Bases de données documentaires et distribuées
Cours NFE04

Création d'un index Solr

Auteurs : Raphaël Fournier-S'niehotta, Philippe Rigaux, Nicolas Travers
prénom.nom@cnam.fr

Département d'informatique
Conservatoire National des Arts & Métiers, Paris, France

le cnam

Construction d'un moteur de recherche

- On va utiliser la base des films de MongoDB et l'indexer
- Dans Solr, un index s'appelle **core**
- Celui par défaut que l'on a utilisé s'appelait "collection1"
- On va en configurer un autre, "movies"

- Dans solr/example/solr :

```
cp -R  collection1 movies
cd movies
```

- éditer "core.properties" pour changer la propriété `name` à `movies`
- vider data/ :

```
rm -Rf data/*
```

- se placer dans le répertoire "conf"

Un document (exemple)

```
{
  "_id": "movie:57",
  "title": "Jackie Brown",
  "year": "1997",
  "genre": "crime",
  "summary": "Jacky Brown, hotesse de l'air, ...",
  "country": "USA",
  "director": "Quentin Tarantino",
  "actors": ["Robert De Niro", "Pam Grier", "Bridget Fonda",
                "Michael Keaton","Samuel Jackson"]
}
```

Schéma de l'index

- Le schéma donne la liste de tous les champs d'un doc Solr
- Nombreuses options :
  - type (numérique, entier)
  - possibilité de calcul de la valeur du champ à partir d'un autre
  - traitements divers sur les valeurs du champ

Squelette

```
<?xml version="1.0" encoding="UTF-8" ?>

<schema name="example" version="1.5">
 <!--  Liste des champs de l'index -->
<fields>
  <field name="_id" type="string" indexed="true" stored="true" required="true" />
  <field name="title" type="string" indexed="true" stored="true" required="true" />
  <field name="summary" type="text" indexed="true" stored="false" required="false" />
  <!-- A completer -->

  <!-- Un champ dans lequel on concatene les autres pour une recherche "plein-texte" -->
  <field name="text" type="text" indexed="true" stored="false"
    multiValued="true" />
  <copyField source="summary" dest="text" />
  <copyField source="title" dest="text" />

      <!-- Un champ "technique" requis par Solr/Lucene -->
  <field name="_version_" type="long" indexed="true" stored="true" />
</fields>

<!-- La cle d'acces a un document dans l'index -->
<uniqueKey>_id</uniqueKey>

  <!-- Configuration des types de champ -->
<types>
  <fieldType name="string" class="solr.StrField" />
  <fieldType name="int" class="solr.IntField" />
  <fieldType name="long" class="solr.LongField" />
  <fieldType name="text" class="solr.TextField">
    <analyzer>
      <tokenizer class="solr.StandardTokenizerFactory" />
              <filter class="solr.LowerCaseFilterFactory" />
    </analyzer>
  </fieldType>
</types>
 </schema>
```

Squelette de l'index

- la liste des champs, dans l'élement "fields", complétée par l'indication du champ de recherche par défaut ;
- le champ qui identifie le document Solr, dans l'élément "uniqueKey" ;
- la liste des **types de champ**, dans l'élément "types".

Note : Pour des besoins internes, tout schéma doit contenir un champ "_version_" défini comme ci-dessus.

Définition des types et de la clé

- Chaque type utilisé dans le schéma d'un index doit apparaître dans un des élements **fieldType** du fichier **schema.xml**
- Solr fournit tout un ensemble de types pré-définis qui suffisent pour les besoins courants ; on peut associer des options à un type
- Les options indiquent d'éventuels traitements à appliquer à chaque valeur du type avant son insertion dans l'index
- ex : type **text**
  - on lui définit un "analyseur" "StandardTokenizerFactory"
  - se charge de découper le texte en **tokens** pour une recherche plein-texte (détails plus tard)
  - retenir : cela permet d'indexer chacun des mots, et donc de faire des recherches sur toutes les combinaisons de mots
- L'élément **uniqueKey** permet de rechercher un document dans l'index par sa clé. Indispensable, ne serait-ce que pour savoir qu'un document est indexé

