Article
Daniel Kutac · Nov 9, 2015 3m read

Tipy a Triky s Caché

Tip dvacátý šestý: objekty a concurrency 2 - swizzling

Když tento seriál p ř ed několika lety za č ínal, byl jeho první díl věnován zajištění izolace instance objektů pro exkluzivní p ř ístup a popisu p ř íslušných API funkcí. Nedávno se mi ale stalo, že mě tento díl dostihl. Jeden ze zákazníků za č al mít problémy v aplikaci, p ř estože důsledně používal exkluzivní zámky pro editování instancí svých objektů. Když po p ř í č ině pátral důsledněji, zjistil (a p ř iznám se že jsem to sám nevěděl) že zámky aplikované na instance objektů mají jednu velmi nep ř íjemnou vlastnost, a to tu, že se neší ř í prost ř ednictvím odkazů (swizzling) na další propojené objekty.

A to byl právě zdroj oněch problémů. V aplikaci se nacházelo několik míst, kde byla dvěma procesy zárove ň modifikována stejná instance objektu. Jednou p ř ímo, podruhé prost ř ednictvím právě odkazu z jiné, p ř ipojené t ř ídy. Jak toto může nastat a jak takové situace ošet ř it si ukážeme v následujícícm modelovém p ř íkladu, jehož zdrojový kód najdete zde.

Po stažení, rozbalení a nainstalování projektu můžeme testovat. Otev ř eme si 2 konzole Caché a p ř epneme se do testovacího názvového prostoru, p ř iravíme si, ALE ZATÍM NESPUSTÍME, p ř íkazy:

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

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

Nejd ř ív spustíme proces na konzoli 1 a během 10 vte ř in i proces na konzoli 2 a sledujeme výstupy. To co uvidíme, může vypadat na první pohled divně, ale je to v po ř ádku, proces v konzoli 2 se snaží otev ř ít novou instanci objektu User.B, a jakmile se mu to poda ř í (poté co proces v konzoli 1 uvolní zámek), na č te jej z DISKU a změní, mezitím proces v konzoli 2 provede opětovné na č tení instance z disku a zobrazí výsledek. V ideálním p ř ípadě budou hodnoty obou instancí konzistentní (v praxi se nám totiž kód obou konzolí může vykonat různě rychle a vrátit pak tomu odpovídající - od níže uvedeného obrázku odlišné - výsledky).

A te ď zavoláme

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

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

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

Obr. 1

Kritický kód uvádím ještě separátně: takto je pot ř eba vždy ošet ř it p ř ístup k odkazovaným objektům, pokud je chceme izolovat.

```
#dim oB as User.B
w !,"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 p ř i p ř ímém p ř ístupu a p ř i použití swizzlingu mohou, jak bylo ukázáno, mít velké důsledky pro aplikaci. Bohužel dokumentace k tomuto chování není dostate č ně podrobná.

Možná se Vám tento p ř íklad může jevit jako p ř itažený za vlasy, ale v praxi je velmi dob ř e možné na takovou kombinaci dvou procesů pracujících nad stejnou instancí jednou p ř ímo a jednou nep ř ímo snadno narazit.
nap ř íklad, provádíte-li ú č etní zápis, pak aktualizujete zůstatky na jednotlivých ú č tech analytické evidence v ú č etní knize ale také měníte zůstatky na p ř íslušných ú č tech syntetické evidence. No, a pokud se někdo rozhodne změnit cokoliv na syntetickém ú č tě v okamžiku provádění ú č etní operace, je zaděláno na neštěstí - pokud tedy nebudete dbát výše uvedených doporu č ení.

#Code Snippet #System Administration #Caché