Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by gprentice on Sat, 20 May 2006 11:11:03 GMT
View Forum Message <> Reply to Message

Quote:1c) My whole theide starts becoming like a "third party tool" Smile but I don't want that! It adds headaches... I want kind of "Firefox" extensions and/or pluggins mechanism! I want an easy exchange between users contributions... Can C++ deliver that?

A possibility was discussed here briefly.
http://www.arilect.com/upp/forum/index.php?t=msg&goto=27 75&&srch=dll#msg_2775

I do not know what the best architecture would be for providing a plugin mechanism for U++ APPs like theIDE. It should be something that avoids the problem JEdit plugins have where the plugins are constantly breaking as JEdit continually moves to a new revision of Java.

TheIDE could provide a C API and a mechanism for loading plugin DLLs that allowed a dll to do things like select the main package or open/close files or dock windows/frames within theIDE.

Another way is for the Esc macro language to allow creation of widgets and forms and for theIDE to provide additional system functions similar to the existing ones that let you build a project. Extending Esc to allow creating of forms/widgets would be great and useful for all U++ apps but a lot of work.

The least work and very effective would probably be for third party extensions to theIDE to be standalone packages or source files built as part of theIDE and for theIDE to provide an interface for functionality such as setting the main package, building a package or selecting a source file etc. Of course this has the problem that the standard release of theIDE can't really include every third party tool that comes along as part of its build, but rebuilding theIDE is not that hard and could even be hidden behind a mechanism that allowed end users to select which third party extensions they wanted to install. However, the result might depend on which version of which compiler used, but a list of "compatible/certified/verified" compilers could be provided. This only works if the application distribution includes the source code, which I certainly wouldn't do if I was trying to make money, so it's not a general solution.

Slickedit provides a "macro language" called slickc and large amounts of slickedit itself are written in slickc and provided to end users for modification and extension. Slickc code itself is actually executed by an interpreter and it's amazingly crash proof - meaning end user slickc code that does silly things doesn't actually crash the editor itself. It also includes a GUI mechanism for creating forms with edit boxes, buttons etc. that can be docked within slickedit.

How do Firefox extensions/plugins work?

Graeme