.NET - NHibernate a revanche (Gerando os arquivos de mapeamento)

Um dos sabores mais usados do **LINQ** , pelo menos para o acesso a dados no SQL Server, é (ou foi) o **LINQ to SQL**; logo em seguida veio o lançamento do **Entity Framework** e parece que as coisas andam meio confusas com relação ao futuro do **LINQ to SQL**.

Tudo começou com uma notícia veiculada no *blog* da equipe da Microsoft :

- http://blogs.msdn.com/adonet/archive/2008/10/29/update-on-linq-to-sql-and-linq-to-entities-roadmap.aspx

que gerou muito comentário:

- http://codebetter.com/blogs/david.hayden/archive/2008/10/31/linq-to-sql-is-dead-read-between-the-lines.aspx
- http://ayende.com/Blog/archive/2008/10/31/microsoft-kills-linq-to-sql.aspx

Assim, se tudo se confirmar, o **LINQ to SQL** esta com os dias contados. (*A Microsoft ainda vai dar suporte, bla,bla,bla...*)

O que eu acho engraçado é que quem optou em usar o **LINQ to SQL** em seus projetos não tem outra opção confiável no momento. (Eu particularmente não usaria o instável e pouco confiável **Entity Framework** para aplicações em produção, pois não tenho vocação para cobaia. E, não estou só nesta questão http://efvote.wufoo.com/forms/ado-net-entity-framework-vote-of-no-confidence/)

Eu cheguei a ler comparações entre o **NHibernate** e o **Entity Framework** nas quais decretava que a ferramenta da Microsoft era superior, isso após a mesma ter acabado de ser lançada. Um absurdo !!!

Nenhum case de sucesso no mercado, nenhuma referência de projeto em produção comercial enquanto que o NHibernate, o porte para Hibernate do Java, embora tenha seus problemas, está há mais tempo no mercado com muita gente usando em produção.

Por falar em **NHibernate**, ele continua a todo vapor e a comunidade tem agregado ferramentas que tornam o trabalho com a ferramenta mais produtivo e menos tediosa.

Se você conhece o NHibernate já sabe do que estou falando. A ferramenta é muito boa mas o trabalho que dá gerar os arquivos de configuração e mapeamento para grandes projetos não é brincadeira.

Felizmente existem algumas opções de ferramentas que tem o objetivo de efetuar a geração dos arquivos de configuração e mapeamento de forma automática e assim facilitar a utilização do NHibernate. Veja abaixo algumas opções:

- AWA - http://www.codeproject.com/useritems/NHibernate_Helper_Kit.asp
- CodeSmith - http://www.codesmithtools.com/ **(trial 30 dias)**
- MyGeneration - http://www.mygenerationsoftware.com/
- ActiveWriter - http://using.castleproject.org/display/Contrib/ActiveWriter

Neste artigo eu vou mostrar como usar o **MyGeneration** para gera os arquivos de mapeamento para o NHibernate tornando assim a sua vida mais fácil e encorajando-o a usar esta poderosa ferramenta que por enquanto é uma opção embassada em projetos de sucesso e que tem uma comunidade ativa.

Gerando os arquivos de mapeamento para o NHibernate

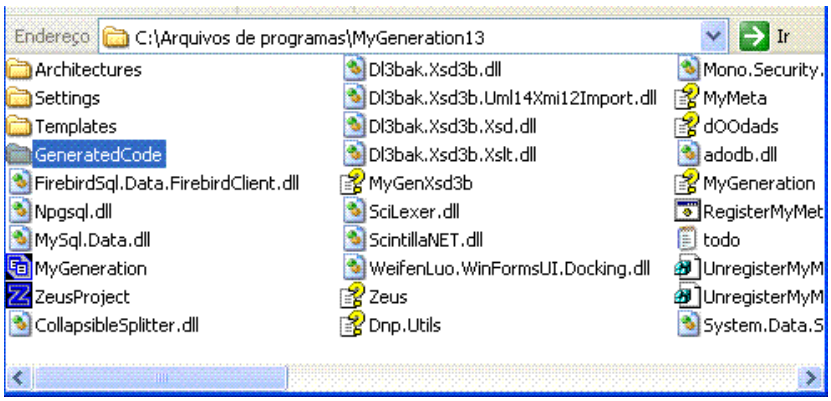
Recursos necessários:

- **NHibernate 2.0.1.GA** *(para ser referenciado no seu projeto)*
- **SQL Server 2005 Express Edition**
- **Microsoft SqlServer Management Studio Express**
- **MyGeneration (versão atual 1.3.0.3)**
- **Templates para NHibernate** *(Os templates pode ser obtidos no site do MyGeneration)*

Roteiro de utilização:

- **Efetuar o download da última versão do NHibernate;** *(para usar no projeto, e isso eu não vou mostrar neste artigo.)*
- **Criação do banco de dados, das tabelas e relacionamentos no SQL Server 2005;**
- **Instalação do MyGeneration;**
- **Instalação dos templates para NHibernate;**
- **Geração dos arquivos de mapeamento;**

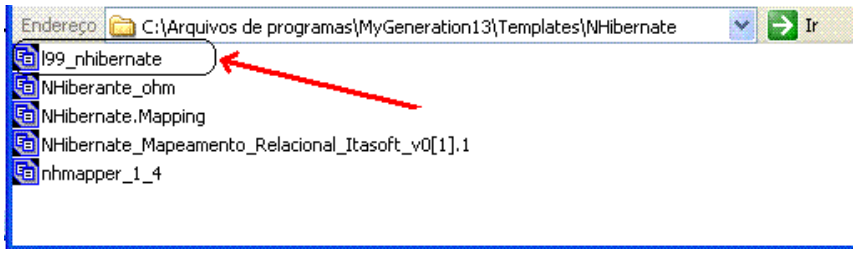
Por padrão a instalação do **MyGeneration** é feita na pasta **\Arquivos de Programas\MyGeneration13;** ***(você pode alterar este comportamento)***



