Equivalencias de lenguajes Visual Basic, C#, Java, C++, Jscript

Tips de Programación en aplicaciones Web

#### INDICE

[EQUIVALENCIAS DE LENGUAJES 2](#_Toc522059823)

[Declarar Variables 2](#_Toc522059824)

[Comentarios 2](#_Toc522059825)

[Asignar valores 3](#_Toc522059826)

[Declaración If...Else 3](#_Toc522059827)

[Declaración CASE 5](#_Toc522059828)

[Lazo FOR 7](#_Toc522059829)

[Lazo WHILE 7](#_Toc522059830)

[Parámetros pasados por valor 8](#_Toc522059831)

[Parámetros pasados por Referencia 9](#_Toc522059832)

[Manipulación de Excepciones estructurado 10](#_Toc522059833)

[Setear la referencia de un objeto a NULO 12](#_Toc522059834)

[BÁSICAS 12](#_Toc522059835)

[Object-oriented programming 16](#_Toc522059836)

[Exception handling 17](#_Toc522059837)

[Decision structures 17](#_Toc522059838)

[Arrays 17](#_Toc522059839)

[Class Scope 18](#_Toc522059840)

[Member Scope 18](#_Toc522059841)

[Misc. Lifetime 18](#_Toc522059842)

[Misc. 18](#_Toc522059843)

[TIPS DE PROGRAMACIÓN 20](#_Toc522059844)

[C# APLICACIONES Web 20](#_Toc522059845)

[HyperLink 20](#_Toc522059846)

[Button 20](#_Toc522059847)

[TextBox 20](#_Toc522059848)

[DataGrid 21](#_Toc522059849)

[DataSet 21](#_Toc522059850)

[DropDownList (Combo) 21](#_Toc522059851)

[Programación 21](#_Toc522059852)

# 

# EQUIVALENCIAS DE LENGUAJES

### Declarar Variables

#### Visual Basic

Dim x As Integer

Public x As Integer = 10

#### Java

int x;

int x = 10;

#### C++

int x;

int x = 10;

#### C#

int x;

int x = 10;

#### JScript

var x : int;

var x : int = 10;

### Comentarios

#### Visual Basic

' comment

x = 1 ' comment

Rem comment

#### Java

//

/\* multiline

comment \*/

/\*\*

Class Documentation

\*/

#### C++

// comment

/\* multiline

comment \*/

#### C#

// comment

/\* multiline

comment \*/

#### JScript

// comment

/\* multiline

comment \*/

### Asignar valores

#### Visual Basic

nVal = 7

#### Java

nVal = 7;

#### C++

nVal = 7;

#### C#

nVal = 7;

#### JScript

nVal = 7;

### Declaración If...Else

#### Visual Basic

If nCnt <= nMax Then

nTotal += nCnt ' Same as nTotal = nTotal + nCnt.

nCnt += 1 ' Same as nCnt = nCnt + 1.

Else

nTotal += nCnt

nCnt -= 1

End If

#### Java

if (nCnt <= nMax){

nTotal += nCnt;

nCnt++;

}

#### C++

if(nCnt < nMax) {

nTotal += nCnt;

nCnt++;

}

else {

nTotal += nCnt;

nCnt --;

};

#### C#

if (nCnt <= nMax)

{

nTotal += nCnt;

nCnt++;

}

else

{hTotal +=nCnt;

nCnt--j}

#### JScript

if(nCnt < nMax) {

nTotal += nCnt;

nCnt ++;

}

else {

nTotal += nCnt;

nCnt --;

};

### Declaración CASE

#### Visual Basic

Select Case n

Case 0

MsgBox ("Zero")

' Visual Basic.NET exits the Select at the end of a Case.

