

```

14 // Ak sParam = "generuj" - nové náhodné vzorky
15 // pre každý bod grafu
16 if (sParam=="generuj")
17 foreach (IB.CGraf.CBod Bod in SvgGraf)
18 {
19     double y = SvgGraf.m_random.NextDouble()*Bod.max;
20     if (Bod.Count<Bod.nBodov )Bod.Add(y);
21 }
22
23 // Zistiť kód požadovanej operácie
24 string sKod = "??";
25 if (Request.Params["kod"] != null) sKod = Request.Params["kod"];
26 string sOdpoved="??";
27 switch (sKod)
28 {
29     case "Obnov" : sOdpoved=Obnov(); break;
30     case "Hodnota" : sOdpoved=Hodnota(); break;
31     .. ďalšie prípady ..
32 }
33 Response.Write(sOdpoved);
34 Response.Expires = 0;
35 Response.End();
36 }
37 }
38 }
39
40 protected string Obnov ()
41 {
42     return
43     SvgGraf.SVGZaciatok() + SvgGraf.SVGMriezka ()
44     + SvgGraf.SVGCGraf() + SvgGraf.SVGKoniec();
45 }
46
47 .. Podobne aj ďalšie funkcie ..

```

Funkcionalitu na strane servera zabezpečuje formulár `reqgraf`. Ten je volaný na spracovanie grafu – objektu triedy `CGraf`, ktorý je zaregistrovaný v

session. Ak taký objekt existuje, získame hodnoty parametrov poslaných klientom. Rozhodujúci pre pripravenie odpovede je kód požadovanej operácie. V prípade, že je to text `Obnov`, je volaná rovnomenná funkcia. Tá vyžiada od objektu `SvgGraf` jednotlivé súčasti, ktoré tvoria kód SVG grafu. Takto získaný kód je odovzdaný metódou `Write` objektu `Response`. Pre stránky, ktorých obsah sa môže na strane servera meniť, je dôležité nastaviť dobu expirácie na 0.

Podobne, ako je tu ukázaná cesta vedúca k spracovaniu žiadosti na obnovu grafu, možno vyriešiť aj obsluhu žiadostí prichádzajúcich s iným kódom. V príkaze `switch` medzi riadkami 28 až 33 stačí doplniť ďalší riadok `case` s volaním príslušnej funkcie, ktorej kód doplníme do zoznamu funkcií.

## Záver

Opísaný postup dynamickej zmeny grafu je na prvý pohľad zložitý. Pri pochopení podstaty riešenia a dopĺňaní nových funkcií však môžeme zúžitkovať poznatky z raz prejdenej cesty. Ak sa pokúsime urobiť odhad náročnosti dodania nového ovládacieho prvku na stránku v podobe počtu riadkov kódu, dostaneme zhruba takéto čísla: jeden až dva riadky kódu HTML, päť až dvadsať riadkov kódu v jazyku JavaScript, desať až dvadsať riadkov v jazyku C#. Viem si predstaviť vývojový nástroj, ktorý by ponúkol zoznam použiteľných komponentov. Potom by stačilo vybrať si komponent a definovať mu vlastnosti. V pozadí však zostane kód. Poznať jeho podstatu môže byť neraz výhodné.

■ IMRICH BURANSKÝ

## Tipy a triky pre Caché I.

### Izolácia

[*Tipy - triky*] - Začínáme voľný seriál o praktických tipoch a trikoch pre databázovú platformu Caché. Cieľom je prinášať vám prakticky použiteľné informácie, ktoré možno ťažko získať iným spôsobom a o ktorých existencii azda ani neviete. Tentoraz sa pozrieme na problematiku konkurenčného prístupu k objektom – izoláciu.

Zatiaľ čo väčšina programátorov, ktorí poznajú niektoré z implementácií M technológie, vo svojom kóde pravidelne a samozrejme používa príkaz `LOCK` skriptovacieho jazyka, spôsob, akým je izolácia implementovaná v objektoch Caché, im môže pripadať pomerne neprehľadný. Pokúsime sa v nasledujúcich riadkoch stručne opísať a interpretovať to, čo možno nájsť v on-line dokumentácii ku Caché.

### Príklad

Proces A volá `L + ^Demo(1):5`, tým sa vytvorí inkrementálny zámok na zázname `^Demo(1)`. Ak proces B volá `L + ^Demo(1):5`, môže pomocou hodnoty `$T` zistiť, či sa pokus o uzamknutie podaril alebo nie. V našom prípade sa, pochopiteľne, nepodarí.

