.NET + SQL Server

ASP .NET: Usando o NHibernate em uma aplicação Web - Parte 02

Segunda-feira, 20/12/2010 às 11h15, por José Carlos Macoratti

Na primeira parte deste artigo, foram realizadas as seguintes tarefas:

- Definição do banco de dados Macoratti.mdf e da tabela Usuarios;
- Criação do projeto no Visual Web Developer 2010 Express;
- Referência ao NHibernate no Projeto ASP NET;
- Criação da classe de negócio Usuario;
- Definição dos arquivos de mapeamento e configuração do NHibernate.

Já temos portanto tudo pronto para usar os recursos do NHibernate em nossa aplicação ASP .NET e realizar as operações CRUD relacionadas com as manutenção das informações da tabela Usuario. Nossa próxima tarefa é definir uma sessão NHibernate.

Mas afinal o que vem a ser uma sessão NHibernate?

Pense em uma sessão NHibernate como uma ligação abstrata ou virtual de um conduto ou fio com o banco de dados. Quando se usa o NHibernate, não temos que nos preocupar em criar uma conexão, abrir a conexão, passar a conexão para o objeto Command e criar um DataReader a partir do objeto Command etc.

Com o NHibernate, o tratamento é diferente. Temos que solicitar um objeto Session para o SessionFactory e usar a sessão para realizar as operações CRUD.

O NHibernate trabalha com um mecanismo de sessão, e para criar uma sessão usamos um objeto do tipo ISessionFactory do NHibernate. Além de criar a sessão, devemos tomar o cuidado de ela ser inicializada apenas uma única vez durante a sessão do usuário na aplicação WEB para evitar degradação no desempenho da aplicação.

Para isso, vamos definir uma classe chamada NHibernateHelper no projeto. No menu Project, selecione o item Add Class, selecione o template Class e informe o nome NHibernateHelper.vb. A seguir, vamos definir o seguinte código nesta classe:

```
Imports NHibernate

Imports NHibernate.Cfg

Public Class NHibernateHelper
```

```
    Private Shared _sessionFactory As ISessionFactory

    Private Shared ReadOnly Property SessionFactory() As ISessionFactory

        Get
            'se a sessão não existir cria e retorna uma sessão

            If _sessionFactory Is Nothing Then

                Dim configuration = New Configuration()

                configuration.Configure()

                configuration.AddAssembly(GetType(Usuario).Assembly)

                _sessionFactory = configuration.BuildSessionFactory()

            End If

            Return _sessionFactory

        End Get

    End Property

    Public Shared Function OpenSession() As ISession

        Return SessionFactory.OpenSession()

    End Function

End Class
```

Note que temos a variável estática (shared) e privada _sessionFactory do tipo ISessionFactory. É através desse objeto que iremos criar as sessões NHibernate, e como ele é estático garantimos que a sessão será criada uma única vez.

O método SessionFactory é o método principal da classe, sendo o responsável por criar o ISessionFactory e retorná-la. No código, verificamos se não existe uma sessão criada e, em caso positivo, criamos a sessão.

Na primeira execução do código, teremos que criar o ISessionFactory e para isso temos que recuperar as configurações do arquivo de configuração do NHibernate através do método Configure() da classe Configuration.

Em seguida, registramos os arquivos de mapeamento através do método AddAssembly que usa o nome do Assembly onde estão os arquivos de mapeamento do NHibernate.

O método OpenSession() cria o SessionFactory e retorna um objeto ISession que representa a sessão NHibernate.

Agora vamos criar uma interface no projeto onde iremos definir os métodos que iremos usar na aplicação: no menu Project -> Add Class, informe o nome IUsuarioRepositorio.vb e clique em Add.

A seguir defina o seguinte código nesta interface:

```
Imports System.Collections.Generic

Public Interface IUsuarioRepositorio

Sub Add(ByVal product As Usuario)
```

```
Sub Update(ByVal product As Usuario)

Sub Remove(ByVal product As Usuario)

Function GetById(ByVal id As Guid) As Usuario

Function GetByLogin(ByVal login As String) As Usuario

Function getAllUsuarios()

Function GetByPerfil(ByVal perfil As String) As ICollection(Of Usuario)


End Interface
```

Apenas para lembrar: em uma interface definimos apenas as assinaturas dos métodos que deverão ser implementados por quem for implementar a interface.

