24. November 2022 | **ADWL Mainz**

ITUG Mitgliederversammlung

itug

TUSCRIPT

EINE EINFÜHRUNG

Slides: **https://itug.github.io/ITUG2022_workshop**

Hans-Werner Bartz | **Thomas Kollatz** | 🐦 **@ITUGeV** | ⬛ **ITUG** | **CC-BY 4.0**

TUSCRIPT

ausführen

```
Gib Kommando >#ma,skriptdatei
```

```
Gib Anweisung >x ,#ma,skriptdatei
```

```
Gib Anweisung >x,   #ma,<editor>
```

```
y,*=execute
y,execute=clr_cmd_line,"x #ma,<editor>",confirm
```

```
Gib Kommando >$meinSkript
```

Skript ist Segment in einer mit `#de,segmentdatei` definierten Segmentdatei und beginnt mit `$$!`

TUSCRIPT

Skriptsprache

```
1  $$ PRINT "Hello world"
```

MODE

itug

TUSCRIPT – DATA – STATEMENT

```
 1  $$ MODE TUSCRIPT,{}
 2  var="Hello world"
 3  PRINT var
 4
 5  MODE DATA
 6  $$ var_stern=*
 7  Hello
 8  world
 9
10  $$ MODE STATEMENT
11  TRACE *var_stern
```

```
 1  $$ MODE TUSCRIPT,{}
 2  var="Hello world"
 3  PRINT var
 4
 5
 6  var_stern=*
 7  DATA Hello
 8  DATA world
 9
10
11  TRACE *var_stern
```

ARBEITEN MIT DATEIEN
CREATE

```
1 $$ MODE TUSCRIPT,{}
2 file="datei.tf"
3 status=CREATE (file,SEQ-o,-std-)
4
5 IF (status!="OK") THEN
6   PRINT status
7   STOP
8 ENDIF
```

```
1 $$ MODE TUSCRIPT,{}
2 file="datei.tf"
3 ERROR/STOP CREATE (file,SEQ-o,-std-)
```

ARBEITEN MIT DATEIEN
anmelden | kopieren | abmelden | umbenennen

```
 1 $$ MODE TUSCRIPT,{}
 2 file="datei.tf"
 3 -- anmelden READ | WRITE
 4 ERROR/STOP OPEN (file,READ,-std-)
 5
 6 -- anlegen SEQ | FDF | RAN ; -o | -t | -p
 7 file_dubble="datei_zwei.tf"
 8 ERROR/STOP CREATE (file_dubble,SEQ-o,-std-)
 9
10 -- kopieren
11 ERROR/STOP COPY (file,file_dubble)
12
13 -- schliessen
14 ERROR/STOP CLOSE (file)
15
16 -- umbenennen
17 old_name=VALUE(file_dubble)
18 new_name="datei_kopie.tf"
19 ERROR/STOP RENAME (old_name,new_name)
```

Variableninhalt in Datei schreiben – Dateien vergleichen

```
1  $$ MODE TUSCRIPT,{}
2  file="datei.tf"
3  -- (zum Schreiben) anmelden
4  ERROR/STOP OPEN (file,WRITE,-std-)
5
6  MODE DATA
7  $$ content=*
8  Es gibt nichts Gutes,
9  ausser man tut es.
10
11 $$ MODE STATEMENT
12 FILE/ERASE $file = content
13
14 file_new="datei_kopie.tf"
15
16
17 content_new=EXCHANGE (content,"|ss|ß|")
18
19 FILE/ERASE $file_new = content_new
20
21 status=COMPARE (file,file_new)
22 TRACE *status
```

```
BUILD X_TABLE ss2buckels=*
DATA |ss|ß|
content_new=EXCHANGE(content,ss2buckels)
RELEASE X_TABLE ssbuckels
```

Dateiinhalt in Variable – Variableninhalt vergleichen

```
 1  $$ MODE TUSCRIPT,{}
 2  file_a="datei.tf"
 3  -- (zum Lesen) anmelden
 4  ERROR/STOP OPEN (file_a,READ,-std-)
 5
 6  content_a=FILE (file_a)
 7  TRACE *content_a
 8
 9  file_b="datei_kopie.tf"
10  ERROR/STOP open (file_b,READ,-std-)
11
12  content_b=FILE (file_b)
13  TRACE *content_b
14
15  TRACE
16  x=FIND_DIFF (content_a,0,0,content_b,0,0,char_a,char_b,position)
```

