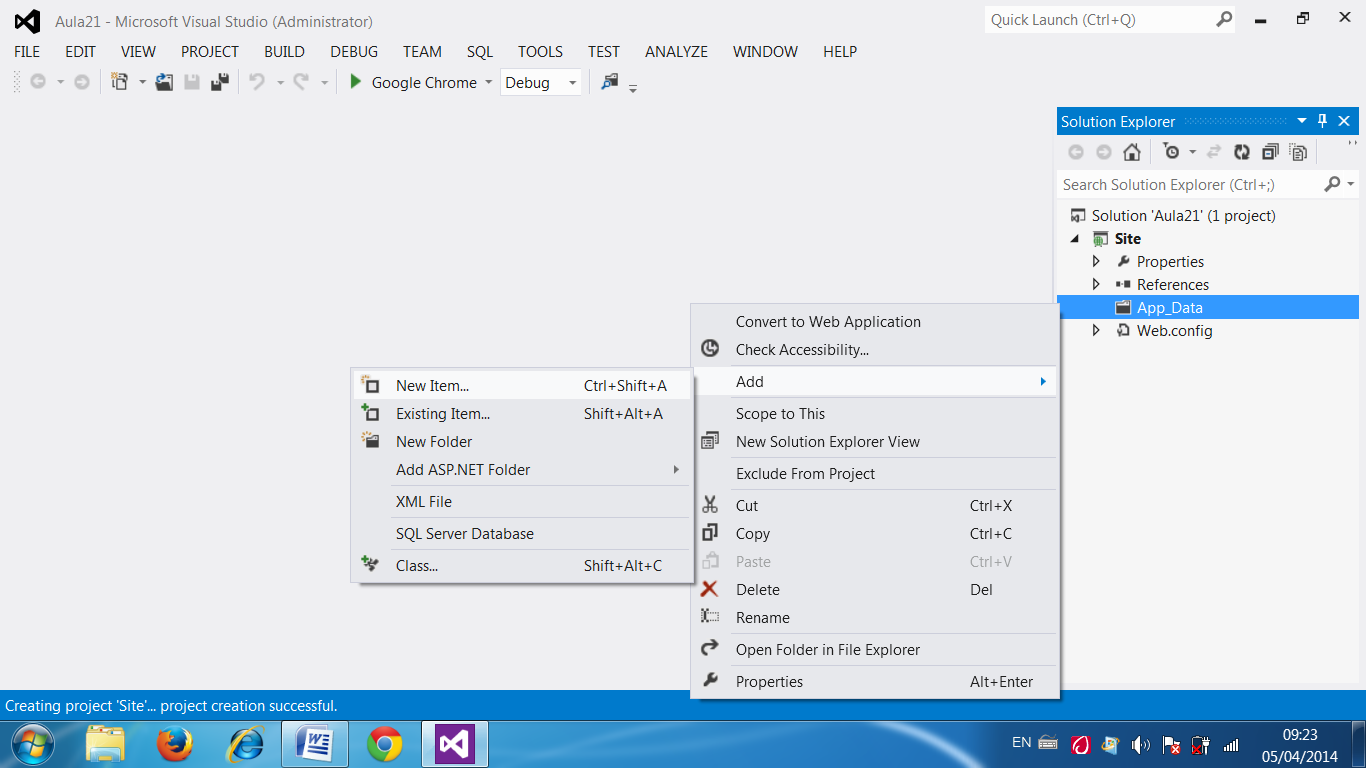
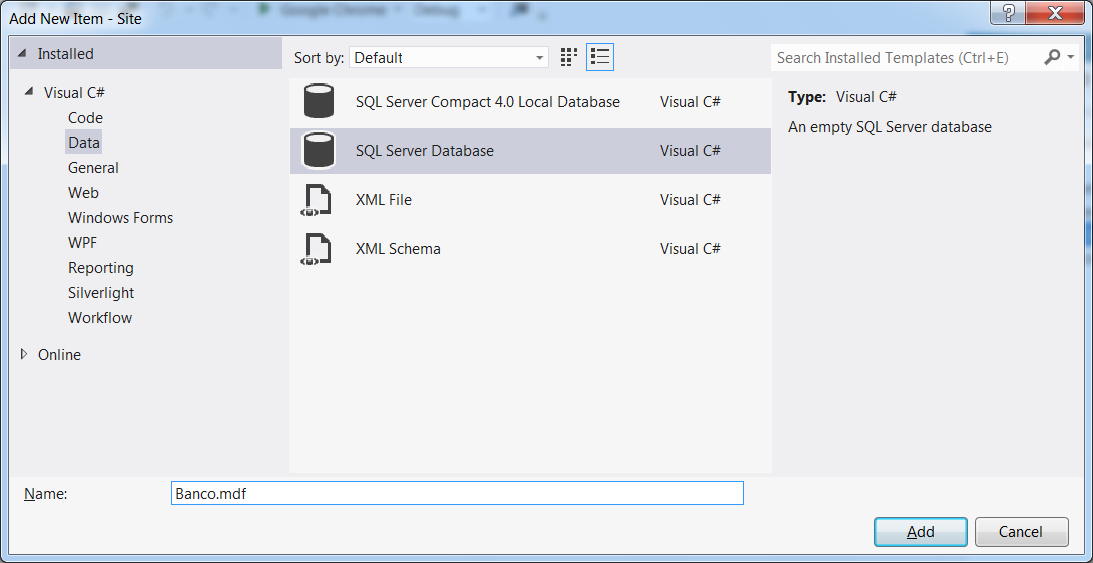
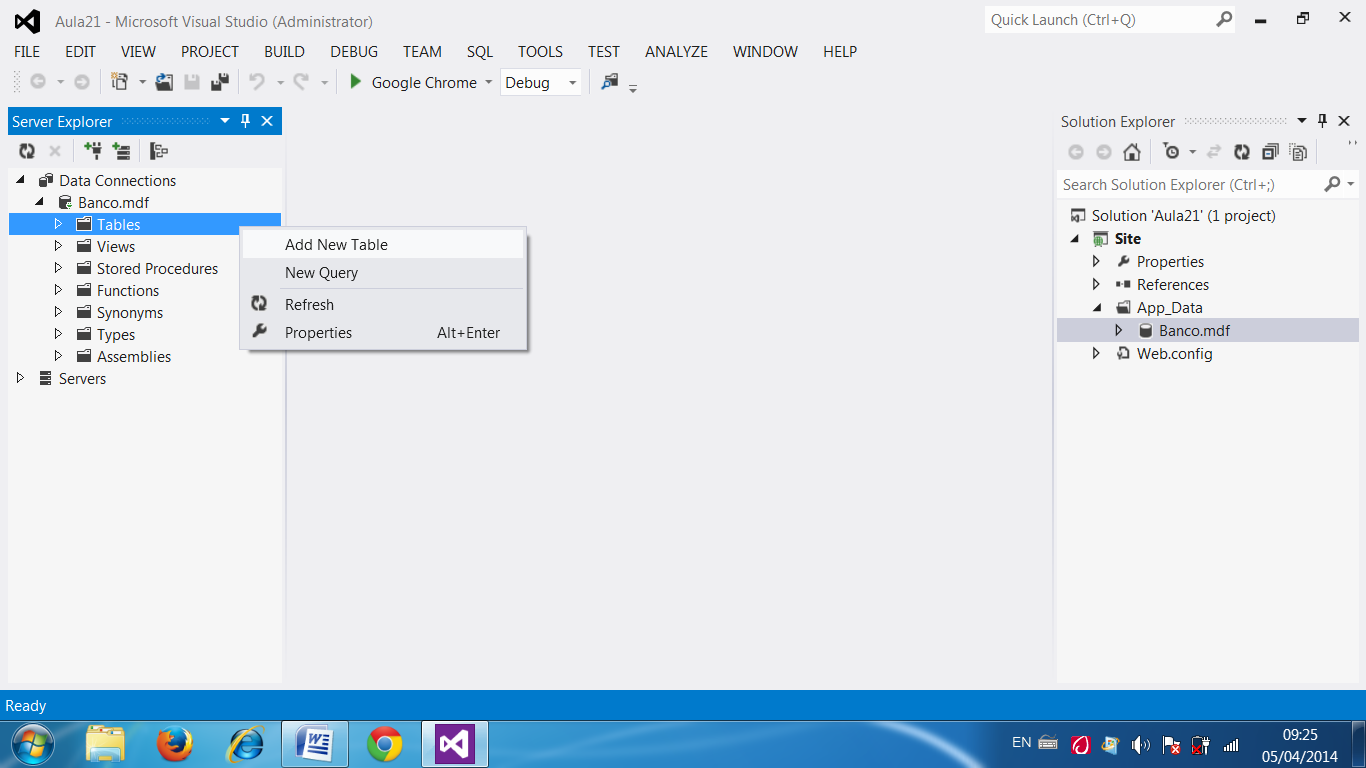


Criando a base de dados...





