Digital Factory 7
Search and Query API under the hood

#jahiaone

Benjamin Papež, QA Architect
Search and Query API under the hood

- Overview on used search engine frameworks and API
- Jahia's extensions to the Jackrabbit search engine
- Maintenance tools and utilities
- What's new in Digital Factory 7
- Query tuning and general tips
- Questions?
Digital Factory: Search&query API under the hood

Overview on used search engine frameworks and API
Jackrabbit extends Lucene

- Improve performance for frequent index updates
- Parse SQL-2, Xpath, QOM to create Lucene syntax queries
- Support hierarchical queries
- Support joins
- Configure index rules, aggregates, analyzers…
Search and query tags / API in Digital Factory

Search tags / API

- Site, document repository and subsection search using full text and simple metadata constraints
- Returns list of hit objects (URL, excerpt, content node, score, references) sorted by score
- HTML form creation via JSP tags. Predefined search result snippet.
- OpenSearch and spellcheck support

Query tags / API

- More flexible, supports all JCR 2.0 query functions (SQL-2, QOM, Xpath)
- Returns iterator through found content nodes or row objects
- No form creation. Provides some edit mode components.
Query tags and API

SQL-2, QOM based
- Automatically handles multilingual content specifics
- Supports faceting
- No support for retrieving excerpts and spellcheck suggestions

Xpath based
- No automatic multilingual content handling
- No support for faceting
- Support for retrieving excerpts (with result highlighting) and spellcheck suggestions
Digital Factory: Search&query API under the hood

Jahia‘s extensions to the Jackrabbit search engine
Handle multilingual content

Jahia’s datamodel stores i18n properties in j:translation_* subnodes

This implementation choice is not visible from looking at CND or using Content manager/picker and Jahia JCR API. Should also not influence queries!
Handle multilingual content

Indexing modifications

- Non-i18n properties are copied down to fields of all translation node’s Lucene documents
  - Some properties are excluded (j:locktoken, jcr:lockIsDeep, jcr:lockOwner, jcr:language, jcr:created, jcr:createdBy, jcr:lastModified, jcr:lastModifiedBy, j:lastPublished, j:lastPublishedBy, j:published)
  - Primary nodetype and mixin types of main node are also copied

- Special fields in document:
  - :TRANSLATION_LANGUAGE and :TRANSLATED_PARENT
  - _ : PARENT field of translated node document points to grand-parent

- A change in the main node state automatically triggers re-indexing of translation nodes
Handle multilingual content

- **Query modifications**
  - Adding current language constraint into queries (SQL-2, QOM)
  - Excluding translation nodes from child- or descendant-node queries
  - In SameNode or ChildNode join comparisons main and translation nodes are treated as they would be the same nodes (UUID)
  - In search results we filter out multiple results per node (due to translation node)
  - When sorting results, the position of the translation node overrules the position of the main node
Faceting support (SQL-2, QOM)

- Integrate Apache Solr
  - SimpleJahiaJcrFacets based on Solr’s SimpleFacets class
  - Reuse solr-common module and use exactly same facet query syntax and result objects as Solr does

- facetable fields (set in CND) are prefixed with FACET: in Lucene document
  - Analyzed with KeywordAnalyzer
  - Date values are indexed with a Solr internal representation

- For hierarchical fields (set in CND) we additionally add a : FACET_HIERARCHY field with setting all the parent UUID nodes (implemented for categories, could be used for other hierarchies)
Faceting support

Facets can be added in Studio or Edit mode or even used programmatically
Count support

- JCR specification does not support „select count“
- In JCR there is just RangeIterator.getSize()
  - returns -1 when precise information is not available
  - Performance/memory overhead as all result objects are created and access rights checked

Digital Factory introduces a rep:count() function:

```sql
select count AS [rep:count(skipChecks=1;approxCountLimit=1000)]
from [jnt:file]
where isdescendantnode('/sites/ACME-SPACE')
```

- `skipChecks` to skip ACL and visibility checks
- `approxCountLimit` to get an approximate count
Query performance improvements

- Access control checks
- Visibility rule checks
- Publication status check
  - Added special fields to index (`_ACL_UUID`, `_CHECK_VISIBILITY`, `_PUBLISHED`)
  - Possible lazy loading of JCR nodes
  - Same read rights and visibility rules are resolved only once
Text extraction from documents is done asynchronously triggered by Quartz scheduler.

Extracted text is stored as property in jnt:content node.

