NHibernate. Part 2

Natalie Vegerina
Software engineer
Infostroy Ltd,
Kharkov, Ukraine

Методы загрузки данных

- SQL query
- HQL query
- Criteria query
- QueryOver
- LINQ to NHibernate

SQL

Scalar queries

```
ISession session = DbSessionFactory.Instance.OpenSession();
ISQLQuery query = session.CreateSQLQuery("SELECT * FROM CATS")
    .AddScalar("ID", NHibernateUtil.Int64)
    .AddScalar("NAME", NHibernateUtil.String)
    .AddScalar("BIRTHDATE", NHibernateUtil.Date);
IList results = query.List();
```

Output:

```
object[] result = new object[3];
result = (object[])results[0];
long id = (long)result[0];
string name = (string)result[1];
DateTime birthdate = (DateTime)result[2];
```

Entity queries

```
ISQLQuery query = session.CreateSQLQuery("SELECT * FROM CATS")
        .AddEntity(typeof(Cat));

ISQLQuery query = session.CreateSQLQuery(@"SELECT ID, NAME,
BIRTHDATE FROM CATS")
        .AddEntity(typeof(Cat));

IList results = query.List();
```

Output:

```
Cat cat = results[0] as Cat;
```

Returning non-managed entities

```
session.CreateSQLQuery("SELECT NAME, BIRTHDATE FROM CATS")
    .SetResultTransformer(Transformers.AliasToBean(typeof (CatDTO)));
```

Named SQL queries

```
<sql-query name="persons">
   <return alias="person" class="eg.Person"/>
   SELECT person.NAME AS {person.Name},
   person.AGE AS {person.Age},
   person.SEX AS {person.Sex}
   FROM PERSON person
   WHERE person.NAME LIKE :namePattern
</sql-query>
```

```
IList people = session.GetNamedQuery("persons")
   .SetString("namePattern", "smith")
   .SetMaxResults(50)
   .List();
```

Named SQL queries

```
<sql-query name="mySqlQuery">
   <return-scalar column="name" type="String"/>
   <return-scalar column="age" type="Int64"/>
   SELECT p.NAME AS name,
   p.AGE AS age,
   FROM PERSON p WHERE p.NAME LIKE 'Hiber%'
</sql-query>
```

Named SQL queries

```
<sql-query name="mySqlQuery">
   <return alias="person" class="eg.Person">
    <return-property name="Name" column="myName"/>
    <return-property name="Age" column="myAge"/>
    <return-property name="Sex" column="mySex"/>
   </return>
   SELECT person.NAME AS myName,
   person.AGE AS myAge,
   person.SEX AS mySex,
   FROM PERSON person WHERE person.NAME LIKE :name
</sql-query>
```

Using stored procedures for querying

```
CREATE PROCEDURE selectAllEmployments AS
   SELECT EMPLOYEE, EMPLOYER, STARTDATE, ENDDATE,
   REGIONCODE, EMPID, VALUE
   FROM EMPLOYMENT
```

Using stored procedures for querying

```
<sql-query name="selectAllEmployments_SP">
   <return alias="emp" class="Employment">
     <return-property name="employee" column="EMPLOYEE"/>
     <return-property name="employer" column="EMPLOYER"/>
     <return-property name="startDate" column="STARTDATE"/>
     <return-property name="endDate" column="ENDDATE"/>
     <return-property name="regionCode" column="REGIONCODE"/>
     <return-property name="id" column="EID"/>
</return>
   exec selectAllEmployments
</sql-query>
```

Custom SQL for create/update/delete

```
<class name="Person">
   <id name="id">
    <generator class="increment"/>
   </id>
   <property name="name" not-null="true"/>
   <sql-insert>INSERT INTO PERSON (NAME, ID) VALUES ( UPPER(?), ? )</sql-insert>
   <sql-update>UPDATE PERSON SET NAME=UPPER(?) WHERE ID=?</sql-update>
   <sql-delete>DELETE FROM PERSON WHERE ID=?</sql-delete>
</class>
```

Stored procedures for create/update/delete

```
<class name="Person">
   <id name="id">
    <generator class="increment"/>
   </id>
   <property name="name" not-null="true"/>
   <sql-insert>exec createPerson ?, ?</sql-insert>
   <sql-delete>exec deletePerson ?</sql-delete>
   <sql-update>exec updatePerson ?, ?</sql-update>
</class>
```

Custom SQL for loading

```
<class name="Person">
   <id name="Id">
    <generator class="increment"/>
   </id>
   <property name="Name" not-null="true"/>
   <loader query-ref="person"/>
</class>

<sql-query name="person">
   <return alias="pers" class="Person" lock-mode="upgrade"/>
   SELECT NAME AS {pers.Name}, ID AS {pers.Id}
   FROM PERSON
   WHERE ID=?
   FOR UPDATE
</sql-query>
```