Criando a tabela de Clientes



Script do banco de dados...

go

create table Cliente(

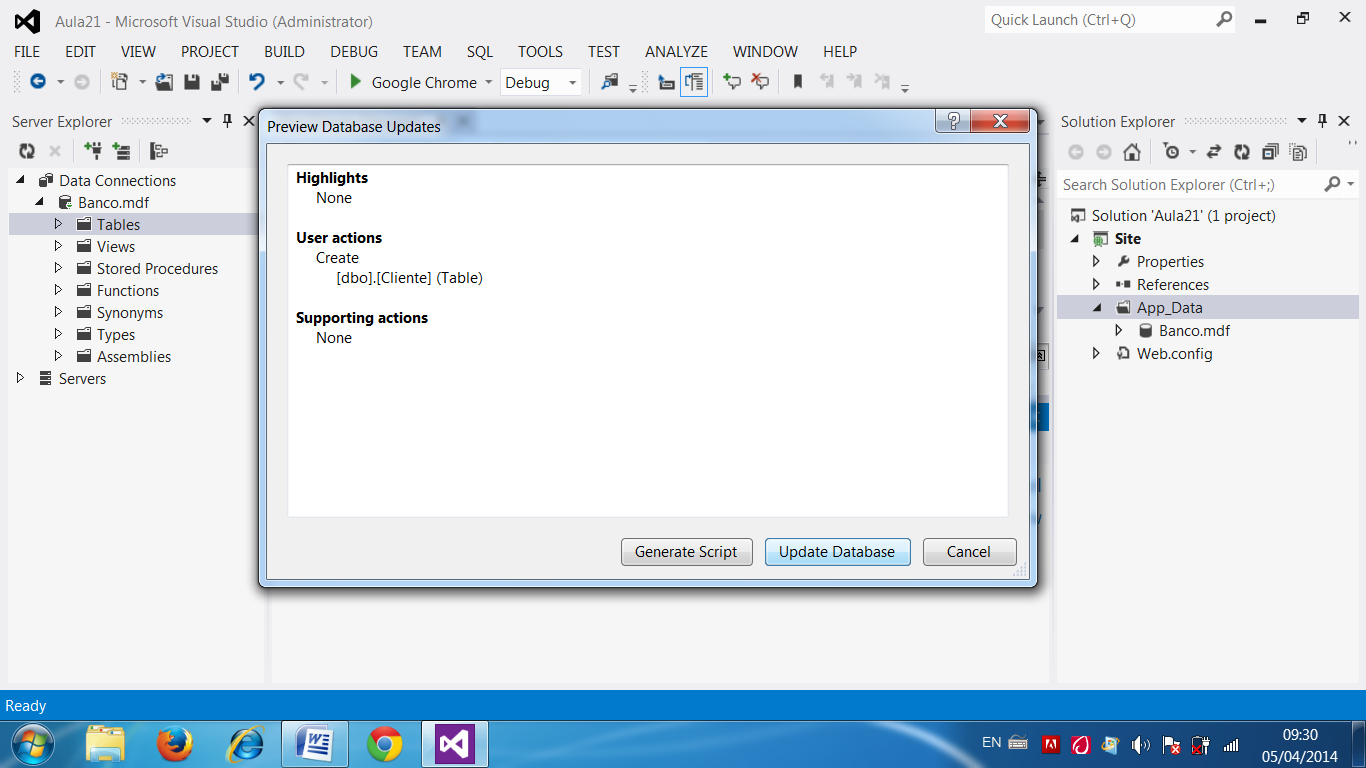
IdCliente integer identity(1,1),

Nome nvarchar(50) not null,

Email nvarchar(50) not null unique,

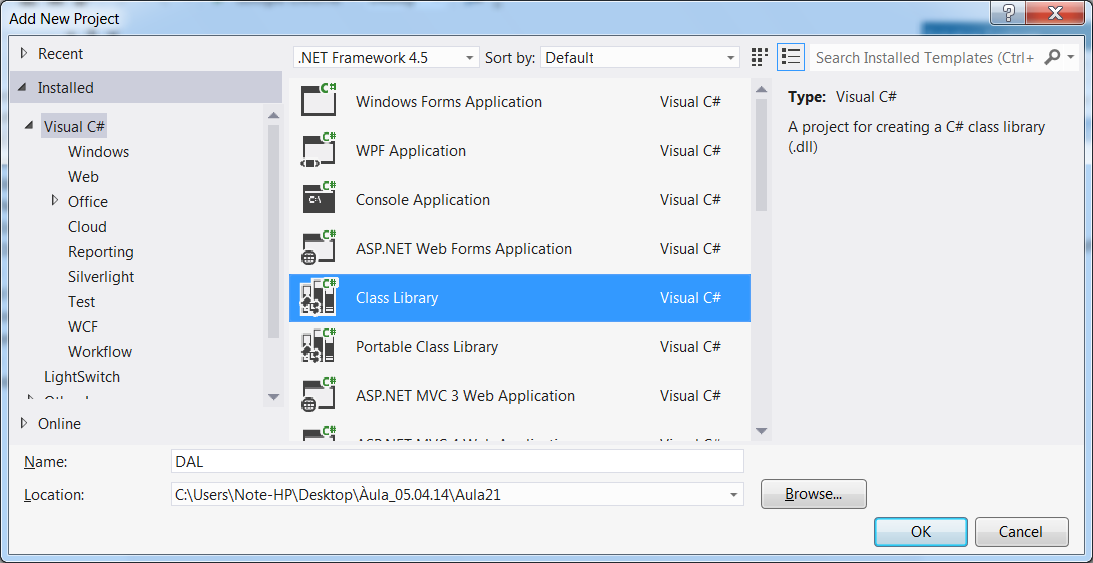
DataCadastro datetime not null,

primary key(IdCliente))



Data Access Layer (DAL)

Camada de acesso a dados

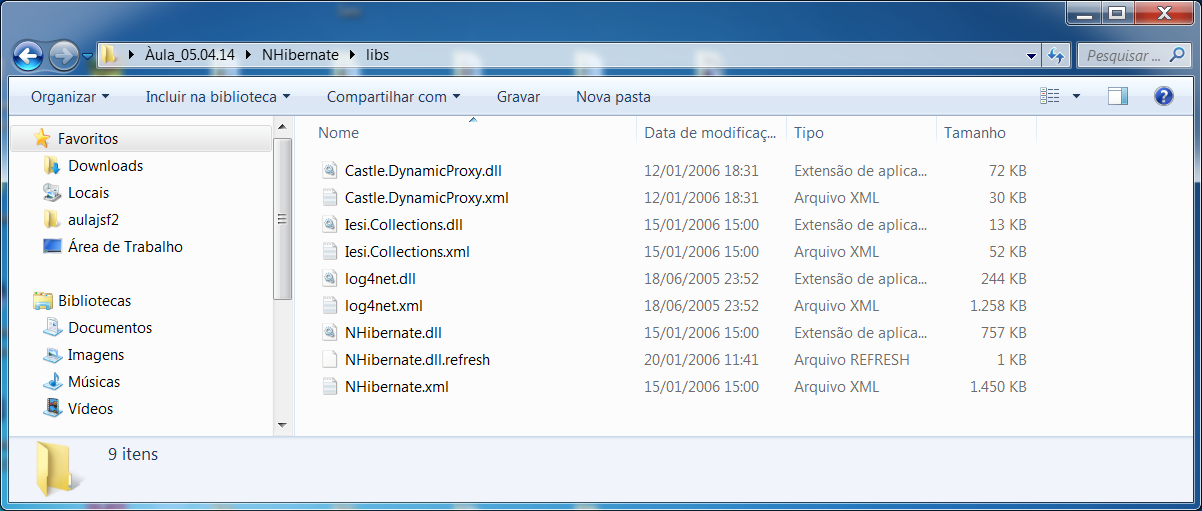


Hibernate

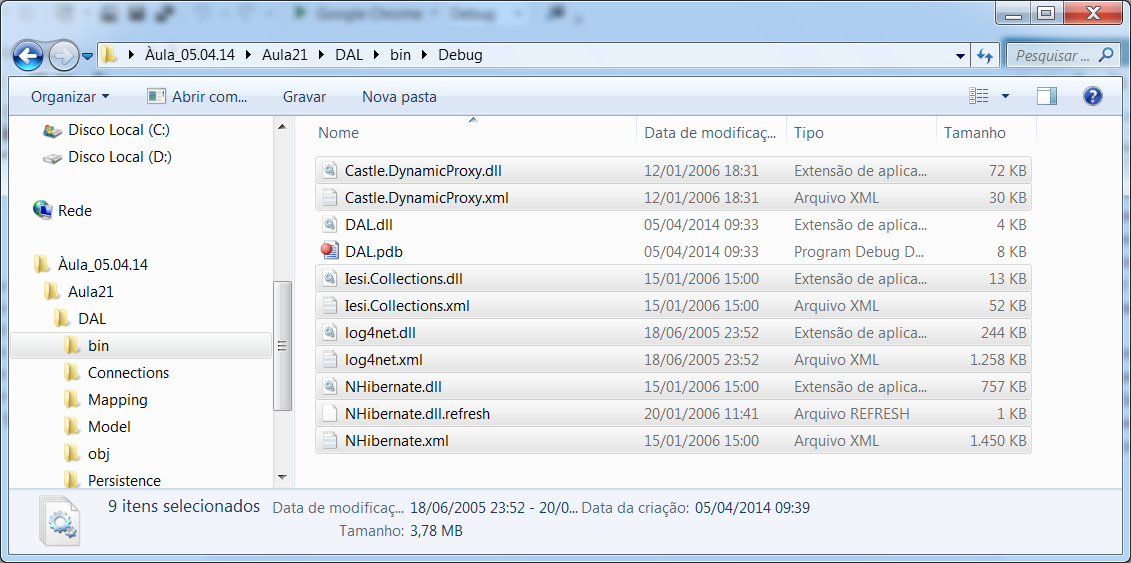
Framework Java para acesso a bases de dados. Possui versão escrita para .NET denominada de NHibernate.

<http://hibernate.org/>

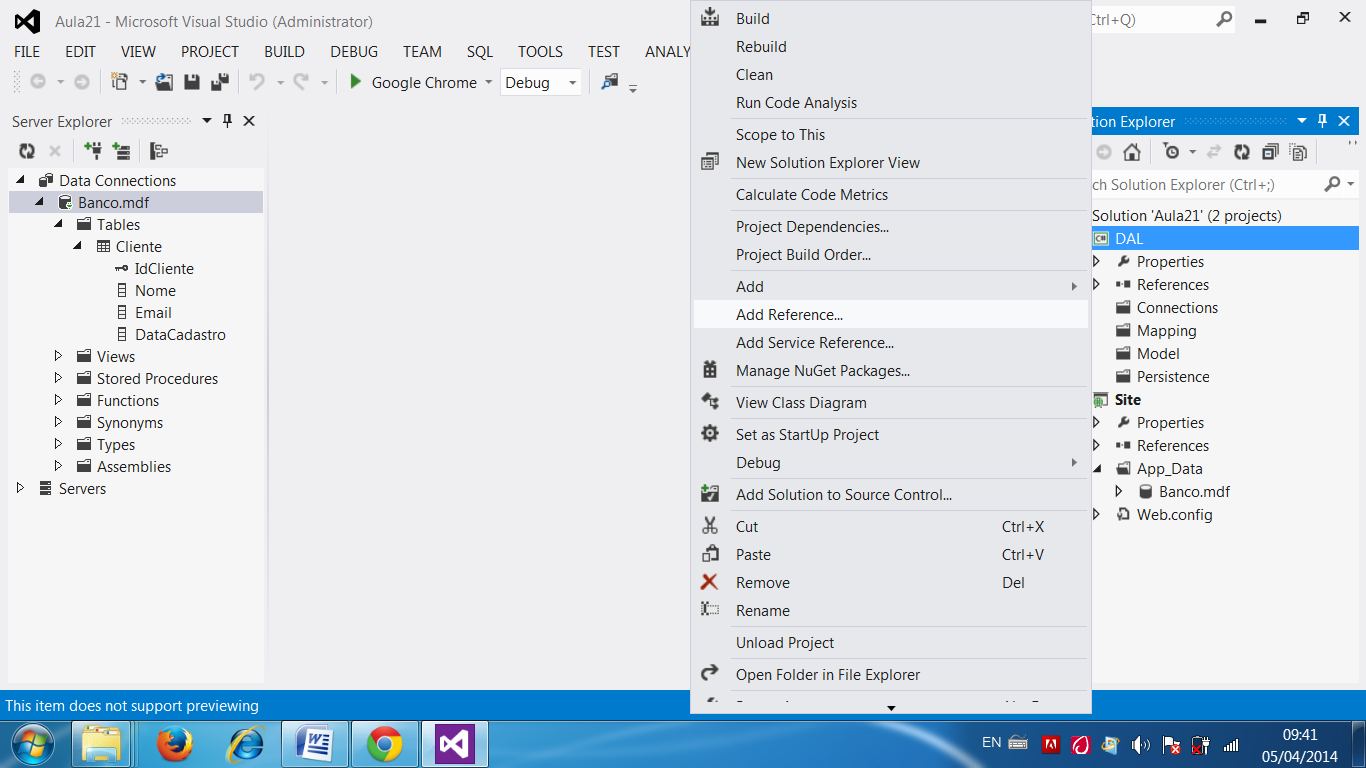
Bibliotecas do NHibernate...