Pokiaľ však proces B zavolá `L + ^Demo(1)`, zostane „visieť“ po celý čas, kým proces A zámok neuvoľní. Proces B teda nemá možnosť oznámiť používateľovi, ktorý ho spustil, čo sa deje, a používateľ nemá možnosť na vzniknutú situáciu zareagovať.

Teraz sa zameriame viac na objekty.

Z dokumentácie sa dá ľahko zistiť, že otvorenie existujúcej inštancie objektu sa vykoná volaním metódy `%OpenId(id)`, kde `id` je reťazec

identifikujúci inštanciu triedy na disku:

```
Set osoba=##class(Demo.Osoba).%OpenId(2)
```

V tomto prípade však systém vôbec pri otvorení inštancie nepoužije zámok. Môže teda nastať situácia, kde proces A modifikuje metódou `%Save()` dáta danej inštancie, zatiaľ čo proces B sa práve snaží danú inštanciu čítať; pokiaľ je trieda zložitá, môže dôjsť k tomu, že proces A stihne zapísať iba časť modifikovaných dát a proces B potom načíta už modifikované dáta, ako aj dáta ešte nezmenené a tým dôjde k načítaniu neplatných dát.

V dokumentácii sa to veľmi nez dôrazňuje, ale metóda `%OpenId()` má v skutočnosti tri argumenty:

```
(id as %String, concurrency as %Integer,
byref status as %Status)
```

Posledné dva sú však nepovinné.

Na správnu funkciu zámokov v prípade práce s objektmi však treba používať všetky tri argumenty. Nestačí dokonca ani použitie iba prvých dvoch. To preto, že v prípade otvorenia inštancie s najvyššou úrovňou izolácie sa vôbec nevytvorí referencia na inštanciu. To môže mať rôzne príčiny, ale iba z premennej pre status sa dozvieme, že metóda zlyhala pre existenciu výhradného zámku vlastneného iným procesom.

```
Set osoba=##class(Demo.Osoba).%OpenId(2,4,sc)
If $System.Status.IsError(sc)
{
    do $System.Status.DisplayError(sc)
}
else
{
    ..... vykonaj požadované akcie ...
}
```

A teraz sa bližšie pozrieme na jednotlivé hodnoty argumentu `concurrency`.

### 0 - bez zámku

V tomto prípade nie sú zámky použité. Dôsledkom bude pravdepodobne nekonzistentné čítanie dát.

### 1 - autonómny zámok

Keď je objekt načítaný, metóda `%LoadData` vytvorí zdieľaný zámok za podmienky, že objekt zaberá viac než jeden uzol (globálu) v databáze. Pokiaľ sú dáta inštancie uložené v jednom uzle, zámok sa nevytvára. Zámok je uvoľnený na konci metódy `%LoadData`.

Pri vytváraní novej inštancie metóda `%SaveData` vytvorí výhradný zámok za predpokladu, že dáta zaberajú viac uzlov v globále, inak sa zámok neuplatní. V prípade aktualizácie existujúcej inštancie sa výhradný zámok vytvára vždy.

V tomto prípade sa teda redukuje pravdepodobnosť nekonzistentného čítania, objekt však nie je chránený proti editácii viacerými procesmi zároveň.

### 2 - zdieľaný zámok

Keď je objekt načítaný, metóda `%LoadData` vždy vytvorí zdieľaný zámok. Zámok je uvoľnený na konci metódy `%LoadData`.

Pri vytváraní novej inštancie metóda `%SaveData` vytvorí výhradný zámok za predpokladu, že dáta zaberajú viac uzlov v globále, inak sa zámok neuplatní. V prípade aktualizácie existujúcej inštancie sa výhradný zámok vytvára vždy.

Zabrání sa teda možnosti nekonzistentného čítania, ale objekt nie je chránený proti editácii viacerých procesov zároveň.

### 3 - zdieľaný a podržaný zámok

Keď je objekt načítaný, metóda `%LoadData` vždy vytvorí zdieľaný zámok. Zámok je uvoľnený metódou `%Close` (alebo pri zničení referencie na objekt pomocou príkazu `Kill`).

