Tema 7; .NET y Primeros programas con C#

.NET

-permite el desarrollo de aplicaciones software y librerías
-contiene el compilador y las herramientas necesarias para construir, depurar y ejecutar estas aplicaciones
-independiente del lenguaje
-permite combinar código escrito en diferentes lenguajes
-No está orientado a un Hardware/Sistema Operativo concreto

Visual Studio .NET

-Entorno de desarrollo para desarrollar aplicaciones que se ejecutan sobre el .NET Framework.
-Simplifica la creación e implantación de Aplicaciones y Servicios Web y Aplicaciones basadas en Windows

Common Language Specification (CLS)

-mínimos estándares que deben satisfacer los lenguajes y desarrolladores si desean que sus componentes y aplicaciones sean ampliamente utilizados por otros lenguajes compatibles con .NET.
-permite crear aplicaciones con la seguridad de que no habrá problemas con la integración de los diferentes lenguajes
-permite heredar de clases desarrolladas en lenguajes diferentes.

.NET Framework

-Es el motor de ejecución
-Proporciona un conjunto de servicios comunes para los proyectos generados en .net, con independencia del lenguaje.
-Extensible
-La jerarquía del .NET Framework no queda oculta al desarrollador
-Herencia y Herencia multilenguaje
-código está organizado en espacios de nombres jerárquicos y clases
-Sistema de tipos común,denominado sistema de tipos unificado, que utiliza cualquier lenguaje compatible con .NET. En el sistema todo es un objeto

Colecciones de datos

-Existen 3 tipos principales de colecciones: ICollection, interfaz IList e interfaz IDictionary
-las colecciones de tipo IList (y las directamente derivadas de ICollection) solo almacenan un valor, mientras que las colecciones de tipo IDictionary guardan un valor y una clave relacionada con dicho valor
-IList se utiliza para acceder a los elementos de un array mediante un índice numérico
-Existen 3 tipos de colecciones que implementan esta interfaz: sólo lectura (no se pueden modificar. Basadas en ReadOnlyCollectionBase), de tamaño fijo (no se pueden quitar ni añadir elementos, pero sí modificarlos) y las de tamaño variable (permiten cualquier acción. Son la mayoría)
-ArrayList, CollectionBase y StringCollection implementan IList
-ArrayList representa una lista de datos, puede modificar su tamaño dinámicamente. Empieza en 0 y se añaden elementos consecutivamente
-using System.Collections ArrayList arraylist = new ArrayList();
-El constructor de ArrayList también puede utilizar un parámetro entero indicando el tamaño del objeto
-Add añade el elemento en la última posición
-Insert lo inserta en una posición indicada
-remove elimina el objeto pasado como parámetro
-removeAt elimina el elemento en la posición especificada
-RemoveRange elimina un grupo de elementos
-Todos los objetos de ArrayList son tratados como objetos
-a diferencia de los array no todos los elementos deben ser del mismo tipo
-Count sirve para conocer la cantidad de elementos que contiene
-Capacity indica la capacidad máxima de la colección
-si Capacity devuelve menos que Count se producirá la excepción ArgumentOutOfRangeException
-Si es un escenario donde no conocemos el tamaño que tendrá la colección y si además será muy probable que el tamaño varíe, entonces será recomendable bajo todas las demás circunstancias usar un ArrayList
-para escenarios donde se conoce de antemano la cantidad total de elementos a almacenar y si todos son del mismo tipo, se debe usar un array normal

Common Language Runtime (CLR)

-es el núcleo de la plataforma .NET
-gestiona la ejecución de las aplicaciones desarrolladas
-ofrece servicios que simplifican el desarrollo la fiabilidad y seguridad de las aplicaciones.
-Ejecución multiplataforma, actúa como una máquina virtual, cualquier plataforma para la que exista una versión del CLR podrá ejecutar cualquier aplicación .NET.
-Existe para todas las versiones de Windows
-para sistemas Unix existe Mono
-la arquitectura del CLR es abierta.
-La integración de lenguajes provee que es posible escribir una clase en C# que herede de otra escrita en Visual Basic.NET que, a su vez, herede de otra escrita en C++ con extensiones gestionadas.
-El CLR incluye un recolector de basura que evita errores de programación muy comunes (Intentos de borrado o acceso de objetos ya borrados o agotamiento de memoria
-comprueba que toda conversión de tipos que se realice durante la ejecución de una aplicación .NET se haga de modo que los tipos origen y destino sean compatibles
-los errores se propagan mediante excepciones

Microsoft Intermediate Lenguage (MSIL)

-Los compiladores de .NET generan código escrito en el lenguaje intermedio conocido como Microsoft Intermediate Lenguage (MSIL)
-Incluye instrucciones que permiten trabajar directamente con objetos, tablas y excepciones

Sistema de Tipos Comunes (CTS)

-define un conjunto de tipos de datos predefinidos que se pueden utilizar para definir variables
-es una parte integral del runtime de lenguaje común y es compartido por los compiladores, las herramientas y el propio runtime
-define las reglas que sigue el runtime a la hora de declarar, usar y gestionar tipos
-establece un marco que permite la integración entre lenguajes, la seguridad de tipos y la ejecución de código con altas prestaciones
-admite tanto tipos de valor (datos directamente, copias en cada variable, operaciones sobre la variable) como de referencia (referencian a objetos, 2 variables pueden referenciar el mismo objeto, las operaciones pueden afectar a varias variables)
-Con signo: sbyte<short<int<long
-Sin signo: byte<ushort<uint<ulong
-float(prec. simple)<double(prec. doble)<decimal(prec. alta)
-un float debe inicializarse con una F al final de la parte decimal (numero = 28.67F)
-por defecto un número no entero es double
-los tipos bool no pueden ser entero o viceversa

