

Parte 1

Atendendo milhares de pedidos resolvi mostrar como você pode desenvolver uma aplicação *Windows no VB 2005* do início ao fim. Será uma aplicação que se propõe a gerenciar uma locadora de filmes (DVD e VHS) com recursos básicos.
A APLICAÇÃO É APRESENTADA EM 8 PARTES.

Primeiro quero deixar claro que tudo o que você vai ler neste artigo e nos artigos seguintes não é de minha autoria. Tudo está baseado nos exemplos da MSDN ([MSDnVideo](#)) que você pode acessar no site da Microsoft. Eu apenas refiz os exemplos citados com algumas adaptações.

Para acompanhar esta série de artigos você vai precisar ter os seguintes recursos instalados e funcionando no seu computador:

-  [Visual Developer 2005 Express](#)
-  [Visual Basic 2005 Express](#)
-  [SQL Server 2005 Express](#)

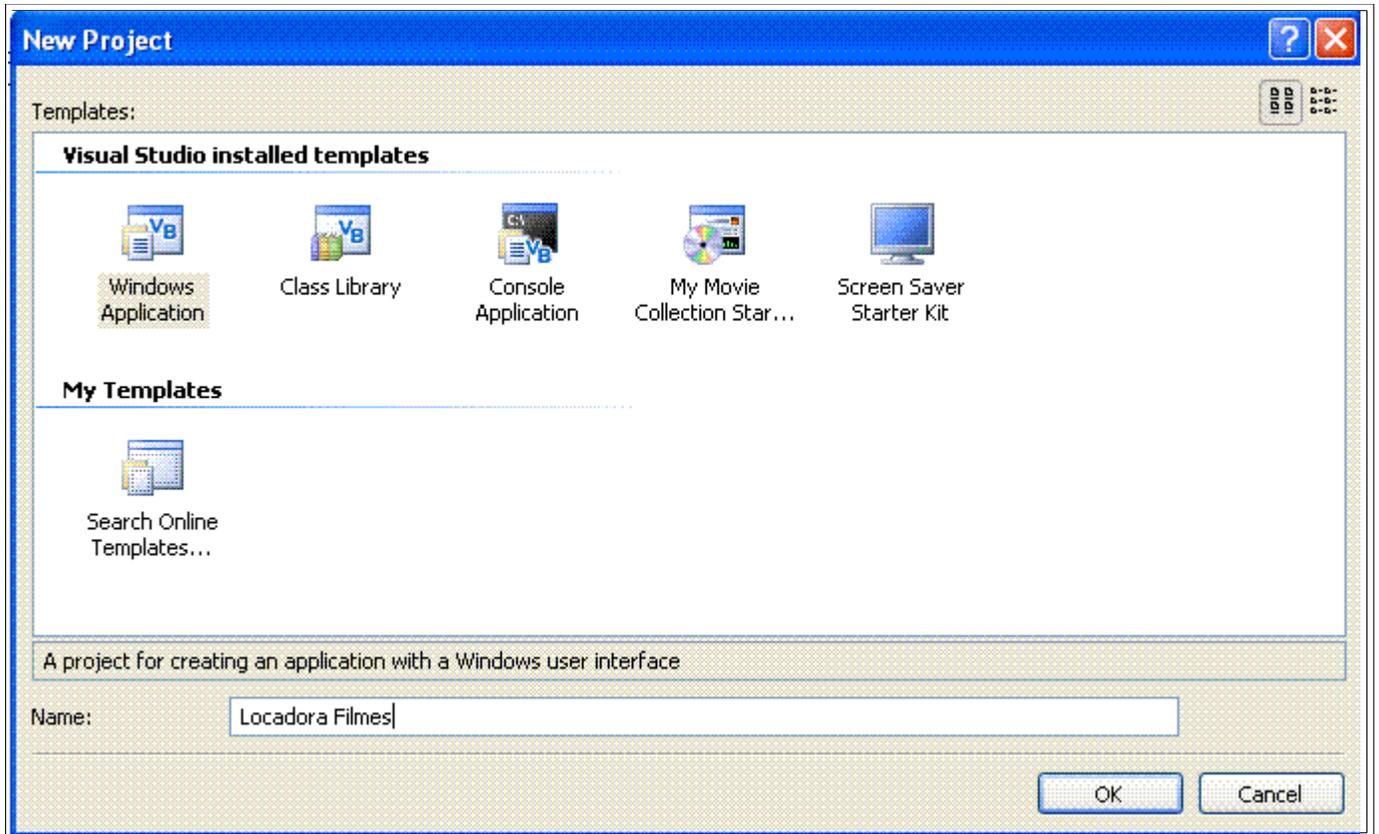
O objetivo principal deste artigo é apresentar os novos recursos de acesso a dados a aplicação será bem simples de forma a tornar possível, em um tempo razoável, a publicação dos artigos. Portanto a modelagem de dados e a definição de requisitos, pontos fundamentais em qualquer aplicação não estarão refletindo o que seria uma aplicação real de produção. A idéia é fornecer os fundamentos básicos para que você possa expandir e melhorar a aplicação e também para que os que estão começando agora possam acompanhar os artigos sem muito trauma.

Nesta primeira parte vou desenvolver os seguintes tópicos:

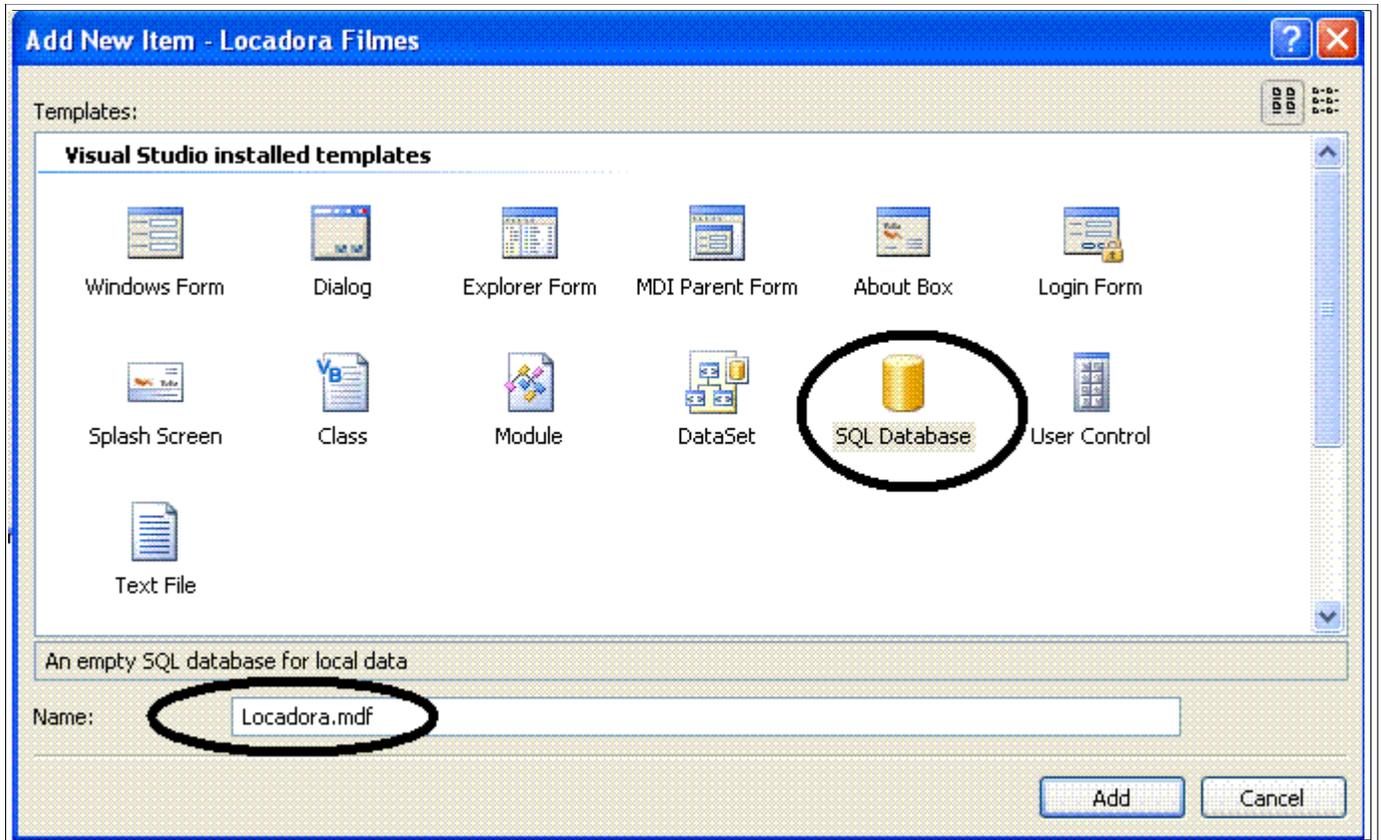
1. Criação do Banco de dados , tabelas , campos , definição de chaves primárias e do relacionamento entre as tabelas do sistema
2. Criação de uma conexão com a fonte de dados e definição dos DataSets

1- Criando o Banco de dados, as tabelas e definindo o relacionamento

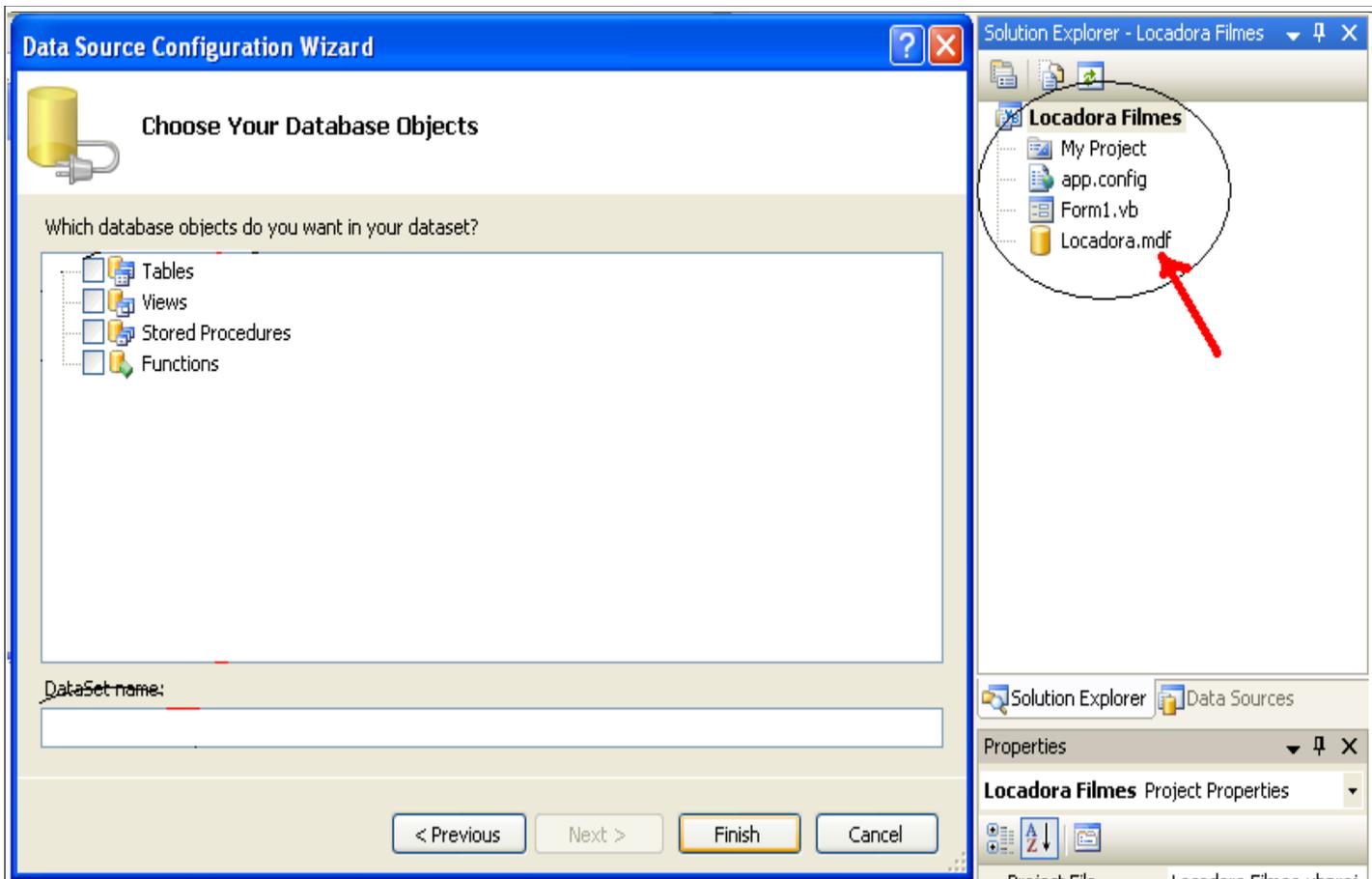
Vamos então arregaçar as mangas e por a 'mão na massa'. Inicie o *Visual Basic 2005 Express Edition* e clique na opção *Create Project*, selecionando a seguir na janela *New Project* o *Template Windows Application*, e, informando o nome do projeto como **Locadora Filmes** (ou algo que melhor lhe apetecer...eu vou usar o nome citado.) conforme figura abaixo:



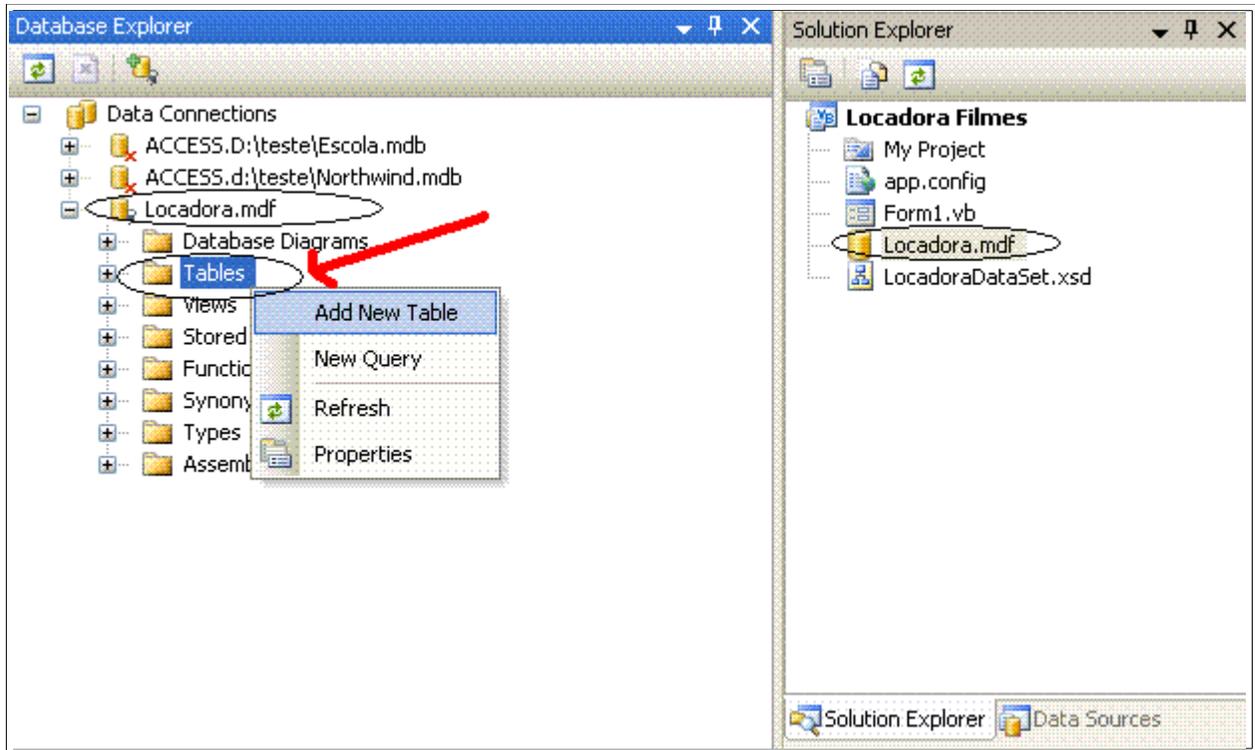
Na janela [Solution Explorer](#) clique com o botão direito do mouse sobre o nome do projeto e selecione do menu suspenso a opção [Add -> New Item](#), e na janela Add New Item , selecione o Template [SQL Database](#) informando o nome de **Locadora.mdf**; a seguir clique no botão [Add](#). (figura abaixo)



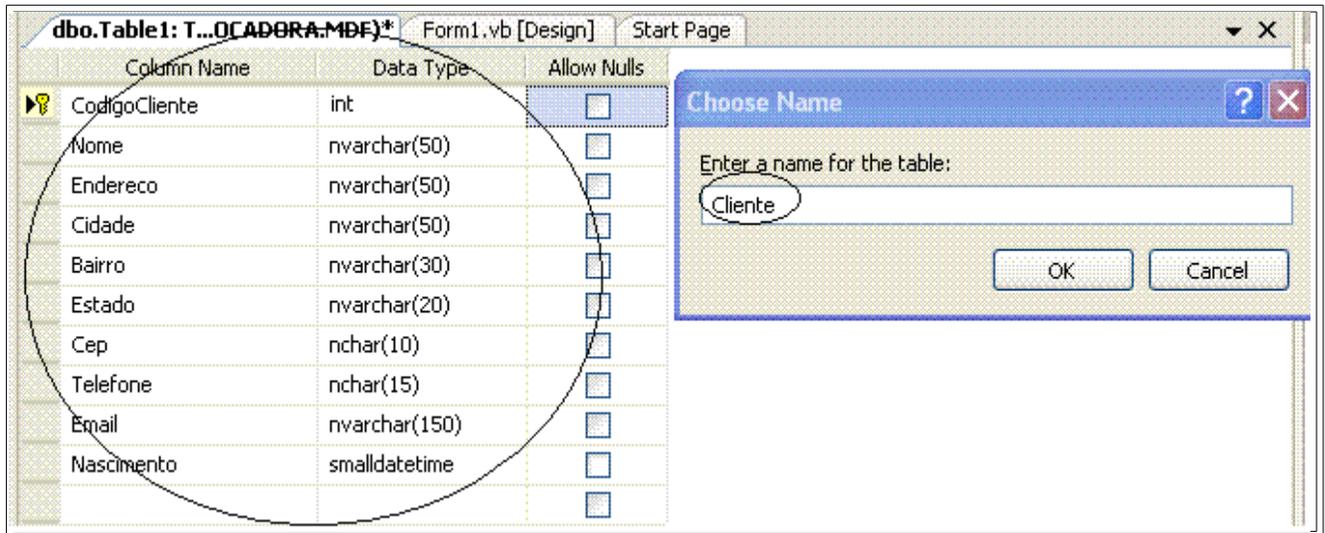
Na janela [Data Source Configuration Wizard](#), como vamos criar as tabelas clique no botão - [Finished](#). Você verá, conforme figura abaixo, o banco de dados **Locadora.mdf** criado e incorporado ao seu projeto. Isto é uma novidade do VB 2005.



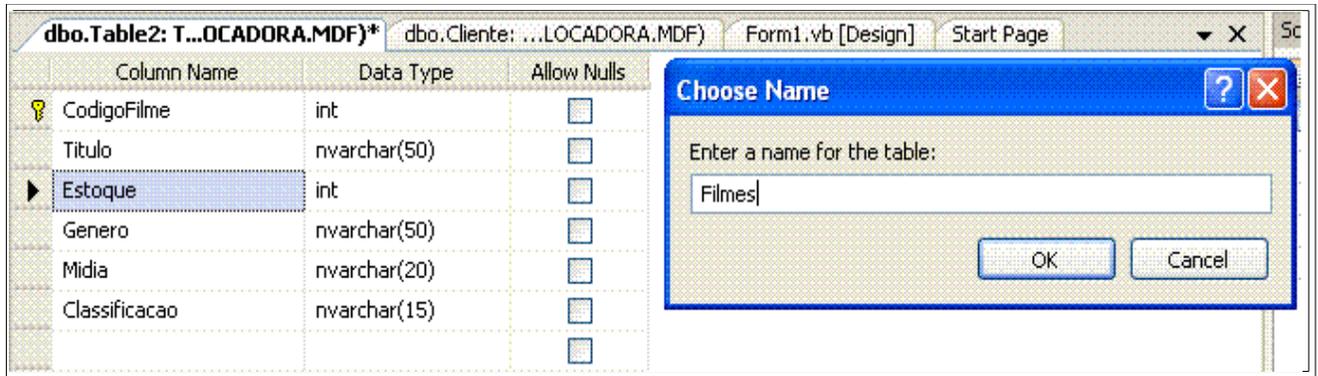
Vamos criar as tabelas do nosso projeto; na janela DataBase Explorer expanda o item Locadora.mdf e clique com o botão direito do mouse sobre o item Tables, selecionando a opção Add New Table.



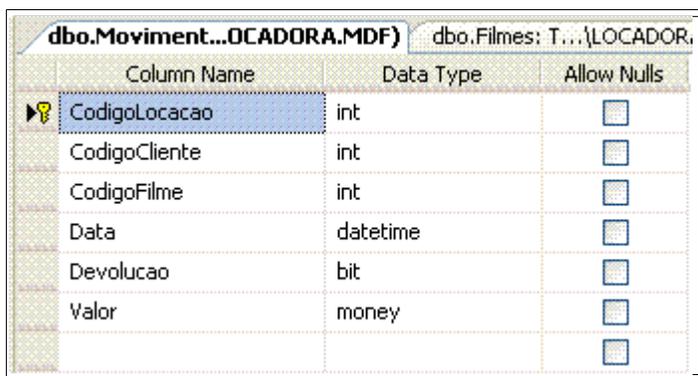
Digite o nome e os tipos de dados de cada campo e a seguir salve a tabela com o nome de Cliente, conforme figura abaixo:



Repita o procedimento adota acima e crie a tabela Filmes conforme figura abaixo:

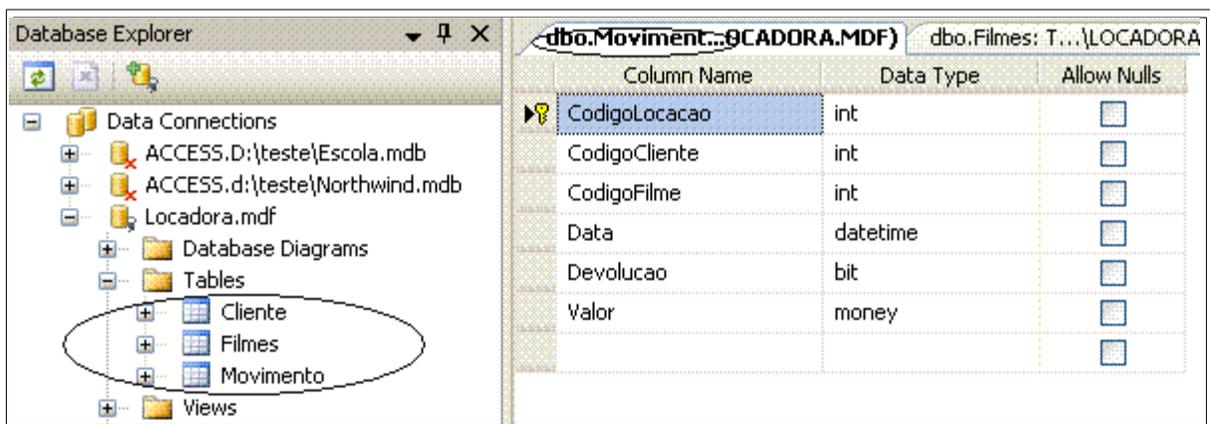


Repita a operação para a tabela Movimento, conforme abaixo:



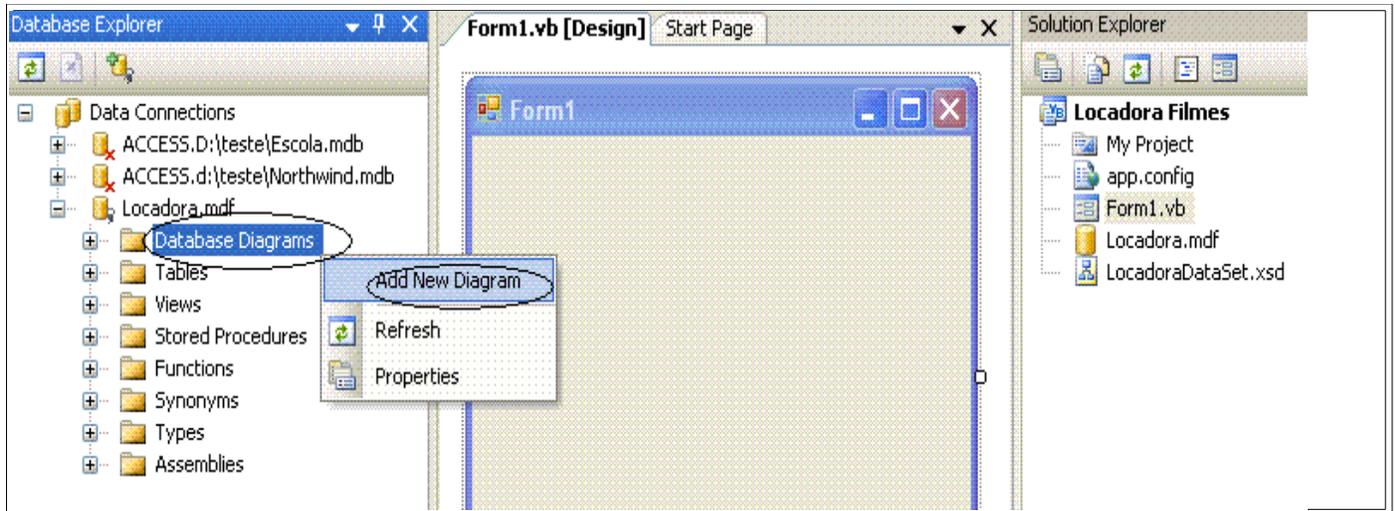
Após terminar você deverá ter 3 tabelas conforme abaixo:

1. **Cliente** - Contém os dados dos clientes da locadora
2. **Filmes** - Contém os dados sobre os filmes oferecidos pela locadora
3. **Movimento** - Contém os dados sobre a locação dos filmes pelos clientes

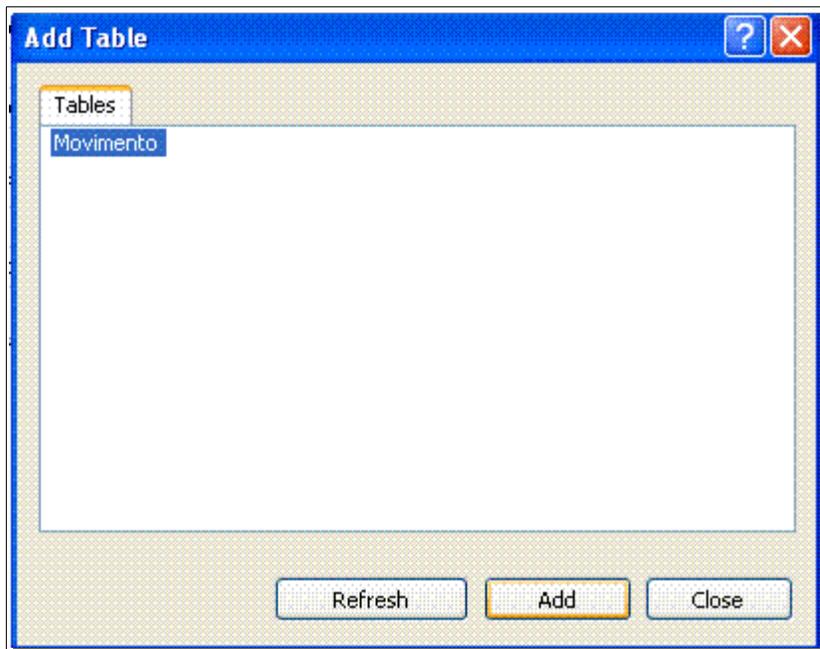


Vamos definir o relacionamento entre as tabelas. Clique agora no item [Database Diagrams](#) e selecione a opção [Add New Diagram](#).

