École Nationale Supérieure des Télécommunications de Bretagne
Projet de développement. Automne 2004.

Lundi 24 janvier 2005

MÉTA-CALENDRIER
AUTOMATIQUE POUR PORTAIL WWW

Rapport technique
Version finale

Groupe de projet P48

- ✓ Benjamin BOUILLIEZ
- ✓ Sanaa LIMOURI
- ✓ Said NABIRH
- ✓ Hai-Nam NGUYEN

Encadrant

- ✓ Ronan KERYELL

Rédacteur : Benjamin. Relecteur : Hai-Nam.

Objectif de ce projet

Le projet Méta-calendrier vise à élaborer un logiciel susceptible de centraliser des informations issues de divers sites et traitant d'événements informatiques. Ces informations, issues de sites différents seront homogénéisées, stockées dans une base de données indépendante, et pourront être consultées et interrogées via une interface Web.

Purpose of this project

The Meta-calendar project aims to create a programme able to centralise data providing from various websites giving informatics conferences calendars. The data is stored in our database and can be accessed via our website.

1. Table des matières

2. Table des illustrations

3. Introduction

Rédacteur : Benjamin Bouilliez. Relecteur : Hai-Nam Nguyen.

Il s'agit de mettre en œuvre un méta-calendrier automatique pour recenser des événements touchant à l'informatique.

Le but du projet est d'élaborer un logiciel capable d'extraire les informations à propos des évènements sur divers sites[1], de les placer dans une base de données qui pourra ensuite être interrogée par l'utilisateur final via une interface Web classique.

La mise à jour du calendrier sera faite deux fois par jour.

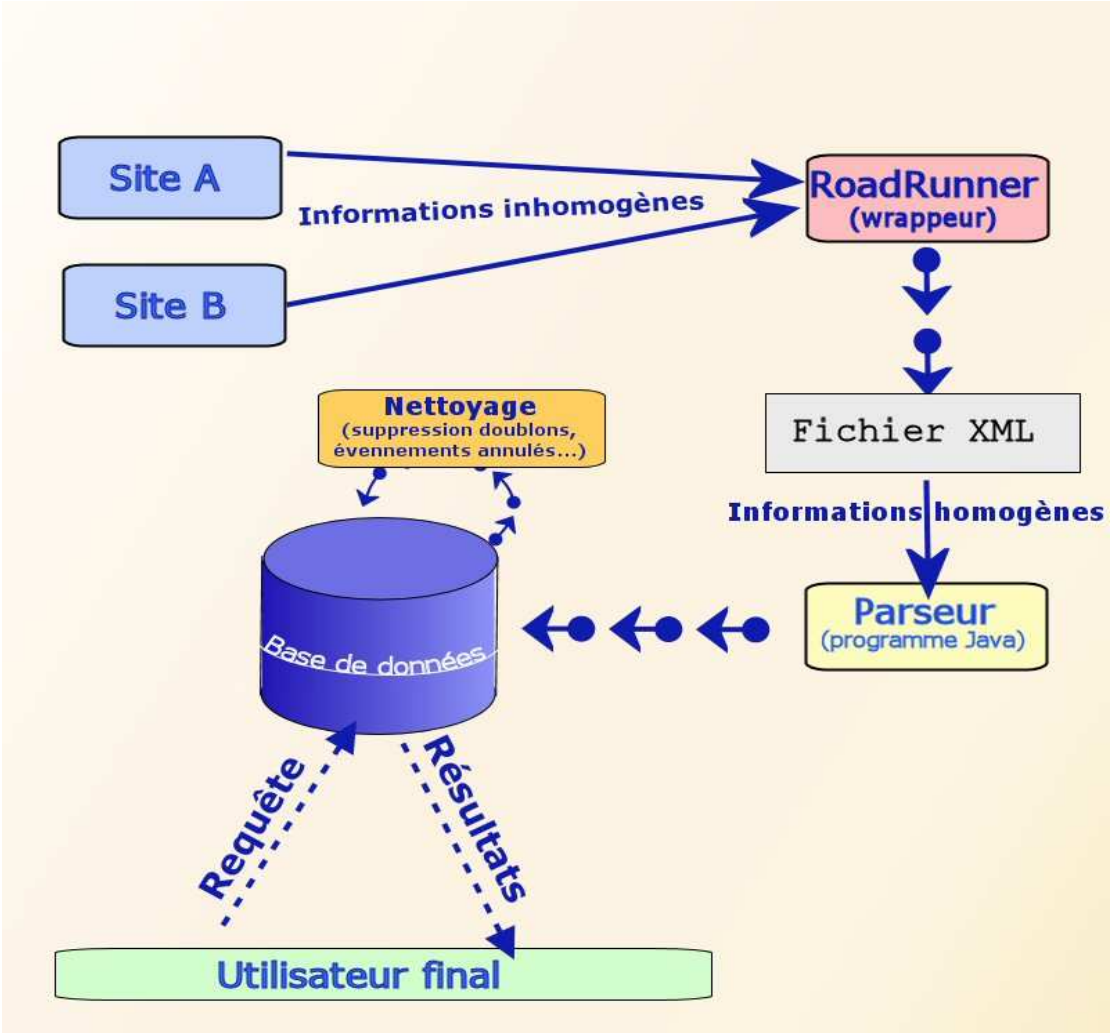


Figure 1. Schéma global du logiciel

[1] Pour la conception du logiciel, c'est le site ACM (Association for Computing Machinery) http://campus.acm.org/calendar/ qui a été analysé.

