

o jeito simples e gratuito de fazer web.
Faça o download e comece agora mesmo!



PORTAL

AGENDA MULTIMÍDIA

BOX

COLETIVOS

IMASTERS PRO

CODE

FÓRUM

INTERCON 2011

Faça L...



.NET

Pesquisar...

[.NET](#) + SQL Server

ASP .NET - usando o NHibernate em uma aplicação Web - Parte 01

Quarta-feira, 15/12/2010 às 11h00, por José Carlos Macoratti

Após muitos artigos dedicados ao Entity Framework, vamos falar novamente do NHibernate, desta vez usando a linguagem C# em um projeto ASP .NET Web Forms.

O NHibernate é um porto do consagrado framework Hibernate para Java, para a plataforma .NET. Dessa forma, o NHibernate surgiu a partir do Hibernate, um framework muito usado na plataforma Java.

Podemos pensar no NHibernate como uma ferramenta ORM de persistência e ele não está só pois existem muitas outras ferramentas que se propõe a fazer a mesma coisa. Veja a relação abaixo:

Lista das ferramentas ORM e/ou Frameworks de persistência mais conhecidos:

- [ADO.NET Entity Framework](#), Microsoft's ORM, part of .Net 4.0
- [AgileFx](#), open source
- [Base One Foundation Component Library](#), free or commercial
- [Devart LinqConnect](#), commercial, an ORM solution for [Oracle](#), [MySQL](#), [PostgreSQL](#), and [SQLite](#)
- [Castle ActiveRecord](#), ActiveRecord for .NET, open source
- [Database Objects .NET](#), Open Source
- [DataObjects.NET](#), GPL and commercial
- [DevForce](#), commercial, N-Tier
- [ECO](#), Commercial but free for use up to 12 classes
- [EntitySpaces](#), commercial
- [Habanero](#), Free open source Enterprise application framework with a Free Code Generation Tool
- [iBATIS](#), Free open source
- [LINQ to SQL](#), Free, .NET Framework component
- [LLBLGen Pro](#), commercial
- [Neo](#), open source

ÚLTIMAS NOTÍCIAS

- 02/01 às 11h40 [Microsoft corrige falhas de segurança no ASP.net](#)
- 08/10/2010 [MS anuncia gerenciador de pacotes de código aberto .NET](#)
- 01/12/2009 [Microsoft adiciona SSL CDN](#)
- 09/02/2009 [Marten Mickos deixa Site MicroSystems](#)
- 30/04/2004 [101 exemplos VB.Net](#)

[VER MAIS NOTÍCIAS](#)

- [NHibernate](#), open source
- [nHydrate](#), open source
- [ObjectMapper .NET](#), GPL and commercial license
- [OpenAccess ORM](#), free or commercial
- [Persistor.NET](#), free or commercial
- [Quick Objects](#), free or commercial
- [SubSonic](#), open source

A grande vantagem do NHibernate é que ela é uma ferramenta gratuita, madura e com uma grande aceitação e utilização na comunidade .NET.

Usando o NHibernate com a linguagem VB .NET em uma aplicação ASP .NET Web Forms

Neste artigo, eu vou mostrar como criar uma aplicação simples que mostra como usar o NHibernate para fazer o mapeamento objeto relacional e persistir dados no SQL Server 2005 Express Edition.

Recursos necessários:

- [NHibernate 2.1.2.GA](#)
- [Visual Web Developer 2010 Express Edition](#)
- [SQL Server 2005 Express Edition](#)

Existem duas formas de usarmos o **NHibernate**:

1. A partir do modelo de dados já criado, realizar o mapeamento ORM para as entidades;
2. Criar a estrutura das tabelas no banco de dados a partir das entidades definidas nos arquivos de configuração (**Model First**).

Neste artigo, eu vou usar a primeira opção, na qual a partir do modelo de banco de dados e tabela já existente, irei realizar o mapeamento, gerando as entidades através dos arquivos de configuração XML (*poderíamos também usar anotações*) do NHibernate.

Roteiro de utilização:

- Efetuar o download da última versão do NHibernate;
- Criação do banco de dados e da tabela no SQL Server 2005;
- Criação do projeto ASP .NET no Visual Web Developer 2010 Express;
- Definição da referência a API do NHibernate;
- Criação da classe que irá definir o domínio da aplicação;
- Criação e definição do arquivo de mapeamento que irá permitir mapear o objeto para a tabela do banco de dados;
- Utilização da API do NHibernate para mapear os objetos e persistir os dados na tabela.

Definindo o modelo de dados: Banco de dados e tabela

Neste exemplo, teremos um banco de dados criado no SQL Server 2005 de nome **Macoratti.mdf** e um única tabela chamada **Usuarios** com a seguinte estrutura:

[CURSOS ONLINE](#)



Gerência de Projetos com Scrum

Neste curso você irá aprender os valores do Manifesto Ágil. Também será possível conceitos, processos e vantagens na utilização do framework Scrum em projetos de qu...



Facebook Marketing

Desenvolva uma visão estratégica de marketing em rede social, apresentando os principais conceitos e as aplicações disponíveis para criação, execução e análise de resultados de campanhas que utilizem meios digitais.



Introdução ao desenvolvimento para iOS

Apresentação das ferramentas de desenvolvimento para aplicativos para iOS (Xcode, Interface Simulator), como conseguí-las e como...

