University of Seville

School of Computer Engineering

Hibernate search

Software engineering

Design and testing 1

2018 – 2019

11/14/2018

Group 26

Index

1. Introduction

Hibernate search offers full-text search support for objects stored by Hibernate ORM, among its characteristics we found the followings:

- search words with text.
- order results by relevance.
- find by approximation (fuzzy search).

2. Deployment

2.1. Dependencies

We need to add a dependency to our maven project, to do this is needed to add the following lines to our **pom.xml** file:

```
<dependency>

        <groupId>org.hibernate</groupId>

        <artifactId>hibernate-search-orm</artifactId>

        <version>5.0.0.Final</version>

</dependency>
```

2.2. Properties

The next step is to add a couple of properties to your Hibernate configuration file.

The Hibernate configuration file is your **persistence.xml,** found in the route:

main/resources/META-INF

The properties are the followings:

```
   <property name="hibernate.search.default.directory_provider"
            value="filesystem"/>
   <property name="hibernate.search.default.indexBase" value="C:\Documents
and Settings\Student\lucene\indexes"/>
```

These properties indicate where hibernate search will store the indexes of our entities. The route can be changed as will.

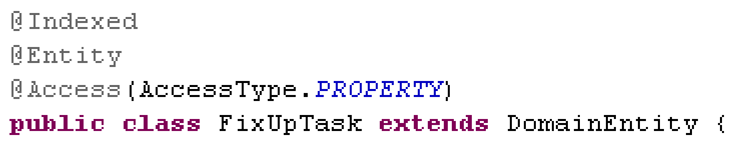
2.3. Annotations

We need to add a few annotations to our entities in order to search information in them. The annotations are:

- @Indexed: marks an entity which needs to be indexed by Hibernate Search.
- @Field: marks the fields you want to make searchable. @Field annotation has the following parameters:
  - index=Index.YES will ensure that the text will be indexed.
  - analyze=Analyze.YES ensures that the text will be analyzed using the default Lucene analyzer.
  - store=Store.NO, ensures that the actual data will not be stored in the index.

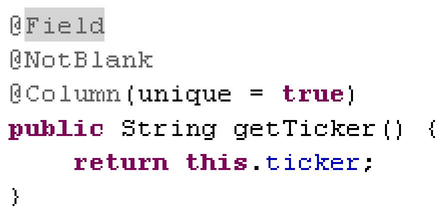
The Field parameters that were described are the default so can be omitted.

So, to index FixUpTask and search in Ticker, Description and address we need to add the followings annotations:



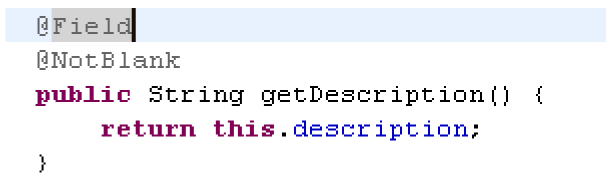
```
@Indexed
@Entity
@Access(AccessType.PROPERTY)
public class FixUpTask extends DomainEntity {
```

Figure 1. Annotations of FixUpTask



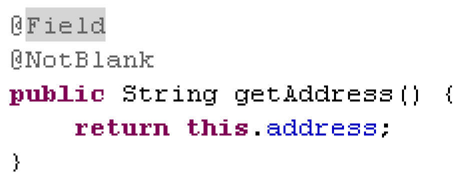
```
@Field
@NotBlank
@Column(unique = true)
public String getTicker() {
    return this.ticker;
}
```

Figure 2. Annotations of Ticker



```
@Field
@NotBlank
public String getDescription() {
    return this.description;
}
```

Figure 3. Annotations of Description



```
@Field
@NotBlank
public String getAddress() {
    return this.address;
}
```

Figure 4. Annotations of Address

2.4. Java search application

To achieve our search, we need to:

- Index the entities.
- Create a Lucene Query
- Wrap the Lucene Query into an Hibernate Query
- Execute the Hibernate Query

Our code is the following:

```
  public class Indexes {

        public static void main(String[] args) throws InterruptedException {


                EntityManagerFactory factory =
Persistence.createEntityManagerFactory("Acme-HandyWorker");

                EntityManager manager = factory.createEntityManager();

                FullTextEntityManager fullTextEntityManager =

        org.hibernate.search.jpa.Search.getFullTextEntityManager(manager);

                fullTextEntityManager.createIndexer().startAndWait();

                factory.close();

        }

  }
```

