Optimizando el filtrado de datos en el control DomainDataSource

Luis Miguel Blanco Ancos

En nuestro anterior artículo [1] sobre el control DomainDataSource nos dedicamos a realizar un repaso general de este control de datos, presentando sus principales características. En esta ocasión, por el contrario, nos detendremos en una de sus funcionalidades en particular, el filtrado, profundizando un poco más en el mismo, con la intención de mejorar la experiencia de usuario.

Podemos considerar este artículo como un complemento del anteriormente mencionado sobre este control de datos, por lo que utilizaremos el mismo proyecto de ejemplo (disponible en http://www.dotnetmania.com). Para más información, sugerimos al lector la revisión de dicho artículo.

Ya sabemos que es posible enlazar un filtro declarado en el DomainDataSource con un elemento de la interfaz de usuario mediante una expresión de enlace a datos (data binding). El ejemplo más típico, como ya se explicó, es un TextBox, donde el usuario escribe el valor del filtro, y éste es aplicado dinámicamente.

Pero en algunos escenarios, los valores para el filtro forman parte de una lista, y debemos asegurarnos de que el usuario introduce solamente uno de tales valores a la hora de filtrar, por lo que la utilización de un control como el TextBox no ofrece muchas garantías de éxito.

ComboBox al rescate

Ya que parece más conveniente buscar otro tipo de control para implementar esta funcionalidad, en esta ocasión nuestro candidato será el control ComboBox, dado su mejor orientación al manejo de listas de valores; para lo que agregaremos al proyecto una nueva página con el nombre FiltrosCombo.xaml.

Supongamos que queremos filtrar los resultados del DataGrid por el campo BillingCountry, usando un ComboBox para almacenar la lista de países. La manera más simple de hacer esto sería incluir tales valores como objetos ComboBoxItem dentro de la declaración del control, estableciendo un enlace a datos entre el FilterDescriptor del DomainDataSource y el ComboBox (listado 1). De esta forma, cada nueva selección en el ComboBox filtrará los datos del DomainDataSource en función del valor seleccionado en la lista desplegable, refrescando así las filas del DataGrid (figura 1).

```
<riaControls:DomainDataSource x:Name="ddsInvoices" QueryName="GetInvoices"
AutoLoad="True">
   <!--.....-->
   <riaControls:DomainDataSource.FilterDescriptors>
       <riaControls:FilterDescriptor PropertyPath="BillingCountry"
         Operator="IsEqualTo"
         Value="{Binding ElementName=cboCountries, Path=SelectedItem.Content}" />
   </riaControls:DomainDataSource.FilterDescriptors>
</riaControls:DomainDataSource>
<!--.....-->
<ComboBox x:Name="cboCountries" Margin="5" Width="150">
    <ComboBoxItem Content="Argentina" />
    <ComboBoxItem Content="Australia" />
    <ComboBoxItem Content="Austria" />
    <ComboBoxItem Content="Belgium" />
    <ComboBoxItem Content="Brazil" />
    <ComboBoxItem Content="Canada" />
    <!--.....-->
</ComboBox>
```

Listado 1.

