Capítulo 10
Controles de datos en WebForms

Objetivo

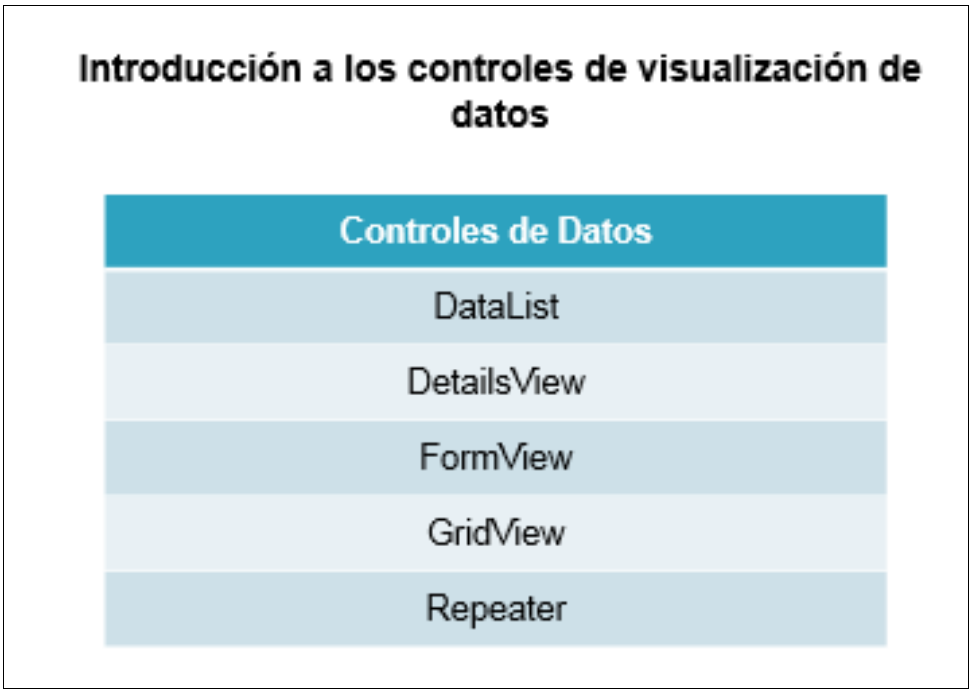
Al finalizar el capítulo, el alumno:

- Identificar los principales controles de visualización de datos, reconociendo sus características y funcionalidad.
- Utilizar apropiadamente las plantillas según el tipo de control.
- Establecer el origen de los datos de un control.

Temas

1. Introducción a los controles de visualización de datos
   - DataList
   - DetailsView
   - FormView
   - GridView
   - Repeater
2. DataSource y templates

1. Introducción a los controles de visualización de datos



Los controles de visualización y manipulación de datos (**DataBound Controls)** permiten que el usuario pueda interactuar libremente con los datos, sin importar la procedencia ni naturaleza de los mismos.

Estos controles pueden llegar a englobar bastante funcionalidad de manera automática, lo que incrementa nuestra productividad de gran manera. Estos controles por lo general, no tienen un equivalente directo en HTML, sino que generan un gran conjunto de elementos HTML.

Estos controles son:

ListBox

El control Listbox es utilizado para mostrar un listado de elementos al usuario, el cual puede hacer una selección simple o múltiple. La propiedad **SelectionMode** es utilizada para habilitar la selección múltiple, ya que por defecto es simple. Además, el control Listbox tiene la propiedad **Rows**, la cual permite definir el alto del control, especificando la cantidad de elementos a mostrar.

La propiedad **Items**, de tipo colección, contiene la colección de **ListItem** del control. Por otro lado, la propiedad **SelectedValue** permite conocer el valor seleccionado del control, mientras que la propiedad **SelectedIndex** permite conocer la posición del ListItem seleccionado.
Para conocer los elementos seleccionados, se debe verificar por cada ListItem**,** la propiedad **Selected**.

Este control dispara el evento **SelectedIndexChanged**, cada vez que el valor del control ha cambiado.