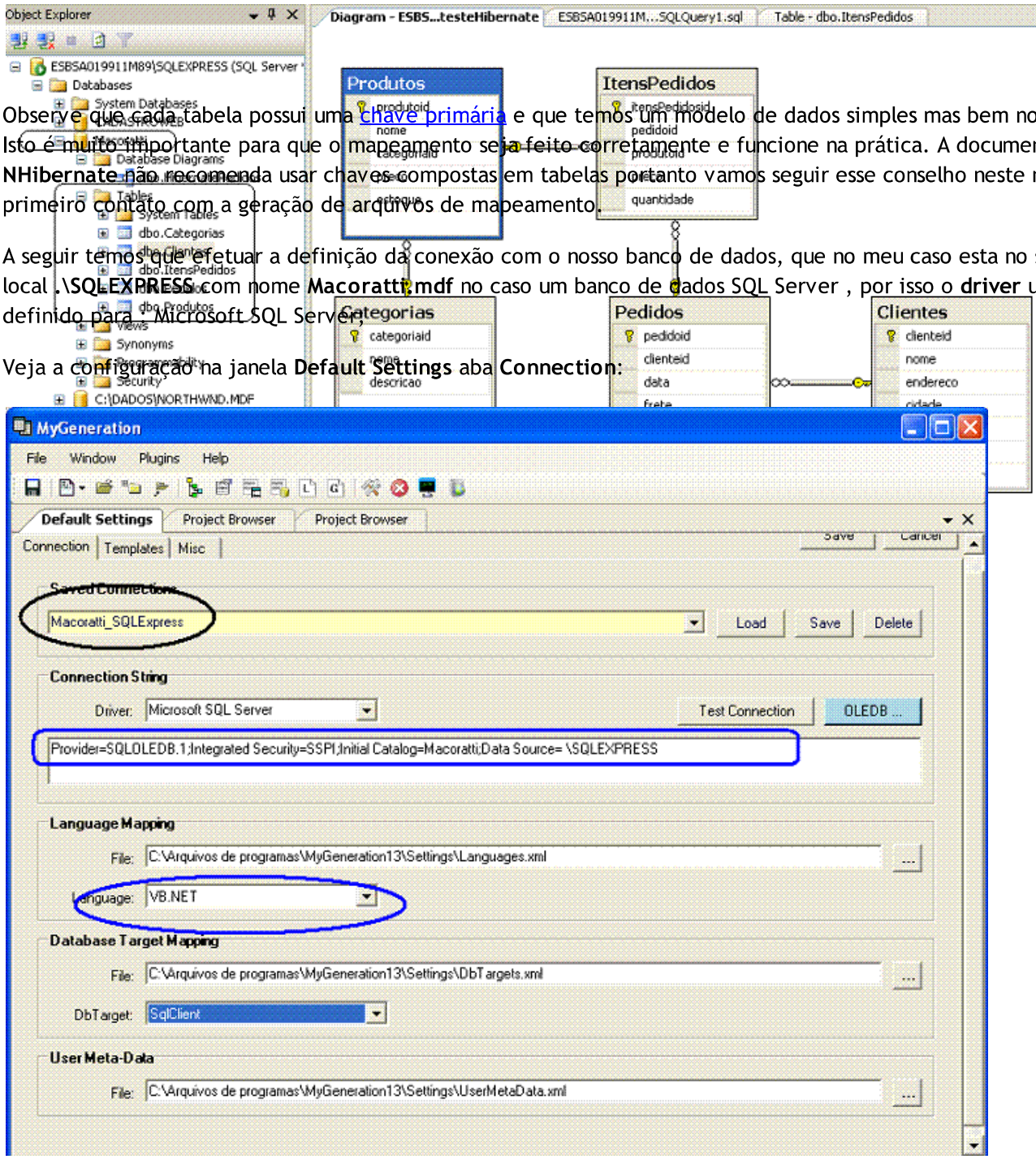
Existem duas sub-pastas importantes na pasta principal de instalação:

- **Templates - Pasta onde devem ser copiados os templates para o NHibernate;**
- **GeneratedCode - Pasta onde serão gerados os arquivos de mapeamento;**

Existem diversos *templates* para o **NHibernate** que você pode usar. Eu baixei alguns *templates* para o **NHibernate** mas neste artigo eu vou usar o **l99_nhibernate** por achar que ele é mais completo. *(Você pode efetuar testes com os demais templates e usar o que achar mais adequado)*



Eu vou criar um banco de dados no **SQL Server 2005 Express Edition** usando a ferramenta **Management Studio Express** chamado **Macoratti.mdf** contendo as seguintes tabelas e relacionamentos: *(Não vou dar detalhes de como criar o banco de dados e as tabelas, veja nas referências links de como fazer este serviço)*



Observe que cada tabela possui uma chave primária e que temos um modelo de dados simples mas bem normalizado. Isto é muito importante para que o mapeamento seja feito corretamente e funcione na prática. A documentação do **NHibernate** não recomenda usar chaves compostas em tabelas portanto vamos seguir esse conselho neste nosso primeiro contato com a geração de arquivos de mapeamento.