C#

-diseñado para .NET
-Diseñado desde cero sin ningún condicionamiento
-primer lenguaje moderno orientado a componentes de la familia de C y C++ (y Java)
-Fácil de integrar con Visual Basic
-alto rendimiento
-permite el acceso a memoria de bajo nivel de C++
-Puede incrustarse en páginas ASP.NET
-Incluye clases, interfaces, delegados,espacios de nombres, propiedades, indexadores,eventos, sobrecarga de operadores, versionado,atributos, código inseguro
-documentación en formato XML ....
-Una aplicación C# es una colección de clases, estructuras y tipos
-Los comentarios son importantes. De una sola línea
-C# es un lenguaje de especificaciones seguras (type--- safe), lo que significa que el compilador de C# garantiza que los valores almacenados en variables son siempre del tipo adecuado
-El compilador C# exige que cualquier variable esté inicializada
-C# es mucho más seguro que C++

-VB inicializa a 0 por defecto las variables
-El valor de la constante se calcula en tiempo de compilación con variables del tipo readonly

Declaración de variables
-Usa letras, el signo de subrayado y dígitos
-Recomendaciones: Evita poner todas las letras en mayúsculas, Evita empezar con un signo de subrayado, Evita el uso de abreviaturas, Use PascalCasing (EstaEsUnaVariable) para nombres con varias palabras
-CamelCasing es lo mismo que PascalCasing, pero con la 1a letra en minúscula (estaEsUnaVariable)
-utilizar PascalCasing variables públicas y camelCasing para privadas
-El ámbito de una variable coincide con su clase contenedora
-Las constantes con la palabra const y se deben inicializar obligatoriamente en la declaración

Conversión de tipos
-Es implícita. Automática, cuando no hay posible pérdida de información
-Explicita. Se debe indicar que se desea realizar una conversión en la que puede haber pérdida de información. Se utiliza casting

Main
-debe ir con 'M'
-es el punto de entrada al programa
-se declara como static void Main
-puede pertenecer a múltiples clases
-La aplicación termina cuando Main acaba o ejecuta un return

La Clase

-es un conjunto de datos y métodos
-Una aplicación C# puede incluir muchas clases
-Se hace referencia a clases por su espacio de nombres mediante la sentencia using

Estructuras de control
-condicionales simples: if/else
-condicionales múltiples: switch/case
-repetición: for/foreach, while/do
-cambio de secuencia: return, break, continue
-foreach permite recorrer todos los elementos de un contenedor con una variable del tipo de los elementos del contenedor
-Return devuelve el control a la rutina llamante actual
-Break sale de la actual estructura de bucle
-Continue obliga a ejecutar la siguiente iteración del bucle

Excepciones
-Es un objeto que se crea cuando se produce una situación de error específica
-el objeto contiene información que permite resolver el problema
-se utiliza try/catch. Try contiene el código normal del programa, Catch contiene el el código de los errores
-finally tiene el código que libera los recursos. Es opcional
-funcionamiento: ejecuta el código de try, si hay error va al catch, si no sigue el programa o salta al bloque finally si es que lo hay

Tema 8. Modelo de capas

arquitectura de 3 capas

Interfaz de usuario, componentes que interactúan con el usuario final
Lógica de negocio, contienen las reglas de negocio de nuestra aplicación
Persistencia, contienen el acceso y almacenamiento de los datos

Entidades de Negocio (EN)

Componentes que representan entidades de negocio del mundo real.
Contienen normalmente la información de una clase de dominio con sus atributos, operaciones y restricciones. Aunque pueden representar una composición de clases
Tienen asociado un CAD que le proporciona el acceso y el mapeo a los datos
Pueden ser representados de múltiples maneras.

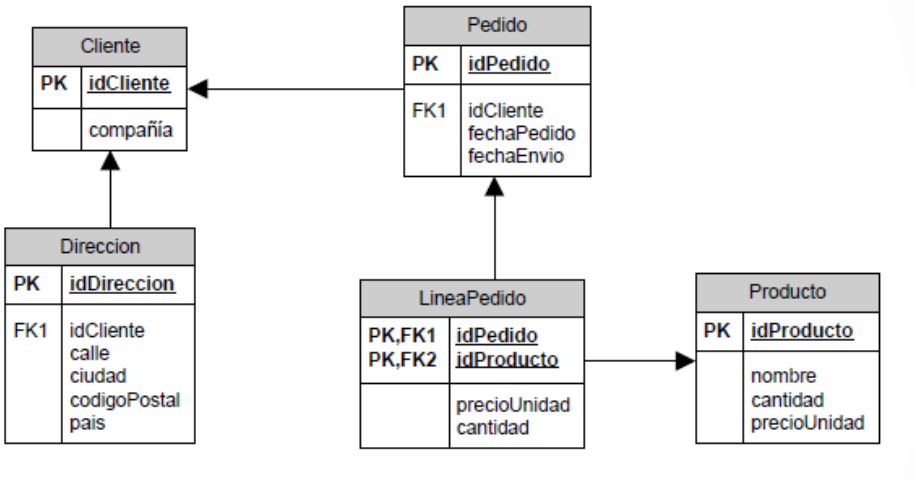
Componentes de Acceso a Datos (CAD)

