélectionner le seul lien que nous laisserons accessible au bot de Google pour éliminer ce risque.


► Fig. 2.11 : Ce chapeau d'article dépend également de cette seconde URL / rubrique secondaire.

L'approche retenue dans ce site exemple est :

- Un article appartient principalement à une seule rubrique que nous appellerons primaire.
- Les autres rubriques auxquelles un article peut appartenir seront appelées secondaires.
- Seuls les liens vers des articles dépendant de manière primaire de la rubrique en affichage via *article.php* doivent être accessibles à Google.
- Ces informations de dépendance sont gérées dans la table `article_rubrique`. Un article a une seule rubrique primaire.
- Les autres liens seront rendus invisibles à Google via la balise `rel="nofollow"` . Ainsi le codeur remplit sa mission : une seule URL pour un seul contenu, avec une URL sélectionnée par la logique applicative du logiciel, donc la plus adéquate à notre référencement :

```
if (isset($_GET["rubrique"]))
  {
  // affichage des 1 à 5 rows du résultat de la requête
  while ($line = mysql_fetch_array($request))
    {
    // on veut savoir si l'article à afficher
    // est dans sa rubrique primaire ou non
    // sql_ps : Primaire ou Secondaire ?
    $id_art = $line["id"] ;
    $sql_ps = "select *
                from 'article_rubrique'
                WHERE article_rubrique.id_article = '$id_art'
                 AND article_rubrique.id_rubrique='$valeur' ";
// on recupère la seule ligne où cet article
// est dans la rubrique ($valeur) à afficher
      if ($request_ps = mysql_query($sql_ps))
      {
      if ($line_ps = mysql_fetch_array($request_ps) )
        {
        if ($line_ps["primaire"] == 1)
          { $ps = "rubrique_primaire" ; }
        else
          { $ps = "rubrique_secondaire" ; }
        }
      else
        { echo ' Debug : ERREUR : ' ; }
    }
// fin du module detectant si l'article appartient ou non
// à sa rubrique principale
```

Pour chaque enregistrement de la table `article`, on veut savoir si la rubrique en cours est primaire ou secondaire.

`$ps` contient le statut de l'article en cours de traitement. Il est dans sa rubrique primaire ou non.

```
preview_article($line,$line_rubrique["nom"],$valeur, $ps);
```

La fonction PHP4 `preview_article` est stockée dans *php_inc/ biblio_inc.php*. Cette fonction a en charge l'affichage de l'article en mode Preview, c'est-à-dire affichage d'un minimum d'informations permettant à l'internaute de savoir s'il désire lire le détail ou non de cet article : chapeau, titre, n° de rubrique.


► Fig. 2.12 : Affichage dans le centre de article.php par preview_article.

Détaillons cette fonction `preview_article` ci après.

```
function preview_article
         ($line,$pathway_prec,$valeur_rubrique,$ps)
{
  echo '
    <div class="bordurebloc3">
      <div class="borduretitre3">
  ' ;
  if ($ps == "rubrique_primaire")
    {
    echo '
      <a href="article/'
              . $line["url"]
              .'.html&src='. $pathway_prec
```

```
              . '">'. $line["titre"]
     .'</a>
      </div>
      <div class="bordureparagraphe3">
      <h2>'. $line["chapeau"] .'</h2> ' ;
      $sentence = cut_sentence($line["texte"],50) ;
// on joue avec le feu ! 50 car. = zone rouge
//
// il faut eviter d'avoir un debut de code HTML
// coupé en plein milieu => risque affichage erratique sinon.
// donc s'il y a des <div> <table>,
// il ne faut pas afficher l'extrait du texte mais que le chapeau
      if (html_dans_string($sentence) )
        echo ' ... ' ;
      else
        echo $sentence . ' ... ' ;
      echo '
    </div>
  <div class="tavoirplus3">
    Lire TOUT l\'article : <a href="article/'
    . $line["url"] .'.html&src='. $pathway_prec . '">'
    . $line["url"] . '</a> ... Article paru le '
    . $line["date-si"] .'
     </div>
   </div>
  ';
    }
  else // donc $ps == "rubrique_secondaire"
       // On doit bloquer les bots de Google
    {
    echo '
    <a href="article/'. $line["url"]
       .'.html&src='. $pathway_prec . '" rel="nofollow">'
       . $line["titre"] .'</a>
    </div>
    <div class="bordureparagraphe3">
      <h2>'. $line["chapeau"] .'</h2> ' ;
      $sentence = cut_sentence($line["texte"],50) ;
      // il faut eviter d'avoir un debut de code HTML
      // coupé en plein milieu => affichage erratique sinon.
      if (html_dans_string($sentence) )
        echo ' ... ' ;
      else
        echo $sentence . ' ... ' ;
      echo '
        </div>
          <div class="tavoirplus3">
```

```
            Lire TOUT l\'article n° '
              . $line["id"] . ' : <a href="article/'
              . $line["url"] .'.html&src='. $pathway_prec
              . '" rel="nofollow">' . $line["url"] . '</a>
                Article paru le '. $line["date-si"] .'
        </div>
      </div>
    ';
  }
}
```

Cette fonction est écrite de la manière la plus simple possible pour en faciliter la compréhension. Il y a des optimisations de code réalisables dans ce source mais encore une fois, ce n'est pas notre objectif de créer le code parfait, quelquefois difficile à lire par un débutant en PHP.

Les `div` utilisées sont décrites dans les feuilles de style. C'est standard donc inutile de s'attarder dessus.

Cette fonction traite les données de l'article à afficher en deux groupes distincts sur un point clé : les liens sont bloqués par un `rel="nofollow"` pour les articles n'appartenant pas de manière primaire à la rubrique en cours d'affichage.

Ainsi, grâce à l'application de cette règle, on évite d'avoir un duplicate content entre la page dédiée à l'article et les différents chemins pouvant y mener. Google ne verra qu'un seul chemin valide, celui de la rubrique d'appartenance principale de l'article.

⚠ Rel="nofollow" évite le duplicate content

En bloquant le bot de tout moteur de recherche grâce à cette balise sur les URL dupliquées vers une même page, on évite les duplicate contents.

Cette logique peut être utilisée pour coder de multiples possibilités de navigation humaine.

Rappelez vous, on pourrait chercher des articles par rubriques mais aussi par contributeurs, sources, etc. Bref, on pourrait créer des liens avec URL rewriting, permettant de chercher selon d'autres critères des articles. Cela serait un plus pour les internautes et la qualité ergonomique du site. Mais cela créerait de nombreux duplicate contents sauf à baliser en `rel="nofollow"` toutes les autres possibilités de navigation créant un potentiel de duplicate content.

Un autre point technique est à évaluer : la qualité des textes à afficher. Dans notre méthode, un chapeau est toujours un texte simple, sans code HTML ou autre. En revanche, le texte de l'article peut contenir de multiples codes HTML plus ou moins

complexes. Hors nous affichons un extrait de ce texte dans `preview_article`. On risque donc un affichage erratique dans la page si on coupe le texte n'importe où.

Pour faire simple et rester centré sur l'objectif de ce livre, nous allons appliquer une politique simple : si un code HTML est détecté dans l'extrait à afficher, on n'affiche rien.

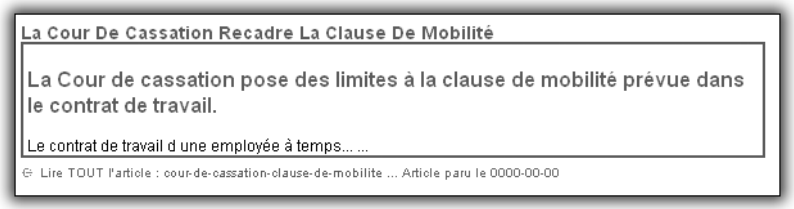
La fonction `html_dans_string($sentence)` renvoie VRAI si un code HTML est détecté dans la string passée en argument. Sur VRAI, on n'affiche rien.

Un autre point reste à gérer dans `preview_article` : jouer ou non avec le feu, avec la limite en `duplicate content` de Google et le risque d'une pénalité associée au franchissement d'un seuil dont on ne sait pas grand-chose.

## Jouer avec le feu : sanction Google possible

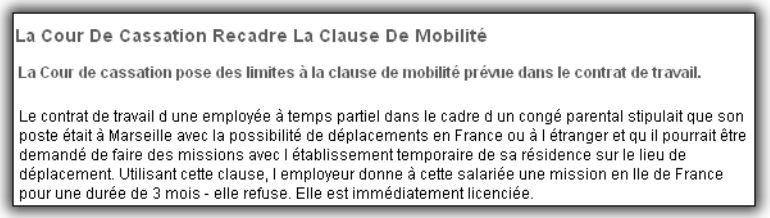
Deux stratégies sont possibles dans `preview_article` :

- *Jouer avec le feu*. On affiche les N premiers caractères du texte principal de l'article. Avantage : on leste en volume de texte notre page en cours ce qui est intéressant en référencement. Inconvénient : on risque de démarrer un duplicate content. En effet, ce même contenu, les N premiers caractères, sera affiché dans la page dédiée à cet article.
- *Sécurité d'abord* : on n'affiche rien ou moins de 30 caractères (N < 30). On peut aussi casser le rythme de l'extrait affiché en y insérant des mots ou un commentaire venant d'une autre source entre chaque extrait.




▶ Fig. 2.13 : Chapeau puis extrait du texte de l'article en duplicate content interne avec le même texte de l'article complet.

Insérer 35 à 50 caractères du texte de l'article dans la page résumé sous le chapeau augmente le volume de texte de notre page et renforce la cohérence du lien reliant cette page à la page dédiée à l'article. Cela renforce mutuellement le référencement de nos deux pages.




► Fig. 2.14 : L'article complet avec le même chapeau et la réédition de l'extrait en tête de texte.

Il est donc tentant pour un référenceur d'aller chercher la limite. Google change fréquemment les seuils et stratégies de déclenchements des duplicate contents. Le risque existe d'avoir une sanction. Les deux pages sont liées par un backlink, donc le texte dupliqué est immédiatement visible. Google possède des seuils de tolérance et des règles comportementales très évolutives et difficiles à cerner dans le détail. Pour faire simple, entre 35 et 50 caractères dupliqués, on est en zone orange : on a une probabilité d'avoir une pénalité. Au-delà, on passe en zone rouge avec une probabilité de pénalité quasi certaine. En dessous, a priori, on ne risque rien. Mais attention, ces chiffres varient en fonction de plusieurs critères. Par exemple, des termes utilisés dans votre contenu : plus les mots utilisés, les formulations écrites seront rares et peu usités, plus vite le seuil de duplicate content approchera.

Conclusion : à chaque référenceur de fixer le curseur N qui lui semble souhaitable.

## Génération de pages dynamiques – zoom.php

Nous venons de voir *article.php* en détail. Ce script, article, affiche des listes de résumés d'articles. Un clic sur un lien dans le résumé entraîne l'affichage de la totalité de l'article.

Le script PHP *zoom.php* a cette mission : afficher une page HTML pouvant être référencée dans Google.

### Appel de zoom.php depuis article.php

Expliquons la forme de cet appel qui sera bien sûr réécrit à la volée.

*/article/capital-non-libere-is-taux-maximum.html&src=Fiscalite-entreprise*

`/article`, lors de requête sur Google de la forme site, permet de trier les pages articles indexées. Nous avons déjà abordé cette astuce :

```
site:www.monsite.com inurl:article
```

`capital-non-libere-is-taux-maximum` : deux usages de ce string.

- Des mots-clés cohérents avec la page appelée et avec la page appelante puisque par définition, *article.php* affiche titre et chapeau au minimum du texte qui sera affiché. On peut même avoir un extrait du texte d'une longueur de 0 à 50 caractères. On renforce le positionnement de ces deux pages, celle affichée par *article.php* et celle affichée par *zoom.php*.
- Ce string est unique. Il est propre à cet article. Il nous sert donc de clé pour accéder au bon enregistrement dans la table article.

`&src=Fiscalite-entreprise` : on ajoute un code, `src` dans cet exemple, contenant le nom de la rubrique sous une forme permettant sa mise en URL. Ainsi on récupère directement via l'URL la rubrique d'où vient le clic. C'est pratique pour réaliser un fil de navigation mais hors sujet dans ce livre. On case aussi deux mots-clés cohérents et pertinents dans l'URL, améliorant ainsi le positionnement.

```
RewriteRule
➥ ^([-_a-zA-Z0-9]+)/([-_a-zA-Z0-9]+).html&src=([-_a-zA-Z0-9]+)$
➥ zoom.php?referencement=$1&url=$2&src=$3
```

L'URL réécrite appelle *zoom.php* avec en argument `referencement =` le string de mots-clés, l' URL permettant de rechercher l'article à afficher dans la table *article*, et enfin `src =` la rubrique d'origine du clic, permettant ainsi de gérer un éventuel fil de navigation.

## Code source de zoom.php – version simple

Voici la première partie de ce code source. Il est autonome et fonctionnel. Une autre partie, la gestion du tags cloud, du nuage de liens, sera ajoutée et documentée dans le chapitre consacré aux backlinks internes à un site :

```
<?php
require_once("php_inc/MiseEnPage.inc.php");
require_once("php_inc/init.inc.php");
require_once("php_inc/biblio_inc.php");

if (isset($_GET["url"]))
  {
  $sql = "SELECT * FROM 'articles'
                  WHERE url='".$_GET["url"]."' LIMIT 1;";
  if ($request = mysql_query($sql))
    {
```

```
    if ($line = mysql_fetch_array($request))
      {
      $title_page = $line["title"] ;
      $meta_description_content = $line["description"] ;
      // ce id article va nous permettre de gérer l'affichage
      // d'info contextuelle dans la colonne de droite
      $id_Article = $line["id"] ;
      }
    }
  }

MiseEnPage::haut($title_page,$meta_description_content,$rep);
?>

<div id="sousban">
  </div>
<div id="pagegauchetete">
  <?php
    MiseEnPage::MenuGauche() ;
 ?>
</div>   <!-- fin page gauche tete -->

<div id="pagecentretete">
  <!-- COLONNE DROITE -->
  <div id="pagedroite">
  <?php
    MiseEnPage::ColDroite($id_Article) ;
 ?>
  </div>   <!-- fin colonne droite -->


    <div id="pagetext">
    <?php
      view_article($line);
 ?>

      <!--fin page texte -->
    </div>
    <!-- fin pagecentre -->
  </div>

  <?php
    MiseEnPage::bas() ;
 ?>
```

On récupère les include PHP dont on a besoin : require_once();

```
require_once("php_inc/MiseEnPage.inc.php");
require_once("php_inc/init.inc.php");
require_once("php_inc/biblio_inc.php");
```

Si on a bien un paramètre URL passé par l'appel à *zoom.php*, on peut construire une requête SQL en charge de récupérer l'article voulu dans la table *article*.

```
if (isset($_GET["url"]))
  {
  $sql = "SELECT * FROM 'articles'
                  WHERE url='".$_GET["url"]."' LIMIT 1;";
  if ($request = mysql_query($sql))
```

Dans l'enregistrement article ainsi lu, on prend la valeur du title de page et on instancie la valeur du meta description content.

```
{
$title_page = $line["title"] ;
$meta_description_content = $line["description"] ;
$id_Article = $line["id"] ;
}
}
}

    MiseEnPage::haut($title_page,$meta_description_content,$rep);
?>
```

On affiche tout le haut de la page.

Ensuite, on passe à l'affichage du menu de gauche et de la colonne de droite. Le principe est identique à *article.php*.

Le ID de l'article permet d'appeler une fonction PHP5 ColDroite() qui affiche le contenu de la colonne droite de notre page par rapport au numéro ID de notre article.

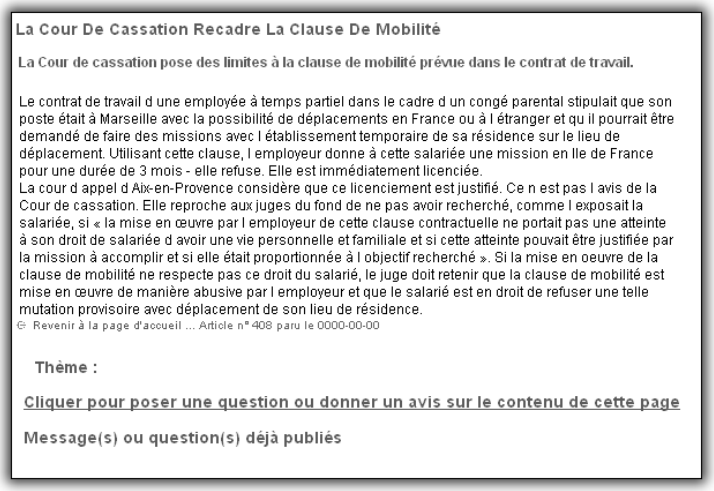
```
  <div id="sousban">
    </div>
  <div id="pagegauchetete">
  <?php
    MiseEnPage::MenuGauche() ;
 ?>
  </div>   <!-- fin page gauche tete -->

  <div id="pagecentretete">
```

```
   <!-- COLONNE DROITE -->
   <div id="pagedroite">
     <?php
           MiseEnPage::ColDroite($id_Article) ;
?>
 </div>   <!-- fin colonne droite -->
```

On peut afficher la totalité de l'article dans le conteneur DIV `pagetext`.

```
   <div id="pagetext">
     <?php
       view_article($line);
?>

     <!--fin page texte -->
   </div>
   <!-- fin pagecentre -->
 </div>

 <?php
    MiseEnPage::bas() ;
?>
```

La fonction PHP4 `view_article($line)`, localisée dans *php_inc/biblio_inc.php*, a en charge d'afficher l'article passé en paramètre `$line`.


La Cour De Cassation Recadre La Clause De Mobilité

La Cour de cassation pose des limites à la clause de mobilité prévue dans le contrat de travail.

Le contrat de travail d une employée à temps partiel dans le cadre d un congé parental stipulait que son poste était à Marseille avec la possibilité de déplacements en France ou à l étranger et qu il pourrait être demandé de faire des missions avec l établissement temporaire de sa résidence sur le lieu de déplacement. Utilisant cette clause, l employeur donne à cette salariée une mission en Ile de France pour une durée de 3 mois - elle refuse. Elle est immédiatement licenciée.
La cour d appel d Aix-en-Provence considère que ce licenciement est justifié. Ce n est pas l avis de la Cour de cassation. Elle reproche aux juges du fond de ne pas avoir recherché, comme l exposait la salariée, si « la mise en œuvre par l employeur de cette clause contractuelle ne portait pas une atteinte à son droit de salariée d avoir une vie personnelle et familiale et si cette atteinte pouvait être justifiée par la mission à accomplir et si elle était proportionnée à l objectif recherché ». Si la mise en oeuvre de la clause de mobilité ne respecte pas ce droit du salarié, le juge doit retenir que la clause de mobilité est mise en œuvre de manière abusive par l employeur et que le salarié est en droit de refuser une telle mutation provisoire avec déplacement de son lieu de résidence.
Revenir à la page d'accueil ... Article n° 408 paru le 0000-00-00

Thème :

Cliquer pour poser une question ou donner un avis sur le contenu de cette page

Message(s) ou question(s) déjà publiés

► Fig. 2.15 : Affichage dans centre de zoom.php par `view_article($line)`

Examinons son code source :

```
// affiche un article complet

function view_article($line){
  echo '
    <div class="jcmb">
      <div class="jcmbtitre">
      '. $line["titre"] .'
      </div>
      <div class="jcmbparalarge">
        <h2>'. $line["chapeau"] .'</h2> '. $line["texte"] .'
      </div>
  ';
  echo '
    <div class="tavoirplus3">
      <a href="index.php"> Revenir à la page d\'accueil</a>
       ... Article n° ' . $line["id"]
         . ' paru le '. $line["date-si"] .'
    </div>' ;

  echo ' </div>'; // le div de fin de page article
}
```

La mise en page est gérée en CSS. Le chapeau est affiché entre balises `<h2>`, le texte est affiché tel quel, ce qui permet d'avoir du HTML dans le champ *texte* de la table *article*. C'est pratique mais dangereux. Les séquences HTML doivent être complètes et correctes sous peine d'affecter l'affichage de la page pour cause de balises non fermées, par exemple. Un lien permet de revenir à la page d'accueil.

## 2.7 Résumé

Nous avons un outil logiciel de publication d'articles. La version que nous venons de voir est simple. Elle va subir quelques améliorations dans les chapitres à venir.

Notre logiciel est écrit en PHP, il y a des syntaxe d'appels de fonction en PHP4 et en PHP5. Il est basé sur une version standard de MySQL.

Ce logiciel de publication automatise un minimum d'actions de référencement : gestion automatisées des title de page, des meta description, du chapeau en balises `h2` dans *zoom.php*, d'un titre en `h1` dans *article.php*.

Nous avons géré un cas de duplicate content assez courant : l'appartenance d'une page à plusieurs URL distinctes, qu'elles soient là pour faciliter la navigation (classement de produits/articles par créateur, propriétaire, date, etc.) ou parce qu'un même produit/article peut appartenir à plusieurs catégories distinctes.

# 3

# Page rank sculpting et backlinks

Le *PR sculpting* est basé sur l'utilisation des liens internes dans un site à des fins de positionnement dans l'index de Google. On gère le flux de Page rank, on l'associe à des mots-clés pertinents ; ainsi, on influence le positionnement de certaines pages du site sur des mots-clés précis dans Google.
En français, on traduirait *Page rank sculpting* par "*positionnement par backlinks internes*".

## 3.1 Introduction

Google a été le premier à utiliser l'existence des liens naturels entre pages HTML pour évaluer la pertinence d'une page face à une requête d'internaute. L'utilisation des backlinks pour le calcul du positionnement a été fondamentale dans le succès de Google.

Désormais, sans backlinks, impossible d'être positionné, quel que soit le moteur de recherche.

Le codeur lisant ces lignes doit se dire :"Mais les backlinks sont issus d'autres sites, c'est du web marketing pur, je n'ai rien à voir avec ça". Rien de plus faux. Le codeur est impliqué de différentes manières dans les backlinks :

- Le site doit avoir la capacité à générer des nuages de liens internes. Nous verrons plus loin en détail pourquoi. À ce stade du livre, il suffit de savoir qu'un nuage de liens bien conçu permet de mettre en avant certaines pages du site dédiées à des mots-clés, les landing pages, vers l'index de Google et de tenter ainsi d'obtenir la meilleure place possible.
- Le site doit pouvoir changer aisément de critères *on page*. Un processus automatique serait le bienvenu. Exemple : l'ajout via back office d'un mot-clé sur une page doit permettre de mettre à jour automatiquement tous les critères de cette page ainsi que les éventuels nuages de liens. Ainsi, depuis une table MySQL ou depuis le back office du site, il sera possible de piloter la stratégie de référencement de tout le site.

## 3.2 Principe d'un backlink

Un lien correctement codé comprend deux parties :

- L'URL de la page visée par le lien : obligatoire.
- Le libellé texte : il sera considéré comme constitué de mots-clés.

```
Exemple :
< a href=http://www.lesite.com/formation-webmestre-webmaster.html>
  Formation pour webmestres
</a>
```

Ce libellé de texte véhicule des mots-clés de la page de départ, `formation` et `webmestres`, vers la page d'arrivée.

Dans le cas d'une image à la place du libellé texte, le contenu de la balise `ALT` de cette image sera considéré comme constitué de mots-clés.

Le Page rank de la page de départ a un effet de levier démultipliant l'efficacité de ce backlink. Plus le PR de la page de départ est élevé, plus la poussée sur le positionnement de la page sur les mots-clés du libellé sera fort.

Des sécurités anti-fraude de Google peuvent néanmoins annuler le backlink. Pour éviter de tomber sur ces sécurités, il convient de ne pas construire un backlink n'importe comment. Nous reviendrons sur ce point un peu plus loin.

## 3.3 Principe du Page rank (PR)

Le Page rank, acronyme PR, est une mesure indiquant grossièrement la popularité d'une page selon la communauté des internautes.

Google part du principe que plus on parle et donc plus on cite une page sur Internet, plus on mentionne son URL, plus cette page a un potentiel capable d'intéresser des internautes à la recherche d'informations. Les internautes votent pour cette page.

Google ausculte cette page et la lie dans son index à quelques mots-clés, avec une forte préférence pour ceux mentionnés dans les libellés des backlinks ayant attiré son attention.

Pour construire le Page rank, Google cartographie le web et analyse tout lien reliant deux pages entre elles. Tout lien transfère du Page rank d'une page vers l'autre. Tout lien ayant un libellé transporte des mots-clés d'une page à une autre. Google mesure ceci, le pondère et calcule le PR d'une page ainsi que le positionnement de ces pages sur des mots-clés.

Le PR d'une page sera donc fonction du volume de backlinks pointant vers elle ainsi que de la qualité de chacun d'entre eux.

La qualité d'un backlink obéit à plusieurs critères divisés en deux familles :

- les *critères quantitatifs* ;
- les *critères de cohérence*.

### Les critères quantitatifs d'un backlink

Ce sont les critères les plus faciles à mesurer :

- Le Page rank de la page. Il est approximativement indiqué par le champ PR de la *Google bar*.
- Le nombre de backlinks sortant vers des pages externes. Ces liens sortants, dont celui que nous analysons pour calculer le PR de la page que nous examinons, se partagent le pourcentage de Page rank alloué aux liens sortants de toute page.

- Le nombre de mots du libellé du lien reliant cette page vers la page dont on est en train de calculer le PR. Ces mots se partage le Page rank, donc la poussée dans l'index. Plus il y a de mots, plus la poussée de chaque mot sera faible.

## Les critères de cohérence d'un backlink

Les mots-clés du libellé doivent être présents sur la page de départ et sur la page visée par le lien.

Plus il y a de présences de ces mots-clés sur ces deux pages, plus le lien est cohérent, assurant ainsi un transfert maximal de Page rank associé aux mots-clés du liens.

Comment envisager une automatisation de liens cohérents de qualité ? Pour concevoir des algorithmes sur le sujet, il faut d'abord cerner la problématique et disposer d'axes de réflexion utilisables par un codeur.

Pour cela, nous allons nous mettre à la place d'un apprenti moteur de recherche qui, justement, a lui aussi cette problématique à résoudre. Ainsi, nous aurons des axes de réflexion pour nos algorithmes.

Pour évaluer des backlinks, notre apprenti moteur de recherche a besoin de modéliser le web. Il existe une manière très simple issue de la théorie des graphes :

- Une page HTML est un sommet.
- Un lien entre 2 pages HTML est un arc entre ces 2 sommets.

Un site Internet est un ensemble de sommets. Chaque sommet est unique : il a une seule et unique URL. Pour les arcs (les liens), c'est un peu plus compliqué. Dans le contexte Google, pour faire simple, il ne doit exister qu'un arc, lien, entre 2 sommets, 2 pages. En effet, un arc peut partir d'un sommet (page) P1 unique, aller vers un autre sommet P2 unique. Mais on peut avoir ainsi plusieurs liens reliant une page P1 à une page P2 au gré des besoins du webmestre pour faire passer des internautes d'une page à une autre.

Le moteur face à deux liens ou plus, partant d'une même page P1 et pointant une même page P2 aura un seul comportement : il sélectionnera un lien unique et ignorera les autres.

Nous avons deux objets : *sommet* et *arc*. Dotons-les de propriétés.

### Propriétés d'un backlink (un arc)

Un lien (un arc) peut être équipé d'une structure de propriétés comme celle-ci :

- URL de la page de départ ;
- URL de la page d'arrivée ;

- texte du libellé du lien décomposé mot par mot ;
- dates où ce lien a été détecté par le moteur la première fois, la dernière fois, etc.

### Propriétés d'une page (un sommet)

À chaque sommet, on peut associer une structure de propriétés. Traduit en Web, cela donne à toute page HTML une liste de propriétés décrite par une structure :

- URL ;
- Page rank ;
- `TITLE` de la page ;
- `META DESCRIPTION` ;
- premier texte entre balises `<h1>` ;
- premier texte en balise `ALT` d'une première image ;
- texte n°1 entre balise `<p>` ou équivalent, etc.

### Calculer une cohérence

Nous avons des structures, des objets, des propriétés, des valeurs. Tout bon codeur voit immédiatement ce que l'on peut en faire. Il est aisé de mettre au point un algorithme capable de calculer une cohérence d'un lien entre 2 pages.

Un exemple simple : Est-ce que le mot présent dans un libellé de lien est présent dans ces critères de la page de départ ?

- Dans le `TITLE` ? Autrement, on minore le transfert de Page rank de 20 %.
- Dans un titre `H1` ? Autrement, on minore le transfert de Page rank de 10 %.
- Dans deux titres `H1` ? Si c'est le cas, on majore le transfert de Page rank de 5 % par titre `H1` offrant ce mot-clé, etc.

On fait la même chose avec la page d'arrivée. Au final, notre apprenti moteur de recherche arrive à calculer un transfert de Page rank via un lien entre deux pages.

Il suffit d'adapter cet algorithme à nos besoins.

### Algorithme de cohérence

Plus la cohérence est bonne, plus la qualité du backlink est satisfaisante.

Pour assurer une bonne cohérence, les mots-clés présents dans le libellé de texte du lien doivent être répartis sur l'ensemble des deux pages. Mais il ne faut pas tomber dans le spamdexing : multiplier les mots-clés partout en grand volume. Cela constituerait une

anomalie statistique ; il y aurait trop de densité. On peut appeler cela une optimisation trop agressive. Elle pourrait entraîner une désindexation de la page par Google.

Nous tenons notre approche pour concevoir un algorithme capable de générer un nuage de liens. Cette base de réflexion pourra être perfectionnée au fil du temps et de ses acquisitions de connaissances par chaque lecteur.

En évaluant les mots-clés de deux pages, on peut évaluer si un backlink composé de mots-clés pourrait les relier.

C'est un bon début mais insuffisant. Il va falloir compléter cette base d'algorithme avant de pouvoir passer à un premier codage.

## 3.4 Algorithme, nuages de liens et landing pages

Le principe est le suivant : Notre ébauche d'algorithme vue précédemment peut tirer des liens cohérents entre pages du site. Exemple : toutes les pages traitant du chocolat sous toutes les formes possibles dans un site de recettes de cuisine sont interconnectées par le même nuage de liens titré ainsi : `Toutes nos recettes à base de Chocolat`.

Sous le couvert de cette assistance à la navigation contextuelle sur le site, le codeur référenceur peut mettre en place une architecture pour manipuler les transferts de Page rank et les leviers sur les mots-clés associés.

Plusieurs formes de transferts de PR sont possibles. En voici quelques-unes :

- Équitable : le Page rank boucle entre les pages d'un même thème. Ce qui sort par un lien d'une page reviendra via un autre lien.
- 20:1 20 pages d'un même thème sont interconnectées par le même nuages de liens présent sur chacune d'entre elles. Chaque page a en plus de son nuage, un lien en dur vers une page spéciale : la landing page. Cette page n'a pas de nuages de liens vers ces 20 pages. Donc elle reçoit beaucoup et donne moins. Cette landing page se retrouvera donc très favorisée par rapport aux autres pages de même thème, sur ses mots-clés, face aux algorithmes de Google.
- Une combinaison des deux précédentes formes. Une forme sert à lisser et à moyenner le Page rank entre pages de même thématique. On sait ainsi sur quoi tabler. Les anomalies de PR par Page sont ainsi partiellement corrigées car lissées entre ces pages. L'autre forme de transfert sert à favoriser une page au détriment des autres pages.

Dans un nuage de liens, chaque lien pousse de un à quelques mots-clés. Comme les pages dans un nuage bien conçu sont de la même thématique, la cohérence est en général plutôt

bonne entre la page de départ de chaque lien et sa page d'arrivée. L'efficacité du backlink est donc maximale. Le Page rank ne se perd pas et circule entre les pages du site.

La page ciblée par toutes les pages de même thème est appelée landing page, page d'atterrissage. Elle est dédiée, comme les autres pages du site, à une ou quelques expressions clés. L'administration d'un nuage de liens ciblant une expression clé permet de sélectionner la page d'atterrissage d'une expression clé. À terme, cette page sera la page mise en avant en premier par Google sur cette expression clé précise.

Nous en déduisons les critères d'une bonne landing page :

- Elle doit être de préférence en haut de la hiérarchie de navigation.
- Elle a un menu de navigation.
- Son URL appartient à un menu de navigation du site. Cela favorise la diffusion du PR rerouté vers cette page

⚠ Trop de liens tuent le nuage

La formule du Page rank est détaillée plus loin. Un mécanisme de cette formule limite l'efficacité d'un nuage de liens si celui-ci a trop de liens. Un nombre de 15 à 20 liens constitue une première limite maximale pour un nuage de liens.

## Algorithmes : Automatiser les liens d'un nuage

Les différents algorithmes automatisant un nuage de liens pour chaque page d'un site peuvent être plus ou moins complexes et lourds à mettre en place.

Pour améliorer et simplifier la mise en place de notre futur programme, augmenter la portée et maximiser l'efficacité de nos algorithmes, il serait bon de mettre en place un cadre de travail.

Par convention, les mots utilisés dans le title de la page, critère le plus important d'une page pour son référencement, seront tous considérés comme mots-clés stratégiques pour la page. Cela implique que ces mots-clés seront bien présents dans les différents autres critères de la page.

On pourra d'ailleurs, via un algorithme de mise en page, s'assurer d'un minimum de réalisation à ce niveau (exemple : afficher en `H2` les mots-clés du title de page quelque part sur la page).

⚠ Renforcer la cohérence.

Le contenu et la mise en page du texte affiché peuvent être insuffisamment optimisés pour les besoins du site. Quelques astuces permettent de limiter les dégâts. Afficher en *H2* dans un coin de la page le contenu du title de page permet d'assurer ou de renforcer une cohérence. Afficher dans une partie de la page en *H1* le contenu du champ *expression_clef* de la table *article* renforce la cohérence de manière encore plus précise.

