

Forum:[Hack Forums](#)

Thread: [C# Hashtables \[Intermediate Level Tutorial\]](#)

Pulser - October 22, 2010, 1:11 pm

C# Hashtables

What Is It:

A Hashtable, unlike an array, will allow you to store data based on keys and values. Each key will have a value. You can use this to store Names, Numbers, anything in your life that would require organization and order! A Hashtable is known as an associative array.

Getting Started:

The very first thing we want to do when working with a Hashtable..is to set it up!

```
Code Hashtable hf = new Hashtable();
```

By reading this code you should have a fairly basic understanding of what it does. It creates a new Hashtable object that is named "hf"

The next thing we want to do is start setting up our keys/values.

```
Code hf["Predictable"] = "is cool";
```

```
hf["Hackforums"] = "is cooler";
```

hf is obviously the name of our Hashtable. "Predictable" is one of the keys in our Hashtable, it is obviously named Predictable!

After the equals sign is our value, the value is the stuff that our key is holding.

Now, we want to display our key/value pairs in a MessageBox, Right?

NO! MessageBox's are getting boring. So, create a new listbox. A listbox is just a component that will display a list of something. To display our keys and values in our new listbox, we do the following:

```
Code foreach (DictionaryEntry h in hf)
```

```
{  
    listBox1.Items.Add(h.Key + " " + h.Value);
```

```
}
```

This is called a foreach loop. We use the word "DictionaryEntry" is what we use when using a foreach loop with a Hashtable, "h" is just the name of our new variable.

We use "listbox1.Items.Add" to add something to the listbox. "h.Key" and "h.Value" simply adds the key/value pairs to our listbox!

Conclusion:

Hashtables can be used for anything, whether it is storing people in your contacts, matching MD5 hashes with plaintext, or simply screwing around.

[IMG]http://i52.[censored].com/302xd1.jpg[/IMG]

Code:

```
Code private void button1_Click(object sender, EventArgs e)
{
    Hashtable hf = new Hashtable();

    hf["Predictable"] = "is cool";
    hf["Hackforums"] = "is cooler";

    foreach (DictionaryEntry h in hf)
    {
        listBox1.Items.Add(h.Key + " " + h.Value);
    }
}
```

BaussHacker - October 22, 2010, 1:19 pm

I wouldn't start use hashtables like this, but thats ur way in coding style.

Maybe u should start explain about certain arrays and other collection types.

You could have done this much easier, but good guide anyway :)

Pulser - October 22, 2010, 2:39 pm

(Oct 22, 2010 01:19 PM)_CookieCoder Wrote: [\[CITE\]](#) I wouldn't start use hashtables like this, but thats ur way in coding style.

Maybe u should start explain about certain arrays and other collection types.

You could have done this much easier, but good guide anyway :)

Thanks a lot for the constructive criticism. I will write a better, more effective tut in the future.

Zer0Fact0r - October 22, 2010, 3:52 pm

Explain Hashtable's run-time efficiency compared to arrays and such.

BaussHacker - October 22, 2010, 4:13 pm

all types of arrays should be declared after use, else they can send alot random null reference exceptions.

hashtables can keep the data for longer time + they can keep more.

Well I dont wanna go deeper in it, if u dont want me to, then I will explain it total out, because this wont explain much, if u dont know about it already.

Zer0Fact0r - October 22, 2010, 4:38 pm

(Oct 22, 2010 04:13 PM)_CookieCoder Wrote: [\[CITE\]](#) all types of arrays should be declared after use, else they can send alot random null reference exceptions.

hashtables can keep the data for longer time + they can keep more.

Well I dont wanna go deeper in it, if u dont want me to, then I will explain it total out, because this wont explain much, if u dont know about it already.

No. Just no...

??????????????

BaussHacker - *October 22, 2010, 4:44 pm*

Example on arraylist.

Code static void Main(string[] args)

```
{
    //Use ararylists temp only, else u will get random errors (null exception eg.)
    //Make sure u copy to a real array, because arraylists are to hold less data
    ArrayList list = new ArrayList();
    Random random = new Random();
    for (int i = 0; i < random.Next(0, 1000); i++)
    {
        list.Add(i);
    }
    int[] buffer = new int[list.Count];
    list.CopyTo(buffer);
    foreach (int i2 in buffer)
    {
        Console.WriteLine(i2);
    }
}
```

more u want?