Los CADs encapsulan la tecnología de acceso a datos y la BD al resto de la aplicación
Permite recuperar los datos y salvar una entidad de negocio
Los CADs contienen lógica de negocio para alcanzar las operaciones relacionadas con los datos
Un CAD debería proveer los métodos para realizar tareas sobre la
BD: Crear registros en la BD, Leer registros en la BD y devolver las entidades de negocio, Actualizar registros en la BD, usando entidades de negocio, Borrar registros de la BD
Estos métodos son llamados CRUD, acrónimo de "Create, Read, Update and Delete"
pueden contener también métodos que realizan algún filtro.
accede a una única BD y encapsula las operaciones relacionadas con una única tabla o un grupo de tablas vinculadas de la BD
pueden Controlar la seguridad y autorización
pueden Realizar la paginación de datos
pueden Realizar Transacciones de entidades complejas
pueden Invocar a procedimientos almacenados

De Relacional a Entidad de Negocio

Una BD contiene múltiples tablas con relaciones y debemos decidir como mapear las tablas en diferentes EN
Cuando se define las EN se debe considerar "como" se usará la información en la aplicación
Es mejor identificar el núcleo de EN que encapsulan la funcionalidad de la aplicación, antes que definir una EN por cada tabla

Ejemplo



Las requisitos funcionales mínimos de una tienda son:
• Obtener información sobre el Cliente, incluyendo sus direcciones

• Obtener la lista de pedidos para un cliente
• Obtener la lista de artículos para un pedido en particular
• Enviar un nuevo pedido
• Obtener o actualizar la información de un producto o colección de productos

Para completar estos requisitos, podemos hacerlo definiendo tres EN lógicas que controlen la aplicación:
• Un Cliente que contendrá sus direcciones
• Un Pedido que contendrá sus líneas de pedido
• Y un Producto

Para cada EN, definimos un CAD que será definido como:
• ClienteCAD: Esta clase provee los servicios para recuperar y modificar los datos de las tablas Cliente y Dirección
• PedidoCAD: Esta clase provee los servicios para recuperar y modificar los datos de las tablas Pedido y LineaPedido
• ProductoCAD: Esta clase provee los servicios para recuperar y modificar los datos de la tabla Producto

Recomendaciones

Básate en las composiciones y herencias UML para componer objetos complejos
No definas EN separadas para representar tablas muchos- a-muchos. Estas relaciones pueden ser implementadas mediante colecciones.
Definir todos los métodos que devuelven un tipo concreto de Entidad de Negocio en un solo CAD

Tema 9; Aplicaciones Web y Capa de Interfaz

Aplicaciones web vs escritorio

Simple creación de interfaz de usuario.
Distribución de actualizaciones más fácil y rápido y menos costoso.
Procesamiento distribuido.
La web provee protocolos estándar (HTTP, HTML, XML) para facilitar aplicaciones n-capa.
Tecnología dependiente del servidor
La ventaja principal radica en la seguridad que tiene el programador sobre su código

Qué es ASP.NET?

Plataforma de para construir Aplicaciones Web y Servicios Web que funcionan bajo IIS.

Aplicaciones Web ASP.NET

Combinación de archivos, páginas, manejadores, módulos y código ejecutable que puede invocase desde un directorio virtual.
Se dividen en varias páginas web.
Comparten un conjunto de recursos y opciones de configuración común.
Una aplicación web sólo existe en un Directorio Virtual, es un recurso compartido identificado por un alias que representa la localización física en el servidor.

Formularios Web

Técnicas para Rápido Desarrollo de Aplicaciones (RAD).
La aplicación se desarrolla para un servidor web.
Los usuarios interactúan con la aplicación a través de un navegador.
Proporcionan una aproximación orientada a: Objetos, eventos y gestión de estado
Programación del lado del servidor para manejar eventos del lado del cliente: pueden ejecutarse, virtualmente, sobre cualquier navegador compatible con HTML
Visual Studio dispone de su propio servidor de desarrollo, por lo que no necesitamos tener instalado IIS (Internet Information Server).

Code-inline vs Code-behind

Único archivo (“Code-inline”)
Archivos separados (“Code-behind”)

Páginas maestras

Las páginas maestras permiten crear un diseño coherente para las páginas de la aplicación.
Se puede definir el aspecto, el diseño y el comportamiento estándar que desea que tengan todas las páginas (o un grupo de páginas) de la aplicación en una sola página maestra.
Realización de cambios de diseño en una sola ubicación.
Reutilización de la interfaz de usuario
Master pages definen el contenido común y los contenedores de contenido, hacen referencia a las paginas maestras y llenan a los contenedores con su contenido.
Desde la perspectiva del usuario, da como resultado una única página.
La dirección URL de esta página es la de la página de contenido.

Secuencia de control IIS (En tiempo de ejecución)

por primera vez se carga la maestra y la página de contenido
por segunda vez ya no hace falta compilar la página maestra solo la de contenido, la página maestra se va actualizando en las demás de contenido.

Controles de servidor

Tienen propiedades que pueden ser establecidas declarativamente o mediante programación.

Soporte para características avanzadas: enlace de datos, plantillas..
Se emplea el prefijo asp: junto con el atributo runat="server".

