Cadenas

Una cadena es un objeto de tipo String cuyo valor es texto. Internamente, el texto se almacena como una colección secuencial de solo lectura de objetos Char. No hay ningún carácter que finaliza en null al final de una cadena de C#; por lo tanto, la cadena de C# puede contener cualquier número de caracteres nulos insertados ('\0'). La propiedad Length de una cadena representa el número de objetos Char que contiene, no el número de caracteres Unicode. Para obtener acceso a los puntos de código Unicode individuales de una cadena, use el objeto StringInfo.

cadena frente System.String

En C#, la palabra clave string es un alias de String. Por lo tanto, String y string son equivalentes y se puede utilizar la convención de nomenclatura que prefiera. La clase String proporciona muchos métodos para crear, manipular y comparar cadenas de forma segura. Además, el lenguaje C# sobrecarga algunos operadores para simplificar las operaciones de cadena comunes. Para más información sobre la palabra clave, consulte string. Para obtener más información sobre el tipo y sus métodos, vea String.

Declaración e inicialización de cadenas

Puede declarar e inicializar cadenas de varias maneras, tal como se muestra en el ejemplo siguiente:

```
// Declare without initializing.
string message1;

// Initialize to null.
string message2 = null;

// Initialize as an empty string.
// Use the Empty constant instead of the literal "".
string message3 = System.String.Empty;

//Initialize with a regular string literal.
string oldPath = "c:\\Program Files\\Microsoft Visual Studio 8.0";

// Initialize with a verbatim string literal.
string newPath = @"c:\Program Files\Microsoft Visual Studio 9.0";

// Use System.String if you prefer.
System.String greeting = "Hello World!";

// In local variables (i.e. within a method body)
// you can use implicit typing.
```

```
var temp = "I'm still a strongly-typed System.String!";

// Use a const string to prevent 'message4' from
// being used to store another string value.
const string message4 = "You can't get rid of me!";

// Use the String constructor only when creating
// a string from a char*, char[], or sbyte*. See
// System.String documentation for details.
char[] letters = { 'A', 'B', 'C' };
string alphabet = new string(letters);
```

Tenga en cuenta que no se usa el operador new para crear un objeto de cadena, salvo cuando se inicialice la cadena con una matriz de caracteres.

Inicialice una cadena con el valor constante Empty para crear un objeto String cuya cadena tenga longitud cero. La representación literal de la cadena de una cadena de longitud cero es "". Mediante la inicialización de las cadenas con el valor Empty en lugar de null, puede reducir las posibilidades de que se produzca una excepción NullReferenceException. Use el método estático IsNullOrEmpty(String) para comprobar el valor de una cadena antes de intentar obtener acceso a ella.

Inmutabilidad de los objetos de cadena

Los objetos de cadena son inmutables: no se pueden cambiar después de haberse creado. Todos los métodos String y operadores de C# que parecen modificar una cadena en realidad devuelven los resultados en un nuevo objeto de cadena. En el siguiente ejemplo, cuando el contenido de s1 y s2se concatena para formar una sola cadena, las dos cadenas originales no se modifican. El operador += crea una nueva cadena que contiene el contenido combinado. Este nuevo objeto se asigna a la variable s1 y el objeto original que se asignó a s1 se libera para la recolección de elementos no utilizados porque ninguna otra variable contiene una referencia a él.

```
string s1 = "A string is more ";
string s2 = "than the sum of its chars.";

// Concatenate s1 and s2. This actually creates a new
// string object and stores it in s1, releasing the
// reference to the original object.
s1 += s2;

System.Console.WriteLine(s1);
// Output: A string is more than the sum of its chars.
```

Dado que una "modificación" de cadena es en realidad una creación de cadena, debe tener cuidado al crear referencias a las cadenas. Si crea una referencia a una cadena y después "modifica" la cadena original, la referencia seguirá apuntando al objeto original en lugar de al objeto nuevo creado al modificarse la cadena. El código siguiente muestra este comportamiento:

```
string s1 = "Hello ";
string s2 = s1;
s1 += "World";

System.Console.WriteLine(s2);
//Output: Hello
```

Literales de cadena regulares y textuales

Utilice literales de cadena regulares cuando tenga que insertar caracteres de escape proporcionados por C#, tal como se muestra en el ejemplo siguiente:

```
string columns = "Column 1\tColumn 2\tColumn 3";
//Output: Column 1        Column 2        Column 3

string rows = "Row 1\r\nRow 2\r\nRow 3";
/* Output:
  Row 1
  Row 2
  Row 3
*/

string title = "\"The \u00C6olean Harp\", by Samuel Taylor Coleridge";
//Output: "The Æolean Harp", by Samuel Taylor Coleridge
```

