DYALOG

APL Germany

Abenteuerliche Texte

Michael Baas (mbaas@dyalog.com)

Inhaltsverzeichnis

Einleitung
oder "Here we go again!"

- Thema meines ersten Vortrags bei APL Germany hatte auch schon mit Literatur zu tun!
- Literal programming (Knuth) und APL

1. Kapitel

DYALOG

APL Germany

Ist jetzt alles Text?

oder

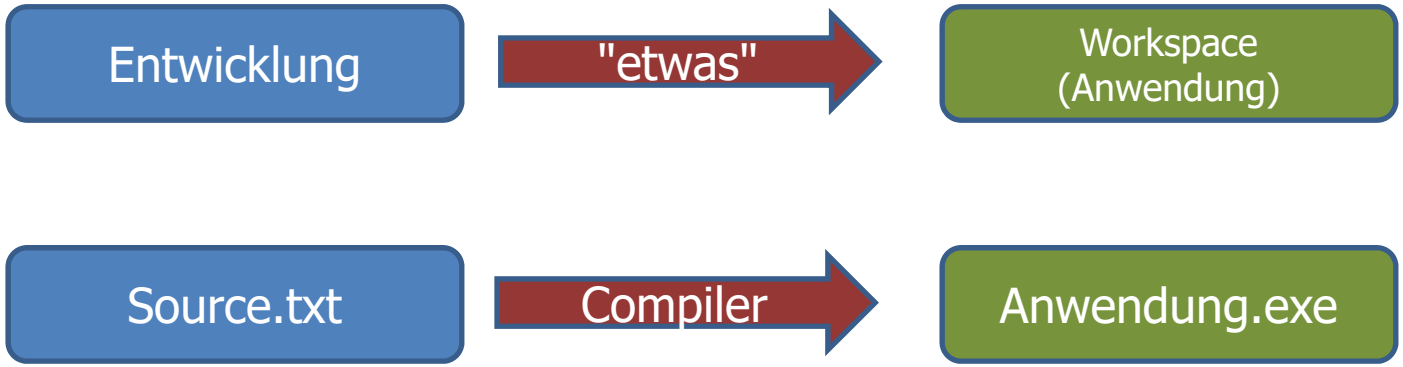
```
)SAVE the workspace!
```

Motivation

- Die "neue Welt" (Source-Code in Dateien) ist ideal für Entwickler
- …aber wie verteilt man die Software?
- …oder schützt sie vor Änderungen durch experimentierfreudige Nutzer?