- Prevent recurring text extraction on re-indexing, publication, mark for deletion,…

- Extract text from richtext field using Apache Tika (remove HTML markup)
Indexing related keywords in content definition (CND)

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nofulltext</td>
<td>property should be excluded from full text search</td>
</tr>
<tr>
<td>indexed=&quot;no&quot;</td>
<td>property should not be indexed at all as queries are never done on it</td>
</tr>
<tr>
<td>scoreboost=2.0</td>
<td>allows specifying the boost factor or &quot;weight&quot; for the property value</td>
</tr>
</tbody>
</table>

- Jackrabbit otherwise only controls these indexing settings via `indexing_configuration.xml`
  - with the ability to use some rules
  - better to use that configuration when you want to override our default property definitions
  - Check: http://wiki.apache.org/jackrabbit/IndexingConfiguration
**jcr:sql**

```xml
<jcr:sql var="result"
    sql="select * from [jnt:virtualsite] as site
     where isdescendantnode(site,'/sites')"/>

<c:forEach items="${result.nodes}" var="node">
```

**jcr:xpath**

```xml
<jcr:xpath var="result"
    xpath="/jcr:root/sites/${functions:xpathPathEncode(renderConte
xt.site.name)}/${functions:xpathPathEncode(descendantNode)}/e
lement(*, jnt:file)[not(@jcr:language)] order by
@jcr:lastModified descending"/>

<c:forEach items="${result.nodes}" var="node">
```
Digital Factory Query JSP tags

```xml
<jcr:qom var="result" scope="request">
  <query:selector nodeTypeName="nt:base"/>
  <query:descendantNode
    path="${renderContext.mainResource.node.path}"/>
  <query:column propertyName="j:tags"
    columnName="rep:facet (nodetype=jmix:tagged
    &key=j:tags&facet.sort=true)"/>
</jcr:qom>

<c:forEach items="${result.nodes}" var="node">
```

© 2002 - 2014 Jahia Solutions Group SA
Digital Factory Query
REST API

- **Find servlet**
  [http://www.jahia.com/community/documentation/jahia-modules/rest.html#Find](http://www.jahia.com/community/documentation/jahia-modules/rest.html#Find)

- **OpenSearch integration**
Digital Factory Search

JSP tags

```jsp
taglib prefix="s" uri="http://www.jahia.org/tags/search" %>
...
<url value='${url.base}${currentNode.properties.result.node.path}.html'
var="searchUrl"/>
<form method="post" class="simplesearchform" action="${searchUrl}">
  <site value="${renderContext.site.name}" display="false"/>
  <pagePath value="${searchPath}" display="false" includeChildren="true"/>
  <nodeType value="${searchType}" display="false" />
  <language value="${renderContext.mainResource.locale}" display="false" />
  <term id="searchCustomTerm" class="text-input" value="Search..."
    match="all_words" searchIn="siteContent,tags"
    onfocus="if(this.value==this.defaultValue)this.value='';"
    onblur="if(this.value=='')this.value=this.defaultValue;'"/>
  <nodeProperty nodeType="jmix:categorized" name="j:defaultCategory"
    id="searchCategories"/>

  <input type="submit" title="Submit" value=""/>
</form>
```

► Default search results (can be overridden):

modules\search\src\main\resources\jnt_searchResults\html\searchResults.jsp
RenderContext context = getContext();

CommaSeparatedMultipleValue oneSite =
    new CommaSeparatedMultipleValue();
oneSite.setValue("mySite");
CommaSeparatedMultipleValue frenchLang =
    new CommaSeparatedMultipleValue();
frenchLang.setValue("fr");

SearchCriteria criteria = new SearchCriteria();
criteria.setSites(oneSite);
criteria.setLanguages(frenchLang);
criteria.getTerms().get(0).setTerm("ACME");
criteria.getTerms().get(0).getFields().setSiteContent(true);
List<Hit<?>> hits = searchService.search(criteria, context).getResults();
Digital Factory Search API

- Custom search result excerpts
  - Custom display (CSS class)
  - Custom excerpt if search term found in tags, category

<param name="excerptProviderClass" value="org.jahia.services.search.jcr.HTMLExcerpt"/>

- Spellchecker (Did you mean)
  - Creates dictionary per site and language
  - Used to suggest terms if current search produces too few results.
  - Spellchecker is able to return more than one suggestion (configurable)
  - Limitation if multiple search terms are used and multiple have more than one suggestion, then only one suggestion is returned
Digital Factory: Search & query API under the hood

Maintenance tools and utilities
In the Digital Factory tools area there are some query/search related tools

- **JCR Query tool**
  - Very useful to test queries and measure their performance
  - Possibility to run actions on the results (deletion, purge unused versions)

- **Search engine management**
  - Check index consistency or trigger re-indexing

- **Document text extractor**
  - Extract and show content from uploaded documents
Other query/search utilities

- Auto-completion
  - Service offers auto-completion suggestions for typing in fields with short property values (users, categories, tags, …)

- Auto-tagging
  - Simple solution used in Wise (parse document and automatically assign tags from an uploaded dictionary of tags)

