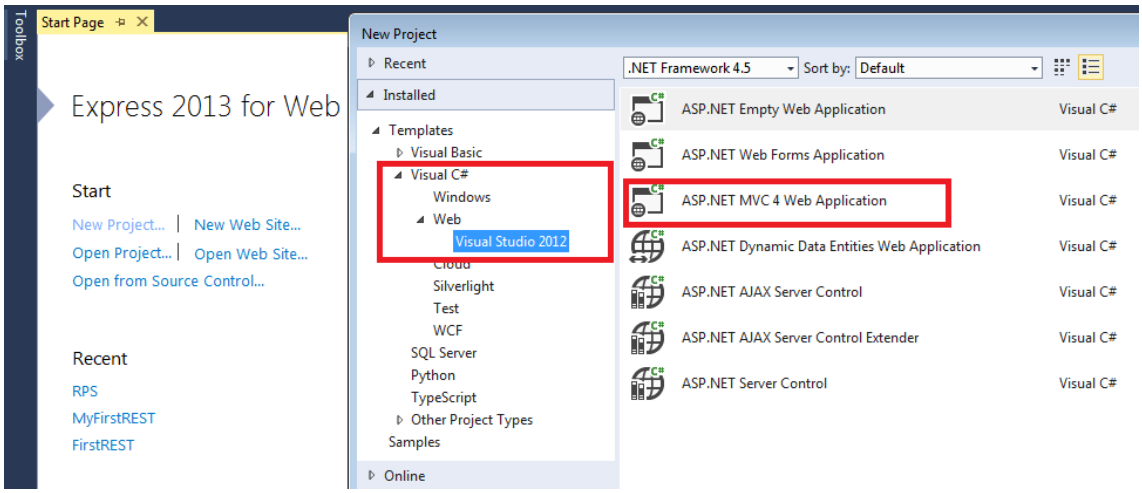
Workshop

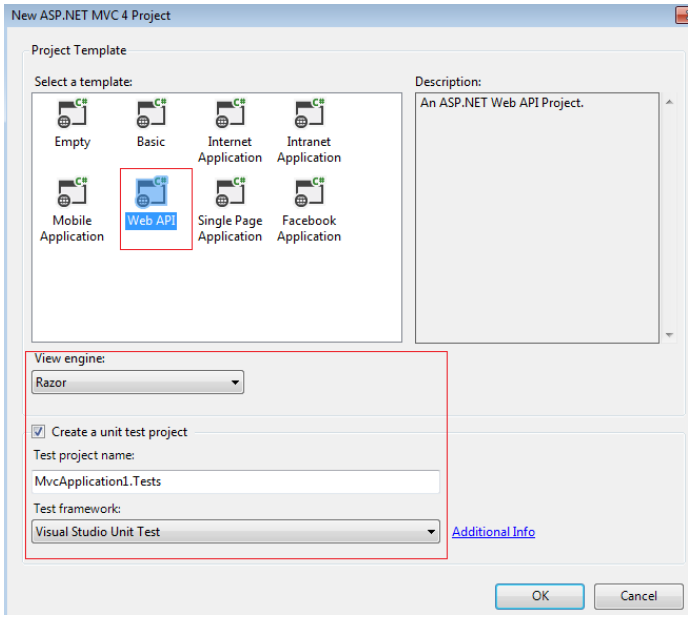
Interoperabilidade Primavera

Conteúdo

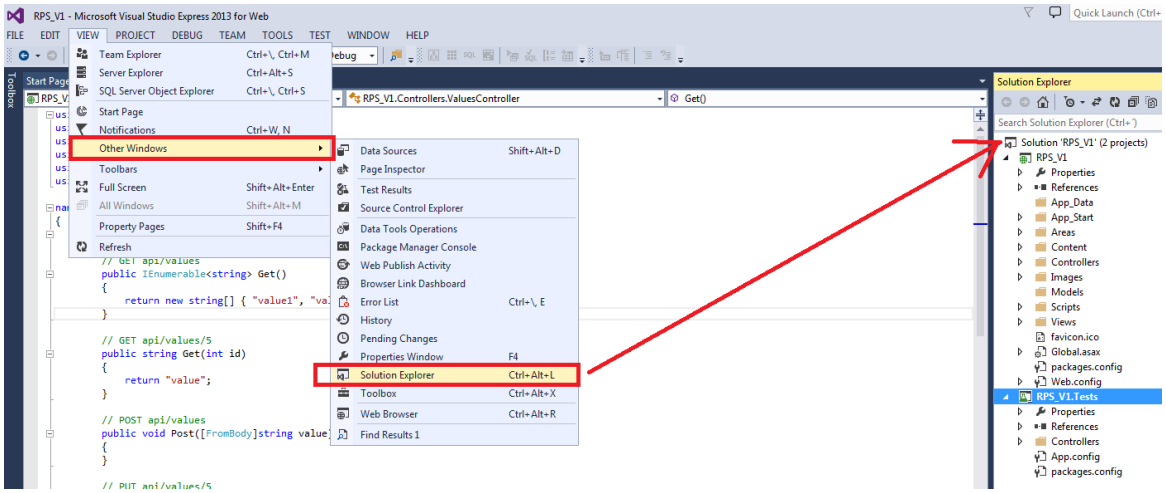
1 - Criar o projecto

File -> New Project

