OpenCms days 2008

Using and extending OpenCms search capabilities

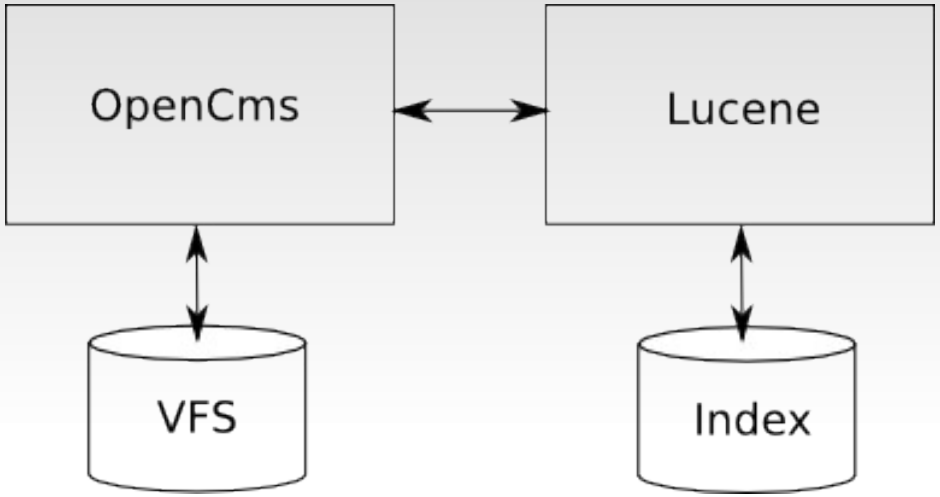
Claus Priisholm
CEO, CodeDroids ApS
www.codedroids.com

codedroids

Contents

- Overview over the built-in features
- Searching with the default setup
- Indexing structured contents
- Customizing the indexing
- Adding other sources to the mix
- Integrating with external search engines
- More searching