Nous décomposerons le travail du logiciel en quatre temps:

- La lecture des données issues des différents sites se fait grâce à un **wrappeur**[2] (ici RoadRunner[3]). Le wrappeur est à même de récolter les informations nous intéressant (date, lieu, description etc.).

  Nous avons choisi d'utiliser un wrappeur existant déjà dans le monde des logiciels libres: RoadRunner. Il livrera les informations récoltées dans un fichier XML[4].

- Le **parseur lira** ces données XML et les insérera dans la base de données.

- Un **nettoyeur** agira sur la base de données pour éliminer les événements qui figurent plusieurs fois, qui sont obsolètes ou qui ont été annulées.

- Enfin, l'utilisateur final pourra consulter les données et faire des recherches sur celles-ci via un **formulaire sur une page internet**.

[2] un programme étant capable d'extraire des morceaux d'information dans un document HTML et les rendre sous forme structurée (e.g. XML)

[3] un projet de deux universités italiennes, écrit en Java, licence GNU GPL. Voir l'annexe 2

[4] eXtented Markup Language, traduire *Langage à balises extensible*

4. Développement

4.1. Réalisation du wrappeur

Rédacteur : Hai-Nam. Relecteur : Benjamin.

4.1.1. Introduction à RoadRunner

Dans cette partie, nous cherchons une méthode simple pour extraire des informations (ici les événements) de différents sites web, puis les mettre dans une base de données.

HTML[5] est un langage de présentation, destiné à être lu et ne comprend donc aucune information explicitant le type de données manipulées : des informations de types différents peuvent être contenues dans des balises[6] de même natures. Il est donc difficile d'extraire les informations voulues (lieu, date, description) si nous ne connaissons pas la structure de page.

Heureusement, la plupart des sites comportant un grand nombre de données comportent des pages HTML produites à partir d'un ou de quelques modèles prédéfinis. Nous pouvons examiner plusieurs pages de même modèle, puis en déduire la structure de présentation des données, que l'on nomme la grammaire. À partir de la grammaire, nous disposons d'outils pouvant collecter les informations recherchées. Mais la construction de la grammaire reste la tache la plus ardue.

La déduction automatique de wrappeur étant bien développée sur le plan technique [Kush2000] , nous avons décidé d'utiliser le logiciel RoadRunner. Ce logiciel est en mesure de générer un wrappeur à partir de quelques pages de même structure. Il est également capable de repérer des répétitions dans un même fichier (une liste d'événements sur un site internet par exemple). C'est à partir de ces répétitions que RoadRunner structure les données contenues dans la page sous forme d'arbre pouvant ultérieurement être consulté, interrogé ou réorganisé.

RoadRunner fonctionne en suivant trois étapes :

- Un travail sur les informations fournies sur les sites en essayant de trouver les points communs au niveau de la structure de présentation.
- Une prise en compte de la connaissance partielle éventuelle de la structure des sites étudiés. Cette structure peut en effet être intégrée au logiciel par un administrateur de ce dernier.
- La génération d'un wrappeur en prenant en compte l'ensemble des informations.

La configuration est primordiale pour un fonctionnement optimal de RoadRunner lors de chaque étape. Cette configuration porte sur quatre parties que nous aborderons ci-dessous.

- La section « Tokenization » : définit la manière de convertir l'arbre DOM[7] en chaîne de tokens c'est à dire la méthode pour extraire les informations contenues dans l'arbre DOM pour pouvoir les utiliser sous forme de chaines de caractères.

---

[5] HyperText Markup Language : un langage avec des balises de formatage indiquant la façon dont doit être présenté le document.

[6] Balise: élément syntaxique du langage HTML spécifiant la mise en page des informations.

[7] Document Object Model : le modèle objet de document définit une hiérarchie d'objets, afin de faciliter leur manipulation [CCM].

- La section « Generation » : définit la manière avec laquelle le Wrappeur analysera la structure du site étudié [Dia2003].
- La section « Labelling » : donne un nom aux différents éléments du fichier XML en sortie du wrappeur
- La section « Output » : spécifie le fichier cible d'enregistrement des fichiers XML et XSLT[8]

4.1.2. Étapes de récupération des données

RoadRunner est constitué de deux classes :

- Shell : déduit le wrappeur (un fichier grammaire `wrapper.xml`) à partir de quelques pages d'un même site. Ces pages sont sauvegardées localement sur le disque.
- Wrapper : génère un fichier XML contenant les données grâce au wrappeur déduit ci-dessus. Cette classe reçoit en entrée une URL et récupère les données de cette URL via les méthodes de la classe `java.net.URL` [Haro1997, Ch. 5]. Ce travail est fait 2 fois par jour grâce à une tâche cron[9].

