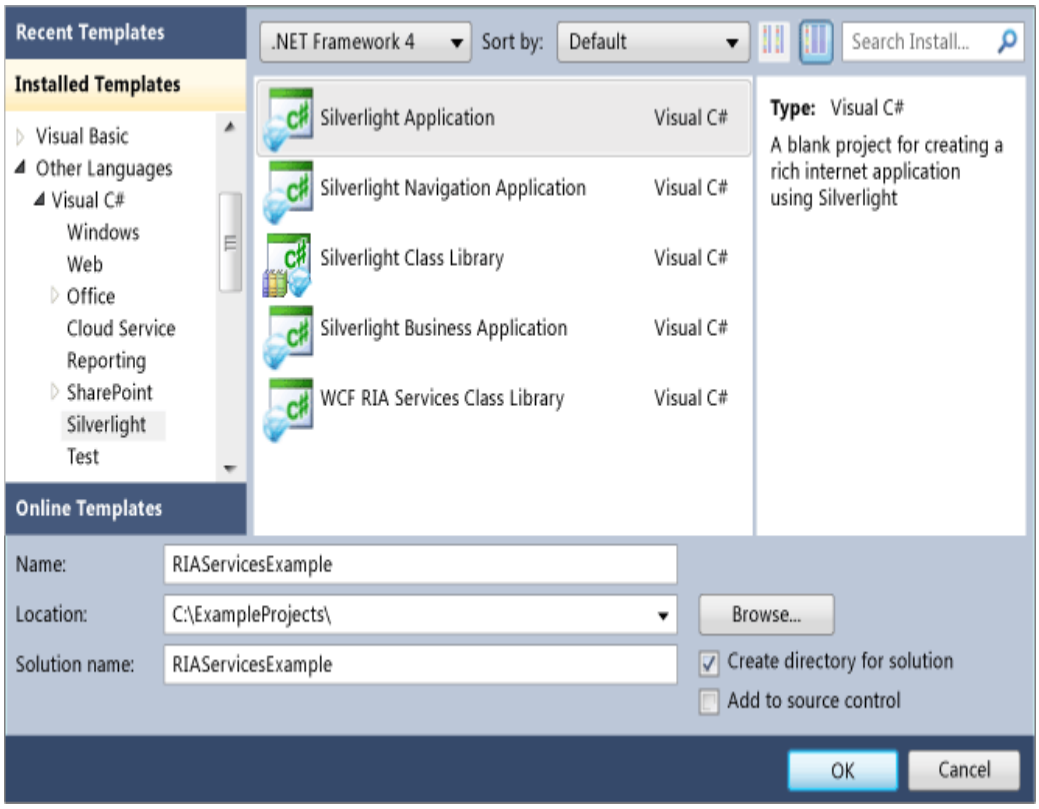
Criar uma solução com um link Serviços RIA entre os projetos

Para configurar uma solução de Serviços de RIA

1. Criar um novo projecto RIA Services no Visual Studio 2010, selecionando **Arquivo, Novo** e, em seguida, **Projeto.**

   A caixa de diálogo **New Project será** exibida.

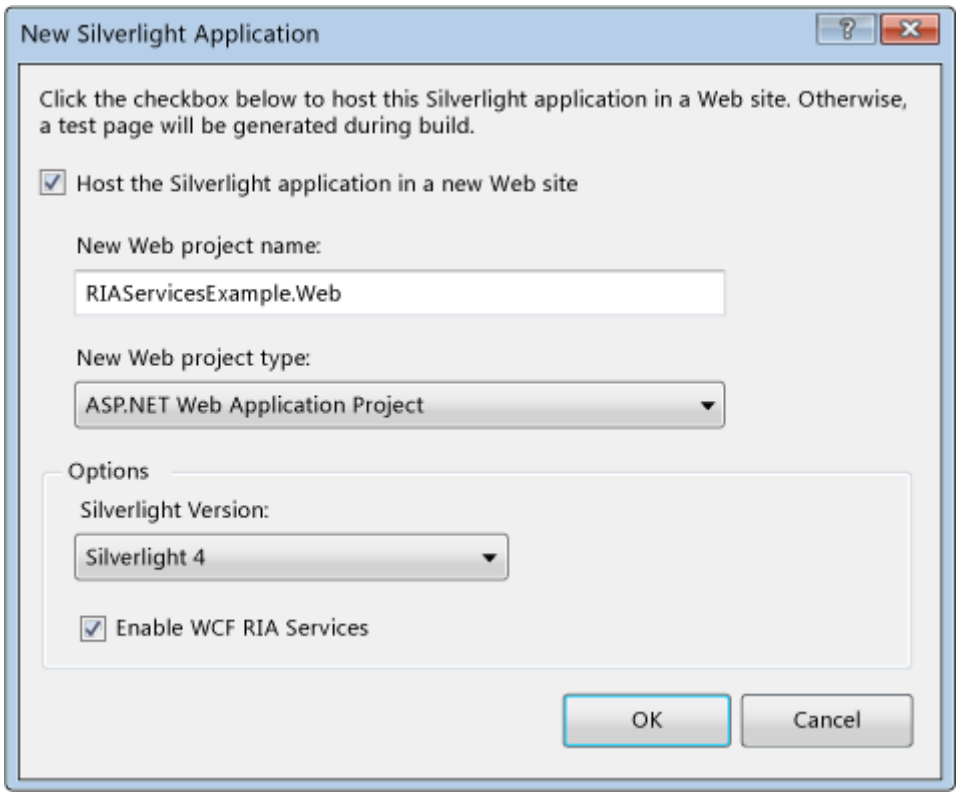
2. Selecione o modelo de **aplicativo** de **Silverlight** grupo dos **Modelos Instalados** e **RIAServicesExamplo** o nome do novo projeto.



