Index

The following Help Topics are available:

For Help on Help, Press F1

Overview

Toolbox is a programers toolbox for working with Borland's C++ Object Windows Library (OWL) product to create windows applications. Toolbox is a File Manager add-in. When you install it, a new menu item (Toolbox) appears on the File Manager menu. The Toolbox provides four major functions:

1. You can associate a default editor to be executed. Instead of having to associated every file type that you might want to edit (i.e. .c, .h, .dlg, .rc, etc.) with an editor, you can simply highlight the file you want to edit and select the Run Editor option from the Toolbox menu. This will invoke the editor you set and pass the highlighted file name to it. See Run Editor for more details.

2. You can perform text searches on selected file types. For instance, you can search for all .cpp and .h files that contain "Listbox". You can choose to see the line in the file that contains the word or a list of files that contain the word. You can then select a file from the search result list and have it opened up in your editor. See Text Search for more details. See Text Search for more details.

3. You can sort thee types of files. The Borland project file (.prj), a string table resource file (.str), or a resource id file (rcinc.h). See ToolBox File Naming Standards for details on file convention naming. See Sorting Files for more details.

4. You can generate template C++ OWL code from your resource dialog and menu files. You can choose to have the main window generated as a regular application window or as a MDI window. See Generating C++ OWL Code for more details.

Registering your shareware

Shareware allows you to try a program and see if it meets your needs before you pay for it. If you don't find it useful and you don't use it, you don't pay for it. However, if you continue to use it after your evaluation period (30 days for ToolBox), you are requested to become a registered user. Registered users receive the most recent version of ToolBox and free support for one year. They also have a voice in future enhancements to the product.

The registered version of ToolBox doesn't keep putting a reminder to register your shareware each time you use it!

To register as a user, send $23 ($20 for the software and $3 for shipping) to:

DISC

P.O. Box 677805

Orlando, FL 32867

(407) 366-DISC (3472)

Print the Order Form and return it with your check or money order.

DISC - Company Information

Dynamic Information Systems Corporation (DISC) was formed in 1991 as a software development/systems integration company. We specialize in Microsoft Windows application development and client/server technologies. If you would like further information or you have some ideas for shareware products you would like to see, contact us at:

DISC

P.O. Box 677805

Orlando, FL 32867

(407) 366-DISC (3472)

or on-line via America Online at DISC1@aol.com.

Possible Future Enhancements

There are a some ideas we have about enhancing ToolBox.

o Generate your own class names. If you have created special classes (other than the OWL ones) that you want generated, it would generate these instead of the OWL ones. For example, if your Listbox class was TMyListBox, whenever we saw a LISTBOX statement in a dialog resource we would generate TMyListBox instead of TListBox.

If you have any other ideas on how to enhance ToolBox, or there is a shareware product you would like developed but haven't seen, contact DISC and we'll see what we can do.

Sorting Files

Overview

The ToolBox contains an option for sorting Borland Project Files, RC Include Headers, and String Table resources. Open File Manager and select a .PRJ file, .H file, or .STR file. Select the Generate (or sort) option from the ToolBox menu. You can also select more than one file at the same time.

Sorting Borland Projects

As your Borland project grows, it becomes harder to find the file you are looking for in the IDE project window because the window is not sorted in any order. Files are added at the cursor location. ToolBox sorts the module section of the Project file, retaining all information associated with the original. This provides a nice orderly listing of the files included in your project. The original project file is backed up to an extension of .~PR. The Modules are sorted in the following order:

1. File extension.
2. File path.
3. File name.

Please make sure that your project file is not open before sorting.

Sorting RC Include Headers

RCINC.H is a file that contains all the resource ids of the resources used in the project (for more information see ToolBox File Naming Standards). It is important to ensure that resource ids are not duplicated on the same dialogs, but it is sometimes tough to keep track of the resource numbers that you are assigning names to. Failure to maintain unique numbers can cause resource errors during runtime. This is where the Toolbox sorter comes in. It renumbers all the resource ids in the rcinc.h file starting at 101, thus eliminating any possibilities of resource id duplication.

