Desenvolvimento - C#

Guia prático para o desenvolvimento de Aplicações C# em Camadas - parte 4

Este é o quarto artigo da série onde vamos demonstrar passo-a-passo a construção de uma aplicação .Net utilizando o conceito de desenvolvimento em camadas.

por Carlos Camacho

Artigo atualizado em 01/08/2008.

Neste artigo vamos implementar as classes da Camada DAL (Data Access Layer).

Criaremos as classes:

· ClientesDAL.cs

· ProdutosDAL.cs

· VendasDAL.cs

· Dados.cs

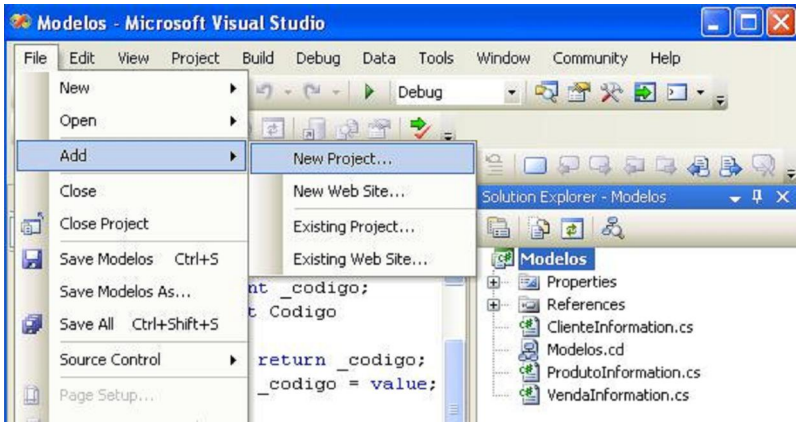
Você verá que a classe ClientesDAL, por exemplo, conterá os métodos de inclusão, alteração, exclusão e consulta referentes a tabela de Clientes. O mesmo ocorrerá para as classes ProdutosDAL e VendasDAL.

Usaremos a classe Dados para armazenarmos a string de conexão com o banco de dados. Vamos começar?

- Abra o Microsoft Visual Studio;

- Abra o projeto Modelos que criamos no artigo anterior.

- Vamos adicionar um novo projeto. No menu **File** do Visual Studio, clique em **Add** > **New Project...**



Na janela Add New Project:

- Escolha **Visual C#** para o tipo de projeto;

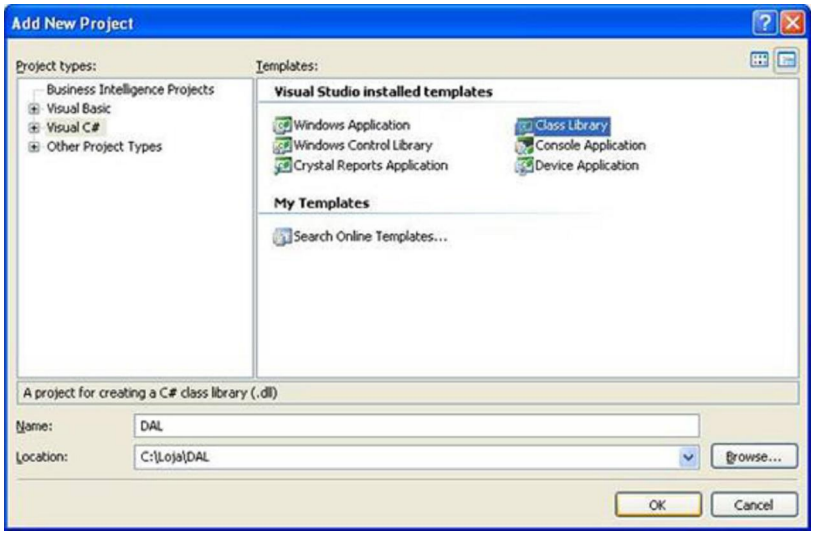
- **Class Library** para o template;

- Digite **DAL** no campo nome;

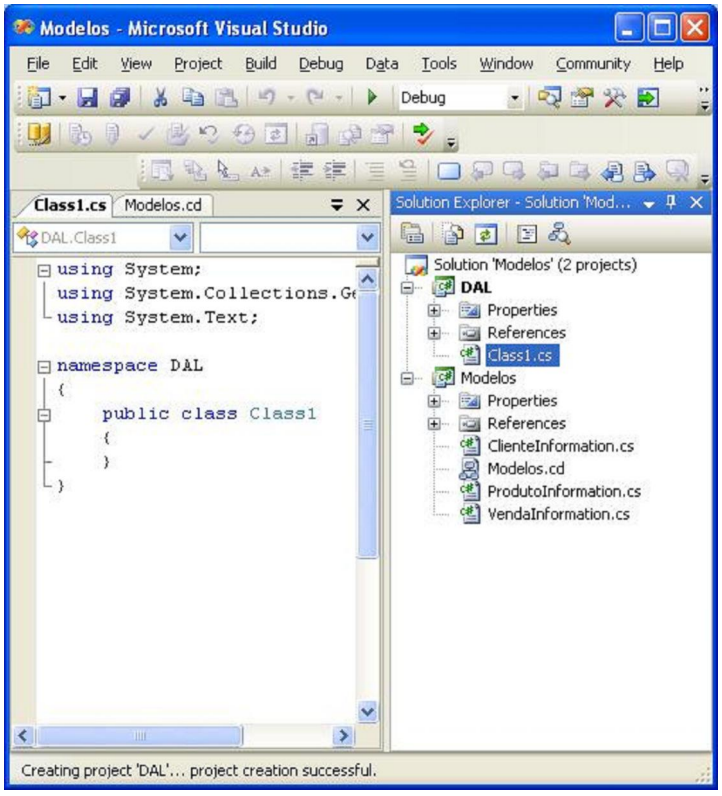
- Digite **C:\Loja\DAL** na localização;

- Clique em **Ok** para adicionar o novo projeto à nossa *Solution* (solução).

Nota: Quando temos um conjunto de projetos reunidos para atender uma necessidade chamamos esse conjunto de *Solution*.



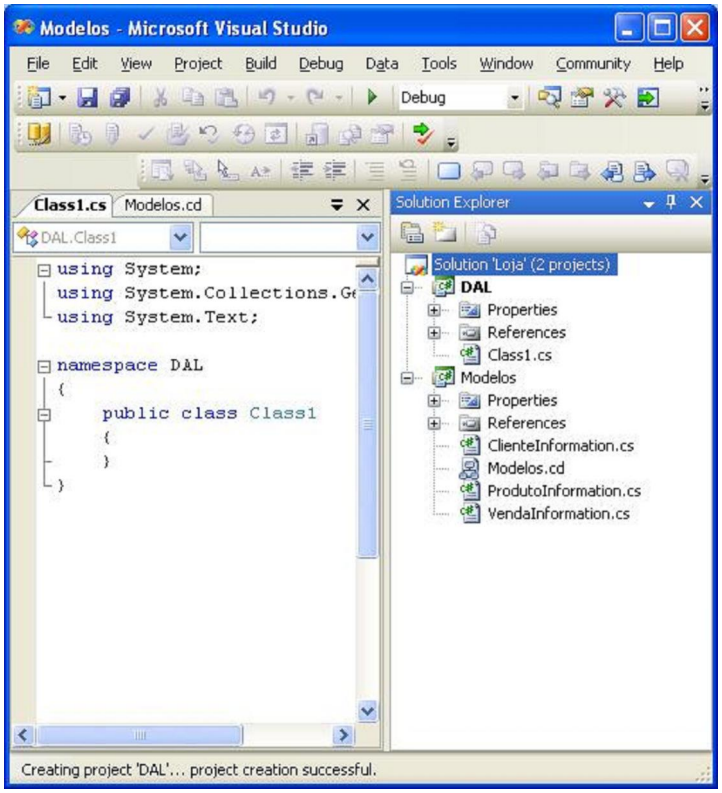
Muito bom! Agora o nosso projeto DAL foi adicionado à nossa Solution:



O nome da nossa Solution está como Modelos porque o MS Visual Studio usa o nome do primeiro projeto criado.