Tipos de controles ASP.NET

Label	Se utiliza para mostrar texto dinámico (cambiamos sus propiedades a través del código del servidor)
Hyperlink	Muestra un enlace a otra página
TextBox.	Permite introducir texto al usuario. Propiedad Textmode valores: Single, Multiline o Password
Image.	Sirve para mostrar una imagen en la página web
Button .	Se usa en un Formulario web para crear un control de tipo submit (evento OnClick) o un control de comando tipo botón (evento OnCommand)
LinkButton	Apariencia de hipervínculo pero funciones de control de un botón (envío formulario)
ImageButton.	Muestra una imagen que maneja eventos tipo click
Checkbox.	Sirven para añadir casillas de verificación a una página
RadioButton.	Crea un botón de radio individual en la página
DropDownList.	Proporciona un buen método para que los usuarios elijan elementos de una lista en un espacio pequeño
ListBox.	Propiedad SelectionMode: Single, Múltiple. Cada elemento se crea con un control Listltem
CheckBoxList.	Permite presentar una lista de opciones pudiendo el usuario seleccionar varias
RadioButtonList.	Permite crear una lista de botones de radio de opciones excluyentes
Panel .	Puede usarse como contenedor de otros controles
Table, TableRow, TableCell.	Permiten crear dinámicamente una tabla mediante programación

Modelo de eventos

Un evento es un mensaje que envía un objeto cuando ocurre una acción.
En una aplicación, se debe traducir el mensaje en una llamada a un método del código.
En las páginas Web ASP.NET, los eventos asociados a los controles de servidor se originan en el cliente pero los controla la página ASP.NET en el servidor Web.
La información del evento se captura en el cliente y se transmite un mensaje al servidor mediante un envío HTTP.
La página debe interpretar el envío para determinar el evento ocurrido y, a continuación, llamar al método apropiado del código del servidor para controlar dicho evento.
.NET Framework define un tipo especial que proporciona la funcionalidad de un puntero a función.

Objeto evento

El objeto del evento se conoce como remitente.
El objeto que captura el evento es el receptor.
En las comunicaciones de eventos, el remitente del evento no sabe qué objeto o método recibirá los eventos que provoca.
Se necesita un intermediario entre el origen y el receptor.
El enlace entre el mensaje del evento y un método específico se lleva a cabo utilizando un delegado de eventos.

Clase delegado

es una clase que puede guardar una referencia a un método

tiene un prototipo y puede guardar referencias únicamente a los métodos que coinciden con su prototipo.
equivale a un puntero a función con seguridad.
los delegados de evento de .NET Framework tienen dos parámetros, el origen que provocó el evento y los datos del evento
Los delegados personalizados sólo son necesarios cuando un evento genera datos de evento.
Los delegados son de multidifusión, pueden guardar referencias a más de un método de control de eventos.

Manejadores de eventos

El prototipo de los métodos manejadores de eventos deben coincidir con el prototipo del delegado EventHandler.
los manejadores de evento en ASPnet devuelven void
tienen dos parámetros:Objeto que lanza el evento y Argumentos del evento(información específica del evento)
Crear una instancia del delegado utilizando una referencia al método controlador de eventos.
Cuando se llama a la instancia de delegado, ésta, a su vez, llama al método controlador de eventos.
Agregar la instancia de delegado al evento: cuando se provoca el evento, se llama a la instancia de delegado y a su método de controlador eventos asociado.

Asociar el manejador al control

Seleccionar el evento de la ventana de propiedades del control
se hace cuando se están creando controles mediante programación.
crear una instancia del delegado EventHandler, a la que se debe pasar la dirección del método al que se va a enlazar.
Después agregar el objeto delegado a la lista de métodos a los que se llama cuando se produce el evento.

Tipos de eventos

Eventos de envío (POSTBACK). Son los eventos lanzados por controles que emiten un envío (post) al servidor de manera inmediata para procesar el Web Form.
Eventos de caché (NO-POSTBACK). Estos eventos se producen en la vista y serán procesados en el servidor cuando se envíe la información mediante un evento de envío.

Eventos postback vs no-postback

Una página se carga después de cada petición: fenómeno conocido como PostBack (=envío).
Eventos no-Postback (o cached): la información se envía en el siguiente evento postback.
Los eventos se guardan en una cola en el cliente hasta que un evento postback ocurre.
Button, Link Button y Image Button causan eventos postback.
TextBox, DropDownList, ListBox, RadioButton y CheckBox, proveen eventos cached.
Sin embargo podemos sobrecargar este comportamiento en los controles para poder realizar eventos postback, cambiando la propiedad AutoPostback a true.

Eventos de página

Las páginas ASP.NET provocan eventos de ciclos de vida como Init, Load, PreRender y otros.
De manera predeterminada, los eventos de página se pueden enlazar a los métodos utilizando la convención de nomenclatura Page_nombreDeEvento.
Las páginas ASP.NET enlazan automáticamente los eventos de páginas a los métodos que tienen el nombre Page_evento.
Este enlace automático lo configura el atributo AutoEventWireup de la directiva @ Page, cuyo valor predeterminado en true.Si establece en false, la página no busca automáticamente los métodos Page_evento
Importante: Los métodos de control de eventos de página no requieren ningún argumento. Estos eventos (como Load) pueden aceptar los dos argumentos estándar, pero en estos argumentos no se pasa ningún valor.

Maquetación con CSS

CSS es una herramienta que permite definir la presentación de un documento
Permite crear un conjunto de estilos en una única localización
Las páginas a las que se aplica la misma hoja de estilos tendrán las mismas fuentes, colores y tamaño Proporcionan una estética homogénea en todas las páginas de un sitio web
Maquetar una pagina web es pasar el diseño a código HTML
Cada Explorador renderiza de manera distinta cada documento HTML

Divs

Las capas, layouts o divs son la misma cosa con distinto nombre, para tener un concepto mental de lo que son, podemos imaginarlos como contenedores o bloques donde podemos meter lo que queramos dentro a los que se le asigna un ancho, alto y posición, de esta manera se van a ir posicionando consiguiendo la estructura que queremos.
Los elementos DIV pueden centrarse utilizando los atributos margin- left: auto; margin-right: auto;
La diferencia entre una clase (.) y un bloque (#) es que el bloque es único e irrepetible en la página, en cambio si lo convertimos en una clase podremos usarlo cuantas veces queramos.
En las estructuras clásicas con tablas, podíamos utilizar tamaños en porcentaje. Aunque aquí, también podemos utilizar porcentajes, el dibujado de la página puede no verse como se esperaba.

