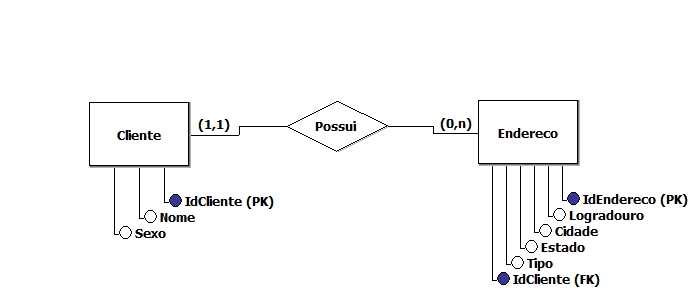
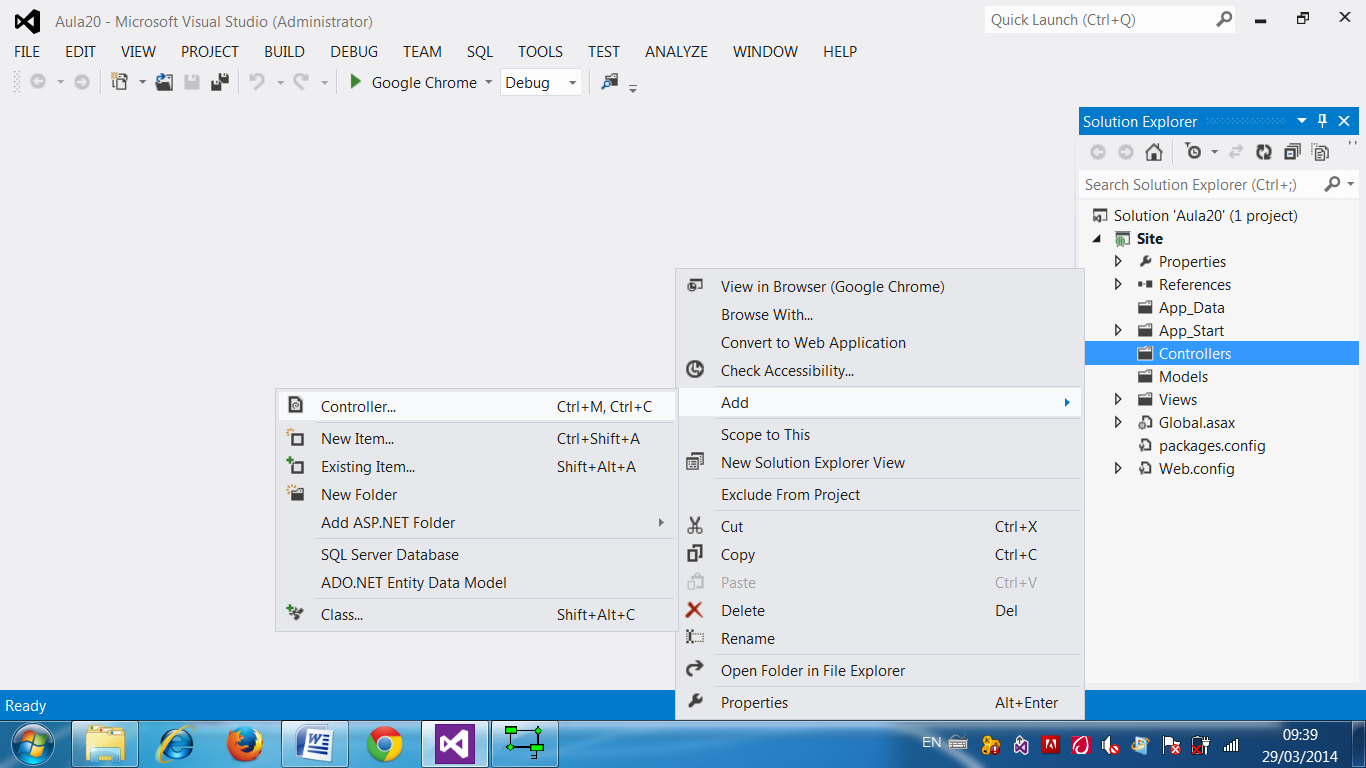
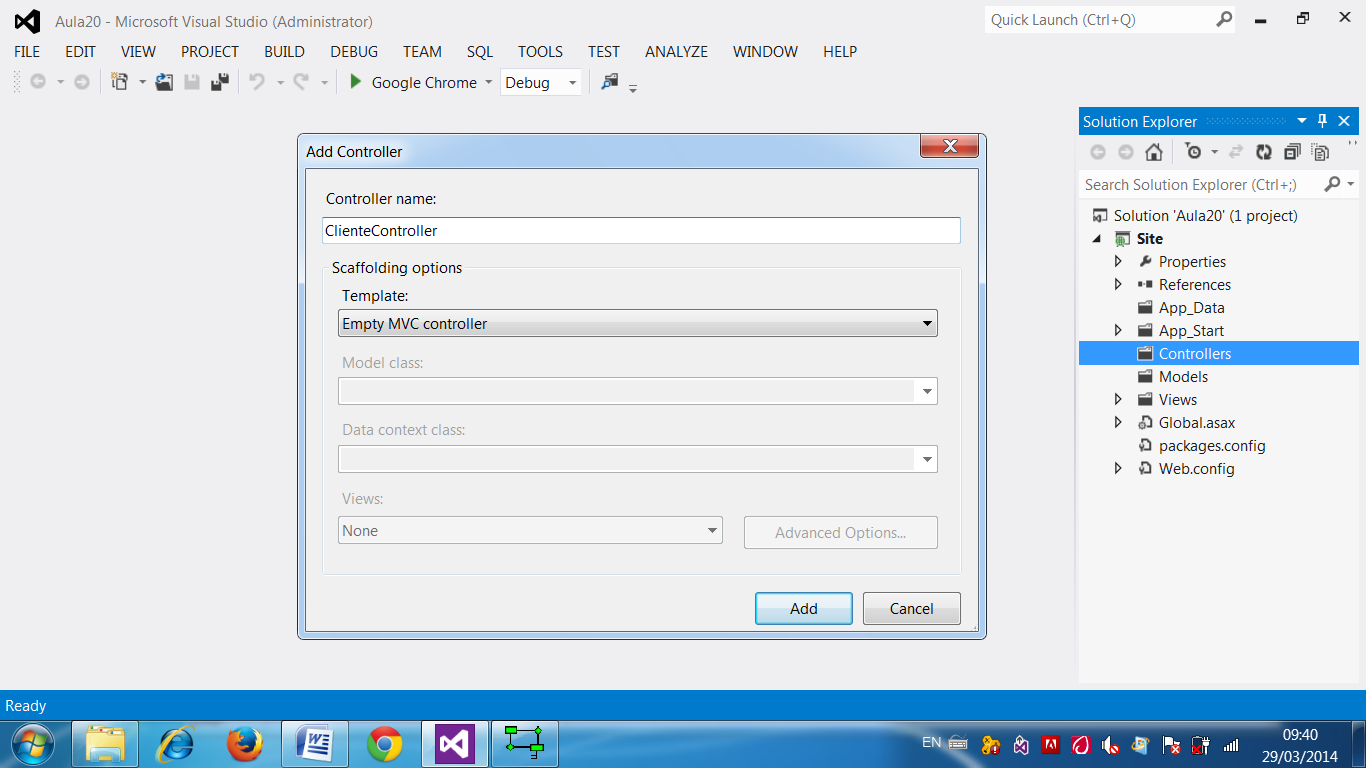