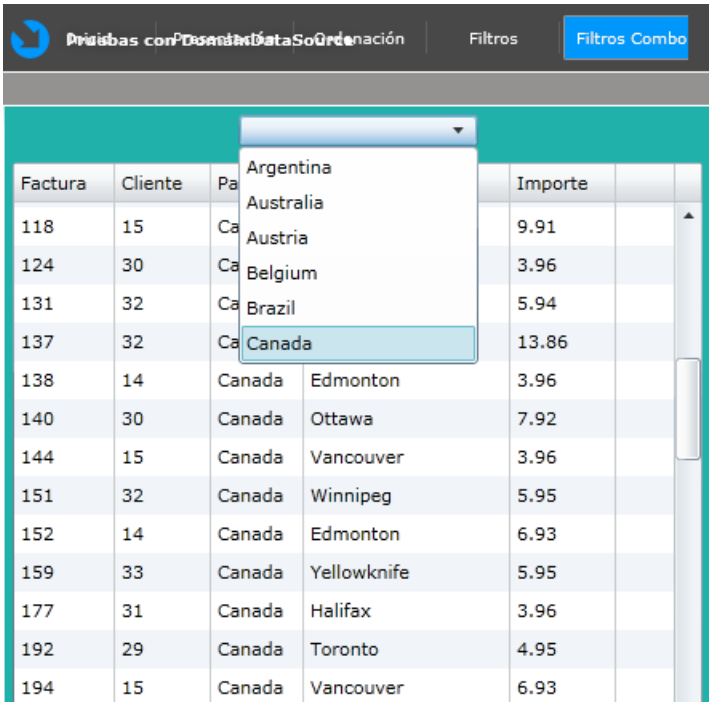

Figura 1. Filtrado mediante un ComboBox.

Consideramos no obstante, que el lector estará de acuerdo en que este procedimiento para el llenado del ComboBox no resulta todo lo óptimo que debiera, ya que supongamos que los valores de la lista tienen su origen en una tabla de la base de datos, la cual puede sufrir modificaciones. Por este motivo, la mejor forma de

mantener actualizados los datos del ComboBox, consiste en cargarlos desde dicha tabla. El listado 2 muestra el script de SQL necesario para crear la tabla de países y llenarla de valores.

```
USE Chinook

CREATE TABLE Country (
CountryId int IDENTITY(1,1) NOT NULL,
CountryName varchar(50) NOT NULL,
PRIMARY KEY CLUSTERED (CountryId) )

INSERT INTO Country
SELECT DISTINCT BillingCountry FROM Invoice ORDER BY BillingCountry
```

Listado 2.

Nuestro siguiente paso consistirá en añadir una nueva entidad al modelo de datos de la aplicación, que represente a la tabla Country; para lo que nos situaremos en el diseñador del modelo de entidades, y haciendo clic derecho sobre el mismo seleccionaremos la opción *Update Model from Database*, que iniciará el asistente mediante el que añadiremos esta entidad.

A continuación escribiremos en el servicio de dominio un método que devuelva la colección de entidades Country (listado 3).

```
public class ChinookDomainService : LinqToEntitiesDomainService<ChinookEntities>
{
    public IQueryable<Country> GetCountries()
    {
        return this.ObjectContext.Countries;
    }
    //....
```

Listado 3.

Seguidamente volveremos a la página FiltrosCombo.xaml, donde añadiremos un nuevo DomainDataSource, que obtenga el contenido de la tabla Country, mediante la llamada al método GetCountries recién creado en el servicio de dominio. Asignando una expresión de enlace a datos en la propiedad ItemsSource del ComboBox, éste se rellenará con la colección de entidades Country. Para que el nombre del país sea mostrado en la lista desplegable, en la propiedad DisplayMemberPath asignaremos el valor CountryName.

Por último, en el enlace a datos del FilterDescriptor, cambiaremos el atributo Path para que la propiedad CountryName del elemento seleccionado en el ComboBox sea la utilizada como valor de filtro (listado 4).

```
<riaControls:DomainDataSource x:Name="ddsInvoices" QueryName="GetInvoices"
AutoLoad="True">
    <!--.....-->
    <riaControls:DomainDataSource.FilterDescriptors>
        <riaControls:FilterDescriptor PropertyPath="BillingCountry"
            Operator="IsEqualTo"
            Value="{Binding ElementName=cboCountries,
Path=SelectedValue.CountryName}" />
    </riaControls:DomainDataSource.FilterDescriptors>
</riaControls:DomainDataSource>

<riaControls:DomainDataSource x:Name="ddsCountries" QueryName="GetCountries"
AutoLoad="True">
    <riaControls:DomainDataSource.DomainContext>
        <domainctx:ChinookDomainContext />
    </riaControls:DomainDataSource.DomainContext>
</riaControls:DomainDataSource>
<!--.....-->
<ComboBox x:Name="cboCountries"
    ItemsSource="{Binding ElementName=ddsCountries,Path=Data}"
    DisplayMemberPath="CountryName"
    Margin="5" Width="150" />

<sdk:DataGrid x:Name="grdInvoices"
    ItemsSource="{Binding ElementName=ddsInvoices, Path=Data}"
    AutoGenerateColumns="False" Margin="5" Height="350">
<!--.....-->
```

Listado 4.

