Hashtable and Dictionary Collection Types

Article • 09/15/2021 • 2 minutes to read

The System.Collections.Hashtable class, and the System.Collections.Generic.Dictionary<TKey,TValue> and System.Collections.Concurrent.ConcurrentDictionary<TKey,TValue> generic classes, implement the System.Collections.IDictionary interface. The Dictionary<TKey,TValue> generic class also implements the IDictionary<TKey,TValue> generic interface. Therefore, each element in these collections is a key-and-value pair.

A Hashtable object consists of buckets that contain the elements of the collection. A bucket is a virtual subgroup of elements within the Hashtable, which makes searching and retrieving easier and faster than in most collections. Each bucket is associated with a hash code, which is generated using a hash function and is based on the key of the element.

The generic HashSet<T> class is an unordered collection for containing unique elements.

A hash function is an algorithm that returns a numeric hash code based on a key. The key is the value of some property of the object being stored. A hash function must always return the same hash code for the same key. It is possible for a hash function to generate the same hash code for two different keys, but a hash function that generates a unique hash code for each unique key results in better performance when retrieving elements from the hash table.

Each object that is used as an element in a Hashtable must be able to generate a hash code for itself by using an implementation of the GetHashCode method. However, you can also specify a hash function for all elements in a Hashtable by using a Hashtable constructor that accepts an IHashCodeProvider implementation as one of its parameters.

When an object is added to a Hashtable, it is stored in the bucket that is associated with the hash code that matches the object's hash code. When a value is being searched for in the Hashtable, the hash code is generated for that value, and the bucket associated with that hash code is searched.

For example, a hash function for a string might take the ASCII codes of each character in the string and add them together to generate a hash code. The string "picnic" would have a hash code that is different from the hash code for the string "basket"; therefore, the strings "picnic" and "basket" would be in different buckets. In contrast, "stressed" and "desserts" would have the same hash code and would be in the same bucket.

The Dictionary<TKey,TValue> and ConcurrentDictionary<TKey,TValue> classes have the same functionality as the Hashtable class. A Dictionary<TKey,TValue> of a specific type (other than Object) provides better performance than a Hashtable for value types. This is because the elements of Hashtable are of type Object; therefore, boxing and unboxing typically occur when you store or retrieve a value type. The ConcurrentDictionary<TKey,TValue> class should be used when multiple threads might be accessing the collection simultaneously.

See also

- Hashtable
- IDictionary
- IHashCodeProvider
- Dictionary<TKey,TValue>
- System.Collections.Generic.IDictionary<TKey,TValue>
- System.Collections.Concurrent.ConcurrentDictionary<TKey,TValue>
- Commonly Used Collection Types