Encontre-nos no Facebook

iMasters
Curtir

9,532 pessoas curtiram **iMasters**.

Letícia Gabriel Ricardo Fran

Michael Adriano William Sérgio

Plug-in social do Facebook

The screenshot shows the SSMS interface. In the Server Explorer, there's a connection to 'mac-pc\sqlexpress.Macoratti'. Under 'Tables', 'Usuarios' is selected. The 'dbo.Usuarios' properties window is open, showing the column definitions:

Column Name	Data Type	Allow Nulls
usuarioid	int	<input checked="" type="checkbox"/>
nome	nvarchar(50)	<input checked="" type="checkbox"/>
login	nvarchar(50)	<input checked="" type="checkbox"/>
senha	nvarchar(50)	<input checked="" type="checkbox"/>
perfil	nvarchar(50)	<input checked="" type="checkbox"/>
email	nvarchar(80)	<input checked="" type="checkbox"/>

Below the table definition, the 'Column Properties' section shows the 'Identity Specification' settings:

- Full-text Specification: No
- Has Non-SQL Server Subscript No
- Identity Specification: Yes
- (Is Identity): Yes
- Identity Increment: 1
- Identity Seed: 1

A nossa aplicação **ASP .NET Web Forms** irá gerenciar os usuários, permitindo inclusão, alteração e exclusão de usuários da tabela **Usuarios**.

Note que definimos uma chave primária - **usuarioid** - do tipo **identity**; dessa forma, esse campo passa a ser gerenciado pelo Banco de dados.

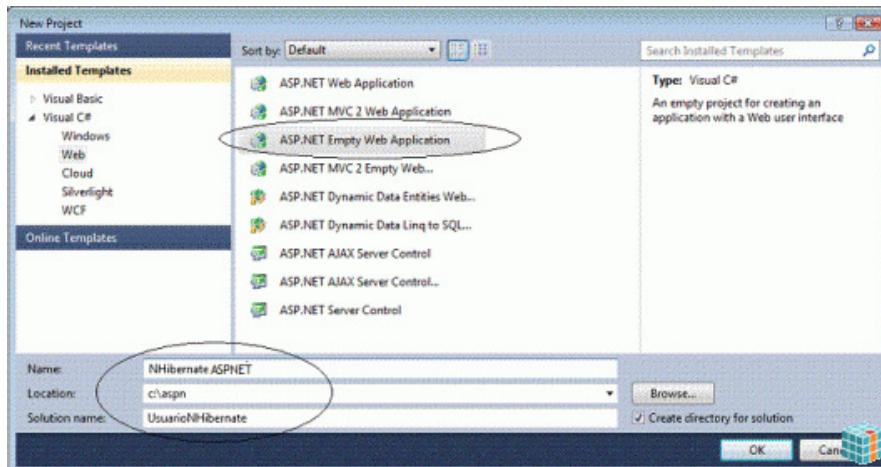
Criação do projeto no Visual Web Developer e referência ao NHibernate 2.1.2

Abra o Visual Web Developer 2010 Express Edition e selecione a opção **New Project**.

A seguir, selecione **Visual C# -> Web**, escolha o template **ASP .NET Empty Web**

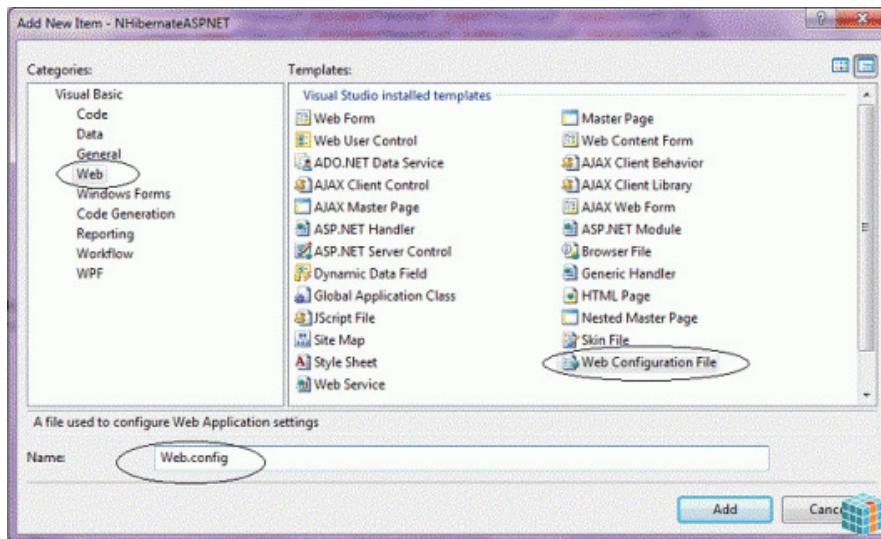
Application e informe o local e o nome do projeto que, no nosso caso, é

NHibernateASPNET.

