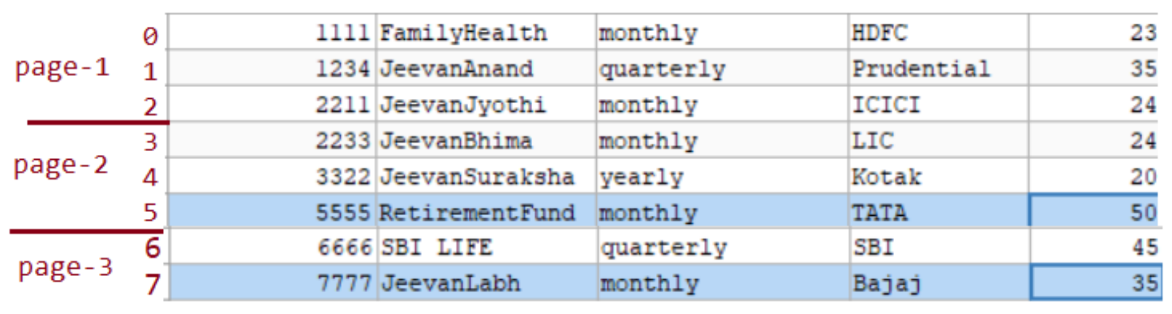
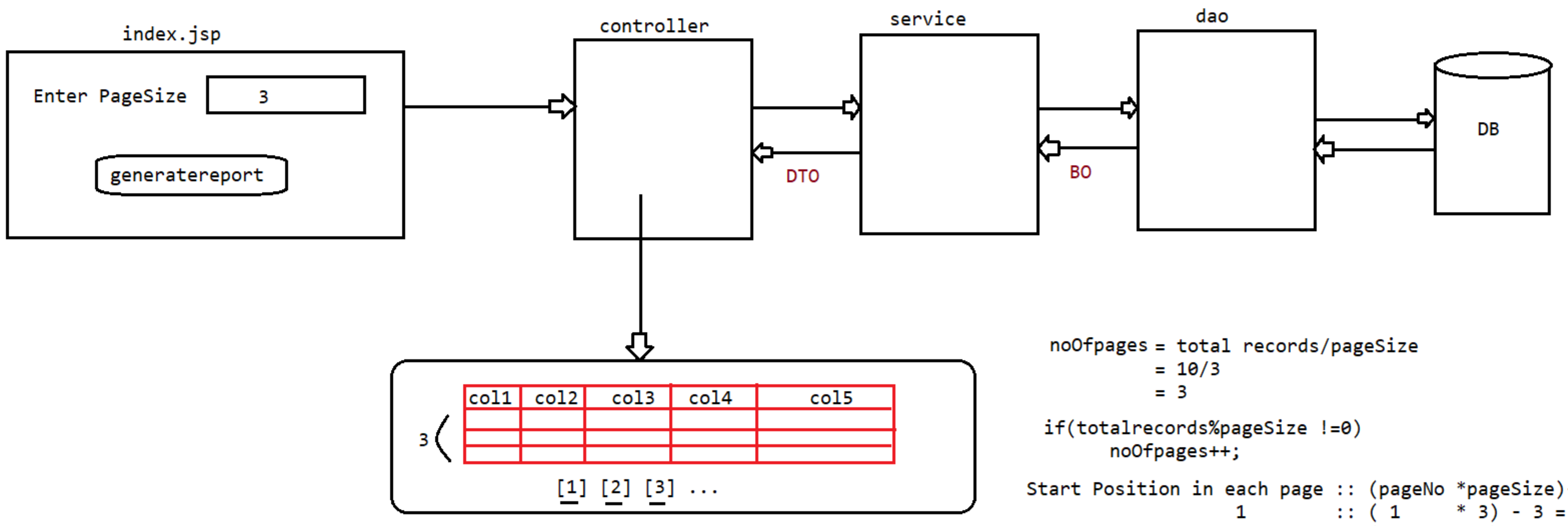
Pagination



	0	1111	FamilyHealth	monthly	HDFC	23
page-1	1	1234	JeevanAnand	quarterly	Prudential	35
	2	2211	JeevanJyothi	monthly	ICICI	24
	3	2233	JeevanBhima	monthly	LIC	24
page-2	4	3322	JeevanSuraksha	yearly	Kotak	20
	5	5555	RetirementFund	monthly	TATA	50
	6	6666	SBI LIFE	quarterly	SBI	45
page-3	7	7777	JeevanLabh	monthly	Bajaj	35



```
noOfpages = total records/pageSize
          = 10/3
          = 3

if(totalrecords%pageSize !=0)
      noOfpages++;

Start Position in each page :: (pageNo *pageSize)-3
                   1        :: ( 1     * 3) - 3 = 0
                   2        :: ( 2     * 3) - 3 = 3
                   3        :: ( 3     * 3) - 3 = 6
                   4        :: ( 4     * 3) - 3 = 9
```

HBFilters

It is a named,parameterized global condition that can be enabled or disabled on session object.

