SILVERLIGHT DATA BIDING

José Antônio da Cunha
Jose.cunha@ifrn.edu.br

Data Binding

Agenda

1. O que é Data Binding?
2. Opções de Binding
3. Elementos de Binding
4. Binding modes
5. Update Modes

Data Binding

Data Binding é a maneira de puxar informações de um objeto e exibi-la na interface do usuário, sem escrita do código.

vamos aprender como usar binding para puxar valores de dados de um objeto, exibi-los, formatá-los e permitir aos usuários editá-los.

Data Binding

O Data Binding

1. Estabelece a conexão entre a UI e a camada de negócios.
2. Quando os dados são alterados, a UI é atualizada imediatamente
3. Quando a UI é atualizada pelo usuário, os dados são alterados automaticamente.

Data Binding

O Data Binding pode atuar entre:

- Propriedades de um objeto
- Elementos de um XML
- Entre propriedades de controles

Data Binding

Elementos do Binding

- Source
- Path (ou XPath)
- UpdateSourceTrigger
- Converter

Data Binding

Binding Modes

•**OneWay**: é a opção padrão usada quando se cria um Binding. Essa opção permite receber modificações de uma propriedade fonte automaticamente. Quando a propriedade fonte é modificada, a propriedade alvo automaticamente mudará, mas a propriedade fonte não se modifica quando o alvo é alterado***.***
•***TwoWay:*** a vinculação TwoWay habilita duas propriedades vinculadas a se modificarem entre si. Uma vinculação TwoWay modifica o alvo quando a fonte é modificada. Se o alvo é modificado, a fonte é atualizada.
•***OneTime***: a opção OneTime atribui a propriedade alvo à propriedade fonte quando um vinculo é inicialmente criado. Quando este BindingMode é usado, qualquer mudança na fonte de dados não será enviada automaticamente para o alvo.

Data Binding

Data Binding de objetos de dados

Na sua forma mais simples, data binding é um processo que diz ao Silverlight para extrair um valor de propriedade de um objeto de origem e usá-lo para definir uma propriedade em um objeto alvo.

Data Binding

Binding - Objeto

A melhor maneira de ver as características de binding do Silverlight é criando um objeto de dados simples. Depois, usar as expressões de binding para mostrar os dados do objeto sem escrever código.

Data object - é um pacote de informações relacionadas. Qualquer classe irá funcionar como um objeto de dados, desde que consiste em propriedades públicas. (Um objeto de dados também podem ter campos e propriedades privadas, mas não se pode extrair as informações desses membros através de expressões data binding). Além disso, se você quiser que o usuário seja capaz de modificar um objeto de dados através de data binding, suas propriedades não pode ser somente leitura.

Data Binding

Aqui está um objeto de dados simples que encapsula as informações de produto.

```
public class Product
    {
        private string modelNumber;
        public string ModelNumber {get { return modelNumber; } set { modelNumber = value; }
        }

        private string modelName;
        public string ModelName {get { return modelName; } set { modelName = value; }
        }

        private double unitCost;
        public double UnitCost {get { return unitCost; } set { unitCost = value; }
        }

        private string description;
        public string Description {get { return description; } set { description = value; }
        }

        public Product(string modelNumber, string modelName, double unitCost, string description) {
            ModelNumber = modelNumber;
            ModelName = modelName;
            UnitCost = unitCost;
            Description = description;
        }
    }
```

Data Binding

Exibindo um objeto de dados com DataContext

Considere a página simples mostrado na Figura 1. Ela mostra as informações para um único produto usando várias caixas de texto em um Grid.

