Manipulación de datos en Silverlight mediante ADO.NET Data Services (II)

Luis Miguel Blanco Ancos

Tras la introducción y acercamiento inicial a los servicios de ADO.NET Data Services realizados en la primera entrega del presente artículo, en esta segunda parte abordaremos su manipulación desde el lado cliente de la aplicación, donde centraremos nuestra atención en algunas de las técnicas disponibles para la obtención y presentación de los datos del servicio en la interfaz de usuario.

Implementando el servicio de datos en Silverlight

Como reza en su título, este es un artículo sobre la aplicación y uso de ADO.NET Data Services en Silverlight, por lo que ha llegado la hora de mostrar el modo de acceder al servicio de datos desde el lado cliente [1].

Con tal finalidad, en esta segunda entrega y en la tercera llevaremos a cabo un conjunto de pasos para implementar las operaciones habituales de lectura, modificación, inserción y borrado sobre la fuente de datos, también conocidas por el acrónimo CRUD –Create, Read, Update, Delete.

Comenzaremos por la consulta, recuperación y presentación de los datos en la interfaz de usuario. La base de nuestros ejemplos seguirá siendo la solución ServicioDatosConsultaSL, que comenzamos a desarrollar en la primera parte, y que se encuentra disponible en el sitio web http://www.dotnetmania.com.

Para poder acceder desde Silverlight a nuestro servicio de datos, debemos establecer una referencia hacia el mismo, por lo que nos situaremos en el Explorador de Soluciones de Visual Studio 2008 y haremos clic derecho sobre el proyecto Silverlight de nuestra solución, seleccionando la opción *Add Service Reference*. Seguidamente se abrirá el cuadro de diálogo para establecer dicha referencia, donde pulsaremos el botón Discover para que haga una búsqueda de los servicios existentes en la solución sobre la que actualmente estamos trabajando.

Como consecuencia, nuestro servicio de datos será localizado, mostrándose en el cuadro de diálogo su URL y el contenido del servicio. Como nombre para la referencia del servicio, en el cuadro de texto Namespace escribiremos RefWDSNorthwind, y aceptaremos el diálogo, quedando establecida la referencia.

A continuación pasaremos a la creación de la interfaz de usuario, para lo que abriremos el diseñador de la página Page.xaml, donde agregaremos un panel de tipo StackPanel, en cuyo interior situaremos un DataGrid y un Button, como vemos en el listado 1.

```
<UserControl xmlns:data="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
   x:Class="ServicioDatosConsultaSL.Page"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   Width="400" Height="325">
   <Grid x:Name="LayoutRoot" Background="White">
      <StackPanel Background="PaleGoldenrod">
         <data:DataGrid x:Name="grdDatos"
                  ItemsSource="{Binding}" Width="390" Height="240" Margin="2" />
         <Button x:Name="btnCargar1" Content="Cargar datos (1)" Width="120" Margin="2" />
      </StackPanel>
   </Grid>
</UserControl>
```

Listado 1. Definición de la página para la visualización de consultas del servicio de datos.

Seguidamente, comenzaremos a escribir la funcionalidad de esta página seleccionando la opción de menú de Visual Studio 2008 *View > Code*, entrando de esta manera en su editor de código.

Comenzaremos por declarar los siguientes espacios de nombre.

- ServicioDatosConsultaSL.RefWDSNorthwind. Contiene las clases del servicio de datos con el que acabamos de establecer una referencia. Estas clases representan las entidades del servicio, así como el contenedor de las mismas.
- System.Collections.ObjectModel. Contiene un conjunto de tipos para manipular colecciones, entre los que se encuentra ObservableCollection<T>, que utilizaremos en los ejemplos del presente artículo, para volcar el contenido resultante de una petición al servicio de datos y su posterior visualización en los controles de la interfaz de usuario.

En los próximos ejemplos presentaremos algunas de las técnicas disponibles para recuperar los elementos de la entidad Suppliers –todos ellos, o un subconjunto mediante filtros—a través del servicio de datos, traspasando el resultado a una colección que mostraremos en el control DataGrid de la página Silverlight.

Primeramente escribiremos un método manipulador para el evento Click del botón btnCargar1, donde crearemos una instancia del servicio de datos a través de la clase NorthwindEntities, cuyo constructor precisa de un objeto Uri, al que pasaremos como parámetro una cadena con el nombre del servicio.

