StringCollection Class

Reference

Definition

Namespace: System.Collections.Specialized

Assembly: System.Collections.Specialized.dll

Represents a collection of strings.

C#

```
public class StringCollection : System.Collections.IList
```

Inheritance Object → StringCollection

Derived System.Configuration.CommaDelimitedStringCollection

Implements ICollection , IEnumerable , IList

Examples

The following code example demonstrates several of the properties and methods of StringCollection.

C#

```
using System;
using System.Collections;
using System.Collections.Specialized;

public class SamplesStringCollection  {

   public static void Main()  {

      // Create and initializes a new StringCollection.
      StringCollection myCol = new StringCollection();
      // Add a range of elements from an array to the end of the
StringCollection.
      String[] myArr = new String[] { "RED", "orange", "yellow", "RED",
"green", "blue", "RED", "indigo", "violet", "RED" };
      myCol.AddRange( myArr );

      // Display the contents of the collection using foreach. This is the
preferred method.
      Console.WriteLine( "Displays the elements using foreach:" );
      PrintValues1( myCol );

      // Display the contents of the collection using the enumerator.
      Console.WriteLine( "Displays the elements using the IEnumerator:" );
      PrintValues2( myCol );

      // Display the contents of the collection using the Count and Item prop-
erties.
      Console.WriteLine( "Displays the elements using the Count and Item prop-
erties:" );
      PrintValues3( myCol );

      // Add one element to the end of the StringCollection and insert another
at index 3.
      myCol.Add( "* white" );
      myCol.Insert( 3, "* gray" );

      Console.WriteLine( "After adding \"* white\" to the end and inserting
\"* gray\" at index 3:" );
      PrintValues1( myCol );

      // Remove one element from the StringCollection.
      myCol.Remove( "yellow" );

      Console.WriteLine( "After removing \"yellow\":" );
      PrintValues1( myCol );

      // Remove all occurrences of a value from the StringCollection.
      int i = myCol.IndexOf( "RED" );
      while ( i > -1 )  {
         myCol.RemoveAt( i );
         i = myCol.IndexOf( "RED" );
      }

      // Verify that all occurrences of "RED" are gone.
      if ( myCol.Contains( "RED" ) )
         Console.WriteLine( "*** The collection still contains \"RED\"." );

      Console.WriteLine( "After removing all occurrences of \"RED\":" );
      PrintValues1( myCol );

      // Copy the collection to a new array starting at index 0.
      // Copy the collection to a new array starting at index 0.
      String[] myArr2 = new String[myCol.Count];
      myCol.CopyTo( myArr2, 0 );


      Console.WriteLine( "The new array contains:" );
      for ( i = 0; i < myArr2.Length; i++ )  {
         Console.WriteLine( "   [{0}] {1}", i, myArr2[i] );
      }
      Console.WriteLine();

      // Clears the entire collection.
      myCol.Clear();

      Console.WriteLine( "After clearing the collection:" );
      PrintValues1( myCol );
   }

   // Uses the foreach statement which hides the complexity of the enumerator.
   // NOTE: The foreach statement is the preferred way of enumerating the con-
tents of a collection.
   public static void PrintValues1( StringCollection myCol )  {
      foreach ( Object obj in myCol )
         Console.WriteLine( "   {0}", obj );
      Console.WriteLine();
   }

   // Uses the enumerator.
   // NOTE: The foreach statement is the preferred way of enumerating the con-
tents of a collection.
   public static void PrintValues2( StringCollection myCol )  {
      StringEnumerator myEnumerator = myCol.GetEnumerator();
      while ( myEnumerator.MoveNext() )
         Console.WriteLine( "   {0}", myEnumerator.Current );
      Console.WriteLine();
   }

   // Uses the Count and Item properties.
   public static void PrintValues3( StringCollection myCol )  {
      for ( int i = 0; i < myCol.Count; i++ )
         Console.WriteLine( "   {0}", myCol[i] );
      Console.WriteLine();
   }
}

/*
This code produces the following output.

Displays the elements using foreach:
   RED
   orange
   yellow
   RED
   RED
   green
   blue
   RED
   indigo
   violet
   RED

Displays the elements using the IEnumerator:
   RED
   orange
   yellow
   RED
   green
   blue
   RED
   indigo
   violet
   RED

Displays the elements using the Count and Item properties:
   RED
   orange
   yellow
   RED
   green
   blue
   RED
   indigo
   violet
   RED

After adding "* white" to the end and inserting "* gray" at index 3:
   RED
   orange
   yellow
   * gray
   RED
   green
   blue
   RED
   indigo
   violet
   RED
   * white

After removing "yellow":
   RED
   orange
   * gray
   RED
   green
   blue
   RED

   indigo
   violet
   RED
   * white

After removing all occurrences of "RED":
   orange
   * gray
   green
   blue
   indigo
   violet
   * white

The new array contains:
   [0] orange
   [1] * gray
   [2] green
   [3] blue
   [4] indigo
   [5] violet
   [6] * white

After clearing the collection:

*/
```

