

Back to Basics – How to iterate through an enumeration (Enum) in C# ?

```
enum WorkingDays
{
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday
}

enum WorkingDays
{
    Monday,
    Tuesday,
    Wednesday,
    Thursday,
    Friday
}
```

WorkingDays.Monday = 0

WorkingDays.Friday = 4

The *enum* type is one of the common features that we used during almost every implementation. While the declarations and usages look very straightforward, there are many things we should be very clear about, and this topic is one of them. ***How to iterate through an Enum in C#?*** or ***can you loop through with all the enum values in C#?*** – yet another frequently asked question ^[1] in an interview for the beginners. Well, there are several ways to achieve this. To answer this in a sort way, the *Enum* class provides the base class for enumerations and *Enum.GetNames()* method, which is used to retrieve an array of the names and returns a string array of the names.

Let's try to understand using simple example. Consider you have following Enumeration

?

```
enum WorkingDays
{
    Monday,
    Tuesday,
```

Wednesday,

Thursday,

Friday

}

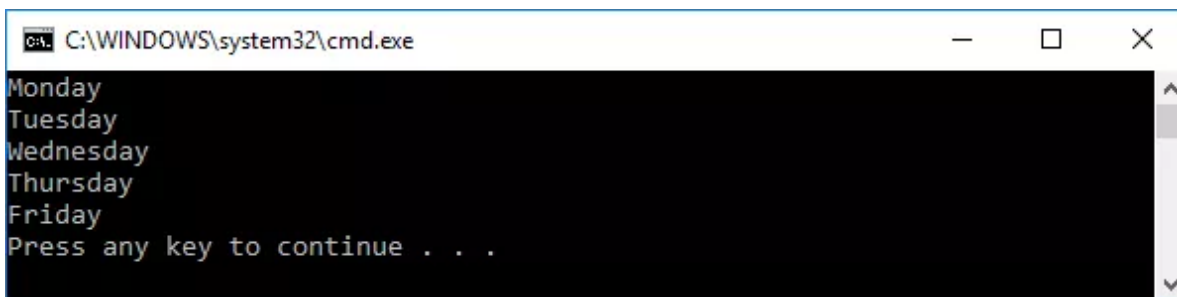
Check out for more similar *.NET Frequently Asked Questions*^[2]

Now using the *Enum.GetNames()* you can iterate through the enumeration as following. *Enum.GetNames()* required the types of *enum* as parameters, which you can pass using *typeof* keyword. That will retrieve an array of the names of the constants in the *WorkingDays* enumeration.

?

```
foreach (var item in Enum.GetNames(typeof(WorkingDays)))  
  
{  
  
    Console.WriteLine(item);  
  
}
```

Once you run this, you will have following output



```
C:\WINDOWS\system32\cmd.exe  
Monday  
Tuesday  
Wednesday  
Thursday  
Friday  
Press any key to continue . . .
```

In another approach, you can also use *Enum.GetValues()* and iterate through the items.

?

```
foreach (var item in Enum.GetValues(typeof(WorkingDays)))
```

```
{

Console.WriteLine(item);

}
```

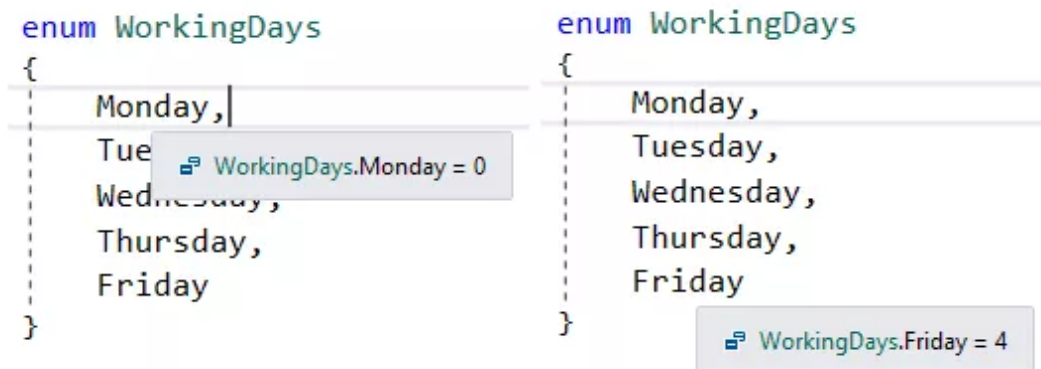
and this will also produce the same output.

```
C:\WINDOWS\system32\cmd.exe
Monday
Tuesday
Wednesday
Thursday
Friday
Press any key to continue . . .
```

Both the approaches works, but now the question would be, *what is the difference between **Enum.GetNames()** and **Enums.GetValues()** ?*

GetNames() will return a string array of the **Names** for the items in the enum where as the *GetValues()* will return an array of the **values** for each item in the Enum.

