SilverLight 4 - Usando o DataForm

Neste laboratório eu vou mostrar um pouco sobre o **DataForm**, um controle do **SilverLight** 4 que pode ser usado para tratar grande quantidade de informações.

O **DataForm** é um controle do **SilverLight ToolKit** e é essencialmente o que o nome sugere: **um formulário.**

Na verdade o controle é um formulário poderoso que fornece um suporte embutido para a geração de campos com gerenciamento de conteúdo com:

- persistência ou cancelamento de alterações , feitas com o suporte da interface **IEditableObject**;
- validação a nível de campo e entidade;
- navegação e paginação automática;

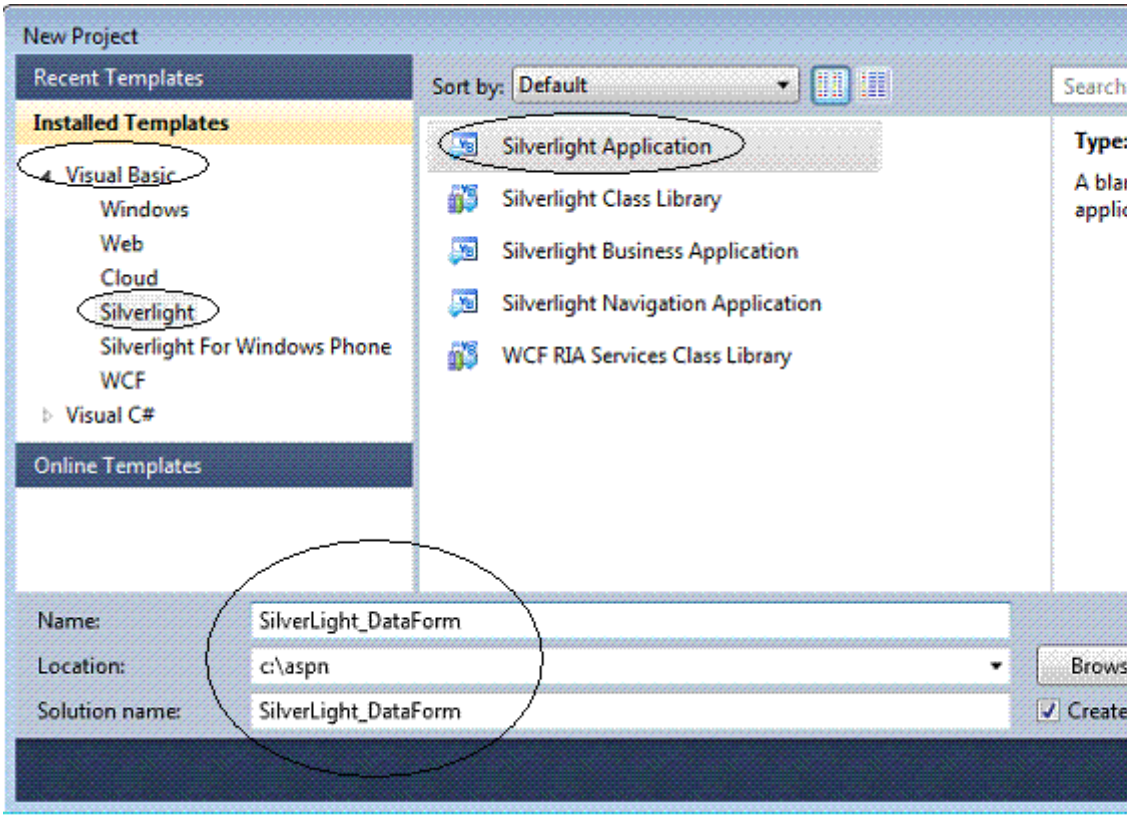
Além disso o controle á altamente customizável o que o torna um companheiro perfeito para o desenvolvimento rápido de aplicações com formulários.