3. Clique em **OK.**

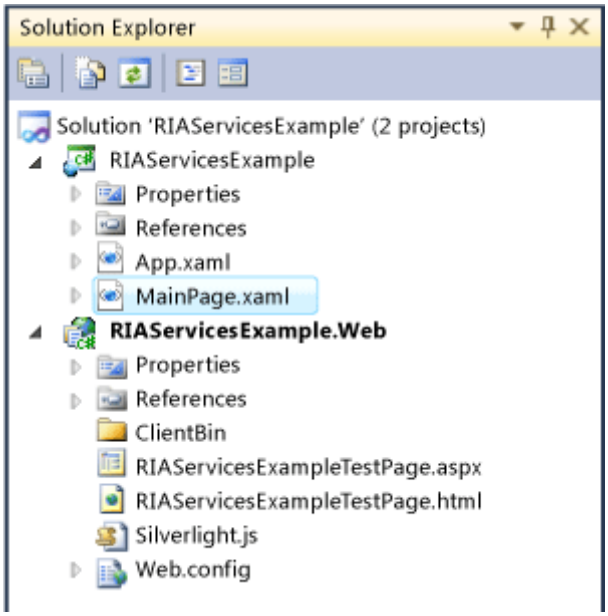
   O **novo Silverlight** caixa de diálogo **Application** aparece.

4. Selecione **Ativar o WCF RIA** caixa de seleção **Serviços** perto da parte inferior da caixa de diálogo. Marcar essa caixa cria um link RIA Serviços entre o projeto do cliente e do projeto de servidor.



5. Clique em **OK** para criar a solução.

A solução contém dois projetos: um projeto do cliente e um projeto de servidor. O projeto do cliente é chamado RIAServicesExample e que contém o código do Silverlight que você usa para criar a camada de apresentação. O projeto de servidor é chamado RIAServicesExample.Web e que contém o código da camada intermediária.



Criação de Modelos de Dados

Nesta seção, você irá criar as classes ADO.NET Entity que representam dados do banco de dados AdventureWorksLT. RIA Services trabalha com uma variedade de classes de modelagem de dados e fontes de dados.

Atenção:
Ao usar um Entity Data Model (EMD) com o Visual Studio 2010, você deve selecionar a opção **Incluir colunas de chave estrangeira na** opção **de modelo.** Por padrão, esta opção é selecionada quando você usa o assistente **Entity Data Model.**

Para tornar os dados disponíveis na camada intermediária

1. No **Solution Explorer,** clique com o projeto de servidor, RIAServicesExample.Web, selecione **Adicionar** e, em seguida, selecione **Novo Item.**

   O **Adicionar Novo Item** caixa de diálogo aparece.

2. Na lista de categorias, selecione **Dados** e selecione o modelo **ADO.NET Entity Data Model.**
3. Nomeie o novo arquivo **AdventureWorksModel.edmx** e clique em **Adicionar.**

   O **Entity Data Model Wizard.**

4. Na tela **Escolher Conteúdo Modelo,** selecione a opção **Gerar banco de dados** e clique em **Avançar.**
5. Na tela **Choose Your Data Connection,** crie uma conexão de dados para o banco de dados e clique em **Avançar.**
6. Na tela **Choose Your Database Objects,** selecione as tabelas Address, Customers e CustomerAddress**.**
7. Confirme que a opção **Incluir colunas de chave externa na** caixa **de** seleção **do modelo** está marcada por padrão e clique em **Concluir.**

   Entidade modelos são criados para as tabelas.