* Modelagem



Passo 1: Criação do Controller





using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace Site.Controllers

{

public class ClienteController : Controller

{

//Método gerador de rota => Abrir página => /Cliente/Cadastro

public ActionResult Cadastro() //Nome da página

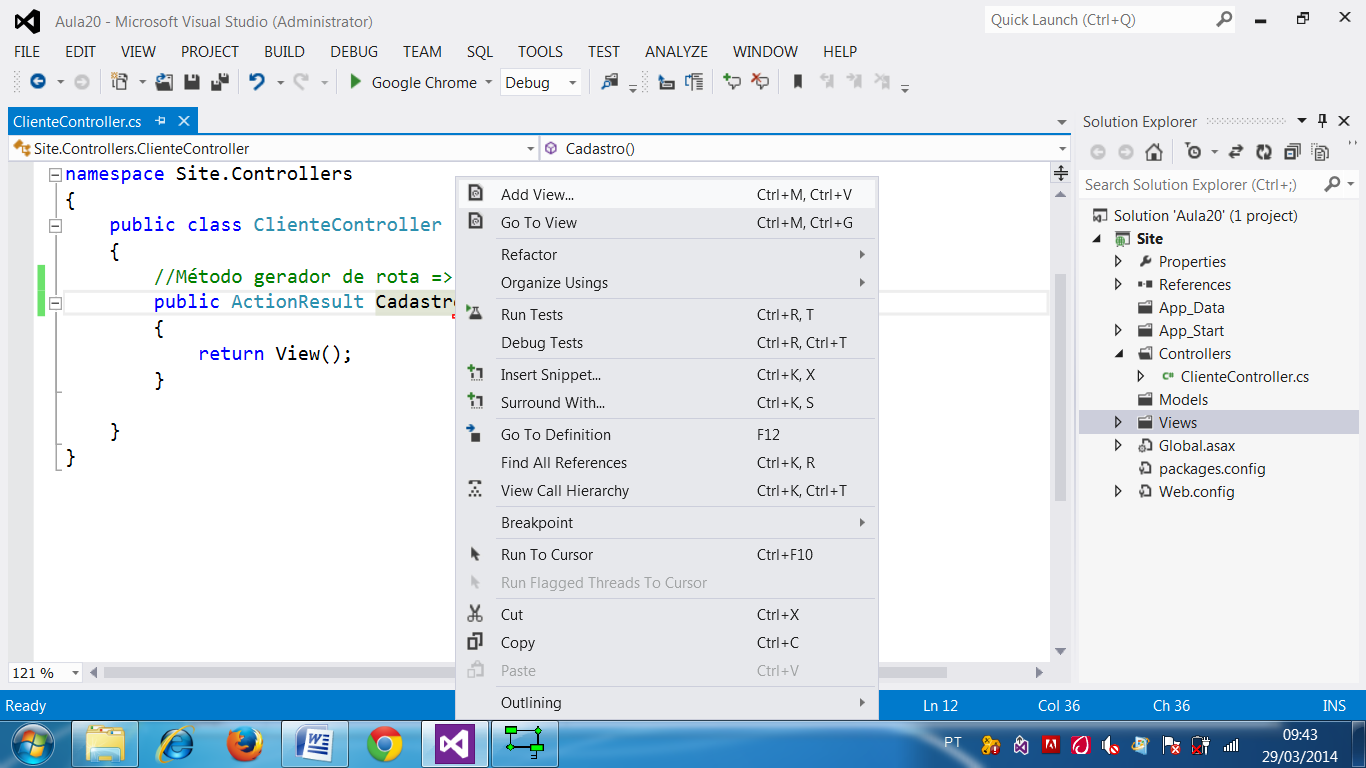
{

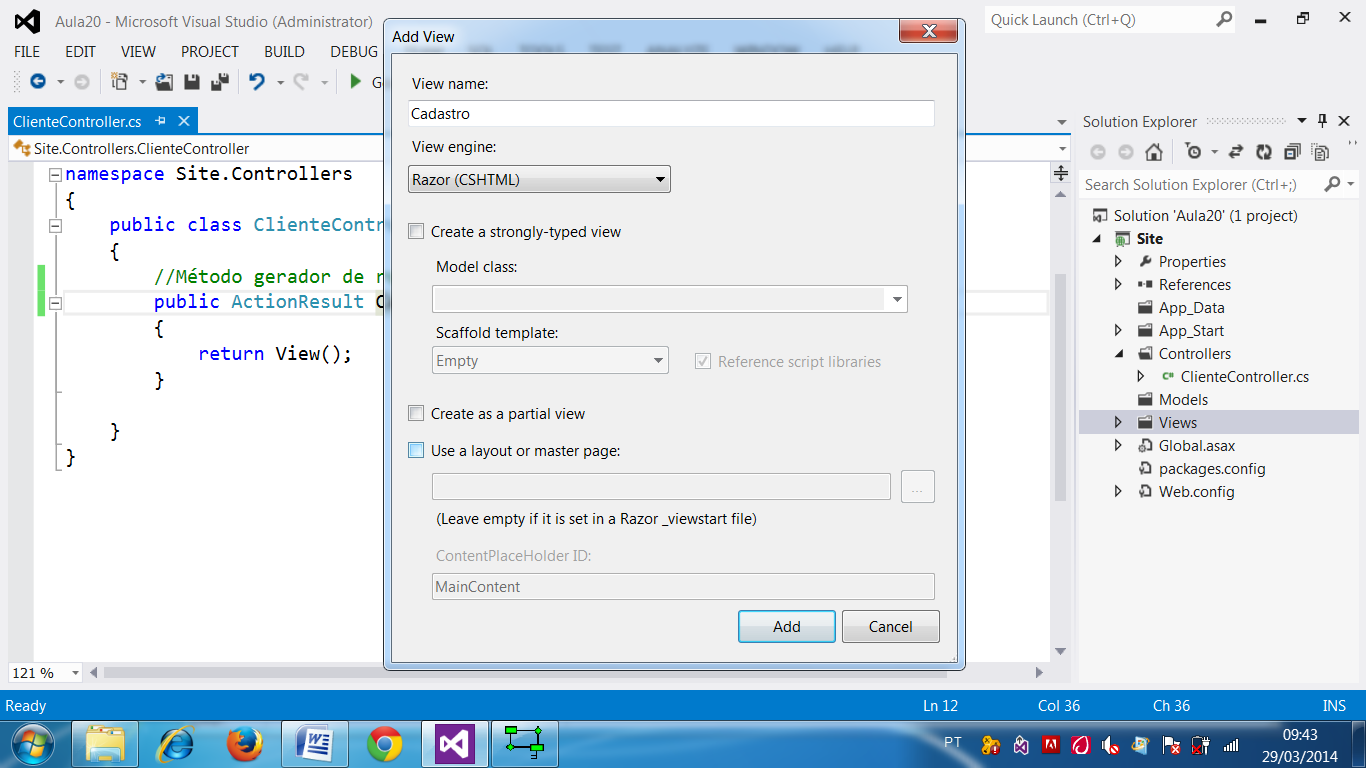
return View();

}

}

}





<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Cadastro</title>

<style type="text/css">

body { font-family: Verdana; font-size: 10pt; padding: 40px; }

</style>

</head>

<body>

<div>

<h3>Cadastro de Clientes</h3>

<p>

Para cadastrar um novo cliente, preencha os campos abaixo:

</p>

</div>

</body>

</html>

Página inicial

/App\_Start/RouteConfig.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Routing;

namespace Site

{

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Cliente", action = "Cadastro",

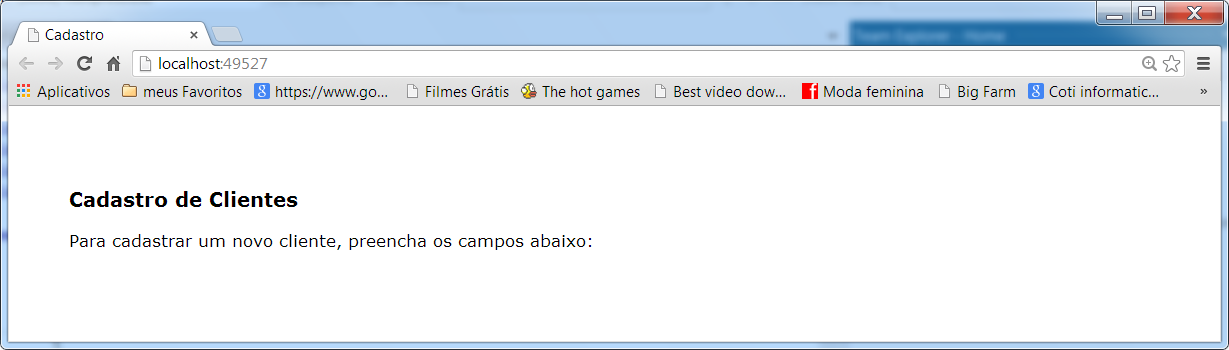
id = UrlParameter.Optional }

);

}

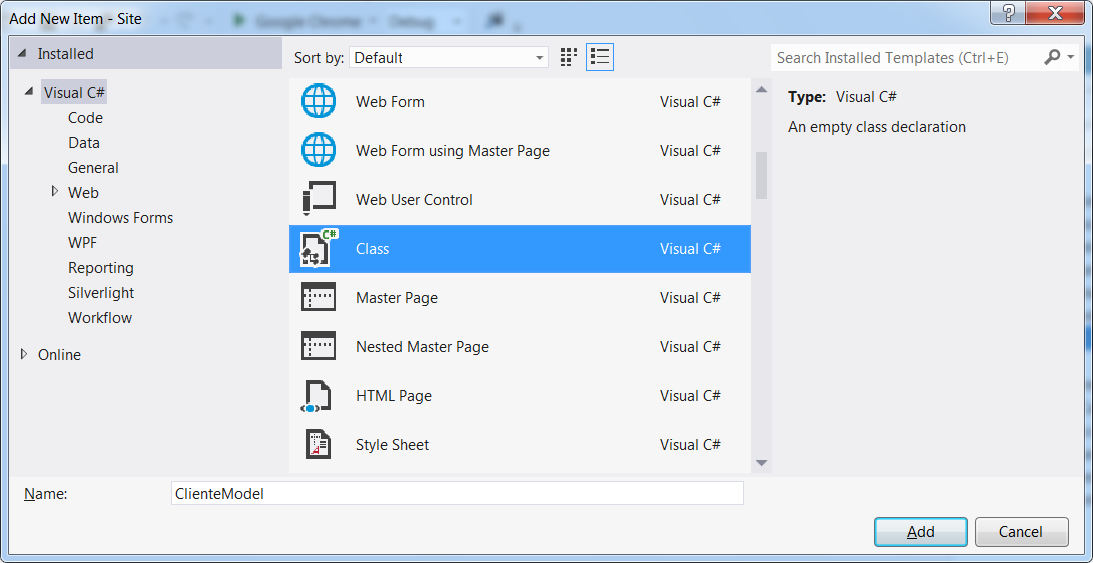
}

}



Camada de Modelo

Responsável pelos dados de entrada / saída



using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.ComponentModel.DataAnnotations; //mapeamento

namespace Site.Models

{

public class ClienteModel

{

[Display(Name = "Nome do Cliente:")]

public string Nome { get; set; }

[Display(Name = "Sexo:")]

public char Sexo { get; set; }

[Display(Name = "Logradouro:")]

public string Logradouro { get; set; }

[Display(Name = "Cidade:")]

public string Cidade { get; set; }

[Display(Name = "Estado:")]

public string Estado { get; set; }

[Display(Name = "Tipo do Endereço:")]

public string Tipo { get; set; }

}

}

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Cadastro</title>

<style type="text/css">

body { font-family: Verdana; font-size: 10pt; padding: 40px; }

</style>

</head>

<body>

<div>

<h3>Cadastro de Clientes</h3>

<p>

Para cadastrar um novo cliente, preencha os campos abaixo:

</p>

<!-- Referenciar a Classe de Modelo -->

@model Site.Models.ClienteModel

<!-- Criando o formulário de cadastro -->

@using(Html.BeginForm())

{

@Html.LabelFor(c => c.Nome) <br />

@Html.TextBoxFor(c => c.Nome)

<br /><br />

@Html.LabelFor(c => c.Sexo) <br />

@Html.RadioButtonFor(c => c.Sexo, "M") <span>Masculino</span>

@Html.RadioButtonFor(c => c.Sexo, "F") <span>Feminino</span>

<br /><br />

@Html.LabelFor(c => c.Logradouro) <br />

@Html.TextBoxFor(c => c.Logradouro)

<br /><br />

@Html.LabelFor(c => c.Cidade) <br />

@Html.TextBoxFor(c => c.Cidade)

<br /><br />

@Html.LabelFor(c => c.Estado) <br />

@Html.TextBoxFor(c => c.Estado)

<br /><br />

@Html.LabelFor(c => c.Tipo) <br />

@Html.RadioButtonFor(c => c.Tipo, "Residencial")

<span>Endereço Residencial</span> <br />

@Html.RadioButtonFor(c => c.Tipo, "Comercial")

<span>Endereço Comercial</span> <br />

<br />

<input type="submit" value="Cadastrar Cliente" />

<p>

<!-- Exibir mensagens -->

@ViewBag.Mensagem

</p>

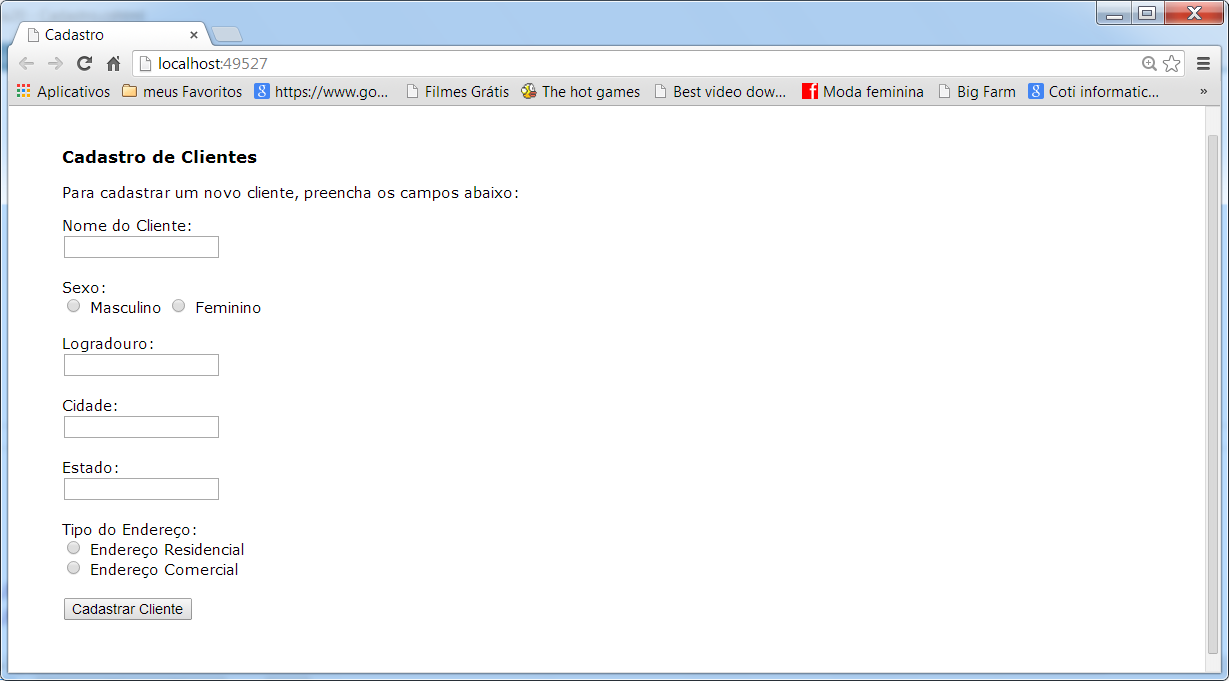
}

</div>

</body>

</html>

Executando...



