AJAX JSF Comparision

AJAX JSF approaches

When writing AJAX JSF components different tag architectures can be chosen.

- Some semi-commercial or ex-commercial frameworks like IceFaces or Trinidad provide a large set of AJAX enabled components each with their own set of component specific behaviour and bugs. Many of these components provide AJAX and non-AJAX functionality that goes far beyond the features o the standard components. These will be hard to maintain as the JSF specification and the standard tag set evolve. We'll call this the multi tag AJAX architecture.
- The single tag AJAX architecture provides an AJAX tag that can be nested inside other JSF tags adding AJAX functionality to the parent tag. Usually certain Javascript functions of the parent component (like onClick or onChange) will trigger an AJAX request and the response will be used to update certain elements in the HTML page. This kind of architecture has a great chance to survive future evolvement of JSF, because enhancements of the standard tags and new fancy components can be AJAX enabled easily. Examples for this architecture are J4Fry and AJAX4JSF (A4J).

For obvious reasons J4Fry has chosen to use the single tag architecture. Using the single tag architecture your application can comply to the JSF standards thus being compliant with the future development of JSF and being easy to maintain for every JSF programmer.

The other important decision concerns the JSF rendering strategy.

- The full render strategy lets JSF run through its lifecycle and render the complete response. In consequence some post-rendering instance needs to restrict the rendered HTML code to the parts needed for the AJAX response. This can be done with a filter on the server side or within Javascript on the client side. A4J combines both strategies reducing the JSF answer through their org.ajax4jsf.Filter and applying it only to the required parts of the page using the "Prototype" Javascript library.
- Using a JSF PhaseListener to stop the JSF lifecycle before the rendering of the JSF page starts allows to render only the components that need to be rerendered by AJAX. We call this the partial render strategy. This strategy reduces the amount of HTML code rendered and the time consumed by the rendering of the components. If you have large components on your page that don't need rerendering the speed of the JSF response will increase significantly.

J4Fry has chosen to use the partial render strategy to reduce the time needed for a complete AJAX roundtrip. We've got a AJAX comparision page in work to demonstrate the difference in performance and the look and feel with some AJAX JSF frameworks. With the strategies chosen J4Fry is the only AJAX framework that needs no manipulation of your application (including the web.xml). You simply add

the jars to your classpath, and the taglib to your JSP and you can start using AJAX right away. J4Fry AJAX JSF is completely non-intrusive to your application.

Before making an AJAX call an AJAX framework needs to collect the values entered on the page. For large pages this is the most time consuming part of the AJAX request. There is a nice comparision of different Javascript AJAX algorithms (without JSF) on the JQuery site. It documents that collecting the submit values and sending them to the server is time critical and error prone. We are working on a AJAX JSF comparision page where performance and robustness of different AJAX JSF libraries are compared.

AJAX JSF frameworks

Apache Trinidad (former Oracle ODF): The Trinidad code has gone through lots of changes as it is derived from the Oracle ADF Framework. Unluckily these changes haven' made the code more mature, but more complicated instead, because the architecture had been adapted to suit modern technologies. On one side there was JSF evolving imposing the need to adapt the JSP/JSF integration. On the other side the ADF AJAX mechanism was initially based on an IFrame solution because the XMLHTTPRequest wasn't supported by the browsers at that time. The component set of Trinidad is impressionable, but lot of work will be necessary to get it stable and to adapt it to new upcoming JSF versions.

IceFaces: The IceFaces code is very much up-to-date, and contains a huge quantity of highly sophisticated AJAX-enabled components. This is the advantage as well as the disadvantage of IceFaces. Having a lot of non-standard components gives you many possibilities, but it requires developers in your project who not only to understand JSF, but are also able to handle IceFaces. Moreover IceFaces will cause then vendor lock-in trap to catch you. To use the more advanced features your complete JSF pages will need to use the entire framework including the ice:view and the ice:form tags. Future development of JSF is kind of guaranteed as SUN is pushing forward the sepicification cycles. IceFaces is not based on specification but rather on well-chosen component features and may or may not fit the future development of JSF.

AJAX4JSF: The AJAX4JSF (A4J) approach is the one which is closest to the J4Fry approach. Still the underlaying technologies differ significantly. AJAX4JSF is a clever combination of different libraries that runs through the full JSF lifecycle rendering the complete HTML page.

- The Prototye library is used to collect the form values and submit the through XMLHttpRequest. As you can see in the comparision this choice is not so good with respect to the quality and speed of the submission.
- On the Server side A4J runs through the complete JSF lifecycle and uses a filter to reduce the amount of HTML redered to the client. The filter needs to be configured inside the applications web.xml.
- On the Javascript side A4J uses Prototype to find the differences between the HTML code returned through AJAX and the page that is currently displayed in the browser. Then Prototype is used to replace parts of the page.
- To make this Prototype comparision possible the pages rendered with A4J need to run through HTML Tidy. Using HTML Tidy may be a good idea with

new projects, because it disables the use of wrong HTML that tolerant browsers may still display. On the other hand this restriction may break the function of an existing application and your projects will have one more foreign dependency.

J4Fry: J4Fry's JSF AJAX uses fast object oriented Javascript code based on the highly mature JSON library. J4Fry is completely non-intrusive: there is no external configuration, neither is the web.xml, nor in the faces-config. J4Fry recommends development based on the standard tags and to add fancyness (like scrollbars) through CSS. We are working on a nice examples page to demonstrate the power of standard JSF plus style sheets. Once you have your application running, you can enrichen it through the use of the J4Fry JSF AJAX tag, that is simply nested inside any editable component, reacts on the components Javascript events and triggers a JSF action.