Subject: Re: Object epiphany: A new way of building widget applications
Posted by John-David T. Smith on Thu, 05 Apr 2001 14:02:25 GMT
View Forum Message <> Reply to Message

OK, I feel I need to clarify just a bit more. I didn't mean to hit such a sensitive nerve. I am not *at all* questioning the motives and qualities of the software being released with the naming convention in question. Not even sort of. I appreciate, just like everyone else, the ability to pick through outstanding pieces of work coded by talented and dedicated individuals like yourself. Heck, sometimes I even use contributed routines as-is ;) So I'm sorry if my opinion cut too deep.

Maybe it was a poor choice of timing, but all I really wanted to do was alert contributors of a potential hidden meaning their naming style might convey to a certain (apparently small) portion of their intended audience. I did not mean to appear critical.

As far as the question of inheriting and updating older code (like FSC_FIELD), this is a trickier issue. One thing that is certain to confuse users, however, is multiple *_FIELD routines out there. Which do I use?

Sometimes this is unavoidable. But consider an example. Suppose in some alternate universe I was good enough at lisp to redo the IDLWAVE mode of emacs substantially to suit my own peculiar needs (not that Carsten doesn't bend over backwards to do that now... but this is hypothetical...). I have two options.

1. Fork the codebase, rename the mode JD-IDLWAVE, and begin distributing a competing version.

2. Talk to Carsten about the features I'd like to see, and come to some agreement about who should really be maintaining it, given the new information.

Version 2 is much trickier, obviously. It takes cooperation, potential conflict resolution, and perhaps compromise. If JD-IDLWAVE were for internal consumption, it probably would not be worth it. But, if JD-IDLWAVE were intended to benefit the larger community, the experience of "real" programmers (unlike myself) has shown that a consistent, continuously evolving work will last longer, receive more attention, and ultimately benefit everyone more, despite the transition to a new maintainer/coder. Fractionation confuses. Unification encourages.

Anyway, these are just very different approaches. And now, back to your regularly scheduled program (and sorry for the interruption).

I close with a truer wisdom:

Honest disagreement is often a good sign of progress.
--Mahatma Gandhi (1869 - 1948)

JD