CONNECT
SYSTEM LIB

CCBUS

Connect System Lib

Table of Contents

1. Introduction

Presents library attending ccbus type application.

2. Requirements

- Java 8
- Maven tool

3. Installation

- Checkout Java Api in you java project folder. Source is shared between all projects

```
terminal$ svn checkout svn://192.168.0.2/projects/ccbus/ccbusSystemLib/trunk ccbusSystemLib
```

```
terminal$ cd ccbusSystemLib
```

```
terminal$ mvn install
```

- *Add module as dependency in you Maven project*

```
<dependency>
    <groupId>ccbus.system</groupId>
    <artifactId>ccbus-system</artifactId>
    <version>1.0.0</version>
</dependency>
```

It could be imported as module in your maven project, but still it have to be installed in mvn repository with "mvn install"

4. Packages

1.1 Query package

1.1.1 Introduction

API that is on top of JPA criteria builder. It allows safe and typed interfaces for building predicates and dynamic queries. It is not tighten to some specific storage engine like hibernate or relevant but level down JPA API.

1.1.2 Requirement

Requires auto generated Meta resources from **Connect DB Tool** .

Meta root class is tree from entity nodes. It provides information as field names, types and relations between entities.

For more information regarding meta generator see **readme-db.pdf**

1.1.3 API Structure

1.1.3.1 Stack Based API

There is one main class that provides facade interface to all inner classes. It functions as query builder on top of a stack. Sequential calls change state of expression on the top. Creating new expression will be added to last one or will pop last one. Calling state methods will affect expression on top of the stack.

Public methods returns reference to the main front class, which allows chain calls.

Point of the top level query build is fetch request. Thus expression build is not ordered (column selection is not before having expression or where expression).

Join expression shouldn't be generated explicitly, because selected field path will generate it implicitly.

- Structure scheme:
    - Build query expression. Meta code , bold text means prefix:

Scheme to follow:

SELECT ... FROM ...

[JOIN]

[WHERE ...]

[GROUP BY ... [HAVING ...]]

[ORDER BY ...]

Meta code representation:

```
selectStackRoot                                  // root query object
.joinLeft(MetaEntity Entity)                     // join left - use to overwrite inner join,  prior select
.joinRight(MetaEntity Entity)
.select(Meta Field,[Meta Field][...])           // select field
            .fFunctionA(Meta Field,[Value]) // select function
            .fFunctionB(Meta Field,[Meta Field][Value])
.whereAnd                                        // push and epxression on top of stack
        .cmpA(Meta Field,Value)                  // add compare function to top - field+value
        .cmpB(Value).par(Meta Field)            // field only, value provided with par method
    .whereOr                                     // add or epxression on top expression :
                                                 //  sql:   where cmp and cmp (cmp or cmp)
          .cmpA(Meta Field,Value)               // add compare function to top - field+value
          .cmpB(Value).par(Meta Field)          // field only, value provided with par method
    .whereClose                                  // pop last expression - whereOr
          .cmpA(Meta Field,Value)               // add expression to top whereAnd
                                                  //  sql:  where cmp and cmp (cmp or cmp) and cmp
  .group(Meta Field,[Meta Field][...])
  .havingOr                                                  //  sql: same as whereOr and whereAnd
             .cmpA(Value).fFunctionA(Meta Field,[Value]) // field is presented by function
  .[...]
   .orderDesc(Meta Field)
   .orderAsc(Meta Field)
```

- Add expression to existing one:

```
selectStackRoot
.addWhere()   // add where expression to first whereOr/whereAnd
      .whereAnd()
              .cmpA(Meta Field,Value) //  sql: where cmp and cmp (cmp or cmp) and cmp and  cmpAdded
.addHaving()                          // same as addWhere
.[...]
```

- Fetch result:

  Result depends on the select stackType, overall input interface is the same:

  - Single result

    RESULT some=selectStackRoot.resultSingle()

    RESULT could be: Entity, Object[], UserDefinedObject, Tuple

    Selecting single column leads to

  - List result

    RESULT some=selectStackRoot.resultList()

    RESULT could be: List<Entity>, List<Object[]>, List<UserDefinedObject>, List<Tuple>

- Create query builder:

Select stack type affects result of the query. Following supported:

- Entity/List<Entity> as result

SelectStackEntity<Shape> selectShape=new SelectStackEntity(Shape.class);

- Object[]/List<Object[]> result

SelectStackArray<Shape> selectShapeA=new SelectStackArray(Shape.class);

- UserDefined/List<UserDefined> result

SelectStackCtor<Shape,DateIdC> selectShapeCtor=new SelectStackCtor(Shape.class,DateIdC.class);

- Tuple/List<Tuple> result

SelectStackTuple<Shape> selectShapeT=new SelectStackTuple(Shape.class);

1.1.3.2 Nested Based API - N/A

1.1.4 Examples

- Entity

```
// Entity
SelectStackEntity<Shape> selectShape=new SelectStackEntity(Shape.class);
selectShape.whereAnd().cmpEqual(Meta.Shape.ShapeType.id,1L);
Shape s=selectShape.resultSingle();
System.out.println(" date:"+s.getDate());

List<Shape> list=selectShape.resultList();
System.out.println(" list:"+list);

// GENERATED QUERY
 select
        shape0_.id as id1_1_,
        shape0_.date as date2_1_,
        shape0_.shapeType_id as shapeTyp3_1_
    from
        Shape shape0_
    inner join
        ShapeType shapetype1_
            on shape0_.shapeType_id=shapetype1_.id
    where
        shapetype1_.id=?
```

