Universidad Abierta Interamericana

Campus Virtual UAIOnline

PROGRAMACIÓN I

LECTURA

UNIDAD 1

INTRODUCCIÓN A LA PROGRAMACIÓN VISUAL

Autor de contenidos:
Nicolás Battaglia

OBJETIVOS

En este texto le presentamos un listado de conceptos fundamentales para iniciar la programación en C#, algunos de ellos aplicables para la resolución de los trabajos prácticos.

Intentamos con un lenguaje sencillo y claro, ofrecer un marco básico para poder encarar la práctica y el estudio en profundidad de este lenguaje. Incluimos con este propósito referencias a los Laboratorios que demandan estos contenidos.

ENUNCIADO

1.1. Tipo System Object

Todo en **.NET** es una clase donde en la parte superior de la jerarquía de objetos se encuentra la clase **System.Object**.

Lo único que no deriva del **.NET Framework** son las interfaces.

Todas las clases heredan (directa o indirectamente) de System.Object. Esto significa que siempre podrá asignar cualquier tipo de dato a una variable System.Object y nunca obtendrá un error de compilación o de tiempo de ejecución.

Por ejemplo, al generar una nueva clase codificamos de la siguiente manera:

```
System.Object MiVar= new OtraClase();

Dim MiVar as System.Objetc = New OtraClase()
```

Veamos algunos **Métodos** del System.Object. Esta clase tiene cinco **métodos públicos** y dos **protegidos**:

Métodos Públicos

1. **Equals:** método reemplazable que verifica si el objeto actual tiene el mismo valor que el objeto que se ha pasado como argumento.
2. **GetHashCode**: método reemplazable que devuelve el *código hash* correspondiente al objeto. Este método se utiliza cuando se emplea el objeto como una clave para las colecciones y las *tablas hash*. Idealmente, el *código hash* debe ser único para cada instancia de objeto por lo que podría verificarse que dos objetos son "iguales" sin más que comparar su *código hash*.
3. **GetType**: método que devuelve un valor que identifica el tipo del objeto. El valor proporcionado se suele utilizar normalmente en operaciones de reflexión.
4. **ToString**: método reemplazable que devuelve el nombre completo de la clase.
5. **ReferenceEquals**: método compartido que acepta dos argumentos de objeto y devuelve *true* si hacen referencia a la misma instancia.

Métodos Protegidos

1. **MenberwiseClone**: devuelve un objeto del mismo tipo. Inicializa sus campos y propiedades a fin de que el nuevo objeto pueda ser considerado una copia.
2. **Finalize**: es un método reemplazable que se llama cuando un objeto ha sido recolectado por no haber sido utilizado (tiempo de vida de un objeto).

Tipo String

Una cadena es una colección secuencial de caracteres que se utiliza para representar texto.

Un objeto **String** es una colección secuencial de objetos System.Char que representan una cadena; un objeto System.Char corresponde a una unidad de código UTF-16.

El valor del objeto String es el contenido de la colección secuencial de objetos System.Char, y ese valor es inmutable (es decir, es de solo lectura).

El tamaño máximo de un objeto **String** en la memoria es de 2 GB, o aproximadamente 1 mil millones de caracteres.

La clase **String** expone numerosos métodos constructores sobrecargados.

Con ellos podrá crear sus cadenas de diversas formas.

Por ejemplo:

```
//declaro cadena
string cadena ;
//declaro he inicializo
string cadena1 = new string('A', 10);
string cadena2 = "";
//declaro cadena nula
string cadena3= null;
//utilizo la barra para que la dirección sea válida
string cadena4 = "c:\\Program Files\\Microsoft Visual Studio 8.0";
//utilizo el @ para que el compilador no crea que las barras son caracteres de escape
string cadena5 = @"c:\Program Files\Microsoft Visual Studio 9.0";
//declaro he inicializo indicando el espacio de nombre
System.String cadena6 = "Hello World!";
//utilizo una variable tipo objeto que admite cualquier valor
var cadena7 = "I'm still a strongly-typed System.String!";
//declaro una cadena que no podrá ser modificada
const string cadena8 = "You can't get rid of me!";
//declaro una cadena como arreglo de caracteres
char[] letras = { 'A', 'B', 'C' };
//declar una cadena como objeto
string cadena9 = new string(letters);
```

Propiedades y Métodos

Las únicas propiedades de la clase **String**

son **Length** y **Chars**.

- **Length**: Devuelve el numero de caracteres que forman la cadena.
- **Chars**: Devuelve el carácter situado en un índice determinado.

Optimización de la Cadena

Un importante detalle que deberá recordar es que un objeto **string** es inmutable (esto quiere decir que una vez que cree un **string** no podrá cambiar su contenido).

