Subject: Re: Understanding packages, assemblies, and nests
Posted by eldiener on Sun, 15 Mar 2015 16:02:08 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 15 March 2015 03:15Quote:
I do not understand why non-main packages do not have build flags. Don't non-main packages have to be built also ? Suppose I create a shared library use the IDE. I assume a shared library is another package, just like anything else being built in the IDE from source files. Don't I need build flags to tell the IDE how to build the shared library ?

This is not the purpose of build flags. They are usually used as 'binary-wide' settings. Some of them are recognized by build system (e.g. MT says that application is multithreaded, which affects the way how packages are compiled, GUI says that it is gui application, which is recognized by Win32 builders etc...), some are application specific, e.g. developer can decide to recognize 'DEMO' build flag, which would restrict the functionality of resulting executable. Your main package can have multiple main configurations.

Quote:
Regarding have the IDE add a package which already exists as a .upp file: are you saying that this package needs to go in an already existing assembly ? Or can I create a new assembly for the package ? Are assemblies just a grouping mechanism for packages in the IDE so that a user of the IDE can open the package by finding it within a particular assembly ?

Yes. It is like include path or say system 'PATH' (for finding executables).

Quote:
Can a package be in more than one assembly, ie. can different assemblies have the same nest in its list of nests ?

Definitely!

More importantly, you can use assemblies to e.g. select version of library. Say you have 'stable' libraries and 'experimental' - you can have two assemblies, one will group your main project with experimental, other with stable library.

Quote:
I also see that one can choose a non-main package to be opened in the IDE, but the menu item says 'File | Set main package...'. Isn't this a misnomer since the end-user can choose a non-main package to build ?

True. Practical needs sometimes blur things... perhaps it would be more correct to have 'main package' and 'package with main configuration' as two distinct terms, but later is too long...

Quote:

You wrote "Usually, your package would by in MyApps, and it would depend on for example on Core, which is located in the uppsrc nest." Does this mean that packages which the end-user creates should go into the MyApps assembly ? I am gathering that the other assemblies are for Ultimate++'s own use, except maybe for MyAppsWBazaar.

It is something like initial suggested setup. E.g. when I start theide on my work laptop, I have about 40 assemblies in the list :)

Quote:
3) A main package is never a dependent of any other package.

Actually, there is no such limitation. Main package in fact can be used in another main package dependency. There is even a macro 'flagMAIN' that is only defined when package is build as main.

Usage scenario: Imagine I would want to create text editor which would provide much of functionality that 'theide' provides, but I would be too lazy to correctly split ide main package sources into correct 'non-main' package. I could have all sources of theide included into new main package by simply adding theide into now project and perhaps putting some of #ifdefs flagMAIN into it. It is not typical, perhaps not even recommended, but sometimes ends justify means...

(rest of your points is correct).

Mirek

Thanks for your explanations !

I still do not understand the difference between a main package and a non-main package as far as Ultimate++ is concerned. Why divide your idea of a package into these two distinct categories ? In other words what does a main package consist of that a non-main package does not have and why is this important to either the IDE or to a person using Ultimate++ to create executables or libraries to be run cross-platform ?

Your explanation suggests, although I might be misunderstanding it, that a main package can have more than one way of building it but a non-main package only is built in one particular way. Is this the distinction between them ? Because if it is I can conceive of very few things which are built in only one way. A library as well as an executable often has different build configurations. But if this is the single distinction between a main package and a non-main package then I do understand the difference. I just don't understand why this distinction is important to the IDE or the programmer using the Ultimate++ IDE.

Also I think I do understand assemblies/nests as merely grouping mechanisms in the IDE in order to specify a package or a particular build of a package in a particular category (assembly). If they exist as anything else please tell me.