- Array

```
// Array
SelectStackArray<Shape> selectShapeA=new SelectStackArray(Shape.class);

selectShapeA.select(
    Meta.Shape.date,Meta.Shape.id)
    .fCount(Meta.Shape.id)
    .fSum(Meta.Shape.id)
    .fCurrentDate()
    .whereAnd().cmpIn(Meta.Shape.id, Arrays.asList(12L,14L,15L))
        .cmpLessThan(21L).par(Meta.Shape.id)
        .cmpLessThan(22L).par(Meta.Shape.id)
    .whereOr()
        .cmpLessThan(20L).par(Meta.Shape.id)
        .cmpLessThan(21L).par(Meta.Shape.id)
    .whereClose()
        .cmpLessThan(22L).par(Meta.Shape.id)
        .par(Meta.Shape.id).cmpLessThan(Meta.Shape.id,20L)
    .group(Meta.Shape.id)
    .havingOr()
        .cmpLessThan(20L).fCount(Meta.Shape.id)
        .cmpLessThan(Meta.Shape.id,20L);

selectShapeA.addWhere()
    .whereAnd()
        .cmpLessThan(21L).par(Meta.Shape.id)
        .cmpLessThan(22L).par(Meta.Shape.id)
    .whereOr()
        .par(Meta.Shape.id)  // will be ignored no compare expression on top
        .cmpLessThan(20L).par(Meta.Shape.id)
        .cmpLessThan(Meta.Shape.id,10L)
    .whereClose()
        .cmpLessThan(100L).par(Meta.Shape.id);



Object[] obj=selectShapeA.resultSingle();

System.out.println(" id:"+obj[0]);
System.out.println(" date:"+obj[1]);
System.out.println(" cnt:"+obj[2]);
System.out.println(" sum:"+obj[3]);
System.out.println(" cur date:"+obj[4]);

// GENERATED QUERY
 select
        shape0_.date as col_0_0_,
        shape0_.id as col_1_0_,
        count(shape0_.id) as col_2_0_,
        sum(shape0_.id) as col_3_0_,
        current_date as col_4_0_
    from
        Shape shape0_
    where
        (
            shape0_.id in (
                ? , ? , ?
            )
        )
        and shape0_.id<?
        and shape0_.id<?
        and (
            shape0_.id<?
            or shape0_.id<?
        )
        and shape0_.id<?
        and shape0_.id<?
        and shape0_.id<?
        and shape0_.id<?
        and (
            shape0_.id<?
            or shape0_.id<?
        )
        and shape0_.id<?
    group by
        shape0_.id
    having
        count(shape0_.id)<?
        or shape0_.id<?
```

- Array#2

```
// Array#2
SelectStackArray<Point> selectPointA=new SelectStackArray(Point.class);
selectPointA.joinLeft(Meta.Shape);  // overwrite inner join to left

selectPointA.select(Meta.Point.id,Meta.Point.Shape.date,Meta.Point.Shape.ShapeType.name)
        .whereAnd().cmpGreaterThan(Meta.Shape.id,0L);



Object[] objp=selectPointA.resultSingle();
System.out.println("id point: "+objp[0]);
System.out.println("date shape: "+objp[1]);
System.out.println("shape type name: "+objp[2]);

// GENERATED QUERY
select
        point0_.id as col_0_0_,
        shape2_.date as col_1_0_,
        shapetype4_.name as col_2_0_
    from
        Point point0_
    left outer join
        Shape shape1_
            on point0_.shape_id=shape1_.id
    inner join
        Shape shape2_
            on point0_.shape_id=shape2_.id
    inner join
        Shape shape3_
            on point0_.shape_id=shape3_.id
    inner join
        ShapeType shapetype4_
            on shape3_.shapeType_id=shapetype4_.id
    where
        point0_.id>?
```

- Constructor

```
// Define class
class DateIdC
{
    public java.util.Date date;
    public Long id;
    public Integer cnt;

    public DateIdC(java.util.Date date, Long id)
    {
        this.date = date;
        this.id = id;
    }

    public DateIdC(java.util.Date date, Long id,Integer cnt)
    {
        this.date = date;
        this.id = id;
        this.cnt=cnt;
    }
}

// Ctor
SelectStackCtor<Shape,DateIdC> selectShapeCtor=new
SelectStackCtor(Shape.class,DateIdC.class);
selectShapeCtor.select(Meta.Shape.date,Meta.Shape.id).whereAnd().cmpGreaterThan(Meta.Sha
pe.id,12L);
DateIdC dic=selectShapeCtor.resultSingle();

System.out.println(" id:"+dic.id);
System.out.println(" date:"+dic.date);

List<DateIdC> listA=selectShapeCtor.resultList();
System.out.println(" date:"+listA.get(1).date);

// GENERATED QUERY
    select
        shape0_.date as col_0_0_,
        shape0_.id as col_1_0_
    from
        Shape shape0_
    where
        shape0_.id>?
and shape0_.id>?
```

- Tuple

```
// Tuple
SelectStackTuple<Shape> selectShapeT=new SelectStackTuple(Shape.class);
selectShapeT.select(Meta.Shape.date,Meta.Shape.id).whereAnd().cmpEqual(Meta.Shape.id,12L
);

Tuple tuple=selectShapeT.resultSingle();

System.out.println(" id:"+tuple.get(0));
System.out.println(" date:"+tuple.get(1));

// GENERATED QUERY
    select
        shape0_.date as col_0_0_,
        shape0_.id as col_1_0_
    from
        Shape shape0_
    where
        shape0_.id=?
```

1.2 ...

Happy coding :)