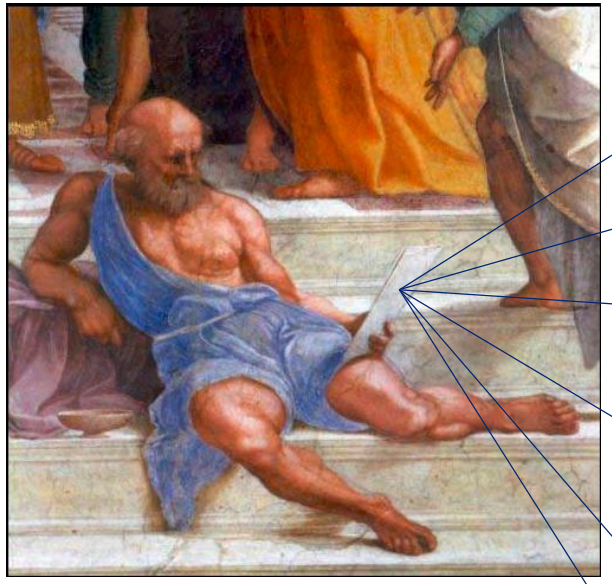
Microsoft AL

06-AL-Basics

Core Statements on the subject of Programming AL

What the wise man says



- Programming AL is almost like programming C/Side TXT files
- Use MS Txt2AL Converter whenever you‘re helpless
- What changes is the object structure: Fobs in Object Designer → AL (txt) Files in File System
- It is crucial to setup a normalized file structure to
  - become more efficient
  - give „everyone“ the chance to „participate a development“ without the need of lengthy project individual introductions
- It is lots more about getting familiar with guidelines and „doings“ than learning to code AL
- if “actions are too cumbersome”, we will think about suitable tool-assistance

Literature / further Reading & Watching

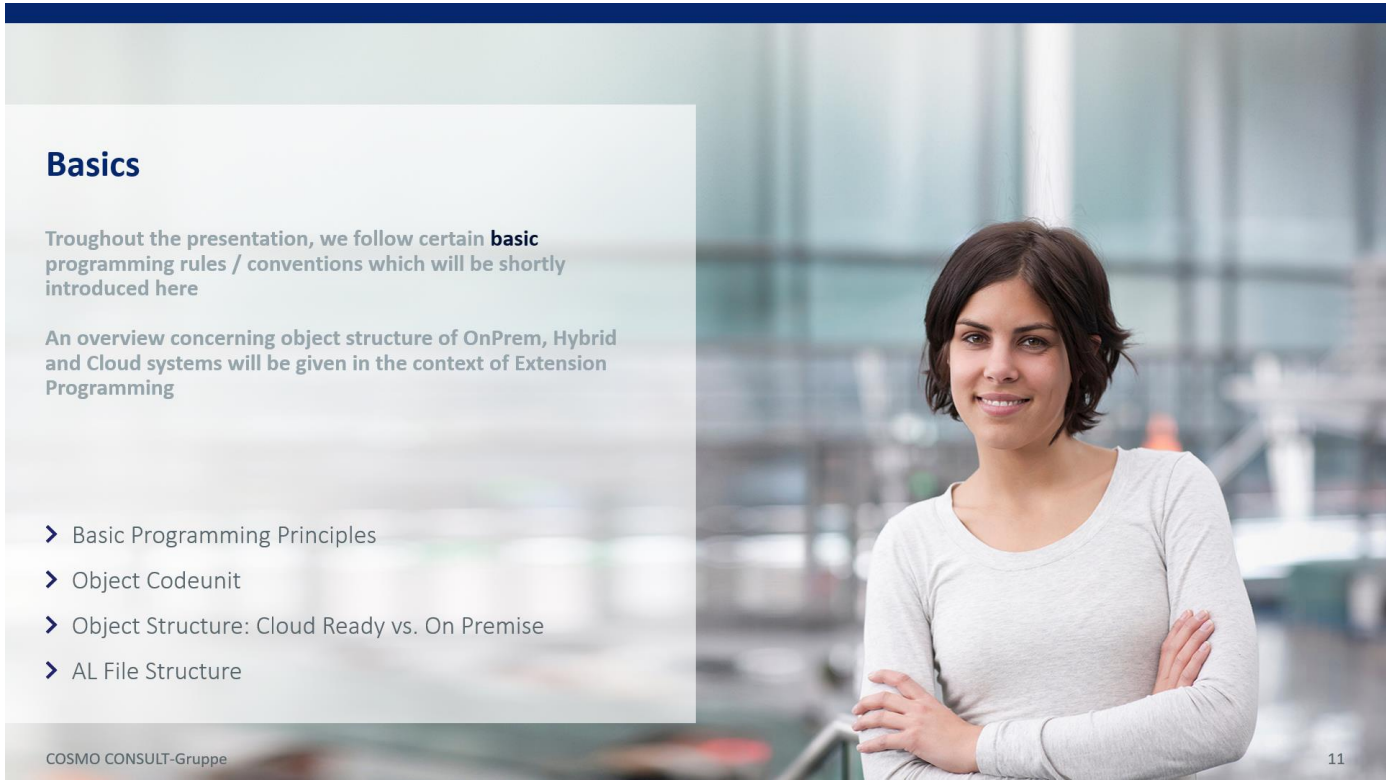
- VS Code
  - https://code.visualstudio.com/docs/getstarted/introvideos
  - https://jpearson.blog/2019/01/31/vs-code-powershell-git-5-things/
  - https://code.visualstudio.com/docs/getstarted/tips-and-tricks: multi cursor editing …
- …

Agenda

1. Basis
2. Boundary Conditions / Setups: HowTo handle a Project - a Repetition
3. Programming in AL
4. Miscellaneous
5. Q & A

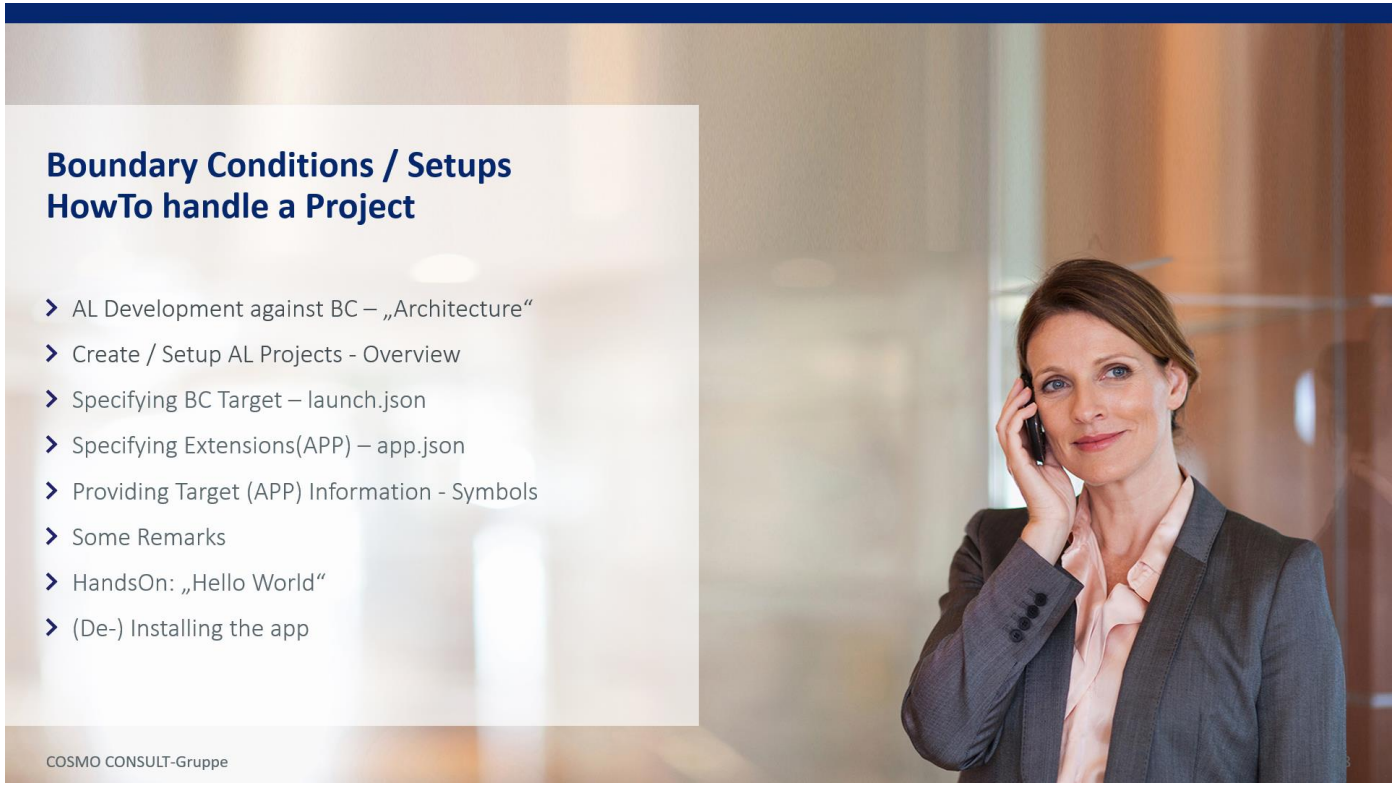
Agenda

1. → Basis ←
2. Boundary Conditions / Setups: HowTo handle a Project - a Repetition
3. Programming in AL
4. Miscellaneous
5. Q & A



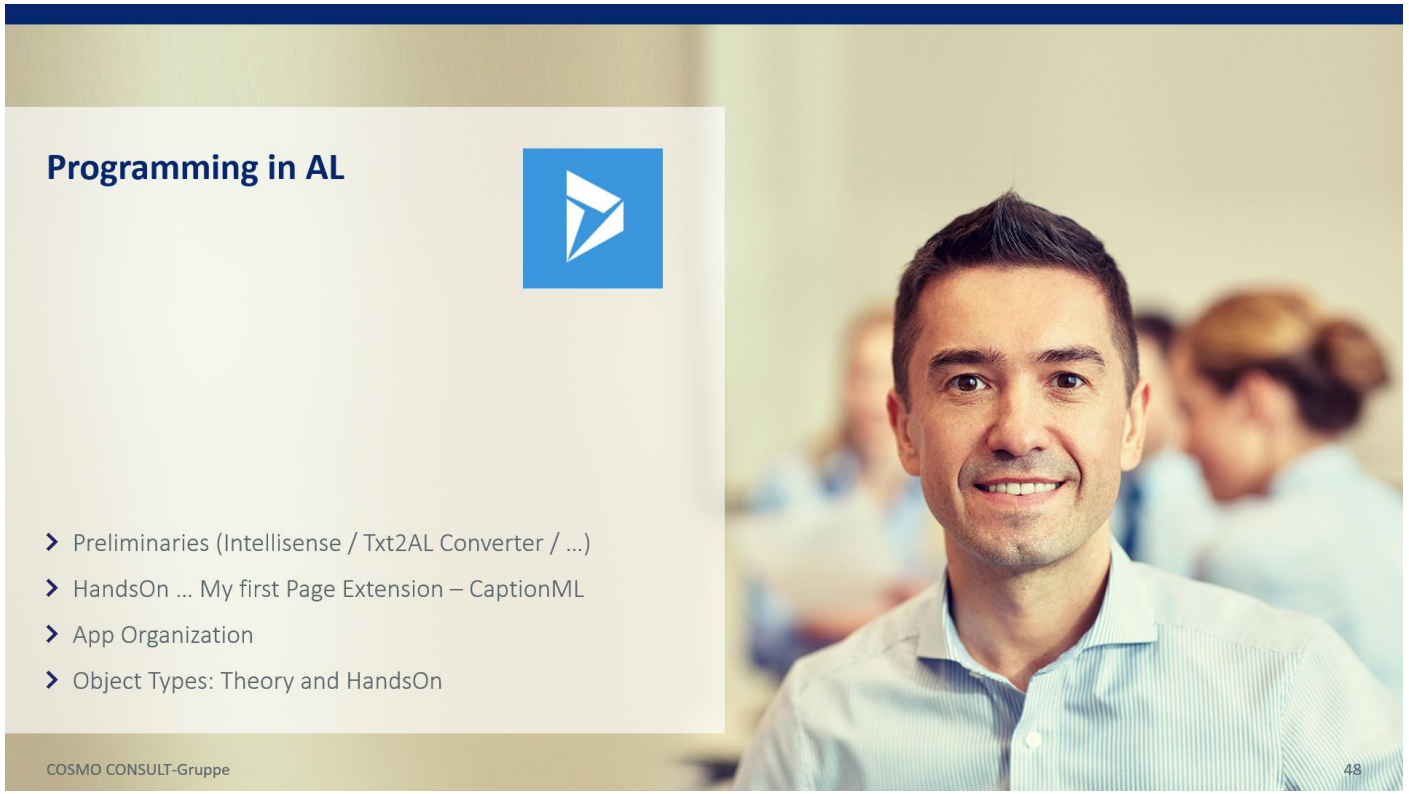
Agenda

1. Basis
2. → Boundary Conditions / Setups: HowTo handle a Project – a short Repetition of “01-VisualStudioCode”?←
3. Programming in AL
4. Miscellaneous
5. Q & A



Agenda

1. Basis
2. Formalities / Boundary Conditions: HowTo handle a Project
3. → Programming in AL ←
4. Miscellaneous
5. Q & A

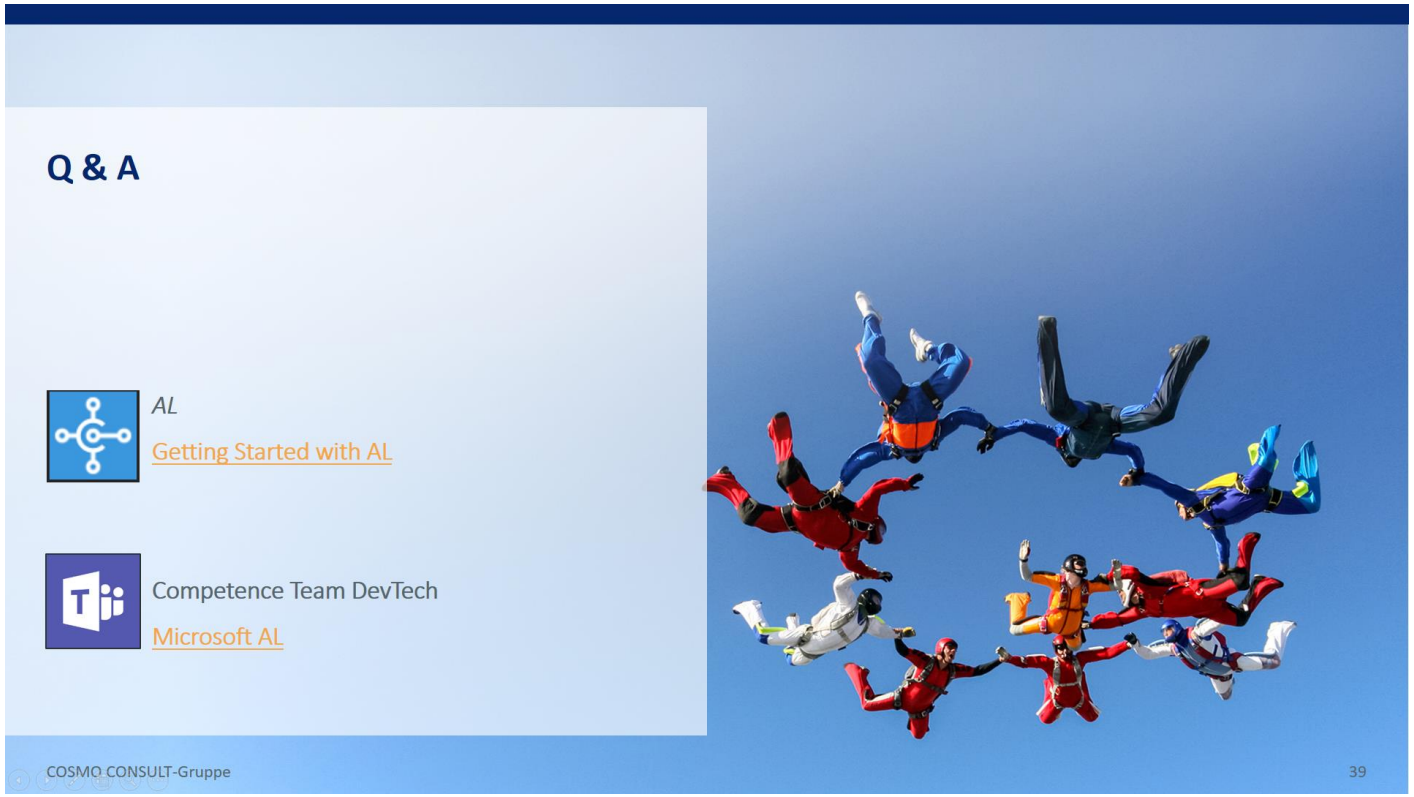


Agenda

1. Basis
2. Formalities / Boundary Conditions: HowTo handle a Project
3. Programming in AL
4. → Miscellaneous ←
5. Q & A

Agenda

1. Basis
2. Formalities / Boundary Conditions: HowTo handle a Project
3. Programming in AL
4. Miscellaneous
5. → Q & A ←



Importants & Exercises

This workshop comes with an extra “Importants and Exercises” brochure (*“06-AL-Importants and ExercisesXXX.pptx”*) where you can find

- Things to remember during the workshop
- Things to look up during the workshop
- Exercises

Basics

Troughout the presentation, we follow certain basic programming rules / conventions which will be shortly introduced here

An overview concerning object structure of OnPrem, Hybrid and Cloud systems will be given in the context of Extension Programming

- Basic Programming Principles
- Object Codeunit
- Object Structure: Cloud Ready vs. On Premise
- AL File Structure

Basics
Basic Programming Principles

I	Agent Principle	Any function can be associated with an actor / agent, the acting entity. In consequence, the function should be located within this entity
II	Code belongs in Codeunits	As far as possible, code should be stored in containers, the Codeunits.
III	Minimum Footprint	Every implementation within standard / product code should be done minimally invasive
IV	Top Down Design / Atomic Functions / Natural Language Programming	Every implementation should read as a iterative sequence of natural language calls, the corresponding ”units” being as atomic (small) as possible. Programming language (nerdy stuff) should appear as late as possible.

Only relevant for OnPrem Implementations since Extensions are minimally invasive by construction

Basics
Object Codeunit

- An „Object Codeunit“ is assigned to each object
- Elements of an Object Codeunit

Element	***Naming***	***Example, agent: Purchase Header Table***	***Remark***
Event Handler (Event Subscriber)	Handle<*Event Function*>[<*Event Publisher Element*>] summarizes all subscriptions of an event in one subscriber, contains a list of the functionalities to be executed	HandleOnAfterValidateDocumentDate DoThis(…); DoThat(…);	Used for manipulating standard / third party behaviour
Trigger Handler (Global Function)	Handle<*Trigger Name*>[<*Trigger Provider Element*>]Trigger summarizes the functionality to be carried out when the trigger is called, defines “our standard behaviour”,contains a list of the functionalities to be executed	HandleOnValidateMyNewFieldTrigger DoLeft(…); DoRight(…);	Used for manipulating behavior of “own elements”, e.g. the validation of “My New Field”
Agent Method (Global Function)	“Name describes / promises behaviour”	PostDocument	Contains the code of a method of the agent. To be called via the agent only!
Local Helper (Local Function)	“Name describes / promises behaviour”	DoThis DoThat DoLeft DoRight AppendVendorNo2PostingDescription	

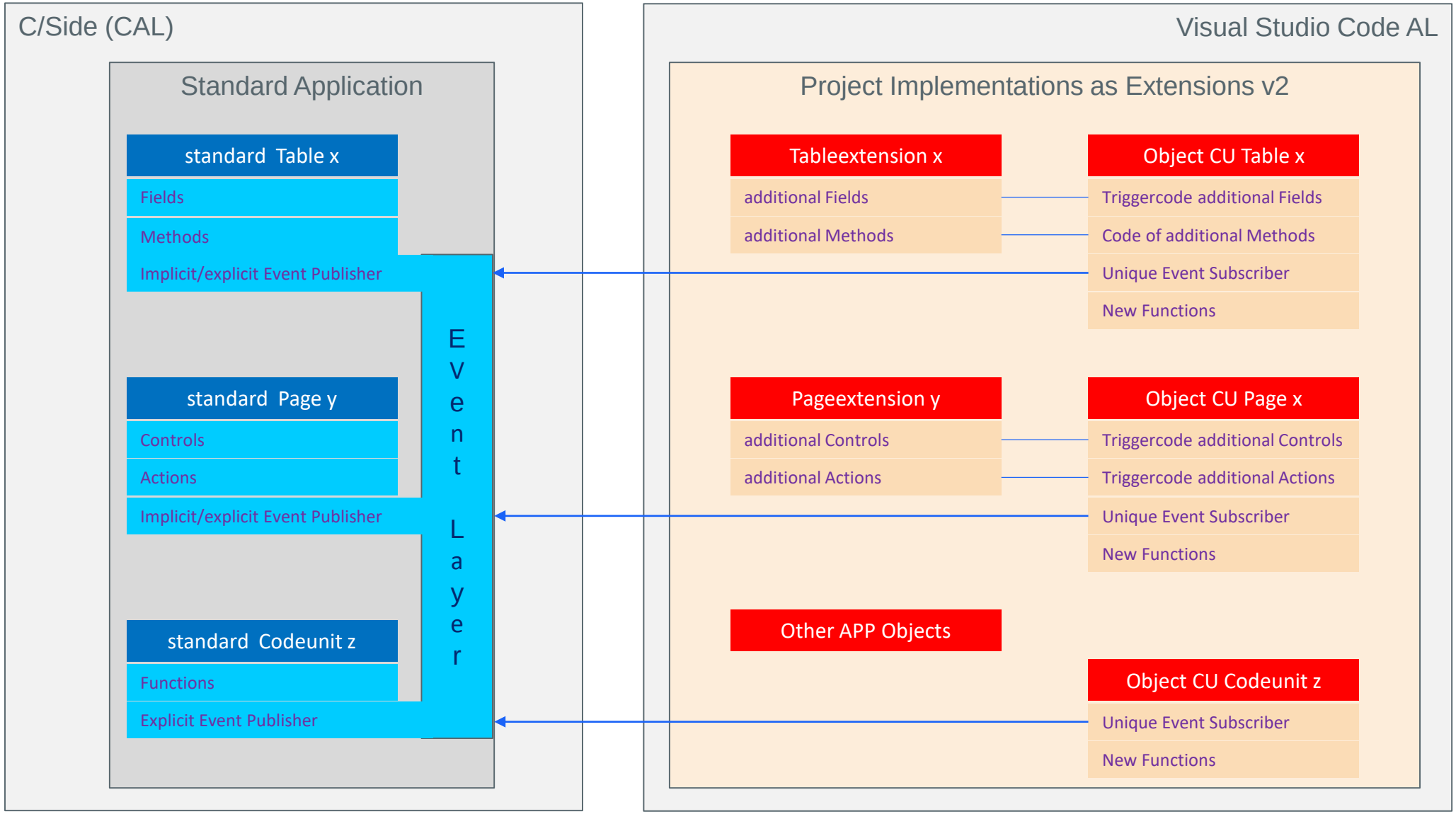
Basics
Object Codeunit

- Remarks
  - Collecting all ToDos in a single Event Handler instead of multiple Event Subscriptions provides a

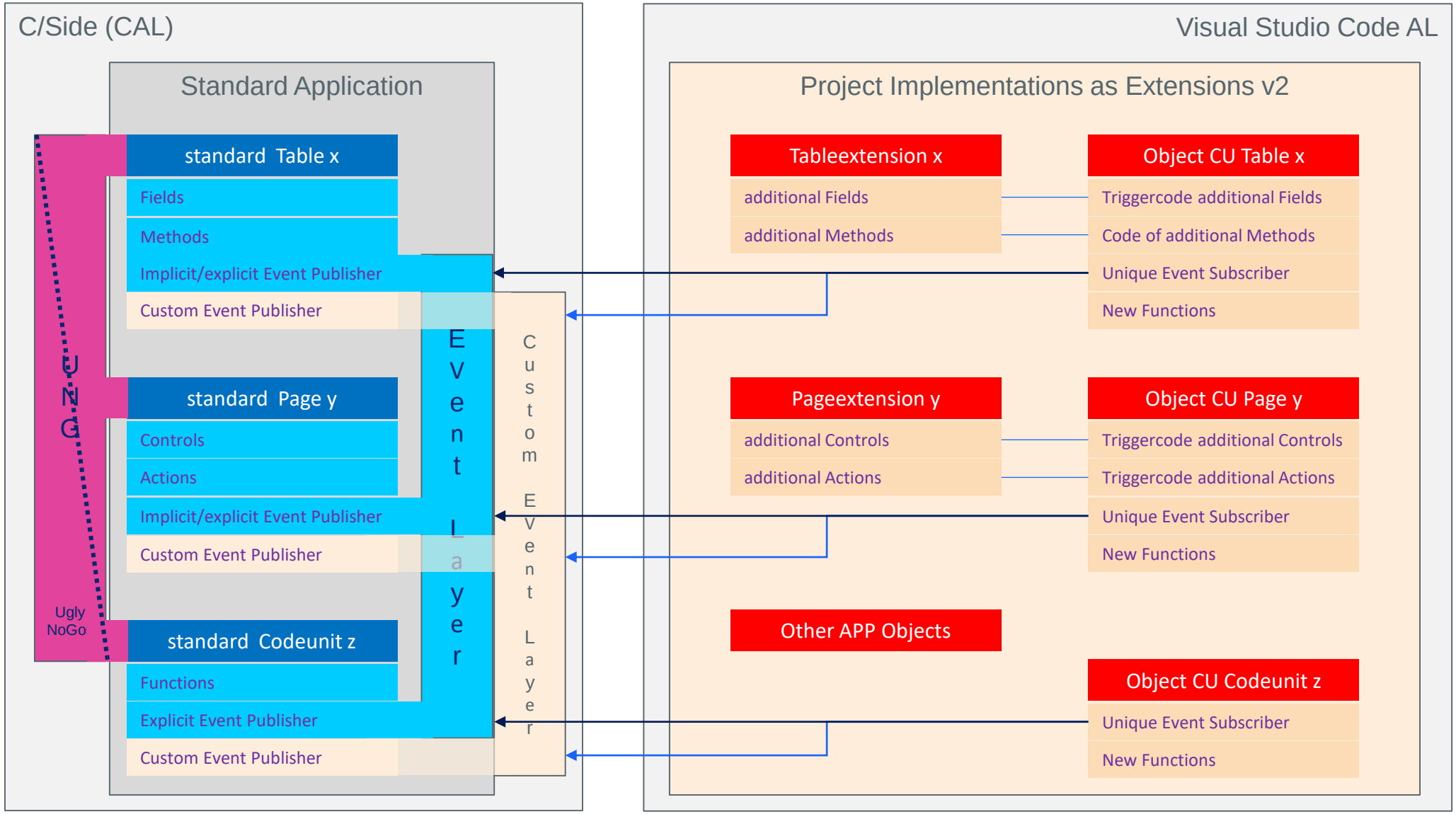
> Simple solution to the problem of undefined execution-order of subscriptions:
>
> **each event is subscribed once only**