Nous avons étudié plusieurs sites qui fournissent des calendriers d'événements (cf. annexe). Dans ces sites, il y a 3 façons de présentation des événements :

- Une seule liste d'événements ;
- Une liste des événements sur la page principale, et des liens vers d'autres listes correspondant à différentes périodes de temps (dans le passé et dans le futur) ;
- Une liste d'événements sur la page principale, et présence d'un formulaire pour accéder aux autres listes.

Le premier cas est traité directement par le wrappeur, les deux derniers demandent un traitement de la page principale en utilisant les expressions régulières et la classe `java.net.URLConnection` [Haro1997, Ch. 10].

Chercher des liens dans une page web

```
<a[^<]*href=(["']?)(\S+)\1.*?>
```

L'expression ci-dessus cherche tous les liens encadrés par ' (apostrophe) ou " (guillemet anglais) ou rien. Les liens se trouvent dans la 2e parenthèse.

Convertir URL relatif en URL absolu

```
import java.net.*;
URL urlAbsolu = new URL(urlRéference, urlRelatif);
```

Soumettre un formulaire

```
import java.net.*;
import java.io.*;
try {
  URL u = new URL("http://www.truc.com/event.cgi");
  //établit la connexion, prépare l'envoie une requête POST
  URLConnection uc = u.openConnection();
```

[8] eXtensible Stylesheet Language Transformation : une recommandation permettant de transformer un document XML en document HTML accompagné de feuilles de style [CCM]

[9] Un démon sous UNIX/Linux qui effectue des tâches prédéfinies régulièrement

```
  uc.setDoOutput(true);

  DataOutputStream dos = new DataOutputStream(uc.getOutputStream());
  dos.writeBytes("mes données");
  dos.close;
}
catch (Exception e) {
  System.err.println(e);
}
```

L'exemple cité de [Haro1997, Ch. 10].

4.2. Le parseur XML

Rédacteur : Said. Relecteur : Sanaa.

4.2.1. Définitions

L'analyseur syntaxique (généralement dit parseur) est un outil logiciel permettant de parcourir un document et d'en extraire des informations.

Dans le cas de XML (on parle alors de parseur XML), l'analyseur permet de créer une structure hiérarchique contenant les données du document XML.

On distingue deux types de parseurs XML :

- *les parseurs « validants » (validating)* permettant de vérifier qu'un document XML est conforme à sa DTD.
- *les parseurs « non validants » (non-validating)* se contentant de vérifier que le document XML est bien formé (c'est-à-dire respectant la syntaxe XML de base).

Les analyseurs XML sont également divisés selon l'approche qu'ils utilisent pour traiter le document. On distingue actuellement deux types d'approches :

- Les API (Application Programming Interface) utilisant une approche **hiérarchique** : les analyseurs utilisant cette technique construisent une structure hiérarchique contenant des objets représentant les éléments du document (les nœuds), et dont les méthodes permettent d'accéder aux propriétés. La principale API utilisant cette approche est **DOM**.

  Javascript et ECMAScript utilisent DOM pour naviguer au sein du document HTML.

- Les API basés sur un mode **événementiel** permettent de réagir à des événements (comme le début d'un élément, la fin d'un élément) et de renvoyer le résultat à l'application utilisant cette API. **SAX** (*Simple API for XML*) est la principale interface utilisant l'aspect événementiel

Ainsi, on tend à associer l'approche hiérarchique avec **DOM** et l'approche événementielle avec **SAX**.

4.2.2. Comparaisons de DOM et de SAX

Les analyseurs utilisant l'interface DOM souffrent du fait que cette API impose la construction d'un arbre en mémoire contenant l'intégralité des éléments du document quelle que soit l'application.

Ainsi pour de gros documents (dont la taille est proche de la quantité de mémoire présente sur la machine) DOM devient insuffisant. De plus, cela rend l'utilisation de DOM lente, c'est la raison pour laquelle la norme DOM est généralement « peu respectée » car chaque éditeur l'implémente selon ses besoins, ce qui provoque l'apparition de nombreux parseurs utilisant des interfaces propriétaires...

Ainsi de plus en plus d'applications se tournent vers des API événementielles telles que SAX, permettant de traiter uniquement ce qui est nécessaire.