A seguir temos que efetuar a definição da conexão com o nosso banco de dados, que no meu caso esta no servidor local **.\SQLEXPRESS** com nome **Macoratti.mdf** no caso um banco de dados SQL Server , por isso o **driver** usado foi definido para ' Microsoft SQL Server';

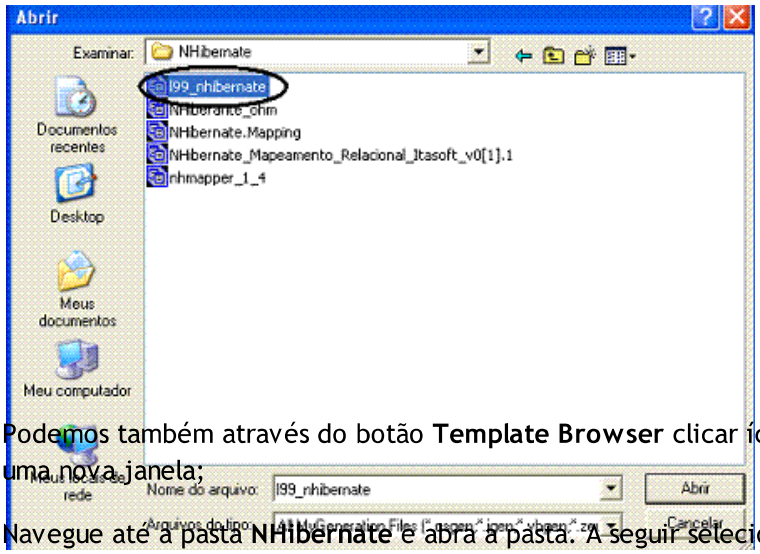
Veja a configuração na janela **Default Settings** aba **Connection**:

Observe que eu selecionei a linguagem **VB .NET** que será usada para gerar os arquivos de mapeamento mas poderia ter usado C#.

Após estas configurações clique no botão - **Test Connection** - e você deverá obter uma mensagem de conexão bem sucedida. Se houver erro verifique a string de conexão e tente novamente.

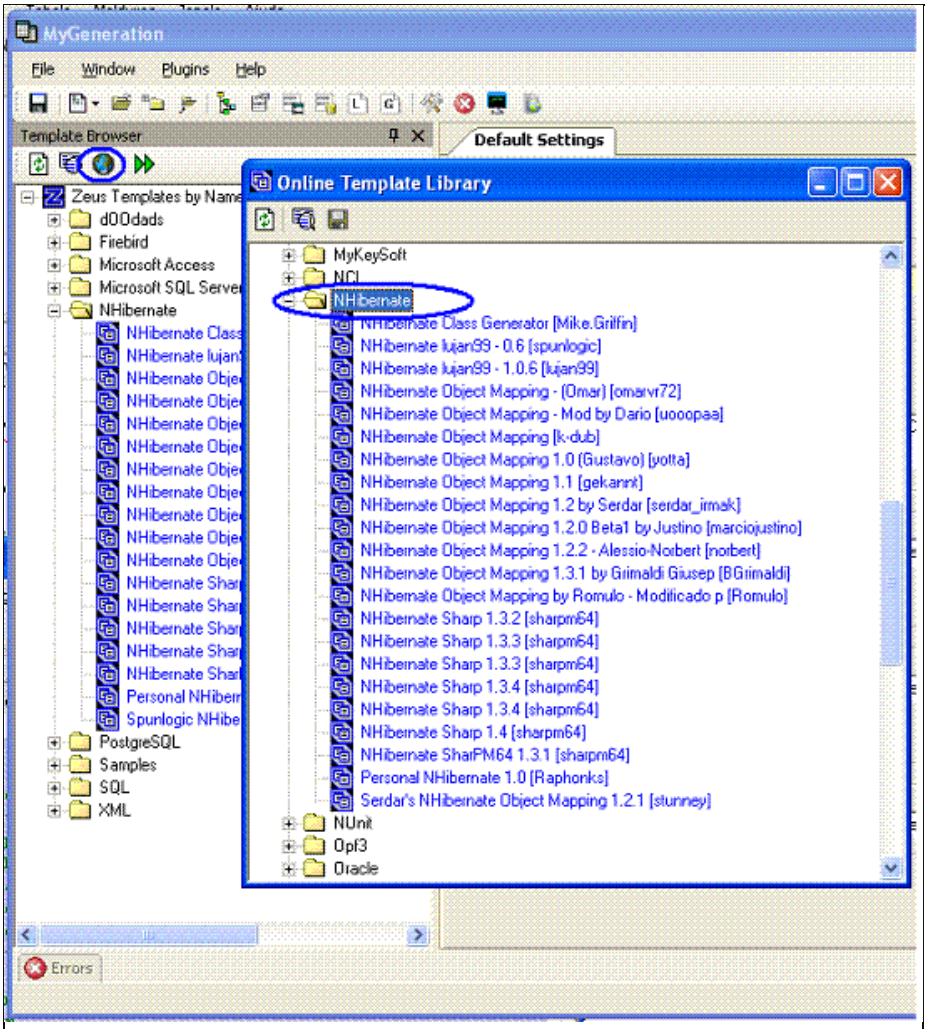
Vamos agora usar um *template* adequado para gerar os arquivos de mapeamento. Clique no menu **File -> Open** e localiza a pasta **Templates** e no seu interior a sub-pasta **NHibernate** onde foram copiados os *templates* para o **NHibernate**.

