Subject: Re: U++ 2019.1.rc4 released
Posted by mirek on Thu, 18 Apr 2019 12:13:34 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Thu, 18 April 2019 14:02The issues is that TheIDE CtrlCore constantly overrides your changes. In a very counter-intuitive way. Chameleon has been doing this for over a decade now. If you want to correct something, you can't easily do it. If you want to implement your own theme on top of the current dark mode? It is fairly hard. It is fairly hard to force a dark app on a light windows or the other way around. Making TheIDE do the same thing is even harder because it also constantly overrides you changes in many places. The same for code editor.

Chameleon has always been a bit hard to use and counter-intuitive.

Maybe because the problem itself is hard and counterintiutive?

Anyway, missing docs about Chameleon is not helping. Will try to fix than.

Quote:
Quote:SetDarkThemeEnabled tells U++ that application is ready to handle the dark theme. It in fact has any meaning only in windows. If you activate that, app starts following theme setting in Windows 10.
IMHO that is a counter-intuitive and problematic design. SetDarkThemeEnabled means "Set my theme to dark": change colors and skins.

OK, we can argue about the name and I am willing to change it. But I really need that function to do exactly what it does.

Quote:
Why cant U++ detect multiple themes and be coded to religiously follow those themes. So that if you change them, I don't have to track down dozen of places to make things just work.

I would use a design where you have a Theme, all theme aware components just read from the theme, nobody can write, the theme knows if it

Thats centralized model which is not extensible - that is the issue I wanted to avoid.

Quote:
- use the theme (just set a pointer and force the gui to refresh)

void Ctrl::SetSkin(void (*_skin)())

Mirek