<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>Cadastro</title>

<style type="text/css">

body { font-family: Verdana; font-size: 10pt; padding: 40px; }

</style>

</head>

<body>

<div>

<h3>Cadastro de Clientes</h3>

<p>

Para cadastrar um novo cliente, preencha os campos abaixo:

</p>

<!-- Referenciar a Classe de Modelo -->

**@model Site.Models.ClienteModel**

<!-- Criando o formulário de cadastro -->

**@using(Html.BeginForm("CadastrarCliente",**

**"Cliente", FormMethod.Post))**

{

@Html.LabelFor(c => c.Nome) <br />

@Html.TextBoxFor(c => c.Nome)

<br /><br />

@Html.LabelFor(c => c.Sexo) <br />

@Html.RadioButtonFor(c => c.Sexo, "M") <span>Masculino</span>

@Html.RadioButtonFor(c => c.Sexo, "F") <span>Feminino</span>

<br /><br />

@Html.LabelFor(c => c.Logradouro) <br />

@Html.TextBoxFor(c => c.Logradouro)

<br /><br />

@Html.LabelFor(c => c.Cidade) <br />

@Html.TextBoxFor(c => c.Cidade)

<br /><br />

@Html.LabelFor(c => c.Estado) <br />

@Html.TextBoxFor(c => c.Estado)

<br /><br />

@Html.LabelFor(c => c.Tipo) <br />

@Html.RadioButtonFor(c => c.Tipo, "Residencial")

<span>Endereço Residencial</span> <br />

@Html.RadioButtonFor(c => c.Tipo, "Comercial")

<span>Endereço Comercial</span> <br />

<br />

<input type="submit" value="Cadastrar Cliente" />

<p>

<!-- Exibir mensagens -->

@ViewBag.Mensagem

</p>

}

</div>

</body>

</html>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Site.Models; //camada de modelo

namespace Site.Controllers

{

public class ClienteController : Controller

{

//Método gerador de rota => Abrir página => /Cliente/Cadastro

public ActionResult Cadastro() //Nome da página

{

return View();

}

//Método para resgatar o envio de dados realizado pelo formulário

[HttpPost] //indicando que o método é chamado por

//formulário com requisição POST

public ActionResult CadastrarCliente(ClienteModel Model)

//classe de modelo

{

ViewBag.Mensagem = "Testando..."; //imprimindo mensagem

//redirecionamento

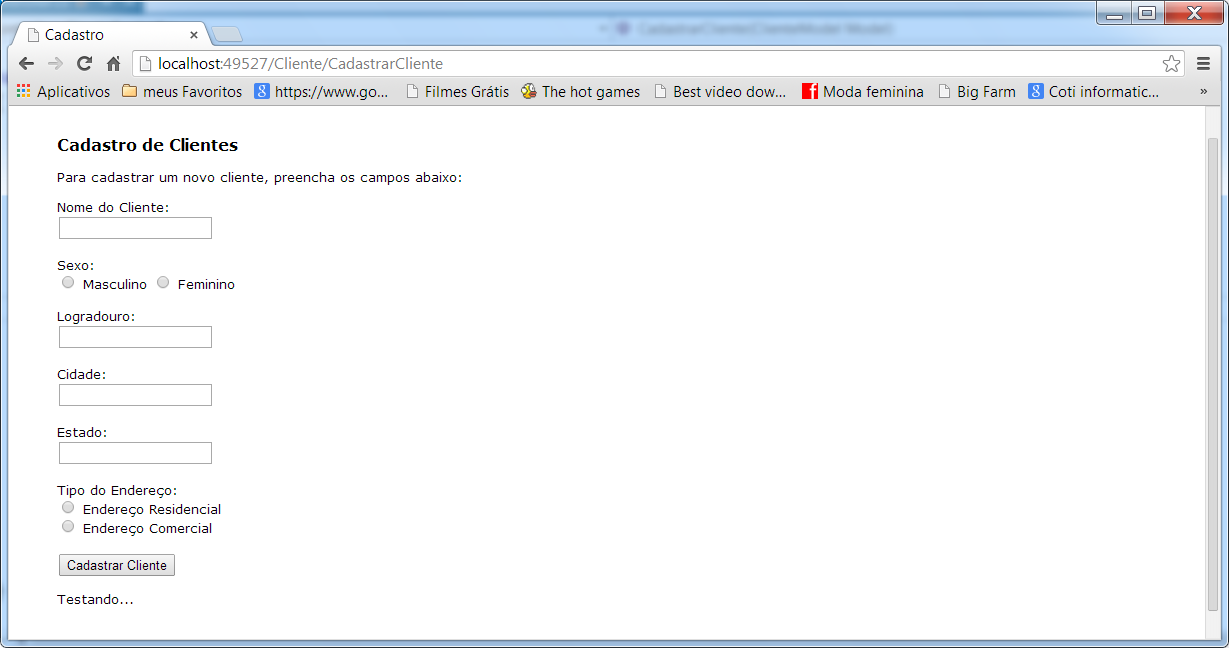
return View("Cadastro"); //nome da página

