StringBuilder

Representa una cadena de caracteres mutable. Esta clase no puede heredarse.

Las operaciones de cadena en .NET están muy optimizadas y en la mayoría de los casos no afectan significativamente al rendimiento. Sin embargo, en algunos escenarios, como los bucles de pequeñas dimensiones que se ejecutan cientos o miles de veces, las operaciones de cadena pueden afectar al rendimiento. La clase StringBuilder crea un búfer de cadena que proporciona un mejor rendimiento si el programa realiza muchas manipulaciones de cadenas. La cadena StringBuildertambién permite reasignar caracteres individuales, algo que el tipo de datos de cadena integrado no admite. Por ejemplo, este código cambia el contenido de una cadena sin crear una nueva:

El StringBuilder.Length propiedad indica el número de caracteres del StringBuilder actualmente contiene el objeto. Si se agregan caracteres a la StringBuilder objeto, su longitud se incrementa siempre que sea igual que el tamaño de la StringBuilder.Capacity propiedad, que define el número de caracteres que puede contener el objeto. Si el número de caracteres agregados hace que la longitud de la StringBuilder objeto supere su capacidad actual, la nueva memoria se asigna el valor de la Capacity se duplica la propiedad, se agregan nuevos caracteres a la StringBuilder objeto y su Length se ajusta la propiedad. Memoria adicional para el StringBuilder objeto se asigna dinámicamente hasta que alcanza el valor definido por el StringBuilder.MaxCapacity propiedad.Cuando se alcanza la capacidad máxima, no se puede asignar ninguna memoria adicional para el StringBuilder objeto y al intentar agregar caracteres o se expande más allá de su capacidad máxima inicia ya sea un ArgumentOutOfRangeException o un OutOfMemoryException excepción.

El ejemplo siguiente se muestra cómo un StringBuilder objeto asigna memoria nueva y aumenta su capacidad dinámicamente a medida se expande la cadena asignada al objeto. El código crea un StringBuilder objeto mediante una llamada a su constructor (sin parámetros) predeterminado. La capacidad predeterminada de este objeto es 16 caracteres y su capacidad máxima es de más de 2 millones de caracteres. Anexar la cadena "Es una frase". da como resultado una nueva asignación de memoria porque la longitud de cadena (19 caracteres) supera la capacidad predeterminada de la StringBuilder objeto. Duplica la capacidad del objeto y 32 caracteres, se agrega la nueva cadena y la longitud del objeto ahora es igual a 19 caracteres. El código, a continuación, anexa la cadena "Es una frase adicional". en el valor de la StringBuilder objeto 11 veces. Cada vez que la operación de anexado hace que la longitud de la StringBuilder objeto supere su capacidad, su capacidad existente se duplica y Append operación se realiza correctamente.

```
using System;
using System.Reflection;
using System.Text;

public class Example
{
   public static void Main()
   {
      StringBuilder sb = new StringBuilder();
      ShowSBInfo(sb);
      sb.Append("This is a sentence.");
      ShowSBInfo(sb);
      for (int ctr = 0; ctr <= 10; ctr++) {
         sb.Append("This is an additional sentence.");
         ShowSBInfo(sb);
      }
   }

   private static void ShowSBInfo(StringBuilder sb)
   {
      foreach (var prop in sb.GetType().GetProperties()) {
         if (prop.GetIndexParameters().Length == 0)
            Console.Write("{0}: {1:N0}    ", prop.Name, prop.GetValue(sb));
      }
      Console.WriteLine();
   }
}
// The example displays the following output:
//    Capacity: 16    MaxCapacity: 2,147,483,647    Length: 0
//    Capacity: 32    MaxCapacity: 2,147,483,647    Length: 19
//    Capacity: 64    MaxCapacity: 2,147,483,647    Length: 50
//    Capacity: 128    MaxCapacity: 2,147,483,647    Length: 81
//    Capacity: 128    MaxCapacity: 2,147,483,647    Length: 112
//    Capacity: 256    MaxCapacity: 2,147,483,647    Length: 143
//    Capacity: 256    MaxCapacity: 2,147,483,647    Length: 174
//    Capacity: 256    MaxCapacity: 2,147,483,647    Length: 205
//    Capacity: 256    MaxCapacity: 2,147,483,647    Length: 236
//    Capacity: 512    MaxCapacity: 2,147,483,647    Length: 267
//    Capacity: 512    MaxCapacity: 2,147,483,647    Length: 298
//    Capacity: 512    MaxCapacity: 2,147,483,647    Length: 329
//    Capacity: 512    MaxCapacity: 2,147,483,647    Length: 360
```

