



The Compass Framework

Search made easy

About Me

- Uri Boness <uri@jteam.nl>
- Software Architect at JTeam
- 8 years experience with Java
- Spring Modules committer
- JSR-303 (Bean Validation) expert group member

Goals

- You should walk away with:
 - Basic understanding of the gap compass fills
 - Basic understanding of compass core constructs
 - Understanding of compass ease of use and integration
 - **Enough arguments and knowledge to give Compass a try in one of your next/existing projects**



Agenda

- Why Search?
- Common Approaches & Solutions
- The origins of Compass
- Compass – Core constructs
- Compass – GPS
- Compass – Spring integration
- Demo
- Conclusion and Q&A



Why Search?

- More than WWW Search Engines
- Gmail – search, don't navigate.
- Web 2.0 – users expect more
- No quick access = Bad user experience
- Examples – Quicksilver, Spotlight, Google Desktop, Copernic, Gmail, JIRA, Lookout for Outlook, etc...



Common Approaches (cont.)

	Pros	Cons
SQL based	Simple - familiar ground	Extremely limited; Poor performance; Only databases
Database specific	Very simple	Clutter database; Tightly coupled to DB; Only databases
Search Engines	Powerful; Coupled with Application; Not only databases	Some learning curve; Synchronization needed

Lucene in a nutshell

- Why Lucene?
- World modeled as Documents and Fields
- Holds data in an index
- Low level API to:
 - Index data
 - Update indexed data
 - Search indexed data
- Text Analyzers & Converters
- Rich Query API – parsers, filters, and hits



Lucene in a nutshell (cont.)

```

public void indexBug(Bug bug) throws Exception {

    Document document = new Document();
    document.add(new Field("id", bug.getId(), Store.YES, Index.UN_TOKENIZED));
    document.add(new Field("title", bug.getTitle(), Store.YES, Index.TOKENIZED));
    document.add(new Field("description", bug.getDescription(), Store.YES, Index.TOKENIZED));

    Directory directory = FSDirectory.getDirectory("bug-index", true);

    IndexWriter indexWriter = new IndexWriter(directory, LuceneBugIndexer.ANALYZER, true);

    indexWriter.addDocument(document);

    indexWriter.optimize();

    indexWriter.close();
}
  
```



The need for Compass

- Lucene provides too low level API
- Lucene does not provide integration API
- Lucene is not transactional
- **Best practices already exist**

Compass to Lucene = Hibernate to JDBC



Introducing Compass

- Search made easy!
- Some of the features:
 - Declarative Configuration
 - Transactional
 - Very good integration support
 - JDBC, Hibernate, iBatis, JPA, etc...
 - Spring
- Yet still leverage the power of Lucene....



Introducing Compass (cont.)

```
public void indexBug(Bug bug) {  
  
    CompassConfiguration config = new CompassConfiguration()  
        .setConnection("compass-index")  
        .addResource("Bug.cpm.xml");  
  
    Compass compass = config.buildCompass();  
  
    CompassTemplate compassTemplate = new CompassTemplate(compass);  
  
    compassTemplate.save(bug);  
  
}
```



Compass - Overview

Compass Gps

Hibernate, JPA, JDO, OJB,
Jdbc

Compass Spring

DAO, Transaction, MVC,
AOP

Compass Core

Transactional Index, Mapping (OSEM/XSEM/RSEM),
Search Engine API & Abstraction,
Lucene extensions and helpers

Compass Core – Core Constructs

- Resources & Properties
- Mapping Definitions (XML & Annotations)
- Indexes & Connections
- Working With Compass API



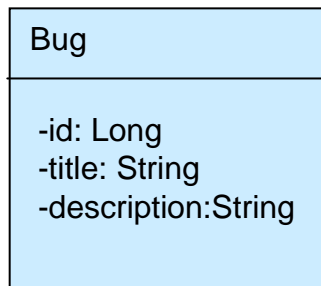
Resources & Properties

- Resource:
 - Maps to Lucene's Document
 - Holds text data as a set of properties
 - Holds one or more id properties
 - Associated with an alias
- Property
 - text based name-value pair
 - Maps to Lucene's Field

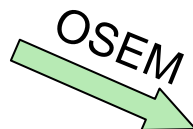
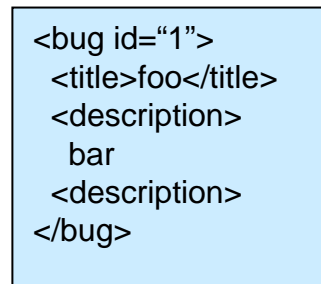
Compass Core – Mapping Definitions

