

Dátový kontajner

Dátový kontajner je interná dátová štruktúra umožňujúca uchovávať jednoduché aj štruktúrované hodnoty podľa kú. Vlastníkom kontajnera je vždy jedna bežiacia inštancia skriptu. Dátový kontajner je možné zdieľať medzi rôznymi skriptami, aj procesmi. Kontajner má vlastný jednoznačný identifikátor.

Vznik kontajnera zabezpečujú akcie [CNT_CREATE](#) - vone použitény kontajner, alebo [GETARCHARR_TO_CNT](#). Druhý typ kontajnera je plnený stránkami, ktoré obsahujú dáta prečítané z archívu.

Takto koncipovaný prístup do archívu je efektívnejší (spotreba pamäte a iasťone rýchlos) ako použitie akcie GETARCHARR. Pre kontajner vytvorený akciou GETARCHARR_TO_CNT sú použiténe len akcie [CNT_GETNR](#), [CNT_FIND](#) and [CNT_DESTROY](#) ([príklad](#)). Kontajner zaniká automaticky zánikom skriptu, ktorý ho vlastní alebo vykonaním akcie [CNT_DESTROY](#).

Vekos kontajnera nie je určená a je obmedzená len vekosou operanej pamäte počítača.

Každá hodnota v kontajneri je jednoznačne identifikovaná pomocou tzv. *kú*. Hodnoty je možné do kontajnera ubovone vkladať, vyhádzať, prezerať a mazať. Hodnoty vkladané do kontajnera môžu byť ubovoného typu (*Int*, *Bool*, *Text*, *Real*, *Time*), prípadne štruktúry (celá štruktúrovaná premenná, riadok, ...). Typ hodnoty kú musí byť *Int*, *Bool*, *Text*, *Real* alebo *Time*, pričom platí, že všetky kú v kontajneri musia byť rovnakého typu.

Prácu s dátovým kontajnerom zabezpečujú akcie uvedené v dokumente [Akcie v skriptoch](#).

Prenos dátových kontajnerov medzi bežiacimi ESL skriptami

Prenos kontajnerov je možný pomocou RPC procedúr. V deklarácii RPC procedúry je potrebné označiť parameter, reprezentujúci handle na kontajner, špeciálnym typom **CNT_HANDLE**. Algoritmus je podmienený existenciou dátových kontajnerov. Ak handle na dátový kontajner je neplatná hodnota alebo ukazuje na neexistujúci dátový kontajner, algoritmus skoní s chybou.

Hodnota lokálnych premenných typu **CNT_HANDLE** je interpretovaná ako celočíselná (INT).

Deklarácia RPC procedúry:

```
RPC PROCEDURE ProcName [([IN] CNT_HANDLE paramName1[,paramName2, ...] [IN] CNT_HANDLE paramName3]...)]  
  
;akcie  
  
END ProcName
```

Poznámky:

- Vlastníkom dátového kontajnera môže byť vždy iba jeden ESL skript, ktorý zároveň zabezpečí aj zrušenie dátového kontajnera.
- Dátové kontajnery je možné prenášať aj medzi ESL skriptami rôznych procesov. Pri tomto spôsobe dochádza k prenosu všetkých dát, ktoré sú umiestnené v kontajneri.
- Ak pri volaní RPC procedúry ako skutočný parameter na mieste formálneho parametra typu CNT_HANDLE použijete obyčajnú hodnotu, ktorá nie je handle, ESL skript vyhadá dátový kontajner podľa vstupnej hodnoty.
 - Ak dátový kontajner označený vstupnou hodnotou existuje, tak ho preniesie.
 - Ak dátový kontajner označený vstupnou hodnotou neexistuje, tak ESL skript vyhlási chybu - kontajner s handle = x nenašiel (nie sú dáta 22).

```
INT _cnt_handle  
  
_cnt_handle := 5  
  
CALL [objIdent] CNT (_cnt_handle) ON procIdent
```