Asignación de memoria

La capacidad predeterminada de un StringBuilder objeto es 16 caracteres y su capacidad máxima del valor predeterminado es Int32.MaxValue. Estos valores predeterminados se utilizan si se llama a
la StringBuilder() y StringBuilder(String) constructores.

Puede definir explícitamente la capacidad inicial de un StringBuilder objeto de las maneras siguientes:

- Al llamar a cualquiera de los StringBuilder constructores que incluye un capacity parámetro cuando se crea el objeto.
- Asignando explícitamente un valor nuevo a la StringBuilder.Capacity propiedad para expandir una existente StringBuilder objeto. Tenga en cuenta que la propiedad produce una excepción si la nueva capacidad sea menor que el existente capacidad o mayor que el StringBuilder capacidad máxima del objeto.
- Mediante una llamada a la StringBuilder.EnsureCapacity método con la nueva capacidad. La nueva capacidad no debe ser mayor que el StringBuilder capacidad máxima del objeto. Sin embargo, a diferencia de una asignación a la Capacity propiedad EnsureCapacity no produce una excepción si la nueva capacidad deseada es menor que la capacidad existente; en este caso, la llamada al método no tiene ningún efecto.

Si la longitud de la cadena asignada a la StringBuilder objeto en la llamada al constructor supera la capacidad predeterminada o la capacidad especificada, el Capacity propiedad está establecida en la longitud de la cadena especificada con el value parámetro.

Puede definir explícitamente la capacidad máxima de un StringBuilder objeto mediante una llamada a la StringBuilder(Int32, Int32) constructor. No se puede cambiar la capacidad máxima, se asigna un nuevo valor para el MaxCapacity propiedad, porque es de solo lectura.

La sección anterior se muestra, siempre que la capacidad existente es inadecuado, más memoria se asigna y la capacidad de un StringBuilder objeto dobles hasta el valor definido por el MaxCapacity propiedad.

En general, la capacidad predeterminada y la capacidad máxima son adecuados para la mayoría de las aplicaciones. Considere la posibilidad de establecer estos valores en las siguientes condiciones:

- Si el tamaño final de la StringBuilder objeto es probable que crezcan extremadamente grande, normalmente por encima de varios megabytes. En este caso, puede haber algunas ventajas de rendimiento desde la configuración inicial Capacity propiedad en un valor alto significativamente para eliminar la necesidad de demasiados reasignaciones de memoria.
- Si la aplicación se ejecuta en un sistema con memoria limitada. En este caso, puede desear considerar la configuración del MaxCapacity propiedad a menos de Int32.MaxValue si la aplicación está controlando las cadenas

de gran tamaño que pueden provocar que se ejecutan en un entorno con restricciones de memoria.

Instancias de un objeto StringBuilder

Crea instancias de un StringBuilder objeto llamando a uno de sus seis constructores de clase sobrecargados, que se enumeran en la tabla siguiente. Crear tres de los constructores de instancias de un StringBuilder objeto cuyo valor es una cadena vacía, pero establecer su Capacity y MaxCapacity valores de forma diferente. Definen los tres constructores restantes un StringBuilder objeto que tiene un valor de cadena específico y la capacidad. Dos de los tres constructores usan la capacidad máxima predeterminada de Int32.MaxValue, mientras que la tercera le permite establecer la capacidad máxima.

