



[Gems on AEM] Search forms made easy with the AEM QueryBuilder

Alexander Klimetschek | Senior Developer, Adobe

**MAKE IT AN
EXPERIENCE**

QueryBuilder - What it looks like

- Search for jar files, and order them newest first:

```
type=nt:file  
nodename=*.jar  
orderby=@jcr:content/jcr:lastModified  
orderby.sort=desc
```

- As URL:

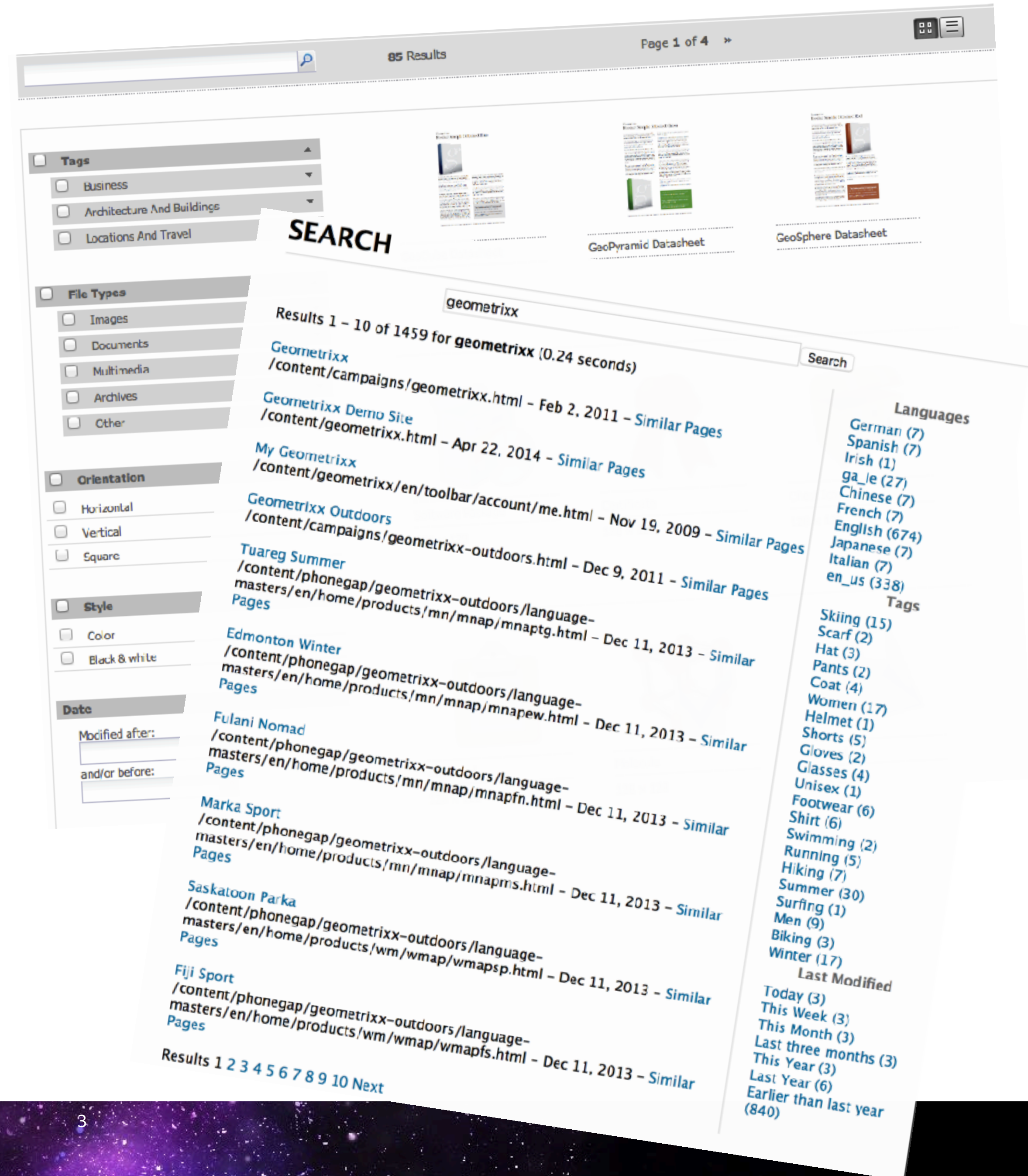
```
http://localhost:4502/bin/querybuilder.json?  
type=nt:file&nodename=*.jar&  
orderby=@jcr:content/jcr:lastModified&orderby.sort=desc
```