DropDownList

El control DropDownList es utilizado para mostrar un conjunto de elementos, del cual, el usuario puede hacer una selección simple. Contiene casi las mismas propiedades que el control **Listbox**.

Repeater

El control **Repeater** renderiza de manera repetitiva, una serie de elementos en forma de lista según las plantillas que se tengan definidas. Como mínimo, el control **Repeater,** debe definir una plantilla para cada elemento, es decir, una plantilla **ItemTemplate**.

DataList

El control DataList es muy semejante al control Repeater. Este control es más configurable, pues incorpora tres propiedades llamadas **RepeaterDirection**, **RepeatLayout** y **RepeatColumns**, que permiten indicar de qué forma se va a ir mostrando y distribuyendo la información sin tener que especificarlo a través de las plantillas.

- RepeatLayout: indica qué distribución se le va a dar a los datos, es decir, a cada elemento de datos incluido en la plantilla **DataItem**. Esta propiedad puede tomar dos valores: **Table**, para indicar un formato de distribución de tabla tipo de HTML con sus filas y columnas correspondientes, y el valor **Flow**, para indicar que los elementos se van a distribuir sin ningún formato de tabla, sino que lo hacen de forma lineal. El valor por defecto de esta propiedad es **Table**.

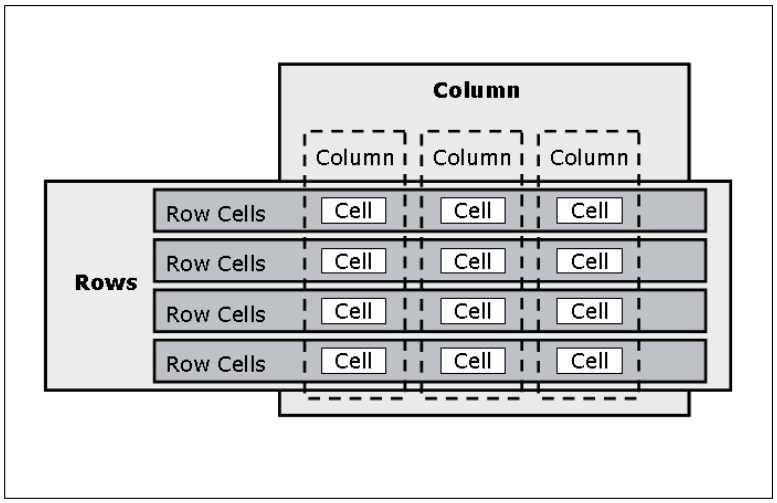
- RepeatDirection: indica si los elementos se van a mostrar de forma horizontal o vertical, es decir, la dirección en la que se van a ir repartiendo los distintos elementos contenidos en el control DataList. Los valores que acepta esta propiedad son Vertical y Horizontal. Por defecto, esta propiedad tiene el valor Vertical.
- RepeatColumns: recibe un valor entero que indicará el número de columnas utilizado para mostrar los datos. Por defecto, el valor de esta propiedad es 1.

El control DataList permite especificar el estilo que va a tener cada uno de los distintos tipos de plantillas utilizados para mostrar la información, esto se consigue mediante una serie de propiedades ofrecidas por el control, así por ejemplo, se tiene una propiedad denominada **ItemStyle-Fond-Bold**, para indicar si la fuente de texto que va a aparecer dentro de una plantilla **ItemTemplate**, va a estar negrita o no, y si existirá una propiedad similar para cada una de las distintas plantillas.

El control **DataList** permite utilizar dos templates, que el control **Repeater** no las soporta, estas plantillas se denominan **SelectedItemTemplate** y **EditItemTemplate**.

GridView

El control GridView muestra los valores de un origen de datos en una tabla donde cada columna representa un campo y cada fila representa un registro. El control GridView permite seleccionar, ordenar y modificar estos elementos. Se renderiza como una tabla HTML. Asimismo, permite implementar la paginación, ordenamiento y edición de sus elementos, sin necesidad de tener que escribir varias líneas de código.