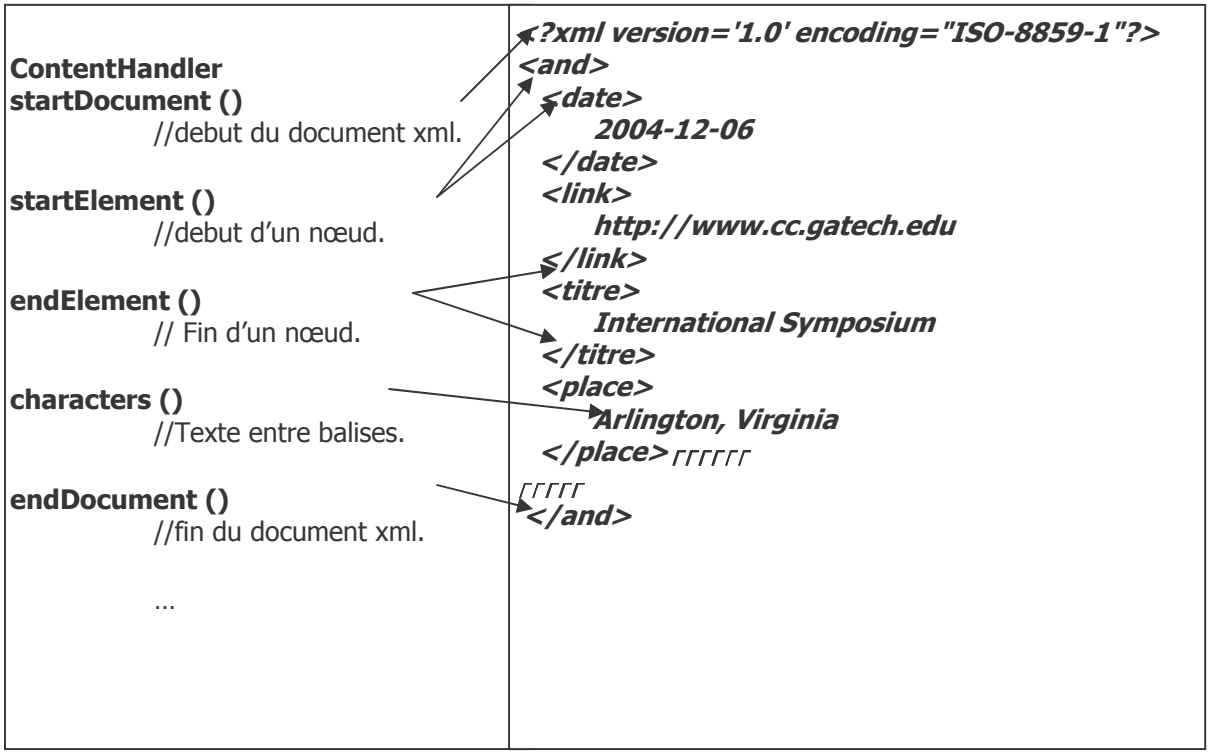
4.2.3. Choix du parseur

Le choix d'un parseur XML de type SAX non validant (le wrapper fournit un fichier XMLl bien formé) a été fais en tenant compte des raisons précédentes.

Comme il existe différents parseurs dans cette même classe on a choisi celui de **Xerces**, celui ci fait partie de la deuxième génération des parseurs SAX (sax.2.0)

4.2.4. Fonctionnement du SAX

Un analyseur SAX délivre les données du document à un processus au fur et à mesure de la lecture puis les traites selon les besoins de l'utilisateur.



Exemple :

Les méthodes du ContentHandler définies ci-dessus sont évoques a chaque fois que le parseur rencontre l'espace auquel elles font référence. Elles sont des méthodes prédéfinies en java dans le package Xerces, mais c'est l'utilisateur qui configure les traitements selon son application.

Ces méthodes entre autres sont toutes indispensables pour le parsing. L'oubli d'une d'entre elles provoque l'arrêt immédiat du parseur.

Si notre application ne nécessite pas une de ces méthodes, on a qu'à la définir avec un corps vide.

```
Public void characters (paramètres) {
    //aucun traitement.
}
```

Après définition de cette classe de traitement on commence le parsing du fichier xml en créant le parseur comme suit :

```
XMLReader saxReader =
XMLReaderFactory.createXMLReader("org.apache.xerces.parsers.SAXParser
");
saxReader.setContentHandler (new SimpleContentHandler () );
saxReader.parse(fichier xml);
```

Les méthodes du ContentHandler définies ci-dessus sont évoquées à chaque fois que le parseur rencontre l'espace auquel elles font références. Ces méthodes étant prédéfinie dans le package Xerces sont indispensable pour le parsing, elles offrent plus de liberté à l'utilisateur qui configure ses traitements selon son application cependant l'oublie de l'une entre elles provoque l'arrêt immediat du parseur.

Si notre application ne fait pas de traitement qui nécessite l'implémentation d'une méthode au sein de la classe il est possible de la définir avec un corps vide comme suit :

```
Public void characters (paramètres) {
    // aucun traitement.
}
```

Après définition de cette classe de traitement on commence le parsing du fichier XML en créant le parseur comme suit :

```
XMLReader saxReader =
XMLReaderFactory.createXMLReader(org.apach.xerces.parsers.SAXParser);
saxReader.seContentHandler(new SimpleContentHandler());
saxReadr.parse(fichier xml);
```

Ainsi le parsing est quasi configuré il reste alors à traiter les exceptions et les irrégularités.

4.2.5. Rôle de notre parseur

Notre parseur reçoit un fichier XML en entrée et doit fournir une liste java de type ArrayList() qui sera traitée par l'interface **SQL** (insertion dans la base de données) puis détaillée par la suite.

Le traitement du fichier XML consiste à extraire le texte entre les nœuds correspondant à des dates, des titres d'événements, des places, et des détails supplémentaires.

Ces textes sont soumis chacun au traitement qui lui convient pour n'en garder à la fin que l'essentiel selon les formats demandés.

