Wicket + Javascript

Asked 16 years, 2 months ago Modified 7 years, 4 months ago Viewed 4k times

5

I'm wrapping up a `Javascript` widget in a `Wicket` component. I want to let the JS side talk to the component. What I've got so far:

Component in question goes like

```
talker = new GridAjaxBehavior();
this.add(talker);
```

in constructor

and then, later on, puts something like

```
"var MyGridTalker = new talker(" + this.talker.getCallbackUrl() + ");";
```

into the JS.

where `GridAjaxBehavior` extends `AbstractDefaultAjaxBehavior`. I want GridAjaxBehavior to spit back some XML when the JS calls it.

Am I doing this the right way? What should GridAjaxBehaviour do to spit back the XML?

Thanks

javascript java ajax xml wicket

Share
Improve this question
Follow

edited Aug 14, 2017 at 9:44
Rahul Gupta
10.1k ● 7 ● 64 ● 69

asked Oct 20, 2008 at 11:49
tmtest
1,571 ● 3 ● 15 ● 14

FYI, there's an error in your javascript emit. You're missing the right hand paren and semicolon, but you're also missing quotes around the supplied string. ... new talker(\"" + this.talker.getCallbackURL() + "\");"; or something like that. – davenpcj Oct 20, 2008 at 15:43

PSA: Wicket ~= a Java Framework. wicket.apache.org/introduction.html – annakata Apr 3, 2009 at 11:30

please show a link to GridAjaxBehavior - i know wicket a bit, but GridAjaxBehavior is new to me. is this a custom developed behavior – Andreas Petersson Apr 3, 2009 at 11:37

2 Answers

Sorted by: Highest score (default)

2

Spit back some XML for what? Presumably to update the model or the view, yes?

The strength of Wicket is that you don't have to worry about the rendered HTML. In Model-View-Controller terms, you set up the Controller to correctly modify the Model, and Wicket takes care of the View.

The separation is not *entirely* clear: in fact you can show/hide view components, or change then, and that can be seen as altering the View.

But what you generally don't have to do is directly manage the browser or javascript. Wicket takes care of that, if you take care of making your changes in the Java code.

In Wicket, the Ajax will call a method on your AjaxBehavior with an AjaxRequestTarget target.

In that method (or in methods called from it), you do whatever you need to do, updating models or views, and then you add to the target any view component that that has changed. Wicket takes care of updating the browser.

Here's an example. It's taken from some code I did, but **heavily altered** just to make explication clearer. The idea is simple: "chained" dropdown choices, where the options in the child change when the select option in the parent changes, as in the series of [State] [County] [District].

(In the actual class, the Model change is passed to the child, which decides for itself if it has changed, and adds itself to the target if it has, then passes the target to its child. I've removed most of that to make a clearer example.)

Here's the ctor, which just adds to itself an anonymous subclass of an AjaxBehavior:

```
public AjaxChildNotifyingDropDownChoice(...code elided for clarity...) {
    this.child = child;

    // Ajax won't work without this:
    setOutputMarkupId(true);
    //
    add( new OnChangeAjaxBehavior() {
        @Override
        public void onUpdate(final AjaxRequestTarget target) {

            // tell child to update its list
            // based on newly selected value

            // when the Ajax is called,
            // my owning component's model
            // is already updated

            // note we could just type getModel()
            // I'm making explicit that we're calling it
            // on the enclosing class
            // (which a non-static inner class has a hidden ref to)
            child.setNewModelBasedOnSelectionOf(
               AjaxChildNotifyingDropDownChoice.this.getModel());

            // now add the child to the target
            // Wicket javascript will receive the new
            // options and re-render the child dropdown
            target.add(child);

        }
    });
}
```

We could also have hidden or un-hidden components, or added behaviors like CSS styles, or even swapped one Panel for another. As long as for each changed component we: 1) called setOutputMarkupId(true); so that the javascript can find it, and 2) added it to the AjaxRequestTarget

Note that different types (subclases) of Ajax Behavior have different callback functions, so be sure you're overriding the right one (add an @Override annotation so the compiler can complain if you got the name wrong).

But again, the basic wicket idea is that instead of sending raw data for the client to parse and act on, you update your model and view, and tell Wicket to re-render what you've changed, by adding the chnaged components to the target.

The only reason I can think of to send straight XML would to be to feed it to non-Wicket javascript. Let me know if that's your aim, and I completely missed the point. ;)

Share

Improve this answer

Follow

edited Apr 4, 2009 at 20:24

answered Apr 4, 2009 at 17:29

tpdi

35.1k ● 11 ● 82 ● 123

0

I don't really know what Wicket is or what it does, but there is a minor bug in your code (as it appears).

This:

`"var MyGridTalker = new talker(" + this.talker.getCallbackUrl();`

You seem to be missing your end parens:

```
"var MyGridTalker = new talker(" + this.talker.getCallbackUrl() + ")";
```

Anyway, not a big deal, but didn't know if it was intentional.

Share Improve this answer Follow

answered Apr 3, 2009 at 11:23

Eric Ryan Harrison

749 ● 2 ● 10 ● 22