

# Hibernate Search Lucene - Spring

L'objectif de cet article est de vous permettre d'appréhender la mise en place d'une indexation avec le moteur Lucene en s'aidant des frameworks java du moment sur le marché à savoir Hibernate et Spring.

Le projet de test que vous allez réaliser est téléchargeable à la fin de cet article, ce projet un projet maven donc pour le build : mvn package .Maven s'occupera de télécharger les dépendances pour vous .

**Hibernate Search** vous permet de profiter du moteur d'indexation **Lucene** pour indexer les objets de votre domaine métiers.

**Lucene** est basé sur une technologie d'indexation des contenus textuels similaire à Google ou d'autres moteurs de recherche. D'ailleurs, un sous-projet de Lucene, Nutch, offre les fonctionnalités de Google et permet de faire de la recherche sur des millions de pages web.

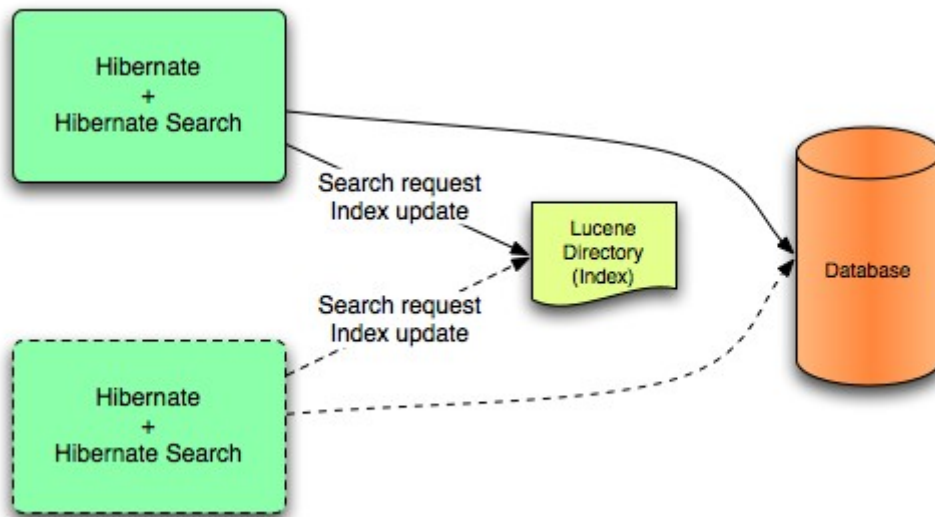
Lucene offre des performances inégalées et est scalable quelle que soit la volumétrie en terme de contenu ou de nombre de recherches.

**Spring** quand à lui est le meta-framework du moment, il intègre les concepts comme **AOP IOC MVC** et fourni un nombre considérable d'helpers pour l'intégrations de technologies, nous l'utiliserons dans notre projet pour manager hibernate et les transactions

L'intégration d'hibernate search est assez aisé et ce fait par le biais des annotations et les events hibernate.

Les annotations permettent de définir les propriétés que vous voulez indexer ainsi que leur stratégies d'indexations.

Les events s'occupent pour vous de synchroniser les indexes Lucene avec vos Entités hibernate



voilà après ce rapide tour d'horizon, nous allons mettre en place hibernate search pour un projet de test

## 1 Mise en place de MAVEN

Maven s'occupe de gérer pour vous les dépendances de vos projet et bien d'autres chose mais ce sera l'objet d'un prochain article.

Création de votre pom.xml avec les dépendances nécessaires :

```

<?xml version="1.0" encoding="UTF-8"?>
<project>
  <name>TutorielHibernateSearch</name>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.karlverger.tutoriel</groupId>
  <artifactId>hibernatesearch</artifactId>
  <packaging>war</packaging>
  <version>1.0.0-SNAPSHOT</version>
  <repositories>
    <repository>
      <id>repository.jboss.org</id>
      <name>JBoss Maven Repository</name>
      <url>http://repository.jboss.org/maven2</url>
      <layout>default</layout>
    </repository>
  </repositories>
  <build>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.5</source>
          <target>1.5</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
  
```

```

        </configuration>
    </plugin>
</plugins>
</build>
<dependencies>
    <dependency>
        <groupId>commons-dbc</groupId>
        <artifactId>commons-dbc</artifactId>
        <version>1.2.1</version>
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-annotations</artifactId>
        <version>3.3.0.ga</version>
    </dependency>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate</artifactId>
        <version>3.2.5.ga</version>
    </dependency>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-search</artifactId>
        <version>3.0.0.GA</version>
    </dependency>

    <dependency>
        <groupId>org.apache.lucene</groupId>
        <artifactId>lucene-core</artifactId>
        <version>2.2.0</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.5</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring</artifactId>
        <version>2.0.5</version>
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>2.5</version>
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-mock</artifactId>
        <version>2.0.5</version>
        <scope>compile</scope>
    </dependency>
    <dependency>

```

```

        <groupId>org.springframework</groupId>
        <artifactId>spring-remoting</artifactId>
        <version>2.0.5</version>
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-support</artifactId>
        <version>2.0.5</version>
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.1</version>
    </dependency>
</dependencies>

</project>