Para poder usar este controle você deve instalar o : **Silverlight 4 Toolkit**

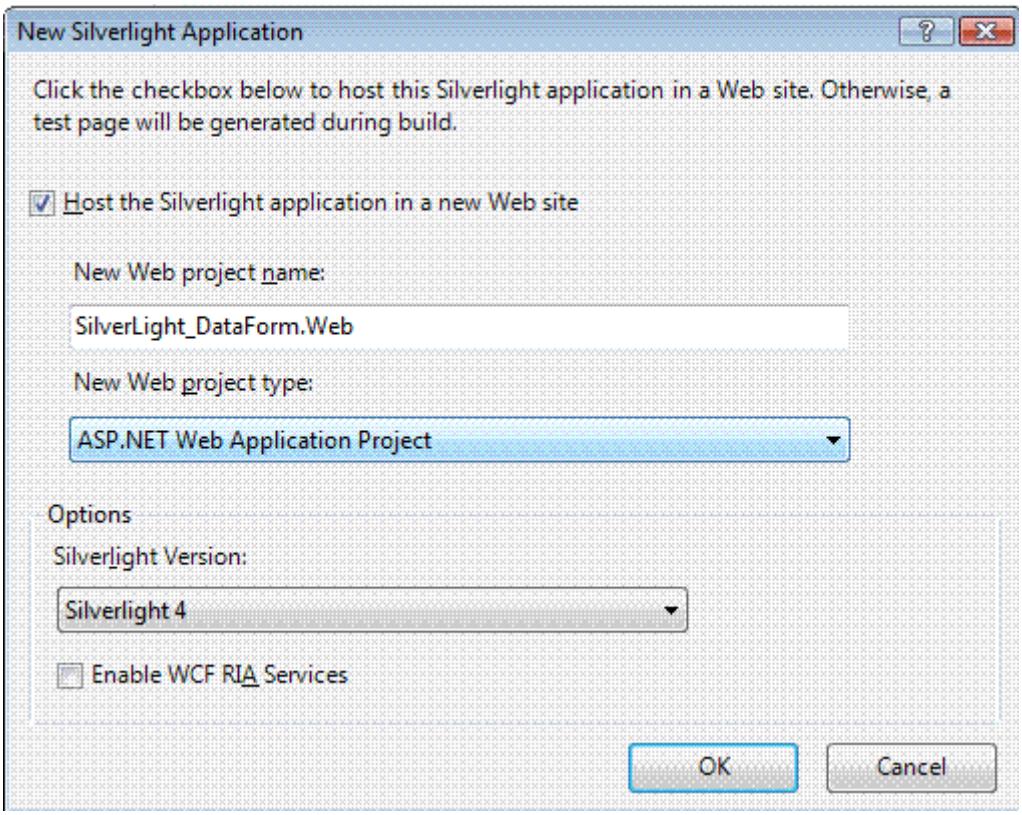
Abra o Visual Web Developer 2010 Express Edition e crie um novo projeto com o nome **SilverLight_DataForm** usando a linguagem C#/Visual Basic;

No menu **File-> New Project** selecione **Visual Basic -> SilverLight** e a seguir o template **SilverLight Application** informando o nome **SilverLight_DataForm**;

Clique em **OK**;