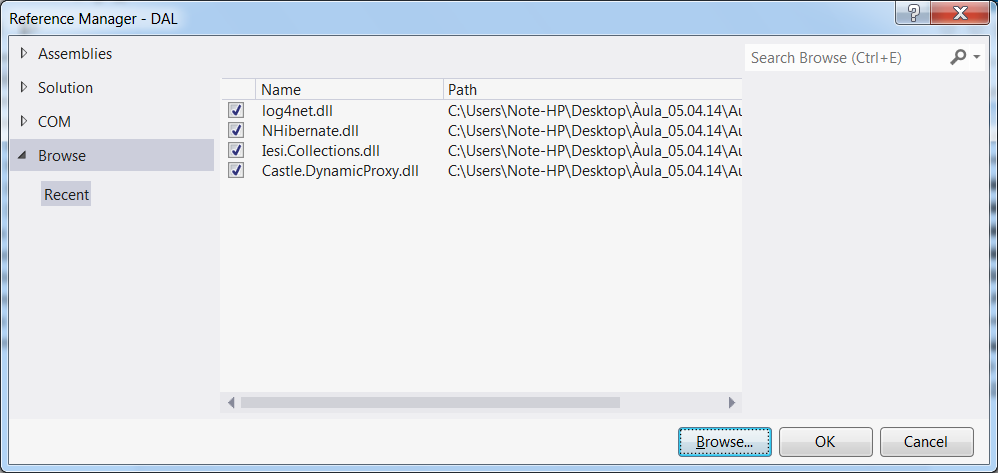


Colar: Aula21\DAL\bin\Debug



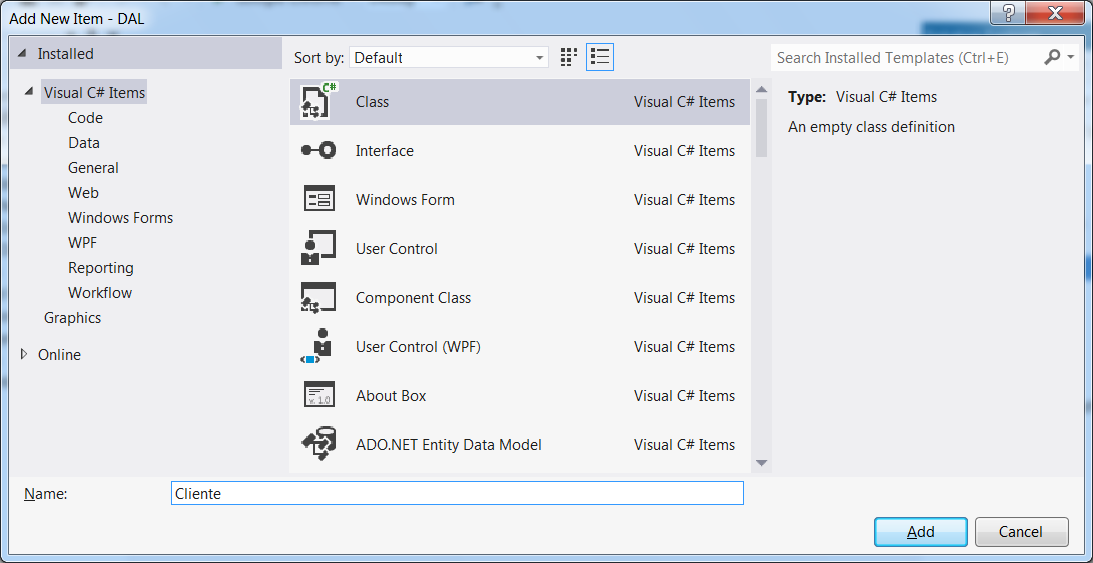
Adicionar referências...





Parte 1:

Modelagem das entidades na forma de Classes



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DAL.Model

{

public class Cliente

{

public int IdCliente { get; set; }

public string Nome { get; set; }

public string Email { get; set; }

public DateTime DataCadastro { get; set; }

}

}

Mapeamento Objeto Relacional

Arquivos XML (hbm 🡪 hibernate mapping file)

/Mapping/Cliente.hbm.xml

<?xml version="1.0" encoding="utf-8" ?>

<hibernate-mapping xmlns="urn:nhibernate-mapping-2.0">

<!-- Mapeamento da Classe / Tabela -->

<class name="DAL.Model.Cliente, DAL" table="Cliente">

<!-- chave primária -->

<id name="IdCliente" column="IdCliente" type="Int32">

<generator class="identity"/> <!-- Geração da chave primaria -->

</id>

<!-- demais colunas da tabela -->

<property name="Nome" column="Nome" type="String"

not-null="true" length="50"/>

<property name="Email" column="Email" type="String"

not-null="true" length="50" unique="true"/>

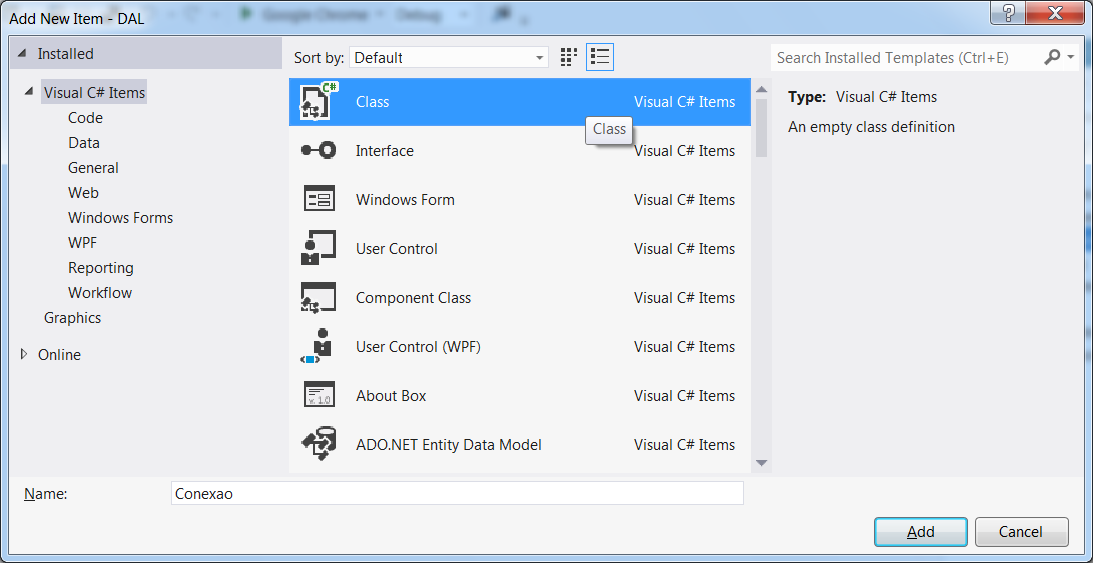
<property name="DataCadastro" column="DataCadastro"

type="DateTime" not-null="true"/>

</class>

</hibernate-mapping>

Classe para realizar a conexão com o banco de dados



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NHibernate;

using NHibernate.Cfg;

namespace DAL.Connections

{

public class Conexao

{

}

}

O Hibernate gera conexões em uma base de dados através de um componente chamado de ISessionFactory (Fábrica de Conexões)

* ISessionFactory (Fábrica de Conexões)
  + hibernate.cfg.xml (mapeamento da conexão)
  + hbm.xml (mapeamento das entidades)

**hibernate.cfg.xml**

<?xml version="1.0" encoding="utf-8" ?>

<hibernate-configuration xmlns="urn:nhibernate-configuration-2.0">

<session-factory>

<property name="connection.provider">

NHibernate.Connection.DriverConnectionProvider

</property>

<property name="dialect">

NHibernate.Dialect.MsSql2000Dialect

</property>

<property name="connection.driver\_class">

NHibernate.Driver.SqlClientDriver

</property>

<property name="connection.connection\_string">

Data Source=(LocalDB)\v11.0;AttachDbFilename=C:\Users\Note-HP\Desktop\Àula\_05.04.14\Aula21\Site\App\_Data\Banco.mdf;Integrated Security=True