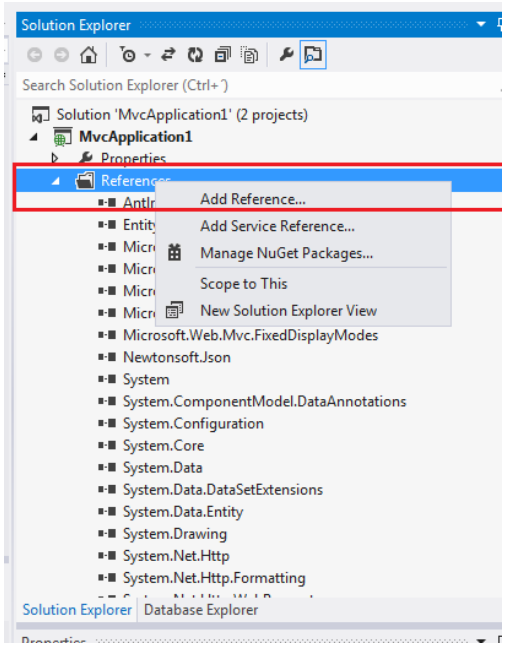




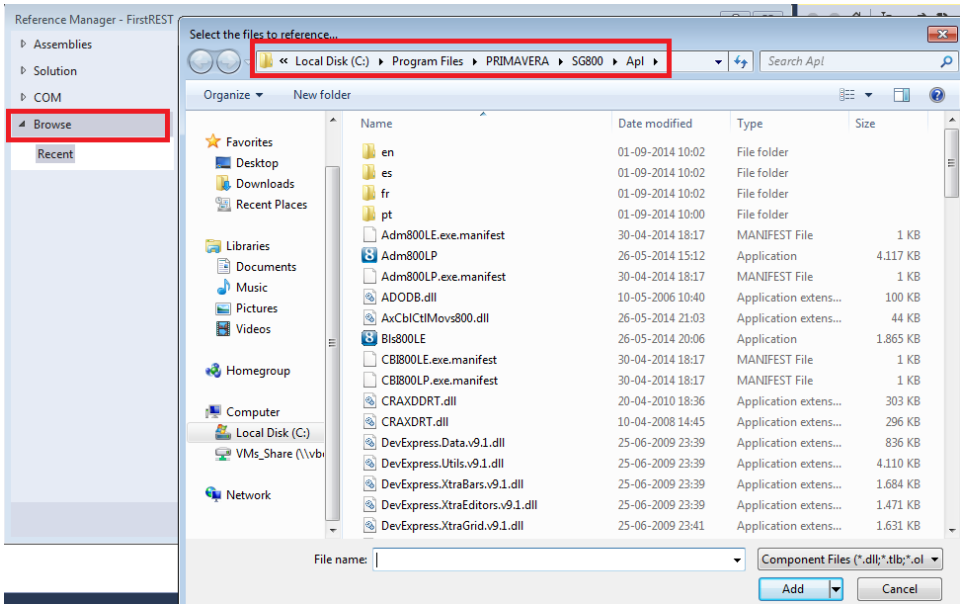
Com a criação do novo projecto um conjunto de objectos necessários à solução ficam agora disponíveis e visíveis no "Solution Explorer".