  - Integration of manual subscription might be done via „Event Switching Codeunits“

Basics: Cloud Ready Object Structure - bare Extension V2



Basics: Hybrid Object Structure – OnPremise only



Basics

AL = File based programming ➔ File Structure

- Considerations on the file structure will be made later on

Boundary Conditions / Setups
HowTo handle a Project

- AL Development against BC – „Architecture“
- Create / Setup AL Projects - Overview
- Specifying BC Target – launch.json
- Specifying Extensions(APP) – app.json
- Providing Target (APP) Information - Symbols
- Some Remarks
- HandsOn: „Hello World“
- (De-) Installing the app

Boundary Conditions / Setups: HowTo handle a Project

AL Development against BC – „Architecture“ → the Past

Microsoft Dynamics NAV

- SQL Database
- Development Environment
- NAV Services
- NAV Clients
- „All“ Code within Database
- …

Boundary Conditions / Setups: HowTo handle a Project

AL Development against BC – „Architecture“ → the Future

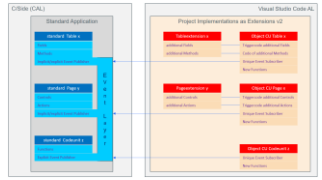
Microsoft Dynamics 365 Business Central

- SQL Database
- „Classical“ Development Environment
- BC Services
- BC Clients
- „Standard“ Code within Database
- …

Seamless coupling

APPs / Extensions v2

- Visual Studio Code
- AL Language Extension
- Code stored in File System
  - .al Files
  - Source Code Management (GIT)
  - Distributed Development
  - …
- Modern Compiler
- VS Code Extensions
- Snippets (Macros)
- …



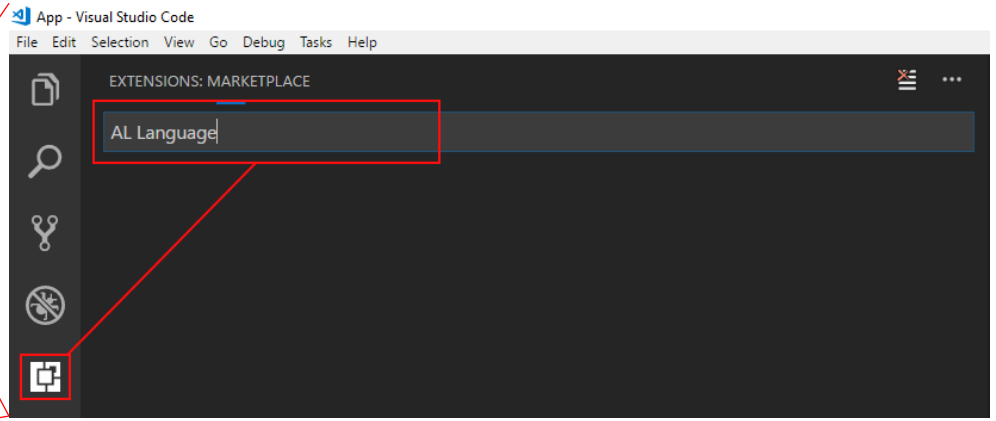
Boundary Conditions / Setups
HowTo handle a Project

- AL Development against BC – „Architecture“
- Create / Setup AL Projects - Overview
- Specifying BC Target – launch.json
- Specifying Extensions(APP) – app.json
- Providing Target (APP) Information - Symbols
- Some Remarks
- HandsOn: „Hello World“
- (De-) Installing the app

Boundary Conditions / Setups: HowTo handle a Project

Create / Setup AL Projects – Overview → Installation & Setup of VS Code AL-Plugin

- Download & Install Visual Studio Code
  - https://code.visualstudio.com/Download
- Download & Install the AL Extension for Visual Studio Code
  - Open Visual Studio Code
  - Type „AL Language“ in the search box of the „Extensions“ section
  - Hit „Install“
  - Hit „Reload“ to relaunch VS Code
- Eventually change Telemetry Setting
- Eventually download and install „AL Code Outline“
  - AL code outline tree provider
  - Browse through meta data of standard objects
  - Autocreate GUI objects, … based on selected table
- Eventuall download and install CRS AL Language Extension



How to disable telemetry reporting

VS Code collects usage data and sends it to Microsoft to help improve our products and services. Read our privacy statement to learn more.

If you don't wish to send usage data to Microsoft, you can set the `telemetry.enableTelemetry` setting to `false`.

From **File** > **Preferences** > **Settings** (macOS: **Code** > **Preferences** > **Settings**), search for `telemetry.enableTelemetry` and uncheck the setting. This will silence all telemetry events from VS Code going forward. Telemetry information may have been collected and sent up until the point when you disable the setting.

If you use the JSON editor for your settings, add the following line:

```
"telemetry.enableTelemetry": false
```

You can inspect telemetry events in the Output panel by setting the log level to **Trace** using **Developer: Set Log Level** from the Command Palette.

> **Important Notice**: VS Code gives you the option to install Microsoft and third party extensions. These extensions may be collecting their own usage data and are not controlled by the `telemetry.enableTelemetry` setting. Consult the specific extension's documentation to learn about its telemetry reporting.

Boundary Conditions / Setups: HowTo handle a Project

Create / Setup AL Projects – Overview → Important Shortcuts

- https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-keyboard-shortcuts

General in Visual Studio Code

Keyboard Shortcut	Action
(F1) Ctrl+Shift+P	Show All Commands
Ctrl+F7	Download source code
Alt+A Alt+L	Go! Generates a HelloWorld project
Ctrl+Shift+B	Package
F5	Publish
Ctrl+F5	Publish without debugging
F6	Publish and open the designer
Ctrl+F2	Update the compiler used by the service tier(s)

Compile in Visual Studio Code

Keyboard Shortcut	Action
Ctrl+Shift+B	Compile and build the solution
Ctrl+F5	Build and deploy

Debugging in Visual Studio Code

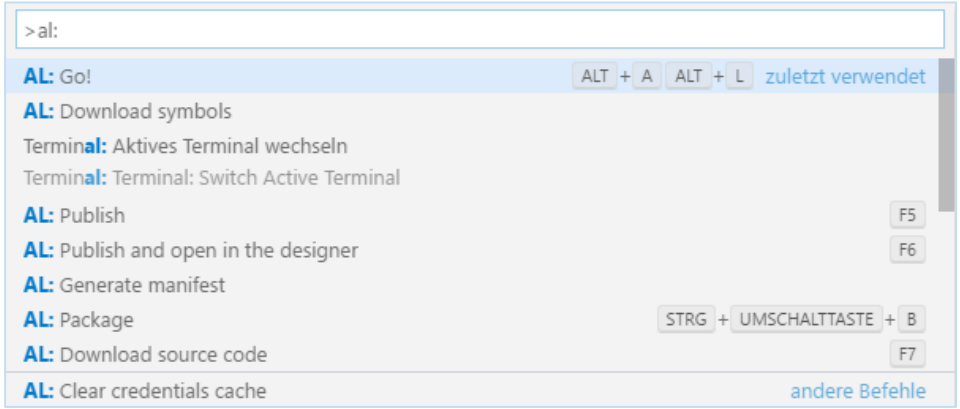
Keyboard Shortcut	Action
F5	Start debugging session

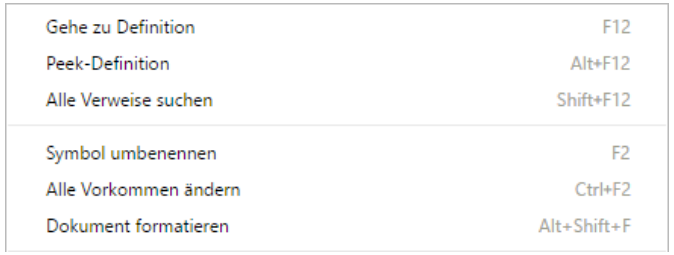
Editing in Visual Studio Code

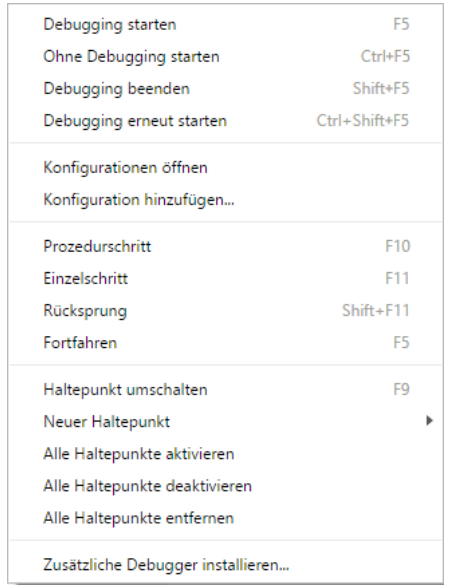
Keyboard Shortcut	Action
Ctrl+Space	Look up suggestions for the current object
Ctrl+X	Cut
Ctrl+C	Copy
Ctrl+V	Paste
Ctrl+F2	Select all occurrences
F12	Go to definition
Alt+F12	Peek definition
Shift+F12	Show References
Ctrl+Shift+Space	Look up parameter hints
Ctrl+K Ctrl+C	Add line comment
Ctrl+K Ctrl+U	Remove line comment
Ctrl+Shift+P	Show All Commands

Boundary Conditions / Setups: HowTo handle a Project
another list of Important Commands

- VS Code Commands [Ctrl + Shift + P] or [F1]
  - AL: Go! [Alt + A] & [Alt + L]
  - AL: Download Symbols
  - AL: Package
  - AL: Download Source Code
  - AL: Publish
  - Keyboard Shortcuts [Ctrl + K] & [Ctrl + S]
- Editor Commands
  - [F2] … **Rename** Object / Property / …
  - [F12] … Goto **Definition**
  - [Ctrl + F12] … Goto **Implementation**
  - [Shift + F12] … Show **Where** the reference is **Used**
  - [Ctrl + B] … Compile / Build
  - [F5] … Publish and Start
  - [Ctrl + F5] … Publish without Debug







Boundary Conditions / Setups: HowTo handle a Project

Create / Setup AL Projects – Overview → „Project Setup“

Load Template

Create „Hello World“
- *Open VS Code*
- *Type F1 or Ctrl+Shift+P to open AL Command Line*
- *Type a fragment of „AL: Go!“*
- *Choose „AL: Go!“* *Oder: „Alt+A, Alt+L“*
- *Enter (AL) Project Path (…\...\HelloWorld)*
- *Choose Environment (Cloud / OnPremise)*

Configure Environment

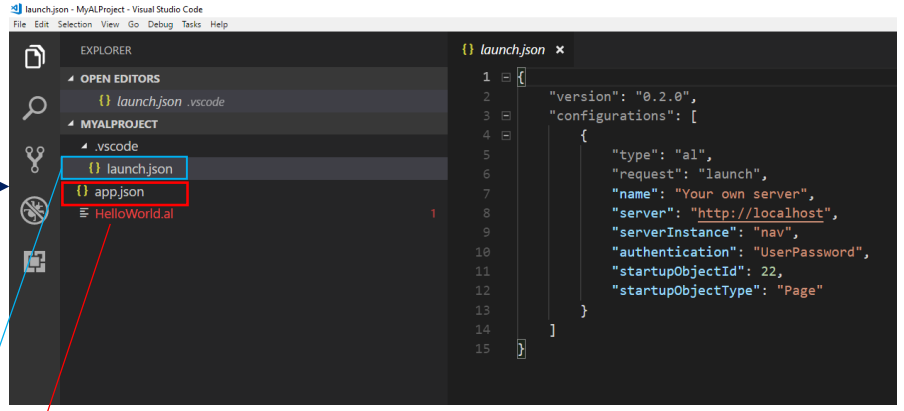
Characterize your environment via proper values in the launch.json file

Define the APP

Characterize your APP via proper values in the app.json file

Introduce the Target DB‘s object structure to your APP

Download Symbols
- *Type F1 or Ctrl+Shift+P to open AL Command Line*
- *Type a fragment of “Download Symbols”*
- *Choose “Download Symbols”*



Boundary Conditions / Setups
HowTo handle a Project

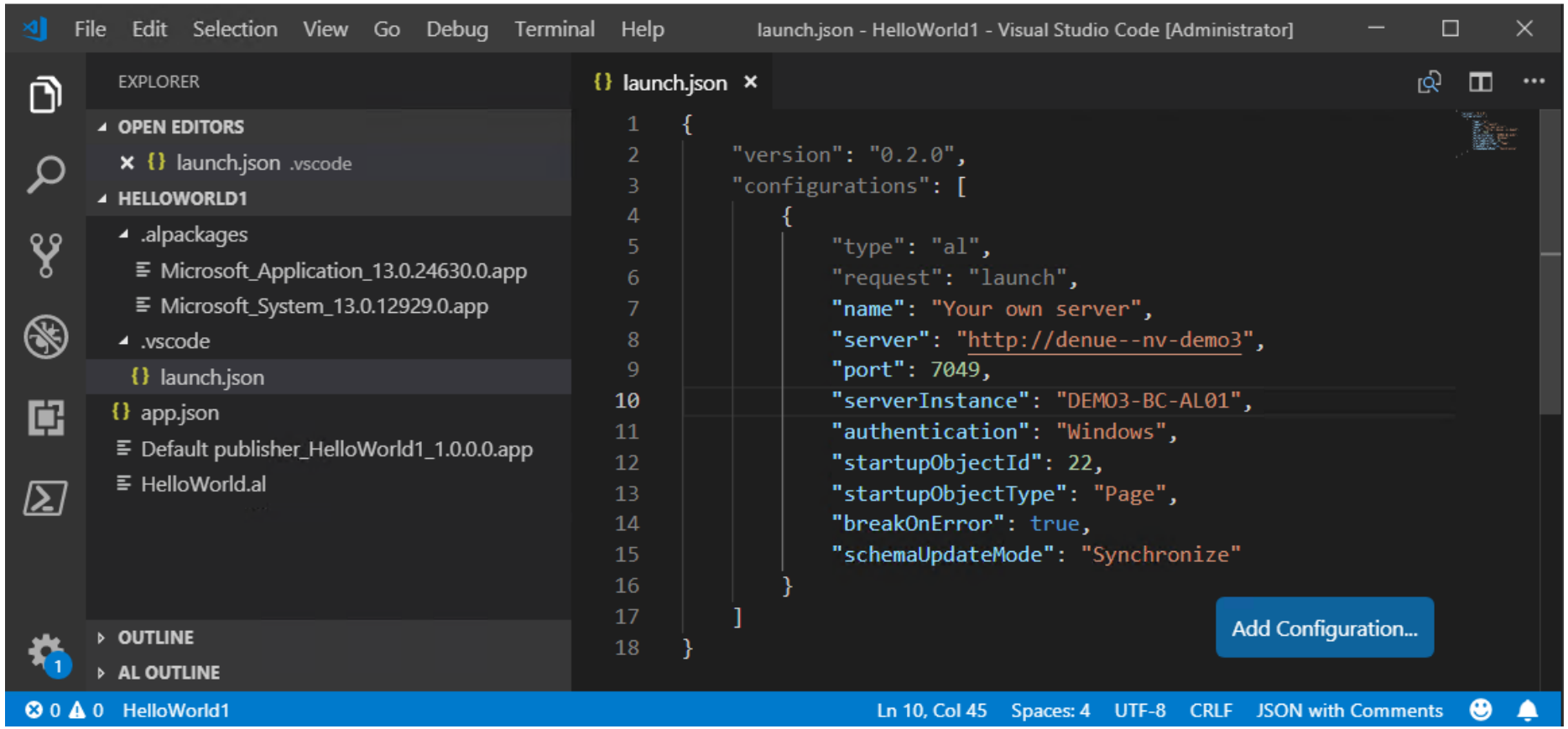
- AL Development against BC – „Architecture“
- Create / Setup AL Projects - Overview
- Specifying BC Target – launch.json
- Specifying Extensions(APP) – app.json
- Providing Target (APP) Information - Symbols
- Some Remarks
- HandsOn: „Hello World“
- (De-) Installing the app

Boundary Conditions / Setups: HowTo handle a Project

Specifying BC Target

→ launch.json

- Defines the „Target System“ and how to start it
- Located in „MyProject“/.vscode/launch.json
  https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-json-files#Launchjson

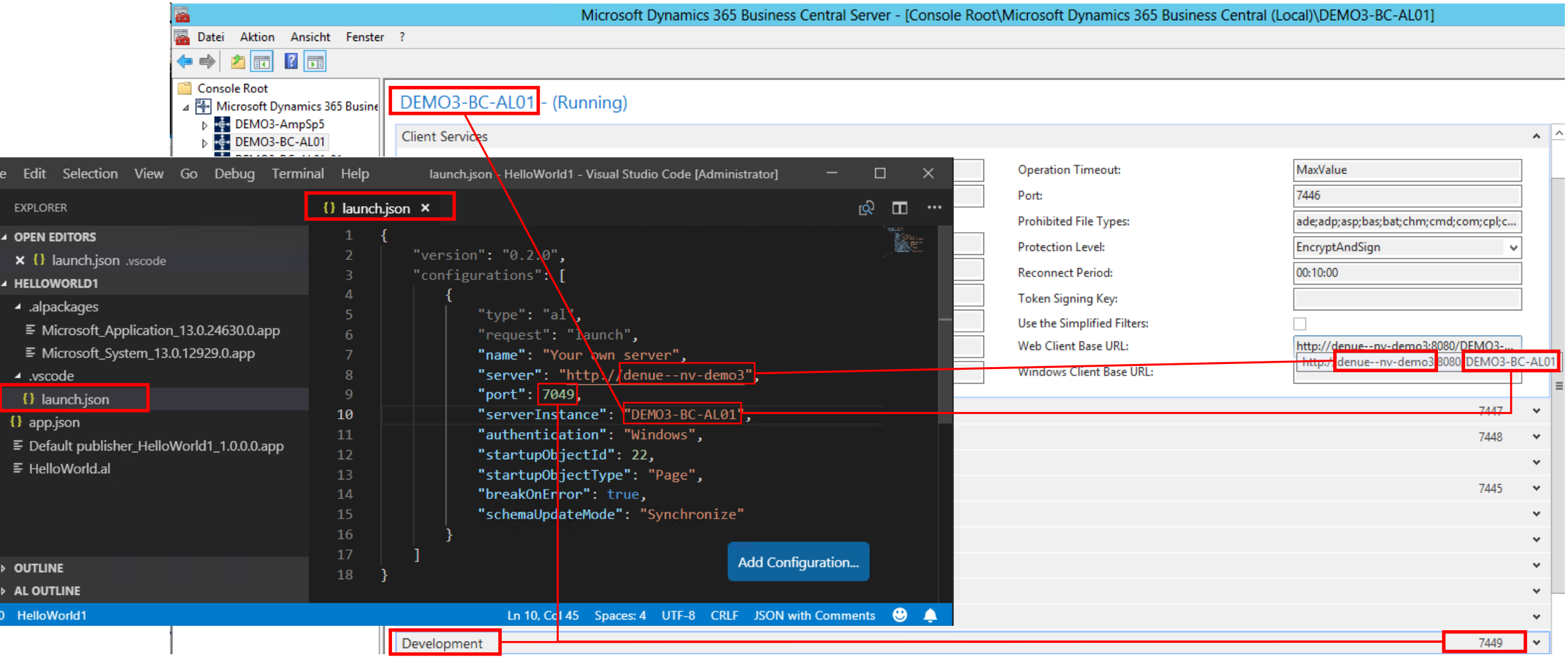


Boundary Conditions / Setups: HowTo handle a Project

Specifying BC Target

→ launch.json

- more detailed



Boundary Conditions / Setups: HowTo handle a Project

Specifying BC Target

→ launch.json

Important Parameter

Parameter Name	Description	Remark
„name“	Name of the „launcher“	Arbitrary but mandatory
„type“	„.al“	Mandatory
„request“	„launch“	Mandatory
„server“	NAV-Server HTTP-URL	
„serverInstance“	NAV-Server Instance-Name	
“port“	Development port	If „server“ not configured as „[URL]:[port]“
„authentication“	NAV-Authentication	Possible Options - AAD: Active Directory - UserPassword: DB - Windows
„startupObjectType“	NAV Object-Type to run / start	- Either „Page“ or „Table“ (ReadOnly)
„startupObjectID“	NAV Object-ID of Object to be started	

Boundary Conditions / Setups: HowTo handle a Project

Specifying BC Target

→ launch.json

Important Parameter

Parameter Name	Description	Remark
„schemaUpdateMode“	Data / schema synchronization mode	Either „Synchronize“ - Tries synchronizing table fields. Fails if data could be lost. - → „No Destructive Changes“ or „Recreate“ - Deletes and creates the tables afterwards. - All data will be lost → Upgrade-Code needed! C.f. 1
breakOnError	Specifies whether to break on errors when debugging. The default value is true	
breakOnRecordWrite	Specifies if the debugger breaks on record changes. The default value is false	

1: https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-retaining-data-after-publishing

Boundary Conditions / Setups
HowTo handle a Project

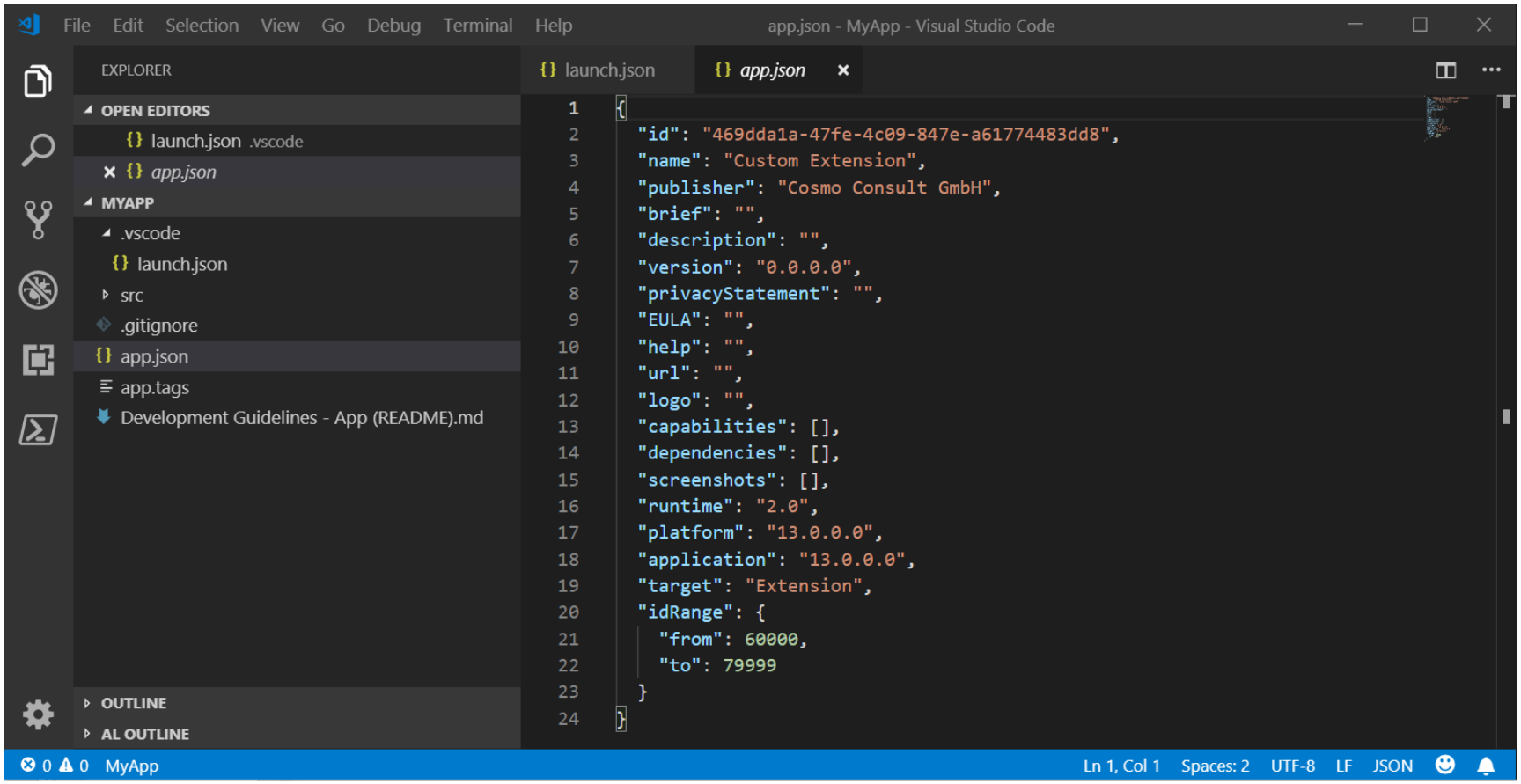
- AL Development against BC – „Architecture“
- Create / Setup AL Projects - Overview
- Specifying BC Target – launch.json
- Specifying Extensions(APP) – app.json
- Providing Target (APP) Information - Symbols
- Some Remarks
- HandsOn: „Hello World“
- (De-) Installing the app

Boundary Conditions / Setups: HowTo handle a Project

Specifying Extensions (APP)

→ app.json

- Defines the „APP“ to be created via the AL Project
- Located in „MyProject“/app.json
  https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-json-files#appjson-file



Boundary Conditions / Setups: HowTo handle a Project

Specifying Extensions (APP)

→ app.json

Important Parameters I

Parameter Name	Description	Remark
„id“	Unique ID of the APP (Extension)	- Generated automatically while project is created - Mandatory
„name“	Unique name of the APP (Extension)	- Might be the project name, the project number or the name of the customer in the case of a customer project. - Mandatory
„publisher“	Name of the publisher of the APP (Extension)	- Might be „Cosmo Consult“? - Mandatory
„version“	Version of the APP (Extension)	Mandatory
„platform“	Minimum NAV-Version for platform-symbols (system tables)	Mandatory for using system tables
„application“	Minimum NAV-Version for application-symbols (application objects)	Mandatory for using application objects
„idRange(s)“	Restriction of object Ids used by the APP (Extension)	- Example: "idRanges": [{"from": 50100,"to": 50200},{"from": 50202,"to": 50300}]. - Error occurs if object is created outside this range(s) - ID is automatically suggested - Use either „idRange“ or „idRanges“!
„showMyCode“	Enables debugging of the source code of the extension	Possible „options“ - true - false (default)

Boundary Conditions / Setups: HowTo handle a Project

Specifying Extensions (APP)

→ app.json

Important Parameters II

Parameter Name	Description	Remark

„target“	Defines the type of APP (Extension)	Possible values
		- Level 0: „Personalization“
		  - only allows for inclusion of UI objects like Profiles/PageCustomizations
		- Level 1: „Extension“ (Default)
		  - Only global C/Side Functions with „FunctionVisibility“=„External“ are accessible
		- Level 2: Solution
		  - Unused at the moment, this (or something very like it) will be used to encompass the capabilities of 'Embed' apps in the future
		- Level 3: „Internal“
		  - All global C/Side functions accessible
		  - NAV-Server setting „ExtensionAllowedTargetLevel“ must be „internal“, too
		NAV Server setting „ExtensionAllowedTargetLevel“ should probably be as high as the desired APP level. Otherwise, the APP can not be installed (c.f.1)

„brief“	Short description of the extension	Mandatory for App Source submission
„description“	Longer description of the extension	Mandatory for App Source submission
„…….“	Other App Source relevant properties	
„dependencies“	List of dependencies to other apps	
„logo“	Relative path to the app package logo from the root of the package.	Mandatory for App Source submission nice otherwise

1. https://github.com/Microsoft/AL/issues/734

Boundary Conditions / Setups: HowTo handle a Project

Specifying Extensions (APP)

→ app.json

- Important Parameters III

Parameter Name	Description	Remark
„helpBaseUrl“		The URL for the website that displays help for the current extension. The default URL is https://docs.microsoft.com/{0}/dynamics365/business-central.
„supportedLocales“		The list of locales that are supported for looking up help. The value on the list is inserted into the URL defined in the helpBaseUrl property. The first locale on the list is default. An example is "supportedLocales": ["da-DK", "en-US"].