Nota: Informe o nome do Diagrama como **DiagramaLocadora**



Na janela [Add Table](#), selecione cada uma das tabelas e clique no botão [Add](#).



Para definir a chave primária para cada tabela clique com o botão direito do mouse sobre o campo que deseja como chave primária e selecione a opção - [Set As Primary Key](#), aceitando o valor padrão das telas seguintes.

Vamos definir os seguintes relacionamentos entre as tabelas:

1- Cliente (CodigoCliente) --> Movimento (CodigoCliente) - Relacionamento do tipo UM-PARA-MUITOS

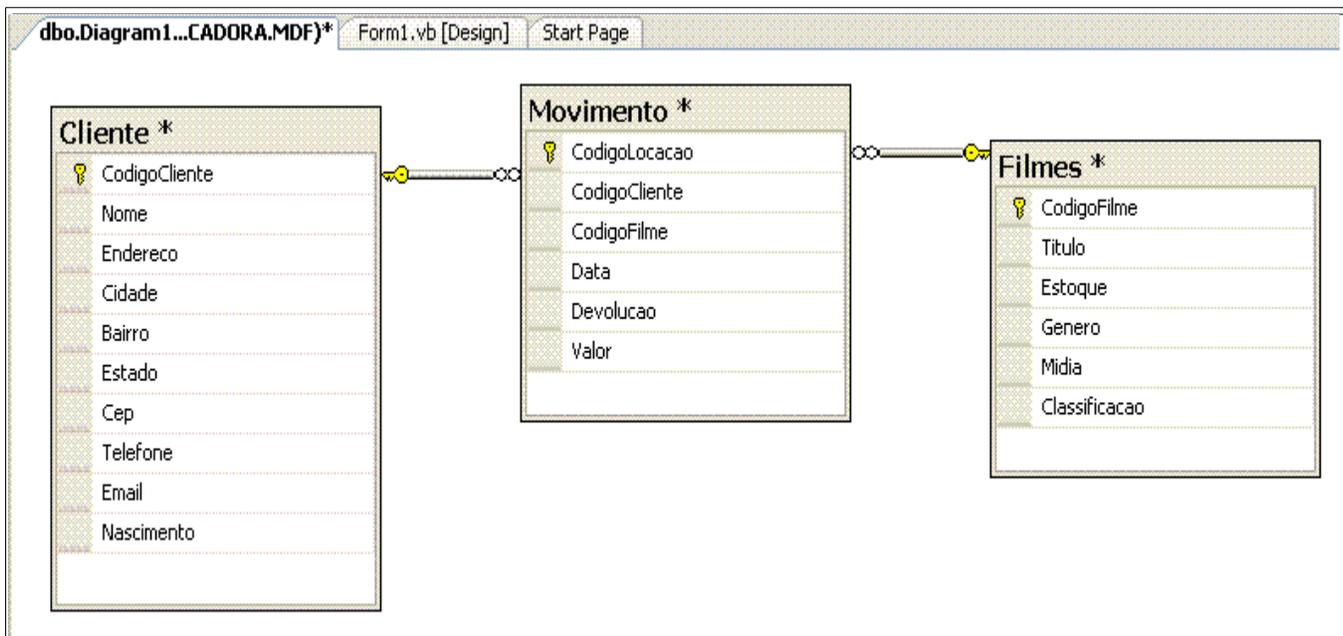
2- Filmes(CodigoFilme) --> Movimento (CodigoFilme) - Relacionamento do tipo UM-PARA-MUITOS

Ou seja:

1- Existe um relacionamento entre a tabela Cliente e a tabela Movimento entre as chaves primárias CódigoCliente onde para cada Cliente da tabela Cliente eu posso ter infinitos Clientes na tabela Movimento.

2- Existe um relacionamento entre a tabela Filmes e a tabela Movimento entre as chaves primárias CódigoFilme onde para cada Filme da tabela Cliente eu posso ter infinitos Filmes na tabela Movimento.

Nota: Esta modelagem foi adotada apenas com objetivo didático e não deve ser usada em uma aplicação de produção pois não está, como podem notar, corretamente normalizada.

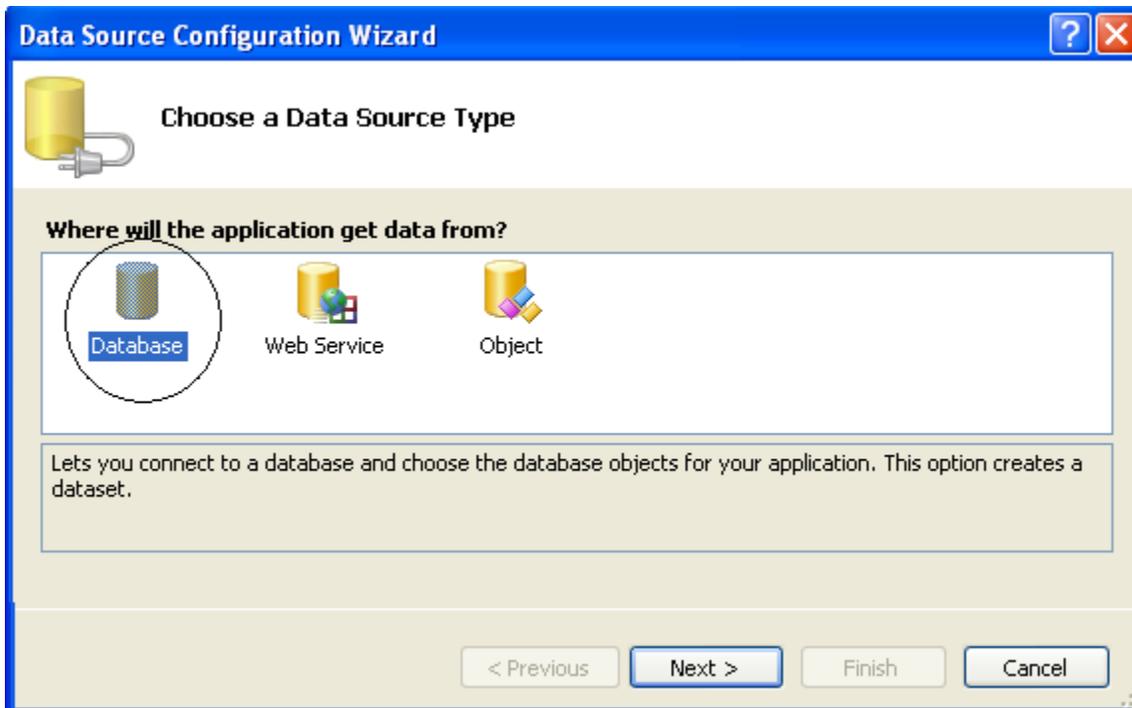


Pronto ! Já temos o banco de dados criado no SQL Server 2005 e as tabelas [Cliente](#), [Filmes](#) e [Movimento](#) também criadas e com campos e relacionamentos definidos. Note que tudo foi feito usando o VB.NET 2005 com ajuda dos seus assistentes.

2- Criação de uma conexão com a fonte de dados e definição dos DataSets

Vamos agora criar a conexão e as fonte de dados do tipo [DataSet](#) usando o assistente do VB 2005.

1- Clique na guia Data Sources e selecione a opção - [Add New DataSource](#); Na janela [Data Source Configuration Wizard](#), selecione a opção DataBase e clique no botão Next>.



Fontes de dados

O Visual Studio 2005 apresenta o conceito de fontes de dados em um projeto. Uma fonte de dados representa os dados disponíveis para um aplicativo. Esses dados não estão necessariamente em um banco de dados, o **Data Source Configuration Wizard** (Assistente para Configuração de Fonte de Dados) que você usa para definir a fonte de dados permite obter os dados de três fontes diferentes:

1. **Banco de dados** — pode ser um banco de dados baseado no servidor, como o SQL Server ou o Oracle, ou um banco de dados baseado em arquivo, como o Access ou o SQL Server Express. O Visual Studio gera automaticamente os **DataSets** de tipos, bem como outras classes, e os adiciona ao projeto.
2. **Objeto** — qualquer objeto com propriedades públicas pode ser a fonte de dados. Não é necessário implementar nenhuma interface especial.
3. **Serviço da Web** — criar uma fonte de dados em um serviço da Web cria objetos correspondentes ao tipo de dados retornado por esse serviço.

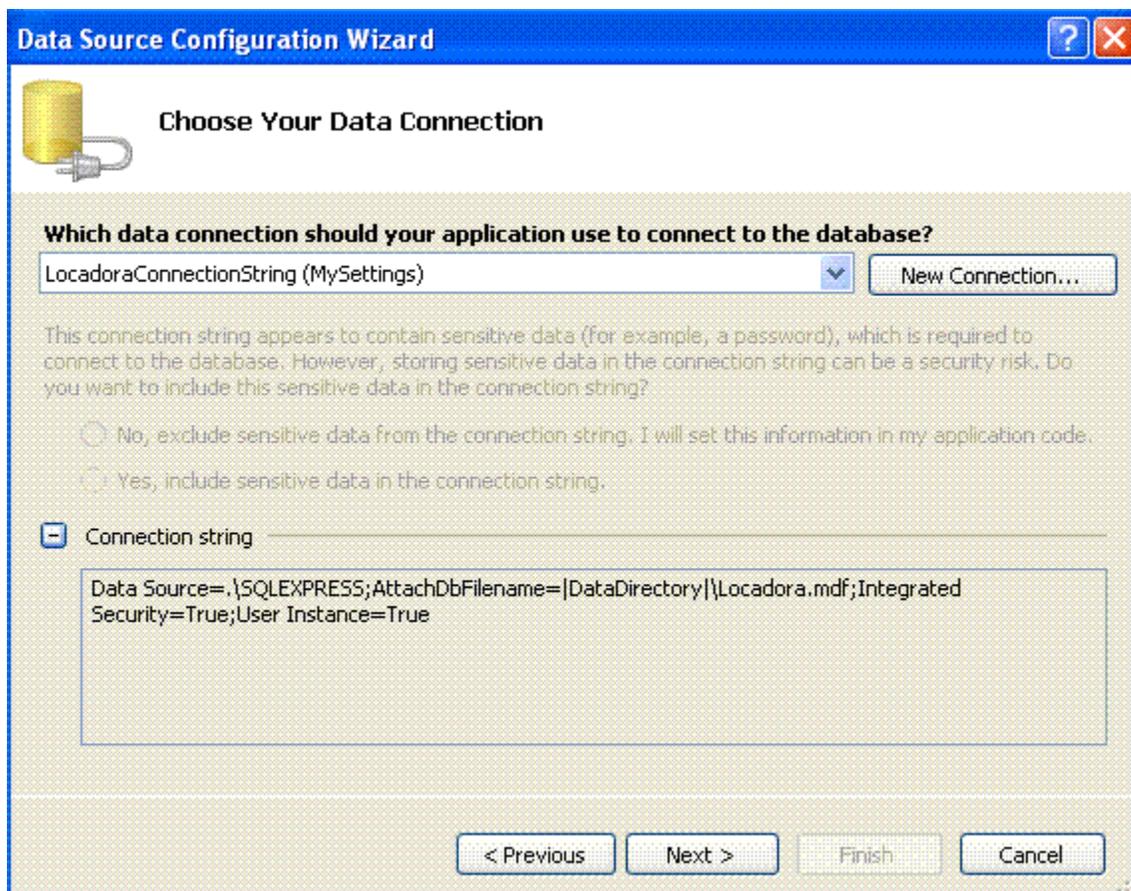
A finalidade da fonte de dados é dupla. Em primeiro lugar, ela torna mais fácil especificar, criar e gerar classes com rigidez de tipos que representam os dados do aplicativo. Em segundo lugar, fornece um mecanismo flexível, porém uniforme, de criação rápida de interfaces de usuário **WinForm** e **WebForms** avançadas e altamente funcionais. Neste artigo, veremos o quanto isso é rápido, fácil e flexível.

Também nos concentraremos na criação de fontes de dados de banco de dados (**DataSet**) e no uso dessas fontes em aplicativos WinForms. No entanto, é importante salientar estes dois pontos:

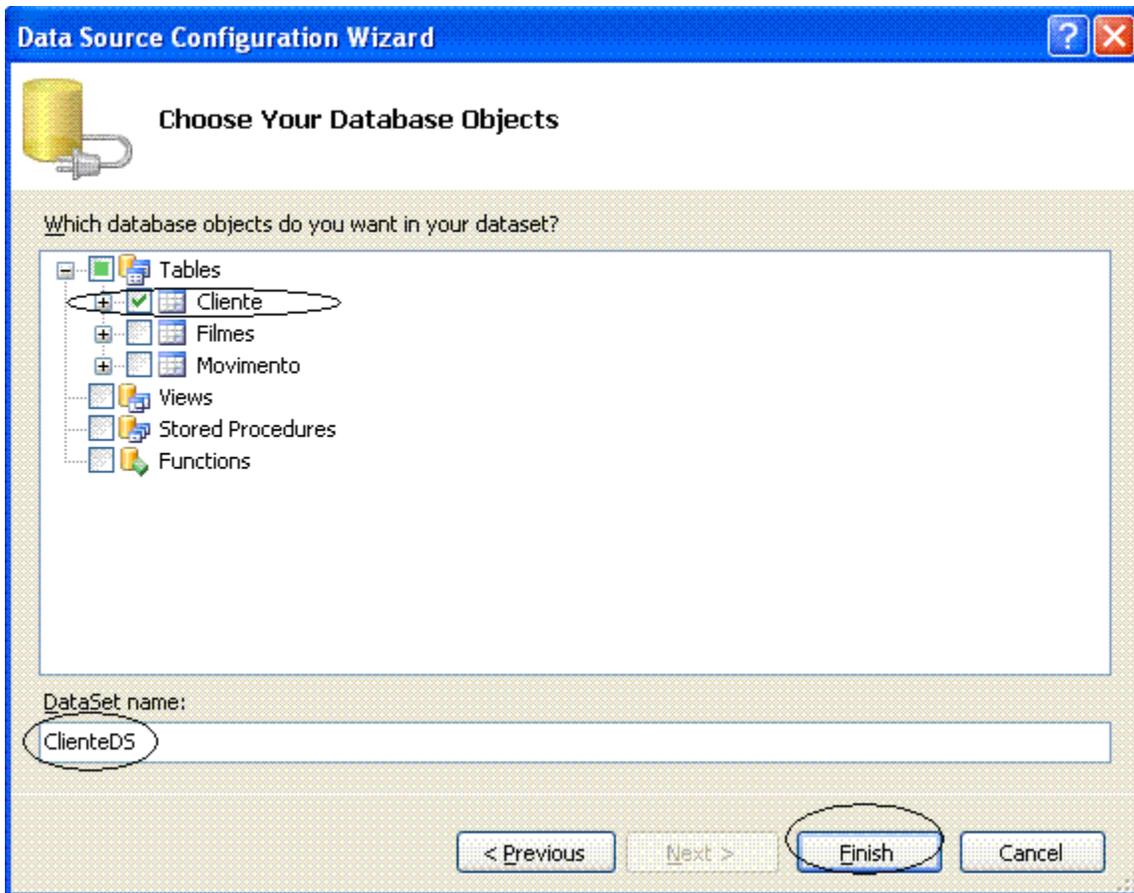
- Após a criação de uma fonte de dados, você a utiliza da mesma maneira, independentemente da origem dos dados. Ou seja, assim como você pode facilmente (e graficamente) ligar uma fonte de dados baseada em um **Database** a uma grade ou a um conjunto de controles, também pode fazer isso com os dados cuja origem é um serviço da Web ou seus objetos comerciais personalizados.
- As fontes de dados são definidas da mesma maneira, independentemente de serem usadas em um aplicativo **WinForms** ou **WebForms**. Os diferentes provedores de dados também são abstratos; assim, se o seu acesso aos dados estiver exposto somente pelo uso de **DataSets** e **TableAdapters**, para alterar o banco de dados real, bastará alterar a seqüência de caracteres de conexão e gerar as classes novamente.

fonte MSDN - Novos recursos de DataSet no Visual Studio 2005 - Jackie Goldstein - Renaissance Computer Systems

2- Na tela seguinte, defina a conexão com o banco de dados Locadora.mdf. Lembrando que podemos exibir a string de conexão. Clique no botão Next>

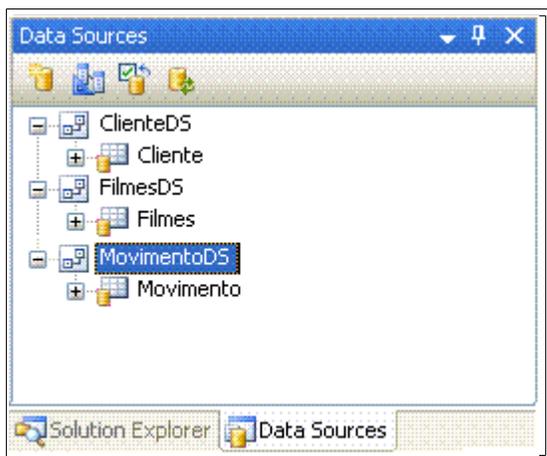


3- No próximo passo vamos criar uma fonte de dados do tipo DataSet para cada uma das tabelas. Poderíamos criar um DataSet com mais de uma tabela se desejamos apresentar um relacionamento do tipo [mestre-detalhes](#). Selecione então a tabela [Clientes](#), informe o nome ClienteDS e clique no botão Finish.

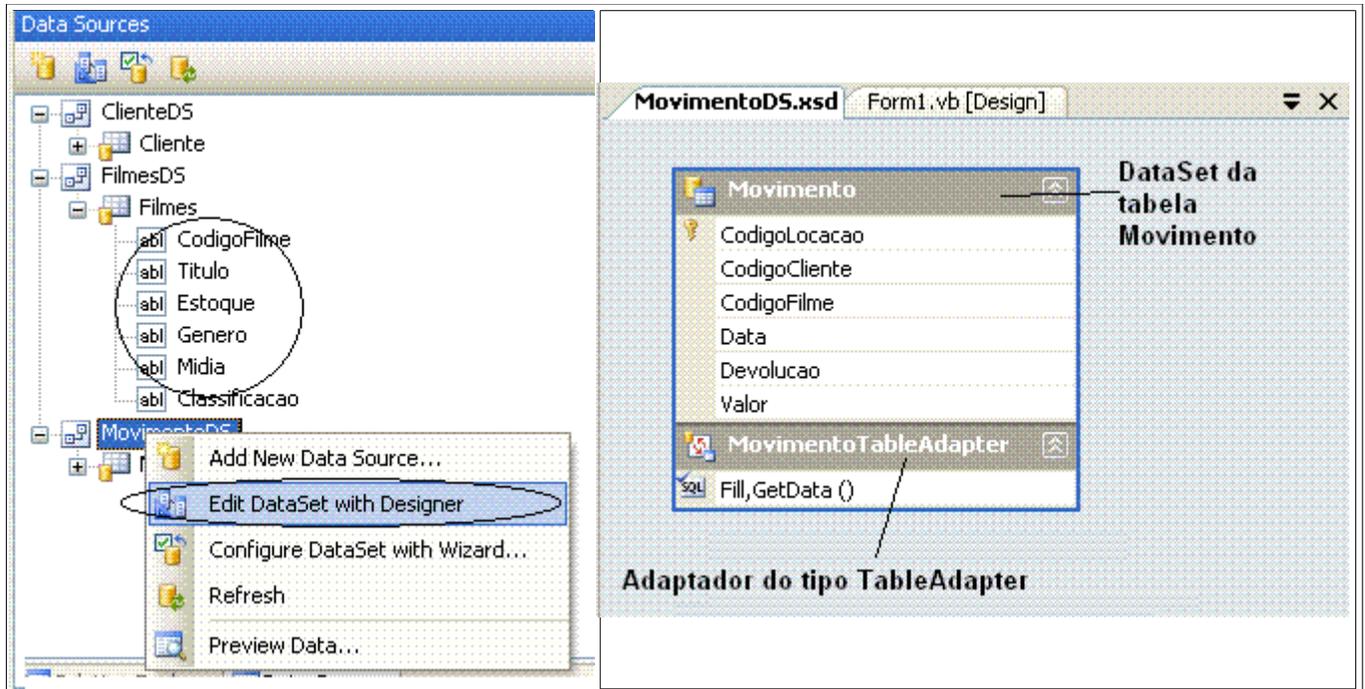


Repita o procedimento acima (item 1 ao 3) e crie as fontes de dados FilmesDS e MovimentoDS para as tabelas [Filmes](#) e [Movimento](#).

Se tudo deu certo você deverá ter 3 fontes de dados do tipo DataSet em sua janela, conforme figura abaixo:



Podemos navegar pelos objetos DataSet criados exibindo os campos de cada um deles e também editar o DataSet. Para isto clique com o botão direito do mouse sobre o DataSet e selecione a opção - [Edit DataSet with Designer](#).



Será exibido a representação do objeto **MovimentoDS**, conforme figura acima, onde temos a tabela com seus respectivos campos e o objeto **TableAdapter** que permite alimentar, atualizar e aplicar filtros a fonte de dados.

E chegamos ao final da primeira parte onde temos a base de dados e as entidades relacionadas a esta base de dados que simbolizam os nossos objetos de negócios os quais são representados pelas tabelas **Cliente**, **Filmes** e **Movimento**; temos também o adaptador que permite sincronizar essas entidades em memória com a base de dados.

Nota:

O conceito de um **TableAdapter**, por outro lado, é novo no Visual Studio 2005. A idéia subjacente é que um **TableAdapter** com rigidez de tipos é o equivalente com rigidez de tipos do **DataAdapter** padrão. Você usa o **TableAdapter** para se conectar a um banco de dados e executar consultas (ou procedimentos armazenados) nesse banco de dados, bem como para preencher com dados um **DataTable** associado. Cada par **DataTable-TableAdapter** é indicado simplesmente como um **TableAdapter**

O **TableAdapter** é essencialmente um invólucro ao redor de um **DataAdapter** padrão, que oferece vários benefícios:

- A mesma classe **TableAdapter** pode ser usada em mais de um formulário ou componente para que qualquer alteração em consultas/comandos seja automaticamente refletida em todas as instâncias. Essa situação é diferente da existente, na qual cada componente que acessa o banco de dados deve ter seu próprio **DataAdapter** configurado individualmente. Dessa forma, fica muito mais fácil garantir a sincronização de **DataTables** e **DataAdapters**.
- Em vez de usar vários **DataAdapters** (ou código de comutação artesanal) para ter várias consultas/comandos para um único **DataTable**, um **TableAdapter** permite definir facilmente vários comandos para um **DataTable** específico.
- Os comandos de preenchimento têm nomes legíveis ("amigáveis") e o **TableAdapter** inclui um código para preencher automaticamente as informações de tipo e valor de todos os parâmetros desses métodos de comando. Você não precisa mais se preocupar em passar tipos de dados específicos do provedor, como **SqlInt**.

fonte **MSDN** - Novos recursos de DataSet no Visual Studio 2005 - Jackie Goldstein - Renaissance Computer Systems

Parte 2

Continuando o desenvolvimento do nosso projeto Locadora de Filmes no *Visual Basic 2005 Express Edition* este artigo irá mostrar como você pode criar a interface com o usuário usando alguns dos recursos disponíveis no VB 2005. Se você está chegando agora recomendo que leia a primeira parte em : **VB.NET 2005 - Criando uma aplicação completa : Locadora de Filmes I**

Os principais formulários que iremos criar para a aplicação são :

1. Principal.vb - formulário MDI que apresenta a aplicação com um menu de opções
2. Clientes.vb - formulário usado para cadastrar os clientes
3. Movimento.vb - formulário que gerencia as informações sobre locações de filmes
4. Devolucao.vb - formulário que gerencia as informações sobre a devolução de filmes alugados.

Criando os formulários da aplicação

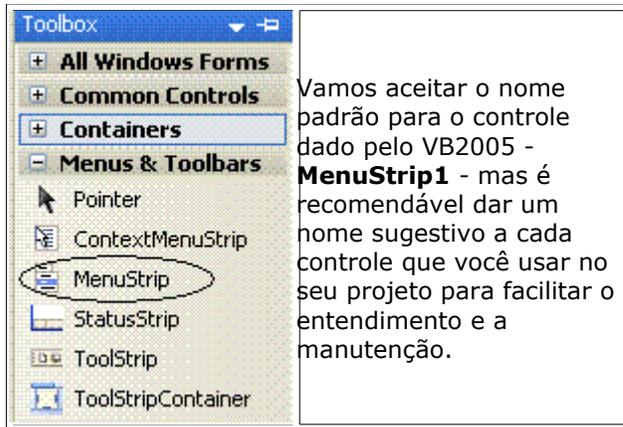
- O formulário Principal

Vamos usar o formulário padrão **form1.vb** para criar o formulário principal da nossa aplicação. Vamos alterar o nome do arquivo para Principal.vb (*clique com o botão direito do mouse sobre o nome do arquivo e selecione Rename*) e o nome do formulário também para Principal (*Name=Principal*) na janela de propriedades.