Utilice cadenas textuales para mayor comodidad y mejor legibilidad cuando el texto de la cadena contenga caracteres de barra diagonal inversa, por ejemplo, en rutas de acceso de archivo. Como las cadenas textuales conservan los caracteres de nueva línea como parte del texto de la cadena, pueden utilizarse para inicializar cadenas multilíneas. Utilice comillas dobles para insertar una comilla simple dentro de una cadena textual. En el ejemplo siguiente se muestran algunos usos habituales de las cadenas textuales:

```
string filePath = @"C:\Users\scoleridge\Documents\";
//Output: C:\Users\scoleridge\Documents\

string text = @"My pensive SARA ! thy soft cheek reclined
    Thus on mine arm, most soothing sweet it is
    To sit beside our Cot,...";
/* Output:
```

```
My pensive SARA ! thy soft cheek reclined
   Thus on mine arm, most soothing sweet it is
   To sit beside our Cot,...
*/

string quote = @"Her name was ""Sara.""";
//Output: Her name was "Sara."
```

Secuencias de escape de cadena

Secuencia de escape	Nombre de carácter	Codificación Unicode
\'	Comilla simple	0x0027
\"	Comilla doble	0x0022
\\	Barra diagonal inversa	0x005C
\0	Null	0x0000
\a	Alerta	0x0007
\b	Retroceso	0x0008
\f	Avance de página	0x000C
\n	Nueva línea	0x000A
\r	Retorno de carro	0x000D
\t	Tabulación horizontal	0x0009
\U	Secuencia de escape Unicode para pares suplentes.	\Unnnnnnnn
\u	Secuencia de escape Unicode	\u0041 = "A"
\v	Tabulación vertical	0x000B
\x	Secuencia de escape Unicode similar a "\u"	\x0041 o \x41 = "A"

Secuencia de escape	Nombre de carácter	Codificación Unicode
	excepto con longitud variable.	

En tiempo de compilación, las cadenas textuales se convierten en cadenas normales con las mismas secuencias de escape. Por lo tanto, si se muestra una cadena textual en la ventana Inspección del depurador, verá los caracteres de escape agregados por el compilador, no la versión textual del código fuente. Por ejemplo, la cadena textual @"C:\files.txt" aparecerá en la ventana Inspección, como "C:\\files.txt".

Cadenas de formato

Una cadena de formato es una cadena cuyo contenido se determina de manera dinámica en tiempo de ejecución. Las cadenas de formato se crean mediante la inserción de *expresiones interpoladas* o marcadores de posición entre llaves dentro de una cadena. Todo lo incluido entre llaves (`{...}`) se resolverá en un valor y se generará como una cadena con formato en tiempo de ejecución. Existen dos métodos para crear cadenas de formato: interpolación de cadenas y formato compuesto.

Interpolación de cadenas

Disponible en C# 6.0 y versiones posteriores, las cadenas interpoladas se identifican por el carácter especial $ e incluyen expresiones interpoladas entre llaves. Si no está familiarizado con la interpolación de cadenas, consulte el tutorial interactivo Interpolación de cadenas en C# para obtener información general rápidamente.

Use la interpolación de cadenas para mejorar la legibilidad y el mantenimiento del código. Con la interpolación de cadenas se obtienen los mismos resultados que con el método String.Format, pero mejora la facilidad de uso y la claridad en línea.

```
var jh = (firstName: "Jupiter", lastName: "Hammon", born: 1711, published:
1761);
Console.WriteLine($"{jh.firstName} {jh.lastName} was an African American poet
born in {jh.born}.");
```

```
Console.WriteLine($"He was first published in {jh.published} at the age of
{jh.published - jh.born}.");
Console.WriteLine($"He'd be over {Math.Round((2018d - jh.born) / 100d) *
100d} years old today.");

// Output:
// Jupiter Hammon was an African American poet born in 1711.
// He was first published in 1761 at the age of 50.
// He'd be over 300 years old today.
```

