QueryDSL как вариант выбора

JPQL, Criteria API и QUERYDSL

JPQL:

+ Выразительность (SQL like)

+ Хороший выбор для статических запросов

- Плохо подходит для динамических запросов

- Не поддерживает автозаполнение в IDE

- Легко написать синтаксически некорректный код

Criteria API

+ Динамическое формирование запроса

- Очень многословный

Query DSL

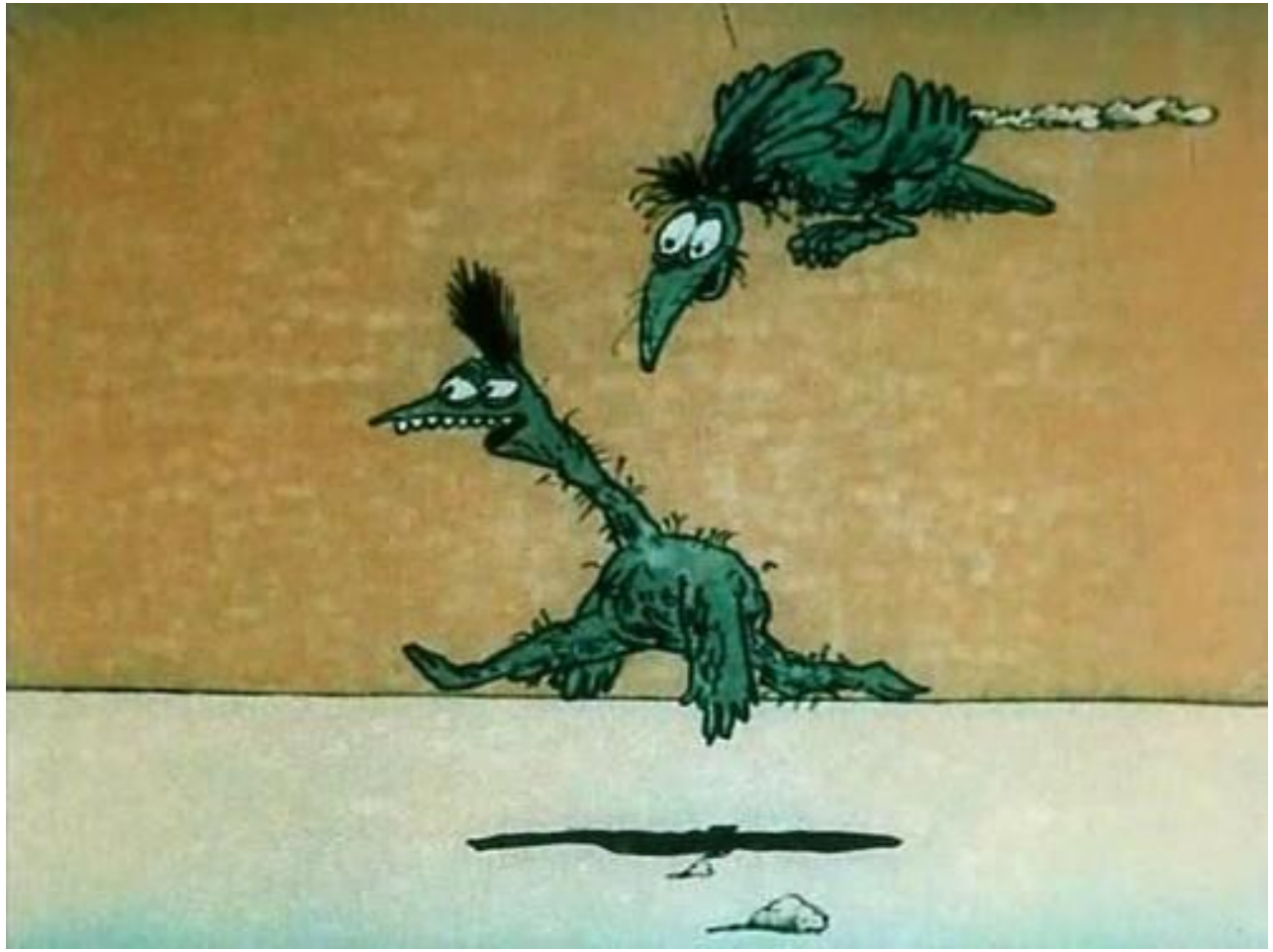
+ Объединяет плюсы

- Не является «отраслевым стандартом»

