<http://www.macoratti.net/09/09/net_wcf1.htm>

Vou iniciar o artigo com a definição do **WCF** do [Site do WCF](http://msdn.microsoft.com/en-us/netframework/aa663324.aspx) - onde lemos que:

# Windows Communication Foundation (WCF) is...

|  |  |  |  |
| --- | --- | --- | --- |
| *a part of the .NET Framework that provides a unified programming model  for rapidly building service-oriented applications that communicate across  the web and the enterprise.****[Read the WCF whitepaper](http://www.macoratti.net/library/dd943056.aspx)****.*   |  |  |  | | --- | --- | --- | | [Get it](http://www.macoratti.net/09/09/dd936009.aspx) | [Beginner's Guide](http://www.macoratti.net/09/09/dd939784.aspx) | [Learn more](http://www.macoratti.net/09/09/dd939788.aspx) | |

Traduzindo...: O **WCF** é uma parte da .NET Framework que fornece um modelo unificado de programação para construir de forma rápida aplicações distribuidas orientadas a serviço (**SOA**).

A tecnologia **WCF** - **Windows Communication Foundation**, surgiu com a .NET Framework 3.0 com o objetivo de unificar as até então existentes tecnologias de programação distribuidas como: **COM+ , MSMQ-Message Queue, Enterprise Services, .NET Remoting e Web Services.**

Com o advento da WCF foi criada uma plataforma com uma API que facilitou de forma considerável o desenvolvimento de aplicações distribuídas visto que o WCF não esta acoplado as regras de negócio que deverão ser expostas pelo serviço.

Para iniciar com o **WCF** existem alguns conceitos básicos que você deve conhecer para projetar, implementar e hospedar os seus serviços. Na verdade os **templates WCF** fornecidos no Visual Studio simplificam muito este processo visto que eles fornecem um serviço modelo que pode ser imediatamente hospedado e testado com as ferramentas de teste WCF. Assim chamar um serviço WCF a partir de um cliente é também uma tarefa simples , basta gerar um proxy e escrever o código contra o seu modelo de objetos.

**Componentes de um serviço WCF**

**O que é um serviço WCF ?**

Podemos também dar a definição formal da Microsoft que diz:  *"All communication with a Windows Communication Foundation (WCF) service occurs through the*endpoints*of the service. Endpoints provide clients access to the functionality offered by a WCF service."*

*Ou seja...:* "Toda comunicação com um serviço WCF ocorre através de endpoints do serviço. Os endpoints fornecem aos clientes o acesso às funcionalidades oferecidas por um serviço WCF."

**O que venha a ser então é esse tal de endpoint ?**

Cada serviço precisa ter um endereço (**Adress**) que define onde o mesmo reside. certo ? http://www.macoratti.net/1_positivo.gif  
Cada serviço deve possuir também um contrato (**Contract**) que define o que o serviço vai fazer. Certo ? http://www.macoratti.net/1_positivo.gif  
Cada serviço também deve possuir uma vinculação (**Binding**) que define como se comunicar com o serviço. Certo ? http://www.macoratti.net/1_positivo.gif

Então no **WCF** o relacionamento entre o **endereço (Address), o contrato (Contract) e a vinculação (Binding)**  é chamada de **EndPoint**.

Ou seja :  **Address + Binding + Contract = Endpoint**  ***(o famoso ABC's endpoints)***

Vamos começar definindo então quais os componentes fazem parte de um serviço WCF:

* **Service Class** - Uma classe de serviço WCF que implementa um serviço com um conjunto de métodos;
* **Host Environment** -  Este é o grande diferencial do WCF pois podemos usar qualquer tipo de aplicação como host. Um **Host Environment** pode ser uma *aplicação Console, um Windows Service, um Web Service, uma aplicação WIndows Forms ou o Internet Information Service* no caso de um web service normal;
* **EndPoints** - Todas as comunicações com um serviço WCF irão acontecer via **endpoints**. O **endpoint** é composto de 3 partes (**ABC´s endpoints)** :
  + **Address** - Consiste em especificar um endereço que define onde o *endpoint* esta hospedado. Cada endpoint possui um endereço especificado com ele que é usado para identificar e localizar o *endpoint*. (*Geralmente é definido pela instância da classe Uri)*
  + **Binding** - Especifica como o cliente irá se comunicar *(qual transporte será usado: Http, Tcp, Ipc, WS, MSMQ, etc.)* com serviço e o endereço onde o endpoint esta hospedado. A definição do Binding pode ser feita no arquivo de configuração ou via programação.
  + **Contract** - Especifica um contrato que define quais métodos da classe de Serviço serão acessíveis via *endpoint*, onde cada *endpoint* poderá expor um conjunto diferente de métodos; (*O contrato é representado por uma Interface que deverá ser decorada com o atributo****ServiceContract****)*

