

Article

[Daniel Kutac](#) · Nov 9, 2015 3m read

## Tipy a Triky s Caché

### Tip dvacátý šestý: objekty a concurrency 2 - swizzling

Když tento seriál před několika lety začínal, byl jeho první díl věnován zajištění izolace instance objektu pro exkluzivní přístup a popisu příslušných API funkcí. Nedávno se mi ale stalo, že mě tento díl dostihl. Jeden ze zákazníků začal mít problémy v aplikaci, přestože dříve používal exkluzivní zámky pro editování instancí svých objektů. Když po pátrání dále sleduji, zjistil (a poznám se že jsem to sám nevěděl) že zámky aplikované na instance objektů mají jednu velmi nepříjemnou vlastnost, a to tu, že se nešíří prostřednictvím odkazů (swizzling) na další propojené objekty.

A to byl právě zdroj oněch problémů. V aplikaci se nacházelo několik míst, kde byla dvěma procesy zároveň modifikována stejná instance objektu. Jednou přímo, podruhé prostřednictvím právě odkazu z jiné, propojené třídy. Jak toto může nastat a jak takové situace ošetřit si ukážeme v následujícím modelovém příkladu, jehož zdrojový kód najdete [zde](#).

Po stažení, rozbalení a nainstalování projektu můžeme testovat. Otevřeme si 2 konzole Caché a přepneme se do testovacího názvového prostoru, přiravíme si, ALE ZATÍM NESPUSTÍME, příkazy:

```
// konzole 1
>w ##class(User.Test).GoObjectB()

// konzole 2
>w ##class(User.Test).GoObjectA(0)
```

Nejdříve spustíme proces na konzoli 1 a během 10 vteřin i proces na konzoli 2 a sledujeme výstupy. To co uvidíme, může vypadat na první pohled divně, ale je to v pořádku, proces v konzoli 2 se snaží otevřít novou instanci objektu User.B, a jakmile se mu to podaří (poté co proces v konzoli 1 uvolní zámek), načte jej z DISKU a změní, mezitím proces v konzoli 2 provede operativní načtení instance z disku a zobrazí výsledek. V ideálním případě budou hodnoty obou instancí konzistentní (v praxi se nám totiž kód obou konzolí může vykonat rychle a vrátit pak tomu odpovídající - od níže uvedeného obrázku odlišné - výsledky).

A teď zavoláme

```
// konzole 1
>w ##class(User.Test).GoObjectB()

// konzole 2
>w ##class(User.Test).GoObjectA(1)
```

Výsledek bude odlišný - stejná instance objektu vrátí různé hodnoty v každém procesu, jak ukazuje souhrnný obrázek:

Obr. 1

Kritický kód uvádím ještě separátně: takto je potřeba vždy ošetřit přístup k odkazovaným objektům, pokud je chceme izolovat.

```
#dim oB as User.B
write !, "Trying to Open B from A"
set oB=oA.relaceB.%Open(oA.relaceB.%Oid(),4,.sc)
$$$THROWONERROR(sc,sc)
set oB.B2 = "B2 Done"
```

Rozdíly v chování zámků při přímém přístupu a při použití swizzlingu mohou, jak bylo ukázáno, mít velké důsledky pro aplikaci. Bohužel dokumentace k tomuto chování není dostatečně podrobná.

Možná se Vám tento příklad může jevit jako přitažený za vlasy, ale v praxi je velmi dobře možné na takovou kombinaci dvou procesů pracujících nad stejnou instancí jednou přímou a jednou nepřímou snadno narazit. Například, provádíte-li účetní zápis, pak aktualizujete záznamy na jednotlivých útech analytické evidence v účetní knize ale také můžete záznamy na příslušných útech syntetické evidence. No, a pokud se někdo rozhodne změnit cokoliv na syntetickém účtu v okamžiku provádění účetní operace, je zadáno na nešťastí - pokud tedy nebudete dbát výše uvedených doporučení.

[#Code Snippet #System Administration #Caché](#)

Source URL: <https://community.intersystems.com/post/tipy-triky-s-cach%C3%A9>