Needle Documentation

Release 0.1a1

Ben Firshman

Apr 23, 2017

Contents

Needle is a tool for testing your CSS and visuals with Selenium and nose.

It checks that visuals (CSS/fonts/images/SVG/etc.) render correctly by taking screenshots of portions of a website and comparing them against known good screenshots. It also provides tools for testing calculated CSS values and the position of HTML elements.

CHAPTER 1

Installation

If you haven't got pip installed:

```
$ sudo easy_install pip
```

As root, or in a virtualenv:

```
$ pip install selenium
$ pip install needle
```

CHAPTER 2

Getting started

Create `test_bbc.py` in an empty directory:

```
from needle.cases import NeedleTestCase

class BBCNewsTest(NeedleTestCase):
    def test_masthead(self):
        self.driver.get('http://www.bbc.co.uk/news/')
        self.assertScreenshot('#blq-mast', 'bbc-masthead')
```

This is a test case which tells the Selenium web driver (by default Firefox) to open BBC News and check the bar across the top of the page looks correct. `assertScreenshot()` take two arguments: a CSS selector for the element we are capturing and a filename for the image.

To create an initial screenshot of the logo, we need to run Needle in 'baseline saving' mode:

```
$ nosetests test_bbc.py --with-save-baseline
```

This will create `screenshots/baseline/bbc-masthead.png`. Open it up and check it looks okay.

Now if we run our tests, it will take the same screenshot and check it against the screenshot on disk:

```
$ nosetests test_bbc.py
```

If a regression in your CSS causes them to become significantly different, the test will fail.

CHAPTER 3

Selecting a WebDriver

You may control which browser is used by Needle by overriding the `get_web_driver()` method:

```
from needle.cases import NeedleTestCase
from needle.driver import NeedlePhantomJS

class MyTests(NeedleTestCase):

    @classmethod
    def get_web_driver(cls):
        return NeedlePhantomJS()

    def test_something(self):
        ...
```

By default Needle uses `NeedleFirefox`, which is a wrapper of Selenium's built-in `selenium.webdriver.firefox.webdriver.WebDriver` class. You may use any of the following WebDrivers: `NeedleRemote`, `NeedlePhantomJS`, `NeedleFirefox`, `NeedleChrome`, `NeedleIe`, `NeedleOpera` and `NeedleSafari`. Refer to Selenium's documentation to learn how to install and configure any of those WebDrivers.

CHAPTER 4

Setting the viewport's size

You may set the size of the browser's viewport using the `set_viewport_size()` method:

```
from needle.cases import NeedleTestCase

class MyTests(NeedleTestCase):

    def test_something(self):
        self.set_viewport_size(width=1024, height=768)
        ...
```

This is particularly useful to predict the size of the resulting screenshots when taking fullscreen captures, or to test responsive sites.

You may also set the default viewport size for all your tests with the `viewport_width` and `viewport_height` class attributes:

```
from needle.cases import NeedleTestCase

class MyTests(NeedleTestCase):
    viewport_width = 1024
    viewport_height = 768

    ...
```

CHAPTER 5

Engines

By default Needle uses the PIL engine (`needle.engines.pil_engine.Engine`) to take screenshots. Instead of PIL, you may also use PerceptualDiff or ImageMagick.

Example with PerceptualDiff:

```
from needle.cases import NeedleTestCase

class MyTests(NeedleTestCase):
    engine_class = 'needle.engines.perceptualdiff_engine.Engine'

    def test_something(self):
        ...
```

Example with ImageMagick:

```
from needle.cases import NeedleTestCase

class MyTests(NeedleTestCase):
    engine_class = 'needle.engines.imagemagick_engine.Engine'

    def test_something(self):
        ...
```

Besides being much faster than PIL, PerceptualDiff and ImageMagick also generate a diff PNG file when a test fails, highlighting the differences between the baseline image and the new screenshot.

Note that to use the PerceptualDiff engine you will first need to download the `perceptualdiff` binary and place it in your `PATH`.

To use the ImageMagick engine you will need to install a package on your machine (e.g. `sudo apt-get install imagemagick` on Ubuntu or `brew install imagemagick` on OSX).

CHAPTER 6

File cleanup

Each time you run tests, Needle will create new screenshot images on disk, for comparison with the baseline screenshots. It's then up to you whether you want to delete them or archive them.

Set the `cleanup_on_success` class attribute to `True` to delete these files for all successful tests. Any screenshots that differ from the baseline will remain on disk for your inspection. Example:

```
from needle.cases import NeedleTestCase

class MyTests(NeedleTestCase):
    cleanup_on_success = True

    def test_something(self):
        ...
```

By default, `cleanup_on_success` is `False`.

You may also activate the file cleanup from the command line by using the `--with-needle-cleanup-on-success` nose plugin:

```
$ nosetests –with-needle-cleanup-on-success
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search