

18 Enums (Aufzählungen)

534	Bestimmung der Werte der Aufzählungselemente
535	Bestimmung der Typen der Aufzählungselemente
536	Konvertieren von Enums
537	Flags-Enum (Flags-Aufzählungen)

Enums (etwa: *Aufzählungen*, nicht zu verwechseln mit *Auflistungen*, den *Collections*) dienen in erster Linie dazu, Programmierern das Leben zu erleichtern. Programmierer müssen sich gerade in den Zeiten von .NET schon eine ganze Menge merken und sind für jede Unterstützung in dieser Richtung dankbar.

Mit Enums rufen Sie konstante Werte, die thematisch zu einer Gruppe gehören, per Namen ab. Gleichzeitig haben Sie die Möglichkeit, die Werte auf diese Namen zu beschränken. Angenommen, Sie haben eine kleine Kontaktverwaltung auf die Beine gestellt, und diese Datenbankanwendung erlaubt es Ihnen, Ihre Kontakte zu kategorisieren. Sie können also einstellen, ob ein Kontakt ein Kunde, ein Bekannter, ein Geschäftskollege ist oder sonst einer Gruppierung angehört. Für jede dieser Kategorien haben Sie eine bestimmte Nummer vergeben. Und in Abhängigkeit bestimmter Kategorien (Nummern) können Sie nun spezielle Funktionen in Ihrer Datenbankanwendung aufrufen oder auch nicht (Ansprechpartner-Kontakte lassen sich beispielsweise Firmenkontakten zuordnen, Lieferanten aber nicht, da diese Kontakte je selbst eine Firma darstellen – nur als Beispiel). Ohne Enums könnte man den Code so schreiben, dass man Ihre Funktionen auch mit der Nummer (z.B. mit einem Integer) aufruft, aber dies ist zum einen schwer zu merken und zum anderen könnten Sie kaum verhindern, dass man die Funktionen mit 42 aufruft; was als Integer vollkommen in Ordnung wäre – nur haben Sie eben keine 42. Kategorie, sondern gerade mal 10 o. Ä. So leisten Enums sowohl die bessere Merkbarkeit als auch die Einschränkung möglicher Werte.

Wie gesagt: Sie können sich eine eigene Liste erstellen und die Zuordnung Kontaktkategorie/Nummer sozusagen im Kopf durchführen. Oder Sie legen praktischerweise eine solche Enum an, die Ihnen die Arbeit erleichtert.

BEGLEITDATEIEN: Die in einem Projekt zusammengefassten Beispiele für dieses Kapitel finden Sie im Verzeichnis `.\VB 2005 - Entwicklerbuch\E - Datentypen\Kap18\Enums\`.

```

Public Enum KontaktKategorie
    Familie
    Freund
    Bekannter
    Kollege
    Geschäftspartner
    Kunde
    Lieferant
    ZuMeiden
    Firma
    AnsprechpartnerBeiFirmenkontakt
End Enum

```

Wie setzen Sie diese Enum-Elemente nun ein? Ganz einfache Geschichte: Wenn nichts anderes gesagt wird, vergibt .NET den Enums Werte, und zwar von oben nach unten bei 0 angefangen in aufsteigender Reihenfolge – was Sie nun aber nicht mehr wissen müssen. Sie können Variablen vom Typ der Aufzählung definieren und anstatt eine Nummer zu verwenden, einfach den Enum-Typ angeben:

```

Sub main()

    Dim locKontakte As KontaktKategorie

    locKontakte = KontaktKategorie.Geschäftspartner
    Console.WriteLine(locKontakte)
    Console.ReadLine()

End Sub

```

Wenn Sie diesen Code ausführen, gibt es Ihnen im Konsolenfenster den Wert 4 aus.

Bestimmung der Werte der Aufzählungselemente

Falls Sie mit der vorgegebenen Durchnummerierung nicht einverstanden sind, legen Sie bei der Aufzählungsdefinition eben selbst Hand an, wie im folgenden Beispiel:

```

Public Enum KontaktKategorie
    Familie = 10
    Freund
    Bekannter
    Kollege
    Geschäftspartner
    Kunde = 20
    Lieferant
    ZuMeiden = 30
    Firma
    AnsprechpartnerBeiFirmenkontakt
End Enum

```

Das gleiche Programm, abermals ausgeführt, liefert anschließend den Wert 14.