8. Construir (com a combinação de teclas Ctrl + Shift + B) da solução.

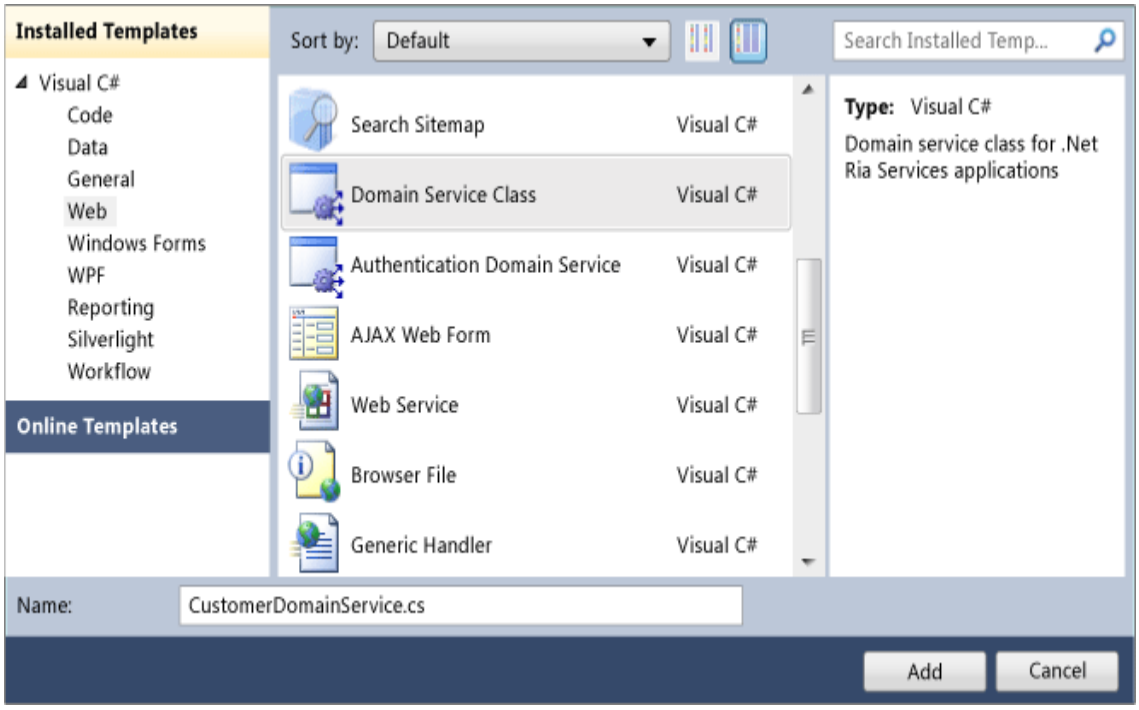
Criando um serviço de domínio

Nesta seção, você irá adicionar um serviço de domínio para o projeto de camada intermediária. Um serviço de domínio expõe as entidades de dados e operações no projeto do servidor para o projeto do cliente. Você pode adicionar lógica de negócios para o serviço de domínio para gerir a forma como o cliente interage com os dados.

Para criar o serviço de domínio

1. Botão direito do mouse o projeto de servidor, e selecione **Adicionar novo item.**
2. Na lista de categorias, selecione **Web** e selecione o Domain Services Class**.**

