Subject: Re: MVC example
Posted by Mindtraveller on Tue, 18 Mar 2008 10:17:42 GMT
View Forum Message <> Reply to Message

In my opinion U++ is standing aside of MVC paradigm.
First of all, MVC assumes a number of child windows inside one parent window (MDI). As far as I know, U++ doesn`t support this paradigm. Yes, you may think it`s that bad. But in my opinion U++ supports much more flexible (and easy to use!) paradigm.

U++ has innovatively attractive support for multi-tab and even multi-window design approach. Of course you have your class which describes one unit of what you edit or view. This can be file/item/etc. U++ approach "everything belongs somewhere" assumes that your class has all requiring GUI objects inside of it. This is exactly what I expect: having GUI controls inside my class as private members nicely fits into OOP paradigm and is what I consider an obligatory part of *good class design*.

These GUI objects are children of Layout class object, which is working not only as simple container, but as layout for them too. Of course, this layout is edited visually in theIDE at design time.

So the general idea is: create layout file with theIDE designer. Include corresponding layout you made as class to your class and have fun. Each time you create your class object, GUI is automatically created and attached where you need it (new tab or new window, or even new docking control).

Let`s take a real example of multi-tabbing interface. Look at simplified version of class COManager. It represents some multi-threading server, working in background. Data it processes is drawn inside it`s tab layout. All we have to do is add parentCtrl member into my class and have it initialized in the constructor.

```
// .h
#define LAYOUTFILE "COManager.lay" //contains InputLayout
#include <CtrlCore/lay.h>

class COManager
{       typedef COManager CLASSNAME;

public:
 COManager(TabCtrl &tabs, int comN);
 ~COManager();
private:
 WithInputLayout<ParentCtrl> parentCtrl; //generated by theIDE in .lay file
};

// .cpp
COManager::COManager(TabCtrl &tabs, int _comN)
{
 CtrlLayout(parentCtrl); //init layout
```

```
 parentCtrl.SizePos(); //make it sizeable
 tabItem = & tabs.Add("...").Slave(&parentCtrl); //add new tab to tab control
}
```

That`s it! It is that simple.
If after a month of usage I decide to have multiple windows insted of multiple tabs inside one window (which is mostly bad decision) all I change is "tabs.Add" to a single line creating new window with my layout. I think that it is exactly what is needed for well structured and rapid GUI development.
