Jetspeed-2 development documentation

Jetspeed- 2 Layouts and decorator handling

Revision	***Author***	***Comment***
0.1	Roger Ruttimann/David Taylor/Scott Weaver	Initial draft

Table of Contents

1.Introduction..1
Glossary..2
2.Current implementation...2
File structure and location of layout information..................................2
Graphical overview..3
Loading the layout and decorators...3
3.Proposed changes..4
Guidelines..4
Changes to the file structure..4
Changes to the Aggregator...5

1.Introduction

The current implementation of Jetspeed includes the initial framework for handling layouts and decorators. The first task would be done finalize the architecture and document it.

The purpose of this document is to describe the current state, propose changes and serve as the design document for the J2 Layouts. Over time chapter 3 (proposed changes) should disappear.

Glossary

Her a list of terms and definitions used in the layout and decorator handling specifications:

Term	Description
Layout	Defines the fashion in which grouping of Fragments will organized relative to the final, aggregated content of a request to the portal. A Page is an example of a grouping of Fragments.
Layout-decorator	Frame around page
Fragment	Content whose source is that of a dynamic nature e.g. generated by a Porlet or a Page The content of a Fragment should not be manipulated by a Decoration.
Decoration	Any static or semi-static markup surrounding a dynamically generated Fragment. Decoration does not have access to alter a Fragment's generated mark up.
Page	An aggregate of Fragments. A Page is considered to be type of Fragment and generates immutable markup as such.
Page decorator	A type of decoration specifically designed decorate a Page's generated markup. Page markup consists of that Page's Layout and Fragments.
Portlet decorator	A type of decoration specifically designed decorate a Portlet's generated markup.

2.Current implementation

File structure and location of layout information

The layout information is stored in the following location:

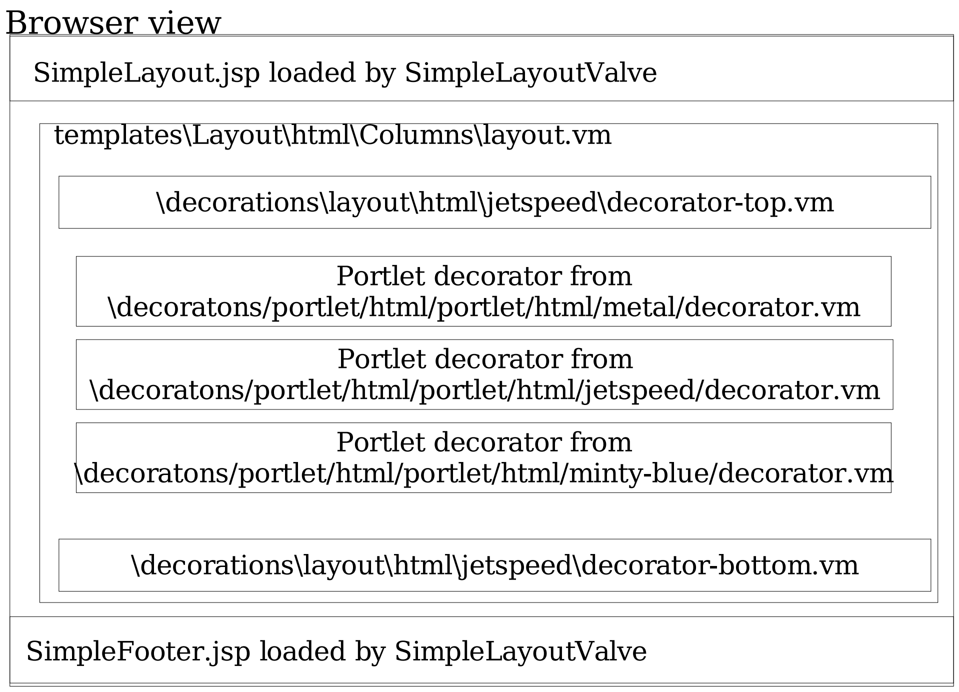
\WEB-INF\templates\layout\html\

The decorator information is stored in the following location:

\WEB-INF\decorations\portlet\html

Graphical overview

The following picture shows the graphical overview (what's on the screen) and what elements (layouts/decorators) are involved. For simplicity we use Jetspeed as the Layout.


Browser view
SimpleLayout.jsp loaded by SimpleLayoutValve
templates\Layout\html\Columns\layout.vm
\decorations\layout\html\jetspeed\decorator-top.vm
Portlet decorator from \decoratons/portlet/html/portlet/html/metal/decorator.vm
Portlet decorator from \decoratons/portlet/html/portlet/html/jetspeed/decorator.vm
Portlet decorator from \decoratons/portlet/html/portlet/html/minty-blue/decorator.vm
\decorations\layout\html\jetspeed\decorator-bottom.vm
SimpleFooter.jsp loaded by SimpleLayoutValve

The above graphic shows that the layout and the decorator are coming out of two different subdirectories in WEB-INF.

Loading the layout and decorators

The loading of the pages are implemented in two components the SimpleLayoutValve and the aggregator. The code in the valve is simple and can be accomplished by the aggregator (see proposed changes in chapter 3).

1. For each request the page header and footer of an HTML page are added by the VerySimpleLayoutValveImpl.
    - The valve implementation isn't mimetype sensitive and just creates HTML format.

2. The AggregatorValve reads the psml file (pages directory) which defines the tempaltes and portlet decorators to load.
    - Two template locators are created (templates & decorations) to find the layout and decorators based on names.

3.Proposed changes

This chapter lists proposed changes to the current implementation.

Guidelines

Once the proposed changes are in the stream the section should be removed from this chapter and added to chapter 2.

Changes to the file structure

Proposal: Update naming convention. Uses pages instead of Layout.

Task:

Change directory structure under WEB-INF/ to from Layout to page (e.g decorations/page/). It's easier to understand the concepts of pages and Layout. A page could switch it's layout from 3 col to 2 col and still retain the same decoration. I like to think of a Page as using or containing a layout, and the decoration decorates the Page itself not the layout, which in fact is itself a portlet generating a fragment.

Proposal: Adding style sheets to decorations

Task: Currently, the style definitions for the portlet decorations is stored in the page decoration (Layout). We need to changes so that each decoration can contribute a self-contained CSS to be added dynamically by the page rendering process.

Location for style sheets:

Portlet decorator:

decorations/portlet/{MIME type}/{decorator name}/css

Page decorator:

decorations/page/{Mime type}/{decorator name}/css

Changes to the Aggregator

Proposal: Add style sheet processing to aggregator engine

Task: The aggregator needs to be aware that the decorations will be contributing to the list of style sheets to be added. We would make it so that the Styles.css is processed dynamically. This would be easy as we could make the velocity servlet also process .css along with .vm files. So the top level css would look like this:

```
#foreach($cssUrl in cssUrls)
 @import url("$cssUrl")
#end
```

Proposal: The aggregator code should use the mimetypes form the capability map in order to load the correct layouts/decorators (html/wml/..).

Task:

> Remove hard coded instances of mimetype (html) with preferred mimetype from the capability map.

Proposal: Remove SimpleLayoutValve and add functionality to the aggregator

Task:

> The SimpleLayoutValve code only includes the header and the bottom of an HTML page which is an overkill. The code needs to be removed and added to the aggregator.

Proposal: Move WEB-Inf\templates directory content under Web-Inf\decorations\Layout

Task:

> Simplify file structure and use just one template locator instead of two. Grovi script should only create one template locator.