Gotchas

WARNING: This page refers to older versions of Tapestry. For Tapestry 5 see http://tapestry.apache.org/frequently-asked-questions.html

Tapestry Gotchas

initialize()

initialize() is sometimes mistakenly used as the place to initialize properties of a page. Actually, initialize() is used to set properties to the state they should be in when returned to the pool--in other words, when you are done with the page (typically set properties to null). Instead, implement PageRenderListener and then initialize your properties in pageBeginRender().

initial-value attribute of property-specification

even the initial-value is the wrong place to initialize properties of a page with client specific information. The value of initial-value has to be a global, client independent value. This value will be seen by the next user that got this page from the page pool. The initial-value attribute should only be used to set 'constant' values with a scope of all users of the application. As mentioned in the previous gotcha implement PageRenderListener and then initialize your properties in pageBeginRender() to any client-specific value.

Form Parameter Direction

If the "form" direction is specified for a component parameter, that component must implement the IFormComponent interface, directly or indirectly (through inheritance). If there is a component which wraps a form components and is, itself, embedded in a form but does not implement the IFormComponent interface, then the parameter directions for any inherited-bindings should be of direction "in", rather than "form"

Example: a page has a property-specification named "myValue". The page also embeds a component of type "MyComponent" which is not an IFormComponent. "MyComponent" embeds a TextField and has a required parameter "textFieldValue". The page binds myValue to textFieldValue. MyComponent, in turn, uses an inherited-binding to bind textFieldValue to the "value" parameter of the TextField component. Even though the parameter "value" is direction "form", the "textFieldValue" parameter should be direction "in".

Form and Submit Listeners: Order Matters

In 3.0, it matters *where* you place your Submit components in your html template. Why? Because components are processed in order during the rewind. This means that if you have a Submit button placed on a Form before some other component whose value is needed, the field's new value will not be included because the Submit listener is called first. If your listener is on the Submit component, then make sure the component is placed last in the Form. Also, it is important to remember that a listener placed on the Form component is *always* called. This can getcha (get you) when, for example, you have a Cancel button with a listener but put your default Form listener in the Form component. **Both** listeners will be called! Either use listeners on your Submit components only, or use them to set flags that are read by the Form listener.

Tapestry with Oracle OC4J

The XML parser that is bundled with Oracle's OC4J seems to be a bit broken. Specifically, it chokes on the contrib:Table component, giving this error: TablePages.jwc<Line 41, Column 26>: XML-0139: (Error) ID value 'informal' is not unique.

The file is actually perfectly valid XML and has no duplicate id="informal" attributes anywhere in the document (even sub-elements). Oracle does not support the use of 3rd party XML parsers, so in a production environment this is big trouble.

The good news is that Oracle has a new version of their XML parser out that seems to fix the issue (Oracle XDK 10.1.0.2 or greater). The bad news is that the new parser does not seem to come with the current OC4J production release (9.0.4). The new XML parser *does* come bundled with their pre-production OC4J (version 10.0.3) and Tapestry deployments to that OC4J version do work. So, right now you have two options if you must use OC4J:

Option 1 Deploy to OC4J 10.0.3. Note that this is pre-production and is only available standalone. Thus, it does not run under grid control or Enterprise Manager. This may work for some people - in our case it probably will.

Option 2 Modify your OC4J 9.0.4 to use the new XML parsers. I HAVE NOT TRIED THIS, but it seems likely that it may work. See my notes about JDeveloper integration for hints as to how this might be done.

If you are interested in testing your applications in Jdeveloper 10G, you need to get JDev's embedded OC4J instance to use the new XML parser. I had many failed attempts at this. If you curious, here is what didn't work. I tried adding Xerces to the classpath and specifying Xerces as the XML parser at start-up, but this did not work for some reason and Oracle's XML parser was still used. In another attempt, I updated the XML parser libraries that come with JDev to the current XSK version (10.1.0.2 for Windows), however, this did not work because JDeveloper and the embedded OC4J instance share the same XML libraries and JDeveloper seems to have a dependancy on the specific version that ships with it - I couldn't get JDev to start after making this change. A final failed attempt was to install the standalone OC4J 10.0.3 and tell Jdev to use it as its embedded server: this completely failed because the new OC4J version uses a different config file structure so JDev was completely lost trying to read/write configurations.

What did work: I created a separate OC4J standalone installation of the same version that comes with JDev (OC4J 9.0.3). In that separate installation I was able to update the XML parser to the latest version with no problems. Specifically, I updated these files with the files I downloaded from Oracles XDK 10.1.0.2 :
<root of OC4J install>/lib/xmlparserv2.jar
<root of OC4J install>/lib/xsqlserializers.jar

Then I told JDev to use the separate OC4J install as its embedded server by going to Tools | Embedded OC4J | Global | OC4J Installation to Use...

Life was good with JDev after making that change. You may need to tell JDev to refresh the datasources, but there is a button to do this in the Embedded preference panel. Also, I had trouble with the bc4j application that seems to be deployed by default in the standalone install. To fix this, I simply removed the bc4j application from the server.xml file (jdev/systemxxx/oc4j-config/server.xml).

Best of luck to anyone else dealing with this. If anyone finds this information incomplete or incorrect, please post a correction so that others don't have to deal with this.

Ordering of Events

There's no specified order to calling pageBeginRender() in a page and its components. The way around this is to use lazy-loading in component properties where this is an issue.

Validation Gotcha in Tapestry 4

If you have defined validators in your .page descriptor, they will be invoked, and will reject invalid values, even if client validation is not enabled and you are not using the validation information they are providing. This means that if you define a validator and forget about it, you will probably at some point be scratching your head wondering why the values you are inputting are not appearing in your page class. This is, of course, perfectly logical and correct, but hopefully if you are reading this you won't waste as many hours as I did trying to work out what is going on.

Don't count on having parameters in pageBeginRender()

Parameters to components will not necessarily be set when pageBeginRender() is called - in fact, in my experience, they NEVER are. This is probably related to the 'Ordering of Events' gotcha above, except there doesn't seem to be any way around it.