The Sorter will accept any .H file, but will echo any item not is not a #define to the top of the sorted header. It then sorts the names of the #defines and outputs the new numbered sequence. The sorter will make a backup of the header file before sorting, with a .~H extension.

You will not want to include certain items in the rcinc.h file if you are going to sort it. They are:

1. Message definitions, such as "#define WM_BUTTONCLICKED (WM_USER + 1)".
2. BMP Resources IDs, where the ID is dependent upon the video mode.
3. Any other identifier whose value is required to remain unchanged.

These resources should be conatined in an auxilarily header, such as globals.h, which can be included into rcinc.h. The #include line will be echoed to the top of the sorted header, thus insuring the inclusion of the identifiers into rcinc.h while maintaining their assigned values.

Sorting String Table Resources

Another candidate for exclusion from the RCINC.H is String Table Identifiers. The identifiers should be grouped into sixteen (16) string segments that contain items that you would like in memory at the same time to optimize performance. The sorter will sort the String Table Identifiers based on the name of the identifier and not based on any optimization technique. So, if you wish to maintain control over which identifiers go into each segment, you should add them to globals.h to ensure that they retain their assigned values.

However, if optimization is not an issue or if you wish to assign identifiers in such a manner that some optimization may be achieved though contiguous identifiers, you can keep the String Table identifiers in the rcinc.h file and use the project sorter to have the String Table reflect the order of the identifiers.

To sort string tables, the string tables should be included into the resource project as a resource file with a .STR extension. Highlight the .STR file and select the Generate (or sort) option from the ToolBox menu. It will be sorted and a backup will be made of the original file, giving it a .~ST extension.

ToolBox File Naming Standards

In order to get the most out of the C++ OWL code generator, you need to follow certain file naming standards.

Resources:

Each resource should be kept in a separate resource script file. We have used the following standard:

.rc	resource file that is nothing more than #includes of actual resources (dialogs, menus, etc.)
.dlg	dialog resources
.mnu	menu resources

The generator only looks at dialog and menu resources, so these are the only ones that are required to be in separate files, but we recommend that all resources should be kept in separate resource script files.

Each resource is given a #define name. These names are used when generating the C++ variable names and class names. Therefore, you can't just use integers in the Resource Workshop, you need to give all your resources a #defined name. Save your resource ids into *rcinc.h*. This header file is included in all generated .cpp files. This file name (*rcinc.h*) is "hard-coded" into the generator. The generator assumes that all the resource ids are in this file. This file can also be sorted and all the resources given unique resource ids. (See Sorting Files for more information.)

Each menu (*filename*.mnu) resource that is generated will create a *filename*.cpp and *filename*.h file. The *filename*.cpp file will contain the C++ OWL code for the main application window. It will be either a regular window or a MDI window depending on your preferences The *filename*.h file will contain the class definitions for the application and main window.

Each dialog (*filename*.dlg) resource that is generated will create a *filename*.cpp and *filename*.h file. The *filename*.cpp file will contain the code for the dialog class and the *filename*.h file will contain the class definition for the dialog class. If there is a *filename*.mnu file with the same filename as a *filename*.dlg file, the dialog will be generated with the assocaited menu. See Generating C++ OWL Code for more details.

A word on directories. We set our projects directories up using the following standard:

project	(placeholder - only contains sub-directories, where project is the project name)
bin	(all .obj and .exe files are placed here)
include	(all .h, .dlg, .rc, .mnu, and other resources placed here)
source	(all .cpp and .prj files are placed here)

If the generator see a directory named "source" as a sibiling the the directory the resource files are in, the generated .cpp files will be placed in it. If there is no "source" directory as a sibling to the current directory, the generated .cpp files will be placed in the same directory as the resource files used in the generation process.

