Programming Dictionary in C#

Mahesh Chand

July 2012, Garnet Valley PA

Introduction

A dictionary type represents a collection of keys and values pair of data.

The Dictionary class defined in the System.Collections.Generic namespace is a generic class and can store any data types in a form of keys and values. Each key must be unique in the collection. Before you use the Dictionary class in your code, you must import the System.Collections.Generic namespace using the following line.

```
using System.Collections.Generic;
```

Creating a Dictionary

The Dictionary class constructor takes a key data type and a value data type. Both types are generic so it can be any .NET data type.

The following The Dictionary class is a generic class and can store any data types. This class is defined in the code snippet creates a dictionary where both keys and values are string types.

```
Dictionary<string, string> EmployeeList = new Dictionary<string, string>();
```

The following code snippet adds items to the dictionary.

```
EmployeeList.Add("Mahesh Chand", "Programmer");
EmployeeList.Add("Praveen Kumar", "Project Manager");
EmployeeList.Add("Raj Kumar", "Architect");
EmployeeList.Add("Nipun Tomar", "Asst. Project Manager");
EmployeeList.Add("Dinesh Beniwal", "Manager");
```

The following code snippet creates a dictionary where the key type is string and value type is short integer.

```
Dictionary<string, Int16> AuthorList = new Dictionary<string, Int16>();
```

The following code snippet adds items to the dictionary.

```
AuthorList.Add("Mahesh Chand", 35);
AuthorList.Add("Mike Gold", 25);
AuthorList.Add("Praveen Kumar", 29);
AuthorList.Add("Raj Beniwal", 21);
AuthorList.Add("Dinesh Beniwal", 84);
```

We can also limit the size of a dictionary. The following code snippet creates a dictionary where the key type is string and value type is float and total number of items it can hold is 3.

```
Dictionary<string, float> PriceList = new Dictionary<string, float>(3);
```

The following code snippet adds items to the dictionary.

```
PriceList.Add("Tea", 3.25f);
PriceList.Add("Juice", 2.76f);
PriceList.Add("Milk", 1.15f);
```

Reading Dictionary Items

The Dictionary is a collection. We can use the foreach loop to go through all the items and read them using they Key ad Value properties.

```
foreach (KeyValuePair<string, Int16> author in AuthorList)
{
 Console.WriteLine("Key: {0}, Value: {1}",
 author.Key, author.Value);
}
```

The following code snippet creates a new dictionary and reads all of its items and displays on the console.

```
public void CreateDictionary()
{
    // Create a dictionary with string key and Int16 value pair
    Dictionary<string, Int16> AuthorList = new Dictionary<string, Int16>();
    AuthorList.Add("Mahesh Chand", 35);
    AuthorList.Add("Mike Gold", 25);
    AuthorList.Add("Praveen Kumar", 29);
    AuthorList.Add("Raj Beniwal", 21);
    AuthorList.Add("Dinesh Beniwal", 84);

    // Read all data
    Console.WriteLine("Authors List");

    foreach (KeyValuePair<string, Int16> author in AuthorList)
    {
        Console.WriteLine("Key: {0}, Value: {1}",
            author.Key, author.Value);
    }
}
```

Dictionary Properties

The Dictionary class has three properties – Count, Keys and Values.

Count

The Count property gets the number of key/value pairs in a Dictionary.

The following code snippet display number of items in a dictionary.

```
Console.WriteLine("Count: {0}", AuthorList.Count);
```

Item

The Item property gets and sets the value associated with the specified key.

The following code snippet sets and gets an items value.

```
// Set Item value
AuthorList["Mahesh Chand"] = 20;
// Get Item value
Int16 age = Convert.ToInt16(AuthorList["Mahesh Chand"]);
```

Keys

The Keys property gets a collection containing the keys in the Dictionary. It returns an object of KeyCollection type.

The following code snippet reads all keys in a Dictionary.

```
// Get and display keys
Dictionary<string, Int16>.KeyCollection keys = AuthorList.Keys;
foreach (string key in keys)
{
    Console.WriteLine("Key: {0}", key);
}
```

Values

The Values property gets a collection containing the values in the Dictionary. It returns an object of ValueCollection type.

The following code snippet reads all values in a Dictionary.

```
// Get and display values
Dictionary<string, Int16>.ValueCollection values = AuthorList.Values;
foreach (Int16 val in values)
{
    Console.WriteLine("Value: {0}", val);
}
```

Dictionary Methods

The Dictionary class is a generic collection and provides all common methods to add, remove, find and replace items in the collection.

Add Items

The Add method adds an item to the Dictionary collection in form of a key and a value.

The following code snippet creates a Dictionary and adds an item to it by using the Add method.

```
Dictionary<string, Int16> AuthorList = new Dictionary<string, Int16>();
AuthorList.Add("Mahesh Chand", 35);
```

Alternatively, we can use the Item property. If the key does not exist in the collection, a new item is added. If the same key already exists in the collection, the item value is updated to the new value.

The following code snippet adds an item and updates the existing item in the collection.

```
AuthorList["Neel Beniwal"] = 9;
AuthorList["Mahesh Chand"] = 20;
```

Remove Item

The Remove method removes an item with the specified key from the collection. The following code snippet removes an item.

```
// Remove item with key = 'Mahesh Chand'
AuthorList.Remove("Mahesh Chand");
```

The Clear method removes all items from the collection. The following code snippet removes all items by calling the Clear method.

```
// Remove all items
AuthorList.Clear();
```

Find a Key

The ContainsKey method checks if a key is already exists in the dictionary. The following code snippet checks if a key is already exits and if not, add one.

```
if (!AuthorList.ContainsKey("Mahesh Chand"))
{
    AuthorList["Mahesh Chand"] = 20;
}
```

Find a Value

The ContainsValue method checks if a value is already exists in the dictionary. The following code snippet checks if a value is already exits.

```
if (!AuthorList.ContainsValue(9))
{
    Console.WriteLine("Item found");
}
```

Sample

Here is the complete sample code showing how to use these methods.

```
// Create a dictionary with string key and Int16 value pair
Dictionary<string, Int16> AuthorList = new Dictionary<string, Int16>();
AuthorList.Add("Mahesh Chand", 35);
AuthorList.Add("Mike Gold", 25);
AuthorList.Add("Praveen Kumar", 29);
AuthorList.Add("Raj Beniwal", 21);
AuthorList.Add("Dinesh Beniwal", 84);

// Count
Console.WriteLine("Count: {0}", AuthorList.Count);

// Set Item value
AuthorList["Neel Beniwal"] = 9;

if (!AuthorList.ContainsKey("Mahesh Chand"))
{
    AuthorList["Mahesh Chand"] = 20;
}
if (!AuthorList.ContainsValue(9))
{
    Console.WriteLine("Item found");
}

// Read all items
Console.WriteLine("Authors all items:");

foreach (KeyValuePair<string, Int16> author in AuthorList)
{
    Console.WriteLine("Key: {0}, Value: {1}",
        author.Key, author.Value);
}

// Remove item with key = 'Mahesh Chand'
AuthorList.Remove("Mahesh Chand");


// Remove all items
AuthorList.Clear();
```