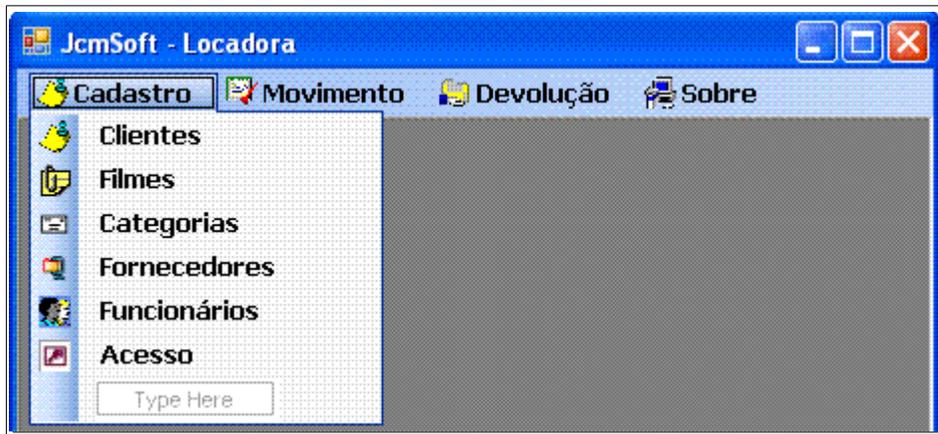
Este formulário será o formulário que irá conter os demais formulários do projeto, e, portanto deverá ser um formulário **MDI**, desta forma altere o valor da propriedade **IsMdiContainer** para **True**. Altere também as propriedades conforme abaixo na janela de propriedades deste formulário:

Propriedade	Valor
StartPosition	CenterScreen
Text	JcmSoft - Locadora <i>(ou algo que mais lhe agrade)</i>

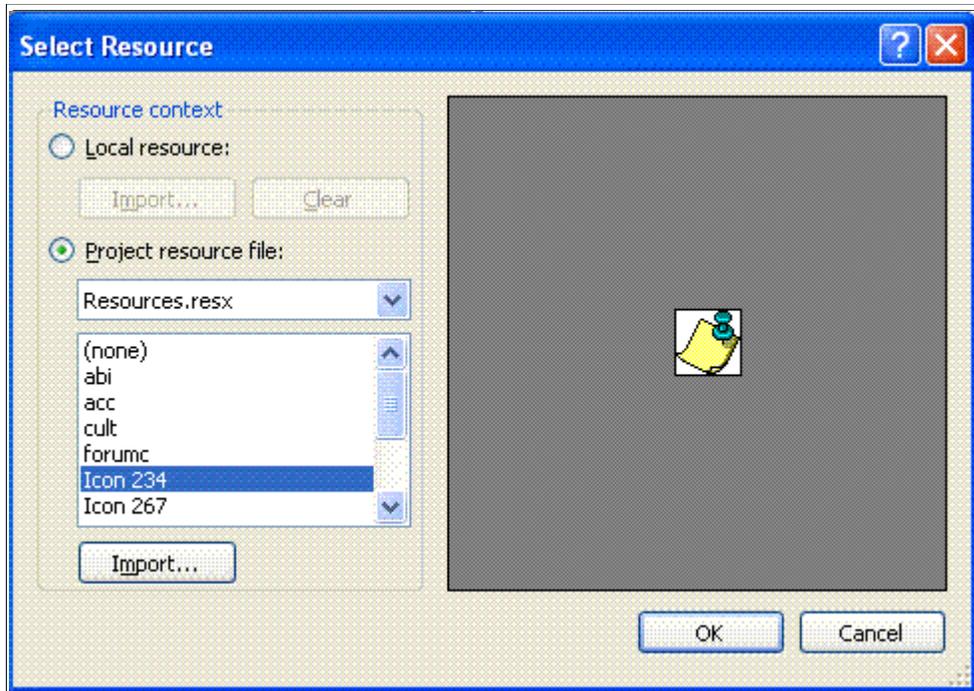
Vamos incluir um menu no formulário de forma a poder gerenciar as opções referente as funcionalidades implementadas. Inclua, a partir da **ToolBox** um controle **MenuStrip** na Seção **Merus & ToolBars**, no formulário Principal.



A seguir Digite diretamente no controle os textos conforme mostrado na figura abaixo.



Perceba que incluímos ícones no menu. Fazemos isto selecionando o texto digitado e na janela de propriedades clicando na propriedade **Image** do item. Na janela **Select Resource** podemos então selecionar a imagem como um recurso local ou como um recurso do projeto clicando a seguir no botão **Import...** e selecionando a imagem:



A seguir vamos incluir um controle **Panel** na parte inferior do nosso formulário diretamente da **ToolBar** a partir da seção **Containers**.

Toolbox

- + All Windows Forms
- + Common Controls
- Containers
- Pointer
- FlowLayoutPanel
- GroupBox
- Panel
- SplitContainer
- TabControl
- TableLayoutPanel

Este controle é do tipo **Container** pois pode conter outros controles. Altere as seguintes propriedades do controle na janela de propriedades:

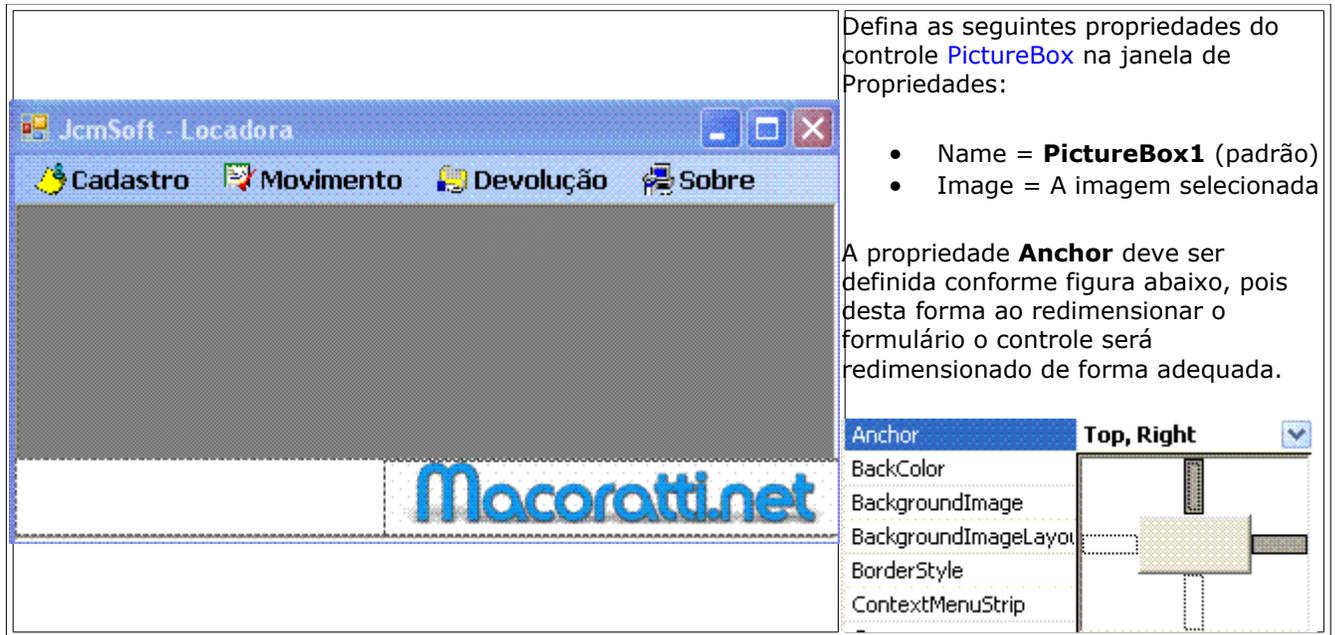
- Aceite o nome padrão - Name = **Panel1**
- BackColor = **White**

A propriedade **Anchor** deve ser definida conforme figura abaixo, pois desta forma ao redimensionar o formulário o controle será redimensionado de forma adequada.

Anchor **ttom, Left, Right**

- AutoScroll
- AutoScrollMargin
- AutoScrollMinSize
- AutoSize
- AutoSizeMode
- BackColor

Inclua um controle **PictureBox** no interior do controle **Panel** que você acabou de inserir no formulário. Posicione o controle a direita do controle **Panel** e inclua uma imagem clicando na propriedade **Image** e selecionando uma imagem via opção **Import** da janela **Select Resource**, conforme mostrado na figura abaixo.



Defina as seguintes propriedades do controle **PictureBox** na janela de Propriedades:

- Name = **PictureBox1** (padrão)
- Image = A imagem selecionada

A propriedade **Anchor** deve ser definida conforme figura abaixo, pois desta forma ao redimensionar o formulário o controle será redimensionado de forma adequada.

Anchor **Top, Right**

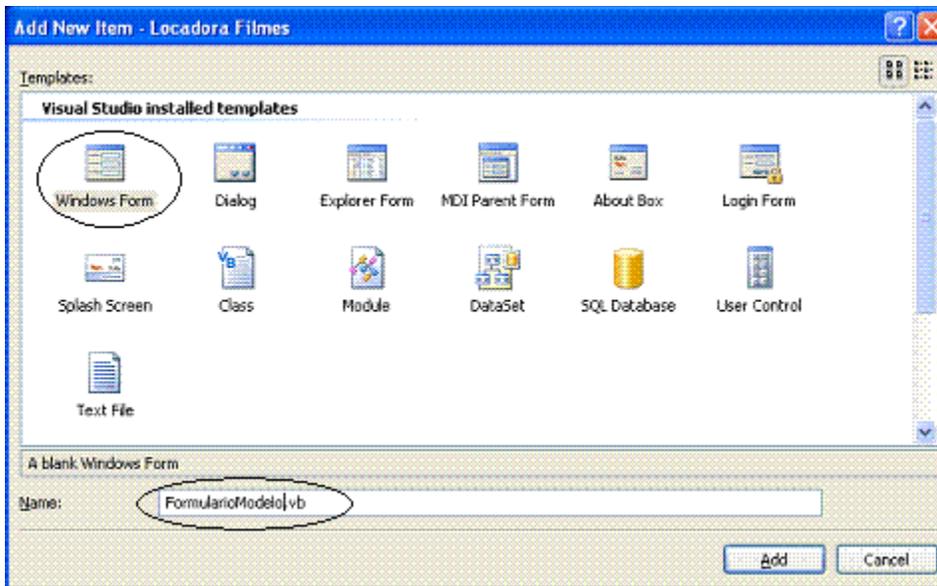
BackColor
 BackgroundImage
 BackgroundImageLayout
 BorderStyle
 ContextMenuStrip

Criando um modelo para formulários

Os formulários [Clientes.vb](#) , [Movimento.vb](#) e [Devolucao.vb](#) são formulários que terão praticamente um mesmo comportamento e identidade visual por este motivo seria interessante que fossem criados a partir de um modelo (*template*) de formulário. É isto que iremos fazer , primeiro vamos criar um formulário com os componentes comuns aos formulários que desejamos criar para em seguida criar os formulários a partir deste modelo. Chamamos este comportamento de herança visual.

- Criando o formulário modelo

Inclua no novo formulário Windows no projeto , [clcando com o botão direito do mouse sobre o nome do projeto](#) e selecionando a opção **Add->New Item** e selecionando a opção **Windows Forms** na janela Add New Item. Informe o nome **FormularioModelo.vb** , conforme figura abaixo.



A seguir vamos incluir os componentes que achamos que serão comuns aos formulários que serão criados a partir deste modelo. Para não complicar vou definir o formulário modelo contendo apenas um controle **Label** e um controle **GroupBox** conforme figura abaixo:

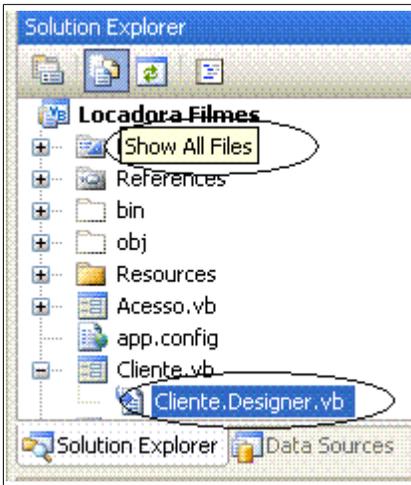
<p>The image shows a screenshot of a Windows Form titled 'FormularioModelo'. The form has a blue title bar. At the top, there is a dark blue header area containing a 'Label1' control. Below the header is a large, empty rectangular area with a light blue background, representing a GroupBox.</p>	<p>Altere as seguintes propriedades do formulário modelo na janela de propriedades:</p> <ul style="list-style-type: none"> • Name = FormularioModelo • BackColor = 192; 192; 255 • ControlBox = False • Font = Arial; 12pt; style=Bold • KeyPreview = True • Text = FormularioModelo
	<p>Altere as seguintes propriedades do controle Label1 :</p> <ul style="list-style-type: none"> • Anchor = Top, Left, Right • BackColor = Navy • Font = Arial; 18pt; style=Bold • ForeColor = White • Modifiers =

	Protected
	Defina a propriedade Anchor do controle GroupBox1 como sendo igual a : Top, Bottom, Left, Right

Criando os formulários a partir do Modelo

Vou mostrar agora como você pode criar formulários a partir do formulário modelo acima. Vou fazer isto uma única vez para o formulário **Cientes.vb** de forma que para os demais bastará repetir o processo.

Inclua um novo formulário Windows com o nome de **Cientes.vb**.



Na janela **Solution Explorer** clique no ícone **Show All Files** para que todos os arquivos que compõem a solução sejam exibidos.

Clique no sinal de mais ao lado do formulário **clientes.vb** e clique duas vezes sobre o item **Clientes.Designer.vb**

Isto irá exibir o código do formulário criado pelo VB2005.

O formulário **Cientes.vb**, por padrão, herda sua identidade visual da classe **Windows.Forms.Form** conforme mostrado na **figura 1** abaixo. Para que o formulário passe a herdar a identidade visual do nosso formulário modelo altere a linha de código **Inherits** de **System.Windows.Forms.Form** para **FormularioModelo** conforme figura 2 abaixo:

```

<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()>
Partial Class clientes
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing AndAlso components IsNot Nothing Then
            components.Dispose()
        End If
        MyBase.Dispose(disposing)
    End Sub

```

Figura 1 - Herdando de **Windows.Forms.Form**

```

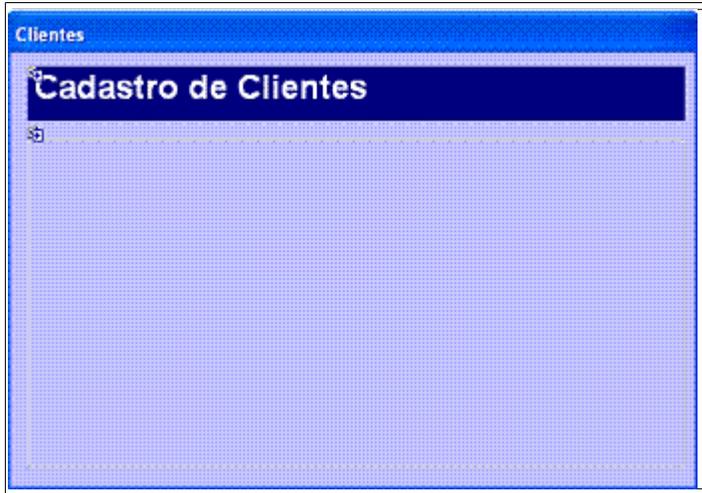
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()>
Partial Class Clientes
    Inherits FormularioModelo

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing AndAlso components IsNot Nothing Then
            components.Dispose()
        End If
        MyBase.Dispose(disposing)
    End Sub

```

Figura 2 - Herdando do formulário - **FormularioModelo**

Agora compile o projeto na opção **Build** do menu, opção **Build Locadora Filmes**. A seguir abra o formulário **Cientes.vb** e altere a propriedade **Text** do controle **Label1** para - **Cadastro de Clientes**, conforme abaixo:



Repita o procedimento acima para os formulário **Movimento.vb** e **Devolucao.vb** de forma a obter o resultado abaixo para os respectivos formulários:



Vamos retornar ao formulário MDI **Principal.vb** e vamos incluir as chamadas para cada formulário que criamos a partir da seleção do menu do formulário. Para isto clique na opção do menu desejada e inclua o código para exibir cada formulário conforme abaixo:

```
Private Sub MovimentoToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MovimentoToolStripMenuItem.Click
    My.Forms.Movimento.MdiParent = Me
    My.Forms.Movimento.Show()
End Sub
```

```
Private Sub DevolucaoToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
```

```

System.EventArgs) Handles DevolucaoToolStripMenuItem.Click
    My.Forms.Devolucao.MdiParent = Me
    My.Forms.Devolucao.Show()
End Sub

Private Sub ClientesToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ClientesToolStripMenuItem1.Click
    My.Forms.Clientes.MdiParent = Me
    My.Forms.Clientes.Show()
End Sub
    
```

Vejamos como o código funciona:

My.Forms.Movimento.MdiParent = Me -> Permite que o formulário seja aberto no container MDI e não em outra janela

My.Forms.Movimento.Show() -> Exibe o formulário

Perceba que estamos usando o recurso **My** do VB2005. Em versões anteriores teríamos que usar o seguinte código:

Dim frmMovimento As New Movimento
frmMovimento.Show()

Nota: Namespace MY
O namespace **My** veio reunir algumas das funções mais utilizadas no .NET Framework, expor novas funções que antes só poderiam ser utilizadas através de APIs, ou até mesmo facilitar o uso de outras funções. O objetivo é ganhar produtividade obtendo as informações de forma mais rápida.

Abaixo as principais classes e suas descrições do **namespace My**:

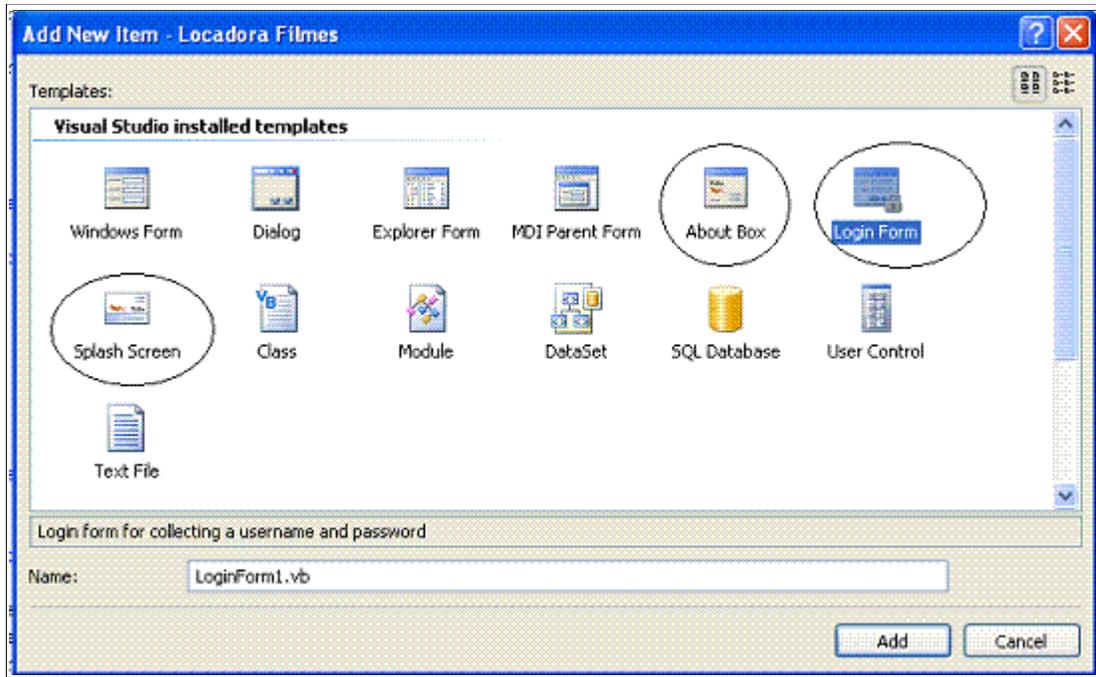
Namespace My	
Application	Sistemas e serviços
Computer	Computador Host e serviços
Forms	Formulários do projeto atual
Resources	Recursos
Settings	parâmetros do sistema e do usuário
User	segurança do usuário

Ex: My.Application... , My.Computer... , My.Forms..., My.Resources..., My.Settings..., My.User...

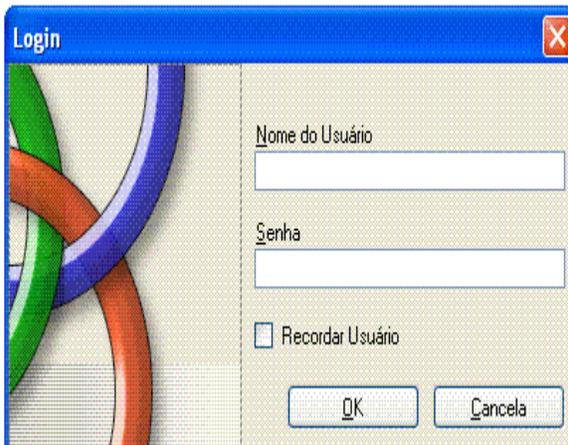
Incluindo alguns modelos prontos

O VB2005 vem com alguns modelos prontos de formulários, basta, pegar e usar. Vou usar os seguintes modelos de formulário já disponíveis no VB2005

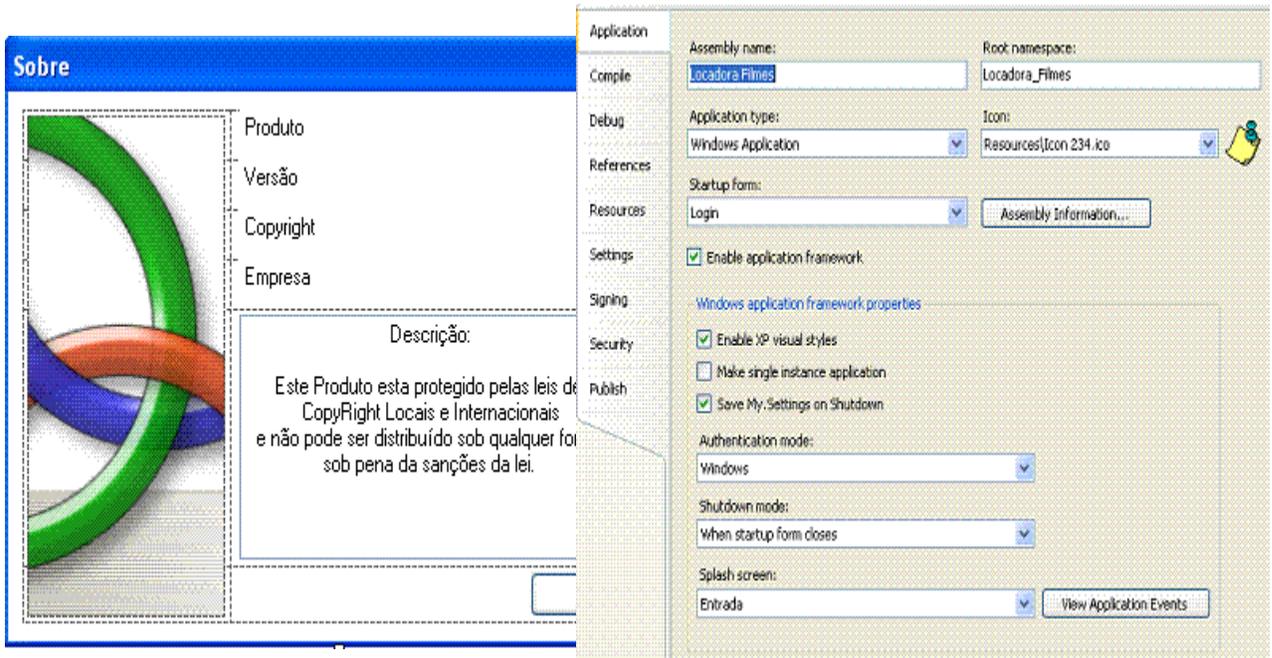
- LoginForm - nome dado Login.vb
- Splash Screen - nome dado Entrada.vb
- About Box - nome dado Sobre.vb



Abaixo estou exibindo cada um dos respectivos formulários com seu layout adotado:

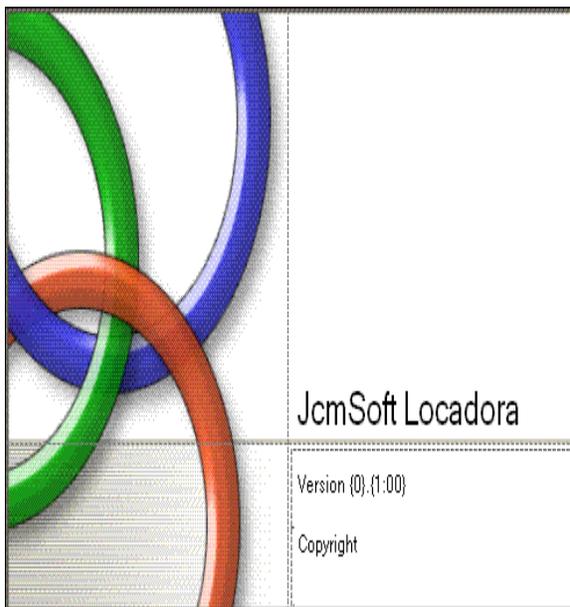


O formulário login.vb permite efetuar uma validação do usuário com senha. Vamos implementar o recurso criando uma tabela chamada Usuarios com os Campos: nomeUsuario e senhaID. A senha será armazenada na forma de um **Hash** que iremos Gerar usando uma Classe.



formulário Sobre.vb

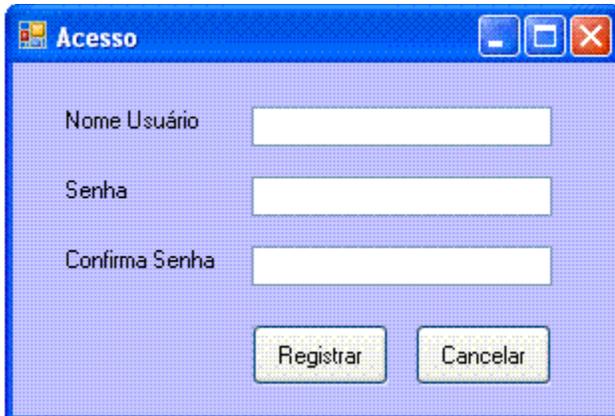
O formulário sobre busca informações definidas na Configuração do projeto na guia Application do recurso My Project



A tela de apresentação é exibida no início da aplicação Basta informar na configuração do projeto no campo Splash Screen (ver figura acima) o nome do formulário que usamos, no caso Entrada.

Como vamos implementar o acesso restrito por chave e senha teremos que incluir um novo formulário chamado **Acesso.vb** que irá permitir a inclusão destas informações na base de dados **Usuarios** (que vou criar em seguida).

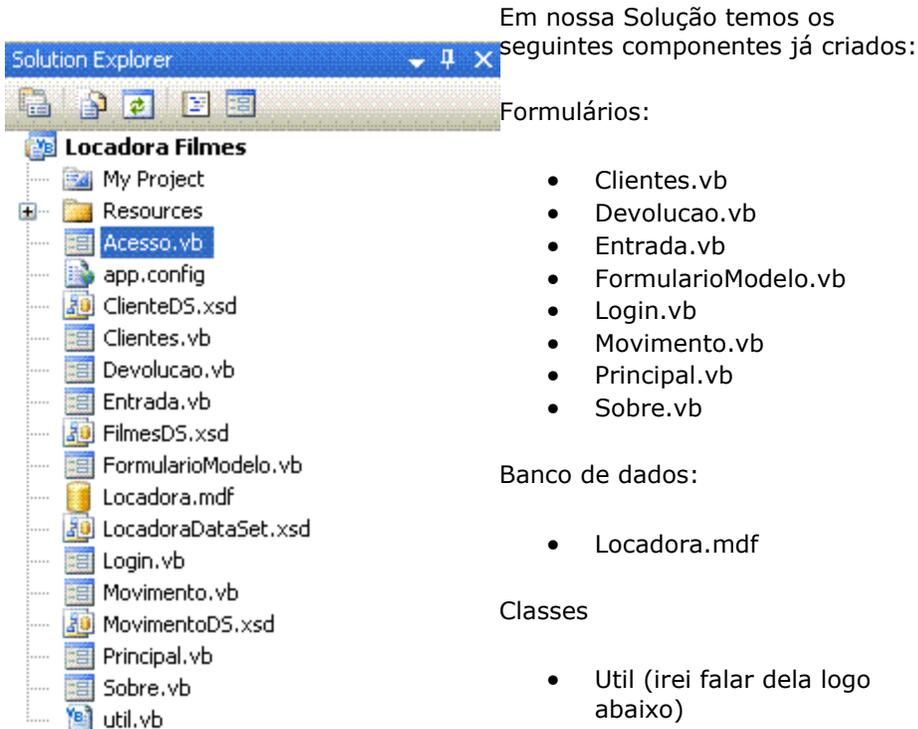
Inclua um novo formulário chamado **Acesso.vb** no projeto conforme o layout abaixo:



Os demais componentes do formulário são:

- TextBox : txtUsuario, TxtSenha e TxtSenha2
- Button : btnRegistrar e btnCancel

Ao visualizarmos a janela **Solution Explorer** do nosso projeto neste momento teremos a seguinte fotografia do mesmo:



Em nossa Solução temos os seguintes componentes já criados:

Formulários:

- Clientes.vb
- Devolucao.vb
- Entrada.vb
- FormularioModelo.vb
- Login.vb
- Movimento.vb
- Principal.vb
- Sobre.vb

Banco de dados:

- Locadora.mdf

Classes

- Util (irei falar dela logo abaixo)

Recursos

- As imagens usadas no menu

Vou continuar falando da implementação do formulário **Login.vb** e do formulário **Acesso.vb** onde iremos fazer o acesso ao banco de dados na tabela **Usuarios** e usar o recurso de gerar um **Hash** de um texto.