No Solution Explorer, clique com o botão direito sobre a Solution 'Modelos' e escolha a opção **Rename**. Renomeie a Solution para **Loja**.

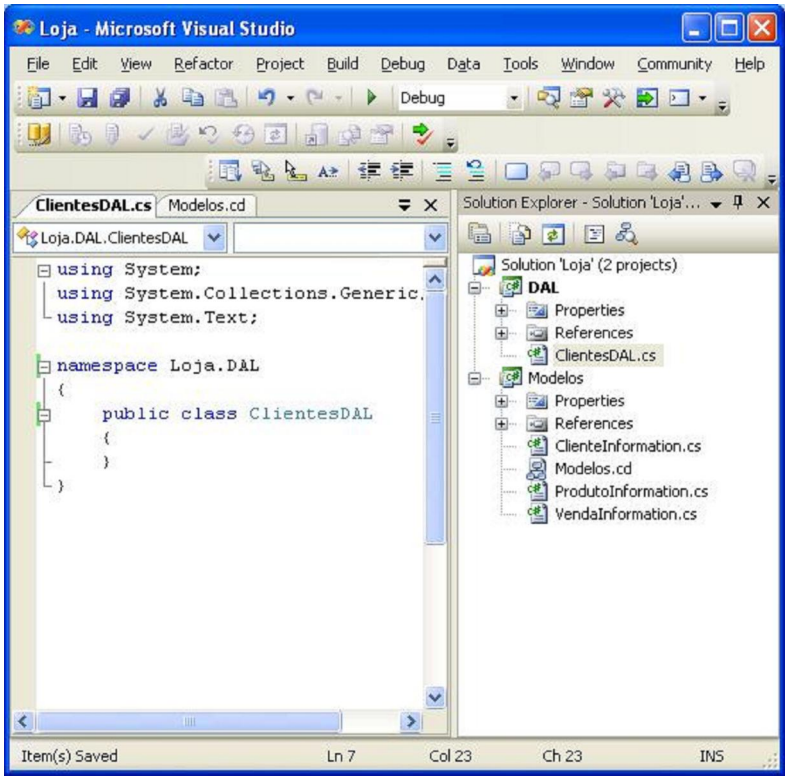
Depois que você renomear a nossa Solution, ela ficará assim:



- No projeto DAL, Renomeie a classe **Class1.cs** para **ClientesDAL.cs**;

- Altere o namespace para namespace Loja.DAL

A nossa classe ClientesDAL ficará assim:



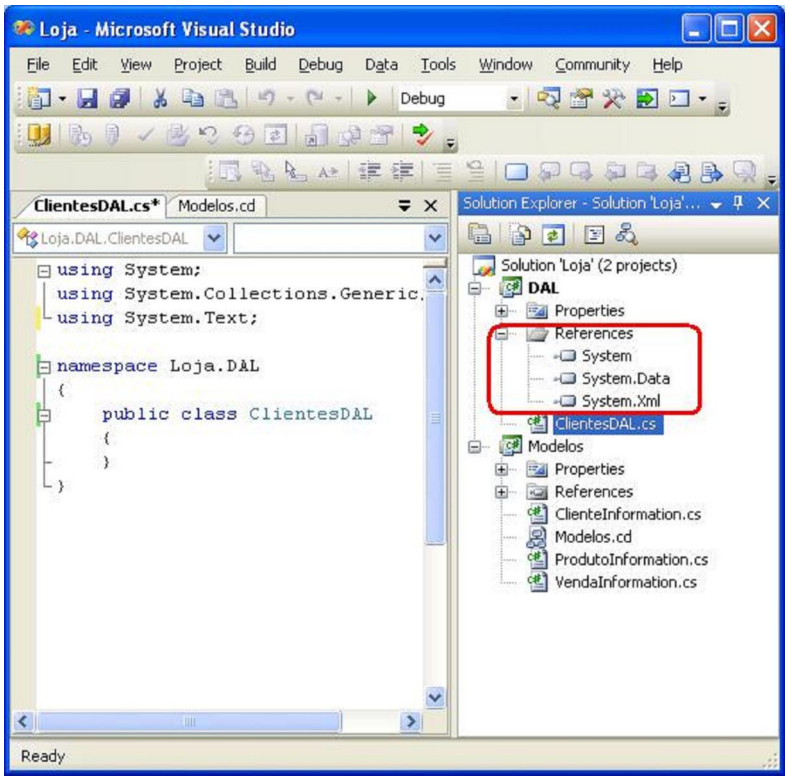
Você se lembra quando comentamos que a nossa implementação aconteceria da camada mais interna para a mais externa do desenho do nosso projeto?

[Modelos] DAL ð BLL ð User Interface

Então, o projeto Modelos é o mais interno e o próximo é a camada de acesso a dados DAL. Isso significa que o nosso projeto DAL poderá usar tudo o que já foi construído no projeto Modelos!

Em termos de programação, dizemos que o projeto DAL faz referência ao projeto Modelos. Para que o nosso projeto DAL consiga ver o que já construímos no projeto Modelos, vamos criar essa referência.

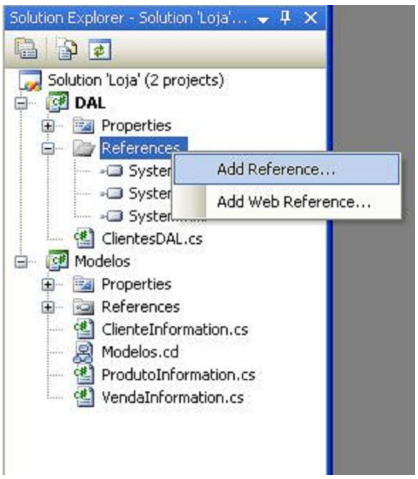
- Abra a pasta References do projeto DAL para ver o que já existe como referência sempre que criamos um novo projeto:



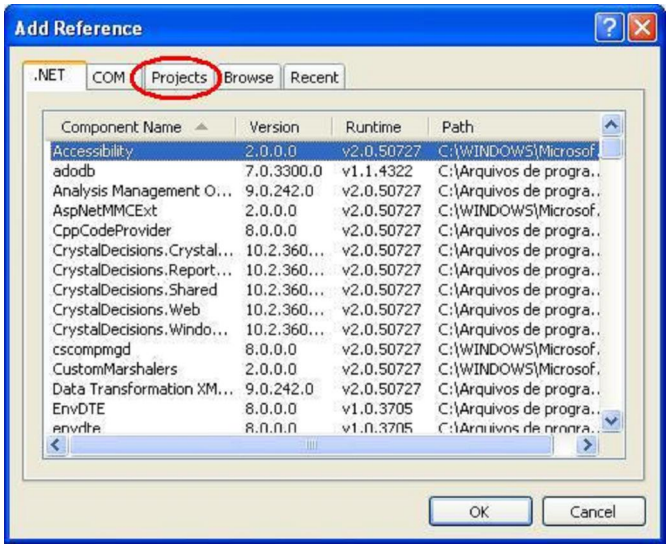
Precisamos incluir no projeto DAL uma referência para o Modelos. Faça o seguinte:

