Create a non NamingContainer Composite Component

{scrollbar}

In theory, a "non explicit" requeriment for a composite component class is it should implement NamingContainer interface. Note I said "non explicit", because in practice there is nothing on JSF 2.0 spec that enforces that.

Let's take a look at this simple example:

xml <h:form id="frmTst"> <ui:include src="demoui1.xhtml"/> <ui:include src="demoui1.xhtml"/> <ui:include src="demoui1.xhtml"/> </h:form> demoui1.xhtml <ui:composition> <h:commandButton value="Hello World!"/> </ui:composition>

The previous code works in both MyFaces and Mojarra. This is valid from facelets 1.1.x. The idea is as long as there is no component with "id" set, the previous code will work. Combinations of c:if and ui:include could be possible, but this work well only on MyFaces 2.0.x and upper and facelets 1.1.x. In Mojarra 2.0.x some changes were done that makes such cases not work correctly (but in my opinion c:if is something evil, because it causes a lot of nightmares).

Now let's try to convert the previous case into a composite component, but this time let's create a normal component that does not implement NamingContainer, because after all the previous snippet followed the rule that no components with id set were added:

xml <h:form id="frmTst"> <test:ccc value="Hello World!"></test:ccc> <test:ccc value="Hello World!"></test:ccc> <test:ccc value="Hello World!"></test:ccc> </h:form> ccc.xhtml <cc:interface componentType="test.ComponentCCC"> <cc:attribute name="value" type="java.lang.String" /> </cc:interface> <cc: implementation> <h:commandButton value="#{cc.attrs.value}"/> </cc:implementation> ComponentCCC.java @FacesComponent("test. ComponentCCC") public class ComponentCCC extends UIOutput { public final static String COMPONENT_TYPE = "test.ComponentCCC"; @Override public String getFamily() { return UINamingContainer.COMPONENT_FAMILY; } }

In theory, since all ids are generated, everything should work like in ui:include case. In practice, the previous code works on MyFaces but does not on Mojarra.

Why somebody could want to do this trick? Because NamingContainer has its effects, specially with UIComponent.findComponent. Suppose you want to create a component that works as a panel. You can imagine something like this:

xml <h:form prependId="true"> <y:myPanel> <h:inputText id="field1" ... /> <!-- more components --> </y:myPanel> </h:form> {code:xml} {code:title=y. xhtml} <cc:implementation> <!-- some javascript markup with html and css --> <cc:insertChildren /> <!-- some javascript markup with html and css --> <cc: implementation>

The user wants that the final clientId be "field1", but instead he/she has "generated id:field1", and the worst part is if you need some call to invokeOnComponent or findComponent, you have to set the id of your composite component and take into account the internals of such cc to get the component instance.

There is an easy solution that works with both MyFaces and Mojarra. The idea is just wrap everything inside cc:implementation with f:subview, which creates a NamingContainer component:

xml <cc:interface componentType="test.ComponentCCC"> <cc:attribute name="value" type="java.lang.String" /> </cc:interface> <cc:implementation> <f: subview id="#{cc.id}_p"> <h:commandButton value="#{cc.attrs.value}"/> <!-- more markup here --> </f:subview> </cc:implementation>

Works great, because the internal markup uses the naming container, but the composite component base is not a naming container, so its children or facets will not be affected. Obviously, if you use cc:insertChildren and cc:insertFacet the components will be relocated and affected by the NamingContainer.

Note this hack becomes useful when you have facelets 1.1.x pages and you want to convert them to composite components, so if you are creating new composite components it is better to keep them as NamingContainer instances, unless by some reason (like in the panel problem) you need the root component not to be a NamingContainer.

See [jsr-344-experts] should really a composite component requires NamingContainer interface? for more information.
{scrollbar}