Dubletten sind erlaubt!

Dubletten sind bei den Aufzählungselementen übrigens erlaubt. So haben in der Aufzählungsdefinition

```
Public Enum Kontaktkategorie
    Familie = 10
    Freund
    Bekannter
    Kollege
    Geschäftspartner = 20
    Kunde
    Lieferant = 19
    ZuMeiden
    Firma
    AnsprechpartnerBeiFirmenkontakt
End Enum
```

sowohl Geschäftspartner als auch ZuMeiden den Wert 20.

Bestimmung der Typen der Aufzählungselemente

Solange nichts anderes gesagt wird, werden die Elemente einer Aufzählung intern als Integer angelegt. Sie können allerdings bestimmen, ob die Aufzählungselemente darüber hinaus als Byte, Short oder Long deklariert werden sollen. Eine entsprechende Typen-Angabe bei der Enum-Definition reicht aus:

```
Public Enum KontaktKategorie As Short
    Familie
    Freund
    Bekannter
    ...
End Enum
```

Ermitteln des Elementtyps zur Laufzeit

Wenn Sie zur Laufzeit ermitteln müssen, welcher primitive Datentyp sich hinter einem Aufzählungselement verbirgt, machen Sie das einfach mit der `GetUnderlyingType`-Eigenschaft:

```
'Ermittelt den Namen des zu Grunde liegenden primitiven Datentyps einer Aufzählung.
Console.WriteLine([Enum].GetUnderlyingType(GetType(KontaktKategorie)).Name)
```

```
'Ermittelt den Typnamen anhand einer Aufzählungsvariablen.
Console.WriteLine([Enum].GetUnderlyingType(100Kontakte.GetType()).Name)
```

Auf unser bisheriges Beispiel angewendet, würden diese beiden Zeilen folgende Ausgabe im Konsolenfenster erscheinen lassen:

```
Int16
Int16
```

WICHTIG: Wenn Sie auf den Enum-Typbezeichner im Quelltext zugreifen, müssen Sie, wie hier im Beispiel, das Schlüsselwort in eckige Klammern setzen, damit es vom Visual Basic-Editor korrekt als Ausdruck verarbeitet werden kann.

Konvertieren von Enums

In vielen Fällen kann es sinnvoll sein, ein Aufzählungselement in seinen zu Grunde liegenden Typ umzuwandeln – beispielsweise um den Wert einer Enum-Variablen in eine Datenbank zu schreiben. Einige Fälle machen es auch nötig, einen Aufzählungswert auf Grund des als Zeichenkette vorliegenden Namens eines Aufzählungselementes entstehen zu lassen oder ein Aufzählungselement in seinen Elementnamen (also in einen String) umzuwandeln.

In Zahlenwerte umwandeln und aus Werten definieren

Um ein Aufzählungselement in seinen Wert umzuwandeln (und im Bedarfsfall auch zurück), verfahren Sie folgendermaßen (das folgende Beispiel geht immer noch von einer Aufzählungstyp-Definition als Short aus):

```
Dim locKontakte As KontaktKategorie
Dim locShort As Short

locKontakte = KontaktKategorie.Geschäftspartner

'Typumwandlung von Aufzählung zu zu Grunde liegenden Datentyp...
locShort = locKontakte
locShort = KontaktKategorie.Firma

'...und umgekehrt, auch nicht schwieriger:
locKontakte = CType(locShort, KontaktKategorie)
```

In Strings umwandeln und aus Strings definieren

Falls Sie wissen wollen, welcher Elementname sich hinter dem Wert einer Aufzählungsvariablen verbirgt, verfahren Sie folgendermaßen:

```
Dim locKontakte As KontaktKategorie = KontaktKategorie.Firma
Console.WriteLine(locKontakte.ToString())
```

Die Ausgabe lautet:

```
Firma
```

Beim umgekehrten Verfahren wird es wieder ein wenig aufwändiger:

```
'Umwandlung zurück in ein Enum-Element aus einem String.
Dim locString As String = "Geschäftspartner"
locKontakte = DirectCast([Enum].Parse(GetType(KontaktKategorie), locString), KontaktKategorie)
```

Hier gilt: Die statische Funktion `Parse` der Enum-Klasse erlaubt das Generieren eines Aufzählungselementes zur Laufzeit. `Parse` erwartet dabei den Typ der Aufzählung, den Sie zunächst mit `GetType` ermitteln müssen. Da `Parse` ein Objekt erzeugt, das ein geboxtes Aufzählungselement enthält, müssen Sie dieses zu guter Letzt mit `DirectCast` aus dem `Object` wieder »entboxen«.

