How do you make a class listen for PropertyChangeEvents caused by other classes, but not this class?

Asked 15 years, 11 months ago Modified 15 years, 11 months ago

Viewed 473 times

0

As described in another question, I have a set of Model objects and an associated set of Panel objects which allow the user to access the data in the Model objects. The Panels are registered as PropertyChangeListeners for the Models, such that if something else updates a value in a Model, it fires a PropertyChangeEvent, and the Panel receives it and knows to resync its values from the Model. (Currently I'm naively just updating all the values, but this could be made more intelligent to pull only the changed property.)

All of this makes sense when a Model is updated by some arbitrary, unknown source, which does happen in my application. However, most often a Model's properties are set by the Panels themselves. In this case, now that I've hooked the Panels up as PropertyChangeListeners for the Models, my code is doing something that makes no sense: after the Panel updates the Model, the Panel receives a PropertyChangeEvent from the Model and pulls the same value from the Model that it originally sent to the Model in the first place. No update needs to occur, and it makes no design sense for this to happen.

So how do I register something as a PropertyChangeListener but then say "Don't notify me of PropertyChangeEvents when I am the source of them?" (Note that I can't answer this by calling PropertyChangeEvent.getSource(); it'll give me my Model, not the Panel that sent the value in the first place; there's no way to look at this and tell what changed the Property.)

java **model-view-controller**

Share

Improve this question

Follow

edited May 23, 2017 at 12:04

Community Bot

1 ● 1

asked Jan 13, 2009 at 20:30

skiphoppy

102k ● 75 ● 179 ● 220

2 Answers

Sorted by: Highest score (default)

3

In all reality, do you really care if you get that event fired back at you? It allows you to handle any times the Model is changed outside the Panel and there really isn't a lot of overhead involved in checking to see if you actually have to update the value.

The PropertyChangeEvent holds the property being changed as well as the old and new values. You can check each incoming event to see if the value in the Panel is the same as the new value, and if it is then discard that event. The Model should tell everyone that is listening to it for PropertyChangeEvents every time it has changed otherwise it will need to know too much about the Objects that are listening to it.

Whatever you do, **DO NOT** create property change event listener loops. You will end up with a situation that very easily can end in an infinite loop if you are not very careful.

Share Improve this answer Follow

answered Jan 13, 2009 at 20:41

Matthew Brubaker

3,117 ● 1 ● 22 ● 18

1

I don't think that there is any measurable detriment by the unneccessary repaint during runtime. So from that point of view there's no need of changing the code. In that situation you should decide having regard to design/architecture. You say:

> ...and it makes no design sense for this to happen...

, but it does! If you want to avoid the repaint, you have to add the source (the panel) to the parameters of the method(s) that change(s) the model, put it into the change event and consider that parameter when evaluating the event again, which makes your code much more complicated. Think what else you have to do, if there are two instances of the same panel, one that changes the model, one that has to be changed after receiving the event...adding object-ids to method calls...What if later changing the design of model or view the way that editing one field of the panel leads to changes of the model that are shown in another part of the panel - additional changes again...etc.

Moreover, if you would do as described, you'll break with the decoupling of model and view, which is never a good idea.

If you want to keep it simple, leave it as it is. The only reason to write complicated code is bad results during runtime and I don't think you'll get that.

Share Improve this answer Follow

answered Jan 13, 2009 at 21:20

Kai Huppmann
10.8k ● 7 ● 50 ● 80