- Ak vstupná hodnota, reprezentujúca handle, je neplatná (Invalid), volajúci ESL skript je ukončený s RunTime chybou.
- Ak je volanie RPC procedúry **asynchrónne**, po takom volaní kontajner v tomto skripte zaniká a vlastníkom sa stáva volaný ESL skript:

```

*****

; volaný ESL skript
RPC PROCEDURE InsertToContainer(CNT_HANDLE _handle)
....
END InsertToContainer

*****

; volajúci ESL skript
INT _cnt_handle

CALL[...] InsertToContainer(_cnt_handle) ASYNC ON
....
; Po takomto volaní, kontajner v tomto skripte zaniká, vlastníkom sa stáva volaný ESL skript

*****

```

- Ak je volanie RPC procedúry **synchronne**, tak sú dve možnosti:
 1. Ak formálny parameter reprezentujúci CNT_HANDLE je oznaený kúovým slovom IN, pri volaní RPC procedúry sa vlastníkom dátového kontajnera **natrvalo** stane ESL skript obsahujúci deklaráciu volanej RPC procedúry.

```

*****

; volaný ESL skript
RPC PROCEDURE InsertToContainer(IN CNT_HANDLE _handle)
....
END InsertToContainer
*****

; volajúci ESL skript
INT _cnt_handle

CALL[...] InsertToContainer(_cnt_handle) ON ....
; Po takomto volaní kontajner v tomto skripte zaniká, vlastníkom sa stáva volaný ESL skript
*****

```

2. Ak formálny parameter reprezentujúci CNT_HANDLE nie je oznaený kúovým slovom IN, pri volaní RPC procedúry sa **doasným** vlastníkom dátového kontajnera stane ESL skript obsahujúci deklaráciu volanej RPC procedúry. Po skonení volanej RPC procedúry sa vlastníkom stáva skript, z ktorého bola RPC procedúra volaná.

```

*****

; volaný ESL skript
RPC PROCEDURE InsertToContainer(CNT_HANDLE _handle)
....
END InsertToContainer
*****

; volajúci ESL skript
INT _cnt_handle

CALL[...] InsertToContainer(_cnt_handle) ON ....
; Po takomto volaní, vlastníkom dátového kontajnera ostáva volajúci ESL skript
*****

```

Príklad 1:

```

INT _INTER_HANDLE

RPC PROCEDURE MakeWithCNT_IN (IN CNT_HANDLE _CNT_Handle, BOOL _bOk)

INT _iKey
INT _value
BOOL _bFound

    _iKey := 1
    _INTER_HANDLE := _CNT_Handle
    CNT_FIND _INTER_HANDLE, _iKey, _value, _bFound

    IF _bFound THEN
        _value := 3
        CNT_INSERT _INTER_HANDLE, _iKey, _value
    ENDIF

END MakeWithCNT_IN

RPC PROCEDURE MakeWithCNT_IN_OUT (CNT_HANDLE _CNT_Handle, BOOL _bOk)

INT _iKey
INT _value
BOOL _bFound

    _iKey := 1
    _INTER_HANDLE := _CNT_Handle
    CNT_FIND _INTER_HANDLE, _iKey, _value, _bFound

    IF _bFound THEN
        _value := 3
        CNT_INSERT _INTER_HANDLE, _iKey, _value
    ENDIF

END MakeWithCNT_IN_OUT

```

Příklad 2:

```

INT _Handle

PROCEDURE Interny_call(BOOL _bOk)
INT _iKey
INT _value
BOOL _bFound

_bOk := @TRUE
_iKey := 1

;*****
;** Prenesie cnt do druhého skriptu a tam vymení na kúí 1 hodnotu na 3
;**
;*****

CALL [E.1] MakeWithCNT_IN_OUT(_Handle,_bOk) ON SELF.EVH

IF !_bOk THEN
RETURN
ENDIF

CNT_FIND _Handle, _iKey, _value, _bFind

_bOk := _bFind & _value = 3

;*****
;** Prenesie cnt do druhého, tento skript sa ho vzdáva
;**
;*****

CALL [E.1] MakeWithCNT_IN(_Handle, _bOk) ON SELF.EVH
END Interny_call

BEGIN

INT _iKey
INT _value

_iKey := 1
_value := 2
CNT_CREATE _Handle
CNT_INSERT _Handle, _iKey, _value

END

```



Súvisiace stránky:

[Akcie v skriptoch](#)