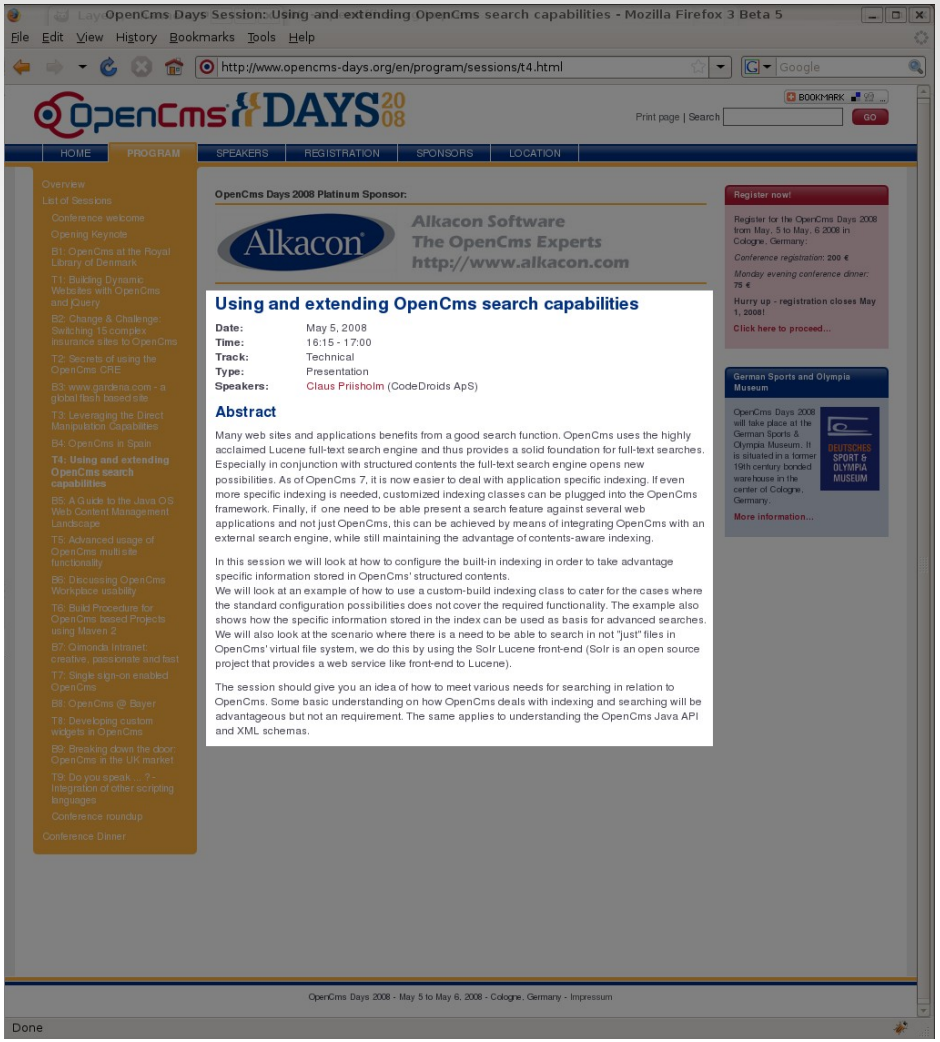
Built-in features



- Indexes contents and properties of VFS resources
- Works on the contents, not the final HTML page
- Flexible definition of multiple indices
- Various fields can be added to the indices
- Automated indexing
- Easy to use search API

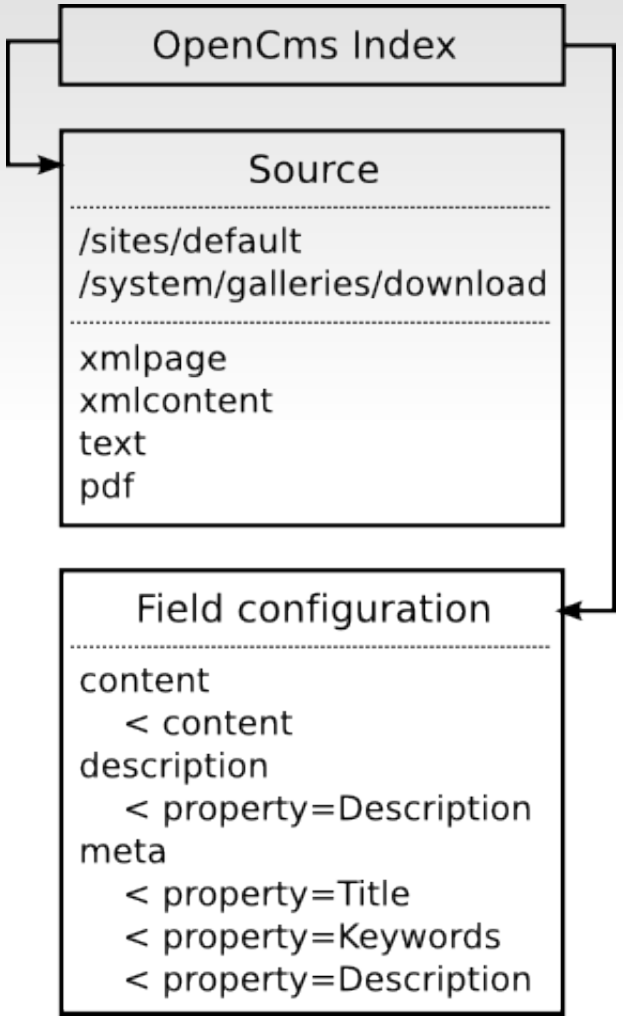
Indexing contents

- Example page, HTML codes stripped:
  - 4233 characters
  - 641 words
- Contents taken from XML file in VFS:
  - 1933 characters
  - 319 words
- Less noise equals better results



codedroids

Setting up an index



- Name, Rebuild, Locale, Project
- Sources
  - Indexer class
  - VFS resources
  - Document types
- Field configuration
  - Name, Description
  - Fields
    - Indexing properties
    - Mappings

Setting up an index

Example: Online project (VFS)

Searching

```
// Setting up the search
//
CmsJspActionElement cms = new CmsJspActionElement(...);
CmsSearch search = new CmsSearch();

search.init(cms.getCmsObject());
search.setDisplayPages(5);
search.setMatchesPerPage(10);
search.setIndex("Online project (VFS)");
search.setField(
  new String[] { "title", "keywords", "description", "content" }
);
search.setQuery(“opencms”); // typically from a request parameter
search.setQueryLength(2);
search.setSearchRoots(new String[] { "/" } );
search.setSortOrder(CmsSearch.SORT_DEFAULT);
```

