Tapestry41AjaxSupport

NOTE: This is outdated information that applies only to Tapestry 4. For the current status of AJAX support in Tapestry, see

http://tapestry.apache.org/ajax-javascript.html

There have been a lot of perfectly valid concerns/questions/etc regarding where/how tapestry plans to provide ajax integration. Just yesterday I answered a private email to someone else outlining in very very slim terms what I hoped to do, but since we're all doing it now I suppose I can't procrastinate with a wiki page promise any longer..

Firstly I would like to say that it is my very sincere hope that nothing going on here would be a suggestion that tapestry should be embracing any one component library over the other. Tacos has a great number of faults, as do other libraries I've seen. Not all are completely perfect, but a healthy community of varying implementations is what keeps us all honest 🙂

All of that being said, I hope everyone knows that what I want to do in tapestry is much more ambitious than what is seen in the current implementation of tacos today 🙂 My only hope is to expose/fix/adjust whatever portions of the framework made doing things in tacos hard, so that doing all of these things is a lot easier. I'd ultimately like to make some additional component contributions/adjustments , but am not overly concerned with getting them into the core as it is just as easy to do it in projects like tacos 😉

There are also some semi political sounding sort of processes I'd like to startup in the hopes that the community can become a little bit more involved and self managing. Like a component promotion/incubation program. If tapestry goes to a top level project then it would be very easy to create a sort of component inspection committee where people like Robert/Ron/Leonardo/etc. could hold votes and make decisions about whether they think a particular component should be promoted/etc.. Don't know yet what the process would be to hand it off to someone in the higher level framework area but it sounds good in theory..

I will start with the core changes and work my out to more specific things like components.

-) Rendering Contributions - Though it is not documented anywhere, and as far as I can tell has never been used by anyone tacos did(does) start the notion of varying ajax client/server implementations, which is one of the reasons why I was able to quickly switch gears and abandon prototype as the tacos core client library and switch to dojo.

This service is more or less an extension plugin-in sort of hivemind contribution point. Involved in the mechanics are the core AjaxDirectService, RenderDelegateFactory and the actual ResponseBuilder implementations. It is the job of the RenderDelegate to look at the incoming http headers and whatever else it can, while working with the response builder contributions to determine what kind of request has been receieved, as well as who would like to handle it. This currently covers prototype and dojo. It used to also cover rico, but that project is dead and now the core developer is working for yahoo.

What all of these configuration/contribution points provide is the ability to "plug in" to all the hard work that makes some of what tacos does possible. Things like partial rendering of responses, dynamic script contribution manipulations, form component semantics, new cycle definitions for the client side state (ie before/during/after render on client browser) , etc...

Most of the architecture of this system is still much less than ideal. I don't want components having to know that they are involved in anything ajax related, much less actually call any specific apis to do anything about it..

I have absolutely no idea what (if any) crossover there would be for this same concept in tapestry. It is my hope that with enough gentle refactoring, well defined extension points, there won'tbe as much of a need to do these taco -ish sort of things anymore.

-) Component Ids / Form Component Ids - Howard seems to be addressing these already with some new ideas. It's an incredibly hard problem to solve and I don't envy him for it, but it's also one huge stumbling block for anything client js related. I could go on and on in this section, but I think everyone knows what we are talking about.

-) Rewind cycle adjustments - I think howard is also working on this, but the basic gist of what I'd like to do would be to have the ability to "partially" validate /parse form submissions so that a finer grained interaction with the client page is also possible.

-) New state management concepts - Though there certainly is a lot of work to do on the server end of the spectrum in regard to state, there are also a number evolving design patterns related to all of these dynamic client interactions. Some of them would be better supported by a clear model for knowing what the client-side state is on the server and vice versa. (Ie widgets, visibility of, etc..) See http://codinginparadise.org/weblog/2006/01/ajaxdhtml-tutorial-should-i-render-my.html .
-) New script contribution nodes - The current @Script contributions are extremely powerful and flexible. I would like to add a couple additional nodes to the existing body/unique/initialization steps. These would mostly serve to make the pre/during/post "client render cycle" a lot easier to handle than defining obscure custom protocols within each library.

-) Hivemind Service COMM Api - Though it is very nice and easy to interact with tapestry to do partial page rendering, it really doesn't always make sense depending on what you are doing on the client. Sometimes you really just need to hit a method on a service and return some data, nothing specific to an actual html block.

I have a number of libraries I've not shared with tacos or anyone else that have started this design interaction. Basically, it's sort of equivalent to things like DWR, only a little more in line with simpler design and of course my undying love of the dojo library 😉

There are also some other more interesting debates/research efforts underway to define more clearly what exactly the best interactions are depending on the context in which you need to communicate. (ie, http://blogs.ebusiness-apps.com/dave/?p=68)

This area should provide a much richer support layer for people that prefer thick client -> thin server interactions.

-) New components ! - Yes of course I want to either enhance or provide new components. This is sort of the very last step though. I think 90% of the work will go into providing the plumbing to make a $_{\text{lot}}$ of new technologies easier to use with tapestry, but not force any one particular implementation on people.

-) Javascript core interactions - While I haven't really worked this part out yet to be honest, I would like to use dojo to do some of the basic tapestry javascript interactions. Originally I had thought this would apply only to very specific "ajax" component implementations, but the more code I saw /user issues I read about the more and more I started thinking that tapestrys javascript portions needed some refactoring as well.

Things like event handling, client side validation, window load / unload semantics (ok, so that's event handling again 😉) are all things that I think tapestry would benefit from greatly if using a more robust client-side library than our own custom implementations.

The other important thing to think about here is that things really have/are changing in our deployment client of choice, the web browser. While we were all happily coding away at this and that in the java world (not all of us, howard has always provided direct Scripting support from what I can tell, don't know how he manages to keep doing things right all the time, we'll catch him slipping at some point 😉) the browser folks quietly sort of fixed a lot of the major problems encountered in the browser.

Javascript is a real language. It isn't java, but it's not intended to be. It's our client technologies most powerful form of interaction and shouldn't be taken lightly. While we can continue to roll our own solutions in regard to this, I've found it incredibly cumbersome and time consuming to do so. I think using a library of some sort makes a lot of sense. It hides all of the cross browser scripting issues we run into, provides unified ways of interacting with the DOM /events/etc.. You could just as easily say that java doesn't make sense as a GUI platform as well.

I'm all for providing as much flexibility and openness as possible, just tell me how to do it. Unless someone is able to provide a layer of abstraction on top of the client library of choice (ie commons-logging, swing, awt, etc..) then I don't know how we're going to do it.

None of this means that dojo needs to necessarily be forced on anyone for their uses of ajax-specific interactions, but at the very least I'd like to use it for basic plumbing jobs in tapestry. Like event handling and form validation.

Tapestry folks should also be aware of the amount of upcoming support being provided in this area as well. Like eclipse's upcoming support via IBM (http://news.zdnet.com/2100-9593_22-6033544.html) , as well as being the growing foundation for a number of other well known entities:

- Yahoo (now moving to the core, not just flickr)
- AOL (ie http://blog.dojotoolkit.org/2006/01/30/dojo-iamalpha-and-cdns)
- Webwork
- Django
- RoR (i know some dev's doing it, don't know if it's openly talked about there or not)
- IBM
- some .Net libraries, don't remember who/ where
- Eclipse
- etc ettc...

There are a lot of others but I can't keep track of them all. It seems like every other day that we here new announcements on the dev list of this or that framework providing direct integrated support for it.