Summit Lab Workbook

L736 – Experience Manager Search Demystified

Ian Reasor, Adobe Partner Experience
Tim Donovan, Adobe Partner Experience

March 29, 2018

Table of Contents

Introduction

This section is not a set of exercises, but rather an introduction to the various tools and techniques that we will be using throughout this lab.

Adobe Experience Manager

Using Chrome, you can log in to the AEM Author at http://localhost:4502/ or by using the AEM bookmark.

Username: admin	**Password:** admin

Chapters 1 – 4 have prepared AEM content pages available via **Sites > Adobe Summit 2018 > L736**. To open the page for a given chapter, select the Chapter page, then click **Edit** in the top action bar, or use the Chapter bookmarks available in Chrome.

Adobe Developer Tools

We will be using the following developer resources in this session:

QueryBuilder Debugger
The QueryBuilder Debugger executes queries written in QueryBuilder syntax, provides the derived XPath expression, and shows the query results. It can be accessed via:

- http://localhost:4502/libs/cq/search/content/querydebug.html

Community Developer Tools

The following tools are maintained by the community and are not supported by Adobe, but we find them to be useful while working to tune queries and indices in AEM.

Oak Index Definition Generator
The index definition generator will generate a suggested index that will satisfy a provided query. The suggested index can then either be created in AEM or merged with an existing index.

- http://oakutils.appspot.com/generate/index

AEM Chrome Plug-in
This Chrome plugin is part of the ACS AEM Tools project and provides detailed logging directly in the browser, via an integration with the Sling Log Tracer:

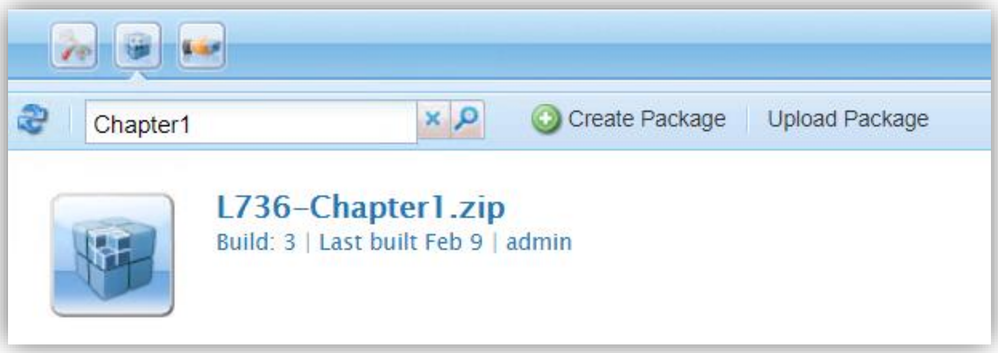
- http://adobe-consulting-services.github.io/acs-aem-tools/aem-chrome-plugin/

Solution Packages

If you have fallen behind, we have provided you with solution packages that will put your AEM environment into the needed state for the end of each chapter. To install a solution package for a chapter, you can do the following:

1. Navigate to CRX Package Manager: **AEM Start > Tools > Deployment > Packages.**
   - http://localhost:4502/crx/packmgr/index.jsp

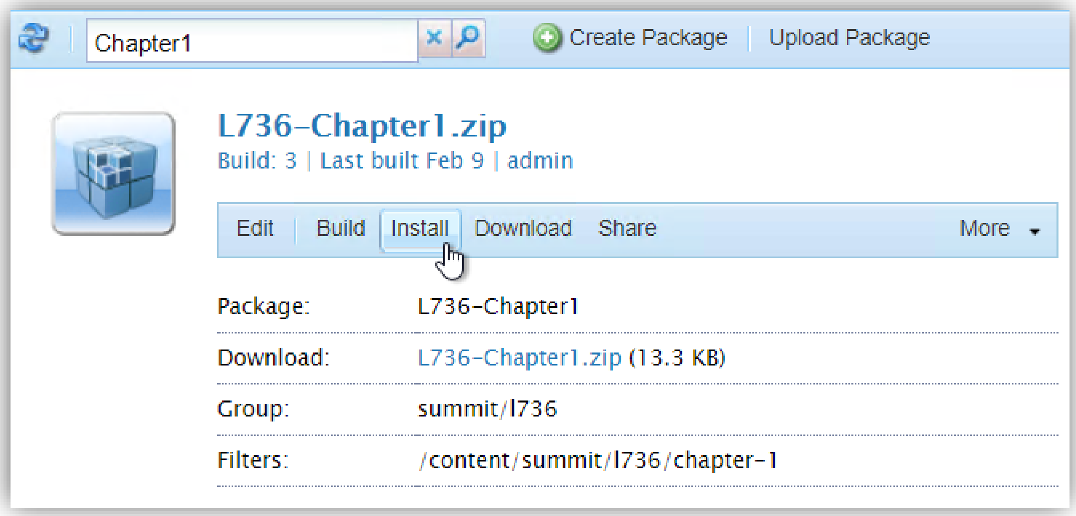
2. Search for the package for your chapter. For example, **Chapter1.**



L736-Chapter1.zip
Build: 3 | Last built Feb 9 | admin

3. Click the package to expand it.

4. Click **Install**.



L736-Chapter1.zip
Build: 3 | Last built Feb 9 | admin

Package:	L736-Chapter1
Download:	L736-Chapter1.zip (13.3 KB)
Group:	summit/l736
Filters:	/content/summit/l736/chapter-1

Chapter 1: Full-text and Search Fundamentals

AEM search supports robust full-text search, powered by Apache Lucene. Lucene property indices are at the core of AEM Search and must be well understood. This exercise covers:

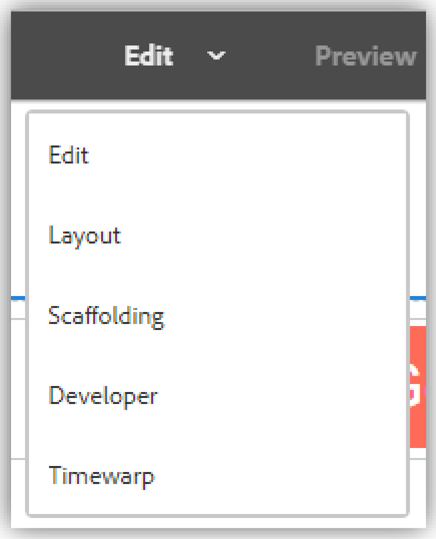
- Definition of the OOTB **cqPageLucene** Oak Lucene property index
- Search query inspection
- Full-text search operators
- Search result excerpts

Exercise

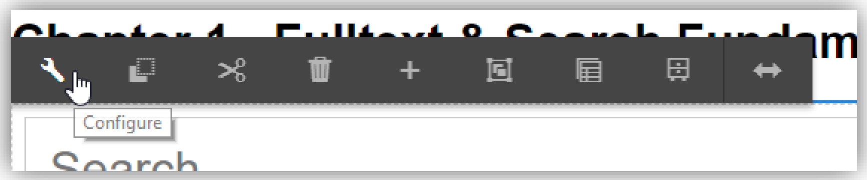
Configuring this lab's Search component

1. Open the page at **Sites > Adobe Summit 2017 > L736 > Chapter 1 – Full-text:**
   - http://localhost:4502/editor.html/content/summit/l736/chapter-1.html

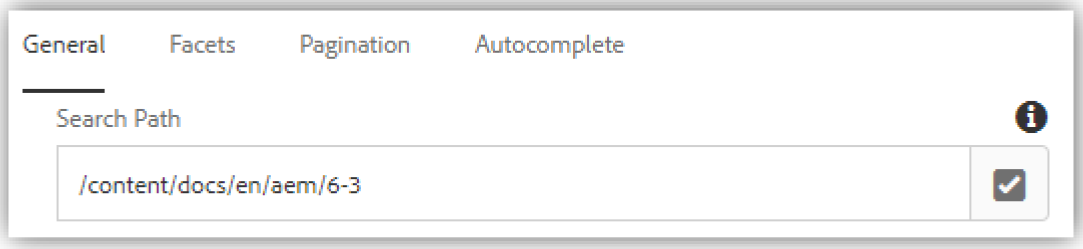
2. Ensure the page is in **Edit mode**, by selecting **Edit** from the drop-down menu in the top right.



3. Select the Search component and click the **configure button (wrench icon)** to edit.



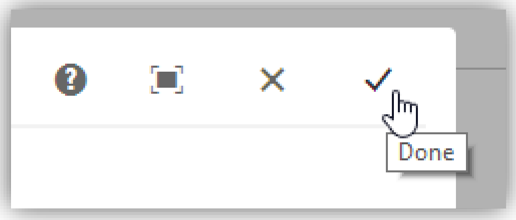
4. In the Search component dialog, set the following:
    a. Search path: **/content/docs/en/aem/6-3**



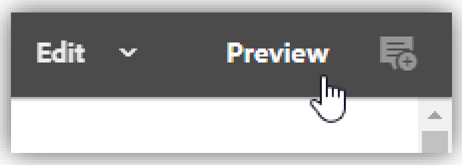
Search Path

/content/docs/en/aem/6-3

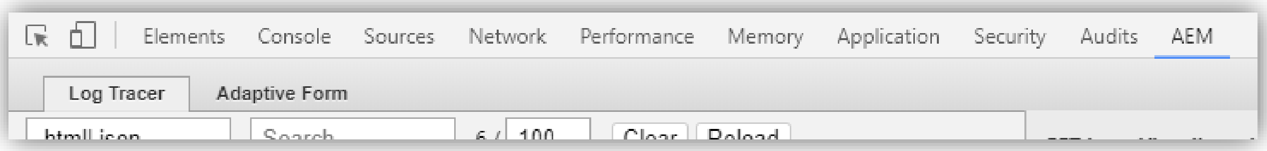
    b. Click the **done icon (checkmark)** in the top right to save dialog changes.



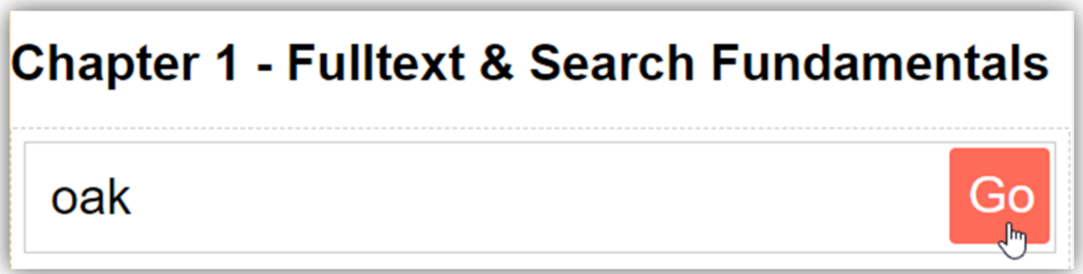
5. Switch to **Preview mode**, by clicking *Preview* in the top right.



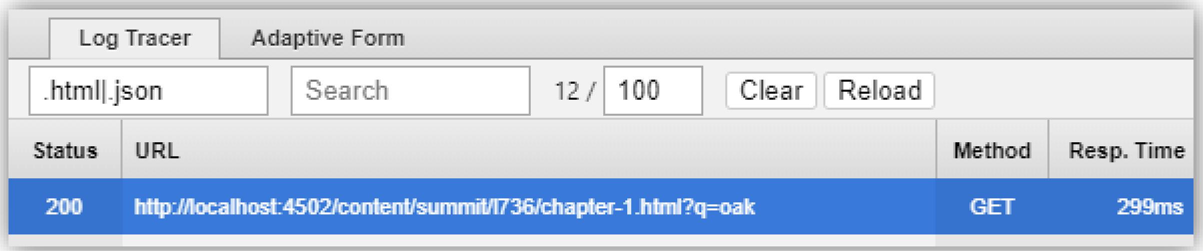
6. Open **AEM Chrome Plug-in.**
    a. While in Chrome, open the **View** menu and then select **Developer > Developer Tools**
    b. Or, on macOS press Option (⌥)– Command (⌘)– I
    c. When the developer tools open, select the AEM tab



7. Enter the term **oak** in the search box and click **Go**.



8. In the left pane of AEM Chrome Plug-in **click** the row matching:
   http://localhost:4502/content/summit/l736/chapter-1.html?q=oak



Status	URL	Method	Resp. Time
200	http://localhost:4502/content/summit/l736/chapter-1.html?q=oak	GET	299ms

9. The right panel will update with logging and query information for this request.

10. In the right panel, click the **Queries** tab to view the executed query:

```
QUERY: /jcr:root/content/docs/en/aem/_x0036_-3//element(*,
cq:Page)[(jcr:contains(., 'oak'))]
```

The plan used is displayed below the query:

```
PLAN: [cq:Page] as [a]
/*lucene:cqPageLucene(/oak:index/cqPageLucene) +:fulltext:oak
+:ancestors:/content/docs/en/aem/6-3 ft:("oak") where
(contains([a].[*], 'oak')) and (isdescendantnode([a],
[/content/docs/en/aem/6-3])) */
```

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. We will visit this index definition often throughout this lab.

Pro Tip: *You will notice the query path contains '_x0036_-3' instead of '6-3' (in the path /content/docs/en/aem/6-3). This is due to ISO9075 and the way Lucene handles directory names. When building queries any directories starting with an integer must be prefixed with the string 'x003' then concatenated using an underscore. For example, the folder '56789' would be queried as '_x0035_6789'.*

Full-text operations

1. Try out the following full-text searches in the search box, using the supported operators and note the change in results:
    a. **"sites assets"** – matches only the exact phrase
    b. **sites assets** – both search terms must appear
    c. **sites OR assets** – either search term must appear

Pro Tip: *You will notice that OR is capitalized. Lucene will not recognize this as a search operator if it is lowercase and will instead process it like any other search word.*

Excerpts operations

1. Put the page in **Edit mode**, by clicking **Edit** in the top right.

2. Click the **Search component**.

3. Click the **wrench icon** to edit.

4. In the Search component dialog, set the following:
    a. **Use Excerpt:** checked
    b. Click the **checkmark icon** to save dialog changes

5. Switch to **Preview mode**, by clicking Preview in the top right.

6. Enter the term **sites** in the search box and click **Go.**
    a. Note the excerpts with **term highlighting** in the results

7. Inspect the query with **AEM Chrome Plug-in** and note the **rep:excerpt(.)** function in the query.

```
QUERY: /jcr:root/content/docs/en/aem/_x0036_-3//element(*,
cq:Page)[(jcr:contains(., 'highlighting'))]/rep:excerpt(.)
```

Bonus Exercise

If you finished this exercise early, try the following bonus exercise:

Inspecting the cqPageLucene index definition

1. Open **CRXDE Lite.**
    - http://localhost:4502/crx/de

2. Select the **/oak:index/cqPageLucene** node.

3. Note the core index configurations on the **cqPageLucene** node.

4. Explore under the **cqPageLucene/aggregates** node to see the full-text aggregate configurations. These define the types of nodes and subnodes that will be included in this index.

5. Explore under the **cqPageLucene/indexRules** node to see the property-specific configurations that are defined for this index.

Chapter 2: Indexing and Filtering

AEM supports the creation of custom indexes to speed up returning search results. AEM also supports filtering based on properties to restrict search results, using the following operators:

- Equals
- Not equals
- Ranges

In this exercise, we will leverage search facets to focus our search results and customize an index definition to ensure performant searches when searching on our custom property.