Una vez obtenida la instancia del servicio, podemos comenzar a solicitarle datos llamando a su método BeginExecute<T>, el cual lanzará una consulta asíncrona –recordemos que las peticiones asíncronas son el modo en que Silverlight trabaja con los servicios web— contra el servicio, siendo su resultado devuelto en un método de devolución de llamada, un delegado de tipo AsyncCallback para ser más exactos.

El primer parámetro de BeginExecute<T> debe ser un objeto Uri con el nombre de la entidad a obtener, lo que sería equivalente a enviar la siguiente ruta desde el navegador: http://localhost/ServicioDatosConsultaSL/wdsNorthwind.svc/Suppliers.

El segundo parámetro es el nombre del método –delegado AsyncCallback— que será invocado automáticamente cuando el servicio haya obtenido todos los datos solicitados y esté listo para devolverlos.

El tercer y último parámetro es un objeto de estado que será pasado en el parámetro del método de devolución de llamada. Para este ejemplo utilizaremos la variable que contiene el servicio de datos.

A continuación, dentro ya del método de devolución de llamada CargarCompletado1, tomaremos el parámetro de tipo IAsyncResult que recibe, obteniendo de su propiedad AsyncState la referencia al servicio de datos. Para completar la llamada al servicio ejecutaremos su método EndExecute<T>, pasándole como parámetro el objeto IAsyncResult, y obteniendo como resultado un enumerador, el cual recorreremos para llenar una colección ObservableCollection<Suppliers>, que finalmente asignaremos como contexto de datos del control DataGrid. El listado 2 muestra el conjunto de acciones que acabamos de describir.

```
using ServicioDatosConsultaSL.RefWDSNorthwind;
using System.Collections.ObjectModel;
//....
public Page()
{
   InitializeComponent();
   this.btnCargar1.Click += new RoutedEventHandler(btnCargar1_Click);
}

void btnCargar1_Click(object sender, RoutedEventArgs e)
{
   NorthwindEntities dtsvcNorthwind = new NorthwindEntities(new Uri("wdsNorthwind.svc",
UriKind.Relative));
   dtsvcNorthwind.BeginExecute<Suppliers>(new Uri("Suppliers", UriKind.Relative), CargarCompletado1,
dtsvcNorthwind);
}

void CargarCompletado1(IAsyncResult oResultado)
{
   NorthwindEntities dtsvcNorthwind = (NorthwindEntities)oResultado.AsyncState;
   IEnumerable<Suppliers> lstSuppliers = dtsvcNorthwind.EndExecute<Suppliers>(oResultado);
   ObservableCollection<Suppliers> cllSuppliers = new ObservableCollection<Suppliers>();
```

```
    foreach(Suppliers oSupplier in lstSuppliers)
    {
        cllSuppliers.Add(oSupplier);
    }

    this.grdDatos.DataContext = cllSuppliers;
}
```

Listado 2. Obtención del contenido de una entidad del servicio de datos.

La figura 1 visualiza la aplicación en ejecución, donde vemos el DataGrid que hemos llenado con los elementos de la entidad Suppliers.

