Strings and String Builders

Strings

* Immutability of strings
* String functions
* string vs System.String
* String Escape sequence list

String Builders

# Strings

A string is a built-in / predefined data type available in the .NET framework that is used to store a sequence of characters as a single string literal. This can be a combination of Alpha Numeric characters and special characters as all fall in the ambit of UNICODE characters.

**Important**:

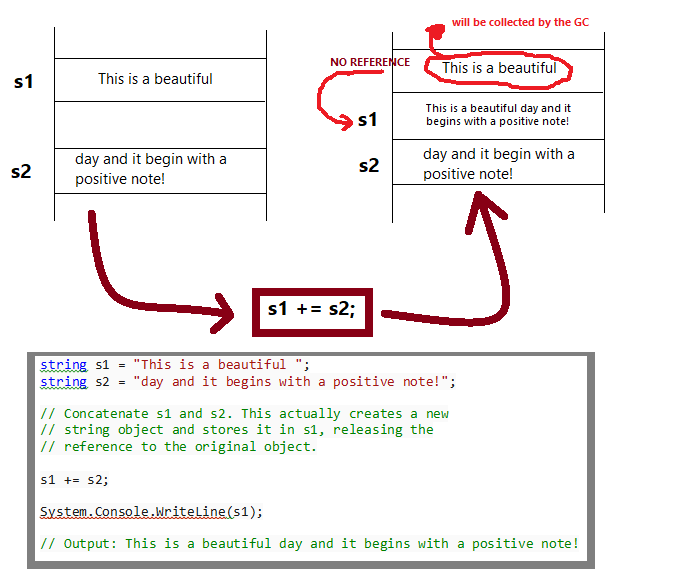
1) There is no null-terminating character at the end of a C# string; therefore a C# string can contain any number of embedded null characters ('\0').

2) The Length property of a string represents the number of Char objects it contains, not the number of Unicode characters.

3) The new operator is used to create a string only when the programmer has to initialize the string with an array of chars.

4) *string str1 = null* & *string str1 = “”*  are not the same. The first one is a null value and the second one is an empty string. Use the IsNullOrEmpty(String) function to check whether the string is null or empty. The programmer can reduce the chances of  NullReferenceException occurring.

## Strings are IMMUTABLE

Strings are immutable which means that they cannot be changed after they have been created.  Look into Fig.1 and the stages 

