L3 année 2007 – 2008

PROJET JAVA

GESTION DES CONNAISSANCES

RECHERCHE DOCUMENTAIRE

Il s'agit d'écrire, de façon modulaire :
- un outil d'indexation des fichiers d'une arborescence donnée
- un outil d'export de cet index dans un format normalisé
- un outil de statistiques sur cet index
- un outil de recherche de document
- brancher votre outil de recherche dans une page html pour obtenir un moteur de recherche (via un script php qui vous sera fourni)

Une des but important d'un outil de gestion de connaissances est de permettre d'accéder de manière uniforme à des ressources variées.

En particulier, la possibilité d'effectuer des recherches sur des documents de formats différents est essentielle. Pour cela, une phase d'indexation permet d'associer des informations standards (liste de mots-clés, langue, ...) à chaque document, quelque soit son format (html, xml, texte simple, pdf, ...) et son origine géographique (fichier local, réseau local, web)

INDEXATION

La classe contenant le main s'appellera fr.umlv.km.Indexer

Pour indexer, on lancera simplement :

```
$ java -jar indexer.jar --build <racine de l'arborescence>
```

L'index sera stocké dans le sous-répertoire *.irkmindex* du répertoire indiqué par la variable d'environnement KM_HOME (par défaut, HOME de l'utilisateur).
Cet index pourra être constitué de un ou plusieurs fichiers (à vous de choisir).

Cette première phase va permettre d'extraire les informations suivantes de chaque document:
- liste de mots-clés avec le nombre d'occurrences de chacun de ces mots dans le document
- langue
- taille en nombre de mots et nombre de caractères

Les documents supportés seront reconnus par leur extension. Il vous est demandé de savoir traiter les :
- document TEXTE (supposé en ASCII ou en ISO-8859-1) : extension *.txt*
- document HTML : extension *.html*
- document XML : extension *.xml*
- document ZIP : extension . *zip* ces fichiers seront traités comme des répertoires
- document OpenDocument texte : extension *.odt*. Ces fichiers sont en fait des fichiers zip. On ne n'interessera qu'au fichier inclus *content.xml*

Optionnellement, vous pouvez supporter tout autyre type de document.

Dans tous les cas, vous devez prévoir qu'on puisse ajouter de nouveaux types de documents et vous devrez décrire dans la doc développeur comment faire.

Optionnellement, si l'indexeur trouvé par l'extension échoue, vous devez essayer de trouver l'indexeur correspondant au contenu réel du fichier.

Les mots-clés

- pour les fichiers HTML, on prendra tous les textes inter-balise
- pour les fichiers XML, on prendra tous les textes inter-balise
- pour les fichiers ASCII divers.
- pour les fichiers content.xml des fichiers ODT, on ne s'intéressera qu'au texte inter-balise des balises <text:title> et <text:p>
- pour les fichiers inclus dans les ZIP, on incluera simplement le nom du zip dans le path complet du document.

On prendra tous les mots (composés uniquement de lettres [a-zA-Z]) de 3 lettres ou plus. En particulier, les ' et – ne seront pas pris en compte (pas de mots-composés ni de l')

Appauvrissement typographique
Dans tous les cas, les lettres suivantes avec accent [é è ê à â ô û ù] seront assimilée à la même lettre sans accent. Idem pour le [ç] assimimé au [c].
cette liste n'est pas exhaustive, vous pouvez optinnellement la compléter.
Dans tous les cas, on se rendra insensible a la casse en sauvant tous ces mots-clés en minuscules.

La langue

- la détermination sera basique : on considérera le texte en français si un des caractères suivants est présent : é è ê à â ô û ù

sinon ça sera de l'anglais.

Les tailles

on comptabilise uniquement (pour chaque fichier) :
- le nombre de mots clés différents et le nombre total d'occurrences
- la taille (en octets) du fichier

AFFICHAGE DE L'INDEX

Attention, le format décrit ici devra être scrupuleusement respecté.

Pour affichage de l'index, on lancera :

```
$ java -jar indexer.jar --print <racine de l'arborescence>
```

Le format obtenu sera le suivant :
- 1 ligne par fichier
- les fichiers affichés avec le chemin relatif par rapport à la racine de l'arborescence, commençant tous par ./
- aucun espace superflu
- des ; comme séparateur de champs

- des , comme séparateur de mots-clés (uniquement en majuscules et donc sans accent)
- un point-virgule à la fin de ligne
- l'ordre d'affichage des fichiers est l'ordre ASCII

soit :
Fichier;type;langue;mots-clés;

avec :
- type devant être XML, HTML,ODT ou TEXT ou les autres types que vous supportez.
- langue devant être le code iso369-1. soit fr pour le Français, et en pour l'anglais.

