Subject: Re: Support for plug-in architecture
Posted by mirek on Sat, 21 Mar 2020 08:54:27 GMT
View Forum Message <> Reply to Message

slashupp wrote on Sat, 21 March 2020 03:21(linux)

I have an app that will work well with plug-ins, i.e. using custom-controls that are compiled
into dynamic shared libs that can be loaded by the executable without needing to recompile the app itself.

My current design generates all as one big blob-executable and needs to be re-build for each small
change or addition of a custom-control, but would be much better using plug-ins.

Since Windows DLL's can be created, why can the linux-equivalent of dynamic shared object (.so) not
be created? It looks like a simple change to compiler and linker flags that will enable this?

I've hacked the [Setup/Build methods] flags to produce a .so lib, which works with well with
'extern "C"'-functions, but fails to pass an accessable custom-control (which is mangled C++) back.

Barring out-of-the-box support for plug-in/.so development, is there any other way to implement a plug-in design using Upp?

Not at the moment. Somehow this is not much required by typical U++ apps - which quite often are niche engineering applications or custom bussines solutions with tens to hunderds users. Not much need for plugins there.

That said, I am really not sure what is wrong with .so, IMO it should work for mangled C++ names as well.

Anyway, the trouble with .so and C++ is usually the problem that it is very fragile. Swap two virtual methods, add single member variable and the whole thing just explodes... :) When I was thinking about the best approach, I have tended to think about separate processes and RPC communication.