El control **GridView** contiene una colección de objetos del tipo **GridViewRow**, asociada a las filas del control, y una colección de objetos **DataControlField**, asociada a las columnas del control. Además, como la clase **GridViewRow** hereda de la clase **TableRow**, tiene una propiedad de tipo colección, denominada **Cells**, la cual es una colección de objetos **DataControlFieldCell**.

Cada **DataControlField**, permite inicializar a todas sus celdas, a través del método **InitializeCell**.

A través del **DataControlField**, se puede definir el tipo de columna a mostrar, el cual puede ser **BoundField**, **TemplateField**, **ButtonField**, **AutoGeneratedField**, **HyperLinkField**, **CommandField**, **CheckBoxField** e **ImageField**.

DetailsView

El control DetailsView permite mostrar, modificar, insertar o eliminar un solo registro cada vez de su origen de datos asociado. El control DetailsView muestra solamente un único registro de datos. El control DetailsView permite editar, eliminar e insertar registros. Si la propiedad **AllowPaging** se encuentra definida en **True**, el control puede ser utilizado para navegar, a través de la fuente de datos.

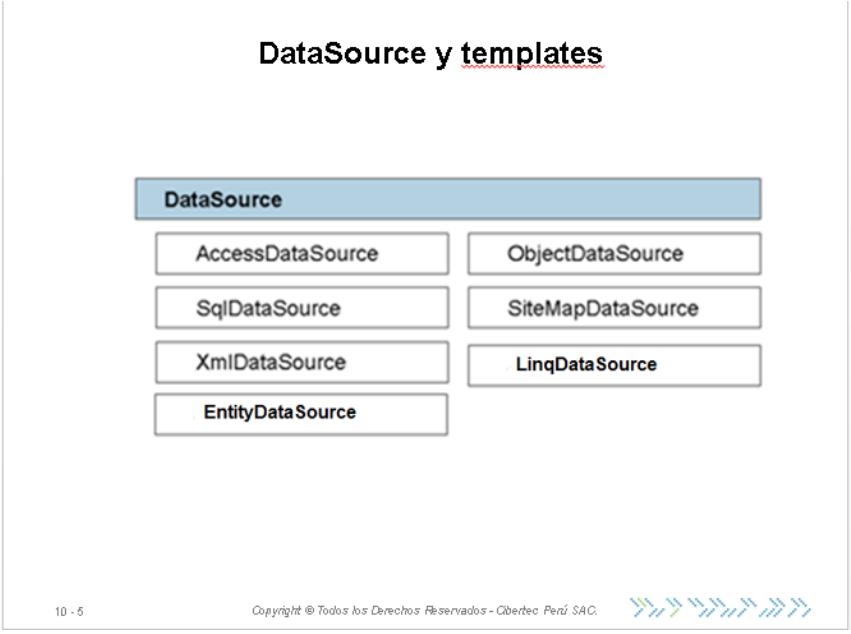
El control DetailsView también puede utilizarse en conjunto con **GridView**, **ListBox** o **DropDownList**, en el caso se desee mostrar un modelo Maestro-Detalle. Sin embargo, no soporta ordenamiento.

FormView

El control FormView permite trabajar con un único registro de un origen de datos de forma similar al control DetailsView. La diferencia entre los controles FormView y DetailsView radica en que el control DetailsView utiliza un diseño tabular en el que cada campo del registro se muestra como una fila independiente. En cambio, el control FormView no especifica ningún diseño predefinido para mostrar el registro.

El control FormView permite editar, eliminar e insertar registros.

2. DataSource y templates



DataSource

Los controles **DataBound** tienen dos propiedades importantes: **DataSource** y **DataSourceID**.

- La propiedad **DataSource,** obtiene o configura al objeto que el control DataBound utiliza para retornar un listado de datos.
  Los **ControlesDataSource,** están conformados por aquellos controles que heredan de la clase **DataSourceControl**, tales como: ObjectDataSource, SqlDataSource, AccessDatasource, XmlDataSource, SiteMapDataSource y LinqDataSource.
  El objeto **DataSource**, comúnmente es una instancia de una clase que implementa las interfaces Innumerable, IListSource, IDataSource o IHierarchicalDataSource.