}

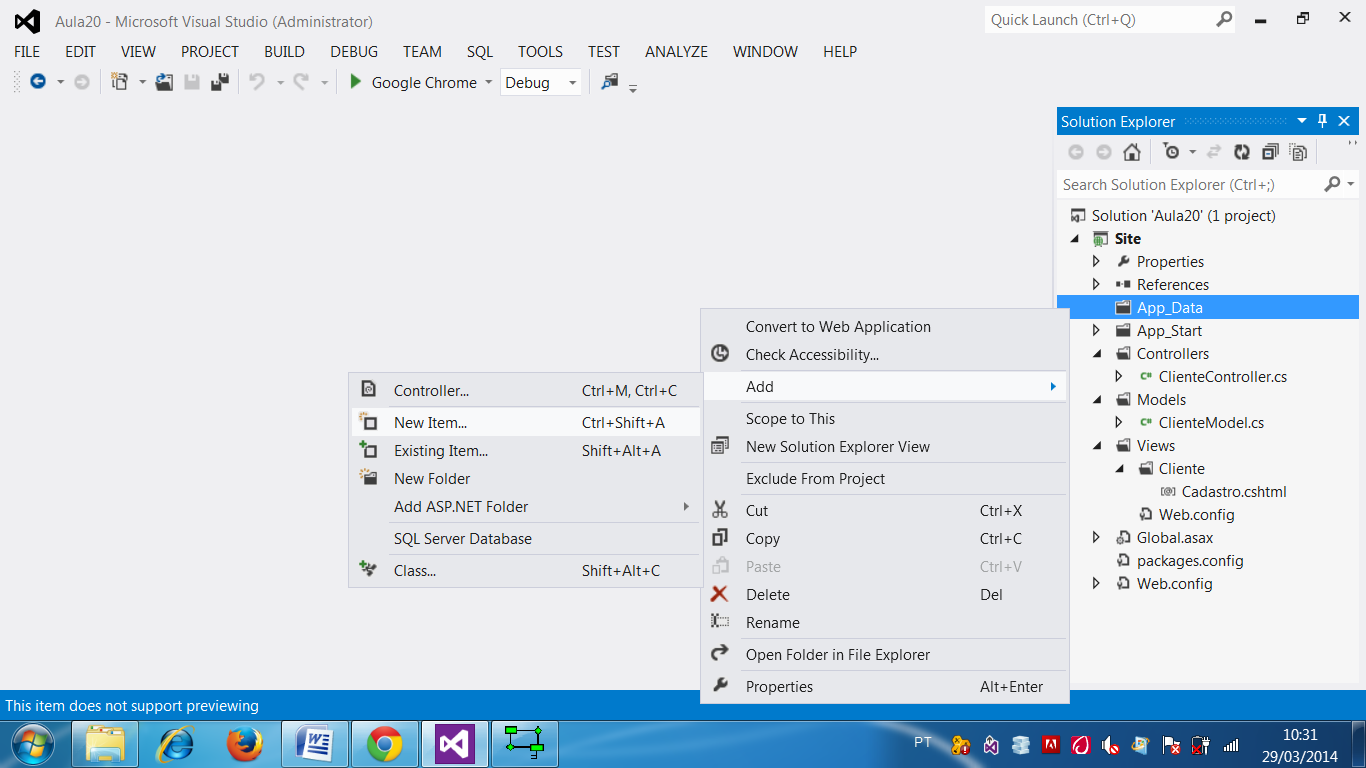
}

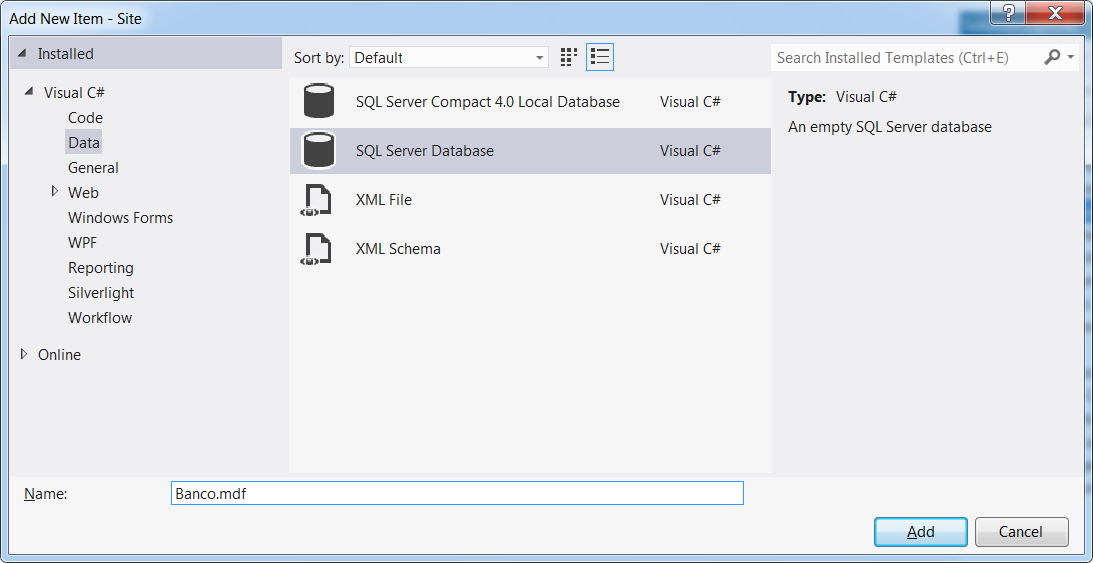
}

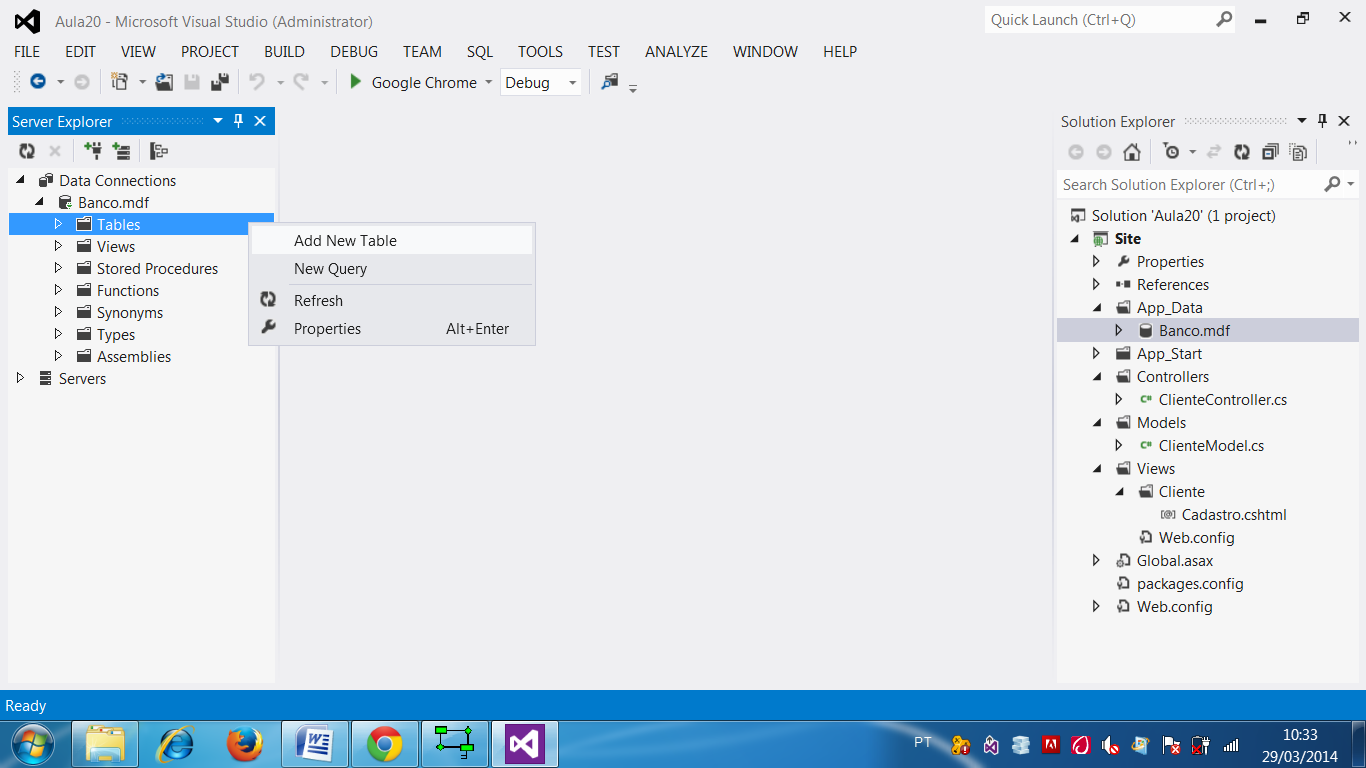
Executando...



Criando a base de dados...







Script do banco de dados...

go

create table Cliente(

IdCliente integer identity,

Nome nvarchar(50) not null,

Sexo char(1) not null,

DataCadastro datetime not null,

primary key(IdCliente))

go

create table Endereco(

IdEndereco integer identity,

Logradouro nvarchar(50) not null,

Cidade nvarchar(50) not null,

Estado nvarchar(50) not null,

Tipo nvarchar(50) not null,

IdCliente integer not null,

primary key(IdEndereco))

go

alter table Endereco add constraint FkEnderecoCliente

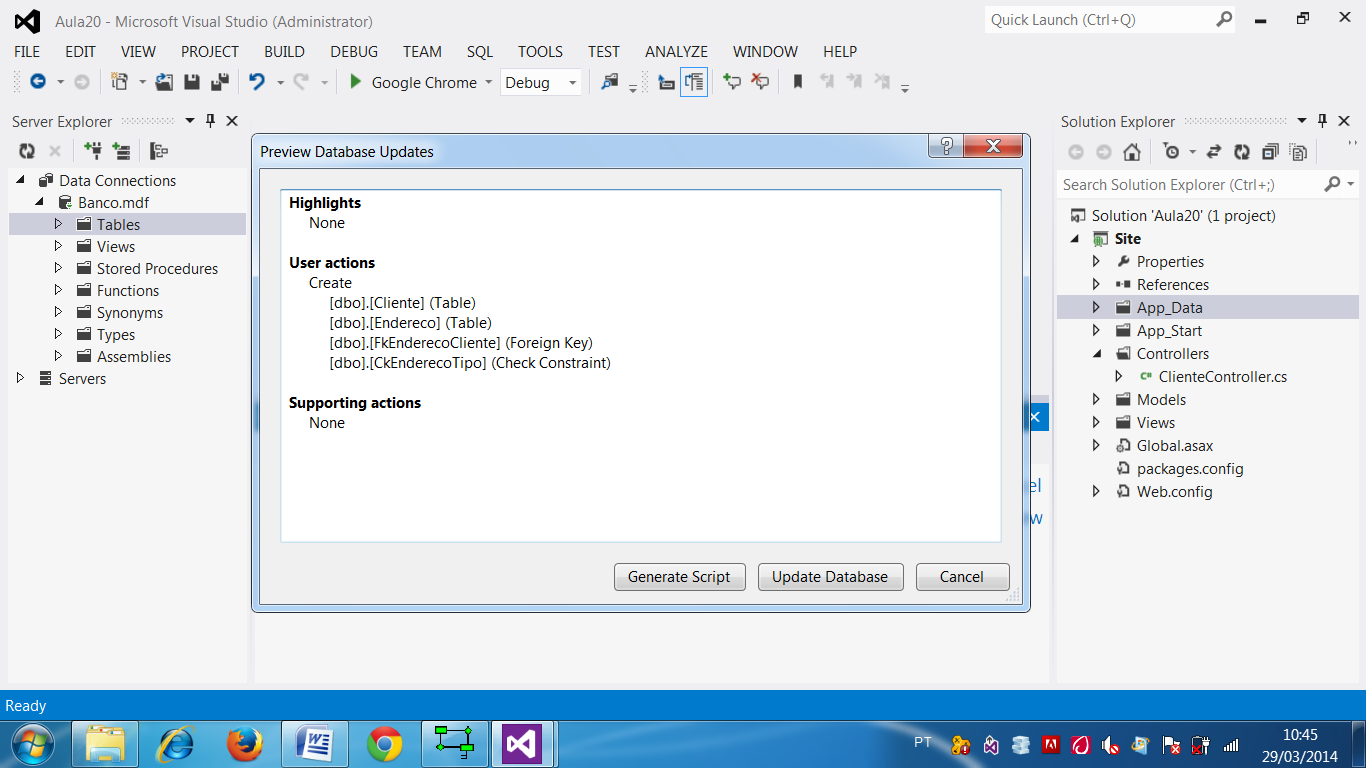
foreign key(IdCliente) references Cliente(IdCliente)

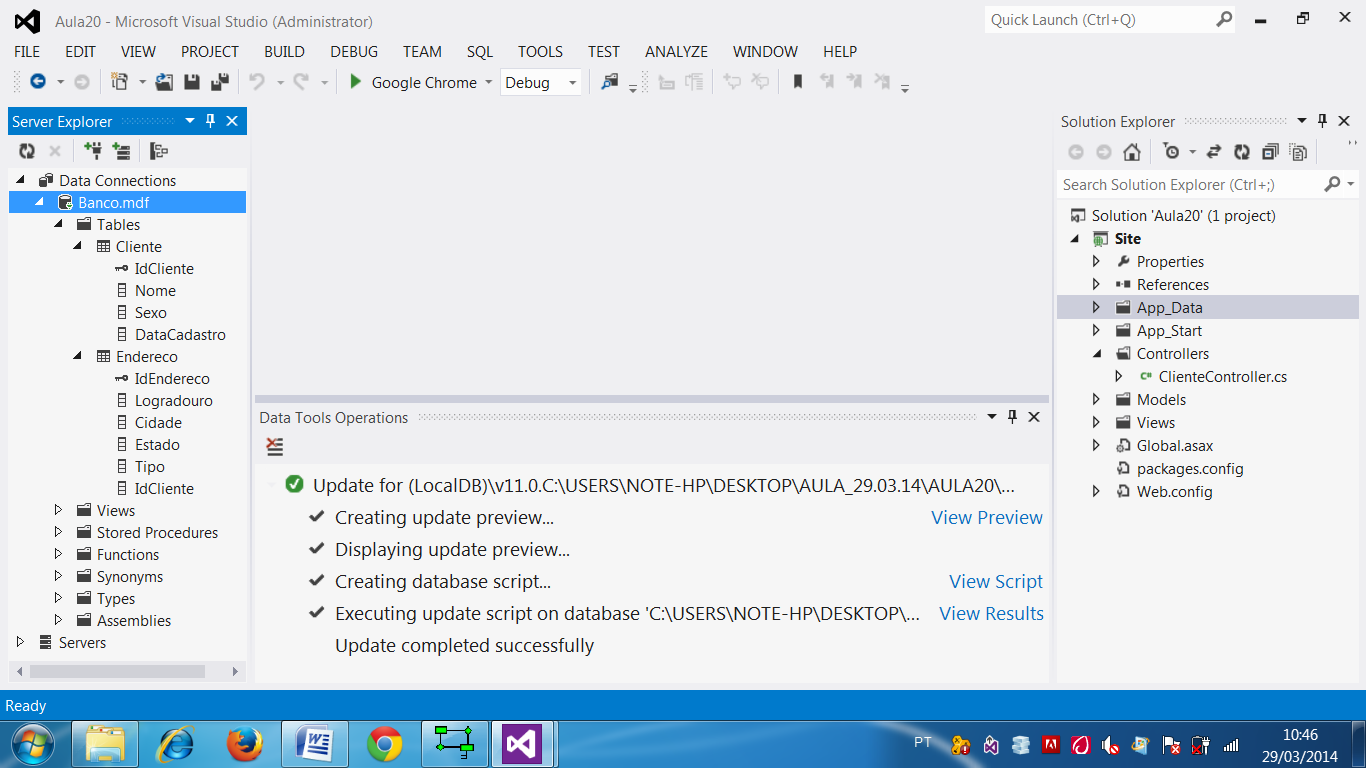
go

alter table Endereco add constraint CkEnderecoTipo

check(Tipo = 'Residencial' or Tipo = 'Comercial')

