.NET + SQL Server

ASP .NET - Usando o NHibernate em uma aplicação Web - Parte 03

Quarta-feira, 05/01/2011 às 11h00, por José Carlos Macoratti

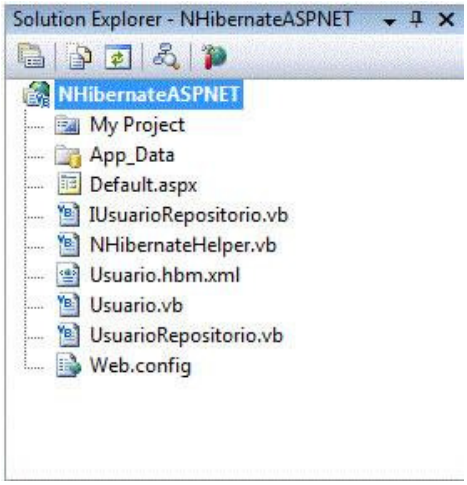
Já seguimos as instruções relacionadas durante a primeira e a segunda etapa deste artigo, relembramos as seguintes tarefas:

- Definição da sessão NHibernate;
- Definição da Interface IUsuarioRepositorio definindo os métodos CRUDs que serão implementados;
- Definição da classe UsuarioRepositorio que implementa a interface definida;

Agora já temos 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.

Só é preciso definirmos a interface da aplicação que será a página ASP .NET Default.aspx e usar os recursos que criamos até aqui no projeto.

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

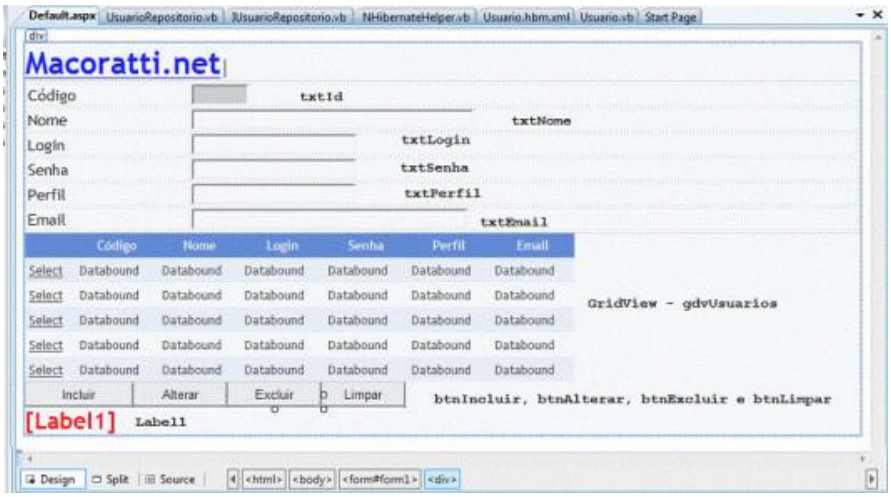
arquivos de configuração no mesmo projeto. Pode-se ainda criar projetos distintos para dividir a solução em camadas o que esta mais aderente as boas práticas.

Então vamos ao trabalho:

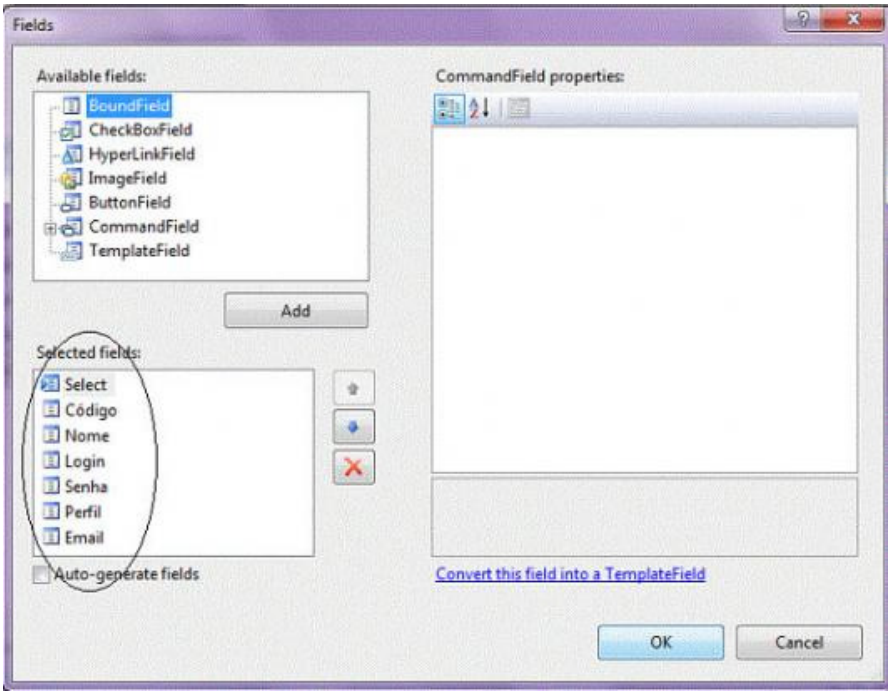
Selecione e abra a página Default.aspx e inclua a partir da toolbox nesta páginas os seguintes controles:

- 7 Labels
- 6 TextBox
- 1 GridView
- 4 Buttons

Conforme o layout da figura abaixo (os IDs dos controles estão definidos abaixo).



É importante definir a propriedade DatKeyNames do controle GridView como sendo igual a usuarioid. Editando o controle GridView na opção Edit Columns deveremos configurar o controle conforme abaixo:



Agora temos que definir o código usado na página Default.aspx no arquivo code-behind Default.aspx.cs. Primeiro definimos o namespace usado na página: Imports NHibernate. A seguir definimos uma variável que representa a classe UsuarioRepositorio que implementa os métodos CRUD e usa a sessão NHibernate: Dim repositorio As UsuarioRepositorio

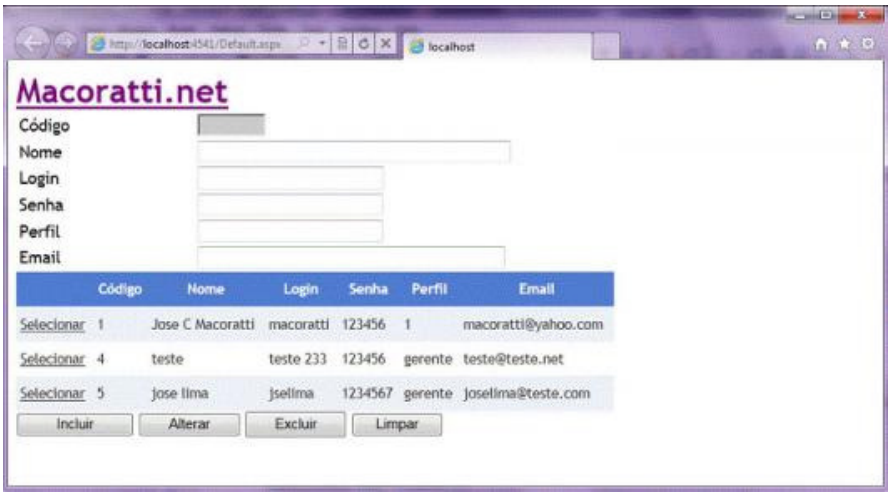
Agora vamos definir o código em cada evento da página conforme a seguir: no evento Load da página temos o código abaixo que chama a rotina carregaUsuarios

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs) Handl

        If Not IsPostBack Then

            carregaUsuarios()

        End If

    End Sub
```

A rotina carregaUsuarios() possui o seguinte código:

```
  Private Sub carregaUsuarios()

        repositorio = New UsuarioRepositorio()

        Dim resultado = repositorio.getAllUsuarios()

        gdvUsuarios.DataSource = resultado

        gdvUsuarios.DataBind()

