Tapestry5TreeComponent

Based on the drag'n drop folder tree from DHTMLGoodies.com, I've created a simple T5 tree component. The original uses AJAX, but since T5.0.5 hasn't got it yet, this one doesn't have it either. Still, it looks rather pretty, and it's quite simple to implement.

http://img502.imageshack.us/img502/7000/treehi8.gif

Each list of objects that you wish to display in a tree needs to implement the ITree interface, basically specifying the depth (depth=1 is top level), and a unique identifier):

```
public interface ITree {
        public int getDepth();
        public long getIdentifier();
}
```

Next, create a new page in tapestry, e.g. TreeDemo.html and TreeDemo.java, where the first just loads some images, js and css files (which can be obtained from the original site - images can be easily overruled) and calls the tree component, t:Tree - each node itself is generated using a standard actionlink component! TreeDemo.java just consists off a bunch of getters and setters.

```
<html xmlns:t="http://tapestry.apache.org/schema/tapestry_5_0_0.xsd">
<head>
<title>Tree demo</title>
<script type="text/javascript" src="js/context-menu.js"></script>
<!-- IMPORTANT! INCLUDE THE context-menu.js FILE BEFORE drag-drop-folder-tree.js -->
<script type="text/javascript" src="js/drag-drop-folder-tree.js"></script>
<link rel="stylesheet" href="css/drag-drop-folder-tree.css" type="text/css"></link>
<link rel="stylesheet" href="css/context-menu.css" type="text/css"></link>
</head>

<body>
  <h1>Tree Demo</h1>
  <p>
   See <a href="http://www.dhtmlgoodies.com/index.html?whichScript=drag-drop-folder-tree">DHTML Goodies</a> for
more information.
  </p>
  <t:Tree treeid='literal:demo' source="treeNodes" currentNode="node">
    <t:actionlink t:id="tree" context="node.identifier">
    ${node.name}
    </t:actionlink>
  </t:Tree>

        <script type="text/javascript">
                treeObj = new JSDragDropTree();
                treeObj.setTreeId('demo');
                treeObj.setImageFolder('img/');
                treeObj.setRenameAllowed(false);
                treeObj.setDeleteAllowed(false);
                treeObj.initTree();
                treeObj.expandAll();
        </script>

        <p> <a href="#" onclick="treeObj.collapseAll()">Collapse        all</a> | <a href="#" onclick="treeObj.
expandAll()">Expand all</a></p>


</body>
</html>
```

and

```
import java.util.ArrayList;
import java.util.List;

import nl.tno.secureit.data.TreeNode;

public class TreeDemo {
        private TreeNode node;

        private List<TreeNode> treeNodes;

        public TreeDemo() {
                super();
                treeNodes = new ArrayList<TreeNode>();
                treeNodes.add(new TreeNode(1, "Food", 1));
                treeNodes.add(new TreeNode(2, "Fruit", 2));
                treeNodes.add(new TreeNode(3, "Red", 3));
                treeNodes.add(new TreeNode(4, "Cherry", 4));
                treeNodes.add(new TreeNode(5, "Yellow", 3));
                treeNodes.add(new TreeNode(5, "Banana", 4));
                treeNodes.add(new TreeNode(6, "Meet", 2));
                treeNodes.add(new TreeNode(7, "Beef", 3));
                treeNodes.add(new TreeNode(8, "Pork", 3));
        }

        /**
         * The action that is called after clicking on a tree node
         * @param index
         */
        void onActionFromTree(long index) {
                treeNodes.add(new TreeNode(9, "Chicken", 3));
        }

        // See also the excellent article on hierarchical trees in SQL at:
        // http://www.sitepoint.com/article/hierarchical-data-database/2
        public List<TreeNode> getTreeNodes() {
                return treeNodes;
        }

        public TreeNode getNode() {
                return node;
        }

        public void setNode(TreeNode node) {
                this.node = node;
        }
}
```

For completeness, I've added the TreeNode.java file:

```
public class TreeNode implements ITree {
	private long identifier;

	private String name;

	private int depth;

	public TreeNode(long identifier, String name, int depth) {
		super();
		this.identifier = identifier;
		this.name = name;
		this.depth = depth;
	}

	public int getDepth() {
		return depth;
	}

	public void setDepth(int depth) {
		this.depth = depth;
	}

	public long getIdentifier() {
		return identifier;
	}

	public void setIdentifier(long identifier) {
		this.identifier = identifier;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

}
```

Finally, the Tree.java component that creates the unsorted list (UL) that is rendered into a tree (there is no Tree.html file needed):

```
import java.util.Iterator;
import java.util.List;

import ITree;

import org.apache.tapestry.MarkupWriter;
import org.apache.tapestry.annotations.Environmental;
import org.apache.tapestry.annotations.Parameter;
import org.apache.tapestry.annotations.SupportsInformalParameters;
import org.apache.tapestry.services.Heartbeat;

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

@SupportsInformalParameters
public class Tree<T extends ITree> {
	/**
	 * Current depth of the node in the tree
	 */
	private int currentDepth;

	/**
	 * Iterator to iterate over all tree elements
	 */
	private Iterator<T> iterator;
```

```
        /**
         * Defines the source Tree to walk over.
         */
        @Parameter(required = true)
        private List<T> source;

        /**
         * Id of the tree
         */
        @Parameter(required = true)
        private String treeId;

        /**
         * Defines the current node of the tree
         */
        @Parameter
        private T currentNode;

        @Environmental
        private Heartbeat heartbeat;

        boolean setupRender(MarkupWriter writer) {
                if (source == null)
                        return false;

                currentDepth = 0;
                this.iterator = source.iterator();
                writer.element("ul", "class", "dhtmlgoodies_tree", "id", treeId);
                return (iterator.hasNext());
        }

        /** Begins a new heartbeat. */
        void beginRender() {
                currentNode = iterator.next();
                heartbeat.begin();
        }

        void beforeRenderBody(MarkupWriter writer) {
                writer.writeRaw(getIndentation(currentNode.getDepth()) + "<li id='node" + currentNode.
getIdentifier() + "'>");
        }

        /** Ends the current heartbeat. */
        boolean afterRender() {
                heartbeat.end();
                return (!iterator.hasNext());
        }

        void cleanupRender(MarkupWriter writer) {
                writer.writeRaw(getIndentation(0));
                writer.end();
        }

        /*
         * 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)
                        while (currentDepth > 0) {
                                currentDepth--;
                                s += "</li></ul>";
                        }
                        return s;
                }
                */
```

```
                if (currentDepth == 0) {
                        // First time
                        currentDepth = 1;
                        return "";
                }
                if (currentDepth > depth) {
                        s = "</li>";
                        while (currentDepth > depth) {
                                currentDepth--;
                                //if (currentDepth > 0)
                                        s += "</ul></li>";
                                //else
                                        //s += "</ul>";
                        }
                } else if (currentDepth < depth) {
                        // The difference can never be more than 1 (which would mean
                        // skipping a level)
                        currentDepth++;
                        s = "<ul>";
                } else // currentDepth==depth
                        s = "</li>";
                return s;
        }
}
```

That's all, folks - Any improvements are welcome!