Exemple :

La date sur le fichier XML est similaire à : ***October 31st-November 3rd*** alors que nous devons en extraire les deux dates suivantes : ***2004-10-31*** et ***2004-11-03***

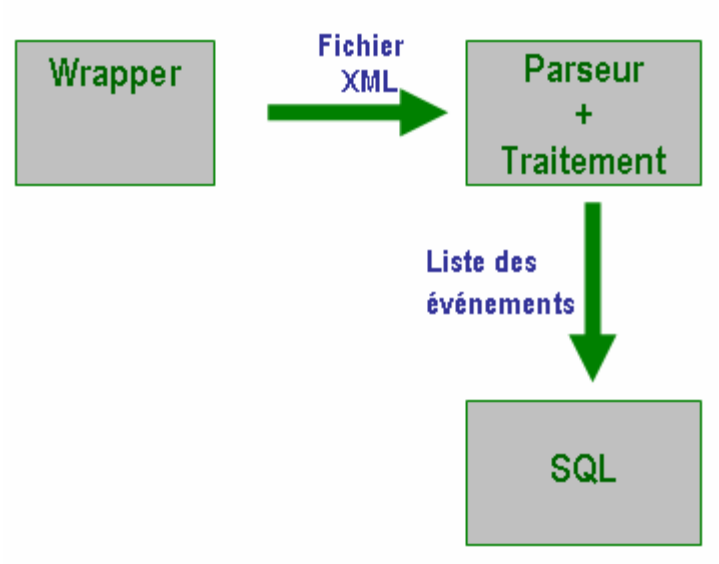


Figure 2. Le parseur XML

4.3. Le nettoyeur

Rédacteur : Benjamin Bouilliez. Relecteur : Said Nabirh.

4.3.1. Rôle et principe

Son rôle est de détecter et supprimer de la base de données toutes les données obsolètes (les événements qui ont déjà eu lieu, annulés (qui ne figurent plus dans le site où ils avaient été lus)) et les doublons (les entrées qui représentent le même évènement et qui ont été lues de deux sites différents).

Le nettoyage se fait périodiquement après chaque mise à jour de la base de données (après l'ajout de nouvelles entrées issues des sites).

La détection des évènements déjà passés ne pose pas de problèmes puisqu'elle se fait via une simple requête SQL sur la date de l'événement.

Pour ce qui est de la détection des événements annulés, voilà la démarche qui a été suivie:

- Lors de la collecte d'information, le wrappeur associe à chaque élément le site source d'une part dans le champ `url`, et la date à laquelle a été fait la collecte dans le champ `timeWrap` d'autre part ;
- Le nettoyeur détecte alors, parmi toutes les dates, la `timeWrap` la plus récente, et toutes les informations n'ayant pas été collectées lors de cette dernière sont donc des événements annulés ou ayant disparu du site.

La détection des doublons se révèle un peu plus compliquée puisque les informations provenant de sites différents, un même événement peut être décrit de deux manières différentes. Notre but sera donc de trouver une solution pour comparer les diverses descriptions et déterminer si un même événement est décrit plusieurs fois en utilisant des formulations différentes.

Par exemple :

Description A: Colloque « Autour du Libre » à l'ENST Bretagne le lundi 03 janvier

Description B: Le 3 janvier, colloque « Autour du Libre » (ENST-Bretagne).

Il sera aussi intéressant de faire cette démarche sur le champ décrivant le lieu des événements. Les champs `dateBegin` et `dateEnd`, quant à eux peuvent facilement être comparés d'un événement à l'autre.

De plus, les erreurs de frappes étant fréquentes dans les descriptions proposées sur Internet, la comparaison de chaînes lettre à lettre est trop contraignante.

Nous avons donc le choix de résoudre ces problèmes en ayant recours à la « fuzzy comparison ».

La comparaison des différents champs `detail` passe ainsi par les étapes suivantes :

1. Les deux séquences à comparer sont découpées mot par mot. Les phrases sont donc stockées dans des listes de mots de type `Vector ;`
2. Les mots « inutiles » tels que : le, la, de, avec,… sont éliminés car ils ne représentent pas des mots-clés. (Ces mots seront lus à partir d'un fichier nommé `motsInutiles.txt`) ;
3. Chaque mot de la première expression est alors comparé avec tous les mots de la seconde expression. A l'issue de ces comparaisons, seule la distance entre ce mot

et le mot le plus proche dans la seconde expression est conservée. Si le mot existe dans le deuxième texte la distance sera nulle ;

4. L'opération est répétée pour tous les mots du second texte et toutes les distances obtenues sont sommées.

La somme obtenue correspond alors à la distance entre les deux textes. De cette manière, si le deuxième texte contient les mêmes mots que le premier mais dans un ordre différent, la distance entre les deux expressions est nulle.

Voyons maintenant comment est faite la comparaison entre les mots.

4.3.2. Algorithme de Levenshtein

La distance de Levenshtein (LD) est une mesure de la similarité entre deux chaînes de caractères, une chaîne source (s) et une chaîne cible (t). La distance entre les deux chaînes de caractères représente le nombre de suppressions, insertions, ou remplacements qui permettent de passer d'une chaîne à l'autre.

- Si s est « examen » et t est « examen », alors LD (s, t) = 0, parce que il n'y a eu aucune opérations qui a modifie le mot source. Les deux chaînes sont identiques;
- Si s est « examen » et t est « examan », alors LD (s, t) = 1, parce qu'il a eu un remplacement (changement « e » à « a ») dans le mot source.

