Subject: Re: subclassing LineEdit is ugly
Posted by hojtsy on Sun, 09 Apr 2006 21:29:16 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 09 April 2006 16:11Ah, well, but where this should stop? Should we make all methods everywhere public and virtual? Lets not mix public and virtual, I was only asking for virtual. LineEdit is clearly made to be subclassed, because the fields and methods are protected and not private. Providing a way to subclass a Ctrl but making it ugly doesn't seem to make sense. Are you trying to do speed optimization with keeping the inflexible non-virtual signatures? The selected methods are only called when a key is pressed, not 1000 times a second, so the minimal cost of virtual call would be unnoticable. I invested some time to find these methods of LineEdit where the behaviour could be most easily changed, not only for this Console thing, but other modifications of LineEdit.

Let me put an example. Somebody wants a LineEdit in which Shift-Tab unindents only if none of the selected lines are already fully unindented. It is clearly a subclassing situation: you are replacing only part of the behaviour. But AlignChar is non-virtual for whatever reason. So you need to do all these steps:
1) create a new method instead of AlignChar
2) override the method Key, copy-paste long code from LineEdit
3) call your new method from Key, instead of AlignChar
4) pray that no further subclasses, or methods of your subclass would accidentally call AlignChar instead of your new method
5) every time a new version of library comes out try to sync the changes to the copy-pasted Key method

Returning to your question whether all methods everywhere should be virtual: I think that complex library classes should be easy to subclass, not just possible, which means to me that
1) any non-speed-critical and non-trivial methods of complex classes should be virtual, and
2) long and complex methods implementing multiple aspects of the behaviour (such as the monster Paint in several Ctrls) should be broken up to multiple virtual methods, to enable overriding only one of them

My reasoning for this is that when you are developping an application yourself and need a subclass it is very easy for you to just make the needed method virtual in the base class, or just insert a branch in the library code itself. But for the clients of the library we are stuck with the amount of flexibility which is readily provided by the library. Imaging working in an environment where you can not change the library, but required to provide slighly different behaviour in some classes. This different working method places different requirements on the library, which may not be realized by you while working on one of your own applications.