Case 1

MsgBox ("One")

Case 2

MsgBox ("Two")

Case Else

MsgBox ("Default")

End Select

#### Java

switch(n) {

case 0:

System.out.println("Zero\n");

break;

case 1:

System.out.println("One\n");

break;

default:

System.out.println("?\n");

}

#### C++

switch(n) {

case 0:

printf("Zero\n");

break;

case 1:

printf("One\n");

break;

case 2:

printf("Two\n");

break;

default:

printf("?\n");}

#### C#

switch(n) {

case 0:

Console.WriteLine("Zero");

break;

case 1:

Console.WriteLine("One");

break;

case 2:

Console.WriteLine("Two");

break;

default:

Console.WriteLine("?");

}

#### JScript

switch(int(n)) {

case 0 :

print("Zero")

break

case 1 :

print("One")

break

case 2 :

print(“Two”)

default :

print(“Default”)

}

### Lazo FOR

#### Visual Basic

For n = 1 To 10

MsgBox("The number is " & n)

Next

For Each prop In obj

prop = 42

Next prop

#### Java

for(n=1; n<11;n++)

System.out.println("The number is " + n);

#### C++

for(int n=1; n<11; n++)

printf("%d\n",n);

#### C#

for (int i = 1; i <= 10; i++)

Console.WriteLine("The number is {0}", i);

foreach(prop current in obj)

{

current=42;

}

#### JScript

for (var n = 0; n < 10; n++) {

print("The number is " + n)

}

for (var prop in obj)

obj[prop] = 42

### Lazo WHILE

#### Visual Basic

While n < 100 ' Test at start of loop.

n += 1 ' Same as n = n + 1.

End While '

#### Java

while (n < 100)

n++;

#### C++

while(int n < 100)

n++;

#### C#

while (n < 100)

n++;

#### JScript

while (n < 100)

n++

### Parámetros pasados por valor

#### Visual Basic

Public Sub ABC(ByVal y As Long) ' The argument Y is passed by value.

' If ABC changes y, the changes do not affect x.

End Sub

ABC(x) ' Call the procedure.

' You can force parameters to be passed by value, regardless of how

' they are declared, by enclosing the parameters in extra parentheses.

ABC((x))

#### Java

Objects are always passed by reference, and primitive data types are always passed by value.

#### C++

MyMethod(i,j);

#### C#

// The method:

void ABC(int x)

{

...

}

// Calling the method:

ABC(i);

#### JScript

ABC(i)

### Parámetros pasados por Referencia

#### Visual Basic

Public Sub ABC(ByRef y As Long)

' The parameter y is declared by by referece:

' If ABC changes y, the changes are made to the value of x.

End Sub

ABC(x) ' Call the procedure

#### Java

Objects are always passed by reference, and primitive data types are always passed by value.

#### C++

// Prototype of ABC that takes a pointer to integer

int ABC(long \*py);

ABC(&VAR);

//Prototype of ABC that takes a reference to integer

int ABC(long &y);

ABC(VAR);

#### C#

// The method:

void ABC(ref int x)

{

...

}

// Calling the method:

ABC(ref i);

#### JScript

/\* Reference parameters are supported for external object, but not internal JScript functions. Use '&' to call by reference \*/

myObject.ByRefMethod(&foo);

### Manipulación de Excepciones estructurado

#### Visual Basic

Try

If x = 0 Then

Throw New Exception("x equals zero")

Else

Throw New Exception("x does not equal zero")

End If

Catch err As System.Exception

MsgBox("Error: " & Err.Description)

Finally

MsgBox("Executing finally block.")

End Try

#### Java

try{

if (x == 0)

throw new Exception ("x equals zero");

else

throw new Exception ("x does not equal zero");

}

catch (Exception err){

if (err.getMessage() == "x equals zero")

System.out.println(err.getMessage());

//Handle Error Here

}

#### C++

\_\_try{

if (x == 0)

throw new Exception ("x equals zero");

else

throw new Exception ("x does not equal zero");

}

\_\_catch(Exception e)

{

Console.WriteLine("Caught Exception");

}

\_\_finally

{

Console.WriteLine("Executing finally block");

}

#### C#

// try-catch-finally

try

{

if (x == 0)

throw new System.Exception ("x equals zero");

else

throw new System.Exception ("x does not equal zero");

}

catch (System.Exception err)

{

System.Console.WriteLine(err.Message);

}

finally

{

System.Console.WriteLine("executing finally block");

}

#### JScript

try {

if (x == 0)

throw "x equals zero"

else

throw "x does not equal zero"

}

catch(e) {

print("Error description: " + e)

}

finally {

print("Executing finally block.")

