Dr. Laura Elena RAILEANU

Florian Poulin

# Spider Web

# Laboratoire Nº 5

## A. Objectifs

Dans ce laboratoire, vous allez implémenter un moteur de recherche Web. Dans un premier temps, vous implémenterez un robot Web, et l'intégrerez ensuite dans votre système de recherche d'informations basé sur le modèle vectoriel. Plus précisément, les points étudiés dans ce laboratoire seront :

- L'implémentation d'un robot Web ;
- L'intégration de ce robot dans le moteur de recherche implémenté dans les laboratoires précédents.

## B. Références

Cours «Recherche d'Information Multimédia » de Laura Raileanu.

## C. Rapport

**A remettre :** Vos sources et un rapport comportant les objectifs, la description des démarches adoptées les résultats obtenus (comprenant les points demandés dans la donnée) et une conclusion personnelle.

**A remettre :** Au plus tard au début de la séance du 19 Décembre 2007.

# D. Donnée

## 1. Généralités

Dans ce laboratoire, vous allez **remplacer la classe CACMFeeder du Framework de RIM par une implémentation récoltant ses informations sur le Web**. Pour l'indexation et la recherche, vous réutiliserez le travail que vous avez réalisé jusqu'au laboratoire n°3.

Utilisez la classe utilitaire `WebParser` fournie avec la donnée de ce laboratoire pour implémenter l'algorithme de Spidering présenté en théorie. Cette classe fournit une méthode statique `parseURL(…)` qui s'utilise comme suit :

```
WebParser.parseURL(new URL("http://www.heig-vd.ch/"));
```

L'invocation de cette méthode retourne un objet de type `WebParser.ParsedData`, qui contient des données reçues de l'URL. Ces données sont énumérées ci-dessous :

- `StatusCode` : `200` (tout va bien), `404` (la page ne peut pas être trouvée), … ;
- `ContentType` : `text/html` (page web), `application/pdf` (document pdf), … ;
- `PageContent` : Contenu de la page web sous forme d'une longue chaîne de caractères contenant le texte affiché sur la page (sans les balises `HTML`) ;
- `PageHrefs` : Références trouvées dans la page Web, un `Set<String>` contenant les valeurs de l'attribut `HREF` pour chaque lien rencontré dans la page.

Notez que les deux dernières données (`PageContent` & `PageHrefs`) ne sont définies que si l'URL désigne une page Web (`text/html`) et que le code de statut vaut `200`. Dans tous les autres cas, elles possèdent la valeur nulle.

Toutes ces valeurs peuvent être récupérées via des accesseurs publiques. Ci-dessous se trouve un exemple complet d'utilisation (sans la gestion des exceptions) :

```
WebParser.ParsedData data = WebParser.parseURL(new URL("http:// … "));
System.out.println("Status code  : " + data.getStatusCode());
System.out.println("Content type : " + data.getContentType());
System.out.println("Page content : " + data.getPageContent());
System.out.println("Page hrefs   : " + data.getPageHrefs());
```

## 2. Algorithme de parcours

Pour l'implémentation de votre *Feeder*, voici **quelques consignes dont vous devrez impérativement tenir compte** :

- Assurez-vous que votre Spider Web reste dans le domaine de l'HEIG-VD (ceci incluant évidemment tous les sous-domaines) ;
- Démarrez votre parcours depuis la page http://www.heig-vd.ch ;
- Visitez les pages en utilisant l'algorithme de parcours vu en théorie ;
- Maintenez les URL's visitées en vous assurant que :
  - Elles font partie du domaine de la HEIG-VD ;
  - Elles n'ont pas déjà été visitées (afin d'éviter les duplicata et les cycles). Pour ce faire, utilisez une fonction de Hachage (MD5 ou SHA) pour récupérer l'empreinte du contenu de la page et stockez cette empreinte (car deux liens différents peuvent pointer sur la même page) ;
  - Elles sont toutes dans un format approprié (pas de lien relatif, pas d'ancre, canonisation).
- Pour chaque URL visitée, indexer son contenu en appliquant les traitements précédemment implémentés (division en *tokens*, suppression des mots vides, indexation de tous les termes). Un fichier contenant une liste de *mots vides francophones* est fourni avec la donnée.

## 3. Questions pratiques

Finalement, ci-dessous se trouve une liste de questions pratiques (certaines nécessitant des interventions dans votre code).

**Les réponses aux questions suivantes devront être affichées dynamiquement sur la console** (et mentionnées dans le rapport) :

- Quel est le nombre total de pages indexées par votre robot ? Quels sont les sous-domaines rencontrés par votre robot, et combien de pages chaque sous-domaine contient-il ? Mettez en évidence les trois sous-domaines contenant le plus de pages (ex : gaps.heig-vd.ch, age.heig-vd.ch, …).

- Combien de termes uniques sont indexés ?
- Quelles sont les pages Web qui contiennent des liens morts (liens vers des pages inexistantes) ? Afficher l'URL de la page et les liens erronés.

Testez votre moteur de recherche avec les requêtes suivantes, et **présentez les résultats dans votre rapport** (uniquement). Pour chaque point, donnez le nombre d'URL's retournées par votre moteur de recherche et jugez la pertinence des deux premiers résultats :

- Offres d'emploi
- Inscriptions et ingénierie
- Horaire TIC trimestre
- + formulez deux requêtes de votre choix

# E. Indications

- La classe `WebParser` est un utilitaire d'aide pour votre laboratoire. Utilisez-la en tant que tel, n'y ajoutez pas directement l'algorithme de Spidering (celui-ci devra figurer dans une autre classe que vous implémenterez complètement).
- Si vous constatez des manques ou des erreurs dans la classe `WebParser`, vous êtes autorisés à la modifier (voire même à la remplacer). Faites-en part à l'assistant ou au professeur en cas de modification, et parlez-en dans votre rapport ;
- En cas de doute, n'hésitez pas à poser des questions à l'assistant ou au professeur.