Run Editor

In Windows, you can associate a file extension with a program. For instance, you can associate .doc files with Word for Windows and .txt files with Notepad. Then, when you double-click on a file with a .txt extension, Notepad will start with the selected file. But, if you don't have an association with an extension, you need to start Notepad yourself and open the file. This is where the Run Editor option of the Toolbox comes into play.

Select the file you want from File Manager and select the Run Editor option. This will start the program you identified as your editor in the preference screen, passing the selected file name to it. For example, you could assign Notepad as your default editor and then when you selected a file and the Run Editor option, Notepad would start and display the file. This means you don't have to associate every possible file extenstion that you may want to edit. Your favorite editor is just a menu selection away.

Text Search

The Text Search function allows you to search selected files for a given string. The search starts at the current directory. Select Toolbox|Text Search and the Text Search screen will be displayed. Fill in the string you want to look for in the *Searching for* field. Then, select the options you want:

Ignore case - if checked, will find a match without regard to upper/lower case. If not checked, the string must match exactly.

Search subdirectories - if checked, will search all subdirectories. If not checked, will only search the current directory.

File names only - if checked, will display a list of files that contain the search string. You can then double-click on a file in this list and invoke the editor specified on the preferences screen. If not checked, a report will be produced that lists the file names and the line that contained the string. This report will be displayed in the editor you specified on the preferences screen.

Update defaults - if checked, will save the current selections to de displayed as the defaults the next time the Text Search screen is opened. If not checked, the current selections will not be saved.

Then, select one or more files from the Files list. Only files that match these will be searched. This list is built from the Text Search list on the preference screen.

Preferences

The preferences screen allows you to change several settings that affect how the Toolbox works.

Developer - Enter the name of the person you want to show up in the "Developer" section in the header comments of the generated C++ (.cpp) files.

Editor - Enter the name of the program that you want to use as your default editor. Note: This program must be in your DOS path. For example, enter "notepad" to have the Windows Notepad editor be your default editor. This editor is used by the Run Editor and Text Search options.

Overwrite existing source files - if checked, when you generate a .cpp file, if a file exists by that name it will be overwritten. If not checked and you generate a .cpp file that already exists, the newly generated file will be named .cpo.

Generate MDI as Main Window - if checked, when you generate a menu resource (.mnu file), the main window will be generated as a MDI window. If not checked, the main window will be generated as a regular window.

Text Search list - list of files that will show up in the Files list in the Text Search screen. You can add or delete from this list by using the New and Delete buttons.

Generating C++ OWL Code

Overview

The ToolBox code generator generates template C++ code for a Borland C++ OWL windows application from windows resource files. Open File Manager and highlight a windows dialog or menu resource file (.dlg or .mnu) and select the Generate (or sort) option from the ToolBox menu.

Building Resources

To start with, you need to create the dialog and menu resource files that your application is going to use. Each of these resources should be kept in a different source file, i.e. one resource per source file. Consult the Borland Resource Workshop guide for saving resources to separate files.

Your Resource Workshop project file should contain nothing more than #includes of header files and other resource files. (In the example, this is *owltest.rc*) It includes the windows header file (*windows.h*) and *rcinc.h*, a file that contains all the resource ids of the resources used in the project. It then includes the main menu resource (*main.mnu*), a dialog resource (*sample.dlg*), and the menu that will appear on the sample dialog (*sample.mnu*).

Each resource is given a #define name. These names are used when generating the C++ variable names and class names. Therefore, you can't just use integers in the Resource Workshop, you need to give all your resources a #defined name. Save your resource ids into *rcinc.h*. This header file is included in all generated .cpp files.

After creating and saving your resources, you are ready to generate an application.

Generating Applications

Open up File Manager and select the resource files you want to generate. Select the Generate (or sort) option from the ToolBox menu. When the generator is finished, it will tell you the names of the files it created.