Exercise

Defining an Oak index rule for Tag-based property filtering

1. Open the page at **Sites > Adobe Summit 2017 > L736 > Chapter 2 – Filtering**
   - http://localhost:4502/editor.html/content/summit/l736/chapter-2.html

2. Open the **AEM Chrome Plug-in:**
   a. While in Chrome, open the **View** menu and then select **Developer > Developer Tools.**
   b. Or, on macOS press Option (⌥)– Command (⌘)– I
   c. When the developer tools open, select the AEM tab

3. Switch to Preview mode by clicking **Preview** in the top right.

4. Enter a search term (e.g., *console*) in the page's search box, and click **Go** several times while noting the time that it takes to execute the query.

   (✎ Time taken   milliseconds)

5. Inspect the Query Plan in the AEM Chrome Plugin's **Queries** tab. In this example, the term *console* was used as a search term:

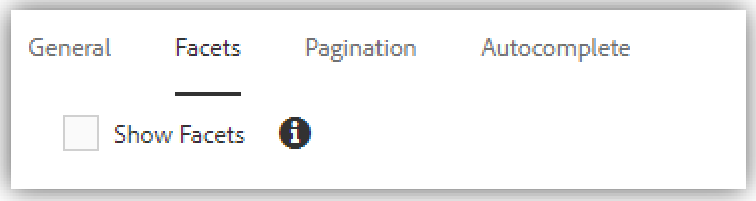
```
QUERY: [cq:Page] as [a] /*
lucene:cqPageLucene(/oak:index/cqPageLucene) +:fulltext:console
+:ancestors:/content/docs/en/aem ft:("console") where
(contains([a].[*], 'console')) and (isdescendantnode([a],
[/content/docs/en/aem])) */
```

6. Note that the cqPageLucene index has been selected for this query.

7. Switch to *Edit* mode by clicking **Edit** in the top right.

8. Select the Search component and click the **wrench icon** to edit the component properties.

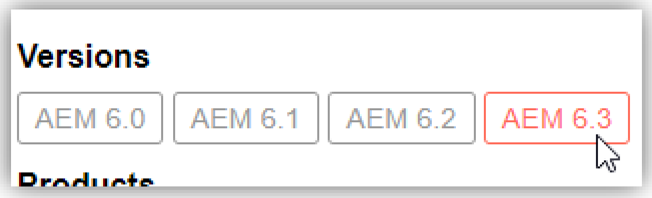
9. In the Search component dialog, set the following:
   a. Click on the *Facets* tab



   b. Check the **Show Facets** checkbox
   c. Click the **checkmark icon** to save dialog changes

10. Switch to Preview mode by clicking **Preview** in the top right.

11. Under **Versions**, click on **AEM 6.3** to filter by pages that have this tag applied:



12. Enter your search term again, i.e., *console*, and click **Go** several times. Note here how long it took to execute the query:

(✎ Time taken   milliseconds)

13. Inspect the Query Plan in the AEM Chrome Plugin's **Queries** tab. In this example, the term *console* was used as a search term:

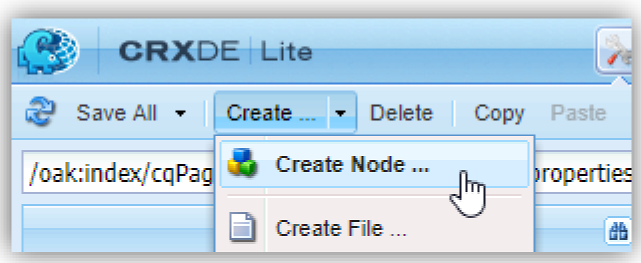
```
Query: [cq:Page] as [a] /*
lucene:cqPageLucene(/oak:index/cqPageLucene) +:fulltext:console
+:ancestors:/content/docs/en/aem ft:("console") where
([a].[jcr:content/cq:tags] in('version:aem63',
'/content/cq:tags/version/aem63')) and (contains([a].[*],
'console')) and (isdescendantnode([a], [/content/docs/en/aem]))
*/
```

14. Note that the cqPageLucene is used to evaluate this query, and there are property restrictions on jcr:content/cq:tags.

15. In a new browser tab, open **CRXDE Lite.**
   - http://localhost:4502/crx/de

16. Select **/oak:index/cqPageLucene/indexRules/cq:Page/properties** node.
    - Each node under **properties** defines how a specific property under the cq:Page hierarchy is indexed
    - Note there is no property index rule for **jcr:content/cq:tags**

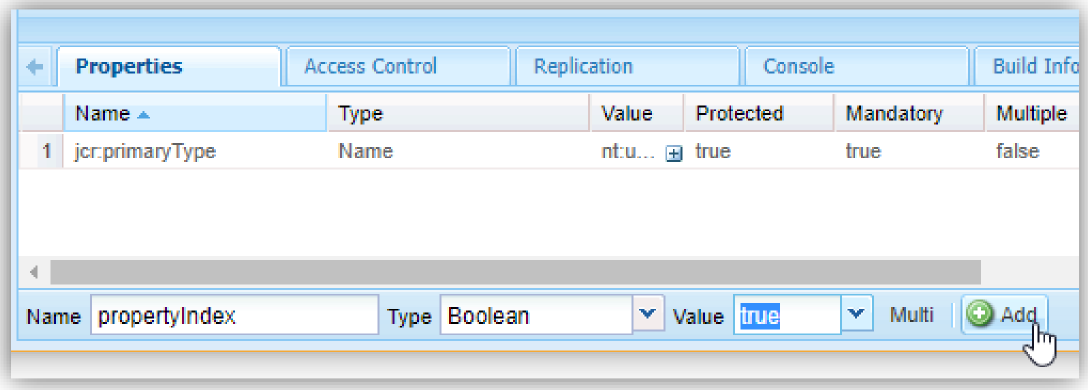
17. Create an index rule for the property [cq:Page]/jcr:content/cq:tags.
    a. While **/oak:index/cqPageLucene/indexRules/cq:Page/properties** is selected, click **Create… > Create Node**



Use the following information:

Node Name	Node Type
cqTags	nt:unstructured

    b. Find the nodes Properties tab on the right side, where you will add new properties to the new **cqTags** node.



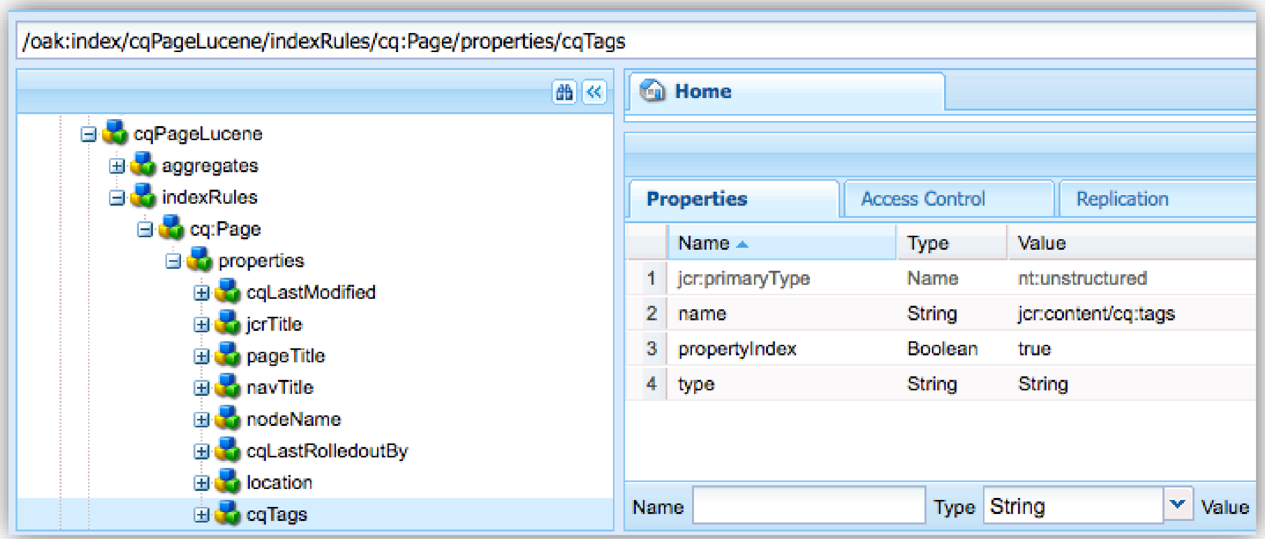
	Name ▲	Type	Value	Protected	Mandatory	Multiple
1	jcr:primaryType	Name	nt:u...	true	true	false

Name propertyIndex Type Boolean Value true Multi Add

Add the following properties:

Property Name	Property Type	Property Value
propertyIndex	Boolean	true
type	String	String
name	String	jcr:content/cq:tags

The properties screen should now appear as below:



/oak:index/cqPageLucene/indexRules/cq:Page/properties/cqTags

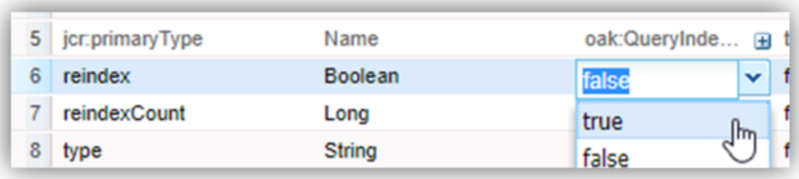
- cqPageLucene
  - aggregates
  - indexRules
    - cq:Page
      - properties
        - cqLastModified
        - jcrTitle
        - pageTitle
        - navTitle
        - nodeName
        - cqLastRolledoutBy
        - location
        - cqTags

	Name	Type	Value
1	jcr:primaryType	Name	nt:unstructured
2	name	String	jcr:content/cq:tags
3	propertyIndex	Boolean	true
4	type	String	String

Name Type String Value

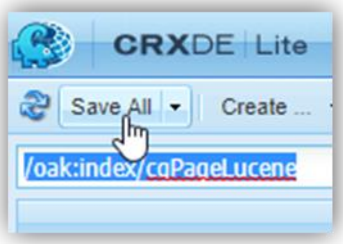
18. Select the node at **/oak:index/cqPageLucene**.

19. Change the value of its **reindex** property to *true* by double-clicking on the property.



5	jcr:primaryType	Name	oak:QueryInde...
6	reindex	Boolean	false
7	reindexCount	Long	
8	type	String	

20. Click **Save All** in the top left to save changes.



/oak:index/cqPageLucene

21. While **/oak:index/cqPageLucene** is selected, click the **Refresh icon** in the top left until the value of the **reindex** property changes back to *false*.

    Be patient! The reindex operation can take as long as a minute or two, but typically completes in a few seconds.

22. Return to the **Chapter 2 – Filtering** page.

23. Ensure the AEM 6.3 filter is still selected, then click **Go** several times to re-issue the query and note the average **Time Taken.**

    The Time Taken should on average be **less than the Time Taken previously in step 4 or step 12**. This is due to the indexed property filter we created.

Warning: While we can quickly rebuild indexes in this small lab environment, it will take a substantial amount of time in a real production environment with large datasets. During this time, some system features may not work as expected. Reindexing should only be done when a change has been made to the index and should usually be planned for an off-hours maintenance window.

Pro Tip: For small content sets like this Lab, the cqPageLucene/aggregates configuration covers [cq:Page]/jcr:content/cq:tags making the property restrictions fast (<100ms). As the body of content grows large (10k's to millions of pages) the index rule greatly improves performance.

Chapter 3: Pagination

Pagination of search results is an important component of any search implementation. AEM's QueryBuilder API includes several options to make pagination easier to implement. In this exercise, we will configure our component to illustrate these pagination features, but our implementation uses the following QueryBuilder properties behind the scenes:

- **p.limit**: Defines the number of results to return for a given query. The default is **10**. A value of **-1** will return all results.
- **p.offset:** A 0-based value that defines where the search results start. Changing this parameter is used to retrieve the next page of search results.
- **p.guessTotal**: Using this parameter can significantly improve the performance of queries that return large result sets because Oak does not need to calculate the exact size of the result set. The disadvantage is that not knowing the exact size can make implementing pagination more difficult.

Exercise

Turn on Pagination

1. Open the page at **Sites > Adobe Summit 2017 > L736 > Chapter 3 – Pagination.**
   - http://localhost:4502/editor.html/content/summit/l736/chapter-3.html

2. While in **Edit** mode, select the Search component and click the **wrench icon** to open the component dialog.

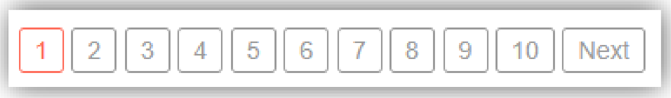
3. Select the **Pagination** Tab.

4. Click the checkbox to **Show Pagination** (leave the other fields as is) and click the **checkmark icon** to save the dialog.

5. Switch the page mode to **Preview** and search for the term *template development*. This will execute a full-text search with multiple pages of results. Note here how long it took to execute the query:

   (✎ Time taken milliseconds)

6. Note the pagination controls at the bottom of the page.

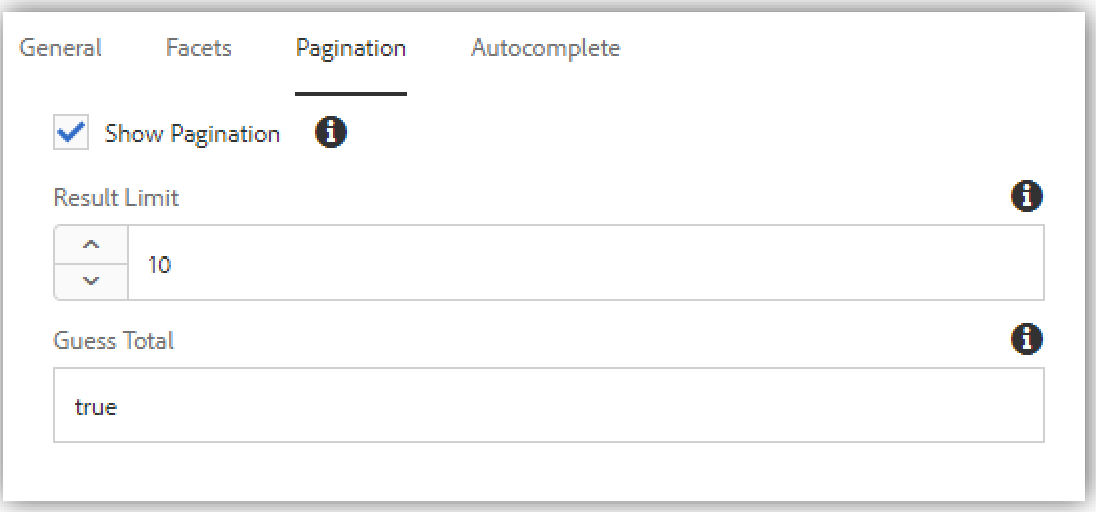


7. Click through the pagination to view different pages of results by clicking **Next**. Click to the last page (you might have to repeatedly click higher numbered pages to get the end). Notice that the **Next** button has disappeared.

Use guessTotal=true to improve performance

1. Switch the page mode to **Edit**, select the Search component and open the component dialog by clicking the **wrench icon**.

2. Navigate to the Pagination tab and type **true** in the ***Guess Total*** field. Click the **checkmark icon** to save and close the dialog.



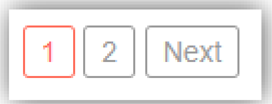
✓ Show Pagination

Result Limit: 10

Guess Total: true

3. Switch the page mode to **Preview** and perform the same keyword search as in the previous step, i.e., *template development*.