A seguir selecione o *template* **l99_nhibernate** conforme abaixo:

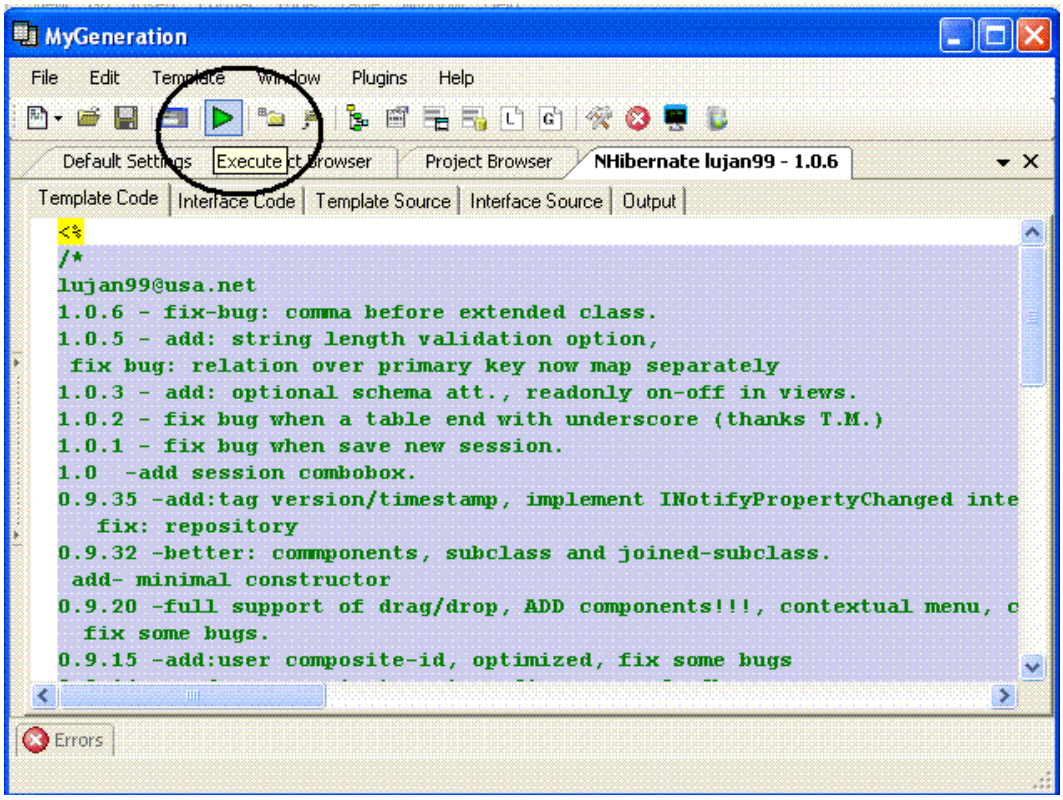


Podemos também através do botão **Template Browser** clicar ícono do globo *(On line Template Library*) que abrirá uma nova janela;

Navegue até a pasta **NHibernate** e abra a pasta. A seguir selecione o template com o qual deseja trabalhar. No nosso exemplo : **lujan99-1.0.6;**

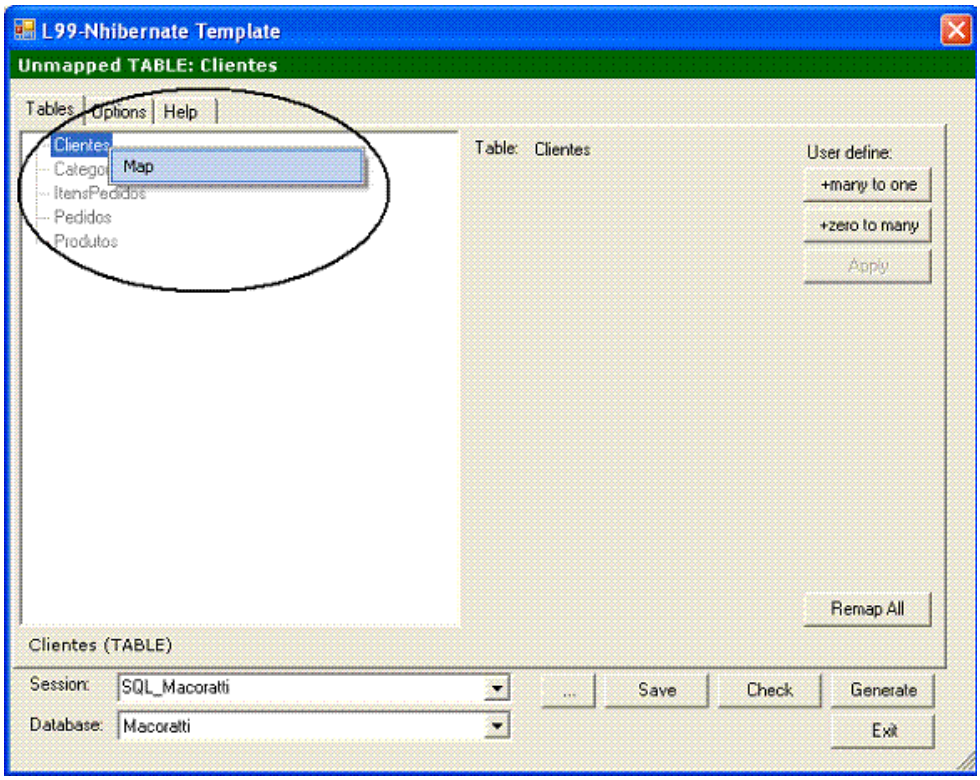


Na próxima tela clique no botão **Execute**:

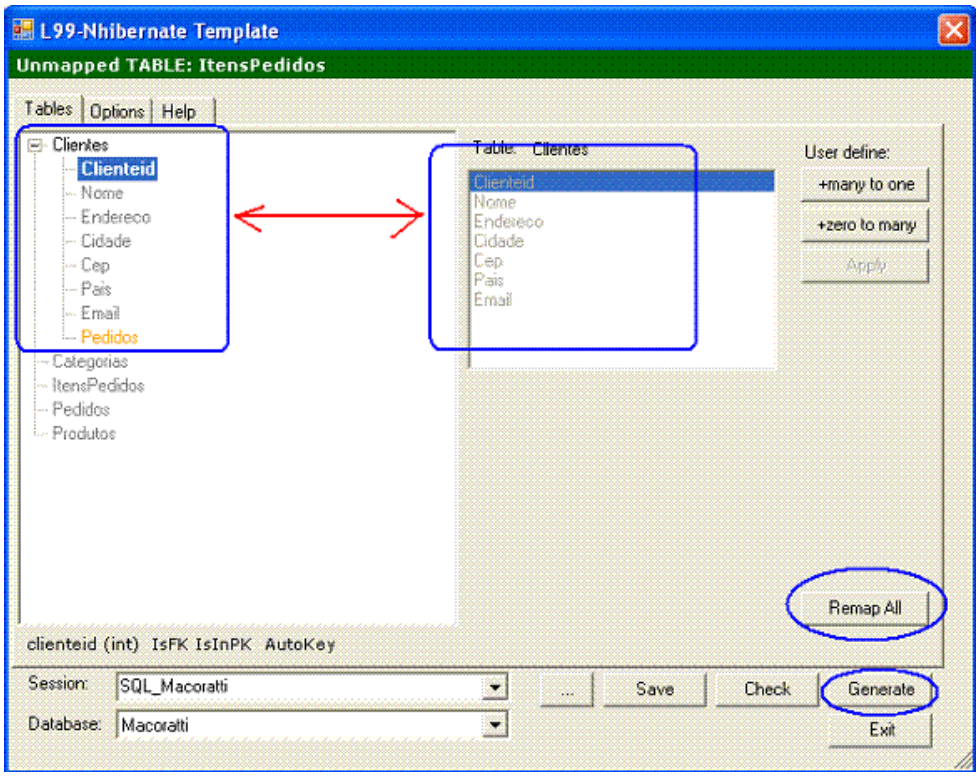


Você verá a tela exibindo as entidades do seu banco de dados, no nosso exemplo **Macoratti.mdf.** Neste momento você pode deve selecionar a entidade que deseja mapear clicando com o botão direito do mouse selecionar a opção **Map**.

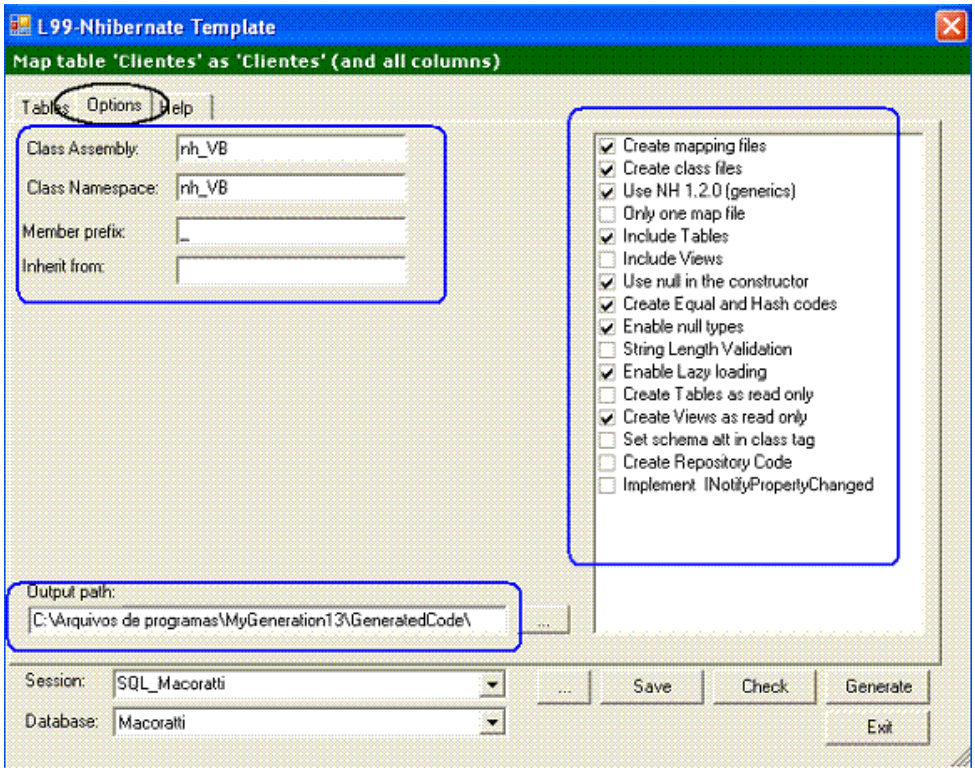
Ao fazer isso serão exibidos as propriedades da entidade mapeada



