

C# - HASHTABLE CLASS

https://www.tutorialspoint.com/csharp/csharp_hashtable.htm

Copyright © tutorialspoint.com

Advertisements

The Hashtable class represents a collection of **key-and-value pairs** that are organized based on the hash code of the key. It uses the key to access the elements in the collection.

A hash table is used when you need to access elements by using **key**, and you can identify a useful key value. Each item in the hash table has a key/value pair. The key is used to access the items in the collection.

Methods and Properties of the Hashtable Class

The following table lists some of the commonly used **properties** of the **Hashtable** class –

Sr.No.	Property & Description
1	Count Gets the number of key-and-value pairs contained in the Hashtable.
2	IsFixedSize Gets a value indicating whether the Hashtable has a fixed size.
3	IsReadOnly Gets a value indicating whether the Hashtable is read-only.
4	Item Gets or sets the value associated with the specified key.
5	Keys Gets an ICollection containing the keys in the Hashtable.
6	Values Gets an ICollection containing the values in the Hashtable.

The following table lists some of the commonly used **methods** of the **Hashtable** class –

Sr.No.	Method & Description
1	public virtual void Add <i>objectkey, objectvalue</i> ; Adds an element with the specified key and value into the Hashtable.
2	public virtual void Clear ; Removes all elements from the Hashtable.
3	public virtual bool ContainsKey <i>objectkey</i> ; Determines whether the Hashtable contains a specific key.
4	public virtual bool ContainsValue <i>objectvalue</i> ; Determines whether the Hashtable contains a specific value.
5	public virtual void Remove <i>objectkey</i> ; Removes the element with the specified key from the Hashtable.

Example

The following example demonstrates the concept –

[Live Demo](#)

```
using System;
using System.Collections;

namespace CollectionsApplication {
    class Program {
        static void Main(string[] args) {
            Hashtable ht = new Hashtable();

            ht.Add("001", "Zara Ali");
            ht.Add("002", "Abida Rehman");
            ht.Add("003", "Joe Holzner");
            ht.Add("004", "Mausam Benazir Nur");
            ht.Add("005", "M. Amlan");
            ht.Add("006", "M. Arif");
            ht.Add("007", "Ritesh Saikia");
        }
    }
}
```

```
if (ht.ContainsValue("Nuha Ali")) {  
    Console.WriteLine("This student name is already in the list");  
} else {  
    ht.Add("008", "Nuha Ali");  
}  
  
// Get a collection of the keys.  
ICollection key = ht.Keys;  
  
foreach (string k in key) {  
    Console.WriteLine(k + ": " + ht[k]);  
}  
Console.ReadKey();  
}  
}  
}
```

When the above code is compiled and executed, it produces the following result –

```
001: Zara Ali  
002: Abida Rehman  
003: Joe Holzner  
004: Mausam Benazir Nur  
005: M. Amlan  
006: M. Arif  
007: Ritesh Saikia  
008: Nuha Ali
```