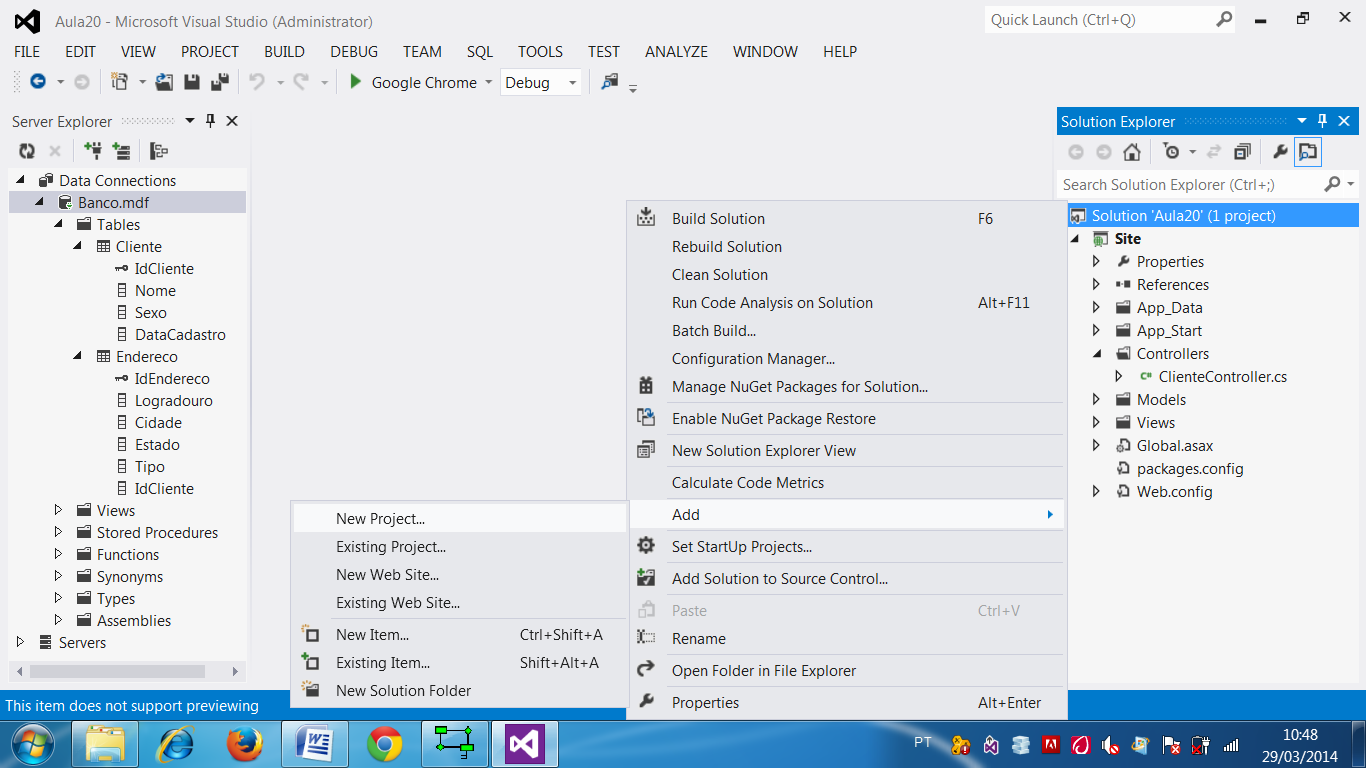
Executando...

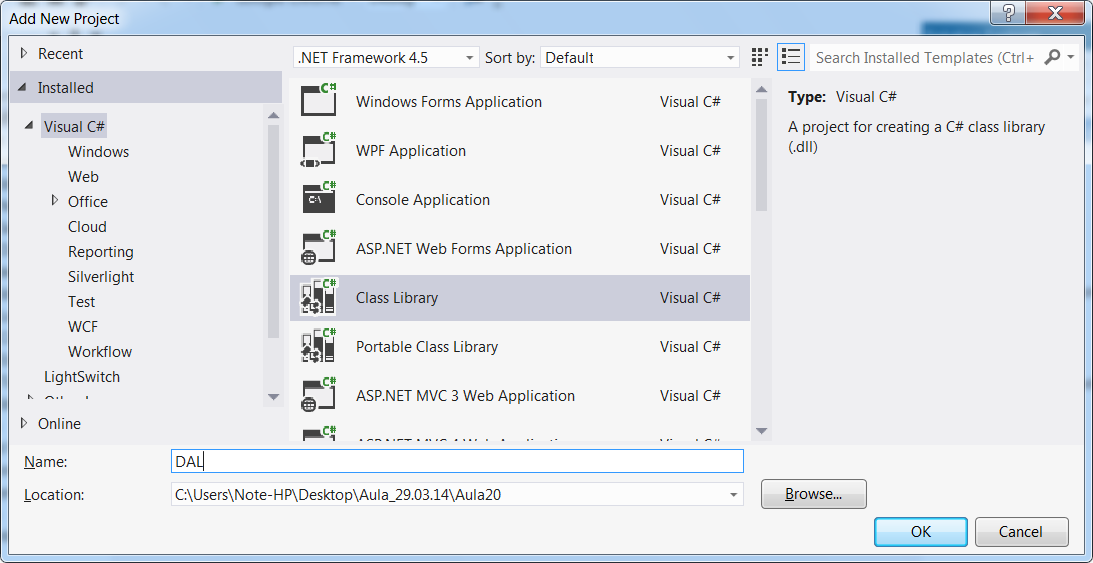




Data Access Layer (DAL)

Camada de acesso a dados



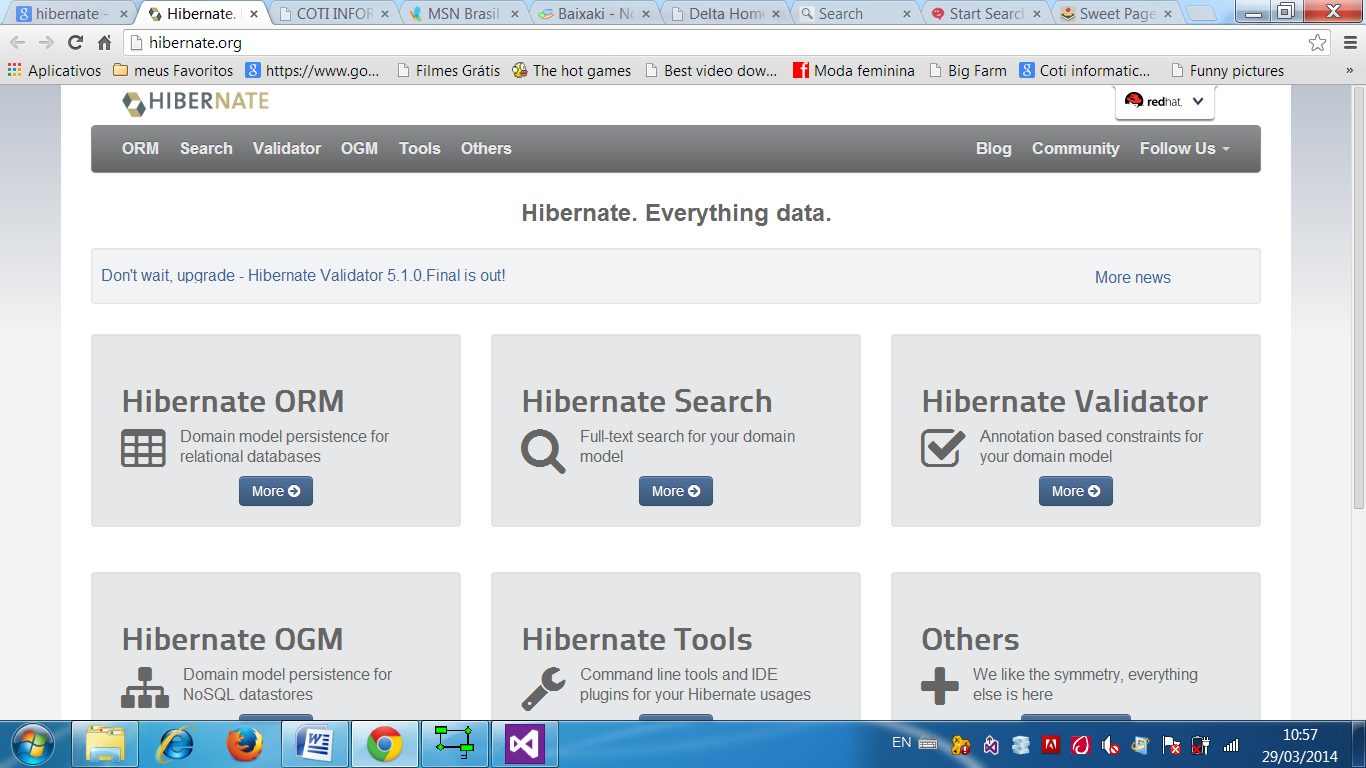


NHibernate

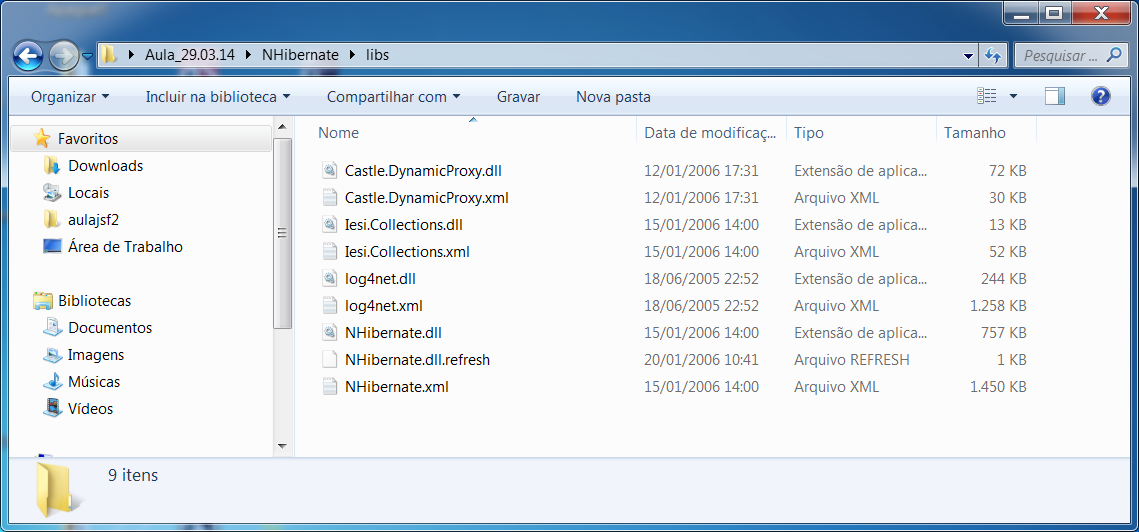
Framework criado pela comunidade Java (denominado de Hibernate) voltado para acesso, transações e consultas a bases de dados. Possui uma versão escrita para .Net chamado de NHibernate.

Possui características como o EntityFramework de mapeamento objeto relacional, conexão otimizada com o banco de dados e tendo como diferencial uma arquitetura que possibilita a conexão com diversos bancos de dados como SqlServer, Oracle, MySql, PostGre, FireBird, etc..