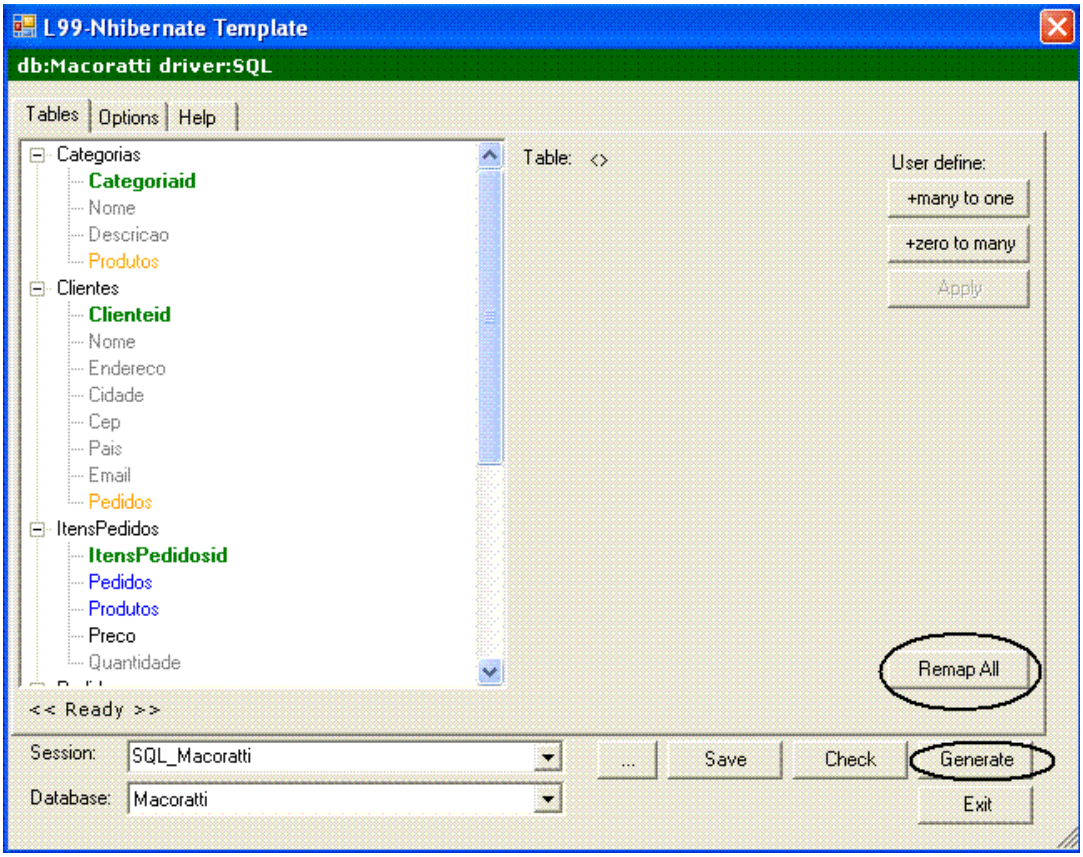
A seguir podemos visualizar as tabelas e as entidades bem como definir alguns mapeamentos;



Na guia **Options** podemos alterar/definir o nome do Assembly, do namespace, o prefixo dos membros e definir opções para geração dos arquivos de mapeamento;

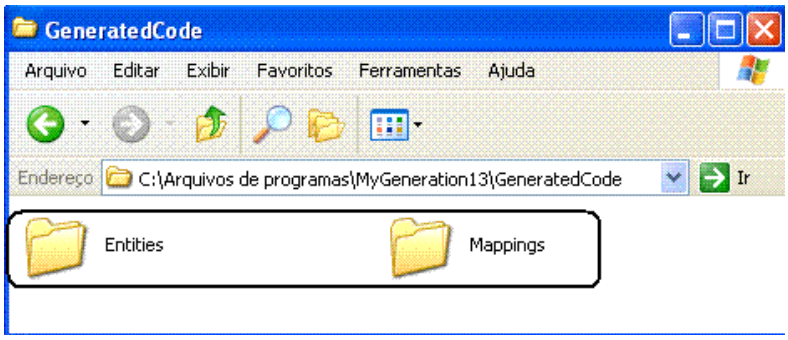


Após definirmos para quais entidades desejamos gerar os arquivos de mapeamento estamos prontos para continuar. No nosso exemplo usamos o botão **Remap All** para mapear todas as entidades;

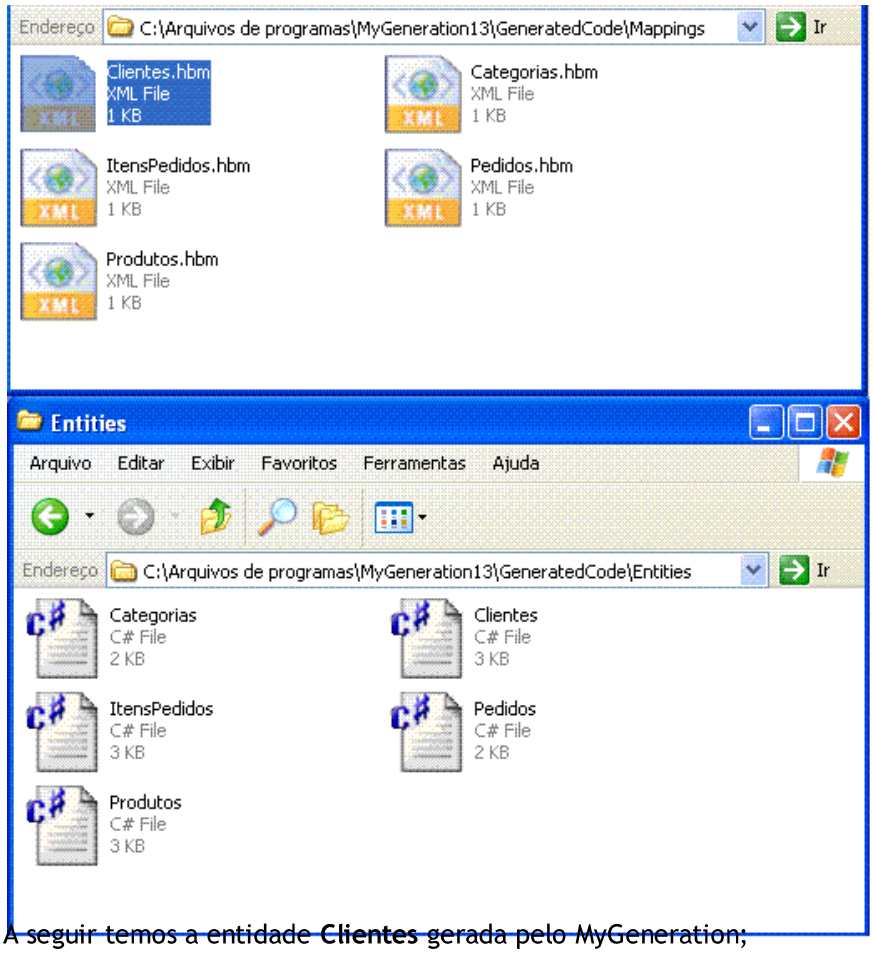


Ao clicar no botão **Generate** serão criadas duas pastas na pasta **GenerateCode** contendo as entidades e os mapeamentos gerados;

Nota: você pode definir outro local para gerar os arquivos de mapeamento.



