02

Recuperando dados

Agora que já vimos como mapear as entidades e seus relacionamentos, vamos aprender como escrever consultas no banco de dados com o NHibernate.

Como vimos anteriormente, a comunicação com o banco de dados é feita através do `ISession`, logo para realizarmos as consultas, precisaremos da sessão do NHibernate:

```
ISession session = NHibernateHelper.AbreSession();
```

Consultas no NHibernate são feitas através da linguagem HQL, a Hibernate Query Language, uma linguagem muito parecida com a SQL, mas que trabalha com entidades do C# ao invés de tabelas do banco de dados.

Utilizando a SQL, quando queremos buscar todos os produtos do banco de dados, utilizamos a seguinte query:

```
select * from Produto
```

A consulta em HQL fica muito parecida com a escrita em SQL:

```
select p from Produto p
```

Como o objetivo da query é simplesmente listar todos os produtos, podemos simplificá-la para:

```
from Produto
```

Agora que temos a query, precisamos criar uma consulta, uma instância de `NHibernate.IQuery`, com a hql que escrevemos. Para criar a consulta, utilizamos o método `CreateQuery` do `ISession`:

```
String hql = "from Produto";
IQuery query = session.CreateQuery(hql);
```

E para recuperarmos a sua lista de resultados, utilizamos o método `List` da interface `IQuery`:

```
String hql = "from Produto";
IQuery query = session.CreateQuery(hql);
IList<Produto> produtos = query.List<Produto>();
```

E agora, podemos, por exemplo, utilizar o foreach para imprimir os produtos que foram recuperados do banco de dados:

```
String hql = "from Produto";
IQuery query = session.CreateQuery(hql);
IList<Produto> produtos = query.List<Produto>();
foreach(var produto in produtos)
```

```
{
    Console.WriteLine(produto.Nome);
}
```

Além disso, podemos pedir para o NHibernate ordenar os resultados de uma consulta utilizando o %%order by%%:

```
string hql = "from Produto p order by p.Nome";
```

Buscando produtos por preço

Agora que já aprendemos como fazer uma query que lista entidades, vamos aprender como colocar restrições na busca. Assim como na sql, utilizamos a palavra `where` para definir condições:

```
from Produto p where condições
```

Queremos buscar todos os produtos com preço maior do que 10.0, então a condição da query deve aceitar apenas produtos com a propriedade Preco maior do que o valor 10.0:

```
from Produto p where p.Preco > 10.0
```

E se também quisermos os produtos com preço maior do que 100.0? Teríamos que escrever uma query diferente. Precisamos definir um parâmetro nessa busca que será o preço mínimo do produto. Fazemos isso colocando uma '?' on queremos marcar um parâmetro:

```
string hql = "from Produto p where p.Preco > ?";
```

Para substituir o valor do parâmetro utilizamos o método `SetParameter` do objeto `IQuery` passando o índice do parâmetro e seu valor.

```
string hql = "from Produto p where p.Preco > ?";
IQuery query = session.CreateQuery(hql);
query.SetParameter(0, 10.0);
IList<Produto> produtos = query.List<Produto>();
```

Porém marcar os parâmetros com '?' pode deixar o código confuso, por isso o NHibernate também permite a criação de parâmetros nomeados. Para criar um parâmetro nomeado, colocamos no texto da query o nome do parâmetro precedido pelo caractere ':'

```
string hql = "from Produto p where p.Preco > :minimo"
```

Nessa query, definimos um parâmetro chamado `minimo` e para substituir seu valor também utilizamos o método `SetParameter` passando o nome do parâmetro e seu valor:

```
string hql = "from Produto p where p.Preco > :minimo";
IQuery query = session.CreateQuery(hql);
```

```
query.SetParameter("minimo", 10.0);
IList<Produto> produtos = query.List<Produto>();
```

Busca de produtos por categoria

Queremos recuperar todos os produtos que pertencem a uma categoria chamada informática, porém a Categoria é um outro modelo do sistema e é representada por uma tabela diferente no banco de dados.

Para realizarmos essa busca, utilizando a SQL, teríamos que utilizar o join:

```
select p.*
from Produto p
     inner join Categoria c on p.CategoriaId = c.Id
where c.Nome = 'Informatica'
```

Com a HQL, estamos fazendo buscas em objetos, então não precisamos nos preocupar com as tabelas do banco de dados, precisamos apenas acessar as propriedades da entidade. Para filtrarmos os produtos cuja categoria tem nome igual a um parâmetro chamado %%categoria%%, utilizamos a seguinte query:

```
from Produto p where p.Categoria.Nome = :categoria
```

Repare que na query acima, não precisamos nos preocupar com o join entre as tabelas, é responsabilidade do NHibernate enviar a SQL correta para o banco de dados.

