ImmutableWidgetDefinitions

How can we make Cocoon Forms' widget-definitions immutable?

The problem comes from having to build up parent-child relationships:

The parent's child-widget-definition-collection cannot be immutable because we build the parent before we build the children, and then it is too late to add the children to the immutable child-widget-definition-collection property of the parent.

We cannot resolve this by building the children first because then the children's parent-widget-definition property cannot be immutable because by the time the parent is created it is too late the set the child's immutable parent-widget-definition property.

Also, with the introduction of repositories of reusable widget-definitions it is possible for widget-definitions to have multiple parents. A child-widget-definition will have one parent for each place where it is (re)used, so we now need to track a context-sensitive stack of ancestors rather than just storing a reference to a single parent-widget-definition.

Solutions

Walker solution (for multiple-parents problem):

Remove the "parent" property from the child-widget-definition, and instead use a widget-definition-walker to track the stack of ancestors for the current context.

This solution will work as long as we only reference widget-definitions via navigating trees of widget-definitions with widget-definition-walker objects. If we ever needed to externally store a reference directly to a widget definition, then we would have to store a copy of the current context's ancestor list with it, but when would we ever do this? Note that this only concerns widget-definitions, not widget instances (which are still single-parented).

Constructor solution (for circular parent-child references problem):

Immutable member variables can be set during the execution of an object's constructor. This means after the parent's constructor has done most of its setup work it can call the builders for the child-widget-definitions via a widget-definition-builder-assistant, and thus be able to initialize its immutable child-widget-definition-collection. The streaming nature of this build strategy is well adapted to building from SAX streams.

Alternate depth-first solution (for circular parent-child references problem):

Use the walker solution above to break the circular reference between the parent and the children widget-definitions, and then build the widget definitions depth-first, starting with the deepest children and working up through the parents. This moves the call to the child-widget-definition builders out of the parent's constructors, at the expense of higher memory use during widget-definition building, since we need to store the parents' configuration info until we are ready to build each parent. The overhead for this build strategy is minimal *if* we already have external reasons for holding the configuration in memory, such as when building from a DOM.