Llamar a métodos de StringBuilder

La mayoría de los métodos que modifican la cadena en un StringBuilder instancia devuelven una referencia a esa misma instancia. Esto le permite llamar a StringBuilder métodos de dos maneras:

- Puede realizar llamadas a métodos individuales y omitir el valor devuelto, como el siguiente ejemplo.

```
using System;
using System.Text;

public class Example
{
   public static void Main()
   {
      StringBuilder sb = new StringBuilder();
      sb.Append("This is the beginning of a sentence, ");
      sb.Replace("the beginning of ", "");
      sb.Insert(sb.ToString().IndexOf("a ") + 2, "complete ");
      sb.Replace(",", ".");
      Console.WriteLine(sb.ToString());
   }
}
// The example displays the following output:
//        This is a complete sentence.
```

- Puede realizar una serie de llamadas de método en una sola instrucción. Esto puede ser conveniente si desea escribir una sola

instrucción que se vincula operaciones sucesivas. El ejemplo siguiente consolida tres llamadas de método del ejemplo anterior en una sola línea de código.

```
using System;
using System.Text;

public class Example
{
   public static void Main()
   {
      StringBuilder sb = new StringBuilder("This is the beginning of a
sentence, ");
      sb.Replace("the beginning of ",
"").Insert(sb.ToString().IndexOf("a ") + 2,
                                                "complete
").Replace(",", ".");
      Console.WriteLine(sb.ToString());
   }
}
// The example displays the following output:
//        This is a complete sentence.
```

Realización de operaciones de StringBuilder

Puede usar los métodos de la StringBuilder clase para recorrer en iteración, agregar, eliminar o modificar los caracteres de un StringBuilder objeto.
Recorrer en iteración los caracteres de StringBuilder

Puede tener acceso a los caracteres de un StringBuilder objeto utilizando el StringBuilder.Chars[Int32] propiedad. En C#, Chars[Int32] es un indizador; en Visual Basic, es la propiedad predeterminada de la StringBuilder clase. Esto le permite establecer o recuperar los caracteres individuales utilizando su índice solo, sin hacer referencia explícita a la Chars[Int32]propiedad. Los caracteres de un StringBuilder objeto comienzan en el índice 0 (cero) y continuar indizar Length - 1.

El ejemplo siguiente ilustra la Chars[Int32] propiedad. Anexa diez números aleatorios a una StringBuilder de objetos y, a continuación, recorre en iteración cada carácter. Si es la categoría del carácter Unicode UnicodeCategory.DecimalDigitNumber, disminuye la cantidad en 1 (o cambia el número a 9 si su valor es 0). En el ejemplo se muestra el contenido de la StringBuilder objeto ambos antes y después de que se han cambiado los valores de caracteres individuales.

```
using System;
using System.Globalization;
using System.Text;

public class Example
{
   public static void Main()
   {
      Random rnd = new Random();
      StringBuilder sb = new StringBuilder();

      // Generate 10 random numbers and store them in a StringBuilder.
      for (int ctr = 0; ctr <= 9; ctr++)
         sb.Append(rnd.Next().ToString("N5"));

      Console.WriteLine("The original string:");
      Console.WriteLine(sb.ToString());

      // Decrease each number by one.
      for (int ctr = 0; ctr < sb.Length; ctr++) {
         if (Char.GetUnicodeCategory(sb[ctr]) ==
UnicodeCategory.DecimalDigitNumber) {
            int number = (int) Char.GetNumericValue(sb[ctr]);
            number--;
            if (number < 0) number = 9;

            sb[ctr] = number.ToString()[0];
         }
      }
      Console.WriteLine("\nThe new string:");
      Console.WriteLine(sb.ToString());
   }
}
// The example displays the following output:
//    The original string:
//
1,457,531,530.00000940,522,609.000001,668,113,564.000001,998,992,883.000001,7
92,660,834.00
//
000101,203,251.000002,051,183,075.000002,066,000,067.000001,643,701,043.00000
1,702,382,508
//    .00000
//
//    The new string:
//
0,346,420,429.99999839,411,598.999990,557,002,453.999990,887,881,772.999990,6
81,559,723.99
```

```
//
999090,192,140.999991,940,072,964.999991,955,999,956.999990,532,690,932.99999
0,691,271,497
//    .99999
```

El uso de la indexación basada en caracteres con la propiedad Chars[Int32] puede ser muy lento en las condiciones siguientes:

- La instancia de StringBuilder es grande (por ejemplo, consta de varias decenas de miles de caracteres).
- StringBuilder es "pesado". Es decir, las llamadas repetidas a métodos como StringBuilder.Append han ampliado automáticamente la propiedad StringBuilder.Capacity del objeto y le han asignado nuevos fragmentos de memoria.

