Stack.Push(Object) Method

Namespace: System.Collections

Assemblies: System.Collections.NonGeneric.dll, mscorlib.dll, netstandard.dll

In this article

Inserts an object at the top of the Stack.

C# Copy

```
public virtual void Push (object obj);
```

Parameters

obj Object

The Object to push onto the Stack. The value can be `null`.

Examples

The following example shows how to add elements to the Stack, remove elements from the Stack, or view the element at the top of the Stack.

C# Copy

```
using System;
using System.Collections;
public class SamplesStack  {

   public static void Main()  {

      // Creates and initializes a new Stack.
      Stack myStack = new Stack();
      myStack.Push( "The" );
      myStack.Push( "quick" );
```

```
      myStack.Push( "brown" );
      myStack.Push( "fox" );

      // Displays the Stack.
      Console.Write( "Stack values:" );
      PrintValues( myStack, '\t' );

      // Removes an element from the Stack.
      Console.WriteLine( "(Pop)\t\t{0}", myStack.Pop() );

      // Displays the Stack.
      Console.Write( "Stack values:" );
      PrintValues( myStack, '\t' );

      // Removes another element from the Stack.
      Console.WriteLine( "(Pop)\t\t{0}", myStack.Pop() );

      // Displays the Stack.
      Console.Write( "Stack values:" );
      PrintValues( myStack, '\t' );

      // Views the first element in the Stack but does not remove it.
      Console.WriteLine( "(Peek)\t\t{0}", myStack.Peek() );

      // Displays the Stack.
      Console.Write( "Stack values:" );
      PrintValues( myStack, '\t' );
   }


   public static void PrintValues( IEnumerable myCollection, char mySeparator )
{
      foreach ( Object obj in myCollection )
         Console.Write( "{0}{1}", mySeparator, obj );
      Console.WriteLine();
   }

}


/*
This code produces the following output.

Stack values:    fox    brown    quick    The
(Pop)        fox
Stack values:    brown    quick    The
(Pop)        brown
Stack values:    quick    The
(Peek)        quick
Stack values:    quick    The
```

```
*/
```

Remarks

If Count already equals the capacity, the capacity of the Stack is increased by automatically reallocating the internal array, and the existing elements are copied to the new array before the new element is added.

`null` can be pushed onto the Stack as a placeholder, if needed. It occupies a slot in the stack and is treated like any object.

If Count is less than the capacity of the stack, Push is an O(1) operation. If the capacity needs to be increased to accommodate the new element, Push becomes an O(`n`) operation, where `n` is Count.

Applies to

.NET Core

3.0 Preview 2, 2.2, 2.1, 2.0, 1.1, 1.0

.NET Framework

4.8, 4.7.2, 4.7.1, 4.7, 4.6.2, 4.6.1, 4.6, 4.5.2, 4.5.1, 4.5, 4.0, 3.5, 3.0, 2.0, 1.1

.NET Standard

2.0

UWP

10.0

Xamarin.Android

7.1

Xamarin.iOS

10.8

Xamarin.Mac

3.0

See also

- Peek()
- Pop()