Flags-Enum (Flags-Aufzählungen)

Flags-Aufzählungen sind eine ideale Einrichtung, wenn Sie Aufzählungen mit Elementen verwenden müssen, die untereinander kombinierbar sind. So kann es durchaus sein, dass – um bei unserem Beispiel zu bleiben – ein Kontakt in Ihrer Datenbank sowohl ein *Freund* als auch ein *Geschäftskollege* ist. Das Framework unterstützt solche Szenarien auf ideale Weise.

Bei der Definition einer *Flags*-Aufzählung müssen Sie drei Dinge beachten: Sie sollten eine Aufzählungsbenennung für keine der Kombinationen definieren (beispielsweise in Form von *Keine* oder *None*). Dieser Eintrag hat den Wert 0. Zweitens: Sie müssen Werte vergeben, die sich bitweise kombinieren lassen – und dazu zählen Sie die einzelnen Werte als Zweierpotenzen hoch. Und drittens: Sie stellen die Aufzählung mit dem `Flags`-Attribut aus.

Das folgende Beispiel zeigt, wie es geht:

```
<Flags(> _
Public Enum KontaktKategorien
    Keine = 0
    Familie = 1
    Freund = 2
    Bekannter = 4
    Kollege = 8
    Geschäftspartner = 16
    Kunde = 32
    Lieferant = 64
    ZuMeiden = 128
    Firma = 256
    AnsprechpartnerBeiFirmenkontakt = 512
End Enum
```

Wenn Sie anschließend kombinierte Zuweisungen an eine Variable vom Typ `KontaktKategorien` vornehmen wollen, nehmen Sie den logischen `Or`-Operator zu Hilfe, wie das folgende Beispiel zeigt:

```
Sub EnumFlags()

    Dim locKontakte As KontaktKategorien
    locKontakte = KontaktKategorien.Freund Or KontaktKategorien.Geschäftspartner
    Console.WriteLine(locKontakte)
    Console.WriteLine(locKontakte.ToString())

    Console.ReadLine()

End Sub
```

Dieses Beispiel erzeugt die folgende Ausgabe:

```
18
Freund, Geschäftspartner
```

Abfrage von Flags-Aufzählungen

Bei der Abfrage von *Flags*-Aufzählungen müssen Sie ein wenig aufpassen, da Kombinationen Werte ergeben, die keinem bestimmten Wert eines Elementes entsprechen. Erst ein so genanntes Ausmaskieren (Ermitteln eines einzelnen Bitwertes) mit dem *And*-Operator ergibt einen richtigen Wert. Auch hier demonstriert ein Beispiel, wie es geht:

```
Dim locKontakte As KontaktKategorien
locKontakte = KontaktKategorien.Freund Or KontaktKategorien.Geschäftspartner

'Achtung bei Flags! Bits müssen ausmaskiert werden!
'Diese Abfrage liefert das falsche Ergebnis!
If locKontakte = KontaktKategorien.Geschäftspartner Then
    Console.WriteLine("Du bist ein Geschäftspartner")
Else
    Console.WriteLine("Du bist kein Geschäftspartner")
End If

'So ist's richtig; diese Abfrage liefert das richtige Ergebnis:
If (locKontakte And KontaktKategorien.Freund) = KontaktKategorien.Freund Then
    Console.WriteLine("Du bist ein Freund")
Else
    Console.WriteLine("Du bist kein Freund")
End If

'Und so funktionieren Kombiabfragen:
If locKontakte = (KontaktKategorien.Freund Or KontaktKategorien.Geschäftspartner) Then
    Console.WriteLine("Du bist ein Freund und ein Geschäftspartner")
End If
```

Wenn Sie dieses Beispiel laufen lassen, erhalten Sie das folgende Ergebnis:

```
Du bist kein Geschäftspartner
Du bist ein Freund
Du bist ein Freund und ein Geschäftspartner
```

Die erste Zeile wird falsch ausgegeben, da der Wert der *Enum*-Variablen *locKontakte* nicht ausmaskiert und dann verglichen, sondern direkt verglichen wird.