Hibernate Search: Googling Your Java™ Technology-Based Persistent Domain Model

Emmanuel Bernard

Core Developer
JBoss, A Division of Red Hat
http://www.jboss.com http://www.hibernate.org

TS-4746

Search: Left Over of Today's Apps

Add a search dimension to your persistent domain model

"Frankly, search sucks on this project"

—Anonymous' Boss

Why? How to fix that? For cheap?

Integrate Full-Text Search and Persistent Domain

- SQL Search vs. Full-text Search
- Object model/Full-text Search mismatches
- **Demo**
- Hibernate Search architecture
- Configuration and mapping
- Full-text based object queries
- Towards a Java™ Persistence API 2.0 (JPA) integration

SQL Search Limits

- Query by word
  - '*hibernate*' a.k.a. % in SQL
- Approximation (or synonym)
  - 'hybernat'
- Proximity
  - 'Java' close to 'Persistence'
- Relevance or (result scoring)
- Multi-"column" search

Full Text Search

- Search information
  - By word
  - Inverted indices (word frequency, position)
- In RDBMS engines
  - Portability (proprietary add-on on top of SQL)
  - Flexibility
- Standalone engine
  - Apache Lucene™ http://lucene.apache.org

Mismatches With a Domain Model

- Structural mismatch
  - Full text index are text only
  - No reference between documents
- Synchronization mismatch
  - Keeping index and database up to date
- Retrieval mismatch
  - The index does not store objects
  - Certainly not managed objects

DEMO

Let’s Google Our Application!

Hibernate Search

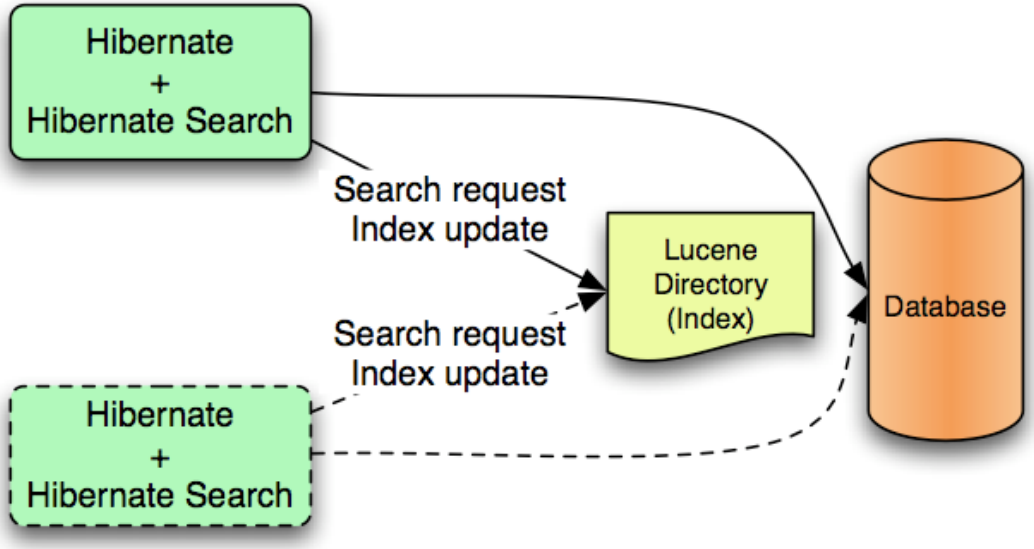
- Under the Hibernate platform
  - LGPL
- Use Apache Lucene™ under the hood
  - In top 10 downloaded at Apache
  - Very powerful
  - Somewhat low level
  - Easy to use it the “wrong” way
- Hibernate Search
  - Solve the mismatches

Architecture

- Indexing triggered by (JPA) event system
  - PERSIST/UPDATE/DELETE
- Convert the object structure into Index structure
- Operation batching per transaction
  - Better Lucene performance
  - “ACID”-ity
  - (pluggable scope)
- Backend
  - Synchronous/asynchronous mode
  - Lucene, Java Message Service (JMS)

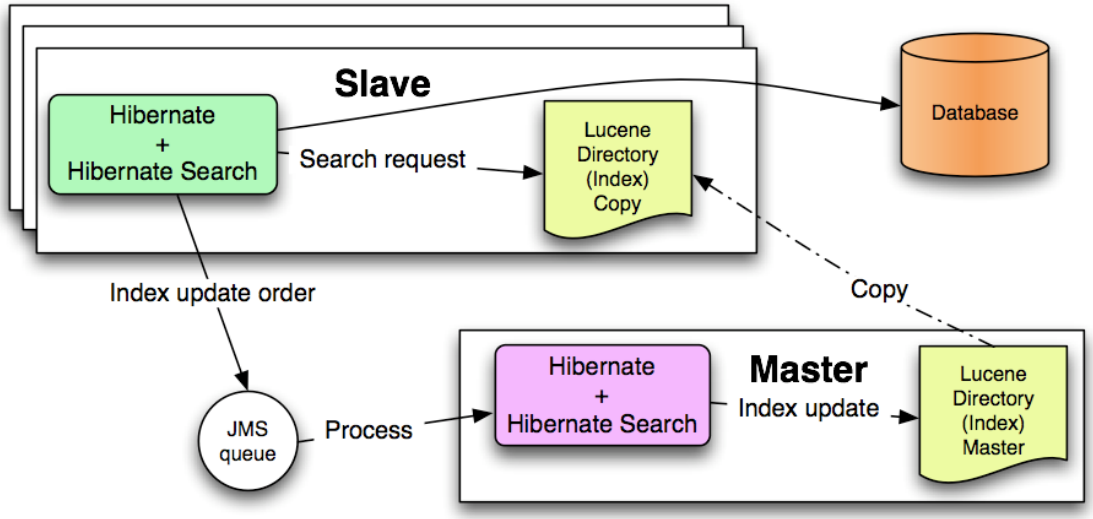
Architecture (Backend)

- Lucene Directory
  - Standalone or symmetric cluster
  - Immediate index change visibility
  - Can affect front-end runtime



Architecture (Backend)

- JMS Technology (Cluster)
  - Search processed locally
  - Changes sent to a master node (JMS technology)
  - Asynchronous indexing (delay)
  - No front-end extra cost



Configuration and Mapping

- Configuration
  - Event listener wiring (transparent in Hibernate Ann.)
  - Backend configuration
- Annotation based
  - @Indexed
  - @Field(store, index)
  - @IndexedEmbedded
  - @FieldBridge

Mapping Example

```
@Entity @Indexed(index="indexes/essays")
public class Essay {
    ...

    @Id @DocumentId
    public Long getId() { return id; }

    @Field(name="Abstract", index=Index.TOKENIZED,
        store=Store.YES)
    public String getSummary() { return summary; }

    @Lob @Field(index=Index.TOKENIZED)
    public String getText() { return text; }

    @ManyToOne @IndexedEmbedded
    public Author getAuthor() { return author; }
}
```

Query

- Retrieve objects, not documents
  - No boilerplate conversion code!
- Objects from the Persistence Context
  - Same semantic as a JPA-QL or Criteria query
- Use org.hibernate.Query
  - Common API for all your queries
  - Pagination support
  - List/scroll/iterate support
- Query on correlated objects
  - “JOIN”-like query

Query Example

```
org.apache.lucene.search.Query luceneQuery;
String queryString = "summary:Festina Or brand:Seiko"
luceneQuery = parser.parse( queryString );

org.hibernate.Query fullTextQuery =
fullTextSession.createFullTextQuery( luceneQuery );

fullTextQuery.setMaxResult(200);

List result = fullTextQuery.list();
//return a list of managed objects

queryString = "title:hybernat~" //Approximate search
queryString = "\"Hibernate JBoss\"~10" //Proximity search
queryString = "author.address.city:Atlanta"
//correlated search
```

Java Persistence API Integration

- Existing integration points
  - Java Persistence API entity listener (index update)
  - Transaction synchronization in JTA (index update)
  - JMS API (clustering)
- Problems
  - Transaction sync in JDBC
  - Initialization lifecycle
    - No one exists today in JPA
  - Standard lazy state detection
    - To avoid loads triggered by the Search engine
  - Standard lazy object initialization
    - To force a lazy loading load in the Search engine

Full-Text Search Without the Hassle

- Transparent index synchronization
- Automatic structural conversion through mapping
- No paradigm shift when retrieving data

- Clustering capability out of the box

For More Information

- Hibernate Search
  - http://search.hibernate.org
- JBoss Seam
  - http://www.jboss.com/products/seam
- Apache Lucene
  - http://lucene.apache.org
  - Lucene In Action
- Java Persistence API with Hibernate

Q&A

Hibernate Search: Googling Your Java™ Technology-Based Persistent Domain Model

Emmanuel Bernard

Core Developer
JBoss, A Division of Red Hat
http://www.jboss.com http://www.hibernate.org

TS-4746