**What is query DSL ?**

It is a framework which enables developer to narrow down the concerns of business logic, where developer can focus on business logic and use this framework to construct type-safe SQL queries.

Internally it uses Fluent API to achieve inline free queries.

Meaning if some want to select all data from table then s/he don’t need to write query

As:

Select \* from Person where firstName = John

Instead it can be achieved by using API method

As:

List data = queryFactory.selectFrom(person)

.where(

person.firstName.eq("John”)

.fetch();

**What is Fluent API ?**

Fluent API is an advanced way of specifying model configuration that covers everything that data annotations can do in addition to some more advanced configuration not possible with data annotations.

Entity Framework Fluent API is used to configure domain classes to override conventions. EF Fluent API is based on a Fluent API design pattern (a.k.a Fluent Interface) where the result is formulated by method chaining.

**How it is used in Nimbus ?**

There lot many crud operations done in Nimbus based on Query DSL, here is example of where data were searched, my comments are in lined with code.

com.antheminc.oss.nimbus.domain.model.state.repo.db.MongoSearchByQuery

private <T> Object searchByQuery(Class<?> referredClass, String alias, SearchCriteria<T> criteria) {

**// outputClass is the object or pojo of desired output**

Class<?> outputClass = findOutputClass(criteria, referredClass);

**// AbstractMongodbQuery is from Query DSL this is where query is constructed and desired output should be of outputClass**

AbstractMongodbQuery query = new QueryBuilder(getMongoOps(), outputClass, alias)

**// Predicate is used to filter data**

buildPredicate((String)criteria.getWhere(), referredClass, alias)

**// buildOrderBy is used to have data in specific order**

.buildOrderBy((String)criteria.getOrderby(), referredClass, alias)

**// At the end get the result from operation**

.get();

**// Projection is used to limit the selected area, to get specific result.**

PathBuilder[] projectionPaths = buildProjectionPathBuilder(referredClass, criteria, query);

if(StringUtils.equalsIgnoreCase(criteria.getAggregateCriteria(), Constants.SEARCH\_REQ\_AGGREGATE\_COUNT.code)) {

**// here DSL query is used to get count of data**

return (Long)query.fetchCount();

}

if(StringUtils.isNotBlank(criteria.getFetch())) {

**// here DSL query is used to get ONE result using projection**

return query.fetchOne(projectionPaths);

}

if(criteria.getPageRequest() != null) {

return findAllPageable(referredClass, alias, criteria.getPageRequest(), query, projectionPaths);

}

**// here DSL query is used to get all projected result**

return query.fetch(projectionPaths);

}