25 tip: Dynamické parametre tried

Prv než sa pustím do témy, len krátke upozornenie. Podarilo sa mi do tohto seriálu zapojiť kolegu Havlíčka. Jeho premiérový tip ste si mohli prečítať ako v poradí dvadsiaty štvrtý. Teraz, keď sme na to dvaja, azda sa nám podarí udržať tento seriál na vyššej úrovni. Držte nám palce...

O tom, že parametre v definícii tried slúžia ako miesto, kde sa spravidla definujú konštantné hodnoty, spoločné pre všetky inštancie tried, určite viete. To, že k hodnotám týchto parametrov môže Caché pristupovať buď v runtime, alebo pri kompilácii tried, iste tiež viete. No možno neviete, že výrazy v parametroch nemusia byť konštantami, ale môžu sa dynamicky vyhodnocovať za behu programu.

Ak chceme zistiť hodnotu parametra, napríklad parametra NameSpace, potom použijeme syntax:

```
write ..#NAMESPACE
```

Pokiaľ má parameter NameSpace hodnotu $ zu (5), potom za behu systému bude uvedený kód interpretovaný takto:

```
write "$zu(5)"
```

My by sme však radi videli výraz, a nie reťazec $ Zu (5), teda radi by sme videli toto:

```
write $zu(5)
```

To docielime použitím podobného operátora, aký používa Caché ObjectScript na nepriamu adresáciu, to jest "@". Ak napíšeme...

```
write @..#NAMESPACE
```

...dostaneme presne to, čo hľadáme. Ukážeme vám jednoduchý konkrétny príklad: Majme nasledujúcu triedu:

```
Class tt.Tip25 Extends %RegistreredObject
{

Parameter SYSNSP  =  "$g ^dk(" "SYSNSP" ") )
" ;

ClassMethod Test ( )
     {
            set   ^dk ("SYSNSP")  = „SYM"
          write   !,@..#SYSNSP
          write   !,..#SYSNSP
    }
|
 }
```

Vykonaním metódy Test získame nasledujúci výstup:

```
SYM>d ##class(tt.Tip25).Test()

SYM
$g(^dk("SYSNSP"))
SYM>
```

No toto nie je jediný spôsob, ako s parametrami pracovať dynamicky. Ukážeme si to na mierne upravenom variante našej triedy.

```
Class   tt.Tip25 Etends %RegisteredObject
{
   Parameter  DK  as  STRING  =  "$zu(5)";

   parameter  DKCOSCODE  as  COSCODE =  "set
x=1   w    " "x=" " _x";
   parameter  DKCOSEXPRESSION  as  COSEXPRES-
SION  =  "$lb(" "bila" ", " "cervena" ", "
"modra" ") ";
ClassMethod  Test ( )  [  ProcedureBlock = 1
]
    {
            w !,"@ STRING
:",@..#DK
            w !," STRING
:",..#DK
            w !," COSCODE             :"
s x=..#DKCOSCODE x x
            w !,"COSEXPRESSION:"
s barvy=..#DKCOSEXPRESSION  w  $lg(barvy,1)
      }
  }
SYM>d ##class(tt.Tip25).Test()
@ STRING     :SYM
STRING       :$zu(5)
COSCODE      :x=1
COSEXPRESSION:bila
SYM>w
x=1
SYM>
```

V tomto variante triedy sme použili dva špeciálne typy parametrov – COSEXPRESSION a COSCODE. Nastavením hodnoty typu parametra sa hovorí kompilátoru triedy, že takto definované parametre sa nemajú považovať za obyčajné reťazce, ale za výraz Caché ObjectScriptu a kód napísaný v ObjectScripte. Na ich vyhodnotenie potom nemusíme použiť operátor "@" pred názvom parametra. Vo výpise si takisto môžete všimnúť, že ukazovatele uvedené v rámci COSCODE sú automaticky označené ako *verejné*.

DAN KUTÁČ, InterSystems

Ďalšie zaujímavé diely seriálu Tipy a triky s Caché, ktoré vychádzajú už od roku 2005, nájdete na stiahnutie tu: **http://www.intersystems.cz/education/university/serialy/tipyTrikyCache.html**