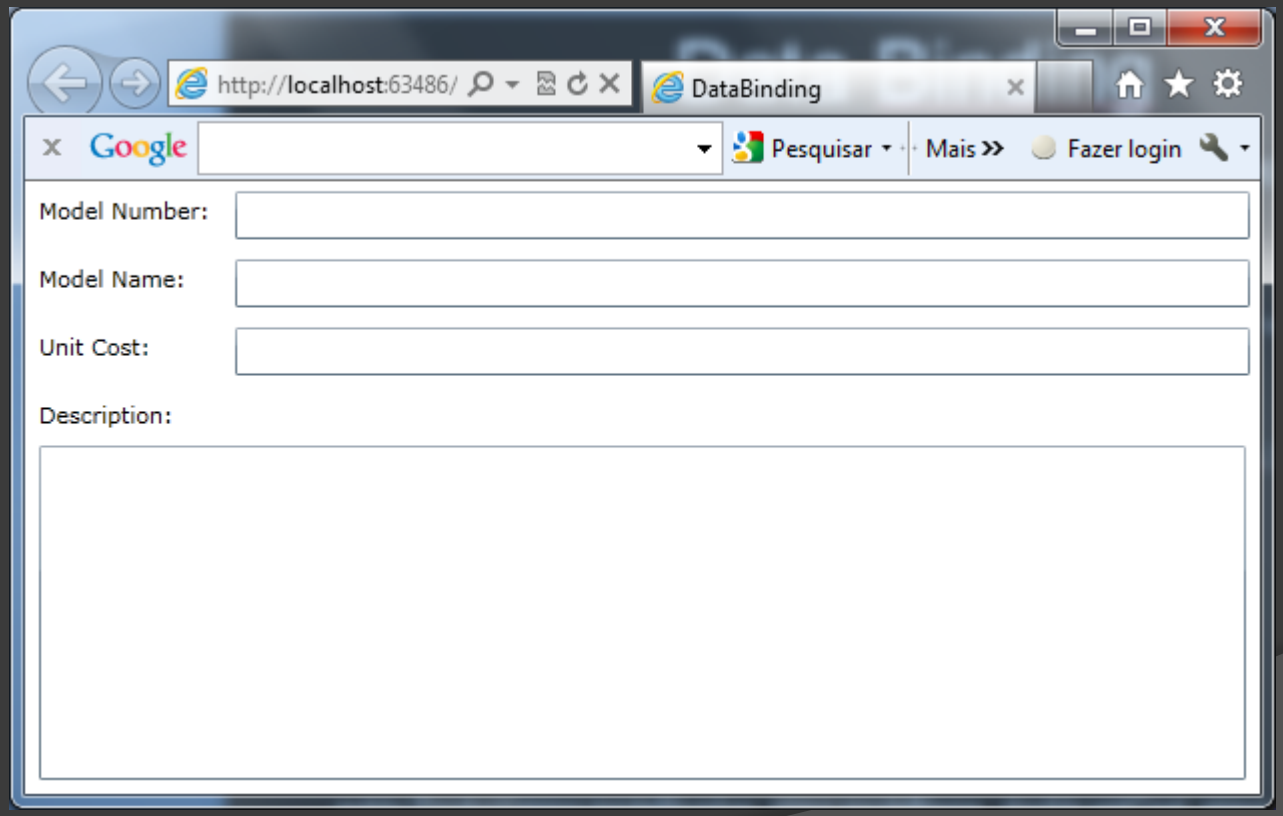

Figura 1. Mostra informações de um objeto produto.

Data Binding

A seguir, temos o código XAML para construção da página anterior

```
<Grid x:Name="gridProductDetails" Background="White">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto"></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"></RowDefinition>
            <RowDefinition Height="Auto"></RowDefinition>
            <RowDefinition Height="Auto"></RowDefinition>
            <RowDefinition Height="Auto"></RowDefinition>
            <RowDefinition Height="*"></RowDefinition>
        </Grid.RowDefinitions>
        <TextBlock Margin="7">Model Number:</TextBlock>
        <TextBox Margin="5" Grid.Column="1" Text="{Binding ModelNumber}"></TextBox>
        <TextBlock Margin="7" Grid.Row="1">Model Name:</TextBlock>
        <TextBox Margin="5" Grid.Row="1" Grid.Column="1" Text="{Binding ModelName}"></TextBox>
        <TextBlock Margin="7" Grid.Row="2">Unit Cost:</TextBlock>
        <TextBox Margin="5" Grid.Row="2" Grid.Column="1" Text="{Binding UnitCost}"></TextBox>
        <TextBlock Margin="7,7,7,0" Grid.Row="3">Description:</TextBlock>
        <TextBox Margin="7" Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="2" TextWrapping="Wrap"
                                    Text="{Binding Description}"></TextBox>
    </Grid>
```

Data Binding

Para exibir as informações de um objeto produto, você pode, obviamente, recorrer ao código C#, como o a seguir:

txtModelNumber = product.ModelNumber;

No entanto, no exemplo da UI que montamos, estamos usando o Data Binding, ou seja, estamos tirando a responsabilidade para exibir os dados do código C # e transferindo para o XAML.