Dernier point et non des moindres : pour éviter de trop forcer sur la densité et donc risquer de faire ainsi du spamdexing, le volume de texte d'une telle page doit être d'un minimum de quelques centaines de mots.

Notre algorithme va opérer ainsi :

- Une table contient les expressions clés du site. Exemple d'une expression clé pour notre exemple : création d'entreprise.
- On cherche toute page contenant cette expression clé dans son title. On obtient une liste.
- On interconnecte les 20 premières pages de cette liste par un nuage de liens.

Une ligne de code pourrait afficher par défaut en `H2` le contenu du champ *expression_clef*.

Nous allons vite rencontrer de nombreux problèmes de mise en œuvre.

Où et comment afficher en `H2` le champ *expression_clef* ? Cela implique que dès la conception du site, ce champ soit pris en compte. Il pourrait être utilisé comme titre du nuage de lien.

Avoir une liste de 20 backlinks ayant tous pour libellé `création d'entreprises` n'est guère pratique à présenter sur chaque page, et cela nous entraînerait dans un spamdexing de 20 mots-clés sur les 20 pages du nuage de liens. Il faut changer cela. Pour chaque lien, le libellé du backlink pourrait être le title de page, tout simplement.

Un nuage de 20 liens n'est guère pratique à afficher. Le plus simple consiste à créer une zone réservée au nuage de liens dans la page, ainsi qu'un conteneur XHTML/CSS destiné à en gérer l'affichage. Le bas de la page est une des possibilités où l'on pourrait poser ce conteneur XHTML/CSS.

⚠ Duplicate content tue les nuages de liens

Trop de nuages de liens identiques sont présents sur différentes pages. Passé un seuil, une page peut être désindexée. Tous ses liens sont ignorés, le PR de la page est perdu et un trou apparaît dans la stratégie de référencement.

Une astuce consiste à ne pas afficher les liens du nuage dans le même ordre.

Au-delà d'une trentaine de caractères, deux chaînes de caractères identiques peuvent entrer en conflit de duplicate content vis-à-vis de Google. Notre nuage de liens repique le contenu du title de la page de destination et on retrouve cette chaîne de caractères répétée jusqu'à 20 fois ou plus dans autant de pages.

Pour éliminer les risques trop élevés de duplicate content, plusieurs approches complémentaires sont à mettre en œuvre afin de "casser" la détection :

- Ne pas afficher les liens dans le même ordre à chaque fois.
- Limiter le nombre de caractères d'un libellé de lien de manière à en avoir moins de 35.
- Afficher de manière aléatoire un segment de N caractères consécutifs du libellé du lien. Si le texte de lien, issu donc du title de la page de destination, fait 80 caractères de long, on en affiche 45 aléatoirement qu'on extrait d'un seul bloc. On limite ainsi les longueurs et le volume de duplicate content des title de liens
- Il faut s'assurer que l'expression clé à pousser est présente dans le bloc retenu. Nous l'évoquons ci-après.

Limiter le nombre de mots peut entraîner l'élimination de notre expression clé. Il faudra donc s'assurer de sa présence dans l'affichage du libellé. On peut afficher les mots environnant notre expression clé, ou uniquement les 4 à 5 premiers mots-clés. Et s'ils ne sont pas présents, on préfixe l'extrait du title de page par le contenu du champ *expression_clef* `"Création d'entreprise : "` et on affiche un extrait de 2 à 4 mots-clés maximum.

Tout codeur pourra imaginer sa propre solution à ce problème.

⚠ Plus de 20 liens en nuage pose de gros problèmes

On peut percevoir ici qu'il vaut mieux se limiter à une vingtaine de liens en nuage sur une expression clé donnée. Passer à plus entraînerait le codeur référenceur vers des limites. Soit on commence à s'accrocher dans des duplicate contents à cause de la répétition de la même string sur trop de page ; soit on expose des pages à du spamdexing. En tentant d'éloigner ces risques, on diminue la cohérence et la présence de l'expression clé.

Enfin, trop de liens tuent les liens pour cause de trop de dispersion du Page rank. Il faudra passer à d'autres algorithmes de nuages de liens pour gérer au mieux les flux de Page rank sur une expression clé stratégique sur laquelle on veut focaliser plus de 20 backlinks internes.

Nous venons de voir un algorithme de nuage de liens entre les pages. Il reste à y intégrer le lien vers la landing page.

Nous disposons d'une table où sont stockées les expressions clés exprimant la stratégie de notre site.

À chaque expression clé, associons une URL en dur, celle de la landing page. Cette URL et un court texte associé comme libellé de lien seront affiché au même endroit que le nuage de liens.

⚠ Piège, le temps et l'obsolescence des liens

Les pages interconnectées évoluent mais pas les backlink avec leurs libellés. Ils deviennent donc incohérents au fil du temps et Google les invalide. En automatisant les nuages de liens, on évite ce piège. Les liens se recalculent régulièrement. Ils restent ainsi à jour.

## 3.5 La formule "officielle" du Page rank

Cette formule a été publiée en 1998 et n'a jamais fait l'objet d'une mise à jour par Google depuis. À plusieurs reprises, Google a indiqué que le Page rank était toujours d'actualité et qu'il constituait une brique fondamentale de sa stratégie.

Sans aucun doute, la formule du Page rank a été modifiée et adaptée au Web du XXI$^{e}$ siècle. Mais peu importe, le concept reste d'actualité. Ce qui compte pour le codeur référenceur est d'en comprendre les principes afin de pouvoir appréhender comment l'utiliser le plus efficacement possible. Sans documentation officielle de Google, il ne sert à rien de polémiquer autour de la précision des calculs et des formules.

Une page *A* reçoit des liens émis par les pages *T1* à *Tn*.

Le paramètre *d* est un facteur d'amortissement pouvant être ajusté entre 0 et 1. Historiquement, en 1998, *d* a la valeur 0,85. Ce paramètre *d* a plusieurs utilités.

*C(A)* est défini comme le nombre de liens émis par la page *A* ; ce sont donc les liens sortants. Le Page rank de la page *A* est défini comme suit :

```
PR(A) = (1-d) + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))
```

Tableau 3.1 : Formule du Page rank

| Composant de la formule | Description |
|---|---|
| `PR(A)` | Le Page rank de la page *A*. Celui que nous voulons calculer |
| `PR(Tn)` | Le Page rank de la page *T*`n`. Les pages *T*, de `1` à `n`, ont chacune un lien pointant vers notre page *A*. |

Tableau 3.1 : Formule du Page rank

| | |
|---|---|
| C(Tn) | Le nombre de liens sortant de la page Tn. Chaque page T a plusieurs liens, enfin au moins un, celui qui pointe vers la page *A*. C(Tn) comptabilise le nombre de liens sortants qui vont devoir se partager, à parts égales, une partie du PR alloué aux liens sortants. |
| D | Coefficient d'amortissement. d est considéré égal à 0,85. |
| (1 – d) | Une astuce pour influencer la convergence |

Le Page rank est affiché dans la barre d'outils Google pour Firefox ou Microsoft IE.

L'échelle du Page rank est logarithmique. Cette formule évolue probablement au fil du temps et des mises à jour par Google de ses algorithmes.

Voici une échelle approximativement correcte. L'important est de comprendre la tendance et d'en maîtriser les conséquences.

Prenons une échelle logarithmique de base 10 pour simplifier nos calculs et bien illustrer notre propos.

Tableau 3.2 : Page rank affiché et calculé

| PR affiché (log base 10) | Page rank "réel", calculé à quelques décimales près, exprimé sans log |
|---|---|
| PR 0 | 0 PR < 1 |
| PR 1 | 1 PR < 10 |
| PR 2 | 10 PR < 100 |
| PR 3 | 100 PR < 1000 |
| PR 4 | 1000 PR < 10000 |
| etc. | |

Chaque niveau de Page rank est 10 fois plus élevé que le niveau précédent. Exprimé simplement, il faut 10 fois plus de travail et de backlinks vers une page pour passer de PR4 à PR5 que pour passer de PR3 à PR4. Pour être plus précis : il faut 10 fois plus de PR transféré via les backlinks pour passer du niveau PR4 à PR5 que pour passer de PR3 à PR4.

Non seulement le volume de backlinks compte, mais aussi leurs qualités prenant en compte la cohérence.

La formule du PR de Google est récursive. Le calcul de PR en cours a un impact sur les PR des autres pages utilisées pour calculer le PR de notre page. Il faut donc le refaire, ce qui modifie à nouveau le résultat, etc. Pour obtenir un résultat juste, il faudrait attendre que le résultat converge vers une valeur à peu près stable. Nous venons de voir une des raisons expliquant pourquoi Google met à jour la petite barre verte seulement quelquefois par an. Il est inutile de la publier trop souvent, cela représenterait un travail important en mises à jour pour une évolution faible. Une mise à jour de la barre verte de la Google bar quelques fois par an est suffisante.

- *Première conséquence de cette formule* : si un site n'évolue pas, son PR se dégrade au fil du temps. En effet, le nombre de pages dans Internet augmentant continuellement, le nombre de liens `C(Tn)` aura tendance à augmenter et le PR transféré depuis chaque page *T*n diminuera.
- *Deuxième conséquence* : le volume de backlinks depuis des pages à faible PR ou depuis des pages surchargées de backlinks entraîne beaucoup de travail pour un faible résultat. Tant qu'à investir du temps, autant en investir une partie dans des backlinks de qualité : bon PR, réel contenu, peu de backlinks externes.
- *Troisième conséquence* : Google n'a nul besoin de disposer des PR de toutes les pages. En itérant plusieurs fois le calcul, chaque itération fait converger le PR vers la valeur finale *juste*. En attendant, il peut prendre une valeur intermédiaire n'ayant pas fini de converger ou une valeur de 0.

Détaillons le fonctionnement. Les pages *T*n ont chacune un lien pointant vers notre page *A*. Peu importe si nous n'avons pas encore toutes les pages *T*n et si le Page rank de certaines de ces pages n'est pas encore finalisé.

En effet, imaginons un cas simple et fort courant : une boucle dans les liens.

La formule est itérée une première fois, le lien de A vers B vient juste d'être cartographié. Une fraction du Page rank passe de la page A à la page B. La formule est itérée une nouvelle fois. Les calculs prennent en compte le nouveau PR de B.

En quelques itérations, le PR de chaque page va converger vers sa valeur finale.

Nous explorerons les conséquences de cette formule pour le référenceur :

- Google l'utilise pour lutter contre le spam de backlinks.
- Comment exploiter le Page Rank de manière optimale pour obtenir un bon positionnement sur les mots-clés stratégiques des pages d'un site web ?

## 3.6 Piloter la stratégie de référencement

Piloter la stratégie de référencement naturelle du site consiste à sélectionner chaque expression clé nécessaire au site et à l'associer à un groupe de pages existantes, afin de pousser en avant une page précise dédiée à cette expression clé.

Exprimé autrement, à une expression clé stratégique donnée :

- On doit associer les pages existantes ayant cette expression clé dans leurs `title` de page puis on doit interconnecter ces pages existantes entre elles via un nuage de liens renforçant le référencement de ces pages les unes les autres.
- Dans chaque nuage de lien placé dans ces pages associées, on place un backlink vers la landing page (page d'atterrissage) dédiée à cette expression clé.

Nous allons mettre en place l'algorithme de génération de nuage de liens sur notre site web exemple. En même temps, tout en administrant le fonctionnement de cet algorithme, nous piloterons la stratégie de référencement naturelle de notre site.

Rappelez-vous, un script *article.php* affiche une liste de résultats. En cas de clic sur un résultat précis, un script *zoom.php* affiche le détail de l'article ainsi sélectionné.

Le nuage de liens sera donc à placer en bas de chaque page affichée par *zoom.php*.

Regardons le code source et l'algorithme dans le détail. Notre code source de *zoom .php* vu dans les pages précédentes se terminait ainsi :

```
    <?php
        view_article($line);
?>

     <!--fin page texte -->
   </div>
   <!-- fin pagecentre -->
 </div>

  <?php
  $map = MiseEnPage::bas() ;
?>
```

Après l'appel à la fonction `view_article($line);` et avant la fin de la page, il va falloir insérer le code gérant le nuage de liens.

## Administrer les nuages de liens

Le générateur de nuages de liens a besoin de diverses informations :

- Quelles sont les expressions clés présentes en title de page à prendre en compte ?
- Quelle est la landing page de chaque expression clé ?

Sur notre site web exemple, nous devons créer un nuage de liens pour toute page ayant une variante d'expression clé de création d'entreprise. Il faut interconnecter jusqu'à 20 pages maximum avec un nuage de liens en évitant les pièges liés au duplicate content.

La table suivante, *expressions_clés*, à ne pas confondre avec le champ *expression_clef* de la table *article*, va nous permettre de gérer les différentes expressions clés déclinées depuis `création d'entreprise` et `reprise entreprise` :

```
--
-- Structure de la table 'expressions_clefs'
--

CREATE TABLE 'expressions_clefs' (
  'id' int(11) NOT NULL auto_increment,
  'expression' varchar(128) NOT NULL,
  'landingURL' varchar(256) NOT NULL
    COMMENT 'optionnel - url en dur visée par expression',
  KEY 'id' ('id')
) ENGINE=InnoDB
  DEFAULT CHARSET=latin1 AUTO_INCREMENT=7 ;

--
-- Contenu de la table 'expressions_clefs'
--

INSERT INTO
   'expressions_clefs'
   ('id', 'expression', 'landingURL')
   VALUES
(1, 'creation entreprise', 'creation-reprise-entreprise.php'),
(2, 'reprise entreprise', 'creation-reprise-entreprise.php'),
(3, 'creation d entreprise', 'creation-reprise-entreprise.php'),
(4, 'creation d''entreprise', 'creation-reprise-entreprise.php');
```

En effet, plusieurs variantes sont possibles autour de cette expression clé `Création d'entreprise`.

Nous retenons l'approche simpliste mais aisée à comprendre de saisir en table chaque variante. Une autre approche plus sophistiquée aurait consistée à identifier la présence de ces 2 mots dans le *title*.

Les avantages et inconvénients des deux approches sont opposés.

Une ligne par variante d'une expression clé dans la table permet d'avoir de multiples nuages de 20 liens maximum sur des expressions clés voisines : création entreprise, création d'entreprise, etc.

On augmente ainsi la volumétrie des liens internes.

Les inconvénients :

- Risque de quelques duplicate contents involontaires mais de faible ampleur sur quelques pages.
- Travail fastidieux. Il faut penser à toutes les variantes possibles.

⚠ Conversion des 2 strings

On peut diminuer ce travail fastidieux de recherche de variantes d'une expression clé en convertissant les 2 strings lors des comparaisons afin de les mettre sur un même jeu de caractères via la fonction `html_entity_decode()` qui convertit toutes les entités HTML en caractères normaux. On évitera ainsi de comparer `é` à `é`, son pendant HTML, par exemple, et d'avoir une erreur de comparaison.

On peut aussi les convertir en caractères non accentués et ainsi simplifier encore plus les comparaisons et les variantes. Nous verrons cela dans le code source.

Dans notre exemple, nous préfèrerons la première solution, un enregistrement par variante de l'expression clé. Cette approche est porteuse de plus d'évolution et d'adaptabilité grâce à la possibilité de gérer plus de nuages de liens autour des différentes déclinaisons d'une expression clé.

## Éviter que notre site web ne s'écroule

Est-il envisageable de construire le nuage de liens à chaque affichage d'une page par *zoom.php* ?

La construction de chaque nuage de liens nécessite de très nombreux accès à la base de données. Cela pénalise les temps de réponse et surcharge notre serveur. L'ajout d'article dans la base n'est pas une opération très fréquente. L'absence de quelques articles durant quelques heures, dans les nuages de liens, n'est pas critique.

Il nous faut donc construire les nuages de liens pour chaque période de 24 heures. Ainsi, on ne va pas refaire le même calcul à chaque demande de page. Ce calcul est effectué une fois et on stocke le résultat dans une table qu'on interrogera à chaque affichage de page via *zoom.php*.

Un programme dont le lancement est automatique, un batch dans le jargon des informaticiens, utilisera donc les tables MySQL pour construire dans une table dédiée, *exp_clef_dynamique*, le nuage de chaque page. Ainsi pour afficher le nuage de liens, il suffira d'interroger cette table. En un seul accès à MySQL, on obtiendra les informations permettant de générer le nuage de liens. En temps réel, sans la préconstruction de tous les nuages en batch, chaque affichage de page impliquerait un traitement incluant des dizaines, voire des centaines de requêtes SQL ; notre serveur web se serait effondré rapidement.

## Préconstruire les nuages de liens

Voici l'exportation de la table *exp_clef_dynamique* :

```
--
-- Structure de la table 'exp_clef_dynamique'
--

CREATE TABLE 'exp_clef_dynamique' (
  'id' int(11) NOT NULL auto_increment,
  'id_article' int(11) NOT NULL,
  'libelle_BL' varchar(128) NOT NULL,
  'id_article_cible' int(11) NOT NULL,
  UNIQUE KEY 'id' ('id')
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=48 ;
```

Chaque affichage par *zoom.php* affiche un article unique identifié par une clé unique : ID de la table *article*.

La table *exp_clef_dynamique* contiendra l'ID *article* de la page où poser un backlink du nuage de liens. Chaque backlink à poser est décrit sous la forme d'un ID *article*, le champ `id_article_cible`, et du libellé texte de ce backlink, `libelle_BL`.

Un nuage de liens est donc un ensemble de triplets (`id_article`, `libelle_BL`, `id_article_cible`) dont le `id_article` est identique.

C'est le script PHP *batch_exp-clef_dynamique.php* qui va remplir cette table des nuages de liens à construire. Ce programme peut être lancé une fois par jour ou plus, automatiquement ou manuellement.

Une autre table est remplie par ce même script PHP : `landing_pages_id_article_url` :

```
--
-- Structure de la table 'landing_pages_id_article_url'
--

CREATE TABLE 'landing_pages_id_article_url' (
  'id' int(11) NOT NULL auto_increment,
```

```
  'id_article' int(11) NOT NULL,
  'libelle_lien' varchar(255) NOT NULL,
  'url_lien' varchar(255) NOT NULL,
  PRIMARY KEY ('id'),
  KEY 'id_article' ('id_article')
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=19 ;
```

Chaque nuage de liens posé sur une page contiendra un backlink vers une landing page. Cette table va nous permettre de gérer cela.

Via chaque ID d'un article, depuis le script *zoom.php*, on pourra afficher dans le nuage de liens ce backlink avec le libellé `libelle_lien` qui pointera vers la page `url_lien`.

Examinons le code source du script PHP *batch_exp-clef_dynamique.php* :

```
require_once("php_inc/MiseEnPage.inc.php");
require_once("php_inc/init.inc.php");
require_once("php_inc/biblio_inc.php");
?>

<h1>gestion dynamique des expressions clefs
entre pages articles</h1>

<?php
// vider la table que l'on va reconstruire
// les liens en nuage
$table = "exp_clef_dynamique" ;
$sql = "TRUNCATE TABLE " . $table . " ;" ;
$id_global = 1 ; // sera le ID de chaque enregistrement
                 // dans table exp_clef_dynamique
$id_landing_page = 1; // sera le ID de chaque row
                    // dans la table
                    // landing_pages_id_article_url
if (!($request = mysql_query($sql) ) )
  {
  echo 'erreur : table ' . $table . '
 :: impossible de vider la table <br />' ;
  exit ;
  }
echo 'info : table ' . $table . ' :: vidée
       <br /><br /><br />' ;

// vider la table que l'on va reconstruire :
// la landing page de chaque nuage
$table = "landing_pages_id_article_url" ;
$sql = "TRUNCATE TABLE " . $table . " ;" ;
if (!($request = mysql_query($sql) ) )
  {
```

```
  echo 'erreur : table ' . $table . '
 :: impossible de vider la table <br />' ;
  exit ;
  }
echo 'info : table ' . $table . '
 :: vidée <br /><br /><br />';
```

On définit les bibliothèques de fonctions via des classiques `require_once`.

Pour suivre la construction des tables et permettre tout débogage, on affichera des informations sur une page HTML via de classiques `echo` ou `print` dans le script PHP.

`$id_global` et `$id_landing_page` seront les ID respectifs de chaque enregistrement que nous allons écrire avec le batch *batch_exp-clef_dynamique.php* dans les deux tables *exp_clef_dynamique* et *landing_pages_id_article_url*.

On vide les tables *exp_clef_dynamique* et *landing_pages_id_article_url* puisque le batch doit les reconstruire complètement toutes les deux.

### *Boucle de niveau 1*

Traiter chaque expression clé pour essayer de mettre en nuage :

```
$sql = "select * from 'expressions_clefs' ;" ;
if ($request = mysql_query($sql))
// on a toutes les exp clef de expressions_clefs à explorer
  {
  while ($line = mysql_fetch_array($request)) // boucle niveau 1
    {
    $expression = $line["expression"] ;
    $url_lien = $line["landingURL"] ;
    $sql = "select * from 'articles' ;" ;
    if ( !($request2 = mysql_query($sql))
      die('Erreur SQL !'.$sql.' '.mysql_error());
```

On récupère chaque expression clé depuis la table *expressions_clefs*. Pour chacune d'elles, on va essayer de trouver une page où déposer le nuage et toutes les pages affichant un article qui seraient cohérentes avec notre lien et la page de départ.

Pour chaque expression, on met de côté les informations afin de construire un lien vers la landing page de l'expression clé. Ces informations sont :

- Le libellé du lien. Ici, nous prendrons l'expression clé elle-même : `$expression`.
- L'URL du lien : `$url_lien`.

Nous avons de quoi construire le lien sous la forme `<a href="$url_lien "> $expression </a>`.

### *Boucle de niveau 2*

Recherche d'une page dont le title possède une sous-chaîne (*substring*) correspondant à l'expression clé, pouvant donc recevoir un nuage de liens :

```
// boucle niveau 2
while ($line_article = mysql_fetch_array($request2))
  {
  $id_article = $line_article["id"] ;
  $s = $line_article["title"] ;
  $s_nohtml = html_entity_decode($s) ;
// plus de HTML, on passe tout en iso standard basique

  $expression_nohtml = html_entity_decode($expression);
  $s_noAccent
    = strtr($s_nohtml,
            'ÁÀÂÄÃÅÇÉÈÊÍÏÎÌÑÓÒÔÖÕÚÙÛÜÝ',
           'AAAAAACEEEEIIIINOOOOOUUUUY');
  $s_noAccent
    = strtr($s_noAccent,
            'áàâäãaçéèêëíìîïñóòôöõúùûüýÿ',
            'aaaaaaceeeeiiiinooooouuuuyy');
// du pur ascii 7bit - on simplifie les comparaisons

  $expression_noAccent
     = strtr($expression_nohtml,
             'ÁÀÂÄÃÅÇÉÈÊËÍÏÎÌÑÓÒÔÖÕÚÙÛÜÝ',
             'AAAAAACEEEEIIIINOOOOOUUUUY');
  $expression_noAccent
     = strtr($expression_noAccent,
             'áàâäãaçéèêëíìîïñóòôöõúùûüýÿ',
             'aaaaaaceeeeiiiinooooouuuuyy');

// on cherche expression clef dans title : oui ou non.
  if (stristr( $s_noAccent, $expression_noAccent ) )
    {
    // si oui, on repère cet article son id),
    // on construit libellé de lien puis faut chercher
    // tous les autres articles à viser
    // on trace ce qui se passe - pratique pour le debug
    // et pour comprendre
    echo ' boucle niv DEUX 2 :
         <b>' . $expression_noAccent . '</b>
         est inclus dans cette string
         ==>> <i> ' . $s_noAccent . '</i> <br>' ;
   // on construit le BL
    $id_depart = $line_article["id"] ;
```

```
    // maintenant il faut lister tous les
    // ID articles à cibler
    $sql = "select * from 'articles' ;" ;
    if ($request3 = mysql_query($sql))
    {
    // on commence par le lien vers la landing page
    $libelle_lien = $expression ;
    $sql2 = "INSERT INTO 'landing_pages_id_article_url'
            VALUES('$id_landing_page','$id_article',
                   '$libelle_lien' , '$url_lien') " ;
    mysql_query($sql2)
      or die('Erreur SQL !'.$sql.' '.mysql_error());
    echo '<br /> article n ' . $id_article
         . ' ' . $libelle_lien . ' ===>>>> '
         . $url_lien . '<br />' ;
      $id_landing_page++ ;

      $compteur_BL = 0 ;
// on compte le nombre de BL que l'on va mettre dans
// la table exp_clef_dynamique pour CE nuage de liens.
// On ne dépasse pas 20
```

Pour chaque enregistrement d'un article (la vue `$line_article`) :

- On récupère son ID. Il nous sera nécessaire au cas où la page serait en adéquation avec notre recherche sur le title.
- On récupère son title de page dans la variable `string $s`.
- On transforme le title et l'expression en texte iso sans code HTML (ou un `é` s'exprime `é` par exemple) via la fonction `html_entity_decode`.
- On élimine tous les caractères accentués via la fonction `strtr()`.
- On recherche dans la chaîne (variable string `$s_noAccent`) contenant le *title* la sous-chaîne `$expression_noAccent`.
- S'il y a présence de l'expression clé dans le title de la page affichée via *zoom.php*, on a une page où poser notre futur nuage de liens.

Pourquoi transformer en caractères latins basiques les contenus de la chaîne (le *title* de page) et de la sous-chaîne (l'expression clé à chercher dans le *title* de page) ? Pour simplifier la gestion du référencement et fiabiliser nos comparaisons. Il existe de multiples déclinaison autour d'une expression clé comme `création entreprise`.

`Création entreprises/creation entreprise/création d'entreprise/crétion entreprise/`, etc. Avec accent, au pluriel, en mode HTML, de nombreuses variantes existent. En éliminant tout code HTML et toute présence d'accent, on simplifie plusieurs choses :

- La saisie à imposer dans le title et les tests liés dans le script PHP. Peu importe désormais que ce soit saisi en HTML ou non. La présence des caractères accentués importe peu également. Le traitement appliqué nous débarrasse de ces problèmes. Nos comparaisons fonctionneront.
- L'administrateur de la table expression n'a plus qu'à se préoccuper des variantes de texte : présence ou non d'un `d` avec apostrophe, sans apostrophe, création au pluriel ou au singulier, etc.

L'autre point concernant l'ordre des opérations porte sur la fiabilité et la pérennité de cette séquence de code. Deux problématiques sont à régler :

- Il existe une multitude de jeux de caractères liés à Internet ou non : sur 8 bits, 16 bits, 32 bits. En les recodant dans une police latine, on se débarrasse de toutes les manières différentes de coder un même caractère.
- La deuxième problématique porte sur les accents. Un même mot peut être orthographié avec des caractères accentués divers, justes ou erronés. La transformation de tout caractère accentué vers sa version non accentuée fiabilise et simplifie le travail de comparaison.

Revenons à notre code source. Nous avons identifié un article où poser un nuage de liens. Il nous faut tracer son ID d'article : la page de départ de chaque lien du nuage. Il nous faut maintenant identifier tous les liens à poser dans le nuage dans une boucle de niveau 3. Nous préparerons cette boucle sur la table article.

On charge tous les articles de la table *article* pour préparer la recherche, cohérence oblige, d'autres articles ayant cette même expression clé dans le title de page affichée par *zoom.php*.

Pour débuter, on insère dans la table *landing_pages_id_article_url* les informations qui nous permettront de coder à la volée le lien vers la landing page de l'expression clé :

- `$id_landing_page` : l'ID unique de chaque enregistrement de cette table ;
- `$id_article` : l'ID de l'article où poser le lien ;
- `$libelle_lien` : le libellé du lien ;
- `$url_lien` : l'URL de la landing page.
- `$compteur_BL`. On doit limiter à une vingtaine le nombre de liens dans un nuage. Ce compteur va nous aider à contrôler le nombre de liens lors de la construction à la

volée du nuage. Dans une version améliorée, les éventuels enregistrements d'articles restant dans la vue en cours de traitement et compatibles avec l'expression clé pourraient être utilisés pour un autre nuage. Dans le cadre de cet exemple, nous nous contenterons du progrès déjà notable d'avoir un seul nuage de liens pour mettre en landing page une expression clé précise.

### *Boucle de niveau 3*

On parcourt les enregistrements de la vue et on compare l'expression à la chaîne *title* de chaque enregistrement, comme dans la boucle de niveau 2 :

```
// boucle niveau 3 - génération du nuage de liens
// pour l'article id_depart
while ($line_article_cible=mysql_fetch_array($request3))
  {
  $s0_nohtml
      = html_entity_decode($line_article_cible["title"]) ;
  $s0_noAccent
      = strtr($s0_nohtml,
        'ÁÀÂÄÃÇÉÈÊËÍÌÎÏÑÓÒÔÖÕÚÙÛÜÝ',
        'AAAAAACEEEEEIIIINOOOOOUUUUY');
  $s0_noAccent
      = strtr($s0_noAccent,
        'áàâäãaçéèêëíìîïñóòôöõúùûüýÿ',
        'aaaaaaceeeeiiiinooooouuuuyy');
  if (stristr($s0_noAccent, $expression_noAccent) )
    {
// il faut entrer une ligne dans la table
// expressions clef dynamiques
    echo ' boucle niv TROIS 3 : '
          . $expression_noAccent
          . ' est inclus dans cette string ==>> '
          . $line_article_cible["title"] . ' <br>' ;

    $libelle_final = $line_article_cible["title"] ;
    $compteur_BL++ ;
    echo $compteur_BL
          . ' éme de ce nuage de liens BL==>>'
          . $libelle_final . ' <<==   '
          . ' ID cible =' . $line_article_cible["id"]
          . '<br>' ;
    $cible = $line_article_cible["id"] ;
    $id_global++ ;
    $sql2 = "INSERT INTO 'exp_clef_dynamique'
            VALUES('$id_global' ,
                  '$id_depart',
```

```
                    '$libelle_final' ,
                    '$cible') " ;
    if ($line_article["id"]==$line_article_cible["id"])
// si on ne pointe pas sur la même page
/: de départ / arrivée le BL est ok, on peut l'insérer
      {
      echo ' **** boucle sur lui-même - pas de BL '
            . $line_article["id"]
            . ' == '
            . $line_article_cible["id"] . '<br />' ;
      }
    else // on insère le BL à générer ds la table
      {
      if ($compteur_BL <21)
// nuage de liens ne doit pas dépasser 20
       {
       mysql_query($sql2)
        or die('Erreur SQL !'.$sql.' '.mysql_error());
       }
      else
        {
        echo '<br /> Nuage de liens saturé
 : 20 max<br />';
        }
      }
    }
  }
```

À chaque enregistrement compatible avec notre expression clé, on récupère les informations nécessaires à la construction du lien. On va les insérer dans la table *exp_clef_dynamique* :

- `$id_global` : l'ID de chaque enregistrement de cette table. On l'incrémente à chaque insertion dans la table.
- `$id_depart` : cette variable a été instanciée dans la boucle 2. C'est l'ID de la page de départ du nuage de liens.
- `$libelle_final` : on récupère le title de la page cible qui sera donc le libellé de notre lien vers elle. Difficile de faire plus cohérent. Mais cette précision nous limite à une vingtaine de liens maximum dans un nuage pour une expression clé précise.
- `$cible` : c'est l'ID de la page cible. Grâce à lui, on pourra construire l'URL vers la page.

Si les deux ID article et ID article cible sont identiques, on est en fait sur la page de départ dont le title contient bien sûr l'expression clé. On ne la traite pas.

Si `$compteur_BL` atteint 21, on arrête d'insérer des liens pour ce nuage. Une amélioration future serait à mettre en place ici pour créer un nouveau nuage de liens qui aurait pour objectif d'utiliser un seul mot-clé composant l'expression clé en cours. C'est un excellent exercice à faire pour mettre en œuvre et améliorer ce que vous apprenez dans ce livre.

Ci-après, on ferme proprement les accolades et la fin de la page HTML affichant les messages de suivi de la construction des deux tables.

```
                                    }
                                  }
                        }

        }
    }
echo '<br><br> == fin du batch ==' ;
?>
  </body>
</html>
```

Nous allons maintenant analyser le code source de *zoom.php*. Nous avons déjà vu le code source de la version sans gestion des nuages de liens au chapitre précédent ? Nous allons revisiter ce script avec cet ajout.

## Afficher un nuage de liens : zoom.php

Nous avons modifié le script PHP *zoom.php* afin de mettre en service l'affichage du nuage de liens.

Les liens à afficher sont contenus dans deux tables distinctes :

- La table *landing_pages_id_article_url* contient le backlink pointant vers la landing page ciblée.
- La table *exp_clef_dynamique* contient les liens du nuage à afficher dans la page de l'article.

Le script doit accéder aux éventuels enregistrements concernant l'ID article qu'il affiche dans ces 2 tables. La clé d'accès est la même pour les deux tables : le champ *id_article* de la page article en cours de publication par *zoom.php*.

## Éviter les duplicate contents involontaires

Un piège guette le codeur ne faisant pas attention aux URL.

Étant donné qu'il existe plusieurs rubriques pour une bonne partie de chacun des articles de la table, il existe plusieurs URL pour accéder à un même article. Il va falloir rester dans le choix opéré en début de projet pour gérer cette problématique. La seule URL valide est celle de la rubrique primaire de l'article. À travers la navigation sur le site, les bots de Google n'ont découvert que les liens vers des articles sous les URL de leurs rubriques principales. Il faut rester dans ce schéma.

Un autre piège attend le codeur : faire des nuages de liens trop semblables. Un nuage ressemble à un grand bloc de texte contenant des ancres. Avec le même grand bloc de texte sur une vingtaine de pages, un duplicate content sur une partie de la page peut survenir, entraînant la désindexation possible d'une ou plusieurs page. Pour limiter les risques, affichons les liens dans un ordre différent. Ce sera une première mesure pour lutter contre les duplicate contents.

Le début de la version simple du script *zoom.php* et de la version avec nuage de liens sont strictement identiques.