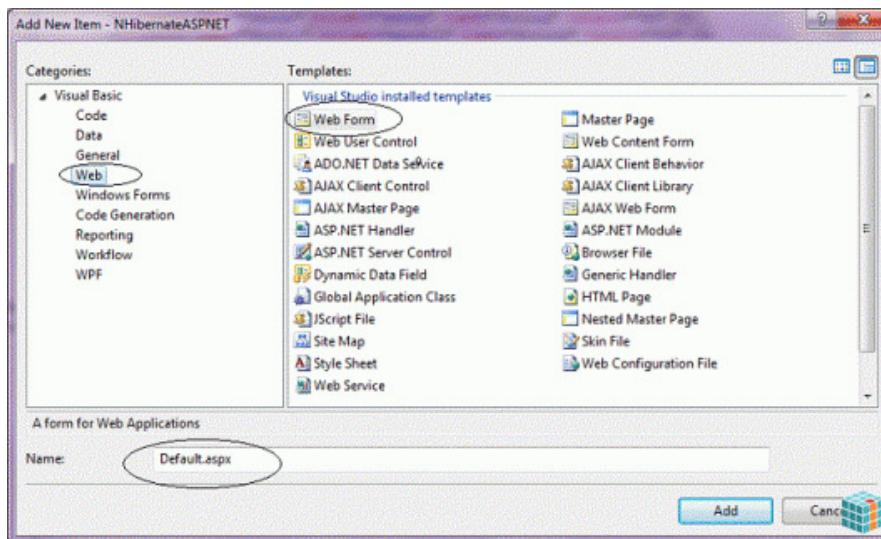


Vamos incluir dois arquivos ao projeto, visto que o mesmo está vazio.

Nome Project selecione Add New Item e, a seguir, selecione o template Web Configuration File e defina as opções conforme a figura abaixo, para incluir o arquivo Web.Config no projeto clicando no botão Add.



Repita o procedimento acima selecionando o template Web Form e informando o nome Default.aspx para incluir o um novo Web Form no projeto.

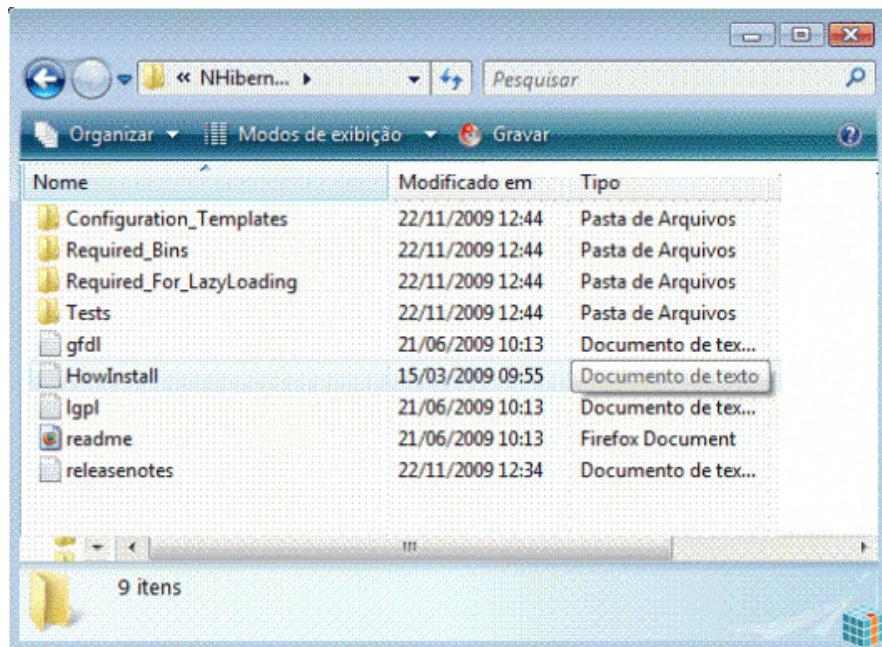


Iremos usar estes arquivos mais à frente.

Agora, baixe o NHibernate 2.1.2 do site: <http://sourceforge.net/projects/nhibernate/files/> e descompacte o arquivo em um local apropriado na sua máquina local.

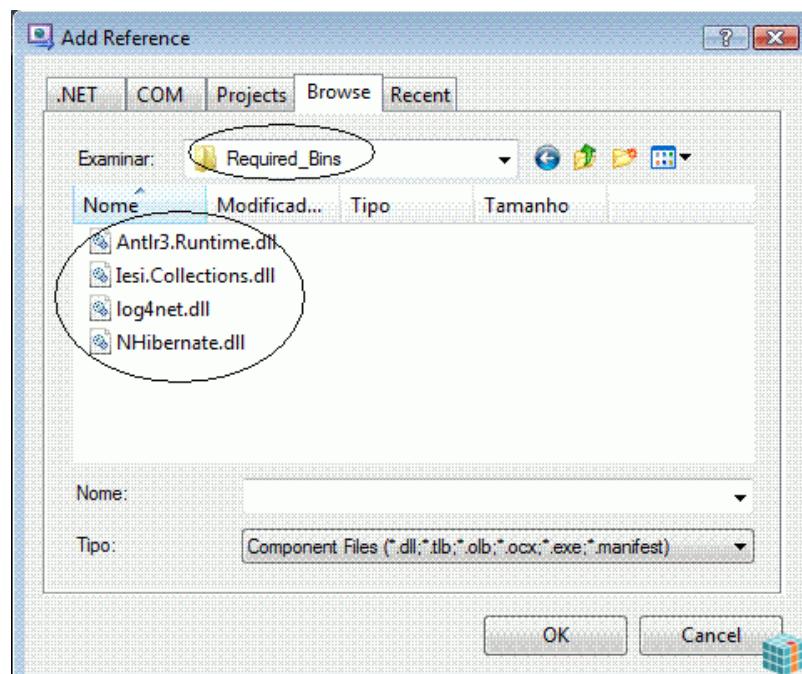
File/Folder Name	Platform	Size	Date	Downloads	Notes/Subscribe
NHibernate-3.0.0.Beta2-arc.zip		6.9 MB	2010-10-31	135	
NHibernate-3.0.0.Beta2-bin.zip		7.9 MB	2010-10-31	789	
All Files					
NHibernate		167.2 MB	2010-10-31	492,464	
3.0.0.Beta2		14.8 MB	2010-10-31	924	
2.1.2.GA		51.2 MB	2009-11-25	165,421	
2.1.0.GA		53.2 MB	2009-07-26	87,256	
2.0.1.GA		7.8 MB	2008-09-29	109,701	
1.2.1.GA		40.3 MB	2007-11-26	129,162	

