

# Re: MVC/MVP swing GUI

---

*Source:* <http://coding.derkeiler.com/Archive/Java/comp.lang.java.gui/2006-02/msg00018.html>

---

- *From:* Thomas Weidenfeller <nobody@xxxxxxxxxxxxxxxxxxx>
  - *Date:* Thu, 02 Feb 2006 12:37:21 +0100
- 

Tomislav wrote:

Can someone point me to a good, clear example of a GUI (preferably in an open source project so I can look at more than just the GUI) written following the MVP pattern.

I would be surprised if you find a "clean" one (of course depending on your criteria for "clean"), for a couple of reasons (in no particular order, this is an endless subject):

- Swing's own architecture isn't modeled as "pure" MVC or MVP, but a variant. Applications following the Swing architecture are unlikely to be "clean".
- Everyone interprets MVC or MVP slightly different. Even if we leave MVC aside (the Smalltalk people claim they know and have the only pure, original and real MVC), MVP is also a problem. E.g. some people simply say the Presenter in MVP is the typical main loop which can be found in many GUI toolkits, others disagree and point to some separate event and interaction "managing" component.
- The above comes into play also, because MVC and MVP are idealized patterns. And just like when you build something real from an idealized blueprint (e.g. an engine from the model of an engine), you soon figure out that there are issues not covered by the model, and that you need to do more. And there your cleanness goes out of the window.
- The Model-View separation doesn't scale too well for real world applications. E.g. there once was a C++ GUI toolkit (recently resurrected) called Fresco, where each and every, really each and every small and large GUI component was modeled that way. Even a single color component of a color was a separate object. When you programmed with that toolkit you got mad about the amount of objects which you need to wire together to get things set up. Many people have learned a lot from Fresco and its predecessor InterViews, and you can still do, but it drove me mad.
- Scaling/scope, more particular, matching the real world to the pattern

## Re: MVC/MVP swing GUI

is also an issue with Swing (or applications based on similar toolkits). E.g. take a simple form-type entry dialog with a couple of text fields for data entry. How should the Model(s) look like? What are they? Where are they? Each text field has an own Model, or should the sum of the text entries (some kind of data set) be the Model? But what if that set of data from that particular dialog window is just an excerpt from some larger data set? Wouldn't it be good to treat that large data set as the Model? But if you do that, what do you provide as input/output to the dialog? All this is solvable and is solved in practice, but it dilutes a "clean" application of MVC.

- Expanding further on the above dialog example, when you implement something like this, you will soon figure out that you need some kind of transactions. Not something which is defined in MVC or MVP. E.g. you don't want that each single keystroke changing data in a text field is immediately applied to the Model. You want that all changes at once (and typically either all changes or non, probably with a chance to undo them, too) are applied to the Model. So where do you hold the intermediate changes? From the GUI widget's point of view, there should be some Model holding this data, but that Model is not the "real" Model which is finally changed from an application's point of view. Again, all this can and is solved in practice, but you won't find anything of this in "clean" MVC or MVP.

/Thomas

--

The comp.lang.java.gui FAQ:

<ftp://ftp.cs.uu.nl/pub/NEWS.ANSWERS/computer-lang/java/gui/faq>

<http://www.uni-giessen.de/faq/archiv/computer-lang.java.gui.faq/>

.