Les modifications liées à la gestion du nuage de liens apparaissent juste après la ligne :

```
            view_article($line);
```

Cette ligne affiche l'article. Dans la version simple vue au chapitre précédent, on ferme la page. Désormais, on passe à la gestion du nuage de liens :

```
view_article($line);
// on commence par le lien vers la landing page
$id_article_depart = $line["id"] ; //
$sql_landing_page
      = "select *
          from landing_pages_id_article_url
          where id_article = " . $id_article_depart
                              . " ;" ;
if ( !($request2 = mysql_query($sql_landing_page)) )
      die('Erreur SQL !'.$sql.' '.mysql_error());
if ($landing_page = mysql_fetch_array($request2) )
// si on a au moins un lien à afficher
  {
// div de début des BL dynamiques
  echo '<div class="jcmbparalarge">' ;
  echo 'Liens connexes à : ' . $line["url"]
      . '<br /><br />' ;
```

```
  echo '<a href="'. $landing_page["url_lien"] . '">'
        . $landing_page["libelle_lien"]
        . '</a> <br /><br />' ;

  $sql_BL_a_poser =
      "select * from 'exp_clef_dynamique'
                where id_article = "
                    . $id_article_depart
                    . " order by md5(rand()) ;" ;
  if ($request_BL_a_poser=mysql_query($sql_BL_a_poser))
  // tous les BL à poser pour l'article où nous sommes
    {
    while ($line_UN_BL_a_poser
      = mysql_fetch_array($request_BL_a_poser))
// liste des liens en nuage à poser pour cet article
      {
      $sql5 = "select *
           from articles
            where ID = "
             . $line_UN_BL_a_poser["id_article_cible"]
             . " ; " ;
      $request5 = mysql_query($sql5);
      $line5 = mysql_fetch_array($request5) ;
      $url_article_cible = $line5["url"] ;
      //
      // quelle est la rubrique PS (primaire)
      // de cet article cible ?
      $sql3 = "select *
              from article_rubrique
              where id_article = "
             .$line_UN_BL_a_poser["id_article_cible"]
             .' AND primaire > 0 ' . " ; " ;
      if ($request3 = mysql_query($sql3))
        {
        // quel est le id_rubrique primaire
        // de cet article, càd la seule URL
        // href valide du site
        // (pour éviter les dup content).
        $line3 = mysql_fetch_array($request3) ;
        //quel est le détail de la rubrique ?
        // là on a juste son ID càd son n°
        // (dont on a aussi besoin pour
        // construire l'URL du BL
        $sql4 = "select * from rubrique
                 where ID = " . $line3["id_rubrique"]
                              . " ; " ;
        if ($request4 = mysql_query($sql4))
```

```
          {
          $line4 = mysql_fetch_array($request4) ;
          //
          // quelle est l'URL à générer
          // depuis la table rubrique ?
          // il faut article/url de l'article cible
          echo ' <a href="article/'
                  . $url_article_cible
                  . '.html&src='. $line4["nom"]
                  . '">'
                  . $line_UN_BL_a_poser["libelle_BL"]
                  . '</a>' ;
          }
        }
      else
        {
        echo '<br /><b>
              Pas de rubrique primaire pour cet article ?
              Impossible / bogue </b> ' ;
        }
      echo '<br /><br />';// br a chaque passe du while
      } // accolade fin du while

    }
// fin de BL dynamiques
  echo '</div>' ; // div de FIN des BL dynamiques
}
?>
```

On commence par générer le backlink vers la landing page. Il nous faut pour cela les informations stockées dans la table *landing_pages_id_article_url*. Le code ci-après extrait ces informations.

```
view_article($line);
// on commence par le lien vers la landing page.
$id_article_depart = $line["id"] ; //
$sql_landing_page =
        "select * from landing_pages_id_article_url
        where id_article = "
                          . $id_article_depart
                          . " ;" ;
if ( !($request2 = mysql_query($sql_landing_page)))
    die('Erreur SQL !'.$sql.' '.mysql_error());
// on récupère notre landing page
if ($landing_page = mysql_fetch_array($request2))
  // si on a au moins un lien à afficher
```

Si on a au moins un lien à afficher dans le nuage, le backlink vers la landing page, on affiche le conteneur, le titre du nuage et le backlink avec le script ci-après. Dans cet exemple et dans la méthodologie de l'exercice, `$lien["url"]` contient obligatoirement les mêmes informations que le champ *expression_clef* de la table *article*. Cela nous dispense d'afficher `expression_clef` dans *zoom.php* comme évoqué précédemment.

```
if ($landing_page = mysql_fetch_array($request2))
  // si on a au moins un lien à afficher
  {
  // div de début des BL dynamiques
  echo '<div class="jcmbparalarge">' ;
  echo 'Liens connexes à
 : ' . $line["url"] . '<br /><br />';

  echo '<a href="'
        . $landing_page["url_lien"]
        . '">'
        . $landing_page["libelle_lien"]
        . '</a> <br /><br />' ;
```

Il nous reste à afficher, ligne par ligne, les liens du nuage :

```
  $sql_BL_a_poser =
        "select * from 'exp_clef_dynamique'
        where id_article = "
                        . $id_article_depart
        . "
        order by md5(rand()) ;" ;
// tous les BL à poser pour l'article où nous sommes
  if ($request_BL_a_poser=mysql_query($sql_BL_a_poser))
    {
```

Par défaut, une requête SQL renvoie les enregistrements dans l'ordre où ils sont trouvés dans la table.

Or pour avoir un affichage aléatoire des liens dans le nuage, il faut stocker dans un autre ordre les articles retournés dans la vue.

`order by md5(rand())`. Cet ajout dans la requête SQL déclenche un renvoi des articles correspondant avec un ordre issu du hasard.

## Risque d'effondrement du serveur web

Attention, la fonction `rand()` dans un `ORDER BY` dans MySQL est dangereuse à utiliser. MySQL va affecter une valeur *random* (aléatoire) à tous les enregistrements de la table

puis exécuter la clause `where`. Sur un site web avec des tables volumineuses en enregistrements et des utilisateurs nombreux, on peut déclencher un effondrement des performances, prélude à un crash du serveur.

⚠ Illustration de surcharge due à rand() dans MySQL

Votre table *article* contient 5 000 enregistrements et votre clause `where` en isolera une douzaine. La fonction `Rand()` va être exécutée sur toute la table, la table va être triée et enfin la clause `where` sera appliquée pour en extraire les lignes voulues dans le désordre souhaité. Cela consomme beaucoup trop de ressources disques, mémoire, processeur. Les temps de réponse seront lents, les performances globales du serveur fortement perturbées.

Pour notre site exemple, ce codage à base de `md5(rand())` est suffisant. Pour un site avec des bases plus importantes et de nombreux utilisateurs affichant des pages, il est obligatoire de coder son propre algorithme de tri aléatoire dans un tableau. Une piste pour un algorithme peu gourmand en ressources :

- Dans un tableau à deux dimensions : une pour les nombres, une pour les chaînes de caractères.
- Dans la colonne chaîne de caractères, copier les enregistrements de la ressource `$sql_BL_a_poser` directement sous la forme de lien `<a href= …</a>` à raison de un lien par entrée dans le tableau.
- Générer un nombre aléatoire à chaque ligne du tableau et le déposer dans la colonne numérique du tableau bidimensionnel.
- Trier le tableau .
- Afficher les chaînes du tableau.

L'endroit idéal pour faire cela est à la fin du script. Au lieu de lancer l'affichage avec des `echo`, on écrit dans le tableau à deux dimensions, on traite pour mettre dans un désordre aléatoire, on affiche.

Il existe plusieurs variantes autour de cette approche. Chacun pourra modifier ou améliorer à sa convenance. Dans cette ébauche de routine à coder, le traitement est effectué uniquement sur la vingtaine de liens autorisés, ce qui limite fortement la consommation de ressources.

## Boucle affichant le nuage de liens

La vue `$line_UN_BL_a_poser` contient un enregistrement par lien. Tous les enregistrements sont dans le désordre :

```
while ($line_UN_BL_a_poser
            = mysql_fetch_array($request_BL_a_poser))
// liste des liens en nuage à poser pour cet article
  {
  $sql5 = "select *
        from articles
        where ID
        = " . $line_UN_BL_a_poser["id_article_cible"]
            . " ; " ;
  $request5 = mysql_query($sql5);
  $line5 = mysql_fetch_array($request5) ;
  $url_article_cible = $line5["url"] ;
```

Pour chacun de ces liens à poser, il va falloir construire le lien avec les mots-clés nécessaires pour l'URL rewriting et éviter des duplicate contents.

Chaque lien doit utiliser les mots-clés prévus. La requête `$request5` permet de récupérer les mots-clé URL de l'article ciblé.

Rappel : le lien doit avoir la forme suivante :

```
<a href="article/ URL de l'article ciblé.html&src=nom de la rubrique
➥ primaire">libellé basé sur le TITLE de l'article</a>
```

Le nom de la rubrique primaire alourdit notre traitement mais nous permet de caser des mots-clés dans l'URL. C'est un travail fort utile.

Pour construire chaque lien de ce nuage, on a donc besoin de récupérer :

- l'URL de article cible ;
- le nom de la rubrique *PRIMAIRE* de cet article ;
- le numéro de la rubrique primaire.

Le code source ci-après réalise cet algorithme pour chaque lien. La dernière ligne de cet extrait affiche le lien construit.

```
// quelle est la rubrique PS (primaire)
// de cet article cible ?
$sql3 = "select *
         from article_rubrique
         where id_article = "
            . $line_UN_BL_a_poser["id_article_cible"]
            . ' AND primaire > 0 ' . " ; " ;
 ($request3 = mysql_query($sql3))
  {
  // quel est le id_rubrique primaire de cet article,
```

```
// càd la seule URL href valide du site
// (pour éviter les dup content).
$line3 = mysql_fetch_array($request3) ;

$sql4 = "select * from rubrique
          where ID = " . $line3["id_rubrique"]
                       . " ; " ;
  if ($request4 = mysql_query($sql4))
    {
    $line4 = mysql_fetch_array($request4) ;
    // quelle est l'URL à générer
    // depuis la table rubrique
    // il faut article/url de l'article cible
    echo ' <a href="article/'. $url_article_cible
                  . '.html&src='
                  . $line4["nom"]
                  . '">'
                  . $line_UN_BL_a_poser["libelle_BL"]
                  . '</a>' ;
```

Le reste du script ferme les accolades ouvertes et traite une éventuelle erreur.

### Fin du script PHP

On ferme les  ouverts et on affiche le pied de page.

```
    </div>
  </div>

  <?php
  $map = MiseEnPage::bas() ;
?>
```

## 3.7 Résumé de ce chapitre

Dans ce chapitre, vous avez appris à faire du *PR sculpting*.

En clair, vous avez vu comment récupérer une partie de votre Page rank et comment l'associer à des mots-clés pour mieux positionner un groupe d'une vingtaine de pages sur leurs mots-clés.

Depuis cette base de 20 pages, vous avez vu comment en favoriser une plus que toutes les autres.

L'URL rewriting est un appoint précieux pour référencer des pages sur leurs mots-clés. Cependant, un usage maladroit de l'URL rewriting peut rapidement aboutir à des maladresses conduisant à des duplicate contents et autres erreurs contreproductives pour le référencement.

# 4

# Référencement et gestion de session

Afin de personnaliser les contenus ou de suivre le comportement des internautes, de nombreux codeurs utilisent la gestion de sessions. Certes cette approche est efficace pour ces objectifs.
En revanche, ce que savent moins les codeurs, c'est que les différentes approches en gestion de sessions peuvent être incompatibles avec les attentes et besoins de Google en accessibilité technique ou sémantique ainsi qu'en garantie de contenu unique.
Dans ce chapitre nous allons voir les pièges de la gestion de session capables de fortement perturber le référencement et le positionnement réussis dans Google. Nous verrons présenterons approches pour les éviter, illustrées d'exemples et de codes sources.

## 4.1 Pourquoi la gestion de session ?

Le gestionnaire d'un site Internet peut avoir besoin, dans de nombreux cas, de tracer le comportement de l'internaute sur son site :

- Quelles sont les pages visitées ?
- Dans quel ordre ?
- Quels sont les mots-clés saisis dans un moteur pour arriver sur une des pages du site ?
- Quels sont les produits regardés ?

Dans de nombreux autres cas, le gestionnaire du site aura besoin d'authentifier un internaute puis de lui assurer un échange sécurisé.

La logique applicative du site pourra avoir besoin de connaître un paramétrage précis :

- L'internaute a demandé un filtre à appliquer sur toutes ses demandes.
- Le site personnalise ses réponses en fonction de l'internaute ou d'un facteur externe.

### ⚠ Gestion de sessions et Google se comprennent mal

Les cookies, à condition de ne pas être obligatoires sur le site, ne gênent pas les bots de Google.

Les URL surchargées peuvent perturber Google de différentes manières. Elles sont trop chargées : Google les ignore. Plusieurs URL gérant des sessions mènent à une même page : le site a généré des duplicate contents avec les problématiques de diminution des performances en référencement qui vont avec.

## 4.2 Cookies et gestion de session par URL

Ces deux techniques vont répondre aux besoins du codeur afin de contenter positivement les multiples demandes et besoins du gestionnaire du site.

### ⚠ Cookies

Le cookie est un fichier placé par le site web sur le "*disque*" local de l'internaute. L'architecture du cookie est faite de telle manière que seul le nom de domaine auteur d'un cookie peut le consulter.

Inutile donc d'essayer de lire les cookies des autres noms de domaine ; ils ne réagiront pas.

Cette technique permet de suivre dans la durée un utilisateur donné, acceptant les cookies, utilisant le même micro ordinateur et le même navigateur, sans jamais faire le ménage dans ses cookies avec un quelconque utilitaire.

SUITE

En fixant une longue durée de vie à son cookie, un site web peut ainsi construire des historiques sur ses internautes sans avoir besoin de les identifier. Le fichier du cookie peut stocker des informations, typiquement un code ou une paire d'identifiant type login/mot de passe. La logique applicative du site peut corréler ce code unique avec les traces laissées par les passages précédents de ce même cookie. Pour peu que l'internaute se soit identifié à un moment quelconque lors de ses passages pour acheter ou demander quelque chose via un formulaire, le site peut savoir qui il est et "tracer" ses activités.

Le cookie est décrit dans la RFC2109 **http://www.ietf.org/rfc/rfc2109.txt**. Son fonctionnement est implémenté dans les outils des acteurs du Net via le suivi de cette RFC2965 : **http://tools.ietf.org/html/rfc2965**.

Le gestionnaire du site peut coder la gestion de session dans l'URL avec plusieurs approches distinctes et complémentaires.

L'URL contient juste un code qui permet d'identifier l'internaute. La logique applicative récupère ce code à chaque page, enregistre les actions. On trace ainsi l'activité de l'internaute. On gère la personnalisation des pages envoyées à cet internaute ainsi que ses droits d'accès à certaines informations ou ressources.

Le comportement de l'internaute sur la page ne peut pas toujours être identifié par le back office du site. Le codeur associe alors à un lien des informations liées aux actions de l'internaute : il a cliqué sur tel bouton quand celui-ci affichait ceci, etc. Ces informations sont passées via l'URL.

### ⚠ Gestion de session par ID dans l'URL

Les sessions gérées par URL ne nécessitent pas l'accord de l'internaute, contrairement aux gestion de session gérées par cookies. Elles ne laissent aucune trace sur son micro-ordinateur, sauf à associer volontairement session et cookie.

En PHP, les variables de session sont dites super globales car elles sont toujours présentes même si la page est rechargée. Les variables de session permettent de transporter des actions de l'internaute sur le site. La logique applicative du site peut alors réagir sur ces informations : les stocker pour historique, renvoyer une information précise, calculer un affichage personnalisé, vérifier un droit d'accès, etc.

Plusieurs problématiques vont survenir avec ces techniques :

- Des utilisateurs n'acceptent pas les cookies.
- Les bots des moteurs n'acceptent jamais les cookies.
- Les bots des moteurs n'aiment pas les URL chargées en paramètres.
- Les URL chargées en paramètres sont en général contre productives en référencement/positionnement.

Néanmoins, il y a des demandes à satisfaire et il va falloir coder en ne se faisant pas piéger dans ces problématiques.

### Pourquoi les internautes n'aiment pas les cookies

C'est le parfait outil pour espionner le comportement, la vie privée, d'un internaute à travers de multiples sites. Voici le processus fort simple à mettre en œuvre :

Un internaute charge une page *Pa* d'un site *Sa*. Cette page *Pa* contient une image venant d'un site *Sz*. Ce site *Sz* place un cookie *Cz* sur le navigateur de notre internaute.

Notre internaute navigue sur le site *Sa* dont toutes les pages ont une image de *Sz*. Celui-ci enregistre toute la navigation de notre internaute grâce au cookie placé dès la 1re page.

Notre internaute quitte *Sa* et va naviguer sur le site web *Sb*. Une image de *Sz* est présente sur toutes les pages de ce site. Le site *Sz*, grâce au cookie *Cz* issu du 1er site *Sa*, vous a identifié de nouveau et suit votre navigation sur le site web *Sb*.

Voici une anecdote. Suite à une fuite d'un fichier *log*, des personnes se sont amusées aux États-Unis à identifier et à deviner la vie privée des personnes ayant laissé des traces. Leurs vies privées et leurs identités ont ainsi été publiées.

## 4.3 Les objectifs du codeur/référenceur en gestion de session

Les demandes de gestionnaire de site doivent être décomposées en groupes distincts :

- les fonctionnalités destinées à tracer les internautes pour constituer un historique ou personnaliser l'affichage ;
- ce qui reste visible du site à tout internaute.

Un exemple pour exposer cette méthodologie. Sur notre site exemple, on désire offrir à l'internaute la possibilité de filtrer les articles par région. Actuellement, tout clic dans la navigation affiche des articles visant la France entière. Comment faire ?

La table article possède un champ *région* qui accepte un entier. On pourra y saisir l'ID de la région. Par défaut, ce sera l'ID de *France entière*.

La table *region* dispose de trois colonnes : *ID*, *nom*, *acronyme*. Elle est préremplie avec les régions de France :

```
--
-- Structure de la table 'region'
--

CREATE TABLE 'region' (
```

```
  'id' int(4) NOT NULL auto_increment,
  'nom' varchar(255) default NULL,
  'acronyme' varchar(255) default NULL,
  PRIMARY KEY ('id')
) ENGINE=MyISAM DEFAULT CHARSET=latin1
  COMMENT='liste des regions françaises' AUTO_INCREMENT=89;

--
-- Contenu de la table 'region'
--

INSERT INTO 'region' ('id', 'nom', 'acronyme') VALUES
(1, 'France', 'France'),
(2, 'Alsace', 'Alsace'),
(3, 'Aquitaine', 'Aquitaine'),
(4, 'Auvergne', 'Auvergne'),
(5, 'Basse Normandie', 'B Normandie'),
(6, 'Bourgogne', 'Bourgogne'),
(7, 'Bretagne', 'Bretagne'),
(8, 'Centre', 'Centre'),
(9, 'Champagne-Ardenne', 'Champagne'),
(10, 'Corse', 'Corse'),
(11, 'Départements d''Outre-Mer', 'DOM'),
(12, 'Franche-Comté', 'Franche-Comté'),
(13, 'Haute-Normandie', 'H Normandie'),
(14, 'Ile-de-France', 'IdF'),
(15, 'Languedoc-Roussillon', 'Languedoc'),
(16, 'Limousin', 'Limousin'),
(17, 'Lorraine', 'Lorraine'),
(18, 'Midi-Pyrénées', 'Mi-Pyrénées'),
(19, 'Nord-Pas-de-Calais', 'Nord-PdC'),
(20, 'Pays de la Loire', 'Loire'),
(21, 'Picardie', 'Picardie'),
(22, 'Poitou-Charentes', 'Poitou-Ch'),
(23, 'Provence-Alpes-Côte-d''Azur', 'PACA'),
(24, 'Rhône-Alpes', 'Rhône-Alpes'),
(25, 'Territoires d''Outre-Mer', 'TOM');
```

Un internaute, pour indiquer qu'il désire voir les articles triés par région, doit cliquer sur la rubrique *Sélection par région*.

Cette rubrique affiche, selon d'autres critères, les mêmes articles que les autres rubriques du site exemple. Les mêmes pages auront des URL différentes. Mais cette fonctionnalité, afficher les articles par région d'appartenance, intéresse les internautes. Il faut donc la laisser.

Pour protéger notre référencement, nous modifierons *article.php* et la `fonction haut()` de la classe `MiseEnPage`, du fichier *php_inc/MiseEnPage.inc.php*.

```
public static function
       haut($title , $content , $repertoire , $follow_index)
// … du code identique
  <title> $title</title>" ;
  if ($follow_index == 0) // page region, ne pas indexer
    Print "<meta name='ROBOTS' content='NOINDEX,NOFOLLOW'>";
    Print "
     <meta name='description' content='$content' />
 // …La suite du code ne change pas.
```

Nous avons un nouvel argument : `$follow_index`. La valeur `0` (zéro) indique qu'il ne faut pas permettre au bot d'indexer cette page. La valeur 1 (un) indique que le bot Google peut parcourir cette page sans risque pour le référencement du site web.

La valeur `0` déclenche la pose du `meta ROBOTS en noindex, nofollow`. Ce meta interdit aux moteurs d'index de suivre les liens de cette page.

Le lien pointant vers la rubrique *Sélection par région* a depuis le début de notre site web exemple une balise `rel="nofollow"` indiquant à Google de ne pas suivre ce lien. Ce lien est dans le menu à gauche de la page. Nous avons remarqué, via la commande `site:`, que Google pouvait outrepasser l'indication `rel="nofollow"`. Par prudence, nous indiquons via `$follow_index`, un meta de blocage envers tout moteur.

Revenons à notre méthode. Nous avons séparé d'un côté ce qui est nécessaire à l'internaute et que l'on peut rendre inaccessible à Google. D'un autre côté, nous avons ce qui doit être impeccable, ce que Google peut ou même doit voir.

Pour atteindre cet objectif, plusieurs approches pratiques sont possibles. En voici deux :

- Comme dans notre exemple précédent, nous avons isolé dans une rubrique protégée par des `noindex nofollow` la fonctionnalité pouvant perturber gravement notre référencement. La rubrique *Sélection par Région* est ainsi inaccessible aux bots de Google et autres moteurs de recherche.
- Une approche mise en œuvre par Amazon ou la Fnac consiste à laisser accessible à Google un site web très hiérarchisé où aucun `session_start()` n'est lancé. Dans le même site web, les mêmes pages et programmes, via une protection type `noindex`, `nofollow` ou autre technique équivalente, utilisent les mêmes données MySQL avec des gestions de sessions permettant une gestion marketing plus efficace sur le comportement des internautes. Pour cela, on lance ou non `session_start()` selon le *user agent HTTP*.

## 4.4 Gestion de sessions en URL référencées

Nous allons modifier notre site web exemple pour lui permettre d'avoir des gestions de sessions en URL sur l'intégralité du site tout en conservant une compatibilité avec les besoins de Google pour accéder techniquement et sémantiquement aux pages du site.

### Session en URL invisible à Google

Le bot de Google ne doit jamais accéder à une page en gestion de session par URL. Mais les pages de notre site web exemple sont toutes sous gestion de session. Voici l'algorithme qui sera appliqué.

Nous allons permettre à l'internaute de naviguer dans le site en filtrant sur une région précise. L'internaute pourra ainsi avoir uniquement les informations propres à sa région, le Nord-Pas-de-Calais, par exemple.

Pour suivre les N internautes naviguant simultanément sur le site, une seule possibilité : suivre par gestion de session en URL chaque internaute pour identifier qui est le possesseur de tel ou tel filtre.

Les URL et sessions générées seront gérées par Apache, soit via Cookies, les URL restent intactes, soit via URL modifiées par l'ajout d'un jeton de session, soit les deux. Ces écarts de comportement dépendent de votre serveur Apache et de son paramétrage. Cela dépend donc de votre hébergeur pour les sites hébergés avec une formule mutualisée.

Un exemple d'URL modifiée pour une gestion de session : **/revenus-chef-entreprise/rubrique/0/1.html?PHPSESSID=057aaae217ac88b23f8412e3e11d2367**

Ce type d'URL déclenche des duplicate contents. À chaque visite, le bot de Google aura une URL différente pour une même page. Voici l'algorithme qui sera implémenté dans notre site exemple :

La page permettant à l'internaute de choisir son filtre est et reste en `noindex`, `nofollow` : la rubrique *Sélection par Région*. Il est inutile de la laisser en accès à Google.

Tout bot est détecté et on ne lance pas le `session_start()` permettant d'initialiser la gestion de session. Ainsi, les bots ne voient qu'un site strictement hiérarchisé, conforme à leurs possibilités de compréhension robotisées.

Tout internaute humain ou détecté comme tel :

- A un `session_start()` pour la page *index* : inutile mais pratique pour déboguer.
- A un `session_start()` pour tout affichage via *article.php*.

- N'a pas de gestion de session pour toute page affichée par *zoom.php* : inutile par définition dans notre exemple. Un article précis demandé est par définition vu à travers l'éventuel filtre donc inutile de contrôler quoi que ce soit.

Tout internaute ayant sélectionné un filtre se voit attribuer une variable de session codant l'ID de la région à filtrer.

## Code source session invisible à Google

Le code source de la fonction est issu d'une base dont l'origine est publiée en libre accès :

- Remi Aubert & Alan Boydell ;
- Remi : **www.remiaubert.com** ;
- Alan : **www.analytics.fr**.

Le code source d'origine servait à traquer le trafic des bots sur un site web via Google Analytics. Ce code source a été modifié et adapté par Gilles Grégoire le 23/12/2008 pour remplacer un code de Gilles moins élégant et dont il était plus difficile d'assurer la maintenance.

Cette fonction, `session_region()`, est en charge de suivre le comportement des bots sur des URL et des pages optimisées SEO. Il bascule sur une action ou une autre selon le visiteur : bot ou humain.

Plusieurs sites fournissent la liste ou des informations sur les bots en circulation :

**http://www.robotstxt.org/db/schema.txt** liste les principaux robots en circulation : plus de 2 000 actifs en 2008 selon ce site. Dans ce site, "notre" utilisation PHP du *http user agent* du robot correspond à la ligne `robot-useragent`.

La liste des bots et le script *patterns.php* dans notre exemple proviennent de : BBclone **www.bbclone.de**.

Cette fonction est dans la classe `MiseEnPage` dans *php_inc/MiseEnPage.inc.php* :

```
public static function session_region() {
require_once ('patterns.php' );
$verbotten = 0 ;
// 0 = mode silencieux 1=> affichage messages debug

$debug = 0 ;
// Mettre à 1 pour passer Chrome en outil de debug
// simule un bot donc on doit éviter session start
// $debug à 0 : internaute, lancer session start()
```

```
$log = 1 ;
// on loggue le trafic de debug en ligne
// Ouverture du fichier en écriture
$nomfic = "log.txt";
if(!($fic_log = fopen($nomfic, "a")))
  {
  print("Erreur: ");
  print("Création ouverture impossible de '$nomfic' \n");
  exit;
  }

$user_agent = getenv("HTTP_USER_AGENT" );
// si on est sur un PC de dév et sous Chrome
if ((strpos('XX'.$_SERVER['SERVER_NAME'],'127.0.0.1')>0)
  or
    (strpos('XX'.$_SERVER['SERVER_NAME'],'localhost')>0))
//alors on est considéré comme un moteur pour permettre les tests
  {
  if ($verbotten) echo '<br /> Chrome Gilles <br />' ;
    $bot_moteur = strpos($user_agent , "Chrome") ;
  if ($bot_moteur > 0)
    {
    $gilles = "Chrome" ; // chrome sur local => le codeur
    $user_agent = "Chrome" ; // être pris pour un bot
   }
  }

// par défaut on considère que visiteur est un internaute
$internaute = 1;

# S'il existe un referer alors internaute humain
# car les robots n'ont pas de referer.
if( empty( $_SERVER["HTTP_REFERER"] ) )
  {
  $internaute = 0;
  }
if ($verbotten)
  echo '<br /> HTTP_REFERER = '
        . $_SERVER["HTTP_REFERER"]
        . ' Vide = Moteur possible sinon c un humain <br />';

// Si le visiteur renvoie un OS connu
// il ne s'agit pas d'un robot
foreach($os as $pattern => $o)
  {
  if ( preg_match( '#'.$pattern.'#msi'
      , $_SERVER["HTTP_USER_AGENT"] ) == 1 )
```

```
    {
    $internaute = 1;
    $os_name = preg_replace ( "/\\s{1,}/i" , '-' , $o );
    if ($verbotten)
      echo '<br /> OS = ' . $os_name
                          . ' internaute == '
                          . $internaute . ' ' ;
    if ($log)
      {
      $uri = $_SERVER["REQUEST_URI"];
      $var_now = time();   //date du jour
      fputs($fic_log,
            '$os_name = ' . "\t" . $os_name
                . "\r\n"
               . ' sur page : ' . "\t" . $uri
                . "\r\n"
                . ' à : ' . "\t\t"
                . date(DATE_RFC822 , $var_now)
                . "\r\n"
                . '=========='
                ."\r\n" . "\r\n");
     }
    break;
    }
  }

// en debug, chrome doit être vu comme un bot.
if ($debug)
  {
  array_push($bots, $gilles);
  if ($verbotten) echo
                    '<br /> debug :
                    Chrome mis dans pattern $bots' ;
  $internaute = 0 ;
  }

if ( $internaute == 0 )
  {
  #On verifie de quel bot il s'agit
  foreach( $bots as $pattern => $bot )
    {
    if ( preg_match( '#'.$pattern.'#i' ,
         $_SERVER['HTTP_USER_AGENT'] ) == 1 )
      {
      //on recupère le nom du bot
      $botname = preg_replace ( "/\\s{1,}/i" , '-' , $bot );
      $uri = $_SERVER["REQUEST_URI"];
```

```
      //Resquested URI by Crawler - Page vue par le visiteur
      $var_now = time(); //date du jour
      if ($verbotten)
        echo '<br />
              $botname =' . $botname
                          . ' sur page : '
                           . $uri . ' à : '
                           . date(DATE_RFC822 , $var_now);
      if ($log)
        fputs($fic_log, '$botname = ' . "\t" . $botname
                                      . "\r\n"
                                      . ' sur page : ' . "\t"
                                      . $uri
                                      . "\r\n"
                                      . ' à : '
                                      . "\t\t"
                                      . date(DATE_RFC822 , $var_now)
                                      . "\r\n"
                                      . '=======
                                      ."\r\n" . "\r\n");
      break;
      }
    }
  }

if ($internaute == 1) // 1 = humain
  {
  session_start();
  if ($verbotten)
  echo '******* debug gg ***   START SESSION pour humain '
        . $user_agent . ' <br>' ;
  }
else
  if ($verbotten)
    echo '******* debug gg ***   PAS SESSION car robot '
        . $user_agent . ' <br>' ;

if ($log) fclose($fic_log);
} // accolade de fin de fonction
```

## Étudions le code source de cette fonction

```
public static function session_region() {
require_once ('patterns.php' );
$verbotten = 0 ;
// 0 = mode silencieux 1=> affichage messages debug
```

```
$debug = 0 ;
// Mettre à 1 pour passer Chrome en outil de debug
// simule un bot donc on doit éviter session start
// $debug à 0 : internaute, lancer session start()

$log = 1 ;
// on loggue le trafic de debug en ligne
// Ouverture du fichier en écriture
$nomfic = "log.txt";
if(!($fic_log = fopen($nomfic, "a")))
  {
  print("Erreur: ");
  print("Création ouverture impossible de '$nomfic' \n");
  exit;
  }
```

Ce sont les déclarations classiques. `$debug` permet de forcer un navigateur *Chrome* comme bot. C'est pratique pour la phase de mise au point ou d'évolution. Il suffit de surfer avec *Chrome* et de regarder le log. `$verbotten` permet de passer en mode Bavard. `$log` valide le suivi en fichier log de certaines actions et informations.

Continuons :

```
$user_agent = getenv("HTTP_USER_AGENT" );
// si on est sur un PC de dév et sous Chrome
if ((strpos('XX'.$_SERVER['SERVER_NAME'],'127.0.0.1')>0)
  or
    (strpos('XX'.$_SERVER['SERVER_NAME'],'localhost')>0))
//alors on est considéré comme un moteur pour permettre les tests
  {
  if ($verbotten) echo '<br /> Chrome Gilles <br />' ;
    $bot_moteur = strpos($user_agent , "Chrome") ;
  if ($bot_moteur > 0)
    {
    $gilles = "Chrome" ; // chrome sur local => le codeur
    $user_agent = "Chrome" ; // être pris pour un bot
    }
  }
```

On récupère l'agent *http* du visiteur. Sur le PC de développement, on peut passer le navigateur *Chrome* comme un bot. Il faudra tester la version "humaine" avec un autre navigateur.

Ensuite :

```
// par défaut on considère que visiteur est un internaute
$internaute = 1;

# S'il existe un referer alors internaute humain
# car les robots n'ont pas de referer.
if( empty( $_SERVER["HTTP_REFERER"] ) )
  {
  $internaute = 0;
  }
if ($verbotten)
  echo '<br /> HTTP_REFERER = '
        . $_SERVER["HTTP_REFERER"]
        . ' Vide = Moteur possible sinon c un humain <br />';

// Si le visiteur renvoie un OS connu
// il ne s'agit pas d'un robot
foreach($os as $pattern => $o)
  {
  if ( preg_match( '#'.$pattern.'#msi'
     , $_SERVER["HTTP_USER_AGENT"] ) == 1 )
    {
    $internaute = 1;
    $os_name = preg_replace ( "/\\s{1,}/i" , '-' , $o );
    if ($verbotten)
      echo '<br /> OS = ' . $os_name
                          . ' internaute == '
                         . $internaute . ' ' ;
    if ($log)
      {
      $uri = $_SERVER["REQUEST_URI"];
      $var_now = time();   //date du jour
      fputs($fic_log,
            '$os_name = ' . "\t" . $os_name
                . "\r\n"
                . ' sur page : ' . "\t" . $uri
                . "\r\n"
                . ' à : ' . "\t\t"
                . date(DATE_RFC822 , $var_now)
                . "\r\n"
                . '=========='
                ."\r\n" . "\r\n");
     }
    break;
    }
  }
```

On considère que tout visiteur est un humain. S'il y a un `http_REFERER`, on confirme que c'est un humain. Il existe des navigateurs humains sans *REFERER*. Testons le système d'exploitation. Si l'Os est connu, c'est un humain. Autrement, on considère que c'est un bot.