Eu estou descompactando o arquivo na pasta **C:\Program Files\NHibernate-2.1.2.GA-bin**; abaixo vemos os arquivos dessa pasta:

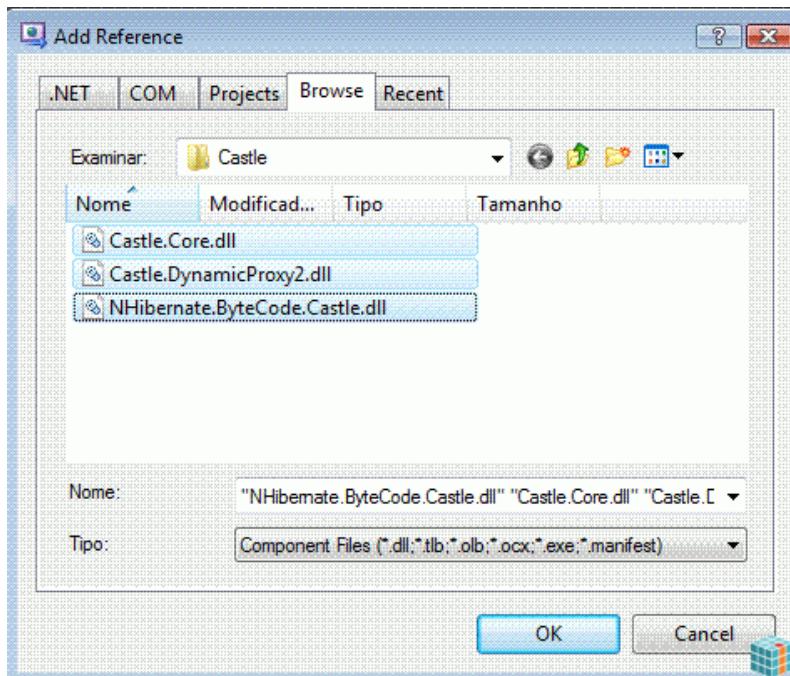


Em seguida, no menu **Project > Add Reference** e na janela **Add Reference**, selecione a guia **Browse** e localize a pasta onde você descompactou os arquivos do NHibernate.

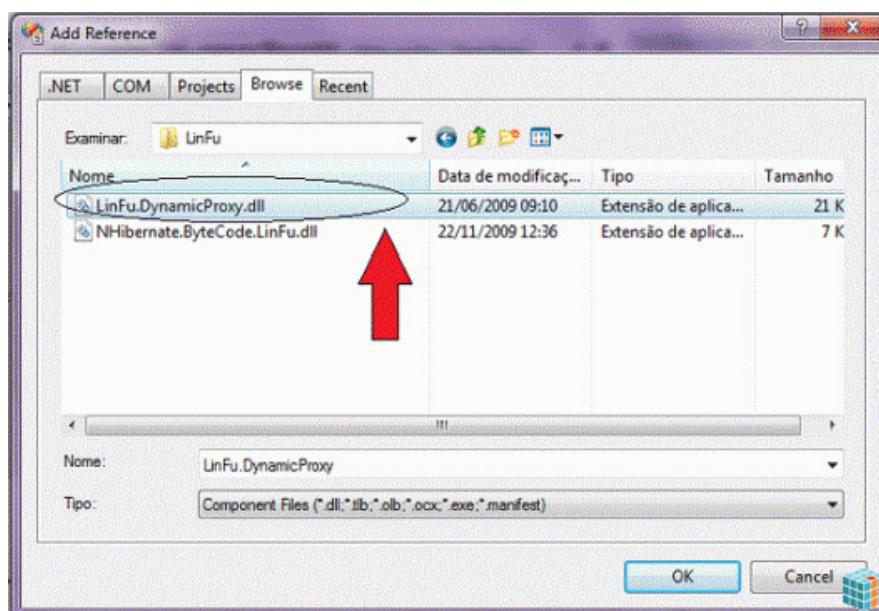
Abra a pasta **Required_Bins** e selecione todos os arquivos e clique em OK.



Repita o processo acima, incluindo uma referência para a pasta **Required_For_LazyLoading** > **Castle**, selecione todos os arquivos e clique em OK.



