

Hibernate Search

Full-text search for Hibernate applications

Emmanuel Bernard

JBoss, by Red Hat

<http://in.relation.to/Bloggers/Emmanuel>



Copyright 2007-2010 Emmanuel Bernard and Red Hat Inc.

- Understand what full-text search does for you
- Understand the magic sauce: analyzers
- Full-text search and applications: how does it fit?
- Bring the *Wow!* effect to existing applications

Emmanuel Bernard



Hibernate Search in Action



blog.emmanuelbernard.com



twitter.com/emmanuelbernard



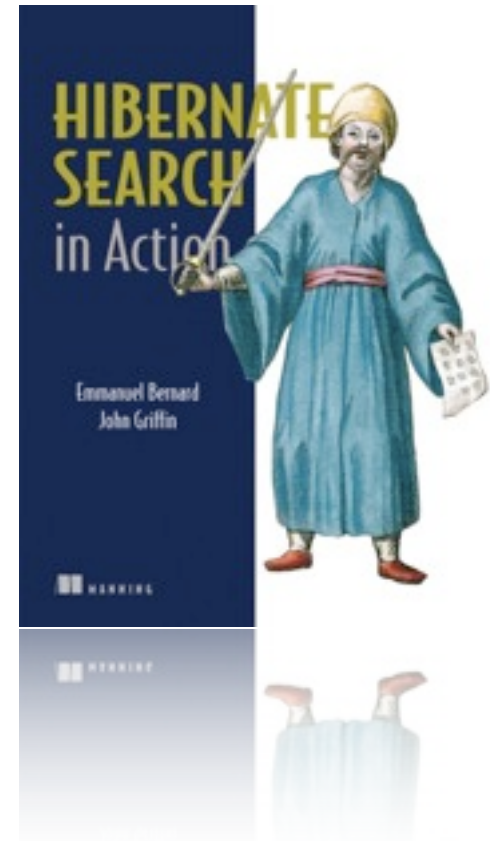
lescastcodeurs.com



Les CastCodeurs

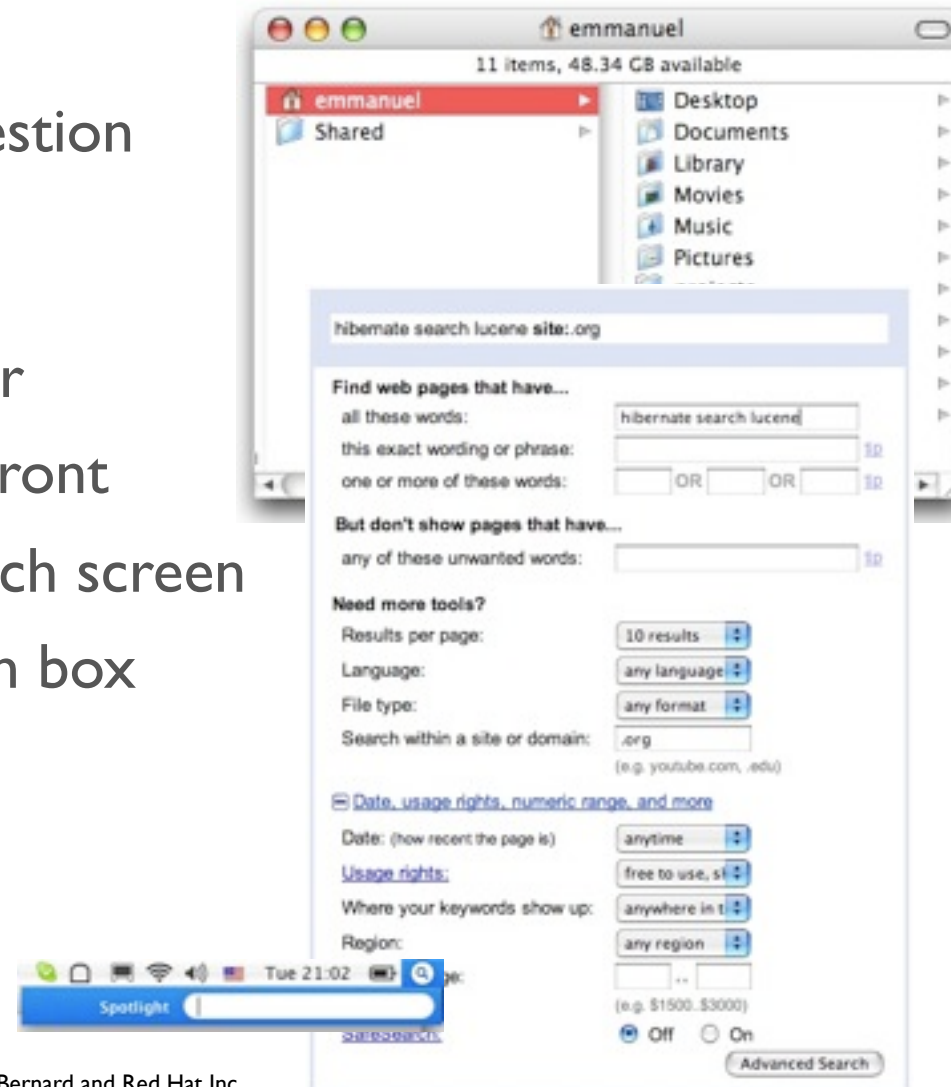
"En Français dans le Code"