En mode Bavard ou Log, on trace les caractéristiques du visiteur :

```
// en debug, chrome doit être vu comme un bot.
if ($debug)
  {
  array_push($bots, $gilles);
  if ($verbotten) echo
                   '<br /> debug :
                   Chrome mis dans pattern $bots' ;
  $internaute = 0 ;
  }
```

*Chrome* est passé comme bot en mode Debug.

Si le visiteur est un bot, qui est-il ? Traçons son passage :

```
if ( $internaute == 0 )
  {
  #On verifie de quel bot il s'agit
  foreach( $bots as $pattern => $bot )
    {
    if ( preg_match( '#'.$pattern.'#i' ,
         $_SERVER['HTTP_USER_AGENT'] ) == 1 )
      {
      //on recupère le nom du bot
      $botname = preg_replace ( "/\\s{1,}/i" , '-' , $bot );
      $uri = $_SERVER["REQUEST_URI"];
      //Resquested URI by Crawler - Page vue par le visiteur
      $var_now = time(); //date du jour
      if ($verbotten)
        echo '<br />
              $botname =' . $botname
                          . ' sur page : '
                          . $uri . ' à : '
                         . date(DATE_RFC822 , $var_now);
      if ($log)
        fputs($fic_log, '$botname = ' . "\t" . $botname
                                      . "\r\n"
                                      . ' sur page : ' . "\t"
                                     . $uri
                                      . "\r\n"
                                      . ' à : '
```

```
                                        . "\t\t"
                                        . date(DATE_RFC822 , $var_now)
                                       . "\r\n"
                                        . '=======’
                                        ."\r\n" . "\r\n");
      break;
      }
    }
  }
```

À ce stade, nous savons si nous avons un visiteur humain ou robot.

Action à entreprendre pour un humain ou un bot :

```
if ($internaute == 1) // 1 = humain
  {
  session_start();
  if ($verbotten)
  echo '******* debug gg ***   START SESSION pour humain '
        . $user_agent . ' <br>' ;
  }
else
  if ($verbotten)
    echo '******* debug gg ***   PAS SESSION car robot '
        . $user_agent . ' <br>' ;

if ($log) fclose($fic_log);
} // accolade de fin de fonction
```

On lance une session pour un visiteur humain. Autrement, on ne fait rien.

## La liste des bots et des OS

Il faudra mettre à jour la liste des bots et des OS régulièrement dans le script PHP suivant. Dans le code source nous avons :

```
require_once ('patterns.php' );
```

Ce script PHP contient deux déclarations de tableau : `$bots` en premier, suivi à la fin du fichier par `$os` :

```
$bots = array(
  'Mediapartners-Google[ /]([0-9.]{1,10})'
              => 'Google Mediapartners',
  'Mediapartners-Google' => 'Google Mediapartners',
  'Googl(e|ebot)(-Image)/([0-9.]{1,10})'
```

```
                => 'Google Image',
  'Googl(e|ebot)(-Image)/' => 'Google Image',
  '^gsa-crawler' => 'Google',
  'Googl(e|ebot)(-Sitemaps)/([0-9.]{1,10})?'
                => 'Google-Sitemaps',
```

Quelques centaines de lignes définissent moins de 1 000 bots.

Le même principe est appliqué aux systèmes d'exploitation :

```
$os = array ('wi(n|ndows)?' => 'windows',
  'linux[ /\-]([a-z0-9._]{1,10})' => 'linux',
  'linux' => 'linux',
  'Mac[ _]?OS[ _]?X[ /]([0-9.]{1,10})' => 'macosx',
```

### Appel à cette fonction session_region

Les scripts PHP *article.php*, *index.php*, *region.php* appellent cette fonction dès le début :

```
<?php
require_once("php_inc/MiseEnPage.inc.php");
require_once("php_inc/init.inc.php");
require_once("php_inc/biblio_inc.php");

MiseEnPage::session_region() ; //
```

## Code source de region.php

Le script *region.php* permet de sélectionner un filtre régional pour afficher les résultats des différentes rubriques du site ou directement dans la page *region.php*, la liste des articles par régions, toutes catégories confondues, d'un seul clic :

```
<?php
require_once("php_inc/MiseEnPage.inc.php");
MiseEnPage::session_region() ;
require_once("php_inc/init.inc.php");
require_once("php_inc/biblio_inc.php");

$title_page = "filtrer par region - selection " ;
$meta_description_content = "filtrer par region";
$follow_index = 0 ; // jamais de bot Google ici
MiseEnPage::haut( $title_page ,
                  $meta_description_content ,
                  $rep, $follow_index ) ;
?>
```

```
<div id="sousban"> </div>
<div id="pagegauchetete">
  <?php
  MiseEnPage::MenuGauche() ;
 ?>
</div>   <!-- fin page gauche tete -->
<div id="pagecentretete">
  <div id="sousban800"> </div>

  <!-- COLONNE DROITE -->
  <div id="pagedroite">
    <?php
    $rubrique = 999 ;
    MiseEnPage::MenuDroite($rubrique) ;
 ?>
  </div>
  <!-- fin page droite -->

<!-- *** CONTENU DES PAGES zone centre *** -->
  <div id="pagetext">
  <h1>Filter les résultats par région françaises </h1>
  <p>
  Par défaut, les résultats des rubriques interrogées sont des
informations propres à toute la France.
  <br /> <br />
Pour obtenir des résultats d'une région française précise, vous pouvez
fixer un filtre ci dessous. Un click vous affichera tous les
articles d'une région donnée. Tous les click suivant seront filtrés.
Pour récupérer la visibilité sur les articles nationaux, il suffit de
cliquer sur<strong> France entière</strong>.
  </p>

<?php
// afficher les regions avec un lien par region
$sql_query="SELECT * FROM 'region' order by id ASC";
if ($request = mysql_query ($sql_query))
  {
// $line : tableau des n rows résultat de la requête
  while ($line = mysql_fetch_array($request))
    {
    echo 'filtrage sur : <a href=\'articles/region/'
            . $line["id"]
            . '/1.html\' rel="nofollow" >'
            . $line["id"] . ' ' . $line["nom"]
            . '</a> <br />' ;
    }
  }
```

```
  else
    {
    echo 'debug : la requête : '
          . $sql_query . ' message SQL : '. mysql_error()
          . ' n\'a pu aboutir - Erreur <br />' ;
    }
?>

      <!--fin page texte -->
    </div>
    <!-- fin pagecentre -->
  </div>

  <?php
  $map = MiseEnPage::bas() ;
 ?>
```

Détaillons les quelques particularités de ce script PHP :

```
<?php
require_once("php_inc/MiseEnPage.inc.php");
MiseEnPage::session_region() ;
require_once("php_inc/init.inc.php");
require_once("php_inc/biblio_inc.php");

$title_page = "filtrer par region - selection " ;
$meta_description_content = "filtrer par region";
$follow_index = 0 ; // jamais de bot Google ici
MiseEnPage::haut( $title_page ,
                  $meta_description_content ,
                  $rep, $follow_index ) ;
?>
```

On commence par les `require_once` pour charger les différentes routines dont nous aurons besoin et par la mise en service ou non du `session_start()` via la fonction `session_region()`.

La page est affichée en `nofollow`, `noindex`. Elle doit rester invisible aux bots de Google.

Continuons :

```
<div id="sousban"> </div>
<div id="pagegauchetete">
  <?php
  MiseEnPage::MenuGauche() ;
 ?>
</div>   <!-- fin page gauche tete -->
```

```
<div id="pagecentretete">
  <div id="sousban800"> </div>

  <!-- COLONNE DROITE -->
  <div id="pagedroite">
    <?php
    $rubrique = 999 ;
    MiseEnPage::MenuDroite($rubrique) ;
 ?>
  </div>
  <!-- fin page droite -->

<!-- *** CONTENU DES PAGES zone centre *** -->
  <div id="pagetext">
  <h1>Filter les résultats par région françaises </h1>
  <p>
  Par défaut, les résultats des rubriques interrogées sont des
informations propres à toute la France.
  <br /> <br />
Pour obtenir des résultats d'une région française précise, vous pouvez
fixer un filtre ci dessous. Un click vous affichera tous les
articles d'une région donnée. Tous les click suivant seront filtrés.
Pour récupérer la visibilité sur les articles nationaux, il suffit de
cliquer sur<strong> France entière</strong>.
  </p>
```

La mise en page de différents éléments et données secondaires sont affichées, rien d'extraordinaire.

Avançons :

```
<?php
// afficher les regions avec un lien par region
$sql_query="SELECT * FROM 'region' order by id ASC";
if ($request = mysql_query ($sql_query))
  {
// $line : tableau des n rows résultat de la requête
  while ($line = mysql_fetch_array($request))
    {
    echo 'filtrage sur : <a href=\'articles/region/'
            . $line["id"]
            . '/1.html\' rel="nofollow" >'
            . $line["id"] . ' ' . $line["nom"]
            . '</a> <br />' ;
    }
  }
  else
```

```
    {
    echo 'debug : la requête : '
          . $sql_query . ' message SQL : '. mysql_error()
          . ' n\'a pu aboutir - Erreur <br />' ;
    }
?>
```

On affiche en colonnes les différentes régions administrées dans la table *region*.

Chaque région est "cliquable". Le lien est constitué de :

- Article/region : via URL rewriting, on va adresser article.php en passant une région ID en argument.
- L'ID de la région entre / (slash).
- *1.html* indique à *article* d'afficher la page 1.
- Le nom de la région.
- Ce lien est balisé en `nofollow` au cas où malgré tout, un bot passerait sur cette page.

## Risques de cette approche

Admettons qu' un bot anti-fraude de Google se fasse passer pour un humain et que ce bot passe donc en mode Session.

Qu'arriverait-il ?

- Une URL modifiée serait utilisée au lieu d'une URL propre.
- Deux URL distinctes mènent à la même page mais l'une n'est pas accessible à Google.
- La page où la pose de filtre intervient est totalement interdite à Google. Les 2 pages seront donc identiques au nuage de liens près qui sera ordonné différemment.

Google n'a rien à reprocher à un tel site. Il n'y a pas de risque de pénalité concernant notre approche.

Existe-t-il des sites pratiquant ce type de technique ? Oui, **amazon.com** par exemple.

Le bot est inconnu et il reçoit une URL modifiée. 95 % des bots ne travaillent pas pour des moteurs. Cette erreur possible ne présente pas de risque particulier sauf si Google (ou Microsoft, ou Yahoo…) change le nom de son bot et que l'outil ne soit pas mis à jour.

L'internaute est pris pour un bot. Il n'aura pas accès au fonctionnement avec le filtre. Cette erreur possible ne présente pas de risque particulier. Elle importune un peu l'internaute et notre suivi webmarketing sera faussé concernant cet internaute.

## 4.5 Protéger son site des hackers

Passer d'un site statique à un site codé en PHP avec des tables MySQL donne la possibilité de fortement améliorer son référencement mais cela crée un risque : se faire pirater le site.

Les répertoires sensibles devraient être protégés par un mot de passe réellement difficile à craquer.

Peu de sites le font. C'est pourtant fort simple. Voici un premier mode opératoire clairement expliqué et simple à mettre en œuvre.

### .htaccess et contrôle d'accès

⚠ Protéger les répertoires des scripts PHP uniquement

Le propos de ce chapitre est d'expliquer simplement comment protéger les répertoires contenant les scripts PHP sensibles. En aucun cas ce chapitre n'est envisagé comme un tutorial sur toutes les possibilités de sécurisation de répertoires. Il existe des ouvrages de sécurité informatique ou des sites web spécialisés sur le sujet.

Seul Apache ou l'administrateur du site web ont besoin de pouvoir accéder aux scripts PHP sensibles, ceux qui donnent accès à l'administration des données typiquement. Il faut donc les regrouper dans un répertoire qu'il s'agira de défendre contre les intrus de toute nature :

- bot de moteur ;
- scan de répertoire : aspirateur de site, logiciel traquant les faiblesses d'un site, etc. ;
- accès direct.

La partie "publique" des scripts se contentant d'afficher sera en répertoire d'accès libre dans cette version.

Dans le répertoire *php_inc*, nous allons :

- créer un répertoire *mdp* (comme *Mot De Passe*) ;
- Créer un fichier *.htaccess*.

Le fichier *.htaccess* du répertoire *php_inc* contiendra quelque chose de ce type :

```
#sur WAMP, on garde comme dans unix les / (pas de backslash \ )
```

```
# fonctionne sur OVH -- protection simple pour répertoire sans
➥ .htpasswd dedans

AuthUserFile / … voir chemin obtenu via script php …
➥ /php_inc/mdp/.htpasswd

AuthGroupFile /dev/null

AuthName "Accès Restreint "

AuthType Basic

<Limit GET POST>

require valid-user

</Limit>
```

*php_inc* contient :

- les include PHP nécessaires pour se connecter à MySQL ;
- les include PHP nécessaires pour accéder aux fonctions permettant de réaliser des entrées/sorties sur la base.

Ce que précise ce fichier *.htaccess* :

- pas de `AuthGroupFile` géré ;
- un chemin d'accès absolu vers un fichier *.htpasswd*.

Un des problèmes des débutants : comprendre quel chemin absolu indiquer. Une ligne en PHP permet de récupérer une telle information.

```
  echo realpath('admin_backoffice.php');
```

Le script *admin_backoffice.php* est situé dans le répertoire à protéger : *php_inc/*. Il contient cette ligne. En exécutant le script, *realpath* donnera le chemin absolu utilisé par l'hébergeur sur votre serveur pour accéder au répertoire à protéger. Il n'y aura plus alors qu'à copier/coller cette information dans les fichiers *.htaccess* nécessaires.

Il faut un utilisateur valide pour accéder aux contenus de ce répertoire :

- Tout utilisateur possédant l'identifiant plus le mot de passe gérés dans le fichier *.htpasswd*.
- Apache : l'appel à ces contenus via un `require_once` ne pose aucun souci. Le login/mot de passe ne sera pas demandé.

Le fichier *.htpasswd* est dans le répertoire *php_inc/mdp/* :

```
gregoire:g8/49Yjq2Xzs0
```

Le plus simple pour générer un mot de passe est de passer par un site comme celui-ci : **http://shop.alterlinks.com/htpasswd/passwd.php**. Vous saisissez votre identifiant, le mot de passe souhaité et vous n'avez plus qu'à recopier cette ligne dans notre fichier *.htpasswd*.

Cette ligne dans le fichier *.htpasswd* signifie que l'identifiant `gregoire` a pour mot de passe un élément crypté.

Mais le cryptage n'est pas suffisant. Il est très facile à casser depuis le code crypté publié : `g8/49Yjq2Xzs0`. Des utilitaires permettent de renverser le code et d'identifier le vrai mot de passe correspondant.

Il faut donc bloquer tout accès à ce fichier *.htpasswd*. Ainsi personne ne pourra tenter de renverser le code contenu dans le fichier et obtenir ainsi le couple (identifiant, mot de passe) lui permettant de pénétrer dans le répertoire sensible. On va, dans notre exemple, interdire l'accès à tout le répertoire.

Dans le répertoire *php_inc/mdp/*, on va créer un fichier *.htaccess* :

```
AuthUserFile /homez.95/apicslil/blog/php_inc/mdp/.htpasswd

AuthGroupFile /dev/null

AuthName "Accès interdit"

AuthType Basic

deny from all
```

`deny from all` interdit le répertoire et ses contenus à quiconque. Seul Apache y a accès ainsi, bien sûr, que FTP afin de pouvoir mettre à jour ce fichier. Si on veut pirater votre site, il faudra d'abord pirater le serveur Apache ou le serveur FTP, ce qui est déjà nettement plus difficile. Votre site sera ainsi beaucoup mieux protégé.

Le présent paragraphe ne dispense nullement un codeur/webmestre de compléter ses connaissances en sécurité informatique pour protéger son site web.

# 5

# Référencement et Web 2.0

Le Web 2.0 est un axe important pour Google. En effet, plus il y a d'internautes interagissant avec les pages et contenus d'un site, plus celui ci présente un intérêt à être indexé et bien positionné.
De même les recommandations d'URL dans les réseaux sociaux, les votes des internautes constituent des gisements de backlinks et d'informations contextuelles importantes pour Google dans ses calculs de positionnement.
Nous allons voir des codes sources pour notre site exemple ciblant deux usages Web 2.0. Ces sources illustreront comment utiliser le Web 2.0 pour améliorer le référencement d'un site.

# 5.1 Web 2.0 et Google

Le Web 2.0 concerne tous les acteurs du net, Google en premier. Deux comportements Web 2.0 intéresseront Google :

- identifier les pages dont on parle sur le Net ;
- identifier les pages à succès

## Réseaux sociaux et URL de pages

En effet, le fait que des internautes discutent d'informations (donc de mots-clés) sur des sites, entre autres sujets, intéresse Google. Pas pour connaître vos goûts personnels, mais pour déterminer des pages de sites qui présenteraient un intérêt potentiel pour les internautes.

Proposer de meilleurs résultats de recherche que ses concurrents est fondamental. Ce sont les internautes qui cliquent sur les liens *AdWords*. Plus il y a d'utilisateurs, plus Google augmente ses revenus publicitaires.

Accéder aux informations publiques des réseaux sociaux, typiquement ceux qui permettent de voter pour tel ou tel site, fait partir de la mission des bots. Le codeur facilitera le travail des internautes en mettant à disposition, sur les pages du site, une boîte permettant de cliquer vers des réseaux sociaux afin de voter ou de mettre en favori l'URL de la page en cours.

Ce serait mieux de proposer des URL propres de tout jeton de session.

## Contribution des internautes Web 2.0

Si vous permettez à des internautes de discuter directement sur votre page, comme dans un blog, quelles en sont les conséquences, les risques éventuels ?

### N'importe qui poste n'importe quoi

En France, vous pouvez vous retrouver avec un procès parce qu'un dealer connecté depuis l'étranger poste des annonces vers des sites illégaux : vente de tabac de contrebande, etc.

Il est donc essentiel de modérer ce type de mécanisme, a priori (on valide avant de publier), ou a posteriori, mais attention aux "requins" des soi disant associations anti-tabac et autres qui traquent la moindre parution illégale, même courte, et attaquent en demandant de fortes sommes d'argent.

Vous pouvez simplement placer un automate détectant les mots-clés sensibles et bloquer à la moindre saisie suspecte :

- Le post-rejet avec message d'erreur.
- L'adresse IP d'origine en mode Ajout de commentaire sur une page
- La classe IP de la personne malintentionnée en mode Ajout de commentaire sur une page. Tous les internautes de ce fournisseur d'accès seront eux aussi bloqués. Vous pouvez les basculer en modération avant publication pour diminuer la gêne occasionnée.

### Ajout de texte et de mots-clés

Dans la plupart des cas, les internautes poseront des questions, feront des commentaires en lien avec le contenu de la page. Les conséquences suivantes participeront au référencement de la page :

- Ajout de texte et de mots plus ou moins clés, plus ou moins cohérents. Le poids de la page augmente en nombre de mots. Il est possible de renforcer la densité, par rapport aux textes d'origine, des mots-clés sur la page par des nouveaux textes. Par exemple, l'ajout automatique d'un titre rappelant le thème de la page tous les 5 posts. Des algorithmes plus sophistiqués pourront être déployés. Dans le cadre de ce livre, nous ferons simple.
- Ajout au fil du temps. Les script PHP ne permettent pas d'identifier la date de mise à jour d'une page car celle-ci est produite à la demande. En revanche, un moteur comme Google peut identifier l'ajout de texte à une page en la comparant à celle présente dans son cache. Il peut tracer la fréquence de mise à jour d'une page et identifier les pages recevant fréquemment du texte cohérent.

## 5.2 Réseau social et référencement

Commençons par le plus simple. Quels sont les site sociaux les plus usités par Google ? Ce seront ceux à privilégier pour faciliter le clic de vote de tout internaute.

Google n'a pas pour habitude de violer les règles en place. Si un site type réseau social déclare que les favoris de ses membres sont confidentiels, même si la sécurité informatique du réseau social est un gruyère, Google n'ira pas collecter ces informations.

Notre attention devra donc se porter sur les sites permettant le libre accès à tous. Outre le jugement de chacun, pour ce livre nous avons retenu de suivre les indications d'un site : **http://www.socialmeter.com**.

Ce site est censé donner un score de présence d'URL dans les réseaux sociaux accessibles à tout bot ou à tout internaute non membre. Peu importe que les informations données par cet outil soit justes ou non. Il nous sert juste de fil conducteur. Testons-le sur **www.lemonde.fr**.


► Fig. 5.1 : Site web lemonde.fr : ses backlinks visibles dans les réseaux sociaux selon socialmeter.com

Il est donc possible de cartographier les URL déposées dans divers réseaux sociaux.

Facilitons le travail des internautes en leur mettant à disposition une barre leur permettant de se connecter d'un simple clic sur leurs réseaux sociaux favoris. Ce serait intéressant de disposer de statistiques sur l'usage de cette barre.

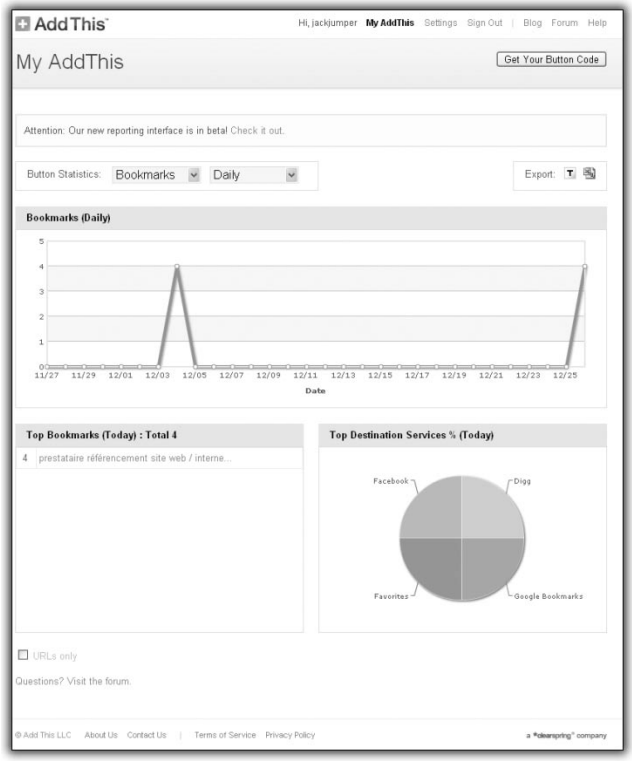
Le site web **http://www.addthis.com** fournit un tel outil (il y aussi **http://justaddme.com/** ainsi que **http://www.letsgetsocialnow.com/** :

Voici un JavaScript prêt à l'emploi pour permettre de *bookmarker* d'un seul clic l'URL en cours :

```
  <!-- AddThis Button BEGIN -->
<script type="text/javascript">
   var addthis_pub = "VotreIdDeCompte chez addthis.com";
```

```
</script>
<a href=
   "http://www.addthis.com/bookmark.php"
   onmouseover="return addthis_open(this,
                    '', '[URL]',
                    '[TITLE]')"
   onmouseout="addthis_close()"
   onclick="return addthis_sendto()">
 <img src="http://s7.addthis.com/static/btn/lg-share-en.gif"
     width="125" height="16" border="0"
     alt="" />
</a>
   <script type="text/javascript"
src="http://s7.addthis.com/js/152/addthis_widget.js">
</script>
<!-- AddThis Button END -->
```

Un compte permettant de suivre les clics des internautes.




► Fig. 5.2 : Suivi des clics de mise en réseau social sur Addthis.com

Addthis permet de personnaliser, dans certaines limites, la barre à cliquer. Ci-après, voici un exemple :

```
var addthis_pub = "VotreIdDeCompte chez addthis.com";
addthis_brand = 'Créer sa Boite';
addthis_options = 'favorites,
                   email,
                   yahoobookmarks,
                   digg, delicious,
                   myspace, facebook,
                  google, live, twitter,
                   yahoobkm, linkedin, blogmarks';
```

Le site web exemple ne sera pas équipé de cette barre de boutons **AddThis**. En effet, il faut ouvrir un compte pour avoir un JavaScript capable de suivre les clics. Il est très facile d'installer ce JavaScript.

L'autre technique consiste à mettre en œuvre sa propre barre de boutons ; un principe simple que chaque codeur pourra ensuite développer à loisir :

- un pictogramme représentant le réseau social (à récupérer sur le site web du réseau social) ;
- un `onclick` lance l'URL du réseau social où il faut se connecter en transmettant l'URL où l'internaute est présent.

Ce code est inséré dans la fonction `MenuGauche()` de *php_inc/MiseEnPage.inc.php*. C'est pour cela que les guillemets sont précédés de backslash :

```
<p>
Partagez
<a rel=\"nofollow\" style=\"text-decoration:none;\"
href=\"http://www.facebook.com/\"
   onclick=
      \"window.open
         ('http://www.facebook.com/sharer.php?u=
            '+encodeURIComponent(location.href)
          +'&t='+encodeURIComponent(document.title));
   return false;\"
   title=\"Bookmark to: Facebook\">

   <img style=\"padding-bottom:0px;padding-top:0px;\"
        src=\"artpackage/facebook.gif\"
        alt=\"Bookmark : Facebook\"
        name=\"Facebook\" border=\"0\">
</a>
</p>
```

## 5.3 Interactivité Web 2.0 et référencement automatisé

Nous avons vu les différents avantages à engager le dialogue avec les internautes sur chaque page du site. Coder une telle approche n'a rien de complexe malgré quelques petits problèmes de codage à résoudre. Dans notre site web exemple, nous allons nous concentrer sur un article qui vous servira de test : **http://localhost/BookN2/V1-00/article/deduire-repas-restaurant-chef-entreprise.html&src=Fiscalite-entreprise**

À chacun d'adapter cette URL à son environnement de développement personnel.


► Fig. 5.3 : Article servant d'exemple. Cliquer pour poser une question

L'objectif sera de permettre à un internaute de poser une question ou de poster un commentaire sur le contenu d'un article affiché par *zoom.php*.

Un lien permet de lancer un formulaire de saisie. Volontairement, pour séparer les codes sources, nous avons sorti ce formulaire vers une page autonome. Le formulaire aurait pu être intégré à la page elle-même. Cela aurait été plus pratique.


► Fig. 5.4 : Formulaire de saisie du post

Ce formulaire, très simple, permet classiquement de saisir un commentaire. Si on remplit les champs obligatoires, le post sera pris en compte immédiatement par le site.


► Fig. 5.5 : Page Merci – retour possible vers la page commentée

Une classique page de remerciement, isolée dans un script PHP, afin de pouvoir utiliser son URL unique pour tracer des statistiques via Google Analytics, par exemple. Un lien permet de revenir vers la page commentée à l'origine.

## Détails des algorithmes

Seuls les articles affichés via *zoom.php* présentent un intérêt pour dialoguer avec les internautes. Le script *zoom.php* reçoit une modification en charge de :

- renforcer la cohérence de l'article affiché ;
- récupérer les variables d'environnement indispensables au bon fonctionnement des différents scripts ;
- lancer les scripts permettant d'afficher les différents posts déjà existants ;
- proposer un lien pour saisir un post.

La fin de notre fichier *zoom.php* devient :

```
        echo '</div>' ; // div de FIN des BL dynamiques
      }
    $expression_clef_strategique = $line["expression_clef"] ;
    $url_de_zoom = $_SERVER['REQUEST_URI'];
    MiseEnPage::message_cet_article
                ($id_Article,
                 $expression_clef_strategique ,
                 $url_de_zoom) ;
 ?>

    </div>
  </div>

<?php
  MiseEnPage::bas() ;
?>
```

- `$expression_clef_strategique = $line["expression_clef"]` : nous allons avoir besoin de l'expression clé stratégique de l'article affiché. Nous le récupérons directement dans l'enregistrement affiché.
- `$url_de_zoom = $_SERVER['REQUEST_URI']` : nous aurons besoin de revenir à la page commentée. Il nous faut mémoriser où nous sommes. On aurait aussi pu recalculer l'URL via l'ID de l'article. Mémoriser l'URI de la page présente quelques avantages en termes de souplesse et d'évolutivité.
- `MiseEnPage::message_cet_article($id_Article, $expression_clef_strategique, $url_de_zoom)` : on regroupe dans la classe `MiseEnPage` la gestion des messages existants pour cet article en particulier.

La bibliothèque *php_inc/MiseEnPage.inc.php* reçoit la fonction suivante :

```
public static function
   message_cet_article($id_article ,
                       $expression_clef_strategique,
                        $url_de_zoom ){
```

```
require_once("RecordPost.inc.php");

$url_de_zoom = str_replace("&","|",$url_de_zoom );

print '
<div class="jcmbparalarge">
<h1>Thème : ' . $expression_clef_strategique . ' </h1>
<span>
  <a href="post_affiche_formulaire_saisie.php?id_article='
           . $id_article .'&url=' . $url_de_zoom
           . '" rel="nofollow">
   Cliquer pour poser une question ou donner un avis
   sur le contenu de cette page
  </a>
</span>
<br /><br/>
' ;

echo '<span>
       Message(s) ou question(s) déjà publiés
      </span>';

$post = PostHelper::findTousById_Article($id_article);
PostHelper::AfficheTousPost($post) ;
echo '</div>' ; // div jcmbparalarge

} // fin de fonction message_cet_article
```

Cette fonction a pour mission :

- d'afficher tous les post déjà existants pour l'article en cours ;
- de proposer de commenter l'article en cours.
- `require_once("RecordPost.inc.php")`. Cette bibliothèque regroupe toutes les fonctions permettant de manipuler des objets type Post. Nous la détaillons un peu plus loin.
- `$url_de_zoom = str_replace("&","|",$url_de_zoom )`. On va transmettre l'URI de la page zoom affichant l'article. Or cette URI possède un caractère `&` incompatible avec une transmission par URL. On remplace `&` par le caractère US peu usité `|` (pipe).
- `<div class="jcmbparalarge">`. On manipule uniquement des affichages classiques. La feuille de style est là pour permettre toute adaptation de chacun. Elle est détaillée un peu plus loin.

- `<h1>Thème : ' . $expression_clef_strategique . ' </h1>`. On renforce la cohérence de la page en affichant entre balises `h1` l'expression clé stratégique de l'article.
- `<a href="post_affiche_formulaire_saisie.php?id_article='. $id_article .'&url=' . $url_de_zoom . '" rel="nofollow">`. Ce lien emmène l'internaute vers le formulaire de saisie de son post. Pour permettre le retour à la page précise d'origine, on transporte dans l'URL différentes informations (ID de l'article, URI d'origine).
- `$post = PostHelper::findTousById_Article($id_article)`. Cette fonction renvoie un tableau d'objets `Post`. Ce sont tous les `Post` existant déjà pour cet `$id_article`.
- `PostHelper::AfficheTousPost($post)`. Cette fonction affiche tous les objets `Post` de ce tableau d'objets `Post`.

Examinons la bibliothèque de fonction autour de l'objet `Post` : *php_inc/RecordPost.inc.php*. Nous nous limiterons aux fonctions utilisées dans notre exemple :

```
<?php
// dédiée aux manipulations de l'objet Post

class PostHelper {
```

Deux classes sont créées dans ce fichier :

- `Class PostHelper` regroupe toutes les fonctions manipulant les objets `Post`.
- `Class Post` regroupe toutes les fonctions créant un objet `Post` et les classiques fonctions ajout/mise à jour/supprimer vis-à-vis de la table *post_article* :

```
  /*
   * findById rechercher un enregistrement
   * par son id dans la table
   */

public static function findById($id) {
  $query = "select * from post_article where id = ".$id;
  $result = mysql_query($query); // on stocke le resultat ds $result
  if ($record = mysql_fetch_object($result))
    {
    // si resultat, $record devient positif et contient tout l'objet
   ➥ enregistrement
    $post = new Post($record);
    }
    else { $post = null; }
```

```
  return $post; // on retourne un objet à null ou contenant le resultat
}


 /*
  * findTousById_Article : renvoie un tableau d'objets Post
  * à null ou avec les N Post à afficher pour $id_article
  */

public static function findTousById_Article($id_article) {
  $query = "select * from post_article
           where id_article = ".$id_article ;
  $result = mysql_query($query) ;
  if (empty($result))
    { $post = null ; }
  else
    {
    $i = 0 ;
    while ($record = mysql_fetch_object($result))
      {
      $post[$i] = new Post($record);
      $i++ ;
     }
    }
  return $post;
  }
```

`findTousById_Article($id_article)` recherche tous les posts d'un même article et les renvoie dans un tableau d'objets `Post`.

```
  /*
   * AfficheTousPost : affiche
   * les N post du tableau $post passé en argument
   */

public static function AfficheTousPost($post) {
  $i=0 ;
  echo '<br /><br />' ;
  while ($post[$i])
    {
    echo '<div class="jcmbfilet570">' ;
    echo '<b>Publié par : </b>' . $post[$i]->nom;
    echo '   ' . $post[$i]->prenom . '<br />' ;
    echo '<h2>' . $post[$i]->titre_post . '</h2>' ;
```

```
    echo '<b>Message ou question</b> <p>' . $post[$i]->text_post . '</p>
  ➥ </div>' ;
    echo ' <br />' ;
    $i++ ;
    }
  }
```

`AfficheTousPost($post)`. Dans un conteneur `<div>`, on affiche chaque objet `Post` du tableau d'objets `$post` :

```
} // de la class PostHelper


class Post {
  /*
   * Attributs de l'objet Post
   */
  public $id;
  public $id_article;
  public $nom;
  public $prenom;
  public $email;
  public $text_post;
  public $titre_post;

  /*
```

Notre objet `Post` est désormais défini. Chacun pourra l'enrichir à sa convenance. Par exemple, l'ajout d'un champ *date* permettrait de noter la date de publication de chaque post :

```
   * Constructeur d'un objet : de $record
   * dans les attributs de l'objet juste créé en mémoire
   */
  public function __construct($record="") {
    if ($record)
      {
      $this->id = $record->id;
      $this->id_article = $record->id_article;
      $this->nom = $record->nom;
      $this->prenom = $record->prenom;
      $this->email = $record->email;
      $this->text_post = $record->text_post;
      $this->titre_post = $record->titre_post;
      }
    }
```

La commande `New` de PHP peut construire un objet `Post` :

```
      /*
       * create un enregistrement dans la table depuis
       * l'objet Post courant (this)
       */
  public function create() {
    $query = "insert into
              post_article
                 (id_article, nom, prenom,
                  email, text_post, titre_post)
                  values
                   (
                    '".$this->id_article."',
                    '".$this->nom."',
                    '".$this->prenom."',
                    '".$this->email."',
                    '".$this->text_post."',
                    '".$this->titre_post."'
                    )";
    mysql_query($query)
      || die( "ERREUR : MySQL : Impossible create post "
             . mysql_error() );

  }

} // de la class Post
?>
```

- `create()` crée un enregistrement correspondant à un objet `Post` dans la table *post_article*.

