

Model, View, Controller

M Jadud

March 4, 2009

About

The Model–View–Controller pattern shows up all over the software world. It is how many graphical applications are structured as well as many web-based applications. The reason for this is it separates the logic of the program (the Model) from the representation of the program’s interface (the View), and explicitly manages the connection between the logic and representation with a third component (the Controller).

It is a challenging pattern. Perhaps you should see it later, after you’ve had much more experience writing object-oriented programs. Or, perhaps I should give it to you as a challenge, to puzzle over and experiment with. I’ve opted for the latter, and we’ll see how this experiment goes.

The Background

You have been given an application that represents a very simple thermometer. It has a Model class that, when instantiated, contains a single variable: the temperature. It also has two methods: `set-temp!` and `get-temp`. These methods do what you might expect; the former changes the value of the temperature variable, and the latter returns it.

Also provided is a very simple View class. When it is told to update, it prints out the current temperature. So that we can see which “view” is updating, I’ve given textual thermometers names. So far, there is one called *Alice and one called Bob*, because I am very creative.

Although it is not a class (in this example), the Controller is represented by a function that instantiates a Model, and instantiates two Views, and ties them together.

The Challenges

I’m going to suggest several challenges.

- ★ Add methods to the model that increment and decrement the temperature by one degree.
- ★★ Create one `frame%` with two buttons, ‘up’ and ‘down’. Those buttons should invoke the increment and decrement functions on a model object. The callback function for each button should do the appropriate (`send ...`) to the model.
- ★★ Create a new view object. It should be a GUI, perhaps with a slider¹. That slider should be updated whenever the model is updated. So, whereas the `text-temperature%` views print the temperature, the `gui-temperature%` view should adjust the slider location.

Note that when you fiddle with the buttons for up and down, you should see both Alice and Bob printing, and your GUI slider moving around. That is, if all goes well. And, assuming you grok MVC along the way.

¹I suggest a slider because it should be a straight-forward GUI widget to work with.