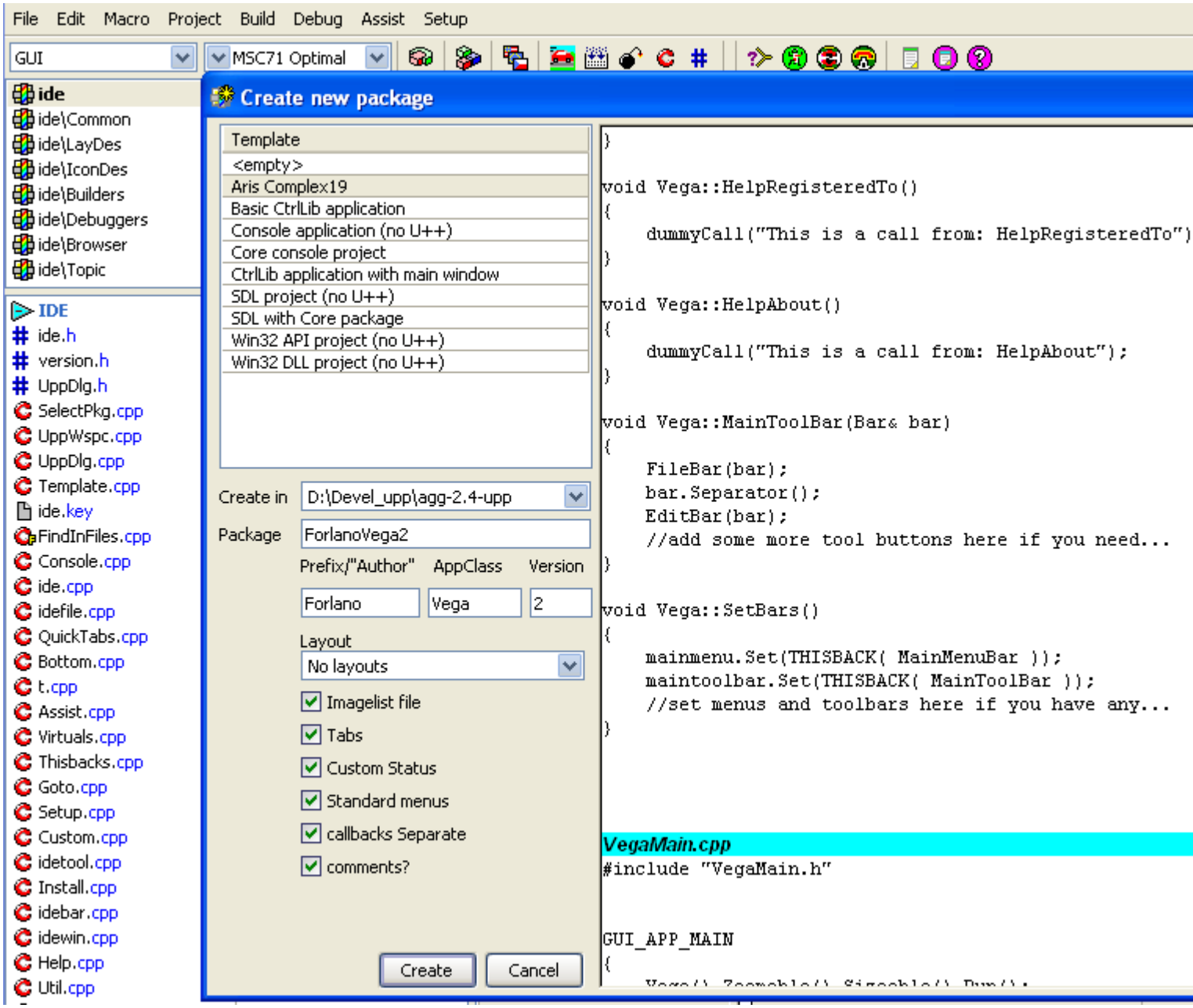
Subject: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by fudadmin on Tue, 16 May 2006 16:01:12 GMT
View Forum Message <> Reply to Message