Nous séparons la manipulation des données des contraintes d'affichage. Dans *php_inc/MiseEnPage.inc.php*, la fonction `message_cet_article` appelle le `<div class="jcmbparalarge">`. Ensuite, cette fonction affiche via de simples balises type `h1`, `h2`, `span` et `p`. Le comportement et l'affichage des contenus gérés par ces balises sont pilotés par la feuille de style dont voici un extrait :

```
.jcmbparalarge {
width:570px;
height: auto; padding: 5px 3px 0 3px; float:right ; }

.jcmbparalarge h1 {
font-size:14px; color:#2C6A93;
```

```
background-color:inherit;height:25px;
font-weight:bolder}

.jcmbparalarge h2 {
font-size:12px; color:#2C6A93;
background-color:inherit;height:25px;}

.jcmbparalarge p {
font-size:12px; color:black; background-color:inherit;
border:1px solid grey; padding: 5px 3px 0 3px;
font-weight:normal}

.jcmbparalarge span {
font-size:14px; color:#2C6A93;
background-color:inherit;height:25px;
font-weight:bolder}

.jcmbparalarge a{
color:#2C6A93; background-color:inherit;
text-decoration: underline}

.jcmbparalarge a:hover{
color:#ffbe33; background-color:inherit;
text-decoration:underline}

.jcmbfilet570{
width:570px; height: auto; padding: 5px 3px 0 3px;
float:right ; border:1px solid #2C6a93;}
```

La feuille de style du formulaire de notre exemple est dérivée d'un exemple ou d'informations techniques exposés dans ces pages :

- **http://www.cssdebutant.com/formulaire-css.html** ;
- **http://www.babylon-design.com/site/index.php/2007/09/24/192-boutons-extensibles-css-compatibles-tous-navigateurs.**

La voici :

```
.cssform p{
width: 400px;
clear: left;
margin: 0;
padding: 5px 0 8px 0;
padding-left: 155px;
/*width of left column : the label elements*/
```

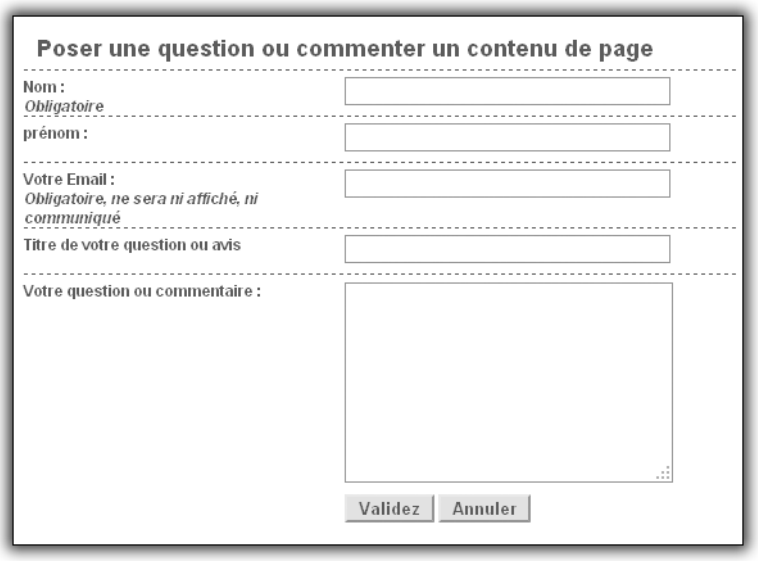
```
border-top: 1px dashed #2C6A93 ;
height: 1%;
}

.cssform label{
font-weight: bold;
color:#2C6A93 ;
float: left;
margin-left: -155px;
/*width of left column*/
width: 250px;
/*width of labels. Should be smaller than left column
(155px) to create some right margin*/
}

.cssform input[type="text"]{
/*width of text boxes. IE6 does not understand this attribute*/
width: 250px;
}

.cssform textarea{
width: 250px;
height: 150px;
}

/* Bouton avec effet - ne marche pas en IE6 ? */
input[type=submit], input[type=reset] {
border:2px outset #EAEAEA ;
background-color: #E3E3E3 ;
font-weight:bold; color:#2C6A93 ;
cursor:pointer;
}
input[type=submit]:hover, input[type=reset]:hover {
border:2px outset white;
background-color:white;
}
input[type=submit]:active, input[type=reset]:active {
border:2px inset #A6BEDE;
background-color:#2C6A93;
color:black;
}
```

En cliquant sur le lien pour ajouter un post à la page, nous accédons au formulaire, volontairement mis à l'écart du code source de *zoom.php*.

Chaque fois qu'on isole les codes sources dans des fichiers séparés, se pose la question des passages d'arguments et de la compatibilité avec le référencement sur Google. Le plus simple est de tout grouper dans une page mais cela alourdit le code source.

Dans notre exemple, nous sommes dans un formulaire dont l'URL sera propre à chaque page article de départ. Pour éviter de massifs duplicate contents, les pages formulaires sont en `noindex, nofollow`, et tout lien menant à elles sera en `rel="nofollow"`.

Voici le code source du formulaire :




► Fig. 5.6 : Le formulaire à afficher

```
<?php
// post_affiche_formulaire_saisie.php

require_once("php_inc/MiseEnPage.inc.php");
$title_page = "créer sa boite
  - formulaire poster un commentaire" ;
$meta_description_content = "créer sa boite
  - formulaire poster un commentaire" ;
$rep = "./" ;
$follow_index = 0 ; // pas de dup content
```

Pour éviter des duplicate contents à cause de ce formulaire identique pour tous les articles, on le passe en `noindex, nofollow` :

```
MiseEnPage::haut($title_page ,
                 $meta_description_content ,
                 $rep , $follow_index ) ;
?>

<div id="sousban"> </div>
<div id="pagegauchetete">
<?php
MiseEnPage::MenuGauche() ;
if (isset($_GET["id_article"]))
  $id_article = $_GET["id_article"];
if (isset($_GET["url"]))
  {
  $url_de_zoom = $_GET["url"]; // où il faudra retourner
  }
```

On mémorise les deux arguments passés en paramètres : `id_article` et `url`. En fait, l'URL contient uniquement l'URI de la page *zoom.php* d'origine. Comme nous le verrons dans le prochain script, c'est suffisant et cela sécurise notre application. En effet, une URL ne peut dépasser 255 caractères. En utilisant strictement l'URI bien plus courte que l'URL complète, on reste éloigné de cette limite :

```
?>
</div>   <!-- fin page gauche tete -->

<div id="pagecentretete">

<h1> Poser une question ou commenter un contenu de page </h1>

<form id="Formulaire_saisie_post"
      name="Formulaire_saisie_post"
      method="post" class="cssform"
      action="<?php
              $s = 'post_traite_formulaire_saisie.php ?id_article='
                           . $id_article . '&url='
                           . $url_de_zoom ;
                        echo $s ;?>" >
```

La seule méthode pour passer une URL avec des arguments dans un champ *action* d'une `form` consiste à faire un `echo` PHP de la chaîne de caractères :

```
<p>
<label for="nom_posteur">Nom : <br /><i>Obligatoire</i></label>
```

```
<input type="text" name="NOM" value="" />
</p>

<p>
<label for="prenom_posteur">prénom : </label>
<input type="text" name="PRENOM" value="" />
</p>

<p>
<label for="email">Votre Email : <br /><i>Obligatoire, ne sera ni
affiché, ni communiqué</i></label>
<input type="text" name="EMAIL" value="" />
</p>

<p>
<label for="titre">Titre de votre question ou avis </label>
<input type="text" name="TITRE_POST" value="" />
</p>

<p>
<label for="comments">Votre question ou commentaire : </label>
<textarea name="TEXT_POST" rows="10" cols="40"></textarea>
</p>

<div style="margin-left: 250px;">
<input type="submit" value="Validez" /> <input type="reset"
value="Annuler" />
</div>

</form>

</div> <!-- fin pagecentre tete -->

<?php
  MiseEnPage::bas() ;
?>
```

Le reste du formulaire est classique et ne demande pas de remarque particulière.

Un clic sur **Envoyer** lance le script *post_traite_formulaire_saisie.php*.

```
<?php
// post_traite_formulaire_saisie.php

require_once("php_inc/init.inc.php");
require_once("php_inc/RecordPost.inc.php");
require_once("php_inc/MiseEnPage.inc.php");
```

Les différentes bibliothèques dont nous auront besoin :

```
// mise en page pour affichage des messages
$title_page = "Page - suivi Post";
$meta_description_content = "Page - suivi Post";
$rep = "" ;
$follow_index = 0 ; // formulaire - pas d'indexation
```

On ne fait pas indexer cette page par Google :

```
MiseEnPage::haut($title_page ,
                 $meta_description_content ,
                 $rep ,
                 $follow_index ) ;

if (isset($_GET["id_article"]))
  $id_article = $_GET["id_article"];
if (isset($_GET["url"]))
  {
  $url_de_zoom = $_GET["url"]; // où il faudra retourner
  }
```

On mémorise les informations pour construire le lien qui nous renverra vers la page d'origine :

```
if( ($_POST['NOM'] == "") || ($_POST['EMAIL'] == "") )
  {
  $champ_non_saisi = "Nom ou email non saisi ... Ces données sont
➥ obligatoires - Merci de les saisir" ;
  }
```

Si les informations obligatoires ne sont pas saisies, on bloque :

```
else
  {
  $post = new post() ; // créer objet vide
```

On crée en mémoire notre objet Post, `$post` :

```
  // puis on l'instancie avec le formulaire
  $post->id_article = $id_article ;
  $post->nom = ereg_replace("<[^>]*>",
               "", trim(strip_tags($_POST['NOM'])) );
  $post->prenom = ereg_replace("<[^>]*>",
               "", trim(strip_tags($_POST['PRENOM'])) );
  $post->email = ereg_replace("<[^>]*>",
               "", trim(strip_tags($_POST['EMAIL'])) );
```

```
  $post->titre_post = ereg_replace("<[^>]*>",
                "", trim(strip_tags($_POST['TITRE_POST'])) );
  $post->text_post = ereg_replace("<[^>]*>",
                "", trim(strip_tags($_POST['TEXT_POST'])) );
```

- `ereg_replace("<[^>]*>", "", trim(strip_tags($_POST['TEXT_POST'])) )`renvoie une chaîne de caractères exempte de tout code HTML ou autres.
- `ereg_replace("<[^>]*>"` retire toute balise HTML. Cette fonction est en doublon apparent avec `strip_tags()`. À une (lointaine) époque, `strip_tags()` n'était pas aussi fiable qu'actuellement.
- `Trim()`supprime les caractères invisibles comme les tabulations, par exemple.
- `strip_tags()`retire toute balise HTML ou PHP.

Cela sécurise le fonctionnement de notre site web, par exemple :

- aucun code HTML perturbant l'affichage ;
- aucun caractère invisible perturbant l'affichage ou le fonctionnement des scripts PHP.

```
  $post->create(); // ecrire dans la table MySQL
```

L'objet `Post` créé depuis le formulaire saisi est ajouté à la table `post_article`.

```
  }

if (!$champ_non_saisi)
  {
  $url_de_zoom = str_replace("|","&",$url_de_zoom );
// on remet le & filtré par $_get
```

Le caractère `&` est filtré lors des passages d'arguments. On ne se retrouve donc pas avec des URI amputées quand on les passe en argument. En remplaçant `&` par `|` puis en inversant quand on a besoin de l'URI originale, on règle ce problème.

```
  Print "
    <h1> Merci </h1>
    <h3>
    Nous vous remercions d'avoir posé une question ou d'avoir commenté
cette page
    <br /><br /><br />
       <a href='http://". $_SERVER['HTTP_HOST'] . $url_de_zoom . "'>
Retour à la page d'origine </a>
```

On reconstruit l'URL complète composée de `http://` concaténée au nom de domaine `$_SERVER['HTTP_HOST']` puis à l'URI d'origine. On ne peut pas faire autrement.

En effet, un JavaScript pilotant l'historique pour faire deux coups en arrière ne recharge pas la page. Donc le post nouvellement ajouté n'y figure pas. Voilà pourquoi on mémorise depuis le début l'URI de notre page de départ. C'était pour pouvoir y retourner avec un rechargement complet de la page donc avec le post juste ajouté.

```
    </h3>" ;
  }
else
  {
  Print "
    <h1> Erreur de saisie </h1>
    <h3> Nom ou email non saisi ... Ces données sont obligatoires -
Merci
    <br /><br /><br />
       <a href='javascript:history.back()'> Retour au formulaire -
contenu déjà saisi maintenu</a>
```

L'internaute peut retourner au formulaire dont la saisie a été conservée. Il peut ainsi corriger et cliquer une nouvelle fois :

```
    <br /><br />
       <a href='index.php'> Retour à la page d'accueil </a>
    </h3>
    ";
  }

MiseEnPage::bas() ;
?>
```

## 5.4 Résumé de ce chapitre

Nous avons vu comment intégrer dans le référencement et profiter de deux parties du Web 2.0 :

- les liens sociaux ;
- l'interactivité avec les internautes.

Dans notre exemple, nous nous sommes contentés de gérer des questions ou commentaires d'internautes par page publiant un article complet.

D'autres aspects, d'autres "jeux" incitant les internautes à intervenir sur une page du site, et ainsi contribuer à son référencement, pourront être mis en œuvre :

- Notez ce contenu.
- Commentez votre note, etc.

Nous avons vu qu'il valait mieux éviter de laisser Google indexer les formulaires. Il est si facile de créer des duplicate contents dès qu'on code avec des passages d'arguments en URL.

Une règle est ainsi rappelée : pas de passage d'arguments en URL ou ces pages ne sont pas soumises à indexation auprès de Google.

Nous avons entr'aperçu une autre approche technique : plutôt que de prévoir N pages se répartissant les scripts, unifiez tout sur une page. Cela simplifie les passages d'arguments en URL. Cela peut poser quelques problèmes avec Google Analytics par exemple (nécessité de disposer d'une URL à dédier à un remerciement pour pouvoir suivre des objectifs).

Les algorithmes de base vus dans ce chapitre sont déclinables à l'infini selon vos cas et besoins particuliers.

# 6

# Pages absentes ou mal positionnées dans Google

De nombreuses causes isolées ou cumulées peuvent conduire à une ou plusieurs pages absentes de l'index de Google ou très mal positionnées. Elles sont d'origine techniques ou rédactionnelles ou les deux à la fois. Les causes techniques sont du domaine du webmestre (HTML, maintenance) ou du codeur (script PHP).
Ces causes sont à l'origine de blocages des bot ou de l'applications de pénalités par Google envers ces pages. Nous allons lister dans ce chapitre l'éventail le plus large possible de ces causes ainsi que les approches pour se dégager de ces problèmes.

## 6.1 Symptômes et causes.

Un webmestre codeur peut se heurter à différents problèmes qui présentent tous les mêmes symptômes, très vagues, sur une ou plusieurs pages d'un site web :

- pages absentes de l'index de Google ;
- pages mal positionnées sur leurs expressions clés.

Généralement, plusieurs causes se combinent pour aboutir à un de ces deux symptômes.

Refaisons un point sur ce que nous avons vu dans les chapitres précédents pouvant occasionner ce type de symptôme. Nous étudierons ensuite les nombreuses autres causes.

### Les pénalités distribuées par Google

La liste n'est pas exhaustive :

- Désindexation de page : la page est difficile à trouver. La commande site permet de l'afficher ainsi qu'une recherche sur un bloc entier de la page. La *Google bar* indique une mesure grisée.
- Désindexation partielle : sur certaines phrases pourtant présentes en contenu, la page est très difficile à faire apparaître en SERP de Google. Quand elle apparaît, elle est seule dans le résultat.
- Liste noire : toutes les pages du site sont inexistantes dans l'index de Google.
- Page absente mais d'autres pages du site sont présentes.
- Le retrait de Page rank. Les pages suspectées d'avoir des liens trop incohérents voient leurs Page rank fondre.

Les désindexations sont des pénalités Google classiques pour différentes formes et différents volumes de duplicate contents.

La mise en liste noire sanctionne des optimisation trop agressives, typiquement du spamdexing. Il ne vous reste plus qu'à faire une demande de réinclusion. Nous conseillons vivement de suivre les recommandation de cette page : **http://www.mattcutts.com/blog/reinclusion-request-howto/**.

Une page absente quand d'autres pages du site sont présentes indique potentiellement :

- Une inaccessibilité de la page par un lien valide. Cherchez les liens en JavaScript ou en Flash vers cette page, vous les trouverez. Cherchez des liens en dur classiques ; vous devriez découvrir qu'ils sont inaccessibles aux bots de Google. Il est donc impossible d'avoir la page indexée.

- La page est enfouie très profondément dans une hiérarchie de niveaux. En conjuguant cela avec un Page rank faible, vous obtenez une page qui mettra du temps avant d'être prise en compte dans l'index de Google.

## Les causes déjà vues

Nous avons vu dans les chapitres précédents plusieurs causes pouvant entraîner des pénalités contre des pages dans un site web :

- Duplicate content : de nombreuses maladresses en gestion de contenus ou en programmation peuvent aboutir à la désindexation d'une partie des pages d'un site. Les différents paragraphes des chapitres précédents sur l'URL rewriting, les URL avec passage d'arguments, les copiés/collés maladroits traitent de ce sujet.
- Une optimisation trop agressive ou maladroite déclenchant un spamdexing et entraînant des pénalités de Google.
- La mise en œuvre de volumes de liens incohérents avec vos contenus faisant passer vos pages pour des fermes de liens à louer, des fermes de *paidlinks*.
- Avoir mis en œuvre des techniques de triche, hors paidlinks, et s'être fait prendre par Google.
- *.htaccess* mal codé entraînant des duplicate contents.

L'absence de qualité dans le travail de référencement n'occasionne pas uniquement des pénalités actives de Google. Cela peut tout simplement faire effondrer le positionnement de toutes les pages du site sur tous les mots-clés visés.

Illustration :

- Un site web peut avoir une page d'accueil avec un Page rank de 5. De nombreux backlinks pointent vers lui.
- Toutes les pages internes possèdent un Page rank confortable, de 2 à 4.
- Aucune pénalité n'est présente sur une page du site.
- Mais toutes les pages du site sont absentes de toute expression-clé, sauf la page d'accueil sur le nom de domaine du site. Là, c'est désastreux.

Nous avons vu au chapitre 1 différentes approches à mettre en œuvre pour réussir un référencement. Négliger ces approches n'implique pas des pénalités actives de Google. Cela peut simplement signifier une absence en positionnement faute de référencement. Vous pourrez avoir tous les backlinks du monde, cela ne changera rien. Vos pages seront mal positionnées faute de référencement de qualité.

⚠ Positionnement et qualité du référencement

Un positionnement réussi passe par un référencement de qualité. Sans un travail de référencement réussi sur l'ensemble des pages du site, sans une stratégie globale de référencement, un site web aura plus ou moins de mal à optimiser le positionnement de ses pages sur des expressions clés.

Nous avons vu que le référencement puis le positionnement réussi de pages précises sur des expressions clés précises est issu de la qualité du travail de référencement sur toutes les pages du site. Si une partie des pages reçoit des pénalités, c'est toute la stratégie de référencement qui est affectée.

Ainsi, des causes apparemment indirectes peuvent se cumuler avec d'autres erreurs ou maladresses et aboutir à un positionnement raté.

## 6.2 Pages mal positionnées : d'autres causes

La maturité croissante des webmestres et des codeurs en référencement, la création massive de nombreux postes de traffic manager en charge de superviser la création de trafic dont le référencement naturel, aboutissent à une compétition plus professionnelle sur les expressions clés.

### Les sites concurrents

Formulé autrement, cela revient à dire que vous êtes peut être mal positionné malgré vos efforts parce que vos concurrents sont meilleurs que vous :

- Ils ont commencé avant vous et les algorithmes de Google donnent une forme de "prime" à l'ancienneté, visant essentiellement ceux qui sont arrivés les premiers en page 1 des SERP. Chaque internaute qui clique sur une URL depuis une SERP Google enrichit des bases de statistiques et cette URL se trouve renforcée dans son positionnement. Difficile d'obtenir ce genre de bonus si on est justement absent des places en or dans les SERP de Google.
- Si l'expression clé visée fait l'objet d'une concurrence acharnée entre de nombreux acteurs, le moindre défaut dans le référencement des pages aboutit à un recul immédiat et important. Un seul critère incohérent et c'est un recul immédiat de plusieurs places, ce qui entraîne un taux de clics moindre et donc, si rien n'est fait, le début d'un décrochage possible hors des places en or dans les SERP.

Le nombre d'expressions clés propres à une profession est plus ou moins limité. Quand on loue des bureaux par exemple, le nombre d'expressions clés est très limité. Vu le nombre de société louant des bureaux, vides ou meublés, sur Paris par exemple, on voit

tout de suite que dès que toutes ces sociétés auront professionnalisé leurs sites et le référencement associé, la bataille pour "*location bureau Paris*" va devenir acharnée.

## Les backlinks

Les sites concurrents ont des backlinks de meilleures qualité et en plus grand volume que votre site. Si vos concurrents gèrent des tags cloud et pas vous, s'ils ont automatisé leur référencement et pas vous, vous savez ce qu'il vous reste à faire.

Mais il y a aussi les backlinks externes. Nous quittons alors le domaine de ce livre et passons au web marketing. Je vous renvoie sur ce sujet au *Guide Complet du référencement sur Google* édité chez Micro Application.

## JavaScripts

Tout lien inséré dans un JavaScript est invisible pour un bot de Google. Si tous vos menus sont en JavaScript, il va falloir prendre d'autres mesures :

- remplacer ces menus déroulants JavaScript, en général très sympathiques et très jolis, par l'équivalent en HTML par exemple ;
- laisser les menus en place et doubler ces menus par des arborescences de liens en bas de page.


► Fig. 6.1 : Menus en arborescence de liens en bas de page

Notre préférence va paraître surprenante ; la deuxième option : doubler les menus JavaScripts par les liens en bas de page.

Voici pourquoi :

- Les menus JavaScript s'adressent exclusivement aux internautes humains. On peut faire tout ce que l'on veut. Aucun impact sur le référencement.
- Les arborescences de liens en bas de page sont peu visibles pour les humains. Ils ne choquent personne. On vient simplement de décliner une forme de tags cloud, de nuage de liens.
- Il n'est pas évident de réussir à coder un menu déroulant en HTML, qui soit lisible par tous les navigateurs, validé par le web marketing, accessible techniquement aux bots des moteurs de recherche et qui contiennent les mots-clés dont le référenceur a besoin.

Personne ne sait exactement comment réagit et comment réagira Google sur les liens en bas de page. Existe-t-il des pénalités quand ils ne sont pas en haut de page ? Quelques tests n'ont rien montré de tel. Par prudence, il serait bien de :

- Faire apparaître des éléments après l'arborescence de liens : un pied de page avec "mentions légales " et autres répétitions, des éléments de charte graphique, etc.
- Écrire ces liens dans une taille de police de 10 minimum.
- Appliquer les même règles de prudences en lutte contre le duplicate content que pour les nuages de liens. L'ordre des liens doit changer, les mots-clés doivent varier, le texte des liens doit légèrement changer à chaque page.

Au vu de nombreux audits et tests, actuellement, le pire ennemi d'une arborescence de liens en bas de page est le duplicate content et non le risque de voir Google ne pas prendre en compte des liens placés trop bas dans une page.

## Frame

Le traitement technique pour les frames est très simple : la poubelle.

Les raisons d'exister de cette technique ont disparu :

- Débit lent et cher du réseau : qui se souvient de RNIS 2 canaux B à 64 kbit/s facturés par France Telecom au double du tarif communication nationale pour un débit réel oscillant entre 1,5 et 5 fois un modem classique ? Les notes Internet étaient de 2 000 francs HT/mois en moyenne. (En tenant compte de l'inflation, cela fait 500 à 600 euros HT/mois d'aujourd'hui.) Les frames diminuaient le besoin de bande passante.
- De tout petits écrans sur les PC et les Mac.

De nombreuses études ont montré que les usagers n'aimaient pas les sites utilisant des frames.

Des spécialistes ont développé des techniques pour réussir le référencement de sites utilisant des frames : pourquoi pas. Mais si on doit conjuguer la complexité des frames et celle automatisant le référencement tel qu'on le voit dans cet ouvrage, les codes sources vont prendre un fort embonpoint préjudiciable à une maintenance aisée et rapide.

## Les limites des feuilles de style

Google appréhende mal les feuilles de style, sauf quand il s'en donne la peine, c'est-à-dire rarement et généralement pour chercher des fraudes.

Nous devons donc rester sur la première option : faire simple et compréhensible en feuille de style afin d'être sûr que Google interprète correctement les balises `Hn`.

Si vous redéfinissez les balises `Hn` de manière invisible à Google via la feuille de style, ou, autre variante, comme l'exemple ci-après, elles seront invisibles comme balises `Hn` pour les bots de Google.

```
<span class="titre4"> titre avec mots clefs </span>
```

Votre codeur XHTML/CSS a défini des classes titres propres au site en feuille de style. Elles sont sûrement très jolies mais sans utiliser les balises `Hn`, le référencement de votre page sera sérieusement handicapé.

Un titre devrait être codé ainsi pour qu'il soit accessible techniquement et sémantiquement à Google :

```
<div class="maclasse">
<h1>Titre avec mots clefs </h1>
<span> Titre destinés aux internautes humains,
jeux de mots, humour etc. autorisé ici
</span>
</div>
```

avec une feuille de style qui ressemblerait à ceci :

```
.maclasse {
width:570px;
height: auto; padding: 5px 3px 0 3px; float:right;}
```

On définit une classe. Dans celle-ci, on va redéfinir toutes les balises à notre convenance, graphique, mais aussi pour le référencement :

```
.maclasse h1 {
```

```
font-size:14px; color:#2C6A93;
background-color:inherit;height:25px;
font-weight:bolder;}
```

Cette balise H1 est personnalisée pour le site mais, avec une taille de 14px, elle est plutôt discrète pour une balise de cette importance. Elle nous permettra de faire des titres truffés de mots-clés mais relativement peu visibles.

```
.maclasse h2 {
font-size:12px; color:#2C6A93;
background-color:inherit;height:25px;}
```

Même chose pour cette balise H2 ; elle est plutôt discrète en visibilité sur une page. Afin de ne pas abuser des balises H1 sur une page, on pourra utiliser à bon escient cette définition de H2.

```
.maclasse p {
font-size:12px; color:black; background-color:inherit;
border:1px solid grey; padding: 5px 3px 0 3px;
font-weight:normal;}
```

Cette définition est classique pour du texte en paragraphe :

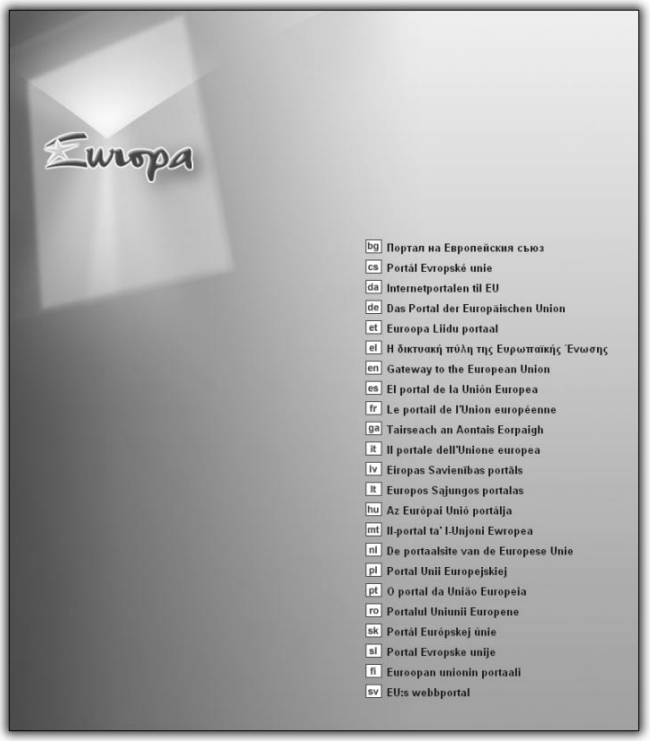
```
.maclasse span {
font-size:20px; color:#2C6A93;
background-color:inherit;height:25px;
font-weight:bolder;}
```

Ce span, lui, est exceptionnel. Colorié, 20px en taille de font, gras, un texte entre balises  sera très visible sur la page. L'usage en est tout trouvé : il servira pour les titres destinés aux humains. Vous pourrez faire usage d'humour, de jeux de mots, etc.

Pour Google ce n'est que du texte, quelques mots bizarres noyés dans les mots de toute la page. Les conséquences négatives sont infimes, négligeables.

Résumons-nous : on a bien redéfini les balises Hn et elles restent lisibles comme balises Hn, ce qui est fondamental pour le référencement réussi d'un site.

## Splash Page




► Fig. 6.2 : http://europa.eu/ La splash page la plus célèbre d'Europe

Les *splash pages* permettent de regrouper des informations directionnelles sur la page d'accueil. Ici, sur le site de l'Union européenne, le choix est offert de choisir parmi les 22 langues actuelles de l'Union européenne (le Groenlandais n'est pas encore une langue officielle ; inutile de la chercher dans la liste).

Une splash page est très pratique quand on ne sait pas trop comment rediriger un internaute humain : choix de la langue, choix du métier de l'entreprise, etc.

Mais une splash page est une catastrophe pour le référencement d'un site :

- Elle bloque avec des informations sans intérêt la page d'accueil qui aurait mieux à faire.
- Elle gaspille le Page rank. Chaque saut dans un niveau hiérarchique coûte en général un point de PR, en tout cas sur un site hiérarchisé, sans nuages de liens et sans interférences avec des backlinks externes.

- Impossible de placer un nuage de liens sur une telle page : elle n'est cohérente avec rien.

Vous l'aurez compris : l'Union européenne peut se permettre une splash page ; faute de concurrence, les inconvénient cités précédemment ne feront guère de mal à son trafic. Le site web d'une PME lui ne peut se permettre un tel luxe pour un usage aussi inutile et superflu.

### ⚠ Splash page

On peut déclarer comme Splash page toute page d'accueil ne contenant pas de texte ou presque pas de texte et servant à rediriger les internautes vers des sous-choix : par langues, par produits, par types de spectacle, etc.

Il y a deux approches pour contourner une splash page :

- On la laisse et on ajoute un copieux bas de page afin de lester cette page en volume de caractères.


► Fig. 6.3 : Un bas de splash page sous forme d'un copieux texte

La page d'accueil du site de la figure C06-30 a tendance à être une splash page. Pas assez de texte unique, texte de page d'accueil pas assez ou mal centré sur les mots-clés, nous avons là une splash page. En ajoutant un copieux bas de page strictement composés de textes uniques et utilisant les mots-clés du métier, de balises `Hn`, le positionnement de cette page d'accueil a réussi.

- L'autre approche : on conçoit une page d'accueil intégrant la redirection des internautes vers des sous-destinations.

Pour le codeur, le problème est simple à résoudre. Soit la page est conçue une nouvelle fois pour éliminer la splash page, soit il suffit de coder une zone de `Hn` et de `<p>` pour permettre le lestage en mots-clés et autres textes de la *home page*.

## Une page : une langue

La norme W3C permet de mixer les langues sur une même page du côté du contenu. Mais la norme impose une seule langue pour la partie "technique" : title de page, meta description.

Google fait plus simple : une page ne peut être qu'en une seule langue, aussi bien pour la partie technique que pour la partie contenue. Pire, Google demande de spécialiser un répertoire par langue.

Si vous mélangez des langues sur une même page, vous risquez d'avoir de mauvaises surprises en positionnement.

## Spamdexing involontaire

Que ce soit en JavaScript ou en script PHP côté serveur, voici la mise en œuvre de divers algorithmes pas assez testés peut aboutir à la création de spamdexing :

- Police de caractère ayant la même couleur que le fond de page.
- Texte variant selon le profil de l'internaute : il accepte ou non les cookies sur une page ayant la même URL. Attention, cela revient à afficher un texte selon l'origine, un moteur de recherche ou un humain : nous avons alors un *cloaking*.
- La même base MySQL fait fonctionner différents sites avec les même données.

Les possibilités d'erreurs sont vastes. Les calques DHTML sont une source d'optimisation intéressante mais aussi de bogues dégénérant en spamdexing. Un texte reste verrouillé en `hidden` sans aucune possibilité de le faire revenir en `visible`. Vous avez un spamdexing très facile à détecter.

