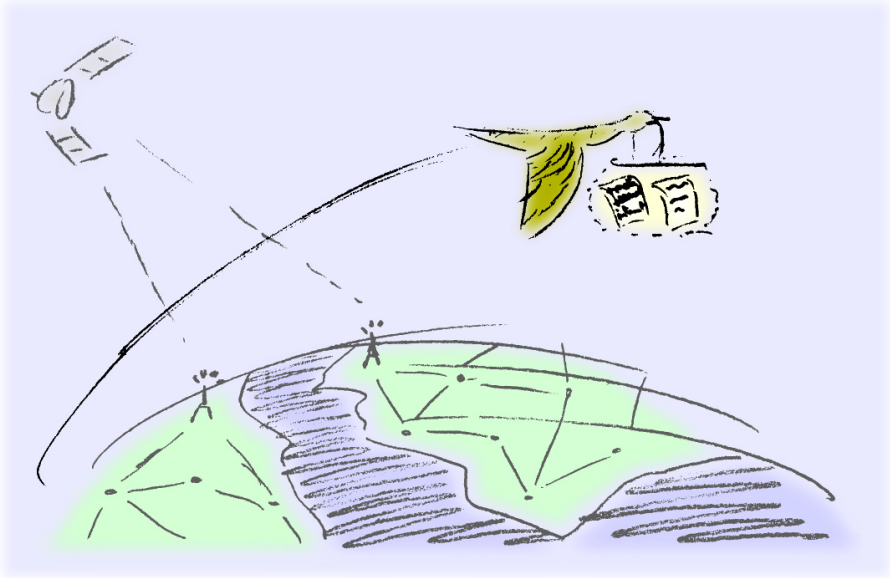
# Donnez des ailes à vos requêtes

**Vous faites régulièrement les mêmes recherches sur Internet (par exemple pour de la veille technologique) et vous avez l'impression de perdre à chaque fois une grande partie de votre précieux temps ? Alors, il vous faut un outil bien plus puissant qu'un simple moteur de recherche : un « meta moteur automatique de groupe ».**



**Par Nicolas ESPOSITO**
*nik@niksnews.com*

Sur le CD-Rom

Nous présentons dans cet article un meta moteur permettant d'automatiser des recherches dans un contexte de travail de groupe. Un prototype à été réalisé, il s'agit de MetaMAG (Meta Moteur Automatique de Groupe), voir **figure 1**. Nous en proposons ici une version minimale. Elle vous permettra de découvrir le concept général de l'approche et un langage particulièrement adapté à ce type de travaux : WebL.

## Les points clés de l'efficacité

**L'automatisation :** toutes les nuits, MetaMAG cherche les nouveaux résultats pour chaque requête. S'il y en a, il les distribue par e-mail aux abonnés.

**Une requête = un groupe :** la liste des abonnés à une requête peut être considérée comme une liste de diffusion. Le groupe ainsi formé peut communiquer autour du thème de la requête.

## Exemple

Supposons que votre domaine d'activité soit le développement de jeux avec Java3D. Alors, vous créez une nouvelle requête dans MetaMAG (par exemple : « +java3d +games ») et vous sélectionnez les moteurs de recherche suivants : Altavista, Yahoo.com et HotBot. Le premier jour, vous recevez par e-mail près d'une trentaine de résultats. Par la suite, vous ne recevrez que les résultats correspondants aux nouvelles pages indexées dans les moteurs que vous avez choisis, de façon tout à fait automatique. Vos collaborateurs pourront s'abonner à votre requête et recevoir eux aussi toute l'actualité Web des jeux utilisant Java3D. Et d'un simple clic, vous pourrez contacter tous les abonnés, par exemple pour leur signaler l'intérêt insoupçonné d'une page.

## Infrastructure

MetaMAG s'appuie sur trois bases de données (voir **figure 2**) : une base qui contient les requêtes et deux bases qui contiennent des résultats. La base « Anciens résultats » contient tous les résultats qui ont déjà été trouvés. « Nouveaux résultats » est une base qui est utilisée temporairement pour stocker les résultats qui seront distribués à la fin de la session.

