# Programmier- und Namenskonventionen



## 1 Variablen

Variablen setzen sich aus Gültigkeit, Typ und Bezeichnung zusammen.
Englische Sprache ist zu verwenden.

### 1.1 Gültigkeit

| | |
|---|---|
| Klassenvariable, privat – m | m |
| Klassenvariable, public | Nicht zulässig /Properties verwenden! |
| Lokale Variablen | (ohne) |

### 1.2 Standard- Typkennungen:

| | |
|---|---|
| Boolean | -b |
| Integer | -l |
| String | -s |
| Decimal | -c |
| Datum | -dt |
| Objekt (typisiert) | -o |
| Objekt (nicht typisiert) | -vnt /o |
| Array | - jeweils ‚a' vor der Typkennung |
| Variable vom Typ enum-x | -en |

### 1.3 Besondere Typangaben:

| | |
|---|---|
| Recordset | -rs |
| XML-String | -sXml |

### 1.4 Bezeichnung

Bezeichnet den Inhalt der Variablen in englischer Sprache. Bei Arbeitsvariablen darf die Bezeichnung entfallen, z.B. „for i as integer = 0 to…"

# 2 Konstanten / Enums

Konstanten werden in Klein-/Groß-Schreibweise mit dem Präfix „ef“ benannt. Danach folgt der Typ analog zu denVariablen.

Beispiel:
Private Const efsResetMethod As String = „gResetUser“

Enums werden zusätzlich mit dem Bezeichner „Enum“ gekennzeichnet:

Beispiel:
```
Public Enum efEnumDataSourceTypes
        efIDataSourceTypeMSSQL = 0
End Enum
```

# 3 Prozeduren / Funktionen

Name setzt sich aus Gültigkeit, Typ, Aktion und Bezeichnung zusammen.

## 3.1 Gültigkeit:

| | |
|---|---|
| Globale Funktionen | -g |
| Klasseninterne Funktionen | -m |

## 3.2 Typ:

Siehe Variablen

## 3.3 Aktion:

Get, Set, Init, Reset, Insert, Delete, Copy, Recalc
(bei häufig eingesetzten Toolfunktionen wie z.B. gsCStr kann auf den Aktionsbezeichner verzichtet werden)

## 3.4 Bezeichner:

Bezeichnet den Objekttyp oder die Aktion in englischer Sprache, z.B.:
gbInsertProduct, gResetQuestions, grsGetAnswerList, mlGetProductNumber

# 4 (Klassen-)properties

Klassenproperties bestehen aus Präfix für den Datentyp und einer englischen Bezeichnung.

# 5 Datenbankobjekte

Namen von DB-Objekten bestehen aus einem kennzeichnenden Präfix und einer englischen Bezeichnung.

| | |
|---|---|
| Datentabellen | -td |
| Systemtabellen | -ts |
| Views | -vw |
| Stored procs | -sp (plus Aktionsbezeichner wie bei Prozedur/Funktion) |
| Felder | - ohne; wichtig: keine Schlüsselworte wie Name, ID, Index verwenden! |
| Primärindex | -aIdx |
| Index | -zIdx |
| Foreign-Key-Constraint | -fkTable1Table2<br>z.B. fkTdSurveysTdSurveyTypes<br>Table1 ist die Master-Tabelle, Table2 ist die Detailtabelle bzw. die, die die Lookupwerte enthält |
| Spaltenpräfix | setzt sich aus markanten Buchstaben des Tabellenbezeichners zusammen + „_“-Zeichen; maximal 3-stellig, z.B. qst_id, qst_text… für Tabelle „tdQuestions“;<br>NN-Tabellen erhalten die Spaltenpräfix mit einem „_“ getrennt. Das Spaltenpräfix, das alphabetisch kleiner ist, wird zuerst verwendet, z.B. „ans_qst_ans_id“; ans_qst_qst_id |
| Name Primärspalte | Spaltenpräfix + „id“, z.B. „qst_id“, „ans_id“ |
| Project-Spalte | In Systemtabellen, in die eigene Projektdaten eintragen werden können ist immer eine „Project“-Spalte (ohne Präfix) vorhanden; hier muss das Projectkürzel eingetragen werden,z.B. tsSysEvents, tsProcesses |
| Spalte Erstellt, Bearbeitet | Datentyp Datetime; Spaltenpräfix + „inserted“, „updated“, z.B. „qst_inserted“, „ans_updated“, |
| Spalte erstellt von und bearbeitet von | “qst_updatedby”, “qst_createdby” |
| Check-Constraint | -chk_<Feldname> |
| NN-Tabellen | Tabellen, die zwischen zwei Tabellen stehen und diese Verknüpfen.<br>zuerst soll der Name der logisch hierarchisch tieferstehenden Tabelle verwendet werden, dann der andere Tabellenname, z.B.: „tdUsersTdUserGroups“<br>Usergroups stehen höher als Users, werden jedoch über NN- miteinander verbunden. |
| Feldnamen in NN-Tabellen | Zuerst das Kürzel der logischer tieferstehenden Tabelle, dann das Kürzel der höheren Tabelle, z.B. in „tdUsersTdUserGroups“ |

| | |
|---|---|
| | heissen die Standardfelder:<br>usr_grp_id<br>usr_grp_usr_id<br>usr_grp_grp_id |

**Eigen erstellte Tabellen** erhalten das Projektkennzeichen, statt „td“ (Systemtabellen werden bei abgeleiteten easyFramework-Projekten nicht erstellt), z.B. svmAnswers, svmQuestions

# 6 Komponenten

## 6.1 Klassen

Klassen erhalten einfache Bezeichnungen ohne Präfix.