Аналоги

jOOQ	http://www.jooq.org/
JINQ	http://www.jinq.org/
JaQue	https://github.com/TrigerSoft/jaque
JaQu	http://www.h2database.com/html/jaqu.html
Linq4j	https://github.com/julianhyde/linq4j
Quaere	http://quaere.codehaus.org/
JSR-341	https://jcp.org/en/jsr/detail?id=341

Лучше день потерять, но за 5 минут долететь…



Актуальность

+ Последний релиз 4.1.3 (30 июня 2016)

+ Есть сообщество и поддержка разработчиков

+ 2 383 результата при поиске “querydsl” на StackOverflow (для сравнения “jpql” - 6,774 и “criteria api” 5,731 results)

+ Бесплатна

- Критические изменения, ломающие обратную совместимость, не только в версии 4.x, но и при переходе 3.6 на 3.7

Сравнение запросов

JPQL:

```
TypedQuery queryJpql = em.createQuery(
        "SELECT c FROM Cat c JOIN c.owner o WHERE o.name=:ownerName", Cat.class);
queryJpql.setParameter("ownerName", "Bill");
List catsJpql = queryJpql.getResultList();
```

Querydsl:

```
JPAQuery queryFactory = new JPAQuery(em);
QCat cat = QCat.cat;
QOwner owner = QOwner.owner;
List catsQdsl = queryFactory.from(cat)
        .join(cat.owner, owner)
        .where(owner.name.eq("Bill"))
        .list(cat);
```

Criteria API:

```
CriteriaBuilder cb = em.getCriteriaBuilder();
CriteriaQuery criteriaQuery = cb.createQuery(Cat.class);
Root catQ = criteriaQuery.from(Cat.class);
Join ownerQ = catQ.join(Cat_.owner);
criteriaQuery.where(cb.equal(ownerQ.get(Owner_.name), "Bill"));
TypedQuery query = em.createQuery(criteriaQuery);
List catsCrQ = query.getResultList();
```

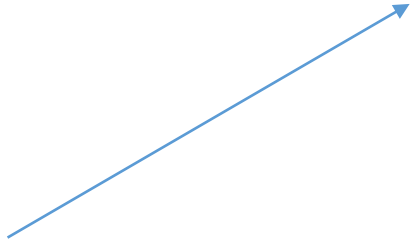
Тестовая БД (H2, in-memory)

Cat

id	integer
name	string
age	integer
owner	integer

Owner

id	integer
name	string



id	name	age	owner
1	Kitty	1	1
2	Tiger	2	2
3	Method	3	3
4	Method	2	3
5	Tiger	4	3
6	Angel	2	4
7	Method	12	4
8	Loki	1	5

id	name
1	John
2	Ron
3	Bill
4	Monica
5	Martin

Метамодель

Стандартная:

```
public abstract class Cat_ {
   public static volatile SingularAttribute owner;
   public static volatile SingularAttribute name;
   public static volatile SingularAttribute id;
   public static volatile SingularAttribute age;
}
```