HQL

Scalar and entity queries

```
string hql = "from Product p";

var products = session.CreateQuery(hql).List();

var products = session.CreateQuery(hql).List<Product>();
```

Filtering, sorting and paging

```
string hql = @"from Product p
where p.Discontinued
and p.Category = :category
and p.UnitPrice <= :unitPrice
order by p.Name";

var cheapFruits = session
    .CreateQuery(hql)
    .SetString("category", "Fruits")
    .SetDecimal("unitPrice", 1.0m)
    .SetFirstResult(10)
    .SetMaxResults(10)
    .List<Product>();
```

Unique result

```
IQuery query = session.CreateQuery("select count(*) from Product");
int count = Convert.ToInt32(query.UniqueResult());

int count = query.UniqueResult<int>();
```

Result transformers

```
var productsLookup = session
    .CreateQuery("select Id as Id, Name as Name from Product")
    .SetResultTransformer(Transformers.AliasToBean<NameID>())
    .List<NameID>();
```

Grouping

```
var productsGrouped = session
    .CreateQuery(@"select p.Category as Category,
count(*) as Count,
avg(p.UnitPrice) as AveragePrice
from  Product p
group by p.Category")
    .List();
```

Grouping with transformers

```
var productsGrouped = session
    .CreateQuery(@"select p.Category as Category,
count(*) as Count,
avg(p.UnitPrice) as AveragePrice
from  Product p
group by p.Category")
    .SetResultTransformer(Transformers.AliasToEntityMap)
    .List<IDictionary>()
    .Select(r => new
                {
                    Category = r["Category"],
                    Count = r["Count"],
                    AveragePrice = r["AveragePrice"],
                });
```

Join

```
var products = session
    .CreateQuery(@"select p from Product p
left join p.Category as c
left join c.Type t
where t.Name = 'Fruits'")
    .List<Product>();
```

```
var hql = @"select p from Person as p
left join fetch p.Hobbies as h";
var listOfPersons = session.CreateQuery(hql)
  .List<Person>();
```

Multi query

```
public void GetPageOfProducts(int pageNumber, int pageSize)
{
    ISession session = DbSessionFactory.Instance.OpenSession();
    int skip = (pageNumber - 1)*pageSize;
    string countHql = @"select count(p.Id) from Product p";
    IQuery countQuery = session.CreateQuery(countHql);
    var productHql = @"from Product p order by p.UnitPrice asc";
    IQuery resultQuery = session.CreateQuery(productHql)
        .SetFirstResult(skip)
        .SetMaxResults(pageSize);
    IMultiQuery multiQuery = session.CreateMultiQuery()
        .Add<long>("count", countQuery)
        .Add<Product>("page", resultQuery);
    long productCount = ((IList<long>) multiQuery.GetResult("count")).Single();
    IList<Product> products = (IList<Product>) multiQuery.GetResult("page");
}
```

Named queries

```
<query name="CountAllProducts">
   <![CDATA[
   select count(p.Id) from Product p
   ]]>
</query>

<query name="GetAllProducts">
   <![CDATA[
   from Product p order by p.UnitPrice asc
   ]]>
</query>
```

Named queries & futures

```
public void GetPageOfProducts(int pageNumber, int pageSize)
{
    ISession session = DbSessionFactory.Instance.OpenSession();
    var skip = (pageNumber - 1)*pageSize;
    var productCount = session.GetNamedQuery("CountAllProducts")
        .FutureValue<long>();
    var products = session.GetNamedQuery("GetAllProducts")
        .SetFirstResult(skip)
        .SetMaxResults(pageSize)
        .Future<Product>();
    var pageCount = (int) Math.Ceiling(
        productCount.Value/(double) pageSize);
}
```

Detached query

```
string hql = @"from Product p where p.Discontinued";
IDetachedQuery detachedQuery = new DetachedQuery(hql);
IQuery executableQuery = detachedQuery.GetExecutableQuery(session);
IList result = executableQuery.List();
```

Bulk data changes

```
var updateHql = "update Product p set p.UnitPrice = 1.1 * p.UnitPrice";
session.CreateQuery(updateHql).ExecuteUpdate();

var deleteHql = "delete Product p where p.Discontinued = true";
session.CreateQuery(deleteHql).ExecuteUpdate();

var insertHql = @"insert into Product(Id, Name, Category, UnitPrice)
select t.Id, t.Name, t.Category, t.UnitPrice
from ProductTemp t";
session.CreateQuery(insertHql).ExecuteUpdate();
```

Criteria

Restrictions

```
List<Product> products = session.CreateCriteria<Product>()
    .Add(Restrictions.Eq("Name", productName))
    .AddOrder(Order.Asc("UnitPrice"))
    .List<Product>();
```