- Soon on the forge
  - Apache Stanbol integration (for entity extraction, auto-tagging)
  - Digital Factory Crawler (using Nutch)
  - Similarity search, MoreLikeThis module
  - Synonym search module
What's new in
Digital Factory 7

Digital Factory: Search&query API under the hood
Framework version upgrades

- Apache Jackrabbit (from 2.2.4 to 2.4.5)
  - Query related performance improvements and bug fixes
  - Lucene native sort in SQL-2

- Apache Lucene (from 2.4.1 to 3.0.3)
  - Performance improvement
  - Less memory usage (no huge byte arrays for norms)

- Apache Solr (from 1.3.0 to 3.1.0)
  - Performance improvement
  - New range facet type (not only date, also other kind of ranges)
Language dependent analyzers

- Indexing of i18n full text properties is now done with the corresponding language specific Lucene analyzer 
  (org.apache.lucene.analysis.*)

- Default analyzers can be overridden or new ones added

```xml
<analyzer-registry>
  <en_US class="org.apache.lucene.analysis.standard.StandardAnalyzer">
    <customizer class="org.mycomp.query.lucene.AnalyzerCustomizer">
      <foo>bar</foo>
      <foo>bar2</foo>
      <key>value</key>
    </customizer>
  </en_US>
</analyzer-registry>
```

- Same language dependent analyzers are used to analyze full text search terms in queries
External data provider

- Same query syntax and API is used to run queries on mapped external data provider

- Results from different data providers are merged

- Search also in extended external data
Other new features

- Limit Spellchecker dictionary creation to content of particular properties
  ```xml
  <spellchecker>
    <include-property>jcr:title</include-property>
  </spellchecker>
  ```

- Store extracted text as binary instead of String
  - Before: huge String properties were always loaded to memory, but only used for indexing

- JCR Query statistics in tools area
  - List slowest and most popular queries
Query tuning and general tips
Ensure that queries perform

- Test duration of queries with the „JCR Query tool“ using production data or a similar test environment

- On Jahia xCM 6.6.x avoid SQL-2 queries with order by a property and a large result-set

- Xpath sometimes perform better than SQL-2

- Avoid that a query is executed too often (e.g. when using top-pager and bottom-pager we store query result in request)

- Avoid cache.expiration=0 on components with queries

- Make use of the „JCR Query statistic“ tool to identify slow or often used queries
Use available features

- Use rep:count() instead of iterating through results

- If you need a "select distinct" feature (which is not supported in JCR), you could use faceting programmatically:

```sql
select ace.[j:roles] AS [rep:facet(facet.mincount=1)]
  from [jnt:ace] as ace
  where ace.[j:aceType]="GRANT" and
  ace.[j:principal] = '"'+fPrincipal+'"' and
  isdescendantnode(ace, ['"'+site.getPath()+'"'])
```
How to enable multiple search hits per page

- Create a template for the nodetype, which should show as distinct search result
  
  ```xml
  <news-template j:applyOn="jnt:news" j:defaultTemplate="true"
    j:priority="100" jcr:primaryType="jnt:contentTemplate">
  ```

- Or trigger URL modification rules
  
  ```java
  rule "Event search result hit"
  when
      A search result hit is present
        - the node is of type jnt:event
  then
      Append URL query-parameter "filter" with
        getEventPageFilter((JCRNodeWrapper)searchHit.getRawHit())
      Append URL query-parameter "calStartDate" with
        (DateFormatUtils.format(((JCRNodeWrapper)searchHit.getRawHit())
        getProperty("startDate").getDate(), "yyyy-MM-dd"))
  end
  ```
Escaping path and fulltext search terms

SQL-2

```xml
<jcr:sql var="result"
    sql="select * from [jnt:post] as post
    where isdescendantnode(post,
        ['${functions:sqlencode(currentNode.path)}']
    ) and
    contains(post.*, '${functions:sqlencode(searchString)}')"/>
```

Xpath

```xml
<jcr:xpath var="result"
xpath="/jcr:root/sites/${functions:xpathPathEncode(renderContext.site.name)}/${functions:xpathPathEncode(descendantNode)}//element(*, jnt:file)
[jcr:contains(., ${functions:xpathSearchEncode(searchString)})]
"/>
```
Queries on multiple nodetypes

- Rather (define and) use a common mixin
- Use joins in SQL-2 / QOM
- Add constraints on jcr:primaryType property

```
select * from [jnt:content]
where ([jcr:primaryType] = 'jnt:contentReference' or [jcr:primaryType] = 'bnt:article') ...
```
Digital Factory: Search&query API under the hood

Questions?
Digital Factory: Search&query API under the hood

Stop searching, you found it

Thank you for your attention

#jahiaone