Las aplicaciones .NET podrán optimizar la administración de las cadenas sin más que mantener un grupo interno de valores de cadena conocido como grupo interno (o pool interno). Si el valor que se ha asignado a una variable de cadena coincide con una de las cadenas contenida ya en el grupo interno, no se reservará memoria adicional y la variable recibirá la dirección del valor

de la cadena en el grupo.

Sin embargo, este paso de optimización no se llevara a cabo en tiempo de ejecución.

Métodos Compartidos

La clase **String** expone otros métodos compartidos (estáticos) a los que podrá llamar sin tener que generar en primer lugar una instancia de un objeto **String**. Por ejemplo, el método **concat** permite el empleo de un número arbitrario de argumentos **String** y devuelve la cadena que resulta de la concatenación de todos los argumentos.

Formato de Valores Fecha y Numéricos

Valores Numéricos

El método compartido **Format** de la clase **String** le permitirá dar formato a una cadena e incluir en ella uno o mas valores numéricos o

de fecha, en una forma similar a como lo hace la función **printf** del lenguaje C o el método **Console.Write**. La cadena a formatear puede contener marcadores de posición para los argumentos, en el formato {N} siendo N un índice que comienza en Cero.

Valores de Fecha

El método **string.Format** también permite el empleo de valores de fecha y hora con formatos estándar y personalizados.

El Tipo Char

La clase **Char** representa a los caracteres simples. No hay mucho que decir sobre esta clase de datos, salvo que expone un cierto número de útiles métodos compartidos que le permitirán probar un carácter siguiendo ciertos criterios.

Todos estos métodos se encuentran sobrecargados y pueden tomar bien un único **Char** o una cadena mas un índice de cadena.

El Tipo StringBuilder

Se puede pensar que los objetos **StrinBuilder** son como un búfer que contiene una cadena y que tiene la capacidad de crecer desde cero caracteres a la capacidad real del búfer. Hasta que exceda esta capacidad, la cadena permanecerá en el búfer y no se asignará ni liberará memoria. Si la cadena llega a superar la capacidad real, el objeto **StringBuilder** crea, de forma transparente, un búfer de mayor tamaño.

Tipo Numérico

Los tipos **short**, **int** y **long** son, simplemente, las clase **int16**, **int32** e **int64** de .NET. Al reconocer que son clases, podrán sacar un mayor partido a estos tipos, por ejemplo, utilizando sus métodos y propiedades.

Propiedades y Métodos

Todos los tipos numéricos exponen el método **ToString**, que convierte su valor numérico a una cadena.

Todas las clases numéricas exponen las propiedades compartidas **Minvalue** y **Maxvalue** que devuelve el menor y mayor valor.

Las clases numéricas que permiten el empleo de valores en punto flotante exponen algunas propiedades compartidas de solo lectura. Por ejemplo la propiedad **Epsilon** devuelve el menor número positivo (distinto de cero) que se pueden almacenar en una variable determinada.

Las clases **Single** y **Double** también exponen algunos métodos de instancia que le permiten comprobar si contiene ciertos valores especiales: **IsInfinity**, **IsNegativeInfinity**, **IsPositiveInfinity** e **isNaN**.

Formato Numérico

Todas las cadenas numéricas disponen de una forma sobrecargada del método **ToString**. Este método utiliza la configuración local para interpretar la cadena de formato.

La clase **NumberFormatInfo** expone numerosas propiedades que determinan la forma en que se va a dar formato a un valor numérico.

Conversión de cadenas en números

Todos los tipos numéricos permiten el empleo del método compartido **Parse**, que analiza la cadena pasada como argumento y devuelve el valor numérico correspondiente. La forma más sencilla del método **Parse** acepta un argumento de cadena.

Clase Convert

La clase **System.Convert** expone varios métodos compartidos que le ayudarán a convertir los distintos tipos de datos disponibles en .Net. En su forma más sencilla estos métodos pueden convertir cualquier tipo base en otro tipo.

La clase **Convert** expone numerosos métodos **ToXxxx**, uno para cada tipo básico: **ToBoolean**, **ToByte**, etc.

Generadores de Números Aleatorios

Visual Basic .NET presenta la clase **System.Randon** que permite el empleo de la generación de números en forma aleatoria. Ésta presenta una serie de constructores que permiten definir entre otras cosas los límites inferior y superior de los números que deseamos obtener.

El Tipo DateTime

System.DateTime es la principal clase .NET para manejar valores de fecha y hora. No solo ofrece un lugar en el cual almacenar valores de datos, también expone varios métodos de utilidad.

Como Sumar y Restar Fechas

La clase **DateTime** expone varias métodos de instancia que le permitirán agregar y restar un numero de año, meses, días, horas, minutos o segundos a un valor Date.