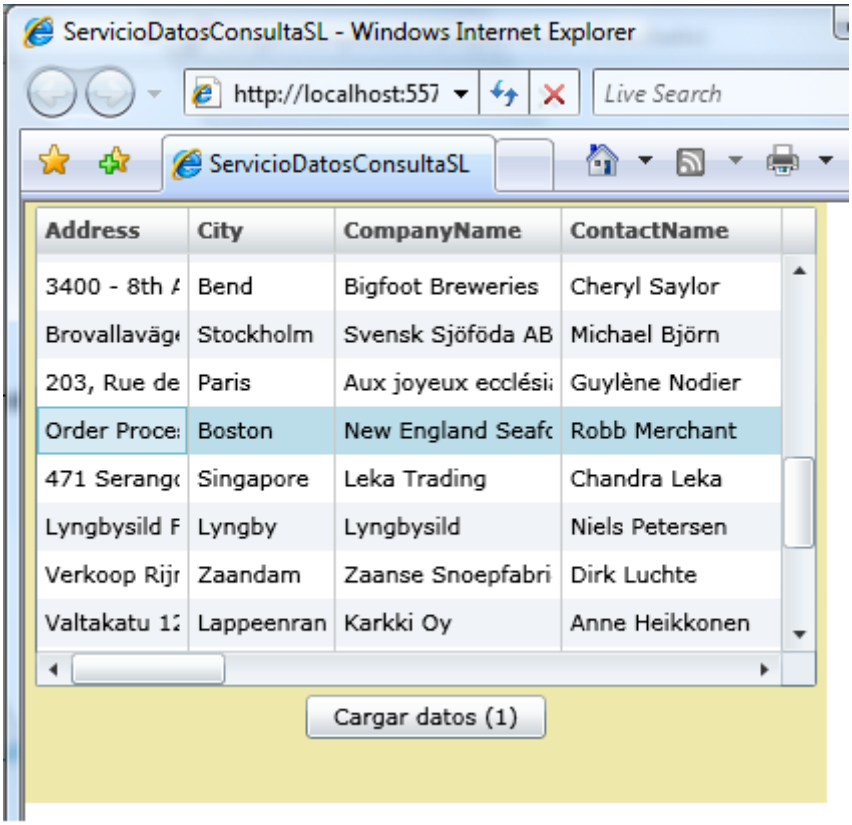


Figura 1. Visualizando los elementos de una entidad en el DataGrid.

Existe una vía alternativa para obtener un resultado idéntico al que acabamos de exponer, consistente en el empleo combinado de las siguientes clases.

- DataServiceContext. Representa el contexto de ejecución de un servicio ADO.NET Data Service.
- DataServiceQuery<T>. Representa una consulta contra el servicio de datos.

Estos tipos se encuentran ubicados en el espacio de nombres System.Data.Services.Client, que tendremos que declarar al comienzo del archivo de código. En el listado 3, después de añadir a la página un nuevo control Button con el

nombre btnCargar2, instanciamos un objeto DataServiceContext, que identifica a nuestro servicio de datos, y acto seguido llamamos a su método CreateQuery<T>, pasándole como parámetro el nombre de la entidad cuyos elementos deseamos obtener. El resultado será la obtención de un objeto DataServiceQuery<T>, del que ejecutaremos su método BeginExecute para comenzar la llamada al servicio de datos, y EndExecute para completarla en el método de devolución de llamada, con un patrón de ejecución similar al explicado en el anterior ejemplo.

```
using System.Data.Services.Client;
//....
void btnCargar2_Click(object sender, RoutedEventArgs e)
{
   DataServiceContext dtsvcNorthwind = new DataServiceContext(new Uri("wdsNorthwind.svc",
UriKind.Relative));
   DataServiceQuery<Suppliers> qrySuppliers = dtsvcNorthwind.CreateQuery<Suppliers>("Suppliers");
   qrySuppliers.BeginExecute(CargarCompletado2, qrySuppliers);
}

void CargarCompletado2(IAsyncResult oResultado)
{
   DataServiceQuery<Suppliers> qrySuppliers = (DataServiceQuery<Suppliers>)oResultado.AsyncState;
   IEnumerable<Suppliers> lstSuppliers = qrySuppliers.EndExecute(oResultado);
   ObservableCollection<Suppliers> cllSuppliers = new ObservableCollection<Suppliers>();

   foreach (Suppliers oSupplier in lstSuppliers)
   {
      cllSuppliers.Add(oSupplier);
   }

   this.grdDatos.DataContext = cllSuppliers;
}
```

Listado 3. Accediendo al servicio mediante las clases DataServiceContext y DataServiceQuery.

Tras cargar de datos el DataGrid empleando cualquiera de los dos ejemplos anteriores, al navegar por sus columnas hallaremos una con el título Products, lo cual indica, que por cada una de las filas de la entidad Suppliers que está mostrando el control, podríamos acceder a los elementos relacionados de la entidad Products; aspecto este que será motivo de un próximo ejemplo.

Este comportamiento se debe a que, por defecto, la propiedad AutoGenerateColumns de este control tiene el valor true, lo cual produce que sea el propio DataGrid quien se encargue de definir columnas para cada una de las propiedades de la colección asignada como fuente de datos para el control; algo que puede no ser conveniente en un caso como el actual, donde posiblemente no querríamos que la columna Products apareciese.

Podríamos definir las columnas para el DataGrid utilizando código XAML, pero para simplificar el ejemplo, lo que haremos en esta ocasión –como muestra el listado 4— será ocultar dicha columna recurriendo al evento AutoGeneratingColumn del DataGrid, en cuyo método manipulador, gracias al parámetro DataGridAutoGeneratingColumnEventArgs que recibe, detectaremos cuándo se va a generar la columna mediante la propiedad PropertyName, evitando su creación al asignar el valor true a la propiedad Cancel.

```
public Page()
{
   //....
   this.grdDatos.AutoGeneratingColumn += new
EventHandler<DataGridAutoGeneratingColumnEventArgs>(grdDatos_AutoGeneratingColumn);
}

void grdDatos_AutoGeneratingColumn(object sender, DataGridAutoGeneratingColumnEventArgs e)
{
   if (e.PropertyName == "Products")
   {
      e.Cancel = true;
   }
}
```

Listado 4. Ocultar columnas del DataGrid en tiempo de ejecución.