- Result as JSON:

```
{  
  success: true,  
  results: 10,  
  total: 155,  
  offset: 0,  
  hits: [{  
    path: "/apps/cloudmgr/install/cq-change-admin-pwd-1.0.1-SNAPSHOT.jar"  
    excerpt: "application/java-archive"  
    name: "cq-change-admin-pwd-1.0.1-SNAPSHOT.jar"  
    ....  
  }, { ... }  
]  
}
```


Use Cases

- End user searches in UI
- Advanced user search forms
 - example: DAM Asset share
 - easy to edit by authors
- Simple, performance-optimized paging
- Endless scrolling
- Site search component
- Simple AJAX JSON query from client side
- Ad hoc developer/admin queries



Do Not Use it For

- Programmatic queries
 - when result is potentially large & needs to be entirely iterated
 - QueryBuilder result API not optimized for that
 - use direct JCR XPath or SQL query instead
 - such a query will be fixed anyway

Philosophy

- QueryBuilder...
 - is a framework to **build queries** for a query engine (JCR XPath underneath)
 - simple to compose query from conditions = **predicates**
 - natively supports URLs & HTML forms
 - complete query can be copy & pasted in text form
 - no worry about escaping
 - post processing: custom ordering, filtering
 - extensible
- QueryBuilder is **not**...
 - a query engine itself (relies on JCR queries)
 - does not have its own search index (relies on Oak indexes)
 - or even cache (except a simple facet cache)

Consequences: URL parameters

- Set of key = value pairs
 - Java: hash maps, property files
- Keep it short for GET requests
 - as fallback, use POST to transport „long“ queries
 - but short queries are more readable
 - => Avoid duplication in parameter names
 - => Allow to write custom „shortcut“ predicates (extensible)
- Order must be encoded in parameter names
 - HTML form GETs/POSTs are required to be in order
 - but the Java servlet spec gives you a hash map....
 - => Conflicts with short names above
 - => Solution is not necessarily intuitive at first, but it gets the job done



`http://localhost:4502/bin/querybuilder.json?
type=nt:file&nodename=*.jar&
orderBy=@jcr:content/jcr:lastModified&orderBy.sort=des`

More Design Consequences

- HTML checkbox behavior
 - checked: `field=on`
 - not checked: *not sent with the request*
- Copy/paste
 - obvious one, but XPath for JCR does not respect it (limit & offset)
 - nothing that can only be done through an API call
 - global settings, given with `p.` prefix, like `p.offset=10`
- Extensible
 - predicates evaluators as OSGi components
 - SCR registration to associate with predicate name


```
fulltext=my search
```

☐ Locations And Travel

File Types

☐ Other

square

Date _____

Page 1 of 4 »

GeoSphere Datasheet



Chest Icon
.....
128 × 128

128 × 128

JCR query underneath

- QueryBuilder will run a JCR XPath query
- Executed normally by Oak
- Same query limits, traversal warnings apply
- Needs indexes to be efficient
 - starting with 6.3 unindexed queries will give a warning/be denied
 - create indexes according to your needs
 - try to cover similar queries by shared index
 - see Gem sessions on Oak queries

QueryBuilder Debugger

Query Builder Debugger

Search >>>

☐ Extract facets
☐ Clear facet cache ([configure](#))
☐ Query is given as URL

type=nt:file
nodename=*.js
path=/libs
orderby=@jcr:content/jcr:lastModified
orderby.sort=asc

[Available predicates](#)

Query tree + URLs

```
ROOT=group: [  
  {nodename=nodename: nodename=*.js}  
  {orderby=orderby: orderby=@jcr:content/jcr:lastModified, sort=asc}  
  {path=path: path=/libs}  
  {type=type: type=nt:file}  
]
```

[JSON QueryBuilder Link](#)
[ATOM Feed QueryBuilder Link](#)

nodename=*.js&orderby=%40jcr%3acontent%2fjcr%3alastModified&orderby..