- Clique com o botão direito na pasta References do projeto DAL e escolha

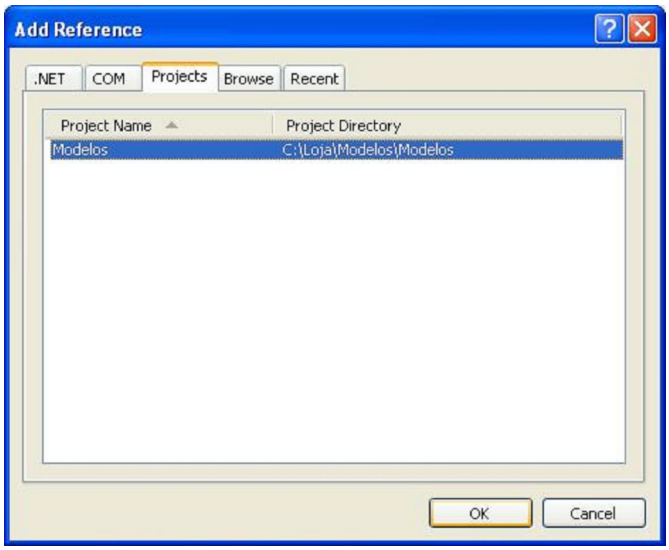
Add Reference...



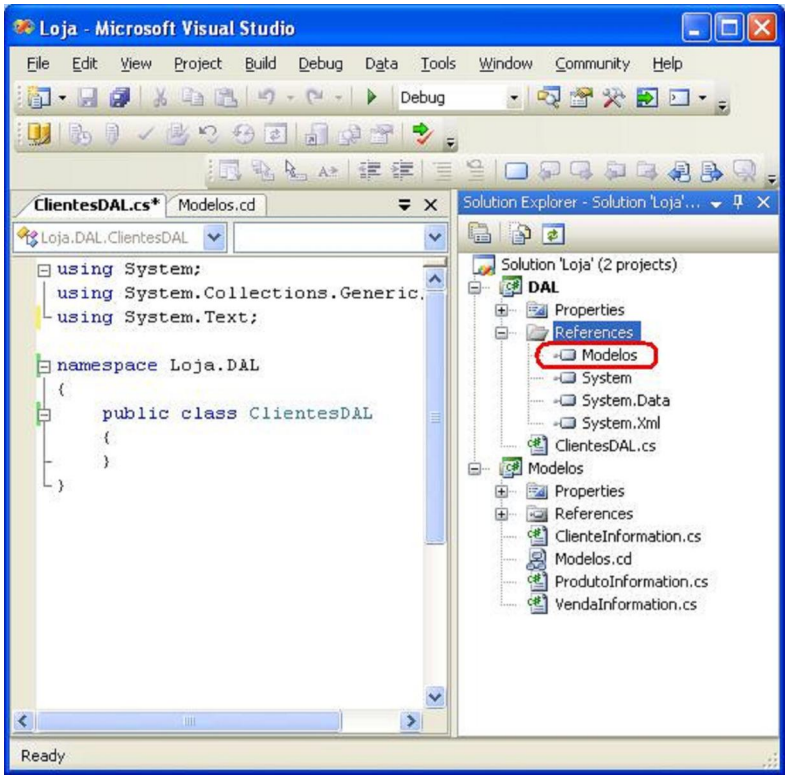
Na janela *Add Reference* que abrirá, clique na aba **Projects**:



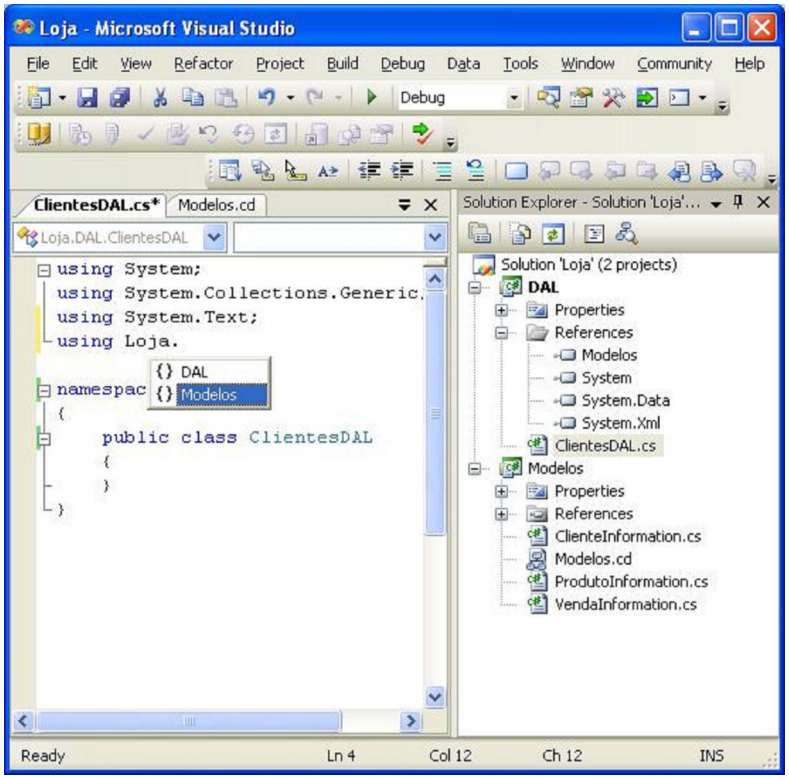
Veremos o nosso projeto Modelos. Clique sobre ele para selecioná-lo e então clique em **Ok** para criar a referência.



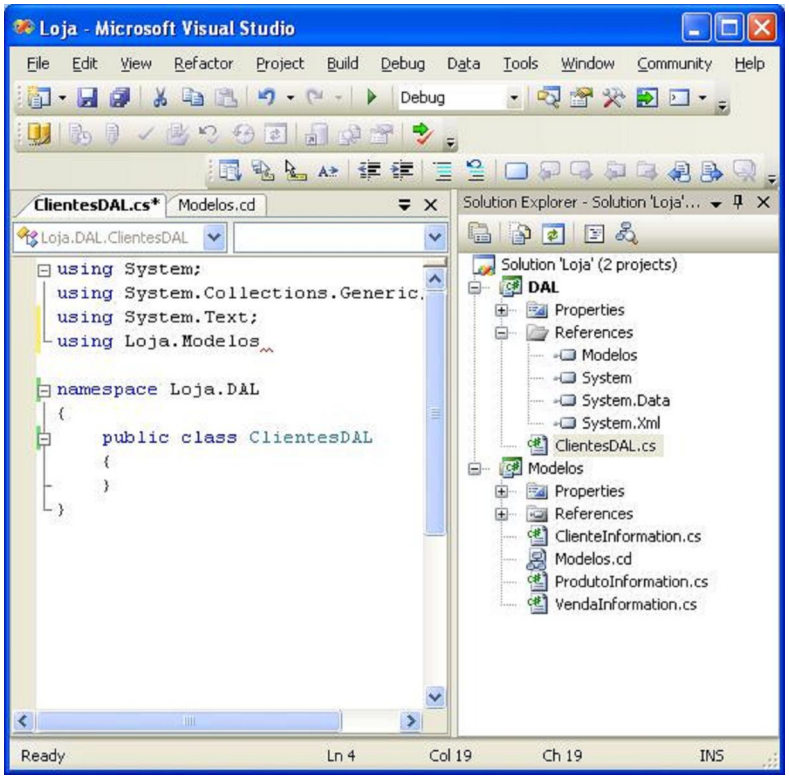
Muito bom! Agora já podemos ver a referência para o nosso projeto Modelos dentro do nosso projeto DAL como a seguir:



