F# - MUTABLE LISTS

The **List<'T>** class represents a strongly typed list of objects that can be accessed by index.

It is a mutable counterpart of the List class. It is similar to arrays, as it can be accessed by an index, however, unlike arrays, lists can be resized. Therefore you need not specify a size during declaration.

Creating a Mutable List

Lists are created using the **new** keyword and calling the list's constructor. The following example demonstrates this −

```
(* Creating a List *)
open System.Collections.Generic

let booksList = new List<string>()
booksList.Add("Gone with the Wind")
booksList.Add("Atlas Shrugged")
booksList.Add("Fountainhead")
booksList.Add("Thornbirds")
booksList.Add("Rebecca")
booksList.Add("Narnia")

booksList |> Seq.iteri (fun index item -> printfn "%i: %s" index booksList.[index])
```

When you compile and execute the program, it yields the following output −

```
0: Gone with the Wind
1: Atlas Shrugged
2: Fountainhead
3: Thornbirds
4: Rebecca
5: Narnia
```

The List*T* Class

The List*T* class represents a strongly typed list of objects that can be accessed by index. It provide methods to search, sort, and manipulate lists.

The following tables provide the properties, constructors and the methods of the List*T* class −

Properties

Property	Description
Capacity	Gets or sets the total number of elements the internal data structure can hold without resizing.
Count	Gets the number of elements contained in the List*T*.
Item	Gets or sets the element at the specified index.

Constructors

Constructor	Description
List*T*	Initializes a new instance of the List*T* class that is empty and has

	the default initial capacity.
List*TIEnumerable*(*T*)	Initializes a new instance of the List*T* class that contains elements copied from the specified collection and has sufficient capacity to accommodate the number of elements copied.
List*TInt*32	Initializes a new instance of the List*T* class that is empty and has the specified initial capacity.

Method

Methods	Description
Add	Adds an object to the end of the List*T*.
AddRange	Adds the elements of the specified collection to the end of the List*T*.
AsReadOnly	Returns a read-only IList*T* wrapper for the current collection.
BinarySearch*T*	Searches the entire sorted List*T* for an element using the default comparer and returns the zero-based index of the element.
BinarySearch*T*, *IComparer*(*T*)	Searches the entire sorted List*T* for an element using the specified comparer and returns the zero-based index of the element.
BinarySearch*Int*32, *Int*32, *T*, *IComparer*(*T*)	Searches a range of elements in the sorted List*T* for an element using the specified comparer and returns the zero-based index of the element.
Clear	Removes all elements from the List*T*.
Contains	Determines whether an element is in the List*T*.
ConvertAll*TOutput*	Converts the elements in the current List*T* to another type, and returns a list containing the converted elements.
CopyTo*T*[]	Copies the entire List*T* to a compatible one-dimensional array, starting at the beginning of the target array.
CopyTo*T*[], *Int*32	Copies the entire List*T* to a compatible one-dimensional array, starting at the specified index of the target array.
CopyTo*Int*32, *T*[], *Int*32, *Int*32	Copies a range of elements from the List*T* to a compatible one-dimensional array, starting at the specified index of the target array.
Equals*Object*	Determines whether the specified object is equal to the current object. *InheritedfromObject*.
Exists	Determines whether the List*T* contains elements that match the conditions defined by the specified predicate.
Finalize	Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection *InheritedfromObject*.
Find	Searches for an element that matches the

	conditions defined by the specified predicate, and returns the first occurrence within the entire List*T*.
FindAll	Retrieves all the elements that match the conditions defined by the specified predicate.
FindIndex*Predicate*(*T*)	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the first occurrence within the entire List*T*.
FindIndex*Int*32, *Predicate*(*T*)	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the first occurrence within the range of elements in the List*T* that extends from the specified index to the last element.
FindIndex*Int*32, *Int*32, *Predicate*(*T*)	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the first occurrence within the range of elements in the List*T* that starts at the specified index and contains the specified number of elements.
FindLast	Searches for an element that matches the conditions defined by the specified predicate, and returns the last occurrence within the entire List*T*.
FindLastIndex*Predicate*(*T*)	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the last occurrence within the entire List*T*.
FindLastIndex*Int*32, *Predicate*(*T*)	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the last occurrence within the range of elements in the List*T* that extends from the first element to the specified index.
FindLastIndex*Int*32, *Int*32, *Predicate*(*T*)	Searches for an element that matches the conditions defined by the specified predicate, and returns the zero-based index of the last occurrence within the range of elements in the List*T* that contains the specified number of elements and ends at the specified index.
ForEach	Performs the specified action on each element of the List*T*.
GetEnumerator	Returns an enumerator that iterates through the List*T*.
GetHashCode	Serves as the default hash function. *InheritedfromObject*.
GetRange	Creates a shallow copy of a range of elements in the source List*T*.
GetType	Gets the Type of the current instance. *InheritedfromObject*.
IndexOf*T*	Searches for the specified object and returns the zero-based index of the first occurrence within the entire List*T*.