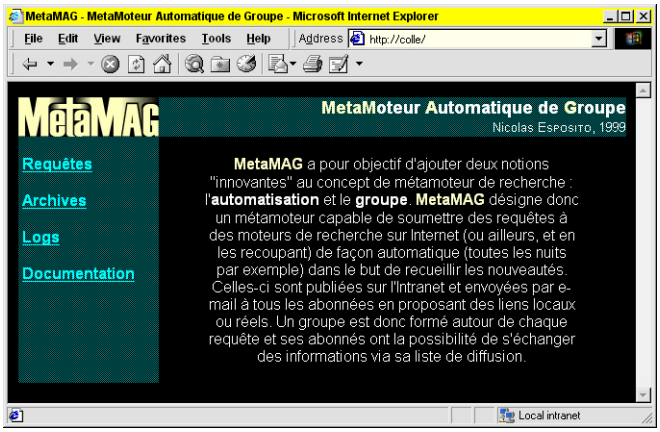
Afin de simplifier le problème du traitement de ces bases, nous avons fait les choix suivants : il y a un répertoire de résultats pour chaque requête et chaque résultat est stocké dans un fichier dont le nom est calculé à partir de son URL. On procède ainsi à deux recoupements implicites : un recoupement au niveau d'un moteur de recherche (s'il propose plusieurs fois un même résultat) et un recoupement entre les différents moteurs. Si le résultat www.niksnews.com est fourni une fois par Altavista et deux fois par Yahoo, il ne sera livré qu'une seule fois aux abonnés.

## Processus général

La version complète de MetaMAG se décompose de la façon suivante :

- vérification des bases de données et suppression des nouveaux résultats de la session précédente ;
- lancement des requêtes vers les moteurs de recherche (Altavista, Yahoo, etc.) et sauvegarde des nouveaux résultats ;
- suppression des résultats déjà trouvés ;

**Nicolas ESPOSITO**
*nik@niksnews.com*
Ingénieur de recherche chez Dassault Systèmes
*www.dsweb.com*
Auteur des Nik's News
*www.niksnews.com*




> **Figure 1 : page d'accueil de la version complète de MetaMAG**

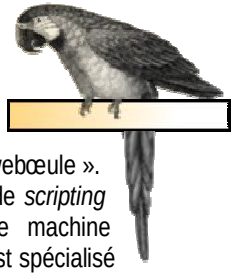
- test de validité des résultats restants (est-ce que les URL pointent vraiment sur des pages ?) ;
- copie en local des nouvelles pages ;
- génération des courriers électroniques et des pages d'archives ;
- distribution des courriers et publication des archives sur le site de MetaMAG ;
- copie des nouveaux résultats dans les anciens résultats pour ne plus les distribuer ;
- génération d'un historique et d'un fichier de statistiques.

Ce processus est basé sur un jeu de préférences qui permet notamment de choisir la langue (français ou anglais) et le système d'exploitation utilisé (Unix ou Windows).

## Choix des langages

Certains se diront que Perl est le langage idéal pour réaliser l'intégralité de cet outil. Mais ce serait sans compter sur l'excellent travail de Hannes MARAIS, des laboratoires de recherche de Digital (maintenant Compaq), qui a mis au point un langage de programmation spécialement destiné à ce type de tâches : WebL.
Perl se révélera quant à lui d'une aide précieuse pour l'interface utilisateur puisque c'est dans le cadre d'un navigateur HTML que nous allons gérer nos requêtes. Nous allons donc profiter de ses librairies pour la programmation CGI.

## WebL, le langage du Web



WebL se prononce « webœule ».
Il s'agit d'un langage de *scripting* qui tourne dans une machine virtuelle Java. WebL est spécialisé dans les applications du Web. Il connaît HTTP, FTP, HTML, XML et il les manipule avec une déconcertante facilité. WebL propose dans ce cadre deux innovations : les combinateurs de services (*Service Combinators* : exécution séquentielle ou concurrente, *Time-out*, répétitions et attentes) et une algèbre des balises (*Markup Algebra* : manipulation « native » des pages).

Outre ces innovations, WebL dispose de caractéristiques très intéressantes : syntaxe simple (entre le C et le Pascal), types de données dynamiques (une variable peut changer de type), programmation orientée objet dynamique (on peut ajouter ou supprimer des propriétés à un objet) avec possibilité de réflexion (le nom d'une propriété ou d'une méthode peut être calculé), manipulation des listes, des chaînes de caractères et des fichiers, accès direct aux classes Java, etc.