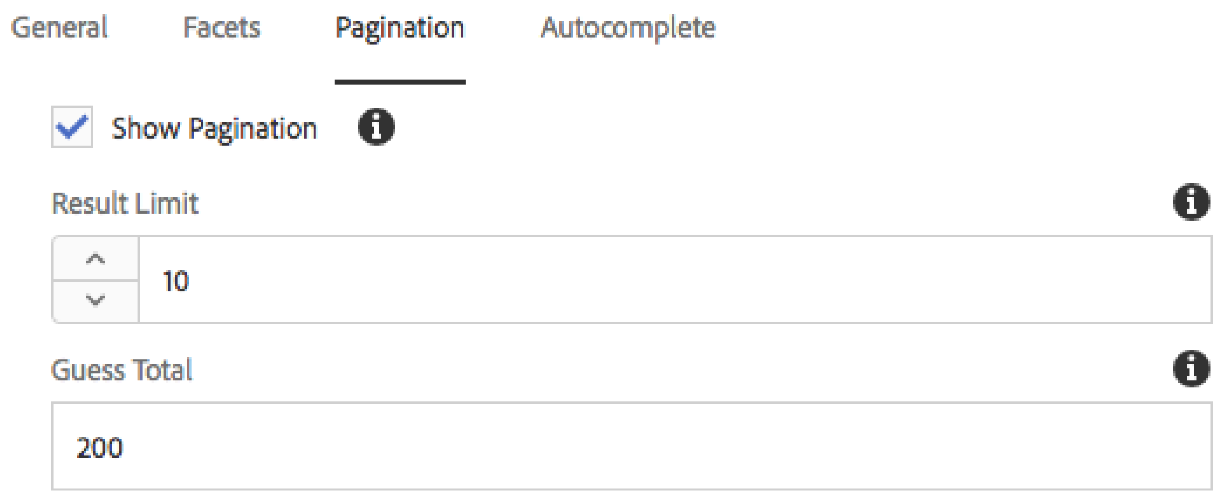
4. Note that the time taken is substantially less than before. Scroll to the bottom of the page and note that the pagination has changed to only show the next page of results. Since we no longer know how many total results there are, we do not know how many pages there are in total. You can still click through to the last page of results, but you will need to click **Next** many more times.



Use guessTotal=200 to read the first 200 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 **200**. Click the **checkmark icon** to save and close the dialog.



Show Pagination: ✓

Result Limit: 10

Guess Total: 200

3. Switch the page mode to **Preview** and perform the same keyword search as in previous steps, i.e., *template development*.

4. Notice that the pagination now shows more than two pages. Click immediately to the last page (10) and then click to the last page again (15).

5. Since **guessTotal** is being used, there is the potential for an extra page of results to be shown. This is because the estimated size of our result set (200) is larger than the size of our actual result set's size.

Pro Tip: For small content sets like we have used in this lab, the difference in response time when using guessTotal may seem trivial, but for larger content sets and more complex queries, it can significantly speed up query performance. We recommend using guessTotal=true whenever returning a fixed size of results or in use cases where knowing the total number of pages is not necessary.

Bonus Exercise

If you have finished this exercise early, try the following bonus exercise:

1. Open the QueryBuilder debugger.
    - http://localhost:4502/libs/cq/search/content/querydebug.html

2. Type the following text in the text area and execute the search:

```
fulltext=aem sites
type=cq:Page
```

3. Notice the number of results and the amount of time it took to execute.

4. Perform the same query but with **guessTotal** turned on:

```
fulltext=aem sites
type=cq:Page
p.guessTotal=true
```

5. Note that while you no longer get the exact number of hits, the time taken to return the results has decreased.

6. Perform the same query but configure it to return all search results:

```
fulltext=aem sites
type=cq:Page
p.limit=-1
```

7. Note that you will receive over 1,000 results. Note that while this may seem convenient at first glance, there may be performance implications to processing this many results all at once in a production application.

Pro Tip: The QueryBuilder Debugger is an essential tool for any AEM Query Developer and is a great way to quickly debug queries.

Reference Links

https://docs.adobe.com/docs/en/aem/6-2/develop/search/querybuilder-api.html

Chapter 4: 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. In this exercise, we will configure stemming to illustrate how analyzers are configured as part of the index.

Analyzers are also available for synonyms, stop words, and HTML stripping. While we do not have time to cover configuring all of these analyzers in the lab, bonus exercises have been provided that cover their configuration. Please see the Oak documentation for more information.

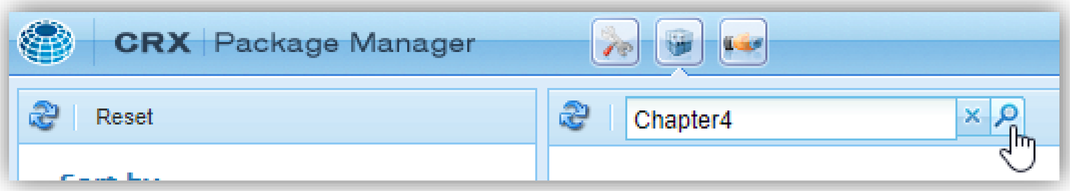
Setup Package

For this exercise, we will need to install the package, **L736-Chapter4-Setup.zip**, via the CRX Package Manager. This package augments the **/oak:index/cqPageLucene** index with the following basic analyzer configurations:

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

1. Navigate to the CRX Package Manager: **AEM Start > Tools > Deployment > Packages.**
   - http://localhost:4502/crx/packmgr/index.jsp

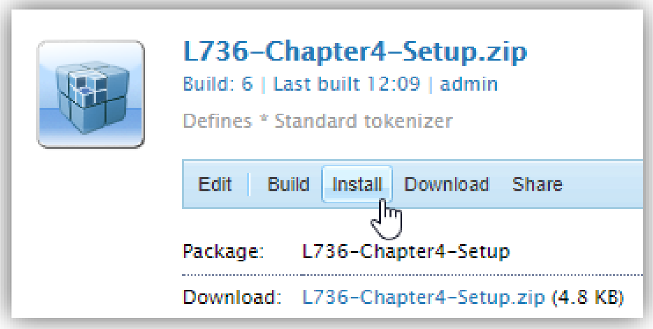
2. Search for **Chapter4.**



3. Find the package named **L736-Chapter4-Setup.zip.**

IMPORTANT Do not install the **L736-Chapter4-*Solution*.zip** package!

4. Click **Install**


L736-Chapter4-Setup.zip
Build: 6 | Last built 12:09 | admin
Defines * Standard tokenizer

Package: L736-Chapter4-Setup
Download: L736-Chapter4-Setup.zip (4.8 KB)

Exercise

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

Stemming is 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:

1. Given the provided term: **developing**
2. The stemmer will derive the root word: **develop**
3. This includes content that contains derived forms such as **developer**, and **development**.

In this exercise, we will configure the Porter Stemmer to enable stemming on our cqPageLucene index:

1. Open the page at **Sites > Adobe Summit 2017 > L736 > Chapter 4 – Analyzers**
   - http://localhost:4502/editor.html/content/summit/l736/chapter-4.html

2. Perform three different searches and note the number of results that each of them returns:
   - **development** (✎ results)
   - **developer** (✎ results)
   - **developing** (✎ results)

3. Note that the **number of results is different** between all three searches.

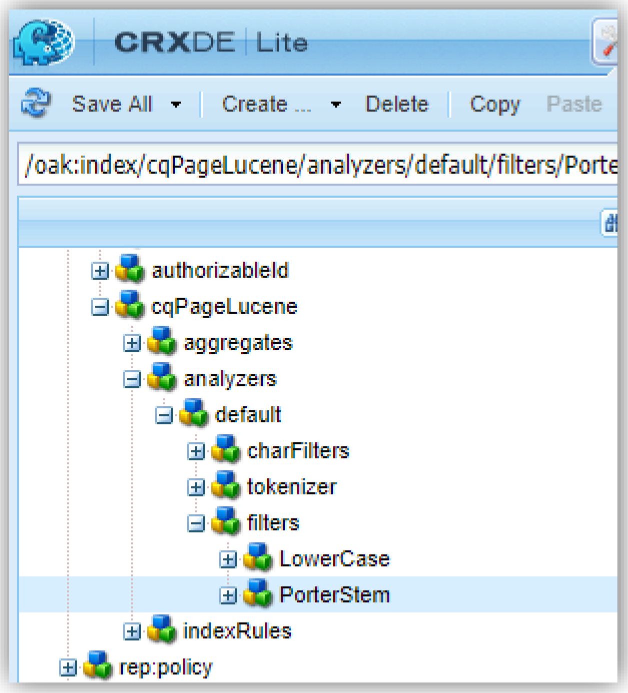
4. Open **CRXDE Lite** in a second tab:
   - http://localhost:4502/crx/de

5. Select **/oak:index/cqPageLucene** and click the **Refresh icon** in the top left of the CRXDE interface.

6. Create a new node under **/oak:index/cqPageLucene/analyzers/default/filters** using the below information:

Node Name	**Node Type**
`PorterStem`	`nt:unstructured`

7. Make sure the new PorterStem node is positioned below the LowerCase node, as shown below:



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

```
authorizableId
cqPageLucene
  aggregates
  analyzers
    default
      charFilters
      tokenizer
      filters
        LowerCase
        PorterStem
  indexRules
rep:policy
```

8. Select the node at **/oak:index/cqPageLucene**.

9. Change the value of the **reindex** property to *true*.

10. Click **Save All** in the top left to save changes.

11. While **/oak:index/cqPageLucene** is selected, click the **Refresh icon** in the top left a few times, until the value of the **reindex** property changes back to *false*.

12. Return to the **Chapter 4 – Analyzers** page and perform the three searches from step 2.

13. Note that all three terms now return the same number of results. This is because the PorterStemmer stemmed both the indexed terms and the query terms to the stem ***develop***.

Bonus Exercise

If you have finished this exercise early, try this bonus exercise.

Configure Synonyms

Synonyms allow different terms with equivalent meaning to be considered the same by full-text search. In this exercise, we will configure synonyms on our cqPageLucene index:

1. Open the page at **Sites > Adobe Summit 2017 > L736 > Chapter 4 – Analyzers.**
    - http://localhost:4502/editor.html/content/summit/l736/chapter-4.html

2. Perform a search using the term ***sightly*** and note the number results.

3. Perform a search using the term ***HTL*** and note that a different number of results are returned.

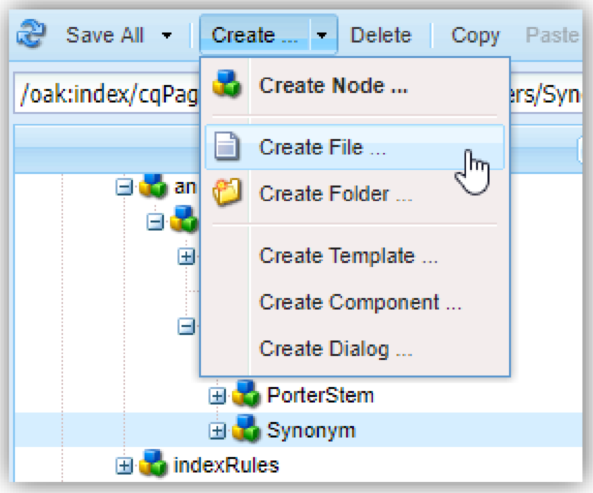
4. Open CRXDE Lite in a second browser tab:
    - http://localhost:4502/crx/de

5. Select **/oak:index/cqPageLucene/analyzers/default/filters** and create a new node named **Synonym** using the following information:

Node Name	Node Type
`Synonym`	`nt:unstructured`

6. Click **Save All**.

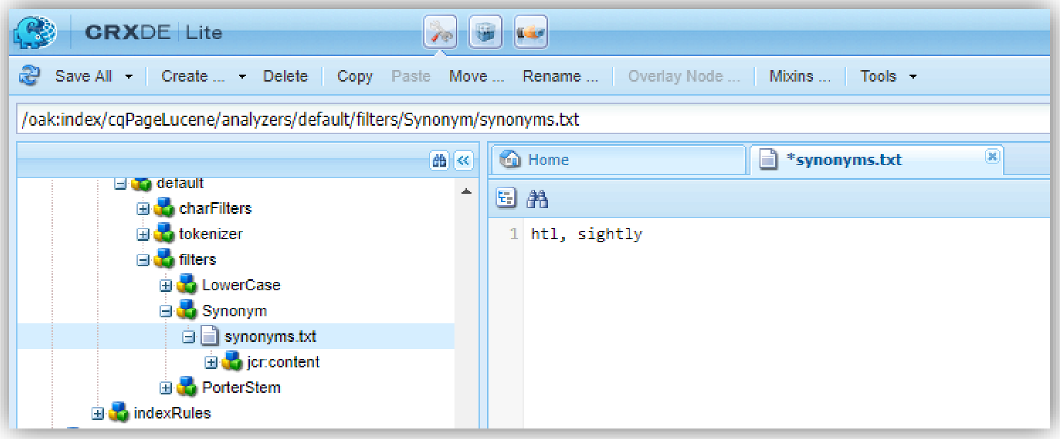
7. With the Synonym node selected, click **Create File** and name the new file **synonyms.txt.**



This will create a file and open an editor for you in the main content area.

8. In the main content area, enter the following text:

```
htl, sightly
```



/oak:index/cqPageLucene/analyzers/default/filters/Synonym/synonyms.txt

```
default
  charFilters
  tokenizer
  filters
    LowerCase
    Synonym
      synonyms.txt
        jcr:content
    PorterStem
indexRules
```

```
htl, sightly
```

9. Add the property **synonyms** to the node **/oak:index/cqPageLucene/indexRules/analyzers/default/filters/Synonym**

Property Name	Property Type	Property Value
`synonyms`	`String`	`synonyms.txt`

10. Click **Save All** in the top left to save changes.

11. If it isn't already, drag the new Synonym node to be positioned **after** LowerCase and **before** the PorterStem node, as seen above, and click **Save All.**

12. Select the node at **/oak:index/cqPageLucene**.

13. Change the value of the **reindex** property to *true*.

14. Click **Save All** in the top left to save changes.

15. While **/oak:index/cqPageLucene** is selected, click the **Refresh icon** in the top left a few times, until the value of the **reindex** property changes back to *false*.

    Be patient! The reindex operation can take as long as a minute or two, but typically completes in a few seconds.

16. Return to the **Chapter 4 – Analyzers** page and ensure you are in **Preview** mode.

17. Search for Sightly and HTL as in step 2. Note that they now yield the same number of results (~17 results), as they are now considered equivalent terms, i.e., synonyms.

Chapter 5: Addressing Query Traversal

Slow queries will impact the user experience of an AEM instance, so it is important to understand how to identify, troubleshoot, and tune queries. For this exercise, we will execute a QueryBuilder-based query, listing all the component nodes that were rolled out by the msm-service user and then order the nodes descending by their roll-out date.

Exercise

Explore traversing queries

1. On the Desktop, open the following folders: AEM > crx-quickstart > logs. Double-click to **open the error.log** file. The log file should open in Console app for macOS.