Nas figuras abaixo vemos os arquivos de mapeamento **.hbm** e as entidades geradas para o nosso exemplo:



A seguir temos a entidade **Clientes** gerada pelo MyGeneration;

```
/*
using MyGeneration/Template/NHibernate (c) by lujan99@usa.net
*/
using System;
using System.Collections;
using System.Collections.Generic;

namespace nh_VB
{
	/// <summary>
	/// Clientes object for NHibernate mapped table 'Clientes'.
	/// </summary>
	[Serializable]
	public class Clientes
	{
		#region Member Variables
		protected Integer _clienteid;
		protected String _nome;
		protected String _endereco;
		protected String _cidade;
		protected String _cep;
		protected String _pais;
		protected String _email;
		protected IList<Pedidos> _pedidos;
		#endregion
		#region Constructors

		public Clientes() {}

		public Clientes(String nome, String endereco, String cidade, String cep, String pais, String email)
```

```
{
	this._nome= nome;
	this._endereco= endereco;
	this._cidade= cidade;
	this._cep= cep;
	this._pais= pais;
	this._email= email;
}

#endregion
#region Public Properties
public  virtual Integer Clienteid
{
	get { return _clienteid; }
	set {_clienteid= value; }
}
public  virtual String Nome
{
	get { return _nome; }
	set {_nome= value; }
}
public  virtual String Endereco
{
	get { return _endereco; }
	set {_endereco= value; }
}
public  virtual String Cidade
{
	get { return _cidade; }
	set {_cidade= value; }
}
public  virtual String Cep
{
	get { return _cep; }
	set {_cep= value; }
}
public  virtual String Pais
{
	get { return _pais; }
	set {_pais= value; }
}
public  virtual String Email
{
	get { return _email; }
	set {_email= value; }
}
public  virtual IList<Pedidos> Pedidos
{
	get { return _pedidos; }
	set {_pedidos= value; }
}
#endregion

#region Equals And HashCode Overrides
/// <summary>
/// local implementation of Equals based on unique value members
/// </summary>
```

```
        public override bool Equals( object obj )
        {
            if( this == obj ) return true;
            if( ( obj == null ) || ( obj.GetType() != this.GetType() ) ) return false;
            Clientes castObj = (Clientes)obj;
            return ( castObj != null ) &&
            this._clienteid == castObj.Clienteid;
        }
        /// <summary>
        /// local implementation of GetHashCode based on unique value members
        /// </summary>
        public override int GetHashCode()
        {
            int hash = 57;
            hash = 27 * hash * _clienteid.GetHashCode();
            return hash;
        }
        #endregion

    }
}
```

Abaixo temos o arquivo de mapeamento para a entidade **Clientes**:

```
<?xml version="1.0" encoding="utf-8"?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
  <!--Build: with lujan99@usa.net Nhibernate template-->
  <class name="nh_VB.Clientes,nh_VB" table="Clientes" lazy="true">
    <id name="Clienteid" column="clienteid" type="Integer">
      <generator class="native" />
    </id>
    <property name="Nome" column="nome" type="String" />
    <property name="Endereco" column="endereco" type="String" />
    <property name="Cidade" column="cidade" type="String" />
    <property name="Cep" column="cep" type="String" />
    <property name="Pais" column="pais" type="String" />
    <property name="Email" column="email" type="String" />
    <bag name="Pedidos" inverse="true" lazy="true" cascade="delete">
      <key column="clienteid" />
      <one-to-many class="nh_VB.Pedidos,nh_VB" />
    </bag>
  </class>
</hibernate-mapping>
```

A seguir temos o arquivo de mapeamento para a entidade **Produtos**:

Clientes.cs | Clientes.hbm.xml | Pedidos.hbm.xml | **Produtos.hbm.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
  <!--Build: with lujan99@usa.net Nhibernate template-->
  <class name="nh_VB.Produtos,nh_VB" table="Produtos" lazy="true">
    <id name="Produtoid" column="produtoid" type="Integer">
      <generator class="native" />
    </id>
    <property name="Nome" column="nome" type="String" />
    <many-to-one name="Categorias" column="categoriaid" cascade="save-update" />
    <property name="Preco" column="preco" type="Decimal" />
    <property name="Estoque" column="estoque" type="Integer" />
    <bag name="ItensPedidos" inverse="true" lazy="true" cascade="delete">
      <key column="produtoid" />
      <one-to-many class="nh_VB.ItensPedidos,nh_VB" />
    </bag>
  </class>
</hibernate-mapping>
```

Agora temos o arquivo de mapeamento para a entidade **Categorias:**

Clientes.cs | Clientes.hbm.xml | Pedidos.hbm.xml | Produtos.hbm.xml | **Categorias.hbm.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
  <!--Build: with lujan99@usa.net Nhibernate template-->
  <class name="nh_VB.Categorias,nh_VB" table="Categorias" lazy="true">
    <id name="Categoriaid" column="categoriaid" type="Integer">
      <generator class="native" />
    </id>
    <property name="Nome" column="nome" type="String" />
    <property name="Descricao" column="descricao" type="String" />
    <bag name="Produtos" inverse="true" lazy="true" cascade="delete">
      <key column="categoriaid" />
      <one-to-many class="nh_VB.Produtos,nh_VB" />
    </bag>
  </class>
</hibernate-mapping>
```