When creating my "clever" templates (e.g. menus generator) I had to use "New Package" interface a lot and I found it quite inconvienent. Some problems I sorted out for myself a long time ago. But in order to make those templates useful (even like tutorials...) for wider public - some changes (or workarounds) would be needed for the official theide.
So, what do you think?:
1. Any fields (was package name?) above templates table are visually tiring.
2. But the most important and useful thing I am using is a compound Package name which is automatically concatenated and consists of Prefix/"Author"+AppClass+version and it doesn't take 3 rows unlike in a case of id's.
3.Also, then, all files names in a package are formed from a short AppClass name (and not long Package name) ...

File Attachments

1) NewPackageCompound.PNG, downloaded 2730 times



Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by fudadmin on Tue, 16 May 2006 18:39:40 GMT
View Forum Message <> Reply to Message

One more idea would be to use ArrayCtrl for user choice id's...

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by fudadmin on Wed, 17 May 2006 15:01:20 GMT
View Forum Message <> Reply to Message

Has anybody noticed this post...?

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by gprentice on Thu, 18 May 2006 09:31:42 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Thu, 18 May 2006 03:01Has anybody noticed this post...?

Well I noticed it but I can't understand a thing you're saying ... of course my opinion doesn't count for a lot but if you want some feedback ...

What do the names of the files in a package have to do with the length of a package name?

Does "automatically concetenated" mean "automatically generated"?
What is appclass?
What is author?
What is prefix?

What problem are you trying to solve?

Graeme

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by forlano on Thu, 18 May 2006 10:27:05 GMT
View Forum Message <> Reply to Message

If I've understood the structure of the code you have generated for my app was done on the fly with your template. It is very useful. In fact I've not touched it in that it was well organized. My opinion is favorable.

Luigi

PS: Each time I add a new *.cpp file the important include appear automatically. Even that is due to your template? What I've to modify if I want to add automatically a new include when adding a new cpp file?

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by mirek on Thu, 18 May 2006 12:21:20 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Wed, 17 May 2006 11:01Has anybody noticed this post...?

I did. Sorry for not responding to everything... Still working on completely unrelated parts of U++....

Ad 2) I am not sure whether it fits everyone's needs. It certainly does not fit mine.

Mirek

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by fudadmin on Thu, 18 May 2006 19:11:07 GMT
View Forum Message <> Reply to Message

forlano wrote on Thu, 18 May 2006 11:27If I've understood the structure of the code you have generated for my app was done on the fly with your template. It is very useful. In fact I've not touched it in that it was well organized. My opinion is favorable.

Luigi

PS: Each time I add a new *.cpp file the important include appear automatically. Even that is due to your template? What I've to modify if I want to add automatically a new include when adding a new cpp file?

Yes, I'm very glad you have asked this particular question... . Some of the problems I'm trying to solve is that with my system it would be possible:
1. - with Esc and/or current macro system analyzing the existing template and/or the package/s generated - to add new files and some/all changes to a new version template-package and automatically sort out all the dependancies.
I hate when I have to edit all those #include by hand. Especially when I want to addopt new packages to U++. I want a kind of inteligent "
template-package-template-version-incremental-upgrader-inclu der " system. Current Packages are too big pieces for me (Java and interpreters clearly have their advantages over C++...). And what if I need a several combinations of Core or etc.? Or to maintain a lot of very similar versions but individual versions to different clients? And because Mirek is "not interested" in many changes I have to maintain my own theide and libs versions. This is natural. One size can't fit all. Maybe not ideal extreme would be to have all the code templated in a database and to assembly (also visually!...) like Lego(tm)... .
2. - even more power would come if connected with uvs/svn.
3. - instead of sending a package or svn updates the users would send some params to click to generate a package...
4. - instead like a new user Luigi spending (1 month?) to create a skeleton for his program consisting of "all-must-have" nearly standard menus, tabs , status bar, file opening-saving, help system, etc. etc. he could have it all with one click... or adjust it with several clicks and param entries.
And immediately (after ~1 min!..) start playing with his data, controls and callbacks and customizing it...