Enviando consultas al servicio de datos mediante LINQ

Además de las técnicas descritas en los anteriores ejemplos para obtener la información de un servicio de datos, existe la posibilidad de emplear también LINQ para lograr el mismo fin, con la ventaja añadida de poder aplicar condiciones para filtrar y ordenar el resultado obtenido.

Si ya tenemos alguna experiencia con LINQ, aunque sea mínima, la creación de una expresión de consulta para un servicio de datos resultará una tarea muy sencilla, puesto que simplemente debemos construir nuestra consulta en el modo habitual, pero especificando el servicio como fuente de datos.

Una vez creada la consulta en LINQ, realizaremos una operación de conversión de tipo sobre la misma para transformarla en un tipo DataServiceQuery<T>, sobre el cual ya podremos ejecutar su método BeginExecute, que dará comienzo a la petición de datos hacia el servicio. El listado 5 muestra este proceso, que llevamos a cabo en el evento Click de un nuevo control Button añadido a nuestra página Silverlight. Obviamos comentar de nuevo el método de devolución de llamada, ya que es idéntico al explicado en los anteriores ejemplos.

```
void btnCargar3_Click(object sender, RoutedEventArgs e)
{
```

```
   NorthwindEntities dtsvcNorthwind = new NorthwindEntities(new Uri("wdsNorthwind.svc",
UriKind.Relative));
   var xConsulta = from Supplier in dtsvcNorthwind.Suppliers
                   where Supplier.ContactTitle.Contains("Manager")
                   orderby Supplier.CompanyName
                   select Supplier;

   DataServiceQuery<Suppliers> qrySuppliers = (DataServiceQuery<Suppliers>)xConsulta;
   qrySuppliers.BeginExecute(CargarCompletado3, qrySuppliers);
}
```

Listado 5. Empleando LINQ para consultar al servicio de datos.

El resultado obtenido al ejecutar este ejemplo mostrará aquellos registros de la tabla Suppliers, en cuya columna ContactTitle aparece la palabra Manager, ordenados por la columna CompanyName.

Visualizar las entidades asociadas (relacionadas) en modo maestro-detalle

En uno de los ejemplos mostrados anteriormente, se comentaba la posibilidad de acceder, desde los elementos obtenidos de una entidad, a los elementos de otra con la que tuviera establecida una asociación, mediante un estilo maestro-detalle –más concretamente, desde Suppliers hasta Products.

El mencionado comportamiento resulta posible gracias a que en la clase que representa la entidad maestra –Suppliers—, una de sus propiedades –Products– es de tipo Collection, permitiendo albergar, para cada elemento de dicha entidad, todos los elementos de la entidad detalle –Products– con los que se relaciona.

No obstante, cuando asignamos a un tipo IEnumerable los elementos de la entidad Suppliers resultantes de una petición contra el servicio de datos, al recorrer dicho enumerador, la propiedad-colección Products de cada instancia de Suppliers estará vacía, ya que, por defecto, el servicio de datos no devuelve los elementos de la entidad relacionada con la que se tiene establecida una asociación. Podemos comprobar este aspecto en particular desde el depurador, observando el contenido de cada objeto Supplier durante la carga de la colección que seguidamente asignamos al DataGrid.

La forma de obtener tales datos pasa por hacer uso del método Expand, presente en las clases que representan a las entidades del servicio. De este modo, cuando solicitemos al servicio de datos la entidad Suppliers, si queremos que se incluyan los datos relacionados de Products, escribiremos nuestra consulta LINQ como se muestra en el listado 6.

```
var xConsulta = from Supplier in dtsvcNorthwind.Suppliers.Expand("Products")
                select Supplier;
```

Listado 6. Utilizando el método Expand en LINQ.

La anterior expresión LINQ sería equivalente a escribir la siguiente URL en el navegador

http://localhost/ServicioDatosConsultaSL/wdsNorthwind.svc/Suppliers?$expand=Products

Si necesitamos presentar esta información en la interfaz de usuario de nuestra aplicación, una posible opción sería emplear sendos controles DataGrid, para entidad maestra y detalle respectivamente.

