

Automatische Stammdaten-Korrektur-Script als Beispiel

Use-Case- Alle Titel entweder exklusiv oder zusätzlich dem Archiv zuordnen.

```
uses Common;

procedure doSetup
begin
    ...
end;

procedure doRun;
var litStock    : TStock;
var IArchiveSegment: TSegment;
var IArchiveSegmentBitMask: TBitmask;

begin
    (* Alle Werte in Archiv. *)
    IArchiveSegment:=varSegments.ItemsName['Archiv'];
    IArchiveSegmentBitMask:=IArchiveSegment.getNrAsBitMask;
    for idxStock:=1 to varStocks.Count do
    begin
        litStock:=varStocks.Items[idxStock-1];
        if (varSegments.getAndToInternalNrAsBool( litStock.StockSegments, IArchiveSegmentBitMask)) then
        begin
            (* Alle Werte zusätzlich dem Archiv zuordnen *)
            litStock.StockSegments:=varSegments.getORToInternalNr( litStock.StockSegments,
IArchiveSegmentBitMask);
            (* oder exklusiv d.h. alle Werte nur ins Archiv verschieben
            itStock.StockSegments:=IArchiveSegmentBitMask;
            *)
        end;
    end;
end;
end;
```

```
begin
  doSetup();
  doRun();
end;
```

Ab der 21.6.x Version werden die Markt und Watchlisten-Zuordnungen über Bitmasken zugeordnet.

Diese sind vom Typ TBitmask:

```
type
  TBitmask = record
    BitFields: Array[1..8] of UInt64;

    const MaxFields = 8;
    const MaxBit = MaxFields*64-1;

    class operator Add(const v1, v2: TBitmask): TBitmask;
    class operator Equal(const v1, v2: TBitmask): Boolean;
    class operator NotEqual(const v1, v2: TBitmask): Boolean;

  end align 16;
```

Das komplette Zurücksetzen eines Titels kann damit mit folgender Konstante erfolgen:

```
const cZeroBitmask : TBitmask = ( BitFields: (0,0,0,0,0,0,0,0) );
```

Beispielzuordnungen als Konstante sind z.B.

```
cEuroStoxx50: TBitmask = (BitFields: (16,0,0,0,0,0,0,0));
cNasdaq      : TBitmask = (BitFields: (1 shl 12,0,0,0,0,0,0,0));
```

Folgende Methoden für alle von TEnhancedList abgeleiteten Klassen z.B. TSegments, TWatchlists, TStocks stehen zur Verfügung:

```
class function getORToInternalNr(orValue, Nr: TBitmask): TBitmask;
class function getORToNr(orValue, Nr: Int64): Int64;

class function getXORToInternalNr(xorValue, Nr: TBitmask): TBitmask;
```

```
class function getAndToInternalNr(andValue, Nr: TBitmask): TBitmask;
```

```
class function getAndToInternalNrAsBool(andValue, Nr: TBitmask): Boolean;
```

```
class function getAndToNrAsBool(andValue, Nr: Int64): Boolean;
```

```
class function getAndNotToInternalNr(Nr, andNotValue: TBitmask): TBitmask;
```

An den Objektklassen selbst stehen zur Umrechnung immer zur Verfügung:

```
function PrimaryKeyID: Int64; virtual;
```

```
function getNrAsBitMask: TBitmask;
```

```
class function convertBitFieldToMask( Nr: Int64 ): TBitmask;
```

```
class function convertToMask( Nr: Int64 ): TBitmask;
```

Revision #1

Created 19 June 2022 07:48:27 by Jens Werschmoeller

Updated 19 June 2022 07:48:55 by Jens Werschmoeller