Code to index FixUpTask

```
public class Search {


        public static void main(String[] args) throws Throwable {


        final EntityManagerFactory factory =
        Persistence.createEntityManagerFactory( "Acme-HandyWorker" );
        final EntityManager manager = factory.createEntityManager();


        final FullTextEntityManager fullTextEntityManager=
           org.hibernate.search.jpa.Search.getFullTextEntityManager(manager);

                manager.getTransaction().begin();
// create native Lucene query unsing the query DSL


                final QueryBuilder qb =
        fullTextEntityManager.getSearchFactory().buildQueryBuilder()
        .forEntity(FixUpTask.class).get();


        ConsoleReader cr = new ConsoleReader();
```

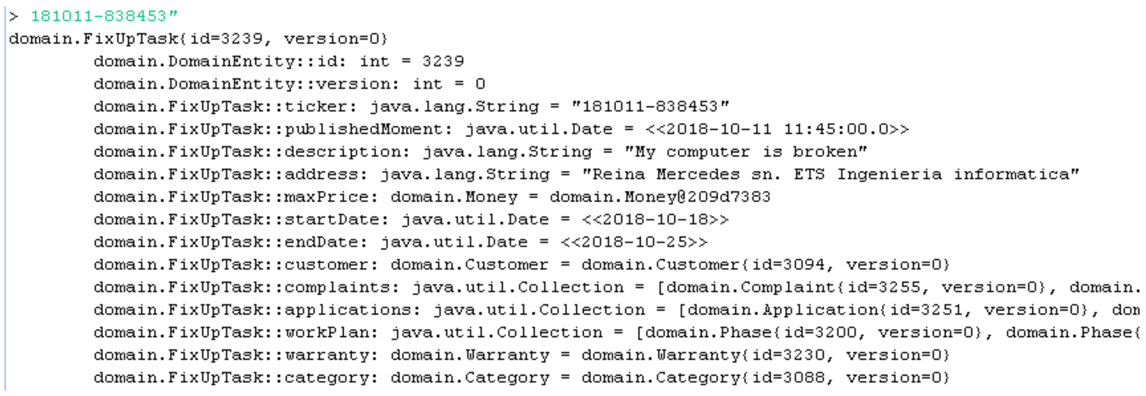
```
                while(true){
                try{
                        String match = cr.readLine();
                        if(match!=null && match.equals( "exit" ))break;

                final org.apache.lucene.search.Query luceneQuery = qb.keyword()
                .onFields( "description" , "ticker" , "address")
                .matching(match).createQuery();

                // wrap Lucene query in a javax.persistence.Query
final javax.persistence.Query jpaQuery =
                   fullTextEntityManager.createFullTextQuery(luceneQuery,
FixUpTask.class);
// execute search
@SuppressWarnings( "unchecked" )
                List<FixUpTask> result = jpaQuery.getResultList();

                if(result!=null && result.size()>0)
                        SchemaPrinter.print(result);
                else
                        System.out.println( "The term has not been found" );
                }catch (Exception e) {

                        System.out.println( "Enter a valid string" );

                                                }

                }


        //Finishing the application


        manager.getTransaction().commit();

        manager.close();

        factory.close();




        }


    }
```

Code of the search engine

2.5. Test

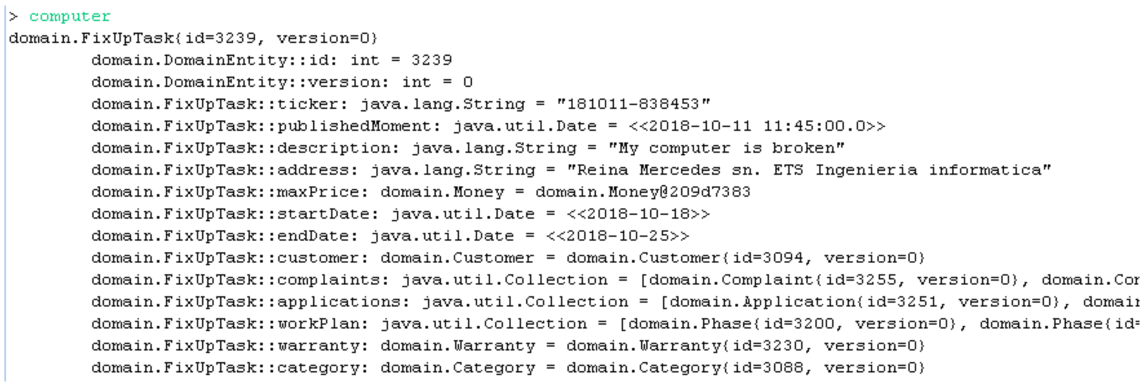
To test our application, we execute the indexes application and then in the search one tried the following cases:

1. Introduce a word that is contained in a ticker.
2. Introduce a word that is contained in a description.
3. Introduce a word that is contained in an address.
4. Introduce a word that is not contained in a ticker, a description or an address.
5. Introduce invalid String.



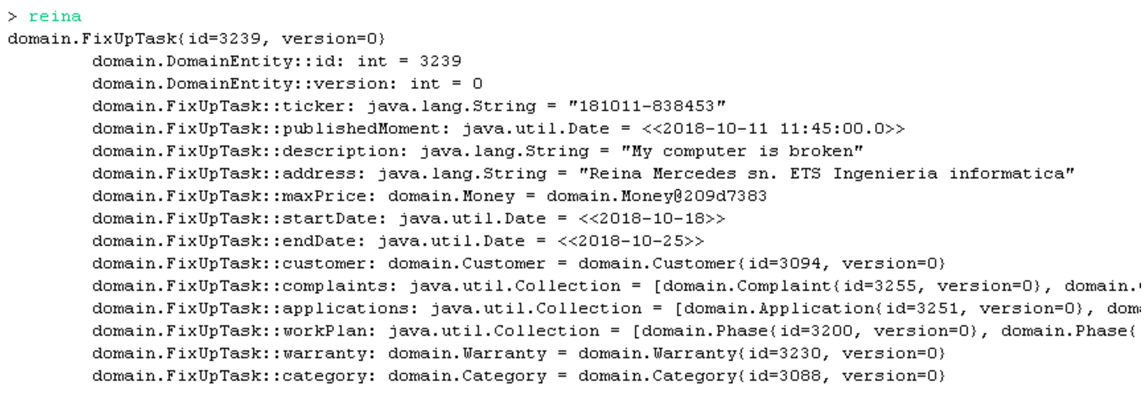
```
> 181011-838453"
domain.FixUpTask{id=3239, version=0}
        domain.DomainEntity::id: int = 3239
        domain.DomainEntity::version: int = 0
        domain.FixUpTask::ticker: java.lang.String = "181011-838453"
        domain.FixUpTask::publishedMoment: java.util.Date = <<2018-10-11 11:45:00.0>>
        domain.FixUpTask::description: java.lang.String = "My computer is broken"
        domain.FixUpTask::address: java.lang.String = "Reina Mercedes sn. ETS Ingenieria informatica"
        domain.FixUpTask::maxPrice: domain.Money = domain.Money@209d7383
        domain.FixUpTask::startDate: java.util.Date = <<2018-10-18>>
        domain.FixUpTask::endDate: java.util.Date = <<2018-10-25>>
        domain.FixUpTask::customer: domain.Customer = domain.Customer{id=3094, version=0}
        domain.FixUpTask::complaints: java.util.Collection = [domain.Complaint{id=3255, version=0}, domain.
        domain.FixUpTask::applications: java.util.Collection = [domain.Application{id=3251, version=0}, dom
        domain.FixUpTask::workPlan: java.util.Collection = [domain.Phase{id=3200, version=0}, domain.Phase{
        domain.FixUpTask::warranty: domain.Warranty = domain.Warranty{id=3230, version=0}
        domain.FixUpTask::category: domain.Category = domain.Category{id=3088, version=0}
```