A primeira etapa para criar um serviço WCF e efetuar a **definição do contrato** pois é o contrato que vai definir quais operações serão expostas pelo serviço, que informações são necessárias para que essas operações sejam executadas e qual o tipo de retorno esperado.

**Obs:** O contrato é uma interface que contém as assinaturas dos métodos que serão expostos. A interface deverá ser decorada com o atributo:**ServiceContract**

## WCF - Tipos de contratos

A seguir temos um resumo dos tipos de contratos existentes onde os código exemplo mostrados foram obtidos do template padrão para **WCF Service** obtido no Visual Web Developer 2008 Express.

Existem 3 tipos de contratos WCF :

**Contratos de serviços (Service Contracts)**- Descrevem as operações que um serviço pode realizar. *(Mapeia os tipos CLR para WSDL)*. Será detalhado mais abaixo. Exemplo:

|  |
| --- |
| **<ServiceContract()> \_**  Public Interface IService  **<OperationContract()> \_**  Function GetData(ByVal value As Integer) As String  **<OperationContract()> \_**  Function GetDataUsingDataContract(ByVal composite As CompositeType) As CompositeType  ' TODO: Add your service operations here  End Interface |

**Contratos de Dados (Data Contracts)**- Descreve a estrutura de dados usada no serviço. *(Mapeia tipos CLR para XSD)*.  Um **Data Contract** é um acordo formal entre um serviço e um cliente que descreve os dados que serão trocados entre ambos. Para estabelecer a comunicação, o cliente e o serviço não necessitam trocar necessariamente os mesmos tipos de dados, **devem trocar apenas os mesmos data contracts**.

Um **Data Contract** especifica para cada parâmetro ou tipo de retorno qual informação será serializada *(convertida em XML)* para ser trocada.  Os DataContracts são definidos através declasses e uma classe **DataContract** deverá ser decorada com o atributo **DataContract** e os campos e propriedades que o tipo possui devem ser decorados com o atributo **DataMember**. *(A****partir do service pack  versão 3.5 da plataforma .NET isso não é mais obrigatório)***

|  |
| --- |
| ' Use a data contract as illustrated in the sample below to add composite types to service operations.  **<DataContract()> \_**  Public Class CompositeType  Private boolValueField As Boolean  Private stringValueField As String  **<DataMember()> \_**  Public Property BoolValue() As Boolean  Get  Return Me.boolValueField  End Get  Set(ByVal value As Boolean)  Me.boolValueField = value  End Set  End Property  End Class |

**Contratos de Mensagens (Message Contracts)** - Define a estrutura de mensagens usadas no serviço. (*Mapeia os tipos CLR para mensagens SOAP)*

Os contratos de mensagens permitem ao serviço interagir diretamente com mensagens e podem ser tipados ou não tipados. Eles são úteis em casos de interoperabilidade e quanto existe um formato de mensagem com o qual devemos dar conformidade e/ou customizar.

Por meio dos contratos de mensagens podemos ter um controle maior sobre a estrutura de mensagem **SOAP** usada pelo **WCF** efetuando uma personalização dos componentes da mensagem.

Os Contratos de Mensagens são definidos em classes e as mesmas devem ser decoradas com o atributo **MessageContract**. Além disso uma classe definida com um contrato de Mensagem deve possuir um construtor sem parâmetros.

Nas classes para especificar se um campo fará parte do **header** ou do **body** da mensagem devemos usar os atributos **MessageHeader** ou **MessageBodyMember**.