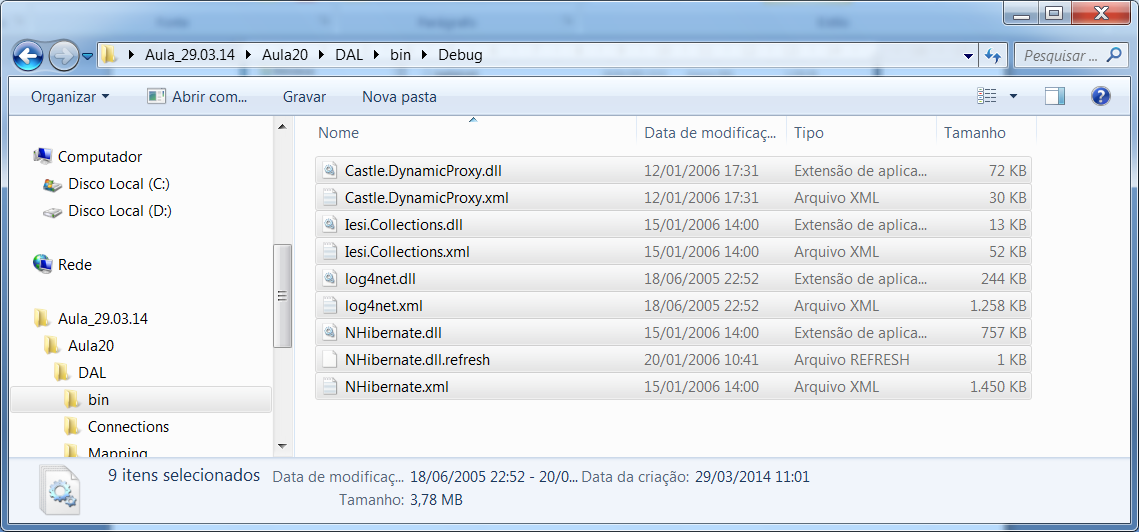
<http://hibernate.org/>



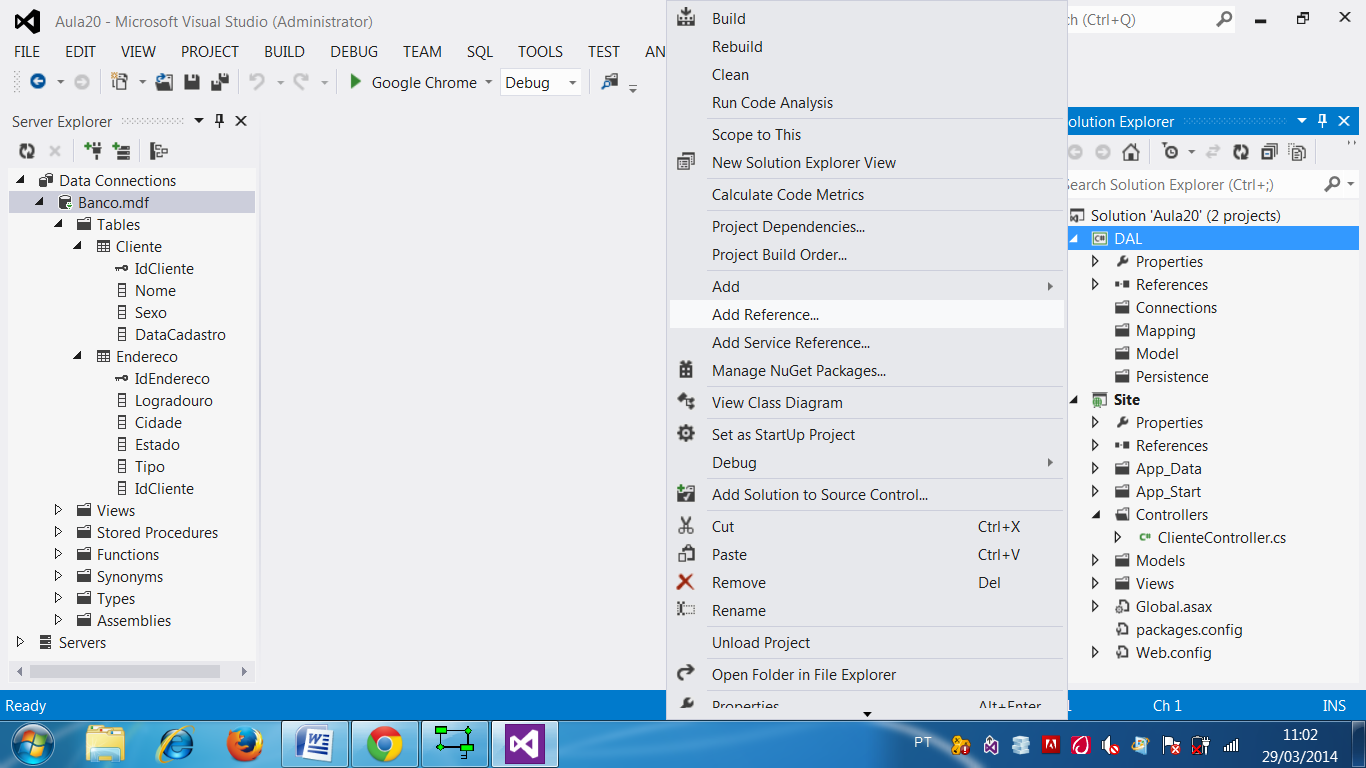
Incluir as bibliotecas do NHibernate no projeto...

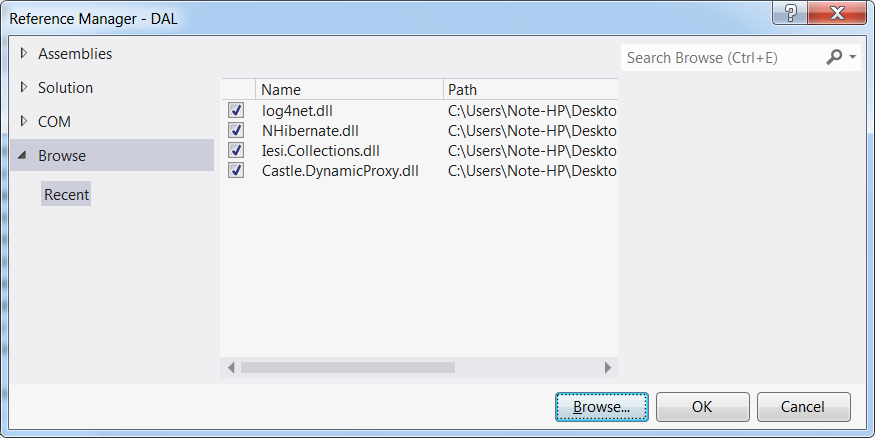


Colar em: Aula20\DAL\bin\Debug

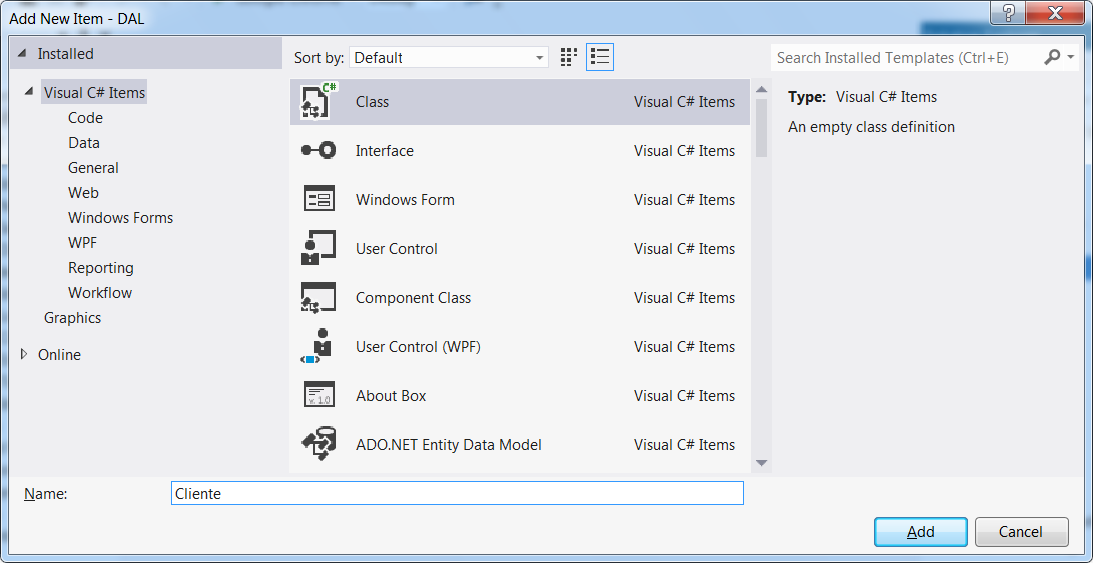


Adicionar referência...





Modelagem das classes de entidade...



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 char Sexo { get; set; }

public DateTime DataCadastro { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DAL.Model

{

public class Endereco

{

public int IdEndereco { get; set; }

public string Logradouro { get; set; }

public string Cidade { get; set; }

public string Estado { get; set; }

public string Tipo { get; set; }

/\*

\* No padrão do NHibernate, chave estrangeira não

\* é incluida como atributo na classe

\*/

public Cliente Cliente { get; set; } //Associação

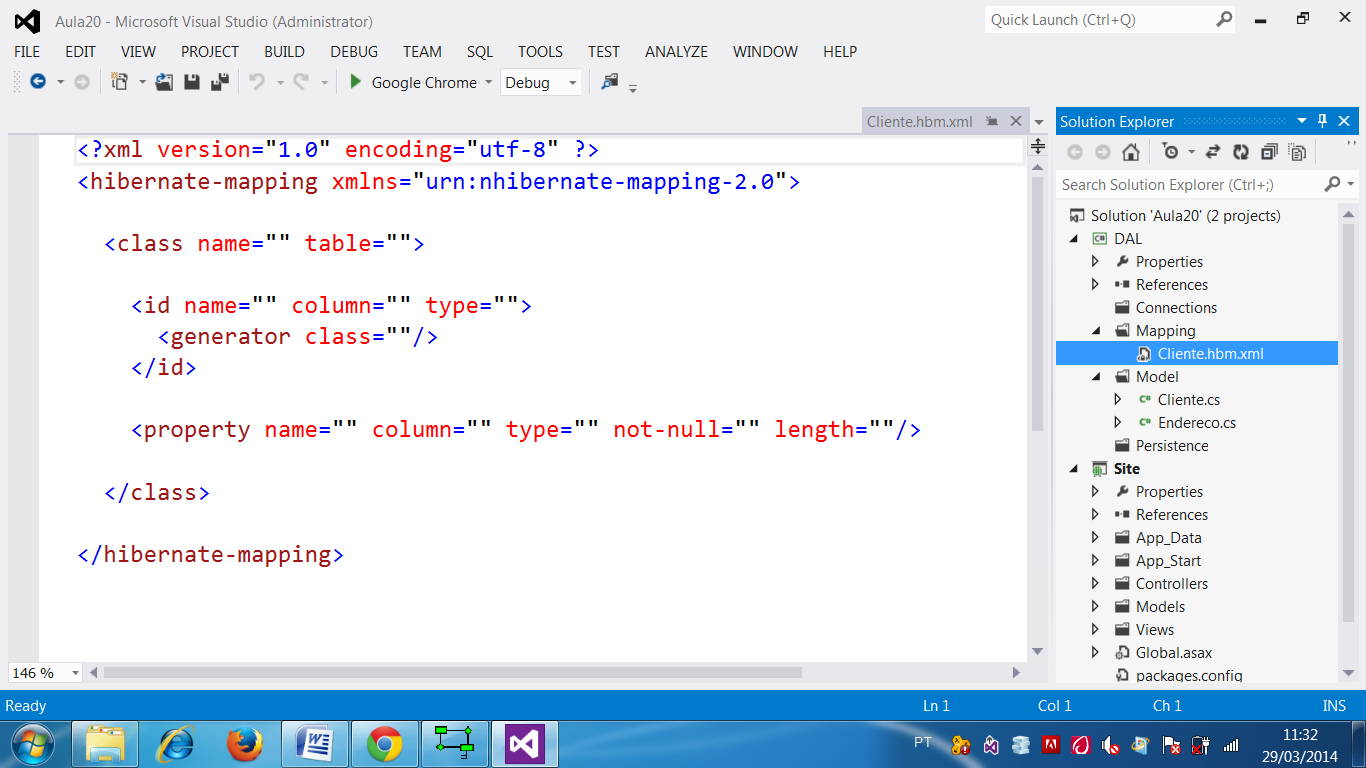
}

}

Mapeamento Objeto Relacional

No NHibernate, o mapeamento das classes será realizado por meio de arquivos XML denominados de **HBM** (**Hibernate Mapping Files**)