Figure 5. Test 1



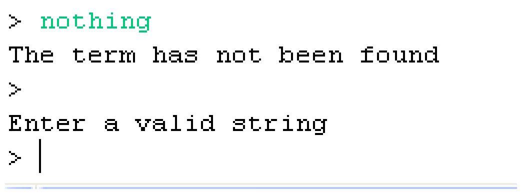
```
> computer
domain.FixUpTask{id=3239, version=0}
        domain.DomainEntity::id: int = 3239
        domain.DomainEntity::version: int = 0
        domain.FixUpTask::ticker: java.lang.String = "181011-838453"
        domain.FixUpTask::publishedMoment: java.util.Date = <<2018-10-11 11:45:00.0>>
        domain.FixUpTask::description: java.lang.String = "My computer is broken"
        domain.FixUpTask::address: java.lang.String = "Reina Mercedes sn. ETS Ingenieria informatica"
        domain.FixUpTask::maxPrice: domain.Money = domain.Money@209d7383
        domain.FixUpTask::startDate: java.util.Date = <<2018-10-18>>
        domain.FixUpTask::endDate: java.util.Date = <<2018-10-25>>
        domain.FixUpTask::customer: domain.Customer = domain.Customer{id=3094, version=0}
        domain.FixUpTask::complaints: java.util.Collection = [domain.Complaint{id=3255, version=0}, domain.Cor
        domain.FixUpTask::applications: java.util.Collection = [domain.Application{id=3251, version=0}, domai
        domain.FixUpTask::workPlan: java.util.Collection = [domain.Phase{id=3200, version=0}, domain.Phase{id=
        domain.FixUpTask::warranty: domain.Warranty = domain.Warranty{id=3230, version=0}
        domain.FixUpTask::category: domain.Category = domain.Category{id=3088, version=0}
```

Figure 6. Test 2



```
> reina
domain.FixUpTask{id=3239, version=0}
        domain.DomainEntity::id: int = 3239
        domain.DomainEntity::version: int = 0
        domain.FixUpTask::ticker: java.lang.String = "181011-838453"
        domain.FixUpTask::publishedMoment: java.util.Date = <<2018-10-11 11:45:00.0>>
        domain.FixUpTask::description: java.lang.String = "My computer is broken"
        domain.FixUpTask::address: java.lang.String = "Reina Mercedes sn. ETS Ingenieria informatica"
        domain.FixUpTask::maxPrice: domain.Money = domain.Money@209d7383
        domain.FixUpTask::startDate: java.util.Date = <<2018-10-18>>
        domain.FixUpTask::endDate: java.util.Date = <<2018-10-25>>
        domain.FixUpTask::customer: domain.Customer = domain.Customer{id=3094, version=0}
        domain.FixUpTask::complaints: java.util.Collection = [domain.Complaint{id=3255, version=0}, domain.
        domain.FixUpTask::applications: java.util.Collection = [domain.Application{id=3251, version=0}, dom
        domain.FixUpTask::workPlan: java.util.Collection = [domain.Phase{id=3200, version=0}, domain.Phase{
        domain.FixUpTask::warranty: domain.Warranty = domain.Warranty{id=3230, version=0}
        domain.FixUpTask::category: domain.Category = domain.Category{id=3088, version=0}
```

Figure 7. Test 3



```
> nothing
The term has not been found
>
Enter a valid string
> |
```

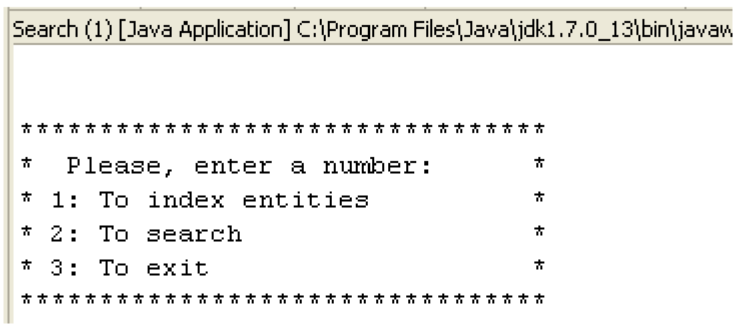
Figure 8. Tests 4 & 5

2.6. Our application

We wrapped the two classes in search.java madding a menu to use hibernate search easier.

The search.java is in the utilities package.

Note that is needed to index the entities to execute the search.



```
Search (1) [Java Application] C:\Program Files\Java\jdk1.7.0_13\bin\javaw

**********************************
*  Please, enter a number:       *
* 1: To index entities           *
* 2: To search                   *
* 3: To exit                     *
**********************************
```

Figure 9. Search.java menu

3. Improved version

We could go further and, making some changes, achieve a better search engine, we can define an analyzer that allows us to search sub-words, for example, if we search “comp” the application will return us a FixUpTask that has a description that contains “computer”.

3.1. Analyzer definition

We need to define an analyzer in the FixUpTask class.