Définition des champs

```
<field name="_id" type="string" indexed="true" stored="true"
      required="true" multiValued="false" />
```

- Les attributs de l'élement XML caractérisent le champ
- Le **nom** et le **type** sont les informations de base
- Ensuite, divers attributs (souvent optionnels) :
  - **indexed** indique simplement que le champ pet être utilisé dans une recherche ;
  - **stored** indique que la **valeur** du champ est **stockée** dans l'index, et qu'il est donc possible de récupérer cette valeur comme résultat d'une recherche, **sans avoir besoin de retourner à la base principale** ; en d'autres termes, "stored" permet de traiter l'index **aussi** comme une base de données ;
  - **required** indique que le champ est obligatoire ;
  - enfin, **multivalued** vaut **true** pour les champs ayant plusieurs valeurs, soit, concrètement, un **tableau** en JSON ; c'est le cas par exemple pour le nom des acteurs.

Définition des champs

Les champs **indexed** et **stored** sont très importants

Toutes les combinaisons de valeur sont possibles :

- **indexed=true**, **stored=false** : on pourra interroger le champ, mais il faudra accéder au document principal dans la base documentaire si on veut sa valeur ;
- **indexed=true**, **stored=true** : on pourra interroger le champ, **et** accéder à sa valeur dans l'index ;
- **indexed=false**, **stored=true** : on ne peut pas interroger le champ, mais on peut récupérer sa valeur dans l'index ;
- **indexed=false**, **stored=false** : n'a pas de sens à priori ; le seul intérêt est d'ignorer le champ s'il est fourni dans le document Solr.

Définition des champs (suite)

Comment peut-on indexer un champ sans le stocker ?

- c'est notamment le cas pour les textes qui sont décomposés en **termes** : chaque terme est indexé indépendamment
- très difficile pour l'index de reconstituer le texte
- d'où l'intérêt de conserver ce dernier dans son intégralité, à part

C'est une question de compromis :

- **stocker** une valeur prend plus d'espace que **l'indexer**
- Dans la situation la plus extrême, on dupliquerait la base documentaire en stockant chaque document **aussi** dans l'index
- un stockage plus important dégrade les performances