Data Binding

Para isto usamos a sintaxe a seguir:

Por exemplo, para criar a propriedade Product.ModelNumber, você usa uma expressão de vinculação como este:

{Binding ModelNumber}

E aqui está como você pode usá-lo para definir a propriedade de texto em uma caixa de texto:

```
<TextBox Text="{Binding ModelNumber}"></TextBox>
```

Usando esta técnica simples, é fácil de construir a página mostrada na **Figura 1**, com as suas quatro expressões de ligação:

Data Binding

Aqui está o código que cria o objeto Produto e define a propriedade *Grid.DataContex* quando a página for carrega pela primeira vez:

```
private void UserControl_Load(object sender, RoutedEventArgs e)
        {

            Product product = new Product("AEFS100", "Portable Defibrillator", 77,
                "Analyzes the electrical activity of a person´s heart and applies " +
                "an electric shock if necessary.");
            gridProductDetails.DataContext = product;
        }
```

Data Binding

Armazenando um Objeto de Dados como um recurso

Você tem uma outra opção para especificar um objeto de dados. Você pode defini-lo como um recurso em seu XAML e, em seguida, alterar cada expressão de ligação, adicionando a propriedade de origem (Source property).

Por exemplo, você pode criar o objeto produto como um recurso usando XAML assim:

```
<UserControl.Resources>
        <local:Product x:Key="resourceProduct"
                       ModelNumber="AEFS100"
                       ModelName="Portable Defibrilator"
                       UnitCost="77"
                       Description="Analyzes the electrical activity of a person´s heart and applies
an electric shock if necessary.">
        </local:Product>
    </UserControl.Resources>
```

Data Binding

Para usar este objeto em uma expressão de binding, você precisa especificar a propriedade Source. Para definir a propriedade Source, você usa uma expressão StaticResource que usa nome da chave do recurso:

```
<TextBlock Margin="7">Model Number:</TextBlock>
    <TextBox Margin="5" Grid.Column="1" Text="{Binding ModelNumber,
        Source={StaticResource resourceProduct} }"></TextBox>
    <TextBlock Margin="7" Grid.Row="1">Model Name:</TextBlock>
    <TextBox Margin="5" Grid.Row="1" Grid.Column="1" Text="{Binding ModelName,
Source={StaticResource resourceProduct}}"></TextBox>
```

Lembre-se de adicionar a referência ao namspace a seguir:

```
xmlns:local="clr-namespace:DataBinding"
```

Data Binding

Edição com Two-Way Bindings

Neste ponto, você pode se perguntar o que acontece se o usuário altera os valores que aparecem nos controles de texto. Por exemplo, se o usuário digita uma nova descrição, o objeto produto em memória muda?

Para investigar o que acontece, vamos fazer um exemplo. Para isto vamos fazer algumas mudanças na página XAML.

Atenção! Para fazer o exemplo rodar a partir deste ponto. Retire o recurso adicionado anteriormente. **StaticResource**.

Data Binding

Modifique a página XAML, acrescente mais duas linhas:

```
<Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"></ColumnDefinition>
        <ColumnDefinition></ColumnDefinition>
 </Grid.ColumnDefinitions>
      <Grid.RowDefinitions>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="100"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
        <RowDefinition Height="Auto"></RowDefinition>
</Grid.RowDefinitions>
```