Plus nombre de différences entre les deux chaînes est grand et plus LD est grande.

Voyons maintenant comment se fait le calcul de cette distance sur un exemple :

Soit s = « CHIENS »

Soit t = « NICHE »

Intuitivement, on voit bien que l'on peut transformer la chaîne A en B en 5 étapes:

1. Suppression du S -> CHIEN
2. Suppression du N -> CHIE
3. Déplacement de I -> CIHE
4. Déplacement de I -> ICHE
5. Ajout du N -> NICHE

Le nombre d'étapes nécessaires au passage d'un mot à l'autre est appelé la distance de Levenshtein. Nous allons maintenant montrer sur le même exemple comment la trouver de manière plus formelle :

Soit n la longueur de la chaîne s (ici n=6)

Soit m la longueur de la chaîne t (ici m=5)

Si n=0 alors retourner d=m et quitter

Si m=0 alors retourner d=n et quitter

Construire une matrice M de m lignes et n colonnes.

Initialiser de la première ligne par la matrice ligne [0, 1, …, n-1, n] et la première colonne par la matrice colonne [0, 1, …, m-1, m]

M:

	C	H	I	E	N	S

	C	H	I	E	N	S
N	0	1	2	3	4	5
I	1					
C	2					
H	3					
E	4					

Soit Cout(i,j)=0 si s(i)=t(j) et Cout(i,j)=1 si s(i)!=t(i)

On a donc ici la matrice Cout:

	C	H	I	E	N	S
N	1	1	1	1	0	1
I	1	1	0	1	1	1
C	0	1	1	1	1	1
H	1	0	1	1	1	1
E	1	1	1	0	1	1

On remplit ensuite la matrice M en utilisant la règle suivante: M[i,j] est égale au minimum de:

1. L'élément directement avant plus 1: M[i-1,j] + 1;
2. L'élément directement au dessus plus 1: M[i,j-1] + 1;
3. Le diagonal précédent plus le cout: M[i-1,j-1] + Cout(i, j).

Voici le résultat de cette méthode appliquée à notre exemple:

	C	H	I	E	N	S
N	0	1	2	3	4	5
I	1	1				
C	2	2				
H	3	2				
E	4	3				

Nous réitérons cette opération jusqu'à remplir la matrice:

	C	H	I	E	N	S
N	0	1	2	3	4	5
I	1	1	2	3	4	5

	C	H	I	E	N	S
C	2	2	2	3	4	5
H	3	2	2	3	4	5
E	4	3	3	4	4	5

La distance de Levenshtein entre les mots s et t se retrouve en M[m,n]

Nous obtenons bien évidemment le même résultat que celui trouvé « intuitivement » (5).

4.4. L'interface pour MySQL

Rédacteur : Sanaa. Relecteur : Hai-Nam.

Une fois arrivées à ce niveau d'extraction et de filtrage, les données doivent être bien évidemment stockées quelque part pour que tout utilisateur puisse y accéder facilement et à n'importe quel moment par la suite.

L'utilisation des fichiers reste toujours possible malgré ses limitations puisqu'elle met en jeu des clients ayant connaissance de leurs organisation tout en sachant que l'accès n'est pas toujours fait de façon ponctuelle car le contenu de ces supports est souvent mal défini et mal désigné ce qui fait que le résultat n'est en aucun cas fiable.

Les données figurant à plusieurs reprises dans chaque fichier créent une certaine redondance et une grande occupation de la mémoire surtout quand on a affaire à des applications rigides, longues et coûteuses à mettre en œuvre.

La recherche de solutions possibles à ces problèmes a abouti au concept de base de données qui est une entité dans laquelle il est possible de stocker des données de façon structurée et avec le moins de redondances possibles. Le grand avantage apporté par cet outil c'est qu'il peut être utilisé par différents clients n'ayant peut-être aucune connaissance de la structuration du système d'informations.

Chaque base de données est fondée sur un support logiciel dit système de gestion de base de données (SGBD) qui permet de décrire, de modifier, d'interroger et d'administrer les données de la base.

Il existe quatre types de bases de données :

4. Bases de données hiérarchique : basées sur la modélisation arborescente des données ;
5. Bases de données déductives : basées sur la représentation des données sous forme de tables exploitées par un langage logique utilisant la logique mathématiques et la théorie des ensembles ;
6. Bases de données orientées objet : basées sur l'implémentation des objets qui vont être par la suite instancié sous forme de classes dont les attributs sont les champs des tables et les méthodes sont les opérations « envisagées » sur les données (requête SQL[10]…) ;
7. Bases de données relationnelles : basées sur la modélisation des données sous forme de tables reliées par des liaisons spéciales (association, CIF[11],…) et en utilisant un langage d'interpellation de haut niveau de type 3GL[12] (Visual Basic, ASP, Java, C++…).

Le choix d'utiliser une base de données relationnelle paraît évident puisque on a pas à gérer une grosse banque d'information ainsi les relations entre les tables constituant la base sont élémentaires vu leurs simplicité.

