

Re: Sorry, but not happy with D2007

Source:

<http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.non-technical/2007-03/msg03169.html>

- *From:* David Erbas-White <derbas@xxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 19 Mar 2007 16:35:56 -0700
-

Allen Bauer (CodeGear) wrote:

This has been the case since day-one of Delphi. The intent here is that until you save or compile, the dropped components aren't "committed." This was a specific design decision since once the uses clause was updated, it is difficult to distinguish between which units the user intended to be there and which ones are only there because of components. On the surface this seems easy, right? The problem comes in when the user manually adds a unit to that list and then later drops a component on the form from that same unit, then deletes that component. Do we keep the unit in the uses list? Do we delete it? There was a *lot* of thought that went into this design back in 1993-'94. A very similar thought process was involved surrounding the auto-removal of empty event handlers when saving or compiling. The simple idea was to allow the user to "try out" lots of things without introducing "cruft" from previously abandoned changes. It is not until the user saves or compiles the source do we consider that they are "done playing" and the changes are then made permanent. It also allows the user to easily correct a change like dropping a TButton on the form when they really wanted a TBitBtn (or vice versa).

Here's the fallacy in this thought (and bear in mind I use BCB5 mostly, but it relates).

The IDE crashes often enough that I automatically hit the 'save' key every few minutes, and there are often times I prefer to put a program stub in place to 'remind me' that I intend to put an event handler there. For example, if I drop an item on a form I typically put stubs in the event handlers that I believe I'll need so that I don't forget about them.

The fact that you've tried to make this 'seamless' fails on two fronts. Because you 'automatically' remove all empty events, it forces me to take the EXTRA time to always had a comment line in each event, which I wouldn't have to do otherwise. And because the IDE crashes so often, means that I have to take the extra actions of saving, PLUS I don't have the advantage of 'trying out' things without extra stuff being added in.

One of my current programs is (frankly) a mess because of the 'crap' that's been added 'automatically' over the years -- but it's too much trouble to actually try and go through and see which items that are included are

Re: Sorry, but not happy with D2007

actually NEEDED.

What would be far better (aside from the obvious of improving stability) is that these options be able to be specified by the user. It should remove empty events, or not, based on a checkbox — and there should be a method to FORCE removal of empty events.

Fast-forward to today with the advent of Error Insight and continuous background compilation. While the unit is in this "uncommitted" state, the background parser only sees the source as it exists in the unsaved editor buffer. So the uses list hasn't been updated, which causes the "false" errors you're seeing. I definitely can see why this would tend to confuse the uninitiated. This has even tripped up the long-time Delphi users, since many of them have never really noticed or worried about this intermediate "uncommitted" state. The source was always updated, just-in-time so that when saved or compiled everything was just fine.

This is clearly an area where we should look at improving the user-experience. One thought would be to hand this "uncommitted" data off to the background parser that it would then use to "fill in the blanks" during this intermediate phase.

My concern in seeing this is that there was no 'advent' of Error Insight that just occurred through natural selection. At some point there should have been some hard, concrete thought put into how adding items such as Error Insight would AFFECT these early design decisions. At the very least, AT THAT POINT IN THE DECISION PROCESS, cleaning up those design decisions should have been put on the 'to do' list. The fact that a software tools company appears to have been caught 'off guard' by the 'advent' of these new features does not engender a feel of security in the development process...

David Erbas-White

.