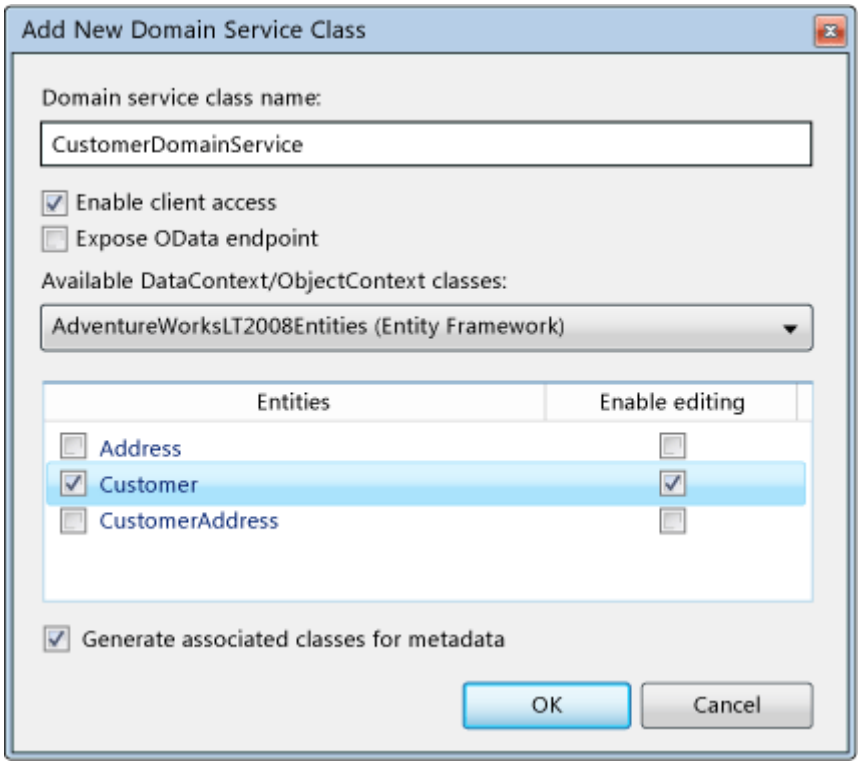
3. Nome do serviço **CustomerDomainService.cs** (ou **CustomerDomainService.vb).**



4. Clique em **Adicionar.**

   A caixa de diálogo **Add New Domain Name Service** aparece**.**

5. Verifique se a caixa **Ativar o acesso do cliente** está marcada.
6. Selecione a entidade do **cliente** e, em seguida, marque a caixa **Ativar Edição** para ele.



7. Clique em **OK.**

   O `CustomerDomainService` classe é gerada em um novo CustomerDomainService.cs.

8. Abra este arquivo. Observe que o arquivo tem as seguintes características:
   - O `CustomerDomainService` classe deriva LinqToEntitiesDomainService classe, que é uma classe abstrata de base no âmbito Serviços RIA. Essa classe de base foi utilizada automaticamente, porque o serviço de domínio expõe uma classe de dados ADO.NET Entity.
   - A classe base genérica está ligada à classe de entidade que foi criada nos passos anteriores pela `AdventureWorksLTEntities` do tipo ObjectContext no seu parâmetro genérico.
   - O `CustomerDomainService` classe é marcada com o EnableClientAccessAttribute atributo para indicar que ela é visível a camada do cliente.
   - Um método de consulta nomeada `GetCustomers` é gerado. Este método retorna todos os itens, sem qualquer filtragem ou classificação.
   - Métodos para inserir, atualizar e excluir os clientes a partir dos registros foram gerados.

Criando o cliente Silverlight

Em outro laboratório, será mostrado como adicionar lógica de negócios para o serviço de domínio. Neste laboratório, você simplesmente irá usar a `GetCustomers` método gerado por padrão.

classes de proxy do cliente são gerados quando você criou a solução. O link RIA Services, que foi estabelecida entre o projeto do cliente e o projeto de servidor gera o código necessário. Essas classes de proxy do cliente fornecer acesso aos dados do cliente.

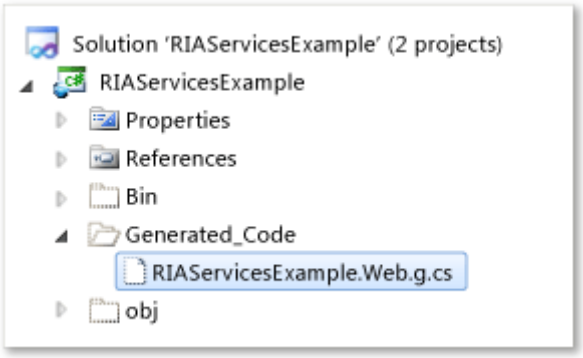
Para ver as classes de proxy gerado cliente

1. Crie a solução.

   Quando você Compila a solução, o código é gerado no projeto do cliente.

2. No **Solution Explorer,** selecione o projeto do cliente **RIAServicesExample** e clique no ícone **Show All Files** na parte superior da janela.

   Observe que a pasta Generated_Code RIAServicesExample.Web.g.cs contém um arquivo (ou RIAServicesExample.Web.g.vb).