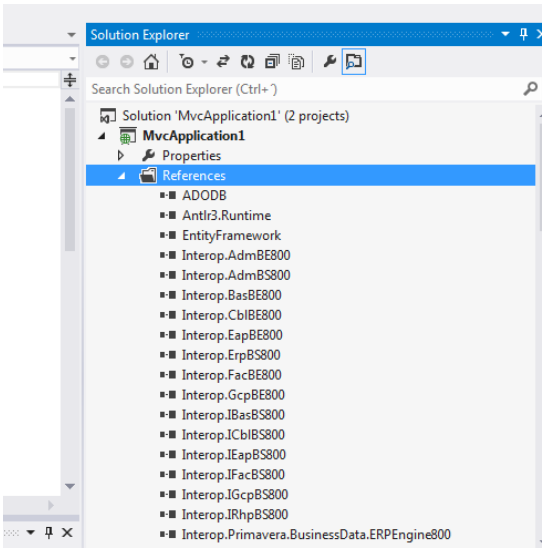
Adicionar referências de Interoperabilidade com primavera.



O directório "c:\program files\Primavera\SG800\Apl\" todas as DLLs necessárias à interoperabilidade. Devemos adicionar estas: Interop*.dll e ADODB.dll



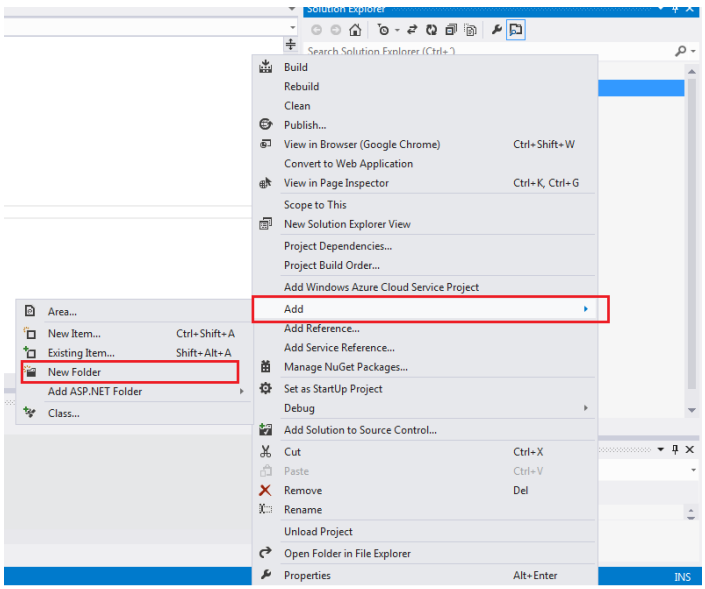
As referências estão agora disponíveis no "Solution Explorer" do projecto, no Folder "References".