Exemplo:

|  |
| --- |
| **[MessageContract]**  public class BankingTransaction  {  **[MessageHeader]** public Operation operation;  **[MessageHeader]** public DateTime transactionDate;  **[MessageBodyMember]** private Account sourceAccount;  **[MessageBodyMember]** public int amount;  } |

## Projetando um contrato para um serviço WCF

Vamos falar agora um pouco mais sobre os **Contratos de serviços (Service Contracts).**

Para implementar um serviço WCF iniciamos pela definição do contrato para o serviço e então implementamos o contrato em um tipo de serviço. Um contrato para um serviço geralmente envolve a aplicação do atributo **ServiceContract**para uma interface e então a aplicação do atributo **OperationContract** para cada método exposto como parte do contrato do serviço.  A seguir veja as regras para a definição de um contrato:

* O contrato deve ser uma interface contendo as assinaturas dos métodos do serviço;
* A interface deverá ser decorada com o atributo : **ServiceContract** de outra forma será lançada a exceção **InvalidOperationException**;
* Todas as operações que serão disponibilizadas devem ser decoradas com o atributo **OperationContract**;
* É obrigatória a definição de pelo menos uma operação com o atributo **OperationContract**;

Na figura abaixo temos um exemplo de um *template* padrão fornecido pelo Visual Studio onde vemos um contrato definido na interface com duas operações:

|  |
| --- |
| http://www.macoratti.net/09/09/netwcf11.gif |

**Criando um serviço WCF na prática**

Vou mostrar um exemplo bem simples onde irei criar um serviço WCF para validar um email usando o Visual Studio 2008 e em seguida irei consumir o serviço em um projeto *Windows Forms.*

Abra o Visual Studio 2008 e crie um novo projeto do tipo (**File-> New Project)**do tipo **Web** usando o template **WCF Service Application** com o nome **Wcf\_Exemplo**;

|  |
| --- |
| http://www.macoratti.net/09/09/net_wcf11.gif |

A seguir vamos definir no arquivo de interface  **IService.vb**o contrato do nosso serviço definindo a operação **ValidarEmail()** que irá validar um email e retornar **True ou False** conforme o resultado. Para isso remova o código existente e defina o seguinte código no arquivo de interface **IService1.vb:**

|  |
| --- |
| http://www.macoratti.net/09/09/net_wcf12.gif |

Agora no arquivo**Service.svc** implemente a interface **IService1** definida acima e o método **ValidarEmail** conforme abaixo:

|  |
| --- |
| http://www.macoratti.net/09/09/net_wcf13.gif |

Execute um **Build** no projeto e verifique se não há erros.*(Se desejar você pode publicar o serviço em: http://localhost/Wcf\_Exemplo)*

Pronto, já temos nosso serviço criado e preparado para ser consumido. Vamos fazer isso em uma aplicação Windows Forms.

**Consumindo o serviço WCF**

A partir do menu **File** selecione **Add -> New Project** e selecione o template **Windows Forms Application** com o nome **windows\_WCF**;

|  |
| --- |
| http://www.macoratti.net/09/09/net_wcf14.gif |

Vamos incluir uma referência ao serviço WCF criado. Para isso clique com o botão direito sobre o nome do projeto **Windows Forms** e selecione **Add  Service Reference;**

A seguir clique no botão **Discover** para procurar por serviços na Solution. Deverá ser encontrado o nosso serviço WCF criado conforme a figura abaixo. Vamos aceitar o nome padrão da referência e clicar no botão **OK**;

|  |
| --- |
| http://www.macoratti.net/09/09/net_wcf15.gif |

No formulário **form1.vb** da aplicação Windows Forms inclua a partir da ToolBox um controle Label, um controle TextBox (txtEmail) e um botão de comando (btnValidarEmail) conforme o leiaute abaixo:

|  |
| --- |
| http://www.macoratti.net/09/09/net_wcf16.gif |

No evento **Click** do botão de comando inclua o seguinte código:

|  |
| --- |
| Private Sub btnValidarEmail\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnValidarEmail.Click  ' criando uma instƒncia do nosso servi‡o WCF  Dim WCF\_ValidaEmail As New ServiceReference1.Service1Client  'usando o m‚todo ValidarEmail do nosso servi‡o  If WCF\_ValidaEmail.ValidarEmail(txtEmail.Text) Then       MsgBox(" Email V lido.")  Else       MsgBox(" Email Inv lido.")  End If  End Sub |

