forrest:views - x formats, one config

This plugin depends on a ViewHelper and a BusinessHelper implementation.

Table of contents

1 How it works 2
2 Resume.. 3

1. How it works

```
This plugin has three components:
  1. viewHelper - delivers contracts in form of xsl:templates
  2. businessHelper - delivers content that is used in the contracts
  3. views - prepares and transforms the requested contracts (viewHelper)
     and populate them with the content (businessHelper)

1. viewHelper
  This is the template producing factory.

a. <map:match pattern="resolve.contract.*.*"> -> Resolving factory
Project implementation of templates have priority before default ones.
If no implementation can be found we use the noFt (~ - no
forrest:template) implementation.-> this match is implemented in the viewHelper plugin

b. <map:match pattern="get.contract.*.*">-> this match is implemented in the
viewHelper plugin
Get the xsl:templates of the requested contract and specific format.

c. <map:match pattern="get.contract-property.*">-> this match is implemented in the
viewHelper plugin
Get the forrest:properties of the requested contract.
This will determine which templates (css, head, body) we have to call
later on

GENERAL:
Contracts are stored in forrest:templates ({contract-name}.ft) please
have a look on the templates/*.ft that the viewHelper implementation contain to get
an idea how your own implementation have to look like.


2. businessHelper
  This is the content producing factory.

NOTE: <map:match pattern="*.page">
  The current factory uses the models the skin (e.g. document2xhtml.xsl from pelt
[default skin]).
  It is only exchanging the last model of the models the skin is producing
(site2xhtml.xsl) till now.
  This will have to be changed in the future.

3. views
  prepares the requested contracts viewHelper and dispatches
the corresponding businessHelper.

a.  <map:match pattern="prepare.view.*"> -> View config resolver
File specific views have priority before default ones.
If no view can be found in the project we use the default one of the
views plugin.

b.  <map:match pattern="prepare.include.*.**">
Aggregate the contract-templates requested by the view with xinclude.
The result is a stylesheet with all needed xsl:templates.

c. <map:match pattern="prepare.properties.*.*">
Aggregate the forrest:properties requested by the *.fv.
The result is an aggregation of properties which defines the templates
to be call.

d. <map:match pattern="prepare.xhtml.*">
Aggregate all contracts-templates requested by the view.
Create a xsl that can be used for the last step of the transformation of
the view.
```

```
e. <map:match pattern="*.html"> -> Last processing step.
Here we are overriding the default skin generation.
-> this match is implemented in the viewHelper plugin

GENERAL:
Views are stored in forrest:view ({file-name}.fv) please have look on
plugins/org.apache.forrest.plugin.internal.structurer/src/documentation/common.fv do
get an idea how your own implementation have to look like.
```

2. Resume

```
The views plugin can be seen as prototype for the next generation
skinning of forrest. It is still in early stage but with extracting the businessHelper
we
hope to make it easier for all devs (not only committer) to get the idea
and enable them to enhance the design of this plugin and their implementation.

Some basic and simple hints:
a) If you want another implementation of a contract then create a folder
"templates" in ${project.resources-dir} and it will be matched before
the standard implementation.

e.g. feedback contract:

<forrest:contract name="feedback" type="nugget"
  xmlns:forrest="http://apache.org/forrest/templates/1.0">
<description>
This function will output the html feedback information.
</description>

<forrest:template
xmlns:forrest="http://apache.org/forrest/templates/1.0"
format="xhtml" name="feedback" inputFormat="xsl" body="true"
head="false" css="true">

<xsl:stylesheet version="1.1"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:jx="http://apache.org/cocoon/templates/jx/1.0">

<xsl:template name="feedback-css">
#feedback {
    color: black;
    background: #CFDCED;
    text-align:center;
    margin-top: 5px;
}
#feedback #feedbackto {
    font-size: 90%;
    color: black;
}</xsl:template>

<xsl:template name="feedback-body">
<div id="feedback"> Modified project implementation
<xsl:value-of select="$config/feedback"/>
<xsl:choose>
<xsl:when test="$config/feedback/@href and
not($config/feedback/@href='')">
  <a id="feedbackto">
    <xsl:attribute name="href">
      <xsl:value-of select="$config/feedback/@href"/>
      <xsl:value-of select="$path"/>
    </xsl:attribute>
    <xsl:value-of select="$config/feedback/@to"/>
  </a>
```

```
</xsl:when>
<xsl:otherwise>
  <xsl:value-of select="$config/feedback/@to"/>
</xsl:otherwise>
</xsl:choose>
</div>
</xsl:template>
</xsl:stylesheet>
</forrest:template>
</forrest:contract>

The @attributes (body="true" head="false" css="true") of the
forrest:template defining which parts of the html page (head, head-css
and body) we have to render. The xsl:templates are following the simple
naming convention {@name}-(css|head|body).


b. If you want a default view for your project then copy the common.fv
from the viewHelper implementation to your ${project.conf-dir} and modify this file.
When
you want another view for a specific file (e.g. ${project.xdocs-dir}/index.html) then
copy
the common.fv to your ${project.xdocs-dir} and renamed it to
${project.xdocs-dir}/index.fv.
```