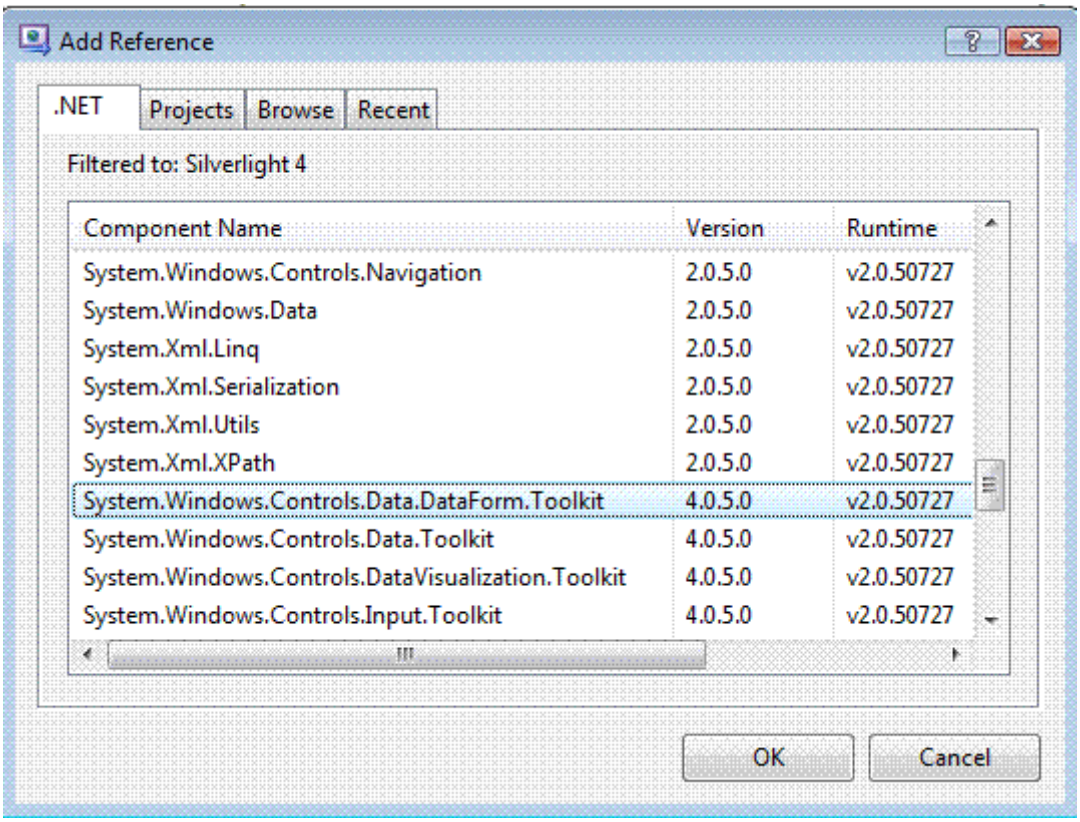


Na seguinte janela de diálogo apenas confirme as opções e clique em OK;



Agora vamos incluir uma referência em nosso projeto SilverLight a **System.Windows.Controls.Data.DataForm.Toolkit** via menu **Project -> Add**

Reference e na janela Add Reference selecionando a guia .NET e o item conforme a figura abaixo:



A seguir selecione o arquivo **MainPage.xaml** e inclua a linha de código abaixo na tag **<UserControl>** referente ao namespace que vai permitir que usemos o **DataForm** que reside o assembly que foi referenciado:

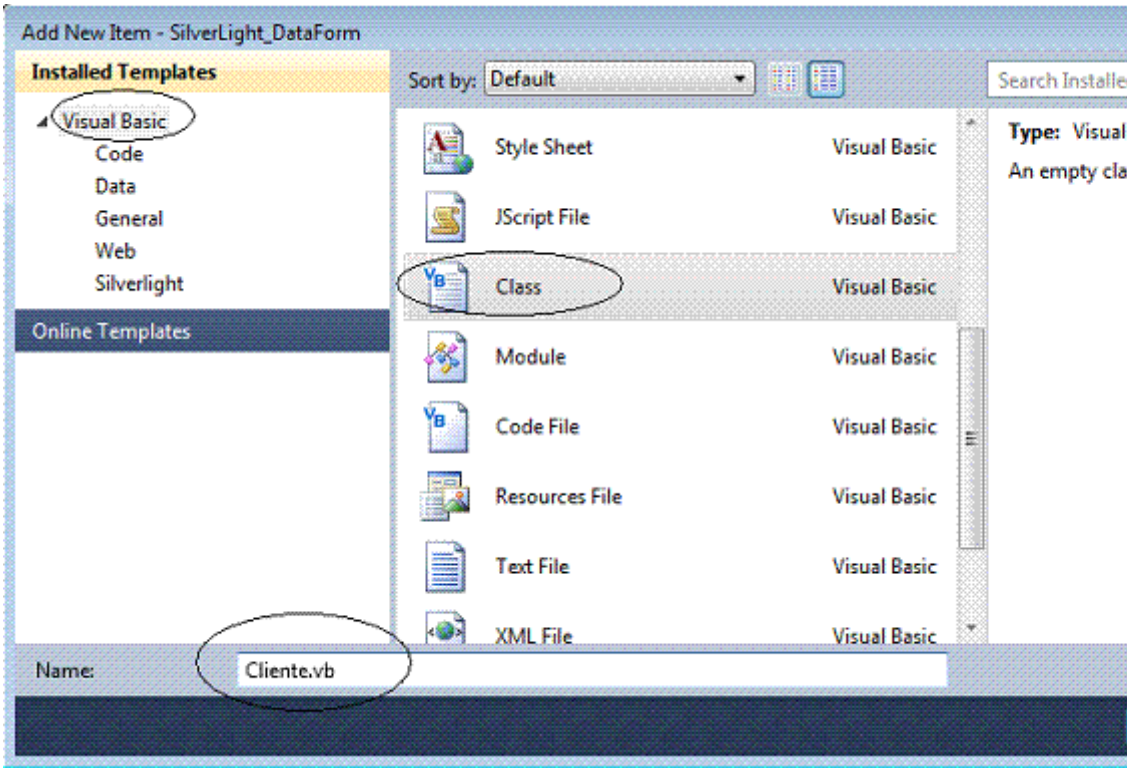
```
xmlns:df="clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data.DataForm.Toolkit"
```