Após isso, precisamos fazer mais uma referência no projeto; trata-se do assembly NHibernate.ByteCode.LinFu.dll que está na pasta Required_For_LazyLoading, subpasta - LinFu:



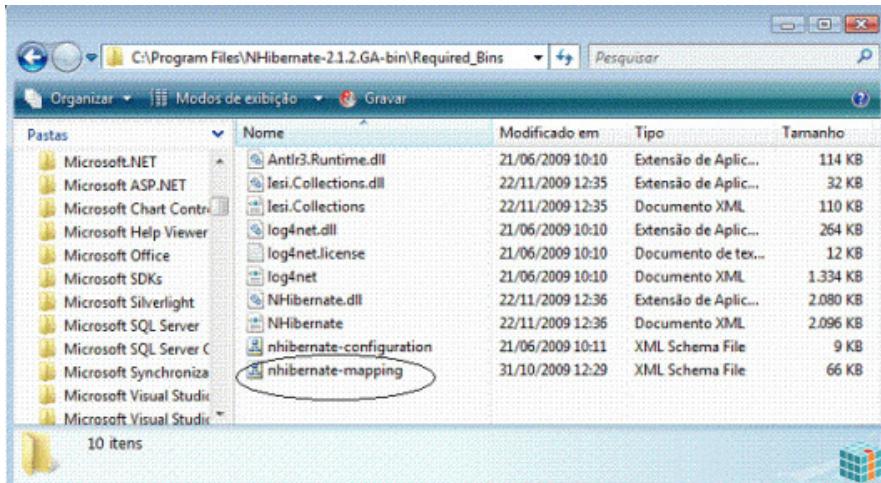
Se não fizermos isso, teremos o erro: *The ProxyFactoryFactory was not configured. Initialize 'proxyfactory.factory_class' property of the session-factory configuration section with one of the available NHibernate.ByteCode providers.*

Assim, já temos todas as principais referências no projeto e com elas garantimos o funcionamento do NHibernate na maioria dos projetos.

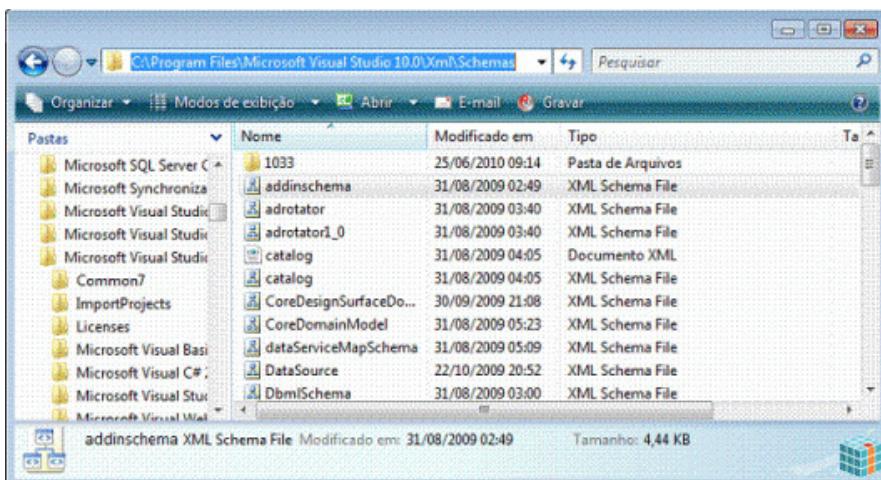
A próxima tarefa é definir os arquivos de mapeamento do NHibernate, a forma mais comum de fazer isso é usar arquivos XML.

Podemos obter o recurso do IntelliSense no Visual ao realizar a definição dos arquivos XML copiando o arquivo nhibernate-mapping.xsd da pasta **Required_Bins** para a pasta **c:\Arquivos de Programas\Microsoft\Visual Studio 10.0\Xml\Schemas**

01. C:\Program Files\NHibernate-2.1.2.GA\bin\Required_Bins



02. C:\Program Files\Microsoft Visual Studio 10.0\Xml\Schemas

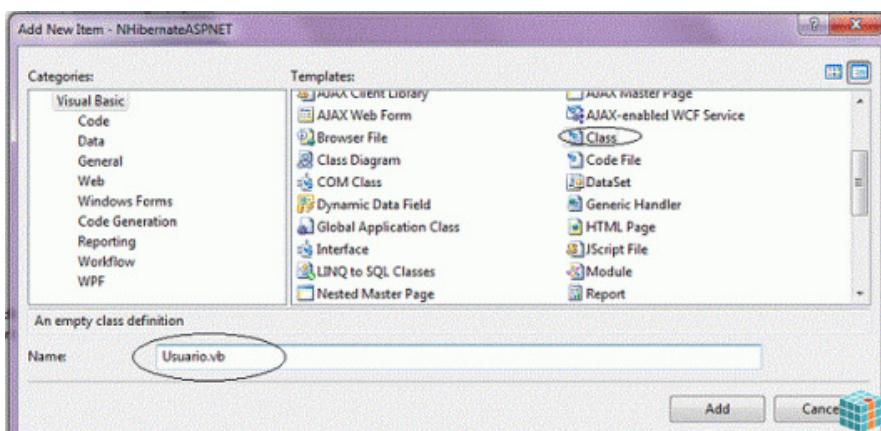


Após isso, já podemos criar os arquivos de mapeamento, tendo o recurso IntelliSense à disposição, o que nos ajuda muito.