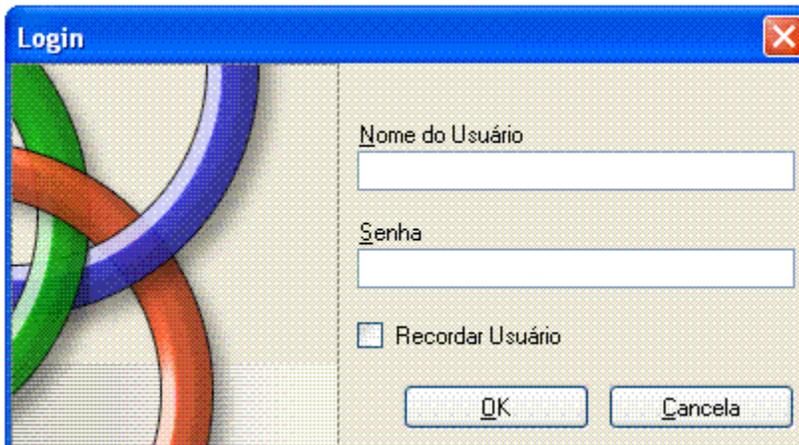
Parte 3

1. Continuando o desenvolvimento do nosso projeto Locadora de Filmes *no Visual Basic 2005 Express Edition* este artigo irá continuar a mostrar a implementação da interface do usuário.

Neste artigo vou mostrar como implementar os formulários **Login.vb** e **Acesso.vb**.

Implementando o formulário de Login

Como já vimos no artigo anterior o formulário de login - **Login.vb** - foi criado a partir de um modelo já existente no VB 2005.



O formulário **login.vb** permite efetuar uma validação do usuário com nome e senha.

Vamos implementar o recurso criando uma tabela chamada **Usuarios** com os campos: **nomeUsuario** e **SenhaId**.

A senha será armazenada na forma de um **Hash** que iremos gerar usando uma classe.

Incluímos um controle **CheckBox** para permitir que o nome do usuário seja armazenado na aplicação.

O código do evento **Click** do botão **OK** para este formulário é exibido abaixo:

```
Private Sub OK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles OK.Click

    Dim conn As New SqlConnection
    Dim comando As New SqlCommand
    Dim senhaID As String

    conn.ConnectionString = My.Settings.LocadoraConnectionString

    Try
        conn.Open()

        comando.Connection = conn
        comando.CommandText = "SELECT senhaID FROM usuarios Where
        nomeUsuario=@nomeUsuario"
        comando.Parameters.AddWithValue("@nomeUsuario", UsernameTextBox.Text)

        senhaID = comando.ExecuteScalar()

        conn.Close()

        If chkLembraUsuario.Checked Then
```

```

    My.Settings.Usuario = UsernameTextBox.Text
    My.Settings.Save()
Else
    My.Settings.Usuario = ""
End If

If util.GeraHash(PasswordTextBox.Text).Equals(senhaID) Then
    My.Forms.Principal.Show()
    Me.Hide()
Else
    MessageBox.Show("Senha/Usuário Incorreto(s)", "Senha incorreta",
    MessageBoxButtons.OK, MessageBoxIcon.Information)
    PasswordTextBox.Focus()
End If

Catch ex As SqlException
    MessageBox.Show("Erro ao efetuar a conexão com a base de dados : " +
ex.Message)
    conn.Dispose()
End Try
End Sub

```

Explicando o código:

A linha de código:

```
conn.ConnectionString = My.Settings.LocadoraConnectionString
```

permite a recuperação da string de conexão usando o recurso **My.Settings**

Após abrir a conexão criamos um comando com a instrução SQL que irá selecionar a senha da tabela **usuarios** onde o nome do usuário é passado como parâmetro conforme o código abaixo:

```
comando.Connection = conn
comando.CommandText = "SELECT senhaID FROM usuarios Where
nomeUsuario=@nomeUsuario"
comando.Parameters.AddWithValue("@nomeUsuario", UsernameTextBox.Text)

```

A seguir obtemos o valor da Senha através da execução do método **ExecuteScalar**

O objeto *Command* fornece o método *ExecuteScalar* que permite retornar um valor único de uma fonte de dados. Este método executa uma consulta e retorna a primeira coluna da primeira linha do conjunto de registros retornado

O método **ExecuteScalar** é o meio mais rápido de se acessar e retornar dados em um base de dados.

```
senhaID = comando.ExecuteScalar
```

Verificamos se o **checkbox** esta marcado; neste caso salvamos o nome do usuário nas configurações da aplicação.

```
If chkLembraUsuario.Checked Then
```

```

    My.Settings.Usuario = UsernameTextBox.Text
    My.Settings.Save()
Else
    My.Settings.Usuario = ""
End If

```

A seguir o método **GeraHash** da classe util é usado para gerar o hash da senha informada para comparar com o hash armazenado no banco de dados. (isto será feito via formulário acesso.vb).

```

If util.GeraHash(PasswordTextBox.Text).Equals(senhaID) Then
    My.Forms.Principal.Show()
    Me.Hide()
Else
    MessageBox.Show("Senha/Usuário Incorreto(s)", "Senha incorreta",
    MessageBoxButtons.OK, MessageBoxIcon.Information)
    PasswordTextBox.Focus()
End If

```

Nota: O recurso My usado em *My.Forms.Principal.Show()* permite exibir o formulário de forma rápida e direta.

Para implementar A classe **util** e seu método **GeraHash** inclua um novo módulo de classe no projeto. Menu **Project** opção **Add Class** inclua o módulo dando ao mesmo o nome util.vb. A seguir inclua o código abaixo na classe:

```

Imports System.Security.Cryptography
Imports System.text

Public Class util

Public Shared Function GeraHash(ByVal texto As String) As
String

' Cria um objeto encoding para assegurar o padrão
'de encoding para o texto origem
Dim Ue As New UnicodeEncoding()

'Retorna um byte array baseado no texto origem
Dim ByteSourceText() As Byte = Ue.GetBytes(texto)

'Instancia um objeto MD5
Dim Md5 As New MD5CryptoServiceProvider()

'Calcula o valor do hash para o texto origem
Dim ByteHash() As Byte = Md5.ComputeHash(ByteSourceText)

'Converte o valor obtido para o formato string
Return Convert.ToBase64String(ByteHash)
End Function

End Class

```

Implementando o formulário de Acesso

O formulário acesso permite o cadastramento de usuários e senhas na base de dados **Acesso.mdb**, tabela **Usuarios**. São armazenados o nome do usuário e o hash da senha informada.

Os demais componentes do formulário são:

- TextBox : txtUsuario, TxtSenha e TxtSenha2
- Button : btnRegistrar e btnCancel

O código do evento **Click** do botão Registrar é mostrado a seguir:

```
Private Sub btnRegistrar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles btnRegistrar.Click

    Dim conn As New SqlConnection
    Dim comando As New SqlCommand
    Dim reg As Integer

    conn.ConnectionString = My.Settings.LocadoraConnectionString

    If txtUsuario.Text = "" Then
        MessageBox.Show("Informe um valor válido.")
        txtUsuario.Focus()
        Exit Sub
    End If

    If txtsenha.Text = "" Then
        MessageBox.Show("Informe um valor válido.")
        txtsenha.Focus()
        Exit Sub
    End If

    If Not txtsenha.Text.Equals(txtSenha2.Text) Then
        MessageBox.Show("A senha não confere.")
        txtSenha2.Focus()
        Exit Sub
    End If

    Try
        conn.Open()
        comando.Connection = conn
        comando.CommandText = "INSERT INTO Usuarios(nomeUsuario,senhaID)
        values(@Usuario,@senha)"
        comando.Parameters.AddWithValue("@Usuario", txtUsuario.Text)
        comando.Parameters.AddWithValue("@senha", util.GeraHash(txtsenha.Text))
        reg = comando.ExecuteNonQuery()
    End Try
End Sub
```

```

    MessageBox.Show(reg.ToString + " registro de Usuário incluído com sucesso.")
Catch ex As SqlException
    MessageBox.Show("Erro ao efetuar a conexão com a base de dados : " + ex.Message)
Finally
    conn.Close()
    conn.Dispose()
End Try
End Sub

```

Neste código estamos incluindo, via instrução **SQL INSERT INTO**, o nome do usuário e a senha informada na tabela **usuarios**. Os parâmetros são montados com os dados das caixas de texto, e, o método **GeraHash()** é usado para gerar o hash do texto da senha informado.

```
conn.Open()
```

```
comando.Connection = conn
```

```
comando.CommandText = "INSERT INTO Usuarios(nomeUsuario,senhaID)
values(@Usuario,@senha)"
```

```
comando.Parameters.AddWithValue("@Usuario", txtUsuario.Text)
```

```
comando.Parameters.AddWithValue("@senha", util.GeraHash(txtsenha.Text))
```

O método **executeNonQuery()** é usado para executar a consulta ação. Este método é usado para executar um comando SQL que não retorna registros.

```
reg = comando.ExecuteNonQuery()
```

Nossa aplicação já possui uma forma de cadastrar os seus usuários via formulário **acesso.vb** e de permitir a validação dos mesmos para acesso a aplicação via formulário **login.vb**.

Nota: Para permitir que a mudança de um campo para outro do formulário seja possível com o pressionamento da tecla **ENTER** altere a propriedade **KeyPreview** do formulário para **True** e inclua o seguinte trecho de código

```

Private Sub Form1_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles MyBase.KeyPress

If e.KeyChar = Convert.ToChar(13) Then
    e.Handled = True
    SendKeys.Send("{TAB}")
End If

End Sub

```

Parte 4

1. Continuando o desenvolvimento do nosso projeto Locadora de Filmes no *Visual Basic 2005 Express Edition* este artigo irá continuar a mostrar a implementação da interface do usuário.

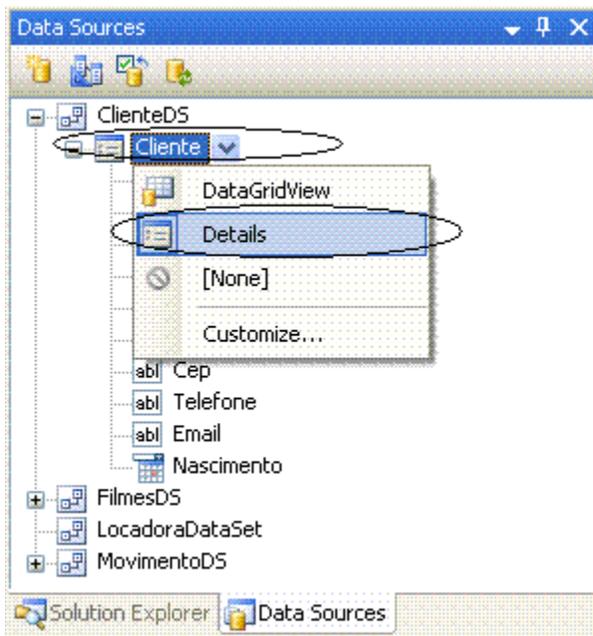
Neste artigo vou mostrar como implementar a interface com o usuário através dos formulários com a seguintes funcionalidades:

1. **Cadastrar clientes - Clientes.vb**
2. **Cadastrar Filmes - Filmes.vb**
3. **Cadastrar Categorias - Categorias.vb**
4. **Registrar Locação de Filmes - Movimento.vb.**

Implementando o formulário para cadastrar Clientes : Clientes.vb

Apenas para você lembrar: já criamos o dataset **ClienteDS** que representa a tabela Clientes(criada no primeiro artigo) e já criamos o formulário Clientes.vb que herda do formulário **FormularioModelo.vb**.

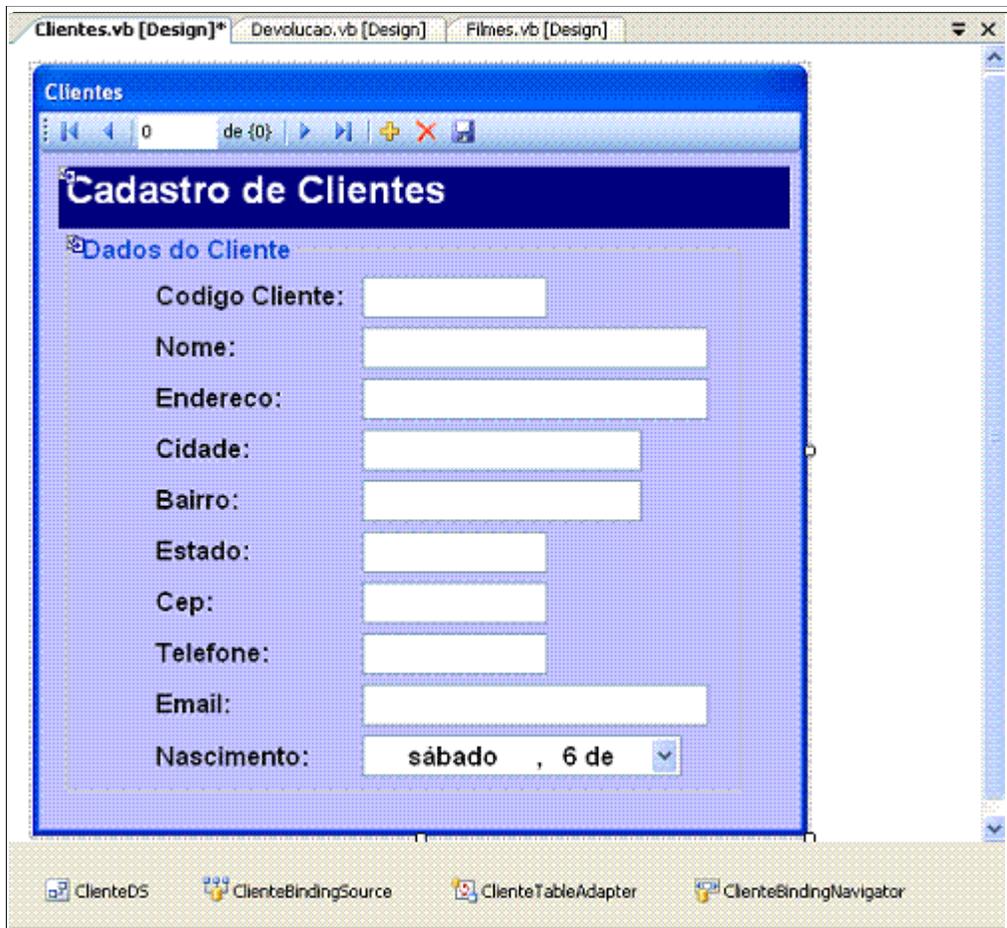
Para implementar o formulário para cadastrar clientes vou usar o recurso do assistente do VB2005. Acompanhe:



1- Na janela DataSource, expanda o DataSet **ClienteDS** e a seguir selecione a tabela **Cliente**

2- Altere a opção de exibição para **Details** conforme figura ao lado

3- A seguir abra o formulário **Clientes.vb** e arraste a tabela **Clientes** da janela **DataSource** para o formulário. Após os ajustes de tamanho nos campos você deverá obter o formulário conforme a figura a seguir:



Note que o tipo do campo Nascimento usa um controle **DateTimePicker**.

Pronto ! acabamos de implementar o formulário de cadastramento de clientes usando os novos recursos do VB2005.

Implementando o formulário para cadastrar Filmes : Filmes.vb

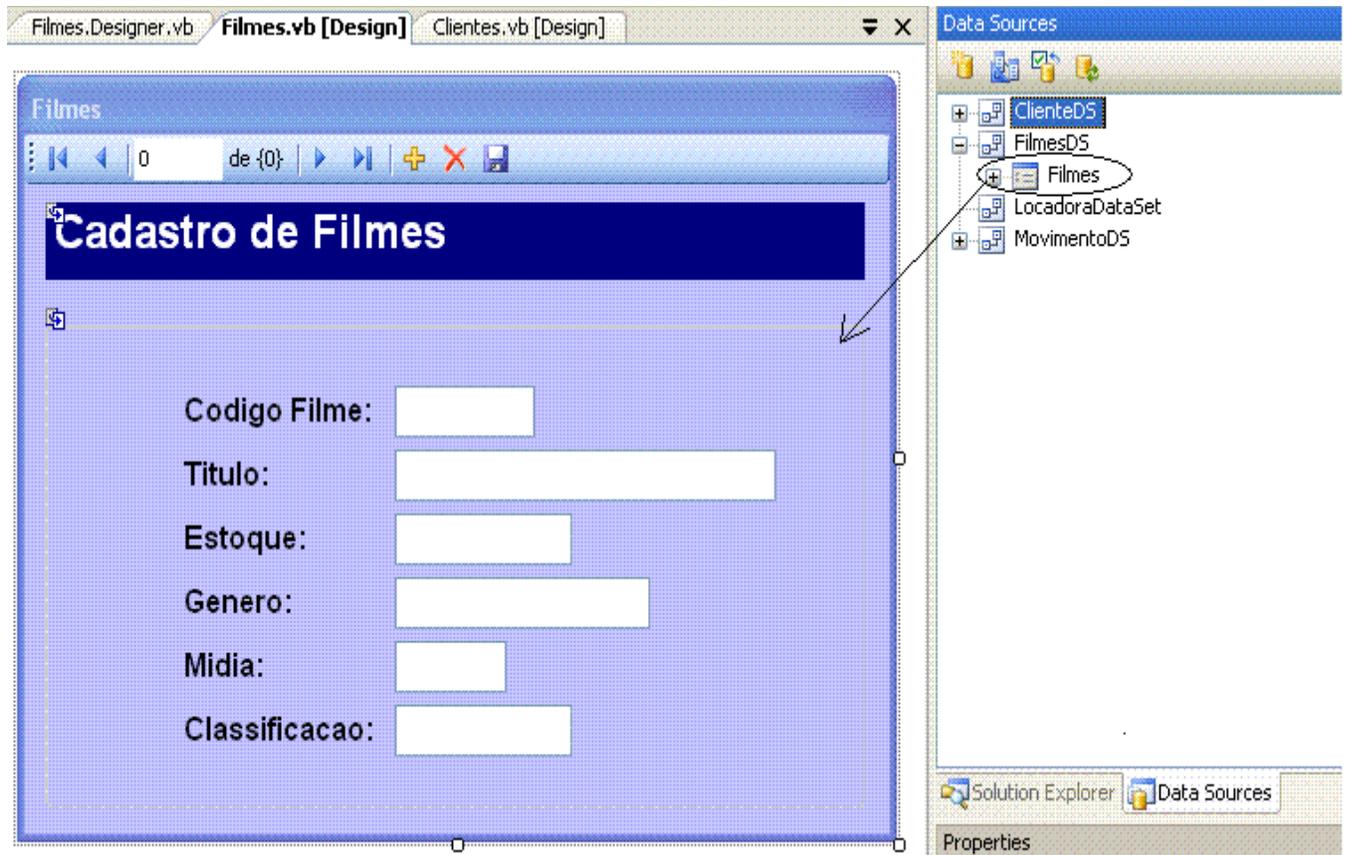
A implementação deste formulário é idêntico ao formulário para cadastrar Clientes.

1- Crie o formulário Filmes.vb incluindo um novo formulário no projeto (**Add -> New Item**) e a seguir , conforme já explicado no segundo artigo , use a identidade visual herdada do formulário modelo **FormularioModelo.vb**.

2- Na janela **DataSource**, expanda o DataSet **FilmesDS** e a seguir selecione a tabela **Filmes**

2- Altere a opção de exibição para **Details**

3- Abra o formulário **Filmes.vb** e arraste a tabela **Filmes** para o formulário. Após alguns ajustes você deverá obter o resultado exibido na figura abaixo:



Implementando o formulário para cadastrar Categorias : Categorias.vb

Tendo sempre em mente que a aplicação Locadora de Filmes, objeto desta série de artigos, tem o propósito de mostrar os novos recursos do VB2005 para desenvolver aplicações Windows Forms, estou fazendo muitas simplificações e adotando algumas estratégias com objetivo didático que não seriam usadas em um aplicação real de produção.

Vamos criar uma tabela chamada **Categorias** que irá armazenar os dados das categorias dos filmes. (Veja como fazer isto no primeiro artigo desta série). A tabela deverá possuir os seguintes campos e propriedades:

Column Name	Data Type	Allow Nulls
Codigo	int	<input type="checkbox"/>
Descricao	nchar(30)	<input type="checkbox"/>
Estado	bit	<input type="checkbox"/>

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

Campos da tabela
Categorias:

1. Código
2. Descrição da
Categoria

3. Estado
(ativo/inativo)

Agora inclua um novo formulário chamado **Categorias.vb** no seu projeto herdando do formulário base **FormularioModelo.vb**.

O próximo passo é criar um novo Data Source para a tabela **Categorias** recém criada. Clique com o botão direito do mouse na janela Data Sources e selecione **Create new Data Source**. Na próxima janela selecione DataBase e clique em **Next>**. Aceite a conexão que já foi criada e clique em **Next>**. Selecione a tabela **Categorias** e informe o nome **CategoriasDS** clicando a seguir em **Finish**.

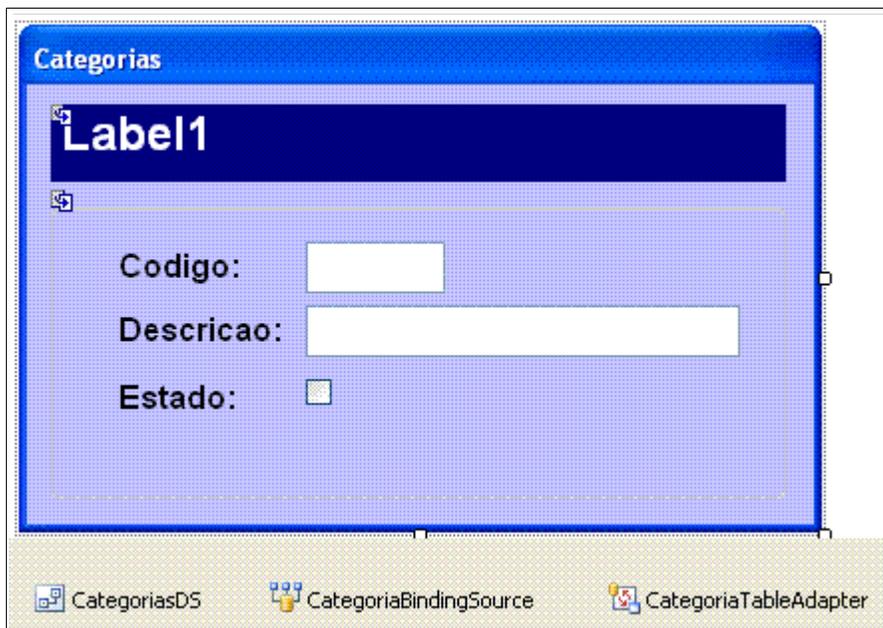
Se você seguiu as orientações terá a seguinte visão do seu projeto:

The screenshot shows the Visual Studio IDE. On the left, the 'Categorias.Designer.vb' form is open, displaying a 'Label1' control. On the right, the 'Data Sources' window is visible, showing a tree view of data sources: 'CategoriasDS', 'Categoria', 'ClienteDS', 'Cliente', 'FilmesDS', 'Filmes', 'MovimentoDS', and 'Movimento'.

Vamos alterar o modo de visão do Data Source **CategoriasDS** para **Details** e arrastá-lo para o formulário **Categorias.vb**.

Quando fazemos esta operação o Assistente, além de criar os campos no formulário e incluir os objetos : [CategoriaBindingSource](#), [CategoriaTableAdpater](#) e [CategoriasBindingNavigator](#), cria também uma barra de navegação que permite, além da navegação pelos registros, realizar as operações CRUD (**read, update e delete**) na fonte de dados.

Não vamos usar esta barra de navegação, vamos excluí-la do formulário de forma a ficar com o seguinte layout:



Se olharmos, neste momento, o código do formulário iremos ver:

```
Public Class Categorias

Private Sub CategoriaBindingNavigatorSaveItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Me.Validate()
Me.CategoriaBindingSource.EndEdit()
Me.CategoriaTableAdapter.Update(Me.CategoriasDS.Categoria)
End Sub

Private Sub Categorias_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
'TODO: This line of code loads data into the 'CategoriasDS.Categoria' table. You can
move, or remove it, as needed.
Me.CategoriaTableAdapter.Fill(Me.CategoriasDS.Categoria)
End Sub
End Class
```

Acima temos o código do botão Salvar da barra de Navegação e o código que carrega todos as categorias na carga do formulário.

Vamos excluir todo este código do formulário e incluir dois botões de comando : **Criar e Cancelar**, conforme abaixo:

No evento **Load** do formulário inclua o seguinte código:

```
Private Sub Categorias_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    Me.CategoriaBindingSource.AddNew()
End Sub
```

Este código inclui um novo registro no Data Source.