- La propiedad **DataSourceID**, permite asociar el ID de un control que contiene la fuente de los datos (DataSource), como un control **SqlDataSource**.
  Los controles **DataBound,** se conectan automáticamente con el control definido en la propiedad **DataSourceID**, llamando al método **DataBind**.

Si ambas propiedades son definidas en un control, la propiedad **DataSourceID** tiene prioridad sobre la propiedad **DataSource**.

Los siguientes **DataSource** pueden utilizarse para enlazar o conectarse a una Base de Datos:

- **AccessDataSource**. Permite enlazarse con una base de datos Microsoft Access.
- **SqlDataSource**. Permite enlazarse a fuentes de datos ODBC, OLEDB, SQL Server, Oracle u otra base de datos que utilice SQL. También puede relacionarse a un Database file de SQL Server, el cual debe estar incluido en el proyecto.
- **XmlDataSource**. Permite enlazarse a un archivo XML, transformarlo o segmentarlo.
- **ObjectDataSource**. Permite enlazarse a un conjunto de Custom Business Entities o a un DataSet.
- **SitemapDataSource**. Permite enlazarse a un archivo que contenga la definición del árbol de navegación de la aplicación.
- **LinqDataSource**. Permite enlazarse a un modelo ORM de Linq para admitir el acceso de L/E hacia la base de datos utilizando un mínimo de código.
- **EntityDataSource**. Representa un Entity Data Model (EDM) para los controles enlazados a datos de una aplicación ASP.NET.

Plantillas (Templates)

Los controles **DataBound** se definen utilizando plantillas o **Templates**. Un control **DataBound** puede permitir el uso de las siguientes plantillas:

- **HeaderTemplate**. Es renderizado al inicio del control. Es opcional.
- **FooterTemplate**. Es renderizado al final del control. Es opcional.
- **ItemTemplate**. Es renderizado por cada fila en la fuente de datos.
- **AlternatingItemTemplate**. Se renderiza de manera intercalada en todas las filas de la fuente de datos. Es opcional.
- **SelectedItemTemplate**. Es utilizado para renderizar de un modo diferente la fila que ha sido seleccionada. Es opcional.
- **SeparatorTemplate**. Crea una separación entre cada elemento. Es opcional.
- **EditItemTemplate**. Muestra a la fila seleccionada, en un modo edición (mostrando cajas de texto para permitir la visualización y modificación de valores).

DataBinder, Eval y Bind

La clase **DataBinder** provee dos métodos: **Eval** y **Bind**, los cuales permiten mostrar los datos cuando se utilizan controles basados en plantillas.

El método **Eval**, es de solo lectura, es decir, solo permite mostrar los datos. Mientras que el método **Bind**, permite mostrar datos y a través de él, insertar o actualizar datos.

Con ambos métodos se puede dar un formato a los datos. La forma de uso es la siguiente:

```
<%# Eval("Nombres") %>
<%# Bind("Nombres") %>

<%# Eval("Precio", "{0 :C}") %>
<%# Bind("Precio", "{0 :C}") %>
```

En la siguiente tabla se muestran los caracteres necesarios, que pueden ser utilizados para formatear valores numéricos:

Carácter de formato	Descripción
c	Indica la moneda del sistema (10.20 €)
g	Formato general (depende del formato del dato)
e	Formato científico (3.77e+01)
f	Formato de coma flotante (3.44)
n	Formato numérico
p	Formato porcentual (20%)
x	Formato hexadecimal

En la siguiente tabla se muestran los caracteres necesarios para formatear fechas:

Carácter de formato	Descripción
d	Fecha corta (d/m/yyy)
D	Fecha larga (dddd. dd mmmm. yyyy)
f	Fecha larga y hora corta (dddd. dd mmmm yyyy hh:mm)
g	Fecha corta y tiempo corto (d/m/yyy hh:mm)
m	Mes y día (dd mmmm)
t	Hora corta (hh:mm)
y	Mes y año (mmmm, yyyy)