How to use Facets to categorize FullTextQuery results

Facetting is another interesting feature provided by Hibernate Search. It allows you to group your *FullTextQuery* results in categories. You often see this in online shops which present the search results in different product categories or on websites which categorize their articles by date.

Prepare your entities for a faceted search

Before you can define a faceted search query, you need to prepare your search index for it. You can do that by annotating the entity attribute you want to use for faceting with a *@Facet* annotation.

```
@Indexed
@Entity
public class Tweet {

	@Column
	@Field(analyze = Analyze.NO)
	@Facet
	private String userName;

	...
}
```

Get faceted results

In the first step, you need to create a *FullTextQuery* for which you to get faceted results. I explained this part in more details in the first post of this series. The *FullTextQuery* in this example selects all *Tweet* entities from the Lucene index.

```
EntityManager em = emf.createEntityManager();

em.getTransaction().begin();


FullTextEntityManager fullTextEm =
Search.getFullTextEntityManager(em);

QueryBuilder tweetQb =
fullTextEm.getSearchFactory().buildQueryBuilder().forEntity(
Tweet.class).get();

Query tweetQuery = tweetQb.all().createQuery();

FullTextQuery fullTextQuery =
fullTextEm.createFullTextQuery(tweetQuery, Tweet.class);
```

How to use Facets to categorize FullTextQuery results

You can then use this query to with a FacetingRequest to get the different facets and their number of elements.

```
FacetingRequest postedAtFR = tweetQb.facet()
    .name("userNameFR")
    .onField(Tweet_.userName.getName())
    .discrete()
    .orderedBy(FacetSortOrder.COUNT_DESC)
    .includeZeroCounts(false)
    .maxFacetCount(3)
    .createFacetingRequest();

FacetManager facetMgr = fullTextQuery.getFacetManager();
facetMgr.enableFaceting(postedAtFR);
List<Facet> facets = facetMgr.getFacets("userNameFR");
```

Use a Facet in your query

Getting the facets of a query result and showing them in the UI is good first step. But what happens if you a user selects one of the facets and wants to see the matching query results?

You obviously need to use the selected facet in your query. You can do that based on the facets you selected in the previous example.

```
// create a FullTextQuery and select Facets
// as shown in previous code snippets

FacetSelection facetSelection = facetMgr.getFacetGroup(
"userNameFR" );
facetSelection.selectFacets( facets.get( 0 ) );

List<Tweet> tweets = fullTextQuery.getResultList();
for (Tweet t : tweets) {
  log.info(t);
}
```