Para enxergar os Modelos dentro da nossa classe ClientesDAL vamos adicionar a cláusula using como segue:

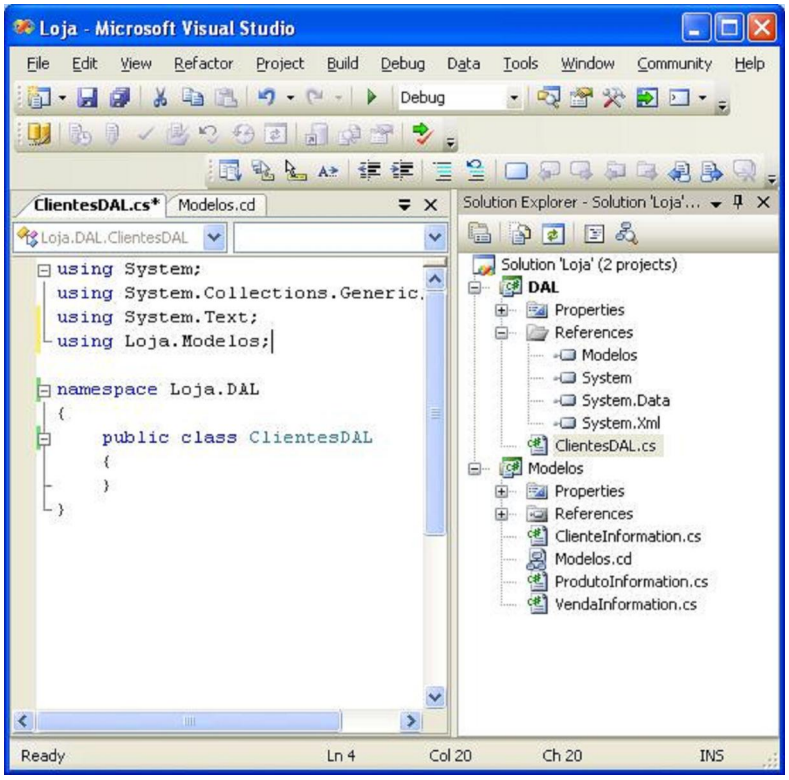


Após digitarmos using **Loja.** (coloque o ponto após digitar Loja) perceba que o MS Visual Studio já nos mostra as referências disponíveis. Com as setas de direção posicione a seleção em **Modelos** e dê um **ENTER** para selecioná-lo.

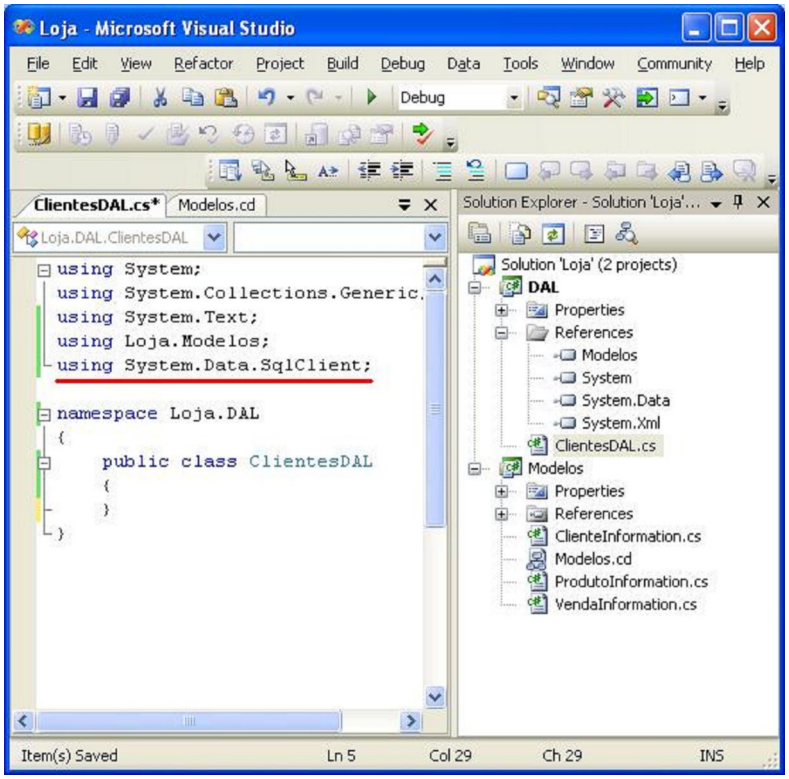


Aquele tracinho vermelho após Loja.Modelos indica que há algo errado. É o comando using que não foi fechado.

- Digite ponto-e-vírgula (;) para fechar o comando using:



Para que tenhamos acesso as definições do namespace do MS SQL Server, vamos inserir mais uma cláusula using para **System.Data.SqlClient** como a seguir:



Para o código da classe ClientesDAL copie e cole o código abaixo entre as chaves da classe:

```
public void Incluir (ClienteInformation cliente)

{

//conexao

SqlConnection cn = new SqlConnection();

try

{

cn.ConnectionString = Dados.StringDeConexao;

//command

SqlCommand cmd = new SqlCommand();

cmd.Connection = cn;

cmd.CommandText = "insert into Clientes(nome,email,telefone) values (@nome,@email,@telefone);
select @@IDENTITY;";

cmd.Parameters.AddWithValue ("@nome", cliente.Nome );

cmd.Parameters.AddWithValue("@email", cliente.Email);

cmd.Parameters.AddWithValue("@telefone", cliente.Telefone);

cn.Open();

cliente.Codigo = Convert.ToInt32 (cmd.ExecuteScalar());

}

catch (SqlException ex)

{

throw new Exception ("Servidor SQL Erro:" + ex.Number);

}

catch (Exception ex)
```

```
{
throw new Exception (ex.Message);
}
finally
{
cn.Close();
}
}
public void Alterar (ClienteInformation cliente)
{
// conexao
SqlConnection cn = new SqlConnection();
try
{
cn.ConnectionString = Dados.StringDeConexao;
SqlCommand cmd = new SqlCommand();
cmd.Connection = cn;
cmd.CommandType = CommandType.Text;
cmd.CommandText = "update Clientes set nome = @nome, email = @email, telefone = @telefone where
codigo = @codigo;";
cmd.Parameters.AddWithValue("@codigo", cliente.Codigo);
cmd.Parameters.AddWithValue("@nome", cliente.Nome);
cmd.Parameters.AddWithValue("@email", cliente.Email);
cmd.Parameters.AddWithValue("@telefone", cliente.Telefone);
cn.Open();
cmd.ExecuteNonQuery();
}
catch (SqlException ex)
{
throw new Exception("Servidor SQL Erro:" + ex.Number);
}
catch (Exception ex)
{
throw new Exception(ex.Message);
}
finally
{
cn.Close();
}
```

```
}
public void Excluir (int codigo)
{
//conexao
SqlConnection cn = new SqlConnection();
try
{
cn.ConnectionString = Dados.StringDeConexao;
//command
SqlCommand cmd = new SqlCommand();
cmd.Connection = cn;
cmd.CommandText = "delete from Clientes where codigo = " + codigo ;
cn.Open();
int resultado = cmd.ExecuteNonQuery();
if (resultado!=1) {
throw new Exception("Não foi possível excluir o cliente " + codigo);
}
}
catch (SqlException ex)
{
throw new Exception("Servidor SQL Erro:" + ex.Number);
}
catch (Exception ex)
{
throw new Exception(ex.Message);
}
finally
{
cn.Close();
}
}
public DataTable Listagem()
{
DataTable tabela = new DataTable();
SqlDataAdapter da = new SqlDataAdapter("select * from clientes", Dados.StringDeConexao );
da.Fill (tabela);
return tabela;
}
```