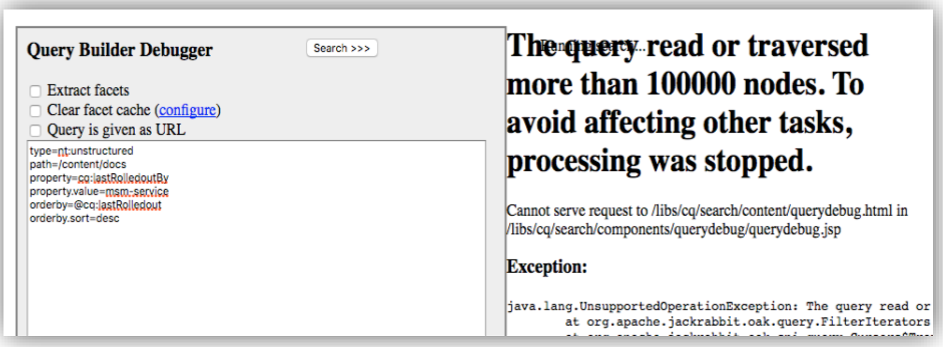
2. In **Chrome**, navigate to **QueryBuilder Debugger**:
    - http://localhost:4502/libs/cq/search/content/querydebug.html

3. Copy the following query into the QueryBuilder and click **Search**.

```
type=nt:unstructured
path=/content/docs
property=cq:lastRolledoutBy
property.value=msm-service
orderby=@cq:lastRolledout
orderby.sort=desc
```

4. While the query executes, watch the logs in the Console app. Note the ***WARN*** messages about query traversal.

5. After several seconds, the query will fail and an exception will appear on the QueryBuilder Debugger web page.



```
type=nt:unstructured
path=/content/docs
property=cq:lastRolledoutBy
property.value=msm-service
orderby=@cq:lastRolledout
orderby.sort=desc
```

The query read or traversed more than 100000 nodes. To avoid affecting other tasks, processing was stopped.

Cannot serve request to /libs/cq/search/content/querydebug.html in /libs/cq/search/components/querydebug/querydebug.jsp

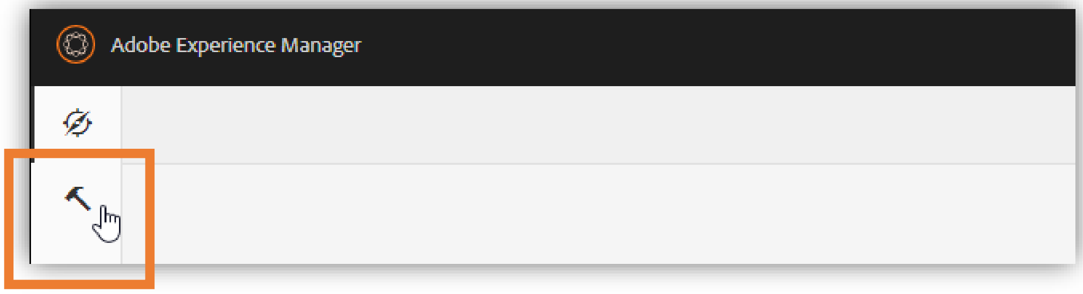
Exception:

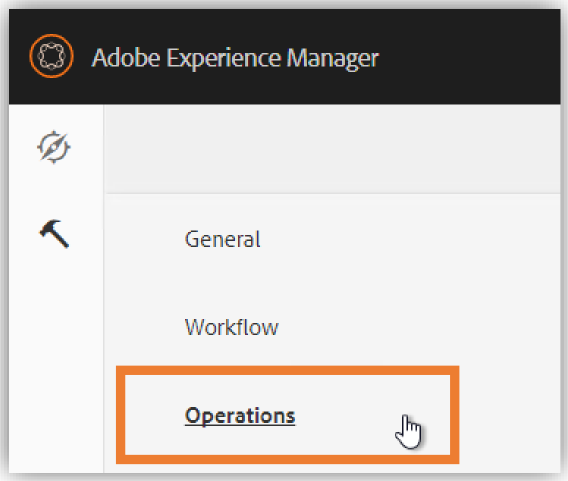
```
java.lang.UnsupportedOperationException: The query read or
        at org.apache.jackrabbit.oak.query.FilterIterators
```

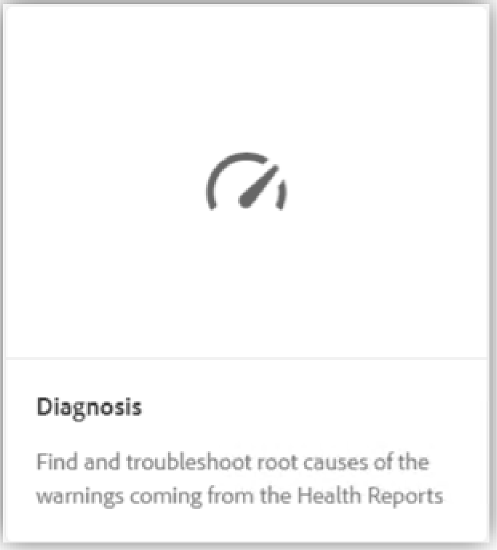
6. Locate the executed XPath query in the logs of the Console app. It is output for each traversal warning immediately after the *xpath* property. The query is bolded in the sample log excerpt below:

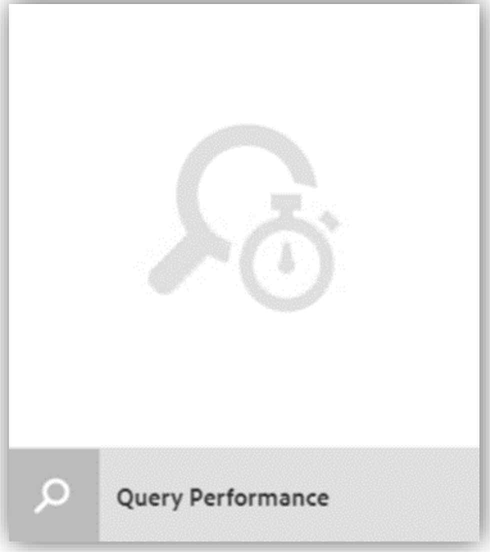
```
08.02.2018 16:24:23.801 *WARN* [0:0:0:0:0:0:0:1 [1518135861600]
GET /libs/cq/search/content/querydebug.html HTTP/1.1]
org.apache.jackrabbit.oak.plugins.index.Cursors$TraversingCursor
Traversed 98000 nodes with filter Filter(query=select [jcr:path],
[jcr:score], * from [nt:unstructured] as a where
[cq:lastRolledoutBy] = 'msm-service' and isdescendantnode(a,
'/content/docs') order by [cq:lastRolledout] desc /* xpath:
/jcr:root/content/docs//element(*,
nt:unstructured)[(@cq:lastRolledoutBy = 'msm-service')] order by
@cq:lastRolledout descending */, path=/content/docs//*,
property=[cq:lastRolledoutBy=[msm-service]]); consider creating
an index or changing the query
```

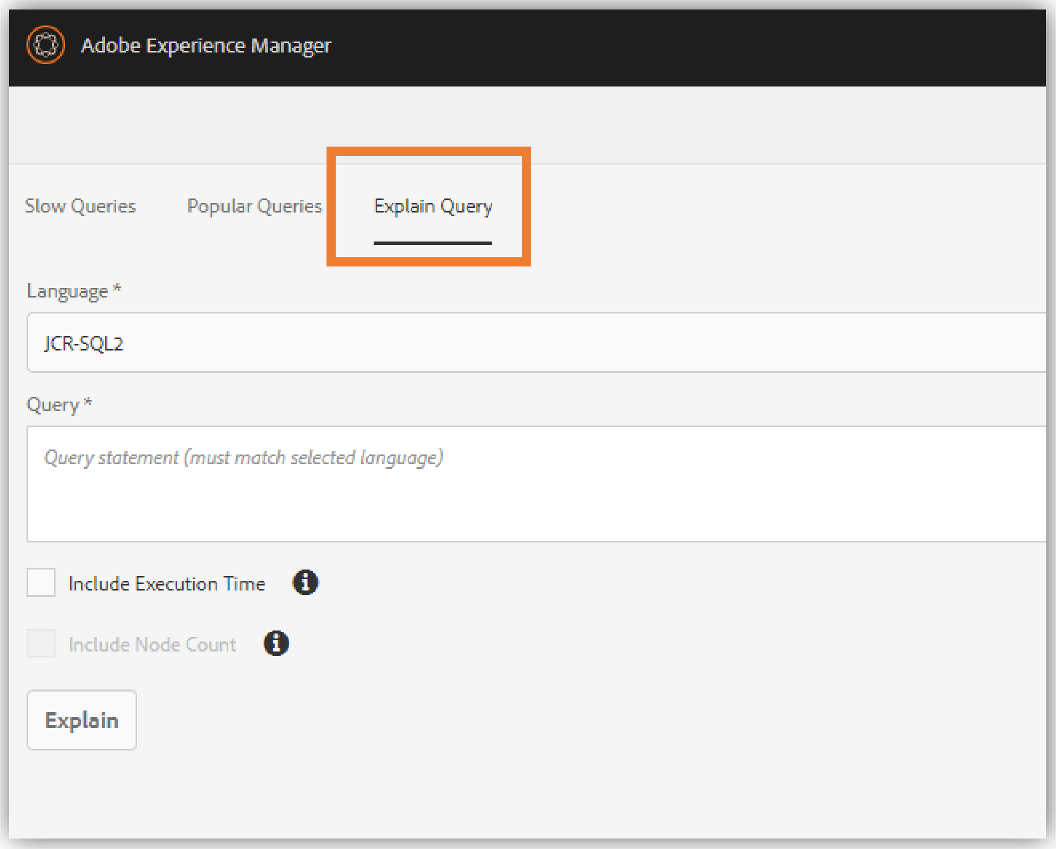
7. In a new browser tab, open the AEM start screen and navigate to the Explain Query tool by opening **Tools > Operations > Diagnosis > Query Performance** and then click on the **Explain Query** tab.




Adobe Experience Manager
General
Workflow
Operations


Diagnosis
Find and troubleshoot root causes of the warnings coming from the Health Reports

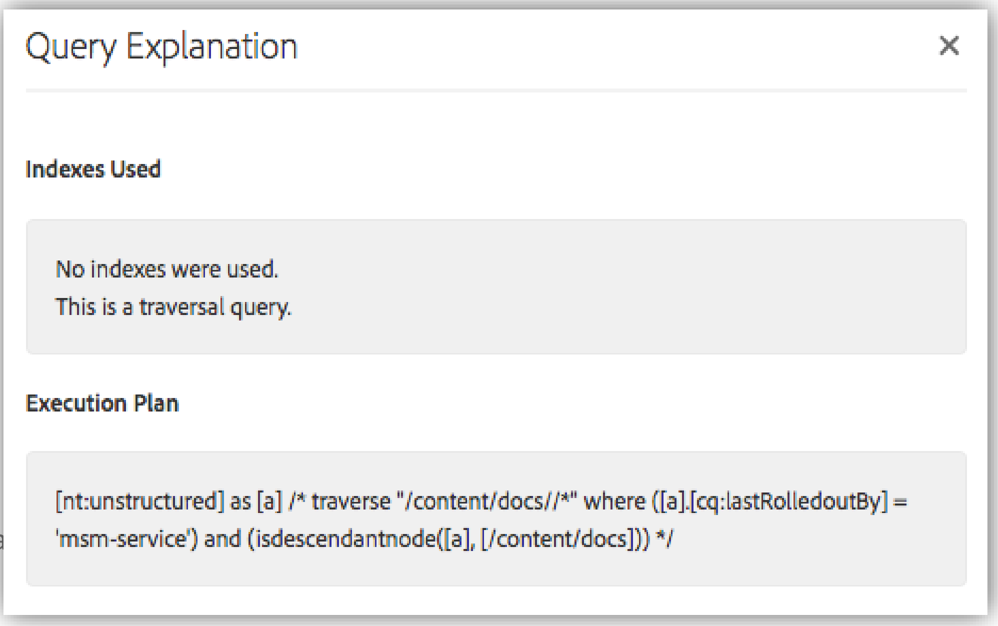

Query Performance



8. Set the **Language** as **XPath**.

9. Paste the XPath **query found in the logs in step 6** into the **Query text box**, and click **Explain.**

10. Note that the explanation tells us that this is a traversal query and that no indexes were used.



Indexes Used

No indexes were used.
This is a traversal query.

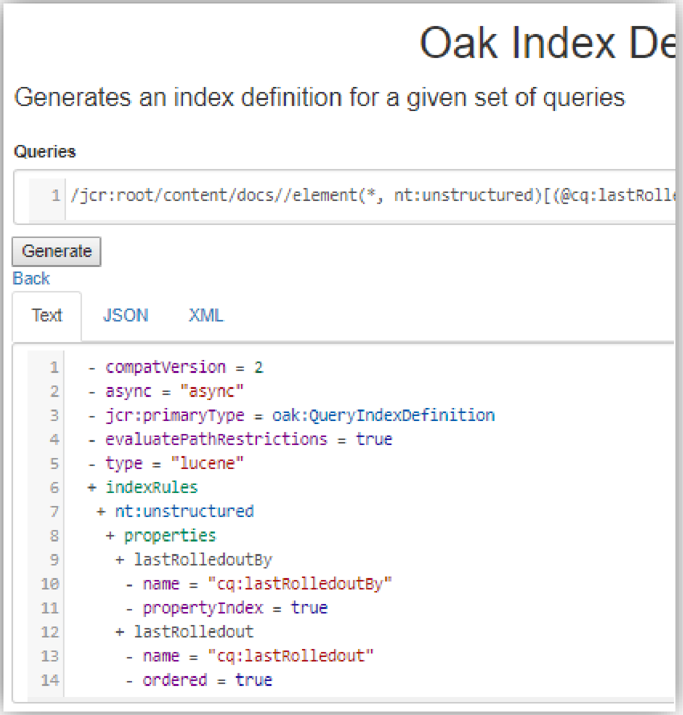
Execution Plan

```
[nt:unstructured] as [a] /* traverse "/content/docs//*" where ([a].[cq:lastRolledoutBy] = 'msm-service') and (isdescendantnode([a], [/content/docs])) */
```

11. In a new browser tab, navigate to **Oak Index Definition Generator**
    - http://oakutils.appspot.com/generate/index

12. Replace the contents of the **Queries text box** with the **XPath statement** used in step 10 and click **Generate**.

13. The second text box populates with the Oak index definition required to satisfy the query. Note that the specified index rule is for an *nt:unstructured* node and indexes the cq:*lastRolledOutBy* and *cq:lastRolledOut* properties.



Oak Index De

Generates an index definition for a given set of queries

Queries

```
/jcr:root/content/docs//element(*, nt:unstructured)[(@cq:lastRoll
```

Generate
Back

Text JSON XML

```
- compatVersion = 2
- async = "async"
- jcr:primaryType = oak:QueryIndexDefinition
- evaluatePathRestrictions = true
- type = "lucene"
+ indexRules
 + nt:unstructured
  + properties
   + lastRolledoutBy
    - name = "cq:lastRolledoutBy"
    - propertyIndex = true
   + lastRolledout
    - name = "cq:lastRolledout"
    - ordered = true
```

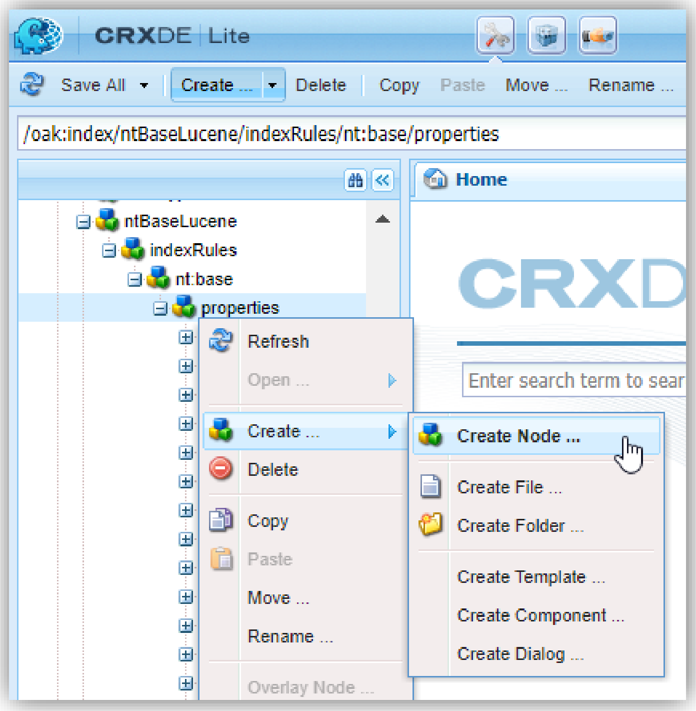
14. As nt:unstructured is a child node of nt:base, this node type is already covered by an index in AEM 6.4. To prevent creating overlapping indexes, we will merge the recommendations from the **Oak Index Definition Generator** into our existing ntBaseLucene index.