</property>

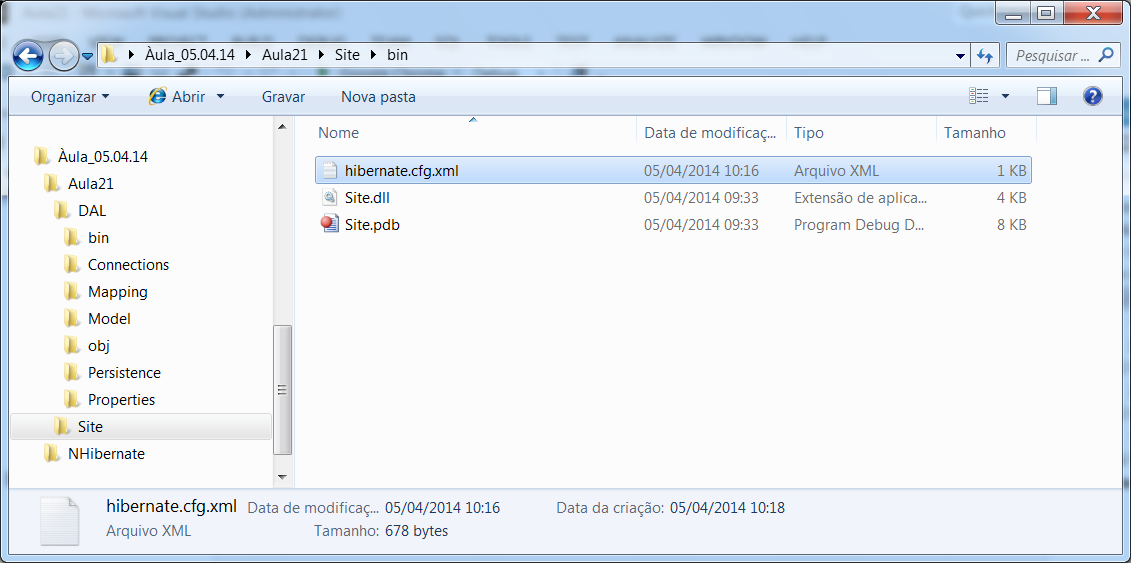
</session-factory>

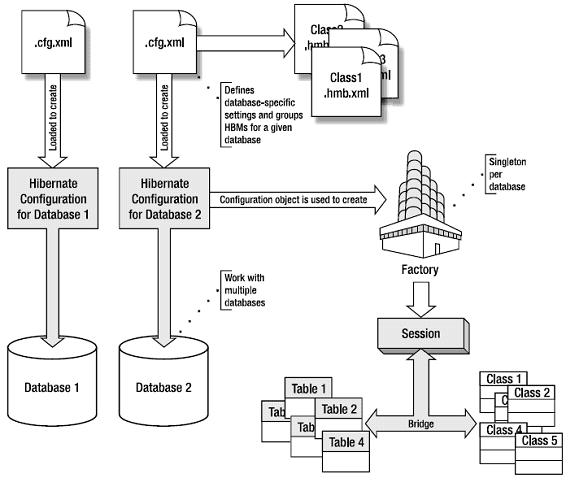
</hibernate-configuration>

Regra:

Este arquivo hibernate.cfg.xml deverá estar armazenado dentro da pasta **\bin** do projeto principal...

Aula21\Site\bin





using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NHibernate;

using NHibernate.Cfg;

namespace DAL.Connections

{

//static -> já garante que a classe estará alocada em memória

public static class Conexao

{

//Método para gerar a fábrica de conexões do Hibernate

public static ISessionFactory GetSessionFactory()

{

try

{

Configuration cfg = new Configuration();

cfg.Configure(); //lê o arquivo 'hibernate.cfg.xml'

cfg.AddAssembly("DAL");

//ler todos os arquivos .hbm.xml contidos no projeto DAL

//Gerar a fábrica de conexões...

return cfg.BuildSessionFactory();

}

catch(Exception e)

{

throw new Exception("Erro ao gerar fábrica de conexões: "

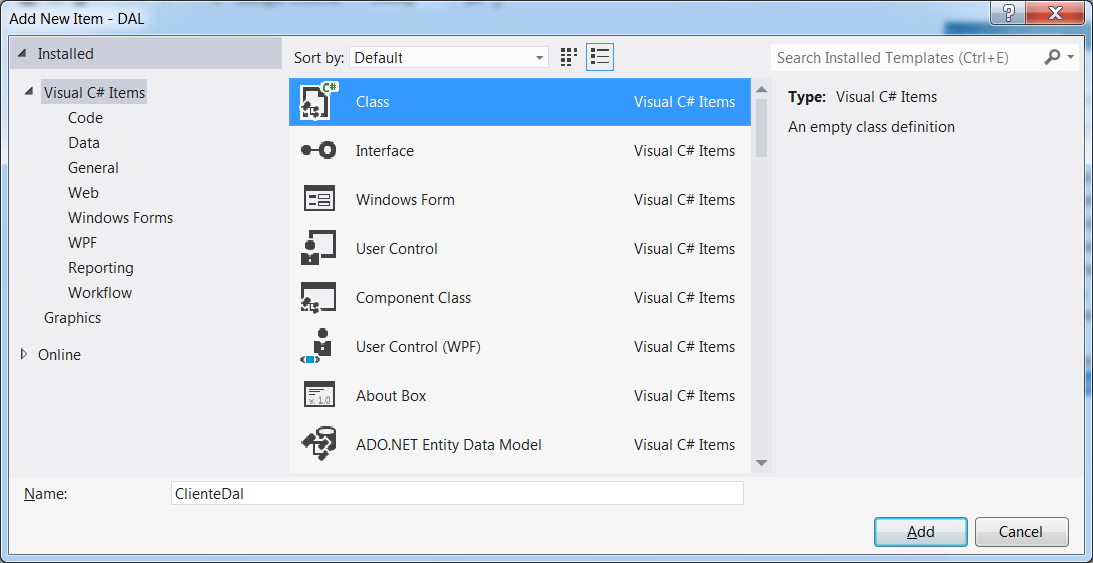
+ e.Message);

}

}

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NHibernate;

using DAL.Connections;

using DAL.Model;

namespace DAL.Persistence

{

public class ClienteDal

{

//Atributos

private ISession session; //SqlConnection Con

private ITransaction transaction; //SqlCommand Cmd

private IQuery query; //SqlDataReader Dr

//Método para gravar Cliente na base

public void Salvar(Cliente c)

{

try

{

session = Conexao.GetSessionFactory().OpenSession();

//conexão aberta

transaction = session.BeginTransaction();

//iniciando uma transação

session.Save(c); //insert

transaction.Commit(); //executar

}

catch(Exception e)

{

transaction.Rollback(); //desfazer

throw new Exception("Erro ao salvar cliente: " + e.Message);

}

finally

{

session.Close(); //fechar a conexão

}

}

//Método para excluir Cliente na base

public void Excluir(Cliente c)

{

try

{

session = Conexao.GetSessionFactory().OpenSession();

//conexão aberta

transaction = session.BeginTransaction();

//iniciando uma transação

session.Delete(c); //delete

transaction.Commit(); //executar

}

catch (Exception e)

{

transaction.Rollback(); //desfazer

throw new Exception("Erro ao excluir cliente: " + e.Message);

}

finally

{

session.Close(); //fechar a conexão

}

}

//Método para atualizar Cliente na base

public void Atualizar(Cliente c)

{

try

{

session = Conexao.GetSessionFactory().OpenSession();

//conexão aberta

transaction = session.BeginTransaction();

//iniciando uma transação

session.Update(c); //update

transaction.Commit(); //executar

}

catch (Exception e)

{

transaction.Rollback(); //desfazer

throw new Exception("Erro ao atualizar cliente: " + e.Message);

}

finally

{

session.Close(); //fechar a conexão

}

}

}

}

HQL – Hibernate Query Language

Linguagem do Hibernate para consulta de dados baseado em Objetos

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NHibernate;

using DAL.Connections;

using DAL.Model;

namespace DAL.Persistence

{

public class ClienteDal

{

//Atributos

private ISession session; //SqlConnection Con

private ITransaction transaction; //SqlCommand Cmd

private IQuery query; //SqlDataReader Dr

//Método para gravar Cliente na base

public void Salvar(Cliente c)

{

try

{

session = Conexao.GetSessionFactory().OpenSession();

//conexão aberta

transaction = session.BeginTransaction();

//iniciando uma transação

session.Save(c); //insert

transaction.Commit(); //executar

}

catch(Exception e)

{

transaction.Rollback(); //desfazer

throw new Exception("Erro ao salvar cliente: " + e.Message);

}

finally

{

session.Close(); //fechar a conexão

}

}

//Método para excluir Cliente na base

public void Excluir(Cliente c)

{

try

{

session = Conexao.GetSessionFactory().OpenSession();

//conexão aberta

transaction = session.BeginTransaction();

//iniciando uma transação

session.Delete(c); //delete

transaction.Commit(); //executar

}

catch (Exception e)

{

transaction.Rollback(); //desfazer

throw new Exception("Erro ao excluir cliente: " + e.Message);

}

finally

{

session.Close(); //fechar a conexão

}

}

//Método para atualizar Cliente na base

public void Atualizar(Cliente c)

{

try

{

session = Conexao.GetSessionFactory().OpenSession();

//conexão aberta

transaction = session.BeginTransaction();

//iniciando uma transação

session.Update(c); //update

transaction.Commit(); //executar

}

catch (Exception e)

{

transaction.Rollback(); //desfazer

throw new Exception("Erro ao atualizar cliente: " + e.Message);

}

finally

{

session.Close(); //fechar a conexão

}

}

//Método para obter 1 Cliente pelo Id

public Cliente ObterPorId(int IdCliente)

{

try

{

session = Conexao.GetSessionFactory().OpenSession();

//conexão aberta

//SQL => select \* from Cliente where IdCliente = @v1

return session.Get(typeof(Cliente), IdCliente) as Cliente;

}

catch(Exception e)

{

throw new Exception("Erro ao obter cliente por id: "

+ e.Message);

}

finally

{

session.Close();

}

}

//Método para listar todos os clientes cadastrados na base

public List<Cliente> ListarTodos()

{

try

{

session = Conexao.GetSessionFactory().OpenSession();

//abrir conexão

//SQL => select \* from Cliente order by DataCadastro desc

query = session.CreateQuery("select c from Cliente as c

order by c.DataCadastro desc"); //HQL

//executar e obter o resultado da consulta

var dados = query.List(); //IList

//retornar os dados obtidos como List<Cliente>

return dados.OfType<Cliente>().ToList();

}

catch(Exception e)

{

throw new Exception("Erro ao listar todos: " + e.Message);