---

[10] Structured Query Language : langage de requêtes structuré, un langage informatique destiné à interroger ou piloter (modifier contenu et structure) une base de données

[11] cointrainte d'integrité fonctionnelle

[12] la troisième génération des langages (de haut niveau)

Comme SGBD on a choisit MySQL parmi Access, Oracle… pour les raison suivantes :

1. MySQL permet de construire facilement des bases de données relationnelles suivant les formes normales ;
2. Il permet la standardisation du langage SQL ;
3. Avec MySQL on n'est pas obligé de tous réinventer (le squelette de la base de données est déjà fait) ;
4. On retrouve dans MySQL les bibliothèques et les modules de plusieurs langages ;
5. SQL est stable autant que SGDB ;
6. Il offre une interface de travail standard ;
7. C'est un logiciel qui est utilisé par un large publique ce qui permet de retrouver facilement la documentation le concernant ;
8. Il est utilisé dans plusieurs domaines (pour les bases de données, pour le système de petites annonces, pour la gestion des utilisateurs d'un système informatique,…).

Cependant SQL n'est pas un outil parfait puisque il présente beaucoup de difficultés d'installation sur un système d'exploitation inconnu. La mise en œuvre de ce logiciel est en général complexe.

La nécessité d'utilisation d'un langage 3GL avec les bases de données relationnelles justifie le choix de Java pour interroger la base sous MySQL, grâce à la technologie JDBC[13] qui offre une interface de programmation en java permettant la connexion à la base de données.

Le JDBC étant relié à Java bénéficie des avantages de ce langage dont la portabilité du code et l'indépendance non seulement du SGBD mais également de la plate-forme.

Les étapes suivies :

1. Création de la base de données sous MySQL : voir [MySQL2004], chapitre 5.6 et 13.2

La base de données étant vide on crée les deux tables selon des permissions propres.

- ✓ `events` : tous événements traités

```
CREATE TABLE events (
  id int(11) NOT NULL default '0',
  datebegin date NOT NULL default '0000-00-00',
  dateend date NOT NULL default '0000-00-00',
  place varchar(100) NOT NULL default '',
  country varchar(50) NOT NULL default '',
  title varchar(255) NOT NULL default '',
  detail text NOT NULL,
  PRIMARY KEY  (id)
) TYPE=MyISAM;
```

[13] Java Database Connectivity : la technologie Java permettant de se connecter au n'importe quel SGBD

Pour la deuxième table, il faut 2 champs `url` et `timewrap` pour des raisons de nettoyage.

- ✓ `events_raw` : tous événements fournis par le parseur XML

```
CREATE TABLE events_raw (
  id int(11) NOT NULL auto_increment,
  url text NOT NULL,
  timewrap int(11) NOT NULL default '0',
  datebegin date NOT NULL default '0000-00-00',
  dateend date NOT NULL default '0000-00-00',
  place varchar(100) NOT NULL default '',
  country varchar(50) NOT NULL default '',
  title varchar(255) NOT NULL default '',
  detail text NOT NULL,
  PRIMARY KEY  (id)
) TYPE=MyISAM;
```

2. L'intégration de la base de données dans l'administrateur de source JDBC en sélectionnant le pilote pour la base et en donnant le nom de cette dernière ainsi que son chemin d'accès
3. La bibliothèque de classes JDBC se charge de la création d'une connexion à la base, l'envoie d'instructions SQL et l'exploitation des résultats provenant de la base.

On expose ci-dessous plus de détails à propos de l'implémentation avec le JDBC et Connector/J :

```
protected String url = "jdbc:mysql://localhost:3306/s3";
protected static String CONNECTION_NAME = "com.mysql.jdbc.Driver";
private Connection conn;
private Statement stmt;

// établir une connexion

try {
   Class.forName(CONNECTION_NAME).newInstance();
   conn = DriverManager.getConnection(url, user, pass);
   stmt = conn.createStatement();
} catch( Exception e ) {
   e.printStackTrace();
}

// requête

ResultSet rs=null;
try{
   rs = stmt.executeQuery(query);
   if (rs == null || !rs.next()) {
      return null;
   }
   else {
      return rs.getInt("id");  // en gros ☺
   }
} catch( Exception ex ){
   ex.printStackTrace();
}
```

Ce processus peut être schématisé comme suit :