Podemos também utilizar diversas condições na query. Assim como na sql, podemos utilizar o `AND` e o `OR` para juntar condições. Por exemplo, se quiséssemos todos os produtos cuja categoria tem um determinado nome e com preço maior do que um valor mínimo, poderíamos utilizar a seguinte query:

```
from Produto p where p.Categoria.Nome = :categoria and p.Preco > :minimo
```

Número de produtos por categoria

Agora que já vimos como a HQL funciona, vamos escrever uma query mais avançada. Queremos recuperar o número de produtos agrupados por categoria. Para resolvermos esse problema, assim como na SQL, utilizaremos o `group by`.

A HQL que resolve o problema proposto deve devolver a categoria junto com o número de produtos daquela categoria:

```
string hql = "select p.Categoria, count(p) from Produto p group by p.Categoria";
```

Agora que temos a HQL, podemos criar o objeto `IQuery` e recuperar a lista de resultados:

```
string hql = "select p.Categoria, count(p) from Produto p group by p.Categoria";
IQuery q = session.CreateQuery(hql);
q.List();
```

Porém qual será o tipo da lista? Nessa busca, cada linha do resultado contém uma categoria junto com o número de produtos pertencentes a aquela categoria e, por esse motivo, nessa situação, o NHibernate devolve um `IList` de `Object[]`.

```
string hql = "select p.Categoria, count(p) from Produto p group by p.Categoria";
IQuery q = session.CreateQuery(hql);
IList<Object[]> resultados = q.List<Object[]>();
```

Trabalhar com o `IList` é muito complicado, portanto guardaremos o resultado da query em um modelo mais amigável. Criaremos a classe `ProdutosPorCategoria` que guardará uma categoria em uma propriedade chamada `Categoria` e o número de produtos daquela categoria na propriedade `NumeroDeProdutos`:

```
public class ProdutosPorCategoria
{
    public Categoria Categoria { get; set; }

    public long NumeroDeProdutos { get; set; }
}
```

Agora precisamos transformar o `IList` em um `IList`:

```
string hql = "select p.Categoria, count(p) from Produto p group by p.Categoria";
IQuery q = session.CreateQuery(hql);
IList<Object[]> resultados = q.List<Object[]>();

IList<ProdutosPorCategoria> relatorio = new List<ProdutosPorCategoria>();
foreach(Object[] resultado in resultados)
{
    ProdutosPorCategoria p = new ProdutosPorCategoria();
    p.Categoria = (Categoria) resultado[0];
    p.NumeroDeProdutos = (long) resultado[1];
    relatorio.Add(p);
}
```

O código para transformar o resultado da query em uma lista de `ProdutosPorCategoria` é muito repetitivo e por isso, o NHibernate nos fornece componentes especializados em transformar o resultado de uma query, os Result Transformers.

Queremos realizar uma transformação que pega o resultado de uma query, aloca um objeto e preenche suas propriedades. No java, objetos que possuem um construtor sem argumentos e cujo acesso é feito através de getters e setters são chamados de Beans. O result transformer que consegue transformar um `IList` em uma lista de beans é o `NHibernate.Transform.Transformers.AliasToBean`.

O `AliasToBean` instancia uma classe através de seu construtor sem argumentos e preenche seus campos através dos setters. Para utilizarmos esse transformer, a query deve dar um nome (alias) para cada valor devolvido, os alias são utilizados para procurar as propriedades do Bean.

No exemplo, queremos preencher as propriedades `Categoria` e `NumeroDeProdutos`, logo o select da query deve devolver dois valores, um chamado `Categoria` e outro chamado `NumeroDeProdutos`.

```
string hql = "select p.Categoria as Categoria, count(p) as NumeroDeProdutos " +
             "from Produto p group by p.Categoria";
```

Depois de definirmos a query com os alias, podemos criar o objeto IQuery que representa essa hql:

```
string hql = "select p.Categoria as Categoria, count(p) as NumeroDeProdutos " +
             "from Produto p group by p.Categoria";
IQuery q = session.CreateQuery(hql);
```

E agora, configuraremos o result transformer. Definimos o result transformer da busca através do método `SetResultTransformer` do `IQuery`. Queremos utilizar o `Transformers.AliasToBean` passando o tipo `ProdutosPorCategoria`

```
string hql = "select p.Categoria as Categoria, count(p) as NumeroDeProdutos " +
             "from Produto p group by p.Categoria";
IQuery q = session.CreateQuery(hql);
q.SetResultTransformer(Transformers.AliasToBean<ProdutosPorCategoria>());
```

E agora que colocamos o result transformer, podemos listar o resultado da query, que será do tipo `ProdutosPorCategoria`:

```
string hql = "select p.Categoria as Categoria, count(p) as NumeroDeProdutos " +
             "from Produto p group by p.Categoria";
IQuery q = session.CreateQuery(hql);
q.SetResultTransformer(Transformers.AliasToBean<ProdutosPorCategoria>());
IList<ProdutosPorCategoria> resultado = q.List<ProdutosPorCategoria>();
```