Defina o projeto Windows Forms como o projeto que será executado clicando sobre o nome do projeto e selecionando a opção - **Set as Startup Project.**

Ao executar projeto efetuando testes com alguns emails iremos obter :

|  |  |
| --- | --- |
| http://www.macoratti.net/09/09/net_wcf17.gif | http://www.macoratti.net/09/09/net_wcf18.gif |
| Email válido | E-mail inválido (existe uma vírgula após yahoo) |

Embora seja muito simples e eu não tenha me preocupado com validações nem com as melhores práticas mostrei como criar um serviço WCF e consumi-lo usando o VS 2008. Podemos fazer isso de outras formas e em breve voltarei ao assunto.

Aguarde... http://www.macoratti.net/1_pisca.gif

Pegue o projeto completo aqui: http://www.macoratti.net/downl.gif  [**windows\_WCF.zip**](http://www.macoratti.net/09/09/windows_WCF.zip)**e**[**Wcf\_Exemplo.zip**](http://www.macoratti.net/09/09/Wcf_Exemplo.zip)

Eu sei é apenas WCF mas eu gosto...http://www.macoratti.net/1_positivo.gif

Referências:

* Fudamentals WCF concepts - <http://msdn.microsoft.com/en-us/library/ms731079.aspx>
* What is WCF - <http://msdn.microsoft.com/en-us/library/ms731082.aspx>
* WCF - Details - <http://msdn.microsoft.com/en-us/library/ms733103.aspx>
* Using Data Contracts - <http://msdn.microsoft.com/en-us/library/ms733127.aspx>
* Using Message Contracts- <http://msdn.microsoft.com/en-us/library/ms730255.aspx>

## [.NET Framework 3.0](http://www.macoratti.net/07/08/net_vs3.htm)

## <http://www.macoratti.net/10/10/wcf_ds1.htm>

**http://www.macoratti.net/maco10.gif  .NET  - Apresentando WCF Data Services**

O **WCF Data Services**ou Windows Communication Foundation Data Services *(originalmente conhecido como ADO .NET Data Services)* é um framework que pode ser usado para expor uma camada de dados via interface **REST** usando **JSON** ou **Atom** como formato de saída.

Por ser uma tecnologia nova alguns termos usados podem gerar dúvidas para quem esta iniciando o aprendizado por isso possuir conceitos sobre  web services , WCF e ADO .NET Data Services vai ajudar muito no entendimento dos recursos apresentados neste artigo. *Sugestão : Vejas as referências no final do artigo.*

O**WCF Data Services** permite a criação e o consumo de serviços de dados para Web ou para Intranet através da utilização do protocolo**Open Data (OData).** O protocolo **OData** permite que você exponha a sua fonte de dados como um recurso que é endereçável via URI. Isto permite o acesso e a alteração de dados através da semântica **REST - Representation State Transfer,** mais especificamente os comandos HTTP padrão: GET, PUT, POST e DELETE.

|  |
| --- |
| O [Open Data Protocol](http://www.odata.org/) (OData) é um protocolo para consulta e atualização de dados na web, sendo um conjunto de extensões para o protocolo [AtomPub](http://en.wikipedia.org/wiki/AtomPub" \t "_blank).   O AtomPub é um protocolo baseado em HTTP usado para o mesmo fim, criar e atualizar os recursos da web, trabalhando em conjunto com o [Atom Syndication Format](http://en.wikipedia.org/wiki/AtomPub" \t "_blank), uma linguagem baseada em XML, usado para a web feeds.   OData anteriormente era conhecido como Protocolo Astoria Protocol, protocolo ADO.NET Data Services ou como um conjunto de convenções / extensões para AtomPub.  *Obs: o Google usa o protocolo Gdata.* |

A seguir temos o diagrama que ilustra a arquitetura dos **WCF Data Services** para exposição de  feeds **OData**:

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_arq1.gif |

Podemos usar o **WCF Data Services** para expor qualquer fonte de dados com suporte a consultas LINQ, ou seja, que implemente**IQueryable/IUpdatable,** através de uma interface **RESTful** a aplicação pede um endereço para o serviço e o endereço é traduzido como uma consulta LINQ que retorna os dados desejados.