MODULE STATISTIQUE

Ce module permet d'obtenir des informations générales sur l'index:

Attention, le format décrit ici devra être scrupuleusement respecté.

3 fonctions sont demandées

1/ obtention de statistiques par type de fichiers

```
$ j java -jar indexer.jar --stat --index <racine de
l'arborescence>  --type
```

affiche exactement (sans aucun espace ni CR superflu):

```
TOTAL:<nombre total de fichiers indexés>
html:<nombre de fichier de ce type>
text:<nombre de fichier de ce type>
xml:<nombre de fichier de ce type>
...
```

la liste doit être ordonnée par nombre décroissant de fichiers.

2/ obtention de statistiques par langue

```
$ java -jar indexer.jar --stat --index <racine de
l'arborescence>  --lang
```

affiche exactement (sans aucun espace ni CR superflu):

```
TOTAL:<nombre total de fichiers indexés>
en:<nombre de fichier de cette langue>
fr:<nombre de fichier de cette langue>
```

la liste doit être ordonnée par nombre décroissant de fichiers

3/ obtention de statistiques par mot-clé

Pour chaque mot-clé, on affichera le nombre total d'occurrences et le nombre de fichiers dans lesquels ce mot-clé a été trouvé.
Les mots-clés seront affichés dans l'ordre ASCII (rappel: tout en majuscule sans accent).

```
$ java -jar indexer.jar --stat --index <racine de
l'arborescence>  --word
```

affiche exactement (sans aucun espace ni CR superflu):

```
TOTAL:<nombre total de fichiers indexés>/<nombre total
d'occurrences>
mots-clé:<nombre total d'occurrences>/<nombre de fichiers>
mots-clé:<nombre total d'occurrences>/<nombre de fichiers>
....
```

RECHERCHE

Ce module permet d'obtenir d'effectuer des recherches sur l'index.

Attention, le format décrit ici devra être scrupuleusement respecté.

Pour lancer le programme :

```
$ java -jar indexer.jar --search --index <racine de
l'arborescence>   --req "<requête>"
```

note : la requête doit être donné comme un argument unique (donc avec des " s'il y a plusieurs mots):

```
$ java -jar indexer.jar --search --index /home/monnom/irkm
--req "reseau informatique"
```

ce module va effectuer une recherche basique.
Il va chercher la liste des fichiers qui "match" avec la requête.

1/ matching
les fichiers trouvés seront ceux contenant tous les mots de la requête
(penser bien sûr à l'appauvrissement typographique)

2/ score (nécessaire pour l'ordre d'affichage)
Chaque fichier trouvé aura un score correspondant à la somme du nombre d'occurrences de chaque mot de la requête présent dans le fichier.

3/ affichage
On affichera le nombre total de fichiers trouvés et la liste des fichiers (leur chemin complet depuis la racine de l'arborescence, commençant tous par ./) avec leur score (séparés par un ; sans aucun espace ni CR superflu).
Les fichiers seront affichés dans l'ordre décroissant du score. Pour les fichiers de même score, on utilisera l'ordre ASCII du chemin complet affiché.

TOTAL:<nombre de fichiers>
chemin_fichier1;<nombre total d'occurrences des mots de la requête dans ce fichier>
chemin_fichier2;<nombre total d'occurrences des mots de la requête dans ce fichier>
...

Intégration page WEB

Vous devez écrire une page PHP qui vous permettra de tester l'intégration de votre moteur de recherche en accès web.

Aide

Une option --help doit permettre d'obtenir l'aide pour toutes les options supportées par votre programme.

Quelques conseils pour démarrer ...

- Commencer par traiter les fichiers textes simples. Quand vous traiterez les fichiers XML ou HTML, seul la génération de l'index changera, mais l'affichage de l'index, les stats et la recherche sont indépendantes du type de fichier.
- Réfléchissez bien à la manière dont vous allez stocker votre index, aussi bien en mémoire que sur disque
- Ne vous occupez pas immédiatement de la détection de la langue.

Packaging ...

travail à rendre

☞ ATTENTION : Le programme sera testé à l'aide d'un jeu unique de scripts de tests. Aucune liberté avec la syntaxe définie n'est donc possible.

Références

- pour récupérer la liste des fichiers voir la classe *java.io.File*
- pour parser les fichiers XML, vous utiliserez le parser SAX *javax.xml.parsers.SAXParser*
- pour manipuler les fichiers ZIP et ODT, regarder la classe *java.util.zip.ZipFile*
- pour le html, utilisez *javax.swing.text.html.parser.Parser* (nous savons qu'il n'y a pas de doc !)
- pour les codes ISO639-1, voir http://www.loc.gov/standards/iso639-2/php/English_list.php
- pour les types mime : http://filext.com/