## Détecter une triche : facile !

Pour illustrer la facilité avec laquelle on peut découvrir une triche implantée dans un programme, voici un algorithme parmi de nombreux autres en charge de traquer les scripts oubliant de passer un texte, indexé par Google, en visible pour l'internaute.

La théorie des graphes s'applique au fonctionnement du Web. Rappelez vous, un point = une page et un arc = un lien. On peut aussi utiliser cette branche des mathématiques pour analyser un code JavaScript manipulant les calques. Rien de plus simple.

Un point = un calque. Un arc = le basculement d'un état vers un autre : `visible` vers `hidden` et inversement.

Un bot anti-fraude suit le fil de la page et cartographie, dans le code JavaScript, les calques et les changements d'état.

Un programme sur le serveur transpose ceci dans une matrice. Très vite, les boucles non fermées sur `hidden` apparaissent. Chacune de ces boucles non fermées correspond à un calque restant bloqué sur `hidden`, donc un texte invisible à l'internaute, mais ce même texte est lu par les bot "*normaux*" de Google et indexé comme du texte normalement apparent aux humains : spamdexing détecté.

Cette technique ou des variantes sont appliquées dans de nombreux domaines du génie logiciel, notamment pour fiabiliser des codes critiques : missiles, fusées, avionique…

Pour conclure cette illustration : aucune technique de triche en programmation ne peut rester indétectable.

Pour aller chercher les limites de Google, il faut aller sur des terrains "*mouvant*" où l'humain, le code et l'environnement se mélangent et saturent ses algorithmes anti-fraudes. C'est un des points abordés au chapitre suivant.

## Optimisations trop agressives

On rentre partiellement dans la Google fiction. Sous ce doux terme, on désigne des techniques *black hat* qui consistent à tricher selon les mécanismes que nous avons déjà abordés. Voilà pour la partie réelle.

La technique la plus répandue, très facile à mettre en œuvre, consiste à noyer sous un même mot-clé une page d'atterrissage. Cette erreur est assez répandue.

Pour la partie fiction, voici des "optimisations trop agressives" non encore traquées par Google, mais qui pourraient le devenir un jour.

L'automatisation des pages permet de personnaliser les balises `ALT` et `TITLE` d'un visuel pour chaque page en fonction du contenu. Un petit rappel de la norme :

- `ALT` est une balise utilisée par les terminaux pour mal voyant. Elle "traduit" le contenu d'une image.
- `TITLE` est une balise info bulle à destination des voyants. Elle complète le visuel par un texte s'affichant quelques secondes à côté de l'image.
- En clair, il n'y a aucune raison que ces descriptifs changent pour chaque page pour un même visuel.

Utiliser les script PHP/MySQL pour personnaliser à outrance le contenu d'une page en personnalisant les balises des images répétées sur chaque page du site est donc un pas trop vite franchi.

On peut aussi personnaliser les pictogrammes, les traits, etc.

Là, on obtient une optimisation trop agressive et facile à découvrir. Les même fichiers images changent de balises `ALT` et `TITLE` à chaque page du site.

Puisque c'est facile à découvrir, utilisons une base MySQL et un script PHP pour changer les noms des fichiers images et les personnaliser pour chaque page en même temps que leurs balises `ALT` et `TITLE`.

Mauvaise idée. En effet, à l'heure où Google tente de finaliser un logiciel capable d'indexer une image d'après son contenu, il serait maladroit de laisser une même image dans des fichiers différents pour des pages distinctes et avec une personnalisation adaptée à chaque page. Nul doute que cela fonctionnera un certain temps. Seul Google sait combien de temps il laissera des black hat flouer ainsi ses algorithmes avec des optimisations trop agressives.

**Une optimisation pour demain**

Utiliser une bibliothèque graphique pour automatiser la personnalisation des images elles-mêmes en fonction du contenu de la page est une idée à creuser, prometteuse. Sur ces images ainsi modifiées, on pourra greffer sans crainte une personnalisation des balises `ALT` et `TITLE`.

## 6.3 Flash et le référencement dans Google

La construction d'une animation flash peut être classée en deux parties distinctes :

- l'animation elle-même ;
- la navigation entre animation, textes et autres objets.

La nuance est importante en référencement.

Il n'est pas possible de référencer correctement un "site" en flash, c'est-à-dire des animations interconnectées entre elles via des liens flash. Les philosophies de développement de programmation sont trop éloignées des normes Google et du W3C pour permettre une adaptation aisée de l'un à l'autre. Même avec l'aide de Adobe, Google pourra difficilement atteindre la qualité technique qu'il a atteint sur le HTML classique. D'autres raisons militent pour une prise en compte minorée des animations flash :

- Tant que Google ne pourra pas injecter du AdWords dans les animations flash, référencer des sites flash complets revient à se tirer une balle dans le pied.
- Les *SEO black hat* pourraient profiter de la situation pour mettre au point des stratégies en charge de flouer les algorithmes de Google.
- Nous ne voyons pas comment un lien social ou un bookmark pourrait pointer une partie précise à l'intérieur d'une animation flash. C'est à l'opposé de la philosophie Google actuelle.

En revanche, des animations flash simples, stockées individuellement sur des pages HTML reliées entre elles par de classiques liens offrent d'excellentes perspectives. On peut référencer ces pages. En effet, petit à petit Google devrait rentrer dans les animations simples pour y lire certaines données. Mais surtout, le plus important est que nous disposons de la page elle-même pour effectuer le travail classique de référencement.

Et puis touche finale, en considérant une animation flash comme un objet multimédia parmi d'autres, on ne contrarie en rien la stratégie de Google.

La lecture des informations publiées par Google conforte cette approche quand on les lit entre les lignes.

## SWFobject2 pour indexer du Flash

**http://code.google.com/p/swfobject/wiki/documentation** sur lequel on trouve l'URL de la traduction française de SWFobject2 : **http://egypte.olympe-network.com/swfobject-francais.html**.

SWFobject2 constitue un indéniable progrès pour les développeurs. Même les référenceurs y trouvent des améliorations. En détaillant la nouvelle formule SWFobject2, on découvre de nombreuses améliorations concernant l'intégration de Flash au monde HTML, mais c'est hors contexte du présent livre et les quelques possibilités offertes en référencement.

Il est à retenir aussi que l'initiative SWFobject2 semblerait devenir le standard pour lancer une animation Flash.

Un contenu alternatif peut être défini et soumis pour chaque animation flash au navigateur de l'internaute. On rejoint la balise `ALT` pour les images en HTML.

Le contenu alternatif à l'animation Flash est soumis à Google et aux autres moteurs de recherche. En examinant de près ce contenu alternatif, nous avons :

- un bloc de texte ;
- associé à l'animation d'un bloc.

En clair, il y a tout ce qu'il faut pour aider au référencement d'une animation Flash simple. Si votre animation Flash contient la totalité de votre "site", vous aurez juste un bloc de texte pour décrire le tout, ce qui se révèlera totalement insuffisant face à des concurrents ayant des sites en HTML.

Il ne vous restera plus qu'à doubler votre site Flash par un site en HTML.

## 6.4 Résumé de ce chapitre

Une technique de programmation pure capable de flouer Google durablement ne peut pas exister.

De multiples maladresses en programmation peuvent expliquer de nombreuses raisons de recevoir des pénalités de Google. Si le codeur ne pense pas en permanence à respecter les *guidelines* de Google, fatalement, une ou des erreurs surviendront et sans aller jusqu'à une black liste, les pénalités de Google conduiront à du trafic perdu.

Google défend la qualité de son index et les maladroits n'ont qu'à louer du AdWords. C'est ce qu'on appelle faire d'une pierre deux coups.

Nous avons examiné quelques techniques d'optimisation de page et détaillé ce qui est à éviter et ce qui est à faire.

Vous avez découvert une des probables pistes futures pour améliorer le référencement : la personnalisation à la volée des images de vos pages, des images elles-mêmes, et pas seulement les balises `ALT` et `TITLE` ou leurs noms de fichier.

Pour finir, nous avons évoqué la technologie Flash face au référencement. Cette technologie est très prisée par les adeptes du beau design et du beau graphisme. Ils sont nombreux sur le Web. Flash est une source de cauchemars pour tout référenceur débutant. Néanmoins, on peut arriver à des résultats très performants quand on comprend comment s'y prendre, à condition de tenir compte du référencement avant tout codage, en se rappelant la règle suivante : ne pas créer un site 100 % Flash mais créer un site de pages HTML contenant chacune une animation Flash ; on arrivera à un excellent résultat en référencement.

# 7

# Google personalized search results

Google et la littérature anglo-saxonne utilisent deux groupes d'appellations très proches pour parler ou écrire sur la personnalisation des résultats de recherche : *personalized search results* ou *customized search results*. Le terme *results* est souvent omis car sous entendu.
Nous nommerons cette évolution *personalized search results* dans ce chapitre pour en simplifier la lecture. Mais attention, actuellement, au moment de la rédaction de ces lignes, le terme officiel en français n'est pas disponible.

## 7.1 Les règles du positionnement vont changer

Des consultants SEO américains prédisent la fin du métier de SEO, de référenceur, suite à la mise en place puis à la prévisible montée en puissance de cette refonte des algorithmes de positionnement dans Google.

Google parle de cette évolution depuis plusieurs années. Simple à énoncer, elle est très complexe à mettre en œuvre. Elle est mise en place depuis l'été 2008 aux USA. Son arrivée en Europe devrait se faire en 2009. Seuls Dieu et Google le savent bien sûr, mais la probabilité d'une mise en place pour 2009 est forte. En effet, la crise économique menaçant de s'étendre, l'historique économique montrant que ce sont ceux qui innovent fortement qui sortent avant tous les autres de la crise, et quelques autres raisons, font que, excepté un blocage technique majeur, *personalized search results* devrait arriver assez vite en Europe et ailleurs.

Cette évolution majeure se doit d'être commentée dans ce livre. Elle va sensiblement changer le métier de référenceur, elle va avoir un impact encore plus fort sur les codeurs de sites web.

Ce chapitre présente des réflexions et des estimations qui permettront au lecteur de mieux saisir la portée de cette évolution ainsi que ses conséquences sur les sites, leurs codages et les actions à entreprendre pour y faire face.

C'est la première fois que Google modifie de manière majeure le principe de tri et de positionnement des pages. Depuis plusieurs années, nous voyons Google mettre en place une mise à jour majeure de ses algorithmes de ranking, c'est-à-dire le principe selon lequel il trie les résultats des recherches dans les SERP.

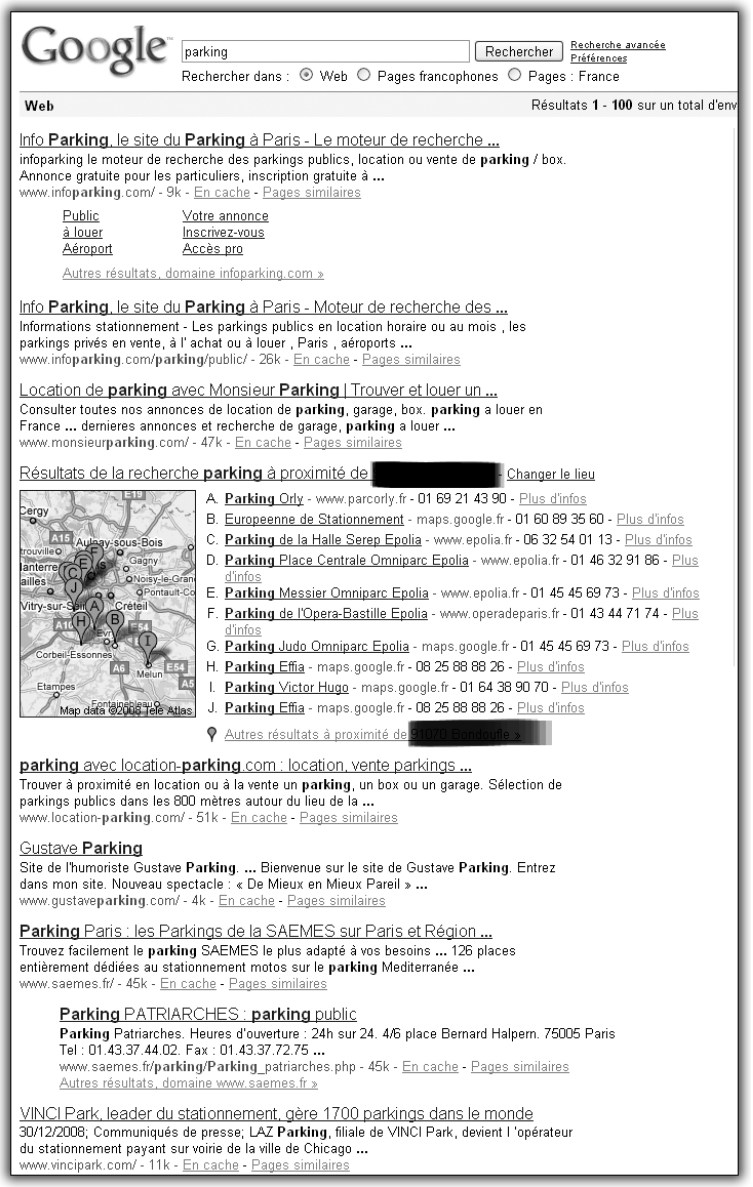
Depuis 1998, le positionnement était basé sur deux approches majeures et successives :

- le contenu et son référencement dans le site lui-même ;
- les backlinks internes ou externes.

Depuis de nombreuses années, Google enrichissait petit à petit ces deux approches par un *filtrage* nommé *personnalized search [results]* en anglais. L'objectif de ce filtrage est d'éliminer des SERP les propositions de site inutiles pour l'internaute. Cela explique pourquoi une requête effectuée à Bruxelles n'affiche pas toujours la même SERP que la même requête saisie à la même seconde depuis Paris.

Les premières règles de filtrage ont été simples ; elles concernaient essentiellement la géolocalisation.

Un exemple : Un internaute est en région parisienne et saisit le mot-clé *parking* depuis son navigateur habituel, Google Chrome, avec lequel il consulte son compte Google, mais il n'est pas connecté sur ce compte au moment de la requête. Il est censé être *anonyme*.




► Fig. 7.1 : SERP sur "parking" depuis Chrome et mon compte.

Le numéro IP a été interprété par Google. Il est manifestement en France, probablement en région parisienne. L'internaute s'est connecté ce matin à son compte Google mais s'est ensuite déconnecté. Les serveurs Google ont manifestement interrogé un cookie sur le navigateur Chrome. La soi-disant session anonyme est en fait parfaitement authentifiée. Google ne s'en cache même pas sur son blog officiel en sous-entendant que, connecté ou non, Google tentera d'identifier tout internaute utilisant ses services (voir un peu plus loin et dans le chapitre annexe webographie les URL à consulter sur ce point).

Le résultat est personnalisé :

- De nombreux parking en région parisienne sont proposés.
- Un *Google map locator* offre des parkings, le plus près possible de la ville du domicile de l'internaute. Clairement, Google a utilisé un cookie et les informations personnelles du compte, en l'occurrence son adresse, pour le localiser aussi précisément.
- Hors la zone géographique, Google n'a aucune information pour personnaliser davantage cette SERP. L'humoriste Gustave Parking figure dans les premiers résultats ainsi que le site de la société Noparking qui commercialise un logiciel. Les algorithmes de Google proposent un mix de résultats couvrant les différents thèmes dépendant de ce mot-clé : essentiellement des parkings pour véhicules, un humoriste, un éditeur de logiciels.

La même requête depuis Firefox et aucun historique ou cookie sur mon compte Google dans ce navigateur modifient la SERP.

Le résultat est plus faiblement personnalisé :

- Comme auparavant, de nombreux parkings en région parisienne sont proposés.
- Hors ma zone géographique, Google n'a là aussi aucune information pour personnaliser cette SERP. L'humoriste Gustave Parking, le site de la société Noparking (logiciel en gestion du temps) et Wikipedia figurent dans les premiers résultats. Plus loin, la SERP affiche des parking en France mais hors région parisienne : Marseille et quelques autres grandes villes.
- Il n'y a aucun Google *map locator* proposant des parkings le plus près possible de la ville correspondant au domicile de l'internaute.

ⓘ Les résultats de recherche personnalisés en 2009

Ce mode de recherche est mis en œuvre par défaut ; il concerne actuellement uniquement une personnalisation géographique de l'information. À partir de 2009, cela changera. Google exploitera davantage d'informations personnelles afin de personnaliser les SERP de chaque internaute.


Google

parking — Rechercher — Recherche avancée — Préférences

Rechercher dans : Web — Pages francophones — Pages : France

Web — Résultats 1 - 10

Info **Parking**, le site du **Parking** à Paris - Le moteur de recherche ...
infoparking le moteur de recherche des parkings publics, location ou vente de **parking** / box. Annonce gratuite pour les particuliers, inscription gratuite à ...
www.info**parking**.com/ - 9k - En cache - Pages similaires

Public à louer Aéroport — Votre annonce Inscrivez-vous Accès pro

Autres résultats, domaine infoparking.com »

Info **Parking**, le site du **Parking** à Paris - Moteur de recherche des ...
Informations stationnement - Les parkings publics en location horaire ou au mois , les parkings privés en vente, à l' achat ou à louer , Paris , aéroports ...
www.info**parking**.com/**parking**/public/ - 26k - En cache - Pages similaires

Location de **parking** avec Monsieur **Parking** | Trouver et louer un ...
Consulter toutes nos annonces de location de **parking**, garage, box. **parking** à louer en France ... dernieres annonces et recherche de garage, **parking** a louer ...
www.monsieur**parking**.com/ - 47k - En cache - Pages similaires

**parking** avec location-**parking**.com : location, vente parkings ...
Trouver à proximité en location ou à la vente un **parking**, un box ou un garage. Sélection de parkings publics dans les 800 mètres autour du lieu de la ...
www.location-**parking**.com/ - 51k - En cache - Pages similaires

Gustave **Parking**
Site de l'humoriste Gustave **Parking**. ... Bienvenue sur le site de Gustave **Parking**. Entrez dans mon site. Nouveau spectacle : « De Mieux en Mieux Pareil » ...
www.gustave**parking**.com/ - 4k - En cache - Pages similaires

**Parking** Paris : les Parkings de la SAEMES sur Paris et Région ...
Trouvez facilement le **parking** SAEMES le plus adapté à vos besoins ... 126 places entièrement dédiées au stationnement motos sur le **parking** Mediterranée ...
www.saemes.fr/ - 45k - En cache - Pages similaires

**Parking** PATRIARCHES : **parking** public
**Parking** Patriarches. Heures d'ouverture : 24h sur 24. 4/6 place Bernard Halpern. 75005 Paris Tel : 01.43.37.44.02. Fax : 01.43.37.72.75 ...
www.saemes.fr/**parking**/**Parking**_patriarches.php - 45k - En cache - Pages similaires
Autres résultats, domaine www.saemes.fr »

VINCI Park, leader du stationnement, gère 1700 parkings dans le monde
30/12/2008; Communiqués de presse; LAZ **Parking**, filiale de VINCI Park, devient l 'opérateur du stationnement payant sur voirie de la ville de Chicago ...
www.vincipark.com/ - 11k - En cache - Pages similaires

France Inter - **Parking** de nuit
En sortiront des courriers/courriels ou messages audio d'auditeurs ayant quelques idées pertinentes pour améliorer le confort du **Parking**, ...
radiofrance.fr/franceinter/em/**parking**denuit - 27k - En cache - Pages similaires

**Parking** Paris - Informations et réservation de places de parkings ...
Réservation de places de parkings à Paris. **Parking** Paris. **Parking** à Paris. Informations parkings. Paris **Parking**. Paris Expo. **Parking** Stade de France.
www.**parking**sdeparis.com/ - 51k - En cache - Pages similaires

**Parking** - Wikipédia
Un **parking**, ou au Canada terrain de stationnement ou simplement stationnement, est un espace ou un bâtiment spécifiquement aménagé pour le stationnement des ...
fr.wikipedia.org/wiki/**Parking** - 49k - En cache - Pages similaires

•▷▷ Garage - **Parking** - Box ANNONCES GRATUITES
I➔➔➔➔ Vente Achat ou Location de votre Garage - **Parking** près de chez vous . VIVASTREET.fr N°1 de l'ANNONCE 100% GRATUITE.
immobilier-**parking**.vivastreet.fr/ - 155k - En cache - Pages similaires

Bienvenue sur le site e-chèque **Parking** de l'aéroport Marseille ...
Site d'achat en ligne : titre de stationnement valable sur le parc Chèque **parking** de l'Aéroport Marseille Provence.
www.cheque**parking**.com/ - 10k - En cache - Pages similaires

Annuaire Automobile LeParking
Annuaire automobile, Le **Parking** du net regroupe l'essentiel des sites automobiles francophones de qualités. Le classement des sites amateurs et ...
le**parking**.autofolie.org/ - 26k - En cache - Pages similaires

Logiciel de gestion du temps et des activités / openTIME
OPENTIME est un logiciel intranet de gestion de temps et de gestion d'activités aidant au pilotage de l'entreprise.
www.no**parking**.net/ - 6k - En cache - Pages similaires

► Fig. 7.2 : SERP sur "parking" depuis Firefox, en anonyme

Les autres applications de la géolocalisation concernent essentiellement la localisation dans le pays ou non des sites. Les sites en *.fr* ou en *.com* hébergés avec une adresse IP française seront privilégiés par rapport aux autres sites. L'objectif pour Google est toujours d'éviter de noyer l'internaute avec des informations inutiles.

Une nouvelle version prenant en compte l'historique de l'internaute était en préparation depuis plusieurs années.

Elle est apparue aux États-Unis en 2008 et devrait arriver en Europe en 2009.

## 7.2 Intérêts des résultats de recherche personnalisés

Une expression clé peut être interprétée avec plusieurs définitions distinctes. Exemple : le mot-clé `lampe`.

Que cherche l'internaute ? Une lampe pour décorer du style de la lampe à huile d'Aladin ? une ampoule ? Une lampe marocaine ? une applique murale ?

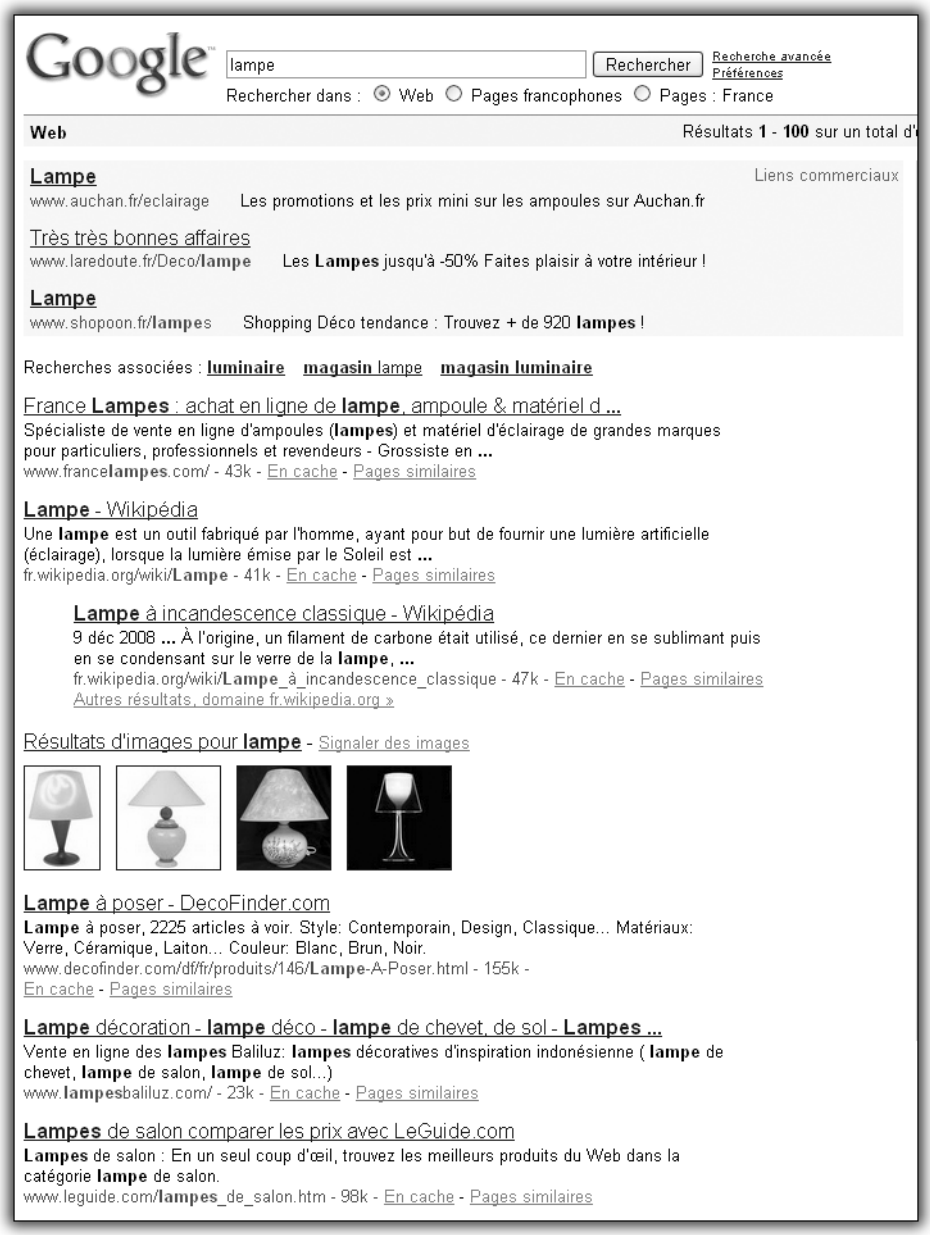
Sans personnalisation poussée, Google pourra juste donner la priorité d'affichage aux sites du même pays évoquant abondamment les *lampes*.

Si cet internaute a déjà étudié d'autres sites comme des sites de décorations orientales, pour rester dans l'exemple, Google va avoir accès aux mots-clés des dernières requêtes. L'internaute a saisi `décoration orientale`. Google aura accès aux divers sites web consultés par notre internaute. Par définition, Google connaît la thématique, les mots-clés des pages de ces sites. Avec *Personalized Search Results* activé, Google pourra compléter par lui-même la requête de l'internaute peu bavard, lancer `lampe décoration orientale` et lui proposer les pages les mieux positionnées sur cette expression clé.

À titre de comparaison, entre une démarche étudiant l'historique d'un internaute et une démarche analysant la moyenne des historiques des internautes sur la même requête, on obtient ceci actuellement :

En haut de page, Google propose trois choix prêts à être cliqués :

- *Luminaire*. De nombreux autres internautes ont saisi `lampe` et ont cliqué sur des site de *luminaires*.
- *Magasin lampe*. De nombreux internautes ont saisi la requête `magasin lampe` ou ont cliqué sur des sites évoquant les termes `magasin` et `lampe`.
- *Magasin luminaire*. Des internautes ont saisi `lampe` et ont cliqué sur des site de *magasin luminaire.*




► Fig. 7.3 : SERP Google sur la requête lampe sans personnalisation poussée

Le système de thésaurus mis à jour par les comportements des internautes met en évidence ces trois expressions clés en rapport avec le mot-clé `lampe`. La SERP les propose en idées de requêtes à notre internaute peu bavard dans sa requête.

On perçoit tout de suite la différence de qualité. D'un côté, `lampe décoration orientale` avec une forte probabilité d'avoir touché juste, de l'autre des résultats mixant de nombreux éléments (lampe pour bibliothèque, annuaire de luminaires, luminothérapie, décoration…), faute de comprendre ce que désire l'internaute.

Personalized Search Results promet un avantage compétitif à Google face à ses concurrents disposant d'un public bien plus faible et donc d'un thésaurus moins complet, moins fiable.

## 7.3 Arrivée de Personalized Search Results

Google évoque cette piste d'évolution depuis plusieurs années. Pour faire simple, fixons cette date à 2005. Elle correspond à la publication de plusieurs communiqués par Google sur ce thème.

Au fil des années, diverses communications et la mise en service d'un peu de "comportemental" sur AdSense/AdWords a permis à Google d'acquérir plus de savoir-faire et de données sur le sujet.

Nous verrons un peu plus loin que le modèle technologique Google repose sur les éléments suivants :

- Des algorithmes "adroits" effectuent des calculs simples.
- Les calculs simples sont réalisés sur d'immenses volumes de données.
- Ces calculs utilisent un thésaurus mis à jour par les internautes eux-mêmes via leurs usages de Google.

Personalized Search a besoin d'un thésaurus parfaitement au point, rôdé et possédant des mises à jour fiables et continues. En effet, toute logique calculant un positionnement par rapport :

- à des mots-clés saisis dans Google ;
- à des mots supposés clés par Google dans des pages cliquées par un même internaute ;

aura besoin d'un thésaurus opérationnel. Autrement, comment évaluer efficacement ce que recherche un internaute en fonction de son historique ? Rappelez-vous l'exemple étudié précédemment, sur `lampe`.

## 7.4 Les informations personnelles utilisables

Nous laissons de nombreux historiques dans notre sillage de navigation sur le Web :

- historique sur l'utilisation de Google ;
- fichier log des possesseurs de cookies, notamment ceux des régies publicitaires ;
- Google Chrome ;
- Google toolbar.

Tout ce qui a été cliqué depuis Google, toutes les requêtes et les clics afférents sont mémorisés par Google par rapport à un cookie ou à un numéro IP.

Un cookie posé par une régie publicitaire peut être lu uniquement par cette régie, auteur de la pose du cookie. Hors une même régie peut avoir des milliers, voire des millions de sites comme clients. Cette régie peut donc vous suivre dans votre navigation. Google a racheté la plus grosse régie publicitaire du secteur : *doubleclic*. Cela lui laisse l'accès aux fichiers de cette régie. En rapprochant les informations avec comme clé de jointure le numéro IP et la date, Google a la possibilité d'exploiter une masse de données personnelles gigantesques. Nous verrons un peu plus loin quelques intérêts stratégiques de Google liés à une telle approche.

### Google affiche des limites

Selon le blog officiel de Google, **http://googleblog.blogspot.com/2008/07/more-transparency-in-customized-search.html**, le moteur se limiterait à des informations exhaustivement listées et informerait en temps réel l'internaute de l'usage fait de ces données sur les SERP fournies par Google sur toute requête.

En théorie, cela semble bien comme éthique. En pratique en revanche, les SEO et autres web marketing vont avoir du souci à se faire.

⚠ Fin du ranking et du positionnement classique ?

La personnalisation des résultats de recherche signifie que de plus en plus souvent, les SERP seront différentes d'un internaute à un autre.

Cela signifie-t-il la mort du référencement tel que nous le connaissons actuellement ? Non, mais le métier va être bouleversé et tout ce que nous venons de voir restera valide. De nouvelles techniques seront ajoutées pour réussir le positionnement d'un site. Nous les évoquerons un peu plus loin dans le présent chapitre.

En effet, la quasi totalité des Internautes ne gèrent pas les différentes options proposées dans leurs navigateurs ou dans les préférences des outils Google.

95 % et plus des internautes ne vont pas gérer leurs historiques et l'accès à leurs données personnelles par Google. Donc celui-ci y accédera sans problème.

Google s'engage à utiliser uniquement les données :

- Géographiques fournies par votre adresse IP.
- Issues de vos récentes recherches, les mots-clés utilisés dans vos requêtes. Cette définition proposée par Google est intéressante. Que signifie "récent" ? Légalement, ce serait 18 mois maximum. Ce filtrage est actif que vous soyez ou non connecté à votre compte Google. Traduction : Google authentifie personnellement chaque internaute via cookie. Il vous authentifie à chaque passage sur un des ses services. Peut-être garde-t-il en réserve la possibilité d'identifier un internaute via son adresse IP, mais cette méthode est théoriquement peu fiable dans 20 % des cas, voire plus. Dans la pratique, avec le nombre de collecteurs de données de Google (Toolbar, doubleclic, Chrome…), l'adresse IP est une méthode d'authentification qui peut être fiabilisée grâce à des recoupements.
- Issues de l'historique web quand vous êtes connecté sur votre compte Google, un très joli texte indique que vous êtes propriétaire de cette historique et que vous pouvez le supprimer ou l'amender à votre convenance.
- Il manque un détail dans le blog officiel de Google : les données personnelles saisies lors de l'ouverture d'un compte Google sont aussi utilisées.

Dans des interviews de 2007 (**http://searchengineland.com/googles-matt-cutts-on-personalization-and-the-future-of-seo-10649**), Google indique ne pas vouloir insérer plus de deux résultats issus de la personnalisation des résultats de recherche sur 10 affichés dans une SERP classique. Google indique aussi laisser intact les premiers de chaque position sur une expression clé. Des interviews de la même époque précisent que Google commencera en douceur pour ne pas heurter ses usagers. Mais ces même interviews indiquent aussi que plus Google aura confiance en cette technologie, plus elle sera utilisée. Les limites d'usage indiquées précédemment ne tiendront donc clairement pas longtemps.

Une limite naturelle existe : ne pas sanctionner les sites qui présentent un vrai plus pour les internautes. Chaque fois que Google met au point une technique anti-fraude, une question se pose : Cela ne va-t-il pas sanctionner des contenus légitimes ? En effet, si trop de contenus légitimes sont exclus par une technique, anti-fraude, anti-paidlink, personnalisation des résultats de recherche…, cela équivaut pour Google à se tirer une balle dans le pied et à rendre possible le retour d'un autre moteur de recherche.

## Résumons-nous sur Personalized search resultats

Tout ce qui est tracé par Google de l'utilisation de son moteur d'un internaute précis sera utilisé pour personnaliser les SERP qui lui sont destinées.

Quatre groupes d'information sont utilisés : géographique, personnelle liée à un compte Google, historiques web et historique de l'usage du moteur (les enchaînements de requêtes).

Ces informations seront utilisées que l'internaute soit connecté ou non à un compte Google. La majorité des internautes laisseront faire.

