

Re: Infinite Loops and Explicit Exits

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2004-12/0492.html>

From: Pete Dashwood (*dashwood_at_enternet.co.nz*)

Date: 12/07/04

Date: Wed, 8 Dec 2004 02:12:43 +1300

"Robert Wagner" <spamblocker-robert@wagner.net> wrote in message
news:l029r0990r1d0116d8nihu34cgrbhs9i0b@4ax.com...

> *On Mon, 6 Dec 2004 23:10:03 +1300, "Pete Dashwood"*

> *<dashwood@enternet.co.nz> wrote:*

>

>>

>> *"Robert Wagner" <spamblocker-robert@wagner.net> wrote in message*

>> *news:g237r0hcilmnrmsqhi94i57up4ov2b9cjp@4ax.com...*

>>> *On Mon, 6 Dec 2004 01:41:27 +1300, "Pete Dashwood"*

<snip>>

> *Tax calculation is very often done in third-party code that's out of*

> *your control. Suppose the entry point is named VERTEX-ITX. You might*

> *want to document the call with a flower-boxed comment.*

>

Or...you MIGHT want mustard and tomato ketchup on your ice cream sundae...

You MIGHT want ANYTHING. The discussion is not about what you MIGHT want. It
is about a specific set of circumstances. (Or at least, my side of it is...

<G>)

> *Suppose your shop has an ancient mainframe naming standard that says*

> *programs must be named ssbbbnp (just in case we have to go back to*

> *VSE). You think I'm making this up, don't you?*

>

I could be forgiven for thinking that, Robert. There is precedent...<G>

If you say it is true on this occasion, I'll take your word for it. It is
not pertinent to the conversation because it comes under MIGHT happenings,
and that isn't what I want to discuss.

>> *Note that if this was an Object Method (and I wrote it) it would contain*

>> *Properties (PUBLIC) for the employee ID and the current pay field. All
the*

>> *other fields it needed (derived database access keys, foreign keys,*

>> *relations, tables, semapores, calc fields, etc.) would be PRIVATE. It is*

a

>> *black box where you simply give it the employee you want paid and it*

returns

> >the amount to pay him. All of his various rates would be accessed from
> >databases, along with the hours or part hours for each rate. Tax and
> >deductions, FICA, pension fund, whatever, would all be found on databases
> >and manipulated by this one Object Method using rules based processing,
> >implemented by soft rules, that can be changed outside the object.
>
> OO is not required to do that. For decades, people have done it with
> called programs, remote procedure calls, etc.

>
No, people have TRIED to do it with those devices and had varying degrees of success. OO lends itself to this type of solution, but I agree it does not preclude other solutions.

> >I would not expect to ever maintain this code again, once it was written,
> >and, as the interface to it would never change either (linkage was not
used
> >to pass data in and out), there would be no impact on code that invokes
it
> >if other innate features of it were activated later.

> >
> >In effect, the whole payroll system would be dependent on proper database
> >design. The behaviour could only be changed by changing the database. I
> >believe that is a good thing.

>
> I design large systems the same way. For instance, at aforementioned
> supermarket company, a frequent activity was determining the cost of a
> product on a given date. I completely re-engineered the tables
> containing that information without touching, recompiling or
> re-testing the many programs that called for cost. Before the Big
> Rewrite, dozens of programs would have needed substantial logic
> changes.

>
Good for you. That was probably forward thinking to the people on the site.

> >In fact, I am coming to the conclusion that there may be a good case for
> >designing proper databases and using stored procedures on them, to
implement
> >the system, with virtually NO application code at all.

>
> Stored procedures ARE application code.

>
Actually, I do know that... (been in this game a while now) <G>. I should have said "non-database resident " application code. I made the mistake of thinking it was obvious... Even after absorbing the Doc's discourse on the subject. Must be getting old...

> >The team I last
> >managed tried to persuade me in this direction, and they were very clever
> >people who had spent their whole professional lives working on this kind
of

- > > *technology for COMPAQ. Guys in their 30s with 15 years experience, all on*
- > > *Database driven systems <G>. For COBOL this is not a good solution ..*
- >
- > *Sure it is. On Oracle, at least, stored procedures can be written in*
- > *any language, including Cobol. The technical term is External*
- > *Procedures. They work just like stored procedures, with an entry in a*
- > *Library table pointing to a .dll or .so.*

>

The difficulty is when you get into extended SQL which has facilities that are database dependent and not available in COBOL. So you embed this SQL into COBOL. What for? It is perfectly capable of standing alone on the database without a COBOL wrapper. I understand you are talking about writing stored procedures in COBOL and I know this facility has been available for some years now. But with extended SQL you simply write the procedures in SQL and there is no requirement for any other host.

- > > *.. and I*
- > > *still believe that embedded SQL should see the DB as purely a repository.*
- > > *with no application code embedded in the database. But OO solutions may well*
- > > *use things that COBOL is not good with, and there could be a case for*
- > > *Extended SQL stored procedures and triggers, in conjunction with transaction*
- > > *processing, and no COBOL (or ONLY minimal OO COBOL...)*
- >
- > *No Cobol?! Are you trying to put us out of work? :)*

>

No, Robert, the COBOL community has largely done that for itself. I am simply looking at solutions and trying not to be biased. For years now I have been advising COBOL people to expand their skill sets. This is an area that might be worth investigating. These triggered procedures can be driven from web based code and scripts as well.

Pete.