Subject: Re: Modal vs non-modal window
Posted by mrjt on Fri, 05 Oct 2007 10:09:51 GMT
View Forum Message <> Reply to Message

There are no stupid questions

There are lots of different ways of handling windows in Upp, and choosing which method to use is dependent on the complexity and type of the window and application as well as personal preference.

Generally I would use seperate classes if the window has more complex behaviour, such as values in one field changing allowable values in others, or Option controls that disbale/enable fields etc. You could do all this from the parent window but it would become untidy and complex, especially if you have lots of them.

If you can make your window a modal dialog (one that blocks other windows) it becomes easier to do more complex windows without a seperate class because it can all be handled in one function. A good example of this that I often use as a reference is void Ide::SetupFormat() in the ide package (ide/Setup.cpp) because it handles quite complex data (look at CtrlRetriever in particular, this might be useful to you) and you can easily compare it to the running version in TheIde.

Once you have non-modal windows it is more difficult becuase you have to seperate the OK button handling, but it really depends on the types of windows you are using. If for instance you have many windows that are just data entry you may be able to simplify the process so that seperate classes are unnecessary by creating a single class that inherits from TopWindow and behaves in a general way. For example, this:

```
class DataEntryWindow : public TopWindow
{
public:
    typedef DataEntryWindow CLASSNAME;

    // Sets the table name to update
    DataEntryWindow &SetTableRecord(String table, int recordid);
    // Links a DB field with a Ctrl
    DataEntryWindow &SetDataSource(Ctrl &source, String field, Value intial_value, int datatype);

    OnOK()            { if (Accept()) { Commit(); OnCancelClose();} }
    OnCancelClose()    { delete this; }
private:
    Commit(); // Builds SQL string using datasources and commits it to DB
};
```

could form the basis of a data enrty window that can be launched from a single function and then forgotten about:

```
void NewEmployee()
{
    WithEmployeeLayout<DataEntryWindow> *wnd = new
WithEmployeeLayout<DataEntryWindow>();
```

```
    //Set tablename, datasources etc
    wnd->Open(this);
}
```

Note that I've never used Upp for SQL, so I'm not sure how well this would work, but something similar should be possible. You may also be able to do something with Serialize().

Other than all that (I got a bit carried away I think, I really should be doing some work) you just need to play around, Upp is very versatile once you know what your doing. Reading the source code (both CtrlLib and ide) helps as it was written by very good programmers.

Feel free to ask anything else, answering questions is always good to get the brain working in the morning

James