Data Binding

Modifique a página XAML, acrescente dois botões e um TextBlock:

```
<TextBlock Margin="7">Model Number:</TextBlock>
 <TextBox Margin="5" Grid.Column="1" Text="{Binding ModelNumber}"></TextBox>
 <TextBlock Margin="7" Grid.Row="1">Model Name:</TextBlock>
 <TextBox Margin="5" Grid.Row="1" Grid.Column="1" Text="{Binding
                                                          ModelName}"></TextBox>
 <TextBlock Margin="7" Grid.Row="2">Unit Cost:</TextBlock>
 <TextBox Margin="5" Grid.Row="2" Grid.Column="1" Text="{Binding UnitCost}"></TextBox>
 <TextBlock Margin="7,7,7,0" Grid.Row="3">Description:</TextBlock>
 <TextBox Margin="7" Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="2"
                        TextWrapping="Wrap" Text="{Binding Description}"></TextBox>
<Button x:Name="btnChange" Content="Change produto" Grid.Row="5" Grid.Column="0"
            Height="30" Width="120"></Button>
<Button x:Name="btnCheck" Content="Check Product" Grid.Row="5" Grid.Column="1"
            Height="30" Width="120" Click="btnCheck_Click"></Button>
<TextBlock x:Name="lblCheck" Grid.Row="6" Grid.ColumnSpan="2" Height="28"
Margin="0,0,0,-28" Text="*****"></TextBlock>
```

Data Binding

Para investigar o que acontece, vamos usar o código a seguir que cata o objeto produto atual do DataContext e exibe suas propriedades em um TextBlock:

```
private void btnCheck_Click(object sender, RoutedEventArgs e)
        {
            Product product = (Product)gridProductDetails.DataContext;

            lblCheck.Text  = "Model Name: " + product.ModelName + "\nModel Number: " +
                product.ModelNumber +  "\nUnit Cost: " + product.UnitCost;
        }
```

Rode a aplicação e veja que, se você mudar os dados nas TextBoxes o objeto Produto não muda. Porque?

Porque o modo de vinculação é o padrão → **OneWay**.

Data Binding

Vamos mudar o modo de vinculação para TwoWay e, verificar o que acontece. Faça as seguintes mudanças no XAML:

```
<TextBlock Margin="7">Model Number:</TextBlock>
        <TextBox Margin="5" Grid.Column="1" Text="{Binding ModelNumber,
Mode=TwoWay}"></TextBox>
        <TextBlock Margin="7" Grid.Row="1">Model Name:</TextBlock>
        <TextBox Margin="5" Grid.Row="1" Grid.Column="1" Text="{Binding ModelName,
Mode=TwoWay}"></TextBox>
        <TextBlock Margin="7" Grid.Row="2">Unit Cost:</TextBlock>
        <TextBox Margin="5" Grid.Row="2" Grid.Column="1" Text="{Binding UnitCost,
Mode=TwoWay}"></TextBox>
        <TextBlock Margin="7,7,7,0" Grid.Row="3">Description:</TextBlock>
        <TextBox Margin="7" Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="2"
TextWrapping="Wrap" Text="{Binding Description,
Mode=TwoWay}"></TextBox>
```

Rode a aplicação e veja que, se você mudar os dados nas TextBoxes o objeto Produto muda.

Data Binding

Em algumas situações, você precisa controlar exatamente quando a atualização é aplicada. Por exemplo, você pode precisar ter uma caixa de texto que aplica suas mudanças a medida que o usuário for digitando os dados, ao invés de esperar por uma mudança de foco. Nesta situação, você precisa fazer o trabalho manualmente chamando o método BindingExpression.UpdateSource() no código.

Aqui está o código que força a caixa de texto para atualizar os dados de origem do objeto cada vez que o usuário digita ou edita o texto:

```
private void txtModelNumber_TextChanged(object sender, TextChangedEventArgs e)
    {
        BindingExpression expression = txtModelNumber.GetBindingExpression(TextBox.TextProperty);
        expression.UpdateSource();
        Product product = (Product)gridProductDetails.DataContext;

        lblCheck.Text = "Model Name: " + product.ModelName + "\nModel Number: " +
            product.ModelNumber + "\nUnit Cost: " + product.UnitCost;
    }
```

```
<TextBox Name="txtModelNumber" Margin="5" Grid.Column="1"
        Text="{Binding ModelNumber, Mode=TwoWay, UpdateSourceTrigger=Explicit}"
            TextChanged="txtModelNumber_TextChanged"></TextBox>
```

Data Binding

ValidatesOnException

ValidatesOnException é o primeiro passo para implementar qualquer tipo de validação. Depois de definir ValidatesOnExceptions para **true. (true/false)**