3. Abra o arquivo de código na pasta Generated_Code.

   Observe que o arquivo tem as seguintes características:

   - A classe `WebContext` que deriva da WebContextBase classe é gerada.
   - A classe `CustomerDomainContext` que deriva da DomainContext classe é gerada. Essa classe tem um método chamado `GetCustomersQuery` que corresponde ao método de consulta criado no serviço de domínio.
   - Um classe `Customer` que deriva da entidade de classe é gerado para a entidade exposta pelo serviço de domínio. O `Customer` de classe de entidade no projeto do cliente corresponde ao `Customer` entidade no servidor.

Para exibir os dados no cliente Silverlight

1. Abra MainPage.xaml.
2. Na caixa de **ferramentas** no lado esquerdo, arraste um controle **DataGrid** para dentro do elemento **de grade** no modo XAML.

   Arrastar o controle **DataGrid** da caixa de **ferramentas** faz com que um espaço para nome `using System.Windows.Controls` que faz uma referência a um conjunto System.Windows.Controls.Data que é adicionado automaticamente.

> **⚠ Atenção:**
>
> Se você adicionar o **DataGrid** sem arrastando-o da **caixa de ferramentas,** você deve adicionar a referência ao conjunto System.Windows.Controls.Data para o projeto do cliente.

3. Altere o valor do `AutoGeneratedColums` como **True,** o nome do `DataGrid` elemento `CustomerGrid` , e ajustar a altura e a largura atributos, como mostrado na seguinte XAML.

```
 <UserControl
     xmlns: data = "clr-namespace: System.Windows.Controls; montagem =
System.Windows.Controls.Data"
     x: Class = "RIAServicesExample.MainPage"
     xmlns =
"http://schemas.microsoft.com/winfx/2006/xaml/presentation"
     xmlns: x = "http://schemas.microsoft.com/winfx/2006/xaml"
     xmlns: d = "http://schemas.microsoft.com/expression/blend/2008"
```

```
     xmlns: mc = "http://schemas.openxmlformats.org/markup-
compatibility/2006"
     MC: Ignorar = "d"
     d: DesignHeight = "300" d: DesignWidth = "400">

     <Grid X:Name="LayoutRoot" Background="White">
       <data:DataGrid Name="CustomerGrid"> </ dados: DataGrid>
     </ Grid>
 </ UserControl>
```

4. Abra o code-behind para MainPage.xaml.
5. Adicione `using` (C #) duas declarações:

`using RIAServicesExample.Web;` e

`using System.ServiceModel.DomainServices.Client;` .

O namespace **RIAServicesExample.Web** é o namespace que contém o código gerado para o projeto do cliente no RIAServicesExample.Web.g.cs.

6. Para instanciar o `CustomerDomainContext` , adicione a linha de código `private CustomerDomainContext _customerContext = new CustomerDomainContext();` no `MainPage` classe.

```
 using System;
 using System.Collections.Generic;
 using System.Linq;
 using System.Net;
 using System.Windows;
 using System.Windows.Controls;
 using System.Windows.Documents;
 using System.Windows.Input;
 using System.Windows.Media;
 using System.Windows.Media.Animation;
 using System.Windows.Shapes;
 using RIAServicesExample.Web;
 using System.ServiceModel.DomainServices.Client;

 RIAServicesExample namespace
 {
     public partial class Principal: UserControl
     {
         CustomerDomainContext _customerContext = new
CustomerDomainContext()

         Mainpage pública ()
         {
             InitializeComponent ();

             LoadOperation loadOp <Customer> =
(this._customerContext.GetCustomersQuery ())
this._customerContext.Load;
             loadOp.Entities CustomerGrid.ItemsSource =;
         }
     }
 }
```

7. Recuperar as entidades cliente chamando o método `GetCustomersQuery` com LoadOperation :

```
LoadOperation<Customer> loadOp =
this._customerContext.Load(this._customerContext.GetCustomersQue
ry());.
```

8. Vincular as entidades carregadas para o **DataGrid** com

```
CustomerGrid.ItemsSource = loadOp.Entities;.
```

Para resumir, o arquivo MainPage.xaml.cs agora deve conter o seguinte código:

```
/ / Namespaces adicionada
usando RIAServicesExample.Web;
usando System.ServiceModel.DomainServices.Client;


RIAServicesExample namespace
{
    public partial class Principal: UserControl
    {
        _customerContext CustomerDomainContext privada
CustomerDomainContext = new ();
        Mainpage pública ()
        {
            InitializeComponent ();

            LoadOperation : LoadOperation<Customer> loadOp =
this._customerContext.Load(this._customerContext.GetCustomersQue
ry());
      CustomerGrid.ItemsSource = loadOp.Entities;

        }
    }
}
```

9. Run (F5) do aplicativo.

Você deverá ver uma grade de dados que é semelhante ao seguinte.