Les conditions d'utilisation par Google de ces informations sont officiellement transparentes. On a vu qu'il manquait un groupe d'information pris en compte dans les calculs de personnalisation sur les résultats de recherche, les données personnelles liées au compte, dès la publication du post sur le blog officiel de Google. Aucune mise à jour n'a été effectuée depuis juillet 2008 sur ce sujet.

Étant donné que très peu d'internautes consultent et réagissent aux conditions d'utilisation de Google, ceux-ci seront enrôlés d'office dans la personnalisation des résultats de recherche ; c'est le mode par défaut, il y a peu de chances que quelqu'un fasse un *opt-out* pour bloquer ce service. D'ailleurs, pourquoi le ferait-il ? La personnalisation est un vrai avantage lors des recherches.

Le référenceur devra prendre en compte les faits suivants :

- *Personalized search resultats* va changer les SERP de nombreux internautes, faisant voler en éclats une partie de travail du référencement et, au minimum, la manière de mesurer le positionnement.
- Google ajoutera au fil du temps de nouvelles données prises en compte. Il officialisera cela ou non. Google peut par exemple prendre en compte les données issues des Google toolbar ou de Chrome en l'annonçant ou non.
- L'immense majorité des internautes laisseront faire.
- La personnalisation des résultats de recherche va bouleverser le travail de référencement ; cela aura donc un impact sur le travail du codeur. Nous l'évoquerons un peu plus loin.
- Google améliorera les algorithmes liés à *Personalized search resultats* mis en place et de plus en plus de SERP s'éloigneront du modèle de ranking habituel.

## 7.5 Pourquoi Personalized search results ?

Plusieurs raisons stratégiques conduisent Google à personnaliser les résultats des SERP.

### L'amélioration des SERP

Mieux les SERP sont adaptées, voire personnalisées, aux attentes des internautes, plus ces derniers sont fidélisés aux outils et au nom de Google. Les avantages sont multiples et évidents :

- plus de revenus de AdWords ;
- plus de partenaires AdSense et donc de revenus liés ;
- plus de revenus publicitaires liés à d'autres outils Google ;
- affaiblissement de la concurrence directe : Microsoft et les autres moteurs.

### Freiner les référenceurs

Tout site web ayant un référenceur compétent diminue les revenus de Google. Des pages bien positionnées naturellement n'incitent guère un site web à souscrire à des dépenses AdWords.

Les SEO sont jaugés sur leurs performances visibles. Actuellement, ils sont évalués sur le ranking des mots-clés des pages du site référencé. Avec une personnalisation des résultats de recherche, ce modèle commence à voler en éclats.

Il va être nécessaire de redéfinir une méthode de mesure de performance du référenceur :

- Le navigateur ou l'outil de mesure devront avoir éliminé tous les cookies de toute nature.
- Google sait que les adresses IP ne sont pas fiables. Renouveler son adresse IP pour ne pas être suivi par Google n'est nullement obligatoire pour fiabiliser une mesure de positionnement et éviter un filtre personnel autre que géographique.
- Il ne faudra pas être connecté sur un compte Google bien sûr et si possible utiliser Firefox sans Googlebar si vous êtes un professionnel en SEO. Autrement, cette précaution est inutile.
- Il faudra songer à bloquer la personnalisation de la recherche dans Google/ Préférences.

Il sera nécessaire de redéfinir le travail du référenceur et de redéfinir la mesure de la qualité de son travail. Nous l'évoquerons plus loin.

## La crise économique

L'économie est durement touchée par la crise (*subprime*, Madoff et autres excentricités de la finance américaine). Face à une possible baisse de revenus, la stratégie commande d'innover pour conserver ses revenus. Deux gisements de bénéfices sont ainsi visés par Google : laminer encore plus la concurrence et repousser les nuées de moustiques référenceurs freinant les revenus AdWords.

## Un avantage concurrentiel important

*Personalized search results* exige pour sa réussite :

- Une puissance de calcul gigantesque : Google dispose de centaines de milliers de serveurs.
- Une algorithmique bien travaillée et optimisée : Google a déjà dû relever de tels challenges et a acquis un indéniable savoir-faire face à de telles problématiques. Nous y reviendrons plus loin.
- De disposer de vastes historiques pour chaque internaute ou presque. C'est le cas. Qui n'a pas un historique associé à un cookie chez Google ? Gmail, Google Map, presque 200 services Google de qualité sont à disposition des internautes de la planète, en des dizaines de langues, le tout gratuitement.

De fait, seul Google dispose des ressources diverses nécessaires à lancer un service aussi lourd. Cette mise en place améliorera son service, augmentera son avance et renforcera un cercle vertueux : plus il y a d'utilisateurs, plus on a de données, mieux on peut personnaliser et plus Google gagne en revenus.

Google dispose également de réserves en informations et historiques pour poursuivre son avancée sur *Personalized search results*. Ces réserves ne sont pas officiellement utilisées ou pas encore : Google toolbar, Chrome, Doubleclic…

Google est une société qui a pour habitude de prendre son temps. Son avance actuelle lui permet de longuement tester et préparer la personnalisation des résultats de recherche, depuis 3 ans environ. Cela lui donne le confort d'une mise en service progressive sans risque.

## 7.6 Les limites imposées à Google

Malgré sa puissance, Google doit agir prudemment. Voici les obstacles à surmonter et ils ne sont pas des moindres.

### Le système doit s'auto éduquer

Google ne dispose pas des ressources humaines pour entretenir un thésaurus. Celui-ci doit auto apprendre.

Google a mis au point un entretien des thésaurus par les internautes eux-mêmes, via leurs propres clics.

Exemple : *cars* est rapidement devenu le synonyme d'automobile à la sortie du dessin animé éponyme de Pixar/Disney.

Ce sont les clics des internautes, dans les requêtes, qui ont conduit à cette mise à jour dans le thésaurus.

De même pour la détection des mots composés jointés dans un seul terme, le thésaurus Google apprend avec les clics des internautes.


AUTOREFLEX : **Voiture** occasion, achat **voiture**, vente **voiture** ...
**voiture** occasion : Plus de 140 545 vente **voiture** par régions pour l'achat **voiture** ou la vente **voiture** occasion, des annonces **auto** et des vente **auto**.
www.**auto**reflex.com/ - 146k - En cache - Pages similaires

Actualité **automobile**, essais nouveaux modèles **auto** : Le Quotidien **Auto**
Avec Le Quotidien **Auto**, retrouvez toute l'actualité **automobile** en continu sur le web, et découvrez les essais des nouveaux modèles **auto** proposés par les ...
**automobile**.nouvelobs.com/ - Pages similaires

**AUTO** PLUS Essai **voiture**, vente **auto** occasion, cote **auto**, achat ...
8 jan 2009 ... Conseils pour l'achat de **voiture** d'occasion ou **auto** neuve, pres de 8000 essais et avis **auto**, la cote **auto** mais aussi l'actualite **automobile**, ...
www.**auto**plus.fr/ - 56k - En cache - Pages similaires

**Voiture** occasion avec 321 **auto** - Achat Vente annonce **automobile** ...
**Voiture** occasion : Acheter et vendre annonce d'occasion, Achat / vente de particulier à particulier, consultation annonce gratuite, petites annonces **auto** ...
www.321**auto**.com/ - 66k - En cache - Pages similaires

**Voiture** occasion : annonce **auto** occasion gratuite , vente , Achat ...
**Voiture** Occasion **AUTO**-SELECTION, achat, vente **voiture** occasion par annonces : Acheter **voiture** occasion , **auto** neuve ou en faire la vente ; Gratuit: annonce ...
www.**auto**-selection.com/ - 44k - En cache - Pages similaires

L'**Auto** Journal - Magazine **automobile** - Actualité - **Voitures** ...
L'**Auto** Journal, le magazine de l'**automobile**. Toute l'actualité **automobile**, essais photos, vidéos mais également les prix du neuf, la cote de l'occasion ou ...
www.**auto**journal.fr/ - 48k - En cache - Pages similaires

► Fig. 7.4 : Détection par Google lors d'une requête sur ~auto : AUTOplus, AUTOreflex

Les internautes via une requête incluant "auto" et cliquant sur de nombreux sites ayant auto accolé un autre mot dans leurs noms de domaines éduquent Google sur l'existence du mot auto, synonyme de automobile.

De même, le thésaurus utilisé par la personnalisation des résultats de recherche devrait automatiquement s'entretenir. Les comportements des internautes et leurs éventuelles corrections vont apprendre au système où sont les erreurs et il pourra ainsi automatiquement les corriger.

## Illustration

Un internaute saisit une requête, `lampe`, après avoir navigué sur des sites web de *décoration de maison*. La personnalisation lui propose une SERP avec majoritairement des pages/URL sur le thème "*lampe décoration*". Il ne clique sur rien et saisit une autre requête : `lampe à pétrole`. Dans la SERP renvoyée, il clique sur une page ayant pour thème "*lampe à pétrole pour la décoration d'une maison*". Si plusieurs internautes font cela, le thésaurus de Google prendra en compte que `lampe décoration [maison || intérieure]` doit inclure aussi des URL lampe à pétrole. Il pourrait en faire figurer une ou deux dans la SERP type à renvoyer vers toute requête. Le thésaurus a été automatiquement mis à jour.

### ⚠ Qui fait quoi ? L'organisation probable de Google

Google développe et déploie les algorithmes génériques de personnalisation des résultats de recherche. Ces algorithmes utilisent les données organisées par un thésaurus.

L'usage des utilisateurs des différents services Google éduque le thésaurus sans intervention humaine. Le thésaurus utilisé par les multiples services Google est issu du comportement réel de la majorité des internautes. Nous tenons une des clés qui permettront de réaliser le référencement sur le Google de demain.

Les historiques d'un internaute contiennent :

- Les mots-clés de ses requêtes : ils dégagent des tendances.
- Son historique web, c'est-à-dire les URL sur lesquelles il a cliqué depuis les SERP de Google.

Hors chaque URL correspond à une page, un objet ayant des propriétés : `TITLE`, titres `H1`, backlinks, etc. Ces propriétés ont des contenus en mots-clés et en syntaxe. Elles dégagent des thématiques : recette de cuisine, décoration d'intérieur…, chacune associée à une orientation plus précise : recette de cuisine sur les gâteaux au chocolat moelleux, luminaire pour la décoration d'intérieur, etc.

Le classement en thématiques, en orientations plus précises, est géré par le thésaurus. Celui-ci crée automatiquement de nouvelles catégories,sous-catégories en fonction du comportement des usagers de Google comme nous l'avons déjà vu.

SUITE

La personnalisation des résultats de recherche va analyser la requête de l'internaute vis-à-vis de ses historiques qui dégagent thématiques et orientations selon le classement du thésaurus de Google. Les SERP contiendront des URL positionnées dans l'index sur le résultat du filtrage sélectionné par la personnalisation : géolocalisation, thématiques, orientations, mots-clés calculés pour correspondre aux attentes de l'internaute.

## Ne pas froisser les utilisateurs

Des centaines de millions d'usagers quotidiens ont l'habitude d'utiliser Google et d'avoir des SERP rapides et à peu près adaptées à leurs besoins.

Il est impossible de froisser ceux qui ont fait Google. La mise en service de nouveaux algorithmes passe toujours par une phase de prise en compte progressive pour ne pas perturber les usagers.

Le changement de positionnement dans les SERP ne sera pas trop révolutionnaire au début. Google a prévu que l'utilisateur puisse débrayer les filtres aisément afin de permettre aux usagers de revenir à tout moment sur l'ancien fonctionnement. Bref, tout est fait pour assurer la jonction entre les deux fonctionnements du moteur : avec ou sans personnalisation des résultats de recherche.

## Ne pas planter le service

Les algorithmes ne doivent pas ralentir ou planter le système. Comme les algorithmes de détection de duplicate contents, il seront "simples". L'objectif semble contradictoire. D'un côté il faut avoir un système de personnalisation performant, de l'autre cela doit être tellement rapide que l'internaute ne perçoit pas de temps d'attente sensible.

Ce n'est pas si contradictoire. Le fonctionnement actuel voit un SEO jouer avec différentes limites face à Google sur des pages accessibles aux deux parties en présence. Pour tenter de ne pas se faire flouer par le référenceur, Google doit dépenser des ressources informatiques afin de traquer triches et paidlinks.

Avec la personnalisation, Google a accès à un important réservoir d'informations inaccessibles à jamais au référenceur : l'historique web et l'historique requête de l'internaute. Il n'y a pas de risque de triche. Il sera relativement aisé d'appliquer des algorithme simples et pas trop gourmands en ressources permettant de cibler, avec un bon taux de succès, ce que cherche l'internaute.

## 7.7 Baisse ou hausse du trafic

Pas de panique : le trafic diminuera des internautes mal ciblés. La personnalisation en écartera une partie. Il pourrait y avoir d'un autre côté une augmentation de trafic avec un meilleur positionnement du site sur des expressions clés pas assez usitées.

Exemple : Reprenons notre exemple de recherche sur *lampe*. Rappelez-vous, la personnalisation a permis à Google de déduire que cet internaute cherchait en fait une *lampe décoration orientale*.

Un site commercialisant de la décoration marocaine, avec un faible budget netlinking, ayant une landing page dédiées aux lampes décoratives, ayant travaillé les différentes déclinaisons des expressions clés autour des lampes commercialisées sera bien placé sur une telle requête : `lampe décoration orientale`. Ce même site est très loin sur le mot-clé `lampe` employé seul. La possibilité de récupérer en trafic de nouveaux internautes mieux ciblés est importante, dans cet exemple.

**Avantage des recherches personnalisées**

Un site ayant du contenu réel, bien organisé, doté de landing pages, d'une gestion dynamique et centralisée de sa stratégie de référencement, a tout à gagner de cette évolution majeure de Google.

En effet, le travail déjà décrit dans cet ouvrage et dans le *Guide Complet du référencement sur Google* (Micro Application) aura une effet démultiplié sur le trafic quand Google mettra en service les résultats de recherches personnalisés.

Cette modification majeure dans les algorithmes de SERP peut avoir des effets très bénéfiques en trafic sur un site web :

- dont le contenu et la valeur ajoutée sont réels ;
- qui présente pas ou peu de duplicate contents ;
- qui a une organisation apparente très hiérarchisée pour Google ;
- qui possède une stratégie de référencement centralisée et ciblée (landing pages, expression clé) ;
- qui met en œuvre une politique de référencement concentrée sur les mots/expressions clés et leurs multiples déclinaisons ;
- qui possède une stratégie de netlinking.

En effet, avoir l'expression clé, celle calculée par la personnalisation du résultat de recherche Google, dans la landing page c'est bien, mais faut-il encore être bien positionné dessus. Hors le nombre de déclinaisons référençables dans une page web n'a pas changé :

approximativement 5 mots-clés centraux, une dizaine maximum autour. Ensuite, il n'y a plus de place en *URL rewriting* et en *title* de page.

Il faudra donc identifier, trier, classer et cibler hiérarchiquement les multiples déclinaisons et variantes d'une expression clé sur sa landing page :

- `lampe décoration orientale` ;
- `lampe décoration marocaine` ;
- `lampe de salon` ;
- `lampe en cuir de chèvre` ;
- `lampe métallique et en cuir`, etc.

Une landing page réussie sur cette expression clé, `lampe décoration orientale`, devra donc détailler les endroits où on peut la mettre (`salon`, etc.), sa composition (`cuir de chèvre` et `métal`), son origine (`marocaine`, `orientale`), sa fonction (`décoration`, `décorative`, `éclairage`). Cela nous montre une landing page dotée de textes, de titres, de visuels si on désire augmenter le trafic.

Ce qui était vrai, avant la mise en service de la personnalisation des résultats de recherches sur Google, est encore plus d'actualité après.

## Les constantes

Un site sera toujours composé de contenu, de backlinks internes et externes. Cela sera toujours pris en compte. En effet, Google doit :

- fournir des SERP aux internautes sans historique ;
- disposer d'une base d'indexation sur laquelle appliquer des filtres de personnalisation des résultats de recherche.

Un site mal conçu aura des statistiques potentiellement incompréhensibles et difficiles à interpréter. La personnalisation pourra avoir un effet dévastateur sur le trafic d'un tel site.

La qualité en structure (navigation claire et fortement hiérarchisée pour les bots de Google), la qualité en référencement (pas de duplicate content, critères bien gérés, accessibilités technique et sémantique), la qualité et le volume des informations (texte, vidéo, images, animations flash , bref le *rich media*), la qualité de la mise en page (des titres, des sous-titres, un chapeau, des paragraphes clairement délimités, des phrases correctement construites et formées, etc.) sont actuellement des critères exigés pour réussir son positionnement via une campagne de netlinking.

Ce sera accentué par l'évolution de Google vers des SERP personnalisées selon l'historique de l'internaute :

- Google pénalise indirectement les contenus mal présentés. Une thématique mal perçue noie la page dans le fin fond de l'index. (Mais on peut avoir du Page rank malgré tout si on a des backlinks externes.)
- Il est toujours plus facile d'interpréter le web analytics d'un site clairement organisé. Un référenceur pourra mieux travailler.

## 7.8 Référenceur 2.0

Le métier va fortement évoluer. C'est évident. Dans un premier temps les référenceurs, codeurs ou non, vont devoir élargir la palette de leurs compétences. Détaillons quelles compétences et pourquoi.

Outre le travail habituel d'audit de mots-clés, le travail sur les critères des pages ou le gabarit, de programmation "de base" ou de conseil en architecture de site, de nouveaux savoir-faire vont devoir être ajoutés ou complétés.

### Mieux comprendre la logique de personnalisation

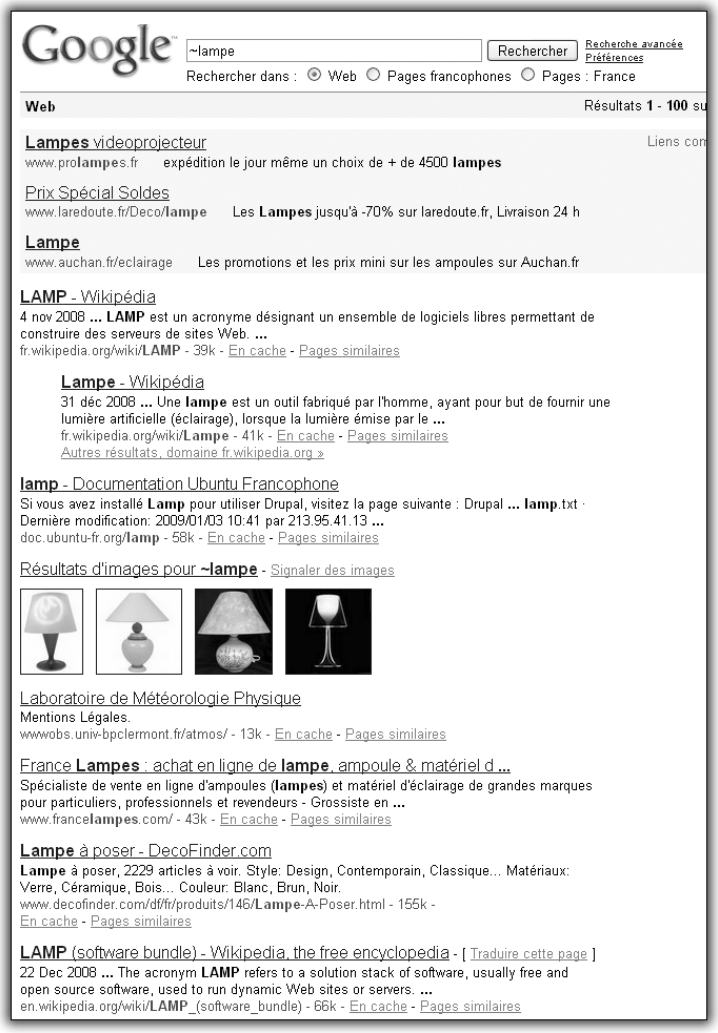
Il va falloir trouver la logique de base des algorithmes de personnalisation. Les SEO ont plus ou moins bien reconstitué le fonctionnement actuel de Google afin de savoir comment référencer efficacement un site et ses pages sur ses expressions clés. Toutes les données nécessaires pour mieux cerner la logique de Google étaient accessibles : les pages à optimiser, les pages mieux placées des concurrents, différents outils Google, etc. Là, le référenceur n'aura pas accès aux historiques des internautes. Il va donc falloir s'y prendre autrement.

On aura d'un côté le moteur manipulant des données simples. Ces manipulations sont en fait l'exécution d'une logique, de calculs donc, destinés à réaliser la personnalisation des résultats de recherche, donc de positionner des pages en liste ordonnée dans des SERP. De l'autre côté, ces données simples sont issues de deux sources :

- Un thésaurus construit par l'usage des internautes eux-mêmes du moteur de recherche. Ce thésaurus fournit les méthodes de classement, bref tout ce qu'il faut pour *comprendre* et classer le contenu des pages des sites d'un côté et les pages des sites web visités depuis des SERP de l'autre.
- Les historiques propres à chaque internaute ayant un compte Google et ayant été authentifié par Google via un cookie.

Pour simplifier, nous partirons du principe que tout internaute a un compte Google (GMail, etc.) et qu'il aura donc un historique. À terme, vu le poids grandissant de Google, vu son statut juridique et ses méthodes, une majorité d'internautes auront sous une forme ou une autre un compte Google ou équivalent, avec lesdits historiques associés.

L'organisation des données du thésaurus est partiellement consultable via les commandes adéquates dans Google. Par exemple : ~ (tilde) qui permet de lister les synonymes d'un mot selon ce thésaurus auto éduqué de Google.




► Fig. 7.5 : SERP de la requête tilde ~lampe sur Google.

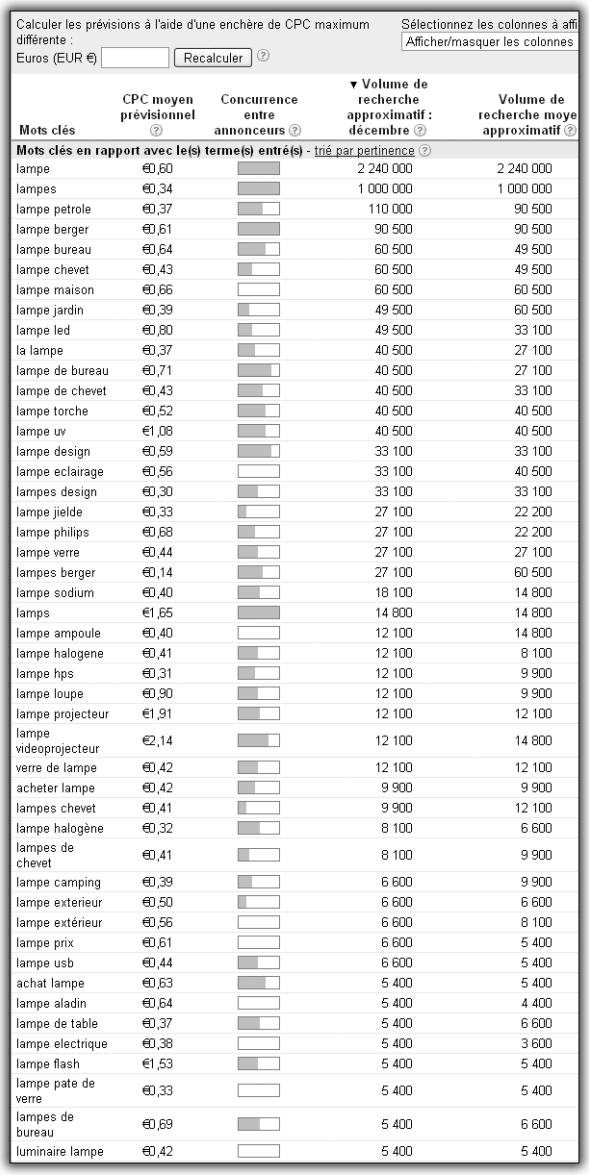
Selon le thésaurus de Google, lampe a pour synonyme l'acronyme LAMP (Linux Apache MySQL PHP). Dans notre cas, cela n'offre aucun intérêt. On notera l'usage massif du mot `lampe` pour parler d'*ampoule* dans de nombreux sites web. Le thésaurus de Google n'a pour le moment pas pris en compte ce synonyme potentiel *lampe – ampoule*.


► Fig. 7.6 : SERP de la requête tilde ~lampe décoration sur Google.

Plus intéressant, `~lampe décoration` saisi en requête dans Google nous apprend que *deco* est un synonyme de *décoration*. On peut ainsi diminuer la longueur des URL et le title de la page. En rédactionnel, on pourra mixer les deux termes.

Via ces exemples, nous venons de voir quelques résultats de quelques règles régissant le thésaurus de Google.

D'autres outils Google permettent de détecter d'autres formes d'associations et règles que ce thésaurus réalise : l'outil de recherche de mots de AdWords, par exemple : **https://adwords.google.com/select/KeywordTool**.



Calculer les prévisions à l'aide d'une enchère de CPC maximum différente :
Euros (EUR €) Recalculer

Sélectionnez les colonnes à affi
Afficher/masquer les colonnes

| Mots clés | CPC moyen prévisionnel | Concurrence entre annonceurs | ▼ Volume de recherche approximatif : décembre | Volume de recherche moye approximatif |
|---|---|---|---|---|
| **Mots clés en rapport avec le(s) terme(s) entré(s)** - trié par pertinence | | | | |
| lampe | €0,60 | | 2 240 000 | 2 240 000 |
| lampes | €0,34 | | 1 000 000 | 1 000 000 |
| lampe petrole | €0,37 | | 110 000 | 90 500 |
| lampe berger | €0,61 | | 90 500 | 90 500 |
| lampe bureau | €0,64 | | 60 500 | 49 500 |
| lampe chevet | €0,43 | | 60 500 | 49 500 |
| lampe maison | €0,66 | | 60 500 | 60 500 |
| lampe jardin | €0,39 | | 49 500 | 60 500 |
| lampe led | €0,80 | | 49 500 | 33 100 |
| la lampe | €0,37 | | 40 500 | 27 100 |
| lampe de bureau | €0,71 | | 40 500 | 27 100 |
| lampe de chevet | €0,43 | | 40 500 | 33 100 |
| lampe torche | €0,52 | | 40 500 | 40 500 |
| lampe uv | €1,08 | | 40 500 | 40 500 |
| lampe design | €0,59 | | 33 100 | 33 100 |
| lampe eclairage | €0,56 | | 33 100 | 40 500 |
| lampes design | €0,30 | | 33 100 | 33 100 |
| lampe jielde | €0,33 | | 27 100 | 22 200 |
| lampe philips | €0,68 | | 27 100 | 22 200 |
| lampe verre | €0,44 | | 27 100 | 27 100 |
| lampes berger | €0,14 | | 27 100 | 60 500 |
| lampe sodium | €0,40 | | 18 100 | 14 800 |
| lamps | €1,65 | | 14 800 | 14 800 |
| lampe ampoule | €0,40 | | 12 100 | 14 800 |
| lampe halogene | €0,41 | | 12 100 | 8 100 |
| lampe hps | €0,31 | | 12 100 | 9 900 |
| lampe loupe | €0,90 | | 12 100 | 9 900 |
| lampe projecteur | €1,91 | | 12 100 | 12 100 |
| lampe videoprojecteur | €2,14 | | 12 100 | 14 800 |
| verre de lampe | €0,42 | | 12 100 | 12 100 |
| acheter lampe | €0,42 | | 9 900 | 9 900 |
| lampes chevet | €0,41 | | 9 900 | 12 100 |
| lampe halogène | €0,32 | | 8 100 | 6 600 |
| lampes de chevet | €0,41 | | 8 100 | 9 900 |
| lampe camping | €0,39 | | 6 600 | 9 900 |
| lampe exterieur | €0,50 | | 6 600 | 6 600 |
| lampe extérieur | €0,56 | | 6 600 | 8 100 |
| lampe prix | €0,61 | | 6 600 | 5 400 |
| lampe usb | €0,44 | | 6 600 | 5 400 |
| achat lampe | €0,63 | | 5 400 | 5 400 |
| lampe aladin | €0,64 | | 5 400 | 4 400 |
| lampe de table | €0,37 | | 5 400 | 6 600 |
| lampe electrique | €0,38 | | 5 400 | 3 600 |
| lampe flash | €1,53 | | 5 400 | 5 400 |
| lampe pate de verre | €0,33 | | 5 400 | 5 400 |
| lampes de bureau | €0,69 | | 5 400 | 6 600 |
| luminaire lampe | €0,42 | | 5 400 | 5 400 |

► Fig. 7.7 : Résultats de Google KeywordTool sur "lampe".

KeywordTool nous donne de précieuses indications même si elles ne sont pas assez précises :

- Nous avons le trafic des recherches sur le mot-clé `lampe`, sur ses déclinaisons.
- Nous obtenons aussi des mots-clés à évaluer selon Google à cause des associations liées à *lampe* et réalisées par les internautes lors de leurs recherches.
- Il nous manque des chiffres plus précis. Mais ceux-ci sont en fait plus ou moins disponibles dans la gestion de compte AdWords. En plaçant des enchères durant une période de quelques semaines, on peut obtenir de très nombreuses informations complémentaires.

| Mots clés | CPC moyen prévisionnel | Concurrence entre annonceurs | Volume de recherche approximatif : décembre | Volume de recherche moyen approximatif |
|---|---|---|---|---|
| **Mots clés en rapport avec le(s) terme(s) entré(s)** - trié par pertinence | | | | |
| lampe décoration | €0,57 | | 90 500 | 40 500 |
| lampes décoration | €0,21 | | 201 000 | 165 000 |
| lampe decoration | €0,49 | | 1 900 | 4 400 |
| lampe de décoration | €0,05 | | Données insuffisantes | 3 600 |
| lampes de décoration | €0,05 | | Données insuffisantes | 720 |
| | | | | To |
| | | | Télécharger tous les mots clés : texte, .csv (p | |
| **Mots clés supplémentaires à envisager** - trié par pertinence | | | | |
| décoration d intérieur | €0,45 | | 110 000 | 135 000 |
| décoration intérieure | €0,36 | | 40 500 | 33 100 |
| décoration intérieur | €0,42 | | 135 000 | 165 000 |
| décoration maison | €0,33 | | 3 350 000 | 2 240 000 |
| lampe deco | €0,56 | | 27 100 | 22 200 |
| decoration interieure | €0,34 | | 33 100 | 33 100 |
| la maison décoration | €0,05 | | Données insuffisantes | 2 900 |
| décoration d interieur | €0,34 | | 8 100 | 6 600 |
| decoration interieur | €0,37 | | 74 000 | 60 500 |
| décoration feng shui | €0,19 | | 2 900 | 2 400 |
| decoration maison | €0,50 | | 110 000 | 90 500 |
| déco | €0,28 | | 1 220 000 | 1 500 000 |
| maison et décoration | €0,40 | | 27 100 | 110 000 |
| décoration | €0,29 | | 9 140 000 | 7 480 000 |
| décoration salle de bain | €0,35 | | 12 100 | 9 900 |

► Fig. 7.8 : Résultats de Google KeywordTool sur "lampe décoration".

Certaines associations suggérées sont intéressantes, associer *lampe décoration* à *intérieur*, *maison*, *deco*.

En faisant un travail itératif sur les multiples combinaisons et extensions proposées par l'outil, on peut mieux cerner la logique du thésaurus. En testant ces combinaisons via Google en ayant un compte Google ouvert, en simulant un historique, en effaçant cet historique (ce que permet Google à ce jour, tout au moins officiellement) et en recommençant, on pourra en tirer des informations essentielles sur la logique associative lui permettant de recréer une nouvelle requête incluant de nouveaux mots-clés pour le compte de l'utilisateur afin de personnaliser le résultat de sa recherche.

De ce que l'on perçoit aujourd'hui, Google transformera probablement une expression clé, celle saisie à l'origine par l'usager, en une autre requête pour le compte de cet usager via la personnalisation. On ne connaîtra pas l'expression finale générée et utilisée, c'est évident. Mais le référenceur aura accès aux mots-clés saisis à l'origine dans la requête sur Google car ils sont véhiculés dans l'environnement de l'internaute arrivant sur la première page du site. Le référenceur pourra deviner ou calculer ou estimer, avec l'aide d'un logiciel de suivi performant et d'un site bien conçu, la requête calculée par l'algorithme de personnalisation.

Peut-on en conclure tout de suite que le référenceur n'aura donc pas accès à des expressions clés/historiques mal prises en compte par le site à référencer ? Non. En effet, même mal positionné sur un mot-clé, un site a tout de même un peu de trafic sur une expression ou mot-clé mal référencés. Il appartient au référenceur de les détecter, de comparer le trafic potentiel existant (on vient de voir qu'il y a des outils Google pour cela) au trafic mesuré en audience, d'en déduire les expressions clés pas assez travaillées dans des landing pages et de corriger ces oublis ou ces erreurs.

Il est probable que Google réemploiera le thésaurus utilisé par d'autres services et applications de Google. En effet, pourquoi en prendre un autre ? Seul Google a accès aux données de l'historique de tout internaute ayant un compte Google, donc reconstituer l'algorithme sera impossible ou très difficile. Mais un référenceur expérimenté, avec un solide sens de la programmation, pourra auditer un site, des expressions clés et en déduire les déclinaisons capables de générer du trafic de qualité, personnalisation des résultats de recherche ou non. Certes, il ne pourra probablement pas calculer toutes les déclinaisons, mais il pourra en identifier suffisamment pour doper le trafic d'un site web.

## Principes probables de la personnalisation

Faute de pouvoir tester en réel en français, nous ne pouvons qu'étendre notre savoir-faire actuel et le projeter vers cette nouvelle approche de Google.

Nous avons vu qu'en utilisant la théorie des graphes puis les notions d'objets et de propriétés, on a défini un premier objet, URL, dotée de propriétés : titre de page, meta description, titre(s) H1, etc. Ces différentes propriétés contiennent des mots. Des algorithmes permettent d'en déduire les mots-clés.

