Tapestry31Status

NOTE: This is outdated information that applies only to Tapestry 3.

April 30 2004

Just finished merging of changes from the hlship-3-1 branch into the trunk. The unit test suite is working and code coverage has only slipped 0.10% from Tapestry 3.0. There's a temporary kludge around OGNL: the old `IResourceResolver` interface extended `ognl.ClassResolver`. The new `org.apache.hivemind.ClassResolver` interface does not. OGNL now uses a static variable to store an instance of `ognl.ClassResolver` and that's a problem if the Tapestry library is on the system classpath and there are multiple WARs deployed ... but it's all temporary until some form of OgnlService is put in place.

Upgrade (3.0 to 3.1) notes:

3.0 name	**3.1 name**
org.apache.tapestry.ApplicationRuntimeException	org.apache.hivemind.ApplicationRuntimeException
org.apache.tapestry.IResourceResolver	org.apache.hivemind.ClassResolver
org.apache.tapestry.ILocation	org.apache.hivemind.Location
org.apache.tapestry.ILocationHolder	org.apache.hivemind.LocationHolder
org.apache.tapestry.IMessages	org.apache.hivemind.Messages
org.apache.tapestry.spec.BaseLocatable	org.apache.hivemind.impl.BaseLocatable
IEngine.getResourceResolver()	getClassResolver()
org.apache.tapestry.engine.IComponentMessagesSource	org.apache.tapestry.services.ComponentMessagesSource
org.apache.tapestry.engine. DefaultComponentMessagesSource	org.apache.tapestry.services.impl. ComponentMessagesSourceImpl
org.apache.tapestry.engine.ITemplateSource	org.apache.tapestry.services.TemplateSource
org.apache.tapestry.engine.DefaultTemplateSource	org.apache.tapestry.services.impl.TemplateSourceImpl
org.apache.tapestry.parse.TemplateParser	TemplateParserImpl, added new interface TemplateParser

Next up:

- Remove some deprecated methods
- Do some more renames to make things sensible

May 9 2004

Been tackling the dependencies; Digester, commons-beanutils and commons-collections are now out. The specification parser has been completely rewritten for efficiency (and to not use Digester). A lot of work, but should parse faster (I didn't check the old speed, but the new code is brutally efficient compared to Digester).

I've also started work on a new, productized test suite that will replace the internal mock unit tests but also be useable for testing end-user's applications. And it uses JakartaHiveMind:SimpleDataLanguage, not XML. It's been slow going since I've been very TestDriven.

Jun 2004

Been trying a different tack; rebuilding Tapestry from the servlet inward around HiveMind. So far, very successful, as more and more monolithic code is moved into small, light services, with plenty of options for extending behavior. So far, application initialization, including locating and loading the application specification, is now HiveMind services.

July 2004

Hard at work on a Tapestry 3.0 + HiveMind application. This has given me more insight into how to best mate the two together; much of that is reflected in recent improvements to HiveMind.

Aug 2004

The ApplicationServlet has been gutted, replaced with a series of services, pipelines and configurations. There will be more natural places to plug in things like Hibernate transaction managers now ... the pipelines are very much like a kind of AspectOrientedProgramming, in that the pipelines provide for all kinds of extensions to the normal control flow.

The test suite has grown to over 516 tests ... in a few cases, existing mock tests (really, integration tests) have been removed and replaced with clearer, more efficient, more maintainable unit tests.

The 3.0 APIs propogated the ApplicationServlet around quite a bit. There's now a service for accessing the ApplicationServlet, and it is represented as HttpServlet, not ApplicationServlet.
`RequestContent.getServlet()` now returns an instance of HttpServlet. I think, in the larger scheme of things, RequestContext is going to go away ... dissolve into a number of related services.

The `hivemodule.xml` was getting very large, so it has been split into a number of sub-modules.

Sep 2004

Added ResetEventCoordinator and ComponentMessagesSource services.