Filtrando mediante controles RadioButton

Planteemos ahora una nueva situación sobre la técnica de filtrado que acabamos de explicar: en lugar de proporcionar la oportunidad de filtrar por todos los países disponibles, supongamos que solamente tenemos que filtrar por un conjunto reducido de los mismos. Aunque este comportamiento también podríamos lograrlo empleando un ComboBox, vamos a aportar una solución basada en otro control distinto: RadioButton, de forma que el abanico de soluciones para el desarrollador ante este tipo de problema sea algo más amplio.

En primer lugar añadiremos a la solución una nueva página llamada FiltrosRadio.xaml, con los habituales DomainDataSource y DataGrid; incorporando además, cinco RadioButton, en cuya propiedad Content asignaremos el nombre de algunos de los países utilizados en el campo BillingCountry de la tabla Invoice.

A continuación incluiremos un filtro en el DomainDataSource, cuyo valor sea proporcionado por el primero de los RadioButton de la página (listado 5). Para poder actualizar el filtro cada vez que el usuario haga clic en un nuevo RadioButton (figura 2), escribiremos un manipulador para el evento RadioButton.Checked, con la particularidad de que será un manipulador genérico para todos los RadioButton de la página. En el código del mismo, comprobaremos si el control DomainDataSource ha sido creado, y en caso afirmativo, obtendremos su objeto FilterDescriptor, asignándole a la propiedad Value, la propiedad Content del RadioButton que acaba de ser pulsado, y que obtenemos a través del parámetro sender del manipulador (listado 6).

```
<riaControls:DomainDataSource x:Name="ddsInvoices" QueryName="GetInvoices"
AutoLoad="True">
    <!--....-->
    <riaControls:DomainDataSource.FilterDescriptors>
        <riaControls:FilterDescriptor PropertyPath="BillingCountry"
            Operator="IsEqualTo"
            Value="{Binding ElementName=rbtBelgium,Path=Content}" />
    </riaControls:DomainDataSource.FilterDescriptors>
</riaControls:DomainDataSource>
<!--....-->
<Border BorderThickness="2" Width="150" Margin="5" BorderBrush="Black">
    <StackPanel Margin="10">
        <RadioButton x:Name="rbtBelgium" Content="Belgium" Width="120"
IsChecked="True" Checked="RadioButton_Checked" />
        <RadioButton Content="France" Width="120" Checked="RadioButton_Checked" />
        <RadioButton Content="Italy" Width="120" Checked="RadioButton_Checked" />
        <RadioButton Content="Spain" Width="120" Checked="RadioButton_Checked" />
        <RadioButton Content="United Kingdom" Width="120"
Checked="RadioButton_Checked" />
    </StackPanel>
</Border>

<sdk:DataGrid x:Name="grdInvoices"
    <!--....-->
```

Listado 5.

```
private void RadioButton_Checked(object sender, RoutedEventArgs e)
{
    if (this.ddsInvoices != null)
    {
        FilterDescriptor oFilterDescriptor = this.ddsInvoices.FilterDescriptors[0];
        oFilterDescriptor.Value = ((RadioButton)sender).Content;
    }
}
```

Listado 6.

Figura 2. Filtrando mediante controles RadioButton.

Conclusiones

Como acabamos de comprobar en el presente artículo, la conexión entre las características de filtrado del DomainDataSource y los controles de la interfaz de usuario, bien de forma exclusivamente declarativa, o en combinación con el code behind de la página, permiten potenciar nuestras aplicaciones, abriendo un campo a través del cual enriquecer la experiencia de usuario.

REFERENCIAS

[1] Blanco, Luis Miguel. "DomainDataSource. Un gestor de datos en Silverlight para la interfaz de usuario". dotNetManía nº 78.