Un autre objet a vite été défini, backlink, avec 3 propriétés pour faire simple : l'URL où son ancre est posée (l'URL départ du backlink), le texte du libellé du lien, l'URL visée.

Avec un peu de mathématiques simples, inclusion, intersection, pondération…, on a vu que très vite on pouvait construire un algorithme capable de calculer le positionnement d'une page précise sur une expression clé précise face à d'autres pages candidates à être positionnées sur cette même expression clé.

Très certainement, Google utilise des algorithmes bien plus sophistiqués pour comprendre le contenu de chaque page. Ces algorithmes existent depuis des années. Le challenge consiste à les faire s'exécuter correctement sur des masses gigantesques de données et à mettre à disposition de centaines de millions d'internautes des temps de réponse rapides lors de leurs recherches ou utilisation des multiples services et fonctionnalités de Google.

Étendons cette logique à la personnalisation des résultats de recherche. L'objectif n'est pas de tenter de faire un *reverse engineering* de Google, mais juste de poser quelques principes. Cela nous permettra de mieux appréhender les actions et méthodes pour réussir un référencement.

Un historique est composé de trois objets au minimum :

- requête : les mots-clés saisis dans une requête ;
- URL visitée : une URL cliquée dans une SERP Google elle-même renvoyée vers un compte Google ayant envoyé une requête ;
- compte Google : un usager utilisant au moins un service Google comme Gmail, Google Analytics, etc.

Nous percevons immédiatement les propriétés évidentes à accrocher à ces objets. Voici une première approche logique dans la lignée de ce que nous avons déjà à peu près établi du fonctionnement des algorithmes de Google.

Requête :

- identifiant requête ;
- compte Google auteur de la requête ;
- date et heure, minute, seconde ;
- mots-clés.

URL visitées :

- identifiant requête à l'origine de la SERP d'où l'URL a été cliquée ;
- date et heure, minute, seconde où le clic a été fait ;
- ID URL (pour accéder aux propriétés de la page visitée ; même objet que celui vu précédemment).

Compte Google, champs supplémentaires à ajouter à la structure déjà existante :

- Thèmes préférés sur les N derniers mois. Illustration : si un internaute saisit de nombreux mots-clés liés à la décoration d'une maison, toutes recherches sur des mots-clés comme `tissu` ou `lampe` verront leurs résultats personnalisés sur `lampe décoration` et `tissu mobilier` ou `tissu décoration maison` plutôt que `lampe ampoule` et `tissu pour vêtement`. Logiquement, les composantes du profil de l'internaute seront organisées pour être compatibles avec le fonctionnement du thésaurus.
- Mots-clés fréquemment utilisés, en requêtes ou présents dans les URL visitées, sur les N derniers mois. Une pondération est possible pour permettre un classement par ordre d'importance des mots-clés utilisés sur cette période.

Une telle structure offre de nombreux avantages. Elle permet d'être "compatible" avec la structure précédente. Elle est apparemment simple à mettre en œuvre. Elle permettrait de multiples réglages à la volée diminuant ou augmentant la consommation de ressources. Elle est compatible avec les rares déclarations de Google sur le sujet.

Elle présente au moins un inconvénient : elle est potentiellement très consommatrice de ressources. Si pour chaque personnalisation de résultats de recherche, le serveur ausculte 18 mois d'historique, il risque d'y avoir une trop forte consommation de ressources informatiques. Ce scénario est incompatible avec le maintien de temps de réponses rapides qui est une des caractéristiques de Google comme moteur de recherche.

De multiples approches algorithmiques sont possibles pour alléger le coût en consommation de ressources informatiques :

- Préconstruire le profil, les mots-clés et les thèmes préférés d'un compte Google. Cela éviterait de parcourir et traiter des centaines ou même des milliers d'enregistrements pour évaluer en temps réel la personnalisation d'un résultat de recherche. Cela permet également une meilleure efficacité. Le traitement étant en batch, une fois de temps en temps, Google pourrait utiliser des algorithmes plus complexes, basés sur de l'IA (Intelligence Artificielle), pour mieux comprendre les contenus visités par l'internaute et donc mieux classer et catégoriser son profil en respectant le système du thésaurus.

- Consulter en temps réel la dernière heure (dernière 30 minutes ou les 100 dernières utilisations, etc.) d'historique en requêtes et en URL visitées liées à ces requêtes.

Le peu d'informations données par Google sur leur architecture, si ces informations sont vraies, indiquent que les techniques de compréhension des textes sont plutôt basées sur de l'IA et qu'ils en sont particulièrement contents depuis début 2008, où, nous citons en traduisant, "un saut qualitatif très net leur permet de plutôt bien appréhender désormais ce que raconte une page [de site web]" (interview en anglais d'un des directeurs techniques de Google).

Des tests sur 2008 ont montré effectivement un saut qualitatif très net. Par exemple, en 2007 les SERP étaient truffées d'URL dont les pages avait du soi-disant contenu mais dont la valeur ajoutée réelle était nulle. Exemple : les annuaires de liens et autres site à contenus bidons étaient bien mieux référencés sur des mots-clés liés au tourisme. Ils encombraient les SERP sur des recherches pour louer un appartement ou une maison. Désormais, ils ont pratiquement disparu des SERP. Le contenu, quoique existant et truffé des bons mots-clés, n'est plus pris en compte par Google.

Différentes campagnes de tests et de comparaisons permettront d'affiner ou d'amender ce modèle.

## Chercher des associations

Les associations les plus courantes réalisées par les internautes seront consultables via AdWords et ses outils. En effet, pour augmenter ses revenus et satisfaire les clients payants, Google doit proposer des AdWords correspondant le mieux possible aux attentes de l'internaute. L'outil *keywordTool* devrait offrir aux clients AdWords la possibilité de louer des mots-clés en lien avec la recherche supposée, personnalisation ou non. L'outil suggèrera les mots-clés dont Google sait qu'ils sont mathématiquement intéressants. Exemple : `Lampe` donne accès aux suggestions *lampe berger*, *lampe bureau*, *lampe maison*, *lampe chevet*, *lampe luminaire*, etc. `lampe Maroc` donne accès aux suggestions *lampe marocaine*, *artisanat marocain*. `lampe artisanat marocain` informe que c'est une impasse : aucune requête. Mais quelque part dans la chaîne, on aperçoit `lampe salon` avec un potentiel `"large"` de 12 000 requêtes mensuelles. En `"exact"`, il reste plus de 5000 requête/mois en moyenne.

Personnalisation ou non, ces informations restent valides. Si la personnalisation modifie les volumes ou les associations de mots-clés, pour garder ses clients et augmenter ses revenus, Google via `keywordTool` devra en tenir compte et mettre à disposition les données résultantes.

Référenceur et mot-clé selon Google

Matt Cutts, personne importante chez Google, indiquait en 2007 que le métier de référenceur se focaliserait dans le futur (avec la personnalisation des résultats de recherche opérationnelle) sur les mots-clés sous-entendus dans leurs sélections et dans leurs accessibilités sémantiques sur le site.

## Hypothèse #1 : historique décousu

Notre internaute, chercheur de lampe pour une décoration orientale, a navigué via des clic dans les SERP de Google sur de multiples sites sans rapport avec une lampe : bourse, informations quotidiennes, recherches pour changer sa voiture, etc. Puis il recherche une lampe (orientale). Nous voulons être positionné naturellement sur cet internaute lors de sa recherche.

S'il saisit juste `lampe`, nous n'avons aucune chance d'être vu. Son historique ne permet pas à Google de comprendre ce qu'il cherche.

L'internaute, devant l'inadéquation des résultats, devrait compléter sa requête. Nous restons dans le cas classique d'un ranking traditionnel.

## Hypothèse #2 : historique faisant sens

Notre internaute cherche une idée de décoration. Il recherche et navigue sur des sites de décoration. Puis il saisit `lampe` dans une requête sur Google.

Les informations de Google publiées en anglais sont claires : le but de la personnalisation des résultats de recherche est d'éviter d'afficher des thèmes sans rapport avec la recherche présumée de l'internaute. A priori, cet internaute exemple ne cherche pas une *ampoule*. Il recherche soit un luminaire (*lampe luminaire*), soit une pure décoration (*lampe décoration*). Selon les pages cliquées depuis Google, son historique, le moteur affichera plutôt soit uniquement des pages liées à "*lampe luminaire*", soit uniquement des pages liées à "*lampe décoration*" ou un mélange des deux thèmes.

## Web analytics et référencement naturel


| | Mot clé | Pages vues | Consultations uniques | Temps passé sur la page | Taux de rebond | Sorties (en %) | Indice de valeur |
|---|---|---|---|---|---|---|---|
| 1. | [redacted] kit | 209 | 111 | 00:00:24 | 15,00 % | 9,57 % | $0,00 |
| 2. | [redacted] en kit | 59 | 29 | 00:00:22 | 12,50 % | 15,25 % | $0,00 |
| 3. | [redacted] kit | 41 | 26 | 00:00:42 | 33,33 % | 21,95 % | $0,00 |
| 4. | kit [redacted] | 22 | 12 | 00:00:31 | 16,67 % | 27,27 % | $0,00 |
| 5. | comment construire [redacted] en kit | 18 | 9 | 00:00:14 | 0,00 % | 5,56 % | $0,00 |
| 6. | notice de montage [redacted] | 18 | 9 | 00:00:09 | 0,00 % | 11,11 % | $0,00 |
| 7. | [redacted] kits en [redacted] | 14 | 6 | 00:00:07 | 0,00 % | 7,14 % | $0,00 |
| 8. | kit [redacted] | 14 | 8 | 00:00:09 | 50,00 % | 14,29 % | $0,00 |
| 9. | [redacted] kit | 13 | 6 | 00:02:00 | 0,00 % | 7,69 % | $0,00 |
| 10. | [redacted] livré en kit + prix | 12 | 5 | 00:00:22 | 0,00 % | 8,33 % | $0,00 |

► Fig. 7.9 : Google Analytics : les mots-clés utilisés pour arriver sur une landing page

*Google Analytics* indique les mots-clés utilisés par les internautes arrivant sur une première page d'un site : **Contenu/Principales pages de destination/Mots clés utilisés (page d'entrée)**.

Dans la figure, cette landing page, un exemple réel, nous l'appellerons "kit", cible l'expression clé *kit* associé à quelques autres mots-clés. 71 déclinaisons différentes utilisant une dizaine de mots-clés ont amené 653 affichages de cette landing page (106

avec cette page comme page d'arrivée dans le site) lors de 379 visites uniques environ (lors d'une visite, le visiteur peut consulter deux fois la même page). Les expressions clés affichées sont celles véhiculées par l'environnement de l'internaute lors de son arrivée sur cette page ou sur une autre page du site.

**Contenu/Principales pages de destination/Récapitulatif de navigation** permet d'identifier clairement, dans Google Analytics, l'usage en landing page (page d'arrivée) ou en page sur le chemin d'une visite.

On voit que le taux de rebond est faible, moins de 15 % en moyenne pour l'expression clé ciblée par notre landing page, on a 33,85 % sinon pour la moyenne de toutes les visites.

Le taux de sortie immédiate est également très faible, de l'ordre de 12 % pour l'expression clé ciblée par notre landing page. On a 19,91 % autrement pour la moyenne de toutes les visites.

Il serait intéressant de savoir ce qu'ont fait ces internautes ensuite. Mais Google Analytics ne sait pas délivrer un détail précis d'un internaute. On peut obtenir une séquence début : *page précédente – notre landing page – page suivante*. Mais c'est insuffisant. Et Google Analytics n'est pas là pour aider un référenceur, mais un webmestre utilisant AdWords. La personnalisation va accentuer les différences de besoins en outils des *AdWorders* d'un côté et des référenceurs naturels de l'autre. Les outils actuels de Google devraient encore moins correspondre aux futurs besoins des référenceurs naturels.

Demain, avec la personnalisation, le référenceur aura besoin de visualiser le chemin complet d'un internaute précis pour tenter de reconstituer une expression clé générée par Google. Il lui faudra au minimum isoler les mots-clés utilisés pour arriver sur telle landing page par des internautes et isoler chaque chemin réalisé ensuite par chacun de ces internautes. Le référenceur avait déjà besoin de faire un peu de *web analytics* pour suivre l'évolution d'un référencement. Cela va devenir indispensable.

### ⓘ Web analytics

Pour s'adapter à la personnalisation des résultats de recherche de Google, le référenceur 2.0 va devoir maîtriser les outils d'audience. Le référenceur codeur aura peut-être besoin de coder certains utilitaires lui-même, notamment pour se faciliter la tâche de reconstituer l'expression clé construite par la personnalisation. Une certaine forme de web analytics va devenir primordiale dans le métier de référenceur.

L'usage web analytics du référenceur sera orienté sur la sélection, le classement, la priorisation des mots et expressions clés. Les autres spécialités du web analytics essaient d'extraire d'autres logiques, comportementales par exemple, pour transformer un internaute prospect en client acheteur.

## Programmation et référencement naturel

Google Analytics n'est pas assez performant, peut-être est-il bridé, pour les besoins d'un référenceur. C'est certes une bonne base, un bon outil d'apprentissage, mais ses fonctionnalités sont insuffisantes pour appuyer un référencement naturel.

Il sera vite indispensable de disposer d'outils générant automatiquement les mots-clés d'arrivée par landing page associés aux chemins de ces internautes. En effet, travailler manuellement pour une expression clé, pourquoi pas. Mais quand un site complet est à prendre en charge, cela devient vite ingérable.

### Coder

Pour travailler sur les bonnes expressions clés afin de prendre en compte au mieux la personnalisation des résultats de recherche, un référenceur devra avoir de solides compétences en programmation.

Soit il en aura besoin pour coder des scripts PHP utilitaires divers et variés afin d'automatiser les tâches fastidieuses mais indispensables nécessaires à son travail. Plus des méthodes fiables sont automatisées, plus efficient sera le travail du référenceur.

Soit il en aura besoin pour évaluer un outil proposé par une société spécialisée.

## Google writing et référencement naturel

Au travers des exemples vus, nous pouvons aussi percevoir qu'une autre dimension s'imposera au niveau du référenceur : la mise en page fonctionnelle et le rédactionnel sous la forme de *Google writing*.

Sélectionner les expressions clés, les factoriser est stratégique. Mais il faut ensuite les implanter en respectant la priorisation des termes. Ce travail pourra être partiellement automatisé avec un programme adéquat.

Exemple :

```
Lampe
   [ [déco] || [orientale | marocaine] ]
   [maison | intérieur]
   [cuir | cuir de chèvre | cuir et métal | cuir de chèvre et métal]
```

- | signifie : *ou exclusif*, soit un terme soit l'autre, jamais les deux en même temps.
- || signifie : *ou inclusif*, soit un terme soit l'autre, soit les deux en même temps.

Cette ligne exprime en factorisation :

- Le mot-clé central stratégique est : `lampe`. Il doit apparaître dans un maximum de critères de page : `TITLE`, `Meta Description`, `ALT`, `H1`, etc.
- `Lampe` doit être associé à `déco`, synonyme de `décoration` selon le thésaurus de Google. `Lampe de décoration` doit être présente dans tous les critères de page.
- `Lampe` doit être associé à `orientale` ou à `marocaine`, synonyme de `Maroc` selon le thésaurus de Google. Ces expressions clés doivent être présentes dans tous les critères de page.
- Puis on peut combiner `lampe décoration` avec soit `orientale`, soit `marocaine`.
- Puis on peut combiner ces expressions clés avec soit `maison`, soit `intérieur`.
- Et enfin, on peut combiner avec les mots décrivant la composition de la `lampe`, du `cuir` de `chèvre` et du `métal`.

En une ligne exprimant une factorisation, on résume les objectifs en expressions clés d'une landing page : ordonnancement des mots et priorisation. Nous avons vu dans les chapitres précédents comment implémenter, dans une table MySQL, de telles propriétés et comment les manipuler en script PHP. Cela automatise partiellement ou totalement l'accessibilité sémantique de chaque page au sein d'une stratégie globale de référencement.

ⓘ Google writing

Pour réussir le positionnement via l'accessibilité sémantique, faire prendre en compte les bons mots-clés par Google pour chaque landing page, les compétences en Google writing devront être utilisées bien plus qu'auparavant. Il va falloir décliner correctement chaque factorisation d'expressions clés.

## Ébauche d'algorithme : référencement et personnalisation

Les enjeux sont :

- reconstituer, même approximativement, une expression clé générée par Google lors de la personnalisation de résultats de recherche ;
- puis corréler avec le taux de transformation afin d'identifier les expressions clés capables d'apporter des internautes transformés, c'est-à-dire devenant clients.

Nous avons vu, au paragraphe précédent, qu'on avait besoin :

- D'un outil collectant les mots-clés des internautes arrivant pour la première fois sur une landing page d'un site. En clair, on a besoin de connaître les performances de

collecte en premiers arrivants de chaque landing page. Sert-elle à quelque chose ? Est-elle bien positionnée sur ses expressions clés, avec ou sans personnalisation des résultats de recherche ? A-t-elle assez de trafic par rapport aux informations issues des outils de Google ?

- De disposer pour chaque internaute arrivant via une landing page donnée, de tout son parcours sur le site. On obtient ainsi les performances de transformation de chaque landing page. Il suffit de détecter le passage de l'internaute sur les pages adéquates : remerciements pour un formulaire rempli, visibilité de plus de 10 secondes de telle promesse marketing, etc.
- D'avoir évalué pour une expression clé, via des simulations d'historiques, le comportement de Google.

On aurait pour une landing page :

- Un nombre limité de mots-clés (`lampe`, `décoration`, `orientale`, etc.) pouvant composer des centaines de combinaisons d'expressions clés.
- Ces mots-clés sont inférieurs à 15 voire moins et sont parfaitement identifiés. Ils sont présents dans une table MySQL. Normalement, si on suit ce qui a été expliqué auparavant, on a pu identifier certaines des expressions clés utilisées par Google en personnalisation et elles sont intégrées dans la factorisation de mots-clés de la landing page.
- Moyennant un bon usage de Google Analytics ou mieux, un petit programme, on peut identifier les mots-clés utilisés par un internaute arrivant sur une première page puis suivre son parcours sur tout le site, y compris s'il revient (via un cookie).
- On a pu exporter dans une table MySQL (chaque mois, chaque trimestre via un fichier *.csv* par exemple) depuis un outil Google comme `keywordTool` le trafic en requêtes estimé ou mesuré réel sur telle ou telle expression clé, en large (c'est-à-dire toute expression clé contenant les termes indiqués dans n'importe quel ordre) ou en strict, c'est-à-dire cette expression clé très précise.
- Un quelconque outil de positionnement nous donne les positions de la landing page sur ces expressions clés directement dans une table MySQL

Nous avons les données, il n'y a plus qu'à automatiser corrélations et suivi dans le temps.

Pour chaque landing page, on afficherait :

- Les expressions clés des premiers arrivants sur les N derniers jours (N = 30 par exemple). Un programme de collecte de ces données via `$_SERVER` en PHP est idéal pour cela.

- Le taux de transformation (une transformation signifie l'atteinte d'une URL comme "*merci d'avoir passé commande*" par un internaute) de ces premiers arrivants sur la landing page. Gérer `$_SERVER` et un cookie permet de tracer le parcours de l'internaute dans un fichier *.txt* puis d'importer ce fichier, la nuit par batch, dans une table MySQL afin de pouvoir identifier si tel internaute est passé par cette URL de remerciement ou non.
- Les mots-clés principaux de la page ou même la factorisation retenue (on peut ajouter un champ à la table *article* pour prévoir cette possibilité).
- Le trafic existant sur ces expressions clés dans Google, en large et en respect strict de l'expression clé.
- Le positionnement de la landing page dans Google, expression clé par expression clé.

Ces informations permettraient de cadrer d'un coup d'œil les performances de chaque landing page sur les 30 derniers jours.

On peut améliorer l'approche en introduisant la notion de suivi sur N mois, de comparaison avec les statistiques globales du site, de comparaison entre landing pages, de benchmarker les positions des sites concurrents sur les expressions clés de chaque landing page, etc. Mais nous avons là le cœur d'un système de suivi des performances.

## 7.9 Résumé de ce chapitre

L'arrivée de *Google Personalized search results* (la personnalisation des résultats de recherche) va profondément bouleverser le référencement et le positionnement des sites web dans les années à venir.

La personnalisation utilisera l'historique des requêtes réalisées sur Google et aussi l'historique web des URL cliquées depuis les SERP de Google.

De nombreuses sources d'informations, Doubleclic, Google ToolBar, Chrome, pourraient théoriquement permettre à Google de tracer l'activité de tout internaute ayant utilisé Google et possédant un compte Google. Le moteur dispose de ressources pour faire progresser et monter en puissance son approche. Ses concurrents, Microsoft et Yahoo, n'ont pas une telle allonge.

La personnalisation des résultats de recherche permettra à Google de laminer la concurrence, d'améliorer les recettes de AdWords en freinant l'activité des référenceurs et de fidéliser encore plus d'internautes à des outils.

Dans ce chapitre nous avons réalisé une première esquisse sur le référencement prenant en compte la problématique de la personnalisation des résultats de recherche.

Le métier du référenceur va se concentrer sur les mots-clés et leurs prises en compte par le moteur (les bons mots-clés sur les bonnes pages avec la bonne mise en exergue).

En appréhendant mieux les principes de programmation d'une personnalisation des résultats de recherche, on perçoit mieux comment procéder pour disposer d'un référencement efficace.

### Mesurer l'efficacité du référencement

Une ébauche d'un algorithme permettant de suivre la qualité de son référencement vous permettra de démarrer une nouvelle approche pour mesurer l'efficacité de votre référencement et de votre référenceur.

# 8

# Annexe Webographie

Vous trouverez dans ce chapitre différentes URL pointant des astuces, des conseils, des outils présentant un intérêt direct ou indirect avec les sujets abordés dans ce livre :

- référencement ;
- forum sur le web marketing dont le référencement naturel n'est qu'une des composantes ;
- programmation web ;
- outils divers facilitant la mise au point de programmes.

À vous de jouer !

## 8.1 Outils pratiques

- **http://www.webrankinfo.com/outils/header.php** est un outil en ligne permettant d'analyser un header *http*. Il s'avère très pratique quand on met au point un *.htaccess* en charge de faire des redirections.
- **http://ip-to-country.webhosting.info/node/view/36** renvoie le pays correspondant à une adresse IP. En effet, avant de faire héberger un site, il est important de savoir où sont installés physiquement les serveurs sous peine de perdre bêtement du trafic. Rappelez-vous, le filtrage Google donne priorité aux "locaux" pour les locaux. Si votre site en *.com* est hébergé en Allemagne ou aux États-Unis par une société française, vous serez vu comme un étranger au marché français. Vous garantissez ainsi une perte de trafic notable à votre site. Ce n'est certes pas une bonne affaire si vous passez par un tel hébergeur pour économiser 0,5 euros par mois.

### Adresse IP

Capturez l'adresse IP de votre fournisseur : la commande UNIX `ping`, disponible sous l'émulateur DOS sur les machines sous Windows, vous la donnera.

```
C:\> ping www.lemonde.fr
```

Cette commande vous renverra `193.159.160.139`, hébergé physiquement en Allemagne. C'est un nom de domaine en *.fr*, donc il est automatiquement vu comme un site français où qu'il soit hébergé.

Un autre exemple : le serveur de **www.webrankinfo.com** a pour adresse IP `194.146.226.133` localisé en France.

- **http://www.hyper-lien.com/articles/ip-site.php** est un petit outil en ligne qui vous donne directement l'adresse IP d'un site web.
- **https://adwords.google.fr/select/KeywordToolExternal?defaultView=3**. Ce logiciel en ligne de Google vous permettra d'évaluer des expressions clés. C'est un des outils indispensables pour auditer des mots et expressions clés.
- **www.google.com/webmasters/tools/?hl=fr**. *Google webmasters tools* donne accès au webmestre à de nombreux outils facilitant le travail de référencement.
- **http://yoast.com/seo-tools/link-analysis/**. Le site néerlandais propose des outils gratuits, payants ou en version d'essai qui aident le référenceur. Il existe par exemple un plugin Firefox qui affiche le Page rank des backlinks listés dans Google webmasters tools. Cela offre un gain de temps pour apprécier en quelques coups d'œil une partie de la qualité des backlinks d'un site.

- http://noscript.net/. Add on pour Firefox afin de sécuriser votre navigateur en limitant la consommation de ressources par les flash mal codés et trop gourmands en ressources de votre processeur.
- http://www.wampserver.com/. L'environnement de développement utilisé : Windows, Apache, MySQL, PHP5 de Anaska. Il permet de développer dans un environnement identique à 99,95 % à celui de votre serveur Linux de production. Les différences notables qui peuvent piéger le codeur débutant portent :
  - Sur des détails d'implémentation de *.htaccess*. Les hébergeurs OVH, Nuxit et quelques autres ont une tendance à "personnaliser" des détails qui peuvent perturber WAMP et rendre un *.htaccess* WAMP incompatible avec celui destiné au serveur Linux de l'hébergeur.
  - Le paramétrage Apache de votre serveur WAMP peut être impossible à reproduire sur un serveur mutualisé, notamment l'URL rewriting.
  - Certains hébergeurs présentent des comportements d'URL rewriting aléatoirement surprenants nécessitant une modification substantielle du *.htaccess* ou de la programmation PHP des liens `HREF` générés. Exemple : l'URL du site, `nomDeDomaine.com/article/charges-sociales/12/1.html` est remplacée par une string du style `…/gp60/stylez~/article/charges-sociales/12/1.html`.
- Pour générer une entrée dans un fichier *.htpasswd*, voici une des multiples URL proposant un outil en ligne : http://www.htaccesstools.com/htpasswd-generator/.

## Pour soumettre une URL

Quand vous préparez la mise en ligne d'un site sous un nouveau nom de domaine, il est préférable d'ouvrir une page d'attente dès que vous disposez du nom de domaine. Vous la remplissez avec un texte de quelques centaines de mots, truffés de mots et expressions clés afin de prendre date avec les moteurs de recherche, d'obtenir un "profilage" de votre futur contenu. Au passage, vous allez acquérir un peu d'ancienneté et ainsi vous débarrasser des protections contre le spamdexing opéré depuis les très jeunes sites web :

- http://fr.siteexplorer.search.yahoo.com/free/submit pour soumettre son site à Yahoo ;
- http://www.google.fr/addurl/ pour soumettre son site à Google ;
- http://search.msn.fr/docs/submit.aspx pour soumettre son site à Microsoft Live Search ;
- http://www.exalead.fr/search??definition=submitYourSitePage pour soumettre son site au moteur français Exalead.

## 8.2 Conseils, astuces et dépannages

- **http://www.webrankinfo.com/forums/index.php**. Forum sur le référencement où interviennent des personnes de tout niveau, du débutant à l'expert. Ce forums et bien d'autres donnent accès à des astuces pour coder en expressions régulières, en PHP…, sans avoir à lire et comprendre toute la documentation sur le sujet. On y trouve aussi des astuces, conseils, coups de main pour se dépanner ou avoir une idée face à une situation incompréhensible.
- Voici un excellent exemple de bonne publication par ce site : **http://www.webrankinfo.com/actualites/200812-le-referencement-en-2009.htm** présente un excellent topo du métier du référencement en 2009 par rapport à 2008.
- **http://www.webmaster-hub.com/index.php** est un autre forum classique et pratique pour le référencement, les problèmes de programmation, les astuces, les coups de main.
- **http://www.forum-marketing.com/index.php** : un dernier forum francophone sur le référencement.
- **http://www.mattcutts.com/blog/** : le blog de Matt Cutts, patron de l'antispam team Google. On y trouvera des sujets inintéressants (les photos de son chat, les logiciels préférés de ses parents …), des informations intéressantes touchant au référencement, plus ou moins précises et des astuces intéressantes (les conseils pour réinclure un site en black list, par exemple).

  *Googlez interview matt cutts* permet d'obtenir de réelles informations intéressantes sur le référencement et Google.

  Une autre expression clé à Google : `Matt Cutts` + *terme de référencement*. Exemple : `Matt Cutts Duplicate Content` renvoie une SERP contenant cette URL pointant une page en français avec des informations intéressantes sur le traitement par Google du propriétaire original d'un contenu : **http://wordpress-tuto.fr/complements-de-matt- cutts-sur-le-duplicate-content-307**.

- Pour s'initier facilement et comprendre les principes de la sécurisation du back office d'un site web, mais avec une faille de sécurité, donc à lire mais à ne pas implémenter tel que : **http://www.siteduzero.com/tutoriel-3-14649-proteger-un-dossier-avec-un-htaccess.html**.

### ⓘ .htpasswd inviolable

Tout *.htpasswd* doit être dans un répertoire dédié accompagné d'un *.htaccess* en `deny all` afin de limiter l'accès à *.htpasswd* à Apache et au protocole FTP ou autre retenu pour mettre à jour le fichier. Cette précaution était omise dans cette page d'explications par ailleurs très bien faite pour expliquer les principes de la sécurisation d'un répertoire dans un site web.

Pour obtenir des pages sur la programmation dans un `.htaccess` pour de l'URL rewriting ou de la redirection par exemple, "googlez" `tutorial .htaccess`.

- Le blog officiel de Google est : **http://googleblog.blogspot.com/**.
- Des informations utiles pour mieux comprendre comment optimiser vos `snippet` et balises sur vos pages : **http://googlewebmastercentral.blogspot.com/2007/12/answering-more-popular-picks-meta-tags.html**.

## 8.3 Résultats de recherche personnalisés

Le blog officiel de Google présente depuis fin juillet 2008 des informations sur cette approche : **http://googleblog.blogspot.com/2008/07/more-transparency-in-customized-search.html**.

"Googlez" `personalized search` ou `Google customized search results` en limitant la date des pages aux derniers 31 jours listera les pages les plus récentes sur ce sujet. En saisissant l'option `inurl:google`, vous filtrerez les résultats efficacement.

## 8.4 Normes et manuels

Nous avons vu que les cookies étaient importants en web marketing. Voici quelques URL présentant un net intérêt pour les codeurs :

- Définition des cookies : la RFC 2109 : **http://www.ietf.org/rfc/rfc2109.txt**.
- Sa remplaçante, la RFC 2965 : **http://tools.ietf.org/html/rfc2965**.

Les codeurs utilisant flash trouveront de l'intérêt dans ces deux URL :

- **http://code.google.com/p/swfobject/wiki/documentation** ou comment implémenter une animation flash en lui donnant un texte alternatif accessible aux bots de Google (et des autres moteurs).
- l'URL de la traduction française de SWFobject2 : **http://egypte.olympe-network.com/swfobject-francais.html**.

### Théorie des graphes

Les mathématiciens et les informaticiens de Google utilisent massivement cette branche des mathématiques dans leurs algorithmes sur les backlinks, probablement pour lutter contre le spamdexing.

Cette branche des mathématiques est enseignée à partir de bac+2 à bac+4 selon les universités ou écoles d'ingénieurs. Tout webmestre ou codeur n'ayant jamais abordé cette

branche des mathématiques devrait s'y plonger pour mieux appréhender le génie logiciel et… Google, bien sûr.

Voici quelques URL sur le sujet :

- http://www.laas.fr/~lopez/cours/GRAPHES/graphes.html ;
- http://fr.wikipedia.org/wiki/Théorie_des_graphes ;
- http://www.apprendre-en-ligne.net/graphes/.

## 8.5 URL de codes sources

Dans ce paragraphe, vous trouverez les sites à l'origine ou ayant inspiré certains codes sources utilisés dans l'exemple présenté dans ce livre.

### Session invisible pour Google

Le code source de la fonction, `session_region()`, en charge de suivre le comportement des bots sur des URL et des pages optimisées SEO et utilisées dans l'exemple est issu d'une base dont l'origine est publiée en libre accès :

- Remi Aubert & Alan Boydell ;
- Remi : www.remiaubert.com ;
- Alan : www.analytics.fr.

Le code source d'origine servait à traquer le trafic des bots sur un site web via Google Analytics. Ce code source a été modifié et adapté par Gilles Grégoire le 23/12/2008 pour remplacer son propre code moins élégant et dont il était plus difficile d'assurer la maintenance. Un bogue figurait dans le code source d'origine. Il a été signalé aux codeurs d'origine qui ont mis à jour leurs sources.

Plusieurs sites fournissent la liste des bots en circulation (plus de 2 000 actifs en 2008 selon l'un de ces sites). Il est recommandé de mettre à jour régulièrement cette liste depuis un de ces sites :

- La liste des bots utilisée dans notre exemple provient de : BBclone www.bbclone.de ainsi que le script *patterns.php*.
- Un autre site régulièrement mis à jour diffuse une liste téléchargeable automatiquement et qu'un codeur pourra utiliser pour mettre à jour sa liste de bots automatiquement : http://www.robotstxt.org/db/schema.txt. Le http user agent du robot, la variable utilisable en PHP, correspond à la ligne `robot-useragent` dans le fichier téléchargeable.

## Feuille de style

Certaines parties des feuilles de style de notre exemple ont été dérivées d'exemples ou d'informations techniques exposés dans ces pages :

- http://www.cssdebutant.com/formulaire-css.html ;
- http://www.babylon-design.com/site/index.php/2007/09/24/192-boutons-extensibles-css-compatibles-tous-navigateurs.

## Réseaux sociaux

Un des exemples de codes sources pour faciliter le bookmarking d'une page d'un site web dans un site réseau social est de l'auteur. D'autres exemples peuvent être librement pris d'un de ces sites :

- http://justaddme.com/.
- L'exemple utilise un code source issu de ce site qui permet de disposer de statistiques sur les clics des usagers : http://www.addthis.com/.
- http://www.letsgetsocialnow.com/.

# INDEX

# !



# A



# B

# C



# D

# E



# F

# G



# H



# I



# J



# K

# L



# M



# N

## O



## P

# Q



# R

# S

# T



# U



# W

# Z

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

# Notes

Composé en France par Jouve
11, bd de Sébastopol - 75001 Paris