O framework Data Services framework  facilita muito a criação de serviços de dados flexíveis que são integrados com a web. O WCF Data Services usa URI's para apontar para porções de dados, isso permite que que formatos de representação como **JSON e ATOM**, baseados em XML, sejam usados para o transporte de dados.

Esse estilo de programação é conhecido como **RESTful** e permite a manipulação de coleções via URIs onde os agentes consumidores podem interagir com os dados através de comandos HTTP como **GET, POST, PUT ou DELETE.**

Pensando dessa forma temos as seguintes possibilidades:

* Permitir que aplicações AJAX acessem banco de dados via HTTP sem ter que escrever um Web Service como um *proxy*;
* Criar um sistema de acesso a dados **RESTful**;
* Expor um modelo de entidades via URI;
* Ser consumido por clientes distintos como : Windows, Web, Console, Ajax e SilverLight;

O**WCF Data Services** cria de forma automática um um**Web Service REST-ful**que gera uma saída baseada em JSON ou Atom em um configuração fornecida. Dessa forma o desenvolvedor não esta obrigado a criar o **Web Service**, serializar os dados para **JSON ou Atom**, tendo assim uma aumento de desempenho no desenvolvimento da aplicação.

**Da Teoria a Prática**

Creio que um exemplo prático simples vai ajudar a esclarecer melhor os conceitos sobre o **WCF Data Services.**

No exemplo deste artigo vamos usar o**WCF Data Services** para expor uma camada de dados gerada pelo Entity Framework e realizar consultas nos dados via URI sem ter que escrever nenhum método para isso. Confira...

Para atingir nosso objetivo vamos seguir as seguintes etapas:

1. Criar uma aplicação ASP.NET Web;
2. Definir um modelo de dados usando o Entity Data Model a partir do banco de dados Northwind;
3. Incluir um serviço de dados  na aplicação Web;
4. Permitir o acesso ao serviço de dados;
5. Realizar consultas no serviço de dados via URI;

Eu estou usando o **Visual Web Developer 2010 Express Edition**  como ferramenta para criar os projetos e serviços deste artigo.

Abra o VWD 2010 Express e no menu File selecione New Project;

A seguir selecione o template ASP .NET Web Application e informe o nome **Web\_Demo\_WCF\_DS** e clique em OK;

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds11.gif |

2-) A seguir no menu **Project** selecione**Add New Item**;

3-) Na janela **Add New Item**, em Templates selecione **ADO .NET Entity Data Model**, informe o nome **Northwind.edmx**e clique em **Add**;

4-) Seguindo o assistente selecione o item : **Generate From DataBase**  e clique em **Next>** ;

5-) Na próxima janela do assistente selecione a conexão com o banco de dados **Northwind.mdf** aceite o nome **NorthwindEntities** e clique em **Next>** ;

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds12.gif |

6-) Na próxima janela selecione as tabelas: **Categories, Customers, Order, Order Details e Products**;  aceite o nome para Model Namespace como **NorthwindModel**, e clique no botão **Finish**;

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds13.gif |

Abaixo temos o modelo conceitual representado graficamente no descritor do **EF**. Além disso foi gerado o ObjectContext que gerencia o estado dos objetos e permite realizar operações **CRUD :**

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds14.gif |

Vamos agora criar no serviço de dados.No menu Project selecione Add New Item e a seguir selecione o template **WCF Data Service** e informe o nome **Wcf\_Northwind\_DS.svc** e clique em **Add**;

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds15.gif |

Dessa forma temos o nosso serviço de dados do tipo WCF Data Service criado.

Até o momento temos o modelo de entidades gerado e o serviço de dados criado. Precisamos informar o serviço de dados qual o contexto que estamos usando e o que ele vai poder disponibilizar.