ARBEITEN MIT DATEIEN
Dateiinhalte auf Variablen lesen

```
 1  $$ MODE TUSCRIPT,{}
 2
 3  -- Belegung der Systemvariable TUSTEP_DSK abfragen
 4  FETCH dsk = TUSTEP_DSK
 5
 6  -- Pfad zur Datei
 7  file="datei.tf"
 8  path2file=ADJUST_PATH (dsk,file)
 9  TRACE *path2file
10
11  -- Dateiinhalt (ohne anmelden) auf Variable legen
12  status=READ (path2file,content)
13  TRACE *content
```

ARBEITEN MIT DATEIEN

WWW-Content auf Variable lesen – REQUEST

```
 1  $$ MODE TUSCRIPT,{}
 2
 3  url="http://www.itug.de"
 4  daten=REQUEST (url)
 5  daten=DECODE (daten,utf8)
 6
 7  ERROR/STOP CREATE ("daten.tf",seq-o,-std-)
 8  path2tf=ADJUST_PATH (dsk,"daten.tf")
 9
10  -- Variableninhalt (ohne anmelden) in Datei schreiben
11  ERROR/STOP WRITE (path2tf,daten)
```

ARBEITEN MIT DATEIEN

Variableninhalt in Datei schreiben

```
 1  $$ MODE TUSCRIPT,{}
 2
 3  -- Variableninhalt von TUSTEP-Datei auf Variable einlesen
 4  tf_file="datei.tf"
 5  FETCH dsk = TUSTEP_DSK
 6  path2tf=ADJUST_PATH (dsk,tf_file)
 7  ERROR/STOP READ (path2tf,daten)
 8  TRACE *daten
 9
10  -- Variableninhalt in Fremddatei schreiben
11  txt_file="datei.txt"
12  ERROR/STOP CREATE (txt_file,FDF-o,-std-)
13  path2txt=ADJUST_PATH (dsk,txt_file)
14  ERROR/STOP WRITE (path2txt,daten,UTF8)
```

ORGANISATORISCHES
LOOP

```
1  $$ MODE TUSCRIPT,{}
2  LOOP n=1 to 9
3  PRINT n
4  ENDLOOP
```

```
1  $$ MODE TUSCRIPT,{}
2  filebasename="datei_"

3  LOOP n
4   IF (n>9) EXIT
5   n=CENTER (n,+2,"0")
6   file=CONCAT (filebasename,n,".tf")
7   ERROR/STOP CREATE (file,SEQ-o,-std-)
8  ENDLOOP
9  PRINT n
```

ORGANISATORISCHES

FILE_NAMES

```
1  $$ MODE TUSCRIPT,{}
2  TRACE
3  files=FILE_NAMES (-,-std-)
4  sz_files_1=SIZE(files)
5
6  FETCH dsk = TUSTEP_DSK
7  files=FILE_NAMES (-,$dsk)
8  sz_files_2=SIZE(files)
```

ORGANISATORISCHES
LOOP

```
1 $$ MODE TUSCRIPT,{}
2 LOOP n=1 to 9
3 PRINT n
4 ENDLOOP
```

```
1 $$ MODE TUSCRIPT,{}
2 filebasename="datei_"

3 LOOP n
4  IF (n>9) EXIT
5  n=CENTER (n,+2,"0")
6  file=CONCAT (filebasename,n,".tf")
7  ERROR/STOP CREATE (file,SEQ-o,-std-)
8 ENDLOOP
9 PRINT n
```

