

Re: Model View Controller basics.

Source: <http://coding.derkeiler.com/Archive/General/comp.object/2004-09/0630.html>

From: Robert C. Martin (unclebob_at_objectmentor.com)

Date: 09/12/04

Date: Sun, 12 Sep 2004 07:53:58 -0500

On Sat, 11 Sep 2004 18:57:53 +0200, "Val"
<valmont_programming@hotmail.com> wrote:

>Goodday everybody.

>

>I'll need to learn to separate my view(s) from everything else by now. I

>don't have the skills to do so. So I've googled around to find the basics

>about the MVC. For Java there is plenty

>but nothing at all for C++. Not for hobby-coders like me -with mediocre C++

>skills that is. Note that the GoF site doesn't demonstrate MVC (that's not

>covered in their book). But they provide a link to a non-software example.

>That is no good to me. No class diagram is shown, nor a sequence diagram.

>I'd like someone to help me set up a

>"do-nothing-console-toy-skeletonprogram" so I can finally move on.

>I know that is a lot to ask. If you don't think that is the way to do it

>then please would you like to guide me if I present a little case so I can

>learn?

>I utterly hate bothering people (even if they like to help), but I remember

>being taught **extremely** well here when I needed tutoring on the Factory

>Method, Template Method and Strategy Design. I silently am hoping for the

>same quality. In return I promise I'll work hard on this subject as usual.

I can give you a simple exercise that will help you get there.

First, write your phonebook application without any GUI. No GUI code whatever. Instead give it a console interface. However, the console interface pretends that it is a GUI. It knows about buttons, and scrolling boxes, and it displays them in a very crude way on the console. e.g. Heres a textBox, Button, and ScrollBox.

[name: Martin] (lookup) |Bob Martin 800-338-6796|
|Frank Martin 800-555-7723|

You get the picture.

Before you write this interface, write a simple lookup engine (a model).

comp.object: Re: Model View Controller basics.

```
class PhoneBook {  
    public:  
        virtual void AddEntry(Entry* e) = 0;  
        virtual bool DeleteEntry(Entry* e) = 0;  
        virtual vector<Entry> LookUp(string pattern) = 0;  
};
```

Then write a completely separate module (a controller) that **knows** that there is a textbox, button, and scrolling list, but does not know about the gui. This module knows when the button should be grey or black. It knows the contents of the scrolling list. It knows what to do when the lookup button is pressed. Get this working **without** the console code working. Use unit tests to make sure it works.

Finally, write the console UI code, and only allow it to talk to the controller.

This is **NOT** model view controller, but it is the first step towards getting there, and is a good exercise in partitioning GUI from Control from Model.

Once you've got that working, then read the writeups on MVC again. You'll see that the partitioning you have created in the exercise will need to be rewired a bit, but isn't too far from what you need.

Robert C. Martin (Uncle Bob) | email: unclebob@objectmentor.com
Object Mentor Inc. | blog: www.butunclebob.com
The Agile Transition Experts | web: www.objectmentor.com
800-338-6716

"The aim of science is not to open the door to infinite wisdom,
but to set a limit to infinite error."
— Bertolt Brecht, Life of Galileo