It should create a [filename].cpp file and a [filename].h file (where [filename] is the filename of the resource file). When you generate a menu file, it will create the code for a new application with the resource as the menu on the window. When you generate a dialog, it will create the code for the dialog. If there is a menu file that has the same name as the dialog file, it will also generate a menu for the dialog.

Generating the Sample Application

1. Start File Manager.

2. Select the *main.mnu* file and select ToolBox|Generate (or sort). (This will create a *main.cpp* and *main.h* file.)

3. Select the *sample.dlg* file and select ToolBox|Generate (or sort). (This will create a *sample.cpp* and *sample.h* file.)

4. Create a new project using the Borland IDE. Add the *main.cpp* and *sample.cpp* files to the project. Add the *owltest.rc* file.

5. Copy the **GetApplication()->ExecDialog(new TSampleDlg(this, DLG_Sample));** line from the *sample.cpp* file and paste it into the CMTest1 function in the *main.cpp* file. Add **#include "sample.h"** into *main.cpp* after the **#include "main.h"**.

6. Compile and run the application. You need to make sure your project has Borland's

includes and libraries in your project's search path. Under the Options menu, the Directories option displays a dialog box to set the search path for includes and libraries. Make sure the classlib\include and owl\include are there and the classlib\lib and owl\lib are in the library. See the Borland documenation for help. There are also sample prj files included in an example directory under the owl directory. You might want to copy one of these prj files and rename it to your own.

Choosing Test1 from the main menu should invoke the sample dialog. You can edit the .cpp files to put in the application specific code.

The *owlapp.h* and *rcinc.h* files are #included into all your .cpp files. Remember to save all your resource ids into the *rcinc.h* file.

Regenerating

The generator does not keep track of changes you have made to the cpp file. i.e. if you modify a dialog and add a new control, when you re-generate the file it will not keep the changes you made to the original cpp file. It will create a [filename].cpo and a [filename].ho file. Your original [filename].cpp and [filename].h files remain unchanged. You can cut and paste the changes from the [filename].cpo and [filename].ho file into your original [filename].cpp and [filename].h files.

Sample Files

The following files should be included:

owlapp.h Header file included in generated .cpp files to include Borland OWL classes.

Sample application files:

owltest.rc	Sample application Borland Resource Workshop project file.
main.mnu	Sample application main menu resource.
sample.dlg	Sample application dialog resource.
sample.mnu	Sample application menu resource for sample dialog.
rcinc.h	Sample application #defines for resource ids.

Creating Your Own Applications

After you generate the sample application and run it, examine the code it generated. Notice where you would add your application specific code into the generated code. Refer to your Borland C++ ObjectWindows Users Guide for more information.

Follow these steps when creating your own applications:

Save each resource file you create in Borland's Resource Workshop into its own source file. (We've used the convention of saving dialogs to .dlg files and menus to .mnu files.)

Name each resource and each control you create in Borland's Resource Workshop with a#define and save the #define to the *rcinc.h* file.

Generate the main menu resource and each dialog resource.

Cut and paste the generated ExecDialog statements from each generated dialog .cpp file into the appropriate function in its parent .cpp file. Add a #include for each header generated into your main .cpp file for each dialog your main menu calls.

Create a project in Borland's IDE and add the .cpp and .rc files to it.

Order Form

Toolbox 1.00 Registration Form/Invoice. Please remit to:

DISC

P.O. Box 677805

Orlando, FL 32867

Ordering by check: You can order by sending a check and this order form to the

address above.

Please check one: 5.25" Disk ____ 3.5" Disk ____

Toolbox: quantity ___ @ $ 20.00 ea. = _____

Shipping & handling $ 3.00   + _____

Florida residents add 7% sales tax   + _____

Total payment   _____

Payments must be in US dollars drawn on a US bank. Prices guaranteed through 1993.

Name: _____Date:_____

Company: _____

Address: _____

City, State, Zip: _____Country: _____

Day Phone: _____ Eve: _____

Electronic Mail address: _____

Where did you get your copy of Toolbox?_____

Comments: