



Mehr über ADAPTOR Scripting

Adaptor ist eine Applikationssprache, die mit einfachen Befehlen den Ablauf der AlligatorSQL Software steuert. Der Dialekt der Sprache ist BASIC, eine Sprache, die sehr leicht zu erlernen ist bzw. viele Anwender schon einmal benutzt haben.

01_003_04

Mehr über ADAPTOR Scripting

Das Adaptor Scripting wird zur Lösung von häufig auftretenden Arbeiten benutzt. Zum Beispiel können Text oder Programmierblöcke; sogenannte Templates; erstellt werden. Diese werden aber nicht einfach in einen Texteditor kopiert, sondern können gezielt in ein zu entwickelndes Programm integriert werden.

Hier ein kurzes Beispiel:

Stellen Sie sich vor, Sie müssen Schnittstellenprogramme für eine Datenbank entwickeln. Dabei erhalten Sie eine Tabellenbeschreibung, die in etwa so aussieht:

```
CustomerId ONLY_NUMERIC_FIELD  
CustomerName ALPHA_NUMERIC_FIELD  
Street ALPHA_NUMERIC_FIELD  
Birthday DATE_FIELD
```

```
.  
.  
.
```

Ihre Aufgabe besteht nun darin, aus diesen Beschreibungen Tabellen in der Datenbank zu generieren. Bei einer einzigen Datei ist es wohl keine Schwierigkeit ein geeignetes Script zur Tabellenanlage zu erzeugen, sei es manuell oder mit einer Makrosprache. Liegen jedoch hunderte solcher Anforderungen vor, raubt Ihnen diese Aufgabe sicherlich einen großen Teil Ihrer Zeit. Hier im einzelnen die Schritte, um Ihnen die Arbeit mit Adaptor Scripting zu erleichtern:

Nachfolgend das Adaptor Script als Lösung:

Adaptor Beispielscript

```
` Bestimme aktuellen Viewindex  
DIM nIndex AS INTEGER  
  
nIndex = GetActualViewIndex()  
  
` Positioniere Cursor an den Anfang  
View(nIndex).Locate(1, 1)  
  
` Lese Anzahl der Datensätze  
DIM nRecCount AS INTEGER  
nRecCount = View(nIndex).GetLineCount()  
  
` Öffne leeres neues Dokument mit Oracle Syntax Hervorhebung  
DIM nNewView AS INTEGER  
CreatView(ORACLE)  
nNewView = GetActualViewIndex()  
  
` Verarbeitung  
DIM nInd AS INTEGER  
DIM sRecord AS STRING  
DIM nBlankPos AS INTEGER  
DIM sLeft AS STRING  
DIM sRight AS STRING  
DIM sOut AS STRING  
DIM sType AS STRING  
DIM nFirst AS INTEGER
```

```

nFirst = 1

SOut = `CREATE TABLE interface (
View(nNewView).AppendLine(sOut)

FOR nInd = 1 TO nRecCount
  sOut = ``

  IF nFirst = 0 THEN
    sOut = sOut + `,`
  ELSE
    nFirst = 0
  END IF

  cString = View(nIndex).ReadLine(nInd)

  nBlankPos = INSTR(cString, ` `)
  IF nBlankPos > 0 THEN
    sLeft = TRIM(LEFT(sRecord, nBlankPos))
    sRight = TRIM(RIGHT(sRecord, LEN(sLeft)-nBlankPos-1))

    IF sRight = `ONLY_NUMERIC_FIELD` THEN
      sType = `NUMBER`
    ELSE IF sRight = `ALPHA_NUMERIC_FIELD` THEN
      sType = `VARCHAR2(200)`
    ELSE IF sRight = `DATE_FIELD` THEN
      sType = `DATE`
    ELSE
      sType = `,unbekannter Schnittstellentyp`
    END IF

    sOut = sOut + sLeft + ` ` + sType

    View(nNewView).AppendLine(sOut)

  END IF

NEXT

sOut = `)`;
View(nNewView).AppendLine()
View(nNewView).Redraw()
END

```

Aus der Schnittstellendatei

```

CustomerId ONLY_NUMERIC_FIELD
CustomerName ALPHA_NUMERIC_FIELD
Street ALPHA_NUMERIC_FIELD
Birthday DATE_FIELD

```

wird

```

CREATE TABLE interface (
CustomerId NUMBER
, CustomerName VARCHAR2(200)
, Street NUMBER
, Birthday DATE
);

```