developer toolbox - Podrška #19840

pivot ui, eventbus - publisher/subscriber model

04.03.2010 17:41 - Ernad Husremović

Status:	Zatvoreno	**Početak:**	04.03.2010
Prioritet:	Normalan	**Završetak:**	
Odgovorna osoba:	Ernad Husremović	**% završeno:**	0%
Kategorija:		**Procjena vremena:**	0.00 sat
Ciljna verzija:			

Opis

Historija

#1 - 04.03.2010 17:41 - Ernad Husremović

- Naslov promijenjeno iz pivot ui, eventbus - pub u pivot ui, eventbus - publisher/subscriber model

#2 - 04.03.2010 17:41 - Ernad Husremović

http://www.eventbus.org/confluence/pages/viewpage.action?pageId=819222#EventBus%26ApachePivotorRefactoringUIstoUsePub-Sub-IntroducingtheEventBus

#3 - 04.03.2010 17:47 - Ernad Husremović

The **EventBus** library is a mature, well-tested and very well-documented pub/sub API with annotations. It can be used in any JavaSE application, not just UIs. The EventBus sticks to a single VM, hence it does not compete with JMS, but is often used by JMS consumers to broadcast events through an application. The EventBus is very lightweight and be used piecemeal alongside an existing codebase. The EventBus works with POJOs, so you can immediately publish any Object that you are working with today. This differentiates the EventBus from Java Observer, Java Events, and every other pub/sub API for any platform. The EventBus does not impose any requirements on what is published. This a key reason why the EventBus requires minimal amounts of code and is extremely non-invasive.

#4 - 04.03.2010 17:50 - Ernad Husremović

http://github.com/michaelbushe/EventBusExtras/tree/914073a8f4187f2796b5f67b7254a924cee2e987/pivotStockTracker/afterEventBus

#5 - 04.03.2010 17:52 - Ernad Husremović

The SymbolPane is the publisher - it creates the event and notifies any other interested parties, which for now, is just the master controller.

#6 - 04.03.2010 18:04 - Ernad Husremović

One of the strongest designs for pub/sub in UIs is decoupling the data model from the UI components that are interested in the data. Without pub/sub, components typically register their interest with a DataManager and ask to be called back. Instead of wiring hard listeners, pub/sub uses the Hollywood Principle - "Don't call us, we'll call you." This avoids a lot of work in managing what is listened to as UI components come and go or as the data they are interested in changes.

#7 - 04.03.2010 18:09 - Ernad Husremović

The EventBus is the common name for this pattern, though in Fowler's EventCollaboration, he calls it a MessageBus (which is more apt, but less in line with convention).

#8 - 04.03.2010 18:10 - Ernad Husremović

Validation Through Veto

The second requirement - displaying information about duplicate symbols - is outside of the realm of the text component, so the EventBus is a good choice.

The EventBus provides **veto subscribers that can stop events from being published**. The VetoSubscriber interface has one method, boolean shouldVeto(event), which returns true to veto the event, false to let it get published. The EventBus checks all VetoSubscribers before any regular subscribers are called. This is a distinct advantage over Swing's PropertyChange Veto mechanism, where a veto can occur after some listeners have already been called, forcing the developer to code impossible rollback mechanisms.

#9 - 04.03.2010 18:14 - Ernad Husremović

We've seen some **WTKX** and some of the Pivot niceties. Let's finishing by publishing an event on the EventBus from WTKX instead of Java. I haven't figured out how to create a class EventSubscriber in Javascript, but **publishing is quite easy**. The addSymbol() method we've been working with is a good example since it's called from both the add button and the enter key. Let's stay DRY and move both calls to WTKX:

#10 - 04.03.2010 18:16 - Ernad Husremović

Hopefully you understand how the EventBus is a great tool for user interface development. It can be used to simplify code, attain true component design, reduce coupling and gain cohesion.

Hopefully you also learned something about the new Pivot RIA and the productive development model it presents.

Programming with pub/sub often takes some getting used to since it's a different mindset from the traditional callstack. It is worth the effort. The StockTracker application is quite small and was improved by using pub/sub. The benefits of pub/sub grow non-linearly with the size and complexity of the application.

#11 - 04.03.2010 18:19 - Ernad Husremović

scala traits

http://blog.objectmentor.com/articles/2008/09/27/traits-vs-aspects-in-scala

Scala traits provide a mixin composition mechanism that has been missing in Java. Roughly speaking, you can think of traits as analogous to Java interfaces, but with implementations.

Aspects, e.g., those written in AspectJ, are another mechanism for mixin composition in Java. How do aspects and traits compare?

Let’s look at an example trait first, then re-implement the same behavior using an AspectJ aspect, and finally compare the two approaches.

#12 - 26.05.2010 15:11 - Ernad Husremović

- Status promijenjeno iz Dodijeljeno u Zatvoreno