ContribTableWithLinks

NOTE: This is outdated information that applies only to Tapestry 4.

One of the first things I wanted to do when learning Tapestry was to have a table that listed some files on the filesystem. I also wanted a column where a link would make things happen for the file on that row (I wanted a delete function). I also wanted the column where the file name was listed to be a link where you could download the file.

Basically something like this:

File	action
somefile.txt	Delete
someotherfile.txt	Delete

So this is how I did it: I created a java class to model the rows. I called mine `FileEntry`. It is a simple bean class, something like this:

```
public class FileEntry {
        private int id ;        // This is an identification I use (a database key)
        private File filename ; // The name to display
}
```

I also implemented the getters and setters (getId/setID etc).

Now, let say that the page that displays the listing is called `FileList` so we have a page class `FileList.java`, an HTML template called `FileList.html` and a page specification called `FileList.page`

In the page specification I have the specification for the table component and some parameters that the table component uses.

```
<component id="fileTable" type="contrib:Table">
        <binding name="source" expression="files" />
        <static-binding
                name="columns"
                value="filename:File:filename,!action"
        />
        <binding name="row" expression="fileEntry"/>
</component>

<property-specification
        name="fileEntry"
        type="nu.zoom.example.FileEntry"
/>

<property-specification
        name="files"
        type="nu.zoom.example.FileEntry[]"
        persistent="yes"
/>
```

Basically what this does is to declare a component called fileTable. The table is told that the source is a variable on the page called files.

The columns declaration tells the table that the first column is called filename, It should be presented with a header string "File" and the attribute that actually supplies the value is called filename (this matches the `FileEntry` class). The second column only have an id.

The row binding tells the table that when it renders each row it should set a variable called fileEntry to the current row from the source. This will be used later on when we render the links in the table.

The property specifications tells tapestry about the page variables we want (Tapestry creates them for us).

In the `FileList.java` I have the following

```
public abstract class FileList extends BasePage implements PageRenderListener
{
        public abstract FileEntry[] getFiles();
        public abstract void setFiles(FileEntry[] archive);

        public void pageBeginRender(PageEvent pageEvent)
        {
                if (getFiles() == null) {
                        FileEntry[] filesToShowInTheTable ;

                        // Here you construct the collection that
                        // is to be displayed
                        // in the table. When you are done 'save'
                        // it by using setFiles

                        setFiles(filesToShowInTheTable) ;
                }
        }
}
```

So what we do is when the page is about to be rendered we check if we have already created a listing of the files we want to present. If not so we go ahead and actually build the array of `FileEntries` that we want to represent.

Finally in the HTML-template the only thing needed is:

```
<span jwcid="resultTable" />
```

This should get you a table that contains the filename but no links. So let's have a go at them.

To render the delete link in the table we start with adding a `DirectLink` component to the page specification, so add the following to `FileList.page`

```
<component id="delete" type="DirectLink">
    <binding name="listener" expression="listeners.deleteAction"/>
    <binding name="parameters" expression="fileEntry.id" />
</component>
```

This tells tapestry that when we click on the link it should call a method on the page class called deleteAction. It will pass in the id of fileEntry as a parameter.

This is how it looks in the `FileList.java`

```
public void deleteAction(IRequestCycle cycle) throws IOException
{
        Object[] parameters = cycle.getServiceParameters();
        Integer id = (Integer) parameters[0];
        trashDocument(id); // Delete the file
}
```

Now we only need to make the table render the link. This can be done in the HTML-template by using a Block component. So we change the template to look like this:

```
<span jwcid="resultTable">
        <span jwcid="actionColumnValue@Block">
                <a jwcid="delete">Delete</a>
        </span>
</span>
```

The trick here is that we give the Block component the "magic" name xxxxColumnValue. xxxx in this case matches the column id "action" as we declared it in the page specification.

This has the effect that the table uses our Block as a renderer for the cell value rather than the default renderer. You can also do the same trick for the column header so that the table renders something other than action as the column header for the link.

This should be enough to get the "delete" link in the table 🙂