Agora adicione a página um controle **DataForm** a partir da **ToolBox** dando a ele o nome de **macDataForm** definindo a propriedade **AutoEdit** para **False** e a propriedade **CommandButtonsVisibility** para **All** conforme o código a seguir:


```
   <Grid x:Name="LayoutRoot" Background="White">
      <Grid.RowDefinitions>
         <RowDefinition Height="40" ></RowDefinition>
         <RowDefinition></RowDefinition>
      </Grid.RowDefinitions>
      <TextBlock Text="Trabalhando com o DataForm" Margin="10"
FontSize="14" >
      </TextBlock>
      <df:DataForm x:Name="macDataForm"
         AutoEdit="False"
         CommandButtonsVisibility="All"
         Grid.Row="1"
         Width="480"
         Height="320"
         Margin="10"
         HorizontalAlignment="Left"
         VerticalAlignment="Top" >
      </df:DataForm>
   </Grid>
```

Agora vamos definir a nossa fonte de dados que será a classe **Cliente** em nosso projeto SilverLight;

No menu **Project -> Add Class** selecione o template **Class** e informe o nome **Cliente.vb** e clique em **Add**;



Em seguida defina o seguinte código na classe **Cliente**:

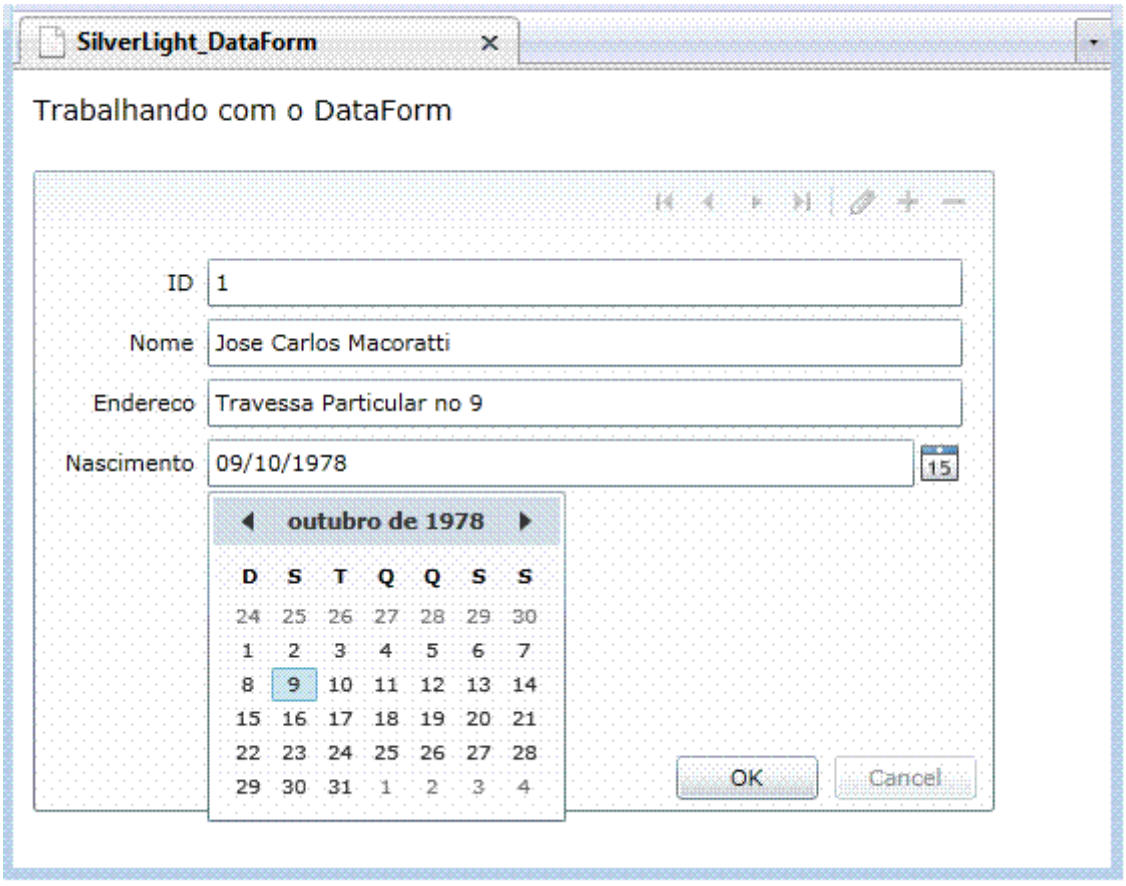
```
public class Cliente: IEditableObject
    {
        public int ID { get; set; }
        public String Nome { get; set; }
        public String Endereco { get; set; }
        public DateTime Nascimento { get; set; }
}
```

