LIFERAY - ELASTICSEARCH

@dmcisneros

Building Search Queries and Filters

1. Include dependencies in build.gradle:

```
{
compileOnly group: "biz.aQute.bnd", name: "biz.aQute.bndlib"
compileOnly group: "com.liferay.portal", name: "release.portal.api”
compileOnly group: "org.osgi", name: "org.osgi.service.component.annotations"
}
```

2. Include Search Services:

```
@Reference protected Queries queries;
@Reference protected Searcher searcher;
@Reference protected SearchRequestBuilderFactory searchRequestBuilderFactory;
```

3. Build Query:

```
TermsQuery termsQuery = queries.terms("fieldName");
MatchQuery matchQuery = queries.match("fieldName", "value");
BooleanQuery booleanQuery = queries.booleanQuery();
booleanQuery.addMustQueryClauses(termsQuery, matchQuery);
```

4. Build Search Request:

```
SearchRequestBuilder searchRequestBuilder = searchRequestBuilderFactory.builder();
searchRequestBuilder.emptySearchEnabled(true);
searchRequestBuilder.withSearchContext( searchContext -> {
        searchContext.setCompanyId(companyId);
        searchContext.setKeywords(keywords); } → if user send a keyword to search from input
);
SearchRequest searchRequest = searchRequestBuilder.query(booleanQuery).build();
//OR: SearchRequest searchRequest = searchRequestBuilder.postFilterQuery(termsQuery).build();
SearchResponse searchResponse = searcher.search(searchRequest);
SearchHits searchHits = searchResponse.getSearchHits();
```

See:
https://help.liferay.com/hc/es/articles/360029046411-Building-Search-Queries-and-Filters#filters
https://github.com/liferay/liferay-portal/blob/master/modules/apps/portal-search/portal-search-api/src/main/java/com/liferay/portal/search/query/Queries.java

Remember

- **Query**: Calculate scoring for each search, are slower than Filters.

```
SearchRequestBuilder.query(Query);
```

- **Filters**: No Scores are calculated, making them faster and easier to cache.

```
SearchRequestBuilder.postFilterQuery(Query);
```

Relevant Query Types

FullText Queries

Match Query: A full text query, scored by relevance.
Multi Match Query: Execute a MatchQuery over several fields.
String Query: Use Lucene query syntax.

Nested Queries

Nested Query: Query nested objects as if they each had a separate document in the index

(Used in ddmFields asociated to structured content)

Compound Queries

Boolean Query: Allows a combo of several query types. Individual queries are as clauses with SHOULD | MUST | MUST_NOT | FILTER

Term Queries

Wildcard Query: Wildcard (* and ?) matching on keyword fields and indexed terms
Fuzzy Query: Scrambles characters in input before matching

Samples

```
MatchQuery titleQuery = queries.match(Field.getLocalizedName(LocaleUtil.US,
Field.TITLE), fieldNameValueTitle);

WildcardQuery wildcardQuery = queries.wildcard(Field.USER_NAME, "Oth*ser*");

NestedQuery nestedQuery = queries.nested("ddmFieldArray", booleanQueryEmp);

BooleanQuery booleanQueryEmp = queries.booleanQuery();
```

@dmcisneros