<http://lescastcodeurs.com>



What is searching?

- Searching is asking a question
- Different ways to answer
 - Categorize data up-front
 - Offer a detailed search screen
 - Offer a simple search box



Human search in a relational DB

- where? (which columns, which tables)
- column != word (wildcard queries?)
- did you say “car” or “vehicle”?
- konference or conpherance?
- Order results by relevance

- How to do that in SQL?

Full Text Search

- Search by word
- Dedicated index
 - inverted indices (word frequency, position)
- Very efficient

- Full text products:
 - embedded in the database engine
 - library embeddable like Lucene

Some of the interesting problems

- bring the “best” document first
- recover from typos
- recover from faulty orthography
- find from words with the same meaning
- find words from the same family
- find an exact phrase
- find similar documents

Find by relevance

- Best results first
 - very human sensitive
- Prioritize some fields over others
- The more matches, the better
 - for a given key word per document
 - for a given document the amount of matching key words
- Similarity algorithm



Extracting the quintessence

- Word: Atomic information
- Analyzer
 - Chunk / tokenize the text into individual words
 - Apply filters
 - remove common words
 - lower case
- One tokenizer
- Some filters

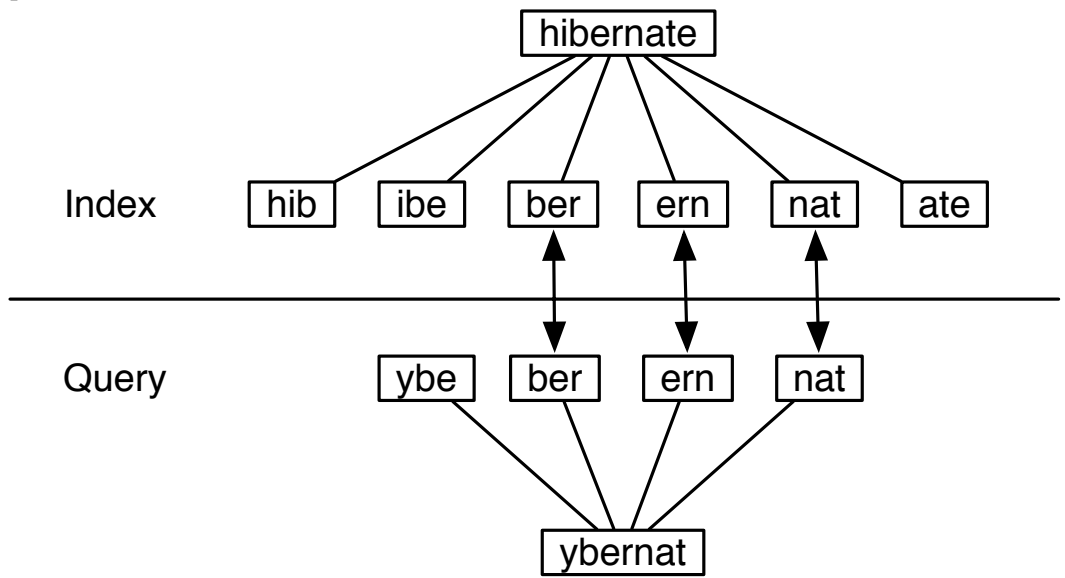
Approximation

- Recover from typos and other approximations
- Fuzzy search
 - query time operation
 - Levenshtein distance (edit distance)

Hibernate

Hibrenate

- n-gram
 - cut the word in parts of n characters
 - index each piece



- Indexing + query time strategy

Demo

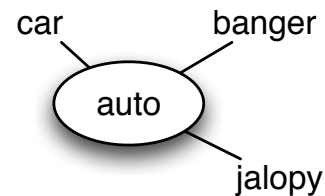
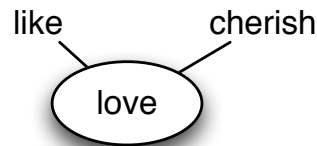
Phonetic search