}

### Setear la referencia de un objeto a NULO

#### Visual Basic

o = Nothing

#### Java

stringVar = null;

#### C++

#### C#

o = null;

#### JScript

o = null

## 

## BÁSICAS

|  |  |  |
| --- | --- | --- |
| **VB.NET** | **C#** | **Descripción** |
| NotInheritable | sealed | Especifica que una clase no puede ser usada como base para otra clase, es decir, no puede ser heredada. |
| NotOverridable | sealed | Especifica que un método no puede ser sobrescrito. |
| MustInherit | abstract | Especifica que una clase sólo puede ser heredada (no puede ser creada una instancia de la clase). |
| MustOverride | abstract | Especifica que un método debe ser implementado en las clases hijas. |
| Overridable | virtual | Especifica que una propiedad de la clase puede ser sobrescrita. |
| Shared | static | Especifica que una propiedad de la clase es compartida por todas las instancias de la clase (es estático). No es necesaria ninguna instancia de la clase para llamar a esta propiedad). |
| Static | no equivalent | Especifica que el valor de una variable local es preservado entre diferentes llamadas. |
| Public | public | Clase o propiedad es visible fuera del proyecto o assembly. |
| Friend | internal | Clase o propiedad es invisible fuera del proyecto o assembly. |
| Private | private | Clase o propiedad es visible sólo en el proyecto. |
| Overloads | not required | Especifica que una propiedad sobrecargada por otra propiedad. |
| Overrides | override | Especifica que una propiedad sobrescrita por otra propiedad. |
| Implements I1 | class C1:I1 | Especifica que la clase (C1) implementa el interfaz I1. |
| Inherits C2 | class C1:C2 | Especifica que la clase (C1) hereda la clase C2. |
| Implements I1  Inherits C2 | class C1:C2,I1 | Especifica que la clase (C1) implementa el interfaz I1 y hereda la clase C2. |
| Shadows | new |  |
| Finalize | ~C1 (destructor) | Método llamado por el sistema justo antes de que el garbage collection reclame el objeto. C1 es el nombre de la clase. |
| New | C1 | Método constructor, llamado cuando el objeto es creado. C1 es el nombre de la clase. |
| Dim x as Int32 | Int32 x | Declara la variable “x” del typo “System.Int32”. |
| Imports | using | Permite llamar a métodos sin necesidad de especificar el nombre del Namespace completo. |
| <> | [] | Especifica parámetros. |
| \_ | ; | Continuación de línea en VB, fin de línea en C#. |
| And  Or | &&  || | Operadores lógicos. |
| no soportado | << >> | Operadores Shift. |
| X+=1 (x=x+1)  X-=1 (x=x-1)  also \*=, /=, ^= etc.. | x++  x– | Incrementos. |
| Dim x(4) as Int32  = 5 items (0 to 4) | Int32[] x = new Int32[4];  = 4 items (0 to 3) | Diferencias entre los elementos creados en las declaraciones de arrays. |
| Dim x as Int32 | Int32 x = 0 | x es inicializado a 0 automáticamente en VB. |
| ReDim Preserve | no equivalent | Redimensiona un array. |
| Optional | no soportado | Especifica que un parámetro es opcional. |
| Select Case x Case True Case Else End Select | switch(I) { case 1:break; default:break; }; |  |
| ByVal  ByRef | not requerido  ref | Pasando parámetros por valor y referencia. |
| Me | this | Referencia al objeto actual. |
| MyBase | base | Referencia a la clase base. |
| MyClass | no soportado | Realiza una llamada non-virtual a un método virtual del objeto actual. |
| Const | const  readonly | Declara una constante. |
| Enum | enum | Declara un enumerator. |
| Structure | struct | Declara una estructura. |
| no soportado | volatile | Declara un objeto que puede ser modificado de forma asíncrona. |
| obj = Nothing | obj == null | Comprueba que una variable objeto no es nula. |
| Option Explicit | Opción por defecto y no puede ser cambiada | Especifica que todas las variables tienen que ser declaradas. |
| IsDBNull | no soportado | Comprobación para un nulo de base de datos. |
| Default | no soportado | Especifica un método por defecto de una clase. |
| WithEvents | no soportado | Declara una variable cuyos eventos queremos manejar. |
| Handles | no soportado | Especifica que un método es llamado por un evento. |
| Try Catch Finally End Try | try{} catch{} finally{} | Manejo estructurado de las excepciones. |
| REM  ‘ | /\* … \*/  // | Líneas de comentarios |
| not soportado | /// | líneas de comentarios XML |
| Dim x As String = “Hola” Dim y As Char = GetChar(x, 1) | string x = “Hola”; char y; y = x[1]; | Recupera un carácter de una cadena de texto. |
| With … End With | no soportado | Evalúa un objeto una vez y permite usarlo varias veces. |
| Dim a() as Long = {1,2,3} | int[] x = new int[4] {1,2,3,4}; | Inicialización de un array. |
| Event  RaiseEvent | event | Declara y lanza un evento. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Purpose** | **Visual Basic Keyword** | **Java Keyword** | **C++ Keyword** | **C# NEW Keyword** | **JScript Keyword** |