Con el fin de no alterar la página Silverlight que hemos construido para los ejemplos desarrollados hasta este momento, lo que haremos será agregar, a la solución con la que estamos trabajando, un nuevo proyecto Silverlight al que daremos el nombre ServicioDatosConsultaSL2. Dentro de este nuevo proyecto, crearemos una referencia al servicio de datos en la forma explicada con anterioridad, mientras que en la aplicación Web estableceremos Default.aspx como página de inicio.

Abriendo el diseñador de Default.aspx, añadiremos varios controles LinkButton para que nos lleven a las diferentes páginas Silverlight, como vemos en el listado 7.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Página inicial</title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
    <asp:LinkButton ID="LinkButton1" runat="server"
PostBackUrl="ServicioDatosConsultaSLTestPage.aspx">1. Consultas varias</asp:LinkButton>
    <br />
    <asp:LinkButton ID="LinkButton2" runat="server"
PostBackUrl="ServicioDatosConsultaSL2TestPage.aspx">2. Maestro-detalle</asp:LinkButton>
    <!--....-->
  </div>
  </form>
</body>
</html>
```

Listado 7. Página inicial para los ejemplos de la solución.

A continuación abriremos el diseñador de la página Page.xaml correspondiente al proyecto ServicioDatosConsultaSL2, y escribiremos el código de marcado que vemos en el listado 8. En esta ocasión vamos a controlar la creación de las columnas para los DataGrid, puesto que solamente definiremos un subconjunto de las que cada entidad dispone.

```
<StackPanel Background="LightYellow">
  <TextBlock HorizontalAlignment="Center">Suppliers</TextBlock>

  <data:DataGrid x:Name="grdSuppliers"
           Width="390" Height="225" Margin="2" Background="Aquamarine"
           AutoGenerateColumns="False" ItemsSource="{Binding}">
    <data:DataGrid.Columns>
      <data:DataGridTextColumn Binding="{Binding SupplierID}" Header="Código" />
      <data:DataGridTextColumn Binding="{Binding CompanyName}" Header="Proveedor" />
      <data:DataGridTextColumn Binding="{Binding City}" Header="Ciudad" />
      <data:DataGridTextColumn Binding="{Binding Country}" Header="País" />
    </data:DataGrid.Columns>
  </data:DataGrid>

  <TextBlock HorizontalAlignment="Center">Products</TextBlock>

  <data:DataGrid x:Name="grdProducts"
           Width="390" Height="130" Margin="2" Background="BlanchedAlmond"
           AutoGenerateColumns="False" ItemsSource="{Binding}">
    <data:DataGrid.Columns>
      <data:DataGridTextColumn Binding="{Binding ProductID}" Header="Código" />
      <data:DataGridTextColumn Binding="{Binding ProductName}" Header="Producto" />
      <data:DataGridTextColumn Binding="{Binding QuantityPerUnit}" Header="Cantidad Unit." />
      <data:DataGridTextColumn Binding="{Binding UnitPrice}" Header="Precio" />
    </data:DataGrid.Columns>
  </data:DataGrid>
</StackPanel>
```

Listado 8. Página XAML para visualizar los datos en maestro-detalle.

Seguidamente pasaremos al codebehind de esta página para implementar la funcionalidad de los controles que acabamos de añadir en diseño.

Como podemos ver en el listado 9, los procesos de obtención y carga de datos de la entidad maestra –Suppliers—no revisten novedad alguna respecto a pasados ejemplos, a excepción del uso del método Expand. Donde centraremos nuestra atención será en el evento SelectionChanged del DataGrid, que se producirá cada vez que el usuario cambie la fila activa. En dicho evento refrescaremos el contenido del DataGrid de detalle, ya que tomaremos el objeto Suppliers seleccionado, y recorreremos los elementos de su propiedad Products para rellenar la cuadrícula de detalle.

```
using ServicioDatosConsultaSL2.RefWDSNorthwind;
using System.Collections.ObjectModel;
using System.Data.Services.Client;
//....
public partial class Page : UserControl
{
  public Page()
  {
    InitializeComponent();
```

```
        this.Loaded += new RoutedEventHandler(Page_Loaded);
        this.grdSuppliers.SelectionChanged += new
SelectionChangedEventHandler(grdSuppliers_SelectionChanged);
    }

    void Page_Loaded(object sender, RoutedEventArgs e)
    {
        NorthwindEntities dtsvcNorthwind = new NorthwindEntities(new Uri("wdsNorthwind.svc",
UriKind.Relative));
        var xConsulta = from Supplier in dtsvcNorthwind.Suppliers.Expand("Products")
                        select Supplier;