Mechanism to map different forms of data to Resources

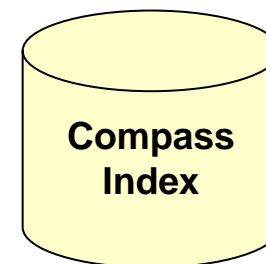
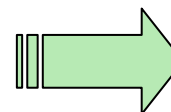
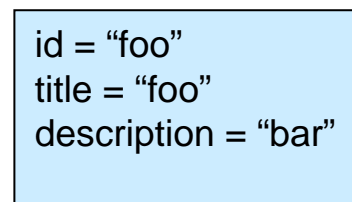
Java object



XML document



Compass Resource



Compass Core - OSEM (XML)

Bug.cpm.xml

```

<compass-core-mapping>

  <class name="jteam.samples.model.Bug" alias="bug">

    <id name="id"/>

    <property name="title">
      <meta-data>title</meta-data>
    </property>

    <property name="description">
      <meta-data>description</meta-data>
      <meta-data>content</meta-data>
    </property>

  </class>

</compass-core-mapping>

```

Bug.java

Bug
<ul style="list-style-type: none"> - Id : String - title: String - description: String
<pre> +setId() +getId() +setTitle() +getTitle() +setDescription() +getDescription() </pre>

Compass Core - OSEM (@Annotations)

```

@Searchable
public class Bug {

    @SearchableId
    private Long id;

    @SearchableProperty
    @SearchableMetaData(name = "title")
    private String title;

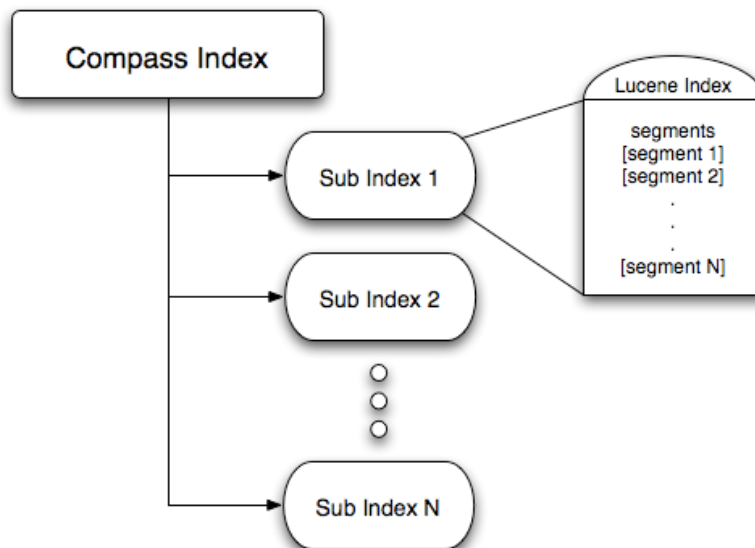
    @SearchableProperty
    @SearchableMetaDatas({
        @SearchableMetaData(name = "description"),
        @SearchableMetaData(name = "content")
    })
    private String description;

    public Bug() {
    }
    ...
}
    
```



Compass Core – Indexes

- Compass index is made out of one or more Lucene indexes
- Compass manages the structure of the index for optimization



Compass Core - Connections

- Defines where the compass index is located
 - File System
 - In Memory
 - JDBC

Compass Core – Compass API

- **CompassConfiguration (Hibernate Configuration)**
 - Uses to configure compass or load compass xml configuration
 - Registry for mapping definitions
- **Compass (Hibernate SessionFactory)**
 - Provide access to search engine management functionality
 - Serves as CompassSession factory
 - Created by CompassConfiguration's `buildCompass()`
 - Heavyweight & Thread safe
- **CompassSession (Hibernate Session)**
 - Provides access to index manipulation and search functionality
 - Opened by Compass's `openSession()`
 - Lightweight & **NOT** thread safe!!!
- **CompassTransaction (Hibernate Transaction)**
 - Compass abstraction of transactions
 - Starts by the session's `beginTransaction()`

Compass Core – Compass API (cont.)

```
CompassConfiguration conf = new CompassConfiguration()
    .configure().addClass(Bug.class);
Compass compass = conf.buildCompass();
CompassSession session = compass.openSession();
CompassTransaction tx = null;
try {
    tx = session.beginTransaction();

    session.save(bug);

    tx.commit();
} catch (CompassException ce) {
    if (tx != null) tx.rollback();
} finally { session.close(); }
```

Compass Core – Compass API (cont.)

- CompassTemplate
 - Based on Spring's Template pattern
 - Relieves the developer from directly dealing with transaction and exception handling



Compass Core – Compass API (cont.)

