Hibernate pending topics
---
1. Locking
2. Pagination
3. StoredProcedure
4. Mapping(tommo i will discuss)
5. NamedQuery,NativeSQL Query and Criterion Api
6. Project

NamedHQLQuery

  => So far our HQL query is specific to one session object becoz query object is created having hard coded HQl query on session object.
  => To make our HQL query accessible and executable through multiple session objects of Mulitple DAO classes or client a[pps we need to
        go for "NamedHQL" query.
  => We defined NamedHQLQuery in mapping file using <query> tag or in Entity class using "@NamedQuery" havinig logical name and we
       access and execute that HQL query in DAO class.

Code using Annotation
==
```
@Entity
@NamedQuery(name = "HQL_INSERT_TRANSFER_POLICIES",
         query = "INSERT INTO
in.ineuron.model.PremiumInsurancePolicy(policyId,policyName,policyType,company,tenu
re)
      SELECT i.policyId,i.policyName,i.policyType,i.company,i.tenure FROM
in.ineuron.model.InsurancePolicy as i WHERE i.tenure>=:tenure")
```

refer:: HibernateNamedHQLInsertOperation

NativeSQL Query

 => It is given to execute plain SQL queries that are supported by underlying DB S/w.
 => We need to use these operations only when it is not possible through HQL
                 eg: inserting a single record.
 => It supports both select and non select operation
 => These queries performance is bit good compared to HQL because they go to sql directly without any conversion.
 => We write a query using table name and column names.

working with select operation
---
1. working with single record.
2. working with all record.

Working with Nonselect operation
1. using NamedNativeQuery
2. Directly writing the query inside DAOImpl class.

refer: HBNativeSQLQuery

Criterion Api

   SRO ====> hibernate persistence methods
   bulkoperations => we use HQL(query written using classname and properties) to write queries.

=> In case of Criterion api, we can perform both singlerow and bulkoperations without using any queries just like java statements.
=> Criterion api will generate SQL queries based on the given entity classnames and properties name.
=> It doesnot support non-select operation,it supports only select operation.
=> Using Criteria object we can add 3 object
    a. Criterion objects(for where clause condition)
    b. Project objects(for scalar select operation)
    c. Order object(for orderBy operations)

There are 2 modes of writing Criterion api
a. HB QBC(Query by Criteria)====> specific to hibernate only
b. JPA QBC ===> common to all ORM framework
refer: HBQBCApi

Pagination
========
Displaying large volume of records into muliptle pages is called as "pagination".
Hibernate supports pagination through QBC
1. setFirstResult(int pageNo)
2. setMaxResult(int maxNo)

StoredProcedure calls in hibernate
==========================
1. we use ProcedureCall(I) to make a call to storedprocedure
2. we use ParameterMode(Enum) to specify the IN,OUT,INOUT Param

To get all the columns value data in the form of ResultSet

```
ProcedureCall procedureCall =
session.createStoredProcedureCall("GET_POLICIES_BY_TENURE",InsurancePolicy.class);
procedureCall.registerParameter(1, Integer.class,
ParameterMode.IN).bindValue(start);
procedureCall.registerParameter(2, Integer.class, ParameterMode.IN).bindValue(end);
List<InsurancePolicy> list = procedureCall.getResultList();
```

To get specific columns based on the input type

```
ProcedureCall procedureCall =
session.createStoredProcedureCall("GET_POLICY_BY_ID");
procedureCall.registerParameter(1, Integer.class, ParameterMode.IN).bindValue(id);
procedureCall.registerParameter(2, String.class, ParameterMode.OUT);
procedureCall.registerParameter(3, String.class, ParameterMode.OUT);
procedureCall.registerParameter(4, String.class, ParameterMode.OUT);

String policyName = (String) procedureCall.getOutputParameterValue(2);
String companyName = (String) procedureCall.getOutputParameterValue(3);
String policyType = (String) procedureCall.getOutputParameterValue(4);
```

refer: HBStoredProcedure