        DataServiceQuery<Suppliers> qrySuppliers = (DataServiceQuery<Suppliers>)xConsulta;
        qrySuppliers.BeginExecute(CargarSuppliersCompletado, qrySuppliers);
    }

    void CargarSuppliersCompletado(IAsyncResult oResultado)
    {
        DataServiceQuery<Suppliers> qrySuppliers = (DataServiceQuery<Suppliers>)oResultado.AsyncState;
        IEnumerable<Suppliers> lstSuppliers = qrySuppliers.EndExecute(oResultado);
        ObservableCollection<Suppliers> cllSuppliers = new ObservableCollection<Suppliers>();

        foreach (Suppliers oSupplier in lstSuppliers)
        {
            cllSuppliers.Add(oSupplier);
        }

        this.grdSuppliers.DataContext = cllSuppliers;
    }

    void grdSuppliers_SelectionChanged(object sender, SelectionChangedEventArgs e)
    {
        Suppliers oSupplier = (Suppliers)this.grdSuppliers.SelectedItem;
        ObservableCollection<Products> cllProducts = new ObservableCollection<Products>();
        foreach (Products oProduct in oSupplier.Products)
        {
            cllProducts.Add(oProduct);
        }
        this.grdProducts.DataContext = cllProducts;
    }
}
```

Listado 9. Obtención de los elementos relacionados de una entidad.

La figura 2 muestra el resultado en ejecución.

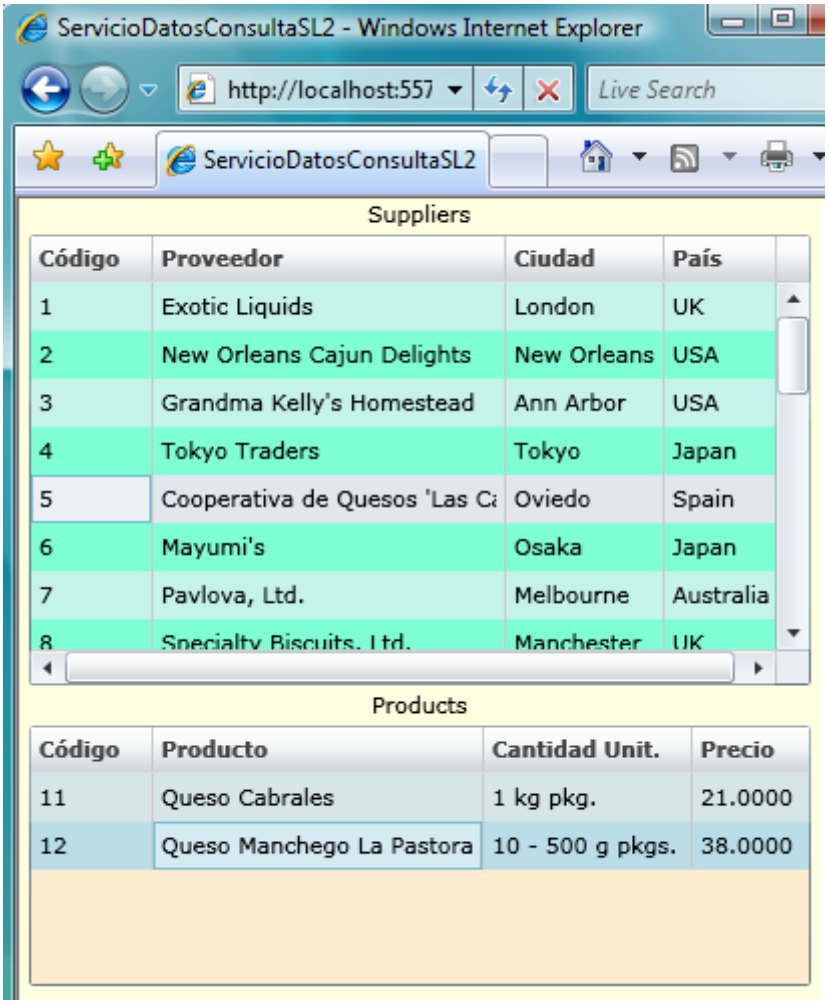


Figura 2. Visualización en modo maestro-detalle.

Maestro-detalle con carga de elementos relacionados bajo demanda

Cuando utilizamos el método Expand, como hemos visto en el ejercicio que acabamos de desarrollar, cargamos todos los elementos de la entidad relacionada sin importar el hecho de que realmente se vayan a utilizar durante la ejecución del programa, lo cual puede suponer un consumo de recursos y una demora de tiempo extra innecesarias.

Supongamos que al obtener el contenido de la entidad Suppliers, queremos recuperar los elementos relacionados de la entidad Products, pero solamente en el caso de que en la propiedad Suppliers.Address exista el valor 9.

Esta forma selectiva en la obtención de los elementos asociados a una entidad es posible gracias al método BeginLoadProperty, presente en la clase que representa al servicio de datos: DataServiceContext, y en la clase generada por el diseñador de EDM, NorthwindEntities.

Como ejercicio ilustrativo del uso de este método, añadiremos a la solución un nuevo proyecto con el nombre ServicioDatosConsultaSL3, al que dotaremos de la misma interfaz de usuario que el ejemplo anterior: dos controles DataGrid.

En lo que atañe a la funcionalidad que tenemos que implementar, al cargar la página haremos una petición al servicio de datos para obtener la entidad Suppliers, pero sin los elementos de Products relacionados. Será en el evento SelectionChanged del DataGrid maestro donde comprobaremos el valor de la propiedad Address del objeto Suppliers seleccionado, y en el caso de que se cumpla la condición antes mencionada, obtendremos los elementos relacionados de Products llamando al método BeginLoadProperty.

BeginLoadProperty es un método que realiza, para una instancia de una entidad, la carga de los elementos de otra entidad con la que guarda relación, todo ello de forma asíncrona.

Como primer parámetro pasaremos la instancia de la entidad; en segundo lugar, una cadena con el nombre de la entidad relacionada; a continuación el nombre del método de devolución de llamada que será invocado cuando BeginLoadProperty finalice su carga de datos; y como último parámetro, un objeto de estado que podremos recuperar en el método de devolución de llamada, para obtener los elementos relacionados.

El listado 10 muestra el código correspondiente al evento SelectionChanged y al método de devolución de llamada recién descrito.