CompanyName	CustomerID	EmailAddress	FirstName
A Bike Store	1	orlando0@adventure-works.com	Orlando
Progressive Sports	2	keith0@adventure-works.com	Keith
Advanced Bike Components	3	donna0@adventure-works.com	Donna
Modular Cycle Systems	4	janet1@adventure-works.com	Janet
Metropolitan Sports Supply	5	lucy0@adventure-works.com	Lucy
Aerobic Exercise Company	6	rosmarie0@adventure-works.com	Rosmarie
Associated Bikes	7	dominic0@adventure-works.com	Dominic
Rural Cycle Emporium	10	kathleen0@adventure-works.com	Kathleen
Sharp Bikes	11	katherine0@adventure-works.com	Katherine
Bikes and Motorbikes	12	johnny0@adventure-works.com	Johnny
Bulk Discount Store	16	christopher1@adventure-works.com	Christopher

Passo a passo: Adicionando métodos de consulta

Serviços WCF RIA

Este passo a passo descreve como adicionar e personalizar os métodos em WCF RIA Services, que consulta uma fonte de dados. Tais métodos, que são referidos como métodos de consulta, deve ser definido com uma assinatura que o framework reconhece como especificar um método de consulta. Os métodos de consulta que atende a esse requisito mediante a aplicação do QueryAttribute . O conjunto de assinaturas de consulta previstos são divididos em duas grandes categorias: as consultas que retornam sempre um único tipo de entidade e as consultas que podem, potencialmente, devolver mais de uma **entidade** do tipo T em um IEnumerable ou IQueryable .

Quando você cria um novo domínio de classe de serviço e especificar as suas entidades no **novo domínio,** o framework RIA Services cria automaticamente um método de consulta para cada entidade exposta pelo serviço. Este método simplesmente retorna todos os registros para a entidade. Nesta etapa do laboratório vou mostrar como adicionar novos métodos de consulta que executam cenários mais complexos, como a filtragem por um valor de parâmetro. Vou mostra como adicionar consultas que retornam uma única entidade e também como adicionar consultas que retornam um conjunto de entidades.

Pré-requisitos

Programas pré-requisito, como o Visual Studio 2010 e do Silverlight Developer Runtime e SDK, ser instalado e configurado corretamente, além de WCF RIA Services e WCF RIA Services Toolkit. Eles também exigem a instalação e configuração do SQL Server 2008 R2 Express com Advanced Services e

Para adicionar um método de consulta que aceita um parâmetro e retorna uma única entidade

1. Abra a solução construída anteriormente, que expõe os dados da tabela Customer.

2. No projeto de servidor, abra o `CustomerDomainService` que expõe os dados da tabela Customer.
3. Adicione um método de consulta que aceita um parâmetro inteiro e retorna o `Customer` entidade com o ID do cliente correspondente.

   Se um método que retorna uma única entidade inclui o atributo **QueryAttribute,** você deve definir o IsComposable propriedade para **false.** Os usuários não podem especificar outras operações de consulta do cliente. Se o método de consulta coincide com a assinatura do esperado para uma consulta, você não precisa aplicar o atributo **QueryAttribute.** O valor de retorno deve ser uma única instância de um objeto de entidade.

```
[Query(IsComposable = false)]
 public Customer GetCustomerByID(int CustomerID)
 {
       return this.ObjectContext.Customers.SingleOrDefault (c =>
       c.CustomerID == CustomerID);
 }
```

Para adicionar um método de consulta que aceita um parâmetro e retorna uma coleção de entidades

1. Abra a classe de serviço do domínio que expõe os dados da tabela Customer.
2. No `CustomerDomainService` classe de serviço de domínio, adicione um método de consulta que aceita um parâmetro string e retorna todos os clientes cujo último nome começa com essa letra.

   O método pode retornar um objeto **IQueryable** porque o usuário pode querer fornecer operação de consulta adicional do cliente.

```
public IQueryable<Customer> GetCustomerByLastLetter(string
startingLastNameLetter)
 {
       return this.ObjectContext.Customers.Where(c =>
       c.LastName.StartsWith(startingLastNameLetter) == true);
 }
```

Para exibir os resultados dos métodos de consulta no projeto do cliente.

1. No projeto do cliente, MainPage.xaml aberto.
2. Adicione dois controles **TextBox** e dois controles **Button** para que o usuário possa filtrar os registros de clientes, quer pela identificação ou pela primeira letra do sobrenome.