No evento **Click** do botão Criar insira o código abaixo:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    If Me.Validate() Then
        Me.CategoriaBindingSource.EndEdit()
        Me.CategoriaTableAdapter.Update(Me.CategoriasDS.Categoria)
    Else
        System.Windows.Forms.MessageBox.Show(Me, "Erro")
    End If
    Me.Close()
End Sub
```

Este código atualiza a tabela Categorias incluindo os dados digitados.

Como o campo **Codigo** da tabela **Categorias** é **Identity** com incremento automático de 1 em 1 iremos desabilitar este campo no formulário definindo para controle **TextBox** - **CodigoTextBox** a propriedade **Enabled=False** e **TabStop=False**.

Além disto vamos validar o campo descrição não permitindo que o mesmo esteja vazio. Para isto inclua o seguinte código no evento **Validating** do **TextBox**:

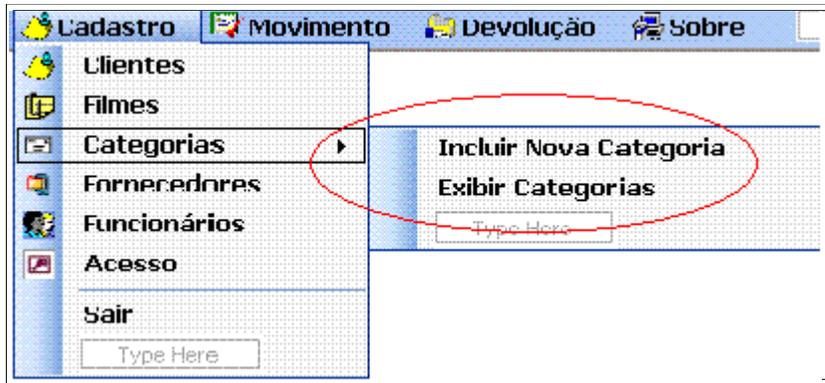
```
Private Sub DescricaoTextBox_Validating(ByVal sender As Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles DescricaoTextBox.Validating
If Me.DescricaoTextBox.Text.Length = 0 Then
    e.Cancel = True
    ErrorProvider1.SetError(DescricaoTextBox, "Informe a descrição da categoria.")
Else
    ErrorProvider1.SetError(DescricaoTextBox, Nothing)
End If
End Sub
```

Para que o código acima funcione corretamente você deve incluir um controle **ErrorProvider** no seu formulário.

Podemos querer exibir em um Grid todas as categorias cadastradas, para isto vamos criar um formulário chamado **TodasCategorias.vb**, e, arrastar o data source **CategoriasDS** no formato **GridView** para o formulário. O resultado será o exibido na figura abaixo:



Teremos que fazer um ajuste no menu permitindo duas opções para **Categorias**: **Incluir Nova Categoria e Exibir Categorias**.



Não esqueça de incluir o código abaixo para exibir os formulários a partir da seleção do usuário via Menu:

```
Private Sub IncluirNovaCategoriaToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles IncluirNovaCategoriaToolStripMenuItem.Click

My.Forms.Categorias.MdiParent = Me
My.Forms.Categorias.Show()

End Sub

Private Sub ExibirCategoriasToolStripMenuItem_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ExibirCategoriasToolStripMenuItem.Click

My.Forms.TodasCategorias.MdiParent = Me
My.Forms.TodasCategorias.Show()

End Sub
```

Implementando o formulário para registrar as locações : Movimento.vb

A implementação do formulário de locação deve registrar as locações efetuadas pelos clientes. No nosso caso o processo de locação funciona da seguinte forma:

1. O cliente escolhe o filme que deseja do acervo e se dirige ao funcionário
2. O funcionário acessa o sistema e informa o código do filme escolhido e o código do cliente que esta efetuando a locação
3. Como geralmente o cliente não memoriza o seu código devemos implementar uma alternativa para selecionar o cliente pelo nome

Vamos recordar a estrutura da tabela **Movimento** para ver os dados que devemos registrar:

- O campo **CodigoLocacao** é um campo autonumerado (**Identity**) e por este motivo não iremos gravar nada neste campo visto que seu valor é incrementado automaticamente

Column Name	Data Type	Allow Nulls
CodigoLocacao	int	<input type="checkbox"/>
CodigoCliente	int	<input type="checkbox"/>
CodigoFilme	int	<input type="checkbox"/>
Locacao	datetime	<input type="checkbox"/>
Devolucao	datetime	<input checked="" type="checkbox"/>
Valor	money	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Devemos registrar valores nos seguintes campos:

- **CodigoCliente** - este dado virá da tabela Clientes
- **CodigoFilme** - o valor virá da tabela Filmes
- **Locacao** - a data de locação (a data atual do sistema)

- Os campos **Devolucao** e **Valor** serão preenchidos quando da devolução do filme pelo cliente (por este motivo aceitam valores **Nulls**, pois não iremos gravar nada na locação nestes campos)

Abra o formulário **Movimento.vb** e inclua dois controles Labels , dois controles TextBox e quatro botões de comando(Button) conforme a figura abaixo:

O que pretendendo aqui é que quando o for efetuar a locação o funcionário irá informar o código do filme e o código do cliente. Se por algum motivo estes dados não estiverem disponíveis implementamos uma busca nas tabelas **Filmes** e **Clientes** que nos retornarão estes valores.

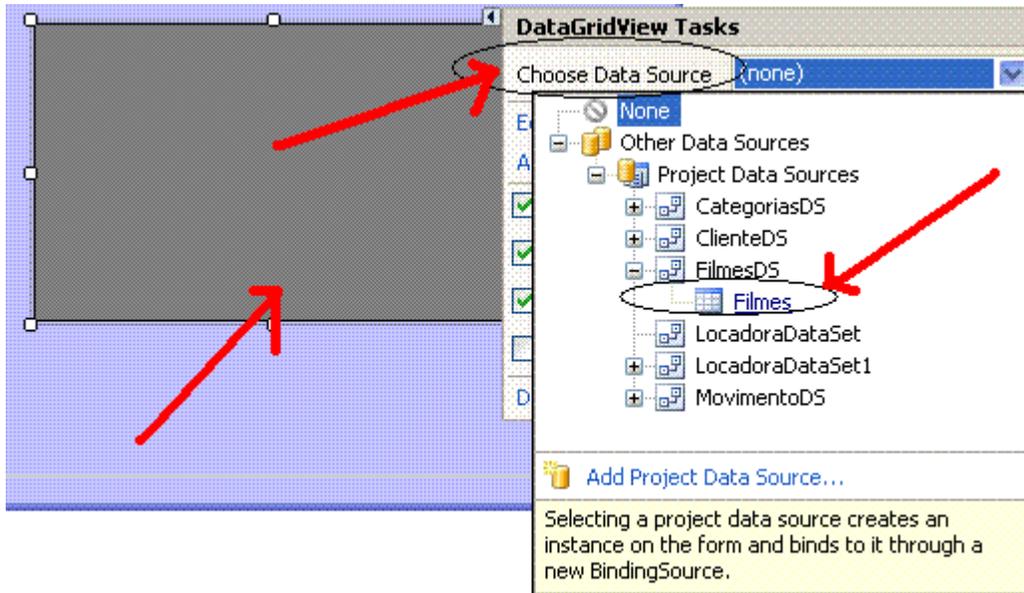
Para implementar este recurso deveremos criar dois novos formulários : **SelecionaFilme.vb** e **SelecionaCliente.vb** que atuarão basicamente de forma idêntica : irão exibir uma caixa de texto, onde o usuário informará o nome do filme ou do cliente, um botão selecionar que deverá extrair os dados da respectiva tabela, e

exibir o resultado da consulta em um [DataGridView](#). A seguir os passos para implementar o recurso para o formulário **SelecionaFilme.vb**.

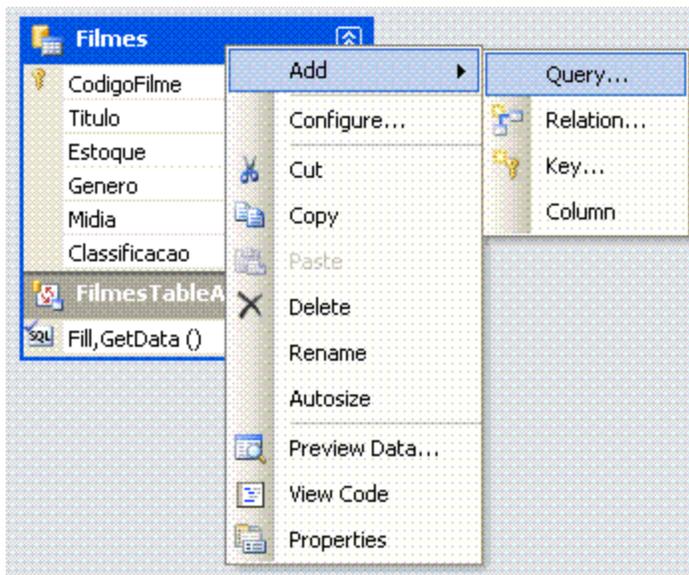
1. Inclua um novo formulário chamado **SelecioneFilme.vb** na sua aplicação herdando a identidade visual do formulário `FormularioModelo.vb`
2. Inclua no formulário os seguintes controles : [Label](#), [TextBox](#) , [Button](#) e [DataGridView](#), conforme abaixo:



3. Quando da inclusão do `DataGridView` , usando as [Smart Tags](#) , selecione o Data Source - **FilmesDS** - conforme a figura abaixo

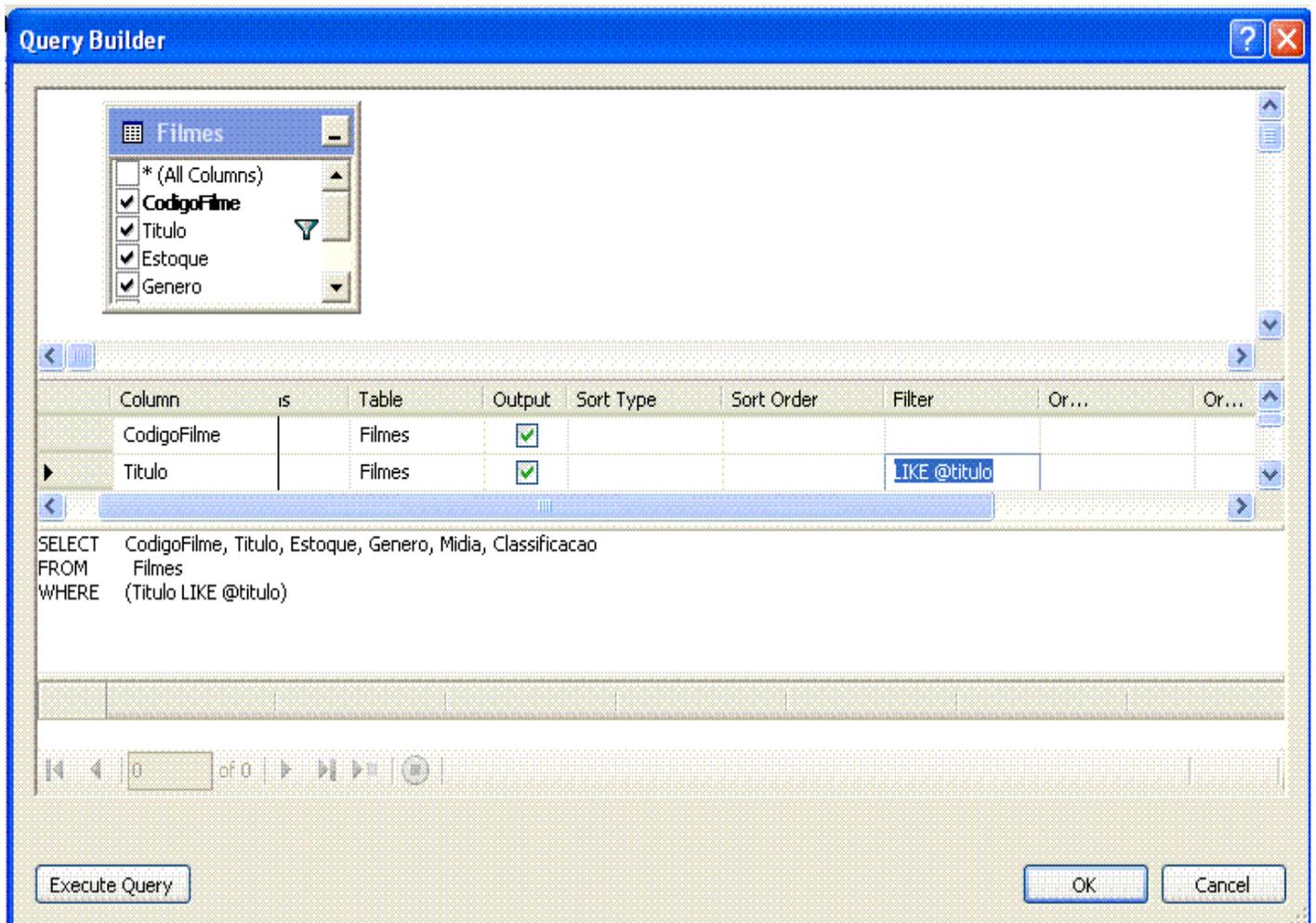


4. Como queremos exibir somente o filme correspondente ao nome informado pelo funcionário devemos incluir um filtro na nossa fonte de dados - Data Source - **FilmesDS**.
5. Na guia Data Source , seleccione **FilmesDS** e clique com o botão direito seleccionando - **Edit Data Set with Designer**
6. A seguir seleccione o Data Set **Filmes** e clique com o botão direito do mouse seleccionando **Add -> Query**

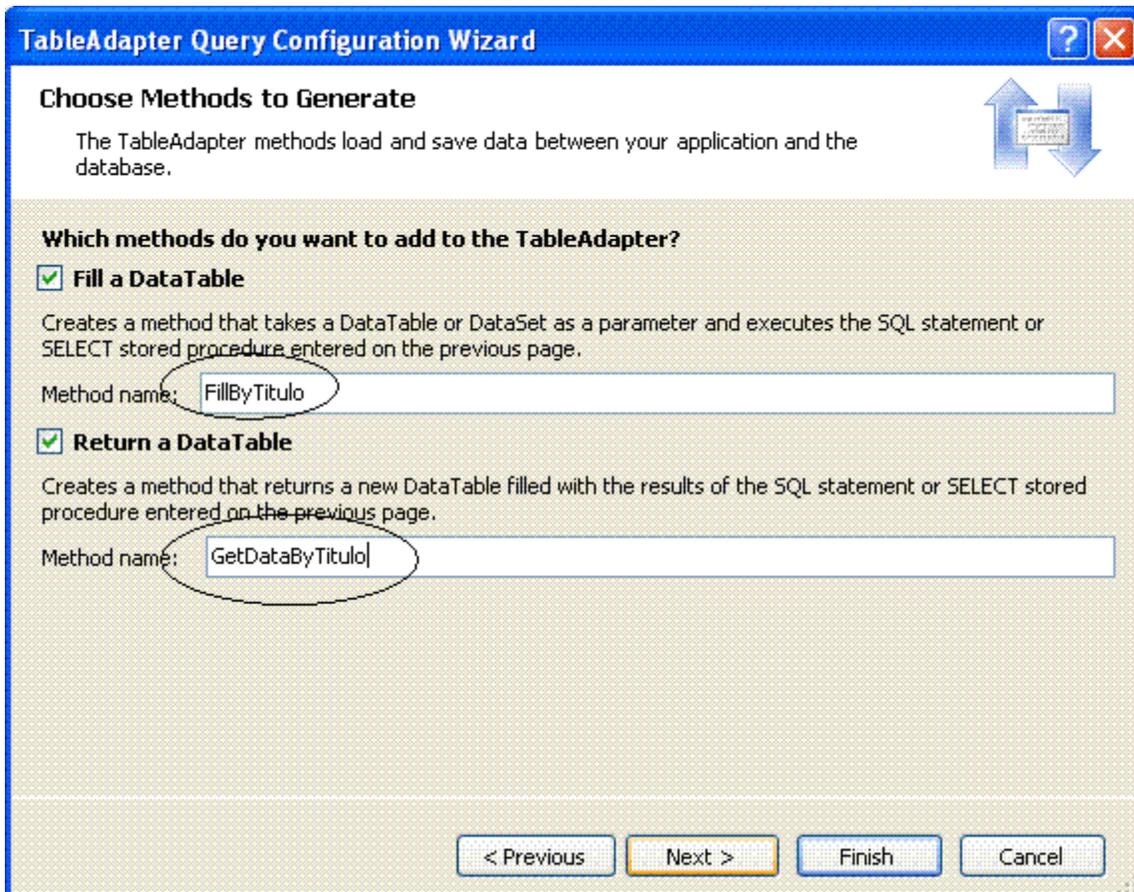


7. Na janela - **TableAdapter Configuration Wizard** - vamos usar uma instrução SQL **Select** , por isto clique no botão **Next>**
8. Como vamos obter linhas de retorno, aceite a seleção do wizard e clique no botão **Next >** novamente

9. Na janela a seguir vamos montar a nossa consulta , por isto clique no botão - **Query Builder**.
10. Estando na janela **Query Builder** selecione o campo **titulo** e na coluna **Filter** informe o parâmetro : **Like @titulo**



11. Podemos testar a consulta clicando no botão **Execute Query**.
12. Clique no botão **Ok** e a seguir informe o nome da consulta criada:



13. A seguir clique em **Next>** e finalmente em **Finished**. Pronto criamos uma consulta para selecionar o filme pelo título informado.

Podemos retornar ao formulário **SelecionaFilme**. Ao clicar no botão **Seleciona** devemos acionar a consulta criada, para isto inclua o seguinte código no evento **Click** do botão:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles Button1.Click

    Me.FilmesTableAdapter.FillByTitulo(FilmesDS.Filmes, "%" & TextBox1.Text & "%")

End Sub
```

Obs: Você deve remover qualquer outro código gerado pelo assistente no formulário.

Neste código eu estou usando do **TableAdapter** e a consulta **FillByTitulo** para extrair dados do dataset **Filmes** segundo o valor informado na caixa de texto **TextBox1.Text**. (eu sugiro que você forneça nomes sugestivos aos controles)

Devemos ainda configurar algumas propriedades do **DataGridView** pois não desejamos que o usuário altere o **DataGridView** ou inclua, exclua ou efetue uma seleção múltipla

no mesmo. Para isto defina como **False** as seguintes propriedades : [MultiSelect](#) , [AllowUserAddRows](#), [AllowUserDeleteRows](#), [AllowUserResizeRows](#) e [AllowUserResizeColumns](#). Vamos definir a propriedade [SelectionMode](#) como **FullRowSelect** para selecionar a linha por inteiro.

Quando o usuário clicar na célula que exibe os dados do título procurado devemos capturar a seleção e os dados do filme escolhido. Fazemos isto tratando o evento **Click** do botão **Aceitar**. Abaixo temos o código que deve ser incluído neste evento:

```
Private Sub btnAceita_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles btnAceita.Click

    codigoFilme = Me.FilmesDS.Filmes(FilmesBindingSource.Position).CodigoFilme
    Me.Close()

End Sub
```

Aqui definimos a variável pública **codigoFilme** no formulário e obtemos o código do filme pois é este valor que vamos gravar na tabela **Movimento**.

Agora voltemos ao formulário Movimento.vb , onde tudo começou. Após efetuar a busca e selecionar o filme pelo título deveremos obter o código do filme e exibir na caixa de texto - **txtCodigoFilme** - do formulário de Locações - **Movimento.vb**. Para isto basta acrescentar o código abaixo no evento **Click** do botão Procurar do formulário:

```
Private Sub btnProcuraFilme_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnProcuraFilme.Click

    My.Forms.SelecionaFilme.ShowDialog()
    txtCodigoFilme.Text = My.Forms.SelecionaFilme.codigoFilme

End Sub
```

A primeira linha abre o formulário para selecionar o filme e a segunda obtém o valor da variável **codigoFilme** declarada como [Public](#) no formulário **SelecionaFilme**.

Todo o procedimento acima deverá ser repetido para o formulário **SelecionaCliente**. Deixo isto como tarefa para você realizar.

O código final do formulário, antes de efetuar o registro da locação, será o seguinte:

```
Public Class Movimento

    Private Sub btnProcuraFilme_Click(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles btnProcuraFilme.Click

        My.Forms.SelecionaFilme.ShowDialog()
        txtCodigoFilme.Text = My.Forms.SelecionaFilme.codigoFilme

    End Sub
```

```
Private Sub btnProcuraCliente_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnProcuraCliente.Click

My.Forms.SelecionaCliente.ShowDialog()
txtCodigoCliente.Text = My.Forms.SelecionaCliente.codigoCliente

End Sub

End Class
```

Para efetuar o registro da locação vou criar uma classe que será responsável por esta tarefa. Mas isto é assunto para o próximo artigo.

Parte 5

1. Continuando o desenvolvimento do nosso projeto Locadora de Filmes no *Visual Basic 2005 Express Edition* este artigo irá continuar a mostrar a implementação da interface do usuário.

Neste artigo vou mostrar como implementar a classe para efetuar o registro da locação dos filmes da locadora de filmes. Lembrando que já foram implementadas as seguintes funcionalidades:

1. **Cadastrar clientes - Clientes.vb**
2. **Cadastrar Filmes - Filmes.vb**
3. **Cadastrar Categorias - Categorias.vb**

Implementando a classe para registrar a locação efetuada no formulário: **Movimento.vb**

Apenas para recordar o formulário **Movimento.vb** possui o seguinte layout:



The screenshot shows a Windows-style application window titled "Movimento". Inside the window, there is a dark blue header bar with the text "Locação de Filmes" in white. Below the header, the form area has a light blue background. It contains two rows of input fields, each with a "Procurar" button to its right. The first row is labeled "Código do Filme" and the second row is labeled "Código do Cliente". At the bottom of the form, there are two buttons: "Cancela" and "Registra".

A estrutura da tabela **Movimento** é a seguinte:

- O campo **CodigoLocacao** é um campo autonumerado (**Identity**) e por este motivo não iremos gravar nada neste campo visto que seu valor é incrementado automaticamente

Column Name	Data Type	Allow Nulls
CodigoLocacao	int	<input type="checkbox"/>
CodigoCliente	int	<input type="checkbox"/>
CodigoFilme	int	<input type="checkbox"/>
Locacao	datetime	<input type="checkbox"/>
Devolucao	datetime	<input checked="" type="checkbox"/>
Valor	money	<input checked="" type="checkbox"/>

Devemos registrar valores nos seguintes campos:

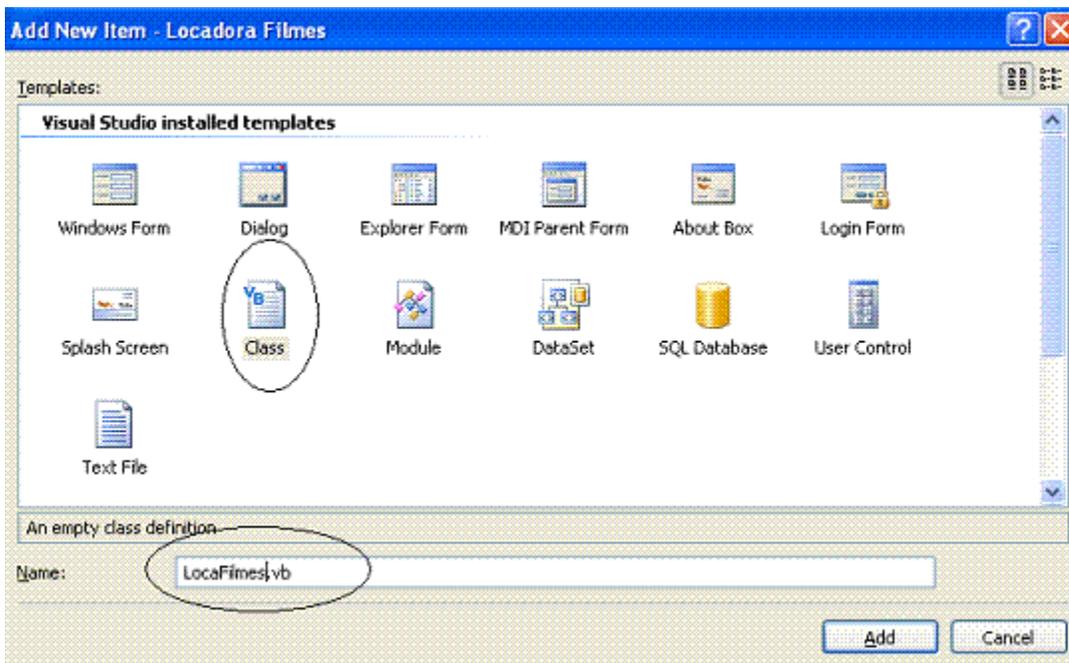
- **CodigoCliente** - este dado virá da tabela Clientes
- **CodigoFilme** - o valor virá da tabela Filmes
- **Locacao** - a data de locação (a data atual do sistema)

- Os campos **Devolucao** e **Valor** serão preenchidos quando da devolução do filme pelo cliente (por este motivo aceitam valores **Nulls**, pois não iremos gravar nada na locação nestes campos)

Uma primeira pergunta que talvez possa surgir seria: Porque eu não implemento o código de locação no próprio formulário ?

Bem , primeiro porque a locação refere-se a ao negócio da locadora de filmes e por este motivo é bom separar a regra de negócio da interface do usuário.

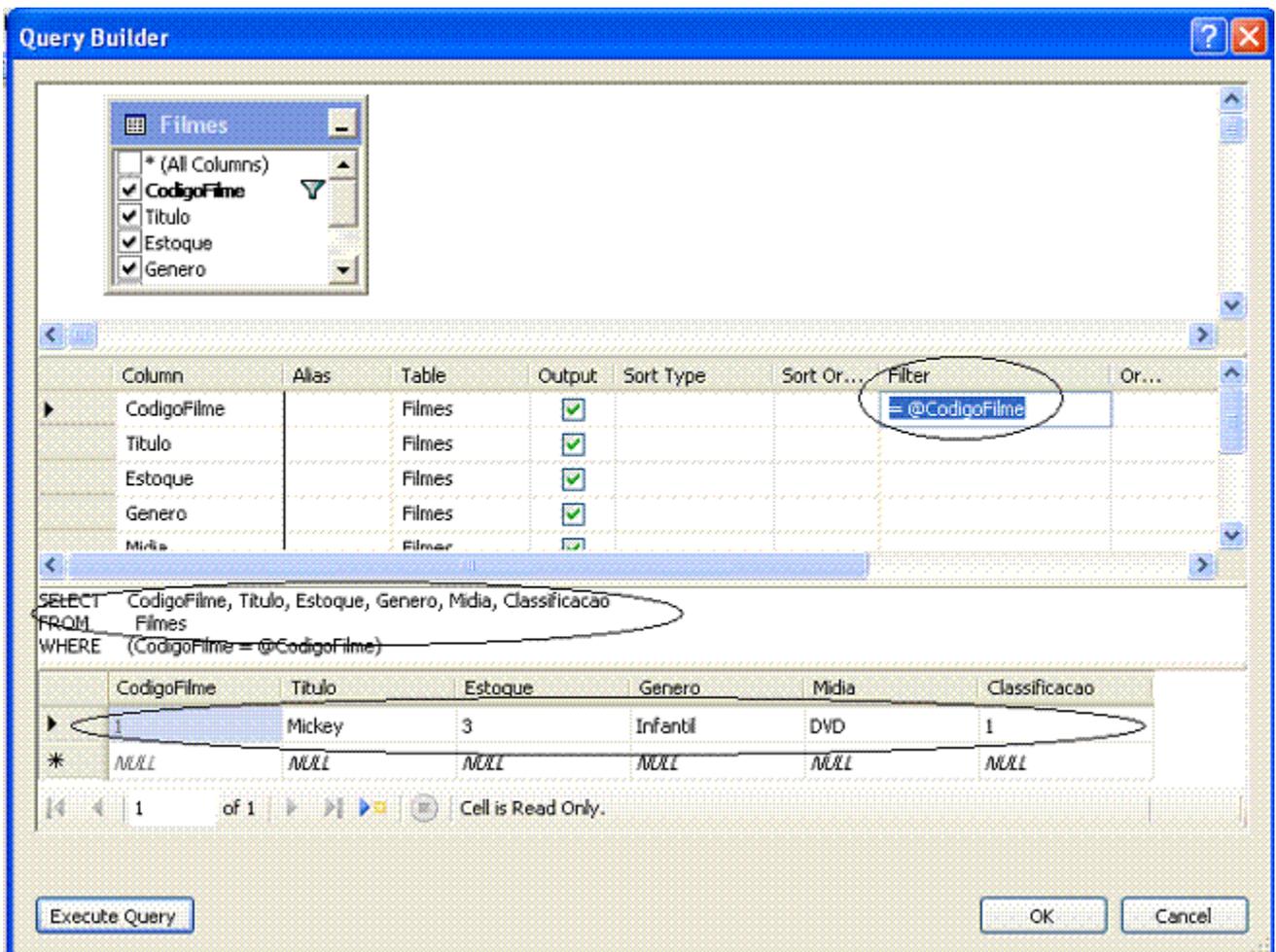
Para criar a classe **LocaFilmes.vb** clique com o botão direito do mouse sobre o nome do projeto e selecione a opção **Add** e a seguir **New Item**; informando a seguir o nome do arquivo e clicando no botão **Add**.