El rendimiento se ve seriamente afectado ya que cada acceso de carácter recorre toda la lista vinculada de fragmentos para buscar el búfer correcto en el que indexar.

Incluso para un gran objeto **StringBuilder** pesado, el uso de la propiedad **Chars[Int32]** para el acceso basado en índice a uno o un número reducido de caracteres tiene un efecto insignificante en el rendimiento; por lo general, es una operación **0(n)**. El impacto significativo en el rendimiento se produce al recorrer en iteración los caracteres del objeto **StringBuilder**, una operación **O(n^2)**.

Si encuentra problemas de rendimiento al usar la indexación basada en caracteres con objetos StringBuilder, puede usar cualquiera de las soluciones alternativas siguientes:

- Convertir la instancia de StringBuilder en una String mediante una llamada al método ToStringy después obtener acceso a los caracteres de la cadena.
- Copiar el contenido del objeto StringBuilder existente en un objeto StringBuilder nuevo de tamaño predefinido. El rendimiento mejora porque el objeto StringBuilder nuevo no es pesado. Por ejemplo:

  ```
  // sbOriginal is the existing StringBuilder object
  var sbNew = new StringBuilder(sbOriginal.ToString(), sbOriginal.Length);
  ```

- Establezca la capacidad inicial del objeto StringBuilder en un valor que sea aproximadamente igual a su tamaño máximo esperado mediante una llamada al constructor StringBuilder(Int32).Tenga en cuenta que esto asigna el bloque completo de memoria aunque StringBuilder rara vez alcanza su capacidad máxima.

Agregar texto a un objeto StringBuilder

El StringBuilder clase incluye los siguientes métodos para expandir el contenido de un StringBuilderobjeto:

- El Append método anexa una cadena, una subcadena, una matriz de caracteres, una parte de una matriz de caracteres, un carácter único se repite varias veces, o tipo de representación de cadena de datos primitivos por un StringBuilder objeto.
- El AppendLine método anexa un terminador de línea o una cadena junto con un terminador de línea en un StringBuilder objeto.
- El AppendFormat método anexa una cadena de formato compuesto a un StringBuilder objeto.Las representaciones de cadena de objetos incluidos en la cadena de resultado pueden reflejar las convenciones de formato de la referencia cultural actual del sistema o una referencia cultural especificada.
- El Insert método inserta una cadena, una subcadena, varias repeticiones de una cadena, una matriz de caracteres, una parte de una matriz de caracteres o la representación de cadena de datos primitivos que se escriba en una posición especificada en el StringBuilder objeto. La posición se define mediante un índice de base cero.

En el ejemplo siguiente se usa el Append, AppendLine, AppendFormat, y Insert métodos para expandir el texto de un StringBuilder objeto.

```
using System;
using System.Text;

public class Example
{
   public static void Main()
   {
      // Create a StringBuilder object with no text.
      StringBuilder sb = new StringBuilder();
      // Append some text.
      sb.Append('*', 10).Append(" Adding Text to a StringBuilder Object
").Append('*', 10);
      sb.AppendLine("\n");
      sb.AppendLine("Some code points and their corresponding characters:");
      // Append some formatted text.
      for (int ctr = 50; ctr <= 60; ctr++) {
         sb.AppendFormat("{0,12:X4} {1,12}", ctr, Convert.ToChar(ctr));
         sb.AppendLine();
      }
      // Find the end of the introduction to the column.
      int pos = sb.ToString().IndexOf("characters:") + 11 +
```

```
                Environment.NewLine.Length;
      // Insert a column header.
      sb.Insert(pos, String.Format("{2}{0,12:X4} {1,12}{2}", "Code Unit",
                                   "Character", "\n"));

      // Convert the StringBuilder to a string and display it.
      Console.WriteLine(sb.ToString());
   }
}
// The example displays the following output:
//    ********** Adding Text to a StringBuilder Object **********
//
//    Some code points and their corresponding characters:
//
//       Code Unit    Character
//            0032            2
//            0033            3
//            0034            4
//            0035            5
//            0036            6
//            0037            7
//            0038            8
//            0039            9
//            003A            :
//            003B            ;
//            003C            <
```