15. Open **CRXDE Lite** in a second tab:
    a. http://localhost:4502/crx/de

16. In the left side panel expand the **oak:index** tree and navigate to the **ntBaseLucene** index.

17. Expand the **ntBaseLucene/indexRules/nt:base/properties** nodes.

18. Right click on the **properties** node and choose **Create > Create Node**.



19. Create a node with **type: nt:unstructured** named **lastRolledoutBy**.

20. Add the following properties to this node:

Property Name	Property Type	Property Value
`name`	`String`	`cq:lastRolledoutBy`
`propertyIndex`	`Boolean`	`true`

21. Click **Save All.**

22. Right click on the **properties node** again and choose **Create > Create Node.**

23. Create another new **nt:unstructured** node named **lastRolledout**.

24. Add the following properties to this node:

Property Name	Property Type	Property Value
`name`	`String`	`cq:lastRolledout`
`propertyIndex`	`Boolean`	`true`

25. Click **Save All.**

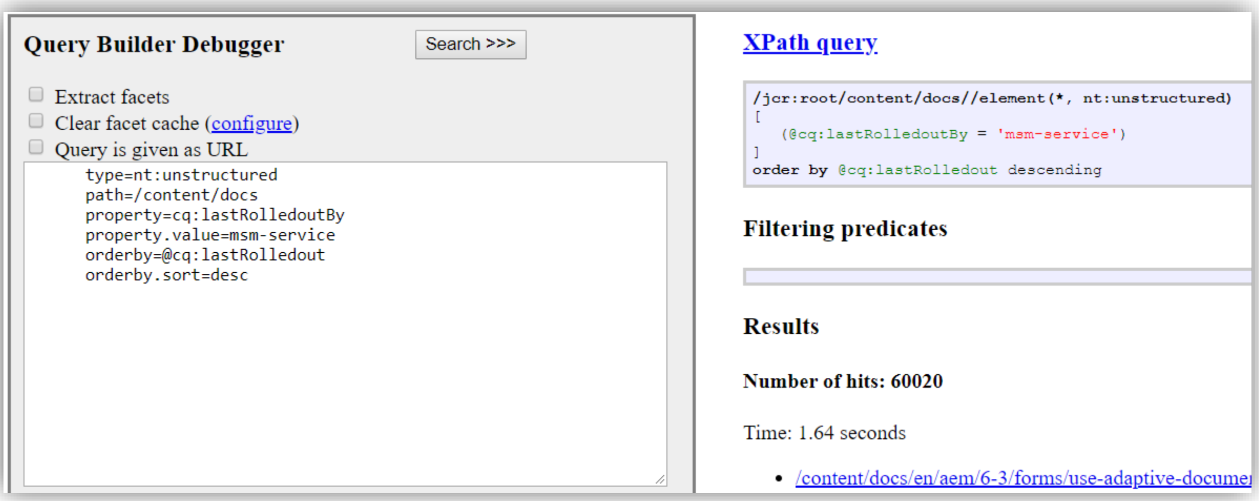
26. Select the node at **/oak:index/ntBaseLucene**.

27. Change the value of the **reindex** property to *true*.

28. Click **Save All** in the top left to save changes.

29. While **/oak:index/ntBaseLucene** is selected, click the **Refresh icon** in the top left a few times, until the value of the **reindex** property changes back to *false*.

30. Execute the query from step 3 again. Note that we now receive results.



```
type=nt:unstructured
path=/content/docs
property=cq:lastRolledoutBy
property.value=msm-service
orderby=@cq:lastRolledout
orderby.sort=desc
```

```
/jcr:root/content/docs//element(*, nt:unstructured)
[
   (@cq:lastRolledoutBy = 'msm-service')
]
order by @cq:lastRolledout descending
```

Number of hits: 60020

Time: 1.64 seconds

- /content/docs/en/aem/6-3/forms/use-adaptive-documen

31. Using the **Explain Query** tool, note that our index is now being selected in the execution plan.

Pro Tip: On real projects, the XML node definition can be copied and pasted into the AEM Code Project for controlled deployment.

Pro Tip: nt:unstructured is a very low-level node type, and the resulting index will be quite large. In real-world scenarios, try to select a more specific node type to ensure that your indexes and searches will be more targeted and performant.

Additional Exercises

As one would expect of an enterprise CMS and DAM, AEM provides robust search capabilities. As a result, there is much more to understanding search in AEM than can be covered in a single lab session. To provide you with a taste of other search features, we have provided the following additional exercises.

Code required to complete all lab exercises will be made available at:

- https://github.com/Adobe-Marketing-Cloud/Summit2018/tree/L736.

Suggestions

Suggestions provide lists of terms or phrases that exist in the content and match a user-provided initial search term. There are two types of suggestion configurations:

Property-based
- Returns the entire value (multi-word) of a property as a suggested term.

Aggregate-based
- Returns a list of single-word terms that match the user-provided search term.

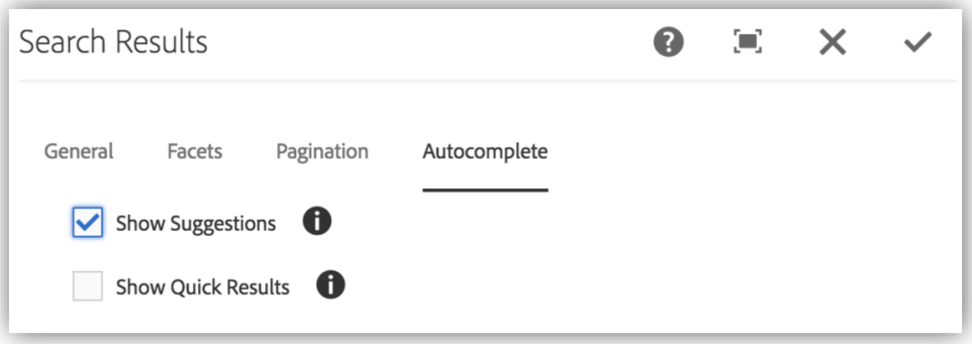
In this exercise, we will look at the default property-based behavior of search suggestions and then configure an index to used aggregate-based suggestions instead.

Open any of the exercise chapter pages.
- e.g., http://localhost:4502/editor.html/content/summit/l736/chapter-1.html

1. While in **Edit** mode, select the Search component and click the **wrench icon** to open the component dialog.

2. Select the **Autocomplete** Tab.

3. Check the box for **Show Suggestions.**



Show Suggestions: ✓
Show Quick Results: ☐

4. Save and close the dialog by clicking on the **checkmark icon** in the upper-right.

Test search suggestions

1. Switch to **Preview** mode.

2. Start typing the search term **metadata** into the search field.
   a. Note that multi-word suggestions appear as the term is typed; because out of the box, property-based suggestions are used.
   b. Click on a suggestion to search and note the number of search results.

3. Open CRXDE Lite in a second browser tab:
   - http://localhost:4502/crx/de

4. Navigate to **/oak:index/cqPageLucene/indexRules/cq:Page/properties.**
   a. Review the properties of the child nodes:
      i. **jcrTitle**
      ii. **nodeName**
   b. Note they both have the **useInSuggest** property set to **true**, which is why the current suggestions are multi-word values of these two properties.

5. Select **/oak:index/cqPageLucene** and create a new node named **suggestion**.

Node Name	Node Type
`suggestion`	`nt:unstructured`

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

Property Name	Property Type	Property Value
`suggestAnalyzed`	`Boolean`	`true`

7. Select the node at **/oak:index/cqPageLucene**.

8. Change the value of the **reindex** property to *true*.

9. Click **Save All** in the top left to save changes.

10. While **/oak:index/cqPageLucene** is selected, click the **refresh icon** in the top left a few times, until the value of the **reindex** property changes back to *false* and the **reindexCount** property is incremented.

11. Return to the exercise page.

12. Start typing in a search query and note how the suggestions are now single search terms rather than multi-word values.

13. Click on a suggestion to search and note the number of results.

Stop Words

Stop words are effectively a blacklist of words that will not be added to the search index and are thus unsearchable. Managed industries may add subjective terms as stop terms, and search implementations for user-generated content may leverage them to keep profanities from being returned.

Open any of the exercise chapter pages.

- e.g., http://localhost:4502/editor.html/content/summit/l736/chapter-1.html

1. Perform searches using the following keywords and note the large number of results.
   - **jsp**
   - **geometrixx**
   - **classic ui**

2. Open CRXDE Lite.
   - http://localhost:4502/crx/de

3. Create a node named **Stop** under **/oak:index/cqPageLucene/analyzers/default/filters**

Node Name	Node Type
`Stop`	`nt:unstructured`

4. Move the Stop node below the Synonym node.

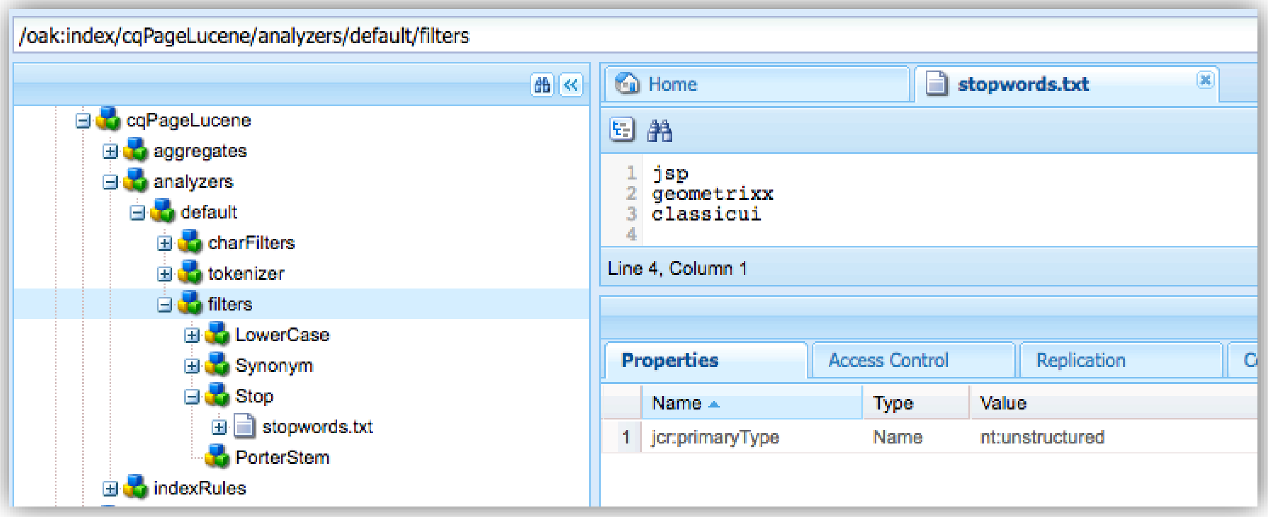
5. Click **Save all** in the top left.

6. Create a File named **stopwords.txt** under **/oak:index/cqPageLucene/analyzers/default/filters/Stop**

Node Name	Node Type
`stopwords.txt`	`nt:file`

7. Double-click to **edit stopwords.txt**, and enter a few stop words, one per line, for example:

```
jsp
geometrixx
classicui
```



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

```
jsp
geometrixx
classicui

```

Line 4, Column 1

	Name	Type	Value
1	jcr:primaryType	Name	nt:unstructured

8. Add the property **words** to **/oak:index/cqPageLucene/indexRules/analyzers/default/filters/Stop**

Property Name	Property Type	Property Value
`words`	`String`	`stopwords.txt`

9. Select the node at **/oak:index/cqPageLucene**.

10. Change the value of the **reindex** property to *true*.

11. Click **Save All** in the top left to save changes.

12. While **/oak:index/cqPageLucene** is selected, click the **refresh icon** in the top left a few times, until the value of the **reindex** property changes back to *false* and the **reindexCount** property is incremented.

13. Perform the three searches in step 1 and note now there are **no results.**

HTML Stripping

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

Open any of the exercise chapter pages.

- e.g., http://localhost:4502/editor.html/content/summit/l736/chapter-1.html

1. Perform a search using the keyword **Tahoma.**

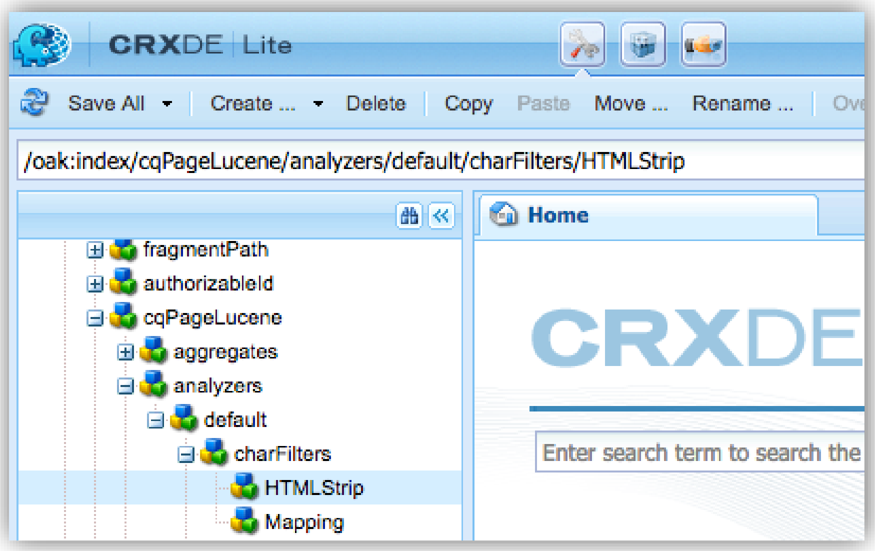
2. Click on any result, view the source on the result page using the developer tools, and search for **tahoma** in the HTML source. Note that it only appears as part of an HTML attribute.

3. Open CRXDE Lite
   - http://localhost:4502/crx/de

4. Create a new node named **HTMLStrip** under **/oak:index/cqPageLucene/analyzers/default**

Node Name	Node Type
`HTMLStrip`	`nt:unstructured`

5. Move the new HTMLStrip node to be **above** the Mapping node.



/oak:index/cqPageLucene/analyzers/default/charFilters/HTMLStrip

6. Select the node at **/oak:index/cqPageLucene**.

7. Change the value of the **reindex** property to *true*.

8. Click **Save All** in the top left to save changes.

9. While **/oak:index/cqPageLucene** is selected, click the **refresh icon** in the top left a few times, until the value of the **reindex** property changes back to *false* and the **reindexCount** property is incremented.

10. Perform the search from step 1 and notice there are no results.

Pro-tip: When possible, avoid storing HTML, CSS or JavaScript in jcr properties!

Boosting

Lucene full-text indexing supports the ability to boost or weight specific metadata properties. This allows specified properties to be ranked higher than others; thus, when a search term is found in a boosted property, the result is moved up in the search results.

Open any of the exercise chapter pages.

