MovingToTapestry

NOTE: This is outdated information that applies only to Tapestry 4.

Decision to Migrate to Tapestry

This is from an email I sent today (2004/10/18), someone asked me to post this to this Wiki... so here it is... I may elaborate and add some more structure to this page at a later date, time permitting...

We are about to release a major rewrite of a legacy app and chose Tapestry as the framework over Struts. This was a big move because we have many Struts developers. It seems that developers who like to learn new things, will welcome using Tapestry. It is true that the spin-up time is hard for someone that is so used to doing something a certain way. In this sense, it might be hard to ever move away from something if you don't try.

As far as functionality, I don't think there's anything you can't do in Tapestry that you can do in Struts, it just comes down to the simplicity of the design behind your application.

I decided to use Tapestry instead of Struts after a very long design phase where I was trying to look at things we can do to really gain development efficiencies. We have an application to rewrite that has many pages (50+) with a lot of functionality to implement. Frankly I didn't look at too many frameworks other than Struts and leveraging the use of XML/XSL heavily on top of Struts. .NET was out of the question. I didn't want to select something that wasn't tried and true. Some friends who I trust pointed me at Tapestry and it didn't take me long to realize the gains. I also considered Echo and Wicket but punted due to maturity. The fact that Tapestry is a Jakarta project doesn't hurt getting buyoff either.

I hope this doesn't sound like I've really drunk the cool-aid, but...

The productivity gains and benefits of NOT using JSP seems considerable to me since your designer/CSS person (if you choose to break that resource out) can make the mockups and these become the final templates that are part of the code base. Last minute design changes, which always come up, are a non-issue. In JSP, these would have to be re-factored. It seems to me that there is more risk in having last minute design change in a JSP application. This alone would have extended our timeline on this project.

As far as pure code scaling...in terms of an application that may have hundreds of pages and classes, and having to maintain that code base for a long period of time by many developers; I believe that a large web site code base in Tapestry will scale better (in terms of sheer number of classes and pages) than a JSP Struts application, but only IF your JSP/Struts discipline is not the greatest. Collectively over time with many developers, this seems difficult if you don't use something to enforce the discipline. From what HLS says in his book, a primary design goal of Tapestry was, make it easier to do the right thing than not. The maintainability aspect is a benefit of achieving this kind design goal in my opinion. Taking JSP out of the equation is a big step toward this end.

Like .NET, I really like the design paradigm of backing a Web page with a Object/Class that extends the framework's base Page class. The Tapestry design does this quite directly. Since I threw .NET into the mix, I do like Tapestry over .NET because of the template language, it enforces better separation in the model and the view code (aspx is analogous jsp). I've managed a large scale .NET application (www.quote.com), so I can speak from experience. To me, the design of using Actions in Struts doesn't seem as intuitive to me as does the notion of a class backing a pure Page. Submits invoke methods on the class, etc...

After doing a somewhat large multi-lingual Tapestry web app, I believe it would have taken about 30-50% more time in JSP/Struts due to the JSP re-factoring of UI requests we would have had to do late in the game. In Tapestry design changes are non-issues since the mockups are the templates. Another reason for taking longer in JSP/Struts would have been the localization features that we leveraged in Tapestry. Page templates themselves can be localized. Our designer resource quickly came up to speed in using property files for strings so all of the UI was done by a design resource, not a traditional web developer. You can also do Localized page flow in Tapestry by just populating the next page as a property. Since the page property files are localized, you can have a different next page property for a different locale.

Jordan Redner
Software Architect
Shopping.com