Formatos compuestos

String.Format emplea marcadores de posición entre llaves para crear una cadena de formato. Los resultados de este ejemplo son similares a la salida del método de interpolación de cadenas usado anteriormente.

```
var pw = (firstName: "Phillis", lastName: "Wheatley", born: 1753, published:
1773);
Console.WriteLine("{0} {1} was an African American poet born in {2}.",
pw.firstName, pw.lastName, pw.born);
Console.WriteLine("She was first published in {0} at the age of {1}.",
pw.published, pw.published - pw.born);
Console.WriteLine("She'd be over {0} years old today.", Math.Round((2018d -
pw.born) / 100d) * 100d);

// Output:
// Phillis Wheatley was an African American poet born in 1753.
// She was first published in 1773 at the age of 20.
// She'd be over 300 years old today.
```

Subcadenas

Una subcadena es cualquier secuencia de caracteres que se encuentra en una cadena. Use el método Substring para crear una nueva cadena de una parte de la cadena original. Puede buscar una o más apariciones de una subcadena con el método IndexOf. Use el método Replace para reemplazar todas las apariciones de una subcadena especificada por una nueva cadena. Al igual que el método Substring, Replace devuelve una cadena nueva y no modifica la cadena original.

```
string s3 = "Visual C# Express";
System.Console.WriteLine(s3.Substring(7, 2));
// Output: "C#"

System.Console.WriteLine(s3.Replace("C#", "Basic"));
// Output: "Visual Basic Express"
```

```
// Index values are zero-based
int index = s3.IndexOf("C");
// index = 7
```

Acceso a caracteres individuales

Puede utilizar la notación de matriz con un valor de índice para adquirir acceso de solo lectura a caracteres individuales, como en el ejemplo siguiente:

```
string s5 = "Printing backwards";

for (int i = 0; i < s5.Length; i++)
{
    System.Console.Write(s5[s5.Length - i - 1]);
}
// Output: "sdrawkcab gnitnirP"
```

Si el método String no proporciona la funcionalidad que debe tener para modificar los caracteres individuales de una cadena, puede usar un objeto StringBuilder para modificar los caracteres individuales "en contexto" y, después, crear una cadena para almacenar los resultados mediante el método StringBuilder. En el ejemplo siguiente, se supone que debe modificar la cadena original de una manera determinada y, después, almacenar los resultados para un uso futuro:

```
string question = "hOW DOES mICROSOFT wORD DEAL WITH THE cAPS lOCK KEY?";
System.Text.StringBuilder sb = new System.Text.StringBuilder(question);

for (int j = 0; j < sb.Length; j++)
{
    if (System.Char.IsLower(sb[j]) == true)
        sb[j] = System.Char.ToUpper(sb[j]);
    else if (System.Char.IsUpper(sb[j]) == true)
        sb[j] = System.Char.ToLower(sb[j]);
}
// Store the new string.
string corrected = sb.ToString();
System.Console.WriteLine(corrected);
// Output: How does Microsoft Word deal with the Caps Lock key?
```

Cadenas nulas y cadenas vacías

Una cadena vacía es una instancia de un objeto System.String que contiene cero caracteres. Las cadenas vacías se utilizan a menudo en distintos escenarios de programación para representar un campo de texto en blanco. Puede llamar a

métodos en cadenas vacías porque son objetos System.String válidos. Las cadenas vacías se inicializan como sigue:

```
string s = String.Empty;
```

En cambio, una cadena nula no hace referencia a una instancia de un objeto System.String y cualquier intento de llamar a un método en una cadena nula produce una excepción NullReferenceException. Sin embargo, puede utilizar cadenas nulas en operaciones de comparación y concatenación con otras cadenas. Los ejemplos siguientes muestran algunos casos en que una referencia a una cadena nula provoca y no provoca una excepción:

```
static void Main()
{
    string str = "hello";
    string nullStr = null;
    string emptyStr = String.Empty;

    string tempStr = str + nullStr;
    // Output of the following line: hello
    Console.WriteLine(tempStr);

    bool b = (emptyStr == nullStr);
    // Output of the following line: False
    Console.WriteLine(b);

    // The following line creates a new empty string.
    string newStr = emptyStr + nullStr;

    // Null strings and empty strings behave differently. The following
    // two lines display 0.
    Console.WriteLine(emptyStr.Length);
    Console.WriteLine(newStr.Length);
    // The following line raises a NullReferenceException.
    //Console.WriteLine(nullStr.Length);

    // The null character can be displayed and counted, like other chars.
    string s1 = "\x0" + "abc";
    string s2 = "abc" + "\x0";
    // Output of the following line: * abc*
    Console.WriteLine("*" + s1 + "*");
    // Output of the following line: *abc *
    Console.WriteLine("*" + s2 + "*");
    // Output of the following line: 4
    Console.WriteLine(s2.Length);
}
```