We're up to 519 tests now. More and more testing has switched over to unit testing, using EasyMock objects. However, the integration tests (*mock tests*) are quite a bit slower now; each tests is paying a startup price for using HiveMind. I suspect that a lot of it is all the class creation (for all the HiveMind proxies and interceptors). I had to up the max memory size on my tests to 384M and it may go higher.

Have to keep an eye on performance. It may make sense to convert the most heavily used services to use the primitive service model. On the other hand, I do suspect that this is just increased startup cost, not (visibly) increased runtime cost.

Oct 2004

Worked on SimplifiedSpecificationsProposal. Introduced the Tapestry 3.1 DTD, which is simpler than the 3.0 DTD. <service> is gone; you have to contribute into a HiveMind configuration now.

<static-binding>, <message-binding> and <inherited-binding> have been removed; its now just <binding> and prefixes (as in HTML templates).

A lot of refactoring over the last few weeks to reorganize things into proper HiveMind services.

OGNL is now represented as its own service, hidden behind an ExpressionEvaluator interface. For now, it's still OGNL 2.x, but will be easy to upgrade to 3.x now.

Tapestry engine services are now full HiveMind services contributed into the tapestry.services.FactoryServices and tapestry.services.ApplicationServices configuration points. The <service> element of the application specification is removed in 3.1 and ignored (with a warning) in 3.0. The EngineServiceView interface has been removed; the functionality is now available as additional HiveMind services that can be injected into engine service implementations.

Refactorings are starting around component class enhancement. 3.1 will have some different behavior than 3.0. By the time I'm through, all parameters will be treated as an improved version of Tapestry 3.0's Direction.AUTO (one that properly handles non-required parameters and caches the bound expression's value).

Changes:

- A <property> will always create a property; the checks for an existing non-abstract method have been removed.
- 3.1 will create a transient property if an abstract accessor exists for a property, even if there is no matching <property> (this falls under *dont repeat yourself*).
- 3.1 will allow the type attribute of <property> to be ignored (more *dont repeat yourself*). The property created will match the accessors (if they exist) or will simply be java.lang.Object (if they don't).
- 3.1 will eventually allow different kinds of persistent properties.
- 3.1 will no longer support binding properties; these were properties used to access the underlying bindings for component parameters. In 3.0, if an abstract accessor was available, Tapestry would provide and use the full implementation.
- The direction attribute of <property> will be removed in 3.1 (and ignored inside 3.0's <property-specification>).

The upshot of this is that component properties and parameters will *just work*.

Nov 2004

Added the <inject> element to specifications.

Abstract properties (that is, properties defined with an abstract accessor) are now turned into transient properties as if a <property> element existed for them. You now only use
<property> to make a property persistent, to provide an initial value, or when you don't need it in code.

Dec 2004

Having trouble keeping up with all the changes here (the project documentation is being kept up to date). Lots of new binding reference prefixes ("asset:", "component:", "listener:"). Lots of simplifications of the DTD:

- <set-property> and <set-message-property> (inside <bean>) are now just <set> and use a binding reference
- <asset> has replaced <context-asset>, <private-asset> and <external-asset>, and there's a new prefixing system for assets
- Order of elements is less needlessly rigid in the DTD

Partial support for modular applications, that is spanning directories, is in place. The Shell component is now all but required, since it needs to write a <base> tag into pages.

Removed support for the JSP tag library.

Added a lot of abstractions to support FriendlyURLs. A typical Tapestry URL is now something like `http://.../context/Home.direct?component=border.link`. Requires more mappings in web.xml and some corresponding configuration in hivemodule.xml.

Jan 2005

Started refactoring property persistence, to make it pluggable.

Feb 2005

Haven't been keeping this page up to date. Property persistence is now pluggable, though there's only the "session" persistence strategy at the moment.

The concept of the Visit and the Global object as been generarlized as ApplicationStateObjects and you can have a factory and manager for each one that controls how and where it is persisted.

It's now possible to inject all sorts of stuff into pages and components.

Things are settling down in 3.1, but the big work (improving the components, tests and documentation) is still ahead.