Estos métodos son: **AddYears**, **AddMonths**, **AddDays**, **AddHours**, **AddMinutes**, **AddSeconds**, **AddMillisecons**, **AddTickes**.

Formato de Fechas

El tipo **DateTime** sustituye al método **Tostring** para proporcionar una representación compacta de los valores de fecha y hora que contiene.

Podrá dar formato a un valor **DateTime** sin más que utilizar algunos métodos particulares que solo expone este tipo.

Análisis de Fecha

La operación complementaria al formato de fecha es el análisis. La clase **Date** proporciona un método **Format** compartido para efectuar trabajos de análisis de cualquier grado de complejidad.

La flexibilidad de este método resultará totalmente aparente cuando le pase un objeto **IFormatProvider** como segundo argumento. Este objeto es conceptualmente similar al objeto **NumberFormatInfo**. Sin embargo, este objeto almacena información sobre separadores y formatos permitidos en valores de fecha y hora.

Manejo de Zonas Horarias

.NET FrameWork permite el empleo de información de la zona horaria gracias al objeto **System.TimeZone**, que podrá utilizar para recuperar información sobre la zona horaria definidas en las opciones regionales de Windows.

Array (Clase)

Proporciona métodos para la creación, manipulación, búsqueda y ordenación de matrices, por lo tanto, sirve como clase base para todas las matrices de **Common Language Runtime**.

Carece de un constructor público porque su procedimiento **New** tiene un ámbito protegido.

System Collections

El espacio de nombre **system collections** expone un gran nuecero de clases que pueden trabajar como contenedores genéricos de datos, tales como colecciones y diccionarios.

Lo más recomendable es conocer las interfaces subyacentes que estas clases ponen a su disposición.

Clase Stack

Visual Basic .Net podrá construir una estructura de pila sin mas que generar una instancia de un objeto **System.Collection.Stack**.

Los tres métodos básicos del objeto **stack** son: **Push**, **Pop** y **Peek**; la propiedad **Count** proporcionan el número de elementos almacenados en la pila.

Clase Queue

En Visual Basic .Net para generar una cola podrá utilizar el objeto **System.Collections.Queue**.

Los Objetos **Queue** tienen una capacidad inicial, pero el búfer interno se amplia automáticamente cuando surge la necesidad.

Podrá crear un objeto **Queue** sin mas que especificar su capacidad inicial y un factor opcional de crecimiento.

Control de flujo

C# ha heredado la sintaxis de la mayoría de las instrucciones que gobiernan el flujo de ejecución, tales como los bucles **If**, **For**, **Do…While, While.**

.

Procedimientos

C# permite la definición de funciones con la misma estructura de C.

Las funciones pueden devolver algún valor, (int, float, char, otros), o nada, (void).

Instrucciones condicionales y de bucle

C# permite el empleo de todas las instrucciones condicionales y de bucle utilizadas en sus predecesores, es decir: los bloques condicionales **If, Switch** y las instrucciones de bucle **For**, **Do While** y **While**. Sin embargo, .Net Framework también ofrece nuevas posibilidades en esta área.

El bucle While

C# permite el empleo de bucles **For** y estos siguen exactamente las mismas sintaxis utilizadas en versiones anteriores del lenguaje.

C# también permite el empleo de la palabra **While**. Podrá salir del bucle **While** utilizando la instrucción **Exit While**.

GoTo y sus Variantes

En versiones anteriores a este lenguaje, permitían el empleo de cuatro tipos de instrucciones de salto intraprocedimientos: **GoTo**, **GoSub**, **On…GoTo** y **On…GoSub**.

Las ultimas tres instrucciones han dejado de ser validas en .Net, solo **GoTo** sigue vigente pero con una sintaxis ligeramente distinta (**Goto** en lugar de **GoTo**).

Introducción a los formularios

Un formulario de C# no es otra cosa que una clase que hereda de la clase **System. Windows.forms.form**; no tiene nada de especial si las comparamos con otras clases .NET

El diseñador de formularios

El diseñador de C# es un sofisticado generador de código. Cuando defina una propiedad de un control en la ventana propiedades estará creando una o más instrucciones en C# que asignarán un valor a dicha propiedad una vez que se genere el formulario.

Nuevas características del diseñador

El diseñador de formularios de C# permite bloquear cada control utilizando la propiedad **locked**.

La jerarquía de clases de Windows Forms

Las clases contenidas en el espacio de nombres **System.Windows.Froms** tienen una jerarquía realmente compleja, en la raíz de la cual se encuentra la clase **System.ComponentModel.Component**, que representa un objeto que se puede introducir en un contenedor.

El objeto Form

El objeto **Form** deriva del objeto **ContainerControl** que, a su vez, deriva de **ScrollableControl** y, en último lugar, de **Control.** Por ello, trabajar con formularios resulta similar a trabajar con un control.