When we define the enum, all the Enum elements will have an assigned a default values starting from 0. For an example, *Monday will have an assigned values 0*, and similarly, *Friday will have values of 4*.



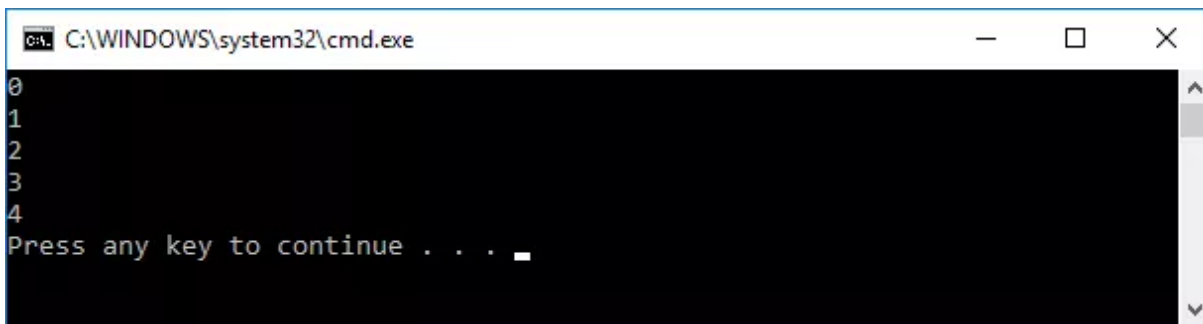
You can assign the custom values as well.

Now, think of this enumerations as *Name & Value* pairs. *Monday = 0, Tuesday = 1, Wednesday = 2, Thursday = 3, Friday = 4*

So with , *GetNames()* will return a string array containing the items “Monday”, “Tuesday” “Friday”.. on the other hand, When used *GetValues()*, Actually, *GetValues()* will return an int

array containing 0,1,2,3,4 . You can easily type cast the values.

```
[  
  
?  
  
foreach (WorkingDays item in Enum.GetValues(typeof(WorkingDays)))  
  
{  
  
Console.WriteLine(Convert.ToInt32(item));  
  
}
```



A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window has a black background with white text. It displays the numbers 0, 1, 2, 3, and 4 on separate lines, followed by the prompt "Press any key to continue . . .".

Like this:

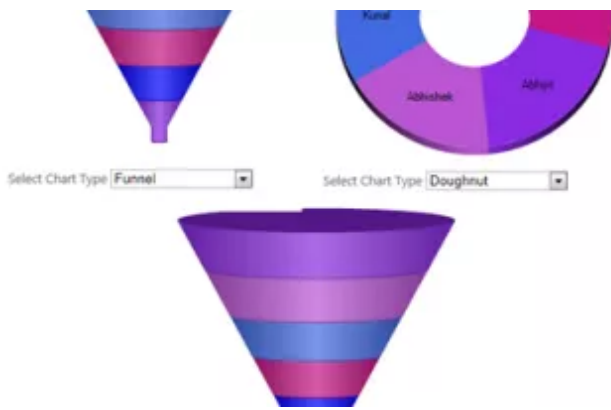
Related



[3]

.NET & C# - Back to Basics Series^[4]

In ".NET"



[5]

Dynamically Change Chart Control Type in ASP.NET 4.0^[6]

In ".NET"

Links

1. <https://dailydotnettips.com/tag/faq/>
2. <https://dailydotnettips.com/tag/faq/>
3. <https://dailydotnettips.com/net-c-back-to-basics-series/>
4. <https://dailydotnettips.com/net-c-back-to-basics-series/>
5. <https://dailydotnettips.com/dynamically-change-chart-control-type-in-asp-net-4-0/>
6. <https://dailydotnettips.com/dynamically-change-chart-control-type-in-asp-net-4-0/>