Aqui está um exemplo que se aplica essa propriedade para UnitCost:

<TextBox Margin="5" Grid.Row="2" Grid.Column="1" Text="{Binding UnitCost, Mode=TwoWay, ValidatesOnExceptions=True}"></TextBox>

Esta simples mudança dá a sua aplicação a capacidade de capturar e exibir erros, desde que você esteja usando data binding two-way com um controle que suporta o grupo de controle ValidationState estados. O controles que suportam esta funcionalidade são:

✓TextBox, PasswordBox, CheckBox, RadionButton, ListBox, ComboBox

Data Binding

Para entender como isso funciona, vamos considerar o exemplo simples de uma caixa de texto com dados inválidos. Primeiro, considere uma versão da classe Produto que usa esse código para capturar os preços negativos e gerar uma exceção:

```
private double unitCost;
        public double UnitCost
        {
            get { return unitCost; }
            set
            {
                if (value < 0) throw new ArgumentException("Este valor não pode ser menor do zero.");
                unitCost = value;
            }
        }
```

Rode a aplicação e tente entrar com o valor do custo negativo.

Data Binding

NotifyOnValidationError

Depois de definir ValidatesOnExceptions para **true**, você também tem a opção de ligar NotifyOnValidationError. Se você fizer isso, o sistema data-binding dispara um evento BindingValidationError quando ocorre um erro:

```
<TextBox Margin="5" Grid.Row="2" Grid.Column="1" Text="{Binding UnitCost, Mode=TwoWay,
ValidatesOnExceptions=True,
                    NotifyOnValidationError=True}"></TextBox>
```

Altere o Grid para ficar como a seguir, ou seja, acrescente o evento BindingValidationError:

```
<Grid x:Name="gridProductDetails" Background="White"
BindingValidationError="gridProductDetails_BindingValidationError">
```

Data Binding

Navegue até o BindingValidationError e digite o seguinte código:

```
private void gridProductDetails_BindingValidationError(object sender, ValidationErrorEventArgs
e)
        {
            //Mostrar o error.
            lblInfo.Text = e.Error.Exception.Message;
            lblInfo.Text += "\nO valor armazendo é: " +
((Product)gridProductDetails.DataContext).UnitCost.ToString();

            //Solicitar outra vez
            txtUnitCost.Focus();
        }
```

Data Binding

O evento BindingValidationError só acontece quando o valor é alterado e a edição está comitada. No caso da caixa de texto, isso não acontece até que a caixa de texto perde o foco. Se você quiser que Erros sejam capturado rapidamente, você pode usar o método **BindingExpression.UpdateSource()** para forçar atualizações imediatas.

Data Binding

A classe de Valiação

Finalmente, é importante notar que você não precisa responder ao BindingValidationError para detectar dados inválidos. Você pode verificar um controle acoplado a qualquer momento usando os métodos estáticos da classe de validação. Validation.GetHasError () retorna true se o controle tem falha na validação, e retorna a coleção apropriada de exceções.

Estes métodos oferecem maior flexibilidade. Por exemplo, você pode verificar HasError () e se recusam a permitir que o usuário continue a uma nova etapa ou executar uma função específica, se existe dados inválidos.

Data Binding

Notificação de mudança

Em alguns casos, você pode querer modificar um objeto de dados depois de ter sido vinculado a um ou mais elementos. Por exemplo, considere este código, o que aumenta o preço atual em 10%:

```
Product product = (Product)griProductDetails.DataContext;
Roduct.UnitCost *= 1.1;
```

Este código não terá o efeito desejado. Embora o objeto produto na memória seja modificado, a mudança não aparece nos controles vinculados. Isso porque uma parte vital da infra-estrutura está faltando - pura e simplesmente, não há nenhuma maneira para o objeto produto, notificar os controles.

Data Binding

Para resolver este Problema, sua classe de dados precisa implementar a interface **System.ComponentModel.INotifyPropertyChanged**. A interface INotifyPropertyChanged define um único evento, que é chamado PropertyChanged. Quando ocorre alteração de propriedade em seu objeto de dados, você deve capturar o evento PropertyChanged e fornecer o nome da propriedade como uma string.