* Cliente.hbm.xml



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

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

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

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

<!-- Chave primária -->

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

<generator class="identity"/>

</id>

<!-- Mapeamento das demais colunas -->

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

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

<property name="Sexo" column="Sexo" type="Char"

not-null="true" length="1"/>

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

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

</class>

</hibernate-mapping>

* Endereco.hbm.xml

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

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

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

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

<generator class="identity"/>

</id>

<property name="Logradouro" column="Logradouro"

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

<property name="Cidade" column="Cidade"

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

<property name="Estado" column="Estado"

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

<property name="Tipo" column="Tipo"

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

<!-- Mapeamento do Relacionamento -->

<!-- one-to-one, one-to-many, many-to-one, many-to-many -->

<!-- many-to-one => Muitos Enderecos para 1 Cliente -->

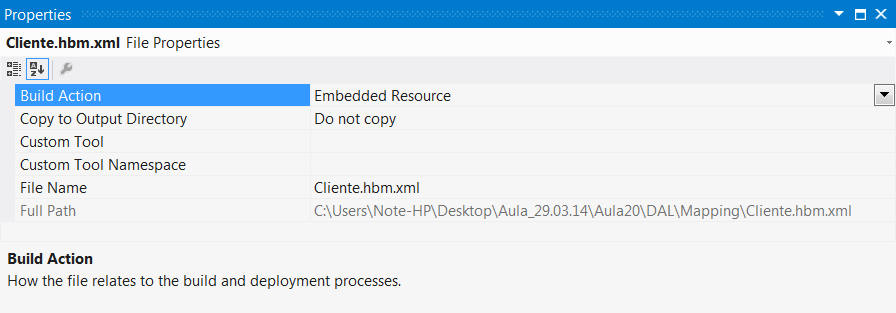
<many-to-one name="Cliente" column="IdCliente"/>

</class>

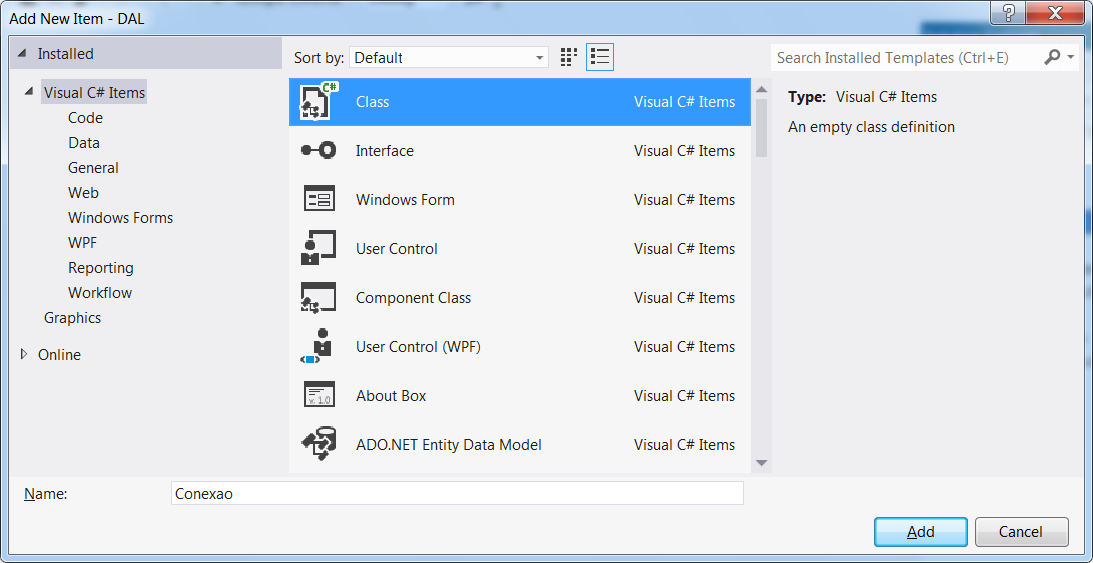
</hibernate-mapping>

Regra:

Para que o Hibernate possa durante a execução do projeto ler e interpretar estes arquivos **hbm.xml**, é necessário que tais arquivos sejam marcados como EmbeddedResource (Recurso incorporado)



Classe para conexão com o banco de dados pelo NHibernate



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

{

//Para que o Hibernate possa gerar conexões com uma base de dados é

//necessário que criemos um componente chamado ISessioNFactory (Fábrica

//de Conexões) -> gerador de conexões

public ISessionFactory Factory()

{

try

{

//Para conectar o hibernate em uma base de dados são necessários

//algumas configurações...

Configuration Cfg = new Configuration(); //Classe para

//configuração do NHibernate

Cfg.SetProperty(NHibernate.Cfg.Environment.ConnectionProvider,

"NHibernate.Connection.DriverConnectionProvider");

Cfg.SetProperty(NHibernate.Cfg.Environment.Dialect,

"NHibernate.Dialect.MsSql2000Dialect");

Cfg.SetProperty(NHibernate.Cfg.Environment.ConnectionDriver,

"NHibernate.Driver.SqlClientDriver");

Cfg.SetProperty(NHibernate.Cfg.Environment.ConnectionString,

@"Data Source=(LocalDB)\v11.0;AttachDbFilename=C:\Users\Note-HP\Desktop\Aula\_29.03.14\Aula20\Site\App\_Data\Banco.mdf;Integrated Security=True");

//Nome do Projeto

Cfg.AddAssembly("DAL");

//Gerar e retornar a fábrica de conexões

return Cfg.BuildSessionFactory();

}

catch(Exception e)

{

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

+ e.Message);

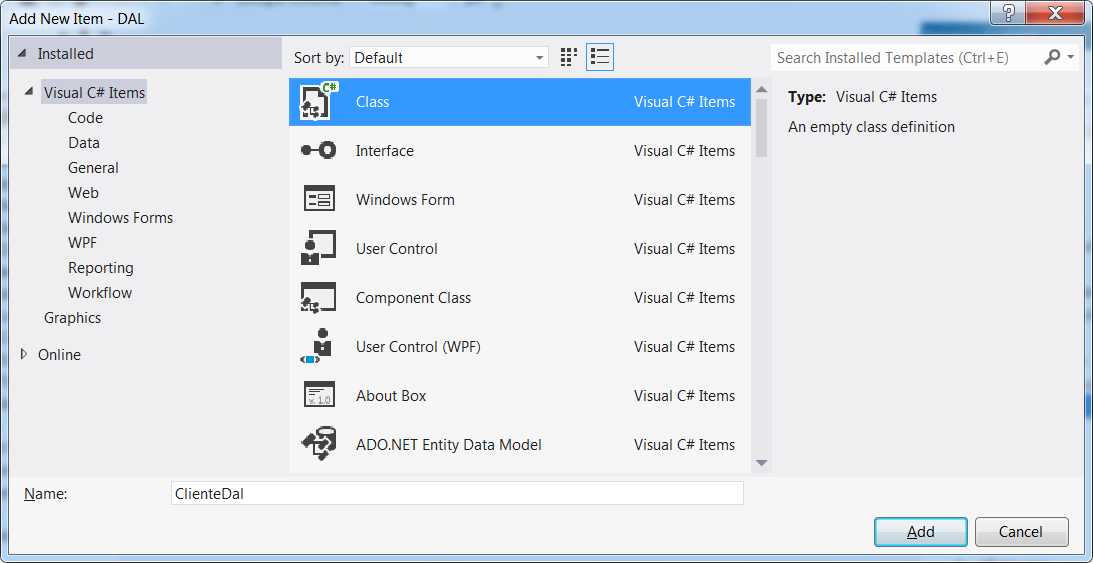
}

}

}

}

Classe de persistência...



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using NHibernate;

using DAL.Model;

using DAL.Connections;

namespace DAL.Persistence

{

public class ClienteDal

{

//Atributos (API do Nhibernate)

private ISession Session;

//Capturar a Conexão com o banco de dados

private ITransaction Transaction;

//Executar transações na base (commit, rollback)

private IQuery Query;

//Executar consultas (linguagem HQL -> Hibernate Query Language)

//Método para gravar Cliente e Endereco

public void Salvar(Cliente c, Endereco e)

{

try

{

Conexao Con = new Conexao(); //Instanciando a Classe de Conexão

Session = Con.Factory().OpenSession();

//abrindo conexão através da fábrica

Transaction = Session.BeginTransaction();

//iniciando uma transação

Session.Save(c); //insert

Session.Save(e); //insert

Transaction.Commit(); //executar

}

catch(Exception ex)