```

## 2 Configuration de descripteur WEB

Création du fichier WEB-INF/web.xml contenant les filtres et listeners nécessaires :

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems,
Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/applicationContext*.xml
        </param-value>
    </context-param>

    <!--
    filtre Spring permettant
    de mettre en place le pattern Session In View Hibernate
    -->
    <filter>
        <filter-name>Hibernate Session In View Filter</filter-name>
        <filter-class>
            org.springframework.orm.hibernate3.support.OpenSessionInViewFilter
        </filter-class>
    </filter>

    <!--
    permet d'appliquer le pattern
    session in view d'hibernate a tous nos controler
    -->

```

```

<filter-mapping>
  <filter-name>Hibernate Session In View Filter</filter-name>
  <url-pattern>*.do</url-pattern>
</filter-mapping>

<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>

<servlet>
  <servlet-name>web</servlet-name>
  <servlet-class>
    org.springframework.web.servlet.DispatcherServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>web</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>

<welcome-file-list>
  <welcome-file>
    index.jsp
  </welcome-file>
</welcome-file-list>
</web-app>

```

voilà, arriver à ce point nous avons un projet web exploitable, il nous reste à configurer hibernate ainsi que le transactionManager et les eventListeners pour la synchronisation entre votre domaine métiers et les indexes Lucene.

Pour ce faire nous allons créer le fichier WEB-INF/applicationContext-hibernate.xml

Le fichier est auto documenté vous devriez comprendre aisément chacun des noeuds déclarer

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:lang="http://www.springframework.org/schema/lang"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-2.0.xsd
http://www.springframework.org/schema/lang
http://www.springframework.org/schema/lang/spring-lang-2.0.xsd">

```

```

<!-- ===== HIBERNATE ===== -->
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
destroy-method="close">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
    <property name="url" value="jdbc:mysql://localhost/test"/>
    <property name="username" value="login"/>
    <property name="password" value="password"/>
</bean>

<bean id="sessionFactory"
class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">
    <property name="dataSource" ref="dataSource"/>

    <!--définition des beans hibernate-->
    <property name="annotatedClasses">
        <list>
            <value>com.karlverger.tutoriel.hibernatesearch.e
ntities.Item</value>
        </list>
    </property>

    <property name="hibernateProperties">
        <value>
            hibernate.cache.use_second_level_cache=false

            hibernate.dialect=org.hibernate.dialect.MySQLInnoDBDialect
            hibernate.show_sql=true
            <!--
d'hibernate-search
            les propriétés nécessaire à la mise en place
            pour le stockage des index
            -->
            hibernate.search.default.directory_provider =
org.hibernate.search.store.FSDirectoryProvider
            hibernate.search.default.indexBase
= /home/ubuntu/lucene/indexes
        </value>
    </property>
    <!--
réalise l'indexation
            en mode synchrones
            -->
    <property name="eventListeners">
        <map>
            <entry key="post-update">
                <bean
class="org.hibernate.search.event.FullTextIndexEventListener"/>
            </entry>
            <entry key="post-insert">
                <bean
class="org.hibernate.search.event.FullTextIndexEventListener"/>
            </entry>
            <entry key="post-delete">

```

```

        <bean
class="org.hibernate.search.event.FullTextIndexEventListener"/>
        </entry>
    </map>
</property>

</bean>

<!-- enable the configuration of transactional behavior based on
annotations -->
<tx:annotation-driven transaction-manager="transactionManager"/>

<!-- a PlatformTransactionManager is still required -->
<bean id="transactionManager"
class="org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory"/>
</bean>
<!-- /===== HIBERNATE ===== -->

<!--Beans DAO
-->
<bean id="itemDAO"
class="com.karlverger.tutoriel.hibernatesearch.dao.ItemDAO">
    <property name="sessionFactory" ref="sessionFactory"/>
</bean>

</beans>

```

Maintenant nous allons créer notre **POJO Item** pour notre test ainsi qu'un DAO correspondant.

Notre POJO Item va se voir ajouter des **annotations** permettant sa **persistance** en base et son **indexation** par le moteur Lucene. Le bean est auto documenté.

```

package com.karlverger.tutoriel.hibernatesearch.entities;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import org.hibernate.search.annotations.DocumentId;
import org.hibernate.search.annotations.Field;
import org.hibernate.search.annotations.Index;
import org.hibernate.search.annotations.Indexed;

/**
 * permet de déclarer le POJO comme entity à persister
 */

