



ColorCloud

HAMBURG

18. - 19.04.2024

**Was zum Henker ist
AL?**

**Grundlagen der
Entwicklung in
Business Central für
Nicht-BC-Entwickler**

Tobias Fenster
Simon Fischer



Tobias Fenster

Managing Director, 4PS Germany
MVP Azure & BizApps, RD, Docker Captain

 Germany

 tobiasfenster

 tobiasfenster

 tfenster@4ps.de

 tobiasfenster.io

#ColorCloudSpeaker 



Simon Fischer

Senior Software Developer, 4PS Germany

 Germany

 SimonOfHH

 SimonOfHH

 sfischer@4ps.de

 SimonOfHH.tech

#ColorCloudSpeaker 

ColorCloud Sponsors

THANK YOU! 

Gold Sponsors



Silver Sponsors



Digital Sponsors



Agenda

- **Überblick**
- Details und Demos
- Und jetzt?

Objekte in AL

- Objekte in Business Central != *Objektorientierung (OOP)* in anderen Sprachen
- Code und Strukturen werden in unterschiedlichen Objekttypen gespeichert
 - Tables (entsprechen Tabellen auf SQL Server)
 - Pages (UI u.a. zum Darstellen von Daten)
 - Codeunits („Container“ für Businesslogik, grob vergleichbar mit DLLs)
 - weitere für unterschiedliche Zwecke (Reports, XMLports, Profile, ...)

Business Logik mit AL entwickeln

- Programmiersprache der 4. Generation („4GL = fourth generation language“) mit wesentlich mehr generativem Ansatz als 3GL (C#, Java, Pascal, ...)
- Optimiert für die Entwicklung von Anwendungslogik
- Sehr schnell erlernbar, relativ einfach einsetzbar
 - Aber wie überall, um überdurchschnittlich gut zu sein braucht es Talent und Fleiß...
- Pascal-ähnliche Syntax

Business Logik mit AL entwickeln

- Business Central mit klassischer 3-Schicht-Architektur: Datenbank, Mittel-/Anwendungsschicht, (Web-)Client
- Code wird in die Datenbank importiert, Mittelschicht generiert C#-Code, SQL-Strukturen, Frontend
- Durch generativen Ansatz zwar weitreichender, aber nicht allumfassender Einfluss auf alle Schichten
- Sehr leistungsfähiger Mechanismus für Lokalisierung und Erweiterung

Business Logik mit AL entwickeln

```
Codeunit 50000 CrInvForCust10000And70000 - C/AL Editor
+ Documentation()
+ OnRun()
- LOCAL CreateInvoiceDocument(VAR SalesHeader : Record "Sales Header")
  WITH SalesHeader DO BEGIN
    INIT;
    "Document Type" := "Document Type"::Invoice;
    "No." := ' ';
    INSERT(TRUE);
  END;
- LOCAL TestIfCustomerIsBlocked()
  WITH Customer DO BEGIN
    GET('10000');
    TESTFIELD(Blocked, Blocked::" ");
  END;
- LOCAL SelectCustomer10000(VAR SalesHeader : Record "Sales Header")
  WITH SalesHeader DO
    VALIDATE("Sell-to Customer No.", '10000');
- LOCAL SaveHeaderToDatabase(VAR SalesHeader : Record "Sales Header")
  WITH SalesHeader DO
    MODIFY(TRUE);
- LOCAL CreateSalesLine(VAR SalesHeader : Record "Sales Header";VAR SalesLine : Record "Sales Line")
+ LOCAL TestIfItemIsBlocked()
+ LOCAL SelectItem70000(VAR SalesLine : Record "Sales Line")
+ LOCAL SaveLineToDatabase(VAR SalesLine : Record "Sales Line")
+ LOCAL OpenSalesInvoicePage(VAR SalesHeader : Record "Sales Header")
```