Searching

```
// Printing the result
//
CmsSearchResultList result = search.getSearchResult();
ListIterator iterator = result.listIterator();
while (iterator.hasNext()) {
  CmsSearchResult entry = (CmsSearchResult)iterator.next();
  String path = cms.getRequestContext()
                                        .removeSiteRoot(entry.getPath())
  out.print("<h3><a href=\"" + cms.link(path) + "\">");
  out.print(entry.getTitle());
  out.print("</a>");
  out.print(" (" + entry.getScore() + ")");
  out.println("</h3>");
  if(!CmsStringUtil.isEmpty(entry.getDescription())) {
    out.println("<p>" + entry.getDescription() + "<p>");
  else
    out.println("<p>" + entry.getExcerpt() + "<p>");
}
```

Searching

Example: Basic search page

“Debugging”

Using Luke to see what
is really going on

Searching

“Out of the box” you have a useful
index for english contents,
just add a search page
using the CmsSearch API.

Indexing revisited

- More than one index
  - Online/offline index
  - Index per site
  - Index per locale
  - Index for specific resources
- More specific indexing
  - Indexing structured contents
  - Customized indexing of fields

Structured contents

- Add new field configuration or alter an existing one
- Add field(s) to the configuration
- Set mapping(s) for the field
- Set index to use the field configuration
- Rebuild index
- Test with index search

Structured contents

Example: add a field for Author names

Customizing

Example of a special value from an xmlcontent file (line breaks added for readability):

```
<LocalControlWords>
<![CDATA[
List 1#sport/teams,
List 1#sport/teams/football,
List 1#sport/teams/handball,
]]>
</LocalControlWords>
```

Customizing

- Subclass one of these classes:
  - org.opencms.search.documents.A_CmsVfsDocument
  - org.opencms.search.documents.CmsDocumentXmlContent
- Override either:
  - I_CmsExtractionResult extractContent(CmsObject cms, CmsResource resource, CmsSearchIndex index))
  - Document createDocument(CmsObject cms, CmsResource resource, CmsSearchIndex index)
- Insert into opencms-search.xml:
  - Enter class for the appropriate <documenttype> declarations

Customizing

```
public Document createDocument(CmsObject cms,
                                CmsResource resource,
                                CmsSearchIndex index)
{
    Document document = super.createDocument(cms, resource, index);
    if( resource needs special treatment ) {
        load and unmarshall the xml file
        extract the relevant data
        Field f = new Field("myfield", term,
                            Field.Store.YES, Field.Index.UN_TOKENIZED));
        document.add(f);
        ...
    }
    return document;
}
```

Customizing

```
<opencms>
    <search>
    ...
        <documenttypes>
            ...
            <documenttype>
                <name>xmlcontent</name>
                <class>my.new.class</class>
                ...
            </documenttype>
            ...
        </documenttypes>
    </search>
</opencms>
```

Other sources

- Indexing sources other than VFS files
- “Forcing” non-VFS data into OpenCms' indexes is not an optimal solution
- Better to have multiple Lucene indexes and then build a search frontend for them
- For database sources there are solutions like Compass, Hibernate search and so forth
- Use Lucene's MultiSearcher class

Integration

- Integrating with external search engine for flexibility and/or more features
- It should ideally work with the contents not the generated HTML page
- Have it traverse your site at regular intervals (using a crawler – e.g. Nutch)
- Better to push contents to it via some interface when publishing (e.g. Solr)

Solr

"Solr is an open source enterprise search server based on the Lucene Java search library, with XML/HTTP APIs, caching, replication, and a web administration interface."

Integrating

- Hook into OpenCms events by implementing I_CmsEventListener
- Check out CmsSearchManager.cmsEvent(CmsEvent)
- Add relevant fields to form XML format and push it to Solr via HTTPClient
- Build search interface that sends of queries Solr and formats the result

More searching

- A lot of times you need to generate lists of articles or other documents
- Usually you will use OpenCms' collectors
- But you can use Lucene as well
- The Danish Royal Library modules include an agent intended for these situations
- Generate RSS feeds
- Use agents as collectors

Links

Lucene:
lucene.apache.org

Solr:
lucene.apache.org/solr

Royal Library modules:
www.kb.dk/en/kb/it/dup/KBSuite.html