UPT "Function Test Frame" and UPT "Function Test"

Documentation

0. Contents

1. Prologue

UPTs (Ultimate++ Package Templates) can be quite powerful tools. Feel free not only to use the below described UPTs but also to learn from them how to write your own.

UPTs are described in "Help Topics / Ultimate++ Project Templates".

And if you like your own UPTs, why not make them available to the community?

2. Description

The common purpose of "Function Test Frame" and "Function Test" is to simplify the creation and test of (non-member) functions which process textually representable values.

"Function Test Frame" generates the files necessary to write and test functions, but these files lack almost all meaningful content. You still have to write/paste not only the function body but all the test logic, too.

"Function Test" is a little bit more sophisticated. You still have to write/paste your own function body, but the complete test environment is already set up and ready to be used. On the other hand you are restricted with regard to return and parameter types.

The package "FnTest.zip" contains the following files:

FunctionTestFrame.upt:	the "Function Test Frame" package template itself
FunctionTest.upt:	the "Function Test" package template itself
FunctionTest.ico:	the "Function Test Frame"/"Function Test" icon

FnTestFrame_FnTest$en-us.tpp: this documentation (Ultimate++ help format)
FnTestFrame_FnTest$en-us.pdf: this documentation (PDF format)

3. Installation

Put "FunctionTestFrame.upt", "FunctionTest.upt", and "FunctionTest.ico" in your "MyApps" directory.

That way the UPTs are available whenever you select your "MyApps" assembly. This makes double sense: You normally need these UPTs only when you write your own applications which you are supposed to build in subdirectories of "MyApps". And this location is secure. Updating Ultimate++ doesn't erase this directory as is true for "Common" and "UppLocal".

Please note that the hard-coded .rc-file part of the UPTs expects "FunctionTest.ico" to reside one directory above your package directory.

4. Overall Issues

4.1 "Create new package window" Layout

When you select "Function Test Frame" or "Function Test" for the first time, the bottommost parameter input field might hide the buttons "Create" and "Cancel". This is caused by the current hard-coded window layout. Just enlarge the window to see the missing interface elements.

4.2 Function Name

Enter the name of your function which you want to write/paste and test into the "Package name" field of the "Create new package window". Please stick to the C/C++ naming convention. For example, the input field allows the entry of "/" which must not be used as a character of an identifier.

The UPT will create a package named "*content of the package name field*" with the following files:

content_of_the_package_name_field.h[pp]
content_of_the_package_name_field.cpp
*content_of_the_package_name_field*Test.h[pp]
*content_of_the_package_name_field*Test.cpp
*content_of_the_package_name_field*Test.lay ("Function Test Frame" only)
*content_of_the_package_name_field*Test.iml
*content_of_the_package_name_field*Test.rc
content_of_the_package_name_field.upp

4.3 Return Type

Although the input field permits any character, do not use leading or trailing white space. The UPTs are not able to filter these characters.

It does make sense to have "void" as return type because you can have and test reference parameters.

4.4 "void" Parameter List

Do not use "void" to indicate that your function doesn't take arguments. Just leave all the parameter input fields empty.

4.5 "Aftermath"

As soon as you call "Create", the package is written and ready to be built, provided that your input data is legal C/C++. The only exception is the case mentioned in topic 5.3. Of course you still have to add your own work to get a useful application.

5. Function Test Frame

5.1 Purpose

"Function Test Frame" generates only file skeletons. Although a layout file is provided, you still have to write/paste not only the function body but all the test logic, too. At least, the basic work has already been done - as the name implies: this UPT is just a package frame. And you have almost every freedom to do what you want to do ...

5.2 Extension Field

You can choose whether your include files end with ".h" or ".hpp". ".hpp" is preselected. (Because of space constraints this option is not available with "Function Test").

5.3 Non-"void" Return Type

If your function returns something, the generated package will not compile successfully until you write/paste an appropriate function body.

5.4 Parameter Fields

Besides the return type field you have six parameter fields to enter zero to six parameter declarations, i. e. any type stuff and the name.

If it makes sense to you, you can use or not use any of these slots. The UPT can handle empty parameter fields at any position.

6. Function Test

6.1 Purpose

"Function Test" provides an almost complete ready-to-run test application. You just have to add your function body to build a useful application. To make this possible via a "simple" UPT you are restricted with regard to return and parameter types. You can use in any combination:

void (return type only)
bool&, bool &
int&, int &
double&, double &
String&, String &
Date&, Date &
Time&, Time &

Other stuff, pointers e. g., is not supposed to work unless you patch heavily.

6.2 No Extension Field

Due to space constraints, in contrast to "Function Test Frame" you cannot choose whether your include files end with ".h" or ".hpp". They always end with ".hpp".

6.3 Return Type

Please be especially careful with white space! To adjust the output to the type, the UPT analyzes your input by very limited means. Leading or trailing white space or - in case of a reference - more than one space in-between will foil a successful build.

6.4 Parameter Type Fields

Besides the return type field you have four parameter type fields to enter zero to four parameter declarations. Please do not enter neither a qualifier or a specifier nor a name into a parameter type field. The former are to difficult to handle for the current version of "Function Test". The latter - if any - goes into the according parameter name field.

Please be especially careful with white space! To adjust the input fields to the types, the UPT analyzes your input by very limited means. Leading or trailing white space or - in case of a reference - more than one space in-between will foil a successful build.

You can enter a type without entering a name into the according field (but not vice versa) if this makes sense to you.

If it makes sense to you, you can use or not use any of the parameter type/name field pairs. The UPT can handle empty parameter type/name field pairs at any position.

6.5 Parameter Name Fields

To each parameter type field belongs a parameter name field. Please do not enter neither a qualifier or a specifier nor a type into a parameter name field. The former are to difficult to handle for the current version of "Function Test". The latter goes into the according parameter type field.

You cannot enter a name without entering a type into the according field and expect a successful build.

If it makes sense to you, you can use or not use any of the parameter type/name field pairs. The UPT can handle empty parameter type/name field pairs at any position.

6.6 Test Environment

After entering your argument(s) - if any - "OK" triggers the function call.

Checking "Honor argument change" makes any new or changed character trigger the function call, too.

Checking "Honor ENTER key" makes hitting the ENTER key trigger the function call, too.

The argument labels are normally green. If the parameter is a reference and if the argument is changed during the function call, the according argument label turns red.

7. Epilogue

Please tell me via the forum, not only if you find bugs, but also whether you find these UPTs useful or not.

The restriction to use only 6 and 4 parameters, respectively, is not "fundamental". The reason for these limits is the current layout of the "Create new package" window. If somebody really uses these UPTs and if there is a true need for more parameters, I'll ask Mirek to change the layout. (The labels and the input fields might be replaced by a scrollable 2-column ArrayControl with the 1st column being read-only. Furthermore, supplementary to "select" there might be a UPT feature which doesn't return the index value (as "select" does") but the indexed value itself ("index"?).)