This concept is very useful to execute SQLQuery without implicit condition.

use cases :The blocked,closed account should not participate in any Query execution.

note: In real time application, blocked/closed account would not be deleted from the table rather status would be marked as closed/blocked.

This is called as "SoftDeletion".

BankAccount.java

```
@Entity
@Table(name = "bankaccount")
@FilterDef(name = "FILTER_BANKACCOUNT_STATUS", parameters = {
                        @ParamDef(type = "string", name = "accType1"),
                        @ParamDef(type = "string", name = "accType2")
                  }
            )
@Filter(name = "FILTER_BANKACCOUNT_STATUS", condition = "STATUS NOT
IN(:accType1,:accType2)")
public class BankAccount implements Serializable {

}
```

SelectApp.java

```
Filter filter = session.enableFilter("FILTER_BANKACCOUNT_STATUS");
      filter.setParameter("accType1", "blocked");
      filter.setParameter("accType2", "closed");

Query<BankAccount> query = session.createQuery("FROM in.ineuron.entity.BankAccount
WHERE balance>=:amt");
      query.setParameter("amt", 2000.0f);

List<BankAccount> resultList = query.getResultList();
      resultList.forEach(System.out::println);

      System.out.println();

session.disableFilter("FILTER_BANKACCOUNT_STATUS");
Query<BankAccount> query1 = session.createQuery("FROM in.ineuron.entity.BankAccount
WHERE balance>=:amt");
            query1.setParameter("amt", 2000.0f);

List<BankAccount> result = query1.getResultList();
      result.forEach(System.out::println);
```

Soft Deletion in hibernate

It is not about deleting the record from the table, it is all about marking the record from DB table as deleted.

When we close the account in the bank, they would not actually delete the record, rather the record would be marked as "blocked".

When we perform soft deletion, for a call of session.delete() 'delete query' should not be executed rather 'update query' should be executed

To do so we need to configure as shown below

a.

```
@Entity
@Table(name = "bankaccount")
@SQLDelete(sql = "UPDATE bankaccount SET status='closed' WHERE accno=?")
@Where(clause = "STATUS NOT IN ('blocked','closed')")
public class BankAccount implements Serializable {


}
```

b.

```
session = HibernateUtil.getSession();
transaction = session.beginTransaction();

BankAccount account = new BankAccount();
account.setAccno(6);
session.delete(account);====> update query will be generated..
```

c.

```
Query<BankAccount> query = session.createQuery("From
in.ineuron.entity.BankAccount");
List<BankAccount> accounts = query.getResultList();
accounts.forEach(System.out::println);
```

Output
======

```
select
        bankaccoun0_.accno as accno1_0_,
        bankaccoun0_.balance as balance2_0_,
        bankaccoun0_.holderName as holderna3_0_,
        bankaccoun0_.status as status4_0_
    from
        bankaccount bankaccoun0_
    where
        (
            bankaccoun0_.STATUS NOT IN (
                'blocked','closed'
            )
        )
BankAccount [accno=7, holderName=dhoni, balance=44000.0, status=active]
```

d.

```
Query query = session.createQuery("UPDATE in.ineuron.model.BankAccount set
status='closed' where accno=:no");
query.setParameter("no", 1234);
rowCount = query.executeUpdate();
```

In the above case, if we keep delete query,then by referring to entity update query won't be executed.
so if we use HQL,NativeSQL,QBC logics then we need to explicity write "update sql query" for soft deletion.

Note: while working with @SQLXXXX annotations/custom queries execution takes place only for single row operation,

Pagination

=> The process of displaying huge no of records page by page is calle pagination.

=> It would avoid loading of all records,load only those records/objects that are required to display in the current page to improve the performance.

eg: Gmail inbox,report generation, google search results ......

Using HQL/NativeSQL/QBC we can achieve pagination as shown below
==

```
      Query<InsurancePolicy> query = session.createQuery("from 
in.ineuron.model.InsurancePolicy");

                  //pagination settings
                  query.setFirstResult(0);
                  query.setMaxResults(3);

      List<InsurancePolicy> insurance = query.list();
      insurance.forEach(System.out::println);
```