Java Web Presentation Technologies and Fluid

Overview

- The special case of DWR
- Overview of Java web presentation approaches
- How do they fit with Fluid's goals?
- Menu of possible "next steps"

DWR

Integrate pure AJAX, pure HTML, and pure Java services

1. Add one JAR to Maven dependencies
2. Minimal configuration to say which Spring beans and methods are exposed to JavaScript
3. Add one JavaScript include line to the HTML file for each bean

Result : Spring bean methods become generic asynchronous JavaScript functions with callbacks

Very satisfying unless you need server-generated HTML.

Why does Java have problems with the web?

As compared to C++, C, Basic, Smalltalk, COBOL, ...?
Beaten by lighter-weight scripting and dynamically-typed languages: Perl, PHP, Ruby-on-Rails, etc.

Reactions

JSP Antica : HTML + Embedded Java + Configuration
"No, Java really *isn't* a dynamic scripting language"

Nouveau JSP : HTML + Embedded non-Java Expression Language + Embedded non-HTML tags + Java + Configuration

HTML-embedded EL + non-HTML tags + Java + Configuration

- JSF
- JSP + various server-side helpers (Spring MVC)
- Facelets + JSF = close enough to HTML to scrape by?
- Tapestry = good HTML previews; idiosyncratic and risky

The revolution: HTML with no embedded EL

- Wicket = HTML + Java
- RSF = HTML + Java + Configuration + Property reference strings

Upsides of revolution

- Cleaner match of human duties to files
- Fewer idiosyncracies to learn, track, and produce support tools for

Downsides of revolution

- Wordier
- Tight conceptual coupling between template and Java, but no automated support
- Hides some important aspects from the template (notably conditionals)

Fluid's split goals

- Guide and support quickly-moving user-centered projects
- Deliver Fluid projects

Fluid requirements for presentation technologies

- User-centered design
    - Request-based work flows (back-button, multiple active contexts)

        - Bookmarkable
        - Able to replicate any HTML + JS

- Faster project delivery
    - Shorten mock-up / delivery turnaround time, preferably with browser-previewable templates
    - Easy integration with Spring

Don't meet requirements

- JSF 1.1
- Early Tapestry
- Early Wicket

Possibly meet requirements

- Tapestry now
- Wicket now
- Facelets + JSF 1.2

The revolutionaries

Wicket and RSF share the most characteristics. Programmers who refuse to use one will probably refuse to use the other. Wicket is a well-run project with a small but fervent community.

RSF is unique among all these frameworks in having the following goals **pre-1.0**:

- Request-based work flows
- Bookmarkable
- Able to replicate any HTML + JS
- Previewable HTML
- Easy integration with Spring
- Working with user interaction / design / accessibility communities

Next steps? Priorities? Personnel assignment?

- RE: Guide and support quickly-moving user-centered projects
    - Put DWR on the approved list?
    - Check for problems with JSP 2, Facelets, Tapestry, Wicket, etc.?
    - ... OR wait for volunteers or requests from experts in those technologies?
    - Spread use of Fluid-developed or approved components in projects using any viable technology?

- RE: Deliver Fluid projects
    - Take advantage of the unique "ground floor" opportunity presented by RSF