```
public partial class Page : UserControl
{
   NorthwindEntities dtsvcNorthwind;
   //....
   void grdSuppliers_SelectionChanged(object sender, SelectionChangedEventArgs e)
   {
      this.grdProducts.DataContext = null;
      Suppliers oSupplier = (Suppliers)this.grdSuppliers.SelectedItem;

      if (oSupplier.Address.Contains("9"))
      {
         dtsvcNorthwind.BeginLoadProperty(oSupplier, "Products", CargarProductsCompletado,
oSupplier);
      }
   }

   void CargarProductsCompletado(IAsyncResult oResultado)
   {
      Suppliers oSupplier = (Suppliers)oResultado.AsyncState;
      dtsvcNorthwind.EndLoadProperty(oResultado);
      ObservableCollection<Products> cllProducts = new ObservableCollection<Products>();

      foreach (Products oProduct in oSupplier.Products)
      {
```

```
            cllProducts.Add(oProduct);
        }

        this.grdProducts.DataContext = cllProducts;
    }
}
```

Listado 10. Obtención de los elementos detalle en base a una condición.

Detalle-maestro. Cambiando la dirección en la obtención de datos relacionados

Como el propio nombre de este apartado sugiere, también es posible obtener la información entre dos entidades relacionadas en sentido inverso, es decir, situados en un elemento detalle, conseguir los valores del elemento maestro con el que guarda relación.

La forma de implementar este comportamiento pasa también por el empleo del método Expand, pero invocado esta vez desde la entidad detalle. Pongamos como ejemplo un DataGrid –que crearemos en un nuevo proyecto de la solución con el nombre ServicioDatosConsultaSL4— que muestra algunas columnas de la entidad Products. Si para cada producto queremos visualizar también el nombre del proveedor que lo suministra y su ciudad, en la expresión LINQ utilizada para obtener la entidad Products llamaremos a su método Expand, pasándole como parámetro el nombre de la entidad relacionada, como vemos en el listado 11.