1: https://github.com/Microsoft/AL/issues/734

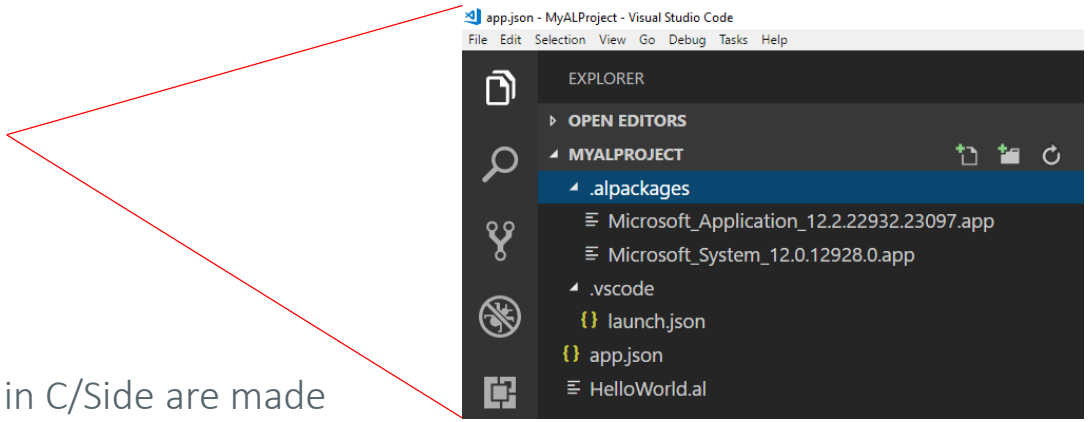
Boundary Conditions / Setups
HowTo handle a Project

- AL Development against BC – „Architecture“
- Create / Setup AL Projects - Overview
- Specifying BC Target – launch.json
- Specifying Extensions(APP) – app.json
- Providing Target (APP) Information - Symbols
- Some Remarks
- HandsOn: „Hello World“
- (De-) Installing the app

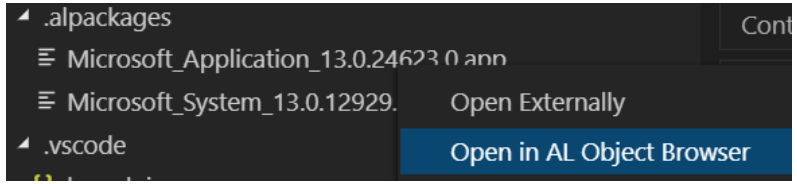
Boundary Conditions / Setups: HowTo handle a Project

Providing Target (APP) Information → symbols

- Metadata of the objects (Platform & Application) with references to fields, functions, etc. are supplied „via symbols“
- No symbols → no Extension programming
- Symbol download from target server yields package files in the „./.alpackages“ folder of the project



- Hybrid Environments
  - Symbols have to be generated and downloaded „whenever“ changes in C/Side are made
  - Generate symbols continuously: use „generatesymbolreference“ option of finsql.exe (c.f. https://docs.microsoft.com/en-us/dynamics-nav/developer/devenv-running-cside-and-al-side-by-side)



- Symbol download successful but „no symbols present“
  - Check symbol presence e.g. via object browser
  - If standard symbols missing, run
    - „<PathToFinSql>\finsql.exe" Command=generatesymbolreference, Database=<DataBase>, ServerName=<NAVServerInstance>

    to (re-)generate symbols – might be a lengthy operation

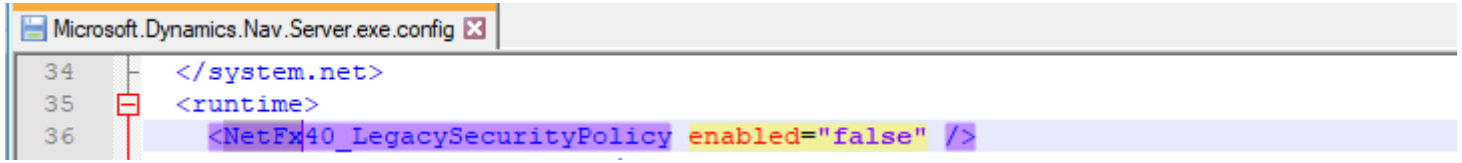
Boundary Conditions / Setups
HowTo handle a Project

- AL Development against BC – „Architecture“
- Create / Setup AL Projects - Overview
- Specifying BC Target – launch.json
- Specifying Extensions(APP) – app.json
- Providing Target (APP) Information - Symbols
- Some Remarks
- HandsOn: „Hello World“
- (De-) Installing the app

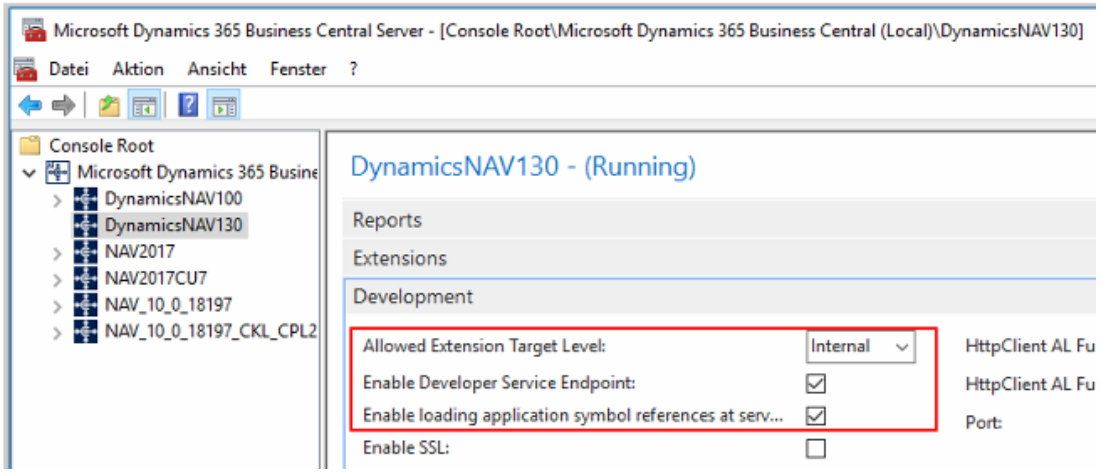
Boundary Conditions / Setups: HowTo handle a Project
Some Remarks

- Always run VS Code „as Admin“
- „Typical errors“, occuring while symbol downloading
  - [2018-11-06 16:39:58.11] The request for path /DynamicsNAV130/dev/metadata failed with code InternalServerError. Reason: NetFx40_LegacySecurityPolicy is enabled and must be turned off in the Microsoft.Dynamics.Nav.Server.exe.config file.

    Solution: change “true” to “false” in NetFx40… key of config file

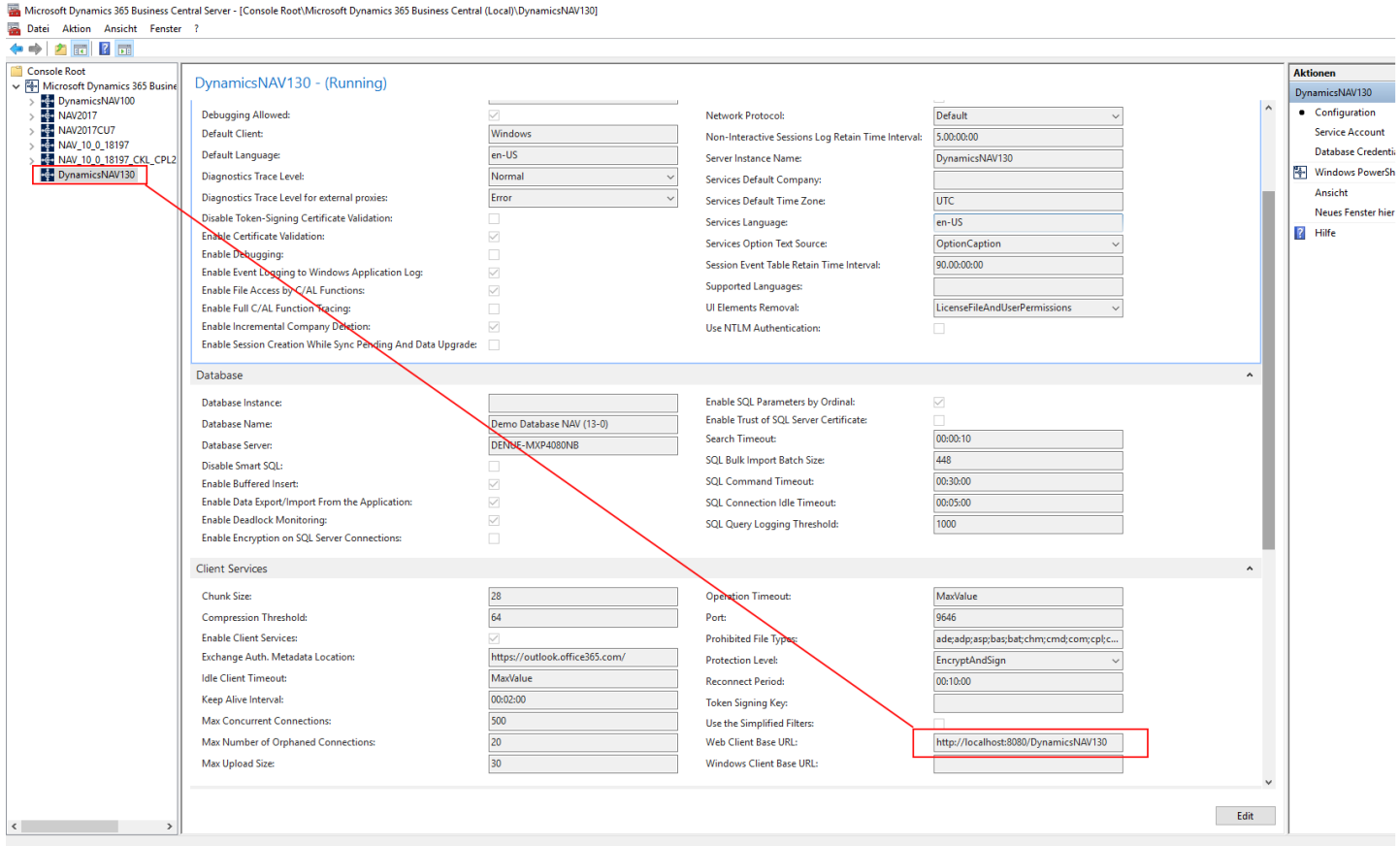


  - [2018-11-07 10:51:10.93] Error: Could not open the specified startup page. Please check that the server configuration key



Boundary Conditions / Setups: HowTo handle a Project
Some Remarks

- "Typical errors", occurring while publishing "Hello World"
  - VS Code not executed „as administrator"
  - [2018-11-07 10:51:10.93] Error: Could not open the specified startup page. Please check that the server configuration key PublicWebBaseUrl has been properly set. ← Define Setting and Restart VS Code



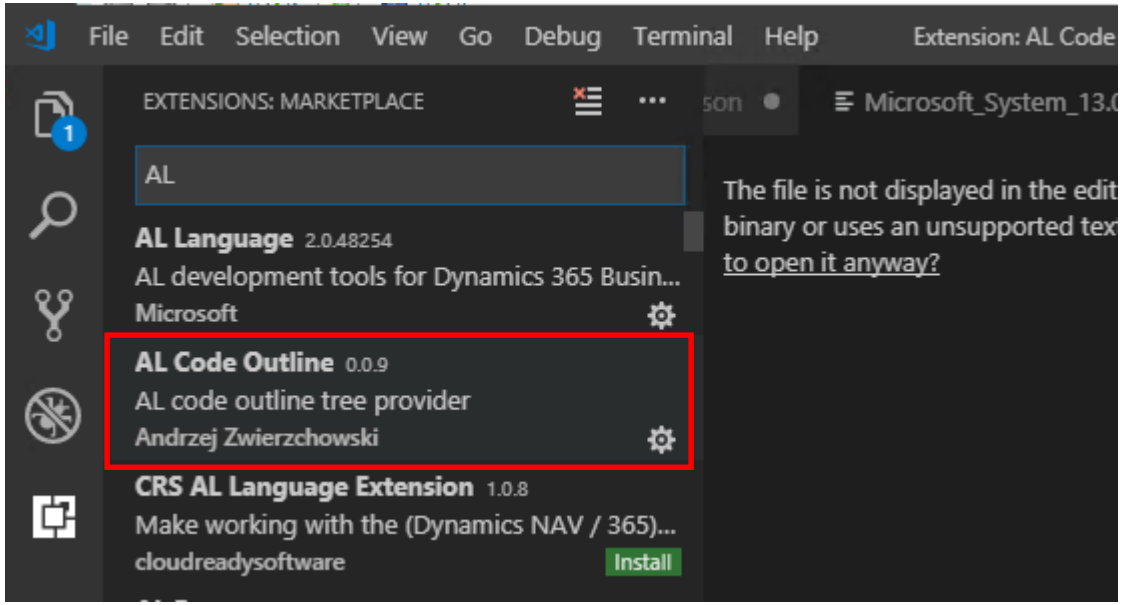
If things don't work as expected:
Restart VS Code
(e.g., if you've changed the „Web Client Base URL"

Boundary Conditions / Setups: HowTo handle a Project
Some Remarks

- “You cannot connect to the Classic Development Environment (Cside)”
- Perhaps, prerequisites are missing,
  Whether you are running CSide using Click-Once, through the navcontainerhelper or you manually copied the binaries to a shared folder, you need to have the pre-requisites installed on the computer on which you are running CSide.
  If CSide complains about missing DLL's or giving an ODBC error, you probably need to install these pre-requisites:
  - https://download.microsoft.com/download/2/E/6/2E61CFA4-993B-4DD4-91DA-3737CD5CD6E3/vcredist_x86.exe
  - https://download.microsoft.com/download/3/A/6/3A632674-A016-4E31-A675-94BE390EA739/ENU/x64/sqlncli.msi

Boundary Conditions / Setups: HowTo handle a Project
Some Remarks

- "For later purposes – to get insights into "app-packages" : Install "AL Code Outline"



Boundary Conditions / Setups
HowTo handle a Project

- AL Development against BC – „Architecture“
- Create / Setup AL Projects - Overview
- Specifying BC Target – launch.json
- Specifying Extensions(APP) – app.json
- Providing Target (APP) Information - Symbols
- Some Remarks
- HandsOn: „Hello World“
- (De-) Installing the app

HandsOn – Exercise 1 | Hello World: our very first page extension

HandsOn

Understand the Example Extension „Hello World“

Configure your Environment / Run the Example

Load Template

Create „Hello World“
- *Open VS Code*
- *Type F1 or Ctrl+Shift+P to open AL Command Line*
- *Type a fragment of „AL: Go!“*
- *Choose „AL: Go!“* *Oder: „Alt+A, Alt+L“*
- *Enter (AL) Project Path*
- *Choose Environment (Cloud / OnPremise)*

Configure Environment

Characterize your environment via proper values in the launch.json file

Define the APP

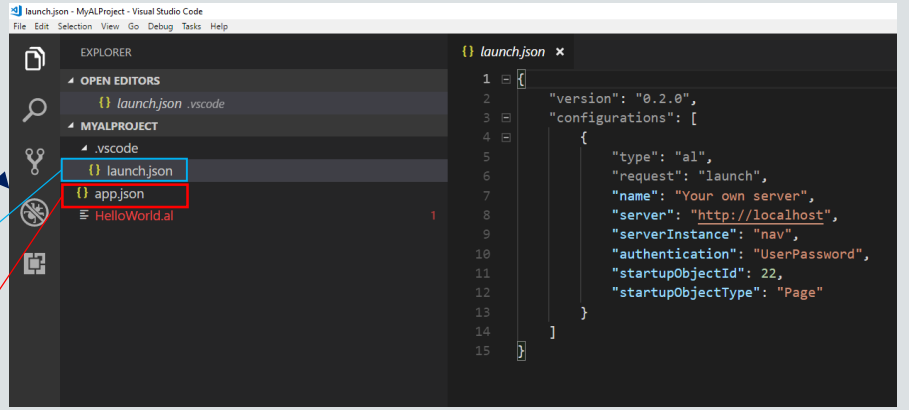
Characterize your APP via proper values in the app.json file

Introduce the Target DB‘s object structure to your APP

Download Symbols
- *Type F1 or Ctrl+Shift+P to open AL Command Line*
- *Type a fragment of “Download Symbols”*
- *Choose “Download Symbols”*

Compile / Publish / Install / Launch the APP

Hit „F5“ in VS Code ….



If things don‘t work as expected:
Restart VS Code
(e.g., if you‘ve changed the „Web Client Base URL“

Boundary Conditions / Setups
HowTo handle a Project

- AL Development against BC – „Architecture“
- Create / Setup AL Projects - Overview
- Specifying BC Target – launch.json
- Specifying Extensions(APP) – app.json
- Providing Target (APP) Information - Symbols
- Some Remarks
- HandsOn: „Hello World“
- (De-) Installing the app

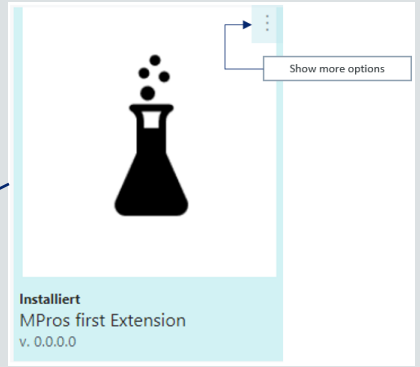
HandsOn – Exercise 2 | Deinstalling the app (via Client)

HandsOn

Open Client

Search „Extension Management“

Press “Show more options”
and choose “Uninstall”



Extension is now „Uninstalled“
but not „Unpublished“

Press “Show more options”
and choose “Unpublish”

Extension is now
„no longer available on the system“

!! data remains !!

Learn - Why this 2 step procedure?
To be able to have an extension available on dedicated tenants only

→ publish into a data base
→ Install into a tenant

Boundary Conditions / Setups: HowTo handle a Project
Installing the app via Client

- On Premise, at the moment, there is no way of installing an app from within client
- Installing via Powershell: wait for later

Programming in AL

- Preliminaries (Intellisense / Txt2AL Converter / …)
- HandsOn … My first Page Extension – CaptionML
- App Organization
- Object Types: Theory and HandsOn

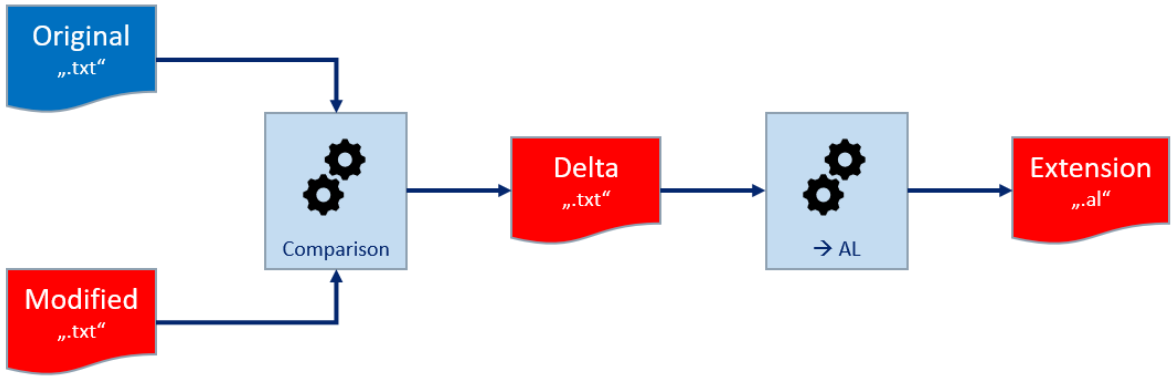
Programming in AL
Preliminaries (Intellisense / Txt2AL Converter / CodeCops / Snippets)

- Context Intellisense
  - shows what is possible in the current context of the cursor.
    - Possible Properties
    - Possible Property-Values
    - Next possible Object-/Field-ID
    - Eventing: Parametervorschlag auf Basis der Publisher Eigenschaften
    - …
  - Shortcut: Ctrl+Space

If you don‘t know, what to do - 1: Press „Ctrl+Space“

Programming in AL
Preliminaries (Intellisense / Txt2AL Converter / CodeCops / Snippets)

- Txt2AL Converter
  - converts Deltas into AL



  - Missing „Original“ → whole „Modified“ is transformed
    - Thus: standard objects can be transformed into AL
    - Allows to see how whole C/AL constructs transform into AL

If you don‘t know, what to do - 2: Use Txt2AL Converter

Programming in AL
Preliminaries (Intellisense / Txt2AL Converter / CodeCops / Snippets)

- Txt2AL Converter consists of
  - CAL2Extension … .ps1
    - Converts Object [Type, ID] into al-file
    - Stores result in „$WorkingDir + \CAL2EXT“ and/or subdirs
    - Needs
      - „$DatabaseServer“: SQL Server of DB
      - „$DatabaseName“
      - „$PathToRTCDir“
      - „$PathToText2ALExe“ (shoud be „$PathToRTCDir + \txt2al.exe“)
    - Optional: „$OriginalsSourceDir“
      - If given, script uses files to compare to, thus generates Extension by transforming deltas
      - If not given: converts objects into al as a whole …
  - BulkCAL2Extension … .ps1
    - Holds list of objects to be transformed
    - Transforms objects one by one using „CAL2Extension … .ps1“
  - At the moment, the scripts which wrap the converter are „crusher bar implementations“ which are in urgent need of improvement

HandsOn – Exercise 3 | Configuring the Txt2AL Converter

HandsOn

- Txt2AL Converter - HandsOn: Convert page „Units of Measure (209)“
  - Configure Converter

CAL2Extension0.99.05.ps1 | BulkCAL2Extension0.99.01.ps1

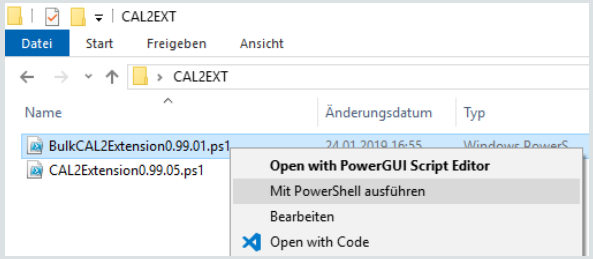
```
$TabIDs = @()
$PagIDs = @(209)
$CuIDs = @()
```

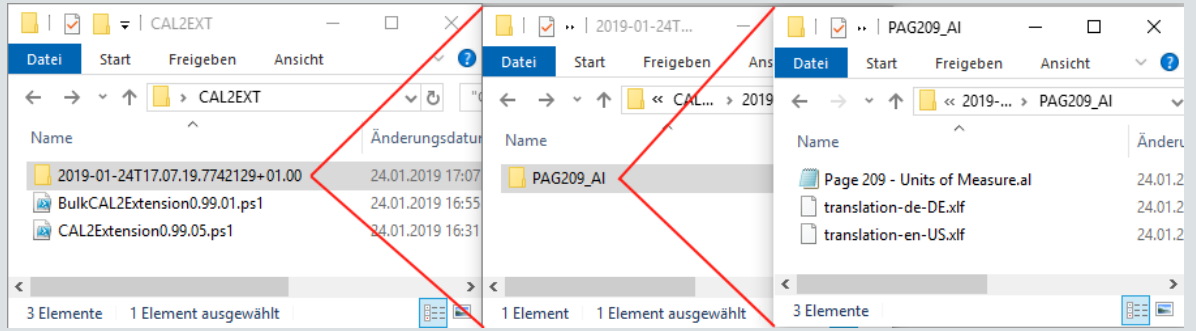
SQL-Server

CAL2Extension0.99.05.ps1 | BulkCAL2Extension0.99.01.ps1

```
# NAV/ BC Installation Specification
$PathToRTCDir                = 'C:\Program Files (x86)\Microsoft Dynamics 365 Business Central\130\RoleTailored Client'
$PathToText2ALExe            = "$PathToRTCDir\txt2al.exe"
# $DatabaseServer              = 'DENUE--NV-SQL05'
$DatabaseServer              = 'DENUE-MXP4080NB'
$DatabaseName                = 'Demo Database NAV (13-0)'
```