ORGANISATORISCHES
SELECT

```
 1 $$ MODE TUSCRIPT,{},MAC
 2 FETCH dsk = TUSTEP_DSK
 3 files=FILE_NAMES (-,$dsk)
 4 sz_files_all=SIZE(files)
 5
 6 BUILD S_TABLE tf=*
 7 DATA |*.tf|
 8
 9 tf_files=SELECT (files,tf)
10 RELEASE S_TABLE tf
11
12 sz_files_tf=SIZE(tf_files)
13
14 -- Bedingung .ne. | != | .eq. | == | .gt. | > | .lt. | <
15 IF (sz_files_all!=sz_files_tf) TRACE *sz_files_all,sz_files_tf
16
17 LOOP n,file=tf_files
18  n=CENTER (n,+3,"0")
19  PRINT/COLOR:9E n," ",file
20 ENDLOOP
```

ORGANISATORISCHES

itug

FILTER | FILTER_INDEX

```
 1 $$ MODE TUSCRIPT,{},MAC
 2 FETCH dsk = TUSTEP_DSK
 3 files=FILE_NAMES (-,$dsk)
 4
 5 BUILD R_TABLE/or tf=*
 6 DATA |*.tf|
 7 DATA |*.tus|*.tstp|
 8
 9 tf_files=FILTER       (files,tf,-)
10 tf_index=FILTER_INDEX (files,tf,-)
11 RELEASE R_TABLE tf
12
13 sz_files_tf=SIZE(tf_files)
14
15 TRACE *sz_files_tf,tf_index
16
17 LOOP n,index=tf_index
18  n=CENTER (n,+3,"0")
19  file=SELECT (files,#index)
20  PRINT/COLOR:E9 n," ",file
21 ENDLOOP
```

modified | TIME_INTERVAL

```
 1  $$ MODE TUSCRIPT,{},MAC
 2  FETCH dsk = TUSTEP_DSK
 3  files=FILE_NAMES (-,$dsk)
 4
 5  time=time(0)
 6  PRINT time
 7
 8  BUILD R_TABLE tf="|*.tf|"
 9  tf_files=FILTER  (files,tf,-)
10  RELEASE R_TABLE tf
11
12  sz_tf_files=SIZE (tf_files)
13
14  n=0
15  LOOP file=tf_files
16   ERROR/STOP OPEN (file,READ,-std-)
17   modified=MODIFIED (file)
18   interval=TIME_INTERVAL (hours,time,modified)
19   IF (interval>1) THEN
20    n=n+1
21    PRINT/COLOR:E9 n," ",file
22   ENDIF
23  ENDLOOP
24  PRINT/COLOR:0F "Ergebnis: "
25  PRINT/COLOR:0F n," tf-Dateien von ",sz_tf_files," sind älter als eine Stunde."
```

```
 1  $$ MODE TUSCRIPT,{},MAC
 2  MODE DATA
 3  $$ xml=*
 4  <?xml version="1.0" encoding="UTF-8"?>
 5  <?xml-model href="http://www.tei-c.org/release/xml/tei/custom/schema/relaxng/tei_lite
 6  <?xml-model href="http://www.tei-c.org/release/xml/tei/custom/schema/relaxng/tei_lite
 7        schematypens="http://purl.oclc.org/dsdl/schematron"?>
 8  <TEI xmlns="http://www.tei-c.org/ns/1.0">
 9   <teiHeader>
10   <fileDesc><titleStmt><title>Test</title></titleStmt>
11   <publicationStmt><p>intern</p></publicationStmt>
12   <sourceDesc>
13   <p>Übungsdatei für den TUSCRIPT workshop</p>
14   </sourceDesc></fileDesc>
15   </teiHeader>
16   <text><body>
17   <div type="letter">
18   <opener><salute hand="#TK">Herzlich Willkommen</salute></opener>
19   <p>Es freut mich, dass <lb/><choice><del>ihr</del><corr>Sie</corr></choice> so <lb/>
20   </div></body></text></TEI>
21  $$ MODE STATEMENT
22  ERROR/STOP CREATE ("test.xml",FDF-o,-std-)
23  FILE/ERASE/UTF8 "test.xml" = xml
```