Método Response.Redirect

Navega a otra página por medio del código.
Los hipervínculos responden a eventos click mostrando la página especificada en la propiedad NavigateURL del control.
Si quieres capturar un click en el código, debes usar los eventos de un LinkButton o ImageButton y utilizai
Utilizar los métodos de la clase HttpServerUtility para codificar los datos
Response.Redirect("WebForm2.aspx?par1 ="+ Server.UrlEncode(“ &hola ”));

Paso de parámetros a en la URL. QueryString

Almacena los datos en la colección QueryString
Los caracteres utilizados deben ser caracteres permitidos en una URL
La información es visible a los usuarios en la barra del navegador
Los usuarios pueden modificar la información provocando errores inesperados
Muchos navegadores imponen un límite en la longitud de la URL

& (para separar múltiple query strings)
+ (alternativa para representar un espacio)
# (especifica un marcador en una página web) Solución:

UrlEncode

reemplaza los caracteres especiales por secuencias de escape

Objeto Request

proporciona información sobre la petición HTTP del cliente que ha provocado la carga de la página actual.
Información sobre el cliente (Request.Browser, Request.Browser.IsMobileDevice, Request.Browser.Id)
Cookies (Request.Browser.Cookies)
Parámetros pasados a la página:
if (Request.QueryString["par1"] != "") Labell.Text = Request.QueryString["par1"];

Tema 10. Capa de Interfaz II

Controles de Lista Sencillos

listBox
dropDownUst
radioButtonList
checkBoxList
dropDownList; Lista desplegable
RepeatDirection
RadioButtonList: lista de botones de radio
CheckBoxList: lista de opciones múltiple

ListItem

Text:Contenido visualizado
Value:Valor oculto del código HTML
Selected:true o false (seleccionado o no)

SelectedIndex

Indica la fila seleccionada como un índice que empieza en cero

SelectedItem

Permite que el código recupere un objeto ListItem que representa el elemento seleccionado

Controles de lista con selección múltiple

ListBox

Pueden seleccionarse varios elementos: propiedad

CheckBoxList

Siempre pueden seleccionarse varios elementos
Para encontrar todos los elementos seleccionados necesitamos: recorrer la colección Items del control lista y comprobar la propiedad ListItem.Selected de cada elemento

Control de navegación menu

Se puede utilizar para crear un menú que colocaremos en la página maestra y otras ayudas de navegación. También podemos añadir submenús y definir menús dinámicos.
Los elementos pueden añadirse directamente en el control o enlazarlos con una fuente de datos.
En las propiedades podemos especificar la apariencia, orientación y contenido.

Static Display y Dynamic Display

El control tiene 2 modos de "Display":
Estático: El control Menu está expandido completamente todo el tiempo.
Dinámico: En este caso solo son estáticas las porciones especificadas, los demás elementos se muestran al hacer click sobre el padre.

StaticDisplayLevels

El número de niveles estáticos que queremos mostrar como raíz del menú (el mínimo es 1).

MaximumDynamicDisplayLevels

Cuantos niveles de nodos que aparecen de forma dinámica se mostraran después del nivel estático.

Dynamic: cantidad de tiempo

Podemos especificar la cantidad de tiempo que queremos que tarde la parte dinámica de un menú en desaparecer.
Lo podemos especificar en milisegundos mediante la propiedad DisappearAfter del menú, el valor por defecto son 500 ms.

Definición del contenido de menu

Añadir objetos individuales MenuItem
se le puede enlazar un archivo XML.
Propiedades del control propiedad Items (colección de objetos MenuItem)

Validación de datos

Debemos asegurar que los usuarios introducen datos correctamente
ASP.Net proporciona un conjunto de controles de validación predefinidos
2 tipos de validación:
Lado del cliente: Utilización de código JavaScript que valida los datos introducidos por el usuario, directamente en el navegador
Lado del servidor: Utilización de código (C#) para validar los datos de formularios una vez han sido enviados al servidor

Controles de validación

ASP.Net detecta si el navegador soporta validación del lado del cliente
Generan el código JavaScript necesario para validar los Datos, en otro caso, los datos del formulario se validan en el servidor
Para mostrar el mensaje de error ErrorMessage, para indicar el control a validar ControlToValidate

Entrada requerida: RequiredFieldValidator: La validación es OK cuando el control de entrada no contiene una cadena vacía.
Coincidencia de modelos: RegularExpressionValidator:. La validación es OK si el valor de un control de entrada se corresponde con una expresión regular especificada.
Comparación con un valor: CompareValidator: La validación es OK si el control contiene un valor que se corresponde con el valor de otro control especificado.
Comprobación de intervalo: RangeValidator: La validación es OK cuando el control de entrada contiene un valor dentro de un intervalo numérico, alfabético o temporal especificado.
CustomValidaton: La validación la realiza una función definida por el usuario.
ValidationSummarv: Este control muestra un resumen con todos los mensajes de error de cada control de validación.
Propiedad display: Sirve para comprobar si el control de validación muestra mensajes de error.

