Laboratório 3 WCF RIA Services

Continuando com nosso estudo sobre o WCF RIA Services vamos desenvolver mais um laboratório para aprendermos mais alguns recursos do Silverlight.

1) Crie uma aplicação Silverlight Business Application.
2) Adicione um novo item Ado Entity Framework e conecte-se ao banco de dados AdventureWorks.
3) Selecione a tabela Employees.
4) Compile na aplicação servidora (Web).
5) Adicione um novo item Domain Service Class, dê o nome de AdvServiceModel.
6) Selecione a entidade Employees e marque-a para ser editável.
7) Veja que foram criados os métodos do crud para você. Altere o método GetEmployee, de forma que fique como mostrado a seguir:

```
public IQueryable<Employee> GetEmployees()
 {
    return this.ObjectContext.Employees.OrderBy(e=>e.NationalIDNumber);
 }
```

8) Adicione a esta classe mais um método, como mostrado a seguir

```
public IQueryable<Employee> getSalariesEmployee()
{
  return this.ObjectContext.Employees.Where(e => e.SalariedFlag == true);
}
```

9) Compile novamente para fazer as atualizações.
10) Compile também a aplicação silverlight para criar a classe de metadados no cliente.
11) Expanda todos os arquivos e der uma olhada nesta classe.
12) Adicione mais um link as opções de menu e, dê o nome de “EmployeeList”, ou seja, abra o XAML da MainPage e acrescente o seguinte código

```
<Rectangle x:Name="Divider1" Style="{StaticResource DividerStyle}"/>

<HyperlinkButton x:Name="Link3" Style="{StaticResource LinkStyle}"
        NavigateUri="/EmployeesListing" TargetName="ContentFrame"
Content="{Binding Path=ApplicationStrings.EmployeePageTitle,
Source={StaticResource ResourceWrapper}}"/>
```

13) Adicione mais uma página Silverlight e nomei como está na Tag NavigateUri, ou seja, como EmployeesListing.

14) No arquivo de recurso da aplicação, na pasta **Assets**, crie a string **EmployeePageTitle e coloque como conteúdo Employees.**

15) No Grid da nova página adicione um componente ScrollView e configure-o como mostrado a seguir

```
<Grid x:Name="LayoutRoot">


    <ScrollViewer BorderThickness="0" VerticalScrollBarVisibility="Auto"
    Margin="-  12" Padding="12,0,12,0">


   </ScrollViewer>
</Grid>
```

16) Adicione dentro do Scrollview o StackPanel e configure-o como mostrado a seguir e adicione ao StackPanel um textBlock.

```
<StackPanel Orientation="Vertical" Margin="0,12,0,12">

    <TextBlock Text="Employee List" Style="{StaticResource HeaderTextStyle}"/>

</StackPanel>
```

17) Rode a aplicação e navegue para a página Employees.
18) Adicione um controle DataGrid abaixo do TextBlock e configure-o como tal

```
<my:DataGrid x:Name="EmployeeGrid" AutoGenerateColumns="True" />
```

Agora vamos popuplar o Grid com os dados dos Employees. Primeiramente vamos usar o cod-Behind e posteriormente vamos fazer a mesma coisa usando apenas as Tags XAML.

19) Clique com o botão direito do mouse em qualquer parte da janela XAML e selecione a opção do menu View Code.
20) Altere o código da página para ficar como a seguir

```
public EmployeesListing()
{
     InitializeComponent();
     Loaded +=new RoutedEventHandler(EmployeesListing_Loaded);
}

void EmployeesListing_Loaded(object sender, RoutedEventArgs e)
{
    AdvDomainContext ctx = new AdvDomainContext();
    EmployeeGrid.ItemsSource = ctx.Employees;
    ctx.Load(ctx.GetEmployeesQuery());
}
```

21) Não esqueça de adicionar a biblioteca
```
     using BusinessApplication2.Web.Services;
```
   pois o contexto pertence a ela.

22) Rode a aplicação para fazer o resultado.
23) O que, agora vamos utilizar apenas o XAML. Apague o código que você criou acima ou apenas comente-os.
24) Adicione logo acima do Scrollview um controle **DomainDataSource** e configure-o como a seguir

```
<my1:DomainDataSource x:Name="MyData" LoadSize="20" QueryName="GetEmployeesQuery"
AutoLoad="True" >
    <my1:DomainDataSource.DomainContext>
        <ds:AdvDomainContext />
    </my1:DomainDataSource.DomainContext>
</my1:DomainDataSource>
```

25) Adicione o seguinte namespace ao XAML

```
xmlns:ds="clr-namespace:BusinessApplication2.Web.Services"
```

26) Altere o DataGrid para ficar como tal

```
<my:DataGrid x:Name="EmployeeGrid" AutoGenerateColumns="True" MinHeight="100"
IsReadOnly="True" ItemsSource="{Binding ElementName=MyData, Path=Data}"/>
```

27) Rode a aplicação e veja o resultado.
28) Agora vamos fazer filtros e Classificação usando o DomianDataSource. Para tal, acrescente um novo StackPanel logo acima do DataGrid e, dentro do mesmo acescente um TextBlock e um TextBox, veja o código a segui

```
<StackPanel Orientation="Horizontal" HorizontalAlignment="Right" Margin="0,-
16,0,0">
    <TextBlock VerticalAlignment="Center" Text="Min Vacation Hours Filter: " />
    <TextBox x:Name="VacationHoursFilter" Width="75" FontSize="11" Margin="4" />
</StackPanel>
```