5. - from "create a new package" it would work like a kind of "Programmable-Program-Structure-Layout-Designer".
You can call it a "fancy generator" but it would save quite a lot of time even for professional users, especially, generating menus. Meanwhile, for the new users it could serve like an "incremental-tutorial-in-action+code-snippets-database".
6. - from inside theide I want it to work like a Templator++... by inserting/including name-parameter-parametrized metods/pieces of code.
7. - similarly to WideStudio it would be able to produce code for different programming languages, GUI toolkits and/or translate from each other. But to achieve that fast maybe I need something more productive than "the most productive language in history and its tools..."

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by fudadmin on Thu, 18 May 2006 20:20:49 GMT
View Forum Message <> Reply to Message

forlano wrote on Thu, 18 May 2006 11:27

PS: Each time I add a new *.cpp file the important include appear automatically. Even that is due to your template? What I've to modify if I want to add automatically a new include when adding a new cpp file?

Sorry, actually, I've already forgotten which template I had used for which of your package. This is one more prove that sophisticated templates must be shipped with packages even if you can't use them... I'll try to think about your wish.

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by fudadmin on Thu, 18 May 2006 23:36:26 GMT
View Forum Message <> Reply to Message

gprentice wrote on Thu, 18 May 2006 10:31fudadmin wrote on Thu, 18 May 2006 03:01Has anybody noticed this post...?

Well I noticed it but I can't understand a thing you're saying ... of course my opinion doesn't count for a lot but if you want some feedback ...

What do the names of the files in a package have to do with the length of a package name?

1. Does "automatically concetenated" mean "automatically generated"?
2. What is appclass?
3 a) What is author?

3 b)What is prefix?

4. What problem are you trying to solve?

Graeme

Thanks for your feedback, Graeme!

4.
 Generally - to increase programmers productivity. Especially beginners. By using highly customizable templates and not only.
But...
 -there is not enough space for "options" in the current dialogue window.
 -it's not possible to assign (write) to :PACKAGE: variable from a template code and have that reflected in the window.
4 and 3 (a) What is author?
3 b)What is prefix?)

It is related to versioning.
 -usually people name their packages consisting of at least to words and a version (you can look at the upp examples and reference examples...) and if you work in a team or want to experiment with packages and have them at your convienence to compare... Or if a begginer wants to experiment with Ulimate++ examples? Don't tell me that 100% programmers (especially beginners!!!) in 100% of cases use CVS, SVN or uvs! Or even if you DO use - do you use the same name to invoke recompilation of all or most of your package files? And then don't forget those stupid BLITZ problems...
 -so how do you copy a new version of a package in u++? what are your actions? how much time do you spend on it? nice problem #include's! Then how do you delete your unwanted packages? then look at the ALL U++ directories? Config files... If you have several hundred packages multiplied by debug,release,GCC, and versions etc..? Then some data files somewhere. Analyze your routine actions... It's a BIG headache and a big waste of time!..
 - "author" or prefix - in case you work with a partner or a team...
 - that helps to preserve the same names of your files inside packages while maintaining different names of versions. Imagine a copy of CtrlLib with all files named under a package name... This is what the current system requires... then try to have different amount of files generated...
 - have you analyzed the supplied templates and tried to improve for your needs... Have you seen:

id   "Main window class name" classname ?
that should answer
 2. What is appclass?

And 1. Does "automatically concetenated" mean "automatically generated"?
When you enter those 3 fields, WhenAction concats and displays the Package name...
(It's no problem to have App classname read from a template but then more changes are needed in theide to callback and display, as I imagine...)
And then depending on options selected...
"A fair chunk C++/U++ program skeletons - produced in one click"

Edit:P.S. According to Luigi they are not bad...
Edit2: Btw, after those changes the next step is available - incremental duplication of packages with a right click from "Select Main Package" window...

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by gprentice on Fri, 19 May 2006 10:23:32 GMT
View Forum Message <> Reply to Message

Hmmm. Thanks for the detailed reply. I'm slowly getting my head round what you're saying. I remember you talking about "finishing your templates" a little while ago. I still don't fully understand though ...