IndexOf*T*, *Int*32	Searches for the specified object and returns the zero-based index of the first occurrence within the range of elements in the List*T* that extends from the specified index to the last element.
IndexOf*T*, *Int*32, *Int*32	Searches for the specified object and returns the zero-based index of the first occurrence within the range of elements in the List*T* that starts at the specified index and contains the specified number of elements.
Insert	Inserts an element into the List*T* at the specified index.
InsertRange	Inserts the elements of a collection into the List*T* at the specified index.
LastIndexOf*T*	Searches for the specified object and returns the zero-based index of the last occurrence within the entire List*T*.
LastIndexOf*T*, *Int*32	Searches for the specified object and returns the zero-based index of the last occurrence within the range of elements in the List*T* that extends from the first element to the specified index.
LastIndexOf*T*, *Int*32, *Int*32	Searches for the specified object and returns the zero-based index of the last occurrence within the range of elements in the List*T* that contains the specified number of elements and ends at the specified index.
MemberwiseClone	Creates a shallow copy of the current Object. *InheritedfromObject*.
Remove	Removes the first occurrence of a specific object from the List*T*.
RemoveAll	Removes all the elements that match the conditions defined by the specified predicate.
RemoveAt	Removes the element at the specified index of the List*T*.
RemoveRange	Removes a range of elements from the List*T*.
Reverse	Reverses the order of the elements in the entire List *T*.
Reverse*Int*32, *Int*32	Reverses the order of the elements in the specified range.
Sort	Sorts the elements in the entire List*T* using the default comparer.
Sort*Comparison*(*T*)	Sorts the elements in the entire List*T* using the specified System. Comparison*T*.
Sort*IComparer*(*T*)	Sorts the elements in the entire List*T* using the specified comparer.
Sort*Int*32, *Int*32, *IComparer*(*T*)	Sorts the elements in a range of elements in List*T* using the specified comparer.
ToArray	Copies the elements of the List*T* to a new array.
ToString	Returns a string that represents the current object.

	InheritedfromObject.
TrimExcess	Sets the capacity to the actual number of elements in the List*T*, if that number is less than a threshold value.
TrueForAll	Determines whether every element in the List*T* matches the conditions defined by the specified predicate.

Example

```
(* Creating a List *)
open System.Collections.Generic

let booksList = new List<string>()
booksList.Add("Gone with the Wind")
booksList.Add("Atlas Shrugged")
booksList.Add("Fountainhead")
booksList.Add("Thornbirds")
booksList.Add("Rebecca")
booksList.Add("Narnia")

printfn"Total %d books" booksList.Count
booksList |> Seq.iteri (fun index item -> printfn "%i: %s" index booksList.[index])
booksList.Insert(2, "Roots")

printfn("after inserting at index 2")
printfn"Total %d books" booksList.Count

booksList |> Seq.iteri (fun index item -> printfn "%i: %s" index booksList.[index])
booksList.RemoveAt(3)

printfn("after removing from index 3")
printfn"Total %d books" booksList.Count

booksList |> Seq.iteri (fun index item -> printfn "%i: %s" index booksList.[index])
```

When you compile and execute the program, it yields the following output −

```
Total 6 books
0: Gone with the Wind
1: Atlas Shrugged
2: Fountainhead
3: Thornbirds
4: Rebecca
5: Narnia
after inserting at index 2
Total 7 books
0: Gone with the Wind
1: Atlas Shrugged
2: Roots
3: Fountainhead
4: Thornbirds
5: Rebecca
6: Narnia
after removing from index 3
Total 6 books
0: Gone with the Wind
1: Atlas Shrugged
2: Roots
3: Thornbirds
4: Rebecca
5: Narnia
```