O código completo da nossa classe ClientesDAL.cs ficará assim:

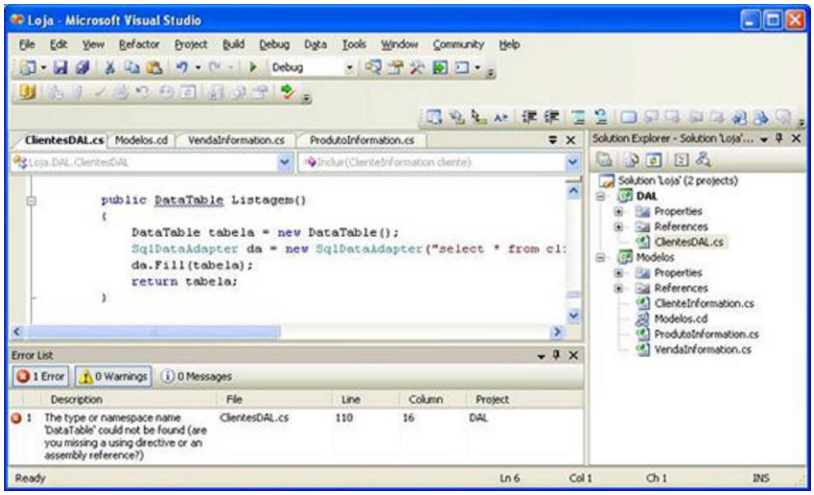
```
using System;
using System.Collections.Generic;
using System.Text;
using Loja.Modelos;
using System.Data.SqlClient;
namespace Loja.DAL
{
public class ClientesDAL
{
public void Incluir(ClienteInformation cliente)
{
//conexao
SqlConnection cn = new SqlConnection();
try
{
cn.ConnectionString = Dados.StringDeConexao;
//command
SqlCommand cmd = new SqlCommand();
cmd.Connection = cn;
cmd.CommandText = "insert into Clientes(nome,email,telefone) values (@nome,@email,@telefone);
select @@IDENTITY;";
cmd.Parameters.AddWithValue("@nome", cliente.Nome);
cmd.Parameters.AddWithValue("@email", cliente.Email);
cmd.Parameters.AddWithValue("@telefone", cliente.Telefone);
cn.Open();
cliente.Codigo = Convert.ToInt32(cmd.ExecuteScalar());
}
catch (SqlException ex)
{
throw new Exception("Servidor SQL Erro:" + ex.Number);
}
catch (Exception ex)
{
throw new Exception(ex.Message);
}
finally
{
cn.Close();
```

```
}

}

public void Alterar(ClienteInformation cliente)

{

// conexao

SqlConnection cn = new SqlConnection();

try

{

cn.ConnectionString = Dados.StringDeConexao;

SqlCommand cmd = new SqlCommand();

cmd.Connection = cn;

cmd.CommandType = CommandType.Text;

cmd.CommandText = "update Clientes set nome = @nome, email = @email, telefone = @telefone where
codigo = @codigo;";

cmd.Parameters.AddWithValue("@codigo", cliente.Codigo);

cmd.Parameters.AddWithValue("@nome", cliente.Nome);

cmd.Parameters.AddWithValue("@email", cliente.Email);

cmd.Parameters.AddWithValue("@telefone", cliente.Telefone);

cn.Open();

cmd.ExecuteNonQuery();

}

catch (SqlException ex)

{

throw new Exception("Servidor SQL Erro:" + ex.Number);

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

cn.Close();

}

}

public void Excluir(int codigo)

{

//conexao

SqlConnection cn = new SqlConnection();

try
```

```
{
cn.ConnectionString = Dados.StringDeConexao;
//command
SqlCommand cmd = new SqlCommand();
cmd.Connection = cn;
cmd.CommandText = "delete from Clientes where codigo = " + codigo;
cn.Open();
int resultado = cmd.ExecuteNonQuery();
if (resultado != 1)
{
throw new Exception("Não foi possível excluir o cliente " + codigo);
}
}
catch (SqlException ex)
{
throw new Exception("Servidor SQL Erro:" + ex.Number);
}
catch (Exception ex)
{
throw new Exception(ex.Message);
}
finally
{
cn.Close();
}
}
public DataTable Listagem()
{
DataTable tabela = new DataTable();
SqlDataAdapter da = new SqlDataAdapter("select * from clientes", Dados.StringDeConexao);
da.Fill(tabela);
return tabela;
}
}
}
```

Sempre que terminamos de codificar uma classe, é bom dar um Build no projeto para nos certificarmos de que está tudo certo. Vamos fazer isso no projeto DAL.

- Clique com o botão direito sobre o projeto DAL e escolha **Build**:



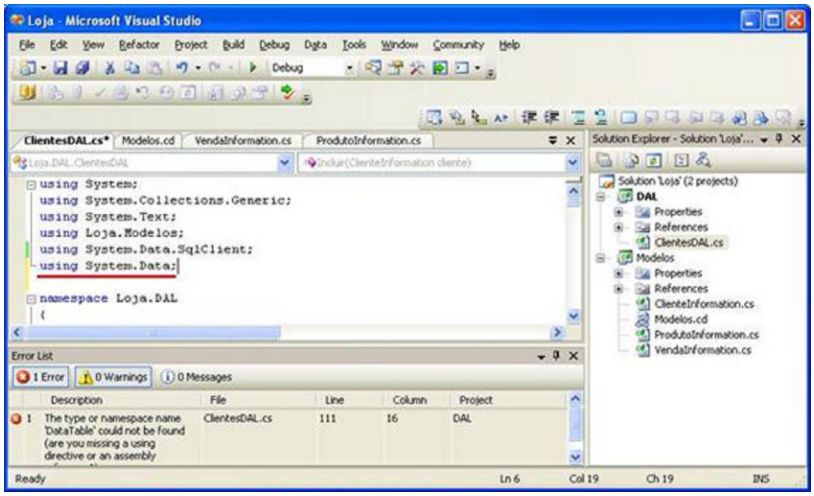
Um erro foi encontrado. Tudo bem, isso não é motivo para pânico. J

Preste atenção na descrição do erro, está escrito que o nome 'DataTable' que usamos como tipo do método Listagem() não pode ser encontrado.