   O XAML a seguir mostra um layout completo, juntamente com o **DataGrid** existentes.

```
<Grid.ColumnDefinitions>
     <ColumnDefinition></ColumnDefinition>
     <ColumnDefinition></ColumnDefinition>
 </Grid.ColumnDefinitions>
 <Grid.RowDefinitions>
     <RowDefinition Height="25"></RowDefinition>
```

```
    <RowDefinition> </RowDefinition>
    <RowDefinition></RowDefinition>
</Grid.RowDefinitions>
<StackPanel Orientation="Horizontal" Grid.Row="0" Grid.Column="0">
    <TextBlock Text="Pesquisar por id: "></TextBlock>
    <TextBox x:Name="IDValue" Width="50"></TextBox>
      <Button Name="IDButton" Click="IDButton_Click"
      Content="Submit"></Button>
</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="0" Grid.Column="1">
    <TextBlock Text="Pesquisar por nome: "></ TextBlock>
    <TextBox Name="LetterValue" Width="30"></ TextBox>
      <Button Name="LetterButton" Click="LetterButton_Click"
      Content="Submit"></Button>
</StackPanel>

<my:DataGrid x:Name="CustomerGrid" AutoGenerateColumns="True"
Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="2"/>
```

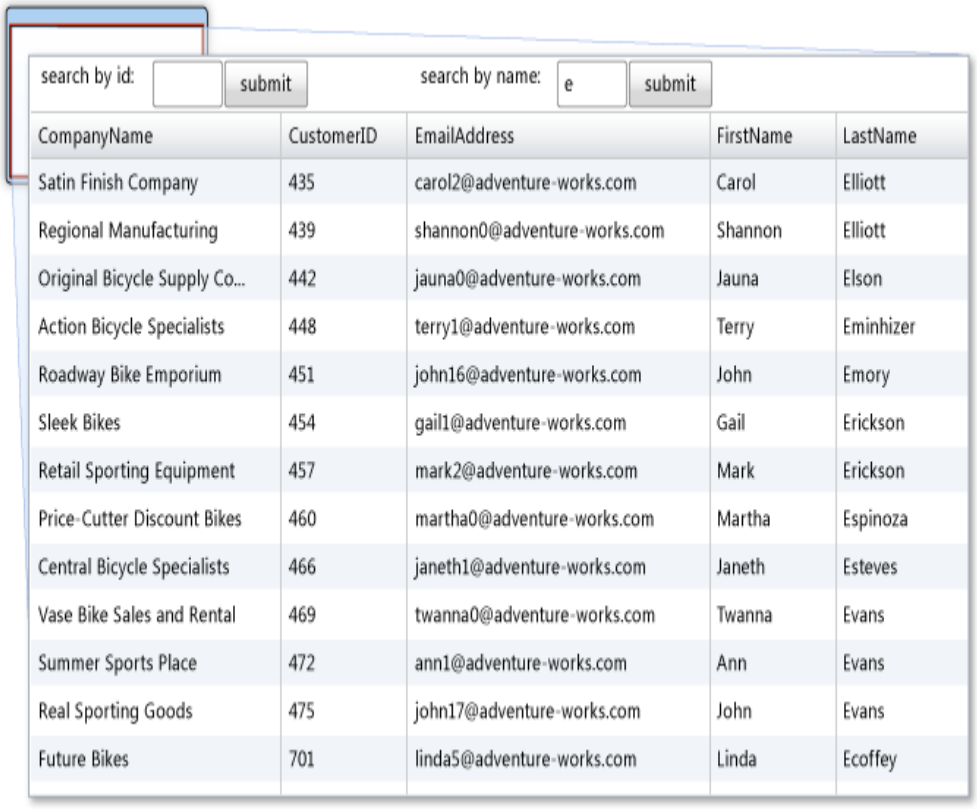
3. Abra a página code-behind para MainPage.xaml (MainPage.xaml.cs ou MainPage.xaml.vb).
4. Adicione o código para recuperar os resultados da consulta com base na entrada do usuário.

```
using System;
usando System.Windows;
usando System.Windows.Controls;
usando RIAServicesExample.Web;
usando System.ServiceModel.DomainServices.Client;

RIAServicesExample namespace
{

    public partial class Principal: UserControl
    {
      private CustomerDomainContext _customerContext = new
CustomerDomainContext();

        Mainpage pública ()
        {
            InitializeComponent ();
        }

  private void LetterButton_Click(object sender, RoutedEventArgs e)
{
    IDButton.IsEnabled = false;
    LetterButton.IsEnabled = false;
    LoadOperation<Customer> loadOp =

      this._customerContex.Load(this._customerContex.GetCustomerByLastLet
      terQuery(LetterValue.Text), CustomerLoadedCallback, null);

    CustomerGrid.ItemsSource = loadOp.Entities;
}
```

```
private void IDButton_Click(object sender, RoutedEventArgs e)
 {
     IDButton.IsEnabled = true;
     LetterButton.IsEnabled = true;
     LoadOperation<Customer> loadOp =

this._customerContex.Load(this._customerContex.GetCustomerByIDQuery(Conver
t.ToInt16(IDValue.Text)), CustomerLoadedCallback, null);
     CustomerGrid.ItemsSource = loadOp.Entities;
 }

void CustomerLoadedCallback(LoadOperation<Customer> loadOperation)
 {
     IDButton.IsEnabled = true;
     LetterButton.IsEnabled = true;
 }


     }
 }
```