Remarks

StringCollection accepts `null` as a valid value and allows duplicate elements.

String comparisons are case-sensitive.

Elements in this collection can be accessed using an integer index. Indexes in this collection are zero-based.

Constructors

StringCollection()	Initializes a new instance of the StringCollection class.

Properties

Count	Gets the number of strings contained in the StringCollection.
IsReadOnly	Gets a value indicating whether the StringCollection is read-only.
IsSynchronized	Gets a value indicating whether access to the StringCollection is synchronized (thread safe).
Item[Int32]	Gets or sets the element at the specified index.
SyncRoot	Gets an object that can be used to synchronize access to the StringCollection.

Methods

Add(String)	Adds a string to the end of the StringCollection.
AddRange(String[])	Copies the elements of a string array to the end of the StringCollection.
Clear()	Removes all the strings from the StringCollection.
Contains(String)	Determines whether the specified string is in the StringCollection.
CopyTo(String[], Int32)	Copies the entire StringCollection values to a one-dimensional array of strings, starting at the specified index of the target array.
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
GetEnumerator()	Returns a StringEnumerator that iterates through the StringCollection.
GetHashCode()	Serves as the default hash function. (Inherited from Object)
GetType()	Gets the Type of the current instance. (Inherited from Object)
IndexOf(String)	Searches for the specified string and returns the zero-based index of the first occurrence within the StringCollection.
Insert(Int32, String)	Inserts a string into the StringCollection at the specified index.
MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object)

Remove(String)	Removes the first occurrence of a specific string from the StringCollection.
RemoveAt(Int32)	Removes the string at the specified index of the StringCollection.
ToString()	Returns a string that represents the current object. (Inherited from Object)

Explicit Interface Implementations

ICollection.CopyTo(Array, Int32)	Copies the entire StringCollection to a compatible one-dimensional Array, starting at the specified index of the target array.
IEnumerable.GetEnumerator()	Returns a IEnumerator that iterates through the StringCollection.
IList.Add(Object)	Adds an object to the end of the StringCollection.
IList.Contains(Object)	Determines whether an element is in the StringCollection.
IList.IndexOf(Object)	Searches for the specified Object and returns the zero-based index of the first occurrence within the entire StringCollection.
IList.Insert(Int32, Object)	Inserts an element into the StringCollection at the specified index.
IList.IsFixedSize	Gets a value indicating whether the StringCollection object has a fixed size.
IList.IsReadOnly	Gets a value indicating whether the StringCollection object is read-only.
IList.Item[Int32]	Gets or sets the element at the specified index.
IList.Remove(Object)	Removes the first occurrence of a specific object from the StringCollection.

Extension Methods

Cast<TResult>(IEnumerable)	Casts the elements of an IEnumerable to the specified type.
OfType<TResult>(IEnumerable)	Filters the elements of an IEnumerable based on a specified type.
AsParallel(IEnumerable)	Enables parallelization of a query.
AsQueryable(IEnumerable)	Converts an IEnumerable to an IQueryable.