Business Logik mit AL entwickeln

```
AL CrInvForCust10000And70000.Codeunit.al X
AL CrInvForCust10000And70000.Codeunit.al > ...
0 references
1 codeunit 50101 CrInvForCust10000And70000
2 {
3     0 references
4     local procedure CreateInvoiceDocument(var SalesHeader: Record "Sales Header")
5     begin
6         SalesHeader.Init();
7         SalesHeader."Document Type" := SalesHeader."Document Type"::Invoice;
8         SalesHeader."No." := '';
9         SalesHeader.Insert(true);
10    end;
11    0 references
12    local procedure TestIfCustomerIsBlocked()
13    var
14        Customer: Record "Customer";
15    begin
16        Customer.Get('10000');
17        Customer.TestField("Blocked", Customer."Blocked"::" ");
18    end;
19    0 references
20    local procedure SelectCustomer10000(var SalesHeader: Record "Sales Header")
21    begin
22        SalesHeader.Validate("Sell-to Customer No.", '10000');
23    end;
24    0 references
25    local procedure SaveHeaderToDatabase(var SalesHeader: Record "Sales Header")
26    begin
27        SalesHeader.Modify(true);
28    end;
29    0 references
30    local procedure CreateSalesLine(var SalesHeader: Record "Sales Header"; var SalesLine: Record "Sales Line") ...
31    0 references
32    local procedure TestIfItemIsBlocked() ...
33    0 references
34    local procedure SelectItem70000(var SalesLine: Record "Sales Line") ...
35    0 references
36    local procedure SaveLineToDatabase(var SalesLine: Record "Sales Line") ...
37    0 references
38    local procedure OpenSalesInvoicePage(var SalesHeader: Record "Sales Header") ...
39    }
40 }
```

Business Logik mit AL entwickeln

- Visual Studio Code als IDE
- Extension „AL Language“ frei und kostenlos verfügbar
- Entwicklung aktuell auf Windows und Mac offiziell supportet, auch Linux möglich
- Server-Komponenten über „bccontainerhelper“ einfach als Container unter Windows aufbaubar
- Keine großen Hürden, um mit Entwicklung zu starten
- Noch einfacher: COSMO Alpaca, Entwicklungsplattform für Business Central

Agenda

- Überblick
- **Details und Demos**
- Und jetzt?

Objekte in AL – Tables und Pages

- Tables beinhalten Felder, Indices, Metadaten
- Definieren, wie Daten in der Datenbank gespeichert werden
- Direkt in der Programmierung angesprochen, kein SQL, ORM-Framework o.ä. notwendig
- Transaktionshandling automatisch, aber beeinflussbar
- Sonderform temporäre Tabellen → nicht persistiert

Objekte in AL – Tables und Pages

- Pages = UI
- Definieren, wie Daten im Client dargestellt oder eingegeben werden können
- Verschiedenen Pagetypen (Liste, Karte, Kopf-/Zeile, Matrix, ...), aber kein Detaileingriff möglich
- Aktionen in Zeilen, im Kopfbereich oder im “Ribbon” ebenfalls hier definiert und gesteuert
- Unterscheidung zwischen direkt sichtbar, erweitert sichtbar oder unsichtbar (aber vom Benutzer einblendbar)



Objekte in AL – Beispiel Table & Page

```
AL Simple.Table.al X
AL Simple.Table.al > Table 50100 "CCSF-Simple Table" > fields > Activ
1 namespace ColorCloud.Samples;
2 1 reference
3 table 50100 "CCSF-Simple Table"
4 {
5     DataClassification = CustomerContent;
6     fields
7     {
8         2 references
9         field(1; Code; Code[20])
10        {
11            Caption = 'Code';
12        }
13        1 reference
14        field(10; Description; Text[50])
15        {
16            Caption = 'Beschreibung';
17        }
18        1 reference
19        field(11; Active; Boolean)
20        {
21            Caption = 'Aktiv';
22        }
23    }
24    keys
25    {
26        0 references
27        key(Key1; Code)
28        {
29            Clustered = true;
30        }
31    }
32 }
33 }
```

```
AL Simple.Page.al X
AL Simple.Page.al > ...
1 namespace ColorCloud.Samples;
2 0 references
3 page 50100 "CCSF-Simple Page"
4 {
5     PageType = List;
6     ApplicationArea = All;
7     UsageCategory = Lists;
8     SourceTable = "CCSF-Simple Table";
9     layout
10    {
11        0 references
12        area(Content)
13        {
14            0 references
15            repeater(repeater)
16            {
17                0 references
18                field(Code; Rec.Code) { }
19                0 references
20                field(Description; Rec.Description) { }
21                0 references
22                field(Active; Rec.Active) { }
23            }
24        }
25    }
26 }
27 }
```

Dynamics 365 Business Central

CCSF-Simple Page

Suchen + Neu Liste bearbeiten Löschen ...