## 6.2 Assemblyname

Assemblies erhalten as Präfix „ef“, danach ein Kürzel für den Bereich, dann eine sprechende Bezeichnung.

Beispiele: efMenuTree, efXmlDialog

**Eigene Projektassemblies** ersetzen „ef“ durch das entsprechende Projektkürzel.

## 6.3 Stammnamespace

Der Stammnamespace beschreibt den Bereich, in dem die Komponente angesprochen werden soll.

Alle Stammnamespace beginnen mit „easyFramework“.

Unterschieden wird dann nach den entsprechenden Bereichen. Zum Beispiel „ef.Frontend.ASP“ wird für Frontends verwendet.

**Eigene Projekte** verwenden hier easyFramework.Projects.[Projektkürzel]. …

# 7 Quellcode

## 7.1 Aufbau Klassendatei:

- Kommentarkopf
- Konstantendefinitionen
- Modulvariablen
- Public Methoden
- Private Methoden

Funktionen / Prozeduren beginnen in der ersten Spalte, jeder Block ist mit Tab (4 Zeichen) einzurücken.

Beispiel:
Public Function gsCStr(ByVal vntValue as Object) as String

```
    If IsNull(vntValue) Then
        gsCStr = ""
    Else
        gsCStr = CStr(vntValue)
    End If

End Function
```

Zeilenumbrüche sind so zu setzen, dass Betrachten des Quellcodes ohne Scrollen möglich ist.

Beispiel:

```
sQry = sQry & „Text1“ & _
    „Text2“
```

# 8 Kommentare

## 8.1 Modulköpfe

Enthalten Namen des Projekts, der Komponente und der Klasse. Dazu eine kurze Doku zum Einsatzzweck. Weiterhin Datum der Erstanlage, Bearbeiter, und jeweils Änderungsdatum und Inhalt der Änderung.

## 8.2 Funktionsköpfe

Enthalten Name der Methode, kurze Beschreibung der Aufgabe, Parameter und ggf. Rückgabewert

## 8.3 Zwischenkommentare

Dienen zur Abgrenzung der einzelnen Bereiche in einem Quellcode-File. Sie dienen quasi als Überschrift für den folgenden Abschnitt. Beispiel: „Public Methoden“

## 8.4 Kommentare im Code

An Stellen, bei denen die Zusammenhänge nicht auf den ersten Blick erkennbar sind, werden Kommentare direkt im Code eingefügt.

Beispiel unter /templates/sourcecode/

```
'==========================================================================================
' Project:      easyFramework
' Copyright:    Promain Software-Betreuung GmbH
' Component:    Class1.vb
'------------------------------------------------------------------------------------------
' Purpose:      for doing a special job about something
'------------------------------------------------------------------------------------------
' Created:      14.03.04 M.Wimmer (mwimmer@promain-software.de)
'------------------------------------------------------------------------------------------
' Changed:          14.03.04 Andy Tropschug
'==========================================================================================
Option Explicit On
Option Strict On

'==========================================================================================
'Const
'==========================================================================================
Private Const As String efsResetMethod = "gResetUser"

'==========================================================================================
'Imports
'==========================================================================================
Imports easyFramework.Sys.Data
Imports easyFramework.Sys.Xml

'==========================================================================================
'Module variables:
'==========================================================================================
Private msUser As String


'==========================================================================================
'Public Methods:
'==========================================================================================

'==========================================================================================
' Method:       gbTransferData
'------------------------------------------------------------------------------------------
' Purpose:      creates the export-file and exports the data
'------------------------------------------------------------------------------------------
' Parameter:    oClientInfo         - ClientInfo
'               sXmlCustomerList    - retrieved by gsGetCustomerList e.g.
'------------------------------------------------------------------------------------------
' Changed:
'==========================================================================================
Function goTransferData(ByVal oClientInfo As ClientInfo, _
    ByVal sMandantNr As String, _
    ByVal sKostenkreis As String) As myfactory.Sys.Xml.XmlDocument


    Dim otnKostenstellen As New SBSFibu.Tsenit.tnKostenstellenFile(sMandantNr)

    Dim oKostenkreis As SBSFibu.Tsenit.tnKostenkreis = _
        otnKostenstellen.AddKostenkreis(sKostenkreis)


    Loop

    'create output-file:
    Return otnKostenstellen.renderToXml

End Function


'==========================================================================================
'Public Methods:
'==========================================================================================
```

# 9 Datenbankdesign

- Jede Tabelle bekommt einen Primärschlüssel
- IDENTITY-Spalten werden nicht verwendet. Bei Bedarf RecordIDs einsetzen.
- Cascading UPDATE und DELETE wird nicht verwendet.
- FOREIGN KEYs sind zulässig
- Alle Stringfelder sind entweder nvarchar oder ntext
- BOOL-Felder sind BIT NOT NULL Default(0) oder DEFAULT (1)
- Beträge sind im Money-Typ
- Ganzzahlen sind im int-Typ
- CHECK-Constraints sind erlaubt