XPath query

```
/jcr:root/libs//element(*, nt:file)  
{  
  jcr:like(fn:name(), '*.js')  
}  
order by jcr:content/@jcr:lastModified
```

Filtering predicates

Results

Number of hits: 3965

Time: 1.20 seconds

- [/libs/sightly/js/3rd-party/q.js](#) (crxde, [html](#), [json](#))
- [/libs/sightly/js/internal/acm.js](#) (crxde, [html](#), [json](#))
- [/libs/sightly/js/internal/page.js](#) (crxde, [html](#), [json](#))
- [/libs/cq/activitymap/touch-ui/clientlibs/activitymapeditor/js/layers/activitymap.Layer.js](#) (crxde, [html](#), [json](#))
- [/libs/cq/activitymap/touch-ui/clientlibs/activitymapeditor/js/activitymap.js](#) (crxde, [html](#), [json](#))
- [/libs/cq/activitymap/touch-ui/clientlibs/activitymapeditor/js/CQ.js](#) (crxde, [html](#), [json](#))
- [/libs/clientlibs/granite/backbone/source/backbone-1.1.2.js](#) (crxde, [html](#), [json](#))
- [/libs/clientlibs/granite/clientlibrarymanager/ClientLibraryManager.js](#) (crxde, [html](#), [json](#))
- [/libs/clientlibs/granite/clientlibrarymanager/DefaultChannelDetector.js](#) (crxde, [html](#), [json](#))
- [/libs/clientlibs/granite/clientlibrarymanager/Timing.js](#) (crxde, [html](#), [json](#))

For even more detailed info, look at the Sling error.log. [Set the logger](#) for `com.day.cq.search` to levels `DEBUG` (query and timing details) or `TRACE` (plus filtering details).

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

Debug logs

- Set `com.day.cq.search` to DEBUG or TRACE level
- DEBUG level (shortened):