| Declare a variable | Private, Public, Friend, Protected, Static1, Shared, Dim | public, private, protected (if a member variabl) const, volatile (Note: these are not required for a declaration.) | declarators (concept, not keyword) | declarators (keywords include user-defined types and built-in types) | var |
| Declare a named constant | Const | final (Applied to a field declaration) | const | const | const Statement |
| Create a new object | New | New | new | new | new |
|  | CreateObject() | n/a | CoCreateInstance() |  | new ActiveXObject() |
| Assign an object to an object variable | = | = | = | = | = |
| Function/method does not return a value | Sub2 | void | void | void | System.Void |
| Overload a function or method (Visual Basic: overload a procedure or method) | Overloads NEW | (No language keyword required for this purpose) | (No language keyword required for this purpose) | (No language keyword required for this purpose) | (No language keyword required for this purpose) |
| Refer to the current object | Me3 | this | this | this | this |
| Make a nonvirtual call to a virtual method of the current object | MyClass | n/a | n/a |  | n/a |
| Retrieve character from a string | GetChar Function NEW | getChar | \*(p + 10) or p[10] where p is a char\* | [] | charAt, substring, substr4 |
| Declare a compound data type (Visual Basic: Structure) | Structure <members> End Structure | n/a | class, struct, union | struct, class, interface | class NEW |
| Initialize an object (constructors) | Sub New()5 | constructors (concept, not keyword) | constructors (concept, not keyword) | Constructors, or system default type constructors. | constructor (concept not keyword) 6 |
| Terminate an object directly | n/a | n/a | ~*ClassName* | n/a | n/a |
| Method called by the system just before garbage collection reclaims an object7 | Finalize NEW  (In Visual Basic 6, Class\_Terminate) | finalize | n/a | destructor | finalize |
| Initialize a variable where it is declared | Dim x As Long = 5  Dim c As New Car(FuleTypeEnum.Gas) | int x = 5; | int x=5; | // initialize to a value:  int x = 123;  // or use default constructor:  int x = new int(); | var x = 5  var y : foo = new foo() |
| Take the address of a function | AddressOf(For class members, this operator returns a reference to a function in the form of a delegate instance) | n/a | Use the name of the function without parentheses | delegate | Use the name of the function without parentheses |
| Callback | n/a | n/a | CALLBACK (a standard type);  callback (IDL attribute) | n/a | n/a |
| Declare that an object can be modified asynchronously | n/a | volatile | Volatile | n/a | n/a |
| Force explicit declaration of variables | Option Explicit | n/a. All variables must be declared prior to use. | n/a. All variables must be declared prior to use. | n/a. (All variables must be declared prior to use) | fast mode (on by default) |
| Test for an object variable that does not refer to an object | obj = Nothing | obj == null | pobj == NULL | obj == null | obj == undefined  obj == null |
| Value of an object variable that does not refer to an object | Nothing | null | n/a | null | null  undefined |
| Test for a database null expression | IsDbNull | n/a | n/a | n/a | x == null |
| Test whether a Variant variable has been initialized | n/a | n/a | n/a | n/a | x == undefined |
| Define a default property | Default NEW | n/a | n/a | by using indexers | n/a |
| Object-oriented programming |  |  |  |  |  |
| Refer to a base class | MyBase NEW | super | \_\_super | base | super |
| Declare an interface | Interface NEW | interface | \_\_interface | interface | interface Statement |
| Specify an interface to be implemented | Implements (statement) | implements (clause on class declaration) | (Just derive from the interface)  class C1 : public I1 | class C1 : I1 | implements |
| Declare a class | ClassNEW <implementation> | class | class | class | class |
| Specify that a class can only be inherited. An instance of the class cannot be created. | MustInherit NEW | abstract | \_\_abstract8 (Only in Managed Extensions for C++) | abstract | abstract |
| Specify that a class cannot be inherited | NotInheritable NEW | final | \_\_sealed (Only in Managed Extensions for C++) | sealed | final |
| Declare an enumerated type | Enum <members> End Enum | n/a | enum | enum | enum Statement |
| Declare a class constant | Const | Static final (Applied to a field declaration) | const | const (Applied to a field declaration) | const |
| Derive a class from a base class | Inherits C2 NEW | class C1 extends C2 | Class C1 : public Base (No language keyword needed for this purpose) | class C1 : C2 | class c1 extends c2 |