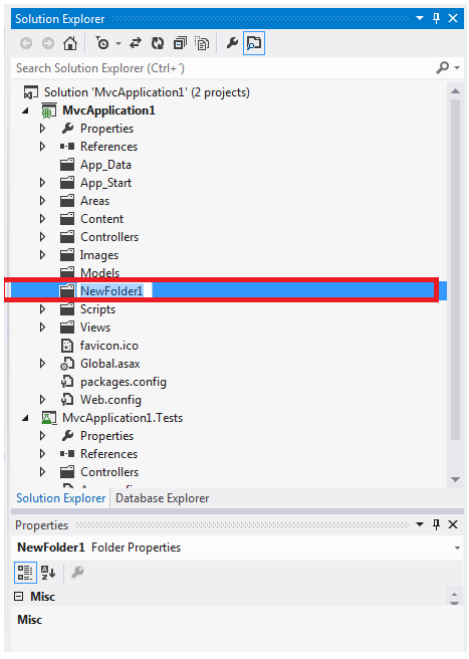


2 - Organizar o nosso projecto

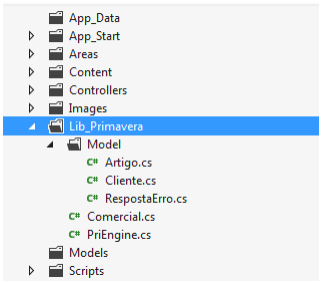
Vamos criar um Folder para organizar o nosso código.

Righ-click em MvcApplication1





Alterar a Folder com uma descrição mais personalizada ao projecto: "Lib_Primavera", e criar as subforlders necessárias à organização das classes a usar.

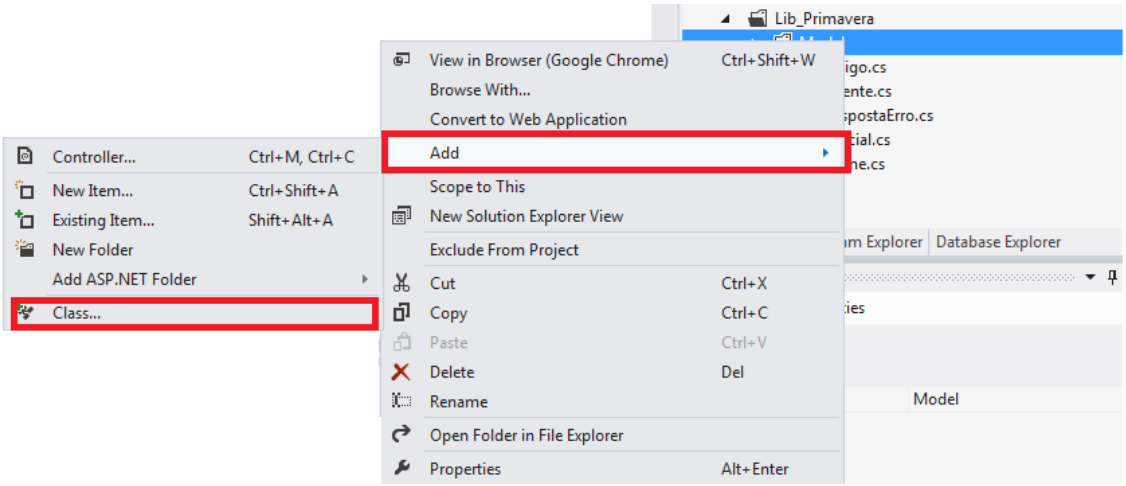


3 - Classes

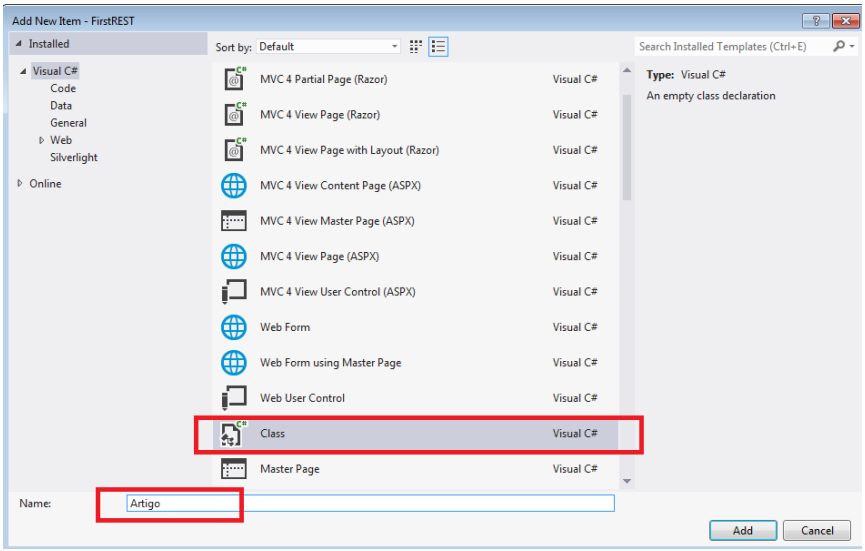
Classes Modelo

São as classes que representam as principais entidades do nosso modelo de dados. Serão usadas principalmente para a passagem de parâmetros nos métodos das classes DAL. No exemplo estão:

- Artigo
- Cliente
- RespostaErro



Classe Model "Artigo"



No exemplo dado, esta classe irá disponibilizar as propriedades (ambas do tipo string):

- CodArtigo
- DescArtigo

Ver código exemplo da classe modelo Artigo.

Classes "Business Rules"

A Classe estática "PriEngine", permite a abertura e autenticação da plataforma, bem como a disponibilização do motor Primavera. Será este "motor" que irá garantir a consistência e integridade com o sistema, uma vez que implementa as regras de negócio.

Classe de acesso aos dados Primavera (DAL)

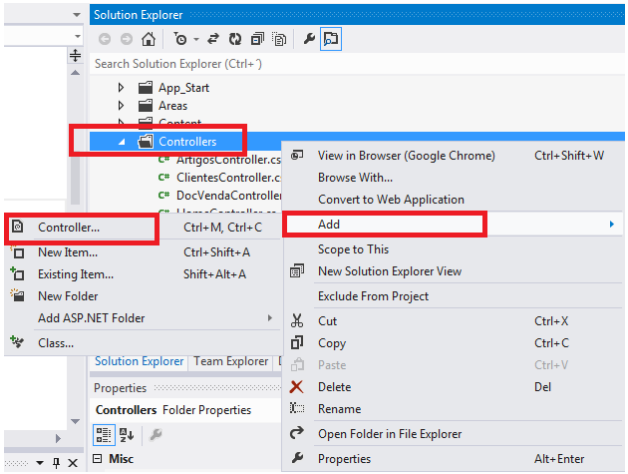
Estas classes serão as responsáveis por gerir os acessos aos dados do ERP Primavera. Os seus métodos vão ser mapeados com os métodos das classes "Controladores", que por sua vez expõem os verbos Get, Post, Put, Delete.

Clientes.ListaCliente ()
Clientes.GetCliente()
Clientes.InsereCliente ()
Clientes.ActualizaCliente ()
Clientes.DeleteCliente ()

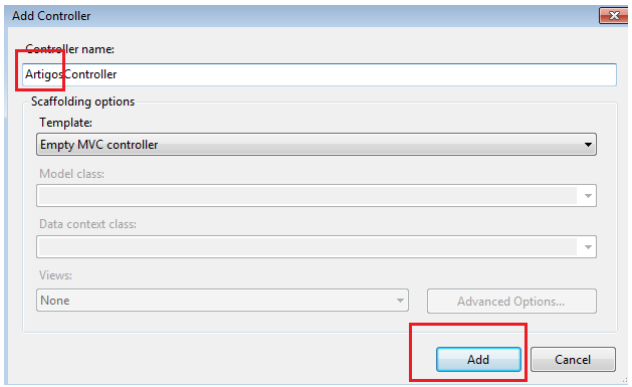
Ver código do exemplo da classe Comercial, disponibilizado no projecto FirstREST.

Classes de Controladores

As classes instanciadas contêm os métodos que ficam expostos para serem consumidos pelos diversos clientes *web*, através da norma RESTfull.



O Nome do controlador deve ser mantido pelo prefixo "Artigos" e pelo sufixo "Controller".