Criando a classe de negócio

Como estamos partindo do modelo de dados já criado, agora vamos definir a classe de negócio que representa a tabela Usuarios.

No menu Project, selecione Add Class, selecione o template Class, informe o nome Usuario.cs e clique em Add.



A seguir, vamos definir o seguinte código no arquivo Usuario.vb:

```
Public Class Usuario

Private _usuarioid As Int32
Private _nome As String
```

```
Private _login As String
Private _senha As String
Private _perfil As String
Private _email As String

Public Sub New()
End Sub

Public Overridable Property Usuarioid() As Int32
Get
Return _usuarioid
End Get
Set(ByVal value As Int32)
Usuarioid = value
End Set
End Property

Public Overridable Property Nome() As String
Get
Return _nome
End Get
Set(ByVal value As String)
Nome = value
End Set
End Property

Public Overridable Property Login() As String
Get
Return _login
End Get
Set(ByVal value As String)
Login = value
End Set
End Property

Public Overridable Property Senha() As String
Get
Return _senha
End Get
Set(ByVal value As String)
Senha = value
End Set
End Property

Public Overridable Property Perfil() As String
Get
Return _perfil
End Get
Set(ByVal value As String)
Perfil = value
End Set
End Property

Public Overridable Property Email() As String
Get
Return _email
End Get
Set(ByVal value As String)
Email = value
End Set
End Property
End Class
```

Nesse código, estamos declarando as propriedades na classe Usuário, na qual cada propriedade será mapeada para o respectivo campo da tabela **Usuarios**.

No VB.NET, indicamos que um método é passível de sobreposição usando a palavra-chave **Overridable** na classe pai (classe base), e a seguir, na classe filha, declaramos novamente o método com a palavra-chave

Overrides. Assim, temos que:

Overridable - declara que o método pode ser sobreposto nas classes que herdarem da classe base

Overrides - indica que o método da classe filha irá sobrepor o método da classe pai.

Para que um método da classe pai não possa ser sobreposto, usamos o modificador - **NotOverridable**. Já para definir que um método seja obrigatoriamente sobreposto em uma classe filha, usamos a palavra-chave - **MustOverride**.

Com a classe **Usuario** definida, podemos passar para a etapa de configuração e mapeamento.

Definindo os arquivos de mapeamento e configuração

Creio que a parte de configuração é a que apresenta maiores problemas para os usuários iniciantes, portanto preste atenção em cada detalhe usado na configuração do NHibernate.

Temos que definir dois tipos de configuração:

- O arquivo de configuração do NHibernate: Onde indicamos o tipo de banco de dados, drivers, conectores e a string de conexão usada;
- O arquivo de mapeamento do NHibernate: Onde definimos a tabela e o mapeamento entre a(s) classe(s) de negócio e a(s) tabela(s).

Definindo o arquivo de configuração

A primeira coisa que vamos fazer é definir o arquivo de configuração para que o NHibernate possa funcionar corretamente.

Podemos definir essas configurações no arquivo **Web.Config** ou em um arquivo a parte chamado de **hibernate.cfg.xml**.

Como já temos o arquivo **Web.Config**, vamos usá-lo para criar essas configurações.

Abra o arquivo **Web.Config** e inclua o código que está em azul, conforme mostrado a seguir:

```
<?xml version="1.0"?>
<!--
For more information on how to configure your ASP.NET application, please
http://go.microsoft.com/fwlink/?LinkId=169433
-->
<configuration>
    <system.web>
        <compilation debug="true" targetFramework="4.0" />
    </system.web>

    <!--tag definindo seção de configuração do NHibernate -->
    <configSections>
        <section name="hibernate-configuration" type="NHibernate.Cfg.Configuration" />
    </configSections>

    <!-- Configuração do NHibernate-->
    <hibernate-configuration xmlns="urn:nhibernate-configuration-2.2" >
        <session-factory>
            <property name="dialect">
                NHibernate.Dialect.MsSql2005Dialect
            </property>
            <property name="connection.provider">

```

```

NHibernate.Connection.DriverConnectionProvider
</property>
<property name="connection.driver_class">
    NHibernate.Driver.SqlClientDriver
</property>
<property name="connection.connection_string">
    Server=.\SQLEXPRESS;
    Database=Acesso;
    Integrated Security=True;
</property>
<property name="proxyfactory.factory_class">

NHibernate.ByteCode.LinFu.ProxyFactoryFactory, NHibernate.ByteCode.LinFu

</property>

</session-factory>
</hibernate-configuration>

</configuration>

```

Arquivo Web.Config

Observe que tivemos primeiro que definir a sessão de configuração, para depois criar a sessão com as configurações do NHibernate.

```

<!-- Configuração do NHibernate-->
<hibernate-configuration xmlns="urn:nhibernate-configuration-2.2" >
    <session-factory>
        <property name="dialect">
            NHibernate.Dialect.MsSql2005Dialect
        </property>
        <property name="connection.provider">
            NHibernate.Connection.DriverConnectionProvider
        </property>
        <property name="connection.driver_class">
            NHibernate.Driver.SqlClientDriver
        </property>
        <property name="connection.connection_string">
            Server=.\SQLEXPRESS;
            Database=Acesso;
            Integrated Security=True;
        </property>
        <property name="proxyfactory.factory_class">

            NHibernate.ByteCode.LinFu.ProxyFactoryFactory,
            NHibernate.ByteCode.LinFu

        </property>
    </session-factory>
</hibernate-configuration>