Para isso abr o arquivo do serviço criado e no lugar de **[[class name]]** informe o nome do nosso modelo de entidades: **NorthwindEntities** criado pelo Entity Framework;

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds16.gif |

Descomente também a linha de código : **config.SetEntitySetAccessRule("MyEntitySet", EntitySetRights.All)**

Substituindo a palavra **MyEntitySet** por um asterísco (\*);

|  |  |
| --- | --- |
| config.SetEntitySetAccessRule("MyEntityset",EntitySetRights.AllRead)   **==>** | config.SetEntitySetAccessRule("\*",EntitySetRights.All) |

Na figura baixo temos como deve ficar o código após essas alterações

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds17.gif |

Com essas alterações informamos ao nosso serviço de dados qual é a classe que contém o mapeamento para as nossas entidades, no caso **NorthwindEntities**; e também definimos que não haverá restrição a nenhuma entidade : **“EntitySetRights.All”**

Executando o projeto iremos obter a seguinte página referente ao nosso serviço de dados:

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds18.gif |

Observe que o nosso serviço de dados WCF listou todas as nossas entidades: **Categories, Customers, Order\_Details, Orders, Products.**

Lembra que dissemos que uma característica de um serviço WCF  é aceitar consultas via URI ?

Pois é isso que vamos fazer...

Para ver todos os **Clientes** digite a seguinte URI: <http://localhost:7410/Wcf_Northwind_DS.svc/Customers>

O resultado exibido será o seguinte:

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds19.gif |

Para melhor visualizarmos o resultado vamos desativar o modo de exibição de **Feeds** do navegador Internet Explorer.

Selecione :**Ferramentas -> Opções da Internet**,  e na janela **Opções da Internet**abra a aba **Conteúdo** e clique em **Configurações para Feeds e Web Services;**

Em seguida desmarque a  opção : **Ativar o modo de exibição de leitura de feed,** conforme mostra a figura abaixo:

|  |  |
| --- | --- |
| http://www.macoratti.net/10/10/wcf_ds1a.gif | http://www.macoratti.net/10/10/wcf_ds1b.gif |

Após isso clique OK até confirmar as alterações e refazendo a execução da URL iremos obter a relação dos **Clientes** no formato XML conforme a figura abaixo:

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds1c.gif |

Observe que cada cliente possui uma referência única que no caso refere-se a chave primária da tabela Customers. Para visualizar um cliente particular basta usar esta referência na URI digitando:  <http://localhost:7410/Wcf_Northwind_DS.svc/Customers('ALFKI>')

O resultado é mostrado a seguir:

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds1d.gif |

Deseja ver os pedidos deste cliente ?

Então digite: <http://localhost:7410/Wcf_Northwind_DS.svc/Customers('ALFKI')/Orders>

E veja o resultado:

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds1e.gif |

Vamos agora exibir os valores para a entidade categories, mais especificamente para a primeira categoria.

Digite a URI: <http://localhost:7410/Wcf_Northwind_DS.svc/Categories(1>)

Iremos obter o seguinte resultado:

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds1g.gif |

Se desejarmos exibir a figura que esta no formato binário no campo **Picture** basta digitar:

<http://localhost:7410/Wcf_Northwind_DS.svc/Categories(1)/Picture/$value>

**Deseja filtrar os clientes por cidade ?**

Vamos exibir somente os clientes de Londres , para isso digite:

**http://localhost:7410/Wcf\_Northwind\_DS.svc/Customers?$filter=City eq 'London'**

O resultado será:

|  |
| --- |
| http://www.macoratti.net/10/10/wcf_ds1f.gif |

Eu poderia continuar mas creio que você já percebeu todo o poder das consultas dos serviços de dados WCF.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Operações suportada por um WCF Data Service:**   |  |  | | --- | --- | | orderby | - realiza a ordenação por um parâmetro qualquer | | top | - seleciona a quantidade de registros informada | | filter | - realiza a pesquisa com base em algum filtro | | value | - exibe um valor |   **Operadores suportados:**   |  |  | | --- | --- | | eq | - igual | | ne | - não igual | | gt | - maior que | | ge | - maior ou igual | | lt | - menor que | | le | - menor ou igual | | and | - AND lógico | | or | - OR lógico | | not | - NOT lógico | |

Além de permitir realizar consultas via URI o **WCF Data Services** permite também que você consulte um serviço de dados a partir de uma aplicação cliente baseada no .NET Framework usando as classes geradas pelo serviço de dados. Você pode executar essas consultas usando os seguintes métodos:

1. Executando uma consulta LINQ contra o **DataServiceQuery** que você obtém a partir do **DataServiceContext** que a ferramenta**Add Data Service Reference** gera;
2. De forma implícita, pela enumeração sobre o **DataServiceQuery**  que você obtém a partir do **DataServiceContext**  que a ferramenta**Add Data Service Reference**gera;
3. Explicitamente, chamando o método Execute no **DataServiceQuery**   ou o método **BeginExecute** para execução assíncrona;

