Exercises: Composite Components 1

1. Make a composite component named “test1” (e.g., <utils:test1/>, or, in general, <yourFolderNameUnderResources:test1>) that just spits out a heading that says “This is a test”. The hardest part about composite components is getting all the pieces in the right place and getting all the funny references right, so this seemingly trivial example is really the most important one of this set of exercises. If you want, you could start by stealing “myComponent.xhtml” from the WebContent/resources/utils/ folder of the source code from the lecture (i.e., “composite-components-1”). Note: Eclipse will not recognize the URL in your main page that refers to the folder of your component, and will sometimes give an incorrect warning that the URL is invalid.

2. Make a composite component named “test2” that spits out a heading that uses a message (required) and style (optional) that are supplied as attributes. E.g., the output of <utils:test2 message="test" styleClass="style1"/> (or, in general, <yourFolderName:test2 ... />) might be something like “Your message is ‘test’ ”, displayed in the “style1” CSS style.

3. Make a composite component that, given a Customer object, outputs a bulleted list of that customer’s ID, first name, last name, and balance. Support an optional CSS style as in the previous problem. You probably have a Customer class lying around from a previous set of exercises, but if not, grab Customer (and TestBean) from my composite-components-exercises project.

4. Make a composite component that accepts a number and two style names. If the number is 0 or greater, it should be displayed in the first style. If the number is negative, it should be displayed in the second style. Hint 1: review the parts in the EL or looping lectures on conditional output. Hint 2: in the JSF expression language, you can use either “<” or “lt” for less than comparisons, and either “>” or “gt” for greater than comparisons. Depending on where you use the operator, you might need to avoid the explicit “<” so that your code will be compatible with the rules of XML.

5. Extend the Customer class to contain an image property. The value of this property should be a simple string such as “ellison.jpg”. Make a new version of your bulleted-list-of-customer info component that also shows the image (assuming that your app has stored the image file in the appropriate place). Make sure it works regardless of what folder the main page is in. Hint: this problem is just about making the image work regardless of what folder the main page is in. Once you have the image reference done correctly, your new component can call your previous component without rewriting it.