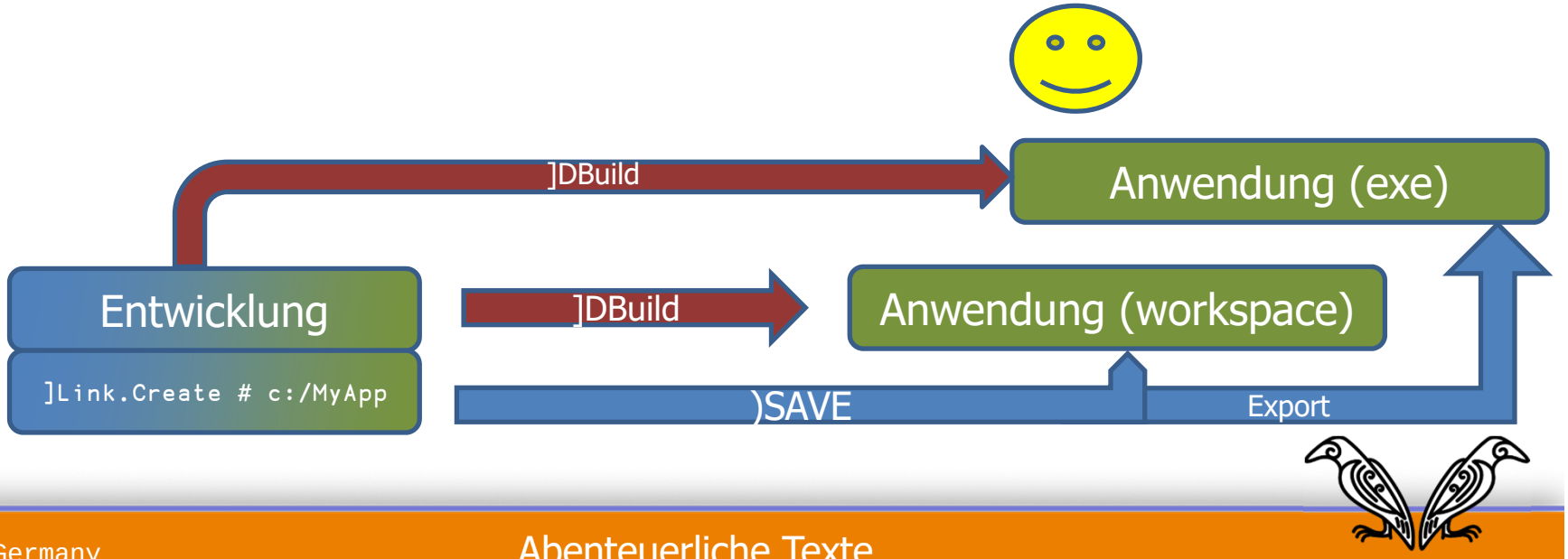
Workspaces

- Sind also nicht "altmodisch" sondern haben ihre Berechtigung.



Workspaces

- Geben wir damit die Interaktivität von APL auf?



Beispiel: DUI

Name	Ext	Size	Modified	Created
.git			30.04.2021 17:34:09	04.06.2020 09:50:49
CommonPages			04.06.2020 09:51:26	04.06.2020 09:51:26
Config			14.01.2021 16:31:34	04.06.2020 09:51:26
Core			05.11.2020 12:24:37	04.06.2020 09:51:26
Documentation Sources			04.06.2020 09:51:26	04.06.2020 09:51:26
Extensions			24.09.2020 09:45:43	04.06.2020 09:51:26
HTML			04.06.2020 09:51:26	04.06.2020 09:51:26
Loadable			04.06.2020 09:51:26	04.06.2020 09:51:26
Logs			21.08.2020 11:27:21	21.08.2020 11:27:21
MS3			14.01.2021 16:31:34	04.06.2020 09:51:26
PlugIns			04.06.2020 09:51:27	04.06.2020 09:51:27
QA			29.04.2021 12:40:47	04.06.2020 09:51:28
Utils			05.11.2020 12:24:37	04.06.2020 09:51:28
wsSource			29.04.2021 12:31:50	04.06.2020 09:51:28
.gitattributes	gitattribut...	1 KB	04.06.2020 09:51:26	04.06.2020 09:51:26
.gitignore	gitignore	1 KB	04.06.2020 09:51:26	04.06.2020 09:51:26
DUI.dws	dws	60 KB	29.04.2021 12:31:50	18.01.2021 19:01:12
DUI.dyalog	dyalog	31 KB	30.11.2020 08:54:35	30.11.2020 08:54:35
LICENCE		2 KB	04.06.2020 09:51:26	04.06.2020 09:51:26
README.md	md	1 KB	04.06.2020 09:51:28	04.06.2020 09:51:28
version		1 KB	04.06.2020 09:51:28	04.06.2020 09:51:28

Name	Ext	Size
DUI.dyalogbuild	dyalogbuild	1 KB
DUI.log	log	1 KB
lx.dyalog	dyalog	2 KB
RuntimeError.dyalog	dyalog	2 KB
Stop.dyalog	dyalog	1 KB

Beispiel: DUI

```
DyalogBuild: 1.24
ID          : DUI, Version=1.1
Description: Dyalog User Interface
Defaults    : ⎕IO←⎕ML←1
TARGET      : ../DUI.dws,save=1,off=1

APL    : *.dyalog, Target=#
APL    : ../Utils/Strings.dyalog, Target=#
⍝ lx needs Strings.lc!
LX     : lx
```

Name	Ext	Size
DUI.dyalogbuild	dyalogbuild	1 KB
DUI.log	log	1 KB
lx.dyalog	dyalog	2 KB
RuntimeError.dyalog	dyalog	2 KB
Stop.dyalog	dyalog	1 KB

Praxis – mit stats

DBuild: Zusammenfassung

-]DBuild {name}.dyalogbuild
- oder, falls in eigene CI-Tools integriert:
  dyalog.exe `lx="⎕SE.UCMD'DBuild <curitem>'"`
  (nur von Unicode-Interpretern unterstützt!)
- Doku unter https://github.com/Dyalog/DBuildTest/wiki
- Kann geladen werden von
  https://github.com/Dyalog/DBuildTest

2. Kapitel

Läuft!

oder

]DTest

DTest

"Ein Modultest (auch von englisch unit test als Unittest oder als Komponententest bezeichnet) wird in der Softwareentwicklung angewendet, um die funktionalen Einzelteile (Units) von Computerprogrammen zu testen, d. h., sie auf korrekte Funktionalität zu prüfen."