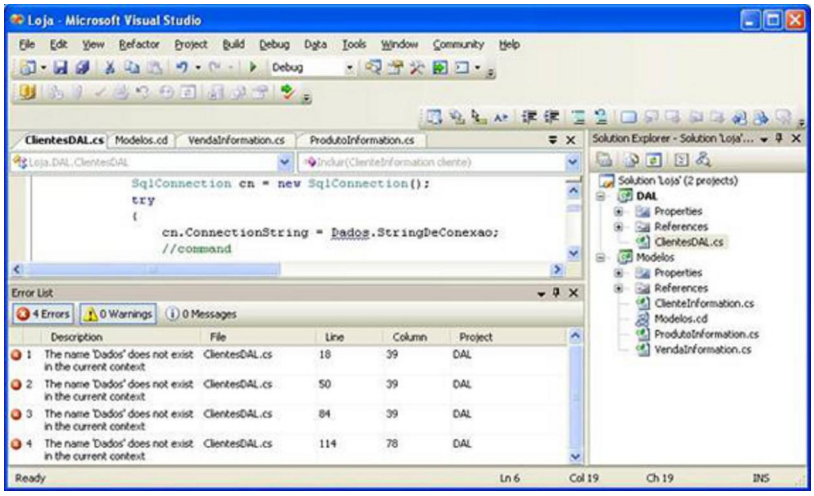
Isso ocorreu porque o tipo *DataTable* está dentro do namespace *System.Data* e nós não demos um *using* nesse namespace. É por isso que o MS Visual Studio não o encontrou no momento da compilação.

Como resolvemos o problema?

Vamos incluir a cláusula *using* para esse namespace no começo da classe:



Agora dê um **Build** novamente no projeto DAL para verificarmos o resultado:



Agora temos 4 erros para corrigir? Hmm... Olhando para a descrição vemos que na verdade é o mesmo erro que ocorreu em 4 linhas diferentes da classe.

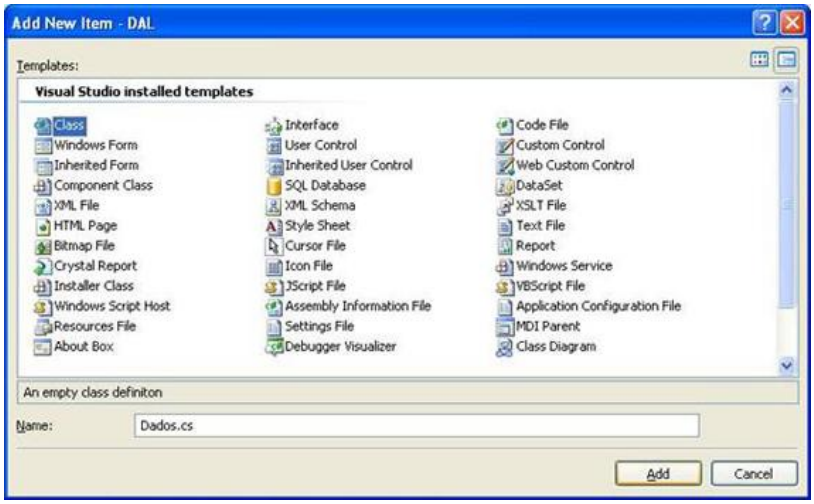
A descrição diz que o nome Dados não foi encontrado.

E está certo, realmente ainda não codificamos a nossa classe Dados que armazenará a nossa string de conexão.

Tudo bem, então vamos codificá-la agora.

- Clique com o botão direito no projeto DAL e selecione **Add** > **New Item...**

- Na janela que se abrirá, vamos escolher o template **Class** e dar o nome de **Dados.cs**. Clique no botão **Add** para adicionar a nova classe ao projeto:



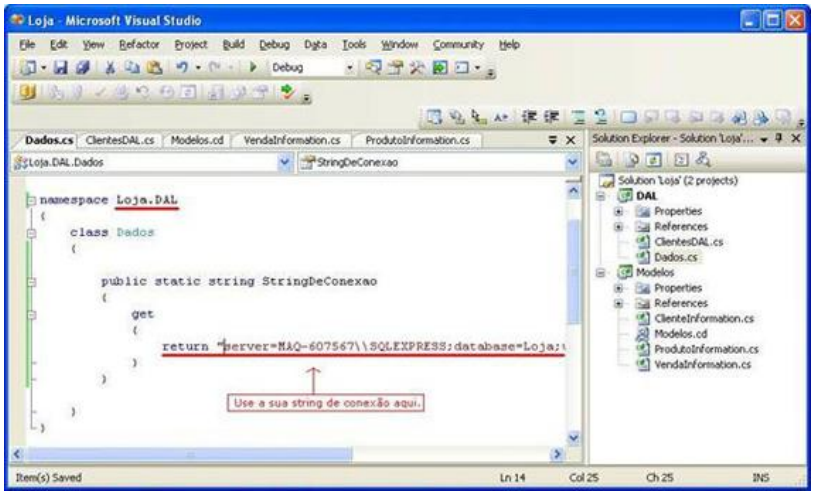
- No Solution Explorer, dê um **duplo clique** na classe **Dados.cs** para abrir o código.

- Altere o namespace para **Loja.DAL**;

- Copie e cole o código entre as chaves da classe Dados:

```
public static string StringDeConexao

{

get

{

Return "server=MAQ-607567\\SQLEXPRESS;database=Loja;user=camacho;pwd=camacho2008a3";

}

}
```

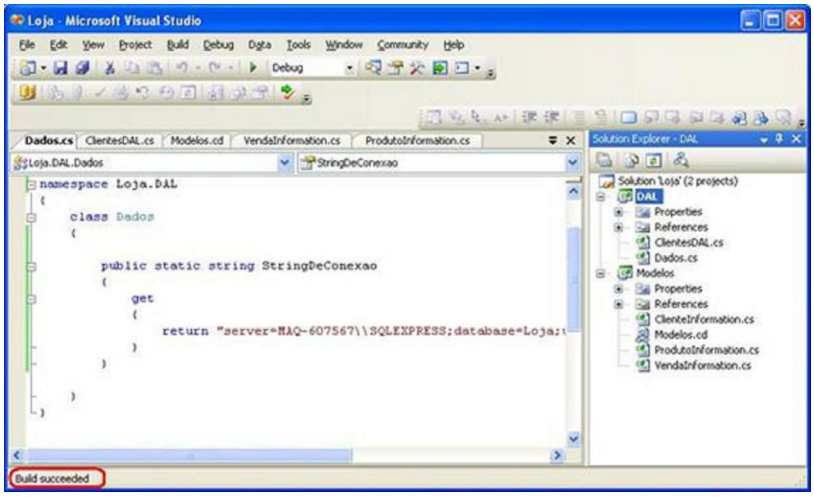
Importante: O conteúdo que você colocar entre as aspas do return deverá ser a sua string de conexão. Os dados da sua string de conexão foram definidos no artigo 2 que trata da criação da infra-estrutura de banco de dados.