Sintaxis Expresiones Regulares

* cero o más ocurrencias del carácter o subexpresión anterior.
+ una o más ocurrencias del carácter o subexpresión anterior
() agrupa una subexpresión que se trata como un único elemento
| Cualquiera de las dos partes (OR)
[ ] se corresponde con un carácter en un intervalo de caracteres válidos [a-c]
{n} exactamente n de los caracteres o subexpresiones anteriores

. cualquier carácter excepto el salto de línea
? el carácter anterior o la subexpresión anterior es opcional
^ comienzo de una cadena
$ fin de una cadena
\s carácter de espacio en blanco \S cualquier carácter no espacio
\d cualquier carácter numérico
\D cualquier carácter no dígito
\w cualquier carácter alfanumèrico (letra, número o carácter de subrayado)

Grupos de validación

Los controles de validación se pueden asociar en grupos de validación para cuando sean comunes se validen juntos
se pueden usar para habilitar o deshabilitar de forma selectiva la validación
hay que definir el nombre del grupo en los controles de validación y en el botón u otros controles de envío que causan validación.

SetFocusOnError

Se establece en controles de validación que causan que el primer control no válido reciba el foco

Objetos Session Application

Los objetos **Session** están asociados a un usuario particular y sirven como manera de transportar y mantener los datos del usuario.
Los objetos **Application** son compartidos por todos los usuarios y permiten almacenar información compartida por toda la aplicación web.
En ASP.NET los objetos Session y Application están implementados como colecciones o conjuntos de pares nombre-valor.

Qué es una sesión?

Una sesión es el período de tiempo en el que un usuario particular interactúa con una aplicación web. Durante esta la identidad única se mantiene internamente.
Los datos se almacenan temporalmente en el servidor.
Una sesión finaliza si hay un timeout o si tú la finalizas.

Cuál es el uso de una sesión?

Las sesiones ayudan a preservar los datos entre accesos sucesivos.
Se hace gracias a los objetos de sesión.
Los objetos de Sesión nos permiten preservar las preferencias del usuario y otra información del usuario al navegar por la aplicación web.

Objeto Session

sirve para almacenar datos pertenecientes a un único usuario .
En asp.net: Las sesiones son tablas Hash en memoria con un timeout especificado.están identificadas usando enteros 32-bit long conocidos como Session IDs.
El motor ASP genera estos session ID’s que son únicos
Al crear la parte privada de la Web utilizaremos variables de sesión para controlar si el usuario ha entrado logueandose o ha entrado poniendo directamente la URL, en cuyo caso la variable de sesión estará vacía y no deberíamos permitir el acceso
¿Inicializar variables que estén disponibles en una sesión y sean las mismas para todos los usuarios? Esto supone que un cambio en el valor de una variable de aplicación se refleja en las sesiones actuales de todos los usuarios.

Session.Abandon l	Abandona (cancela) la sesión actua
Session.Remove	Borra un elemento de la colección de estado de la sesión.
Session. RemoveAII	Borra todos los elementos de estado de la sesión.
Session.Timeout	Establece el the timeout (en minutos) para una sesión
SessionID	Recupera el ID de la sesión (propiedad de sólo lectura de una sesión) para la sesión.
Session. IsNewSession	Es para comprobar que la sesión del usuario se creó con la petición actual

Objeto Application

Application: proporciona una manera sencilla de almacenar en el servidor datos comunes a todos los visitantes de nuestro sitio web.

Global.asax

Permite escribir código de aplicación global
no contiene etiquetas HTML ni ASP.NET
se utiliza para definir variables globales y reaccionar a eventos globales
contiene código de tratamiento de eventos que reacciona a los eventos de aplicación o sesión
se añade a la aplicación web como un nuevo elemento
Clase de aplicación global
Cualquier cambio en el archivo global.asax reiniciará la aplicación ,Sólo puede haber un archivo global.asax, debe estar en la raíz.

Tema 11; Acceso conectado a BBDD

ActiveX Data Object

-ADO.NET es la tecnología que las aplicaciones asp.net utilizan para comunicarse con la BD.
-Optimizada para aplicaciones distribuidas (como aplicaciones web).
-Basada en XML
-Entorno conectado vs desconectado

Entorno conectado

-los usuarios están conectados continuamente a una fuente de datos
-es más fácil de mantener
-La concurrencia se controla más fácilmente
-datos estén más actualizados
-Debe existir una conexión de red constante
-Escalabilidad limitada
-Connection: se utilizan para establecer las conexiones al proveedor de datos adecuado (método Open).
-Command: sirven para ejecutar sentencias SQL y procedimientos almacenados.
-System.Data.OleDb y System.Data.SqlClient: clases responsables del acceso a datos desde fuentes SQL Server y OLE DB. Al trabajar con SQL llevarán el prefijo Sql y al emplear Ole DB llevarán OleDb
-Para recuperar datos de la BD necesitas: Una sentencia SQL, Un Command que ejecute la SQL y Un DataReader que capture los registros recuperados
-En caso de insertar, actualizar o borrar datos de la BD necesitas: Un Command para ejecutar la SQL
-ExecuteNonQuery obtiene el número de registros afectados. Ejecuta un comando y no devuelve ningún resultado
-try/catch para el manejo de excepciones

DataReader

-proporciona un flujo de datos firme
-un cursor de sólo lectura que avanza por los registros sólo hacia delante.
-Mantienen una conexión viva con el origen de datos, pero no permiten realizar ningún cambio.
-ExecuteReader Ejecuta un comando y devuelve un comando que implementa DataReader (Permite iterar a partir de los registros recibidos)
-.Read() recupera una fila. True si se ha recuperado una fila de información correctamente, False si hemos intentado leer después del final del conjunto de resultados
-Podemos acceder a los valores de una fila con el nombre del campo
-.Close() cierra el DataReader
-Más ligero y veloz que DataSet. Mejor elección para acceso a datos simple
-Recomendado para consultas de sólo lectura utilizadas una única vez (listar todos los artículos de un almacén)

Espacios de nombres

System.Data	
System.Data.Common	
System.Data.OleDb	Ms access, Oracle.. DB
System.Data.SqlClient	Ms SQL Server 7.0 DB
System.Data.SqlTypes	contiene clases para trabajar con tipos de datos nativos de SQL Server
System.Configuration	proporciona clases para trabajar con información de configuración almacenada en archivos de configuración.

