MoreFrequentlyAskedQuestions

WARNING: This is outdated content. For Tapestry 5 see http://tapestry.apache.org/frequently-asked-questions.html

Where is the (some random HTML) component? (e.g., TD)

Any component that does not have a direct Tapestry component can be represented with the Tapestry ANY component. For example, say you want to have Tapestry generate the background color for your table using a TD component you could use <td jwcid="@Any" bgcolor="ognl:backgroundColor">< /td>.

Why isn't Initialize setting my variables when the page is created?

Don't worry; you aren't the first person to make this mistake. Initialize isn't called when a page is created - it is called when a page is returned to the pool to be reused. Initialize is not to be used to setup a page but instead return it to a pristine state so it can be recycled in the pool of available pages. See the pageRenderListener interface if you want to setup state before a page is used.

KentTong: I don't think this is true anymore (not sure if it was true in the past). Initialize is indeed called when a page is created. It is also called when it is returned to the pool. So it can be used to setup a page to some fixed initial state. Of course, you could do that using a <property-specification> too.

Where do I "initialize" values for a page?

You'll probably want to use the pageRenderListener class to perform any work you need when the page is accessed. Alternatively you can do some lazy initialization where objects are created the first time they are used. This isn't as clean as using Tapestry properties but it should work. For example:

AppointmentPage:

```
protected Date date = null;
public Date getEvent() {
  if ( null == date ) {
    date = new Date();
  }
  return date;
}
```

How do I use the pageRenderListener to setup my page before it is used?

Simply implement the PageRenderListener interface and override the pageBeginRender() routine. Often times you can just call initialize() from pageBeginRender() to setup the page - your logic in both routines may be the same.

```
public abstract class AppointmentPage extends BasePage implements PageRenderListener {

private Date date;
// this is called by Tapestry when the page is called up
public void pageBeginRender(PageEvent event) {
        // do all your pre page processing here
        date = new Date();
}

// this is called by Tapestry before it is return to the pool for reuse
public void initialize() {
        date = new Date();
}
```

How do I pass information from page to page?

You can use a Visit object to keep state from page to page but this is probably overkill and cumbersome if you just want to pass some information from one page to the next. Instead you will want to create an instance of the next page, pass the parameter and then call (activate) the next page. Here is some sample code:

Calling Page Java:

```
public void submitAction(IRequestCycle cycle) {
  // The next page we want to go to is the Result page
  AppointmentPage next_page = (AppointmentPage)cycle.getPage("Appointment");
  next_page.setDate(new Date());
  next_page.setEvent("Birthday Party");
  cycle.activate(next_page);
}
```

Appointment Page:

```
<property-specification name="date" type="java.lang.Date" persistent="yes"/>
<property-specification name="event" type="java.lang.String" persistent="yes"/>
```

Appointment Page Java:

```
/*
**  Generated derived class implements these routines and connects them to the properties.
**  You only need to create these abstract routines if you are referencing them within the
**  abstract class.
*/
public abstract void setDate(Date date);
public abstract void setEvent(String event);
public abstract Date getDate();
public abstract String getEvent();
```

HowardLewisShip: I call this the *Tapestry Bucket Brigade*.

I just created a new page and now I get a 'class instantiation problem'. Why can't it instantiate my class?

Most likely you created your class abstract when it didn't need to be. Tapestry doesn't create an enhanced subclass if there are no <property-specification> elements. A concrete class is never created by Tapestry in this case and since abstract classes cannot be instantiated, this exception pops up. You either need to create a <property-specification> element or make your class concrete by removing the abstract identifier.

HowardLewisShip: This is a bug fixed in Tapestry 3.0.1. In 3.0.1, if you class is abstract but there is not need for enhancement, Tapestry will quitely enhance your class anyway, just so it isn't abstract.

How can I share data across two Tapestry applications?