Kein Test der gesamten Anwendung, sondern einzelner Funktionen

Conga

```
r←test_multiroot dummy;ret;z;c;iC2;data
⍝ Basic test using different roots for Client and Server

r←''
→(9.1=⎕NC⊂'iConga')⍴0 ⍝ Bypass this test if iConga is the v2 compatibility NS

data←'hello'
:Trap 0
    iC2←#.Conga.Init'R2'
:Else
    →fail⊣r←'Conga.Init of 2nd root failed: ',⊃⎕DMX.DM
:EndTrap

:If 0≠⊃ret←iConga.Srv'S1' '' 5000 'Text'
    →fail⊣r←'Srv failed: ',,⍕ret ⋄ :EndIf
:If 0≠⊃ret←iC2.Clt'C1' '127.0.0.1' 5000 'Text'
    →fail⊣r←'Clt failed: ',,⍕ret ⋄ :EndIf
:If (0 'Connect')≠(⊂1 3)⌷ret←iConga.Wait'S1' 10000
    →fail⊣r←'Wait for Connect failed: ',,⍕ret ⋄ :EndIf
:If 0≠⊃ret←iC2.Send'C1'data
    →fail⊣r←'Sent failed: ',,⍕ret ⋄ :EndIf
:If (0 'Block')≠(⊂1 3)⌷c←iConga.Wait'S1' 5000
    →fail⊣r←'Wait for Srv.Block/BlockLast failed: ',,⍕c ⋄ :EndIf
:If 0≠⊃ret←iConga.Send(2⊃c)(⍕⍬4⊃c)
    →fail⊣r←'Respond failed: ',,⍕ret ⋄ :EndIf
...
fail:

:Trap 0
    z←iConga.Close'S1'
    z←iC2.Close'C1'
:EndTrap
```

- r← Ergebnis enthält Text, der die beobachteten Fehler beschreibt.
- →fail beendet den Test

DSL

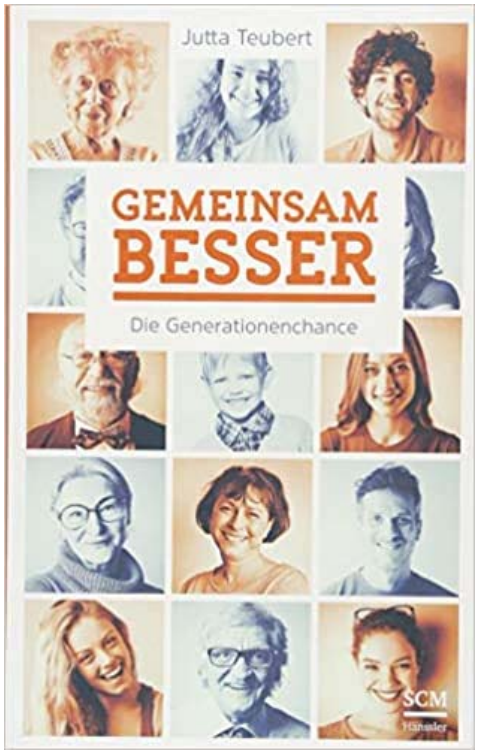
```
:If 0≠⊃ret←iConga.Srv'S1' '' 5000 'Text'
      →fail⊣r←'Srv failed: ',,⍕ret ⋄ :EndIf

:If 0 Check ⊃ret←iConga.Srv'S1' '' 5000 'Text'
     →fail Because'Srv failed: ',,⍕ret ⋄ :EndIf

]DTest Tests/conga -halt
```

Weitere Möglichkeiten

- setups & teardown
- DSL: IsNotElement
- returncode 20/21 + log-Dateien
- [context] RandomVal max[,count]
- https://github.com/Dyalog/DBuildTest
- nicht beschränkt auf Funktionstests

```
:For setup :in Setups
   ⍎setup
   :For test :in Tests
       ⍎test
   :endfor
:endfor
:for t :in teardowns
  ⍎t
:endfor
```

Praxis



3. Kapitel

Konzertierte Aktion

oder

CITA

CITA – Continous Integration Tool for APL

- Testen ist wichtig!
- Unter CI ist Testen noch wichtiger!
- "Läuft's?" wird zu "Läuft's immer und überall?"
  - auch unter Linux?
  - mit Classic Edition?
  - mit v17?

Not macht erfinderisch

- Windows
- Linux
- MacOS
- Raspberry Pi
- AIX
- 32bit / 64bit
- Classic oder Unicode

Das Orchester

CITA.json5 im Repository

```
{
    Tests: [{
        "DyalogVersions": "12.1+",
        "Test": "./Tests/DTest.dyalogtest",
        "EMail": [{
            "All": "mbaas@dyalog.com"
        }],
        "TacitVersion": ["commit", "latest"],
    }, ]
}
```

Famous last words

- .json
- .dcfg

…und so testen sie noch fröhlich viele Jahre und konnten viele schlimme Fehler beseitigen.