Aqui temos o arquivo de mapeamento para a entidade **ItensPedidos**. Observe a o relacionamento com as entidades **Produtos** e **Pedidos** esta definida em um mapeamento many-to-one:

ItensPedidos.hbm.xml

```
<?xml version="1.0" encoding="utf-8"?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
  <!--Build: with lujan99@usa.net Nhibernate template-->
  <class name="nh_VB.ItensPedidos,nh_VB" table="ItensPedidos" lazy="true">
    <id name="ItensPedidosid" column="itensPedidosid" type="Integer">
      <generator class="native" />
    </id>
    <many-to-one name="Pedidos" column="pedidoid" cascade="save-update" not-null="true" />
    <many-to-one name="Produtos" column="produtoid" cascade="save-update" not-null="true" />
    <property name="Preco" column="preco" type="Decimal" not-null="true" />
    <property name="Quantidade" column="quantidade" type="Short" />
  </class>
</hibernate-mapping>
```

Finalmente temos o arquivo de mapeamento para a entidade **Pedidos** onde temos o relacionamento com a entidade **Clientes** definido no mapeamento many-to-one.

Clientes.cs | Clientes.hbm.xml | Pedidos.hbm.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
3    <!--Build: with lujan99@usa.net Nhibernate template-->
4    <class name="nh_VB.Pedidos,nh_VB" table="Pedidos" lazy="true">
5      <id name="Pedidoid" column="pedidoid" type="Integer">
6        <generator class="native" />
7      </id>
8      <many-to-one name="Clientes" column="clienteid" cascade="save-update" />
9      <property name="Data" column="data" type="DateTime" />
10     <property name="Frete" column="frete" type="Decimal" />
11     <bag name="ItensPedidos" inverse="true" lazy="true" cascade="delete">
12       <key column="pedidoid" />
13       <one-to-many class="nh_VB.ItensPedidos,nh_VB" />
14     </bag>
15   </class>
16 </hibernate-mapping>
```

Vejamos a seguir a explicação para alguns dos atributos usados nestes arquivos de mapeamento gerados:

- O atributo **name** indica o nome da propriedade/campo do objeto;

- O atributo **column** indica o nome da coluna na base de dados;

- O elemento **<generator>** indica a forma como o id é gerado. Temos as seguintes possibilidades:.

1. **Increment** - Lê o valor máximo da chave primária e incrementa;
2. **Identity** - Mapeado para colunas identity no *DB2, MySQL, MSSQL, SyBase, Informix*;
3. **sequence** - Mapeado em sequências no *DB2, PostGreSQL, Oracle, SAP DB, Firebird*;
4. **hilo** - Usa um algoritmo **high/low** para gerar chaves únicas;
5. **uudi.hex** - Usa uma combinação do IP com um *timestamp* para gerar um identificador único.
6. **guid** - Usa o novo **system.guid** como identificador;
7. **assigned** - será gerado pela aplicação;
8. **native** - Deixa o banco de dados cuidar da geração;

- O atributo **lazy** definido como **true** indica *'quanto mais tarde melhor'*, isto é, a coleção só vai ser preenchida, quando realmente for necessária e quando se acessar a alguma propriedade da coleção que necessite dessa informação. Evitam-se assim acessos desnecessários na base de dados;

- O elemento **<key>** indica qual é a chave estrangeira. (é o que vai aparecer na cláusula **WHERE**);

- O elemento **<one-to-many>** indica a cardinalidade da relação e qual a classe com que esta se relaciona.

- O atributo **cascade** indica se as operações devem ser feitas em cascata. Por exemplo, se eu excluir um pedido que tenha itens associados, estes também devem ser excluídos ;

- O atributo **inverse** indica que esta é a direção inversa da relação. Em relações de um-para-muitos ou de muitos-para-um, deve ficar sempre do lado do **muitos**, em relações **muitos-para-muitos**, isto é indiferente. Este atributo serve para que o Nhibernate não tente incluir os dados duas vezes na base de dados, o que geraria uma exceção de violação da chave primária.

Nota: Para termos acesso as coleções , podemos utilizar um dos vários tipos de mapeamento de coleções disponibilizados pelo **NHibernate**, através dos elementos: **<bag>, <set>, <array> e <list>**

- O elemento **bag** representa o **iList** da classe, é a responsável por cadastrar ou trazer as listas que um determinado objeto tem como propriedade sem a necessidade de mais consultas *hql*, usando apenas o mapeamento e uma chamada **hql. (***HQL é uma linguagem que é parecida com o SQL, porém é independente do SGBD).*

Vemos que o trabalho para geração dos arquivos de mapeamento pode ser feito por uma ferramenta como o **MyGenaration** aumentando assim a produtividade. Existem outras ferramentas como **Castle ActiveRecord** que é uma implementação do padrão **Active Record** desenvolvido em .NET em cima do **NHibernate** que permite a possibilidade de mapeamento de classes via atributos e não dos arquivos xml como também um mecanismo de validação eficiente e flexível.

Aguarde mais artigos sobre o **NHibernate** em breve... 

Eu sei é apenas **NHibernate**, mas eu gosto...

Referências:

- NHibernate - Efetuando o mapeamento XML 1:N
- VB .NET- Usando o NHibernate com o VB 2008
- NHibernate Users FAQ
- Databases supported by NHibernate
- Miscellaneous NHibernate resources
- NHibernate Quick Start Guide
- NHibernate 1.2 Migration Guide
- VB .NET - LINQ to SQL - Criando entidades
- http://www.castleproject.org/
- http://www.nhibernate.org/

José Carlos Macoratti