Even if you're running two Tapestry applications within the same Application Server (i.e. Tomcat) you can't share data between them within the Application Server. You can merge the two applications together into one application and then share data through singleton objects within the Application Server. Alternatively the two applications can share data at a lower level common level, i.e. the operating system (message passing) or a database.

HowardLewisShip: It is possible to have two different applications within the same WAR, never mind, EAR, in which case, singletons can be stored in the ServletContext. If you *carefully* check the specification and template file resolution rules (in the Users Guide), you can see how to keep the pieces seperate from each other. However, this is not widely used and there's a possibility it won't make it into Tapestry 3.1.

What is Spindle?

Spindle is a Tapestry 3 plugin for the Eclipse IDE. Eclipse is available at www.eclipse.org and Spindle is available at spindle.sourceforge.net. The Spindle adds intelligence about Tapestry into Eclipse. The result is an IDE that is aware of Tapestry Components and Pages by providing Wizard interfaces and navigation widgets for the creation and viewing of Tapestry Components and Pages. It is well worth a look if you intend on developing with Tapestry. Download Eclipse first and then install Spindle through the auto update feature of Eclipse.

What are the jwcid=$content$ tags that Spindle places into my HTML files?

Tapestry does not process any text outside of a <span jwcid="$content$"></span>. This allows a developer to place documentation, header files or static HTML (that should not be rendered by the Tapestry engine). Spindle places these tags into the HTML out of convenience. The can be deleted without any problems.

When would I use the $remove$ tag?

The $remove$ tag is used as an aid to view static HTML pages. The Tapestry Engine removes the tag and any text insert within when the page is rendered. This is useful when the text would otherwise be generated dynamically and an HTML designer needs to mock up the pages statically. In the following example, three lines would be displayed when the page is statically viewed while the Tapestry Engine would render the one line dynamically when it is iterated through the Foreach component.

```
<table>
<tr jwcid="employeeForeach">
  <td><span jwcid="employeeName">Jane</span></td>
  <span jwcid="$remove$">
  <td>Sally</td>
  <td>Sue</td>
  </span>
</tr>
</table>
```

Why is my Visit object always null or throws NoClassDefFound?

You might want to check that your Visit class is defined in the application specification. If it is not defined you will run into problems when calling page.getVisit(). Make sure you put a fully qualified class name in the property.

MyApp.application:

```
<property name="org.apache.tapestry.visit-class" value="some.example.Visit" />
```

AppointmentPage.java:

```
Public void submit() {
        // getVisit() will need to be casted to the defined Visit class
        some.example.Visit visit = (some.example.Visit)getPage().getVisit();
}
```

How do I make a property in a Component persistent?

Pages support persistent properties so place the persistent property in the encompassing Page and pass it into the Component. The property will remain persistent and be passed into the Component as a parameter. The Component will need not know, or care, that the property is persistent.

HowardLewisShip: The above is technically correct, but <property-specification> inside a component can still use the persist="yes" attribute and be persistent! The mechanism is exactly the same for pages as it is for components, just the HttpSession attribute key is a bit longer (it incorporates the page name, the component id path and the property name).

Where can I find a Tapestry tutorial?

- Kevin Dorf Tapestry 3.0 beta 3 Tutorial
- Simple CRUD Model Tutorial
- Kent Tong's Tutorial
- Cloud Nine Tutorial
- T-Deli components

Where can I host my Tapestry application?

You've got several choices ranging from a hosted third party Java Servlet container to a dedicated (or even virtual) server. Some companies recommend through the Tapestry mailing list include:

- www.kgbinternet.com
- www.kattare.com
- hosting.groovesystems.com
- www.mmaweb.net/index-google.html
- www.eapps.com/ManagedHosting/JBoss.jsp
- oxxus.net
- www.johncompanies.com
- 1and1.com

How can I implement skins in Tapestry?

You'll want to take a look at CSS (Cascading Style Sheets). Tapestry can be programmed to pull in the appropriate CSS as needed. For an example of the capabilities of a CSS please look the CSS Zen Garden.