- e.g., http://localhost:4502/editor.html/content/summit/l736/chapter-1.html

1. Perform a search with the search term **forms**
   - Note the top result titled **Overview;** this does not have the keyword **forms** in the title

2. Keep this tab open for comparing results after we enable boosting.

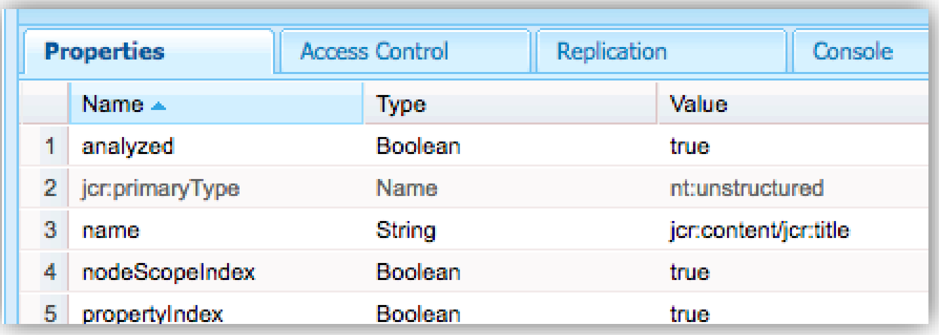
Update cqPageLucene index to enable boosting

1. In a new tab navigate to CRXDE Lite:
   - http://localhost:4502/crx/de/index.jsp

2. In the left side panel expand the oak:index tree and navigate to **/oak:index/cqPageLucene/indexRules/cq:Page/properties/jcrTitle**

3. Add the following property to the **jcrTitle** node:

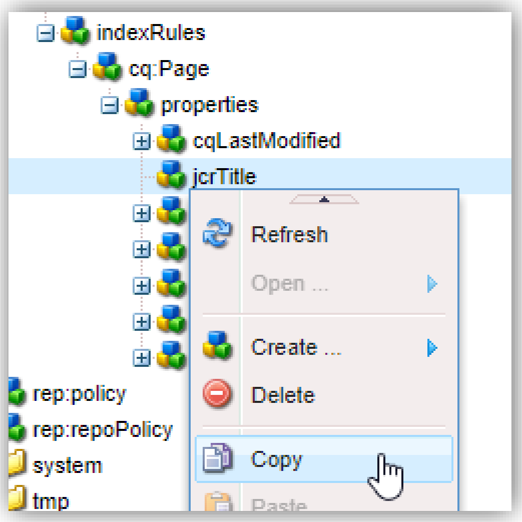
Property Name	Property Type	Property Value
`analyzed`	`Boolean`	`true`



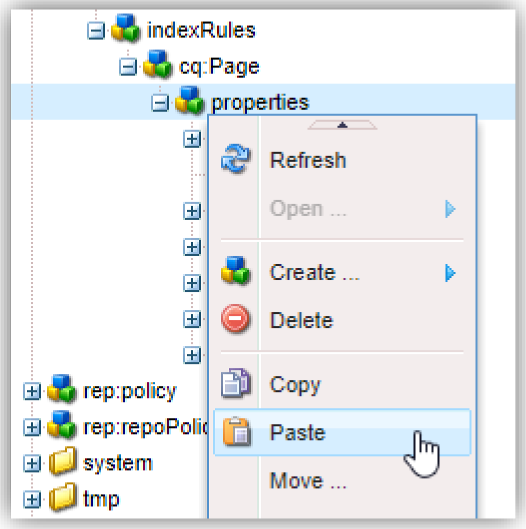
	Name	Type	Value
1	analyzed	Boolean	true
2	jcr:primaryType	Name	nt:unstructured
3	name	String	jcr:content/jcr:title
4	nodeScopeIndex	Boolean	true
5	propertyIndex	Boolean	true

4. Click **Save All** in the upper left corner to save the changes to the node properties.

5. Right-click the **jcrTitle** node and select **Copy** from the menu.

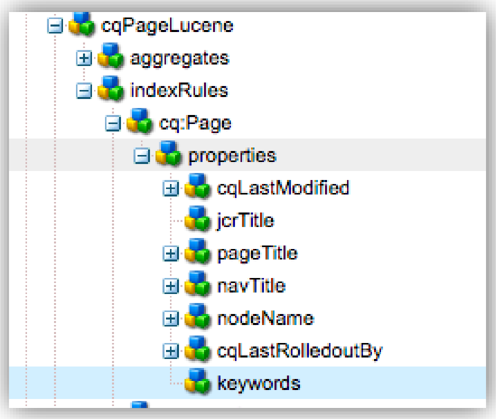


6. Right-click the **properties** node (parent of the jcrTitle node) and click **Paste.**



7. A new node will be created named **Copy of jcrTitle**.

8. Right-click this node and rename to **keywords.**



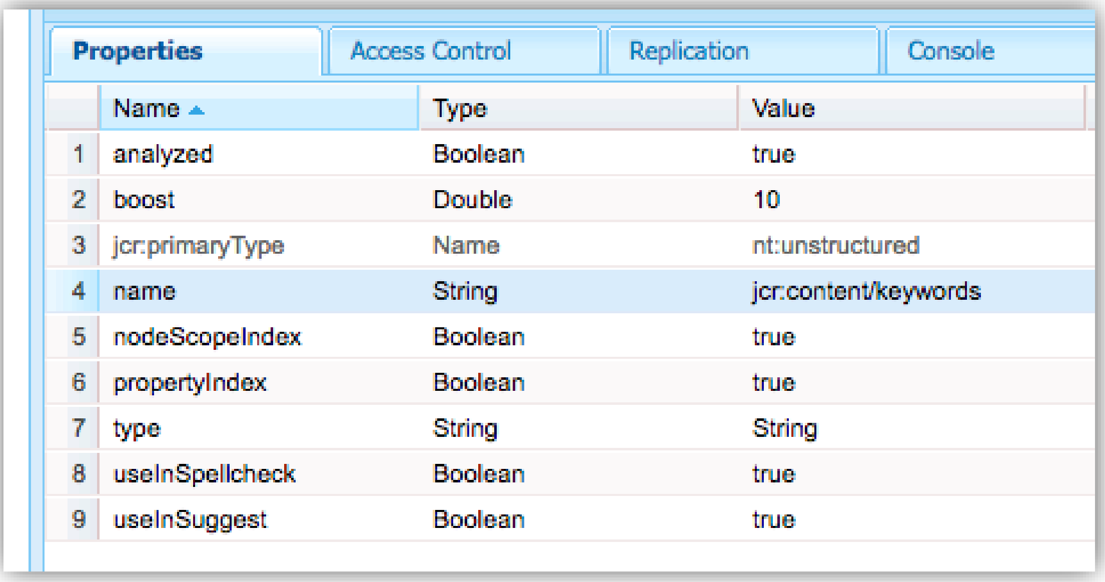
9. Right-click the keywords node and click **Refresh**. The properties of the node should now appear on the right side.

10. Update the **name** property from **jcr:content/jcr:title** to **jcr:content/keywords**

11. Add a new property with the following values:

Property Name	**Property Type**	**Property Value**
`boost`	`Double`	`10`

12. The keyword node should now have the following properties:



	Name	Type	Value
1	analyzed	Boolean	true
2	boost	Double	10
3	jcr:primaryType	Name	nt:unstructured
4	name	String	jcr:content/keywords
5	nodeScopeIndex	Boolean	true
6	propertyIndex	Boolean	true
7	type	String	String
8	useInSpellcheck	Boolean	true
9	useInSuggest	Boolean	true

13. Click **Save All**.

14. Select the node at **/oak:index/cqPageLucene**.

15. Change the value of the **reindex** property to *true*.

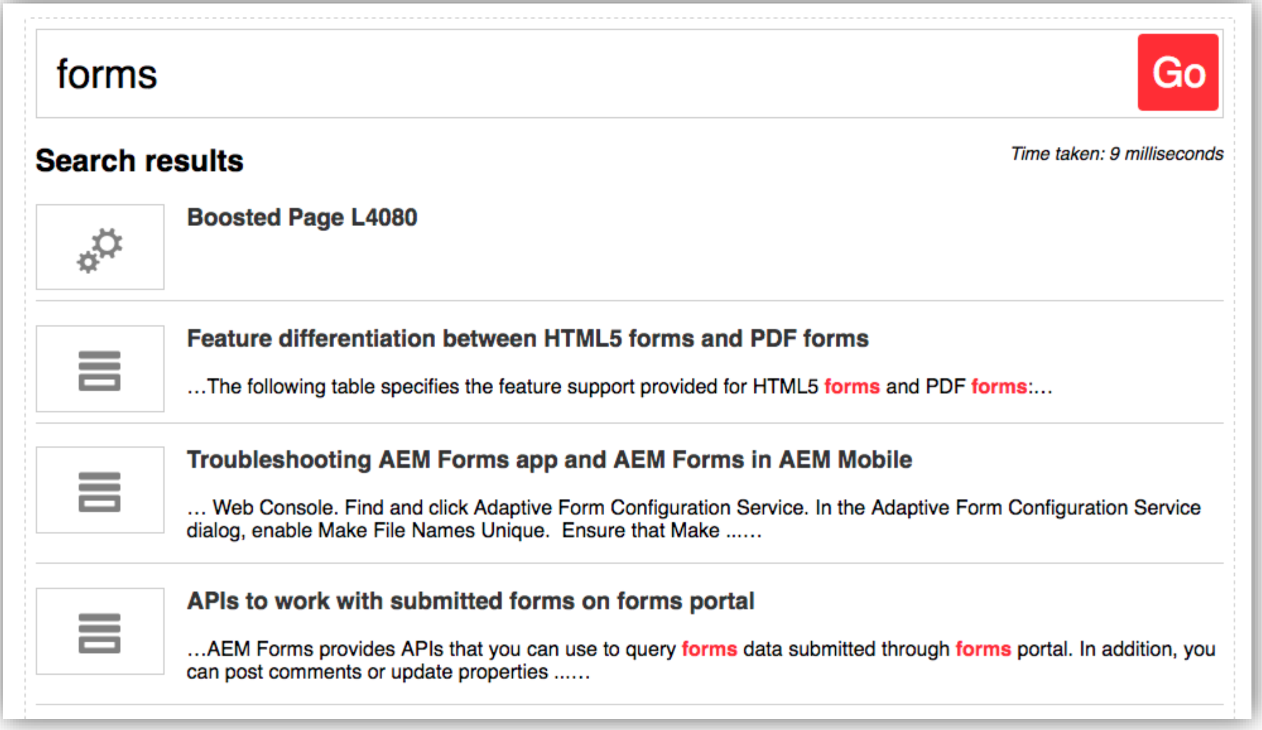
16. Click **Save All** in the top left to save changes.

17. While **/oak:index/cqPageLucene** is selected, click the **refresh icon** in the top left a few times, until the value of the **reindex** property changes back to *false* and the **reindexCount** property is incremented.

Perform Search with boosting

1. Open a new tab and navigate to the page that you used for the beginning of this exercise. (Use a different tab than at the beginning of the exercise so that you can compare results)
   - e.g., http://localhost:4502/editor.html/content/summit/l736/chapter-1.html

2. In Preview mode, perform a search of the keyword **forms.**



Search results

Time taken: 9 milliseconds

Boosted Page L4080

Feature differentiation between HTML5 forms and PDF forms

...The following table specifies the feature support provided for HTML5 forms and PDF forms:...

Troubleshooting AEM Forms app and AEM Forms in AEM Mobile

... Web Console. Find and click Adaptive Form Configuration Service. In the Adaptive Form Configuration Service dialog, enable Make File Names Unique. Ensure that Make ......

APIs to work with submitted forms on forms portal

...AEM Forms provides APIs that you can use to query forms data submitted through forms portal. In addition, you can post comments or update properties ......

The page of results returned should be different than at the beginning of the exercise.

3. The first result should be a page named **Boosted Page L736**. This page does not have any content except for a **keywords** metadata property with a value of **forms**. Boosting this property has made it more important than other properties in our search results.

4. Notice that the other search results on the first page all have the term **forms** in the title. This is because we are now analyzing this property for full-text searches.

Inspect the Query Plan using AEM Chrome Plug-in

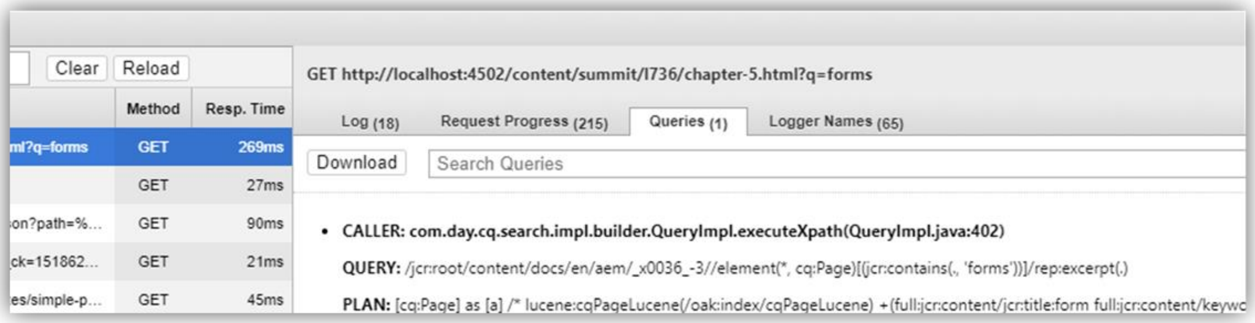
1. Open **AEM Chrome Plug-in.**
   a. While in Chrome, open the View menu and then select **Developer > Developer Tools.**
   b. Or, on macOS press Option (⌥)– Command (⌘)– I
   c. When the developer tools open, select the AEM tab

2. Navigate to the AEM tab.

3. Perform a full-text search of the term **forms.**

4. Select the **http://localhost:4502/content/summit/l736...** request in the left panel.

5. In the right panel, select the **Queries (1)** tab.



GET http://localhost:4502/content/summit/l736/chapter-5.html?q=forms

	Method	Resp. Time
ml?q=forms	GET	269ms
	GET	27ms
on?path=%...	GET	90ms
ck=151862...	GET	21ms
es/simple-p...	GET	45ms

- **CALLER:** com.day.cq.search.impl.builder.QueryImpl.executeXpath(QueryImpl.java:402)

  QUERY: /jcr:root/content/docs/en/aem/_x0036_-3//element(*, cq:Page)[(jcr:contains(., 'forms'))]/rep:excerpt(.)

  PLAN: [cq:Page] as [a] /* lucene:cqPageLucene(/oak:index/cqPageLucene) +(full:jcr:content/jcr:title:form full:jcr:content/keywc

6. In the PLAN, you should now see full-text applied to the **jcr:title** property and boosting of **10** applied to **keywords** property.

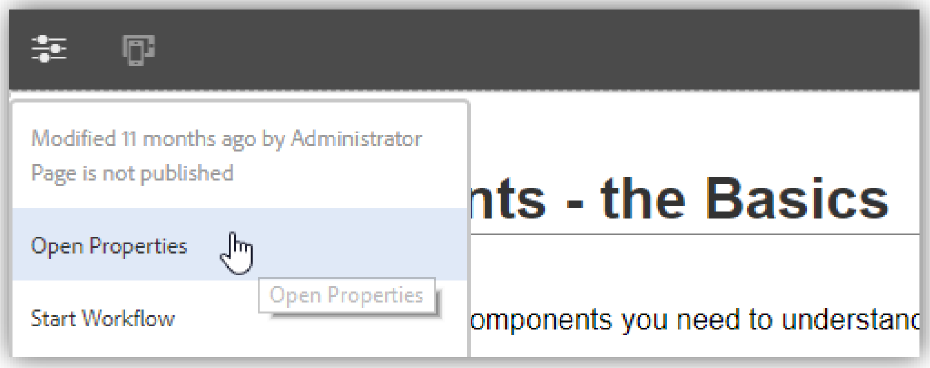
```
PLAN: [cq:Page] as [a] /*
lucene:cqPageLucene(/oak:index/cqPageLucene)
+(full:jcr:content/jcr:title:forms
full:jcr:content/keywords:forms^10.0 :fulltext:forms)
+:ancestors:/content/docs/en/aem/6-3 ft:("forms") where
(contains([a].[*], 'forms')) and (isdescendantnode([a],
[/content/docs/en/aem/6-3])) */
```

Pro Tip: Use explicit boosting sparingly. In most cases, simply adding the property to the full-text index and setting analyzed=true will suffice. The Lucene algorithm already does a good job of evaluating what properties are more important based on text length. For example, a title field is shorter in length than a description field and will thus be ranked higher in search results.