    End Sub
```

Neste código criamos uma instância da classe UsuarioRepositorio e usamos o método getAllUsuarios() para obter os usuários e exibir no GridView. Ao ser executada o projeto teremos a exibição da página abaixo:

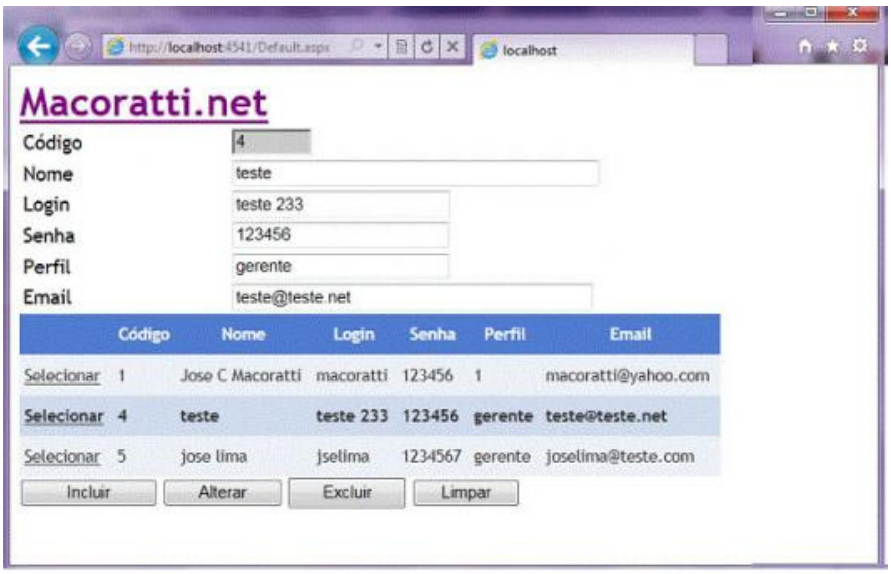


No evento SelectedIndexChanged temos o código que verifica se uma linha do grid foi selecionada e em caso positivo exibe os dados nos controles TextBox da página:

```
Protected Sub gdvUsuarios_SelectedIndexChanged(ByVal sender As Object, ByVa

        If gdvUsuarios.SelectedDataKey IsNot Nothing Then

            Label1.Text = String.empty

            txtId.Text = gdvUsuarios.SelectedDataKey.Value.ToString()

            txtNome.Text = gdvUsuarios.SelectedRow.Cells(2).Text

            txtLogin.Text = gdvUsuarios.SelectedRow.Cells(3).Text

            txtSenha.Text = gdvUsuarios.SelectedRow.Cells(4).Text

            txtPerfil.Text = gdvUsuarios.SelectedRow.Cells(5).Text
```

```
            txtEmail.Text = gdvUsuarios.SelectedRow.Cells(6).Text

        End If

    End Sub
```

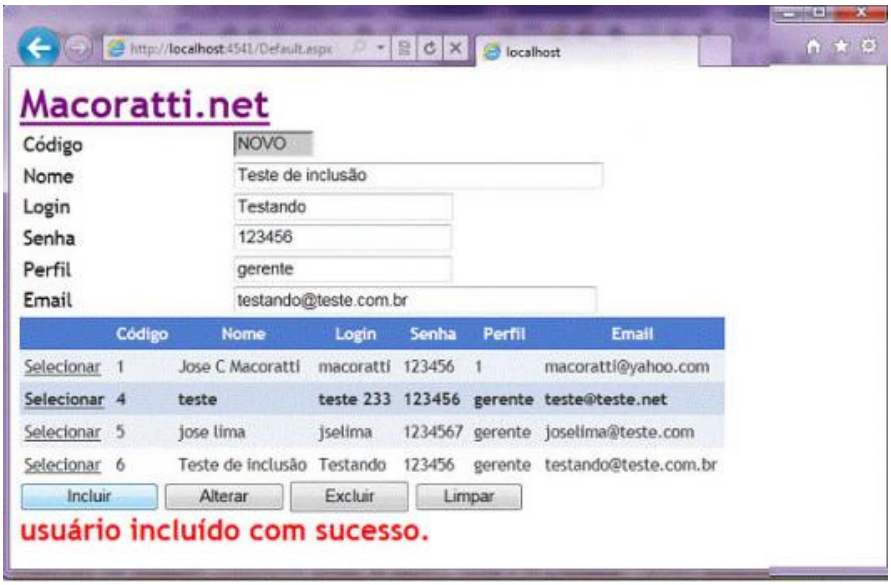
O resultado pode ser visto na figura abaixo:



O código do evento Click do botão Incluir é mostrado a seguir. No código é criado uma instância de usuario e atribuído os valores informados na página web e em seguida usamos o método Add para incluir o usuário.