Para implementar a interface, vamos criar uma classe concreta: No menu Project -> Add Class, informe o nome UsuarioRepositorio.vb e clique em Add.

A seguir, vamos definir o código que vai implementar cada um dos métodos definidos na interface:

```
Imports NHibernate

Imports NHibernateASPNET

Imports NHibernate.Criterion

Public Class UsuarioRepositorio

    Implements IUsuarioRepositorio

Public Sub Add(ByVal usuario As Usuario) Implements IUsuarioRepositorio.Add

        Using session As ISession = NHibernateHelper.OpenSession()

            Using transaction As ITransaction = session.BeginTransaction()

session.Save(usuario)

transaction.Commit()

            End Using

        End Using

    End Sub


Public Function GetById(ByVal id As System.Guid) As Usuario Implements IUsu

        Using session As ISession = NHibernateHelper.OpenSession()

Return session.[Get](Of Usuario)(id)

        End Using

    End Function


Public Function GetByLogin(ByVal login As String) As Usuario Implements IUs
```

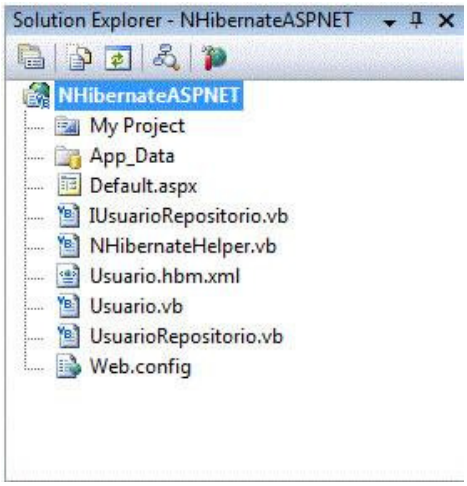
```
        Using session As ISession = NHibernateHelper.OpenSession()

            Dim product As Usuario = session.CreateCriteria(GetType(Usuario

            Return product

        End Using
    End Function



Public Sub Remove(ByVal usuario As Usuario) Implements IUsuarioRepositorio.
       Using session As ISession = NHibernateHelper.OpenSession()

            Using transaction As ITransaction = session.BeginTransaction()

session.Delete(Usuario)

transaction.Commit()

            End Using

        End Using

    End Sub



Public Sub Update(ByVal usuario As Usuario) Implements IUsuarioRepositorio.

        Using session As ISession = NHibernateHelper.OpenSession()

            Using transaction As ITransaction = session.BeginTransaction()

session.Update(Usuario)

transaction.Commit()

            End Using

        End Using

    End Sub

Public Function getAllUsuarios() As Object Implements IUsuarioRepositorio.g

        Using session As ISession = NHibernateHelper.OpenSession()

            Dim usuarios As IList = session.CreateCriteria(GetType(Usuario)

            Return usuarios

        End Using

    End Function

Public Function GetByPerfil(ByVal perfil As String) As System.Collections.G

        Using session As ISession = NHibernateHelper.OpenSession()

            Dim usuarios = session.CreateCriteria(GetType(Usuario)).Add(Res

            Return usuarios
        End Using
    End Function
```

```
End Class
```

Destaques:

- Note que usamos a palavra-chave implements e o nome da interface que vamos implementar;
- Em cada requisição de uma sessão estamos usando o recurso Using, de forma que o recurso usado é liberado ao término da utilização;
- As transações que envolvem atualização no banco de dados usam transações.

Neste momento, a nossa solução deverá ter a seguinte estrutura:



Eu optei criar as classes e os arquivos de configuração em um mesmo projeto e no mesmo local. Fiz isso porque o projeto é bem simples e para tornar mais fácil a visualização da estrutura.

Para projetos maiores, é aconselhável criar pastas distintas para as classes e para os arquivos de configuração no mesmo projeto. Pode-se ainda criar projetos distintos para dividir a solução em camadas o que está mais aderente às boas práticas.

Agora só falta definirmos a interface da aplicação que será a página ASP .NET Default.aspx e usar os recursos que criamos até aqui no projeto.

Referências:

- http://sourceforge.net/projects/nhibernate
- VB.NET - Primeiros passos - Conceitos - V
- NHibernate - Usando o NHibernate 2.1 com o SharpDevelop 3.0 (C#)
- NHibernate - Usando

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

QUAL A SUA OPINIÃO?

Escreva seu comentário aqui...

PARCEIROS