Tapestry5HowToCreateYourOwnComponents

I wanted to create a tree component, based on http://www.dhtmlgoodies.com/scripts/drag-drop-folder-tree/drag-drop-folder-tree.html, and it is actually really simple (thanks to Francois Armand, who pointed me in the right direction).

1. Create a new package in src/main/java, *org.example.myapp*.**components**. Note that the first part (in italics) should follow the same naming convention as your other packages (e.g. org.example.myapp.pages), but the components part is compulsory.
2. Within the component package, create a new class, e.g. Tree.java
3. Study the diagram on component rendering at http://tapestry.apache.org/component-rendering.html (reproduced at the bottom of this page). Based on this diagram, you can see which states you need to implement as annotations.
4. For example, the Tree.java file below creates an unordered list <ul>, which is reformatted by the CSS and JS to the DHTMLgoodies drag-drop-folder-tree.
5. That's all there is too it: you can now reference your tree component in every HTML page as follows, e.g. you can write in Start.html:

```
<t:Tree source="pos" currentNode="node" parentNode="parent">
   <t:actionlink context="node.getCreatedDT()" title="${node.getDescription()}">${node.getId()}</t:actionlink>
</t:Tree>
```

Note that in Start.java, the source *pos* (an `ArrayList` of object that contain a createdDT time stamp, a description, an id and a depth field), the *node* and *p arent* all need to be present, and require getters and setters. Furthermore, in order to convert the unordered list to the pretty DHTML goodies tree, you also need to include the CSS, JS and image files in this HTML file, as well as the bit of javascript you see at the end of the DHTML goodies example (`treeObj = new JSDragDropTree();` etc. I'm still looking into how to integrate all of these items into the component, but the basis is there.

Tree.java

```
package org.example.myapp.components;

import java.util.ArrayList;
import java.util.Iterator;

import org.apache.tapestry.ComponentResources;
import org.apache.tapestry.MarkupWriter;
import org.apache.tapestry.annotations.AfterRender;
import org.apache.tapestry.annotations.AfterRenderBody;
import org.apache.tapestry.annotations.BeforeRenderBody;
import org.apache.tapestry.annotations.BeginRender;
import org.apache.tapestry.annotations.CleanupRender;
import org.apache.tapestry.annotations.Environmental;
import org.apache.tapestry.annotations.Inject;
import org.apache.tapestry.annotations.Parameter;
import org.apache.tapestry.annotations.SetupRender;
import org.apache.tapestry.annotations.SupportsInformalParameters;
import org.apache.tapestry.services.Heartbeat;

import nl.tno.secureit2.data.PhysicalObject;

/**
 * @author Erik Vullings
 * Implements a www.dhtmlgoodies.com drag-and-drop-folder tree component
 */

@SupportsInformalParameters
public class Tree {
        /*
         * Current depth of the node in the tree
         */
        private int currentDepth;
        /*
         * Iterator to iterate over all tree elements
         */
        private Iterator<PhysicalObject> iterator;

        /**
         * Defines the source Tree to walk over. This is the ''source='' input of your component in the Start.
html file, as well as their getters and setters.
         */
        @Parameter(required = true)
        private ArrayList<PhysicalObject> source;

        /*
```

```
         * Defines the current node of the tree: as the parentNode, these objects need to be defined in the
Start.java file, as well as their getters and setters.
         */
        @Parameter
        private PhysicalObject currentNode;

        /*
         * Defines the parent node of the tree
         */
        @Parameter
        private PhysicalObject parentNode;

        @Environmental
        private Heartbeat heartbeat;

        @Inject
        private ComponentResources resources;

        // The first state to render a component: perform initialization here
        @SetupRender
        boolean setupRender() {
                if (source == null)
                        return false;

                currentDepth = 0;
                this.iterator = source.iterator();
                return (iterator.hasNext());
        }

        /** Begins a new heartbeat: The heartbeats allow you to loop over every item in the tree. */
        @BeginRender
        void begin() {
                parentNode = currentNode;
                currentNode = iterator.next();
                heartbeat.begin();
        }

        // Before the body is being rendered (note that the actual contents of the body are rendered in the
HTML page),
        // I write the <ul><li> based on the current depth. There are several writers, also ones that make
certain that
        // you generate correct XHTML syntax, but I use the raw format since I already had this algorithm.
        @BeforeRenderBody
        void beforeRenderBody(MarkupWriter writer) {
                writer.writeRaw(
                                getIndentation(currentNode.getDepth()) +
                                "<li id='node" + currentNode.getCreatedDT() +"'>");
                resources.renderInformalParameters(writer);
        }

        /** Ends the current heartbeat: if the iterator sees more items to process, return false (== not ready)
and start the next heartbeat */
        @AfterRender
        boolean afterRender() {
                heartbeat.end();
                return (!iterator.hasNext());
        }

        /* Any final cleanup that needs to be performed can be added here.
        @CleanupRender
        void cleanupRender(MarkupWriter writer) {
                writer.writeRaw(getIndentation(-1));
                resources.renderInformalParameters(writer);
        }


        /*
         * Helper function, which returns the <ul><li> etc. based on the currentDepth
         */
        String getIndentation(int depth) {
                String s = "";
```

```
                if (depth == -1) {
                        // Reset function (currentDepth remains the same between calls of
                        // the page)
                        currentDepth = 0;
                        return "</li></ul>";
                }
                if (currentDepth == 0 && depth == 1) {
                        // First time
                        currentDepth = 1;
                        return "<ul id='dhtmlgoodies_tree2' class='dhtmlgoodies_tree'>";
                }
                if (currentDepth > depth) {
                        s = "</li>";
                        while (currentDepth > depth) {
                                currentDepth--;
                                s += "</li></ul>";
                        }
                } else if (currentDepth < depth) {
                        // The difference can never be more than 1 (which would mean
                        // skipping a level)
                        currentDepth++;
                        s = "<ul>";
                } else
                        s = "</li>";
                return s;
        }
}
```

http://tapestry.apache.org/tapestry5/images/component-render-states.png