Code ↑	Beschreibung	Aktiv
→ TEST001	Test Entry 1	<input type="checkbox"/>
TEST002	Test Entry 2	<input checked="" type="checkbox"/>
TEST003	Test Entry 3	<input type="checkbox"/>

Objekte in AL – Codeunits und Erweiterbarkeit

- Codeunits (sollten) komplexen Code beinhalten
- Unterschiedliche Arten der Instanziierung und Ausführung
- Auch hier keine klassische Objektorientierung, aber Interfaces
- Unterschiedliche Sichtbarkeiten

Objekte in AL – Codeunits und Erweiterbarkeit

- Vorhandene Objekte (oft, aber nicht ausschließlich aus MS-Standard) können erweitert werden: Table Extension, Page Extension, ...
- Ergänzung von Funktionalitäten, z.B. für branchenspezifische Anforderungen
 - Kennzeichnung eines Geschäftspartners als Subunternehmer oder Generalunternehmer in der Baubranche
- Events erlauben Einfluss auf Standard-Verhalten
 - Bevor eine Rechnung gebucht wird, soll immer ein spezifisches Stück Code ablaufen

Objekte in AL

- Beispiel für Table/Page-Erweiterung

```
AL ClothingType.Enum.al > Enum 50101 "CCSF-ClothingType"
2 references
1 enum 50101 "CCSF-ClothingType"
2 {
3     Extensible = true;
4
5     value(0; "")
6     {
7         Caption = '';
8     }
9     0 references
10    value(1; Shoe)
11    {
12        Caption = 'Shoe';
13    }
14    1 reference
15    value(2; Shirt)
16    {
17        Caption = 'Shirt';
18    }
19    0 references
20    value(3; Pants)
21    {
22        Caption = 'Pants';
23    }
24 }

AL ShirtSize.Enum.al > Enum 50100 "CCSF-ShirtSize"
1 reference
1 enum 50100 "CCSF-ShirtSize"
2 {
3     Extensible = true;
4
5     0 references
6     value(0; XXS)
7     {
8         Caption = 'XXS';
9     }
10    0 references
11    value(1; XS)
12    {
13        Caption = 'XS';
14    }
15    0 references
16    value(2; S)
17    {
18        Caption = 'S';
19    }
20    0 references
21    value(3; M)
22    {
23        Caption = 'M';
24    }
25 }

AL Simple.TableExt.al > TableExtension 50100 "CCSF-Item"
0 references
1 tableextension 50100 "CCSF-Item" extends Item
2 {
3     fields
4     {
5         2 references
6         field(50100; CCSFClothingType; enum "CCSF-ClothingType")
7         {
8             DataClassification = CustomerContent;
9             Caption = 'Clothing Type';
10        }
11        1 reference
12        field(50101; CCSFShirtSize; enum "CCSF-ShirtSize")
13        {
14            DataClassification = CustomerContent;
15            Caption = 'Shirt Size';
16        }
17        0 references
18        trigger OnValidate()
19        begin
20            // needs to be a shirt to have a shirt size
21            Rec.TestField(CCSFClothingType, enum::"CCSF-ClothingType"::Shirt);
22        end;
23    }
24 }

AL Simple.PageExt.al > PageExtension 50100 "CCSF-Item"
0 references
1 pageextension 50100 "CCSF-Item" extends "Item Card"
2 {
3     layout
4     {
5         addlast(Item)
6         {
7             0 references
8             field(CCSFClothingType; Rec.CCSFClothingType)
9             {
10                ApplicationArea = All;
11                ToolTip = 'Specifies the type of clothing';
12            }
13            0 references
14            field(CCSFShirtSize; Rec.CCSFShirtSize)
15            {
16                ApplicationArea = All;
17                ToolTip = 'Specifies the size of the shirt, if "Clothing Type" is "Shirt"';
18            }
19        }
20    }
21 }
```

Agenda

- Überblick
- Details und Demos
- **Und jetzt?**

Und jetzt?

- [MS Docs: Get started with AL](#)
- [AL Language VS Code Extension](#)
- [BC container helper](#)
- Community Link-Sammlungen:
 - [Nützliche VS Code Extensions für AL](#) von Natalie Karolak
 - [Community-Ressourcen](#) von Steve Endow
 - [BC Archipelago Map](#) von Jeremy Vyska
- [BC Code auf Github](#) (gepflegt von Stefan Maron)



ColorCloud
HAMBURG
18. - 19.04.2024

THANK YOU!



ColorCloud

www.colorcloud.rocks

[#ColorCloud2024](https://twitter.com/ColorCloud2024)



ColorCloud
HAMBURG
18. - 19.04.2024

Q&A



ColorCloud

www.colorcloud.rocks

[#ColorCloud2024](https://twitter.com/ColorCloud2024)

Feedback

Please rate our session to help us improve!

👉 **GIVING FEEDBACK PAYS OFF:**
We are giving away special prizes to all feedback submitters! 🌈

