3Controles Reutilizables

Son los que nosotros vamos a realizar (vamos a codificar) para lo cual lo primero que tenemos que tener en cuenta es:

Si ya no existe un control que haga lo que nosotros queramos.

Lo otro a tomar en cuenta es que tipo de funcionamiento queremos que tenga, esto es queremos que todos los objetos creados de ese control hagan exactamente lo mismo o no, en caso de que todos hagan lo mismo le generamos operaciones.

Ahora si queremos que su comportamiento varie en cada objeto creado, tenemos que generar eventos (esto se define en la clase) cuestión que cada vez que tenga un objeto de ese control pueda asignarle un controlador diferente y por lo tanto actúe diferente .

Son 3 las formas de hacerlos:

**User control**

Para crear este tipo de controles no importa si es para web o windows (cambia el nombre de clase en windows) lo que tengo que agregar al proyecto es un elemento llamado control de usuario (son clases que heredan siempre de la clase user control no importa si es para web o windows) La diferencia estriba de donde proviene esa clase en el caso de **web** es **System.WebControl** y en el caso de **Windows System.Winforms.**

En el caso web el user control tiene que estar incluido dentro del proyecto web (de lo contrario no lo va a reconocer)

En el caso de windows podemos guardar el UserControl dentro de una biblioteca de controles

El armado de estos controles la parte gráfica (web o windows) la vamos a armar usando controles ya existentes recordar que el espacio que ocupe la parte gráfica (los dibujos de los controles).

El web es lo que ocupa en el formulario sin embargo cuando lo diseñamos en windows el espacio real no el ocupado es lo que va a tener impacto en el formulario

La parte gráfica se sustenta de otros controles(los que ponemos nosotros)

El comportamiento de UserControl tanto en web como en Windows se sustenta en lo que nosotros hacemos en la clase (es el código de la clase del controlador podemos crear los 5 elementos tal cual una clase)

Cuando colamos un user control dentro de un formulario para ver las propiedades que nosotros creamos simplemente vamos a la ventana de propiedades(se van a mezclar con las que vienen predefinidas) y si usamos el botón del rayito vamos a ver los elementos que le definimos (de este vamos a hacer 3 ejemplos)

**Custom control**

En este caso lo que vamos a tener siempre es una clase esto significa que vamos a tener un solo archivo .cs

La parte gráfica se maneja pura y exclusivamente sobre el código.

La primera forma es crear un controlador desde 0, en este caso en web vamos a heredar la clase de webcontrol y si estoy en windows vamos a heredar de control esta herencia nos da los elementos básicos estos van a ser atributos propiedades y operaciones que son comunes a todos los controles (se hereda el true o false(visible, enable, etc) el comportamiento lo escribimos nosotros)

Para crear la parte gráfica tenemos que usar el constructor en caso de que sean WEB OBLIGATORIAMENTE tenemos que implementar la operación Render (ponerle codigo osea definir que tipo de etiquetas HTML van a representar a mi control cuando se renderiza la página)En el caso de **windows** lo que tenemos que hacer es sobreescribir la operación **onPaint** (es donde realmente se dibujan las cosas en pantalla)

Otra forma es creando otra clase nueva pero esta hereda de un control ya existente (text box drop down list etc etc) la primer ventaja la parte grafica ya esta resuelta (esto va con ejemplo)

Cualquiera de los 2 tipos se debe generar dentro de una biblioteca de controles porque son clases

Porque usar una **biblioteca de clases con controles adentro**?

1 - Porque se reutilizan fácilmente sin exponer código

2- Por la forma de poner los controles en la parte gráfica,

Cuando se agrega una biblioteca a un proyecto automáticamente visual hace es leer el manifiesto de la biblioteca (este se crea cuando leemos una biblioteca) y nos dice que elementos públicos accesibles hay, si encuentra una definición de controles cuando lee el manifiesto automáticamente en el cuadro de herramientas crea una nueva bandita con el nombre de la biblioteca(adentro pone un ítem por cada control que encuentre)

Cuando hacemos referencia pueden dar 2 errores

1- que no aparece nada en el cuadro de herramientas, esto significa que la biblioteca no compilo

2- Cuando usamos el control desde el cuadro de herramientas y lo vamos a tirar al pagina nos da el símbolo de prohibido esto quiere decir que está private en la biblioteca

Controles Composite

Es un híbrido entre UserControl y CustomControl

User control de la parte gráfica se basa en los controles que contengan. la diferencia es que tenemos que crearlos en variables (en vez de tirarlos)

CustomControl se hace a partir de una clase esto significa que todas las modificaciones gráficas que quiera hacer las voy a tener que codificar

En caso que esté tratando de hacer un control composite para web heredó de la clase webControl y tenemos que escribir cómo se van a mostrar todos los controles contenidos en la operación **CreateChildControls**, aca tambien se pueden poner etiquetas HTML para que tengan un formato visible

En caso de que estemos haciendo uno para Windows heredamos de la clase ContainerControl y tenemos que la operación OnPaint definir gráficamente donde queremos que vayan los control definidos

Todo esto va en un biblioteca

**Manejo de memoria en un sitio web**

Genérico a específico ==> Application - Session - ViewState

**Application**

Es un conjunto de datos que se guarda en la memoria del servidor y que puede ser accedida por cualquier usuario en cualquier momento (memoria a temporal) (General: Para todos los usuarios)

**Session**

Es un espacio físico que se tiene en memoria pero que es totalmente temporal esto significa que el usuario se desconecta del sitio y se borra y es individual por el usuario esto nos sirve para intercambiar datos entre las páginas que usa un mismo usuario

**ViewState**

Este no se almacena en la memoria del servidor texto que viaja con la página y desde el cliente hacia el servidor (este lo usamos sin darnos cuenta los controles) esto se genera en xml.

