



Table of Contents

Chapter 0	3
Adobe Developer Tools	3
Community Developer Tools	4
Re-indexing Oak Indexes via Index Manager	4
Chapter 1: Full-text & search fundamentals	6
Exercise	6
Solution Package	8
Chapter 2: Filtering	9
Exercise	9
Solution Package	11
Chapter 3: Pagination	12
Exercise	12
Bonus Exercise	16
Solution Package	17
Reference Links	17
Chapter 4: Suggestions	18
Exercise	18
Solution Package	19
Chapter 5: Analyzers	20
Setup Package	20
Stemming	21
Synonyms	23
Stop words	25
HTML Strip	27
Solution Package	28
Chapter 6: Boosting	29
Exercise	29
Solution Package	34
Reference Links	34
Chapter 7: Similarity	35
Exercise	35
Reference Links	36
Chapter 8: Putting it all together	37
Exercise	37
Download the code	37
Chapter 9: Traversing queries	38
Bonus Exercises	







4. In the left pane of AEM Chrome Plug-in **click** the row for
/content/summit/l4080/chapter-1.html?q=oak
5. The right panel will update to with logging and query information for this request.
6. Click the **Queries** tab, and the executed query displays:



with the plan below it



7. The plan describes what Oak index will be used to execute this query; in this case the Lucene index named **cqPageLucene** is selected for use.

Inspecting the cqPageLucene index definition





hierarchy is indexed.

Note there is no property index rules for `jcr:content/cq:tags`

7. **Create a index rule for the property `[cq:Page]/jcr:content/cq:tags`**

While `/oak:index/cqPageLucene/indexRules/cq:Page/properties` is selected

Click **Create... > Create Node**

Node Name	Node Type
-----------	-----------



Solution Package

1. Navigate to CRX Package Manager: AEM Start > Tools > Deployment > Packages
 - a. <http://localhost:4502/crx/packmgr/index.jsp>
2. Search for **Chapter-2**
3. Click the package to expand: L4080-Chapter2.zip
4. Click install





2. Switch the page mode to **Preview** and perform the same keyword search as in the previous step.
Search **template development**.
3. Notice that the pagination has changed to only show the next page of results. You should be able to click through to the last page of results, but you will need to click 'Next' many more times. The time taken should be faster than in previous steps.



Use guessTotal=100 to read the first 100 results

1. Switch the page mode to **Edit** and open the Search Component dialog.
2. Navigate to the **Pagination** Tab and update the **Guess Total** field value to **100**. Save and close the dialog.



3. Switch the page mode to **Preview** and perform the same keyword search as in previous steps
Search ***template development***.
4. Notice that the pagination now shows more than 2 pages. However, since **guessTotal** is being used there is the potential for an extra page to be shown. Click immediately to the last page. Depending on the size of the result set you may end up on a page with no results because the calculated offset exceeded the result set total size.

Update the search result size

1. Switch the page mode to **Edit**









- b. Click on a suggestion to search and note the number of search results.
3. Open CRXDE Lite
 - a. <http://localhost:4502/crx/de>
4. Navigate to **/oak:index/cqPageLucene/indexRules/cq:Page/properties**
 - a. Review the properties for the nodes
 - i. **.../properties/jcrTitle**
 - ii. **... /properties/nodeName**
 - b. Note they both have the **useInSuggest** property set to **true** which is why the current suggestions are the multi-word values of these 2 properties.
5. Create a new node named **suggestion**, under **/oak:index/cqPageLucene**

Node Name	Node Type

6. Add the following properties to the **suggestion** node

Property Name	Property Type	Property Value

7. Click **Save All**



Chapter 5: Analyzers

AEM search allows Analyzers to be configured per index. Analyzers dictate how content is indexed into the search indexes, and can also augment how queries are executed against them. This exercise set up Stemming, Synonyms, Stop words and HTML Stripping.

Setup Package

For this series of exercises, install the Package **L4080-Chapter5-Setup.zip** via CRX Package Manager. This package augments the **/oak:index/cqPageLucene** index with a basic analyzer configurations.

- Standard character mapping
- Standard tokenizer
- Lower-case token filter

1. Navigate to CRX Package Manager: AEM Start > Tools > Deployment > Packages



Stemming

Stemming converts user-provided search words into their linguistic “root” thereby intelligently expanding the scope of the full-text search.

Stemming both an index time and query time activity. At index time, stemmed terms (rather than full terms) are stored in the full text index. At query time, the user provided search terms are stemmed and passed in as the full-text term.

For example

Given the provided term: **developing**

The stemmer will derive the root word: **develop**

Which includes content that contains derived forms such as “**developer**”, and “**development**”





Synonyms

Synonyms allow different terms with equivalent meaning to be considered the same by full-text search.