Agora na página **MainPage.xaml.cs** vamos iniciar um **cliente** conforme o código abaixo:

```
  Cliente c = new Cliente();
    c.ID = 1;
    c.Nome = "José Antônio da Cunha";
    c.Endereco = "Rua Coronel Dantas, 123";
    c.Nascimento = Convert.ToDateTime( "10/10/1960");

    macDataForm.CurrentItem = c;
```

Rode o projeto pressionando F5 para obter o resultado mostrado na figura abaixo:



Mas o que fizemos para obter esse resultado ??? 

Para começar, precisávamos de algo para mostrar no nosso **DataForm**: a entidade **Cliente**. É por isso que criamos a classe **Cliente** que foi vinculada ao DataForm, definindo a propriedade **CurrentItem** a um objeto do tipo **Cliente**.

Dessa forma garantimos que o **DataForm** irá gerar automaticamente os campos necessários. Ela *'olha'* para todas as propriedades públicas do nosso objeto **Cliente** e gera o controle correto, dependendo do tipo. Uma sequência de caracteres será exibido como um **TextBox**, um valor booleano será exibido como um **CheckBox**, e assim por diante...

Quando definimos a propriedade **CommandButtonsVisibility** do DataForm para **All**, temos um ícone **Editar** na barra de comandos na parte superior do DataForm. (*Definindo AutoEdit para False garante que começamos no modo de exibição, ao invés do modo de edição).*

Quando você clica no ícone **Editar**, o **DataForm** mostra a pessoa no modo editável (usando o **EditItemTemplate**) e um botão OK aparece. Clicando no botão **OK** irá reverter o formulário para o modo de exibição regular. Tenha em mente que as mudanças que você faz para a pessoa imediatamente são persistentes na memória.

Se for preciso você pode escrever código extra para manter o objeto **Cliente** a partir da memória persistindo os dados subjacentes ao manipular o evento **ItemEditEnded** na DataForm.

Até este momento temos um DataForm exibindo um único item que você pode ver ou editar. Mas e se você quiser cancelar a edição?

O botão **Cancel** parece que está desativado e como as alterações feita no DataForm são persistidos para o objeto na memória o cancelamento vai exigir um trabalho extra mas nada muito complicado.

A primeira coisa que vamos ter que fazer é implementar a interface **IEditableObject** na classe Cliente que irá certificar de que o cancelamento é possível e como resultado o botão Cancelar não será desativado. Para tal defina o código a seguir na classe Cliente:

```
public class Cliente: IEditableObject
{
   public int ID { get; set; }
   public String Nome { get; set; }
   public String Endereco { get; set; }
   public DateTime Nascimento { get; set; }
}
```

Observe que estou usando os namespaces **System.ComponentModel** e que estou implementando a interface **IEditableObject**.

Esta interface expõe 3 métodos: **BeginEdit, CancelEdit e EndEdit;**

O que vamos fazer agora é implementar a lógica de negócio extra nestes métodos e como não estamos usando um banco de dados vamos usar um implementação no método **BeginEdit** onde salvamos os valores atuais do cliente e se a edição for cancelada nós retornamos os valores anteriores à edição. Confira a implementação no código abaixo:

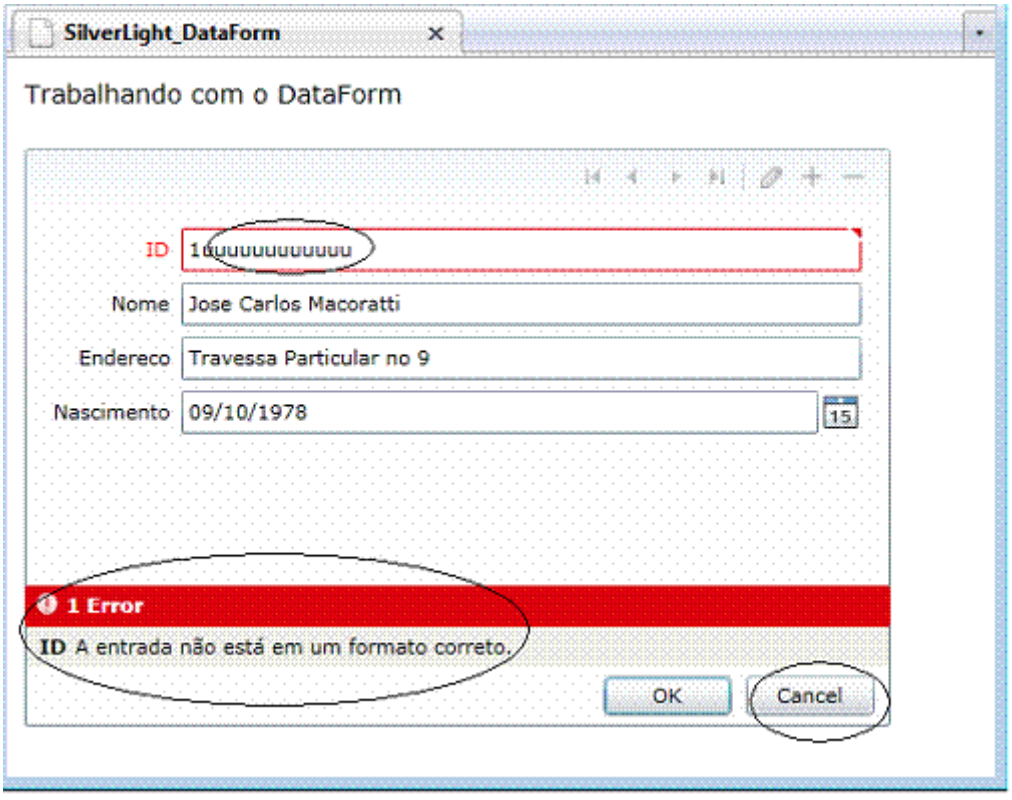
```
private Cliente tmpCliente;

public void BeginEdit()
{
    tmpCliente = new Cliente
     {
        ID = this.ID,
        Nome = this.Nome,
        Endereco = this.Endereco,
        Nascimento = this.Nascimento
     };
}
```

```
public void CancelEdit()
{
    ID = tmpCliente.ID;
    Nome = tmpCliente.Nome;
    Endereco = tmpCliente.Endereco;
    Nascimento =
tmpCliente.Nascimento;
}
public void EndEdit()
{
}
```

Após esta implementação a edição e o cancelamento estão habilitados.

Executando o projeto agora já podemos notar que o botão **Cancel** esta ativo e que se digitarmos um valor inválido a validação é feita e uma mensagem de erro é exibida. *(este é outro recurso do DataForm)*



Existem muitos mais recursos no **DataForm** do que expomos neste laboratório.

Referências:

- http://msdn.microsoft.com/en-us/library/ms668604.aspx
- SilverLight 3.0 - Introdução - Macoratti.net

- .NET - Apresentando o Silverlight e o Expression Blend
- SilverLight 3.0 - Apresentando o DataBinding
- .NET RIA Services: From Vision to Architecture
- SilverLight 3.0 - Introdução - Macoratti.net
- Apresentando o SilverLight - Macoratti.net
- Curso SilverLight 4
  - http://channel9.msdn.com/learn/courses/Silverlight4/