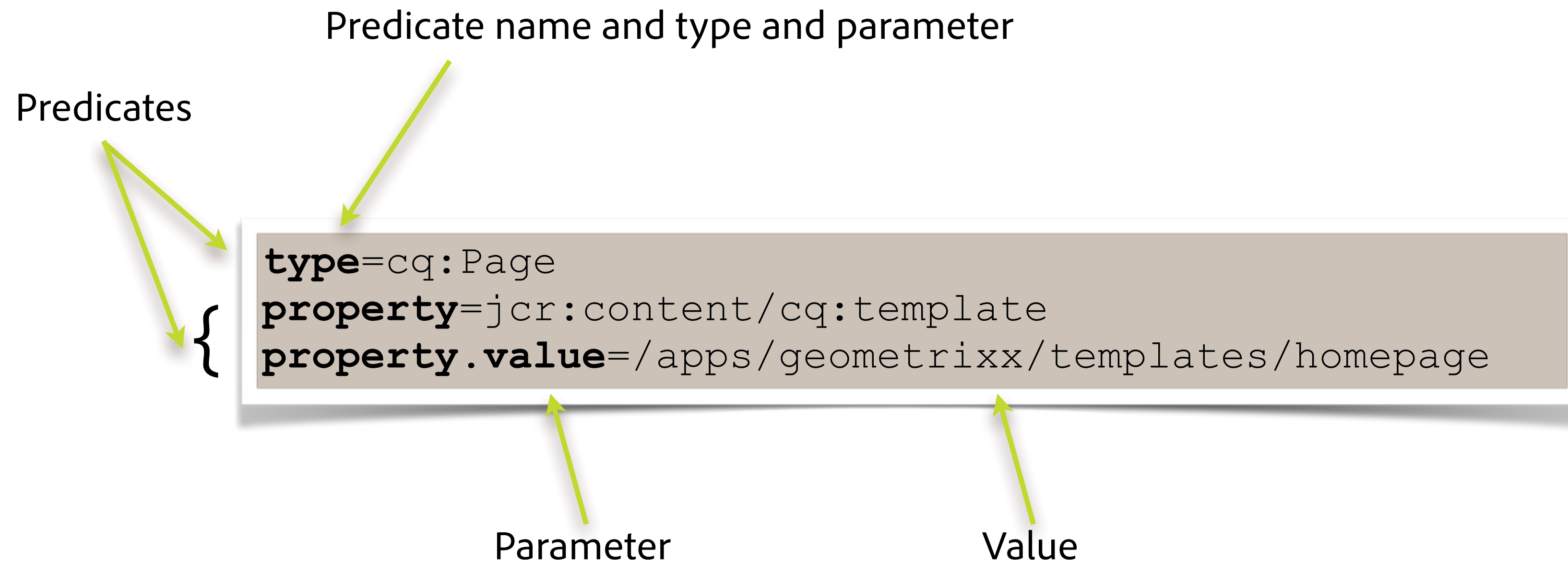
```
21.02.2017 23:26:27.823 *DEBUG* ...QueryImpl executing query (URL):  
nodename=*.%5bp%5dng&type=dam%3aAsset  
21.02.2017 23:26:27.823 *DEBUG* ...QueryImpl executing query (predicate tree):  
ROOT=group: [  
  {nodename=nodename: nodename=*. [p]ng}  
  {type=type: type=dam:Asset}  
]  
21.02.2017 23:26:27.823 *DEBUG* ...QueryImpl xpath query: //element(*, dam:Asset)  
21.02.2017 23:26:27.824 *DEBUG* ...QueryImpl xpath query creation took 1 ms  
21.02.2017 23:26:27.824 *DEBUG* ...QueryImpl filtering predicates: {nodename=nodename: nodename=*. [p]ng}  
21.02.2017 23:26:27.852 *DEBUG* ...QueryImpl >> xpath query returned 1407 results (iterated)  
21.02.2017 23:26:27.852 *DEBUG* ...QueryImpl >> after filtering there are 76 results  
21.02.2017 23:26:27.852 *DEBUG* ...QueryImpl filtering overhead was 6 ms  
21.02.2017 23:26:27.852 *DEBUG* ...QueryImpl consider using 'p.guessTotal' if you don't need the full total with every query  
21.02.2017 23:26:27.857 *DEBUG* ...QueryImpl filtering overhead was 1 ms  
21.02.2017 23:26:27.857 *DEBUG* ...QueryImpl entire query execution took 34 ms
```

- TRACE will also show filtering per node (useful when writing custom filtering)
- Avoid DEBUG or lower in production

Oak Query Debug logs

- To see underlying Oak query details, set DEBUG level
 - `org.apache.jackrabbit.oak.plugins.index`
 - `org.apache.jackrabbit.oak.query`
- might be useful to push into separate `query.log` file
- Will mention index used
- Lists available indexes and costs

Anatomy of a query



Predicate's type is mirrored as parameter internally:

```
type.type=cq:Page  
property.property=jcr:content/cq:template  
property.value=/apps/geometrixx/templates/homepage
```


Predicate resolution & execution

- Internally, a **predicate evaluator** is resolved
- Based on the **type**
- OSGi component (using factories)
- Handles:
 - mapping to xpath (required)
 - filtering of results
 - custom ordering mechanism
 - facet extraction

Multiple predicates of the same type

- Fixed numbering scheme
- Name = **<nr>_<type>**
- Allows to define an order
 - work around hash maps

```
type=cq:Page  
1_property=jcr:content/cq:template  
1_property.value=/apps/geometrix/templates/homepage  
2_property=jcr:content/jcr:title  
2_property.value=English
```

- No custom names possible!