| Override a method | Overrides NEW | (No language keyword required for this purpose) | (No language keyword required for this purpose) | override | (No language keyword required for this purpose) |
| Declare a method that must be implemented in a deriving class | MustOverride NEW | abstract | Put **= 0** at the end of the declaration (pure virtual method) | abstract | abstract |
| Declare a method that can't be overridden | NotOverridable NEW  (Methods are not overridable by default.) | final | \_\_sealed (Only in Managed Extensions for C++) | n/a | final |
| Declare a virtual method, property (Visual Basic), or property accessor (C#, C++) | Overridable | (Methods are virtual by default) | virtual | virtual | (Methods are virtual by default) |
| Declare a typesafe reference to a class method | Delegate NEW | n/a | \_\_delegate (Only in Managed Extensions for C++) | delegate | n/a |
| Specify that a variable can contain an object whose events you wish to handle | WithEvents | n/a | n/a | (Write code - no specific keyword) | (Write code - no specific keyword) |
| Specify the events for which an event procedure will be called | Handles NEW (Event procedures can still be associated with a WithEvents variable by naming pattern.) | n/a | n/a | n/a | n/a |
| Evaluate an object expression once, in order to access multiple members | With objExpr <.member> <.member> End With | n/a | n/a | n/a | with (obj) {  prop = 42;  method();  }9 |
| Exception handling |  |  |  |  |  |
| Structured exception handling | Try NEW <attempt> Catch <handle errors> Finally <always execute> End Try | try, catch, finally, throw | \_\_try, \_\_except, \_\_finally | try, catch, finally,  throw | try, catch, finally, throw |
| C++ exception handling | n/a | n/a | try, catch, throw | n/a | n/a |
| Decision structures |  |  |  |  |  |
| Decision structure  (selection) | Select Case …, Case, Case Else, End Select | switch, case, break; default; | switch, case, default, goto, break; | switch, case, default, goto, break | switch, case, break |
| Decision structure (if … then) | If … Then, ElseIf … Then, Else, End If | if, else | if, else | if, else | if, else |
| Loop structure (conditional) | While, Do [While, Until] …, Loop [While, Until] | do, while, continue | do, while, continue | do, while, continue | do, while, continue |
| Loop structure  (iteration) | For …, [Exit For,] Next  For Each …, [Exit For,] Next | for, break | for | for, foreach | for (x=0;x<10;x++){…}  for (prop in obj) {  print (obj[prop]);  } |
| Arrays |  |  |  |  |  |
| Declare an array | Dim a() As Long | int[] x = new int[5]; | int x[5]; | int[] x = new int[5]; | var x : int[] |
| Initialize an array | Dim a() As Long = {3, 4, 5} | int[] x = new int[] {1,2,3,4,5}; | int x[5]= {1,2,3,4,5}; | int[] x = {1, 2, 3, 4, 5}; | var x : int[] =[1, 2, 3] |
| Reallocate array | Redim | n/a | n/a | n/a | arr.length=newSize (only for JScript arrays)10 |
| Class Scope |  |  |  |  |  |
| Visible outside the project or assembly | Public | Public | public | public | public |
| Invisible outside the assembly (C#/Visual Basic) or within the package (Java) | Friend | (Omitting the scope keyword specifies "package scope") | n/a | internal | internal |
| Visible only within the project (for nested classes, within the enclosing class) | Private | private | private | private | private |
| Member Scope |  |  |  |  |  |
| Accessible outside class and project (Java/C++/Visual Basic) or module (Visual Basic) | Public | Public | public | public | public |
| Accessible outside the class, but within the project (C#, Visual Basic) or package (Java) | Friend | (Omitting the scope keyword specifies "package scope") | n/a | internal | internal |
| Only accessible within class (Java/C++/Visual Basic) or module (Visual Basic) | Private | private | private | private | private |
| Only accessible to current and derived classes | Protected NEW | protected | protected | protected | protected |
| Specify that a function or another class has access to private members of the declaring class | n/a | n/a | friend (Not allowed in the Managed Extensions for C++) | n/a | n/a |
| Misc. Lifetime |  |  |  |  |  |
| Preserve procedure's local variables | Static11 | static | static | n/a | n/a |
| Shared by all instances of a class | Shared NEW | static | static | static | static |
| Misc. |  |  |  |  |  |
| Comment code | '  Rem | //  /\*\*  \*/  /\*  \*/ | //, /\* \*/ for multiline comments | //, /\* \*/ for multiline comments | //, /\* \*/ for multiline comments |
| Case-sensitive? | No | Yes | Yes | Yes | Yes |
| Call Windows API | Declare <API> | n/a | n/a | use Platform Invoke | n/a |
| Declare and raise an event | Event, RaiseEvent | n/a | n/a | event | n/a |
| Threading primitives | SyncLock | Contained in Object and Thread classes |  | lock | n/a |
| Go to | Goto | n/a | goto | goto | n/a |