QueryDSL (включает функционал эквивалентный CriteriaBuilder):

```
public class QCat extends EntityPathBase {
    private static final long serialVersionUID = 1195528346L;
    private static final PathInits INITS = PathInits.DIRECT2;
    public static final QCat cat = new QCat("cat");
    public final NumberPath age = createNumber("age", Integer.class);
    public final NumberPath id = createNumber("id", Integer.class);
    public final StringPath name = createString("name");
    public final QOwner owner;
    public QCat(String variable) {
        this(Cat.class, forVariable(variable), INITS);
    }
    public QCat(Path<? extends Cat> path) {
        this(path.getType(), path.getMetadata(), path.getMetadata().isRoot() ? INITS : PathInit
    }
    public QCat(PathMetadata<?> metadata) {
        this(metadata, metadata.isRoot() ? INITS : PathInits.DEFAULT);
    }
    public QCat(PathMetadata<?> metadata, PathInits inits) {
        this(Cat.class, metadata, inits);
    }
    public QCat(Class<? extends Cat> type, PathMetadata<?> metadata, PathInits inits) {
        super(type, metadata, inits);
        this.owner = inits.isInitialized("owner") ? new QOwner(forProperty("owner")) : null;
    }
}
```

Общие принципы использования

В целом используется цепочка методов:

from

innerJoin, join, leftJoin, fullJoin, on

where

groupBy

having

orderBy

limit, offset, restrict

Примеры

1) Фильтр по полю связанной сущности:

- до 4.x

```
List<String> catNames = queryFactory.from(cat).join(cat.owner, owner)
        .where(owner.name.eq("Bill")).list(cat.name);
```

- 4.x

```
List<String> catNames = queryFactory.select(cat.name).from(cat).join(cat.owner, owner)
        .where(owner.name.eq("Bill")).fetch();
```

2) Агрегация

- Максимальный возраст кота

```
Integer maxAge = queryFactory.from(cat)
        .singleResult(cat.age.max());
```

- Максимальный возраст котов, сгруппированных по имени

```
Map results = queryFactory.from(cat)
        .transform(GroupBy.groupBy(cat.name).as(GroupBy.max(cat.age)));
```

- Количество котов по хозяевам

```
List results = queryFactory.from(cat)
        .join(cat.owner, owner)
        .groupBy(owner)
        .orderBy(owner.name.asc())
        .list(owner.name, cat.count());
```

3) Сортировка

- По имени кота (собственное поле)

```
List cats = queryFactory.from(cat)
        .orderBy(cat.name.asc())
        .list(cat);
```

- Котов по имени хозяина (по полю связанной сущности)

```
List cats = queryFactory.from(cat)
        .orderBy(cat.owner.name.asc())
        .list(cat);
```

4) Динамическое формирование критерий и пагинация

```
List predicates = new ArrayList<>();
if (true) {
    predicates.add(cat.name.like("%ge%"));
}
if (true) {
    predicates.add(cat.age.eq(2));
}
com.mysema.query.types.Predicate where = ExpressionUtils.allOf(predicates);
JPAQuery queryFactory = new JPAQuery(em);
List catNames = queryFactory.from(cat)
        .where(where)
        .restrict(new QueryModifiers(10L, 1L))
        .orderBy(cat.name.asc(), cat.id.asc())
        .list(cat.name);
```

4) Фильтрация по полю связанной коллекции (one-to-many relationship)

- С использованием distinct

```
List ownersDistinct = queryFactory
        .distinct()
        .from(owner)
        .innerJoin(owner.cats, cat)
        .where(cat.name.eq("Method"))
        .list(owner);
```

- Более эффективный подход

```
com.mysema.query.types.Predicate catNamePredicate = new JPASubQuery()
        .from(cat).where(cat.owner.eq(owner).and(cat.name.eq("Method"))).exists();
List owners = queryFactory
        .from(owner)
        .where(catNamePredicate)
        .list(owner);
```

Список литературы

- http://www.querydsl.com
- https://gist.github.com/EdwardBeckett/5377401 - конфигурация с gradle
- https://github.com/akorob/queryDsl - исходный код, использованный в презентации
- Проблема с источниками. Подавляющая часть приведена на официальном сайте.