How to use the forrest:view config-DSL

This How-To describes the usage of the forrest:view config Domain Specific Language to create a beautiful website in no time.

Table of contents

1. Intended Audience

Warning:
"Views" are a new functionality which is still in development phase. That is why it is in the "whiteboard" section of the Forrest distribution. This HowTo is far from being finished.

This part of the views is dedicated to web designers and users with some knowlegde of css.

2. Purpose

This how-to will show you how to write a **forrest:view** from the ground up. We will focus on html as the output format and show how to add your own css implementation to the view.

3. Prerequisites

- You have a ready-to-go new seed (newSeed) based on views like described in Install views.
- This includes as well all additional plugins that are mentioned in Install views.
- Reading the Intall views HowTo is also a good idea to help understand the dir-structure used in this how-to.

4. Steps

Note:
When developing with views we assume you are using 'forrest run' and the following workflow:
change files -> refresh browser

We developed **forrest:view** to let the user decide where to place elements in, e.g., html pages. We started this work with the `skinconf.xml` where you could configure certain elements and their positions. These elements were known by established names and it was up to the skin-designer to support this configuration and the elements.

We started by grouping the elements (from skinconf) and using css contracts that we added as @attributes, e.g. `<div id="content-main"/>` . That made it possible to use the same elements in different skins. For the full list of these contracts refer to the initial contract list .

Around these contracts we developed a configuration Domain Specific Language and called it **forrest:view** . **forrest:view** allows us to define the order in which **forrest:contract**s appear, and also to group them using **forrest:hook**s.

forrest:hooks are containers that are only used for layout reasons. They **do not** add any content nor functionality to the output. They add **only** layout information to the output. Actually a `<forrest:hook name="layoutId"/>` will be transformed to `<div id="layoutId"/>`

forrest:contracts add functionality or extra content that a skin can use to display the requested document (content-main). Sometimes a contract delivers **format-specific markup**, other times it delivers a **format-independent string**.

4.1. Empty view file

```
<forrest:views xmlns:forrest="http://apache.org/forrest/templates/1.0" >
  <forrest:view type="xhtml">
  </forrest:view>
</forrest:views>
```

forrest:view is designed to be available for any format that can use **forrest:view** as a configuration file. The only format we have implemented is xhtml for now. This is also true for the delivered contracts.

Now lets start to skin our site. :)

4.2. Creating your first view

In this section we will create a new view. We will override the default view of the view plugin for the index page of the newSeed. To do this we will create a file called `index.fv` and save it in our xdocs directory. This will **only** make the index.html page look different from the rest of the project.

Remember: pointing your browser to `http://localhost:8888/ls.contracts.html` will show a page with all contracts that you can use in your project.

Let's use the blank view from the earlier step and add the content-main contract. In ls.contracts.html we find the information about how to use the contract in our view. Our `index.fv` should look like:

```
<forrest:views xmlns:forrest="http://apache.org/forrest/templates/1.0" >
  <forrest:view type="xhtml">
    <forrest:contract name="content-main"/>
  </forrest:view>
</forrest:views>
```

Let's try our new view by pointing to `http://localhost:8888/index.html`. We will see only the main content. :) Now let's add the section navigation to our view. The contract usage in the view can be looked up in ls.contracts.html. Our view now looks like:

```
<forrest:views xmlns:forrest="http://apache.org/forrest/templates/1.0" >
  <forrest:view type="xhtml">
    <forrest:contract name="content-main"/>
    <forrest:contract name="nav-section"/>
  </forrest:view>
</forrest:views>
```

We now see the main content and the section navigation in the order we placed them in the view, but we want them next to each other (left: nav-section; right: content-main).

4.3. Hooks in views

We will now use for the first time a `<forrest:hook name="layoutId"/>`. Hooks are the styling side of views. We can imitate an arbitrary html skeleton with their help. Before we explain how to use your own css in views we will use the default css. In the default.css we can find

```
/* menu */
#leftbar {
    width: 25%;
    float: left;
    background: #eae8e3;
    border: thin dashed #565248;
}
```

With this information we can see the purpose of `<forrest:hook name="leftbar"/>` and add

contracts into that container.

If we want to put the nav-section contract into the left side position of the site we need to place the contract into that hook like so:

```
<forrest:hook name="leftbar">
  <forrest:contract name="nav-section"/>
</forrest:hook>
```

Our view will then look like:

```
<forrest:views xmlns:forrest="http://apache.org/forrest/templates/1.0" >
  <forrest:view type="xhtml">
    <forrest:hook name="leftbar">
      <forrest:contract name="nav-section"/>
    </forrest:hook>
    <forrest:contract name="content-main"/>
  </forrest:view>
</forrest:views>
```

4.4. CSS in views

We now know how to place contracts and hooks in our view, but until this stage we have only used the default.css. CSS support within views is as easy as placing contracts/hooks. To override the default.css stylesheet we use another tag within our view `<forrest:css url="default.css"/>`.

Now we will create a new file as `{project:skins-dir}{path}/{name}.css`. In our case we will save a file called howTo.css in newSeed/src/documentation/skins/css/howTo.css containing only the following css:

```
/* menu */
#leftbar {
    width: 25%;
    float: left;
    background: #CCCCFF;
    border: thin solid #000000;
}
```

We just changed the border-style to 'solid', the background to '#CCCCFF' and the color to '#000000'. Now we have to add a new tag to tell that we want to override the default.css. We are doing this by adding the tag `<forrest:css url="howTo.css"/>` to our view.

Note:
`<forrest:css url="howTo.css"/>` has to be the direct son of `<forrest:view type="xhtml">`!!!

```
<forrest:views xmlns:forrest="http://apache.org/forrest/templates/1.0" >
  <forrest:view type="xhtml">
          <forrest:css url="howTo.css"/>
    <forrest:hook name="leftbar">
      <forrest:contract name="nav-section"/>
    </forrest:hook>
    <forrest:contract name="content-main"/>
  </forrest:view>
</forrest:views>
```

Now you see a white page where the menu is sourrounded by a solid border with the defined background. For a second example let's change the content-main by adding another hook `<forrest:hook name="content"/>`. We need to add the new layout container to our howTo.css:

```
/* The actual content */
#content {
    margin-left: 25%;
    padding: 0 20px 0 20px;
    background: #B9D3EE;
}
```

Then we have to add the 'content-main' contract to the 'content' hook. The resulting view looks like:

```
<forrest:views xmlns:forrest="http://apache.org/forrest/templates/1.0" >
  <forrest:view type="xhtml">
          <forrest:css url="howTo.css"/>
    <forrest:hook name="leftbar">
      <forrest:contract name="nav-section"/>
    </forrest:hook>
    <forrest:hook name="content">
      <forrest:contract name="content-main"/>
    </forrest:hook>
  </forrest:view>
</forrest:views>
```

We are now able to place contracts and layout containers and add custom css to the view.

FIXME (thorsten):

Let us now look into advanced features of views. I will write a how-to for advanced contracts soon. :)

5. Further Reading

Congratulations, you are now able to work with the view DSL. From here we recommend reading the following How-To's:

- Create your own contract implementation

6. Feedback

Please provide feedback about this document via the mailing lists.