**Obs1 : Um DataServiceQuery representa uma simples requisição de consulta a um serviço de dados (data service);  
Obs2 : Um DataServiceContext representa um contexto em tempo de execução de um serviço de dados (data service)**

No exemplo a seguir temos um trecho de código que mostra como usar um **DataServiceContext** para explicitamente executar uma consulta que retorna todos os clientes (*customers*) de um serviço de dados gerado para o banco de dados **Northwind**:

|  |
| --- |
| ' Define uma requisição URI que retorna os clientes Dim customersUri = New Uri(svcUri, "Northwind.svc/Customers")  ' Cria um DataServiceContext usando o serviço URI. Dim context = New DataServiceContext(svcUri)  Try    ' Percorre o resutlado da consulta    For Each customer As Customer In context.Execute(Of Customer)(customersUri)       Console.WriteLine("Nome do Cliente: {0}", customer.CompanyName)     Next Catch ex As DataServiceQueryException     Throw New ApplicationException("Ocorreu um erro durante a execução da consutla.", ex) End Try |

Este é apenas um exemplo de como usar os recursos do serviços de dados , aguarde em outro artigo onde eu mostrarei como efetivamente consumir os **WCF Data Services**.

Até o próximo artigo WCF Data Services...http://www.macoratti.net/1_bye.gif

|  |
| --- |
| [**Veja os**](http://www.macoratti.net/destaque.htm)[**Destaques e novidades do SUPER DVD Visual Basic  (sempre atualizado) : clique e confira !**](http://www.macoratti.net/destaques.htm)  **Quer migrar para o VB .NET ?**  Veja mais sistemas completos para a plataforma .NET no [**Super DVD .NET**](http://www.macoratti.net/superdvd.htm)**, confira...**  **Quer aprender C# ??**  **Chegou o [Super DVD C#](http://www.macoratti.net/superdvdc.htm)**com exclusivo material de suporte e vídeo aulas com curso básico sobre C#.  Veja também os Cursos com vídeo aulas e projetos exemplos:   * [**VB .NET Básico - Vídeo Aulas**](http://www.macoratti.net/curso_vbnet_basico.htm) * [**C# Básico - Vídeo Aulas**](http://www.macoratti.net/curso_cshp_basico.htm) |

**João 8:45 Mas porque eu digo a verdade, não me credes.**

**João 8:46 Quem dentre vós me convence de pecado? Se digo a verdade, por que não me credes?**

**João 8:47 Quem é de Deus ouve as palavras de Deus; por isso vós não as ouvis, porque não sois de Deus.**

             Gostou ?  http://www.macoratti.net/face32.jpg [Compartilhe no Facebook](http://www.macoratti.net/10/10/wcf_ds1.htm)  http://www.macoratti.net/twiter32.jpg [Compartilhe no Twitter](javascript:%20void(0);)

Referências:

* [Seção VB .NET do Site Macoratti.net](http://www.macoratti.net/pageview.aspx?catid=1)
* [Super DVD .NET - A sua porta de entrada na plataforma .NET](http://www.macoratti.net/superdvd.htm)
* [Super DVD Vídeo Aulas - Vídeo Aula sobre VB .NET, ASP .NET e C#](http://www.macoratti.net/Vendas/Produtos.aspx?ID=22)
* [Seção C# do site Macoratti.net](http://www.macoratti.net/pageview.aspx?catid=18)
* [WCF - Introdução](http://www.macoratti.net/09/09/net_wcf1.htm)

## [ADO .NET Data Services](http://www.macoratti.net/09/02/adn_ds1.htm)

## [Usando ADO .NET Data Services](http://www.macoratti.net/09/02/adn_ds2.htm)

## [Introdução ao ADO .NET Data Service](http://www.macoratti.net/10/03/adn_ds1.htm)

* [Windows Communication Foundation (WCF) Data Services](http://msdn.microsoft.com/en-us/data/bb931106.aspx)
* <http://www.odata.org/developers/protocols>

[José Carlos Macoratti](http://www.macoratti.net/)