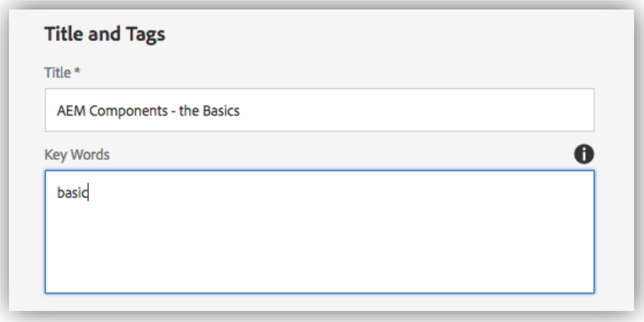
Update the keywords property of a different page

1. Open any page from the search results. Add the /editor.html prefix to the beginning of the URL to allow editing of Page Properties.
   - e.g., http://localhost:4502/editor.html/content/docs/en/aem/6-3/develop/components/components-basics.html

2. In the upper left select the menu and from the drop-down click **Open Properties**.



3. Add a new value to the Keywords field i.e., **basic.**



Title and Tags

Title *

AEM Components - the Basics

Key Words

basic

4. Save and close the dialog.

5. Return to the page used for the previous exercises.
    - e.g., http://localhost:4502/editor.html/content/summit/l736/chapter-1.html

6. Switch the page mode to **Preview.**

7. Perform a search of the term **basic.**

8. The page modified in step 1 should be the first result.

Reference Links

Best Practices for Queries and Indexing - https://adobe.ly/2Ev7Yhy

Oak Boost and Search Relevancy - http://bit.ly/2Hh8dLe

Lucene FAQ – Matching Titles - http://bit.ly/2EJiOQJ

Similarity Search

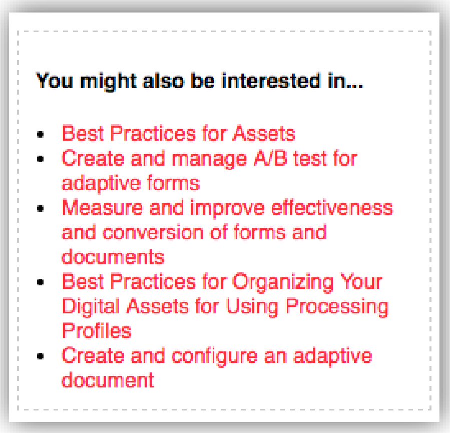
Oak Lucene indexes also support Similarity Queries. The idea behind the similarity query is that it will return nodes that have similar content to the node specified in the query. This can be useful when attempting to implement a "More Like this…" component.

View the Similar Results Component

1. Open any of the exercise chapter pages.
   - e.g., http://localhost:4502/editor.html/content/summit/l736/chapter-1.html

2. Perform a full-text search with term **asset metadata.**
   - Click one of the search results to open the corresponding page
   - e.g.,

http://localhost:4502/content/docs/en/aem/6-3/administer/content/assets/metadata.html



You might also be interested in...

- Best Practices for Assets
- Create and manage A/B test for adaptive forms
- Measure and improve effectiveness and conversion of forms and documents
- Best Practices for Organizing Your Digital Assets for Using Processing Profiles
- Create and configure an adaptive document

3. Notice the box in the side bar with the heading **You might also be interested in…**

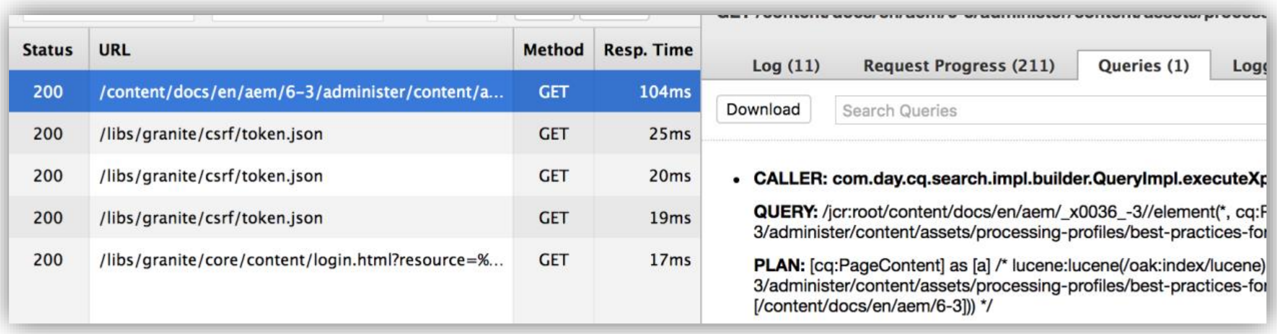
4. There should be five links that have similar content as the current page.

5. Click some of the links in the component and see how the similar results change as you navigate to other pages.

Inspect the Similar Results Query

1. While on one of the results pages open the **Chrome Developer tools > AEM Tab.**

2. Refresh the page and select the page request in the left panel.



Status	URL	Method	Resp. Time
200	/content/docs/en/aem/6-3/administer/content/a...	GET	104ms
200	/libs/granite/csrf/token.json	GET	25ms
200	/libs/granite/csrf/token.json	GET	20ms
200	/libs/granite/csrf/token.json	GET	19ms
200	/libs/granite/core/content/login.html?resource=%...	GET	17ms

- **CALLER: com.day.cq.search.impl.builder.QueryImpl.executeXp**

  QUERY: /jcr:root/content/docs/en/aem/_x0036_-3//element(*, cq:F
  3/administer/content/assets/processing-profiles/best-practices-for

  PLAN: [cq:PageContent] as [a] /* lucene:lucene(/oak:index/lucene)
  3/administer/content/assets/processing-profiles/best-practices-for
  [/content/docs/en/aem/6-3])) */

3. In the main panel select the Queries (1) tab.

4. Notice the **rep:similar** function in the Query. The query searches for other jcr:content nodes that are similar to the one beneath the current page.

Pro-tip: We have found that the easiest way to take advantage of similarity search is to search against the cq:PageContent (jcr:content) node beneath a page. Then, before returning our results to our HTL script, we move the hit result path up one level and return the containing cq:Page.

Reference Links

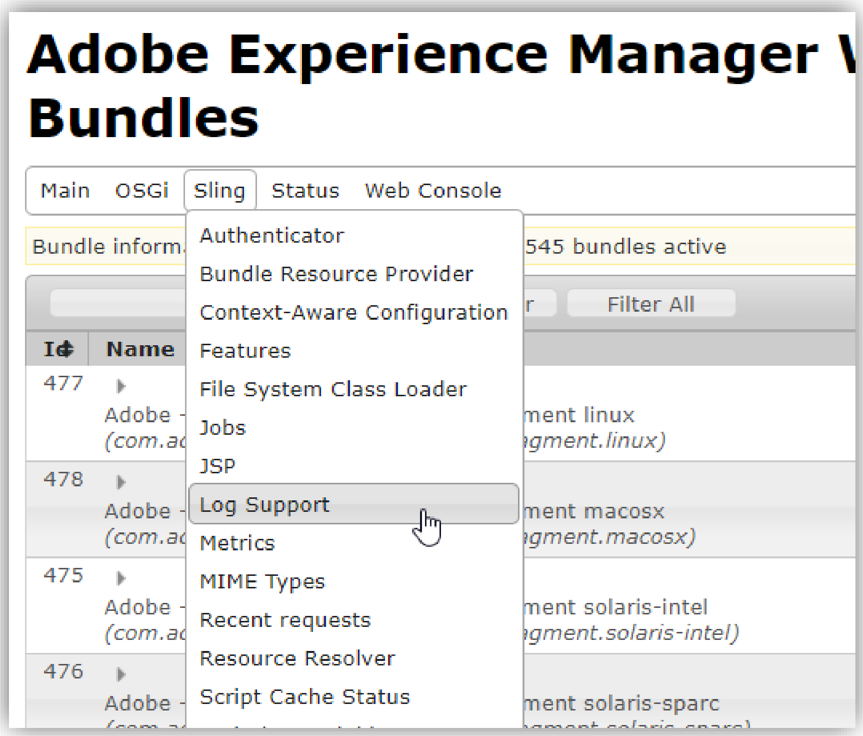
Jackrabbit Similarity Queues - http://bit.ly/2FXnMHu

Logging for Search

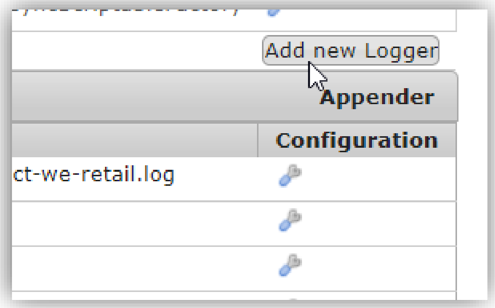
AEM Chrome Plug-in is an efficient view into AEM search logging; however, it is not always available. The same level of information can be obtained via standard AEM logging.

1. As admin, navigate to AEM's OSGi Web console.
   - http://localhost:4502/system/console

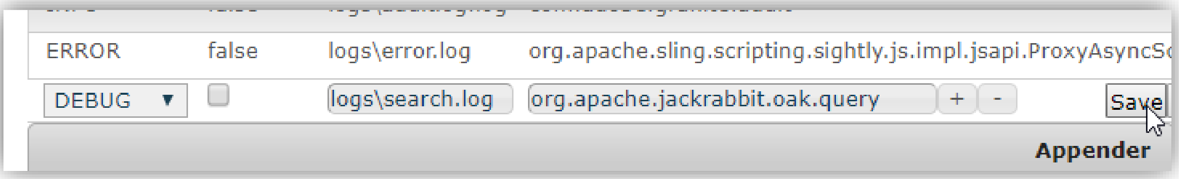
2. In the top menu bar, click **Sling > Log Support.**
   - http://localhost:4502/system/console/slinglog



3. Click **Add new Logger**.



4. Configure the newly appeared logger row, to expose Oak query execution details:
   - Log level: DEBUG
   - Additive: false
   - Log file: logs/search.log
   - Logger: org.apache.jackrabbit.oak.query



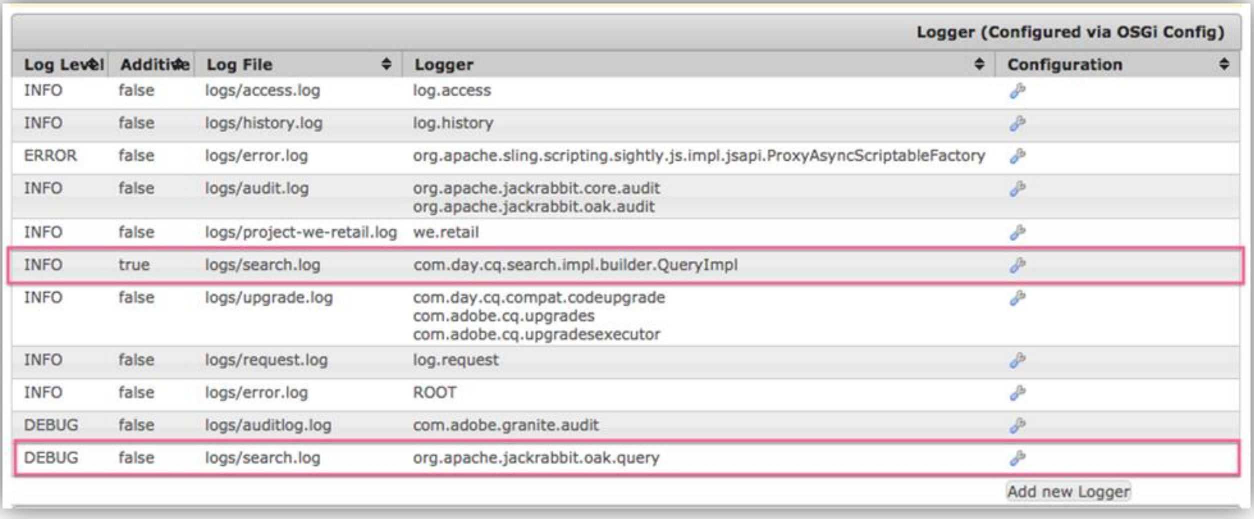
ERROR	false	logs\error.log	org.apache.sling.scripting.sightly.js.impl.jsapi.ProxyAsyncSc	
DEBUG ▼	☐	logs\search.log	org.apache.jackrabbit.oak.query + -	Save

5. Click **Save.**

6. Create a second logger by clicking **Add new logger**.

7. Configure the new logger to expose the Query Builder details:
   - Log level: INFO
   - Additive: true
   - Log file: logs/search.log
   - Logger: com.day.cq.search.impl.builder.QueryImpl

8. Click **Save**.



Logger (Configured via OSGi Config)

Log Level	Additive	Log File	Logger	Configuration
INFO	false	logs/access.log	log.access	
INFO	false	logs/history.log	log.history	
ERROR	false	logs/error.log	org.apache.sling.scripting.sightly.js.impl.jsapi.ProxyAsyncScriptableFactory	
INFO	false	logs/audit.log	org.apache.jackrabbit.core.audit org.apache.jackrabbit.oak.audit	
INFO	false	logs/project-we-retail.log	we.retail	
INFO	true	logs/search.log	com.day.cq.search.impl.builder.QueryImpl	
INFO	false	logs/upgrade.log	com.day.cq.compat.codeupgrade com.adobe.cq.upgrades com.adobe.cq.upgradesexecutor	
INFO	false	logs/request.log	log.request	
INFO	false	logs/error.log	ROOT	
DEBUG	false	logs/auditlog.log	com.adobe.granite.audit	
DEBUG	false	logs/search.log	org.apache.jackrabbit.oak.query	

9. Open the new search.log file in Console (or tail –f from the command line) and perform several searches using Chapter 8.

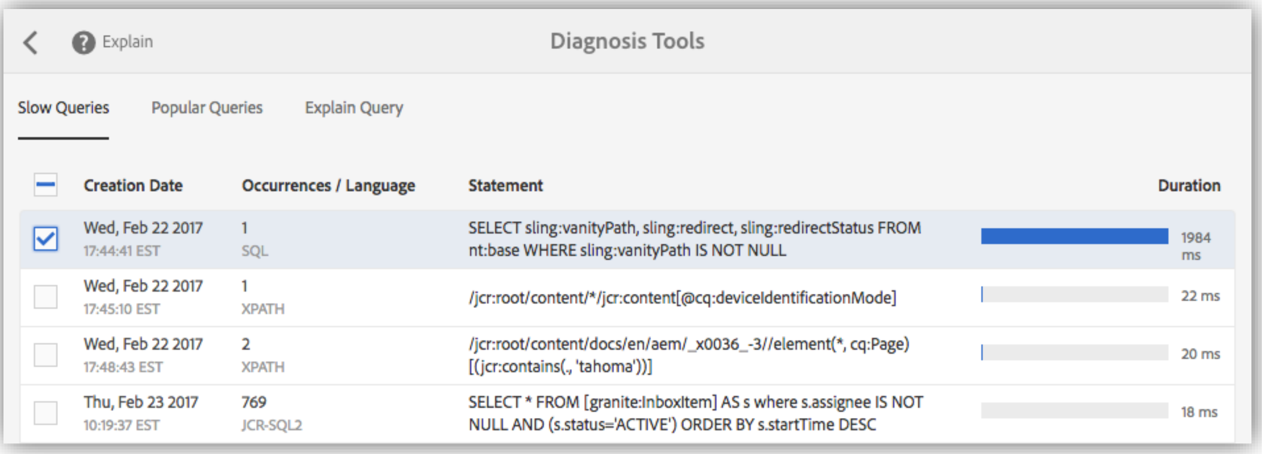
The new logging shows:

- 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 cost evaluation
- The query plan the Oak query engine generates based on the provided query

Pro-tip: Traversal queries will show up prominently in the logs. Any traversal queries MUST be fixed before production deployment to avoid performance issues.

