**Enums - Enumerations**

בחלק זה נלמד כיצד ליצור טיפוסים חדשים מסוג  **enum**

### מהו טיפוס מסוג enum ?

טיפוס מסוג **enum** הוא טיפוס שיכול לקבל ערכים רק מתוך קבוצה קטנה של ערכים.   
לדוגמא, ניתן להגדיר טיפוס מסוג **enum** בשם Month שיוכל לקבל רק אחד מהערכים הבאים:

### כיצד עובדים עם enum ?

ראשית יש להגדיר את הטיפוס החדש, משתמשים במילה **enum** בשביל לציין שזהו טיפוס מסוג **enum**, לאחר מכן נותנים את שם הטיפוס ולאחריו את רשימת הערכים מופרדת בפסיקים.

לדוגמא, הקוד הבא מגדיר **enum** בשם Month המכיל את ערכי החודשים השונים:

enum Month  
{  
    January,   
    February,   
    March,   
    April,   
    May,   
    June,   
    July,   
    August,   
    September,   
    October,   
    November,   
    December  
}

נציין שהגדרת הטיפוס צריכה להיות מחוץ לפונקציית ה **Main** ואפילו מחוץ למחלקה **Program** שכן אנחנו מגדירים טיפוס חדש.

בפרויקטים בדרך כלל מיצור עבור ה- enum קובץ נפרד או שנמקם אותו בקובץ המכיל מספר טיפוסים לשימוש כללי.

כעת נוכל להשתמש בטיפוס החדש בתוכנית הראשית:

static void Main(string[] args)  
{  
    Month myMonth = Month.December;  
    Console.WriteLine(myMonth);  
}

### מתי נשתמש ב-enum ?

השימוש בטיפוס מסוג **enum** עוזר לקריאות של התוכנית ומונע שגיאות.

לדוגמא, התבוננו בקטע הקוד הבא:

if ((documentState == 1) && (userSelection == 2))  
{  
    SaveDocument();  
}  
else if ((documentState == 1) && (userSelection == 1))  
{  
    DiscardDocument();  
}

לא ברור מה הייתה כוונת המשורר, מה מייצגים המספרים 1,2 והאם יש להם משמעות יחידה..   
לעומת זאת, אותו הקטע תוך שימוש ב**enum** יראה בצורה הרבה יותר ברורה:

if ((documentState == DocumentState.NotSaved) &&   
    (userSelection == DocumentOperation.Save))  
{  
    SaveDocument();  
}  
else if ((documentState == DocumentState.NotSaved) &&   
         (userSelection == DocumentOperation.Discard))  
{  
    DiscardDocument();  
}

כאשר במקרה זה נגדיר שני טיפוסים מסוג **enum** באופן הבא:

enum DocumentOperation  
{   
    Load,  
    Save,  
    Discard  
}  
   
enum DocumentState  
{   
    Saved,  
    NotSaved  
}

בהגדרת ה- enum אנו למעשה מגדירים סט של קבועים. השימוש הבסיסי דומה לשימוש ב-C/C++:

דוגמא נוספת.

using System;  
  
public class EnumTest   
{  
      enum Days {Sun = 1, Mon, Tue, Wed, Thu, Fri, Sat};  
  
      public static int Main()   
      {  
            int x = (int) Days.Sun;  
            int y = (int) Days.Fri;  
            Console.WriteLine("Sun = {0}", x);  
            Console.WriteLine("Fri = {0}", y);  
  
            return 0;  
      }  
}

נשים לב רק לעובדה כי ב-C# הקבועים משויכים ל-enum, כלומר עלינו לכתוב Days.Sun על מנת להשתמש באחד מאברי ה-enum, ואיננו יכולים להסתפק רק בכתיבה של הביטוי Sun למטרה זו.

כמו כן, נביט בדוגמא הבאה:

using System;  
  
class MainClass   
{  
      enum colors { BLACK, WHITE, RED, BLUE };  
  
      public static int Main()    
      {  
            colors i = colors.WHITE;  
            Console.WriteLine("{0}", i);  
            return 0;  
      }  
}

על המסך תודפס המילה WHITE, ולא הערך המספרי 1.

כברירת מחדל, enum מוחזק בזיכרון כמשתנה מטיפוס של int. ערכו של הקבוע הראשון הוא אפס וערכו של כל קבוע אחריו גדל באחד. ניתן לשנות גם את סוג המשתנה וגם את ערכי הקבועים:

enum grades : short { FAIL = 40, PASSED = 55, EXCELLENT = 100 };

סוגי הטיפוס המותרים ל-enum הינם: byte, sbyte, short, ushort, int, uint, long, ulong.

מתחת לפני השטח נוצרת מחלקה היורשת את System.Enum ומכילה מספר שדות קבועים, ההמרה למחרוזת נעשית באמצעות שם כל שדה וההמרה מערך בחזרה ל-Enum מתבצעת באמצעות מילון (Hash Table) אשר מתחולל במחלקה.

אפשרות להצבה של שני ערכים ויותר בערך רשימה אחד:

[Flags]

public enum WriteAccess : short

{

None = 0,

Read = 1,

Write = 2,

ReadWrite = 3

}

וכך נוכל ליצור מופע של הרשימה ולהציב לתוכו שני ערכים:

WriteAccess wa= WriteAccess. Read | WriteAccess. Write;

[Flags]

public enum Flavors

{

None = 0,

Cherry = 1,

Grape = 2,

Orange = 4,

Peach = 8

}

המרות של enums :

האם ניתן להמיר ערך רשימה באופן אוטומטי למחרוזת ?

enum Rank { A, B, C }

Rank myRank = Rank.A;

string myString = Rank.A; // Error: Cannot implicitly convert type 'Rank' to 'string'

string myString2 = Rank.A.ToString(); // OK: but is extra work

Console.Writeline(Rank.A);

Also, just to be clear, there's nothing special about Console.WriteLine, it just has an overload that takes an Object and the enum is getting boxed, passed as an object and then WriteLine calls ToString on it. –

ועוד צורות להמרה:

From number to ENUM:

MyEnum e = (MyEnum)3;

You can check if it's in range using Enum.IsDefined:

if (Enum.IsDefined(typeof(MyEnum), 3)) { ... }

From a string:

YourEnum foo = (YourEnum) Enum.Parse(typeof(YourEnum), yourString);

From an int:

YourEnum foo = (YourEnum)yourInt;

Update : From number you can also:

YourEnum foo = Enum.ToObject(typeof(YourEnum) , yourInt);