Standard predicates

- path
 - supports multiple paths
- type
 - node type
- property
 - JCR property
 - different operations
 - depth allows to search inside nested structures
- boolproperty
- fulltext
 - full text search
- range
 - supports decimals
- daterange
- relativedaterange
- nodename
- similar
 - rep:similar
- tagid & tag
- tagsearch
 - searches for matching tag first
- language
 - page languages
- mainasset
 - DAM: asset vs. sub assets
- memberOf
 - sling collection membership
- hasPermission
 - jcr privileges, aka „can write“
- savedquery
 - include predicates from saved query

Ordering

- Use **orderby** predicate
 - sort ascending by default, use `orderby.desc=true` for descending
 - support case insensitive `orderby.case=ignore` (since 6.2)
- (1) Order by JCR properties
 - `orderby=@cq:tags`
 - `orderby=@jcr:content/cq:tags`
- (2) Reference predicate by name
 - `orderby=mypredicate`
 - predicate evaluator must provide ordering
 - simply a list of properties (=> used in xpath query)
 - or a custom Comparator (=> run after filtering)
- Multiple orderings
 - `1_orderby=@cq:tags`
 - `2_orderby=@cq:lastModified`
 - `3_orderby=nodename`

Grouping of predicates

- Special **group** predicate

```
fulltext=Management  
group.p.or=true  
group.1_path=/content/geometrixx/en  
group.2_path=/content/dam/geometrixx
```

- Like brackets:

```
(fulltext AND (path=... OR path=...))
```

- Nested:

```
fulltext=Management  
group.p.or=true  
group.1_group.path=/content/geometrixx/en  
group.1_group.type=cq:Page  
group.2_group.path=/content/dam/geometrixx  
group.2_group.type=dam:Asset
```

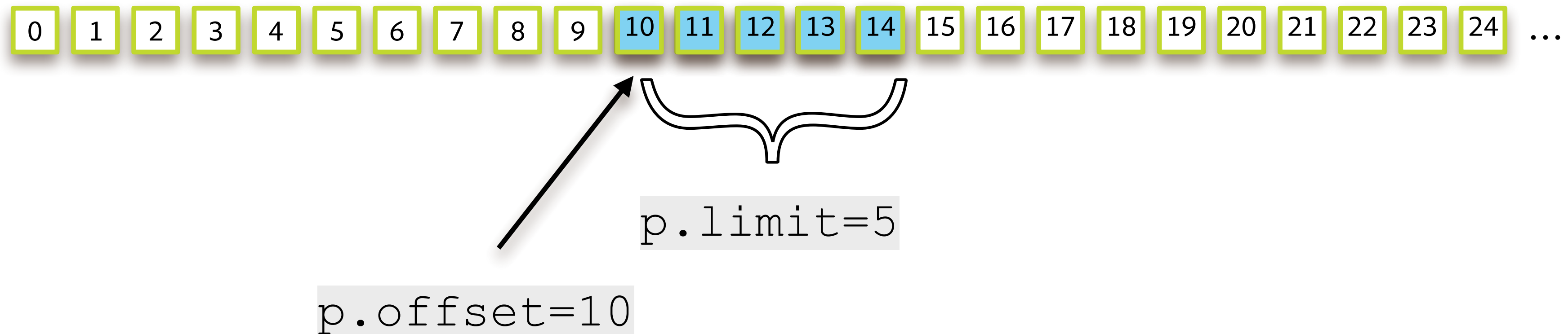
```
(ft AND ( (path= AND type=) OR (path= AND type=) ))
```


Union

- Since 6.3, union queries are now supported by Xpath
 - aka groups with `p.or=true`
 - No longer potentially slow filtering necessary, but passed through
 - Oak can leverage indexes
- Common example: multiple paths
 - `group.p.or=true`
 - `group.1_path=/content/dam/we-retail`
 - `group.2_path=/content/dam/geometrixx`
- Maps to this xpath
 - `(/jcr:root/content/dam/we-retail//* | /jcr:root/content/dam/geometrixx//*[`