Pour profiter de tout ça, vous devez avoir accès à une machine virtuelle java et WebL.jar doit faire partie de votre variable d'environnement CLASSPATH, exemple sous Unix :

```
export CLASSPATH=
   $CLASSPATH:/opt/webl/WebL.jar ↵
```

Exemple sous Windows :

```
set CLASSPATH=
   %CLASSPATH%;C:\webl/WebL.jar ↵
```

Lancement d'un script WebL :

```
java WebL monscript.webl ↵
```

Exemple de script simple (récupération d'une page Web et affichage de son texte) :

```
var Page = GetURL("http:// www.altavista.com/"); ↵
PrintLn(Text(Page)); ↵
```

## Implémentation

Nous vous présentons donc ici une version minimale de MetaMAG :

- **Check** : vérification des bases de données et suppression des nouveaux résultats de la session précédente ;
- **Research** : lancement des requêtes et sauvegarde des nouveaux résultats ;
- **Select** : suppression des résultats déjà trouvés ;
- **Distribute** : génération et distribution des courriers ;
- **Integrate** : copie des nouveaux résultats dans les anciens résultats pour ne plus les distribuer.

La première chose à faire est de se consistuer un petit environnement d'exécution comme dans l'exemple suivant pour Windows (fichier environnement.bat) :

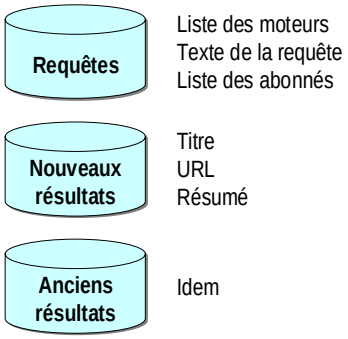
```
@echo off ↵
echo Environnement MetaMAG (JDK + WebL) ↵
set PATH=%PATH%;C:\java\bin ↵
set CLASSPATH=
   C:\java\lib\classes.zip;C:\webl\WebL.jar ↵
```

Ce fichier est à modifier en fonction de votre configuration logicielle (emplacements de Java et de WebL). Recopiez le répertoire metamag du CD-ROM sur votre disque dur et mettez à jour les fichiers environnement.bat et preferences.webl (notamment pour le serveur de mail). La modification de ces deux fichiers permettra aussi d'adapter MetaMAG à n'importe quel type de système d'exploitation disposant d'une machine virtuelle Java.
Il nous faut ensuite un script de lancement (fichier metamag.bat) :

```
@echo off ↵
echo Lancement de MetaMAG... ↵
java WebL metamag.webl ↵
```

Et voici l'exemple d'une session complète d'exploitation que l'on peut bien sûr automatiser :

- lancement de la ligne de commande (icône MS-DOS) ;
- changement de répertoire courant, par exemple : « cd metamag » ;
- lancement du script environnement.bat pour positionner les variables nécessaires à l'exécution ;
- lancement du script metamag.bat.




> **Figure 2 : bases de données principales**

Le fichier minimal de préférences (preferences.webl) se présente ainsi :

```
export var Directory = "C:\\metamag\\data"; ↲
export var Slash = "\\"; ↲
↲
export var MailServer =
   "mail.your-domain.com"; ↲
export var MailPort = 25; ↲
export var Administrator =
   "administrator@your-domain.com"; ↲
↲
export var Requests = Directory + Slash +
   "requests"; ↲
export var NewResults = Directory + Slash +
   "newresults"; ↲
export var Results = Directory + Slash +
   "results"; ↲
export var Directories = [NewResults] + [Requests]
   + [Results]; ↲
↲
export var ToBeReplaced = " %/:?&*|<>\\\""; ↲
export var ReplaceWith  =
   "@@@@@@@@@@@@"; ↲
```

Les variables qui y sont exportées pourront être utilisées dans d'autres modules (par exemple : preferences_Directory). ToBeReplaced et ReplaceWith sont utilisées par une fonction du module utilitaire (fichier tools.webl) pour remplacer tous les caractères *interdits* dans les noms de fichier.

## Exécution des modules

Le script de départ (fichier metamag.webl) va simplement se charger d'appeler un à un les différents modules de MetaMAG, par exemple pour le module de recherche (importation du module puis appel à l'une de ses fonctions) :

```
import research;
research_Research();
```

## Vérification et ménage

Ce premier module (Check, voir **code 1**) propose dans un premier temps de vérifier l'existence des répertoires correspondants aux bases de données de la **figure 2**. Dans un deuxième temps, il supprime les entrées de la base « Nouveaux résultats » qui ont été créées lors de la précédente session. Pour cela, il parcourt chacun des sous-répertoires de la base et il y efface tous les fichiers. On remarquera les appels au module Files (Files_List, Files_IsDir, etc.) et l'instruction every qui permet d'exécuter une séquence de code pour chacun des éléments d'un ensemble (« pour chacun des répertoires, je fais ça »).

```
import Files;
import preferences;
import tools;

export var Check = fun()
  PrintLn("Verification de l'environnement");
  every Directory in preferences_Directories do
    if !Files_IsDir(Directory) then
      tools_Error("le repertoire " + Directory + " n'existe pas");
    end;
  end;
  PrintLn("Suppression des resultats de la session precedente");
  every Directory in Files_List(preferences_NewResults) do
    Directory = preferences_NewResults + preferences_Slash +
      Directory;
    if Files_IsDir(Directory) then
      every File in Files_List(Directory) do
        File = Directory + preferences_Slash + File;
        if Files_IsFile(File) then
          Files_Delete(File);
        end;
      end;
    end;
  end;
end;
```

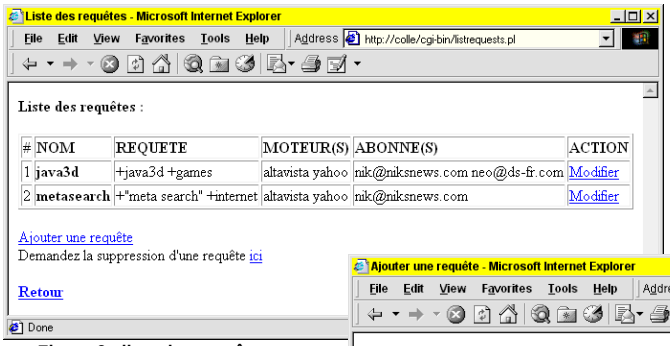
> **Code 1 : gestion de l'environnement (check.webl)**

## Au cœur de la recherche

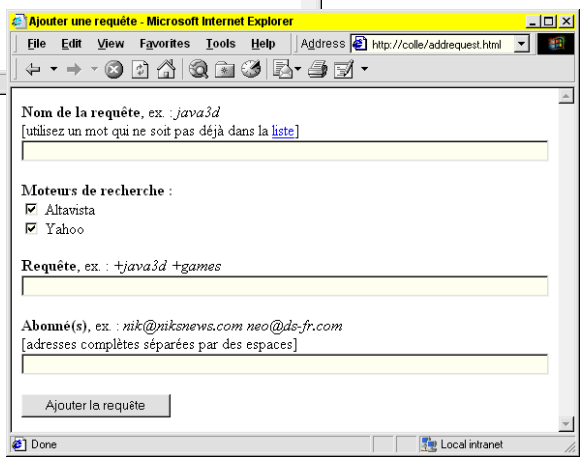
C'est dans le module Research (voir **code 2**) que l'activité de MetaMAG est la plus intéressante.

```
import Files;
import Str;
import Url;
import preferences;
import tools;

var Answers;
var Item;
var Title;
var URL;
var Abstract;

var Engines = [.
  altavista = [.
    Program = "http://www.altavista.com/cgi-bin/query",
    Parameters = fun(Request)
      [. pg = "q", q = Request .]
    end,
    Init = fun(Results)
      Answers = Elem(Results, "dl")
    end,
    Extract = fun(Answer)
      Item = Elem(Answer, "a")[0];
      Title = Text(Item);
      URL = Item.href;
      Abstract = Str_Trim(Text(Children(Elem(Answer, "dd")[0])[0]))
    end
  .],
  yahoo = [.
    Program = "http://search.yahoo.com/bin/search",
    Parameters = fun(Request)
      [. p = Request .]
    end,
    Init = fun(Results)
      Answers = Elem(Results, "li");
    end,
    Extract = fun(Answer)
      Item = Elem(Answer, "a")[0];
      Title = Text(Item);
      URL = Item.href;
      Abstract = Text(Answer);
    end
  .]
.];

export var Research = fun()
  PrintLn("Lancement des requetes");
  every File in Files_List(preferences_Requests) do
    var CompleteFile = preferences_Requests + preferences_Slash + File;
    if Files_IsFile(CompleteFile) then
      var Page = Files_LoadFromFile(CompleteFile, "text/html");
      var ListEngines = Str_Split(Text(Elem(Page, "p")[1]), " ");
      var Request = Text(Elem(Page, "p")[2]);
      every Engine in ListEngines do
        Print('.');
        var Results = GetURL(Engines[Engine].Program,
          Engines[Engine].Parameters(Request)) ? tools_Error(Engine +
          " n'est pas disponible");
        var Directory = preferences_NewResults + preferences_Slash + File;
        if !Files_IsDir(Directory) then
          Files_Mkdir(Directory);
        end;
        Engines[Engine].Init(Results);
        every Answer in Answers do
          Engines[Engine].Extract(Answer);
          var FileName = tools_GoodFileName(Str_Trim(URL)) + ".html";
          var Result = "<html><body>\n<p>Resultat de MetaMAG version " +
            "minimale</p>\n<p>" + Title + "</p>\n<p>" + URL + "</p>\n<p>" +
            Abstract + "</p>\n</body></html>\n";
          Files_SaveToFile(Directory + preferences_Slash + FileName, Result) ?
            tools_Error("impossible d'ecrire " + FileName);
        end;
      end;
    end;
  end;
  Print('\n');
end;
```

