

GRAILS +
PLUGINS +
SEARCH =
SEARCHABLE PLUGIN

WHO'S THIS?

- Maurice Nicholson
- Java developer
- yadda, yadda

SEARCH

SEARCH

- what do I mean by search?
- you know - *search*

DO YOU KNOW ABOUT SEARCH?

- yes / no

WHY NOT?

- not interested in it (uh oh!)
- no need
- lack of opportunity
- apparent complexity

WHAT DO I KNOW?

- I've read *Lucene In Action* ;-)
- work experience at dealchecker.co.uk
- current project leans heavily on search

MY STORY SO FAR...

- I used raw Lucene: almost drowned in code
- Lucene indexes are like DBs: every one is unique, but you have to get to know it

COMPASS and LUCENE

- Lucene is like JDBC: low level (focus on data-source), entirely code-driven and verbose
- Compass is like Hibernate: high level (focus on domain), more declarative (config files/annotations) and mostly quite low on LOC
- Hibernate = ORM, Compass = OSEM

LUCENE or COMPASS?

- Compass allows you to concentrate on your domain and write your business logic
- Compass abstracts index format and index access - you mostly don't need to care
- I was able to delete, like, 3k LOC!
- There's more to Compass than mapping domain models...

ALTERNATIVES?

- There are others competing with Lucene, but Lucene is de facto standard for "backend"
- On top of Lucene, Hibernate Search is one to watch

GRAILS PLUGINS

WHAT ARE THEY?

- re-usable Grails application micro-frameworks
- well thought out and simple to use

WHAT CAN THEY DO?

- add servlets/filters to web.xml
- add new beans to the Spring app context
- add taglibs, controllers, views, domains, services
- add new scripts
- add new methods to classes/objects - even user-defined classes!
- lots more

WHY BOTHER?

- re-use
- breaks application up into modular pieces; aspects if you like
- forces you to generalise/abstract: usually a good thing
- can improve quality, documentation, etc
- community
- make friends and influence people ;-)

PLUGIN DEVELOPMENT

- RTFM ;-)
- infrastructure by *codehaus*: SVN, JIRA, distribution mechanism

SEARCHABLE PLUGIN

AIMS

- make your Grails app's domain searchable
- minimal time investment and config
- give you easy ways to search your domain
- attract beginner to advanced users
- integrate well with Grails

WHAT IT DOES

- detects (user-defined) "searchable" domain classes
- generates Compass mapping
- builds search index
- syncs domain class (Hibernate) CRUDs
- adds new methods to domain classes
- provides a service
- provides a controller and view

IT FEEDS ON DOMAIN CLASSES

```
class Restaurant {
    static hasMany = [tags: Tag, reviews: Review]
    BigDecimal stars
    String name
    String description
    String variety
    Menu menu
    City city
    Long lat
    Long lng
    Address address
}
```

MAPPING REQUIRED

```
class Restaurant {  
    static searchable = true  
    static hasMany = [tags: Tag, reviews: Review]  
    BigDecimal stars  
    String name  
    String description  
    String variety  
    Menu menu  
    City city  
    Long lat  
    Long lng  
    Address address  
}
```

- plugin uses Grails's domain class API

MORE ADVANCED MAPPINGS

```
class Restaurant {  
    static searchable = {  
        name(boost: 1.5)  
        variety(index: 'un_tokenized')  
        address(cascade: 'create,delete')  
        menu(component: true)  
    }  
    // rest as before...  
}
```

- more closure mapping options to come...
- Compass XML and annotations supported

SEARCH != DB QUERIES

- DB: better for relational data, data that you understand and exact criteria
 - results always match criteria: ALL or NOTHING
- Search: better for text data, semantically vague data, and fuzzier criteria
 - results are ordered (usually) by relevance: some are MORE RELEVANT than others

DB QUERIES A LA GORM

```
def restaurants = Restaurant.findAll(
  "from Restaurant where city = ? and variety = ?",
  [london, 'MORROCAN']
)

def restaurants = Restaurant.createCriteria().list {
  menu {
    eq('vegetarianOption', true)
  }
  gt('stars', 1)
  eq('city', paris)
  address {
    'in'('districtCode', [8, 17, 9, 18])
  }
}
```

SEARCH QUERIES A LA SP

```
def result = Restaurant.search(  
  "local organic suppliers"  
)
```

- uses Lucene's own relevance formula
- uses Lucene query string syntax
- this query: matches one or more of those words in any order
- this query: searches in the "all" field created by Compass

SEARCH QUERIES: BOOSTS

- boosting affects natural relevance

```
def results = Restaurant.search {  
  queryString("curry house london")  
  gt('stars', 2, [boost: 2.0])  
}
```

- can be used with term-vectors for *suggestions/related* impl
- class properties can also be mapped with fixed boosts

SEARCH QUERIES: BOOLEANS

```
def topHit = Restaurant.searchTop {
  for (tag in ['favourite', 'authentic']) {
    term('tags', tag)
  }
  spanNear("all", inOrder: false, slop: 3) {
    add("live")
    add("music")
  }
  must(queryString('tasty tapas'))
  mustNot {
    multiPhrase('all') {
      add(["bad", "terrible"] as String[])
      add("service")
    }
  }
}
```

CONCLUSIONS

GROOVY and GRAILS

- Grails is full-stack app framework, and more (“grails” command, console)
- Groovy is expressive; makes DSLs easy
- can leverage existing Java technologies (Lucene, Compass)
- can simplify for the end user

GRAILS PLUGINS

- easy re-use across any number of apps:
learn once, use everywhere!
- Searchable Plugin would be nothing
without Grails domain classes
- ideal for orthogonal functionality

GRAILS, HIBERNATE + COMPASS

- all allow domain-driven-thinking
 - domain classes/objects have behavior that would live in services in classic JEE
 - but functionality can be externalised so does not clutter domain class itself
- yet Grails has “services”, so also supports classic JEE

WHAT ARE YOU WAITING FOR?

- <http://grails.org/>
- <http://grails.org/Plugins>
- user@grails.codehaus.org
- maurice@freeshell.org
- maurice.nicholson@dealchecker.co.uk