Parámetros de conexión

Connection TimeOut	Define el tiempo de espera máximo que debe esperar una conexión para intentar conectar con éxito con el servidor de base de datos. En caso de superar este tiempo se genera una excepción. El tiempo por defecto definido es de 15 segundos.
Data Source	Recibe el nombre del servidor SQL Server utilizado en la conexión, o en caso de utilizar bases de datos de usuario Access el nombre del archivo utilizado.
Initial Catalog / Database	Nombre de la base de datos con la que vamos a trabajar.
Integrated Security	Configura nuestra conexión de un modo seguro o no. Recibe como valores True, False y SSPI, siendo True y SSPI el mismo modo de seguridad.
AttachDBFilename	Si se utiliza un nombre de archivo para conectar con la base de datos, se simplifica la implementación de la base de datos con la aplicación (especificamos el archivo mdf)
Persist Security Info	Si recibe el valor True, se devuelve la contraseña junto con la conexión si ha sido abierta o permanece abierta, esto supone un riesgo de seguridad, por lo que se suele dejar como está configurada por defecto, False.
Password	Contraseña para la identificación de inicio de sesión en SQL Server.
Provider	Utilizada únicamente para conexiones OleDbConnection, establece o devuelve el nombre del proveedor.
User ID	Nombre de usuario para el inicio de sesión en SQL Server 2005, login.

Proveedores
-SQLOLEDB: proveedor OLEDB de SQL
-MSDAORA: proveedor OLEDB para una bd Oracle
-Microsoft.Jet.OLEDB.4.0: proveedor OLEDB de Access

Comandos de conexión
-Crear una conexión

```
string s =
"datasource=.\\SQLEXPRESS;IntegratedSecurity=SSPI;AttachDBFilename=|DataDirectory|\\Database1.mdf;User Instance=true";

SqlConnection c=new SqlConnection(s);
```

-.Open() / .Close() abren / cierran la conexión con la BD

DataDirectory
-es una cadena de sustitución que indica la ruta de acceso de la base de datos.
-elimina la necesidad de definir la ruta de acceso completa. |DataDirectory|
-se establece en AppDomain llamando a AppDomain.SetData. AppDomain.CurrentDomain.SetData("DataDirectory", newpath);
-Si no se establece DataDirectory Para las aplicaciones que se coloquen en un directorio en el equipo cliente, la ruta de acceso de la base de datos será la carpeta en la que se coloque la aplicación. Para aplicaciones Web, se tendrá acceso a la carpeta App_Data.

web.config
-Archivo de configuración de la aplicación ASP.NET basado en XML
-Incluye las opciones del proyecto
- Para evitar almacenar cadenas en el código, se puede almacenar en el archivo web.config.
-La cadena de conexión se puede almacenar en el archivo de configuración en el elemento <connectionStrings> donde el nombre se puede utilizar para buscar el valor almacenado en el atributo connectionString en tiempo de ejecución

Tema 12; Acceso desconectado BBDD

Entorno desconectado

-los datos pueden modificarse de forma independiente y los cambios se escriben posteriormente en la base de datos
-Las conexiones se utilizan durante el menor tiempo posible
-menos conexiones den servicio a más usuarios
-mejora la escalabilidad y el rendimiento de las aplicaciones
-Los datos no siempre están actualizados, pueden producirse conflictos
-varios usuarios pueden modificar los datos de los mismos registros al mismo tiempo
- En modo desconectado SIEMPRE necesitamos recuperar los datos (SELECT) para poder trabajar con ellos (INSERT, UPDATE, DELETE)
-Para recuperar datos se necesita: Un DataAdapter que ejecute la SQL y un DataSet donde guardar el resultado
-Mejor si se va a trabajar con más de una tabla o más de una base de datos

DataSet

-representación de una base de datos relacional en memoria. Crea una BD virtual
-Almacena datos en un caché distinto de la base de datos
-No necesidad de conexión continua
-permite trabajar con una copia de las partes de la BD con las que queremos trabajar, liberando la conexión
-System.Data.OleDb y System.Data.SqlClient NO proporcionan clases para DataSet
-Recomendado para acceso a datos complejo; recibir y almacenar datos (guardar artículos por tipo de un proveedor)

DataAdapter

-DataAdapter se utiliza para insertar datos en un DataSet
-Utiliza comandos para actualizar el origen de datos después de hacer modificaciones en el objeto DataSet
-Abre y cierra la conexión automáticamente
-CommandBuilder recibe un DataAdapter y crea los comandos SQL para actuar sobre la BD
-System.Data.OleDb y System.Data.SqlClient proporcionan clases para DataAdapter y CommandBuilder

DataRow

-Representa una única fila de información de la tabla
-Se puede acceder a los valores individuales usando el nombre de campo o un índice

DataColumn

-No contienen ningún dato real.
-Almacenan información sobre la columna (tipo de datos, valor predeterminado, restricciones..)