Por defeito o wizard não implementa "ApiController", mas apenas "Controller". Devemos proceder às alterações apresentadas na figura seguinte, para podermos posteriormente aceder aos webmethods disponibilizados pela "ApiController"

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace FirstREST.Controllers
{
    public class ArtigosController : Controller
    {
        //
```

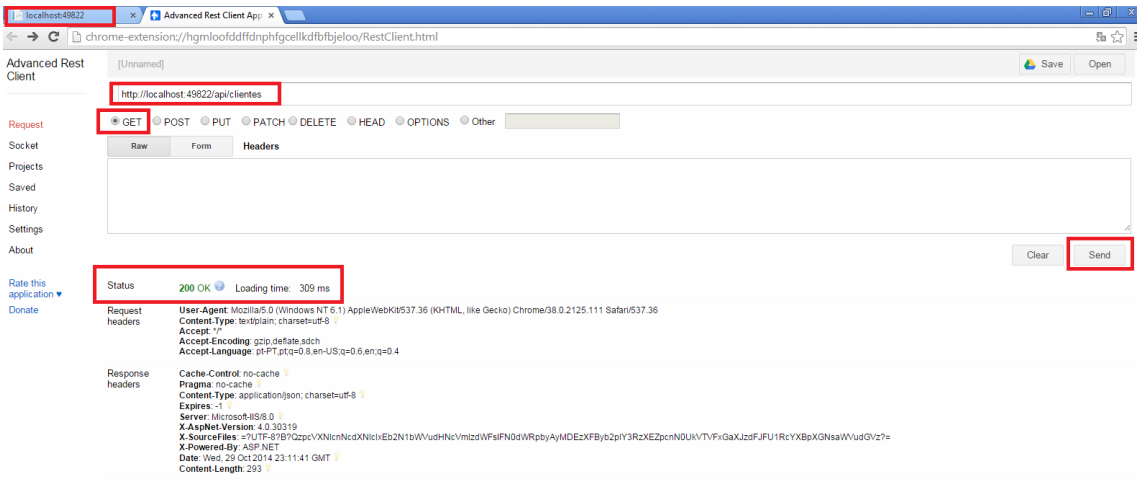
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using FirstREST.Lib_Primavera.Model;

namespace FirstREST.Controllers
{
    public class ArtigosController : ApiController
    {
        //
        // GET: /Artigos/
```

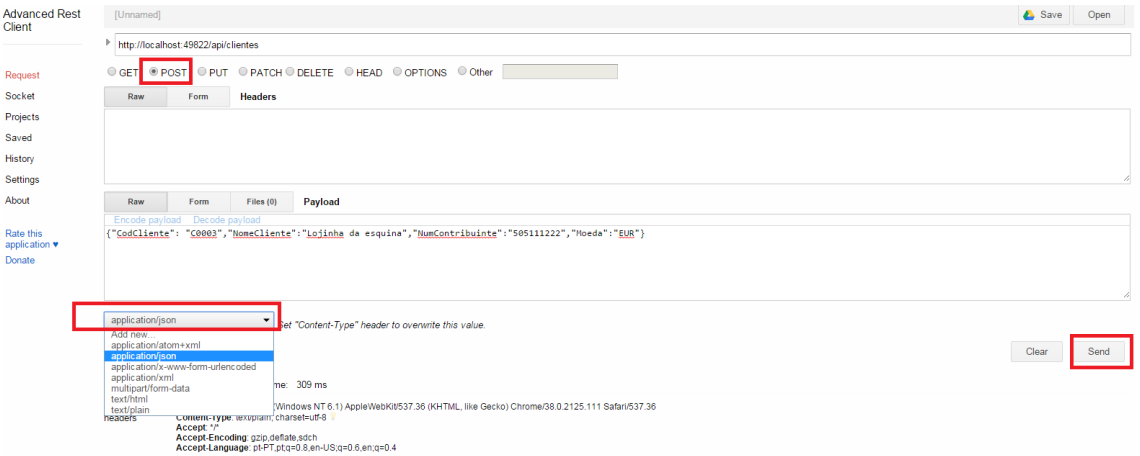
Ver código exemplo da classe "ClientesController" no projecto FirstREST, que implementa os 4 verbos: get, post, put, delete.

4 – Advanced REST Client

Usar o verbo get



Uso do verbo post



5 - Use Case para gestão de Clientes