> **Code 2 : interfaces vers les moteurs et lancement des requêtes (research.webl)**




> **Figure 3 : liste des requêtes**




> **Figure 4 : saisie d'une nouvelle requête**

La première partie constitue la déclaration de l'objet Engines. Il contient un sous-objet pour chaque moteur de recherche avec lequel MetaMAG peut s'interfacer. Pour envoyer ses requêtes vers HotBot par exemple, il suffira d'ajouter à Engines le sous-objet hotbot et de modifier l'interface HTML.
Cette dynamicité est possible grâce à un traitement unique quel que soit le moteur de recherche. Ayant récupéré le nom du moteur dans la base des requêtes, on peut adresser son interface par réflexion de la façon suivante :

```
Engines["nom du moteur"].Méthode(...);
```

Par exemple, pour extraire les résultats de la page HTML Results sachant que Engine contient le nom du moteur :

```
Engines[Engine].Init(Results);
```

Pour chaque résultat, on génère un fichier qui sera enregistré dans la base des nouveaux résultats. WebL manipule si simplement les fichiers HTML avec son algèbre des balises que ce format a été choisi pour les requêtes et les résultats. L'objet Engines en est un bon exemple. Autre exemple, la fabrication d'une liste des moteurs associés à une requête :

```
var ListEngines =
    Str_Split(Text(Elem(Page, "p")[1]), " ");
```

« Je prends le deuxième paragraphe du fichier HTML de la requête [Page], j'en extrait le texte et je construis la liste en considérant les espaces comme des séparateurs. »

## Suppression et intégration

Les modules Select et Integrate sont assez simples (fichiers select.webl et integrate.webl). Select est chargé de la suppression des résultats déjà trouvés : il supprime les fichiers de la base « Nouveaux résultats » qui sont déjà présents dans la base « Anciens résultats ». Integrate est quant à lui chargé de la copie des résultats restants, dans les anciens : il ajoute les résultats de la base « Nouveaux résultats » à la base « Anciens résultats » (copie des sous-répertoires), ils ne seront donc plus distribués aux abonnés puisque le module Select les éliminera lors de la session suivante.

## Distribution

Le module de distribution mérite plus d'attention. Pour chaque requête pour laquelle il y a des nouveaux résultats, on récupère les informations qui la concernent dans la base des requêtes (moteurs, requête, abonnés). Puis, on ajoute au texte de l'e-mail en préparation les informations de chaque nouveau résultat. On envoie finalement l'e-mail grâce à la fonction SendMail du module Tools.
L'instruction suivante qui en est extraite illustre bien les capacités de WebL :

```
var SMTPsocket = Timeout(10000,
    Retry(Java_New("java.net.Socket", mailServer,
    MailPort))) ? Error("connexion impossible a " +
    MailServer);
```