Eliminación de texto de un objeto StringBuilder

El StringBuilder clase incluye métodos que pueden reducir el tamaño del elemento actual StringBuilder instancia. El Clear método quita todos los caracteres y establece el Length propiedad en cero. El Remove método elimina un número especificado de caracteres a partir de una posición de índice determinada. Además, puede quitar los caracteres del final de un StringBuilder objeto estableciendo sus Length propiedad con un valor que es menor que la longitud de la instancia actual.

En el ejemplo siguiente se quita alguna parte del texto de un StringBuilder objeto, muestra su capacidad resultante, capacidad máxima y los valores de propiedad de longitud y, a continuación, llama a la Clear método para quitar todos los caracteres desde la StringBuilder objeto.

```
using System;
using System.Text;

public class Example
{
```

```
   public static void Main()
   {
      StringBuilder sb = new StringBuilder("A StringBuilder object");
      ShowSBInfo(sb);
      // Remove "object" from the text.
      string textToRemove = "object";
      int pos = sb.ToString().IndexOf(textToRemove);
      if (pos >= 0) {
         sb.Remove(pos, textToRemove.Length);
         ShowSBInfo(sb);
      }
      // Clear the StringBuilder contents.
      sb.Clear();
      ShowSBInfo(sb);
   }

   public static void ShowSBInfo(StringBuilder sb)
   {
      Console.WriteLine("\nValue: {0}", sb.ToString());
      foreach (var prop in sb.GetType().GetProperties()) {
         if (prop.GetIndexParameters().Length == 0)
            Console.Write("{0}: {1:N0}    ", prop.Name, prop.GetValue(sb));
      }
      Console.WriteLine();
   }
}
// The example displays the following output:
//    Value: A StringBuilder object
//    Capacity: 22    MaxCapacity: 2,147,483,647    Length: 22
//
//    Value: A StringBuilder
//    Capacity: 22    MaxCapacity: 2,147,483,647    Length: 16
//
//    Value:
//    Capacity: 22    MaxCapacity: 2,147,483,647    Length: 0
```

Modificar el texto en un objeto StringBuilder

El StringBuilder.Replace método reemplaza todas las apariciones de un carácter o una cadena en todo el StringBuilder objeto o en un intervalo de caracteres determinada. En el ejemplo siguiente se usa el Replace método para reemplazar todos los signos de exclamación (!) con signos de interrogación (?) en el StringBuilder objeto.

```
using System;
using System.Text;

public class Example
```

```
{
   public static void Main()
   {
      StringBuilder MyStringBuilder = new StringBuilder("Hello World!");
      MyStringBuilder.Replace('!', '?');
      Console.WriteLine(MyStringBuilder);
   }
}
// The example displays the following output:
//       Hello World?
```

Buscar el texto en un objeto StringBuilder

El StringBuilder clase no incluye métodos similares a los String.Contains, String.IndexOf, y String.StartsWith métodos proporcionados por el String (clase), que le permiten buscar el objeto para un determinado carácter o una subcadena. Determinar la presencia o la posición de carácter de una subcadena inicial requiere que buscar un String valor mediante un método de búsqueda de cadena o un método de expresión regular. Hay cuatro maneras de implementar dichas búsquedas, como se muestra en la tabla siguiente.

Convertir el objeto StringBuilder en string

Debe convertir primero el objeto StringBuilder en un objeto String para poder pasar la cadena representada por el objeto StringBuilder a un método con un parámetro String o mostrarla en la interfaz de usuario. Realizar esta conversión mediante una llamada a la StringBuilder.ToStringmétodo. Para ver una ilustración, vea el ejemplo anterior, que llama a la ToString método para convertir un StringBuilder objeto en una cadena para que se puede pasar a un método de expresión regular.