Kamus Perintah Hibernate Criteria Query – MySQL

Untuk rekan – rekan yang sudah terbiasa dengan Java Platform mungkin sudah tidak aneh dengan Hibernate, begitu pula untuk rekan – rekan yang sudah terbiasa dengan Hibernate kemungkinan juga sudah tidak aneh dengan istilah Hibernate Criteria Query. Pada kesempatan ini kita akan coba bahas mengenai persamaan fungsi antara Criteria Query dengan MySQL, dengan tujuan membantu rekan – rekan yang baru mempelajari tentang Hibernate dan bagaimana kesamaan fungsi nya dengan perintah – perintah standar yang sudah ada dalam MySQL.

Berikut akan kita mulai pembahasan mengenai hal tersebut diatas, pertama kita akan ambil contoh sebuah table dalam MySQL yang kemudian akan di konversikan kedalam sebuah model java class mengikuti kaidah yang sudah ditetapkan Hibernate. Dalam hal ini kita akan menggunakan Hibernate Annotation sehingga tidak menggunakan file *.xml untuk mapping field table terhadap field class. Untuk contoh sederhana kita akan coba sebuah table dengan nama "karyawan" begitu pula kita akan menggunakan model class dengan nama yang sama.

Nama Field	Type Field	Primary Key
id	Integer / Numeric	Yes
nama	Varchar(30)	
tgl_masuk	Date	
upah	Decimal(20,10)	

Dengan perintah membuat table di MySQL sebagai berikut:

```
CREATE TABLE `karyawan` (
  `id` int(11) NOT NULL auto_increment,
  `nama` varchar(30) default NULL,
  `tgl_masuk` date NOT NULL default '1970-01-01',
  `upah` decimal(20,10) default NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=DYNAMIC
```

Kemudian class model Karyawan dibuat dengan syntax seperti berikut:

```
@Entity
@Table(name="karyawan")
public class Karyawan implements Serializable {
    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;

    @Column(name="nama")
    private String nama;

    @Temporal(TemporalType.DATE)
    @Column(name="tgl_masuk",nullable=false,columnDefinition="date")
    private Date tglMasuk;

    @Column(name="upah")
```

```
    @Type(type="big_decimal")
    private BigDecimal upah;

    public Karyawan() {}

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public Date getTglMasuk() {
        return tglMasuk;
    }

    public void setTglMasuk(Date tglMasuk) {
        this.tglMasuk = tglMasuk;
    }

    public BigDecimal getUpah() {
        return upah;
    }

    public void setUpah(BigDecimal upah) {
        this.upah = upah;
    }
}
```

Dengan mengasumsikan Hibernate sudah dikuasai sesuai dengan referensi artikel "Proses CRUD Dengan Hibernate Annotations Menggunakan Netbean 6.0", maka kita akan langsung membahas mengenai penggunaan Criteria Query, adapun class yang bisa digunakan adalah "Criteria" dari "org.hibernate.Criteria", "DetachedCriteria" dari "org.hibernate.criterion.DetachedCriteria" juga harus diikuti dengan class "Expression" dari "org.hibernate.criterion.Expression", "Restriction" dari "org.hibernate.criterion.Restriction", kemudian dengan asumsi jika kita akan mengambil seluruh data dari table "Karyawan" maka kita akan menggunakan perintah SQL sebagai berikut:

```
select * from Karyawan;
```

sedangkan jika menggunakan Criteria Query baik itu "Criteria" ataupun "DetachedCriteria" akan mengembalikan nilai dari baris perintah seperti berikut:

```
getSession().createCriteria(Karyawan.class).list();
```

Berikut kita akan menampilkan beberapa contoh perintah Criteria Query dan persamaan perintah dengan Query MySQL:

MySQL	Hibernate Criteria Query
select * from Karyawan where id = 1;	getSession().createCriteria(Karyawan.class) .add(Expression.eq("id", Long.valueOf(1))) .list();
Select * from Karyawan where id != 1;	getSession().createCriteria(Karyawan.class) .add(Expression.ne("id", Long.valueOf(1))) .list();
Select * from Karyawan where id <> 1;	getSession().createCriteria(Karyawan.class) .add(Expression.ne("id", Long.valueOf(1))) .list();
select * from Karyawan where nama = "sesuatu";	getSession().createCriteria(Karyawan.class) .add(Expression.eq("nama", "sesuatu")) .list();
select * from Karyawan where tgl_masuk = "1900-01-01";	getSession().createCriteria(Karyawan.class) .add(Expression.eq("tglMasuk", new java .util.Date(java.sql.Date .valueOf("1900-01-01")))) .list();
Select * from Karyawan where upah = 10000;	getSession().createCriteria(Karyawan.class) .add(Expression.eq("upah", BigDecimal .valueOf(10000))) .list();
Select * from Karyawan where id > 10;	getSession().createCriteria(Karyawan.class) .add(Expression.gt("id", Long.valueOf(10))) .list();
Select * from Karyawan where id < 10 and id > 30;	getSession().createCriteria(Karyawan.class) .add(Expression.lt("id", Long.valueOf(10))) .add(Expression.gt("id", Long.valueOf(30))) .list();
Select * from Karyawan where id < 10 or id > 30;	getSession().createCriteria(Karyawan.class) .add(Expression.or(Expression.lt("id", Long.valueOf(10)), Expression.gt("id", Long.valueOf(30)))) .list();
Select * from Karyawan where id <= 10;	getSession().createCriteria(Karyawan.class) .add(Expression.le("id", Long.valueOf(10))) .list();
Select * from Karyawan where id >= 10;	getSession().createCriteria(Karyawan.class) .add(Expression.ge("id", Long.valueOf(10))) .list();
Select * from Karyawan where nama like 'sesu%';	getSession().createCriteria(Karyawan.class) .add(Expression.like("nama", "sesu", MatchMode.END)) .list();
Select * from Karyawan where nama like '%sua%';	getSession().createCriteria(Karyawan.class) .add(Expression.like("nama", "sua", MatchMode.ANYWHERE)) .list();
Select * from Karyawan where nama like '%atu';	getSession().createCriteria(Karyawan.class) .add(Expression.like("nama", "atu", MatchMode.START)) .list();
Select * from Karyawan where nama like 'sesuatu';	getSession().createCriteria(Karyawan.class) .add(Expression.like("nama", "sesuatu", MatchMode.EXACT)) .list();
Select * from Karyawan where id between 0 and 100;	getSession().createCriteria(Karyawan.class) .add(Expression.between("id", Long.valueOf(0), Long.valueOf(100))) .list();
Select * from Karyawan where nama = '';	getSession().createCriteria(Karyawan.class) .add(Expression.isEmpty("nama")) .list();
Select * from Karyawan where nama <> '';	getSession().createCriteria(Karyawan.class) .add(Expression.isNotEmpty("nama")) .list();

Select * from Karyawan where nama is null;	getSession().createCriteria(Karyawan.class) .add(Expression.isNull("nama")) .list();
Select * from Karyawan where nama is not null;	getSession().createCriteria(Karyawan.class) .add(Expression.isNotNull("nama")) .list();

Select * from Karyawan where id in (1,3,5,7);	List<Long> idlist = new ArrayList<Long>();
	idlist.add(1);
	idlist.add(3);
	idlist.add(5);
	idlist.add(7);
	getSession().createCriteria(Karyawan.class)
	.add(Expression.in("id", idlist))
	.list();

Select * from Karyawan where id not in (1,3,5,7)	List<Long> idlist = new ArrayList<Long>();
	idlist.add(1);
	idlist.add(3);
	idlist.add(5);
	idlist.add(7);
	getSession().createCriteria(Karyawan.class)
	.add(Expression.not(Expression.in("id", idlist)))
	.list();

Select * from Karyawan where id > 1 order by id ASC;	getSession().createCriteria(Karyawan.class) .addOrder(Order.asc("id")) .add(Expression.ge("id", Long.valueOf(1))) .list();
Select * from Karyawan where id > 1 order by id DESC;	getSession().createCriteria(Karyawan.class) .addOrder(Order.desc("id")) .add(Expression.ge("id", Long.valueOf(1))) .list();

Sebagai catatan perintah "Expression.not" atau "Restriction.not" tidak dapat dipakai untuk perintah Query MySQL seperti berikut "`Select * from Karyawan where id <> 1`" atau "`Select * from Karyawan where id != 1`" meskipun dalam perintah tersebut memperlihatkan where kondisi "`tidak sama dengan`" atau "`not equal`" dikarenakan perintah ini sudah terwakili dengan perintah "Expression.ne" atau "Restriction.ne". Sebagai contoh penulisan yang benar, jika Hibernate yang hendak kita tulis merupakan bagian dari web-app menggunakan Spring, maka contoh – contoh diatas harus ditulis didalam Class DAO Implementation, tetapi jika tidak terintegrasi dalam Spring maka ditulis dalam Class object biasa. Dengan contoh sebagai berikut:

```
@SuppressWarnings("unchecked")
public List<Karyawan> loadContohSatu() {
    List<Long> idlist = new ArrayList<Long>();
    idlist.add(1);
    idlist.add(3);
    idlist.add(5);
    idlist.add(7);

    return getSession().createCriteria(Karyawan.class)
      .add(Expression.not(Expression.in("id", idlist))).list();
}
```

```
@SuppressWarnings("unchecked")
public List<Karyawan> loadContohDua() {
    return getSession getSession().createCriteria(Karyawan.class)
      .addOrder(Order.asc("id"))
```

```
        .add(Expression.ge("id", Long.valueOf(1))).list();
}
```

Untuk kesempatan ini baru beberapa contoh syntax yang simple seperti yang sudah tertulis diatas yang dapat ditampilkan, dan akan dilanjutkan pada artikel selanjut untuk pembahasan yang lebih kompleks dan Criteria Query yang melibatkan beberapa table sekaligus beserta relasi antar table dengan menggunakan Hibernate Criteria Query.

Mungkin saat ini hanya sekian ilmu yang bisa dibagi dengan pembaca semoga bermanfaat, jika ada kekurangan atau masukan yang dapat meningkatkan kemampuan jangan sungkan untuk memberikan komentar.