# TIPS DE PROGRAMACIÓN

## C# APLICACIONES Web

### HyperLink

* Propiedad NavigateURL indica la página web a irse.

### Button

* Código Response.redirect(“http://pagina.htm”) para cargar otra página

### TextBox

* Evento **Leave** en lugar de LostFocus
* Propiedad **.text** devuelve el texto del TextBox
* Propiedad **TextMode** para indicar que se ingrese un password.
* Propiedad **ReadOnly** o **Enabled** para que no pueda editar un TextBox

Para validar que se ingrese un campo obligatoriamente

* Arrastre el control ***RequiredFieldValidator***
* En la propiedad **text** digitar “\*”
* En la propiedad **controltovalidate** escoger el control (textbox1)
* En la propiedad **ErrorMessage** digitar el menaje de error.
* Arrastrar ***ValidationSummary*** para que aquí despliegue los mensajes de error (Sólo se necesita uno por cada form web)

Para validar que ingrese la dirección de un email

* Arrastre el control ***RegularExpressionValidator***
* En la propiedad **text** digitar “\*”
* En la propiedad **Display** escoger Dynamic
* En la propiedad **controltovalidate** escoger el control (textbox1)
* En la propiedad **ErrorMessage** digitar el menaje de error.
* En la propiedad **ValidationExpression** escoger Internet email address
* Arrastrar ***ValidationSummary*** para que aquí despliegue los mensajes de error (Sólo se necesita uno por cada form web)

Para validar que ingrese el password igual a confirmar password

* Arrastre el control ***CompareValidator***
* En la propiedad **text** digitar “\*”
* En la propiedad **Display** escoger Dynamic
* En la propiedad **controltocompare** escoger el control para comparar(textbox2)
* En la propiedad **controltovalidate** escoger el control (textbox1)
* En la propiedad **ErrorMessage** digitar el menaje de error.
* En la propiedad **Type** escoger string
* Arrastrar ***ValidationSummary*** para que aquí despliegue los mensajes de error (Sólo se necesita uno por cada form web)

[Visual Basic]

Private Sub MyValidatingCode()

' Confirm there is text in the control.

