CocoonFormsJSF

Note: Please post your comments at dev@cocoon.apache.org if you think that information on this page is out of date or wrong (both technologies are in the flux)

A comparison between Cocoon Forms (formerly known as *Woody*) and JavaServer Faces.

Separation of Concerns

- JSF heavily mixes concerns, as all is defined in a single page: page layout, field definition and validation, event handlers, etc. Cocoon Forms, in contrary, cleanly separates them. This can be considered as an additional complexity, but it shows its power when a web designer and a programmer have to work together on a project (this happens quite often!).
  - I also have a prototype 20-lines XSL on my HD that allows to use a "pure", untouched HTML page as a Cocoon Forms template. No mixing at all, and the webdesigner doesn't have to worry about what technology is used to animate the page. -- *Unknown Hero*
  - It's possible to "un-mix" concerns in JSF, by implementing ViewHandler and your own form definition language, and move all component definitions, validations in there.
- The CocoonForms framework is NOT the controller because being the a controller is not the concern of a forms framework (CocoonForms uses Action or Flowscript as controller) - most other forms frameworks (Tapestry, JSF, Struts) are both in one.

Rendering, Styling

- JSF separates widget definition from theirs representation through what is called a "RenderKit". But render kits must be implemented in Java, one class per component, and output final page markup (default implementation: HTML) into subclass of Writer (javax.faces.context.ResponseWriter) using a-bit-SAX-like methods, or simply... write()! Back to the pre-JSP days! And the definition of render kits is about 1/3rd of the JSF specification. Cocoon Forms' rendering is performed by FormsTransformer, which generates markup-independent XML SAX events for consumptions down the pipeline.
  - Thankfully, default JSF RenderKit implementation, BASIC_HTML, does not use write() much, which made it possible to replace JSF's default view layer, JSPs, with Cocoon pipeline generated view (See Faces block in SVN). Faces block does not solve multichanneling problem though - you still have to write XML render kit (and XSLs) to achive this.
- JSF BASIC_HTML render kit output can be styled only to the extent of specifying style and class attributes in the resulting HTML elements. For more extensive changes in the output, render kit have to be re-implemented, or new render kit should be written. Cocoon Forms' styling is performed by few XSLs, and can be heavily modified to your needs without changing any Java code.
- JSF can render into multiple markup languages. To achive this, though, you will have to implement render kits for each target markup language, which amounts to dozens of classes. Plugging a new styling or another target into the Cocoon Forms' (e.g. WAP) is very easy - just write couple of XSLs.

Binding

- JSF only allows to bind the form to a JavaBean, and I'm not sure about how it handles complex bindings such as tables, etc. Cocoon Forms, although the binding is not totally mature, offers a very flexible binding that can map to JavaBeans and XML documents, but isn't limited to this.
- Furthermore some generic "binders", such as the JavaScriptBinding, allow to implement any kind of complex bindings: for example, we use it to bind a repeater to different "parallel" collections in an XML document.

Data types

- Widgets in CocoonForms hold strongly typed data, and that validation happens on this data, not on string values. IIRC JSF only does data conversion as part of the binding.

What we learned from JSF

But JSF also has good things that we have "copied" into Cocoon Forms:

- a server-side form model (no need for a FormBean like in Struts)
- some clearly defined processing phases (binding, reading from request, validation, etc) and
- server-side event handlers (mainly the fact that events have to be buffered until the end of a processing phase to ensure consistency of the form model).

More readings

- Cocoon Forms
- Integrating JSP/JSF and XML/XSLT - The best of both worlds by Eric Bruchez and Omar Tazi
- JSR127 - the JSF Spec
- JSF community page
- More on Tapestry and JSF Interesting reading about Tapestry and JSF, by Howard Lewis Ship (founder of Tapestry)
- Improving JSF by Hans Bergsten (O'Reilly book author: JavaServer Faces)

JSF and CocoonForms - different comments

- JSF impressions by Matthew

Cocoon Mailing lists

- http://marc.theaimsgroup.com/?t=108310430200001&r=1&w=2
- http://marc.theaimsgroup.com/?l=xml-cocoon-dev&m=108366040310715&w=2