5. Run (F5) do aplicativo.

A ilustração abaixo mostra uma lista de clientes filtrados pelo último nome que aparece quando o aplicativo é executado.



CompanyName	CustomerID	EmailAddress	FirstName	LastName
Satin Finish Company	435	carol2@adventure-works.com	Carol	Elliott
Regional Manufacturing	439	shannon0@adventure-works.com	Shannon	Elliott
Original Bicycle Supply Co...	442	jauna0@adventure-works.com	Jauna	Elson
Action Bicycle Specialists	448	terry1@adventure-works.com	Terry	Eminhizer
Roadway Bike Emporium	451	john16@adventure-works.com	John	Emory
Sleek Bikes	454	gail1@adventure-works.com	Gail	Erickson
Retail Sporting Equipment	457	mark2@adventure-works.com	Mark	Erickson
Price-Cutter Discount Bikes	460	martha0@adventure-works.com	Martha	Espinoza
Central Bicycle Specialists	466	janeth1@adventure-works.com	Janeth	Esteves
Vase Bike Sales and Rental	469	twanna0@adventure-works.com	Twanna	Evans
Summer Sports Place	472	ann1@adventure-works.com	Ann	Evans
Real Sporting Goods	475	john17@adventure-works.com	John	Evans
Future Bikes	701	linda5@adventure-works.com	Linda	Ecoffey

Como adicionar lógica de negócios para o Serviço de Domínio

Serviços WCF RIA

Neste tópico, você aprenderá como adicionar lógica de negócio para um serviço de domínio em um aplicativo de Serviços RIA. Um serviço de domínio RIA Services contém os métodos atualizar, inserir e excluir por padrão, mas muitas vezes você precisa adicionar lógica de negócios que modifica os dados. Você também pode precisar adicionar métodos que executam operações que não são os métodos tradicionais de consulta, inserção, atualização ou excluir. Neste tópico, você vai aprender como modificar operações de dados para atender aos requisitos de negócio.

Para adicionar lógica de negócios para os métodos de modificação de dados

1. Como você sabe os métodos padrões de inserir, atualizar ou excluir foram criados para você pelo RIA Services.
2. Nos métodos inserir, atualizar ou excluir, adicione código que especifica a lógica para o processamento do solicitado.
3. Adicione quaisquer métodos adicionais que são necessários para atender ao requisito de negócio. Marque com o atributo como IgnoreAttribute se você não quer que o método seja exposto como um serviço.

   O exemplo a seguir mostra um método de inserção que atribui ao vendedor se não for atribuído. O `RetrieveSalesPersonForCompany` método recupera o nome do vendedor de uma empresa. O método está marcado com o atributo **IgnoreAttribute** para impedir que o método seja chamado como um serviço do cliente.

```
public void InsertCustomer(Customer customer)
 {
     //Nova regra de negócio
     if (customer.SalesPerson == String.Empty )
     {
       customer.SalesPerson =
       RetrieveSalesPersonForCompany(customer.CompanyName);
     }

     if ((customer.EntityState != EntityState.Detached))
     {

       this.ObjectContext.ObjectStateManager.ChangeObjectState(customer,
       EntityState.Added);
     }
     else
     {
         this.ObjectContext.Customers.AddObject(customer);
     }
 }

 [Ignore]
 public string RetrieveSalesPersonForCompany(string companyname)
 {
     string salesPersonToAssign = "Não atribuído.";
```

```
        List<Customer> customers = GetCustomers().Where(c => c.CompanyName ==
        companyname).ToList();
        if(customers.Count > 0)
        {
            salesPersonToAssign = customers.First().SalesPerson;
        }
        return salesPersonToAssign;
    }
```