```
Protected Sub btnIncluir_Click(ByVal sender As Object, ByVal e As EventArgs

        Dim usuario As New Usuario()

        usuario.Nome = txtNome.Text

        usuario.Login = txtLogin.Text

        usuario.Senha = txtSenha.Text

        usuario.Perfil = txtPerfil.Text

        usuario.Email = txtEmail.Text

        repositorio = New UsuarioRepositorio()

        repositorio.Add(usuario)

        Label1.Text = "usuário incluído com sucesso."

        carregaUsuarios()

    End Sub
```

O resultado é exibido a seguir:



No evento Click do botão Alterar temos o código que cria uma nova instância de usuario e usa o método Update para alterar o registro selecionado:

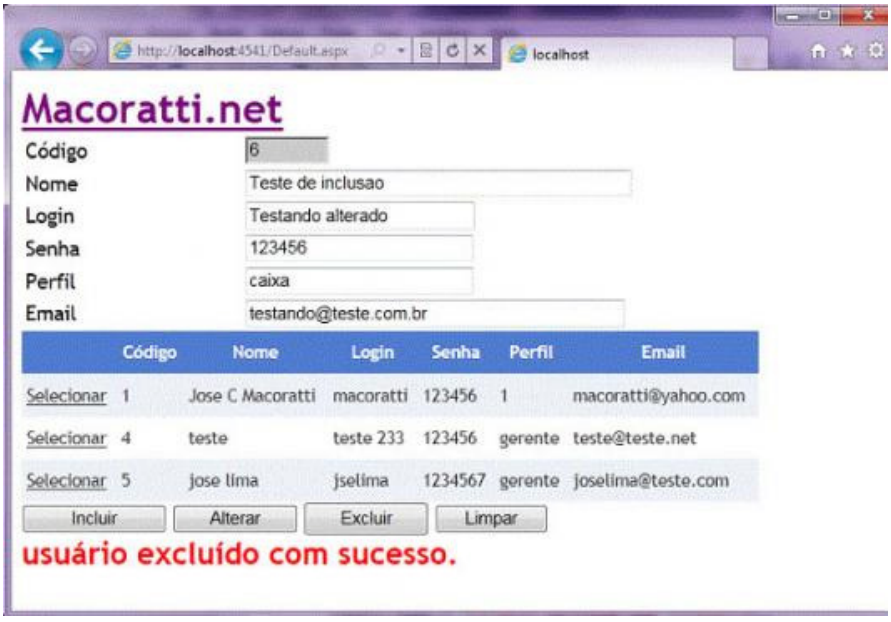
```
Protected Sub btnAlterar_Click(ByVal sender As Object, ByVal e As EventArgs

         Label1.Text = String.empty

        Dim usuario As New Usuario

        usuario.UsuarioId = Convert.ToInt32(txtId.Text)

        usuario.Nome = txtNome.Text

        usuario.Login = txtLogin.Text

        usuario.Senha = txtSenha.Text

        usuario.Perfil = txtPerfil.Text

        usuario.Email = txtEmail.Text

        repositorio = New UsuarioRepositorio()

        repositorio.Update(usuario)

        Label1.Text = "usuário alterado com sucesso."

        carregaUsuarios()

    End Sub
```

Abaixo vamos o resultado da alteração de um registro:

Para excluir um registro temos no evento CLick do botão Excluir o código abaixo que usa o método Remove passando o objeto usuario a ser excluído.

```
Protected Sub btnExcluir_Click(ByVal sender As Object, ByVal e As EventArgs

        Label1.Text = String.empty

        Dim usuario As New Usuario

        usuario.UsuarioId = Convert.ToInt32(txtId.Text)

        repositorio = New UsuarioRepositorio()

        repositorio.Remove(usuario)

        Label1.Text = "usuário excluído com sucesso."

        carregaUsuarios()

    End Sub
```

A exclusão do registro é mostrada na figura a seguir: (Podemos melhorar incluindo uma mensagem solicitando a confirmação da exclusão)



O código do botão Limpar é dado a seguir e apenas limpa o conteúdo dos controles da página. Ele é usado para incluir um novo usuário e para limpar os controles:

```
Protected Sub btnLimpar_Click(ByVal sender As Object, ByVal e As EventArgs)

        txtId.Text="NOVO"

        txtNome.Text=""

        txtLogin.Text=""

        txtSenha.Text=""

        txtPerfil.Text=""

        txtEmail.Text = ""

        txtNome.Focus()

    End Sub
```

Dessa forma encerramos a série de três artigos onde mostramos como usar o NHibernate para realizar as operações CRUD de uma maneira mais consistente usando a sessão NHibernate de uma forma otimizado.

Podemos melhorar ainda mais criando mais recursos em nosso projeto mas isso é assunto para outro artigo. Aguardem!

Enquanto isso, pegue o projeto completo aqui: NHibernateASPNET.zip (Sem as referência ao NHibernate) e baixe os três artigos condensados no formato PDF: NHibernate_ASPNETPDF.zip

Eu sei é apenas NHibernate mas eu gosto...

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

PARCEIROS