O código completo da nossa classe Dados.cs ficará assim:

```
using System;

using System.Collections.Generic;

using System.Text;

namespace Loja.DAL

{

class Dados

{

public static string StringDeConexao

{

get

{

return "Use a sua string de conexão aqui.";

}

}

}

}
```

Agora vamos dar um Build no projeto DAL novamente:

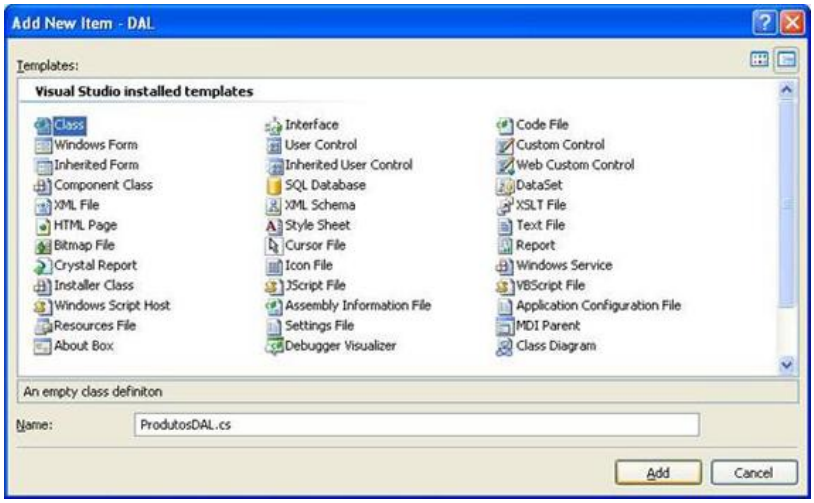


Muito bem! Podemos ver a frase "*Build succeeded*" na barra de status indicando que o nosso projeto foi compilado com sucesso.

Agora vamos implementar a classe ProdutosDAL.cs.

- Clique com o botão direito no projeto DAL, **Add** > **New Item...**

- Nomeie a classe como **ProdutosDAL.cs** e clique em **Add**:



- Copie e cole a listagem completa da classe ProdutosDAL.cs a seguir:

```
using System;

using System.Collections.Generic;

using System.Text;

using System.Data;

using System.Data.SqlClient;

using System.Collections;

using Loja.Modelos;

namespace Loja.DAL

{

public class ProdutosDAL

{

public ArrayList ProdutosEmFalta()

{

SqlConnection cn = new SqlConnection(Dados.StringDeConexao);

SqlCommand cmd = new SqlCommand("Select * from Produtos Where Estoque < 10", cn);

cn.Open();

SqlDataReader dr = cmd.ExecuteReader();

ArrayList lista = new ArrayList();

while (dr.Read())

{

ProdutoInformation produto = new ProdutoInformation();

produto.Codigo = Convert.ToInt32(dr["codigo"]);

produto.Nome = dr["nome"].ToString();

produto.Estoque = Convert.ToInt32(dr["estoque"]);

produto.Preco = Convert.ToDecimal(dr["preco"]);

lista.Add(produto);

}

dr.Close();
```

```
cn.Close();

return lista;

}

public void Incluir(ProdutoInformation produto)

{

//conexao

SqlConnection cn = new SqlConnection();

try

{

cn.ConnectionString = Dados.StringDeConexao;

//command

SqlCommand cmd = new SqlCommand();

cmd.Connection = cn;

cmd.CommandText = "insert into Produtos(nome,preco,estoque) values (@nome,@preco,@estoque);
select @@IDENTITY;";

cmd.Parameters.AddWithValue("@nome", produto.Nome);

cmd.Parameters.AddWithValue("@preco", produto.Preco);

cmd.Parameters.AddWithValue("@estoque", produto.Estoque);

cn.Open();

produto.Codigo = Convert.ToInt32(cmd.ExecuteScalar());

}

catch (SqlException ex)

{

throw new Exception("Servidor SQL Erro: " + ex.Number);

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

finally

{

cn.Close();

}

}

public void Alterar(ProdutoInformation produto)

{

//conexao

SqlConnection cn = new SqlConnection();

try
```

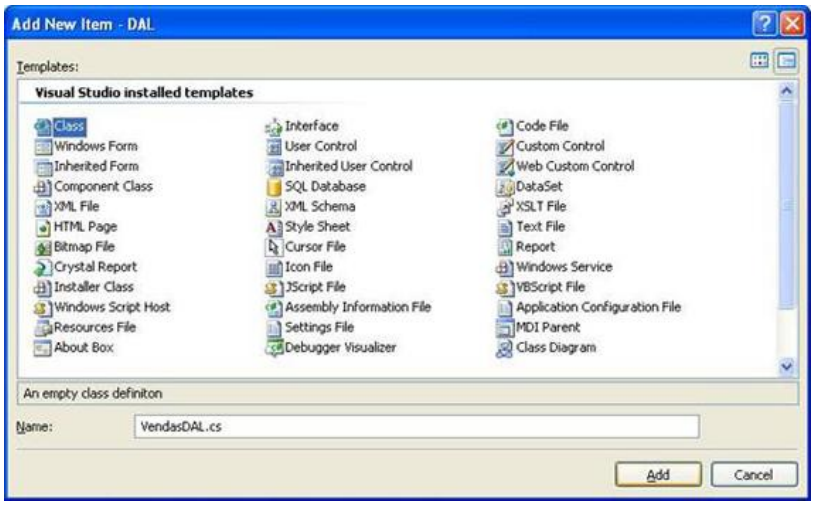
```
{
cn.ConnectionString = Dados.StringDeConexao;
//command
SqlCommand cmd = new SqlCommand();
cmd.Connection = cn;
cmd.CommandText = "AlterarProduto";
cmd.CommandType = CommandType.StoredProcedure;
cmd.Parameters.AddWithValue("@codigo", produto.Codigo);
cmd.Parameters.AddWithValue("@nome", produto.Nome);
cmd.Parameters.AddWithValue("@preco", produto.Preco);
cmd.Parameters.AddWithValue("@estoque", produto.Estoque);
cmd.Parameters.Add("@valorEstoque", SqlDbType.Int);
cmd.Parameters["@valorEstoque"].Direction = ParameterDirection.Output;
cn.Open();
cmd.ExecuteNonQuery();
decimal valorEstoque = Convert.ToDecimal(cmd.Parameters["@valorEstoque"]);
if (valorEstoque < 500)
{
throw new Exception("Atenção! Valor baixo no estoque");
}
}
catch (SqlException ex)
{
throw new Exception("Servidor SQL Erro: " + ex.Number);
}
catch (Exception ex)
{
throw new Exception(ex.Message);
}
finally
{
cn.Close();
}
}
public void Excluir(int codigo)
{
}
public DataTable Listagem()
{
```

```
DataTable tabela = new DataTable();

SqlDataAdapter da = new SqlDataAdapter("select * from produtos", Dados.StringDeConexao);

da.Fill(tabela);

return tabela;

}

}

}
```

Agora vamos implementar a classe VendasDAL.cs.

