Escuela de Sistemas y Tecnologías

Transparencias de **ANALISTA DE SISTEMAS**

Edición 2017 – Materia: **Diseño de Aplicaciones Web**

TEMA: Controles Reutilizables

Plantel y Contactos

- **Bedelía:**
  - Mail: bedeliasistemas@bios.edu.uy

- **Encargado de Sucursal:**
  - Pablo Castaño
  - Mail: pablocasta@bios.edu.uy

Agenda

- Introducción
- User Controls
- Custom Controls
- Composite Controls

Recursos

- **Recursos Imprescindibles:**
  - Sitio Web de material
    (comunicarse con Bedelía por usuario/contraseña).
  - Transparencias del Curso.
  - Contar con el software necesario

Introducción

Tipos

Al implementar un control reutilizable dentro de .NET existen tres opciones:

1. **Controles de usuario (UserControl):** Estos son contenedores en los que se puede introducir controles web estándar (**TextBox**, **ComboBox**, **Calendar**, etc.) y controles de formatos, entre otros.
2. **Controles personalizados (CustomControl)**
   - Un control personalizado puede ser una clase escrita por un desarrollador que se deriva de la clase **Control** o de la clase **WebControl**.
   - También puede ser extender en una nueva clase un control ya existente.
3. **Controles Composite:** Este tipo de control utiliza los controles contenidos para crear su IU y generar la lógica.

Tabla Comparativa

UserControl	CostumControl
Es necesaria una copia independiente del control para cada aplicación (en ASP.NET)	Sólo es necesaria una copia del control, ubicada en la caché del ensamblado global (GAC)
No se puede agregar al Cuadro de herramientas en Visual Studio (en ASP.NET)	Se puede agregar al Cuadro de herramientas en Visual Studio mediante la biblioteca que lo contiene
Idóneo para el diseño estático	Idóneo para el diseño dinámico

User Controls

Generalidades (1)

- En ASP.NET los controles de usuario juegan un papel primordial en la reusabilidad de código.
- Los controles de usuario (archivo **.ascx**) son similares a las páginas web (archivo **.aspx**).
- Al igual que con las páginas, los controles de usuario pueden tener el código necesario para manipular su contenido e incluso realizar tareas como el enlace de datos (databinding).
- Los controles de usuario no se pueden ejecutar como archivos independientes.

Generalidades (2)

- Los controles de usuario no manejan los tags **<html>**, **<body>** y **<form>**.
- Pueden agregar cualquier control Web de ASP.NET (**Button**, **TextBox**, **Calendar**, **DropDownList**, etc.)
- Los controles de usuario permiten desacoplar código embebido en una página y convertirlo en un componente reutilizable a nivel de toda la aplicación.

Implementación

- Para implementar un control de usuario se debe crear una archivo con la extensión **.ascx**, el cual contiene obligatoriamente la directiva:

```
<% @Control Language="C#"
            ClassName="MiControl" %>
```

- Dicha directiva se encuentra encerrado en los bloques de representación en línea <%%>, lo que logra que este código sea interpretado por el compilador en el servidor.

Ubicación Código Trasero

- Se puede colocar normalmente en un archivo **.ascx.cs**
- Se puede colocar dentro del propio archivo **.ascx**, ubicado en una etiqueta SCRIPT. En estos casos no se podrá utilizar la directiva ***using***.

```
<script runat="server"
        language="lenguaje_a_implementar"
    >
   ...el código va aquí...
</script>
```

Atributos (1)

- La directiva @ **Control** habilita a introducir, entre otras, las siguientes propiedades:
  - **Inherits**: habilita a la clase o interfaz actual a heredar los atributos, propiedades, operaciones y eventos de otra clase o conjunto de interfaces.
  - **CodeFile**: ruta de acceso al archivo de código en el servidor al que se hace referencia para el control. Este atributo se utiliza junto con el atributo **Inherits** para asociar un archivo de código fuente a un control de usuario. El atributo sólo es válido para los controles compilados.
  - **Languaje**: lenguaje de Visual en el cual se codifica el control

Atributos (2)

- **EnableViewState:** indica si se mantiene el estado de vista entre solicitudes de control. Es **true** si se mantiene (valor por defecto) y **false** si no. El *view state* (estado de vista) es lo que el usuario ve en el formulario al momento de cargarse y en los posteriores *postback* al servidor.
- **AutoEventWireup** : indica si los eventos del control se conectan automáticamente. Es **true** si la auto-conexión de eventos está habilitada, en caso contrario, es **false**. El valor predeterminado es **true**.

Propiedades y Métodos

- Dentro de los controles de usuario se pueden generar propiedades y métodos de la misma forma que dentro de una página web o una clase.
- Tanto las propiedades como los métodos sirven para interactuar entre controles de usuario y para interactuar con la página en la cual se ha incluido el control de usuario.

Almacenamiento de Datos

- Como con las páginas web, el control de usuario se reinicializa con cada request (luego de su viaje al servidor se produce un refresh en el cliente).
- Por lo tanto se deben almacenar los valores de las propiedades en una ubicación persistente entre las request .
- Recordar que cada vez que un elemento del control ejecute un evento, el control viajará al servidor a atenderlo, y al igual que con las páginas, los valores que no se tengan almacenados se perderán.