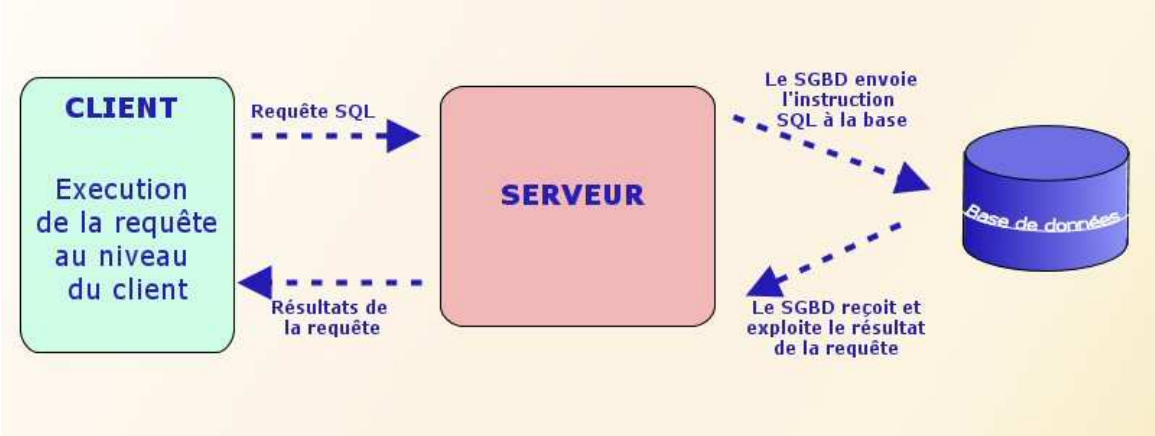


Figure 3. L'interface MySQL

4.5. L'interface web

Rédacteur : Hai-Nam. Relecteur : Benjamin.

Le Méta-calendrier est équipé d'une interface web réalisée en JSP[14] pour :

- L'administration : supprimer des doublons non détectés ;
- La recherche : chercher des événements par des critères multiples (date, lieu, sujet ...) ;
- L'affichage : lister des événements en cours et à venir.

L'utilisateur final à accès au module de recherche et au module d'affichage pour consulter la liste d'événements et faire des recherches sur celle-ci.

[14] Java Server Pages, un standard permettant de développer des applications Web interactives, c'est-à-dire dont le contenu est dynamique.

5. Conclusion

Rédacteur : Hai-Nam. Relecteur : Sanaa.

Le travail réalisé a été développé à partir d'un ancien projet. Les parties les plus importantes ont été complétées, il ne reste plus que des implémentations sous divers systèmes.

Le Méta-calendrier gère des données assez variées : des événements informatiques (l'exemple de base traitée dans le projet) ainsi que des événements sportifs, ou de musiques .Il gère également des calendriers ordinaires pour les différents départements d'une société ou d'une université par exemple en associant le tout dans un seul logiciel. L'implémentation du ce dernier est simple, effective et ne demande presque aucune action administrative lors de son lancement.

Cependant quelques problèmes sont apparus dans la partie du « wrappeur ». Même si il n'y a pas une grande difficulté dans la création des wrappeurs en général à partir des sites donnés (en cas d'ajout de nouveau site), il faut quand même un administrateur pour le faire. Cron est un outil qui ne marche que sous UNIX/Linux. En attendant que les machines Windows en bénéficient les chercheurs essayent toujours d'en développer un remplaçant qui puisse satisfaire aux besoins de Microsoft.

Les programmes qui ont servis dans le projet contribuent également au monde du logiciel libre en utilisant plusieurs outils libres. Outre, nous avons un autre objectif, c'est de joindre le domaine des services Web (faisable en cas d'utilisation du méta-calendrier pour les événements d'informatiques, ceux de sports ou d'actualités).

6. Bibliographie

Kush2000 KUSHMERICK, N. (2000) *Wrapper induction: Efficiency and expressiveness.* Artificial Intelligence J. 118(1-2):15-68 (special issue on Intelligent Internet Systems)

Dia2003 Valter CRESCENZI and Giansalvatore MECCA. *On Automatic Information Extraction from Large Web Sites*, RT-DIA-76-2003, Gennaio 2003 http://www.dia.uniroma3.it/db/roadRunner/publications/dia-76-2003.ps.gz

Haro1997 Elliotte Rusty HAROLD. *Programmation réseau avec Java*. Éditions O'REILLY, Paris, 1997, ISBN 2-84177-034-6

VLDB2001 Valter CRESCENZI, Giansalvatore MECCA et Paolo MERIALDO. *RoadRunner: Towards Automatic Data Extraction from Large Web Sites*. Proceedings of the 27th VLDB Conference, Roma, Italy, 2001.

JDBC2004 Richard G. BALDWIN. *Using JDBC with MySQL, Getting Started*. Disponible sur http://www.developer.com/java/data/article.php/3417381 (consulté le mardi 23 novembre 2004)

ConnJ3 Mark MATTHEWS. *MySQL Connector/J Documentation*. Manuel officiel pour Connector/J.
Disponible sur http://dev.mysql.com/doc/connector/j/en/index.html

CCM Encyclopédie informatique libre.
Disponible sur http://www.commentcamarche.net/

7. Annexe

7.1. Outils utilisés

- ✓ Java 1.5 : langage de programmation indépendant de plate-forme.
  http://java.sun.com
- ✓ MySQL 4.0.22 : système de gestion de base de données
  http://www.mysql.com
- ✓ MySQL Connector/J 3.0.16 : le pilote MySQL officiel, Java natif, pour JDBC
  http://www.mysql.com/products/connector/j/
- ✓ RoadRunner : générer des wrappeurs. Ses cibles sont plutôt des sites larges et pleins de données.
  http://www.dia.uniroma3.it/db/roadRunner/
- ✓ Xerces : Java parseur, utilisé par RoadRunner
  http://xml.apache.org/xerces2-j/
- ✓ NekoHTML : corrige des erreurs dans un document HTML, utilisé par RoadRunner
  http://www.apache.org/~andyc/neko/doc/html

7.2. Code source

Disponible sur le site du projet :

http://p48.gforge.priv.enst-bretagne.fr/