}

finally

{

session.Close(); //fechar conexão

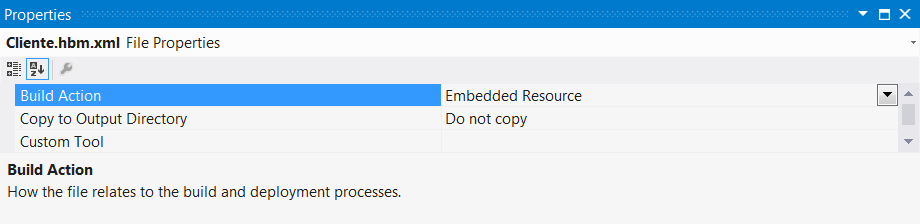
}

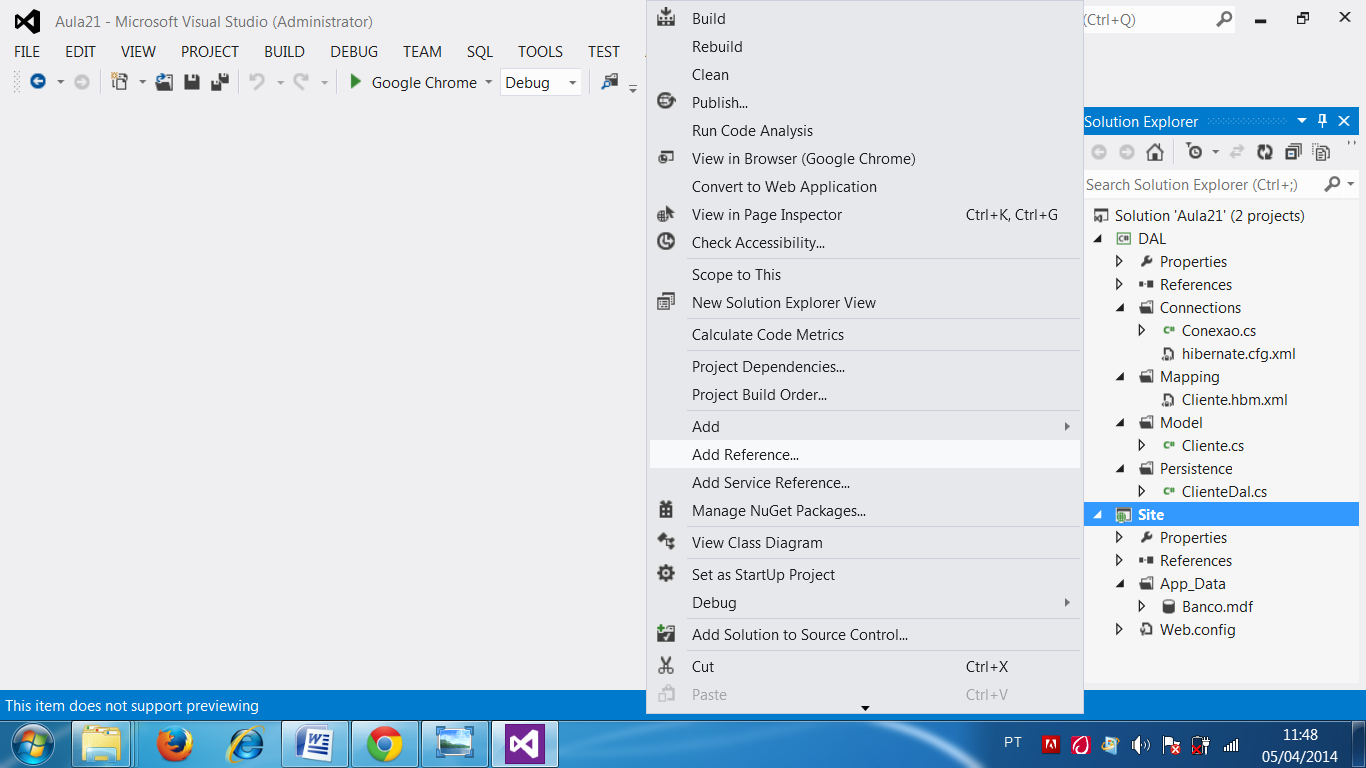
}

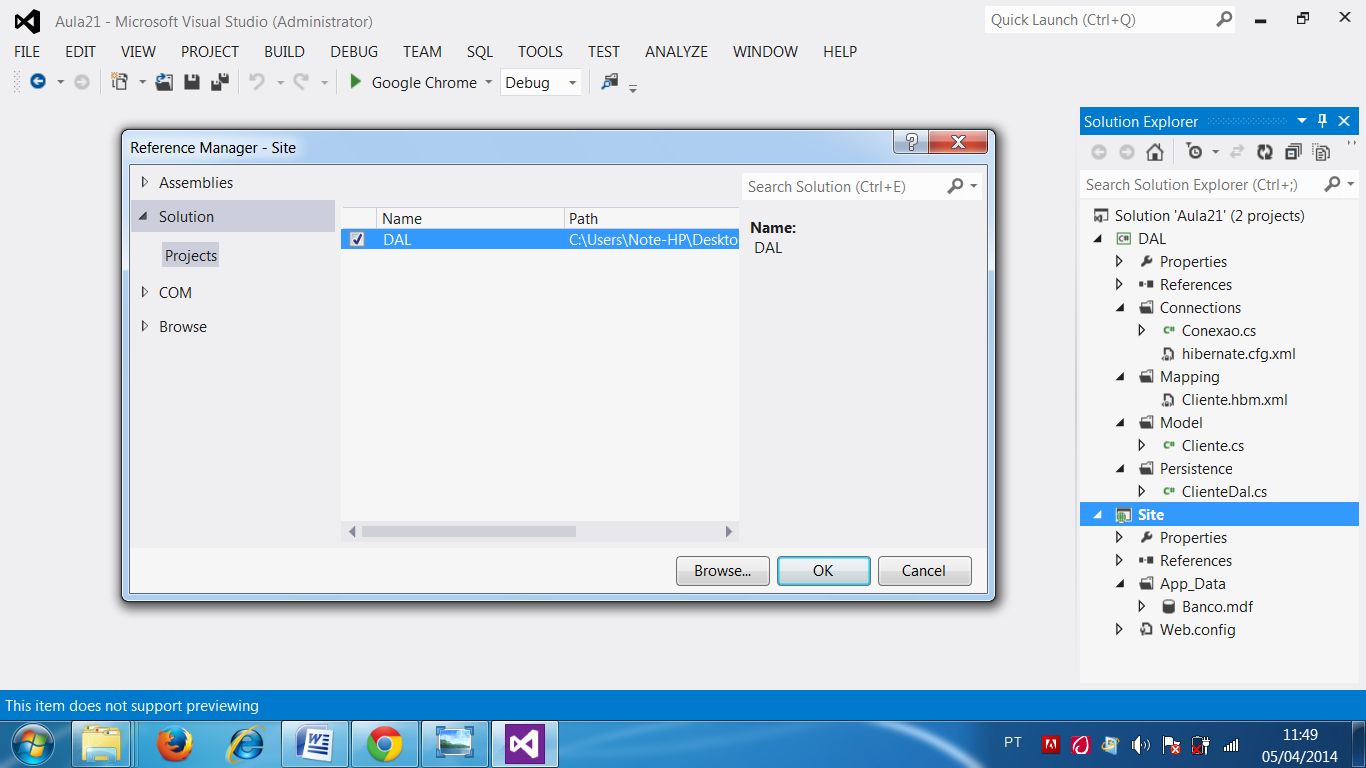
}

}

Importante: Cada Arquivo **hbm.xml** localizado dentro do projeto DAL deverá ser marcado como EmbeddedResource