```

Na sessão de configuração do arquivo web.config, definimos os seguintes itens:

- No arquivo acima, definimos o provedor de acesso a base de dados SQL Server 2005: **NHibernate.Dialect.MsSql2005Dialect**;
- O provedor usado: **NHibernate.Connection.DriverConnectionProvider**;
- O driver apropriado: **NHibernate.Driver.SqlClientDriver**;
- Definimos também a string de conexão usada:
Server=.\SQLEXPRESS;
Database=Acesso;

Integrated Security=True;

- A proxyFactory: NHibernate.ByteCode.LinFu.ProxyFactoryFactory,
NHibernate.ByteCode.LinFu

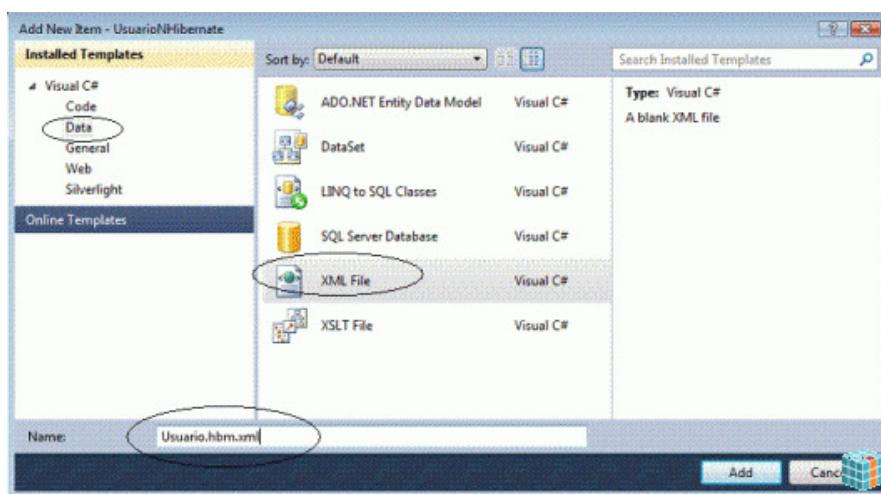
Definindo o arquivo de mapeamento

Como já temos a classe de negócio criada, vamos definir o arquivo de mapeamento baseado nesta classe. O mapeamento é o coração do NHibernate e, se não for feito corretamente, vai lhe dar muita dor de cabeça.

No arquivo de mapeamento, simplesmente especificamos qual tabela no banco de dados estamos relacionando com a classe de negócio. Podemos definir esse mapeamento em um arquivo XML separado ou usando atributos nas classes, propriedades e membros.

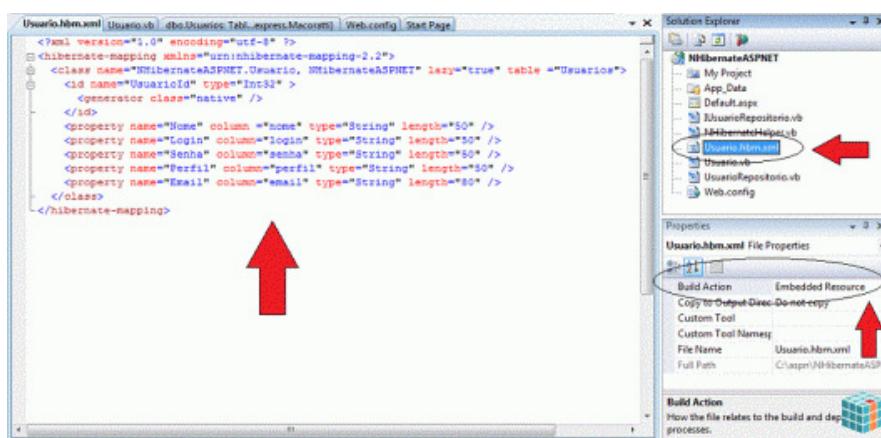
Vamos criar um arquivo XML no projeto. No menu **Project**, selecione **Add New Item**.

A seguir, selecione o item **Data** e o template **XML File** e informe o nome **Usuario.hbm.xml** e clique em **Add**.



A seguir, devemos realizar duas tarefas com o arquivo **Usuario.hbm.xml**:

01. Alterar a sua propriedade Build Action para: Embedded Resource.



02. Definir a tabela e o mapeamento entre a classe de negócio e as tabelas usadas:

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">
<class name="NHibernateASPNET.Usuario, NHibernateASPNET" lazy="true" table="Usuarios">
<id name="UsuarioId" type="Int32" >
<generator class="native" />
</id>
<property name="Nome" column="nome" type="String" length="50" />
<property name="Login" column="login" type="String" length="50" />
<property name="Senha" column="senha" type="String" length="50" />
<property name="Perfil" column="perfil" type="String" length="50" />
<property name="Email" column="email" type="String" length="80" />
</class>
</hibernate-mapping>
```