Naturalmente, los formularios son más complicados y ricos en funcionalidad que los controles.

Menús

El diseñador de Windows **Form** permite ahora crear la estructura de menús de sus aplicaciones utilizando una técnica bastante más directa y sencilla, mediante un editor **WYSIWYG**. Ahora podrá incluso mover elementos y submenús completos utilizando la técnica "arrastrar y soltar".

Formularios MDI

El espacio de nombres Windows Forms no dispone de una clase independiente para los **formularios MDI**: un **formulario MDI** no es otra cosa más que un objeto **Form** regular cuya propiedad **IsMdiContainer** se ha definido como **true**.

La única limitación reseñable de los contenedores MDI es que su contenido no es desplazable. Si intenta asignar el valor **True** a la propiedad **IsMdiContainer**, la propiedad **AutoScroll** se reconfigurara como **False** y viceversa.

Técnicas avanzadas de formularios

En esta describiré algunas técnicas avanzadas de formularios, tales como la la localización y la personalización de formularios.

La propiedad Opacity

La propiedad **Opacity** le permitirá aplicar un nivel de transparencia a todo el formulario, incluyendo su barra de títulos y bordes.

Es un valor **Double** perteneciente al rango 0 a 1, por lo que podrá modificar el nivel de opacidad con una gran precisión.

Formularios Localizados

Los diseñadores de Windows han utilizado tradicionalmente archivos de recursos para crear aplicaciones multilenguajes. El problema relacionado con los archivos de recurso es que no se prestan demasiado bien al Desarrollo Rápido de Aplicaciones (RAD).

Este problema se ha resuelto en el diseñador de Visual Studio .NET de una forma simple, elegante y eficaz.

El objeto Clipboard

El objeto **Clipboard** le permitirá copiar datos en el portapapeles de Windows y, a continuación, pegarlos en cualquier otro lugar.

El objeto **Clipboard** permite el empleo de varios formatos, cada uno de ellos identificado por una constante de cadena expuesta por la clase **DataFormats**.

También existe un formato **CommaSeparatedValu**e que le permitirá importar datos en formato **CSV** desde hojas de cálculo y otras muchas aplicaciones.

El objeto Application

El objeto **System.Windows.Forms.Application** expone algunas propiedades, métodos y sucesos de gran interés. Todos los miembros de esta clase son compartidos y no pueden crear una instancia del objeto **Application**.

El objeto Cursor

La clase cursor tiene un doble propósito:

- Sus propiedades y métodos estáticos le permitirán controlar varias características del cursor del ratón.
- Su método constructor le permitirá crear un nuevo cursor del ratón, que podrá asignar posteriormente la propiedad estática **Mouse.Current** o a la propiedad **Cursor** de cualquier control.

La clase Sendkey

Con su método compartido **Send** podrá enviar una o más pulsaciones de tecla a la aplicación activa.

La clase Help

Esta clase permite el acceso a la ayuda válida que se encuentra en el ámbito de Docs. Microsoft.

Para obtener la ayuda correcta debemos seleccionar la palabra buscada y presionar F1.

Proveedores de controles

La arquitectura de Windows Forms es bastante extensible. Por ejemplo, puede crear los denominados *Controles definidos por el Usuario*, que agregan nuevas propiedades a todos lo controles contenidos en el formulario.

El espacio de nombres de Windows forms incluye 3 tipos: **ToolTip**, **ErrorProvider** y **HelpProvider**.

El **Control ToolTip** permite que cualquier control contenido en un formulario muestre un ToolTip (mensaje de ayuda) cuando el usuario pase por encima.

El control **ErrorProvider** le permitirá desarrollar aplicaciones de Windows Forms utilizando un método de validación que ahora resulta familiar a todos los usuarios que pasan parte de su tiempo en internet.

Esta técnica de validación permite que los usuarios finales pasen de un campo a otro y los rellenen en el orden que prefieran.

El control **HelpProvider** funciona como un puente entre su programa y la clase Help para que pueda mostrar sencillos mensajes de ayuda o paginas de ayuda mas complejas cuando el usuario pulse la tecla F1 y el foco se encuentre sobre el control contenido en su formulario.

El control Splitter

El control **Splitter** facilita al máximo la creación de barras de división, es decir, aquellos divisores que puede utilizar para dividir el espacio contenido en el formulario entre los distintos controles.

El control ImageList

Podrá solicitar un control **ImageList** con cualquier control que permita la propiedad **Image**, no solo con controles comunes de Windows tales como **TreeView** o **ListView**.

El control ListView

La ventana propiedades le permitirá definir todos los objetos **ColumnHeade**r que desee definir. Otra gran mejora es la capacidad para modificarla anchura de las columnas en tiempo de diseño.