django-selenium Documentation

Release 0.9.5

Roman Prokofyev

February 16, 2017

Contents

Contents

CHAPTER 1

Django 1.4 note

Django 1.4 got built-in selenium support, and you can continue to use **django-selenium** with it, while keeping the same shortcut webdriver functions. How to use django-selenium on django 1.4:

- specify preferred webdriver in the **SELENIUM_DRIVER** setting
- create test classes subclassing the **SeleniumLiveTestCase** inside the standard **tests.py** file.

CHAPTER 2

What is it?

Django-selenium is a library that provides seamless integration for Django framework with a Selenium testing tool. Additionally it provides syntactic sugar for writing and maintaining selenium tests (see *MyDriver class* section).

It allows to write and execute **selenium tests** just as normal ones.

CHAPTER 3

Dependencies

- Django 1.3 and above.
- Selenium 2.5.0 and above.
- django-jenkins if you are going to use **JenkinsTestRunner** from this package.

CHAPTER 4

How to use it

Define **selenium specific** settings in your **settings.py** file.

Local

Local run in this case means that you're using Firefox, Chrome or IE driver, and therefore you don't need running selenium server, because these drivers work with the browsers directly.

So, if you plan to use **selenium locally**, then you should define the following settings:

- Set `SELENIUM_DISPLAY` if you plan to run selenium tests on display other than "`:0`" (on VNCServer/Xvfb for example). See `settings.py` for other settings available.
- Set `SELENIUM_DRIVER` for corresponding browser driver in selenium.
- Optionally, set `SELENIUM_DRIVER_OPTS` as a dictionary with options to be passed to the selenium webdriver. This option can be used for instance to pass a custom firefox profile path to be used, or a custom path for `chromedriver`. See the Selenium webdriver's drivers page for more information.

Remote

- Set `SELENIUM_DRIVER = 'Remote'` in your settings file.
- Set `SELENIUM_CAPABILITY` to the desired value.
- Probaly set `SELENIUM_PATH` to point to the selenium-server.jar on your system, for example `/home/dragoon/selenium-server-2.6.jar`. This setting is required if you want to start selenium server along with tests. You don't need this if you keep your selenium server running using other methods.
- Set `SELENIUM_HOST` to point to the IP/hostname of your remote selenium server.
- Set `SELENIUM_TESTSERVER_HOST` to the IP address/hostname of the machine where test server is running (e.g. **192.168.1.2**).

See settings.py file to see some examples.

Common

1. Set `TEST_RUNNER = 'django_selenium.selenium_runner.SeleniumTestRunner'` or subclass `SeleniumTestRunner` to make your own test runner with extended functionality.

2. Write some selenium tests for your apps in a module `seltests.py`. Subclass selenium tests from `django_selenium.testcases.SeleniumTestCase`.
3. Add custom management command to override default test command:

```
from django_selenium.management.commands import test_selenium

class Command(test_selenium.Command):

    def handle(self, *test_labels, **options):
        super(Command, self).handle(*test_labels, **options)
```

   Place it somewhere in your app in `management/commands/test.py` (don't forget the __init__.py files in each directory)

5. Run `manage.py test` like you normally do. Now you have two extra options: `--selenium` and `--selenium-only`. First runs selenium-specific tests after the usual ones, the last runs only selenium tests.

And that's it.

To see the integration in action, check out the test application included in the source.

CHAPTER 5

Django Jenkins

There is also a special test runner to execute selenium tests using django-jenkins integration: `django_selenium.jenkins_runner.JenkinsTestRunner`.

You can specify this class for `JENKINS_TEST_RUNNER` setting, and `manage.py jenkins` command will also execute selenium tests and generate reports for them.

CHAPTER 6

MyDriver class

`MyDriver` class from `django_selenium.testcases` offers extended functionality on top of `selenium.webdriver.remote.webdriver.WebDriver` class.

This class contains a number of convenient shortcuts to handle frequently used operations described below:

CHAPTER 7

South

You use South to migrate your applications ? Ok, south is also overriding the django test commands, therefore you will need to modify your custom management command as follows:

```
from django_selenium.management.commands import test_selenium
from south.management.commands import test as test_south

class Command(test_south.Command, test_selenium.Command):

   def handle(self, *test_labels, **options):
       super(Command, self).handle(*test_labels, **options)
```

In addition, you need to set the following:

- `SOUTH_TESTS_MIGRATE = False` in your `test_settings.py`
- if you called your command "test" (test.py), then the app containing your command should come after the "south" in the `INSTALLED_APPS` list.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search