```

```

@Entity
/**
 * permet de déclarer à hibernate search qu'il faut prendre en compte
 * cette entité afin de l'indexer
 */
@Indexed
/**
 * permet de déclarer la table correspondante à ce pojo
 */
@Table(name = "item")
public class Item implements Serializable {
    /**
     * Anotation définissant cette propriété comme PK
     */
    @Id
    /**
     * Propriété spéciale d'hibernate search permettant d'assurer l'unicité
     * de l'index pour une entité
     */
    @DocumentId
    /**
     * Définit la stratégie de création de l'identifiant de la pk
     */
    @GeneratedValue(strategy=GenerationType.AUTO)
    /**
     * définit de manière explicite la colonne de base de données mappé avec
     * cette propriété
     */
    @Column(name = "ITEM_ID", nullable = false)
    private Long itemId;

    @Column(name = "NOM")
    /**
     * permet de définir la colonne comme source pour les index lucene
     * de l'entité, noté que tous les champs ne doivent pas être
     * TOKENIZED par exemple une date ne doit pas l'être ainsi que les
     * propriétés qui sont prévues pour effectuer des tris
     */
    @Field(index=Index.TOKENIZED)
    private String nom;

    public Long getItemId() {
        return itemId;
    }

    public void setItemId(Long itemId) {
        this.itemId = itemId;
    }

    public String getNom() {
        return nom;
    }

    public void setNom(String nom) {
        this.nom = nom;
    }
}

```

```
}  
  
}
```

Le fichier **ItemDAO** étend **HibernateDaoSupport** afin de bénéficier des facilités qu'apporte **Spring** pour la gestion de la **session hibernate**.

```
package com.karlverger.tutoriel.hibernatesearch.dao;  
  
import com.karlverger.tutoriel.hibernatesearch.entities.Item;  
import java.util.List;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import org.apache.lucene.analysis.standard.StandardAnalyzer;  
import org.apache.lucene.queryParser.MultiFieldQueryParser;  
import org.hibernate.search.FullTextSession;  
import org.hibernate.search.Search;  
import org.springframework.orm.hibernate3.support.HibernateDaoSupport;  
  
public class ItemDAO extends HibernateDaoSupport{  
  
    public void update(Item item) {  
        getHibernateTemplate().update(item);  
    }  
  
    public void create(Item item) {  
        getHibernateTemplate().persist(item);  
    }  
  
    /**  
     * effectue une recherche fullTexte Lucene sur les indexes  
     * défini dans le model  
     * Item, a savoir nom par rapport a la demande passé en parametre.  
     *  
     * Cette méthode permettra de tester l'indexation  
     *  
     * @param searchPattern  
     * @return  
     */  
    public List luceneSearch(String searchPattern) {  
        List result = null;  
        try{  
            FullTextSession fullTextSession =  
Search.createFullTextSession(getHibernateTemplate().getSessionFactory().getCurrent  
tSession());  
            MultiFieldQueryParser parser = new MultiFieldQueryParser(new  
String[]{"nom"}, new StandardAnalyzer());  
            org.apache.lucene.search.Query query = parser.parse(searchPattern);  
            org.hibernate.Query hibQuery =  
fullTextSession.createFullTextQuery(query, Item.class);  
            result = hibQuery.list();  
        }catch (org.apache.lucene.queryParser.ParseException ex) {  
            Logger.getLogger(ItemDAO.class.getName()).log(Level.SEVERE, null,  

```

```
ex);  
    }  
    return result;  
  }  
}
```

A présent nous pouvons effectuer un petit test unitaire pour valider notre mise en place

**pour rappel, hibernate est configuré pour pointer sur la base test sur localhost avec  
username : login  
password : password**

1 nous allons créer la table nécessaire

```
CREATE TABLE `test`.`item` (  
  `ITEM_ID` BIGINT(20) AUTO_INCREMENT,  
  `NOM` VARCHAR(250) ,  
  PRIMARY KEY(`ITEM_ID`)  
)  
ENGINE = InnoDB;
```

Maintenant nous pouvons créer notre test unitaire, celui ci s'appuie sur le framework spring afin de bénéficier de l'injection et du contexte transactionnel.

```
package com.karlverger.tutoriel.hibernatesearch.dao;  
  
import com.karlverger.tutoriel.hibernatesearch.entities.Item;  
import java.util.List;  
import  
org.springframework.test.AbstractTransactionalDataSourceSpringContextTests;  
  
public class ItemDAOTest extends  
AbstractTransactionalDataSourceSpringContextTests {  
  
    protected ItemDAO itemDAO;
```

```

public ItemDAOTest () {
    setPopulateProtectedVariables(true);
}

public void testCreate(){
    Item item = new Item();
    item.setNom("un nouvel item");
    getItemDAO().create(item);
    setComplete();//nécessaire pour que le test ne soit pas rollbacké en base
}

public void testLuceneSearch(){
    List<Item> items = getItemDAO().luceneSearch("un");
    System.out.println("Les items suivants ont été trouvés : ");
    for (Item item : items) {
        System.out.println(item.getNom());
    }
}

public ItemDAO getItemDAO() {
    return itemDAO;
}

public void setItemDAO(ItemDAO itemDAO) {
    this.itemDAO = itemDAO;
}

@Override
protected String[] getConfigLocations() {
    return new String[] {"applicationContext-hibernate.xml"};
}
}

```

ce test vous permet de créer un nouvel item en base, les listener hibernate search s'occuperont de l'indexer auprès du moteur Lucene, puis d'effectuer une recherche sur les indexes.

Voilà en espérant que ce tutoriel vous a été utile. Je rentrerais plus en détail sur les stratégies de mapping et d'indexation dans un prochain article.