- Clique com o botão direito no projeto DAL, **Add** > **New Item...**

- Nomeie a classe como **VendasDAL.cs** e clique em **Add**:



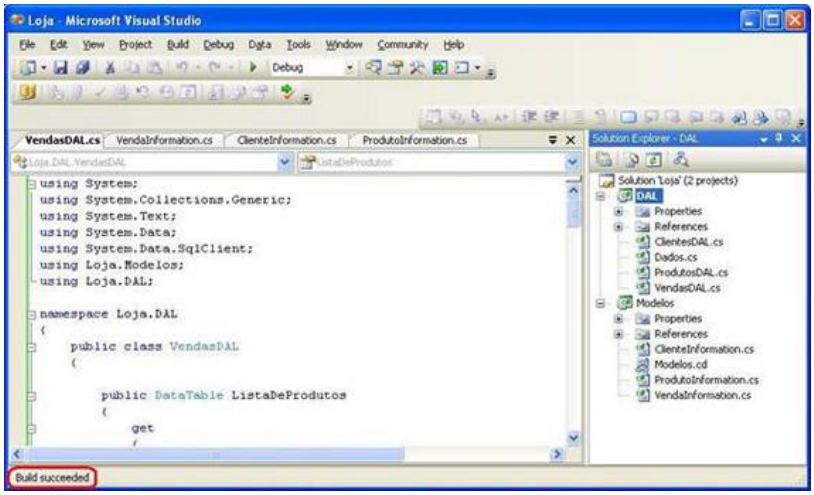
- Copie e cole a listagem completa da classe VendasDAL.cs a seguir:

```
using System;

using System.Collections.Generic;

using System.Text;

using System.Data;

using System.Data.SqlClient;

using Loja.Modelos;

using Loja.DAL;

namespace Loja.DAL

{

public class VendasDAL

{

public DataTable ListaDeProdutos

{

get

{

SqlConnection cn = new SqlConnection();

cn.ConnectionString = Dados.StringDeConexao;
```

```
cn.Open();
SqlDataAdapter da = new SqlDataAdapter("select * from produtos", cn);
DataTable dt = new DataTable();
da.Fill(dt);
cn.Close();
return dt;
}
}
//Propriedade que retorna uma Lista de Clientes
public DataTable ListaDeClientes
{
get
{
SqlConnection cn = new SqlConnection();
cn.ConnectionString = Dados.StringDeConexao;
cn.Open();
SqlDataAdapter da = new SqlDataAdapter("select * from clientes", cn);
DataTable dt = new DataTable();
da.Fill(dt);
cn.Close();
return dt;
}
}
public void Incluir(VendaInformation venda)
{
//conexao
SqlConnection cn = new SqlConnection();
SqlTransaction t = null;
try
{
cn.ConnectionString = Dados.StringDeConexao;
//command
SqlCommand cmd1 = new SqlCommand();
cmd1.Connection = cn;
cmd1.CommandText = @"insert into vendas
(CodigoCliente,
CodigoProduto,
Data,
Quantidade,
```

```
Faturado)
VALUES
(@CodigoCliente,
@CodigoProduto,
@Data,
@Quantidade,
@Faturado);select @@IDENTITY;";
SqlCommand cmd2 = new SqlCommand();
cmd2.Connection = cn;
cmd2.CommandText = @"Update Produtos
Set Estoque = Estoque - @Quantidade
Where Codigo=@CodigoProduto";
cn.Open();
t = cn.BeginTransaction(IsolationLevel.Serializable);//default
cmd1.Transaction = t;
cmd2.Transaction = t;
cmd1.Parameters.AddWithValue("@codigocliente", venda.CodigoCliente);
cmd1.Parameters.AddWithValue("@codigoproduto", venda.CodigoProduto);
cmd1.Parameters.AddWithValue("@data", venda.Data);
cmd1.Parameters.AddWithValue("@quantidade", venda.Quantidade);
cmd1.Parameters.AddWithValue("@faturado", venda.Faturado);
cmd2.Parameters.AddWithValue("@quantidade", venda.Quantidade);
cmd2.Parameters.AddWithValue("@codigoproduto", venda.CodigoProduto);
venda.Codigo = Convert.ToInt32(cmd1.ExecuteScalar());
cmd2.ExecuteNonQuery();
t.Commit();
}
catch (Exception ex)
{
t.Rollback();
throw new Exception("Servidor no Servidor:" + ex.Message);
}
finally
{
cn.Close();
}
}
}
}
```

- Agora, clicando com o botão direito do mouse sobre o projeto DAL, dê um **Build** para compilar o projeto:



A frase so rodapé indica que a compilação foi realizada com sucesso.

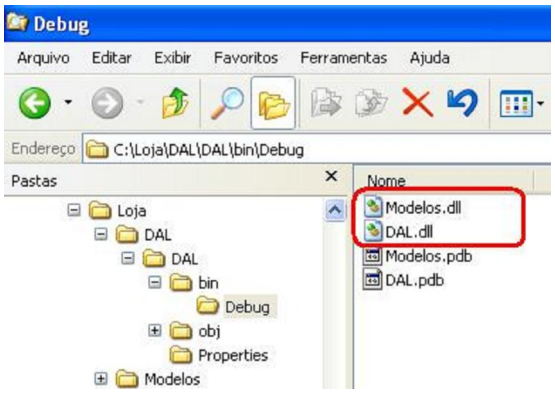
Parabéns! Acabamos de implementar todas as classes do nosso projeto DAL – Camada de Acesso à Dados.

Vou comentar brevemente alguns pontos interessantes sobre as classes que implementamos neste artigo.

· No método Inserir da classe VendasDAL estamos usando uma transação. A inserçaõ de uma venda só vai ocorrer se dois comandos forem concluídos com sucesso: a inserção na tabela de vendas e a atualização do estoque do produto.

· Quando criamos a nossa classe Dados usamos a palavra *static*. Com isso nós não precisamos instanciar um objeto para ter acesso a propriedade *StringDeConexao*. Se você precisar criar uma propriedade deste tipo no VB .Net, pode usar a sintaxe: *Public Shared ReadOnly Property*

· Se você abrir o projeto DAL no Windows Explorer, poderá verificar que na pasta de DLLs do projeto existem dois arquivos com a extensão DLL. Um do próprio projeto DAL e outro para a referência que fizemos ao projeto Modelos.



Compare isso com o nosso modelo em camadas a seguir:

[Modelos] DAL ð BLL ð User Interface

No próximo artigo onde implementaremos a camada BLL (regras de negócio), teremos na pasta do projeto BLL três arquivos dll. Assim podemos distinguir claramente as nossas camadas pois nesta solução elas estarão separadas em arquivos.

A idéia é aproveitarmos o código desenvolvido nas camadas anteriores. Mais adiante veremos que quando for necessário fazer manutenção na camada de regras de negócio, por exemplo, precisaremos abrir e atualizar apenas o projeto BLL.

Até o próximo artigo!

Carlos Camacho - Carlos Olavo de Azevedo Camacho Júnior é mestrando em Tecnologias da Inteligência e Design Digital pela Pontifícia Universidade Católica de São Paulo PUCSP. Pós-graduado em Análise e Projeto de Sistemas pela Universidade Paulista UNIP. Bacharel em Ciência da Computação pela Universidade Paulista UNIP e possui Licenciatura Plena em Matemática pelas Faculdades Oswaldo Cruz.
MCP .Net, MCP SQL Server, Carlos Camacho leciona disciplinas técnicas na área de Ciências Exatas e é Consultor em Tecnologia da Informação para Instituições Financeiras.

Leia também

Lambda Expressions x SQL: Comparando a sintaxe de consultas comuns
C#

List: trabalhando com listas genéricas em C#
C#

Criando Gráficos usando C# e API do Google
C#

Imprimindo um Panel com C#
C#

Consumindo um Web API em C#
C#