Code of the analyzer

```
@AnalyzerDef(name = "customAnalyzerFinder",
 tokenizer = @TokenizerDef(factory = WhitespaceTokenizerFactory.class),
filters = {
// Replace accented characters by their simpler counterpart (è => e, etc.)
    @TokenFilterDef(factory = ASCIIFoldingFilterFactory.class),
//Lowercase all characters
    @TokenFilterDef(factory = LowerCaseFilterFactory.class)
// Generate prefix tokens
}
)
```

Additional filter

```
    @TokenFilterDef(factory = EdgeNGramFilterFactory.class,
      params = {
              @Parameter(name = "minGramSize", value = "1"),
              @Parameter(name = "maxGramSize", value = "40")
      })
```

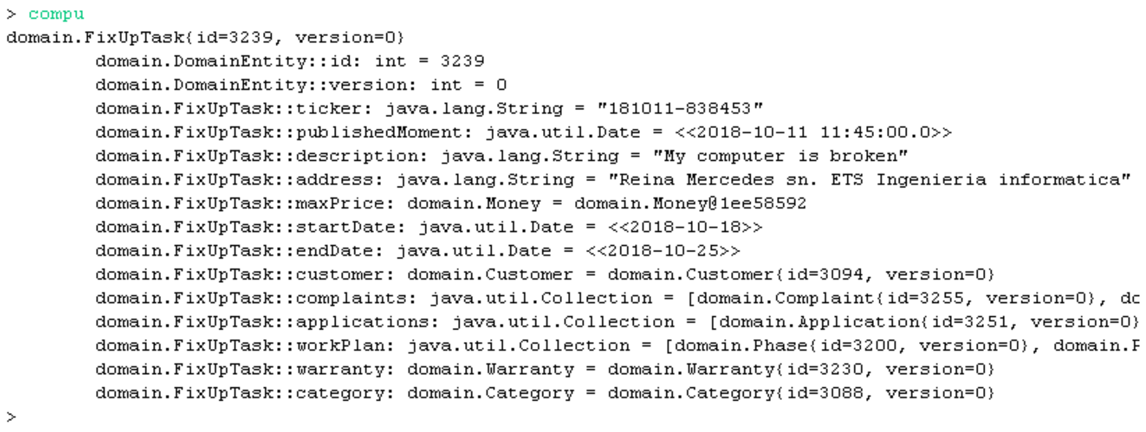
The additional filter is only used when we are indexing our entities, if we are querying we need to remove it.

In the getters we need to add the following:

```
@Field(index=Index.YES, analyze=Analyze.YES, store=Store.NO,

analyzer = @Analyzer(definition = "customAnalyzer"))
```

3.2. Test

To test our improved application, we execute the indexes application with the additional filter and then in the search one tried, without the filter, to find an incomplete word:



```
> compu
domain.FixUpTask{id=3239, version=0}
        domain.DomainEntity::id: int = 3239
        domain.DomainEntity::version: int = 0
        domain.FixUpTask::ticker: java.lang.String = "181011-838453"
        domain.FixUpTask::publishedMoment: java.util.Date = <<2018-10-11 11:45:00.0>>
        domain.FixUpTask::description: java.lang.String = "My computer is broken"
        domain.FixUpTask::address: java.lang.String = "Reina Mercedes sn. ETS Ingenieria informatica"
        domain.FixUpTask::maxPrice: domain.Money = domain.Money@1ee58592
        domain.FixUpTask::startDate: java.util.Date = <<2018-10-18>>
        domain.FixUpTask::endDate: java.util.Date = <<2018-10-25>>
        domain.FixUpTask::customer: domain.Customer = domain.Customer{id=3094, version=0}
        domain.FixUpTask::complaints: java.util.Collection = [domain.Complaint{id=3255, version=0}, dc
        domain.FixUpTask::applications: java.util.Collection = [domain.Application{id=3251, version=0}
        domain.FixUpTask::workPlan: java.util.Collection = [domain.Phase{id=3200, version=0}, domain.F
        domain.FixUpTask::warranty: domain.Warranty = domain.Warranty{id=3230, version=0}
        domain.FixUpTask::category: domain.Category = domain.Category{id=3088, version=0}
>
```

Figure 10. Test "compu"

1. References

[1]. Hibernate search documentation, http://hibernate.org/search/documentation/getting-started/