XML-ZUGRIFF
Elemente und Attribute

```
 1 $$ MODE TUSCRIPT,{},MAC
 2 FETCH dsk=TUSTEP_DSK
 3 file="test.xml"
 4 path2file=ADJUST_PATH (dsk,file)
 5 ERROR/STOP READ (path2file,daten,UTF8)
 6 -- TRACE *daten
 7 ACCESS q: READ/VARIABLE/STREAM daten s,a+t+e,typ,stack,stack_num
 8 LOOP
 9  READ/EXIT q
10  IF (a.hn."salute") PRINT/COLOR:0F t
11  IF (a.ha."hand") THEN
12   handvalue=GET_ATTRIBUTE (a,"hand")
13   PRINT/COLOR:0F handvalue
14  ENDIF
15 ENDLOOP
16 ENDACCESS q
```

XML-ZUGRIFF
Leere Elemente sammeln

```
 1  $$ MODE TUSCRIPT,{},MAC
 2  FETCH dsk=TUSTEP_DSK
 3  file="test.xml"
 4  path2file=ADJUST_PATH (dsk,file)
 5  ERROR/STOP READ (path2file,daten,UTF8)
 6
 7  ACCESS q: READ/VARIABLE/STREAM daten s,a+t+e,typ,stack,stack_num
 8  LOOP
 9   READ/EXIT q
10   IF (typ==4) THEN
11     tagname=GET_TAG_NAME (t)
12     PRINT tagname
13   ENDIF
14  ENDLOOP
15  ENDACCESS q
```

XML-ZUGRIFF
Tags in //text sammeln

```
 1 $$ MODE TUSCRIPT,{},MAC
 2 FETCH dsk=TUSTEP_DSK
 3 file="test.xml"
 4 path2file=ADJUST_PATH (dsk,file)
 5 ERROR/STOP READ (path2file,daten,UTF8)
 6 tags=*
 7 ACCESS q: READ/VARIABLE/STREAM daten s,a+t+e,typ,stack,stack_num
 8 LOOP
 9  READ/EXIT q
10  IF (stack.ct."|<text>|") THEN
11    IF (typ==4) THEN
12      tagname=GET_TAG_NAME (t)
13    ELSE
14      tagname=GET_TAG_NAME (a)
15    ENDIF
16    IF (tagname!="") tags=APPEND(tags,tagname)
17  ENDIF
18 ENDLOOP
19 ENDACCESS q
20 TRACE *tags
```

Tags in //text in DICTIONARY sammeln

```
 1 $$ MODE TUSCRIPT,{},MAC
 2 FETCH dsk=TUSTEP_DSK
 3 file="test.xml"
 4 path2file=ADJUST_PATH (dsk,file)
 5 ERROR/STOP READ (path2file,daten,UTF8)
 6 DICT tags CREATE
 7 ACCESS q: READ/VARIABLE/STREAM daten s,a+t+e,typ,stack,stack_num
 8 LOOP
 9  READ/EXIT q
10  IF (stack.ct."|<text>|") THEN
11    IF (typ==4) THEN
12     tagname=GET_TAG_NAME (t)
13    ELSE
14     tagname=GET_TAG_NAME (a)
15    ENDIF
16    IF (tagname!="") DICT tags ADD/QUIET/COUNT tagname
17  ENDIF
18 ENDLOOP
19 ENDACCESS q
20 DICT tags UNLOAD tagnames,num,cnt
21 DICT tags SIZE anzahl
22 PRINT anzahl, " tags"
23 tag_count=JOIN (tagnames," ",cnt)
24 TRACE *tag_count
```

XML

TAGS_INDENT

```
 1 $$ MODE TUSCRIPT,{},MAC
 2 FETCH dsk=TUSTEP_DSK
 3 file="test.xml"
 4 path2file=ADJUST_PATH (dsk,file)
 5 ERROR/STOP READ (path2file,daten,UTF8)
 6
 7 daten=JOIN(daten,"")
 8
 9 ASK "Sollen lb ignoriert werden?  a/nein":antwort= ""
10
11 IF (antwort.ab."nein") THEN
12  daten=INDENT_TAGS (daten,-,"    ")
13 ELSE
14  BUILD R_TABLE/TEXT/OR ign ="|*<lb>|"
15  daten=INDENT_TAGS (daten,ign,"    ")
16 ENDIF
17
18 LOOP/CLEAR line=daten
19  IF (line=="") CYCLE
20  daten=APPEND(daten,line)
21 ENDLOOP
22 TRACE *daten
```