Limiting

- `p.offset` defines start
- `p.limit` defines page size
- will display results `p.offset` to `p.offset + p.limit`
- part of predicates, works everywhere, API, json servlet, debugger



Paging and result size

- Control offset and page size
 - `p.offset` = start or how many results to skip
 - `p.limit` = page size or how many results to show
- JSON servlet response
 - `total`: total number of results
 - `offset`: same as `p.offset`
 - `results`: same as `p.limit`

Efficient queries

- Enemy: large results (100k, millions) and iterating/counting them entirely
- Typical solution: display „1000 results and more“
- Possible with `p.guessTotal`
 - `true`: will only go to minimum `p.offset + p.limit`
 - `<number>`: will count total up to that (since 6.0 SP2)
 - use number below which users want exact numbers
 - ...but is small enough to have great performance
 - defaults to false... could change
- Result will tell if there are „more“ results
 - `more: true` in JSON
 - `SearchResult.hasMore()`

Important

Efficient queries: examples

- actual result 2500
 - `p.guessTotal=1000`
 - will return `total: 1000` and `more: true`
- actual result 73
 - will return `total: 73` and `more: false`
- actual result 2500
 - `p.guessTotal=1000`
 - `p.offset=1000 & p.limit=20`
 - will return `total: 1020` and `more: true`

Paging client side

- show total using `r.total` plus „and more“ if `r.more == true`
- show next page button if `r.more || (r.offset + r.results < r.total)`
- show previous page button if `currentPage > 1`
- `currentPage = Math.floor(r.offset / qb.limit) + 1`
- Using QueryBuilder JS components can help
 - `/libs/cq/search/widgets`
 - could need an update though
 - see also `/libs/dam/components/assetshare/querybuilder`

Running queries

- JSON servlet: <http://localhost:4502/bin/querybuilder.json>
- Atom feed: <http://localhost:4502/bin/querybuilder.feed>
- Java API

```
PredicateGroup root = PredicateGroup.create(request.getParameterMap());
Query query = queryBuilder.createQuery(root, session);

SearchResult result = query.getResult();
for (Hit hit : result.getHits()) {
    String path = hit.getPath();
    // ....
}
```

JSON servlet

- JSON: <http://localhost:4502/bin/querybuilder.json>
- `p.hits=` selects how the hits are written
 - **simple**
 - path, lastmodified, etc. only
 - **full**
 - full sling json rendering of node
 - by default entire subtree
 - `p.nodedepth=1` as in sling's json (0 = infinity)
 - **selective**
 - `p.properties` is an array of properties to write
 - just the node itself

Open JSON servlet

- Warning: it is a generic query endpoint, DoS attacks
- Sometimes desire to disable it
- Safe guard with query limits in Oak
 - `-Doak.queryLimitInMemory=500000`
 - `-Doak.queryLimitReads=100000`



Java API

- From HTTP request:

```
Session session = request.getResourceResolver().adaptTo(Session.class);
PredicateGroup root = PredicateGroup.create(request.getParameterMap());
Query query = queryBuilder.createQuery(root, session);
```

- From hash map:

```
Map map = new HashMap();
map.put("path", "/content");
map.put("type", "nt:file");
Query query = builder.createQuery(PredicateGroup.create(map), session);
```

- From predicates:

```
PredicateGroup group = new PredicateGroup();
group.add(new Predicate("mypath", "path").set("path", "/content"));
group.add(new Predicate("mytype", "type").set("type", "nt:file"));
Query query = builder.createQuery(group, session);
```


Persisted Queries

- Store query
 - in Java properties file format
 - in the repository
 - as file node
 - or as string property

```
Query query = .... // create query normally

// store query as file
queryBuilder.storeQuery(query, „/content/myquery“, true, session);

// load it again
Query query = queryBuilder.loadQuery(„/content/myquery“, session);
```