É claro que antes de efetuar a locação eu preciso verificar o estoque de filmes para ver se existe o filme disponível. Se houver terei que diminuir uma unidade , referente a unidade locada e acrescentá-la a tabela de locações efetuadas : **Movimento**.

Antes de entrar no código teremos que criar uma nova consulta relacionada com a tabela **Filmes** que nos devolva um determinado filme conforme o seu código.

Na janela Data Source vamos editar o DataSet **FilmeDS** e vamos incluir uma nova consulta, conforme já vimos nos artigos anteriores, para filtrar os filmes selecionados pelo código do filme. Usando o **Query Builder** informamos o parâmetro **@codigoFilme** para montar a instrução SQL conforme abaixo:



Ao salvar a consulta acrescente o sufixo **Codigo** conforme tela da figura 1.0 abaixo. Na figura 2.0 vemos o **DataSet** exibindo a nova consulta criada:

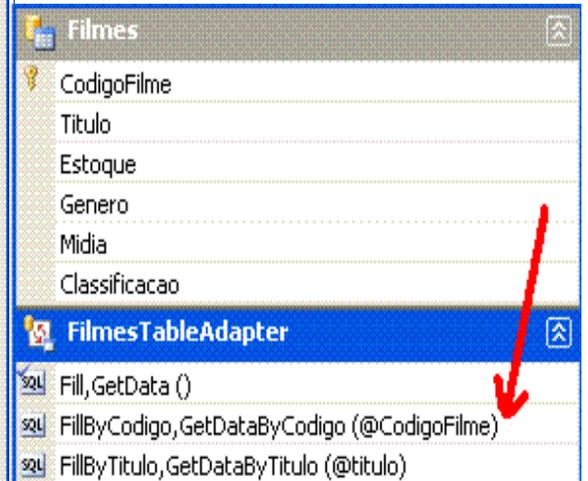
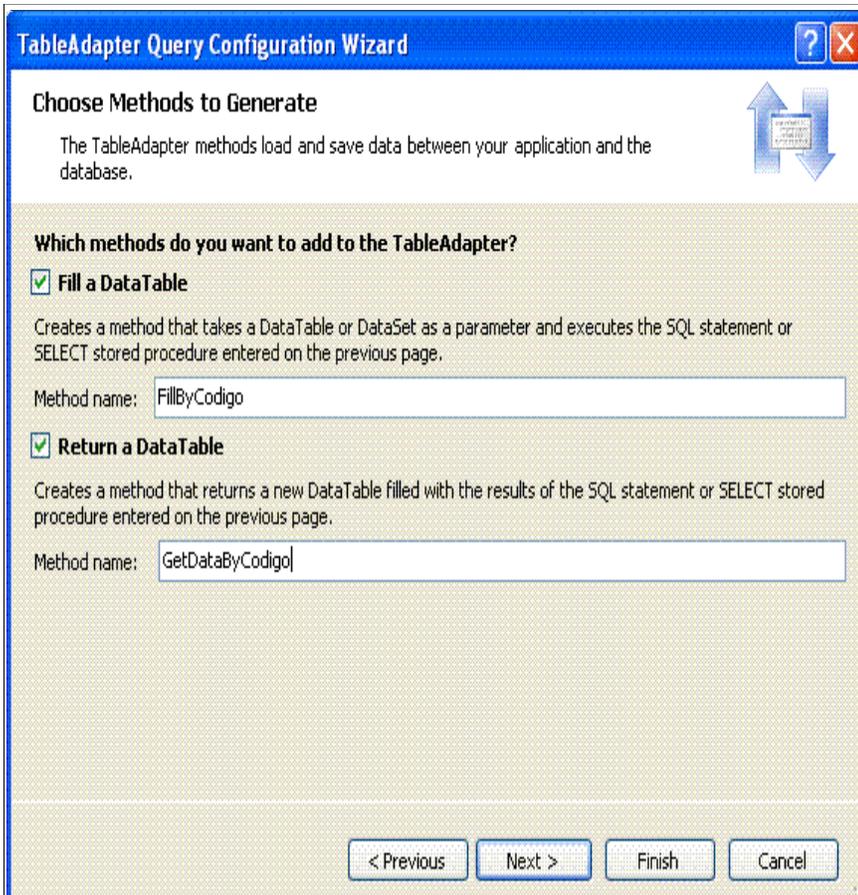
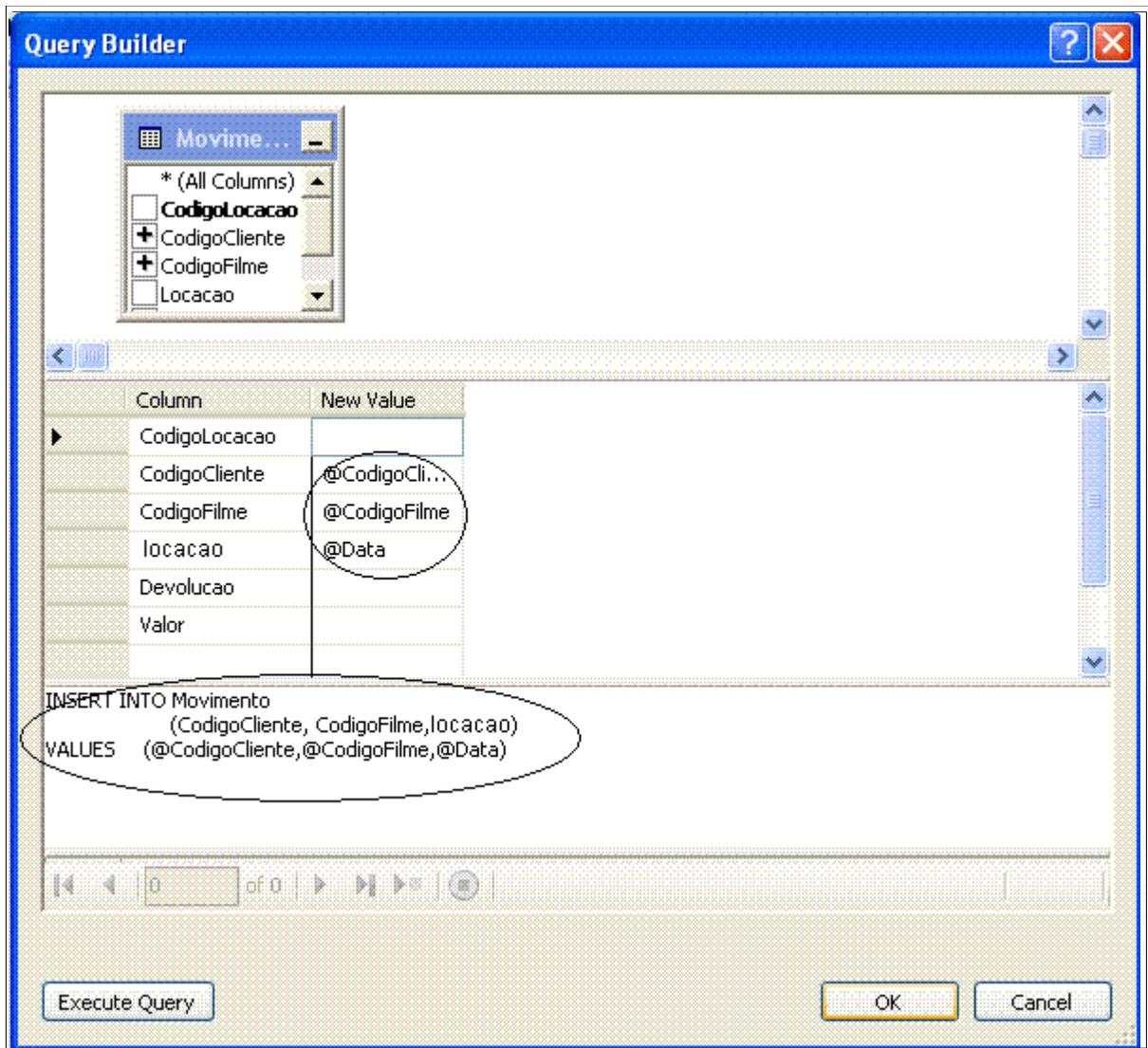


Figura 1.0 - Nomeando a instrução SQL para selecionar dados

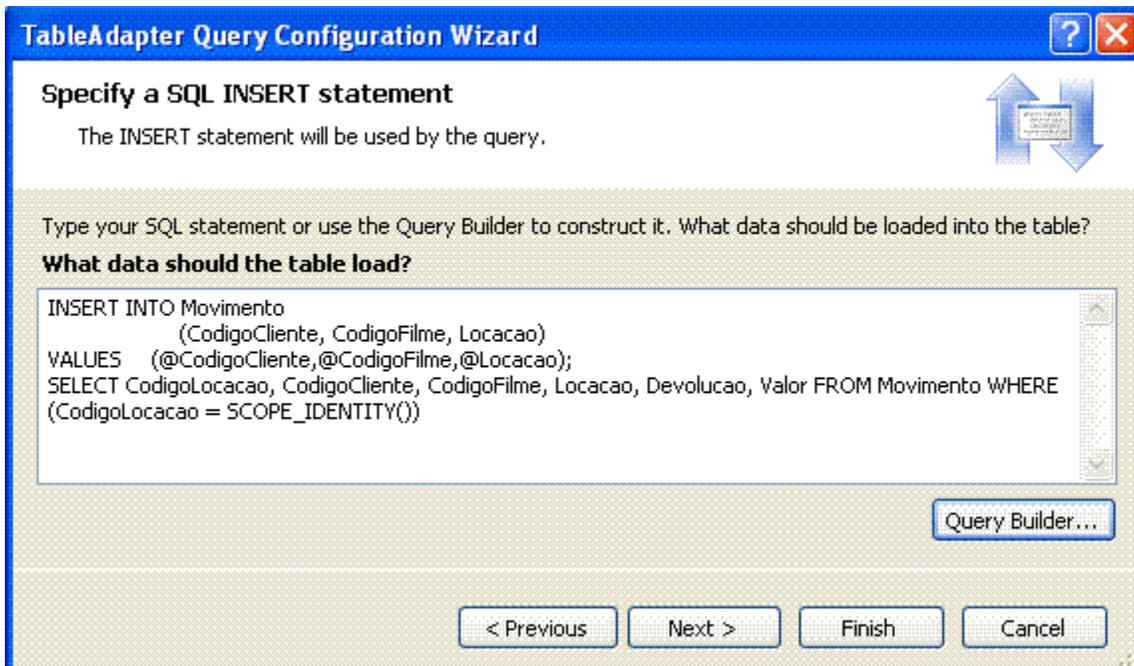
Figura 2.0 - O dataset exibindo a consulta criada

Vamos precisar criar também uma nova consulta de inclusão de dados na tabela **Movimento** usando a instrução SQL **INSERT/INTO**. Na janela **Data Sources** clique com o botão direito do mouse sobre o DataSet **MovimentoDS** e selecione - **Edit DataSet with Designer**. A seguir clique no dataset com o botão direito e selecione **Add** e a seguir **Query**.

Usando o **Query Builder** construa a instrução SQL para inserir uma linha na tabela **Movimento** , conforme figura abaixo:



A janela - **TableAdapter Query Configuration Wizard** - deverá exibir a instrução **INSERT INTO** conforme figura abaixo:



Clique em **Next>** e Informe o nome da consulta - **InsertQueryMovimentoLocacao** - conforme figura 3.0. Na figura 4.0 temos o resultado final onde o dataset já exibe a instrução SQL criada.

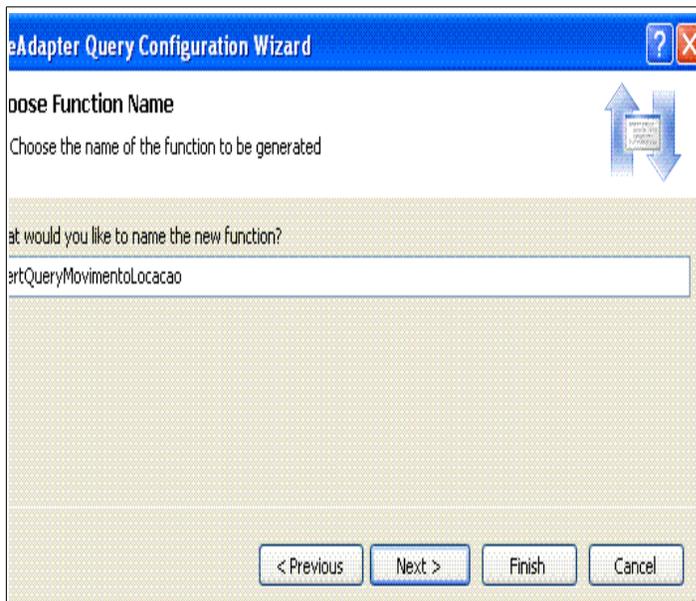


Figura 3.0 - Nomeando a consulta criada

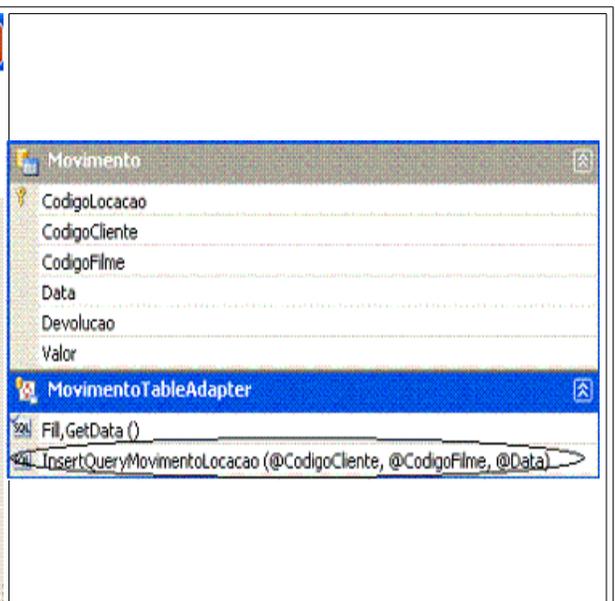


Figura 4.0 - O DataSet exibindo a instrução SQL para incluir dados na tabela

Podemos então partir para o código da classe **LocaFilmes.vb**:

```
Public Class LocaFilmes
    Public Sub alugarFilmes(ByVal codfilme As Integer, ByVal codcliente As
```

```

Integer, ByVal dataLocacao As Date)

'verificando o estoque da locadora
'cria um adapter e um dataset
Dim adapterFilmes As New FilmesDSTableAdapters.FilmesTableAdapter
Dim filme As New FilmesDS

'filtra os filmes pelo código informado usando a consulta SQL criada -
FillByCodigo
adapterFilmes.FillByCodigo(filme.Filmes, codfilme)

'se não encontrou o filme com o código informado dispara exceção
If filme.Filmes.Count = 0 Then
    Throw New ArgumentException("Não existe o filme informado no
acervo.")
End If

'se não tem filme no estoque avisa
If filme.Filmes(0).Estoque = 0 Then
    Throw New ArgumentException("Não existem unidades cadastradas no
acervo.")
End If

'se houver filme no estoque tenho que diminuir a unidade locada do estoque
filme.Filmes(0).Estoque -= 1

'atualiza o banco de dados
adapterFilmes.Update(filme)
filme.AcceptChanges()

'registrar a locação
'cria um adapter e um dataset
Dim adapterLocacao As New
MovimentoDSTableAdapters.MovimentoTableAdapter
Dim movimentoLocacao As New MovimentoDS.MovimentoDataTable

'utiliza a instrução SQL criada para incluir dados na tabela Movimento -
InsertQueryMovimentoLocacao
adapterLocacao.InsertQueryMovimentoLocacao(codcliente, codfilme,
dataLocacao)
adapterLocacao.Update(movimentoLocacao)

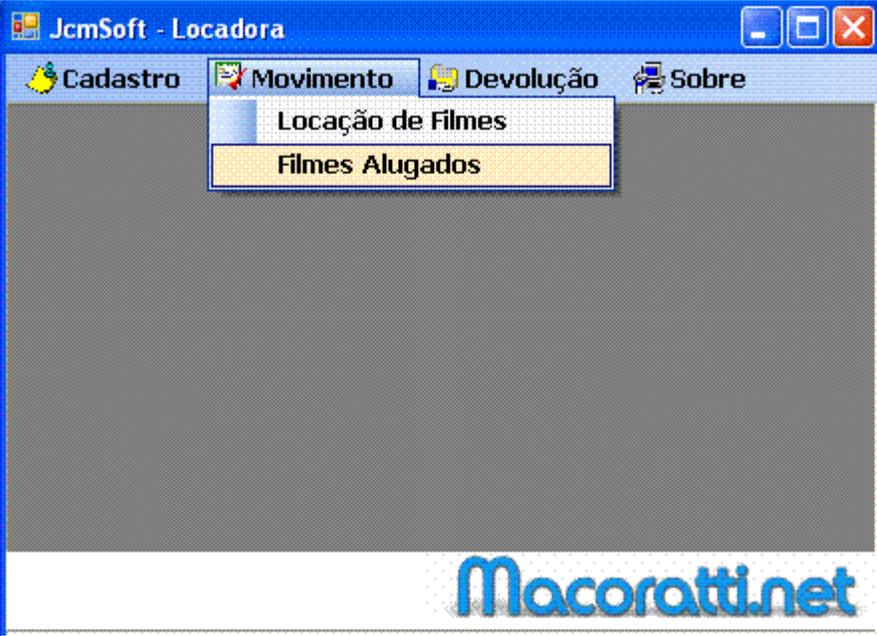
movimentoLocacao.AcceptChanges()
End Sub
End Class

```

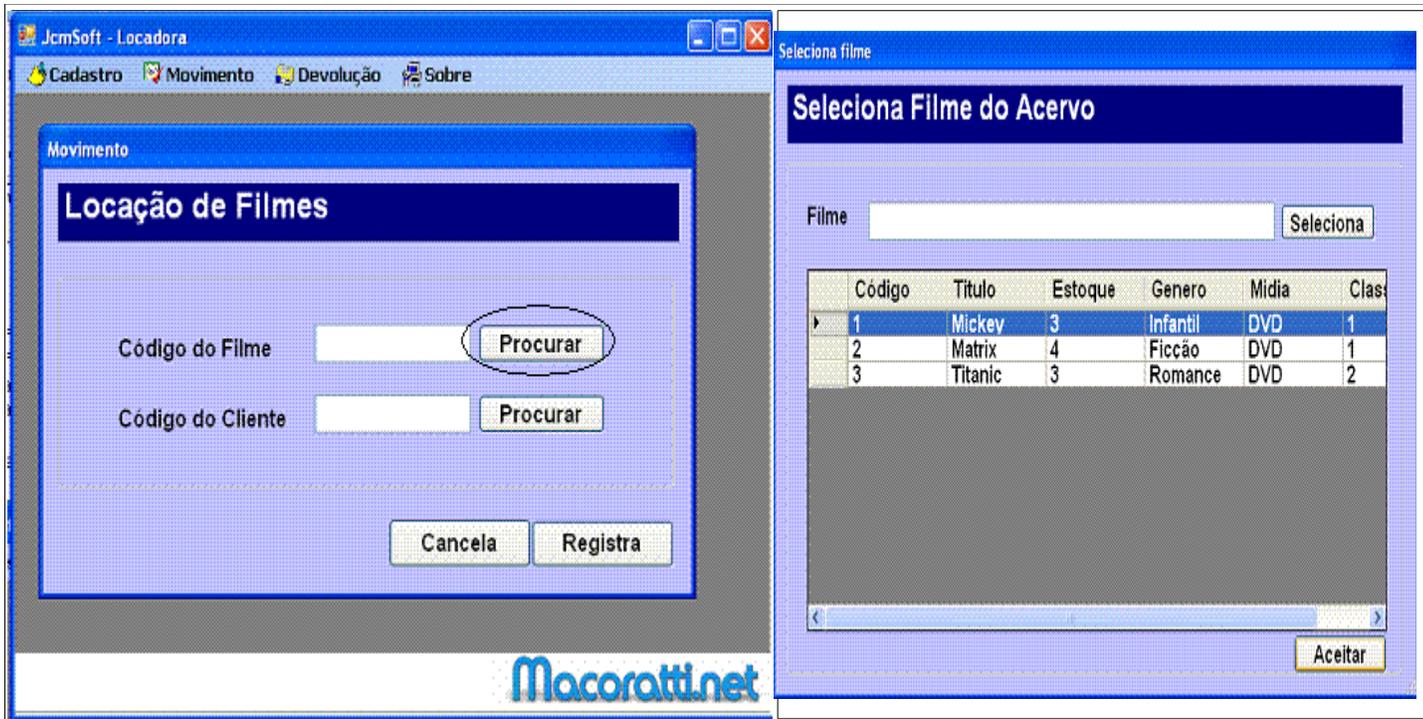
No código da classe acima exibido estou realizando as seguintes tarefas:

- verificando se o filme existe no acervo
- verificando se existem filmes no estoque
- atualizando o estoque de filmes
- atualizando o movimento de locação

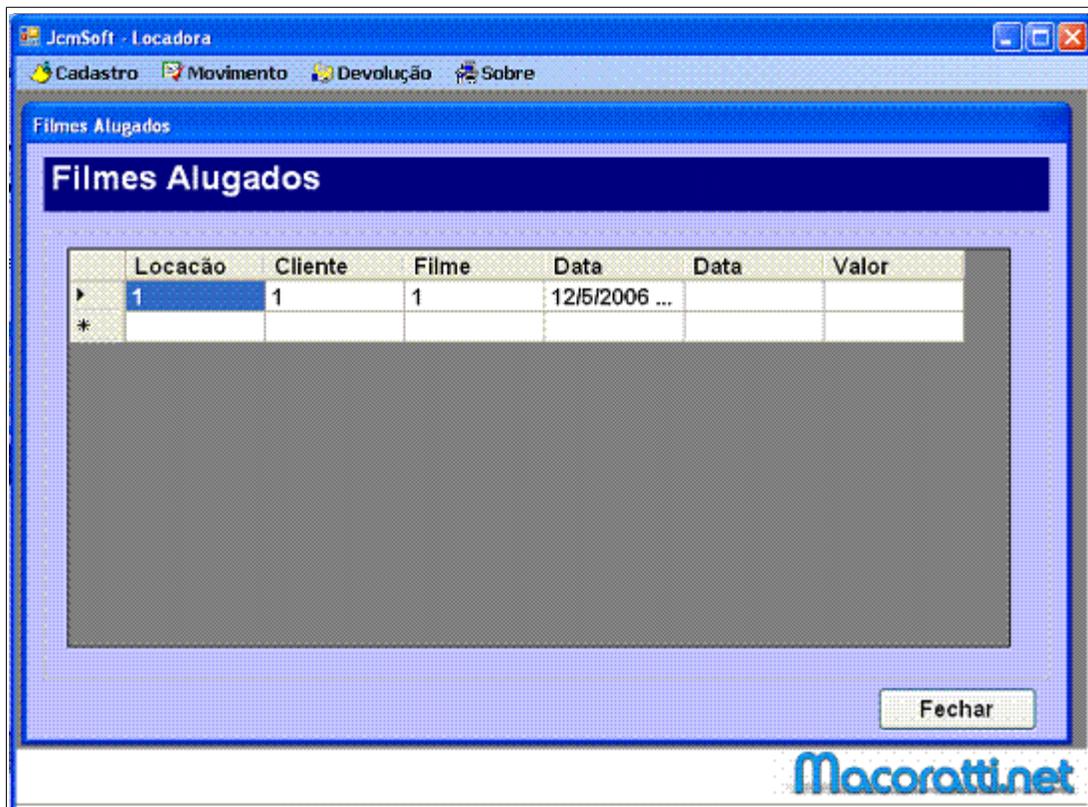
Como o código está comentado vou apenas executar a aplicação e acompanhar de perto o que implementamos para ver se está tudo funcionando:

	<p>- Esta é a tela do Menu Principal</p> <p>- Perceba que na opção do Menu Movimento criamos duas opções:</p> <ul style="list-style-type: none">• Locação de Filmes - para efetuar a locação de filmes• Filmes alugados - para exibir os filmes alugados <p>Nota: Você deverá incluir um novo formulário chamado FilmesAlugados.vb da mesma forma que fizemos com os demais formulários e arrastar a tabela Movimento para o formulário. Após fazer isto exclua o BindingNavigator que o assistente incluiu no formulário pois não iremos efetuar manutenção neste formulário.</p>
---	---

Clicando na opção - **Locação de Filmes**- temos o formulário onde iremos informar o código do filme e o código do cliente. Lembre que implementamos os formulário **SelecionaFilme.vb** e **SelecionaCliente.vb** para podemos selecionar um cliente pelo nome. Abaixo temos a exibição da tela do formulário **Movimento.vb** e de **SelecionaFilme.vb** após clicarmos em Procurar:



Após efetuar a seleção do Filme e do Cliente e clicar no botão registra a locação será efetuada. Podemos ver isto na opção **Filmes Alugados**:



Se consultarmos o estoque de filmes para o filme de código igual a 1 iremos ver que houve uma diminuição de uma unidade do estoque referente a locação efetuada. Quando da devolução deveremos repor o filme no estoque.

The screenshot shows a web browser window with the title 'Filmes'. The address bar shows '1 de 3'. The main content area has a dark blue header with the text 'Cadastro de Filmes'. Below this is a section titled 'Dados dos Filmes do Acervo' containing a form with the following fields:

Codigo Filme:	1
Titulo:	Mickey
Estoque:	2
Categoria:	Infantil
Midia:	DVD
Classificacao:	1

A red circle is drawn around the 'Estoque' field, and a red arrow points to it from the right side of the form.