  - Convert page „Units of Measure (209)“

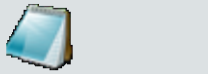




HandsOn – Exercise 3 Continued | Configuring the Txt2AL Converter

HandsOn

Result



Page 209 - Units of Measure.al

Page 209 - Units of Measure.al - Editor

Datei Bearbeiten Format Ansicht Hilfe

```
page 209 "Units of Measure"
{
    // version NAVW113.00

    ApplicationArea = Basic,Suite;
    Caption = 'Units of Measure';
    PageType = List;
    SourceTable = "Unit of Measure";
    UsageCategory = Administration;

    layout
    {
        area(content)
        {
            repeater(Control1)
            {
                ShowCaption = false;
                field("Code";Code)
                {
                    ApplicationArea = Basic,Suite,Invoicing;
                    ToolTip = 'Specifies a code for the unit of measure, which you can select
                }
                field(Description;Description)
                {
                    ApplicationArea = Basic,Suite,Invoicing;
                    ToolTip = 'Specifies a description of the unit of measure.';
                }
                field("International Standard Code";"International Standard Code")
                {
                    ApplicationArea = Basic,Suite,Invoicing;
                    ToolTip = 'Specifies the unit of measure code expressed according to the UI
                }
            }
        }
        area(factboxes)
        {
            systempart(Control1900383207;Links)
            {
                Visible = false;
            }
            systempart(Control1905767507;Notes)
            {
```

Programming in AL
Preliminaries (Intellisense / Txt2AL Converter / CodeCops / Snippets)

- Just to mention them at this point, details will follow below
  - Code Analysis / CodeCops
    - functionality to have the code checked automatically on different subtlety levels
  - Snippets: Macros within Visual Studio Code
    - Macro recorde existing (Visual Studio Code Extension)
    - Basic snippets predefined:
    - type „t<snippet name>“ to activate AL snippet

If you don‘t know, what to do - 3: Press „t<context>“

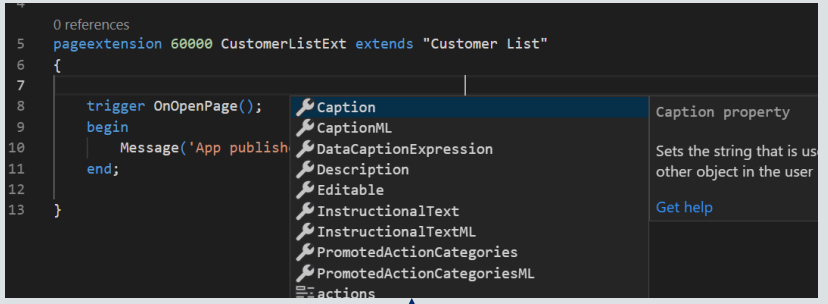
HandsOn – Exercise 4 | intelliSense

HandsOn

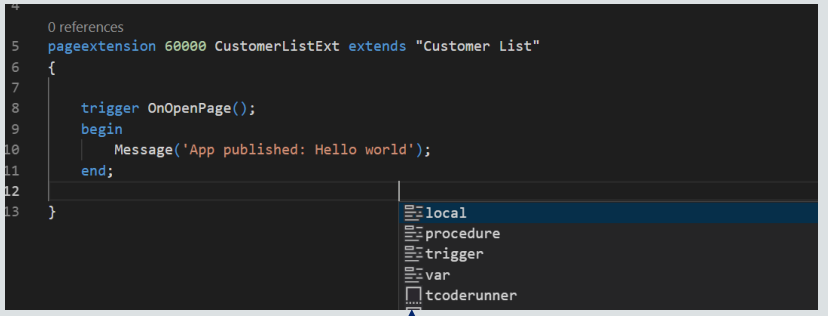
Open VS Code on „HelloWorld“ Project

Open file „HelloWorld.al“

Place the cursor above “trigger OnOpenPage()” and press “Ctrl+Space” to activate Intellisense



Place the cursor below “trigger OnOpenPage()” block and press “Ctrl+Space” again



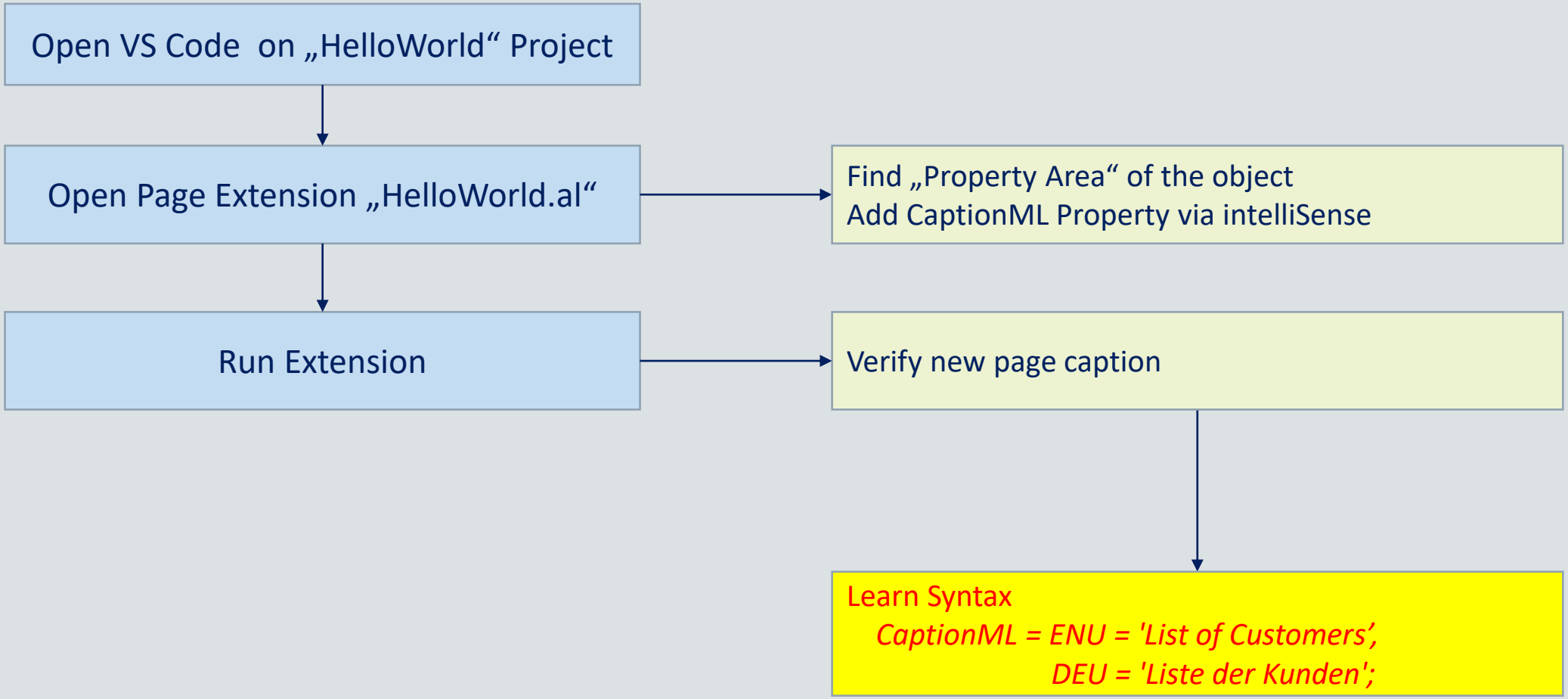
Learn that
1) the suggestions depend on the actual context of your code – it’s “Intelli”
2) Elements of an object have to be placed in a certain order …. .

Programming in AL

- Preliminaries (Intellisense / Txt2AL Converter / …)
- HandsOn … My first Page Extension – CaptionML
- App Organization
- Object Types: Theory and HandsOn

HandsOn – Exercise 5 | CaptionML

HandsOn



Programming in AL

- Preliminaries (Intellisense / Txt2AL Converter / …)
- HandsOn … My first Page Extension – CaptionML
- App Organization
- Object Types: Theory and HandsOn

Programming in AL
APP Organization - Unique Namespace: Prefixing

- Forced by the possibility of changing environment: suitable Prefixing to create uniqueness
  - (separated by a „Space“ from the object‘s name)
- Use the Prefix *CCO* (Cosmo Consult Operations) for the following:
  - Objects
    - Name
  - Object Codeunits
    - Name
  - Object Extensions
    - Name
    - Global Procedures
    - Fields (Table)
    - Keys / KeyNames
    - Controls (Page)
    - Actions (Page)
    - Values (Enum)
- Prefix „CCOT“ for Test Objects

Programming in AL
APP Organization - Philosophy

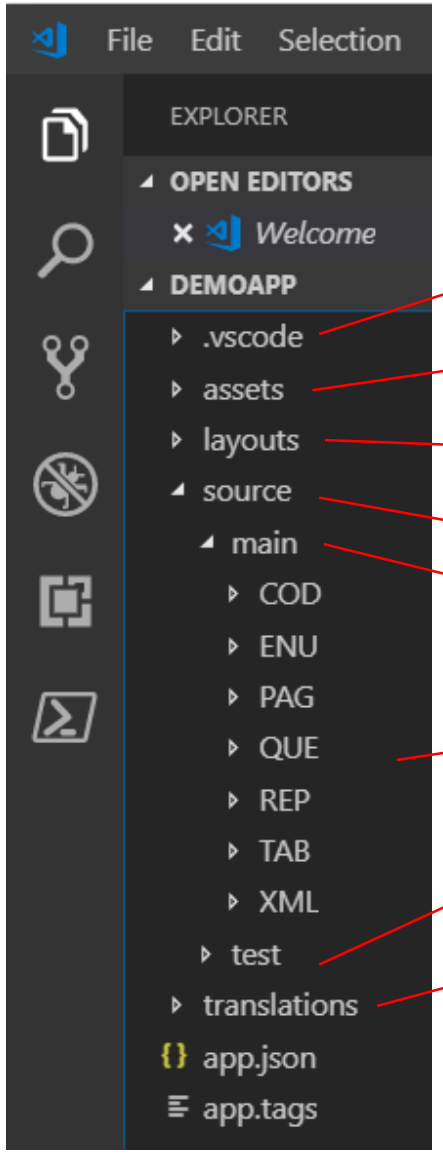
Philosophy

- Group together what belongs together
- Forget about IDs as sort criterion
- Create an understandable and easily enforceable rule on where code should be stored and how files should be named

Table x	or	Extension of Table x	Object Codeunit of Table x		*xliff ?*	*log of Table x*	„Agent Folder“	Table x
Page y	or	Extension of Page y	Object Codeunit of Page y		*xliff ?*	*log of Page y*	„Agent Folder“	Page y
Report z	or	*(Report Extension)*	Object CU of Report z	Layout	*xliff ?*	*log of Report z*	„Agent Folder“	Report z
Codeunit u			Object CU of Codeunit u		*xliff ?*	*log of Codeunit u*	„Agent Folder“	Codeunit u
XML Port v	or	*(?XML Port Extension?)*	Object CU of XML Port v		*xliff ?*	*log of XML Port v*	„Agent Folder“	XML Port v
Query w	or	*(?Query Extension?)*	Object CU of Query w		*xliff ?*	*log of Query w*	„Agent Folder“	Query w
Enum q	or	Enum Extension			*xliff ?*	*log of Enum q*	„Agent Folder“	Enum q
Control Add-In r			Object CU of *Ctrl. Add-In r*		*xliff ?*	*log of Ctrl Add-In r*	„Agent Folder“	Ctrl Add-In r

Programming in AL

APP Organization - Enveloping Folder Structure



contains „launch.json“

contains assets like a logo …

contains report layouts to be published (c.f. below.)

organizes the actual program files, distinguishes …

„main“: distinguishes object types via subfolder structure <type>

<type> folders: contains **agent object folders** (location of program files, c.f. below)

„test“: contains test codeunits per GAP

contains translation file of the app, if existing (c.f. below.)

app.json file

app.tags (file): contains collection of „GAP“-tags representing al GAPs implemented so far, c.f. below

Programming in AL

APP Organization - Naming Conventions Agent Folders / Files

Folder Names	<Type>	.	<Agent Object Name, abbreviated>	*[.<Origin>]*		
File Names	<Type>	.	<Agent Object Name, abbreviated>	*[.<Origin>]*	*[.<Subtype>]*	.al
Object Names	CCO	*[Agent Type]*	<Agent Object Name>	*[_+]*		
Log File	<Folder Name>			.log		(for docu-purposes, c.f. below)

Type	denotes the Object Type of the source Object could be TAB, PAG, REP, COD, …
„abbreviated“	means without spaces …, as suggested as Variable Name by the Object Designer „Purchase Header → PurchaseHeader“
Origin	[„“,“cc“]; „“ if Object is app-external „cc“ if Object is app object
Subtype	[„“, „ext“, „code“]; „ext“ for extension objects; „code“ for Object Codeunits
„Agent Type“	Used for Object CUs only: indicates the type of the related object [T, P, R, X, C, …]
Appendix „+“	indicates in object names that the object extends a standard object.

Programming in AL APP Organization - Naming Conventions: Examples)

Object	Name of Object File	Name of Object	Object Folder
Tableextension *Example 1: Item* *Example 2: Purchase Header*	TAB.<Name of extended Tab, …>.ext.al *TAB.Item.ext.al* *TAB.PurchaseHeader.ext.al*	CCO_<Name of extended Tab>_+ *"CCO Item +"* *"CCO Purchase Header +"*	TAB.<Name Table, …> *TAB.Item* *TAB.PurchaseHeader*
Pageextension *Example: Item Card*	PAG.<Name of extended Page, …>.ext.al *PAG.ItemCard.ext.al*	CCO_<Name of extended Page>_+ *"CCO Item Card +"*	PAG.<Name of Page, …> *PAG.ItemCard*
Table (== new Table) *Example: My App Table*	TAB.<Name of new Table, …>.al *TAB.MyAppTable.cc.al*	CCO_<Name of new Table> *"CCO My App Table"*	TAB.<Name Table, …>.cc *TAB.MyAppTable.cc*
Page (== new Page) *Example: My App Page*	PAG.<Name of new Page, …>.al *PAG.MyAppPage.cc.al*	CCO_<Name of new Page> *"CCO My App Page"*	PAG.<Name of Page, …>.cc *PAG.MyAppPage.cc*
Report (== new Report) *Example: My App Report*	REP.<Name of new Report, …>.al *REP.MyAppReport.cc.al*	CCO_<Name of new Report> *"CCO My App Report"*	REP.<Name of Report, …>.cc *REP.MyAppReport.cc*
ObjectCU of APP-external Tabelle *Example 1: Item* *Example 2: Purchase Header*	TAB.<Name of Table, …>.code.al *TAB.Item.code.al* *TAB.PurchaseHeader.code.al*	CCO_T_<Name of Table>_+ *"CCO T Item +"* *"CCO T Purchase Header +"*	TAB.<Name of Table, …> *TAB.Item* *TAB.PurchaseHeader*
ObjectCU of APP-internal Tabelle *Example: My App Table*	TAB.<Name of new Table, …>.cc.code.al *TAB.MyAppTable.cc.code.al*	CCO_T_<Name of Table> *"CCO T My App Table"*	TAB.<Name of Table, …>.cc *TAB.MyAppTable.cc*
ObjectCU of APP-external Page *Example: Item Card*	PAG.<Name of Page>.code.al *PAG.ItemCard.code.al*	CCO_P_<Name of Page>_+ *"CCO P Item Card +"*	PAG.<Name of Page, …> *PAG.ItemCard*
ObjectCU of APP-internal Page *Example: My App Page*	PAG.<Name of Page>.code.al *PAG.MyAppPage.cc.code.al*	CCO_P_<Name of Page> *"CCO P My App Page"*	PAG.<Name of Page, …>.cc *PAG.MyAppPage.cc*
ObjectCU of APP-external Report *Example: Order Confirmation*	REP.<Name of Report>.code.al *REP.OrderConfirmation.code.al*	CCO_R_<Name of Report> *"CCO R Order Confirmation +"*	REP.<Name of Report, …> *REP.OrderConfirmation*
ObjectCU of APP-internal Report *Example: My App Report*	REP.<Name of Report>.code.al *REP.MyAppReport.cc.code.al*	CCO_R_<Name of Report> *"CCO R My App Report +"*	REP.<Name of Report, …>.cc *PAG.MyAppReport.cc*
…	…	…	…

Programming in AL

APP Organization - Documentation: log-files and app-tags

- Documentation via Log Files
  - One Log per Folder
  - Log file contains well known docu headers (per GAP)

```
<GAP-Tag>_<Date>_<Developer ID>: <Title of GAP>
        Information on changes …
```

A „GAP-Tag“ consists of the GAP Identifier and a two digit suffix counting the generations of implementations of the gap.

If a GAP Implementation tsarts, the suffix is „.01“, if is reopened after a first completion, it gets suffix „.02“ and so on.

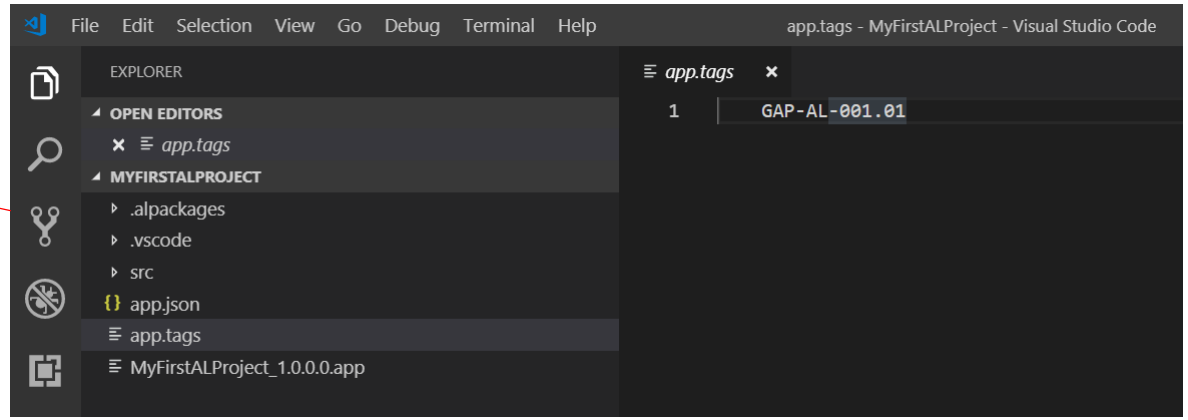
```
GAP-FI-001.01 01.01.2019 DENUE.MPRO: Default Description
        Created
                New field 55000 „Description 3“
        Modified
                …..
```

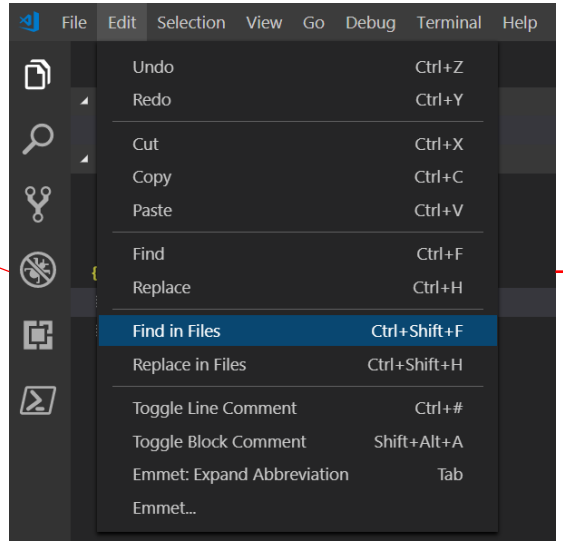
  - Remark: perhaps, we will be able to skip documentation in the future, using Azure DevOps …

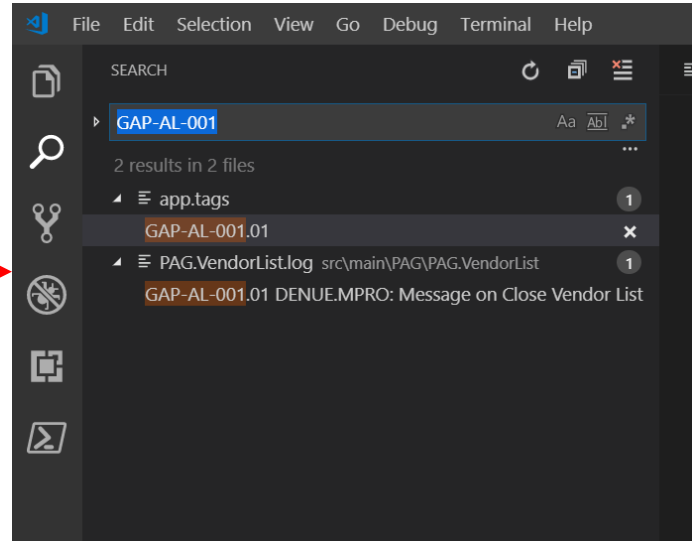
Programming in AL

APP Organization - Documentation: log-files and app-tags

- The „app.tags“ file
  - Loctaed in the root folder of the app
  - Contains al list of the GAPs implemented up to the current project state
  - [GAP-ID].[xx]: the „xx“ denotes the „development phase“ of the GAP
  - Enables „app-wide“ search for GAP Implementations







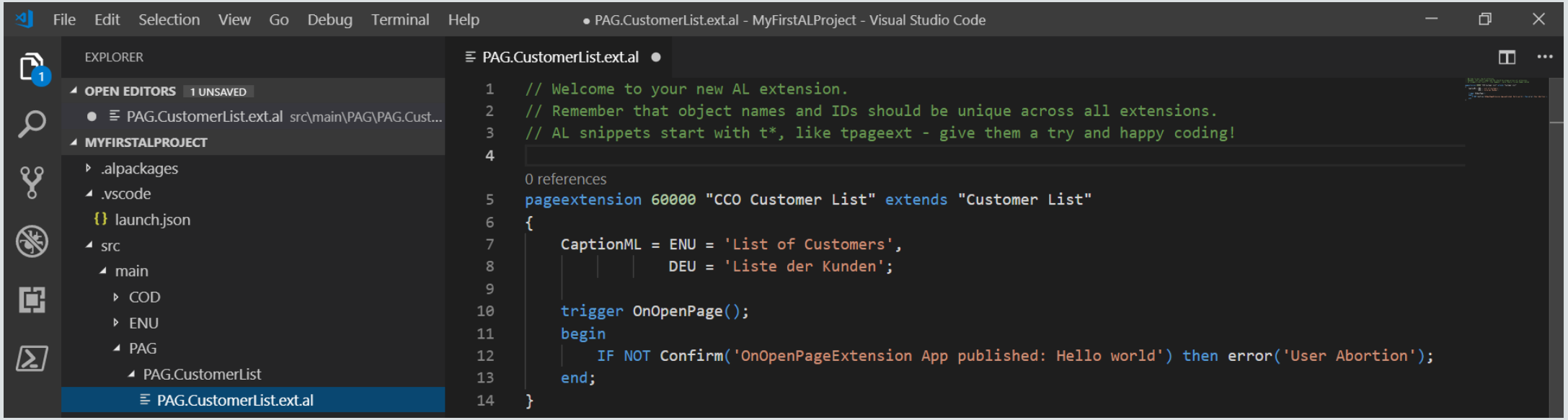
HandsOn – Exercise 6 | AL file structureProgramming in AL

HandsOn

AL Project: Setting up the proper File / Folder structure

Clean up your project

- Open „Hello World“ Project
- Setup a proper file / folder Structure
- Eventually rename your object files
- Eventually rename your objects



HandsOn – Exercise 7 | Codeunits and the Txt2AL Converter

HandsOn

Excercise CU programming via Txt2AL converter: Message on open Vendor List

Open C/Side Dev Environment

Implement the Object CU „classically“

Create Codeunit 55000 „CCO P Vendor List +“
- Add some properties to the CU, e.g.
  - TableNo = Vendor
- Add Event Subscriber „HandleOnOpenPage“
- Add function „LaunchMessageVendorListIsOpen“
- Call function from within subscriber

Test Functionality

Convert to AL

Configure and use Txt2AL converter

Remove Codeunit from object designer

Integrate solution into „MyFirstALProject“

Eventually create the Project, setup app.json / launch.json
Setup proper file structure
Integrate the converted file
View file content

Run the Project

Eventually modify „launch.json“ to get vendor list launched immediately

HandsOn – Exercise 7.5 | Extension Programming and the Txt2AL Converter

HandsOn

Extension programming via Txt2AL converter – GAP-AL-000: „Description 3“ on „Item Card“

New „Description 3“ field in table „Item“, shown on „Item Card“
New Action on „Item Card“ to copy „Description 3“ from „Description“

Open C/Side Dev Environment

Implement the Object CU „classically“

- Create backup of table „Item“ and page „Item Card“
- Add a field „Description 3“ to the Item Table
- Add the new field on the „Item Card“ (where ever you want)
- Implement a new action „Copy Description 3 from Description“
- Implement the Object CU to hold relevant code

→ Don‘t forget naming conventions

Test Functionality

Convert to AL

- Configure and use Txt2AL converter, configure „Originals“-Folder!
  Here: „C:\Users\Public\Desktop\Original“
- Remove Modifications from object designer

Integrate solution into „MyFirstALProject“

- Eventually create the Project, setup app.json / launch.json
- Setup proper file structure
- Integrate the converted file
- View file content
- Eventually recreate „CaptionMLs“ which are not provided by the converter

Run the Project

- Eventually modify „launch.json“ to get „Item Card“ launched immediately

Programming in AL

- Preliminaries (Intellisense / Txt2AL Converter / …)
- HandsOn … My first Page Extension – CaptionML
- App Organization
- Object Types: Theory and HandsOn

Programming in AL
Object Types: Theory and HandsOn - Codeunits

- https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-codeunit-object
- Snippet "*tcodeunit*"
- Remember Naming

File Names	<Type>	.	<Agent Object Name, abbreviated>	*[.<Origin>]*	*[.<subtype>]*	.al
Object Names	CCO	*[Agent Type]*	<Agent Object Name>	*[_+]*		

Type	denotes the Object Type of the source Object could be TAB, PAG, REP, COD, …
„abbreviated“	means without spaces …, as suggested as Variable Name by the Object Designer „Purchase Header → PurchaseHeader“
Origin	[„“,“cc“]; „“ if Object is app-external „cc“ if Object is app object
[.<subtype>]	If CU is an Object Codeunits, else empty
„Agent Type“	Used for Object CUs only: indicates the type of the related object [T, P, R, X, C, …]
Appendix „+“	indicates in object names that the object extends a standard object.

Programming in AL
Object Types: Theory and HandsOn - Codeunits

- Examples

	Object Name	Object File	Object Folder
Object CU of Table „Item“	„CCO T Item +“	TAB.Item.code.al	TAB.Item
Object CU of Page „Item Card“	„CCO P Item Card +“	PAG.ItemCard.code.al	PAG.ItemCard
Object CU of „New APP Table“	„CCO T New App Table“	TAB.NewAppTable.code.al	TAB.NewAppTab
Object CU of CU 90 „Purch.-Post“	„CCO C Purch.-Post +“	COD.PurchPost.code.al	COD.PurchPost
New „App Library“ Codeunit	„CCO App Library	COD.AppLibrary.al	COD.AppLibrary

File Names	<Type>	.	<Traditional Object Name, abbreviated>	*[.<Origin>]*	*[.<subtype>]*	.al
Object Names	CCO	*[rel.Obj. Type]*	<Traditional Object Name>	*[_+]*		

Type	denotes the Object Type of the source Object could be TAB, PAG, REP, COD, …
„abbreviated“	means without spaces …, as suggested as Variable Name by the Object Designer „Purchase Header → PurchaseHeader“
Origin	[„“,“cc“]; „“ if Object is app-external „cc“ if Object is app object
[.<subtype>]	If CU is an Object Codeunits, else empty
„rel.Obj. Type“	Used for Object CUs only: indicates the type of the related object [T, P, R, X, C, …]
Appendix „+“	indicates in object names that the object extends a standard object.