{

Transaction.Rollback(); //desfazer

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

}

finally

{

Session.Close(); //fechar a conexão

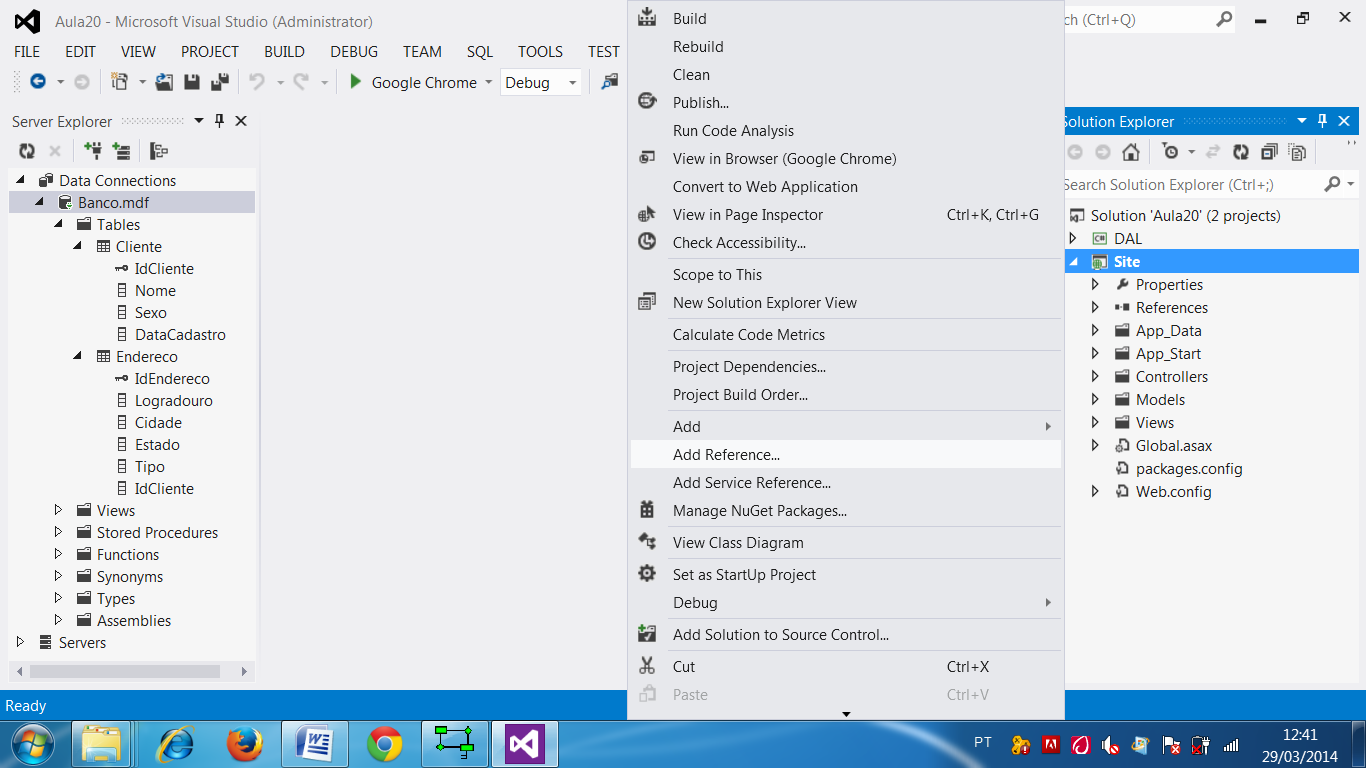
}

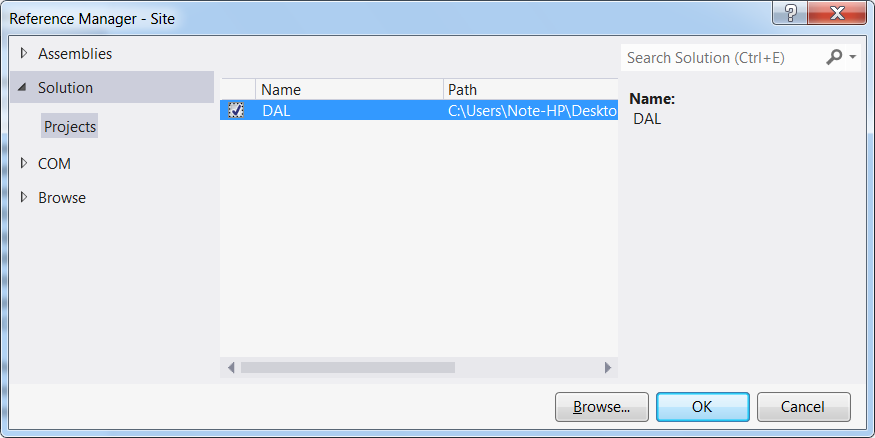
}

}

}

Adicionando referência...





using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using Site.Models; //camada de modelo

using DAL.Model;

using DAL.Persistence;

namespace Site.Controllers

{

public class ClienteController : Controller

{

//Método gerador de rota => Abrir página => /Cliente/Cadastro

public ActionResult Cadastro() //Nome da página

{

return View();

}

//Método para resgatar o envio de dados realizado pelo formulário

[HttpPost] //indicando que o método é chamado por formulário

//com requisição POST

public ActionResult CadastrarCliente(ClienteModel Model)

//classe de modelo

{

try

{

Cliente c = new Cliente(); //entidade mapeada

c.Nome = Model.Nome;

c.Sexo = Model.Sexo;

c.DataCadastro = DateTime.Now;

Endereco e = new Endereco(); //entidade mapeada

e.Logradouro = Model.Logradouro;

e.Cidade = Model.Cidade;

e.Estado = Model.Estado;

e.Tipo = Model.Tipo;

//Relacionar

e.Cliente = c; //faz com que o hibernate grave

//relacionado (foreign key)

ClienteDal d = new ClienteDal();

d.Salvar(c, e); //gravação

ViewBag.Mensagem = "Cliente cadastrado com sucesso.";

ModelState.Clear(); //limpa o conteudo dos campos do formulário

}

catch(Exception ex)

{

ViewBag.Mensagem = ex.Message;

}

//redirecionamento

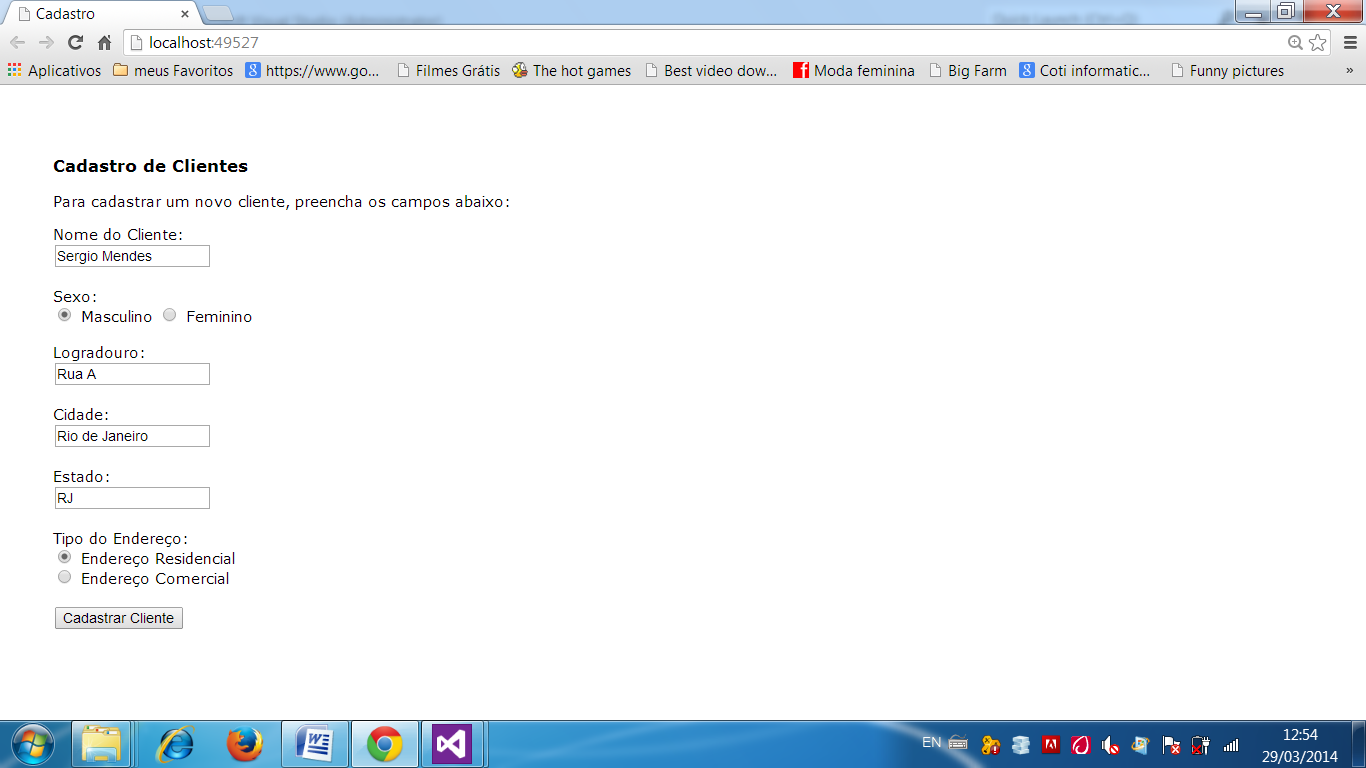
return View("Cadastro"); //nome da página

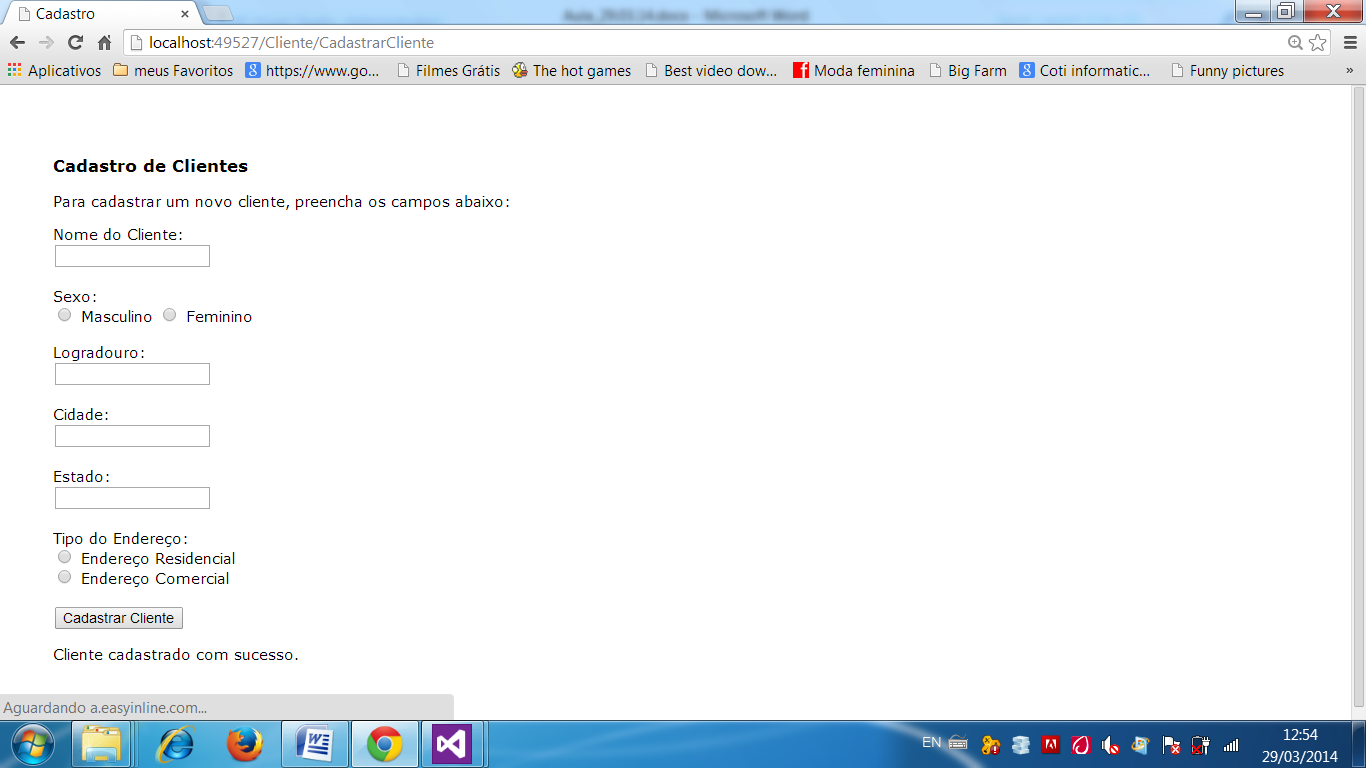
}

}

}

Executando...





No SqlServer...