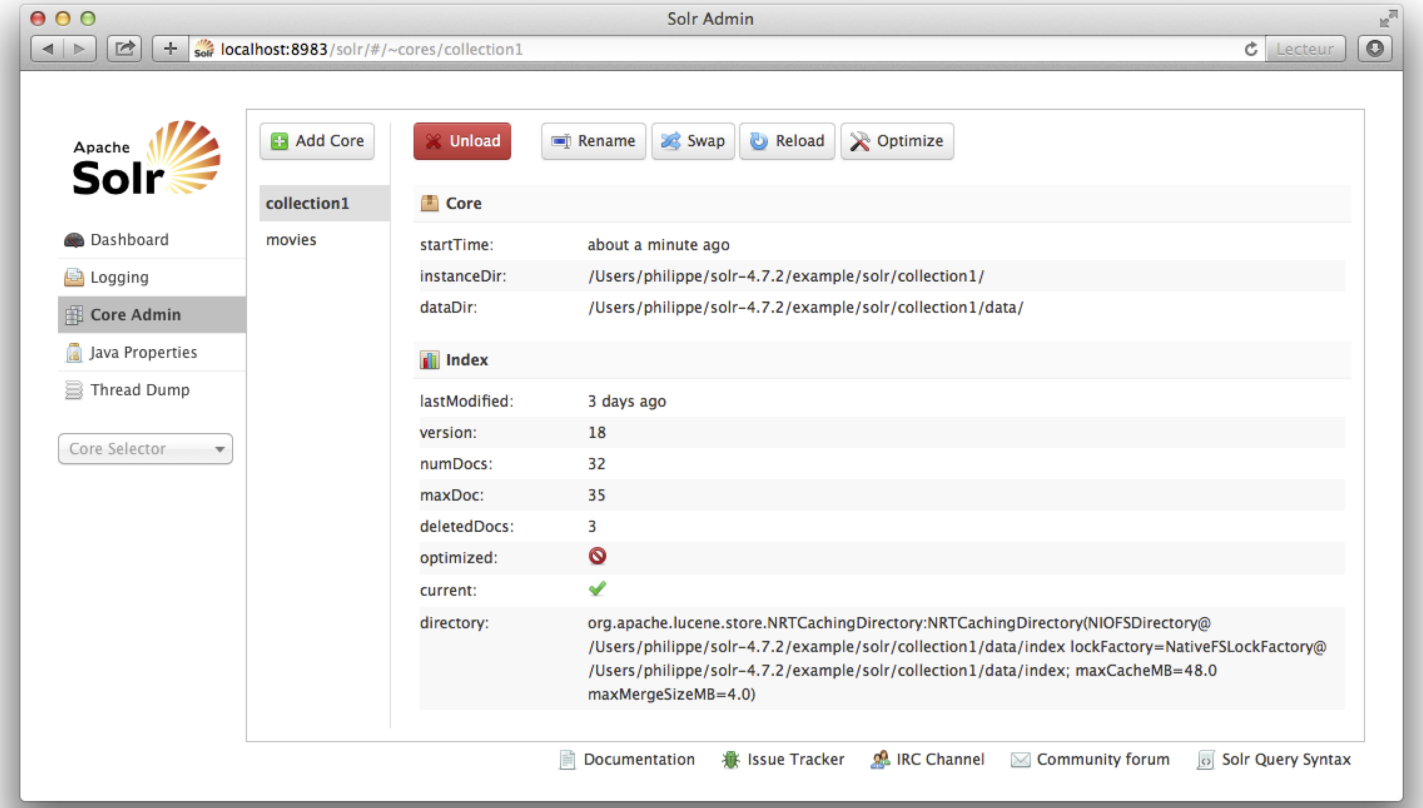
Le squelette de schéma comprend également un champ *calculé*, le champ **text**.

Les instructions **copyField** indiquent qu'au moment de l'insertion d'un document, on va "copier" certains champs dans celui-ci.

- le type du champ **destination** correspond à un mode particulier d'indexation, éventuellement différent et complémentaire de celui du champ **origine** ;
  => par exemple le contenu d'un titre est indexé comme une chaîne de caractères dans le champ **title**, et comme un texte "tokenisé" quand on le copie dans le champ **text** ;
- si **toutes** les occurrences de chaînes de caractères sont concaténées dans un même champ, on obtient, en prenant ce champ pour cible, une recherche plein-texte globale.

Recharger le schéma

- Après tout changement de schéma, il faut **recharger** l'index.
- Pour recharger un index, à partir de l'interface d'administration, utilisez l'option **Reload** après avoir sélectionné le **core**



Rechargement

```
curl "http://localhost:8983/solr/movies/update/json?commit=true" \\
--data-binary @solr_doc.json -H "Content-type:application/json"
```

- Attention, si l'index existant ne correspond pas au nouveau schéma, le rechargement échouera.
- Avec Solr, il est (plus) difficile de faire évoluer un schéma (qu'avec BDD classique)

Reconstruction (destruction puis validation) :

```
curl http://localhost:8983/solr/movies/update  \\
    --data '<delete><query>*:*</query></delete>' \\
    -H 'Content-type:text/xml; charset=utf-8'
curl http://localhost:8983/solr/movies/update \\
  --data '<commit/>' -H 'Content-type:text/xml; charset=utf-8'
```