- Is it “jiroscop” or “gyroscope”
 - not so useful in daily life
- Several phonetic algorithms
 - Soundex
 - Metaphone (JRSKP)
 - mostly for latin languages
- index the phonetic equivalent of a word

- Indexing + query time strategy
 - use a TokenFilter

Synonyms

- Based on a synonym dictionary
- index a reference word in the index



I like to drive my auto around		
I love to drive my banger around		I love to drive my auto around
I cherish to drive my car around		

- Indexing + query time strategy
 - use a TokenFilter

Words from the same family

- love, lover, loved, loving
- Stemming
 - Porter algorithm for English
 - Snowball Stemmer for most Indo-European languages
- Indexing + query time strategy
 - use a TokenFilter

Demo

What's the catch

- Lucene is quite low level
- Integration into an application model
- Index synchronization
- Object model conversion
- Programmatic mismatch

Integration in Java SE / EE

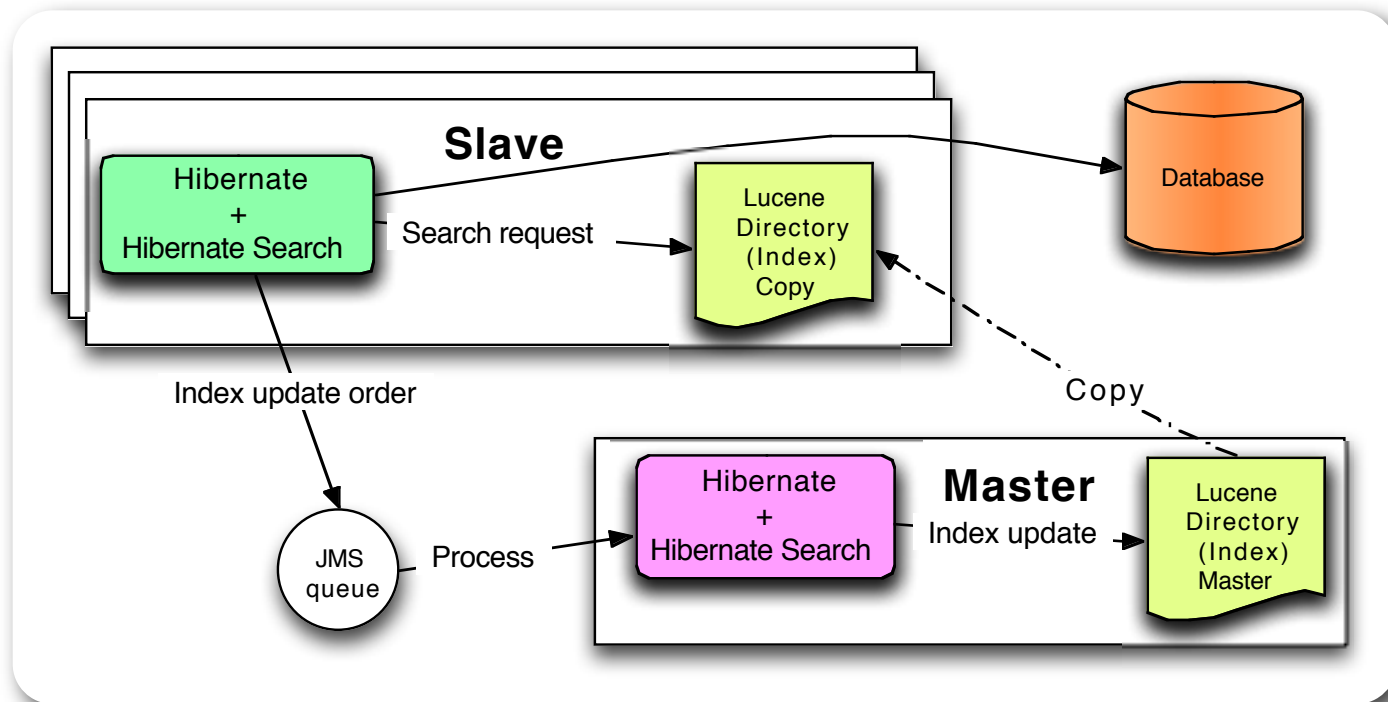
- Hibernate Search bridges
 - Hibernate Core and Java Persistence
 - JBoss Cache & Infinispan (More to come)
 - Apache Lucene
- Transparent index synchronization (event based)
- Metadata driven conversion (annotation based)
- Unified programmatic model
 - API
 - semantic

