Tapestry41Roadmap

NOTE: This is outdated information that applies only to Tapestry 4.1. For the current Tapestry roadmap see http://tapestry.apache.org/introduction.html#Introduction-Roadmap

Process Changes

Tapestry 4.0 had a very (too) long release cycle. New committers came (and, unfortunately, went) and the frayed edges of the process became manifest towards the end of the cycle.

Release Numbering

Tapestry 4.0's release cycle has clearly shown the limitations of Tapestry's release numbering system. Other projects have benefited from a numbering scheme that doesn't not explicitly identify alpha/beta/rc status, since that status is voted upon after the release has been produced.

To merge into the Apache and Open Source mainstream, Tapestry numbering shall change; the first 4.1 release will be 4.1.1 (not 4.1-alpha-1). Once the code changes are reasonably stable, an existing release will be voted in as the first beta release (it may be, say, 4.1.7). Further beta releases will follow. Eventually a stable beta release can be voted as the final 4.1 release (say, 4.1.12).

Build Environment

Tapestry 4.0's home grown, Ant based build environment has been a growing issue. While lean, and mean, it is too difficult for end users to set up, and typically required the users download the source to HiveMind and Forrest as well. Currently (Jan 2006), HiveMind 1.2 is being converted to build using Maven 2.0. Once that is successful, a conversion of Tapestry to Maven 2.0 will occur as well.

Having an easily built framework will reduce the need for frequent releases, as it will be a simple, standard process for anyone to download latest source from Subversion and build locally.

Functionality Changes

Asset Management

The focus of this milestone will be on refactoring the various Asset services in order to provider richer support for a more natural development style in relation to the Shell and asset inclusion logic.

- Shell: The current Shell component is the main barrier to being able to easily contribute global resources, such as css/javascript inclusions from component libraries. A refactoring of the Shell and some of the asset service notions will enable component libraries to conditionally be able to contribute their assets to the head portion of html pages via the Shell in a much easier manner.
- **Done(01/05/06)** Relative/Unprotected Resources: The current AssetService requires that all resources be tracked and validated against an md5sum digest in order to provide security against malicious users. This scheme works, but also limits the ability for people to do more dynamic things, like optionally include image assets within css spec files, or to do library inclusion requests on resources not previously digested (as with dojo javascript toolkit). Refactorings here should result in a new notion of "unprotected" resources that don't require any special security considerations. These resources and the pattern that describes them will be completely configurable via hivemind configuration points.

- [BuildingDojo] New javascript package library inclusions in tapestry.

Component Identities/Rewind Cycle/URL rewriting

A much more impactual and overarching change to the framework will hopefully result in a unified system for identifying and dealing directly with components, as well as improving overall handling of form input field value parsing. These changes will result in a number of great improvements, the least of which will allow a very dynamic and intuitive request cycle for handling ajax style logic.

Form Support

The current Form and validation framework provides a lot of nice functionality, this milestone will hopefully finish up some of the work started here to provide an even more intuitive means of dealing with forms / form submissions / and validation of input values.

- Javascript: The Tapestry namespace scripts may need some additional refactoring/re-packaging in order to support a cleaner/more dynamic notion of dealing with client side interactions in general. It may even be possible to start introducing outside libraries at this point specifically for things like value type validation and form event interactions.
- Eliminate translator: The translator binding is needed, but it is hoped that the type of values being validated can be inferred automatically from the existing environment, filling the last major chink in the armor of form support that creates a lot of confusion around the validation system.

Component Cleanup

With the other three milestones completed it should finally be possible to start incorporating some of the framework feature enhancements into the existing tapestry components to bring them more up to date with current web standards.

Integration Testing Framework

Tapestry has had an internal integration framework for some time now, but it has never been "productized". A new or revised integration framework is a necessity for Tapestry 4.1.

HowardLewisShip: I've been thinking about using jWebUnit, but replacing the guts somehow to send requests to a simulated servlet container.

Ajax Support

- Tapestry41AjaxSupport : High level overview of design goals for ajax and tapestry 4.1
- 41RenderCycleDesign : High level design overview of current/future rendering cycle plans.