Rule Matching

If a rule does not match the theme, it is silently ignored.

If a rule does not match the content, the element is dropped from theme.

Tips

Make sure all necessary parts of the content are being brought in to the theme. This would include things like scripts for analytics, or accessibility helpers like the skip links.

Also for a CMS, you'll need to check that all the editing interfaces function and are properly styled.

Newer versions of Plone's plone.app.theming product have a control panel available within the Plone site for being able to edit the theme. The interactive interface has a theme inspector and rule builder along with a preview to help with building the theme.

Reference

http://docs.diazo.org
https://pypi.python.org/pypi/plone.app.theming

six feet up
where sophisticated web projects thrive

Six Feet Up is a woman-owned business that develops, hosts and supports sophisticated enterprise content management and collaborative intranet projects. Our clients include Harvard, UCLA, Eli Lilly & Oxfam.

Check out our other free Quick Reference cards at www.sixfeetup.com/quickref

six feet up

Quick Reference Series

GETTING STARTED WITH DIAZO THEMING

Diazo is a method of mapping dynamic website content (from a content management system) to display inside of a static theme. This is a great benefit to designers, because they can build a theme with normal HTML and CSS, and hook it up to a CMS without having to learn much of anything about the CMSs and their templating languages. In Plone, Diazo is utilized within the add-on plone.app.theming, which has been included since Plone 4.2.

If you already know how to build a static theme, then the part left to learn is writing the rules. The rules are what connect the dynamic content to the static theme. The folder containing the static theme will generally contain a minimum of the following:

- index.html
- styles.css
- rules.xml
- manifest.cfg

The `manifest.cfg` file is used to provide information about the theme (like the title) which will display in Plone's theming control panel. It also provides a way of creating variables that can be used within the templates.

Syntax

The rules file uses XML syntax with the pre-existing tags.

Each rule generally has two selectors - one for the `content`, and one for the `theme`.

```
<replace theme="/html/head/title"
content="/html/head/title"/>
```

The content selector specifies which dynamic content to grab from the CMS. The theme selector then specifies where in the static HTML file the content will be displayed. This example uses XPath selectors, but CSS selectors can also be used, if that is what you are used to:

```
<replace css:content="#portal-globalnav
a" css:theme="#nav a"/>
```

You can also specify to use the `content-children` or `theme-children` to manipulate the items inside of the selected element:

```
<replace css:content-children="#portal-
-globalnav" css:theme-children="#nav"/>
```

Tags

theme - Specifies which HTML file to use for the theme: `<theme href="index.html" />`

replace - Remove the selected element from the theme, and put in the selected element from the content

before - Place the content before the selected element in the theme

after - Place the content after the selected element in the theme

prepend - Place the content inside the selected element in the theme as the first item

append - Place the content inside the selected element in the theme as the last item

drop - Only uses the `theme` or `content` selector to remove the specified element

strip - Similar to `drop`, but only removes the tag, leaving the content inside the tag

merge - Applies to the attributes of an element, e.g. display the css classes of an element from both the content and theme

copy - Used for copying the attributes of an element in the content to the theme.

Attributes

For `merge` and `copy`, the attributes to be copied need to be specified. This example will merge the classes applied to the body from both the CMS and static theme:

```
<merge attributes="class" css:theme=
"body" css:content="body" />
```

Conditions

Conditions can be used on a rule to determine whether the rule should be applied or not.

if-path - rule will only apply if URL is matched.

if-content / css:if-content - Rule will only apply if specified element exists in the content.

if - Rule will only apply if condition is true. This can check variables that were set up in manifest.cfg.

Rules can also be nested inside a condition. This can be helpful when working with various templates

```
<rules css:if-content="#personal-bar">
<after css:theme-children="#header-box"
css:content="#user-prefs"/>
<after css:theme-children="#header-box"
css:content="#logout"/>
</rules>
```

Order of execution

The order in which the rules are executed is not based on what order they appear in the rules file. Instead, it is done by order of tag type:

- before (*)
- drop
- replace (*)
- strip
- rules for attributes (merge, copy)
- before, replace, after for theme-children
- after (*)

(*) These rules are for all selectors except for those with theme-children. Rules with theme-children are executed 6th.