**Fig. #1**: String immutability

## Standard String Functions

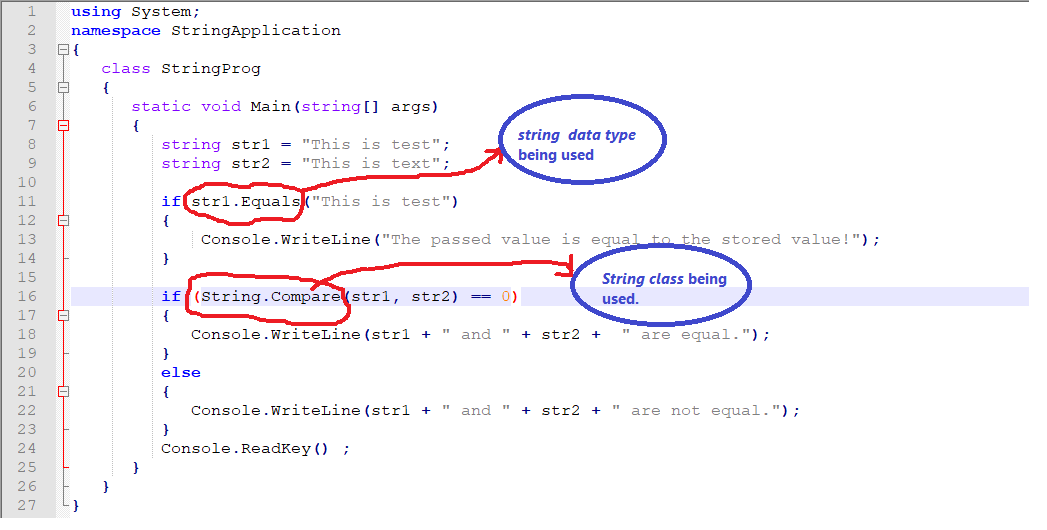
|  |  |
| --- | --- |
| **SL.NO** | **STRING** **FUNCTION** |
| 1 | **public bool Equals (string value)**  Determines whether the current String object and the specified String object have the same value. |
| 2 | **public static bool Equals (string a, string b)**  Determines whether two specified String objects have the same value. |
| 3 | **public bool StartsWith (string value)**  Determines whether the beginning of this string instance matches the specified string. |
| 4 | **public bool EndsWith (string value)**  Determines whether the end of the string object matches the specified string. |
| 5 | **public bool Contains (string value)**  Returns a value indicating whether the specified String object occurs within this string. |
| 6 | **public static string Concat (string str0, string str1)**  Concatenates two string objects. |
| 7 | **public static string Concat (string str0, string str1, string str2)**  Concatenates three string objects. |
| 8 | **public static string Concat (string str0, string str1, string str2, string str3)**  Concatenates four string objects. |
| 9 | **public static int Compare(string strA, string strB)**  Compares two specified string objects and returns an integer that indicates their relative position in the sort order. |
| 10 | **public static int Compare (string strA, string strB, bool ignoreCase)**  Compares two specified string objects and returns an integer that indicates their relative position in the sort order. However, it ignores case if the Boolean parameter is true. |
| 11 | **public static string Copy (string str)**  Creates a new String object with the same value as the specified string. |
| 12 | **public void CopyTo (int sourceIndex, char[] destination, int destinationIndex, int count)**  Copies a specified number of characters from a specified position of the String object to a specified position in an array of Unicode characters. |
| 13 | **public string Insert (int startIndex, string value)**  Returns a new string in which a specified string is inserted at a specified index position in the current string object. |
| 14 | **public string Remove (int startIndex)**  Removes all the characters in the current instance, beginning at a specified position and continuing through the last position, and returns the string. |
| 15 | **public string Remove (int startIndex, int count)**  Removes the specified number of characters in the current string beginning at a specified position and returns the string. |
| 16 | **public string Replace(char oldChar, char newChar)**  Replaces all occurrences of a specified Unicode character in the current string object with the specified Unicode character and returns the new string. |
| 17 | **public string Replace(string oldValue, string newValue)**  Replaces all occurrences of a specified string in the current string object with the specified string and returns the new string. |
| 18 | **public string[] Split (params char[] separator)**  Returns a string array that contains the substrings in the current string object, delimited by elements of a specified Unicode character array. |
| 19 | **public string[] Split (char[] separator, int count)**  Returns a string array that contains the substrings in the current string object, delimited by elements of a specified Unicode character array. The int parameter specifies the maximum number of substrings to return. |
| 20 | **public string Trim()**  Removes all leading and trailing white-space characters from the current String object. |
| 21 | **public string ToLower()**  Returns a copy of this string converted to lowercase. |
| 22 | **public string ToUpper()**  Returns a copy of this string converted to uppercase. |
| 23 | **public char[] ToCharArray()**  Returns a Unicode character array with all the characters in the current string object. |
| 24 | **public char[] ToCharArray (int startIndex, int length)**  Returns a Unicode character array with all the characters in the current string object, starting from the specified index and up to the specified length. |
| 25 | **public int IndexOf (char value)**  Returns the zero-based index of the first occurrence of the specified Unicode character in the current string. |
| 26 | **public int IndexOf (string value)**  Returns the zero-based index of the first occurrence of the specified string in this instance. |
| 27 | **public int IndexOf (char value, int startIndex)**  Returns the zero-based index of the first occurrence of the specified Unicode character in this string, starting search at the specified character position. |
| 28 | **public int IndexOf (string value, int startIndex)**  Returns the zero-based index of the first occurrence of the specified string in this instance, starting search at the specified character position. |
| 29 | **public int IndexOfAny (char[] anyOf)**  Returns the zero-based index of the first occurrence in this instance of any character in a specified array of Unicode characters. |
| 30 | **public int IndexOfAny (char[] anyOf, int startIndex)**  Returns the zero-based index of the first occurrence in this instance of any character in a specified array of Unicode characters, starting search at the specified character position. |
| 31 | **public int LastIndexOf (char value)**  Returns the zero-based index position of the last occurrence of the specified Unicode character within the current string object. |
| 32 | **public int LastIndexOf (string value)**  Returns the zero-based index position of the last occurrence of a specified string within the current string object. |
| 33 | **public static string Format (string format, Object arg0)**  Replaces one or more format items in a specified string with the string representation of a specified object. |
| 34 | **public static string Join (string separator, string[] value, int startIndex, int count)**  Concatenates the specified elements of a string array, using the specified separator between each element. |
| 35 | **public static string Join(string separator, params string[] value)**  Concatenates all the elements of a string array, using the specified separator between each element. |
| 36 | **public static bool IsNullOrEmpty (string value)**  Indicates whether the specified string is null or an Empty string. |

Click here to download the function samples

## string vs. System.String

There is often confusion in string and String – string is a built-in / predefined data type available in the .NET framework whereas - String is a predefined class in the .NET Framework.

They are used interchangeably in the practical scenario. Fig.1 points out where to use string and where to use String.



**Fig.1** string keyword versus the String class

## String escape sequences list

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.** | **Escape sequence** | **Character name** | **Unicode encoding** |
| 1 | \' | Single quote | 0x0027 |
| 2 | \" | Double quote | 0x0022 |
| 3 | \\ | Backslash | 0x005C |
| 4 | \0 | Null | 0x0000 |
| 5 | \a | Alert | 0x0007 |
| 6 | \b | Backspace | 0x0008 |
| 7 | \f | Form feed | 0x000C |
| 8 | \n | New line | 0x000A |
| 9 | \r | Carriage return | 0x000D |
| 10 | \t | Horizontal tab | 0x0009 |
| 11 | \v | Vertical tab | 0x000B |
| 12 | \u | Unicode escape sequence (UTF-16) | \uHHHH (range: 0000 - FFFF; example: \u00E7 = "ç") |
| 13 | \U | Unicode escape sequence (UTF-32) | \U00HHHHHH (range: 000000 - 10FFFF; example: \U0001F47D = "👽") |
| 14 | \x | Unicode escape sequence similar to "\u" except with variable length | \xH[H][H][H] (range: 0 - FFFF; example: \x00E7 or \x0E7 or \xE7 = "ç") |

Click here to download the escape sequence samples

# String Builder

Strings in .NET are IMMUTABLE. This means that in some scenarios such as tight loops that are executing many hundreds or thousands of times, string operations can affect performance.

The StringBuilder class creates a string buffer that offers better performance if your program performs many string manipulations.

StringBuilder string also enables you to reassign individual characters, something the built-in string data type does not support. This code, for example, changes the content of a string without creating a new string.

In a nutshell, StringBuilder represents a mutable string of characters. This class cannot be inherited.

public sealed class StringBuilder : System.Runtime.Serialization.ISerializable

System.Text.StringBuilder sb = new System.Text.StringBuilder("Thi");

sb[0] = 'C';

System.Console.WriteLine(sb.ToString());

System.Console.ReadLine();

**Illustration #1** string keyword versus the String class

Click here to download the StringBuilder sample