WysiwygWithoutBorder

NOTE: This is outdated information that applies only to Tapestry 4. For the equivalent functionality in Tapestry 5 see

http://tapestry.apache.org/layout-component.html

The Border pattern has become quite entrenched with Tapestry user, but I would like to offer an alternative approach that has merits from a web designers perspective. Consider a classic page template using a custom Border component.

```
<html>
<head>
<title>Tapestry Hangman</title>
<link rel="stylesheet" type="text/css" href="css/hangman.css"/>
</head>
<body jwcid="$content$">
<span jwcid="@Border">

. . .

</span>
</body>
</html>
```

As pointed out in a previous article it needs some reworking to validate. I would add to that; When viewed in a browser it doesn't render like the live page will. I other words: If a web designer hands you a set of static pages that render properly you have to pull them apart to use the @Border as shown here. Or you need to maintain full page html in both the page and Border templates.

If you work with a page design that is recently modern and not some archaic nested <table> tag soup, the html structure is actually quite modest.

The typical structure involves 's containing common content such as titles, footers, menus, adds, etc. Your main objective for these are often to keep them consistent across the site and that is why you might want to use @Border. Instead you should make custom components for those specific 's. Consider this page template:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html jwcid="@Shell" xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en" stylesheets="bean:
stylesheets" title="Tapestry Hangman">
<head jwcid="$remove$">
<title>Tapestry Hangman</title>
<link rel="stylesheet" type="text/css" href="css/hangman.css"/>
</head>
<body jwcid="@Body">
<div id="header" jwcid="@Header">Header DIV</div>
<div id="menu" jwcid="@Menu"><ul><li><a href="#">Link 1</a></li></ul><ul><li><a href="#">Link 1</a></li></ul><
/div>
<div id="content">
. . .
</div>
<div id="footer" jwcid="@Footer">Footer</div>
</body>
</html>
```

The difference between this and the static html is limited and it shouldn't be hard to teach a web designer to design the template himself. While you have to put a bit more lines on each page template, it is fundamental to the css design so it will only change if the site is completely redesigned.

The structure of the templates for Header, Menu and Footer are very simple so there is no need to ensure that they render properly in a browser.

Room for improvement:
1) It would be nice to be able to clear prior output when Shell starts rendering and have it output a common DOCTYPE.

2) Doctype,mimetype,css,language should be determined by a common object referenced by Shell rather than the shell.

3) Include working sample app.

Other reasons to use this approach.
1) You can add tags to the <head> on individual pages, and specify the title in an intuitive way.

2) You can give <body> an id tag for special css rules.

3) You can vary the structure menus/titles on individual pages.

4) Quirks/Standard rendering is the same static or dynamic.