Pro-tip: *Place the Synonym filter node after LowerCase but BEFORE PorterStem*

Synonyms exercise

Create a custom synonym list for the **cqPageLucene** index.

1. Perform a search using the keyword **sightly** and note the lack of results, perform a search using the keyword **HTL** and note there are many results.
2. Open CRXDE Lite
<http://localhost:4502/crx/de>
3. Create a new node named **Synonym** under
/oak:index/cqPageLucene/analyzers/default/filters





Stop words

Stop words are effectively a black list of words that will not be added to the search index and thus unsearchable. Managed industries may add subjective terms as stop terms, or search over user-generated content may leverage them to keep profanities being searchable.

Pro-tip: Place the Stop filter node after both the LowerCase and Synonym filter nodes

Stop words exercise

Create a custom Stop Words Filter for the **cqPageLucene** index.

1. Perform searches using the keywords, and note the large number of results.
jsp
geometrix
classic ui
2. Open CRXDE Lite
<http://localhost:4502/crx/de>
3. Create a node named **Stop** under **/oak:index/cqPageLucene/analyzers/default/filters**



/oak:index/cqPageLucene/analyzers/default/filters/Stop

Node Name

Node Type



HTML Strip

HTML can be automatically removed from the search index; so as non-content elements don't populate the content search space. This can be helpful when HTML is stored in page properties, such as with Rich Text editors, Table or Content Fragment components.

HTML Strip exercise

Create an HTMLStrip CharFilter to the cqPageLucene index to prevent any HTML artifacts on the Page82 684529 c .3.nflum BT 55218 529.g; so as resul q .24 0 0 0.24 176.4125 588.2.8e 0 50 09 74 422



5. Move the new HTMLStrip node to be ABOVE the Mapping node
6. 





3. Add two properties to the **jcrTitle** node with the following values:

Property Name	Property Type	Property Value





Title and Tags

Title *

AEM Components: the Basics

Key Words

basic

4. Save and close the dialog
5. Return to the Chapter 6 page
<http://localhost:4502/editor.html/content/summit/l4080/chapter-6.html>
6. Switch the page mode to 'Preview'
7. Perform a search with the term **basic**
8. The page modified in Step 1 should be the first result

Solution Package

1. Navigate to CRX Package Manager: AEM Start > Tools > Deployment > Packages
 - a. <http://localhost:4502/crx/packmgr/index.jsp>
2. Search for **Chapter-6**
3. Click the package to expand: L4080-Chapter6-Solution.zip
4. Click install

Reference Links

<https://docs.adobe.com/docs/pt-br/aem/6-2/deploy/best-practices/best-practices-for-queries-and-indexing.html#Tips%20for%20Creating%20Efficient%20Indexes>

<http://jackrabbit.apache.org/oak/docs/query/lucene.html#boost>

https://wiki.apache.org/lucene-java/LuceneFAQ#How_do_I_make_sure_that_a_match_in_a_document_title_has_greater_weight_than_a_match_in_a_document_body.3F



Chapter 7: Similarity

Oak Lucene indexes also support Similarity Queries. The idea behind the similarity query is that it will return nodes that have similar content to the node specified in the query. This can be useful when attempting to implement a “More Like this...” component.

Exercise

View the Similar Results Component

1. Open **Sites / Adobe Summit 2017 / L4080 / Chapter 7 – Similarity**
<http://localhost:4502/editor.html/content/summit/l4080/chapter-7.html>
2. Perform a full-text search with term **asset metadata**
3. Click one of the search results to open
<http://localhost:4502/content/docs/en/aem/6-3/administer/content/assets/metadata.html>
- 4.



Chapter 8: Putting it all together

We have included a Search Page with all the features enabled in previous exercises. Experiment with the different search capabilities and use the tools in previous exercises to analyze the queries running behind the scenes.

Exercise

4. Navigate to **Sites / Adobe Summit 2017 / L4080 / Chapter 8 – Putting it all together**
 - a. <http://localhost:4502/editor.html/content/summit/l4080/chapter-8.html>
2. Search!

Note that this final implementation enables Quick Results that display below the Suggestions. This makes use of the SearchResults Sling Model exposed as JSON via Sling Model Exporter.

Download the code

Visit the public github.com repository

Summit



5. In a new browser tab, navigate to **Explain Query**



The new logging shows

- QueryBuilder predicate definition

- The query executed by the Oak query engine

 - This query can be used in Explain Query as described in the Explain Query section

- The Oak index coc0(The Oa936.06iuaET Q q 0.24 0 0 0.24 180.6139 180.372826m BT 50 0 0 50 0 C



Upon explaining, Explain Query will provide the query explanation in a modal overlay.