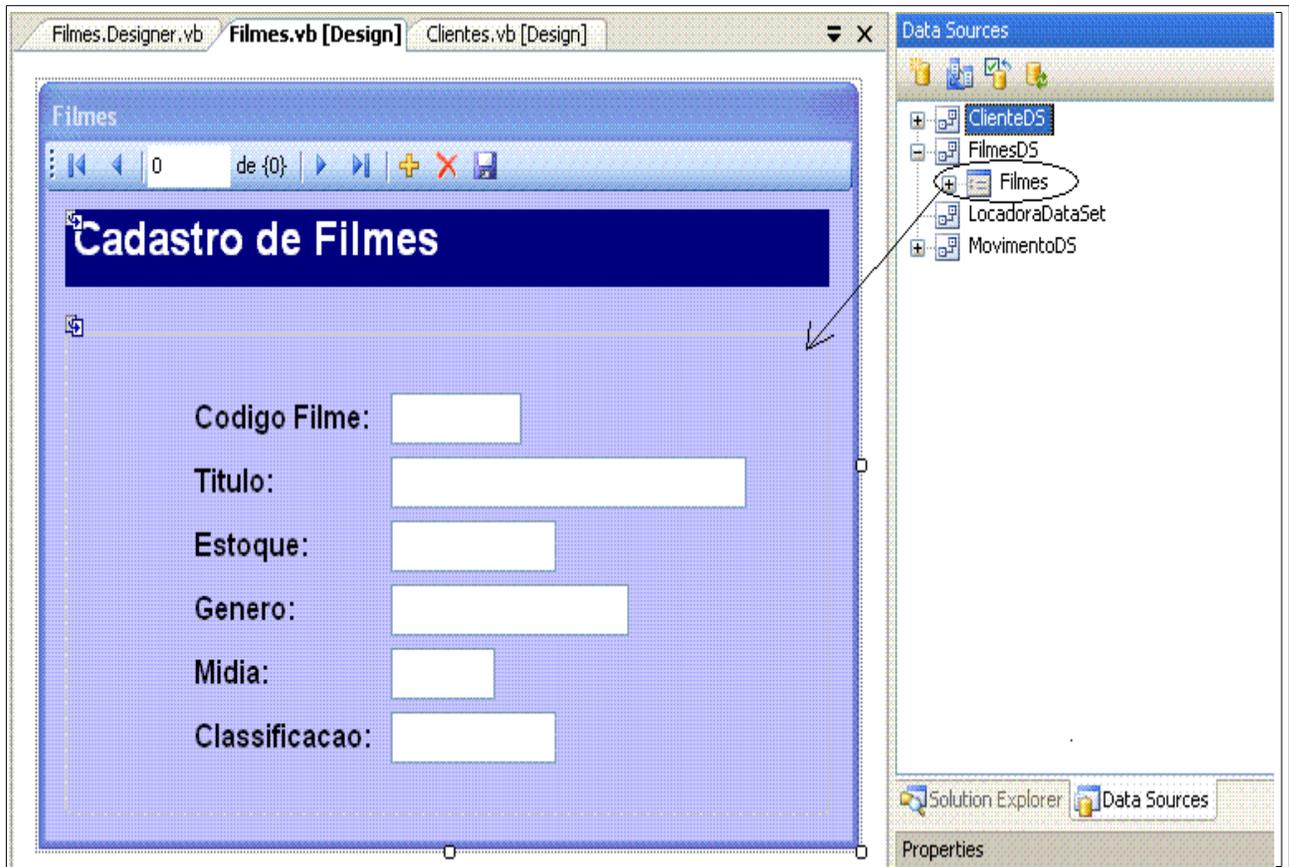
Dica : Sugiro que você implemente uma transação no código para que se algo der errado durante o processo ele não seja concluído.

Parte 6

Neste artigo vou fazer alguns ajustes na interface do usuário já criada, além de criar mais dois formulários nos mesmos padrões já abordados em artigos anteriores da série.

Incrementando o formulário de cadastro de filmes.

Apenas para recordar, o formulário Filmes.vb possui o seguinte layout:



Vamos incrementar a interface com o usuário do formulário de cadastro de filmes usando os novos recursos do VB 2005. Como os dados que o usuário deve informar para os campos Genero, Midia e Classificação devem estar pré-cadastrados em suas respectivas tabelas (Com exceção da campo Midia onde usarei um valor pré-definido) vamos mudar os controles dos campos Genero, Midia e Classificação, do formulário Filmes.vb, de TextBox para ComboBox e fazer a vinculação com as respectivas tabelas. Além disto vou passar a exibir o campo status do filme indicando se o mesmo esta disponível, alugado ou reservado. O novo leiaute deverá ficar como na mostrado na figura a seguir:

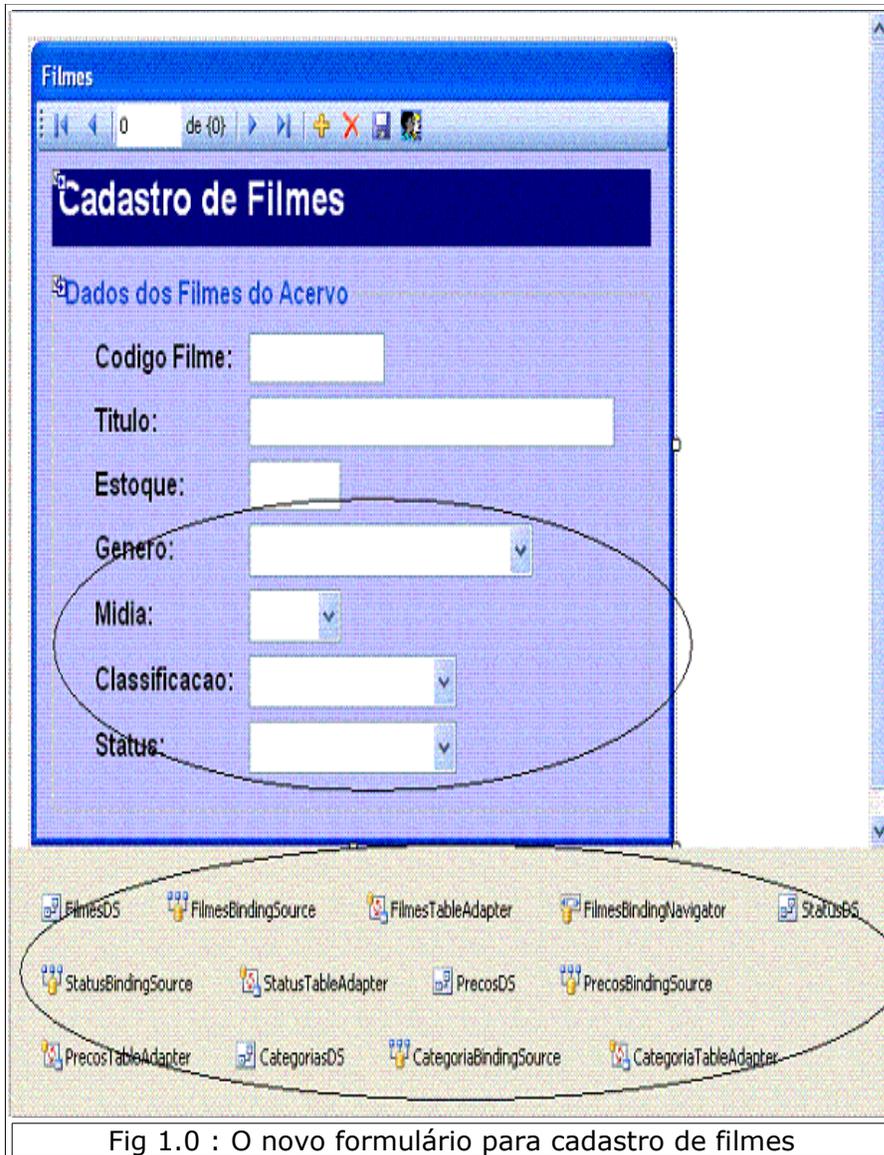


Fig 1.0 : O novo formulário para cadastro de filmes

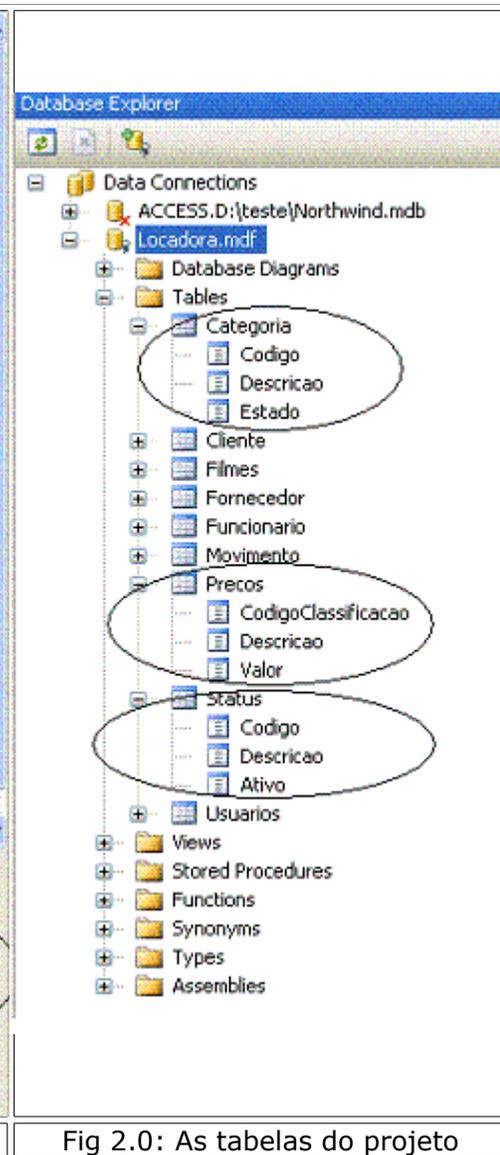


Fig 2.0: As tabelas do projeto

- O campo **genero** deverá exibir os dados da tabela **Categorias**
- O campo **Midia** deverá exibir os valores **DVD e Fita VHS**
- O campo **Classificação** deverá exibir os dados da tabela **Precos**
- O campo **Status** deverá exibir os dados da tabela **Status**

A estrutura das novas tabelas **Precos** e **Status** é mostrada na figura 3.0 abaixo.

dbo.Precos: T...\LOCADORA.MDF)				dbo.Status: T...\LOCADORA.MDF)			
Column Name	Data Type	Allow Nulls		Column Name	Data Type	Allow Nulls	
CodigoClassificacao	int	<input type="checkbox"/>		Codigo	int	<input type="checkbox"/>	
Descricao	nvarchar(50)	<input type="checkbox"/>		Descricao	nvarchar(20)	<input type="checkbox"/>	
Valor	money	<input type="checkbox"/>		Ativo	bit	<input type="checkbox"/>	

Fig 3.0 : Estrutura das tabelas Precos e Status

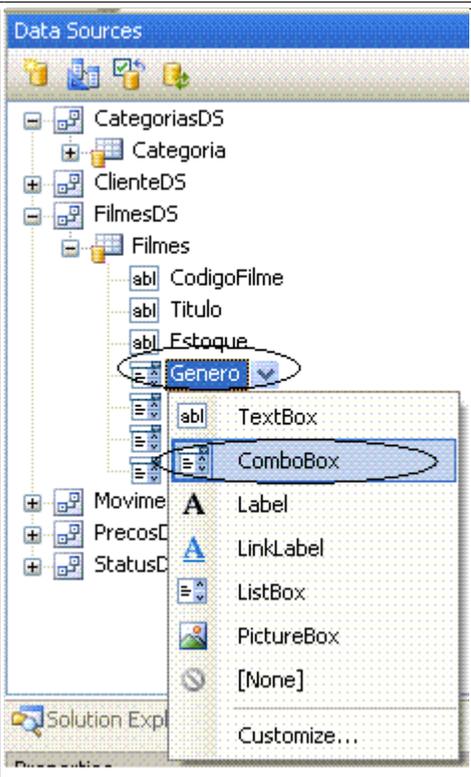
Vejamos como implementar esta funcionalidade no formulário **Filmes.vb**:

1- Primeiro remova os campos do formulário

2- A seguir abra a janela **Data Sources**; selecione e expanda o DataSet **FilmesDS** e exiba os campos da tabela **Filmes**.

2- Selecione o campo que deseja alterar e clique na seta para baixo selecionando a opção **ComboBox** conforme a figura ao lado, repetindo o processo para os demais campos.

3- Arraste os campos com as novas formatações para o formulário novamente.



O próximo passo é definir o preenchimento da **combobox** no formulário; para isto vamos usar o assistente da Smart Tag do controle:

Seleccione o controle e clique na pequena seta superior direita

- Marque a opção : Use **data bound items**

1- Informe o Data Source , no nosso caso - **CategoriaBindingSource**

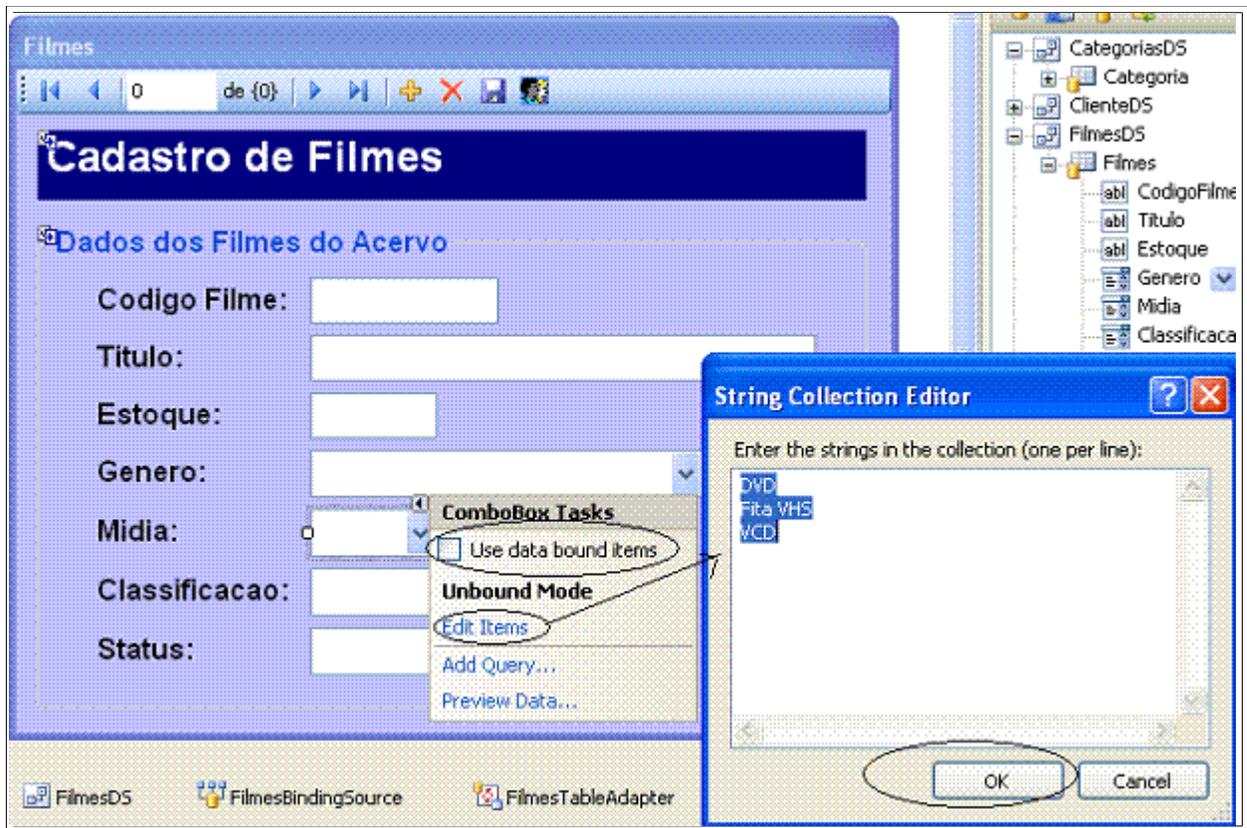
2- Informe o que deseja exibir no campo. No nosso caso vamos querer exibir o nome do **Genero**.

3- Informe o valor do SelectedItem

4- Deixe o último campo como **None**

Repita o procedimento para os campos **Classificação** e **Status**.

No campo **Midia** não marque a opção - **data bound items**. Clique em **Edit Items** e informe os valores na janela - **String Collection Editor**.



Você percebeu que a medida que você vai vinculando os controles, as respectivas instâncias de cada **DataSet** são incorporadas na base do formulário.

O resultado obtido é exibido na figura abaixo. Além de facilitarmos a vida do usuário estamos evitando que ocorra um erro na informação manual que anteriormente deveria ser feita via digitação pelo usuário. Tudo isto sem digitar uma única linha de código.

The screenshot shows a Windows application window titled "Filmes". At the top, there is a navigation bar with a page indicator showing "2 de 5". Below this is a dark blue header with the text "Cadastro de Filmes". The main area is a light blue form titled "Dados dos Filmes do Acervo". It contains several input fields and dropdown menus:

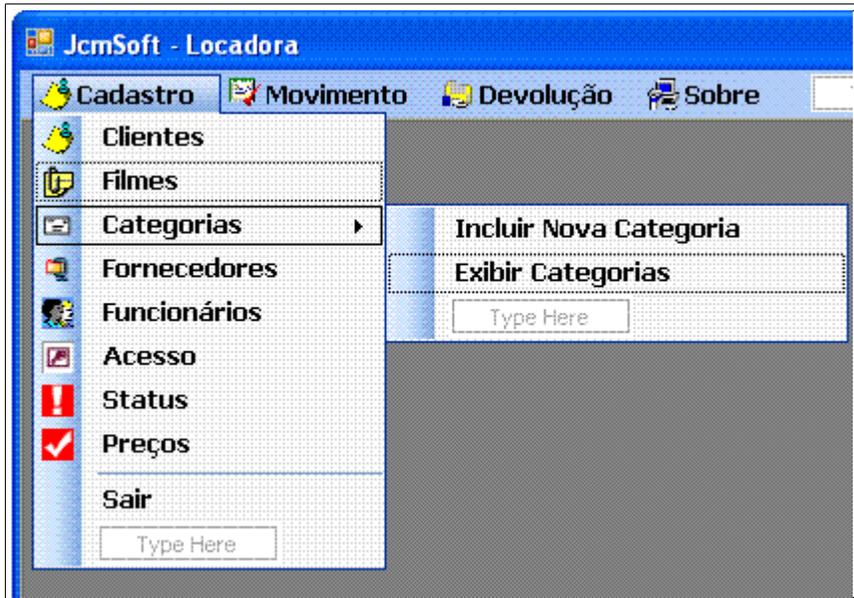
- Codigo Filme: 2
- Titulo: Matrix
- Estoque: 4
- Genero: Ficção
- Midia: DVD
- Classificacao: Lançamento
- Status: Disponível

A dropdown menu for the "Status" field is open, displaying the following options: Disponível, Alugada, Reservada, Danificada, and Defeito.

Foram criados também os formulário **Status.vb** e **Precos.vb** para permitir o cadastramento dos valores usados na locadora. Ambos herdam a aparência do nosso formulário modelo e foram criados como mostrado em artigo anterior. Abaixo temos a figura destes formulários:

The image shows two separate Windows application windows side-by-side. The left window is titled "Status dos Filmes" and has a page indicator "0 de {0}". It contains a form with three fields: "Codigo" (text box), "Descricao" (text box), and "Ativo" (checkbox). The right window is titled "Precos" and also has a page indicator "0 de {0}". It contains a form with three fields: "Codigo Classificacao" (text box), "Descricao" (text box), and "Valor" (text box).

Outro ajuste feito foi incorporar ao menu principal - **Principal.vb** - as novas opções conforme abaixo:



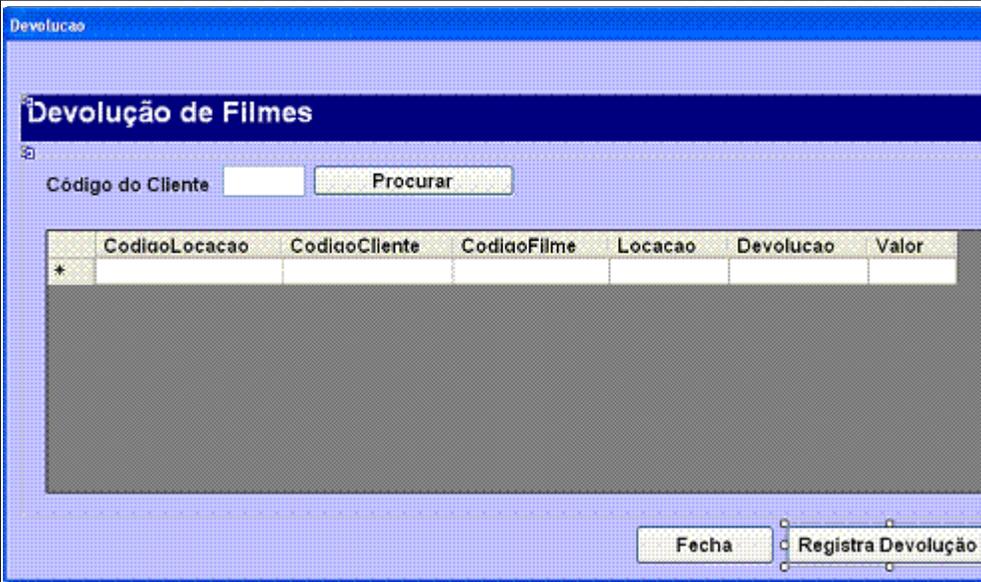
Parte 7

Implementando a classe para registrar a devolução de um filme

A lógica para efetuar a devolução de um filme alugado vai estar na classe **DevolucaoFilmes** presente no arquivo **DevolucaoFilmes.vb**. As tarefas que deverão ser efetuadas na devolução de um filme são:

- Verificar se o filme indicado existe no estoque da locadora
- Aumentar uma unidade referente ao filme que esta sendo devolvido ao estoque da locadora
- Calcular o valor devido pela locação do filme
- Registrar a devolução gravando a data de devolução e o valor devido na tabela Movimento

Tudo começa quando o cliente apresenta o filme para devolução. O funcionário acessa o sistema e exibe o formulário de devolução de filmes: **devolucao.vb** mostrado abaixo:



O funcionário informe o código do cliente ou efetua a busca acionando o botão procurar.

Após selecionar o cliente, o **DataGridView** é preenchido com a relação dos filmes que o cliente possui alugados.

O funcionário seleciona um ou todos os filmes e efetua o registro da devolução.

O evento **Click** do botão procurar possui o seguinte código:

```
Private Sub btnProcuraCliente_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnProcuraCliente.Click

    My.Forms.SelecionaCliente.ShowDialog()

    txtCodigoCliente.Text = My.Forms.SelecionaCliente.codigoCliente
    lblNomeCliente.Text = My.Forms.SelecionaCliente.nomeCliente

    Me.MovimentoTableAdapter.FillByCodigoClienteEdevolucaoNull(MovimentoDS.Movimento,
```

```
CType(txtCodigoCliente.Text, Integer))
```

```
End Sub
```

O formulário **SelecionaCliente** é exibido e nele o funcionário informa o nome do cliente para efetuar a seleção.

A seguir o código do cliente é exibido na caixa de texto - **txtCodigoCliente**- e o seu nome no controle Label - **lblNomeCliente**.

O DataGridView é então preenchido pela consulta previamente incluída no TableAdapter **MovimentoDS** chamada **FillByCodigoClienteDevolucaoNull** que possui a seguinte estrutura:

```
SELECT CodigoLocacao, CodigoCliente, CodigoFilme, Locacao, Devolucao,
Valor
FROM Movimento
WHERE (CodigoCliente = @codigoCliente) AND (Devolucao IS NULL)
```

A consulta irá exibir os dados da tabela Movimento cujo campo **CodigoCliente** é igual ao código do cliente foi selecionado e cujo campo **Devolucao** é **Null**.

A seguir o funcionário seleciona o filme que é objeto da devolução e clica no botão - **Registra Devolução** - cujo código do evento Click é exibido a seguir:

```
Private Sub btnDevolucao_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDevolucao.Click

Dim fila As DataGridViewRow
Dim selecao As MovimentoDS.MovimentoRow
Dim registraDevolucao As New DevolucaoFilmes

For Each fila In MovimentoDataGridView.SelectedRows
    selecao = MovimentoDS.Movimento(fila.Index)
    registraDevolucao.calcula(selecao.CodigoFilme, selecao.Locacao, Now)
    registraDevolucao.devolveFilme(selecao.CodigoFilme, selecao.Locacao, Now,
registraDevolucao.valorLocacao)
Next

Me.Validate()
Me.MovimentoBindingSource.EndEdit()

MsgBox("Devolução registrada com sucesso...", MsgBoxStyle.Information)
Me.MovimentoTableAdapter.Update(MovimentoDS.Movimento)

End Sub
```

Após declarar uma variável do tipo **DataGridView** e outra do tipo **MovimentoRow** estamos instanciando um objeto da classe **DevolucaoFilmes**:

```
Dim fila As DataGridViewRow
Dim selecao As MovimentoDS.MovimentoRow
```

Dim registraDevolucao As New DevolucaoFilmes

A seguir para cada linha do **DataGridView** selecionada estamos invocando os seguintes métodos da classe **DevolucaoFilmes**:

1. [registraDevolucao.calcula\(selecao.CodigoFilme, selecao.Locacao, Now\)](#) - O método **calcula** utiliza os parâmetros : **CodigoFilme**, a data de locação (**selecao.Locacao**) e a data atual (**Now**) que representa a data da devolução do filme.
2. [registraDevolucao.devolveFilme\(selecao.CodigoFilme, selecao.Locacao, Now, registraDevolucao.valorLocacao\)](#) : O método **devolveFilme** usa os parâmetros : **CodigoFilme**, data de locação (**selecao.Locacao**) , a data atual (**Now**) e o valor da locação calculada (registraDevolucao.valorLocacao)

Estes métodos estão implementados na classe **DevolucaoFilmes** que você deve incluir no projeto clicando com o botão direito do mouse sobre o nome do projeto e seleciona a opção **Add->New Item** , e a seguir selecionar o template **Class** informando o nome da classe.

A seguir temos o código da classe **DevolucaoFilmes** contendo os dois métodos : **calcula e devolveFilme** e a propriedade **valorLocacao** pela qual podemos obter o valor calculado da locação.

```
Public Class DevolucaoFilmes

    'variável de classe valor_
    Private Shared valor_ As Double

    Public Sub devolveFilme(ByVal codfilme As Integer, ByVal datalocacao As Date, ByVal
    dataDevolucao As Date, ByVal valor As Double)

        Dim codLocacao As Integer

        'cria um adapter e um dataset
        Dim adapterMovimento As New MovimentoDSTableAdapters.MovimentoTableAdapter
        Dim movimento As New MovimentoDS

        'filtra os filmes pelo codigo informado
        adapterMovimento.FillByCodigoFilme(movimento.Movimento, codfilme)
        codLocacao = movimento.Movimento(0).CodigoLocacao

        'registrar a devolucao do filme
        'cria um adapter e um dataset
        Dim adapterDevolucao As New MovimentoDSTableAdapters.MovimentoTableAdapter
        Dim movimentoDevolucao As New MovimentoDS.MovimentoDataTable
        'atualiza a tabela movimento incluindo a data de devolução e o valor a ser pago
        adapterDevolucao.UpdateQueryDevolucao(dataDevolucao, valor, codLocacao)
        adapterDevolucao.Update(movimentoDevolucao)
    End Sub

    Método calcula

    Public Sub calcula(ByVal codfilme As Integer, ByVal datalocacao As Date, ByVal
    datadevolucao As Date)
```

```

Dim classificacao As String
Dim valor As Double
'verificando o estoque da locadora
'cria um adapter e um dataset
Dim adapterFilmes As New FilmesDSTableAdapters.FilmesTableAdapter
Dim filme As New FilmesDS
'filtra os filmes pelo codigo informado
adapterFilmes.FillByCodigo(filme.Filmes, codfilme)
'se não encontrou o filme com o codigo informado avisa
If filme.Filmes.Count = 0 Then
    Throw New ArgumentException("Não existe o filme informado no acervo.")
End If

'se não tem filme no estoque avisa
If filme.Filmes(0).Estoque = 0 Then
    Throw New ArgumentException("Não existem unidades cadastradas no acervo.")
End If

'se houver filme no estoque tenho que aumentar uma unidade que esta sendo devolvida
filme.Filmes(0).Estoque += 1
filme.Filmes(0).Status = "Disponível" ' atualiza o status para codigo 1 - Disponível

classificacao = filme.Filmes(0).Classificacao
'
----verifica o preco do filme para calcular o valor
Dim adapterPrecos As New PrecosDSTableAdapters.PrecosTableAdapter
Dim preco As New PrecosDS

adapterPrecos.FillByClassificacao(preco.Precos, classificacao)
valor = preco.Precos(0).Valor
Dim dias As Integer = DateDiff("d", datalocacao, datadevolucao)
Dim precoPagar As Double = dias * valor

valor_ = precoPagar
End Sub

Propriedade valorLocacao

Public Property valorLocacao() As Double
    Get
        Return valor_
    End Get
    Set(ByVal valor As Double)
        valor_ = valor
    End Set
End Property
End Class

```

Perceba que no método calcula estamos incrementando uma unidade, referente ao filme devolvido, ao estoque e alterando o **status** do filme para "**Disponível**":

```

'se houver filme no estoque tenho que aumentar uma unidade que esta sendo devolvida
filme.Filmes(0).Estoque += 1
filme.Filmes(0).Status = "Disponível" ' atualiza o status para codigo 1 - Disponível

```

Abaixo temos a janela exibida pela execução do formulário **devolucao.vb**:

The screenshot shows a window titled "Devolucao" with a sub-header "Devolução de Filmes". Below the header, there is a search form with "Código do Cliente" set to "1", a "Procurar" button, and the name "Jefferson Dias". A table displays the following data:

	CodigoLocacao	CodigoCliente	CodigoFilme	Locacao	Devolucao	Valor
▶	1	1	1	12/5/2006		
	3	1	1	3/6/2006 15:52		
	4	1	2	3/6/2006 15:52		
	5	1	3	3/6/2006 15:52		
*						

At the bottom right, there are two buttons: "Fecha" and "Registra Devolução".