```
</class>
</hibernate-mapping>
```

Como nosso exemplo usa somente uma tabela (de propósito), fica fácil definir o mapeamento conforme o código acima.

Agora vamos entender o significado deste código deste arquivo:

- Este é um arquivo de mapeamento usado pelo NHibernate que mapeia as classes para a tabela do banco de dados e possui a terminação nomeclasse.hbm.xml. (no exemplo o nome da classe é Usuario)
- Para cada classe deve haver um arquivo de mapeamento
- Os elementos para o mapeamento objeto relacional encontram-se entre as tags `<hibernate-mapping>` e `<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2">`

Essa linha indica o início do mapeamento (hibernate-mapping) feito NHibernate e deve ser usada na versão 2.1.2 (rn:nhibernate-mapping-2.2).

Atenção: se você estiver usando uma versão anterior do NHibernate, **não** use esta sintaxe:

```
<class name="NHibernateASPNET.Usuario, NHibernateASPNET" table="Usuarios">
```

- Aqui definimos o nome da classe na tag class :`<class name="NHibernateASPNET.Usuario, NHibernateASPNET"` (note que incluímos o nome do Assembly)
- O atributo name deve conter o namespace e o nome do Assembly
- O atributo table deve indicar o nome da tabela: `table="Usuarios"` (se o nome da tabela for igual ao da classe não precisa ser declarado)

Vejamos agora o mapeamento da chave primária:

```
<id name="UsuarioId" column="usuarioId" type="Int32">
    <generator class="native" />
</id>
```

- Nesta definição, temos o mapeamento da chave primária na qual a propriedade UsuarioId é mapeada para a coluna da tabela usuarioId.
- A tag id identifica a chave primária, e o atributo name="UsuarioId" informa o nome do atributo da classe .NET que se refere à chave primária da tabela.
- O atributo column informa ao NHibernate que coluna na tabela é a chave primária, no caso usuarioId.
- O atributo generator informa qual a estratégia para geração da chave primária. Para o nosso exemplo, usamos a estratégia native que significa que o NHibernate irá usar a estratégia que melhor se adequar ao banco de dados podendo ser: identity, sequence ou hilo dependendo das capacidades do banco de dados.

Outros valores possíveis são:

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. **uuid.hex:** usa uma combinação do IP com um timestamp para gerar um identificador único;
6. **guid:** usa o novo system.guid como identificador.

Após isso, temos os mapeamentos das propriedades persistentes através da tag property.

```
<property name="Nome" column ="nome" type="String" length="50" />
<property name="Login" column="login" type="String" length="50" />
<property name="Senha" column="senha" type="String" length="50" />
<property name="Perfil" column="perfil" type="String" length="50" />
<property name="Email" column="email" type="String" length="80" />
```

As tags property indicam propriedades simples dos objetos, onde os nomes das propriedades das classes são definidos pelo atributo name, o tipo pelo atributo type e a coluna da tabela a que se refere pelo atributo column.

No nosso exemplo, temos o mapeamento das propriedades Nome, Login, Senha, Perfil e Email para as colunas nome, login, senha, perfil e email do mesmo tipo.

Se o atributo column não aparecer no mapeamento da propriedade, o NHibernate considera que a coluna na tabela do banco de dados a que se referencia possui o mesmo nome que o definido pelo atributo name.

Dessa forma, definimos os arquivos de configuração e mapeamento e já temos a classe de negócio definida e o banco de dados criado.

Só falta definir a interface no projeto ASP .NET na qual iremos usar a página Default.aspx para realizar a manutenção dos dados dos usuários na tabela Usuarios e definir uma sessão NHibernate para realizar as operações desejadas persistindo-as no banco de dados.

Aguarde os próximos artigos!

[Tweet](#) { 18 }

0

[Like](#)

4

[Send](#)



José Carlos Macoratti

é referência em Visual Basic no Brasil e autor dos livros "Aprenda Rápido: ASP" e "ASP, ADO e Banco de Dados na Internet". Mantenedor do site macoratti.net.

[Página do autor](#) [Email](#)

Leia os últimos artigos publicados por jose_carlos_macoratti

[VB.NET - Populando o controle TreeView com tabelas e colunas do SQL Server](#)

[VB .NET - Populando o controle TreeView com tabelas e colunas do MS Access](#)

[ASP .NET 4.0 - Usando os recursos do Ajax \(ModalPopup\)](#)

[WPF - DataBinding com Entity Framework 4.1 e Code First - Parte 02](#)

[WPF - DataBinding com Entity Framework 4.1 e Code First - Parte 01](#)

3 COMENTÁRIOS

[COMENTE TAMBÉM](#)



Roger

Nossa... poderia ter feito uma video-aula...

Há 1 ano · [Responder](#)



Gabriel

Entity Framework e NHibernate tem a mesma finalidade, certo? Qual dos dois é melhor? Ou depende da aplicação/finalidade?

Fica a sugestão de escrever um artigo comparando os dois frameworks.

Gabriel

www.dotdicas.com

Há 1 ano · [Responder](#)

**Rodolfo Wagner**

Ótimo artigo. parabéns.

o maior problema q encontrei é colocar o namespace e o nome do assembly.

pois minha solution é Fabrica e o projeto web é FabricaObjeto.

como iria ficar?

]]'s

Há 8 meses · [Responder](#)**QUAL A SUA OPINIÃO?**

Escreva seu comentário aqui...

o jeito simples e gratuito de fazer web.
Faça o download e comece agora mesmo!

Microsoft®
WebMatrix

PARCEIROS