
hexagonit.portletstyle Documentation

Release 1.0

Hexagon IT

January 13, 2012

CONTENTS

hexagonit.portletstyle is a Plone 4.x add-on that allows you to assign a CSS style to a portlet. You can chose from a list of pre-defined (configurable through Plone Control Panel) classes.

- [Source code @ GitHub](#)
- [Releases @ PyPI](#)
- [Sphinx docs @ ReadTheDocs](#)

INSTALLATION

To install `hexagonit.portletstyle` you simply add `hexagonit.portletstyle` to the list of eggs in your buildout, run buildout and restart Plone. Then, install *hexagonit.portletstyle* using the Add-ons control panel.

USAGE

Click on `Manage portlets` and add or edit a portlet. Select your desired portlet style from the *Portlet Style* drop-down menu and save. That's it.

DEFAULT PORTLET STYLES

By default, this package gives you three default portlet styles:

- No header
- No footer
- No header and no footer

MANAGING AVAILABLE PORTLET STYLES

You can add, edit and remove available portlet styles by going to the *Plone Control Panel* and clicking on the Portlet Styles configlet. Pointing your browser directly to `http://<zope_ip>:<zope_port>/<plone_id>/@@portletstyles` also does the trick.

Here, you can enter your styles, one by line, with a pipe (|) character delimiting CSS class and style title. For example, a line `dummy|Dummy style` would produce a `Dummy style` drop-down menu item that would give the portlet an additional CSS class of `foo`.

Lines are checked for formatting and validity of CSS classes. One style can have multiple CSS classes, for example, the following is valid: `one two|Double class style`.

SUPPORTED PORTLETS

As of this moment, the following portlets are supported (as in, you can select a style when editing them):

- *Events portlet*
- *Navigation portlet*
- *News portlet*
- *Recently changed items portlet*
- *RSS portlet*
- *Collection portlet*
- *Static text portlet*

The rest of out-of-the-box Plone portlets are non-editable ones as you normally need only one instance per site. For these (and other) reasons the following portlets *do not support* choosing a style for them:

- *Calendar portlet*
- *Classic portlet*
- *Language portlet*
- *Login portlet*
- *Review portlet*
- *Search portlet*

IMPORTING PORTLET STYLES FROM YOUR OWN PACKAGE

This package uses *plone.app.registry* to store portlet styles. The added benefit of this is that you can easily control which styles you want to have in your site with GenericSetup. Just add `registry.xml` to `/profiles/default` folder and reinstall your custom product:

```
<?xml version="1.0"?>
<registry>
  <records interface="hexagonit.portletstyle.interfaces.IPortletStyles">
    <value key="portlet_styles" purge="false">
      <element>first-style|First style</element>
      <element>secondStyle|Second style</element>
      <element>multi style|Multiple CSS classes style</element>
    </value>
  </records>
</registry>
```

If you want to first remove whatever styles are already stored in the registry, simply use `purge="true"`.

STYLING THIRD-PARTY PORTLETS

Follow the steps below to convince your third-party portlets to support selecting a style for them.

7.1 Portlet Assignment

You portlet's Assignment class must have an `__init__()` and inside this method it must call base assignment's `__init__()`. To put it in other words, `plone.app.portlets.portlets.base.Assignment` (from which you portlet's Assignment is likely inheriting from) has an `__init__()` method that sets the `self.portlet_style` value. You need to call this `__init__()` from your portlet's assignment's `__init__()`.

An example of how this can be achieved:

```
from plone.app.portlets.portlets import base
class Assignment(base.Assignment):
    implements(IMyCustomPortlet)

    def __init__(self, *args, **kwargs):
        base.Assignment.__init__(self, *args, **kwargs)
        self.foo = kwargs.get('foo', 5)
        self.bar = kwargs.get('bar', "bar")
```

7.2 Portlet AddForm

Each portlet also has an AddForm class with a `create` method. This method must also pass the portlet style as a parameter. To make things simpler, just pass in the entire data.

```
class AddForm(base.AddForm):
    form_fields = form.Fields(IMyCustomPortlet)

    def create(self, data):
        return Assignment(**data)
```

7.3 Template

Use the style in the template to assign an additional CSS class to your portlet:

```
<dl class="portlet portletMyCustom"
    tal:attributes="class string:portlet portletMyCustom ${view/portlet_style}"
    i18n:domain="plone">
```


TRANSLATIONS

Rebuild POT:

```
$ i18ndude rebuild-pot --pot locales/hexagonit.portletstyle.pot --create hexagonit.portletstyle .
```

Sync a translation file with POT:

```
$ find locales -name '*.po' -exec i18ndude sync --pot locales/hexagonit.portletstyle.pot {} \;
```


TODO

- use in the wild
- maybe patch non-editable portlets (login, calendar, search, etc.)
- check how *collective.weightedportlets* patches portlets and learn from it

CREDITS

- Idea, mentoring and sponsoring provided by Hexagon IT Oy.
- Code monkeying by Nejc Zupan, NiteoWeb Ltd.

CHANGELOG

11.1 1.3 (2011-12-02)

- Don't break portlets if this product is installed in buildout but not installed with QuickInstaller. [zupo]
- Cchange default value of portlet_style field to ‘‘ ‘‘ to it's the same as the *Defaul value* we inject into the drop-down menu. [zupo]

11.2 1.2 (2011-11-29)

- If a portlet has a style assigned that is no longer listed in portlet_styles, than it is added to the drop-down menu, so it's still possible to select it. [zupo]
- Renamed `No style` default style into `Default style` and set it as default selected value for the Styles drop-down menu. [zupo]
- Styles formatting and CSS class validation. [zupo]

11.3 1.1.1 (2011-11-26)

- Re-releasing because README syntax was broken. [zupo]

11.4 1.1 (2011-11-26)

- Human-readable styles. [zupo]
- Improving docs. [zupo]

11.5 1.0 (2011-11-20)

- Slovenian translation. [zupo]
- Added translation support for portlet_style field in patched IPortletDataProvider. [zupo]

11.6 1.0a3 (2011-11-07)

- Re-releasing, hoping this fixes problems with `jarn.mkrelease`. [zupo]

11.7 1.0a2 (2011-10-27)

- Added whitespace to *.rst* files in docs/ to fix reStructuredText indentation errors. [zupo]
- Removed `..code-block::` elements from README.rst as they are Sphinx-specific block elements and are not valid reStructuredText. [zupo]

11.8 1.0a1 (2011-10-27)

- Initial release. [zupo]

LICENSE

Copyright (c) 2011, Hexagon IT Oy All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Hexagon IT Oy nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL HEXAGON IT OY BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*