JSF and Ajax

An update on the latest strategies for building Ajax applications with JavaServer Faces

Chris Schalk
Co-Author of “JSF: The Complete Reference” / Google Developer Advocate
The Basics - A bit of review for JSF

• JSF’s original goals (from 2001)
  – Provide an easy to use component based Web application development framework
  – Make it easy to integrate with IDEs
  – Encourage OpenSource community to contribute libraries of components
The Basics - A bit of review for JSF

• So are we there yet?
• Yes…
  – Numerous IDEs support visual development of JSF Applications
    • (Studio Creator/Netbeans, Eclipse, JDeveloper, Exadel ..)
  – Many Component libraries now exist..
    • JSFCentral.com has a good list
JSF and Ajax - do they work together?

• So what about Ajax?
  – Is it possible to build Ajax applications with JSF?
  – Yes, but there are different approaches.
JSF and Ajax development development approaches

- How JSF and Ajax can work together

  1. Just use Ajax JavaScript in your page templates
  2. Code your Ajax JavaScript into JSF components
  3. Leverage from a growing collection of Ajax-JSF frameworks
1. Placing JS in your template pages

- The most practical/easy way to inject some Web 2.0 functionality into your JSF UI
  - Can add your own JS, or leverage from existing Ajax JavaScript toolkits/frameworks
    - Dojo
    - Yahoo UI
    - JQuery
    - Prototype, Scriptaculous
    - …
Placing JS in your template pages - Pros/Cons

- **Pros**
  - Technically straightforward
  - Easy for simple/small stuff

- **Cons**
  - Working with more advanced JavaScript can be challenging for Java crowd.
    - i.e. you have to get your hands dirty with client script.
  - Can be hard to debug application
    - Look for errors on both server and client
2. Code your Ajax JavaScript into JSF components

- By using JSF UI component technology it is possible to writing your own components that render JavaScript into the client
  - This is doable because JSF can render any markup on the client

- // Render customized Javascript for slider
- writer.write("<script language="JavaScript" src="" + contextPath + "js/JSSlider.js"></script>");
Doing your own JSF Ajax components - Pros/Cons

• **Pros**
  – Architecturally solid
  – End user/Application developer can easily use - instead of dealing with JS.

• **Cons**
  – Can be challenging to build from a technical standpoint
    • Have to be an Ajax/client expert as well as an serverside/JSF expert
  – Hard to debug when developing components
3. Leverage from existing JSF Ajax Components

- Can now download various JSF component libraries which are largely OS
  - Can plug and go!
  - IDE support is also available for many
    - Exadel, JDeveloper …
  - The most interesting ones are:
    - JMaki
    - RichFaces
    - IceFaces
    - ADF Faces (Trinidad) Rich Client
Leverage existing JSF Ajax Components - Pros/Cons

• Pros
  – Easiest approach
  – Don’t have to deal with JavaScript complexities
  – Can pick from a growing collection of libraries

• Cons
  – If everything works, great!
  – Can still be hard to debug on the client
    • This is because client code is rendered from client - you have little or no control over this
Ajax JSF Components Demos...

Checkout these components...

- JBoss RichFaces (Ajax4JSF)
- Jmaki
- ICEfaces.org
- ADF Faces Rich Client
What about the future of JSF?

• JSF 2.0 in the works
  – Being architected to make Ajax programming easier
    • Easier to bypass full JSF lifecycle
      – For Ajax requests/partial component tree traversal
      – Build tree/render tree
    • Bookmarkable JSF pages
    • Easier to build custom components
    • Easier configuration
    • Standard component set will include rich components (datepicker, tree, tabview etc.)

• Targeted for Java EE 08
More Info...

- JSF The Complete Reference
  - Chris Schalk/Ed Burns

- Covers Ajax JSF component Development

- Blog:
  - http://jroller.com/page/cschalk