- List component allows this

Facets

- Extract set of possible values found in current result
- Options for a more specific query
- Facet = set of buckets
 - Facet = tag
 - Buckets = product, business, marketing
- Buckets can also be custom ranges
 - Facet = daterange
 - Buckets = yesterday, last week, last year...

```
Map<String, Facet> facets = result.getFacets();
for (String key : facets.keySet()) {
    Facet facet = facets.get(key);
    if (facet.getContainsHit()) {
        writer.key(key).array();
        for (Bucket bucket : facet.getBuckets()) {
        }
    }
}
```


Run new query based on facet/bucket

- Simple as that:

```
String bucketURL =  
query.refine(bucket).getPredicates().toURL();
```

- Facets are extracted for all predicates in the current query
 - keep them „empty“ if they should not search

```
type=cq:Page  
1_property=jcr:content/cq:template  
1_property.value=/apps/geometrixx/templates/homepage  
2_property=jcr:content/jcr:title
```

No value for 2_property

Facet Performance & Cache

- Always iterates entire result set (p.guessTotal benefit lost)
 - Linear performance with size of result set, affected by query limits
- Simple facet cache available since 6.0
- Disabled per default, enable & configure via osgi config
`com.day.cq.search.impl.builder.QueryBuilderImpl`
- Caches facet result by query
 - only reusable for exact same query
- Oak Lucene Index has facet support, would be desirable to tap into it in the future

Extension Points

- Custom predicate: `PredicateEvaluator`
 - generate xpath: `getXPathExpression()`
 - filter via code: `includes()`
- Custom sorting
 - `PredicateEvaluator.getOrderByComparator()`
 - `PredicateEvaluator.getOrderByProperties()`
- Custom facet extraction
 - `PredicateEvaluator.getFacetExtractor()`
 - `FacetExtractor, Facet, Bucket`
- Custom hit format for JSON servlet: `ResultHitWriter`

Extending: Writing custom predicate evaluators

```
/** @scr.component metatype="no"
 *      factory="com.day.cq.search.eval.PredicateEvaluator/event" */
public class EventPredicateEvaluator extends AbstractPredicateEvaluator {
    public String getXPathExpression(Predicate p, EvaluationContext context) {
        final String from = p.get(„from“);
        final String to = p.get(„to“);

        // build xpath snippet
        return „@start = ,...‘ and @end = ,...““;
    }

    public String[] getOrderByProperties(Predicate predicate, EvaluationContext ctx) {
        return new String[] { „start“ };
    }

    public boolean canFilter(Predicate predicate, EvaluationContext context) {
        return false;
    }

    public boolean canXpath(Predicate predicate, EvaluationContext context) {
        return true;
    }
}
```


Extending: Filtering & Facet extraction

- In addition or alternatively to xpath, a predicate can filter
- goes over nodes in result and says include or drop

```
public boolean includes(Predicate p, Row row, EvaluationContext context)
```

- Facet extraction is „lazy“
- Evaluator returns a **FacetExtractor**
- Base implementations available
 - PropertyFacetExtractor
 - DistinctValuesFacetExtractor
 - PredefinedBucketsFacetExtractor

Documentation & Links

- Main documentation plus some examples
 - <https://docs.adobe.com/docs/en/aem/6-2/develop/search/querybuilder-api.html>
- Javadocs
 - <https://docs.adobe.com/docs/en/aem/6-2/develop/ref/javadoc/com/day/cq/search/package-summary.html>
(click next package to see others)
- Predicate evaluators in Javadocs
 - <https://docs.adobe.com/docs/en/aem/6-2/develop/ref/javadoc/com/day/cq/search/eval/package-summary.html>
- QueryBuilder Debugger
 - <http://localhost:4502/libs/cq/search/content/querydebug.html>
- Configuration
 - <http://localhost:4502/system/console/configMgr/com.day.cq.search.impl.builder.QueryBuilderImpl>
- Oak indexes
 - <https://docs.adobe.com/docs/ja/aem/6-2/deploy/platform/queries-and-indexing.html>



Adobe