Restrictions

- Eq, EqProperty
- Ge, Gt, GeProperty, GtProperty
- Le, Lt, LeProperty, LtProperty
- Like
- In
- Between
- Not
- IsNull
- IsNotNull
- Where
- And
- Or

Restrictions

```
List<Product> products = session.CreateCriteria<Product>()
    .Add(Restrictions.And(
        Restrictions.Ge("UnitPrice", minPrice),
        Restrictions.Le("UnitPrice", maxPrice)
            ))
    .AddOrder(Order.Asc("UnitPrice"))
    .List<Product>();
```

Join

```
IList<Category> categories = session.CreateCriteria<Category>()
    .CreateCriteria("Products", JoinType.InnerJoin)
    .Add(Restrictions.Eq("Discount", 0))
    .List<Category>();
```

Paging

```
List<Product> products = session.CreateCriteria<Product>()
    .Add(Restrictions.Eq("Name", productName))
    .SetFirstResult(10)
    .SetMaxResults(10)
    .List<Product>();
```

Projections

```
IList products = session.CreateCriteria<Product>()
.Add(Restrictions.Eq("Name", productName))
.SetProjection(Projections.ProjectionList()
   .Add(Projections.Property("Id"))
   .Add(Projections.Property("Name")))
.SetResultTransformer(Transformers.AliasToBean(typeof(NameID)))
.List();
```

Aggregate functions

```
var productCount = session.CreateCriteria<Product>()
    .Add(Restrictions.Eq("Name", productName))
    .SetProjection(Projections.RowCount())
    .UniqueResult();
```

Multi criteria

```
public void GetPageOfProducts(int pageNumber, int pageSize)
{
   ISession session = DbSessionFactory.Instance.OpenSession();
   int skip = (pageNumber - 1) * pageSize;
   ICriteria rowCount = session.CreateCriteria(typeof(Product))
      .SetProjection(Projections.Count(Projections.Id()));
   ICriteria criteria = session.CreateCriteria(typeof(Product))
      .Add(Restrictions.Gt("UnitPrice", 0))
      .AddOrder(Order.Asc("UnitPrice"))
      .SetFirstResult(skip)
      .SetMaxResults(pageSize);
   IMultiCriteria multiCriteria = session.CreateMultiCriteria()
      .Add<long>("count", rowCount)
      .Add<Product>("page", criteria);
   long productCount = ((IList<long>)multiCriteria.GetResult("count")).Single();
   IList<Product> products = (IList<Product>)multiCriteria.GetResult("page");
}
```

Detached criteria

```
DetachedCriteria detachedCriteria = DetachedCriteria.For<Product>()
    .Add(Restrictions.Like("Name", productName));
IList results = detachedCriteria.GetExecutableCriteria(session).List();
```

QueryOver

QueryOver

```
IList<Product> results = session.QueryOver<Product>()
    .Where(x => x.ProductType == ProductTypes.ProductTypeA)
    .OrderBy(x => x.Name)
    .Desc
    .List();
```

QueryOver

```
IList<Product> results = session.QueryOver<Product>()
    .WhereRestrictionOn(x => x.UnitPrice)
    .IsBetween(20).And(50)
    .OrderBy(x => x.Name)
    .Desc
    .List();
```

QueryOver

```
IList<NamePrice> results = session.QueryOver<Product>()
    .Select(m => m.Name, m => m.UnitPrice)
    .List<object[]>()
    .Select(props =>
          new NamePrice()
            {
               Name = (string) props[0],
               Price = (decimal) props[1]
            });
```

Projections

```
var result = session.QueryOver<Product>()
    .Select(Projections.Avg<Product>(m => m.UnitPrice))
    .SingleOrDefault<double>();
```

Join, paging

```
IList<Category> categories = session.QueryOver<Category>()
    .Inner.JoinQueryOver(x => x.Products)
    .Skip(20)
    .Take(10)
    .List<Category>();
```

Session methods

Session actions

```
using (ISession session = DbSessionFactory.Instance.OpenSession())
{
   using (ITransaction transaction = session.BeginTransaction())
   {
      // create, update, delete, or read data
      transaction.Commit();
   }
}
```

Session methods

- Insert
- Save
- Update
- SaveOrUpdate
- Delete
- Get
- Load
- Merge
- Flush
- Refresh
- Evict
- Clear
- Close

Литература

- Benjamin Perkins - Working with Nhibernate 3.0
- Jason Dentler – Nhibernate 3.0. Cookbook
- Dr. Gabriel Nicolas Schenker, Aaron Cure – Nhibernate 3. Beginners guide
- SQL in Nhibernate - http://knol.google.com/k/nhibernate-chapter-14-native-sql#