En important le module Java (import Java), on a la possibilité de manipuler directement des objets Java. Ici : java.net.Socket pour établir une connexion avec le serveur d'e-mail.
La structure Timeout(10000, Retry(...)) ? Error(...) montre bien l'utilisation des combinateurs de services : « J'essaye jusqu'à ce que ça fonctionne [Retry]. Si au bout de dix secondes ça ne fonctionne toujours pas [Timeout], il y a erreur. ».
Voici un exemple d'e-mail de résultats :

```
Moteur(s) de recherche : altavista yahoo
Requete : +java3d +games
Abonne(s) : nik@niksnews.com neo@ds-fr.com

TITRE : java3d-interest: [java3d] Games in Java3d
URL : http://java.sun.com/products/java-media/mail-
archive/3D/2607.html
RESUME : java3d] Games in Java3d. Danielle
Rousy Dias da Silva (drds@di.ufpe.br) Tue, 11 May
1999 13:13:13 -0300 (EST)

TITRE : Javadomain.com - Java Games and Fun
URL : http://javadomain.com/java_mailing_list.html
RESUME : Java Games and Fun - The Javadomain
is a resource for Internet users that would like to
add Java Applets to their own Web sites. It curren-
tly houses  more than 200 free working Applets,
along with instructions for  downloading and
including them in other Web pages.

2 nouvelle(s) page(s) trouve(e)
```

## Une interface HTML

MetaMAG est une application qui fonctionne au sein d'un Intranet ou sur Internet. Son interface utilisateur est donc composée de pages HTML dont une partie est générée à l'aide de scripts Perl :

- index.html (voir **figure 1**, accès aux requêtes) ;
- listrequests.pl (voir **figure 3**, liste des requêtes) ;
- addrequest.html (voir **figure 4**, saisie d'une requête) ;
- addrequest.pl (enregistre une requête et demande l'affichage de la liste) ;
- modifyrequest.pl (modification d'une requête).

Ces fichiers se trouvent sur le CD-ROM, dans les sous-répertoires data/html et data/cgi-bin.

## Configuration du serveur Web

Afin que notre interface utilisateur soit accessible depuis un navigateur HTML, il faut la *référencer* auprès d'un serveur HTTP : où trouver les pages HTML, où trouver les scripts Perl et quelle extension ils auront.
Exemple des modifications à apporter au fichier conf/httpd.conf du serveur Apache sous Windows :

```
DocumentRoot "C:/metamag/data/html"
<Directory "C:/metamag/data/html">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>
ScriptAlias /cgi-bin/ "C:/metamag/data/cgi-bin/"
<Directory "C:/metamag/data/cgi-bin">
    AllowOverride None
    Options None
</Directory>
AddHandler cgi-script .pl
```

D'autre part, il faudra modifier la première ligne de chaque fichier .pl du sous-répertoire data/cgi-bin. Cette ligne doit faire référence à votre interpréteur Perl, par exemple : #!C:\mksnt\perl.exe.

## Utilisation

Une fois cette configuration effectuée et en automatisant le lancement quotidien de MetaMAG, l'utilisation de notre outil se fait donc via un navigateur HTML : abonnement à des requêtes existantes, création et modification de requêtes. Ensuite, il ne reste plus qu'à surveiller sa boîte aux lettres.

## Sans limite

En plus des différents traitements décrit dans le processus général et qui ne figurent pas dans cette version minimale (interfaces vers de nombreux moteurs, test de validité des résultats, copie en local des pages, gestion des archives, d'un historique et d'un fichier de statistiques), une infinité de fonctionnalités pourra être ajoutée au gré de vos besoins. Voici quelques exemples :

- calculer un score de pertinence des résultats ;
- régler la profondeur des recherches ;
- proposer d'autres sources de données qu'Internet ;
- permettre les requêtes privées.

## Conclusion

L'utilisation d'un tel outil montre à quel point le Web est vivant. On est impressionné par la vitesse de renouvellement de la première page de résultats des moteurs de recherche. En moyenne, un résultat sur dix est nouveau par rapport à la veille.
Un suivi au jour le jour est donc pleinement justifié. Mais le travail que cela représente est trop important si on se contente d'un navigateur HTML. C'est pourquoi MetaMAG pourra devenir votre compagnon de recherche idéal. ■

## Liens

WebL Home Page, *www.research.digital.com/SRC/WebL/*
Overview of Java Platform Product Family, *java.sun.com/products/OV_jdkProduct.html*
Perl, *www.perl.com*
Apache Project, *www.apache.org*