Programming in AL
Object Types: Theory and HandsOn - Codeunits

- Structure

Codeunit <ID> "<Name>"
{
[Properties]
[Variables]
[Triggers]
[Procedures]
}

PAG.VendorList.code.al

```
0 references
codeunit 60000 "CCO P Vendor List +"
{

    Permissions = TableData Vendor = rimd;

    var
        0 references
        MyIntegerVariable: Integer;
        0 references
        Vendor: Record Vendor;

    trigger OnRun()
    begin
    end;


    [EventSubscriber(ObjectType::Page, Page::"Vendor List", 'OnClosePa
    0 references
    local procedure HandleOnClosePage()
    begin
        LaunchMessageVendorListClosed();
    end;

    1 reference
    local procedure LaunchMessageVendorListClosed()
    begin
        Message('Vendor List is now beeing Closed');
    end;
}
```

Programming in AL
Object Types: Theory and HandsOn - Codeunits

- Properties
  - https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-codeunit-properties
  - Syntax

    <Property> = <Value>;

    Use Context IntelliSense to see possible Properties.

    Use Context IntelliSense after the "=" to see possible Values.
  - Example

```
Permissions = tabledata "Item" = RIMD,
              tabledata "Item Ledger Entry" = RIMD;
```

Programming in AL
Object Types: Theory and HandsOn - Codeunits

- Variables
  - Syntax

    var

    [Attributes]

    <Name>:<Definition>

    Possible Attributes: „InDataSet“, „RunOnClient“

    Use context intelliSense …
  - Examples

    var

    Item: Record Item;

    ItemNo: Code[20];

    [InDataSet]

    DescriptionVisible: Boolean;

Programming in AL
Object Types: Theory and HandsOn - Codeunits

- Procedures
  - Syntax

[Attributes]
[local] procedure <Name>([Parameters]) [ReturnValue]
[Variables (local)]
begin
[Code]
end;

Use context intelliSense …

[Attributes]

- *[IntegrationEvent(<IncludeSender>, <GlobalVarAccess>)]*
- *[BusinessEvent(<IncludeSender>, <GlobalVarAccess>)]*
- *[EventSubscriber(ObjectType::<ObjectType>, <ObjectId>, <EventName>, <ElementName>, <SkipOnMissingLicense>, <SkipOnMissingPermission>)]*

Use Context IntelliSense to see possible EventNames.

Use Context IntelliSense to see possible ElementNames.

Programming in AL
Object Types: Theory and HandsOn - Codeunits

[Parameters]

[var] <Name>: <Definition>

Multiple Parameters separated by semicolon.

Example:

procedure SelectItem(var Item: Record "Item"; DefaultItemNo: Code[20]) Selected: Boolean

[ReturnValue]

[Name]: <Definition>

Multiple Parameters separated by semicolon.

Example:

procedure SelectItem(var Item: Record "Item"; DefaultItemNo: Code[20]) Selected: Boolean

[Variables (local)]

c.f. above

Programming in AL
Object Types: Theory and HandsOn - Codeunits

- Triggers
  - Syntax

```
Trigger <Name>()
[Variables (local)]
begin
    [Code]
end;
```

Use Context IntelliSense to see possible Triggers (<Name>: in the case of Codeunits, its only OnRun)

Always Remember **If you don‘t know, what to do - 2: Use Txt2AL Converter** to get detailed insight

HandsOn – Exercise 8 | Procedures and Snippets

HandsOn

Implement „Message Vendor List is closed“ using Snippets

Open (My first) AL Project

Find and Open Object CU of Vendor List

Implement new Functionality

Run and Test the Project

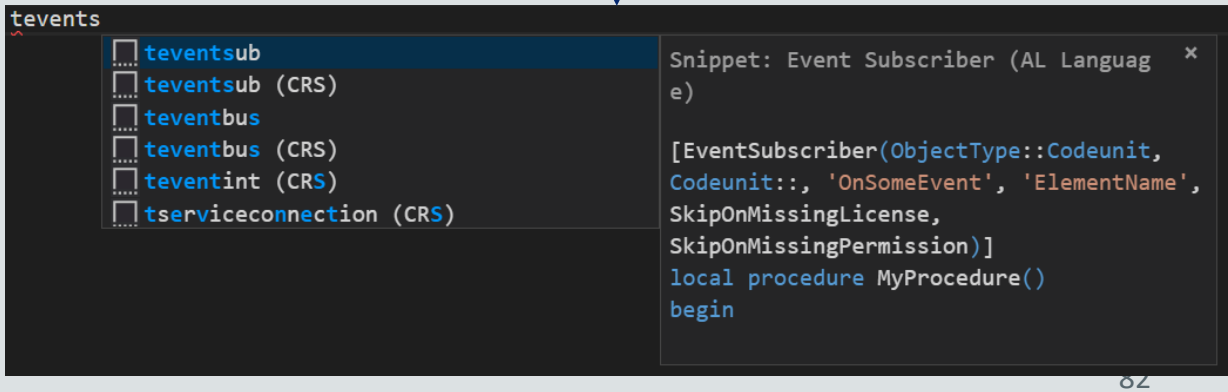
Place cursor in new line (beyond existing Event Subscriber Procedure)

Type (leading) fragment of „teventsubscriber“ to start intelliSense

Implement the Desired Function „LaunchMessageVendorListClosed“

```
local procedure LaunchMessageVendorListClosed()
begin
    Message('Vendor List is Closed');
end;
```

Choose the right snippet and implement the Event Subscriber



HandsOn – Exercise 9 | Documentation „logs and tags“

Finish „GAP-AL-000“: „Description 3 on Item Card“ and „GAP-AL-001: Message on Close Vendor List “

HandsOn

Open (My first) AL Project

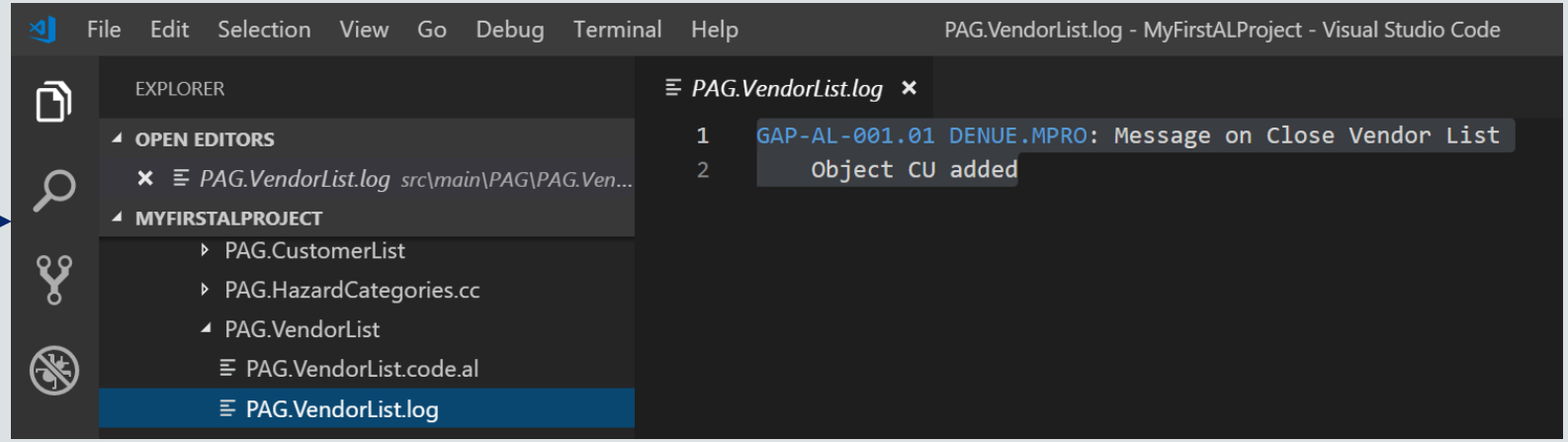
Create Documentation (log-file(s))

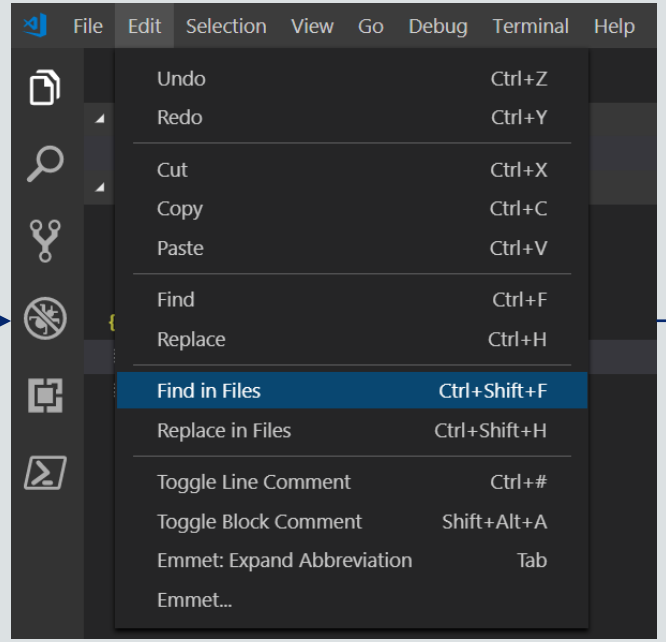
Log your GAPs within the „app.tags“-file

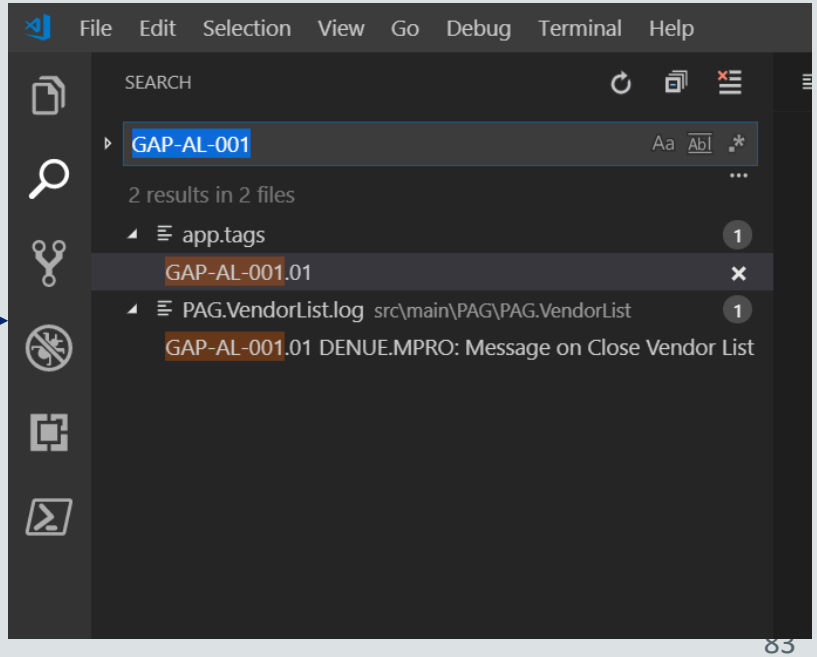
```
app.tags
1  GAP-AL-000.01
2  GAP-AL-001.01
```

Close open Editors and search for the GAP

Run and Test the Project







Programming in AL
Object Types: Theory and HandsOn - Pages

- https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-page-object
- Snippet „tpage“ (Card, API, List)
- Naming

File Names	PAG	.	<Agent Object Name, abbreviated>	.cc	.al
Object Names	CCO	<Agent Object Name>			

„abbreviated“ means without spaces …,
as suggested as Variable Name by the Object Designer
„Purchase Header → PurchaseHeader“

- Example

	Object Name	Object File	Object Folder
New Page „My new Page“	„CCO My new Page“	„TAB.MyNewPage.cc.al“	„PAG.MyNewPage.cc“

- Note that all Page files have a „.cc“ in their names …. all of them are new, always!

Programming in AL
Object Types: Theory and HandsOn - Pages

- Structure

```
Page <ID> "<Name>"
{
    [Properties]
    [Layout]
    [Actions]
    [Variables]
    [Triggers]
    [Procedures]
}
```

Syntax as before
Special Properties (amongst others)
UsageCategory
https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-usagecategory-property
https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-al-menusuite-functionality
AplicationArea
https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-applicationarea-property

Integrates page into search functionality

Complex but easy syntax. Use Txt2AL Converter to get more insight

See below and / or Use Txt2AL Converter to get more insight.

Globals of the Page, Syntax as before

Page Triggers, Syntax as before

Syntax as before

Layout Elements:
Group
Label
Field
Repeater
…

- If you don‘t know Syntax of [Layout] and [Actions], use Txt2AL Converter (or intelliSense)
- Page snippet comes with „empty“ frameworks of pages of certain types, showing both „Layout“ and „Action“ syntax
- VS Code Extsnsion „CRS AL Language Extension“ comes with additional page snippets

Programming in AL
Object Types: Theory and HandsOn - Pages

- Actions Syntax

```
actions
{
    area(<area name>)
    {
        group(<group name>)
        {
            [Properties]
            action(<name>)
            {
                [Properties]
                [Trigger OnAction]
            }
            [actions]
            [groups]
        }
        [groups]
    }
    [areas]
}
```

„Syntax as usual“

```
actions
{
    0 references
    area(navigation)
    {
        0 references
        group("D&ebitor")
        {
            Caption = '&Customer';
            Image = Customer;
            0 references
            action(Dimensionen)
            {
                ApplicationArea = Dimensions;
                Caption = 'Dimensions';
                Image = Dimensions;
                RunObject = Page "Default Dimensions";
                RunPageLink = "Table ID"=CONST(18),
                              "No."=FIELD("No.");
                ShortCutKey = 'Shift+Ctrl+D';
                ToolTip = 'View or edit dimensions, such as area, pr
            }
        }
    }
}
```

HandsOn – Exercise 9.5 | Pages

HandsOn

Convert a Page of your choice from C/Side to AL and analyze the „Layout“

```al
page 257 "Source Codes"
{
    ApplicationArea = Basic,Suite;
    Caption = 'Source Codes';
    PageType = List;
    SourceTable = "Source Code";
    UsageCategory = Administration;

    layout
    {
        area(content)
        {
            repeater(Control1)
            {
                ShowCaption = false;
                field("Code";Code)
                {
                    ApplicationArea = Basic,Suite;
                    ToolTip = 'Specifies the source code.';
                }
                field(Description;Description)
                {
                    ApplicationArea = Basic,Suite;
                    ToolTip = 'Specifies a description of what the code stands for.';
                }
            }
        }
        area(factboxes)
        {
            systempart(Control1900383207;Links)
            {
                Visible = false;
            }
        }
    }
```

```al
    actions
    {
        area(navigation)
        {
            group("&Herkunft")
            {
                Caption = '&Source';
                Image = CodesList;
                action(Fibujournale)
                {
                    ApplicationArea = Basic,Suite;
                    Caption = 'G/L Registers';
                    Image = GLRegisters;
                    RunObject = Page "G/L Registers";
                    RunPageLink = "Source Code"=FIELD(Code);
                    RunPageView = SORTING("Source Code");
                    ToolTip = 'View posted G/L entries.';
                }
            }
        }
    }
}
```

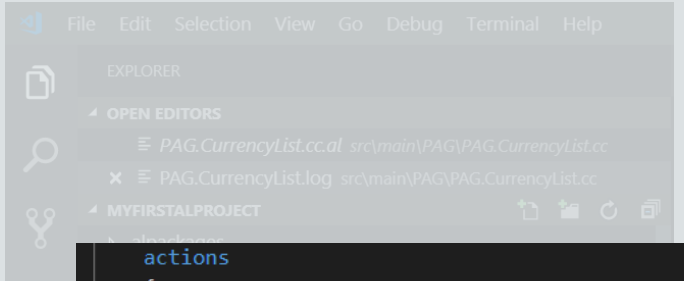
HandsOn – Exercise 10 | Pages

HandsOn

Implement „GAP-AL-002: Currency List“

Create a simple List Page „Currency List“, which shows a list of all Currencies and their Descriptions. It shall not be possible to edit the list. Via an Action „Show extended List“, it shall be possible to open the standard currency list „Currencies“.

- Open (My first) AL Project
- Create the file structure needed
- Implement the page without the action
- Verify functionality
- Implement the desired action
- Run and Test the Project




```
actions
{
    0 references
    area(Processing)
    {
        0 references
        action(ActionName)
        {
            ApplicationArea = All;
            Caption = 'Exch. &Rates';
            Image = Currencies;
            Promoted = true;
            PromotedCategory = Process;
            RunObject = Page Currencies;
            RunPageOnRec = true;
            // RunPageLink = Code = FIELD (Code);

            trigger OnAction()
            begin

            end;
        }
    }
```

PAG.CurrencyList.cc.al

```
page 60001 "CCO Currency List"
{
    PageType = List;
    CaptionML = DEU = 'Währungsliste',
                ENU = 'Currency List';
    ApplicationArea = All;
    UsageCategory = Lists;
    SourceTable = Currency;
    Editable = false;

    layout
    {
        0 references
        area(Content)
        {
            0 references
            repeater(Currencies)
            {
                0 references
                field(Code; Code)
                {
                    ApplicationArea = All;
                }
                0 references
                field(Description; Description)
                {
                    ApplicationArea = All;
                }
            }
        }
    }
}
```

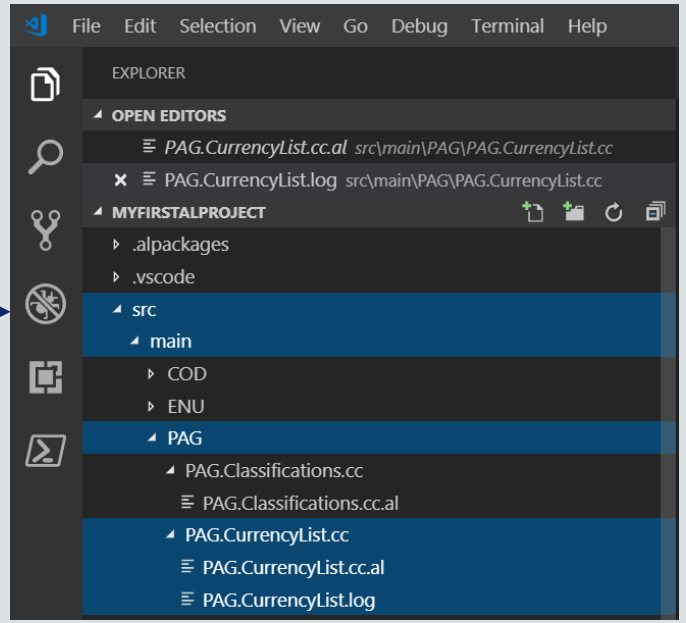
HandsOn – Exercise 10 | Pages

HandsOn

Implement „GAP-AL-002: Currency List“

Create a simple List Page „Currency List“, which shows a list of all Currencies and their Descriptions. It shall not be possible to edit the list. Via an Action „Show extended List“, it shall be possible to open the standard currency list „Currencies“.

- Open (My first) AL Project
- Create the file structure needed
- Implement the page without the action
- Verify functionality
- Implement the desired action
- Run and Test the Project

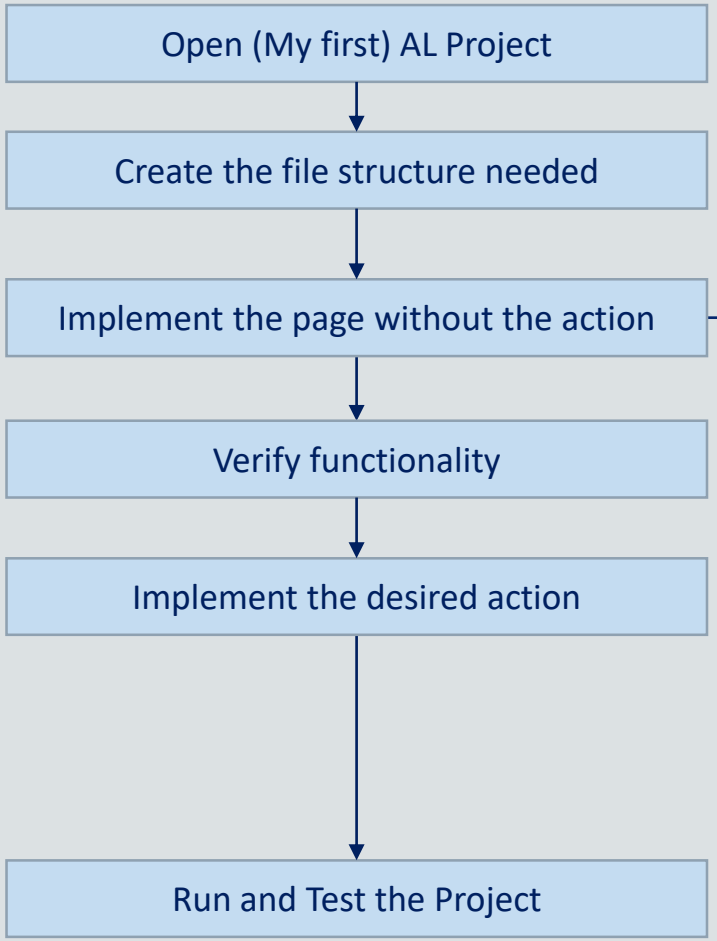


HandsOn – Exercise 10 | Pages

HandsOn

Implement „GAP-AL-002: Currency List“

Create a simple List Page „Currency List“, which shows a list of all Currencies and their Descriptions. It shall not be possible to edit the list. Via an Action „Show extended List“, it shall be possible to open the standard currency list „Currencies“.



PAG.CurrencyList.cc.al

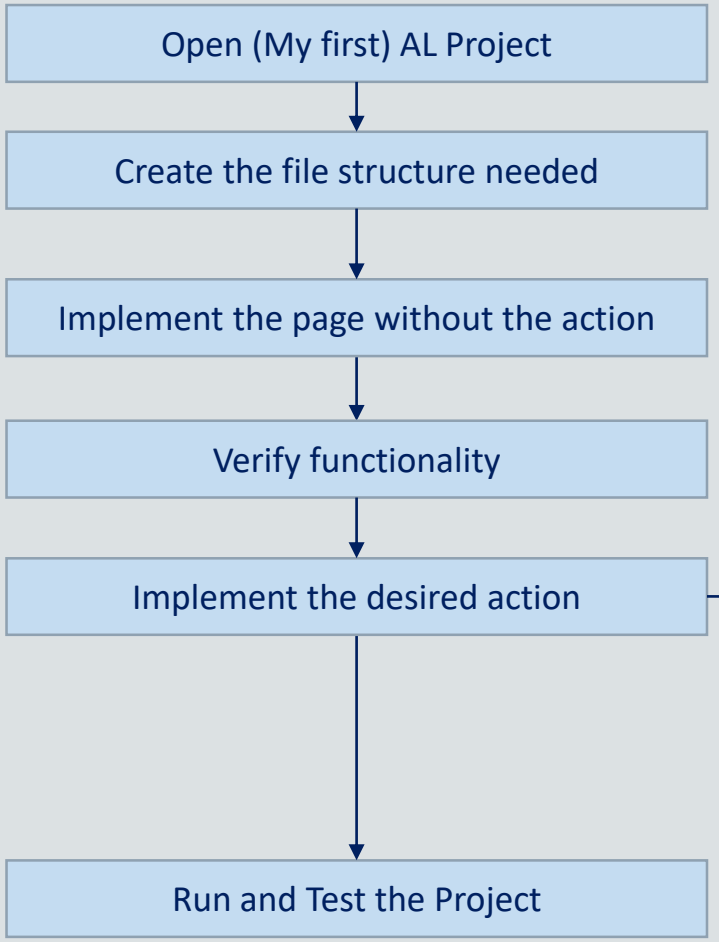
```
page 60001 "CCO Currency List"
{
    PageType = List;
    CaptionML = DEU = 'Währungsliste',
                ENU = 'Currency List';
    ApplicationArea = All;
    UsageCategory = Lists;
    SourceTable = Currency;
    Editable = false;

    layout
    {
        0 references
        area(Content)
        {
            0 references
            repeater(Currencies)
            {
                0 references
                field(Code; Code)
                {
                    ApplicationArea = All;
                }
                0 references
                field(Description; Description)
                {
                    ApplicationArea = All;
                }
            }
        }
    }
}
```

HandsOn – Exercise 10 | Pages

HandsOn

Implement „GAP-AL-002: Currency List“

Create a simple List Page „Currency List“, which shows a list of all Currencies and their Descriptions. It shall not be possible to edit the list. Via an Action „Show extended List“, it shall be possible to open the standard currency list „Currencies“.




```
actions
{
    0 references
    area(Processing)
    {
        0 references
        action(ActionName)
        {
            ApplicationArea = All;
            Caption = 'Exch. &Rates';
            Image = Currencies;
            Promoted = true;
            PromotedCategory = Process;
            RunObject = Page Currencies;
            RunPageOnRec = true;
            // RunPageLink = Code = FIELD (Code);

            trigger OnAction()
            begin

            end;
        }
    }
}
```

Programming in AL
The Truth about Programming in AL

Don‘t know how to do it in AL?

But: If you know how the Standard does it in C/Side

Always Remember

If you don‘t know, what to do - 2: Use Txt2AL Converter

You could think about scetching the „unknown“ syntax in C/Side yourself and then …
You could even think about implementing the desired functionality in C/Side as a whole and then …

HandsOn – Exercise 11 | Object / File Structure

HandsOn

Implement „GAP-AL-003: Currency List advanced“

Extend the functionality of „GAP-AL-002 in a way that the user is prompted „Do you want to open the standard currency list“ when he hits the „Show extended List“ button.