It seems you've created your own version of the "create new package" dialog (with a nice icon!), perhaps based on existing upp code ??? - can't you submit this as a "third party tool" though? Why do you need theIde package dialog to change ??

I did notice a while ago the existing "create package" dialog was in danger of running out of room for extra checkboxes from user created templates.

I'm not sure I understand but it seems the purpose of the author/prefix/version items is to make it easier to duplicate the package in future, however I don't understand why this makes it easier ?? Is your "duplicate package" command going to go through all the package source files and change #include <packagenameV1/xxx.h> to #include <packagenameV2/xxx.h> ??

With the HelloWorld example, you can duplicate the package by copying the directory and calling it HelloWorld2 and renaming the upp file to HelloWorld2. The source files in the package don't explicitly use the package name in any #includes.

I'm wondering why would a package source file use its own package name in a #include ??. In the top level package folder, you can just do #include "xxx.h". For header files that are in sub folders of the top level package folder I guess you need to use the package folder name sometimes - you can also use relative paths though (e.g. ../../something/xxx.h)

My feeling is that packages that have nested folders are "serious" packages that don't get duplicated very often and can possibly use the assembly nest path mechanism. However I'm not sure I really understand so I don't want to pour cold water on what you're doing

(Why are there 61 downloads of the png attachment on your post when you can see the image in the forum ???).

Anyway, I admire your enthusiasm

Graeme

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by forlano on Fri, 19 May 2006 11:50:43 GMT
View Forum Message <> Reply to Message

gprentice wrote on Fri, 19 May 2006 12:23I'm not sure I understand but it seems the purpose of the author/prefix/version items is to make it easier to duplicate the package in future, however I don't understand why this makes it easier ?? Is your "duplicate package" command going to go through all the package source files and change #include <packagenameV1/xxx.h> to #include <packagenameV2/xxx.h> ??

You have anticipated my same question.
Quote:
With the HelloWorld example, you can duplicate the package by copying the directory and calling it HelloWorld2 and renaming the upp file to HelloWorld2. The source files in the package don't explicitly use the package name in any #includes.

Instead it seems the package name appear in the sourcefile although not in the #include. I'm referring to the LAYOUT and IMAGE file via #define. For example in my code appear:

#define IMAGECLASS   Vega3Img
#define IMAGEFILE   PACKAGE_DIR/Vega.iml>
#include <Draw/iml.h>

#define LAYOUTFILE   PACKAGE_DIR/Vega.lay>
#include <CtrlCore/lay.h>

Where PACKAGE_DIR is in turn substituted by a #define with the package directory. A way to proceed that is not satisfactory as you noted in another post but it works.

Luigi

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by fudadmin on Fri, 19 May 2006 21:35:17 GMT
View Forum Message <> Reply to Message

gprentice wrote on Fri, 19 May 2006 12:23I'm not sure I understand but it seems the purpose of the author/prefix/version items is
1. to make it easier to duplicate the package in future, however I don't understand why this makes it easier ??
2. Is your "duplicate package" command going to go through all the package source files and change #include <packagenameV1/xxx.h> to #include <packagenameV2/xxx.h> ??

1. Yes. And easier because of:
2a. First of all, try duplicate (your usual "copy directory" way) not HelloWorld but packages which have *.iml and *.lay. Tell your experience!

2b. Imagine you work with a team. Try to produce 2 different packages from 1 template... Or in 2 steps create 2 packages (a package which uses the other package which will be also versioned (with *.iml and *.lay). And test quickly 2*2 combinations. Experiences?
Or imagine creating a step-by-step tutorial-in-action... with different subtopics and combinations...
Edit2: Or having your testable and combinable (!) your code snippets database...

2c. And because the trick (remember, we were talking about)

```
 #define IMAGEFILE     PACKAGE_DIR/Vega.iml>
```

doesn't work well with GCC and BLITZ...