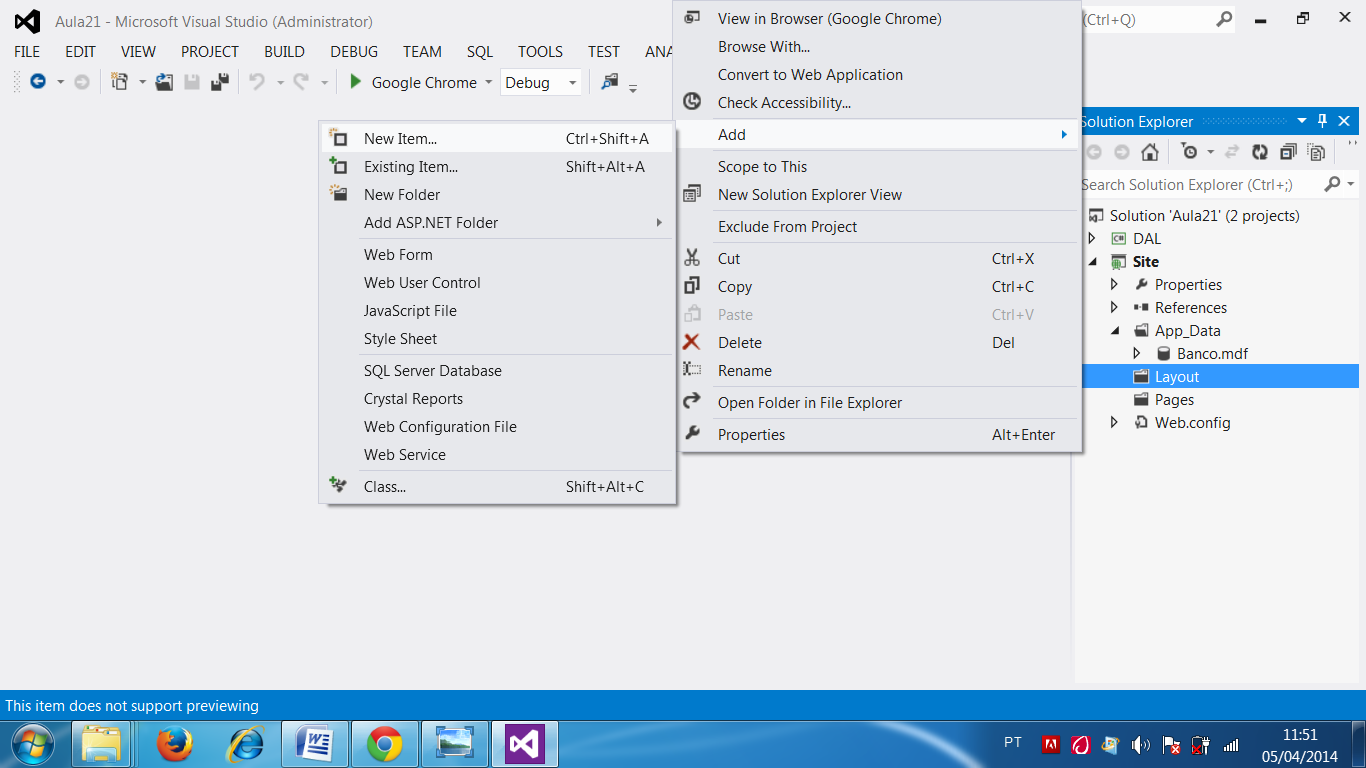


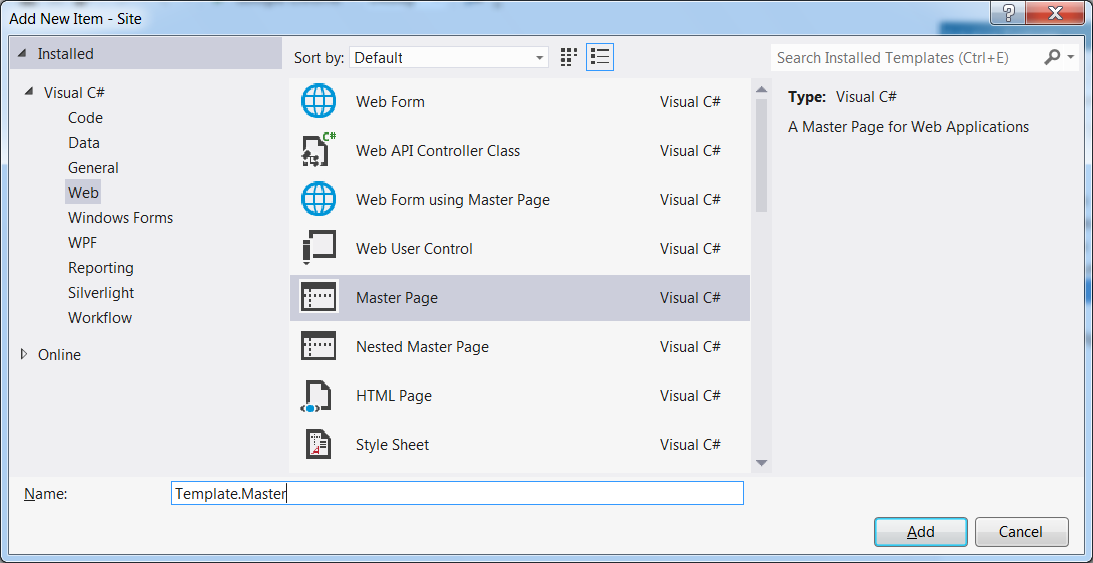




MasterPage

Página de layout para o projeto...





<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Template.master.cs" Inherits="Site.Layout.Template" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

<style type="text/css">

body /\* formatação da tag <body> \*/

{

font-family: 'Trebuchet MS'; /\* tipo da fonte \*/

font-size: 10pt; /\* tamanho da fonte \*/

padding: 30px; /\* margem interna \*/

}

</style>

</head>

<body>

<form id="form1" runat="server">

<div>

<h3>Projeto Asp.Net Hibernate</h3>

<hr />

<!-- Local da MasterPage onde será incluido o conteudo

das demais páginas -->

**<asp:ContentPlaceHolder ID="Conteudo" runat="server">**

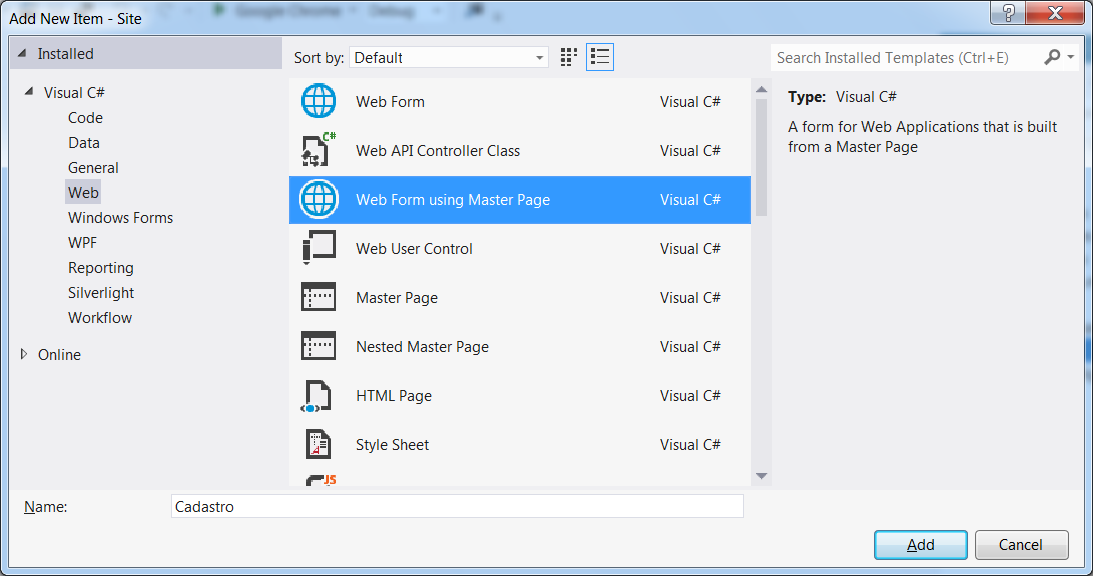
**</asp:ContentPlaceHolder>**

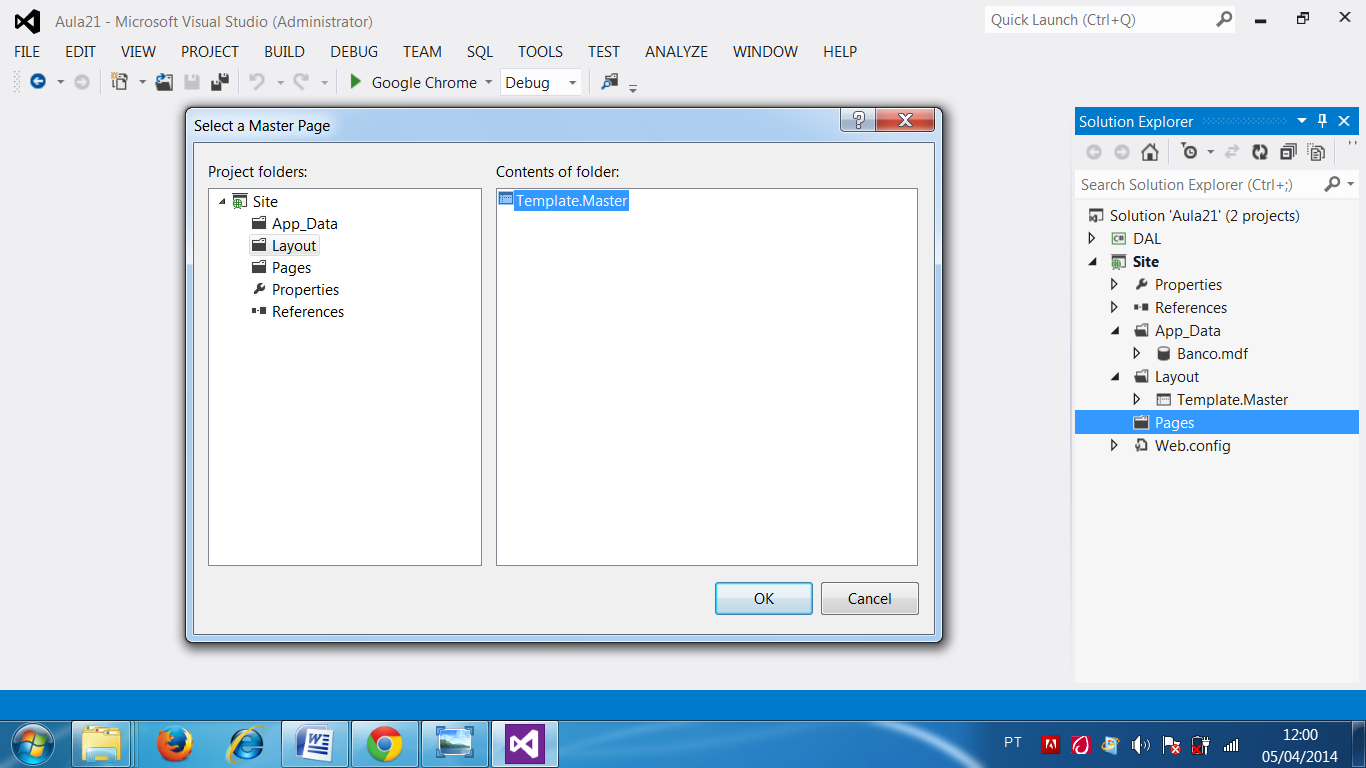
</div>

</form>

</body>

</html>





<%@ Page Title="" Language="C#" MasterPageFile="~/Layout/Template.Master" AutoEventWireup="true" CodeBehind="Cadastro.aspx.cs" Inherits="Site.Pages.Cadastro" %>

<asp:Content ID="Content1" ContentPlaceHolderID="Conteudo" runat="server">

<p>

Formulário de cadastro de clientes:

</p>

Nome do Cliente: <br />

<asp:TextBox ID="txtNome" runat="server" />

<br /><br />

Email: <br />

<asp:TextBox ID="txtEmail" runat="server" />

<br /><br />

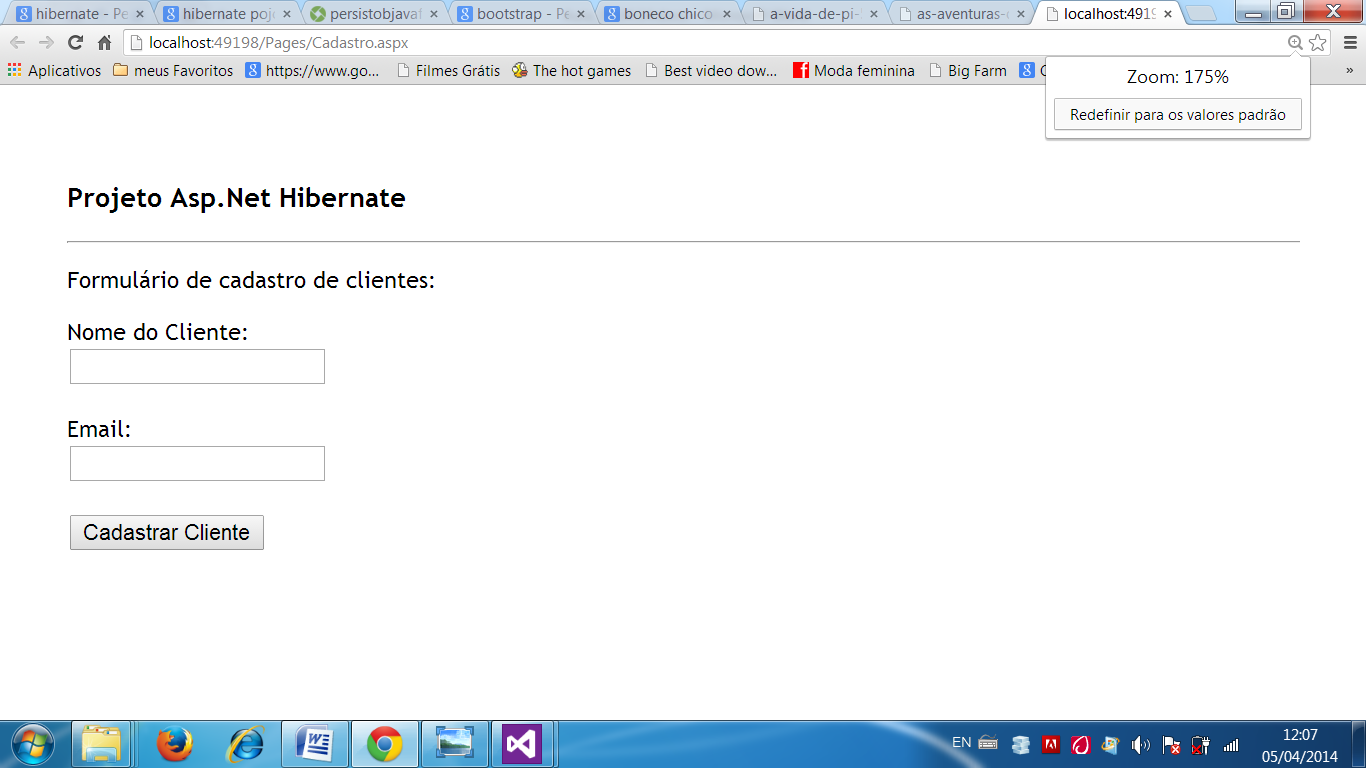
<asp:Button ID="btnCadastro" runat="server" Text="Cadastrar Cliente" />

<p>

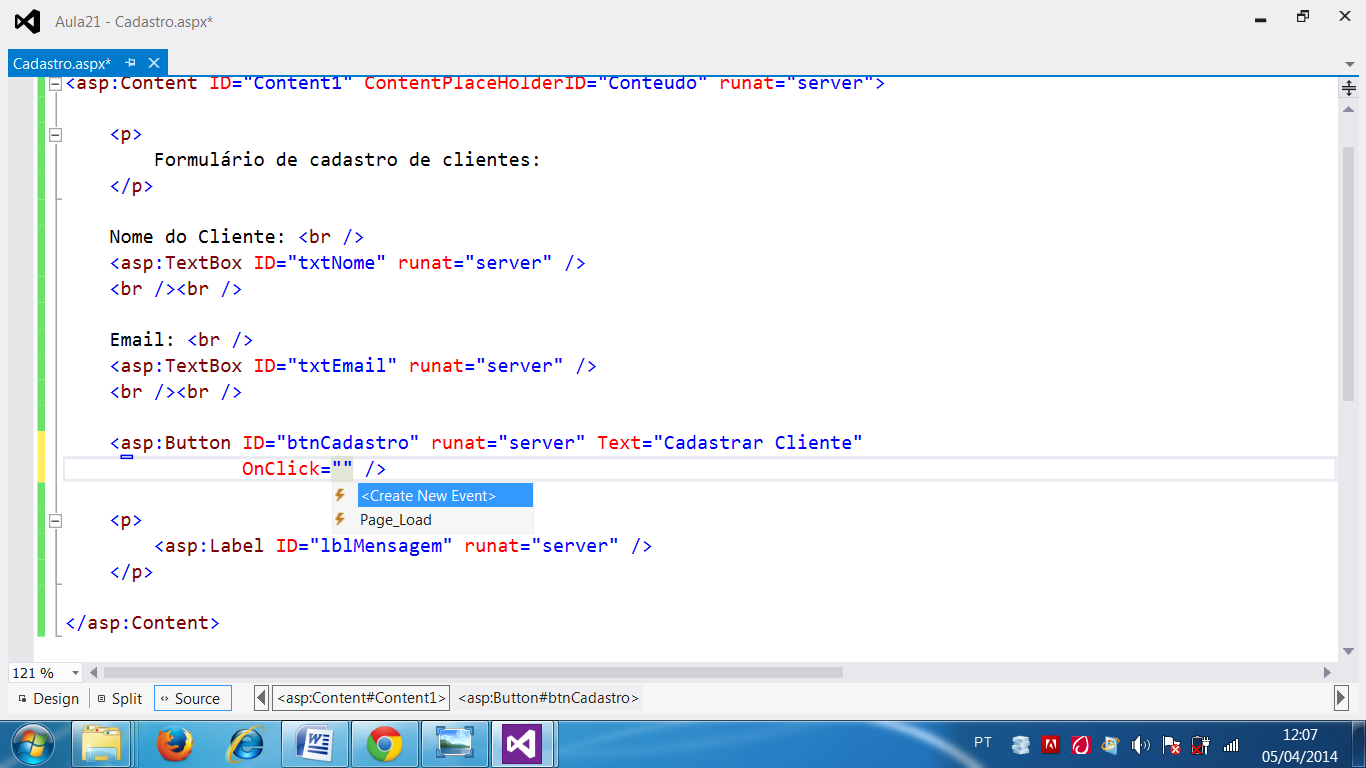
<asp:Label ID="lblMensagem" runat="server" />

</p>

</asp:Content>



Criando o evento...



<%@ Page Title="" Language="C#" MasterPageFile="~/Layout/Template.Master" AutoEventWireup="true" CodeBehind="Cadastro.aspx.cs" Inherits="Site.Pages.Cadastro" %>

<asp:Content ID="Content1" ContentPlaceHolderID="Conteudo" runat="server">

<p>

Formulário de cadastro de clientes:

</p>

Nome do Cliente: <br />

<asp:TextBox ID="txtNome" runat="server" />

<br /><br />

Email: <br />

<asp:TextBox ID="txtEmail" runat="server" />

<br /><br />

**<asp:Button ID="btnCadastro" runat="server" Text="Cadastrar Cliente"**

**OnClick="btnCadastro\_Click" />**

<p>

<asp:Label ID="lblMensagem" runat="server" />

</p>

</asp:Content>

CodeBehind...

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Site.Pages

{

public partial class Cadastro : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

}

**protected void btnCadastro\_Click(object sender, EventArgs e)**

{

}

}

}

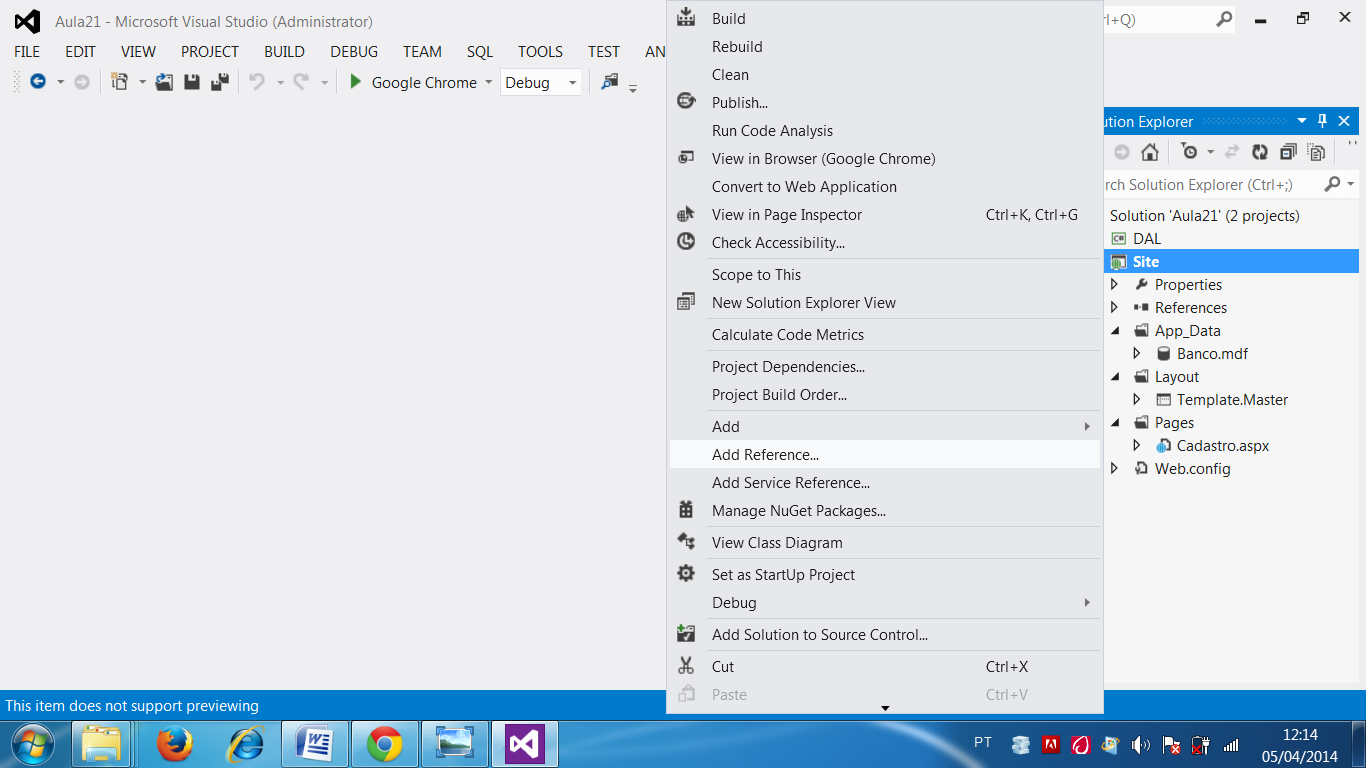
Log4net

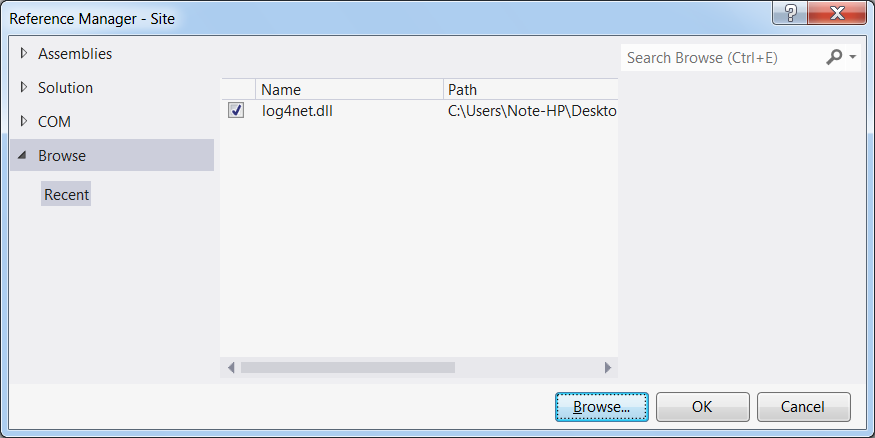
Ferramenta do NHibernate para geração de **logs** em aplicações .NET

Logs são arquivos texto gerados pelo Hibernate que contém toda a descrição do que o framework está executando na base de dados.