Open (My first) AL Project

Reimplement the Action

Run and Test the Project

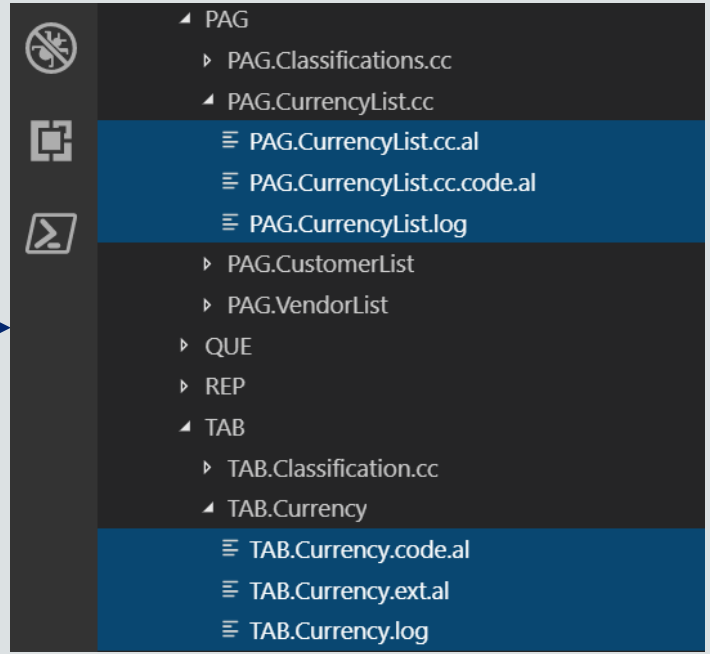
Page: OnAction Trigger
calls Object CU

ObjectCU(Page):
- HandleOnActionShowExtendedListTrigger calls
- QueryLaunchExtendedList(…)
calls „Agent“

Table Currency: CCOLaunchExtendedList(HideDialog, …)
calls Object CU

ObjectCU(Table) : LaunchExtendedList(HideDialog, …)
contains functionality

There might be other ways of carrying out this implementation



Programming in AL
Object Types: Theory and HandsOn - Tables

- https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-table-object
- Snippet „ttable“
- Naming

File Names	TAB	.	<Agent Object Name, abbreviated>	.cc	.al
Object Names	CCO	<Agent Object Name>			

„abbreviated“ means without spaces …,
as suggested as Variable Name by the Object Designer
„Purchase Header → PurchaseHeader“

- Example

	Object Name	Object File	Object Folder
· New Table „My new Table“	„CCO My new Table“	„TAB.MyNewTable.cc.al“	TAB.MyNewTable.cc

- Note that all Table files have a „.cc“ in their names …. all of them are new, always!

Programming in AL
Object Types: Theory and HandsOn - Tables

- Structure

Table <ID> "<Name>"
{
[Properties]
[Fields]
[Fieldgroups]
[Keys]
[Variables]
[Triggers]
[Procedures]
}

c.f. below

Syntax as before

- key name: must be prefixed
- First key in list is primary key

```
keys
{
    - reference
    key(Key1;"No.")
    {
        // Properties
        Clustered = true;

    }
    - reference
    key(Key2;"Vendor Item No.","Vendor No.")
    {
        // Properties
        Enabled = true;

    }
}
```

```
Clustered
Description                    Description property
MaintainSiftIndex
MaintainSqlIndex
ObsoleteReason
ObsoleteState
SqlIndex
SumIndexFields
```

Programming in AL
Object Types: Theory and HandsOn - Tables

- Structure

Table <ID> "<Name>"
{
[Properties]
[Fields]
[Fieldgroups]
[Keys]
[Variables]
[Triggers]
[Procedures]
}

[Properties]
[Triggers]

```
fields
{
    0 references
    field(70000; MyNewField; Code[20])
    {
        // Properties
        DataClassification = CustomerContent;
        // ....

        trigger OnValidate()
        var
            MyLocalVariable: Integer;
        begin
            // Validation Code
        end;

        trigger OnLookup()
        var
            MyLocalVariable: Integer;
        begin
            // OnLookupCode
        end;
    }
}
```

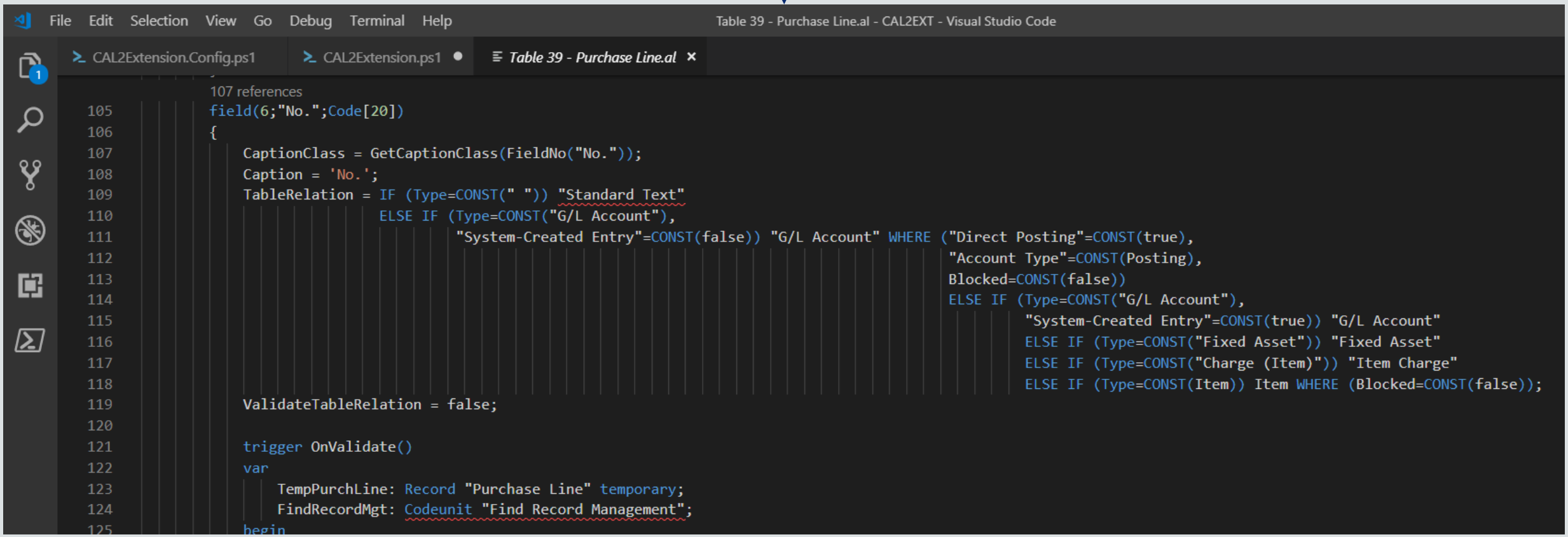
Excercise:
TableRelation Syntax

HandsOn – Exercise 11.5 | Table Relations and the Txt2AL converter

Gain insight into the syntax of Table Relations by „Txt2AL Converting“ table 39 „Purchase Line“

HandsOn

Configure Txt2AL COnverter → Convert Table 39 „Purchase Line“ → Learn the syntax of „Table Relation Property“



Programming in AL
Object Types: Theory and HandsOn - Tables

- Structure

Table <ID> "<Name>"
{
[Properties]
[Fields]
[Fieldgroups]
[Keys]
[Variables]
[Triggers]
[Procedures]
}

```
fieldgroups
{
    - reference
    fieldgroup(DropDown;"No.",Description,"Base Unit of Measure","Unit Price")
    {
    }
    - reference
    fieldgroup(Brick;"No.",Description,Inventory,"Unit Price","Base Unit of Measure","Description 2",Picture)
    {
    }
}
```

Programming in AL
Object Types: Theory and HandsOn - Tables

- Some „special“ properties
  - ***ObsoleteState***
    - https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-obsoletestate-property
    - Marks whether the table, field, or key is or will become obsolete (deprecated).

No	Not obsolete; Default
Pending	Will become obsolete; No Effect on Table and Data
Removed	Is obsolete; No Effect on Table; Data access only in upgrade codeunits, else Runtime Error

  - ***ObsoleteReason***
    - „Info field” to inform Developers
    - https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-obsoletereason-property
  - *TableType, ExternalName, ExternalSchema, LinkedObject, LinkedInTransaction*
    - «Properties to organize connections to external data sources»
    - https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-tabletype-property
    - https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-externalname-property
    - https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-externalschema-property
    - https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-linkedobject-property
    - https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/properties/devenv-linkedintransaction-property

HandsOn – Exercise 12 | Module Hazard Category Part 1

HandsOn

FRD

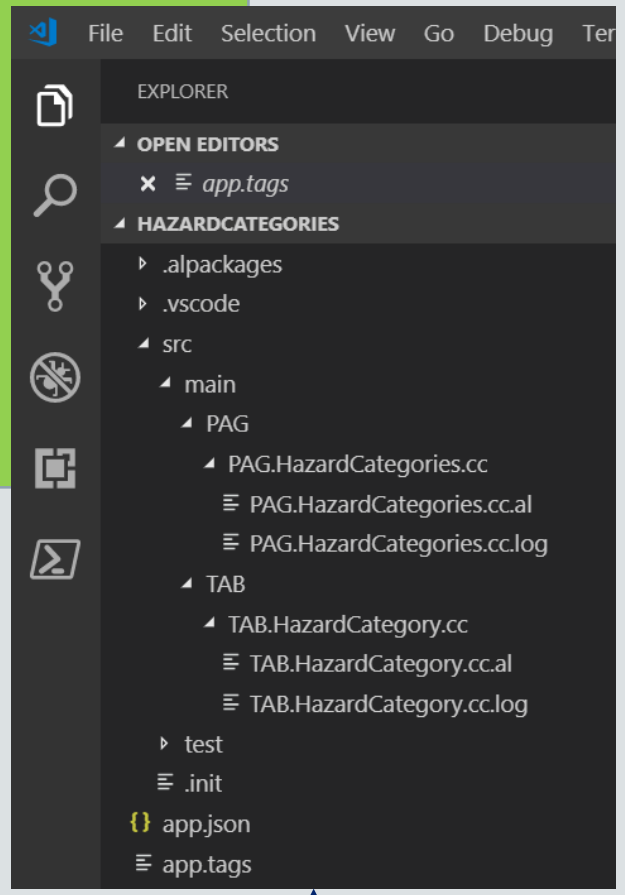
It shall be possible to group Items according to their "Hazard Category". "Hazard Categories" should be freely definable. When an article is purchased via ordering, its "Hazard Category" is to be displayed in the purchasing document and should be changeable there. During the course of item posting, the "Hazard Category" has to be transferred into the "Item Ledger Entries" and, if indicated, into the posted purchase documents.

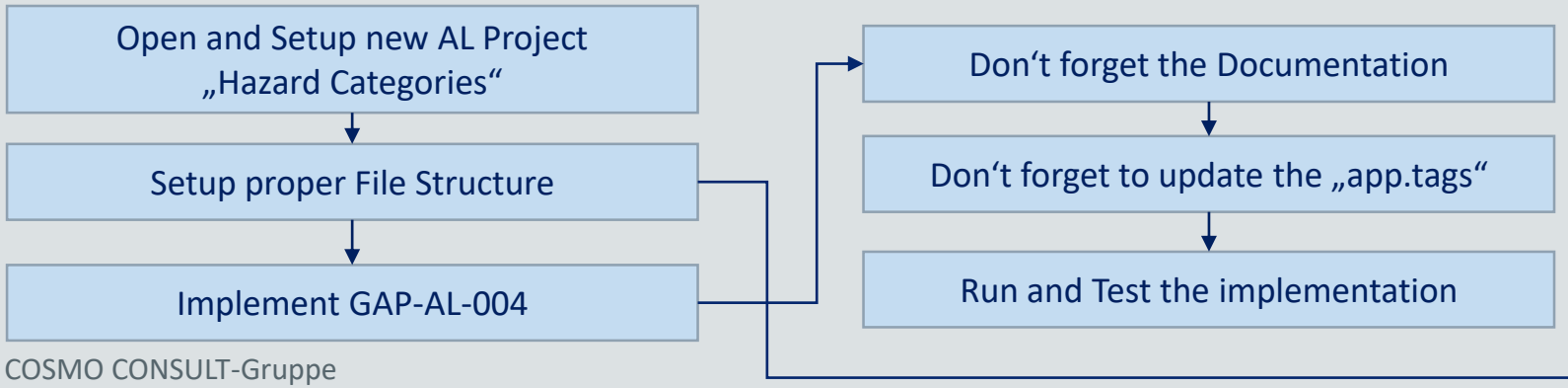
No other processes are to be respected.
No further details have to be respected concerning, e.g., manipulations of partially posted documents etc..

GAP-AL-004: Hazard Categories
A new Data Structure “Hazard Category” is to be implemented
(Table 60100 “Hazard Category” and List Page 60100 “Hazard Categories”).
Hazard Category

Code	*Code 20*
Description	*Text 80*

It shall not be possible to enter “empty” categories.
The new page shall be accessible via the search functionality.





Programming in AL
Object Types: Theory and HandsOn – Table Extensions

- https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-table-object
- Snippet „ttableext“
- Naming

File Names	TAB	.	<Name of extended Table, abbreviated>	*.ext*	.al
Object Names	CCO	<Name of extended Table>		+	

„abbreviated“ means without spaces …,
as suggested as Variable Name by the Object Designer
„Purchase Header → PurchaseHeader“

- Example

	Object Name	Object File	Object Folder
Table Extension of Table „Item“	„CCO Item +“	„TAB.Item.ext.al“	TAB.Item

Programming in AL
Object Types: Theory and HandsOn – Table Extensions

- Structure

tableextension <ID> <Name> extends <Name of extended Table>

{

[Properties]

[Fields]

[Fieldgroups]

[Keys]

[Variables]

[Triggers]

[Procedures]

}

c.f. Table Elements
Syntax as before

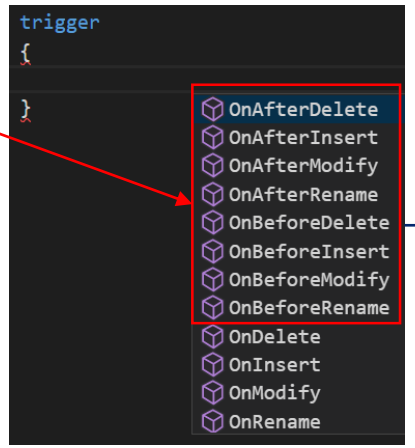
If you don‘t know, what to do - 1: Press „Ctrl+Space“

- Use intelliSense to get info on the available „items“
- Enter [Property] section and type Ctrl+Space to get info on the Properties you can modify
- Enter [Trigger] section and type … to get info on the available trigger to „hook in“.

Some modifications possible, Field Trigger code can not be changed here.
Use proper events or OnBefore-/OnAfterValidate Triggers … → c.f. below …

For adding / extending DropDown and / or Brick groups via a „addlast“ command
c.f. https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-field-groups

New Trigger coming with Extensions



What about the sequence of execution compared to
- the corresponding Table Event
- the corresponding Database Event

???? → see below

Programming in AL
Object Types: Theory and HandsOn – Table Extensions

- Remark 1 - Order of execution: how Table Extension Trigger fits into the picture

"Event / Trigger"	Insert(TRUE) / Rec in DB	INSERT(FALSE) / Rec in DB	MODIFY(TRUE)
OnBefore Event Table	+ / -	+ / -	+
OnBefore Trigger Extension	+ / -	-	+
On Trigger Table	+ / -	-	+
OnAfterGlobal Event	+ / -	-	+
OnBeforeDatabase Event	+ / -	+ / -	+
OnAfterDatabase Event	+ / -	+ / -	+
OnAfter Trigger Extension	+ / +	-	+
OnAfter Event Table	+ / +	+ / +	+

Programming in AL
Object Types: Theory and HandsOn – Table Extensions

- Remark 2 - On the „List of Properties / Things that could be changed“ on Table Extension level
  - MS Recommendation

I don't think we have documentation with an exhaustive list of the property that are modifiable. But, you can get access to it by declaring a small table extension in AL modifying a field and triggering IntelliSense.

```
tableextension 50100 MyExtension extends Customer
{
    fields
    {
        modify(Name)
        {
            // trigger IntelliSense here.
        }
    }
}
```

Intellisense will suggest only the properties that you can modify. All the ones that do not appear cannot be used so would be exported as warnings / comments by the txt2al.

Taken from https://github.com/Microsoft/AL/issues/2459

Programming in AL
Object Types: Theory and HandsOn – Table Extensions

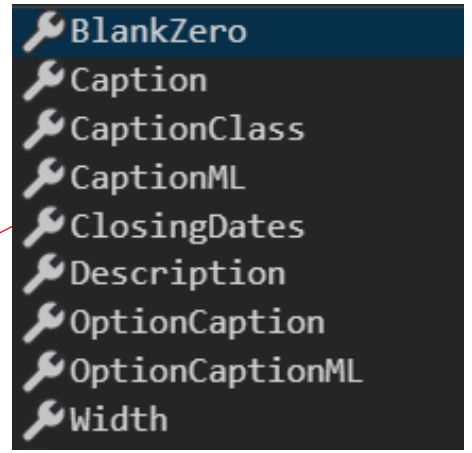
- Remark 3 - On the „List of Properties / Things that could be modified“ on field level: what intelliSense yields

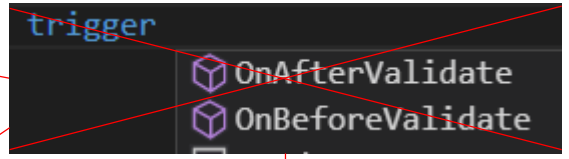
tableextension <ID> <Name> extends <Name of extended Table>
{
[Properties]
[Fields]
[Fieldgroups]
[Keys]
[Variables]
[Triggers]
[Procedures]
}

(c.f. „Table Elements“ for additional (new) fields)

Field Modifications

```
fields
{
    modify(Description)
    {
        // Properties

        // Trigger

    }
}
```





Normally, we would change / add field Trigger behaviour via subscribing the field events …
instead of using this extension trigger

For the order of execution …
See exercise below

Programming in AL
Object Types: Theory and HandsOn – Table Extensions

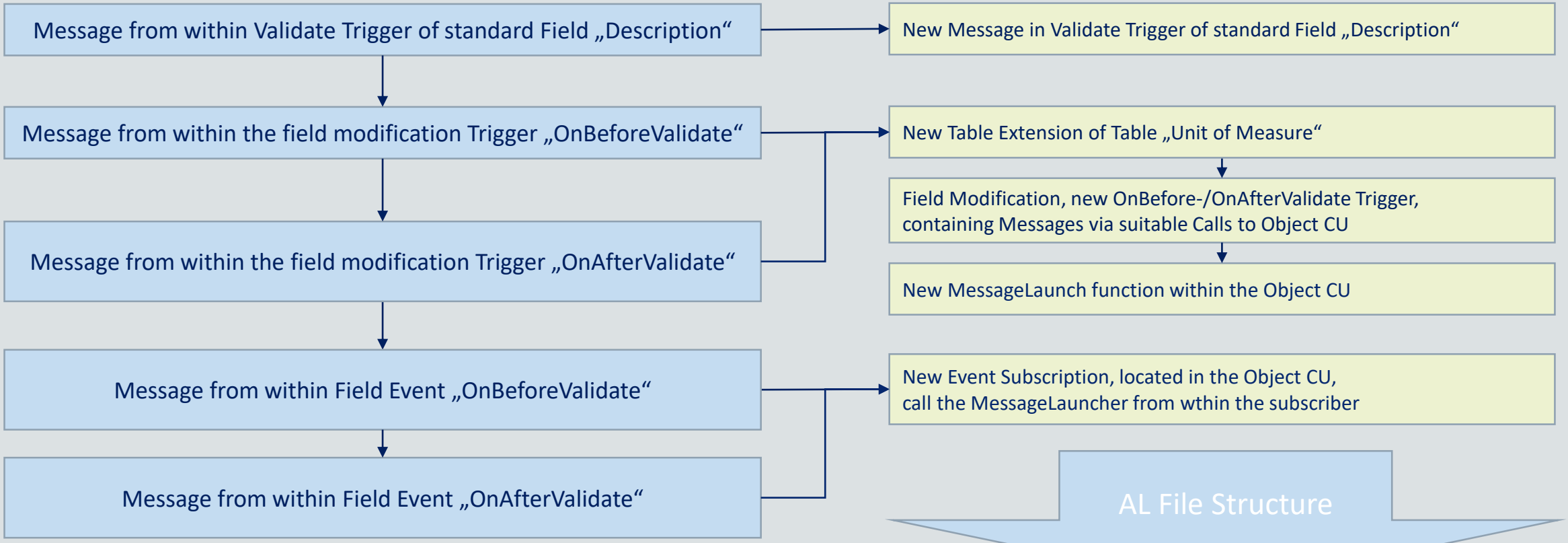
- Remark 4 – Extending keys
  - Up to today, besides of the primary key fields, a key field list must not contain fields from „outside“ the app!

HandsOn – Exercise 13 | Table Extension (Table) Trigger

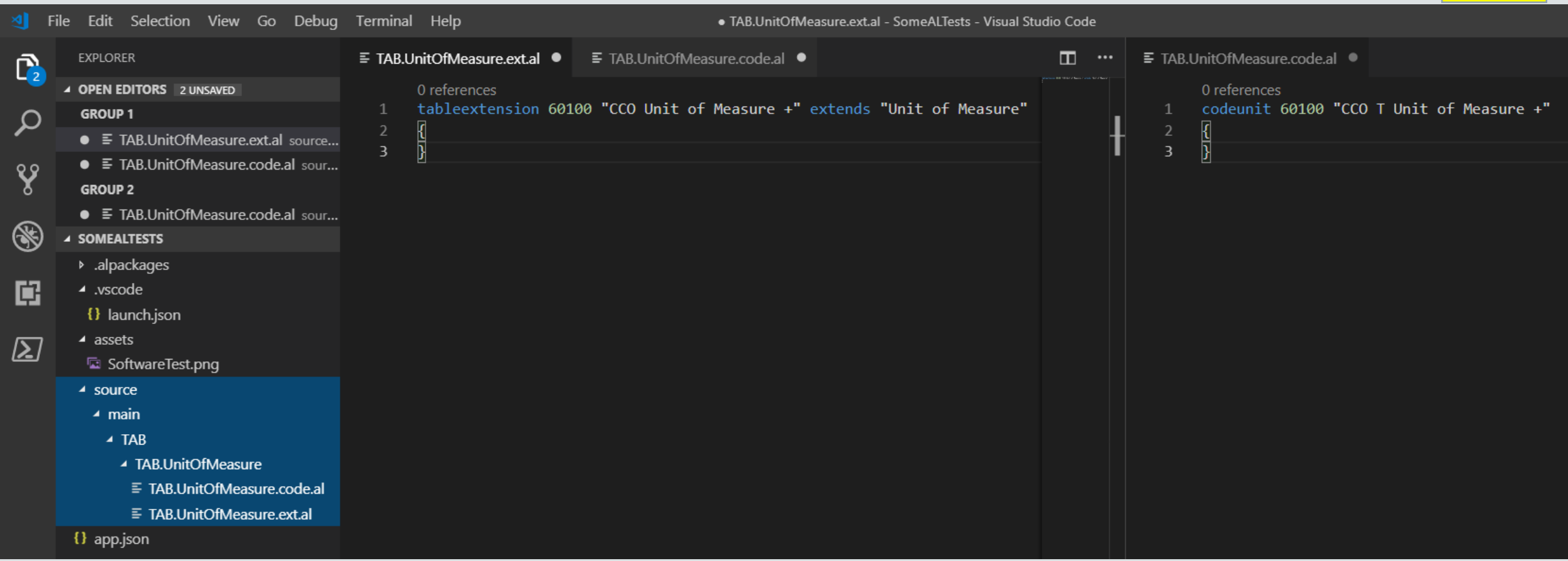
HandsOn

Sequence of Execution of table extension field triggers compared to relevant standard triggers and events.
Implement an Extension which is suited to show the order of execution of „OnBeforeValidate (Extension Trigger)“, „OnBeforeValidate (Table Field Event)“, „OnValidate (Table Field Trigger)“, „OnAfterValidate (Table Field Event)“ and „OnAfterValidate (Extension Trigger)“, use „Description“ field of „Unit of Measure“ as an example.
Implement a Message-Chain which is suited to show the order of execution.

Scetch of Design



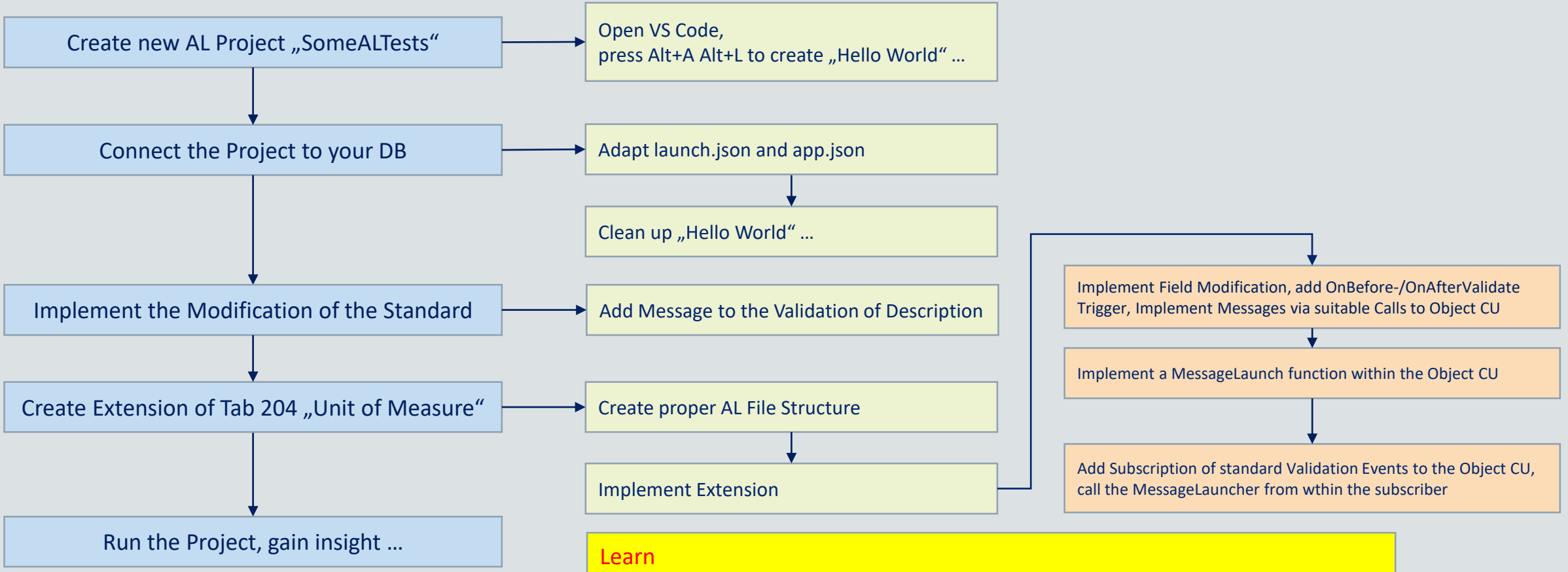
HandsOn – Exercise 13 | AL File Structure



HandsOn – Exercise 13 | Doing

HandsOn

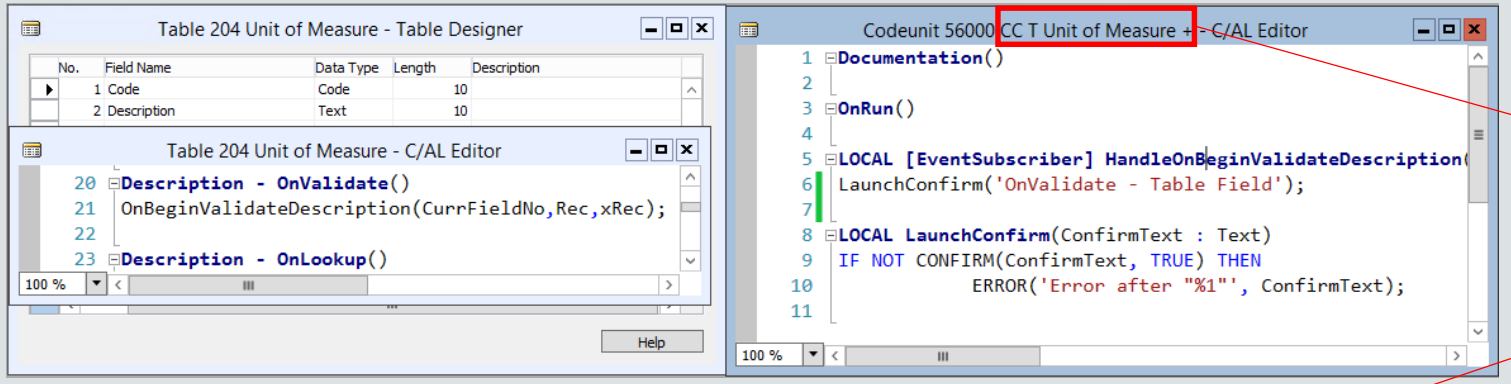
Sequence of Execution of table extension field triggers compared to relevant standard triggers and events.
Implement an Extension which is suited to show the order of execution of „OnBeforeValidate (Extension Trigger)“, „OnBeforeValidate (Table Field Event)“, „OnValidate (Table Field Trigger)“, „OnAfterValidate (Table Field Event)“ and „OnAfterValidate (Extension Trigger)“, use „Description“ field of „Unit of Measure“ as an example.
Implement a Message-Chain which is suited to show the order of execution.