Pri vytváraní novej inštancie metóda %SaveData vytvorí výhradný zámok za predpokladu, že dáta zaberajú viac uzlov v globále, inak sa zámok neuplatní. V prípade aktualizácie existujúcej inštancie sa výhradný zámok vytvára vždy.

Všetky procesy teda môžu otvoriť inštanciu iba na čítanie, žiadny nemôže získať exkluzívny zámok, a teda zapisovať zmeny.

#### 4 - exkluzívny zámok

Keď je objekt načítaný, metóda %LoadData vždy vytvorí výhradný zámok. Zámok je uvoľnený metódou %Close (alebo pri zničení referencie na objekt pomocou príkazu Kill).

Pri vytváraní novej inštancie metóda %SaveData vytvorí výhradný zámok za predpokladu, že dáta zaberajú viac uzlov v globále, inak sa zámok neuplatní. V

prípade aktualizácie existujúcej inštancie sa výhradný zámok vytvára vždy.

Iba prvý proces má teda úplný prístup k inštancii. Žiadny ďalší proces danú inštanciu ani nenačíta.

#### O autorovi

Daniel Kutáč pracuje ako Senior Sales Engineer pre spoločnosť InterSystems od roku 2000. Vyštudoval ČVUT, Fakultu jadrového a fyzikálneho inžinierstva, potom pracoval pre Komerčnú banku v oblasti obchodu s cennými papiermi, následne pre aplikačného partnera InterSystems, anglickú spoločnosť MISYS UK, kde bol zodpovedný za implementáciu systémov na obchodovanie s cennými papiermi v rôznych štátoch strednej a východnej Európy. Je ženatý, má jednu dcéru a k jeho záujmom patrí cyklistika a turistika. ■

### Knižné novinky

## OpenOffice.org 2.0

Kniha oboznamuje čitateľa s kancelárskym balíkom **OpenOffice.org**, ktorý je dosť rozsiahly. Autor sa preto rozhodol z každej aplikácie podchytiť tie funkcie, ktoré začínajúci používateľ bude najviac využívať.

Prvá kapitola je „zoznamovacia“, obsahuje stručný opis balíka programov OpenOffice.org, dočítate sa, ako program inštalovať a spúšťať.

Druhá kapitola obsahuje podrobný opis prostredia OpenOffice.org, dočítate sa niečo o tom, ako nastaviť základné parametre programu.

Treťou kapitolou sa začína opis práce s jednotlivými aplikáciami, prvou z nich je textový editor Writer. Najdôležitejšiu časť tvorí opis práce so štýlmi, na záver



kapitoly nasleduje stručné oboznámenie s tabuľkami a tabuľkami.

Štvrtá kapitola je určená pre priaznivcov matematických vzorcov a tabuliek v aplikácii Calc cez vkladanie dát, operácie s bunkami až po vkladanie vzorcov a prácu s funkciami. Grafickým

aplikáciám je venovaná piata kapitola.

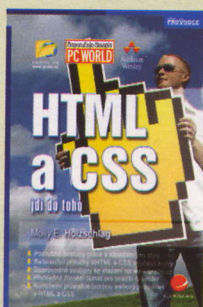
Záverečná kapitola sa stručne dotkne aplikácie Base a na malom konkrétnom príklade ukáže, ako vytvoriť jednoduchú databázu a ako vytvárať formuláre alebo zostavy.

Autor: **Josef Pecinovský**  
 Vydavateľ: **GRADA Publishing**  
 Počet strán: **292**  
 Cena: **459 Sk**

## HTML a CSS

Chcete si vytvoriť vlastné webové stránky? Alebo ich už máte a chcete ich zdokonaľiť? Táto kniha je určená práve pre vás. Nájdete v nej desiatky efektívne a podrobne opísaných príkladov, zameraných na HTML, XHTML

a CSS, ktoré môžete ľubovoľne používať a vzájomne bez obmedzení kombinovať. Vďaka nim bez problémov bezpečne zvládnete najlepšie techniky



dneška - preniknete priamo k jadrú bez zbytočných okľúk a rozvláčných teórií. Žiadny iný sprievodca HTML a CSS neobsahuje toľko informácií a nenaučí vás techniku tvorby dynamických webových stránok tak rýchlo.

Autor: **Molly E. Holzschlag**  
 Vydavateľ: **GRADA Publishing**  
 Počet strán: **264**  
 Cena: **383 Sk**