Slow and Popular Queries

AEM maintains a list of slow and popular queries that have been recently executed. These lists can be helpful in locating and identifying problematic queries or simply queries that should be tuned for maximum efficiency (optimize the common case).



	Creation Date	Occurrences / Language	Statement	Duration
✓	Wed, Feb 22 2017 17:44:41 EST	1 SQL	SELECT sling:vanityPath, sling:redirect, sling:redirectStatus FROM nt:base WHERE sling:vanityPath IS NOT NULL	1984 ms
	Wed, Feb 22 2017 17:45:10 EST	1 XPATH	/jcr:root/content/*/jcr:content[@cq:deviceIdentificationMode]	22 ms
	Wed, Feb 22 2017 17:48:43 EST	2 XPATH	/jcr:root/content/docs/en/aem/_x0036_-3//element(*, cq:Page)[(jcr:contains(., 'tahoma'))]	20 ms
	Thu, Feb 23 2017 10:19:37 EST	769 JCR-SQL2	SELECT * FROM [granite:InboxItem] AS s where s.assignee IS NOT NULL AND (s.status='ACTIVE') ORDER BY s.startTime DESC	18 ms

1. As admin, navigate to AEM's Query Performance console
   - **AEM > Tools > Operations > Diagnostics > Query Performance**

2. The **Slow Queries** tab:
   a. Show the recent slowest queries, their query time and execution count
   b. Note that Traversal queries will likely not show on this list as in AEM 6.3 they auto-terminate

3. The **Popular Queries** tab:
   c. Show the recent slowest queries, their execution count and query time

4. Any of the queries can be selected and Explained via the Explain Query button in the top left.
   a. *Explain Query explained in more detail below*

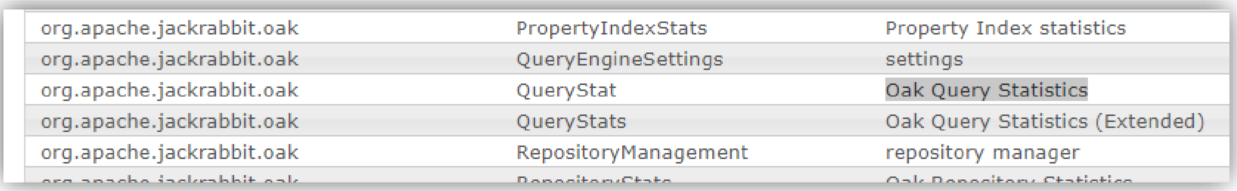
Slow and Popular Queries JMX MBean

Slow and Popular queries can be cleared via the JMX MBean "QueryStat" available in the AEM OSGi Web console.

1. As admin, navigate to AEM's OSGi Web console.
   - http://localhost:4502/system/console

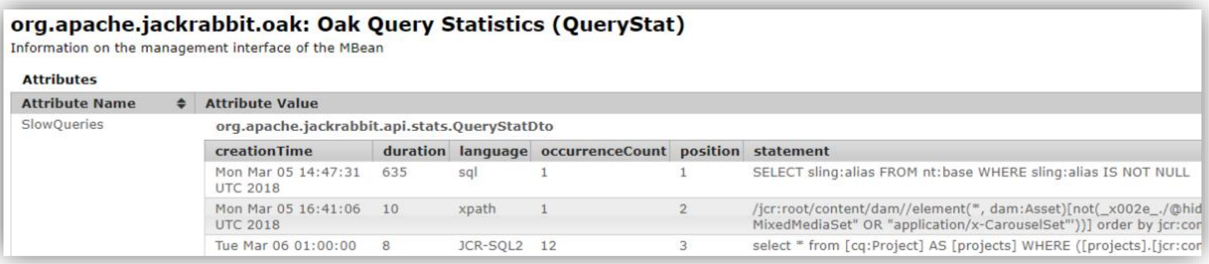
2. In the top menu bar, click **Main > JMX.**
   - http://localhost:4502/system/console/jmx

3. Scroll down and click on the row:
   org.apache.jackrabbit.org QueryStat Oak Query Statistics



org.apache.jackrabbit.oak	PropertyIndexStats	Property Index statistics
org.apache.jackrabbit.oak	QueryEngineSettings	settings
org.apache.jackrabbit.oak	QueryStat	Oak Query Statistics
org.apache.jackrabbit.oak	QueryStats	Oak Query Statistics (Extended)
org.apache.jackrabbit.oak	RepositoryManagement	repository manager

4. Slow and Popular queries are displayed at the top of the page.



org.apache.jackrabbit.oak: Oak Query Statistics (QueryStat)

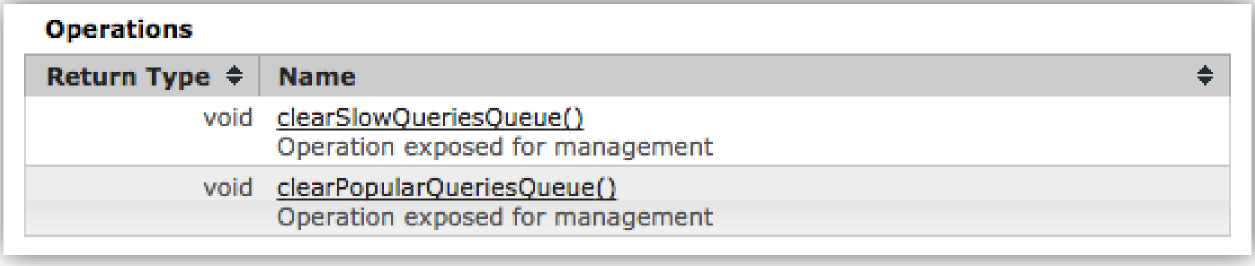
Information on the management interface of the MBean

Attributes

Attribute Name	Attribute Value
SlowQueries	org.apache.jackrabbit.api.stats.QueryStatDto

creationTime	duration	language	occurrenceCount	position	statement
Mon Mar 05 14:47:31 UTC 2018	635	sql	1	1	SELECT sling:alias FROM nt:base WHERE sling:alias IS NOT NULL
Mon Mar 05 16:41:06 UTC 2018	10	xpath	1	2	/jcr:root/content/dam//element(*, dam:Asset)[not(_x002e_./@hid MixedMediaSet" OR "application/x-CarouselSet"))] order by jcr:con
Tue Mar 06 01:00:00	8	JCR-SQL2	12	3	select * from [cq:Project] AS [projects] WHERE ([projects].[jcr:con

5. Scroll to Operations at the bottom of the page and click the appropriate method name to clear the corresponding lists.
   - Clear slow queries lists: **clearSlowQueriesQueue()**
   - Clear popular queries lists: **clearPopularQueriesQueue()**



Operations

Return Type	Name
void	clearSlowQueriesQueue() Operation exposed for management
void	clearPopularQueriesQueue() Operation exposed for management

Explain Query

Explain Query is a powerful tool that provides detailed information on how Oak evaluates and executes search queries. Explain Query shows the:

- Executed query
- Execution cost per index
- Query plan
  - The Oak indices are used
  - What query constraints are evaluated
- Query time
- Number of results

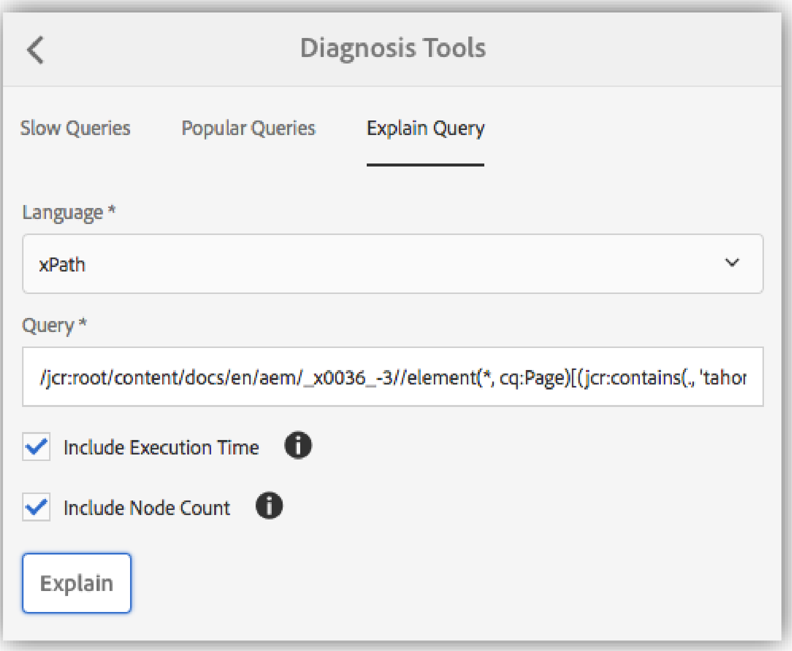
Explain query accepts XPath, JCR-SQL and JCR-SQL2 queries. QueryBuilder queries must be evaluated to their XPath query statement, and said XPath query statement is to be provided to Explain Query.

1. As admin, navigate to AEM's Query Performance console and click on the Explain Query tab.
   - **AEM > Tools > Operations > Diagnostics > Query Performance > Explain Query**

2. Select your query **Language.**
   - XPath, JCR-SQL, JCR-SQL2

3. Provide the **Query** statement**.**

4. Optionally include **Execution Time** and **Node Count.**



Language *
xPath

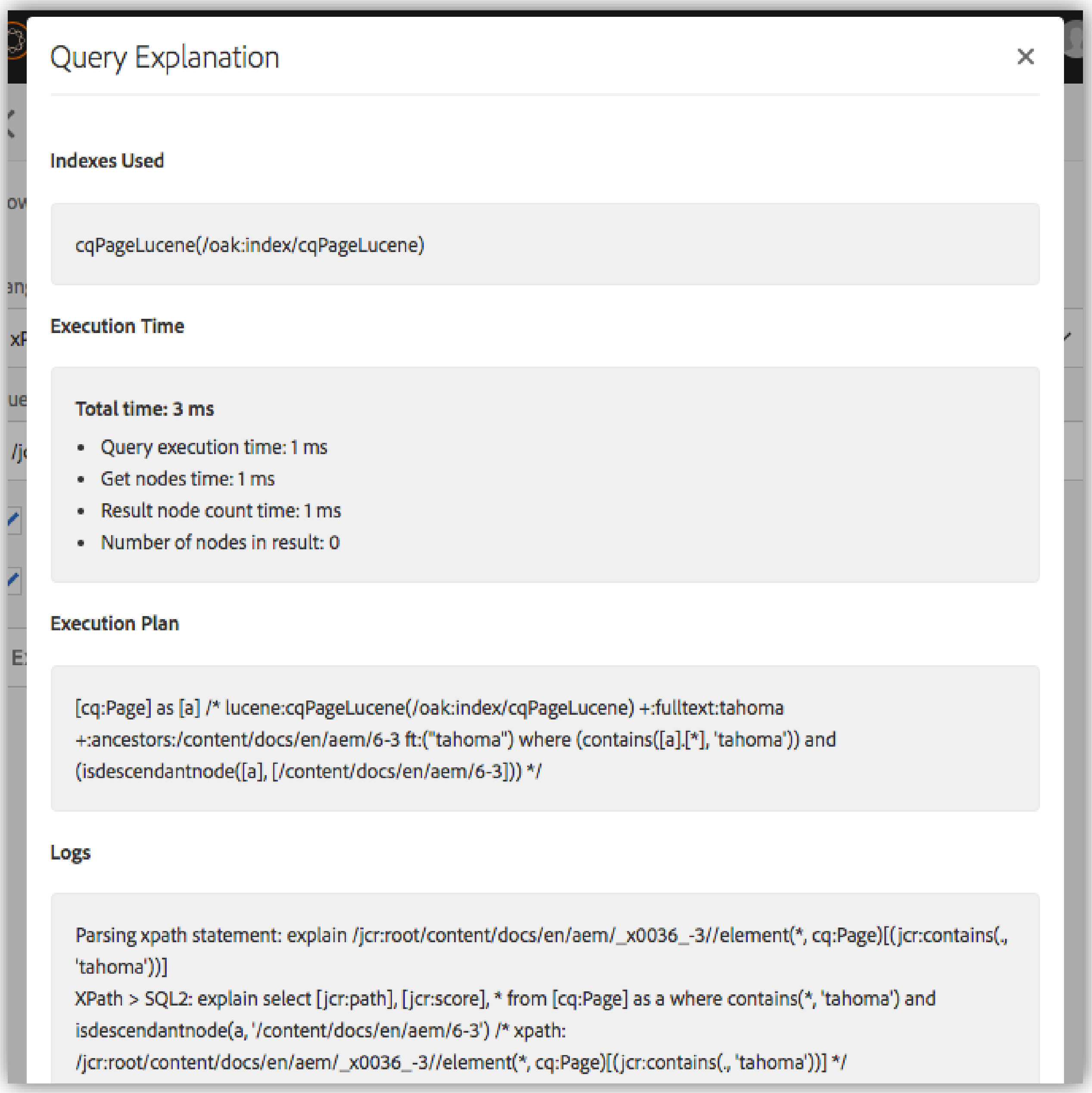
Query *
/jcr:root/content/docs/en/aem/_x0036_-3//element(*, cq:Page)[(jcr:contains(., 'tahoı

✓ Include Execution Time

✓ Include Node Count

Pro-tip: Execution Time and Node Count will execute the provided query, which, depending on the query, could be resource intensive. It is best to only Explain the query first and ensure it is not a Traversal before including Execution Time and Node Count.

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



Indexes Used

```
cqPageLucene(/oak:index/cqPageLucene)
```

Execution Time

Total time: 3 ms
- Query execution time: 1 ms
- Get nodes time: 1 ms
- Result node count time: 1 ms
- Number of nodes in result: 0

Execution Plan

```
[cq:Page] as [a] /* lucene:cqPageLucene(/oak:index/cqPageLucene) +:fulltext:tahoma
+:ancestors:/content/docs/en/aem/6-3 ft:("tahoma") where (contains([a].[*], 'tahoma')) and
(isdescendantnode([a], [/content/docs/en/aem/6-3])) */
```

Logs

```
Parsing xpath statement: explain /jcr:root/content/docs/en/aem/_x0036_-3//element(*, cq:Page)[(jcr:contains(.,
'tahoma'))]
XPath > SQL2: explain select [jcr:path], [jcr:score], * from [cq:Page] as a where contains(*, 'tahoma') and
isdescendantnode(a, '/content/docs/en/aem/6-3') /* xpath:
/jcr:root/content/docs/en/aem/_x0036_-3//element(*, cq:Page)[(jcr:contains(., 'tahoma'))] */
```