DataRelation

-Especifica una relación padre/hijo entre dos tablas diferentes de un DataSet

DataView

-Proporciona una vista sobre una DataTable.
-Cada DataTable tiene al menos un DataView, que se usa para la vinculación de datos.
-DataView muestra los datos de DataTable sin cambios o con opciones especiales de filtro u ordenación

DataTable

-Se utiliza en los DataSet para modificar las tablas
-Tables[“tabla”] devuelve la tabla reclamada

GridView

-presentación de datos en forma de tabla
-permite: Selección, Paginación, Ordenación, Edición y es extensible mediante plantillas

-.DataSource es el origen de datos. Podemos asignarle una tabla de la BD como origen de datos, o los registros obtenidos de una SQL
-.DataBind() muestra el contenido del GridView
-Para la paginación en GridView debemos usar AllowPaging = true y PageSize = X (número de elementos por página). Si no se usa el asistente se debe usar en el código del GridView un PageIndex
-Los tipos de columnas son:

BoundField	Muestra el texto de un campo de la BBDD
ButtonField	Muestra un botón para cada item
CheckBoxField	Muestra un checkbox para cada item
CommandField	Proporciona funciones de selección, edición y borrado
HyperLinkField	Muestra el texto de un campo de la BBDD como un hipervínculo
ImageField	Muestra una imagen
TemplateField	Permite especificar múltiples campos y controles personalizados

-EmptyDataText Se utiliza para mostrar un mensaje cuando no existen datos que mostrar en el GridView
-EmptyDataTemplate permite personalizar el mensaje mostrado cuando el GridView está vacío

Gestión de conflictos

-Concurrencia pesimista: Cuando una fila es leída, esta queda bloqueada para su lectura para cualquier otro que la demande hasta que aquel que la posee la libere
-Concurrencia positiva: Las filas están disponibles para su lectura en todo momento, pueden ser leídas por distintos usuarios al mismo tiempo. Cuando alguno intenta modificar una fila ya modificada se produce un error y no se modifica
-Last win: No existe control. El último cambio en escribirse es el que permanece

Concurrencia positiva

-el DataSet mantiene dos versiones de las filas que leímos: la original y la actualizada
-Cuando se actualiza la fila, se comparan los valores originales con la fila real de la BD, para ver si ha sido modificada. Si ha sido modificada, hay que capturar una excepción, si no, se actualiza

El evento RowUpdated

-Se produce Al actualizar una fila: después de cada operación pero antes de lanzar cualquier excepción
-permite examinar los resultados e impedir que se lance una excepción para conservar los cambios

Tema 13. Aplicaciones Web (IV)

Cookies:

Para almacenar datos relativos a un usuario y que éstos se preserven después de cerrar la ventana del navegador es necesario utilizar las cookies .

Una cookie se representa por la clase HttpCookie
Las cookies del usuario se leen a través de la propiedad Cookies del objeto Request
Las cookies del usuario se modifican a través de la propiedad Cookies del objeto Response
Por defecto las cookies expiran cuando se cierra el navegador
Borrar cookies, La única forma es reemplazarla por una cookie con una fecha de expiración que ya ha pasado

EMAIL

ASP.NET permite automatizar escribir cada email
System.Net.Mail

- SmtpClient
- MailMessage
- Attachment
- AttachmentCollection
- MailAddress
- MailAddressCollection

MailMessage

- From
- To
- CC
- Bcc
- Attachments
- Subject
- Body
- IsBodyHTML

Clases para trabajar con ficheros y streams:

File: Métodos para trabajar con ficheros
FileStream: Representa un stream para leer y escribir en ficheros
StreamReader: Lee caracteres de un fichero de texto
StreamWriter: Escribe caracteres en un fichero de texto
Path: Métodos para manipular un fichero o directorio

FileUpload:

HasFile: True si el usuario ha subido un archivo
FileName: El nombre del fichero como string
FileContent: Stream que puede ser utilizado para leer el contenido
FileBytes: Array de bytes que puede ser utilizado para leer el contenido
PostedFile: Un objeto HttpPostedFile
ContentLength: El tamaño del fichero en bytes
ContentType: El tipo MIME (image/gif)
SaveAs: Se le pasa un string que contiene la ruta y el nombre del archivo a guardar

Controles de usuario

Se pueden crear controles personalizados y reutilizables en diferentes páginas : CONTROLES DE USUARIO
Los cambios se ven reflejados.
Contienen uno o varios controles ASP.NET junto con el código necesario para que los controles realicen la funcionalidad deseada.
En lugar de una directiva @Page tiene @Control

Los controles de usuario no se pueden ejecutar como archivos independientes. En su lugar, debe agregarlos a las páginas ASP.NET.
El control de usuario no contiene elementos html, body o form

Como crear una pagina

Agregar un nuevo elemento al proyecto, de tipo user control
Una vez creado, trabajamos como si de una página .aspx normal se tratara
Para añadir el control a cualquier sitio de nuestra Web, no tenemos más que arrastrarlo al lugar que queramos..

Incluir un control de usuario en una página Web ASP.NET

Se puede hacer de dos maneras:
Manualmente añadimos la siguiente directiva en el aspx
Atributo src: define la ruta al control de usuario
Atributo tagprefix: permite asociar un prefijo al control de usuario
Atributo Tagname: asociamos un nombre al control de usuario
Arrastramos el control desde nuestro explorador de soluciones a la página donde queramos incluirlo y automáticamente se añadirá esta directiva