Após efetuar o [registro da devolução](#) podemos verificar os valores atualizados na tabela movimento na opção **Movimento-> Filmes Alugados** do menu principal:

The screenshot shows a window titled "Filmes Alugados" with a sub-header "Filmes Alugados". A table displays the following data:

	Locação	Cliente	Filme	Data	Data	Valor
▶	1	1	1	12/5/2006	3/6/2006 1...	2,0000
	2	2	2	15/5/2006	3/6/2006 1...	4,0000
	3	1	1	3/6/2006 1...		
	4	1	2	3/6/2006 1...		
	5	1	3	3/6/2006 1...	3/6/2006 1...	4,0000
*						

At the bottom right, there is a "Fechar" button.

E com isto fechamos o ciclo desta aplicação de exemplo onde o objetivo foi mostrar alguns dos novos recursos do [ADO.NET 2.0](#) e como você pode criar uma aplicação [Windows Forms](#) de forma rápida usando os novos assistentes para auxiliá-lo em tarefas que antes exigiriam muito código. Não pode ser considerada uma aplicação de produção pois não me preocupei com os requisitos fundamentais que uma locadora real deveria possuir para estar funcionando. Aliás, quero chamar a atenção para o fato de que o levantamento dos requisitos é uma das fases cruciais e que geralmente determinam o sucesso ou o fracasso de um projeto de software. Nossa aplicação não resistiria a uma crítica a começar pela modelagem de dados. No entanto ela serviu aos meus propósitos que foi poder mostrar o desenvolvimento de uma aplicação windows com alguns recursos no [Visual Basic 2005 Express](#).

Parte 8

Instalação da aplicação

A parte final do nosso projeto **Locadora de Filmes** no Visual Basic 2005 Express Edition é justamente montar o pacote de instalação da aplicação. Vamos criar o pacote de instalação usando o **Inno Setup**, pois é um software **free** e de grande versatilidade e facilidade de utilização.

Se você não possui o **Inno Setup** pegue a última versão no sítio : www.innosetup.com. (A última versão estável era a 5.17)

Se você deseja distribuir a **NET framework 2.0** junto com sua instalação vai precisar do arquivo **NET framework 2.0 redistributable package** que pode ser baixado do sítio: <http://www.microsoft.com/downloads/details.aspx?FamilyID=9655156b-356b-4a2c-857c-e62f50ae9a55&DisplayLang=pt-br> (São 22 MB)

	<p>Antes de iniciar o processo de gerar o pacote de instalação você precisa localizar a pasta onde sua aplicação foi gerada e criar nela uma pasta chamada Install e outra pasta chamada support subordinada a pasta Install.</p> <p>A seguir copie o arquivo dotnetfx.exe para a pasta support.</p> <p>Para encerrar crie uma pasta chamada OutPut na pasta Support.</p> <p>Obs:O arquivo dotnetfx.exe é a NET framework 2.0 redistributable package (22 MB)</p>
--	--

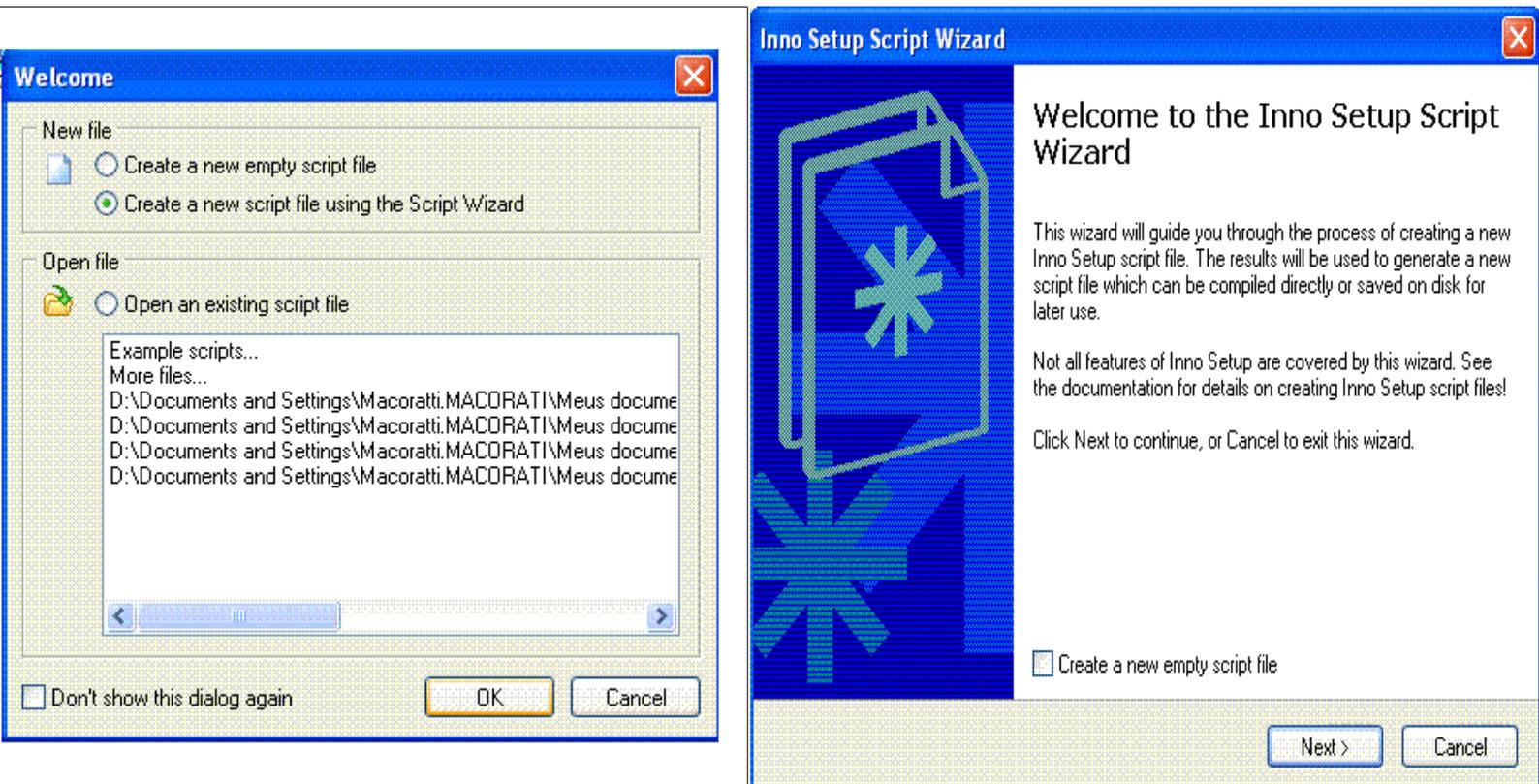
Não esqueça de compilar o seu projeto e deixá-lo pronto para a instalação. Menu **Build - Build seu_projeto**.

Nota: Se você está chegando agora recomendo acompanhe o desenvolvimento da aplicação nos artigos :

1. [VB.NET 2005 - Criando uma aplicação completa : Locadora de Filmes I](#)
2. [VB.NET 2005 - Criando uma aplicação completa : Locadora de Filmes II](#)
3. [VB.NET 2005 - Criando uma aplicação completa : Locadora de Filmes III](#)
4. [VB.NET 2005 - Criando uma aplicação completa : Locadora de Filmes IV](#)
5. [VB.NET 2005 - Criando uma aplicação completa : Locadora de Filmes V](#)
6. [VB.NET 2005 - Criando uma aplicação completa : Locadora de Filmes VI](#)
7. [VB.NET 2005 - Criando uma aplicação completa : Locadora de Filmes VII](#)

Criando o pacote de instalação usando o Inno Setup

Inicie o Inno Setup e na tela inicial marque a opção - Create a new script using the Script Wizard e clique no botão OK. A seguir, na tela de apresentação do Wizard, deixe a opção - Create a new script file - desmarcada e clique no botão Next>.



A próxima janela de diálogo do assistente solicita algumas informações básicas sobre a aplicação:

The screenshot shows the 'Application Information' dialog box in the Inno Setup Script Wizard. The title bar reads 'Inno Setup Script Wizard'. The main heading is 'Application Information' with a sub-instruction: 'Please specify some basic information about your application.' Below this, there are several input fields: 'Application name:' with the text 'Locadora de Filmes'; 'Application name including version:' with 'Locadora de Filmes 1.0'; 'Application publisher:' with 'Macoratti.net Inc.'; and 'Application website:' with 'http://www.macoratti.net'. At the bottom left, it says 'bold = required'. At the bottom right, there are three buttons: '< Back', 'Next >', and 'Cancel'.

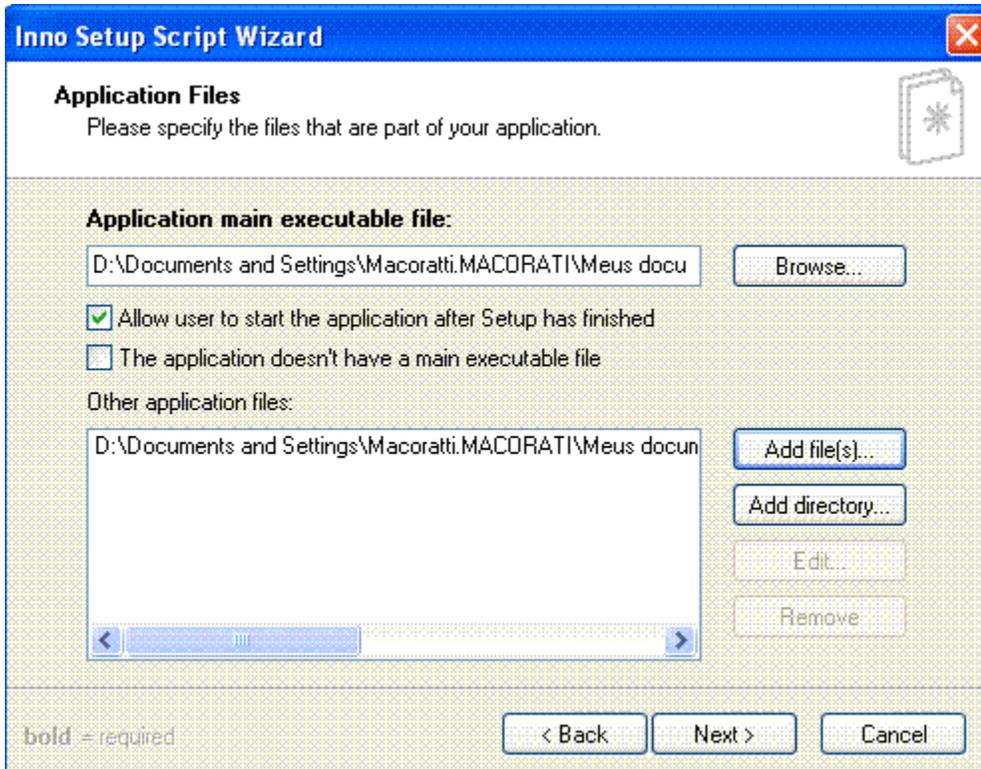
Forneça as informações necessárias e clique no botão **Next>**.

A próxima janela solicita informações sobre o local de instalação da aplicação.

The screenshot shows the 'Application Directory' dialog box in the Inno Setup Script Wizard. The title bar reads 'Inno Setup Script Wizard'. The main heading is 'Application Directory' with a sub-instruction: 'Please specify directory information about your application.' Below this, there are several options: 'Application destination base directory:' with a dropdown menu set to 'Program Files directory'; 'Application directory name:' with a text field containing 'Locadora de Filmes'; a checked checkbox for 'Allow user to change the application directory'; and an unchecked checkbox for 'The application doesn't need a directory'. At the bottom left, it says 'bold = required'. At the bottom right, there are three buttons: '< Back', 'Next >', and 'Cancel'.

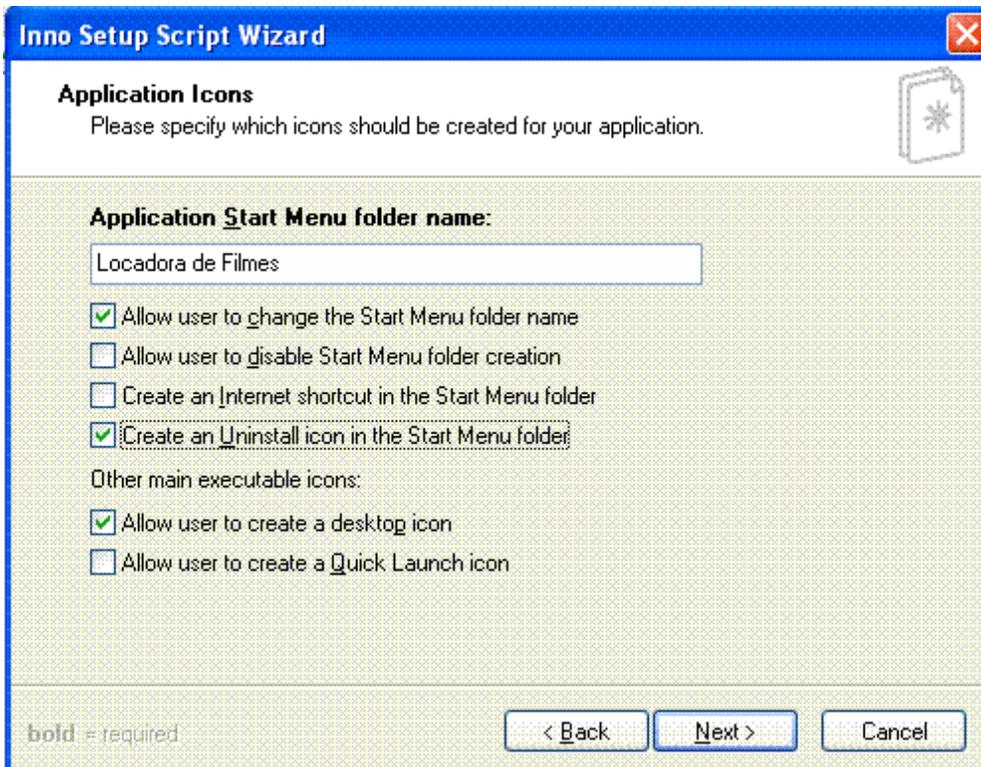
Geralmente a instalação é feita na pasta - **Program Files Directory** . Informe o nome da pasta e clique no botão **Next>**.

A próxima janela define os arquivos da aplicação:



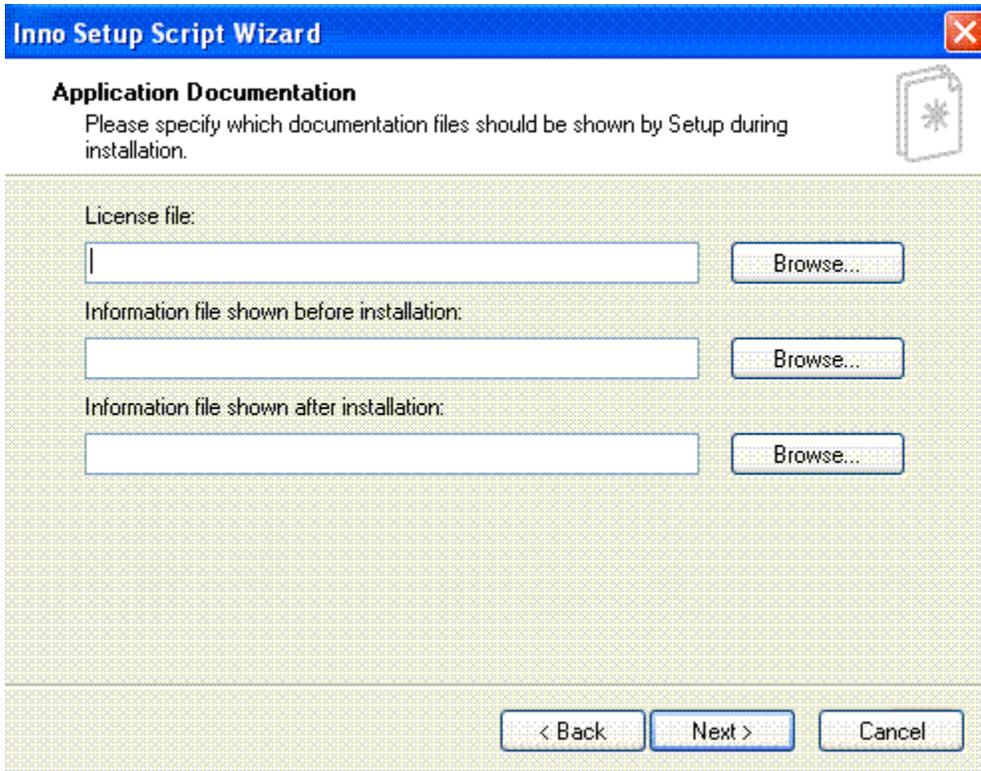
Selecione o executável, clicando no botão **Browse**. Geralmente o arquivo principal executável fica na pasta **bin\release** da sua aplicação. A seguir use o botão **Add File(s)** para incluir o arquivo **.config** para sua aplicação. Clique no botão **Next>**

Após definir os arquivos de instalação na próxima janela vamos criar definir o nome inicial e alguns ícones para aplicação.

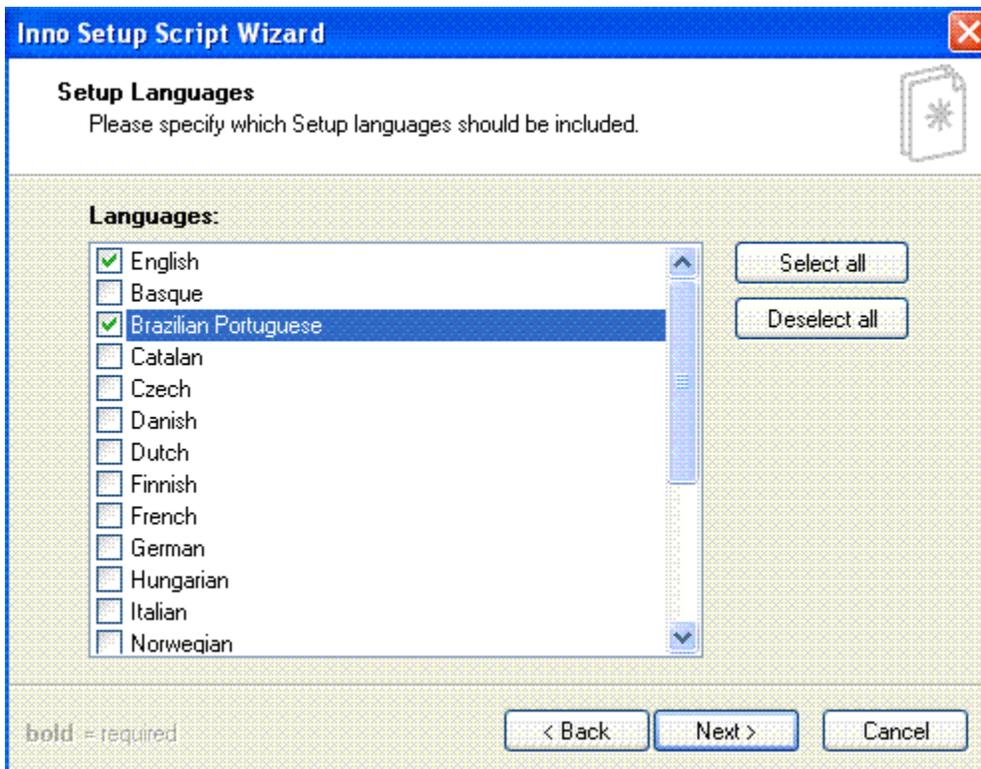


A seguir será apresentada a tela onde você pode definir um arquivo de licenciamento para a aplicação. Se você definir um arquivo de licença o usuário não poderá prosseguir com a instalação sem concordar com o conteúdo do arquivo.

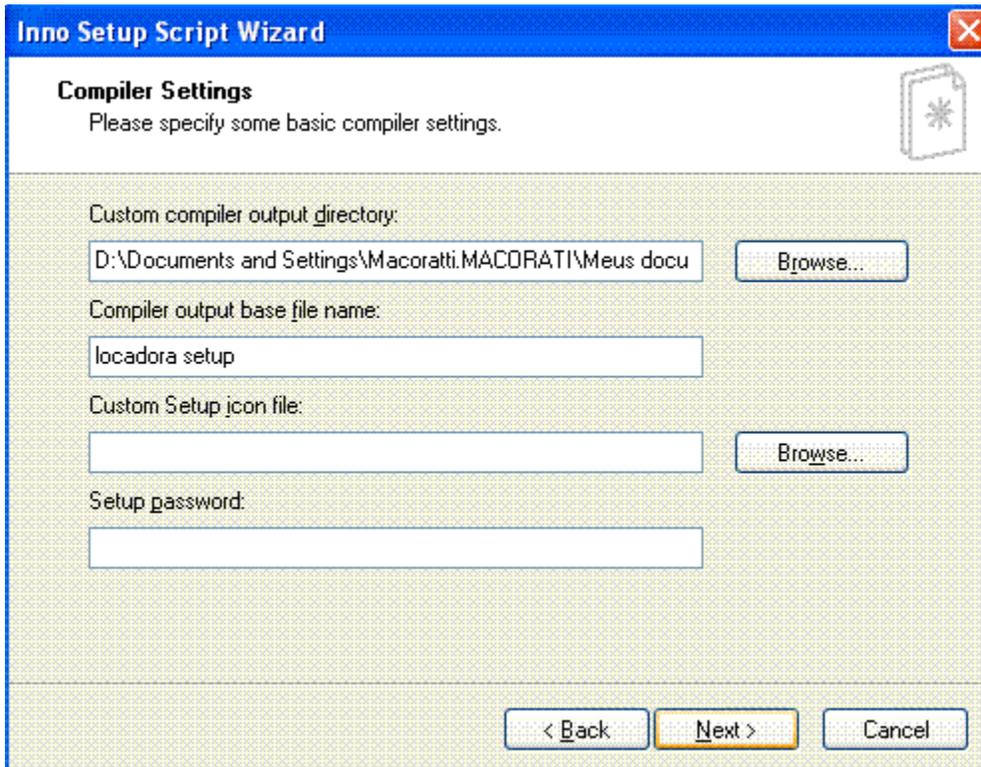
A nossa aplicação não possui este arquivo clique em **Next>**



Na próxima janela defina as linguagens para as quais estarão disponíveis durante o **setup** da aplicação.



A última janela defina a pasta base de destino da compilação, selecione a pasta **OutPut** que foi criada no interior da pasta **Support**.



Clique no botão **Next>** e a seguir no botão **Finish**.

Na tela final do assistente do Inno Setup será apresentado o arquivo de Script semelhante ao da figura abaixo:

```

Name: "brazilianportuguese"; MessagesFile: "compiler:Languages\BrazilianPortuguese.isl"

[Tasks]
Name: "desktopicon"; Description: "{cm:CreateDesktopIcon}"; GroupDescription: "{cm:Additi

[Files]
* Source: "D:\Documents and Settings\Macoratti.MACORATI\Meus documentos\Visual Studio 2005\
* Source: "D:\Documents and Settings\Macoratti.MACORATI\Meus documentos\Visual Studio 2005\
; NOTE: Don't use "Flags: ignoreversion" on any shared system files

[Icons]
* Name: "{group}\Locadora de Filmes"; Filename: "{app}\Locadora Filmes.exe"
* Name: "{group}\{cm:UninstallProgram,Locadora de Filmes}"; Filename: "{uninstall.exe}"
* Name: "{userdesktop}\Locadora de Filmes"; Filename: "{app}\Locadora Filmes.exe"; Tasks: d

[Run]
* Filename: "{app}\Locadora Filmes.exe"; Description: "{cm:LaunchProgram,Locadora de Filmes

Compressing Setup stub
Updating version info

*** Finished. [16:23:01, 00:05,758 elapsed]

Compiler Output | Debug Output |
1: 1 | Modified | Insert

```

Este é o script que está preparado para realizar uma instalação básica na máquina de destino. Selecione a opção Save no menu File do Inno Setup e salve o script na pasta Install do projeto.

Podemos dar suporte a inclusão da .NET Framework na instalação da aplicação. O arquivo dotnetfx.exe que foi baixado e copiado para a pasta Support precisa ser incluído no instalador para poder rodar como parte do processo. Isto vai aumentar o tamanho do arquivo de instalação pois o dotnetfx.exe possui 22 MB de tamanho, mas com isto estamos assegurando que se a máquina de destino não possuir o .NET Framework 2.0 instalado o instalador irá instalar.

Para fazer isto precisamos instruir o Inno Setup para incluir o arquivo dotnetfx.exe no instalador pois queremos ter este arquivo disponível na instalação como um arquivo temporário que será deletado no final da instalação.

Os arquivos que fazem parte da instalação estão na seção [Files] do script. Nós definimos a fonte e o destino para cada arquivo usando chaves especiais que indicam o comportamento do arquivo. No nosso caso a fonte será a pasta support/dotnetfx.exe e o destino será uma pasta especial chamada {tmp} que mapeia para o diretório temporário na máquina de destino.

Inclua então a seguinte linha na seção [Files] do arquivo de script:

```
Source: support\dotnetfx.exe; DestDir: {tmp}; Flags:  
deleteafterinstall
```

Para completar o processo precisamos incluir uma chamada para o script de instalação para executar o arquivo antes que a instalação se complete. Inclua na seção [Run] do arquivo de script os seguintes parâmetros e mensagem que será exibida :

```
Filename: {tmp}\dotnetfx.exe; Parameters: "/q"; WorkingDir: {tmp};  
StatusMsg: Instalando DotNET Framework 2.0
```

O parâmetro /q indica que a instalação será silenciosa e a mensagem será :
Instalando o DotNet Framwork 2.0

Agora que tudo esta pronto podemos proceder a compilação do script que irá gerar o arquivo de Setup na pasta Output. No Menu **Build** do Inno Setup clique na opção **Compile**. Agora é só testar a instalação da sua aplicação.

Com isto terminamos o ciclo de desenvolvimento e instalação de uma aplicação completa no Visual Basic 2005 Express Edition.