**Comparisons in .Net Framework**

2012-03-06 PV

IComparable<T> interface

* Int CompareTo(T other) returns <0/0/>0

Type-specific comparison method for ordering instances. Compares an object with another.

IComparable interface

* Int CompareTo(object other) returns <0/0/>0; other is always of the same type

Old non-generic interface

Comparer<T> class

* public static Comparer<T> Default { get; }

Provides a sort order comparer for type T, returns a Comparer<T> that implements IComparer<T> based on IComparable<T> or IComparable interfaces

IComparer<T> interface

* int Compare(T x, T y)

Compares two objects.

IComparer interface

* int Compare(object x, object y)

Comparison<T> delegate

* public delegate int Comparison<in T>(T x, T y)

Represents the method that compares two objects of the same type.

This delegate is used by the Sort<T>(T[], Comparison<T>) method overload of the Array class and the Sort(Comparison<T>) method overload of the List<T>class to sort the elements of an array or list.

Example of use of this delegate:

list.Sort((x,y) => string.Compare(x.Name, y.Name));

' Conversion Comparer --> Comparison

Dim co As Comparison(Of Integer)

co = AddressOf Comparer(Of Integer).Default.Compare

Summary:

* Base type T implements IComparable<T> or IComparable
* Class Comparer<T> has a static Default methods that returns an instance implementing IComparer<T> interface
* Comparison<T> delegate represents the type of int Compare(T x, T y) of IComparer<T>

Note: When implementing IComparable, you must also override Object.Equals and overload the language-specific operators for equality, inequality, less than, or greater than (<http://msdn.microsoft.com/en-us/library/ms182163.aspx>).

Implement IComparer instead of IComparable if your objects are mutable : Since the IComparer is just a disposable object used for sorting at that point in time, your object can have any mutable semantics you desire. Furthermore, it doesn't require or even suggest using Equals, GetHashCode, or == - you're free to define it in any way you please.

StringComparer class is recommended over Comparer<string>, various properties of StringComparer class return predefined instances that perform string comparisons with different combinations of culture-sensitivity and case-sensitivity, namely:

- CurrentCulture Gets a StringComparer object that performs a case-sensitive string comparison using the word comparison rules of the current culture.

- CurrentCultureIgnoreCase Gets a StringComparer object that performs case-insensitive string comparisons using the word comparison rules of the current culture.

- InvariantCulture Gets a StringComparer object that performs a case-sensitive string comparison using the word comparison rules of the invariant culture.

- InvariantCultureIgnoreCase Gets a StringComparer object that performs a case-insensitive string comparison using the word comparison rules of the invariant culture.

- Ordinal Gets a StringComparer object that performs a case-sensitive ordinal string comparison.

- OrdinalIgnoreCase Gets a StringComparer object that performs a case-insensitive ordinal string comparison.