Tipos de Almacenamientos

- Hay dos formas típicas para el almacenamiento de los valores de las propiedades:
  - **VIEWSTATE:** Es el más recomendado ya que es una variable del ámbito de petición de una misma página (solo permitido para elementos serializables).
  - **SESSION:** Ésta mantendrá los valores dentro de la instancia de usuario y se podrán ver los valores de las propiedades en diferentes momentos y lugares de la aplicación, para la misma sesión de usuario.

Controles Anidados

- Los controles de usuario permiten ser anidados unos dentro de otros:
  - Facilita la reutilización de código.
  - Al dividir hacia arriba la funcionalidad de los controles de usuario independientes, el desarrollo de los controles puede beneficiarse de volver a utilizar el mismo código de un control específico para un problema específico.
  - Sirve para mantener un bajo acoplamiento de los problemas a resolver.

Eventos (1)

- Si dentro de un control de usuario existen controles de servidor, se pueden desarrollar controladores de eventos para éstos (**TextBox**, **Calendar**, etc.)
- Por defecto, los eventos producidos por los controles secundarios dentro de un control de usuario no están disponibles para la página host.
- Es posible definir en el control de usuario eventos y producirlos de modo que se notifiquen a la página host, para que ésta se encargue de su atención.

Eventos (2)

- El ejemplo siguiente muestra la manera de enlazar el evento Click del control “BtnEjecuto” a un método denominado “MiMetodo”:

```
BtnEjecuto.Click += new
        System.EventHandler(this.MiMetodo);
```

- Esto producirá que cada vez que el usuario realice la acción Click sobre “BtnEjecuto”, se dispare el método “MiMetodo”.

Eventos (3)

- Estos controles también provocan eventos de ciclo de vida en cada paso del procesamiento (como **Init**, **Load** y **PreRender**) los cuales se pueden implementar.
- De manera predeterminada, dichos eventos se pueden enlazar a métodos utilizando la convención de nombres **Page_*nombreDeEvento***.
- Por ejemplo, con el fin de crear un controlador para el evento **Load** del control, se puede crear un método denominado **Page_Load**.

Acceso al Contenedor

- Suponer que se tiene una propiedad denominada **MiNombre** dentro de la página contenedora llamada **MiPagina** y se quiere consultar su valor desde el control de usuario:

```
protected void MiMetodo()
{
    MiPagina pagina = (MiPagina)this.Parent;
     string nombrePagina= pagina.MiNombre;
}
```

Utilización

- Se debe utilizar la directiva **@Register** para definir al control dentro de una página. Sus principales atributos son:
  - **TagPrefix**: Asocia un prefijo al control de usuario.
  - **TagName:** Asocia un nombre arbitrario que se asocia a una clase. Este atributo sólo se utiliza para los controles de usuario.
  - **Src**: Ubicación (relativa o absoluta) del archivo declarativo del control de usuario que se asocia. Sirve para ubicar el control de usuario.

Custom Controls

Introducción

- Un control personalizado es una clase escrita por un desarrollador que puede heredar tanto de la clase **Control** como de la clase **WebControl**.
- A diferencia de los Controles de Usuario, los controles personalizados no sólo se pueden utilizar en un proyecto, sino que su alcance se extiende a varios proyectos. Esto es gracias a que se define en un componente de tipo biblioteca de clases, ya que su definición esta contenida dentro de una clase.

Clase Control

- Es la clase principal de la que se debe derivar para desarrollar controles personalizados, que no tiene una interfaz grafica propia, o que combina otros controles que generan sus propias interfaces de usuario
- Esta clase permite implementar propiedades y métodos para el desarrollo de controles.
- Se encuentra en el espacio de nombres **System.Web.UI**

Clase WebControl

- Clase que hereda de **Control**, proporciona las propiedades, métodos y eventos para controlar la apariencia y el comportamiento, que comparten todos los controles de servidor.
- Para controlar la apariencia y el comportamiento de un control de servidor se establecen las propiedades definidas en esta clase y éstas se trasladan a todo el control
- Se encuentra en el espacio de nombres **System.Web.UI**

Composite Controls

Definición

- Un Composite Control utiliza los controles hijos (es decir contenidos) para crear la interfaz de usuario y generar la lógica, por lo que relega su funcionalidad en los controles hijos (y sus propias funcionalidades).
- Es más fácil desarrollar este tipo de control que implementar todas las funcionalidades de los controles manualmente.
- Se utilizan:
  - Delegados.
  - Creación manual de eventos.
  - Agregación de controladores de eventos a controles contenidos.

Elementos Básicos

- Se define como una clase, la cual debe heredar comportamiento de ***WebControl*** (para poder desplegarse dentro de una pagina web); y debe implementar la Interface ***INamingContainer*** (permite identificar a un control contenedor dentro de la jerarquía de controles de una pagina web).
- Se debe sobrescribir el método ***CreateChildControls***, para poder crear los controles contenidos dentro del nuevo control.
- La interfaz grafica de este tipo de controles no es visible hasta el momento de ejecución, que es cuando realmente se crean los objetos contenidos y que le brindan dicha interfaz.