Este tipo de arquivo de log é útil quando precisamos verificar as queries por exemplo que o NHibernate executa na base bem como as transações, etc...

* Adicionando referência para o log4net





Configurar o log4net

\Web.config.xml

<?xml version="1.0"?>

<configuration>

**<!-- Log4Net Início -->**

**<configSections>**

**<section name="log4net"**

**type="log4net.Config.Log4NetConfigurationSectionHandler,**

**Log4net"/>**

**</configSections>**

**<log4net>**

**<root>**

**<level value="DEBUG" />**

**<appender-ref ref="LogFileAppender" />**

**</root>**

**<appender name="LogFileAppender"**

**type="log4net.Appender.RollingFileAppender" >**

**<param name="File" value="log\Hibernate.log" />**

**<param name="AppendToFile" value="true" />**

**<rollingStyle value="Size" />**

**<maxSizeRollBackups value="10" />**

**<maximumFileSize value="10MB" />**

**<staticLogFileName value="true" />**

**<layout type="log4net.Layout.PatternLayout">**

**<param name="ConversionPattern" value="[%d][%-5p] %c - %m%n" />**

**</layout>**

**</appender>**

**</log4net>**

**<!-- Log4Net Fim -->**

<system.web>

<compilation debug="true" targetFramework="4.5" />

<httpRuntime targetFramework="4.5" />

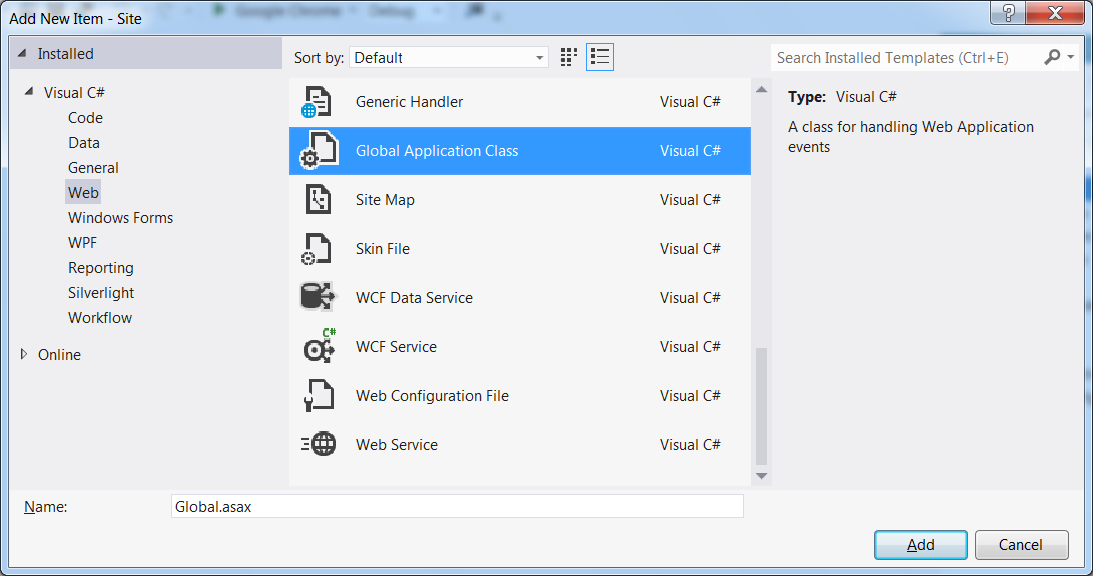
</system.web>

</configuration>

Classe para eventos globais

Eventos que são executados independente da página do sistema, por exemplo, que está sendo acessada. São eventos como “Iniciar o Projeto”, “Encerrar o Projeto”, “Erro no Projeto”, etc...

Global.asax



using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Security;

using System.Web.SessionState;

using log4net;

namespace Site

{

public class Global : System.Web.HttpApplication

{

//Evento executado toda vez que o projeto é iniciado

protected void Application\_Start(object sender, EventArgs e)

{

//Ativar o log

**log4net.Config.XmlConfigurator.Configure(); //inicia o log4net**

}

protected void Session\_Start(object sender, EventArgs e)

{

}

protected void Application\_BeginRequest(object sender, EventArgs e)

{

}

protected void Application\_AuthenticateRequest(object sender,

EventArgs e)

{

}

protected void Application\_Error(object sender, EventArgs e)

{

}

protected void Session\_End(object sender, EventArgs e)

{

}

protected void Application\_End(object sender, EventArgs e)

{

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using DAL.Model;

using DAL.Persistence;

namespace Site.Pages

{

public partial class Cadastro : System.Web.UI.Page

{

protected void Page\_Load(object sender, EventArgs e)

{

if( ! IsPostBack ) //página executando pela 1º vez

{

CarregarDados();

}

}

protected void btnCadastro\_Click(object sender, EventArgs e)

{

try

{

Cliente c = new Cliente();

c.Nome = txtNome.Text; //campo nome

c.Email = txtEmail.Text; //campo email

c.DataCadastro = DateTime.Now; //data do sistema

ClienteDal d = new ClienteDal();

d.Salvar(c);

txtNome.Text = string.Empty; //vazio

txtEmail.Text = string.Empty; //vazio

lblMensagem.Text = "Cliente " + c.Nome

+ ", cadastrado com sucesso.";

CarregarDados();

}

catch(Exception ex)

{

lblMensagem.Text = ex.Message;

}

}

private void CarregarDados()

{

try

{

ClienteDal d = new ClienteDal();

gridClientes.DataSource = d.ListarTodos(); //preenche o grid

gridClientes.DataBind(); //exibir

}

catch(Exception e)

{

lblMensagem.Text = e.Message; //mensagem de erro

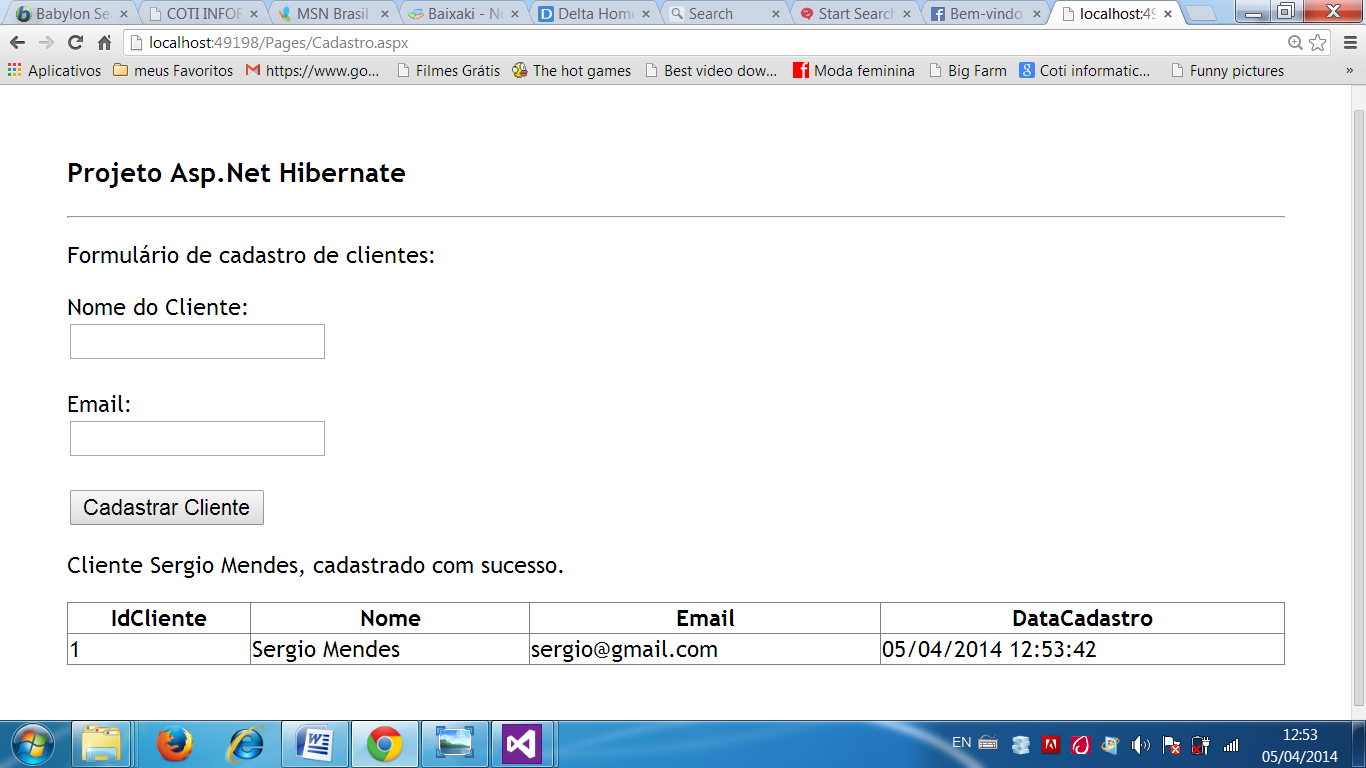
}

}

}

}

Executando...



Log gerado...