Learn
1) how extension field trigger fit into the total field trigger / event scenario
2) that invoking intelliSense is always a good idea

HandsOn – Exercise 13 | Hybrid Implementation

HandsOn



!!!

TAB.UnitOfMeasure.ext.al

```
0 references
tableextension 60100 "CCO Unit of Measure +" extends "Unit of Measure"
{
    fields
    {
        modify(Description)
        {
            trigger OnBeforeValidate()
            begin
                ObjectCU.LaunchConfirm('OnBeforeValidate - Table Extension');
            end;

            trigger OnAfterValidate()
            begin
                ObjectCU.LaunchConfirm('OnAfterValidate - Table Extension');
            end;
        }
    }


    var
        2 references
        ObjectCU: codeunit "CCO T Unit of Measure +";
}
```

TAB.UnitOfMeasure.code.al

```
1 reference
codeunit 60100 "CCO T Unit of Measure +"
{
    [EventSubscriber(ObjectType::Table, Database::"Unit of Measure", 'OnBeforeValidateEvent', 'Description', false, false)]
    0 references
    local procedure HandleOnBeforeValidate()
    begin
        LaunchConfirm('OnBeforeValidate Event');
    end;

    [EventSubscriber(ObjectType::Table, Database::"Unit of Measure", 'OnAfterValidateEvent', 'Description', false, false)]
    0 references
    local procedure HandleOnAfterValidate()
    begin
        LaunchConfirm('OnAfterValidate Event');
    end;

    4 references
    procedure LaunchConfirm(ConfirmText: Text)
    begin
        if not Confirm(ConfirmText, true) then
            Error('Error after "%1"', ConfirmText);
    end;
}
```

Programming in AL
Object Types: Theory and HandsOn – Page Extensions

- https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-page-ext-object
- Snippet „tpageextension“
- Naming

File Names	PAG	.	<Name of extended Page, abbreviated>	*.ext*	.al
Object Names	CCO	<Name of extended Page>	+		

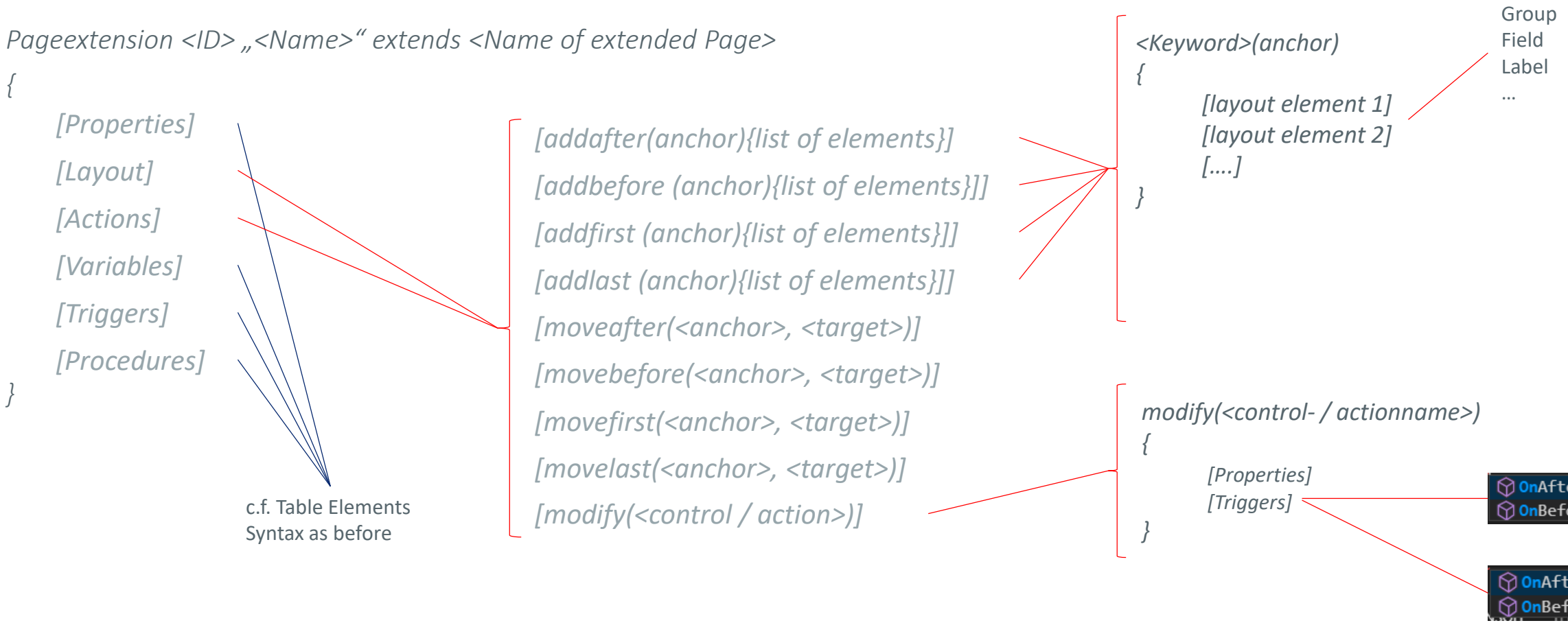
„abbreviated“ means without spaces …,
as suggested as Variable Name by the Object Designer
„Purchase Header → PurchaseHeader“

- Example

	Object Name	Object File	Object Folder
Extension of Page „Item Card“	„CCO Item Card +“	„PAG.ItemCard.ext.al“	„PAG.ItemCard“

Programming in AL
Object Types: Theory and HandsOn – Page Extensions

- Structure (similar to the structure of pages …)



HandsOn – Exercise 14 | Page Extension Layout

HandsOn

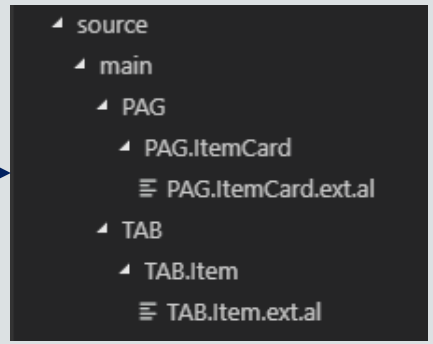
Create an Item (Card) extension

- Item Table: Add a new field „My new Description“ (Text 50)
- Item Card
  - Add a label „Here comes the Item No.“ on first place of fast tab „Item“
  - Move the „Replenishment System“ from where ever it is at the right bottom of the „Item“ fast tab.
  - Add a „Description Group“ at the right bottom of the „Item“ fast tab, containing all three Item Descriptions
  - Rename „Item“ fast tab to „Item Fast Tab“ („Artikelregister“)
  - Add a new first fast tab „Descriptions“, place all three „Descriptions“ on this fast tab
  - Add an action to the „Sales“ section of the „Navigate“ menu which launches the currency list.

Open / Create AL Project „SomeALTests“

Create proper file structure

Add field to „Item Extension“




```
0 references
tableextension 70000 "CCO Item +" extends Item //27
{
    fields
    {
        2 references
        field(70000; "CCO My New Description"; Text[80])
        {
            CaptionML = ENU = 'My new Description',
                        DEU = 'Meine neue Beschreibung';
            DataClassification = CustomerContent;
        }
    }
}
```

There might be other ways of carrying out this implementation

HandsOn – Exercise 14 | Page Extension Layout, continued 1

HandsOn

Create an Item (Card) extension

- Item Table: Add a new field „My new Description“ (Text 50)
- Item Card
  - Add a label „Here comes the Item No.“ on first place of fast tab „Item“
  - Move the „Relenishment System“ from whereever it is at the right bottom of the „Item“ fast tab.
  - Add a „Description Group“ at the right bottom of the „Item“ fast tab, containing all three Item Descriptions
  - Rename „Item“ fast tab to „Item Fast Tab“ („Artikelregister“)
  - Add a new first fast tab „Descriptions“, place all three „Descriptions“ on this fast tab
  - Add an action to the „Sales“ section of the „Navigate“ menu which launches the currency list.

Add a label „Here comes the Item No.“ on first place of fast tab „Item“

Check Result

Move the „Relenishment System“ from where ever it is at the right bottom of the „Item“ fast tab.

Check Result

Add a „Description Group“ at the right bottom of the „Item“ fast tab, containing all three Item Descriptions

```
pageextension 70000 "CCO ItemCard +" extends "Item Card" //30
{
    layout
    {
        addbefore
        {
            0 refe
            label
            {
                C
            }
        }
    }
    //
    }
    }
    // move
    //    (a
    movelast
}
```

```
    //    (actually: "Warehouse") at "head" of tab Item
    movelast(Item; "Replenishment System")

    addlast(Item)
    {
        0 references
        group(DescriptionBlock)
        {
            CaptionML = DEU = 'Beschreibungen',
                        ENU = 'Descriptions';

            0 references
            field(DescriptionAgain; Description)
            {
                ApplicationArea = All;
            }
            0 references
            field(Description2; "Description 2")
            {
                ApplicationArea = All;
            }
            0 references
            field("CCO My New Description"; "CCO My New Description")
            {
                ApplicationArea = All;
            }
        }
    }
}
```

There might be other ways of carrying out this implementation

HandsOn – Exercise 14 | Page Extension Layout, continued 2

HandsOn

Create an Item (Card) extension

- Item Table: Add a new field „My new Description“ (Text 50)
- Item Card
  - Add a label „Here comes the Item No.“ on first place of fast tab „Item“
  - Move the „Relenishment System“ from whereever it is at the right bottom of the „Item“ fast tab.
  - Add a „Description Group“ at the right bottom of the „Item“ fast tab, containing all three Item Descriptions
  - Rename „Item“ fast tab to „Item Fast Tab“ („Artikelregister“)
  - Add a new first fast tab „Descriptions“, place all three „Descriptions“ on this fast tab
  - Add an action to the „Sales“ section of the „Navigate“ menu which launches the currency list.

Rename „Item“ fast tab to „Item Fast Tab“ („Artikelregister“)

Check Result

Add a new first fast tab „Descriptions“, place all three „Descriptions“ on this fast tab

Check Result

Add an action to the „Sales“ section of the „Navigate“ menu which launches the currency list.

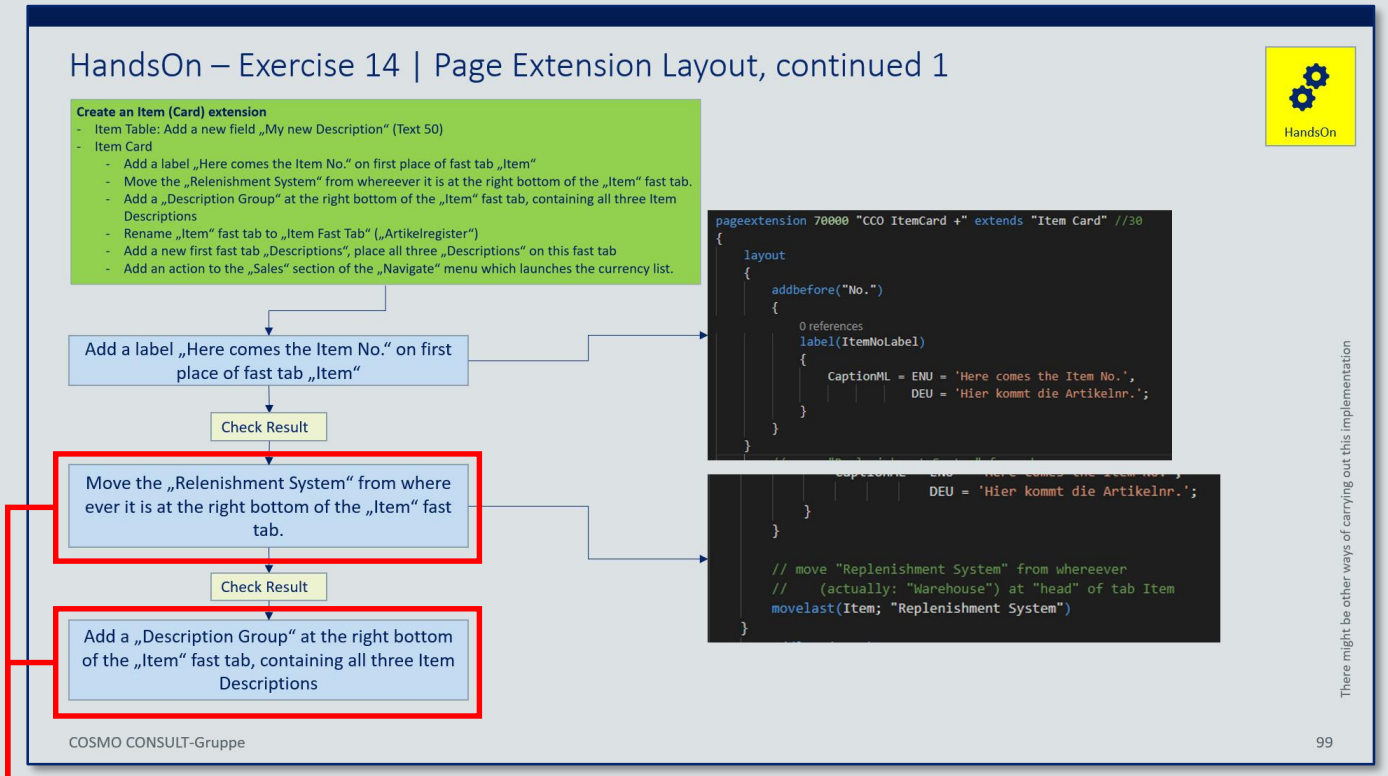
```
modify(Item)
{
    CaptionML = DEU = 'Artikelregister',
                ENU = 'Item Fast Tab';
}

addbefore(Item)
{
    // area(content)
    // {
    0 references
    group(Desc
    {
        Captio
        0 referen
        field(
        {
            Ap
        }
        0 referen
        field(
        {
            Ap
        }
        0 referen
        field(
        {
            Ap
        }
    }
}
```

```
                {
                    ApplicationArea = All;
                }
            }
        }

    actions
    {
        addlast("S&ales")
        {
            0 references
            action(CCOCurrencies)
            {
                CaptionML = ENU = 'Currencies',
                            DEU = 'Währungen';
                RunObject = page Currencies;
            }
        }
    }
}
```

There might be other ways of carrying out this implementation

HandsOn – Exercise 14 | Page Extension Layout, continued 4

HandsOn



Learn

1) how to influence the layout of a page
2) Page extensions are „executed“ from top to bottom … „the last one wins“

HandsOn – Exercise 15 | Module Hazard Category Part 2

HandsOn

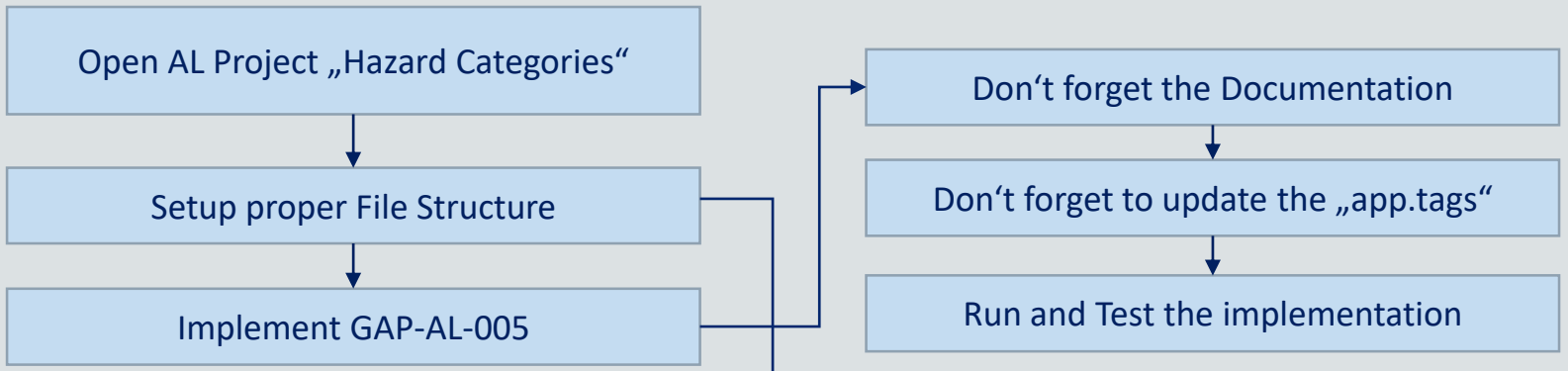
FRD

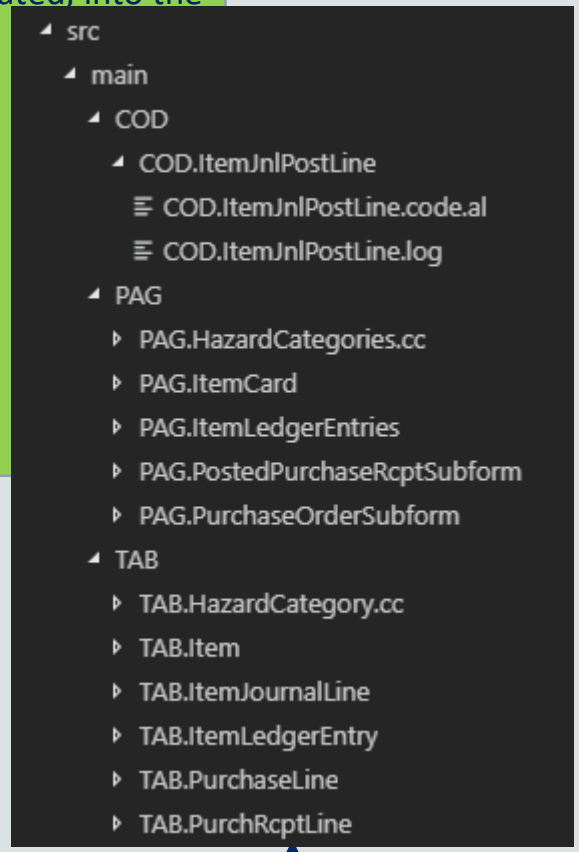
It shall be possible to group Items according to their "Hazard Category". "Hazard Categories" should be freely definable. When an article is purchased via ordering, its "Hazard Category" is to be displayed in the purchasing document and should be changeable there. During the course of item posting, the "Hazard Category" has to be transferred into the "Item Ledger Entries" and, if indicated, into the posted purchase documents.

No other processes are to be respected.
No further details have to be respected concerning, e.g., manipulations of partially posted documents etc..

Remember: New data structure was setup with GAP-AL-005

GAP-AL-005: Hazard Categories – Functionality
Implement new fields in tables “Item”, “Purchase Line”, “Purch. Rcpt. Line”, “Item Journal Line” and “Item Ledger Entry”.
Implement functionality.





HandsOn – Exercise 15 | Module Hazard Category Part 2, continued

HandsOn

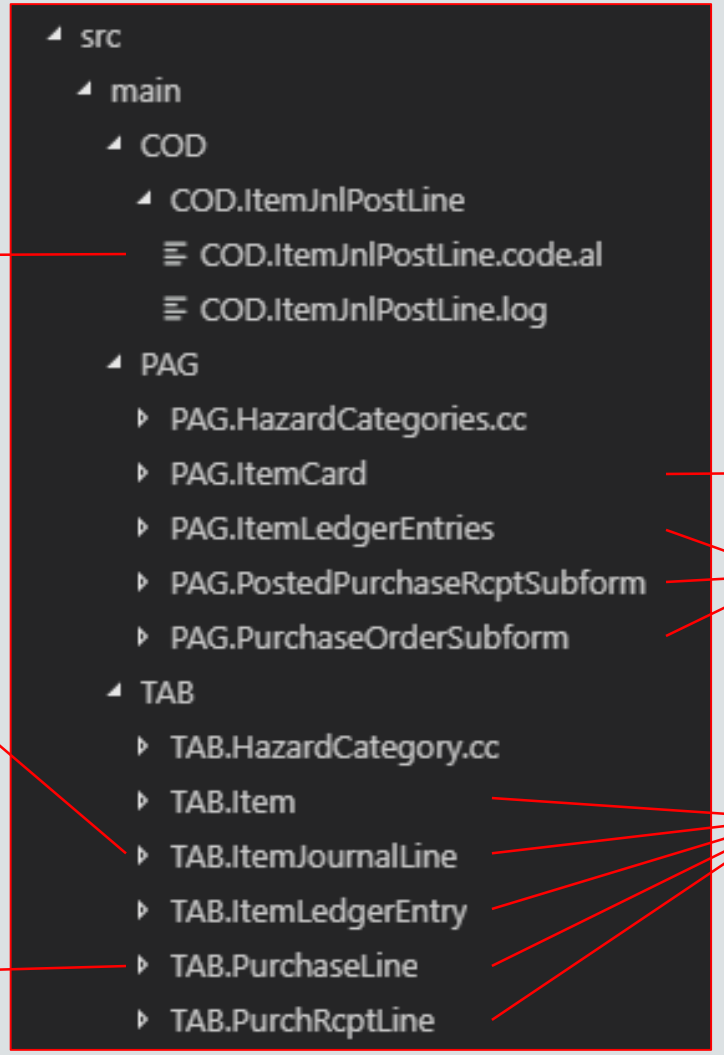
Functionality

Posting:
Transfer of Hazard Category from Jnl Line to Entry
(via *OnAfterInitItemLedgEntry*)

Posting:
Transfer of „Hazard Category“ from Purchase Line to „Jnl Line“
(via *OnAfterCopyItemJnlLineFromPurchLine*)

Preassignment & Consistecy Checks:
- Copy „Hazard Category“ from Item if not yet filled
(via *OnAfterAssignItemValues*)
- Ensure „Hazard Category“ only present with Items …

Involved Objects



Data Structure & GUI

New field „Hazard Category Code“ on Fast Tab „Item“

New field „Hazard Category Code“

New field „Hazard Category Code“ Code[20]

Programming in AL
Object Types: Theory and HandsOn – Enums

- https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-extensible-enums
- Snippet „tenum“
- Naming

File Names	ENU	.	<Enum Name, abbreviated>	.cc	.al
Object Names	CCO	<Enum Name>			

„abbreviated“ — means without spaces …,
as suggested as Variable Name by the Object Designer
„Purchase Header → PurchaseHeader“

- Example

	Object Name	Object File	Object Folder
• New Enum „My new Enum“	„CCO My new Enum“	„ENU.MyNewEnum.cc.al“	„ENU.MyNewEnum.cc“

- Note that all Enum files have a „.cc“ in their names …. all of them are new, always!