```
var xConsulta = from Product in dtsvcNorthwind.Products.Expand("Suppliers")
                select Product;
```

Listado 11. Acceso a la información maestra desde el detalle.

Omitimos el resto del codebehind de este proceso por seguir la misma mecánica de los anteriores ejemplos.

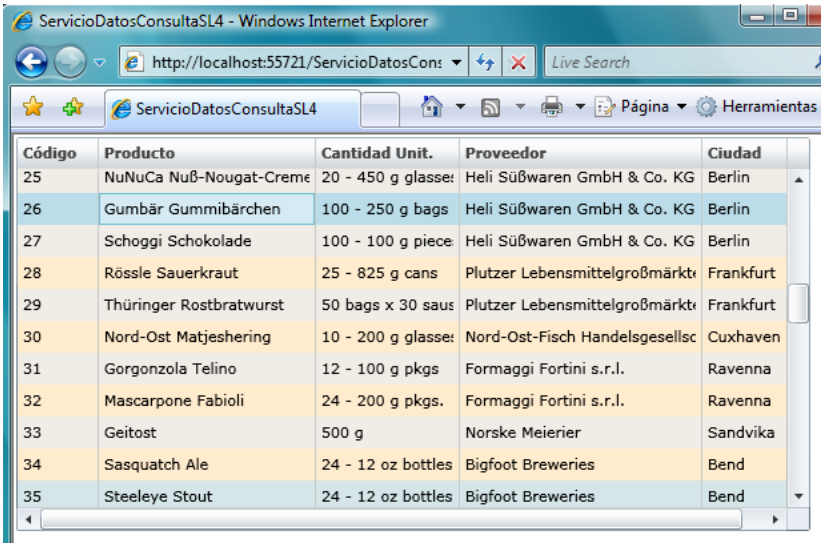
Por otro lado, dentro del código de marcado de la página, en la definición de columnas del DataGrid, además de las pertenecientes a la propia entidad Products, añadiremos aquellas que muestran los valores relacionados de Suppliers, como vemos en el listado 12.

```
<data:DataGrid x:Name="grdDatos"
    Margin="2" Background="BlanchedAlmond"
    AutoGenerateColumns="False" ItemsSource="{Binding}">
    <data:DataGrid.Columns>
        <data:DataGridTextColumn Binding="{Binding ProductID}" Header="Código" />
        <data:DataGridTextColumn Binding="{Binding ProductName}" Header="Producto" />
        <data:DataGridTextColumn Binding="{Binding QuantityPerUnit}" Header="Cantidad Unit." />
        <data:DataGridTextColumn Binding="{Binding Suppliers.CompanyName}" Header="Proveedor" />
        <data:DataGridTextColumn Binding="{Binding Suppliers.City}" Header="Ciudad" />
    </data:DataGrid.Columns>
```

```
</data:DataGrid>
```

Listado 12. Definición de página para mostrar información de detalle y maestra.

Del anterior listado debemos destacar la sintaxis para crear el enlace en las columnas de la entidad Suppliers, donde mediante el uso del operador punto, especificamos el nombre de la propiedad perteneciente a la entidad maestra. El resultado en ejecución lo muestra la figura 3.



Código	Producto	Cantidad Unit.	Proveedor	Ciudad
25	NuNuCa Nuß-Nougat-Creme	20 - 450 g glasses	Heli Süßwaren GmbH & Co. KG	Berlin
26	Gumbär Gummibärchen	100 - 250 g bags	Heli Süßwaren GmbH & Co. KG	Berlin
27	Schoggi Schokolade	100 - 100 g piece	Heli Süßwaren GmbH & Co. KG	Berlin
28	Rössle Sauerkraut	25 - 825 g cans	Plutzer Lebensmittelgroßmärkte	Frankfurt
29	Thüringer Rostbratwurst	50 bags x 30 saus	Plutzer Lebensmittelgroßmärkte	Frankfurt
30	Nord-Ost Matjeshering	10 - 200 g glasses	Nord-Ost-Fisch Handelsgesellsc	Cuxhaven
31	Gorgonzola Telino	12 - 100 g pkgs	Formaggi Fortini s.r.l.	Ravenna
32	Mascarpone Fabioli	24 - 200 g pkgs.	Formaggi Fortini s.r.l.	Ravenna
33	Geitost	500 g	Norske Meierier	Sandvika
34	Sasquatch Ale	24 - 12 oz bottles	Bigfoot Breweries	Bend
35	Steeleye Stout	24 - 12 oz bottles	Bigfoot Breweries	Bend

Figura 3. Visualizando la información maestra a partir del detalle.

Conclusión

Una vez completado el ejemplo anterior, damos por concluida esta segunda parte del artículo. La tercera entrega –la cual finaliza esta serie—estará dedicada a la edición de datos, donde trataremos las operaciones habituales de alta, baja y modificación a través de de ADO.NET Data Services, y siempre desde el prisma de Silverlight como capa de interfaz de usuario.

BIBLIOGRAFIA

[1] Client Applications for Silverlight (ADO.NET Data Services/Silverlight)

http://msdn.microsoft.com/en-us/library/cc838191(VS.95).aspx