WoodyEventHandling

Event Handling

Some types of widgets can emit events. For example, the action widget produces ActionEvents and the field widget produces ValueChangedEvents. Next to these events, there are also processing phase events, fired in between the various phases of the processing of a request.

Handling events can be done in two ways:

- by defining event listeners in the form definition (as child of wd:on-action for the action widget, or wd:on-value-changed for the field widget, ...). These event listeners will be triggered when the widget on which they're defined fires an event.
- by registering a `org.apache.cocoon.woody.event.FormHandler` on the Form object. This FormHandler will receive all events from all widgets.

When are events processed? (Request processing phases)

To answer the question "When are events processed?", we have to look a bit deeper into how a form request is handled. This is separated in a couple of phases, more specifically the following ones:

- Any outstanding events are broadcasted to the event listeners.
  The reason this is done is because events might have been collected while the form was loaded with values by the binding framework.
- ProcessingPhaseListeners are informed that the *LOAD_MODEL* phase has ended.
- All widgets in the widget tree read their value from the request. If a widget decides it has to produce an event, it is added to a global (i.e. form-level) list (but not yet executed).
- Once all widgets had the opportunity to read their value from the request, the events are broadcasted to the event listeners. This assures that event listeners have access to the values of all widgets in the tree.
- ProcessingPhaseListeners are informed that the *READ_FROM_REQUEST* phase has ended.
- It is possible that processing ends now. This usually happens when an action widget has caused an event.
- All widgets in the widget tree validate themselves.
- ProcessingPhaseListeners are informed that the *VALIDATE* phase has ended.

Defining event handlers in the form definition

Event handlers can be specified as part of the form definition, as child of the various wd:on-xxx elements, such as wd:on-action for the action widget.

Event handlers can be written in either javascript or java. The form definition syntax is as follows:

```
<wd:on-xxxx>
  <javascript>
    ... some inline javascript code ...
  </javascript>
  <java class="..."/>
</wd:on-xxxx>
```

You can specify as many <javascript> and/or <java> event listeners as you want.

Javascript event listeners

Objects available in the Javascript snippet:

- **event**: a subclass of WidgetEvent. See the WoodyWidgetReference for information on the type of WidgetEvent, and see the Javadocs for what these provide (the samples might also give a good start),
- **viewData**: any "bizdata" supplied to the showForm function from flowscript.
- if the form processing was started from a flowscript, then everything available from the scope of that flowscript, such as global variables, functions and the **cocoon** object (see also Flow Object Model).

Java event listeners

The Java class specified in the class attribute on the java element should implement a certain interface. Which interface depends on the type of widget. These are mentioned in the WoodyWidgetReference.

Note: *In the future, the Java event listener classes will also be treated as Avalon components, so that they get access to e.g. the Logger and the ServiceManager, but this was not yet implemented at the time of this writing (October 31, 2003)*.

Handling events using the FormHandler

To handle events using a FormHandler, write a class implementing the `org.apache.cocoon.woody.event.FormHandler` interface. Alternatively you can extend from the abstract class `org.apache.cocoon.woody.event.AbstractFormHandler`, which will split ActionEvents and ValueChangedEvents to two different methods. See the Javadocs of these interfaces and classes for more details.

Once you created the FormHandler, register it on a form instance by calling the method setFormHandler(FormHandler formHandler) on it.

Note: _If you're doing this from flowscript, you can access the Java Form instance by calling the method getWidget() on the flowscript form object.

eg:

```
        var f1h  = new Packages.org.apache.cocoon.woody.event.Form1Handler();
        form.getWidget().setFormHandler( f1h );
```

_