Programming in AL
Object Types: Theory and HandsOn – Enums

- Structure

Enum <ID> „<Name>“
{
[Properties]
[Values]
}

One Property only: *Extensible* to rule if particular Enum may be extended

Value(<Unique „No.“ of value>; <Unique Enum Value (Name)>)
{
[Properties]
}

Available Properties: *Caption and CaptionML (runtime 2.3 and later)*

ENU.Classification.cc.al

```
1 reference
enum 80100 "CCO Classification"
{
    Extensible = true;

    1 reference
    value(0; None)
    {
        Caption = '---';
    }
    1 reference
    value(1; Bronze)
    {
        Caption = 'Bronze Customer';
    }
    1 reference
    value(2; Silver)
    {
        Caption = 'Silver Customer';
    }
    1 reference
    value(3; Gold)
    {
        Caption = 'Gold Customer';
    }
}
```

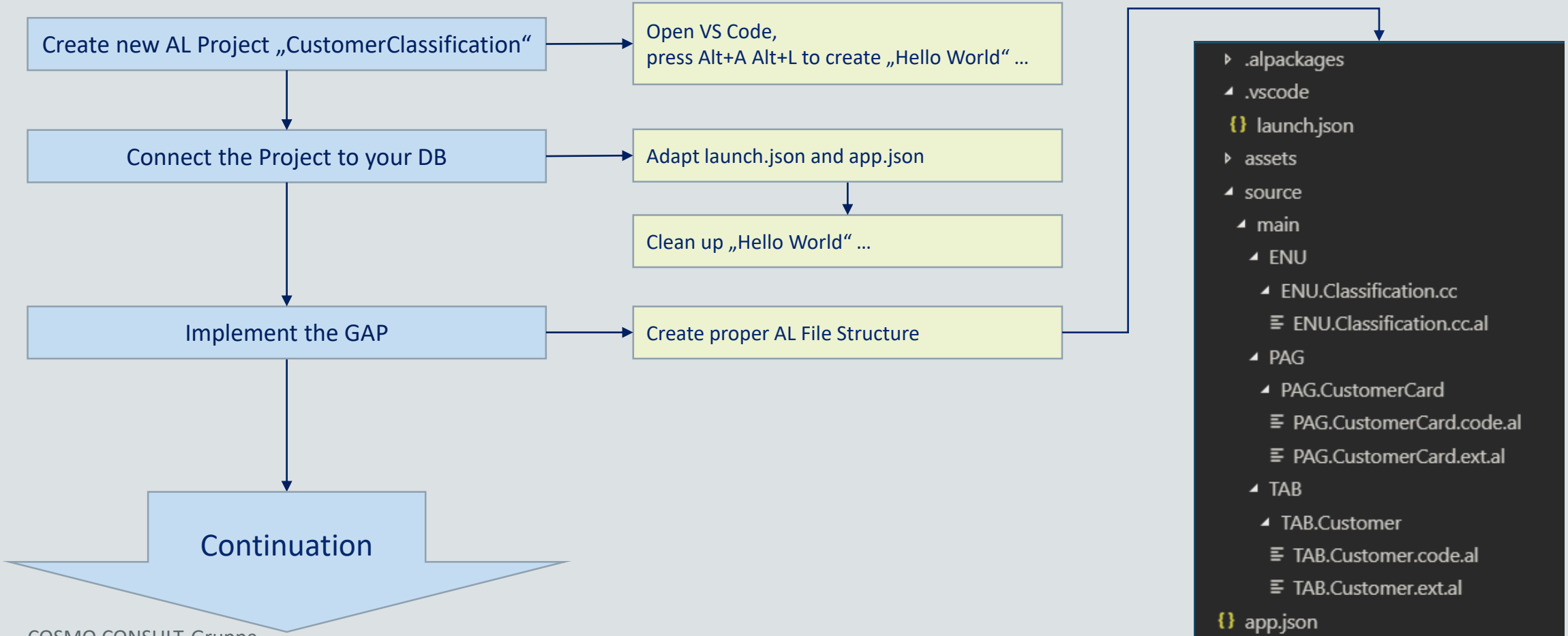
Programming in AL
Object Types: Theory and HandsOn – Enums

- Table Fields / Variables can be of type „Enum“, subtype „Enum ID“
- Enums „replace“ Options
- Syntax like Option Syntax
  - Enum Values are referenced via „<Enum Name>::<Unique Enum Value Name>
  - *Enum Values Names* are to be prefixed
- More and more Enums will enter the standard, more and more will be *extensible*
  - Note that „Extensibility“ here means that any code which „cases“ options must be extensible, too

HandsOn – Exercise 16 | Customer Classification, Enums

HandsOn

GAP-AL-006: Customer Classification
It shall be possible to setup a classification of customers by assigning them one of the „values“ „---“, „Bronze Customer“, „Silver Customer“ or „Gold Customer“. An Action „Suggest Salutation“ in the Customer Action Group shall suggest a salutation depending on the particular classification. The Action starts a respective message „Hello <name>“, „Dear bronze <Name>“, „Dear silver <name>“ or „Dear most revered gold <name>“.



There might be other ways of carrying out this implementation

HandsOn – Exercise 16 | Customer Classification, Enums continued 1

HandsOn

GAP-AL-006: Customer Classification
It shall be possible to setup a classification of customers by assigning them one of the „values“ „---“, „Bronze Customer“, „Silver Customer“ or „Gold Customer“. An Action „Suggest Salutation“ in the Customer Action Group shall suggest a salutation depending on the particular classification. The Action starts a respective message „Hello <name>“, „Dear bronze <Name>“, „Dear silver <name>“ or „Dear most revered gold <name>“.

Implement the GAP

- Create proper AL File Structure
- Create Enum Object
- Create Table Extension of Customer Table with new Classification Field
- Show the field on Customer Card
- Add Action functionality
- Don‘t forget about Documentation (logs and tags)

Run / Test the Project

Learn Enum Syntax

TAB.Customer.ext.al

```
0 references
tableextension 80100 "CCO Customer +" extends Customer
{
    fields
    {
        9 references
        field(80000; "CCO Classification"; enum "CCO Customer Classification")
        {
            CaptionML = ENU = 'Classification',
                        DEU = 'Klassifizierung';
            DataClassification = ToBeClassified;
        }

    }
}
```

ENU.Classification.cc.al

```
1 reference
enum 80100 "CCO Customer Classification"
{
    Extensible = true;

    1 reference
    value(0; None)
    {
        Caption = '---';
    }
    1 reference
    value(1; Bronze)
    {
        Caption = 'Bronze Customer';
    }
    1 reference
    value(2; Silver)
    {
        Caption = 'Silver Customer';
    }
    1 reference
    value(3; Gold)
    {
        Caption = 'Gold Customer';
    }
}
```

There might be other ways of carrying out this implementation

HandsOn – Exercise 16 | Customer Classification, Enums continued 2

HandsOn

GAP-AL-006: Customer Classification : Hints

TAB.Customer.code.al

```
1 reference
codeunit 80100 "CCO T Customer +"
{
    1 reference
    procedure SuggestSalutation(Rec: Record Customer)
    var
        SalutationTxt: Text;
        Handled: Boolean;
        SuggestedSalutationTxt: TextConst ENU = 'Suggested Salutation:\%1',
                                          DEU = 'Vorgeschlagene Anrede:\%1';
    begin
        Handled := true;
        case true of
            Rec."CCO Classification" = Rec."CCO Classification"::None:
                SalutationTxt := 'Hallo %1';
            Rec."CCO Classification" = Rec."CCO Classification"::Bronze:
                SalutationTxt := 'Dear bronze %1';
            Rec."CCO Classification" = Rec."CCO Classification"::Silver:
                SalutationTxt := 'Dear silver %1';
            Rec."CCO Classification" = Rec."CCO Classification"::Gold:
                SalutationTxt := 'Dear most revered gold %1';
            else begin
                    Handled := false;
                    InvokeOnCreateSalutation(Rec, SalutationTxt, Handled);
                end;
        end;
        IF Handled then
            Message(STRSUBSTNO(SuggestedSalutationTxt, STRSUBSTNO(SalutationTxt, Rec.Name)))
        else
            Error('Impossible to suggest Salutation');
    end;

    1 reference
    local procedure InvokeOnCreateSalutation(var Rec: Record Customer; var SalutationTxt: Text; var Handled: Boolean)
    begin
        // Event publisher could be located here, too, but was implemented within the "Customer" (agent) table
        Rec.CCOOnCreateSalutation(Rec, SalutationTxt, Handled);
    end;
}
```

There might be other ways of carrying out this implementation

Programming in AL
Object Types: Theory and HandsOn – Enum Extensions

- https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-extensible-enums
- Snippet „tenum“
- Naming

File Names	ENU	.	<Enum Name, abbreviated>	*.ext*	.al
Object Names	CCO	<Enum Name>	+		

„abbreviated“ — means without spaces …,
as suggested as Variable Name by the Object Designer
„Purchase Header → PurchaseHeader“

- Example

	Object Name	Object File	Object Folder
New Enum Extension „My Enum Extension“	„CCO My Enum Extension +“	„ENU.MyEnumExtension.ext.al“	„ENU.MyEnumExtension“

Programming in AL
Object Types: Theory and HandsOn – Enum Extensions

- Structure

enumextension <ID> „<Name>“ extends <Name of extended Enum>
{

[Values]

}

Value(<Unique „No.“ of value>; <Unique Enum Value (Name)>)
{
[Properties]
}

(Available Properties: *Caption and CaptionML (runtime 2.3 and later)*)

ENU.Classification.ext.al

```
- reference
enumextension 80300 "CCO Classification +" extends "CCO Classification"
{
    1 reference
    value(80300; CCOPlatinum)
    {
        Caption = 'Platinum Customer';
    }
}
```

Programming in AL
Object Types: Theory and HandsOn – Reports

- See workshop **Microsoft AL [Reports]**

Miscellaneous

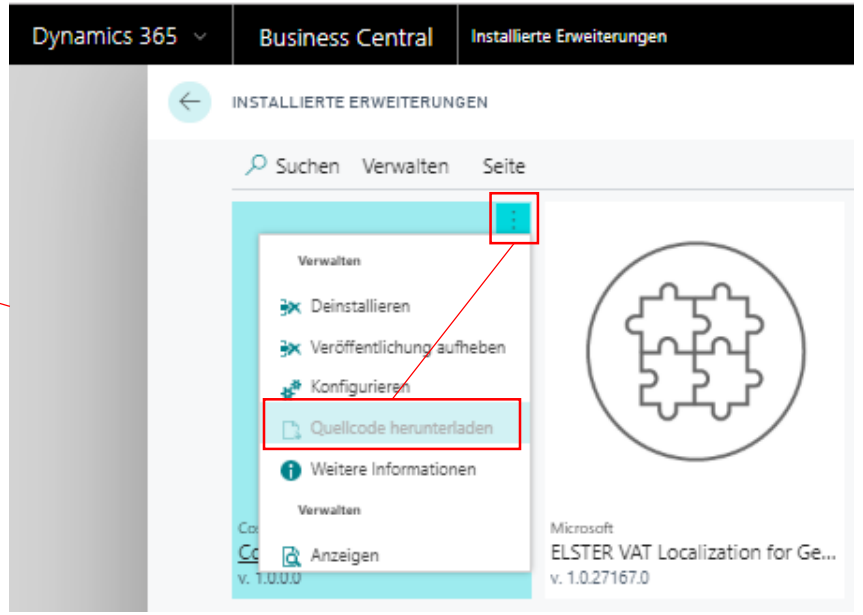
- APP Dependencies
- The “CU Launcher”
- Debugger
- Life Cycle Management
- Language Features
- … to be added …

Miscellaneous
APP Dependencies

- It is possible to extend Extensions
- Dependencies are configured within the app.json file

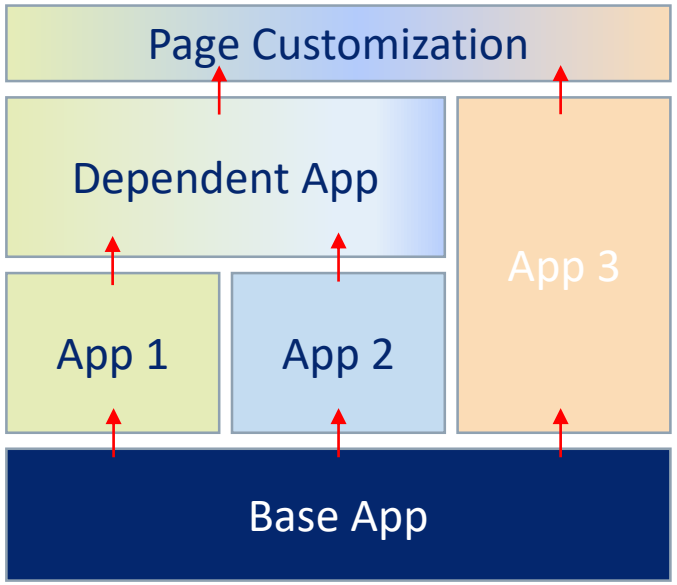
```
{} app.json
13    "capabilities": [],
14    "dependencies": [
15      {
16        "appId": "e1956ed4-fdfe-4b15-a724-13d2e0427049",
17        "name": "Customer Classification",
18        "publisher": "MPro",
19        "version": "1.0.0.0"
20      }
21    ],
```

- Symbols of a additional „root“ app
  - can be downloaded, if „root“ app is present in the target database
  - If available as file, they can be copied directly into the „.alpackages“ directory
  - Sometimes, source code (including symbols) can be downloaded from productive environment



Miscellaneous
APP Dependencies

- „Dependency Graph“



Want to install „modifiction“ of „App 1“

↓

Deinstall „Dependent App“ and „Page Customiztion“ first

HandsOn – Exercise 17 | Extending Customer Classification, Extended Enums

HandsOn

GAP-AL-007: Extended Customer Classification
The Customer Classification system shall be extended by a new category „Platinum Customer“.
The corresponding salutation suggestion shall be „Hallo supreme most revered <name>“

Create new AL Project „CustomerClassificationExtension“

Open VS Code, press Alt+A Alt+L to create „Hello World“ …

Connect the Project to your DB

Adapt launch.json and app.json

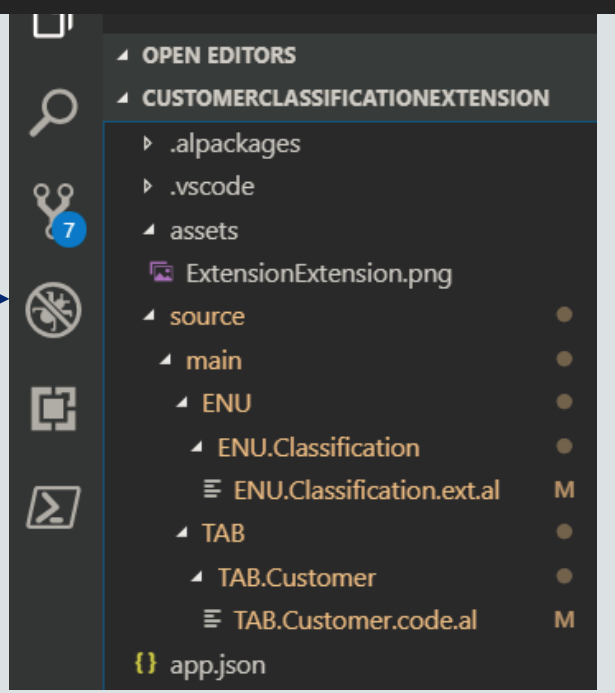
Clean up „Hello World“ …

Define the Dependency via app.json

```
"dependencies": [
  {
    "appId": "e1956ed4-fdfe-4b15-a724-13d2e0427049",
    "name": "Customer Classification",
    "publisher": "MPro",
    "version": "1.0.0.0"
  }
],
```

Implement the GAP

Create proper AL File Structure



Continuation

There might be other ways of carrying out this imple

HandsOn – Exercise 17 | Extending Customer Classification, Extended Enums

HandsOn

GAP-AL-007: Extended Customer Classification
The Customer Classification system shall be extended by a new category „Platinum Customer“.
The corresponding salutation suggestion shall be „Hallo supreme most revered <name>“

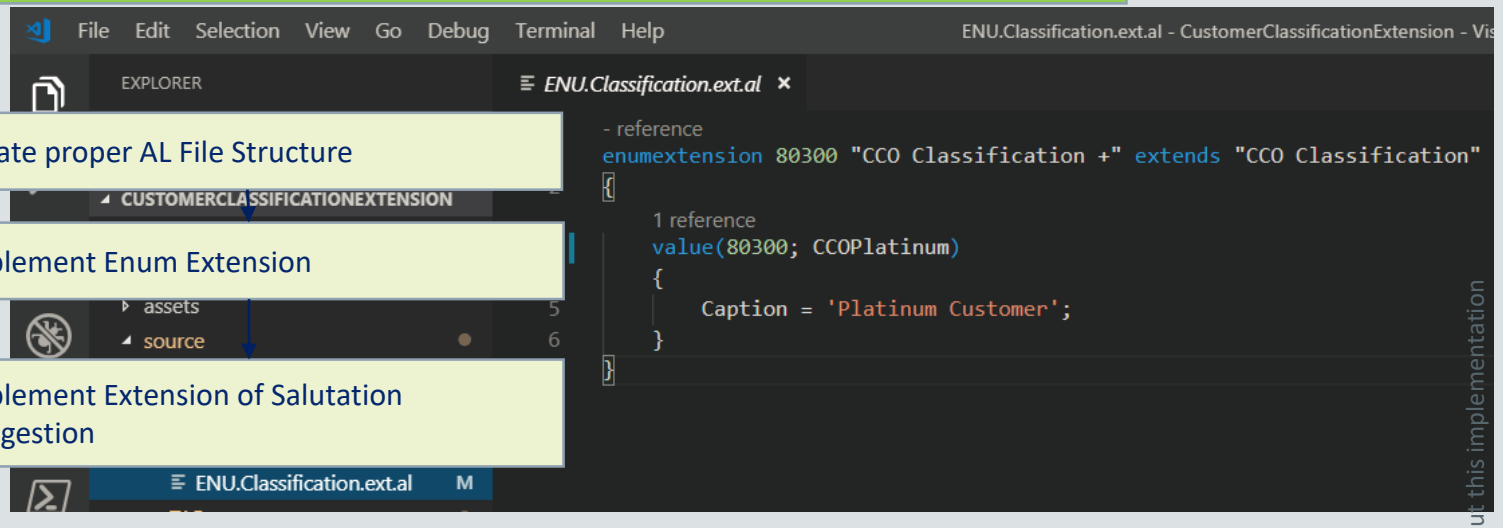
Implement the GAP

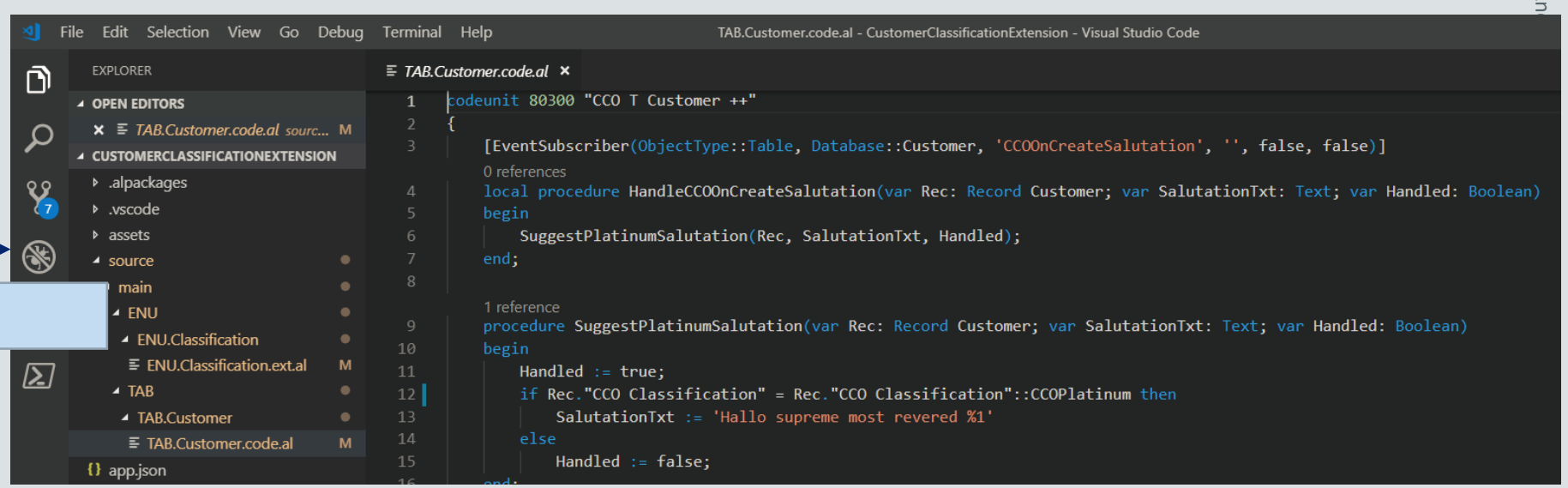
Create proper AL File Structure

Implement Enum Extension

Implement Extension of Salutation Suggestion

Run / Test the Project





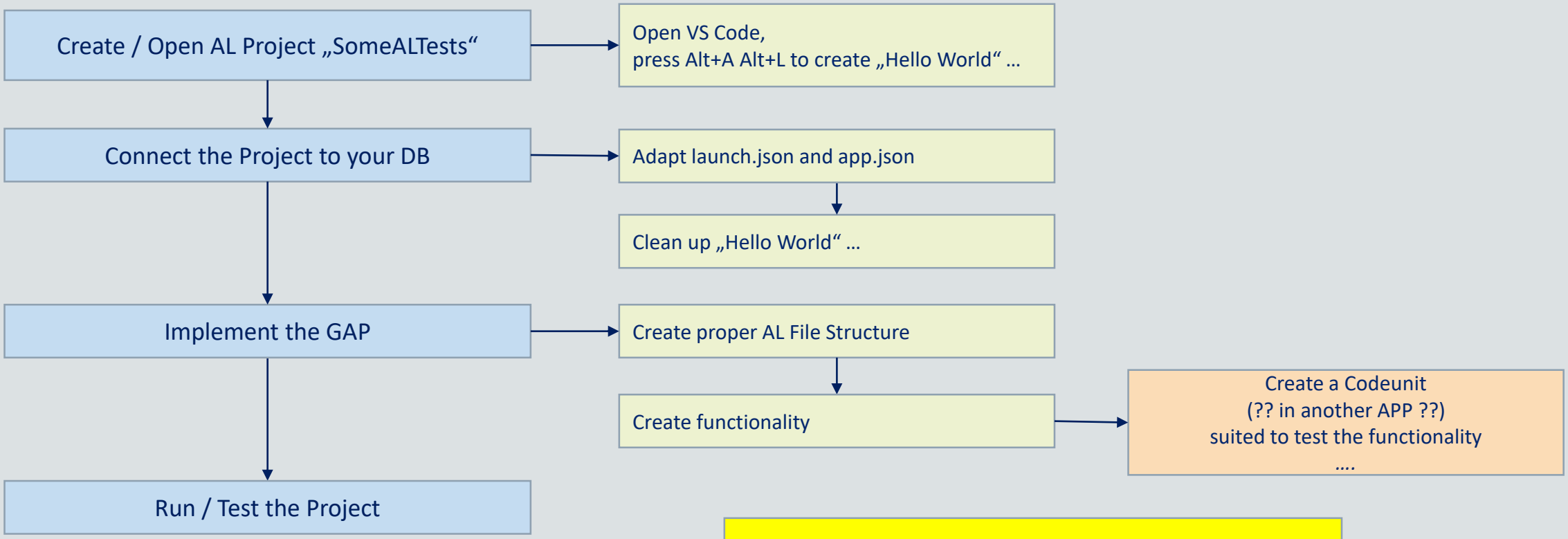
Miscellaneous

- APP Dependencies
- The “CU Launcher”
- Debugger
- Life Cycle Management
- Language Features
- … to be added …

HandsOn – Exercise 18 | Tools: CU Launcher

HandsOn

GAP-AL-008: CU Launcher
A functionality is to be implemented, which allows to start particular codeunits via GUI.
Design
- Page „CU Launcher“
  - with control „CU ID to launch“
  - Action „Launch CU“ which launches CU (after proper confirmation)



Learn: Objects can be „cross-app“ adressed by ID

There might be other ways of carrying out this implementation

Miscellaneous

- APP Dependencies
- The “CU Launcher”
- Debugger
- Life Cycle Management
- Language Features
- … to be added …

Miscellaneous
Debugging

- Available „OnPremise“ and within docker sandbox containers of cloud systems (...)
- not yet really available on cloud sandbox systems (nor on cloud production systems)
- Hit F9 in VS Code to define Breakpoins, F5 to start debugging
- Launch.json setting: „breakOnError“, „breakOnRecordWrite“
- Remark

> ⓘ **Important**
>
> To enable debugging the `NetFx40_LegacySecurityPolicy` setting in the Microsoft.Dynamics.Nav.Server.exe.config file must be set to **false**. This requires a server restart.

There are a number of limitations to be aware of:

- "External code" can only be debugged if the code has the `showMyCode` flag set. For more information, see Security Setting and IP Protection.
- Not all AL types yet show helpful debugging.
- The debugger launches a new client instance each time you press F5. If you close the debugging session, and then start a new session, this new session will rely on a new client instance. We recommend that you close the Web client instances when you close a debugging session.

Miscellaneous

- APP Dependencies
- The “CU Launcher”
- Debugger
- Life Cycle Management
- Language Features
- … to be added …

Miscellaneous
Life Cycle Management

- Installing, Upgrading, …
- To be dealt with in a separate workshop (*07-Microsoft-AL-Application-Lifecycle*)
- https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-app-life-cycle

Miscellaneous

- APP Dependencies
- The “CU Launcher”
- Debugger
- Life Cycle Management
- Language Features
- … to be added …

Miscellaneous
Language Features

- For complete list, see https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/methods-auto/library
- Examples
  - List Data Type
    - provides arranged list of elements of „simple data type“
    - Example: MyTextList: List of [Text]
    - „Array-like“ but not „bound by declaration“
    - Comes with bunch of methods to manipulate lists
  - Text Builder Data Type
    - “Wrapper” for the .Net implementation of StringBuilder.
  - Json-”…” Data Types
    - JsonArray, JsonObject, JsonToken, JsonValue
    - Come with bunch of methods to handle / manipulate Json files
  - „Lots of“ Xml-”…” Data Types
    - Come with bunch of methods concerning xml handling

Miscellaneous

- APP Dependencies
- The “CU Launcher”
- Debugger
- Life Cycle Management
- Language Features
- … to be added …

Miscellaneous
…to be added …

- Notes on
  - Permission Sets

To export permission sets using Visual Studio Code

1. In Visual Studio Code, open your extension.
2. Create extension with some objects like Page, Report, Table, Query, Codeunit, or XmlPort.
3. Open the command palette using the `Ctrl+Shift+P` keys and select the **AL: Generate permission set containing current extension objects** command.

   > ⓘ Note
   >
   > If you do this repeatedly, Visual Studio Code will probe for overwriting the file, there is no support for merging manual corrections into newly generated content.

4. Publish the app.

Now, you have the XML file with default permissions to all your objects.

  - DataType „Dictionary“
  - HowTo handle xlifs

    Store in „agent folder“, automatically collect in central folder during build process
  - HowTo handle layouts
    - Store in „agent folder“, automatically copy to central folder during build process

Miscellaneous
Code Runner in “Test” Apps

```
codeunit 90080 "Client Code Runner"
{
    // OnCompanyOpen, code placed in fct. RunCode is executed
    //   may be used for running GUI independent code for test purposes
    [EventSubscriber(ObjectType::Codeunit, Codeunit::LogInManagement, 'OnAfterCompanyOpen', '', true, true)]
    0 references
    local procedure HandleOnAfterCompanyOpen()
    begin
        If not (CurrentClientType() in [ClientType::Windows, ClientType::Web, ClientType::Desktop, ClientType::Tablet, ClientType::Phone]) then
            exit;
        RunCode();
    end;

    1 reference
    local procedure RunCode()
    begin
        //  place your Code to run
        Message('OnAfterCompanyOpen');
    end;
}
```

Q & A

AL

Getting Started with AL

Competence Team DevTech

Microsoft AL