

Types d'énumération (référence C#)

Article • 07/04/2023

Un *type d'énumération* (ou *type enum*) est un [type valeur](#) défini par un ensemble de constantes nommées du type [numérique intégral](#) sous-jacent. Pour définir un type d'énumération, utilisez le `enum` mot clé et spécifiez les noms des *membres enum* :

C#

```
enum Season
{
    Spring,
    Summer,
    Autumn,
    Winter
}
```

Par défaut, les valeurs constantes associées des membres enum sont de type `int` ; elles commencent par zéro et augmentent de un en suivant l'ordre textuel de la définition. Vous pouvez spécifier explicitement n'importe quel autre type [numérique intégral](#) comme type sous-jacent d'un type d'énumération. Vous pouvez également spécifier explicitement les valeurs constantes associées, comme le montre l'exemple suivant :

C#

```
enum ErrorCode : ushort
{
    None = 0,
    Unknown = 1,
    ConnectionLost = 100,
    OutlierReading = 200
}
```

Vous ne pouvez pas définir une méthode à l'intérieur de la définition d'un type d'énumération. Pour ajouter des fonctionnalités à un type d'énumération, créez une [méthode d'extension](#).

La valeur par défaut d'un type d'énumération `E` est la valeur produite par l'expression `(E)0`, même si zéro n'a pas le membre enum correspondant.

Vous utilisez un type d'énumération pour représenter un choix à partir d'un ensemble de valeurs mutuellement exclusives ou d'une combinaison de choix. Pour représenter une combinaison de choix, définissez un type d'énumération en tant qu'indicateurs binaires.

Types énumération comme indicateurs binaires

Si vous souhaitez qu'un type d'énumération représente une combinaison de choix, définissez des membres enum pour ces choix, de sorte qu'un choix individuel soit un champ de bits. Autrement dit, les valeurs associées de ces membres enum doivent être les puissances de deux. Ensuite, vous pouvez utiliser les [opérateurs logiques binaires](#) | ou & pour combiner des choix ou croiser des combinaisons de choix, respectivement. Pour indiquer qu'un type d'énumération déclare des champs binaires, appliquez-lui l'attribut [Indicateurs](#) . Comme le montre l'exemple suivant, vous pouvez également inclure certaines combinaisons classiques dans la définition d'un type d'énumération.

C#

```
[Flags]
public enum Days
{
    None      = 0b_0000_0000,  // 0
    Monday    = 0b_0000_0001,  // 1
    Tuesday   = 0b_0000_0010,  // 2
    Wednesday = 0b_0000_0100,  // 4
    Thursday  = 0b_0000_1000,  // 8
    Friday    = 0b_0001_0000,  // 16
    Saturday  = 0b_0010_0000,  // 32
    Sunday    = 0b_0100_0000,  // 64
    Weekend   = Saturday | Sunday
}

public class FlagsEnumExample
{
    public static void Main()
    {
        Days meetingDays = Days.Monday | Days.Wednesday | Days.Friday;
        Console.WriteLine(meetingDays);
        // Output:
        // Monday, Wednesday, Friday

        Days workingFromHomeDays = Days.Thursday | Days.Friday;
        Console.WriteLine($"Join a meeting by phone on {meetingDays &
workingFromHomeDays}");
        // Output:
        // Join a meeting by phone on Friday

        bool isMeetingOnTuesday = (meetingDays & Days.Tuesday) ==
Days.Tuesday;
        Console.WriteLine($"Is there a meeting on Tuesday: {isMeetingOn-
Tuesday}");
        // Output:
        // Is there a meeting on Tuesday: False

        var a = (Days)37;
        Console.WriteLine(a);
```

```
// Output:  
// Monday, Wednesday, Saturday  
}  
}
```

Pour plus d'informations et d'exemples, consultez la [System.FlagsAttribute](#) page de référence d'API et les [membres non exclusifs](#) et la [section Attributs indicateurs](#) de la page de référence d'API [System.Enum](#) .

Le système. Type enum et contrainte d'énumération

Le type [System.Enum](#) est la classe de base abstraite de tous les types d'énumération. Il fournit plusieurs méthodes pour obtenir des informations sur un type d'énumération et ses valeurs. Pour plus d'informations et des exemples, consultez la page de référence API [System.Enum](#).

Vous pouvez utiliser `System.Enum` dans une contrainte de classe de base (appelée [contrainte d'énumération](#)) pour spécifier qu'un paramètre de type est un type d'énumération. Tout type d'énumération respecte également la contrainte `struct`, qui est utilisée pour spécifier qu'un paramètre de type est un type valeur non-nullable.

Conversions

Pour tout type d'énumération, il existe des conversions explicites entre le type d'énumération et son type intégral sous-jacent. Si vous [castez](#) une valeur enum vers son type sous-jacent, le résultat est la valeur intégrale associée d'un membre enum.

C#

```
public enum Season  
{  
    Spring,  
    Summer,  
    Autumn,  
    Winter  
}  
  
public class EnumConversionExample  
{  
    public static void Main()  
    {  
        Season a = Season.Autumn;  
        Console.WriteLine($"Integral value of {a} is {(int)a}"); // out-  
put: Integral value of Autumn is 2  
    }  
}
```

```
var b = (Season)1;
Console.WriteLine(b); // output: Summer

var c = (Season)4;
Console.WriteLine(c); // output: 4
    }
}
```

Utilisez la méthode [Enum.IsDefined](#) pour déterminer si un type d'énumération contient un membre enum avec la valeur associée certaine.

Pour tout type d'énumération, il existe respectivement, des conversions [de boxing et d'unboxing](#) vers et à partir du type [System.Enum](#).

spécification du langage C#

Pour plus d'informations, consultez les sections suivantes de la [spécification du langage C#](#) :

- [Énumérations](#)
- [Valeurs et opérations des énumérations](#)
- [Opérateurs logiques d'énumération](#)
- [Opérateurs de comparaison d'énumération](#)
- [Conversions d'énumération explicites](#)
- [Conversions d'énumération implicites](#)

Voir aussi

- [Informations de référence sur C#](#)
- [Chaînes de format d'énumération](#)
- [Lignes directrices de conception - Conception d'énumération](#)
- [Lignes directrices de conception - Conventions d'affectation de noms enum](#)
- [switch, expression](#)
- [instruction switch](#)

 Collaborer avec nous sur
GitHub


La source de ce contenu se
trouve sur GitHub, où vous
pouvez également créer et



Commentaires sur .NET

.NET est un projet open source.
Sélectionnez un lien pour fournir des
commentaires :

examiner les problèmes et les demandes de tirage. Pour plus d'informations, consultez notre [guide du contributeur](#).

 [Ouvrir un problème de documentation](#)

 [Indiquer des commentaires sur le produit](#)