Por lo que yo puedo guardar en el viewstate solamente elementos que hayan sido definidos como serializados es decir que ellos solamente puedan traducirse al xml,por ahora no hay ninguna clase serializable (por lo que no puedo guardar un objeto de entidades compartidas en viewstate

Sin embargo los tipos de datos básico de netFramework ya vienen marcados como serializables, también son vienen serializables los objetos para el manejo de una base de datos

este siempre se castea (convert es mejor que casteo) y este siempre se refresca cada vez que cambia un valor

dentro del espacio system.drawing vamos a encontrar todas las funcionalidades necesarias para manejar las cosas gráficas Font (letras) Color(colores) image (imagenes) y demas

la clase color tiene un conjunto de constantes las cuales representan el concepto de color (no son objetos color son constantes)por lo cual para modificar una propiedad de color necesito un objeto color

la clase color tiene una operación de clase FromName que dada una de esas constantes genera y me devuelve un objeto color (las constantes tienen que ser las existentes)’

Todos los controles y eso incluye a las páginas poseen la propiedad control estos son un conjunto genérico con puntero a control(no tiene especificación de tipo) que contiene todos los controles directamente incluidos (esto significa que cada vez que saque algo de esa colección voy a tener que castear) por otro lado todo control posee una propiedad **Parent** que es un puntero genérico (tipo control) hacia el control y/o página que lo contiene vamos a poder desde un user control a los controles contenidos en una página

el botón aplicar host va a buscar 3 controles en la página usercontrol para cambiarles el control

la propiedad findcontrol dado el nombre identificatorio del control me lo busca en la colección control en caso de que encuentre un control devuelve el puntero en caso de que no encuentre nada devuelve null ( no da excepción)

**Directivas de Pre Compilación**

Las directivas de pre compilación las vamos a encontrar en el código html y sirven para identificar de qué tipo de elemento estamos hablando (esta es la primera línea y tiene que estar antes de la primera etiqueta de html)

Sin una directiva de pro compilación el compilador no sabe con qué elemento está trabajando y extensiones de elementos no ayudan aca.

NO sabe frente a qué elemento está.

El hecho de borrarla hace que no compile

Estas directivas van entre etiquetas de código embebido de ASP <%@ %>

este codigo tiene que ser resuelto antes de la renderización de la página dentro de estas etiquetas tenemos 2 símbolos:

**@** esto indica que es una directiva de pre compilación osea identifica el tipo de documento web que tenemos presente

**#** este significa que tengo que resolver el código y mostrar en ese lugar el resultado final (esto significa que hay va un valor)

Cuando hay una directiva **@Page** me indica que estoy frente a un formulario web (osea voy a tener controles y lógica) hay 2 tipos de páginas:

La primera es la página web convencional que se muestra sola o puede representar una página de contenido (Content page éstas se muestran dentro de una masterPage).

Para saber a qué tipo de pagina estoy hay que fijarse si la directiva de pre compilación posee el atributo **MasterPageFile** si solo tenemos el atributo **page** estamos frente a una pagina comun.

**@Master** esta nos dice que estamos frente a una página maestra la cual no se va a poder mostrar sola en el navegador

**@Control** me indica que lo que estoy haciendo es un usercontrol el cual obviamente no se muestra solo en el navegador

Estas 3 primeras son las que se ejecutan estas no pueden desaparecer ni duplicarse (Page - Master - Control).

**@Registrer** es una directiva de pre compilación que no puede ir sola porque lo que representa es la definición de un control de usuario (uno personalizado hecho por nosotros) La definición no está en la página de definición ni en ninguna biblioteca de net framework

@Registrer no es nunca la primera línea !,no hay límite para registrer

Etiquetas básicas a saber de HTML

<HTML>

Es la primer etiqueta que tiene que aparecer para poder escribir HTML esta es única por página,no se pueden anidar una dentro de la otra.

<HEAD>

Representa el cabezal de la pagina,titulo de una pagina etc etc,solo podemos tener una sola dentro de la etiqueta de HTML,estas no se duplican.

<BODY>

Es única dentro del html y dentro de la pagina,aca va todo lo visible, todo lo statico o dinamico, lo que quieras que se vea en la página del cliente uno solo por página

<FORM>

Es la etiqueta donde va lo dinámico,osea los controles, no necesariamente tienen que ser únicos lo importante es no anidarlos

Las 4 etiquetas pueden estar solamente presentes en una directiva de página común o de masterpage no puede estar ninguna de ellas content page ni en un control de usuario

Tanto la página de contenido y el control para usarse deben colocarse dentro de otro elemento que ya tiene las etiquetas (la apgina de