Subject: Re: MVC example
Posted by Mindtraveller on Tue, 18 Mar 2008 13:05:53 GMT
View Forum Message <> Reply to Message

More of that.
I think that MVC paradigm is some kind of glue attaching bad program design to OOP. According to OOP paradigm, each class must have everything possible within it. It must be it`s own model, view and controller - in one place. Or you will have too strong dependencies between different parts of program. This makes it hard to debug and a complete nightmare to upgrade.

My experience acknowledges this approach - less others know about your class internal structure - better your life is. In Borland C++ Builder I had a nightmare of sending UI events between classes hierarchy levels. This all was because of the fact that GUI objects could belong only to main window class. But this is wrong approach: most of time this window has nothing in common with those controls which serviced some internal objects. And I had very long series of useless member function calls - everything in attempt to comply to OOP paradigm. It was horrible but a bit better than classic MVC idea:

```
//MyMainWindow.cpp
void MyMainWindow::OnThisButtonClick (TObject *Sender)
{
   workManager::ActivateSomething();
}

//WorkManager.cpp
void WorkManager::ActivateSomething()
{
   if (currentTab < 0)
      return;
   workersArray.GetItem(currentTab)->DoSomething();
}

//Worker.cpp
void Worker::DoSomething()
{
   ParallelWorker::StartTask();
}

//ParallelWorker.cpp
void ParallelWorker::StartTask()
{
   //DOING ACTUAL WORK
}
```

And all this code is because of unnatural default position of controls inside window/form class. Dynamic creating/deleting of controls was i nightmare too because it is hard to synchronize creating of Workers with it`s controls.

The situation described is far *better* than classic MVC approach, where all the data is public.

So U++ approach in my opinion meets global OOP requirement having everything about this class inside this class. Now I simply write:

```
void ParallelWorker::ParallelWorker(SomeKindOfParent &parentContainer)
{
   CtrlLayout(myLayout);
   parentContainer.AddUnit(myLayout);
   myLayout.thisButton <<= THISBACK(StartTask);
}

void ParallelWorker::StartTask()
{
   //DOING ACTUAL WORK
}
```

It is a way shorter, 10 times as more readable, and more important - inside my class. Now I`m free of debug & upgrade nightmare.

---