CompassTemplate usage:

```
CompassConfiguration conf = new CompassConfiguration()
    .configure()
    .addClass(Bug.class);

Compass compass = conf.buildCompass();

CompassTemplate template = new CompassTemplate(compass);

template.save(bug);
```

Compass Core – Compass API (cont.)

Indexing

```
Bug bug = new Bug(1, "title", "description");  
session.save(bug);
```

Searching

```
CompassHits hits = session.find("title:foo -description:bar ");  
  
CompassHits hits = compassTemplate.find(queryText);  
for (int i=0; i<hits.length(); i++) {  
    CompassHit hit = hits.hit(i);  
    Bug bug = (Bug)hit.getData(0);  
    float score = hit.getScore();  
}
```

Compass Gps – Overview

- Built on top of Compass Core
- Provides integration with different indexable data sources
- Indexable data sources can be:
 - File system
 - Database
 - Web
 - ORM solutions
 - Anything... just use your imagination



Compass Gps - Hibernate Device

- Relies Compass's OSEM

DB → Hibernate → Object → Gps → Compass

- Knows what objects to index from hibernate configuration
- Supports real time data mirroring for hibernate 3



Programmatic configuration

```
Compass compass = ... // set compass instance
CompassGps gps = new SingleCompassGps(compass);

CompassGpsDevice hibernateDevice =
    new Hibernate3GpsDevice("hibernateDevice", sessionFactory);

gps.addDevice(hibernateDevice);
gps.start();
```

Compass – Spring integration

- Configuration
- Dao support
- Transactions
- Spring AOP
- Spring MVC

Compass - Spring Integration (cont.)

```

<beans>
  <bean id="compass" class="org.compass...LocalCompassBean">
    <property name="resourceLocations">
      <list>
        <value>classpath:Bug.cpm.xml</value>
      </list>
    </property>
    <property name="compassSettings">
      <props>
        <prop key="compass.engine.connection">compass-index</prop>
        <prop key="compass.transaction.factory">
          org.compass.core.transaction.LocalTransactionFactory
        </prop>
      </props>
    </property>
  </bean>
</beans>
  
```



Compass in Spring Application Context (2)

Spring 2.0 Namespaces!!!

```
<beans>

  <compass:compass name="compass">
    <compass:connection>
      <compass:file path="compass-index" />
    </compass:connection>
    <compass:transaction factory="org...LocalTransactionFactory"/>
    <compass:mappings>
      <compass:resource location="Bug.cmp.xml"/>
    </compass:mappings>
  </compass:compass>

</beans>
```

Compass Spring Integration - Transactions

- Supports Spring Transactions
- In Managed Environments:
 - No need to manage transactions
 - No need to close sessions
 - Hmm... no need for CompassTemplate!!!
 - @CompassContext

Compass - Spring Integration (cont.)

```

<beans>

  <bean id="transactionManager".../>

  <compass:compass name="compass" txManager="transactionManager">
    <compass:connection>
      <compass:file path="compass-index" />
    </compass:connection>
    <compass:mappings>
      <compass:resource location="Bug.cmp.xml"/>
    </compass:mappings>
  </compass:compass>

  <!-- defines CompassContextBeanPostProcessor -->
  <compass:context/>

</beans>
  
```



Compass - Spring Integration (cont.)

And the code can now look like this:

```
public class CompassBugDao implements BugDao {  
  
    @CompassContext  
    private CompassSession compassSession;  
  
    // assuming our index only contains bugs  
    public CompassHits findBugs(String query) {  
        return compassSession.find(query);  
    }  
  
    ....  
}
```

Demo

Compass – What's next?

- **Many more features**
 - XSEM & RSEM
 - Analyzers & Filters registration
 - Common Metadata
 - Query Parsers & Builders & Highlighters
 - Index Hashing & Optimizers
 - Lucene JDBC Directory
 - And much more...
- **Future directions:**
 - Integration with more web frameworks
 - Distributed indexes/compass support
 - Compass Administration
 - Compass crawler
 - What would you like to see in compass?

Compass - Conclusion

- Utilizes Lucene's power
- Improves usability
- Transactional
- Very good and simple integration
 - Hibernate/JPA/ORM
 - Spring

Resources

- Compass Framework website
<http://www.opensymphony.com/compass>
- Lucene website
<http://lucene.apache.org>
- Shay Bannon's Blog (the creator of Compass)
<http://jroller.com/page/kimchy>
- Uri Boness's Blog
<http://uri.jteam.nl>
- JTeam website
<http://www.jteam.nl>



Q&A