More on Hibernate Search

- Asynchronous clustering (JMS, JGroups)
- Projection
- Filters
- Index sharding
- Custom DirectoryProvider (eg. JBoss Cache, Infinispan based)
- Infinispan / JBoss Cache full text searchable
- Native Lucene access

Asynchronous cluster

- Search local / change sent to master
- Asynchronous indexing (delay)
- No front end extra cost / good scalability



Summary

- Search for humans
- Full text tackles those problems
 - relevance
 - (human) fault tolerance
 - stemming and synonyms
 - incremental search
- Barrier of entry has lowered: Go for it!
 - POJO based approach
 - infrastructural code tackled by frameworks
 - unified programmatic model

Present & Future

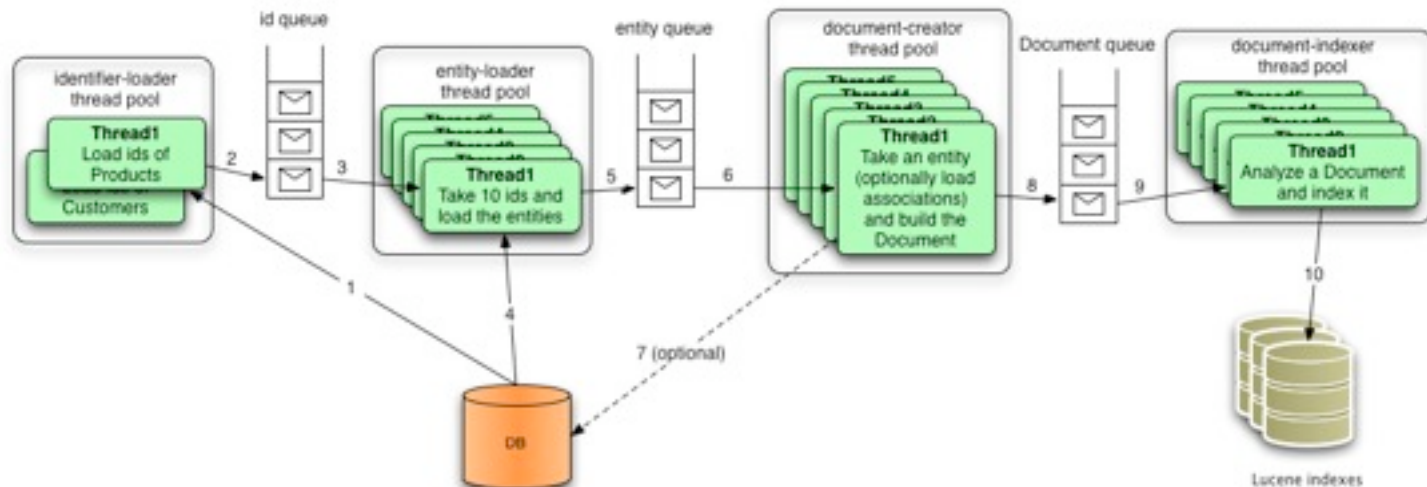
- Perf
 - on mass indexing
 - on simple deployments
- Ease of use
 - programmatic API to express mappings
 - fluent API to write queries
 - Error handling
- Cluster
 - Infinispan based Lucene directory

Massive indexing

```
fullTextSession.createIndexer().startAndWait();
```

```
fullTextSession
```

```
.createIndexer( User.class )  
.batchSizeToLoadObjects( 25 )  
.cacheMode( CacheMode.NORMAL )  
.threadsToLoadObjects( 5 )  
.threadsForSubsequentFetching( 20 )  
.startAndWait();
```



Programmatic configuration

```
SearchMapping mapping = new SearchMapping();
mapping.analyzerDef( "stem", StandardTokenizerFactory.class )
    .tokenizerParam( "name", "value" )
    .tokenizerParam( "name2", "value2" )
    .filter( LowerCaseFilterFactory.class )
    .filter( SnowballPorterFilterFactory.class)
    .param("language", "English")
.entity(Address.class).indexed().indexName("Address_Index")
    .property("street1", ElementType.FIELD)
    .field()
    .field()
        .name("street1_iso")
        .store( Store.YES )
        .index( Index.TOKENIZED )
        .analyzer( ISOLatin1Analyzer.class)
    .field()
        .name("street1_ngram")
        .analyzer("ngram")
.entity(User.class).indexed()
    .property("name", ElementType.METHOD)
    .field();
```

Questions

- <http://search.hibernate.org>
- <http://lucene.apache.org>
- Hibernate Search in Action
 - Manning
- <http://in.relation.to>
- <http://blog.emmanuelbernard.com>