2d. And/or create a template which simply produces
#include ForlanoVega1/Vega.h etc...
customizable and shows them in the dialogue each time you enter a char!..

2e. And/or updates those names according to template params...

Edit:
2. Yes. and *.upp

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by fudadmin on Sat, 20 May 2006 00:43:24 GMT
View Forum Message <> Reply to Message

gprentice wrote on Fri, 19 May 2006 11:23

1. It seems you've created your own version of the "create new package" dialog (with a nice icon!), perhaps based on existing upp code ??? - can't you submit this as a "third party tool" though? Why do you need theIde package dialog to change ??

2. I did notice a while ago the existing "create package" dialog was in danger of running out of room for extra checkboxes from user created templates.

...
3. Anyway, I admire your enthusiasm

Graeme

1a) Well, for even a "third party tool" there must be a mechanism(s) inside theide... Then, I don't want to duplicate the pieces of code which already present in teide. I want to use them. Maybe *.usc widgets would serve better for such "kind of pluggins" or extensions?

1b) Also, I don't like to have many clicks and starting-closing theide... with Assistant++ reading times (solution automated on/off?). I also support unodgs tabbed workspaces idea.

1c) My whole theide starts becoming like a "third party tool" 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?

2. By writing "etc." in the topic name I had in mind that. And "danger of running out of room for extra checkboxes from user created templates" I regard as a bug... But thinking about the possible solutions drove me to the wider things I've written in this topic.

3. Thanks for your time and appreciation! I'm just looking for ways and trying to make U++ more useful for myself and other users and more attractive for wider public. I see "templates-packages-creator-manager", "templates-packages-tutorial", "templates-packages-snippets-database" - things which can speed up learning and using not 4 times but 100's or 1000's.

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

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by gprentice on Sat, 20 May 2006 11:37:51 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Sat, 20 May 2006 09:35gprentice wrote on Fri, 19 May 2006 12:23I'm not sure I understand but it seems the purpose of the author/prefix/version items is
1. to make it easier to duplicate the package in future, however I don't understand why this makes it easier ??
2. Is your "duplicate package" command going to go through all the package source files and change #include <packagenameV1/xxx.h> to #include <packagenameV2/xxx.h> ??

1. Yes. And easier because of:
2a. First of all, try duplicate (your usual "copy directory" way) not HelloWorld but packages which have *.iml and *.lay. Tell your experience!

2b. Imagine you work with a team. Try to produce 2 different packages from 1 template... Or in 2 steps create 2 packages (a package which uses the other package which will be also versioned (with *.iml and *.lay). And test quickly 2*2 combinations. Experiences?
Or imagine creating a step-by-step tutorial-in-action... with different subtopics and combinations...
Edit2: Or having your testable and combinable (!) your code snippets database...

2c. And because the trick (remember, we were talking about)

 #define IMAGEFILE   PACKAGE_DIR/Vega.iml>

doesn't work well with GCC and BLITZ...

2d. And/or create a template which simply produces
#include ForlanoVega1/Vega.h etc...

customizable and shows them in the dialogue each time you enter a char!..

2e. And/or updates those names according to template params...

Edit:
2. Yes. and *.upp

Ok, so layout files are a problem because they include path information in the LAYOUTFILE #define - but it's not hard to search for all LAYOUTFILE in the package and change the pathname. Similarly with #includes that include the package name - they can be changed very quickly with a regular expression and global replace over all package files, but it seems you're duplicating packages so frequently this probably doesn't appeal to you

Unfortunately, the mechanism you've used, which the uninitiated would, not unreasonably, expect to work, probably doesn't work portably

Graeme

Subject: Re: Compound New <:PACKAGE:> name etc. [FEATURE REQUEST]
Posted by fudadmin on Sat, 20 May 2006 12:35:09 GMT
View Forum Message <> Reply to Message

gprentice wrote on Sat, 20 May 2006 12:11

How do Firefox extensions/plugins work?

Graeme

Extensions are javish jars. I've tried to program them a little. But I don't know anything about their mechanisms from Firefox side. I mentioned them just as easy-to-use examples.