If textBox1.Text.Length = 0 Then

Throw New Exception("Email address is a required field")

Else

' Confirm that there is a "." and an "@" in the e-mail address.

If textBox1.Text.IndexOf(".") = - 1 Or textBox1.Text.IndexOf("@") = - 1 Then

Throw New Exception("Email address must be valid e-mail address format." + \_

ControlChars.Cr + "For example 'johndoe@somedomain.com'")

End If

End If

End Sub

Private Sub textBox1\_Validating(sender As Object, \_

e As System.ComponentModel.CancelEventArgs) Handles textBox1.Validating

Try

MyValidatingCode()

Catch ex As Exception

' Cancel the event and select the text to be corrected by the user.

e.Cancel = True

textBox1.Select(0, textBox1.Text.Length)

' Set the ErrorProvider error with the text to display.

Me.errorProvider1.SetError(textBox1, ex.Message)

End Try

End Sub

Private Sub textBox1\_Validated(sender As Object, \_

e As System.EventArgs) Handles textBox1.Validated

' If all conditions have been met, clear the error provider of errors.

errorProvider1.SetError(textBox1, "")

End Sub

[C++, JScript] No example is available in C++ or JScript. To view a Visual Basic or C# example, click the Language Filter button in the upper-left corner of the page.

### DataGrid

* Obligatoriamente debe llamarse a la función **DataBind()** para que se llene o se conecte el DataGrid.
* Click derecho y PropertyBuilder para personalizar al grid.
  + La propiedad **DataSource** se conecta con el DataSet o DataTable
  + La propiedad **DataMember** se conecta con el nombre de la tabla.
  + Desactivar Creat column at runtime para personalizar las columnas ancho, color, columnas que se debe desplegar, etc.
* La propiedad **ReadOnly** para no poder modificar los datos de las filas

### DataSet

Para crear una tabla de forma visual

* Arrastrar un **DataSet.**
* Escoger que el dataset este sin conexión
* En la propiedad **Tables** se ingresa una colección de tablas.
  + Se tiene la colección **Columns** (para ingresar las columnas)
  + Se tiene la colección **Constraints** (para ingresar la clave primaria o foránea)
* En la propiedad **Relations** se ingresa las relaciones de las tablas.
* Para actualizar los cambios de una tabla en el Dataset:

ds.Tables[0].AcceptChanges();

### DropDownList (Combo)

* Obligatoriamente debe estar **AutoPostBack**=True para que se cambie los datos del evento SelectedItemChanged, o sea para que indique el texto seleccionado.
* La propiedad **Items** se ingresa una colección de los datos que quiere que despliegue el combobox.
* La propiedad **DataSource** se conecta con el DataSet o DataTable
* La propiedad **DataMember** se conecta con la tabla.(string).
* La propiedad **DataTextField** se conecta con la columna de la tabla (string), es el texto que aparece en el dropdownlist
* La propiedad **DataValueField** se conecta con una columna de la tabla. (string), este dato se almacena como valor para cada texto y no se muestra en el dropdownlist
* La propiedad **SelectedItem.value** devuelve el valor seleccionado
* La propiedad **Text** devuelve el texto seleccionado

### Programación

* Codigo Response.redirect(“http://pagina.htm?codigo=123&nombre=texto”) para cargar otra página y enviar datos como parámetros.
* Para recibir los parámetros int cod=Request.Parms[“codigo”];
* Convert. 🡪 esta clase te permite convertir cualquier tipo de dato a otro tipo de dato.
* DataGrid1.DataBind()
* DropDownList 🡪 AutoPostBack=true Debe programarse Page\_Load para indicar los valores que toma la forma web al cargar la página.
  + IsPostBack es falso si es la primera vez que se carga la página. Es conveniente cuando queremos que realice algo una sola vez, la primera vez que se carga la pagina
* Para acceder a los datos de una tabla o dataset es como matrices, así:
  + Var=(string)Dataset.Tables[int o string].rows[int][int o string];

Tipo de dato que devuelve Fila columna