Aqui está a definição da classe Produto renovada que usa a interface INotifyPropertyChanged, e o código para a implementação do evento **PropertyChanged**:

```
public class Product: INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;
        public void OnPropertyChanged(PropertyChangedEventArgs e)
        {
            if (PropertyChanged != null)
                PropertyChanged(this, e);
        }
...
```

Data Binding

Agora, você precisa acionar o evento PropertyChanged em todos as propriedade set:

```
public double UnitCost
        {
            get { return unitCost; }
            set
            {
                if (value < 0) throw new ArgumentException("Este valor não pode ser menor do
                zero.");
                unitCost = value;

                OnPropertyChanged(new PropertyChangedEventArgs("UnitCost"));
            }
        }
```

Se você usar esta versão da classe Produto no exemplo anterior, você tem o comportamento que você espera. Quando você alterar o objeto do produto atual, as novas informações aparecem nas caixas de texto vinculadas imediatamente.

Data Binding

Construção de um Serviço de Dados (Building a Data Service)

Os exemplos anteriores, serviu para mostrar, como o Silverlight utiliza o binding com objetos. No entanto, em uma aplicação real, seu aplicativo Silverlight, necessita de uma fonte externa, para recuperar os objetos de dados, como um serviço web.

O primeiro passo é mover a definição de classe para o objeto de dados para o site ASP.NET. O objeto de dados precisa de algumas modificações: a adição dos atributos DataContract e DataMember para torná-lo serializável, e a adição de um construtor sem argumentos público que permite que ele seja serializado.

Data Binding

Aqui está uma lista parcial do código, que mostra o esquema geral que é necessário:

```
[DataContract()]
    public class Produto: INotifyPropertyChanged
    {
        public event PropertyChangedEventHandler PropertyChanged;
        public void OnPropertyChanged(PropertyChangedEventArgs e)
        {
            if (PropertyChanged != null)
                PropertyChanged(this, e);
        }
        private string modeloNum;
        [DataMember()]
        public string ModeloNum
        {
            get { return modeloNum; }
            set
            {
                modeloNum = value;
                OnPropertyChanged(new PropertyChangedEventArgs("ModeloNum"));
            }
        }

        ...

       public Produto() { }
```

Data Binding

Com o objeto de dados pronto, você precisa de um método de serviço da web que possa usá-lo. Porntanto, adicione um Silverlight-enabled WCF Service ao projeto Web. Aqui está o esquema básico:

```
[ServiceContract(Namespace = "")]
[SilverlightFaultBehavior]
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class StoreDB
{
    private string connectionString =
WebConfigurationManager.ConnectionStrings["StoreDB"].ConnectionString;

    [OperationContract()]
public Produto GetProduto(int ID)
{
        ...
}
```

Data Binding

```
<connectionStrings>
    <add name="StoreDB" connectionString=
            "Data Source=localhost;Initial Catalog=Store;Integrated Security=True"/>
  </connectionStrings>
```

Data Binding

Chamando o Serviço de Dados

Para usar o serviço, você necessita inicialmente adicionar uma web referencia ao seu projeto Silverlight. Feito isso, você está pronto para usar o código de serviço gerado automaticamente em sua aplicação web. Neste caso, é uma classe chamada StoreDBClient.

Data Binding

No manipulador de evento do botão Get Produto, insira o seguinte código para recuperar um produto e exibi-lo no contexto.

```
StoreDBService.StoreDBClient client = new StoreDBService.StoreDBClient();


private void btnGetProduto_Click(object sender, RoutedEventArgs e)
        {
           client.GetProdutoCompleted+=new
EventHandler<StoreDBService.GetProdutoCompletedEventArgs>(client_GetProdutoCompleted);
           client.GetProdutoAsync(Convert.ToInt32(txtProdutoID.Text));
        }

        private void client_GetProdutoCompleted(object sender, GetProdutoCompletedEventArgs e)
        {
            try
            {
                gridProdutoDetalhes.DataContext = e.Result;
            }
            catch (Exception err)
            {
                throw new Exception("Erro: " + err.Message);
            }
        }
```

Data Binding

Lingando uma coleção de objetos (Binding to a Collection of Object)