WxSmith tutorial: Adding support for events in custom items

From Code::Blocks

Contents

Introduction

In previous tutorials we managed to add external wxChart item into wxSmith. This class is nice solution for presenting basic charts but has one disadvantage - it does not send any back events which may help to catch some user's action. Since in this tutorial we will talk about events, we will have to choose other item. The easiest solution for me was wxCustomButton class wihch is part of wxThings project (http://wxcode.sourceforge.net/showcomp.php?name=wxThings).

This class offers two events which may be processed by user:

- EVT_BUTTON - event generated similarry to EVT_BUTTON in standard wxButton class
- EVT_TOGGLEBUTTON - event generated when button is in toggle mode (similarry to wxToggleButton)

We will add support for both events (even though using given settings only one may be generated).

I won't show the full implementation of the class (it may be easily done by following previous tutorials) so we will focus on events only.

Event block

By default all items have block in cpp file which looks like this:

```
WXS_EV_BEGIN(<Events array name>)
        ...
WXS_EV_END()
```

Those macros automaticaly build array which contains descriptions of events. Such array may be also generated automatically but this won't be covered in this tutorial.

Between WXS_EV_BEGIN and WXS_EV_END macros are declarations of events supported by this class and there's also possibility to ad category for events (categories are not used now in wxSmith, but it will be added in future). To add event you can use one of those macros:

- WXS_EV(Entry,Type,ArgumentType,FunctionBase) - this macro can be used to add event which does not use id of item which generated event (like EVT_LEFT_DOWN when only function is specified in event table)
- WXS_EVI(Entry,Type,ArgumentType,FunctionBase) - this macro can be used to add event which does use single id with id of item that generated event (like EVT_BUTTON where you must specify function name and id in event table)
- WXS_EV2I(Entry,Type,ArgumentType,FunctionBase) - this macro can be used to add events which require range of ids (like EVT_COMMAND_RANGE), this mode is currently not used since one item may have only one id

Arguments for these macros are as follows:

- Entry - name of entry in event table for this event (this name will be visible in property browser)
- Type - type of event (usually it's Entry with extra "wx" prefix)
- ArgumentType - type of argument passed to event processing function (like wxCommandEvent for EVT_BUTTON method)
- FunctionBase - base name which will be used to generate default name of function processing event. It should describe what action generated this event.

To start new category of events you may use WXS_EV_CATEGORY(Name) macro.

There's also one more macro called WXS_EV_DEFAULTS() which adds all standard events processed by controls like paint, mouse or keyboard. This macro should be used carefully since wxWidgets does not allow using all standard events for all items and even if such support is added on one platform, it is not on other.

Events for wxCustomButton

Ok, now let's give some real example. At the beginning we talked about adding events for wxCustomButton class, so here's valid declaration:

```
    WXS_EV_BEGIN(wxsCustomButtonEvents)
        WXS_EVI(EVT_BUTTON,wxEVT_COMMAND_BUTTON_CLICKED,wxCommandEvent,Click)
        WXS_EVI(EVT_TOGGLEBUTTON,wxEVT_COMMAND_TOGGLEBUTTON_CLICKED,wxCommandEvent,Toggle)
        WXS_EV_DEFAULTS()
    WXS_EV_END()
```

Note that no arguments inside this block use quotes.

This few lines of code is almost all needed to make events up and running. There's only one thing left. When connecting events (which is done through Connect function, not event table) compiled will require event type, also when user will add event handler which will hanve non standard argument, this argument will also have to be declared. All this means that there are few extra includes and class declarations needed. Those may be added in OnBuildCreatingCode. So the example list of headers for wxCustomButton will be added this way:

```
void wxsCustomButton::OnBuildCreatingCode()
{
    switch ( GetLanguage() )
    {
        case wxsCPP:
        {
            AddHeader(_T("<wx/things/toggle.h>"),GetInfo().ClassName);
            AddHeader(_T("<wx/tglbtn.h>"),_T(""),hfLocal);
            ...
```

Here extra <wx/tglbtn.h> header is required because it declares wxEVT_COMMAND_TOGGLEBUTTON_CLICKED define required while connecting events. Note that I didn't provided name of define it declares (by using empty string). That's because all local headers are always included (never replaced with forward declarations) so the define name would never be used.

Up and running

Now we have wxCustomButton class in wxSmith which also provide some events. The full implementation can be found in current wxSmithContribItems code (Header (http://svn.berlios.de/wsvn/codeblocks/trunk/src/plugins/contrib/wxSmithContribItems/wxthings/wxscustombutton.h?op=file&rev=0&sc=0) and Source (http://svn.berlios.de/wsvn/codeblocks/trunk/src/plugins/contrib/wxSmithContribItems/wxthings/wxscustombutton.cpp?op=file&rev=0&sc=0))