29) Acrescente o seguinte o código a tag DomainDataSource, veja o código a seguir. Este código permite realizar filtros.

```
<my1:DomainDataSource x:Name="MyData" LoadSize="20" QueryName="GetEmployeesQuery"
AutoLoad="True" >
   <my1:DomainDataSource.DomainContext>
      <ds:AdvDomainContext />
   </my1:DomainDataSource.DomainContext>
   <my1:DomainDataSource.FilterDescriptors>
       <my1:FilterDescriptor PropertyPath="VacationHours"
                  Operator="IsGreaterThanOrEqualTo"
                  Value="{Binding ElementName=VacationHoursFilter, Path=Text}"/>
       </my1:DomainDataSource.SortDescriptors>
   </my1:DomainDataSource>
```

30) Vamos fazer algo parecido com o código acima para realizar Classificação(Sorting).

```
<my1:DomainDataSource x:Name="MyData" LoadSize="20" QueryName="GetEmployees"
AutoLoad="True" >
    <my1:DomainDataSource.DomainContext>
       <ds:AdvDomainContext />
    </my1:DomainDataSource.DomainContext>
    <my1:DomainDataSource.FilterDescriptors>
       <my1:FilterDescriptor PropertyPath="VacationHours"
                Operator="IsGreaterThanOrEqualTo"
                Value="{Binding ElementName=VacationHoursFilter, Path=Text}"/>
    </my1:DomainDataSource.FilterDescriptors>
    <my1:DomainDataSource.SortDescriptors>
       <my1:SortDescriptor PropertyPath="BirthDate" Direction="Ascending" />
    </my1:DomainDataSource.SortDescriptors>
</my1:DomainDataSource>
```

31) Vamos agora adiconar o novo controle para fazer paginação. Então adicione logo abaixo do DataGrid um DataPager e configure-o como a seguir

```
<my:DataPager x:Name="DataPager" PageSize="5" Margin="0,-1,0,0" Source="{Binding
ElementName=MyData, Path=Data}"/>
```

32) Rode a aplicação para fazer alguns teste.

Vamos agora adicionar mais um controle e tentarmos fazer edição nos dados através do mesmo.

33) Adicione um controle DataForm logo acima do controle DataPager e configure-o de acordo com o código a seguir.

```
<toolkit:DataForm x:Name="EmployeeDetails" Header="Employee Information"
AutoGenerateFields="False" AutoCommit="True"  AutoEdit="False"
CurrentItem="{Binding ElementName=EmployeeGrid, Path=SelectedItem}">
<toolkit:DataForm.EditTemplate>
   <DataTemplate>
      <StackPanel>
        <toolkit:DataField Label="Employee ID">
        <TextBox Text="{Binding EmployeeID, Mode=TwoWay}"/>
     </toolkit:DataField>
         <toolkit:DataField Label="Login ID">
         <TextBox Text="{Binding LoginID, Mode=TwoWay}"/>
     </toolkit:DataField>
     <toolkit:DataField Label="Hire Date">
           <TextBox Text="{Binding HireDate, Mode=TwoWay}"/>
     </toolkit:DataField>
     <toolkit:DataField Label="Marital Status">
         <TextBox Text="{Binding MaritalStatus, Mode=TwoWay}"/>
     </toolkit:DataField>
     <toolkit:DataField Label="Gender">
         <TextBox Text="{Binding Gender, Mode=TwoWay}"/>
     </toolkit:DataField>
     <toolkit:DataField Label="VacationHours">
       <TextBox Text="{Binding EmployeeID, Mode=TwoWay,
       NotifyOnValidationError=True, ValidatesOnExceptions=True}"/>
     </toolkit:DataField>
   </StackPanel>
 </DataTemplate>
</toolkit:DataForm.EditTemplate>
</toolkit:DataForm>
```

34) Rode a aplicação e faça alguns teste. Veja que a edição é possível mais os dados ainda não são persistidos. Observe que alguns campos são obrigatórios

35) Adicione um StackPanel abaixo do DataForm e configure-o com tal e, dentro do StackPanel adicione um Button e configure de acordo com o código a seguir.

```
<StackPanel Orientation="Horizontal" HorizontalAlignment="Right"
Margin="0,12,0,0">
   <Button Content="Save changes" x:Name="SaveChangesButton" Width="75"
   Height="23" Margin="4,0,0,0" Click="SaveChangesButton_Click"/>
</StackPanel>
```

36) Navegue para o código do evento click do botão e digite o seguinte código.

```
private void SaveChangesButton_Click(object sender, RoutedEventArgs e)
{
     MyData.SubmitChanges();
}
```

37) Rode a aplicação e faça alguns teste. Verifique que agora as mudanças são persistidas.
38) Faça as seguintes mudanças no metadados para formatar a entrada de dados.

```
[RegularExpression("M|m|F|f")]
public string Gender { get; set; }

[Range(0,80, ErrorMessage="Sua mensagem")]
public short VacationHours { get; set; }
```

39) Compile a aplicação, e rode e faça alguns testes.
40) Adicione um novo StackPanel e dentro deste um botão, conforme o código a seguir

```
<StackPanel Orientation="Horizontal" HorizontalAlignment="Right"
Margin="12,0,0,0">
       <Button x:Name="NewEmpButton" Width="90" Height="23" Content="New
       Employee" Margin="4,0,0,0" Click="NewEmpButton_Click"></Button>
</StackPanel>
```