django-jenkins Documentation

Release 0.19.0

Mikhail Podgurskiy, Gabriel Le Breton

Jun 04, 2017

Contents

Plug and play continuous integration with django and jenkins

There are a lot of ways to enable continuous integration with your django project. But most of them require a lot of time to configure and setup ci server and make changes in your django project.

This tutorial introduces a new way to enable continuous integration for django project, with minimal projects modifications by means of with django-jenkins.

Previously, ad-hoc integration django with jenkins required using nose testing frameworks instead of django native unittests. Since nose project uses custom test loader it leads to a lot of problems with django code - duplicate signals subscriptions, module loading and other test-only realted errors. Django-jenkins uses a standard unittest runner, so the code under test works like in production. No difference between django-jenkins run and manage.py test so keeps your hands free for real work.

CHAPTER 1

Indices and tables

- genindex
- search

CHAPTER 2

Getting started

Configuring django project

To enable django_jenkins you need only to add `django_jenkins` to `INSTALLED_APPS` in `settings.py`.

Running

```
$ python manage.py jenkins
```

will do the same job as

```
$ python manage.py tests
```

but also will create reports folder in the root of your django project with jenkins parsable pylint, test coverage and tests reports.

Django-jenkins have support for several other testing tools/ To enable it, you should include tools task to `JENKINS_TASKS` settings variable.

```
JENKINS_TASKS = (
    'django_jenkins.tasks.run_pep8',
    'django_jenkins.tasks.run_pyflakes',
    'django_jenkins.tasks.run_jslint',
    'django_jenkins.tasks.run_csslint',
    'django_jenkins.tasks.run_sloccount'
)
```

Please, note that corresponding task tool should be installed on jenkins server manually. Please refer to django-jenkins README for specific task dependencies list.

In order to get the right coverage, `'django_jenkins'` app should be included as early as possible in `INSTALLED_APPS`

This tutorial doesn't cover the library dependency management and deploying your django project on external server. Basically you could setup the CI server as you did in your local environment.

But if you prefer automatically installation and configuration dependencies on CI server, you could easily add virtalenv support for your project.

Add requirements to your `requirements.txt` file:

```
Django
django-jenkins
# any other libraries for your project
```

Running

```
$ virtualenv --python=python2.6 env
$ env/bin/pip install -r requirements.txt
```

will create local folder env with the required libraries for your project. Running those commands on other servers will help to sync the external dependencies.

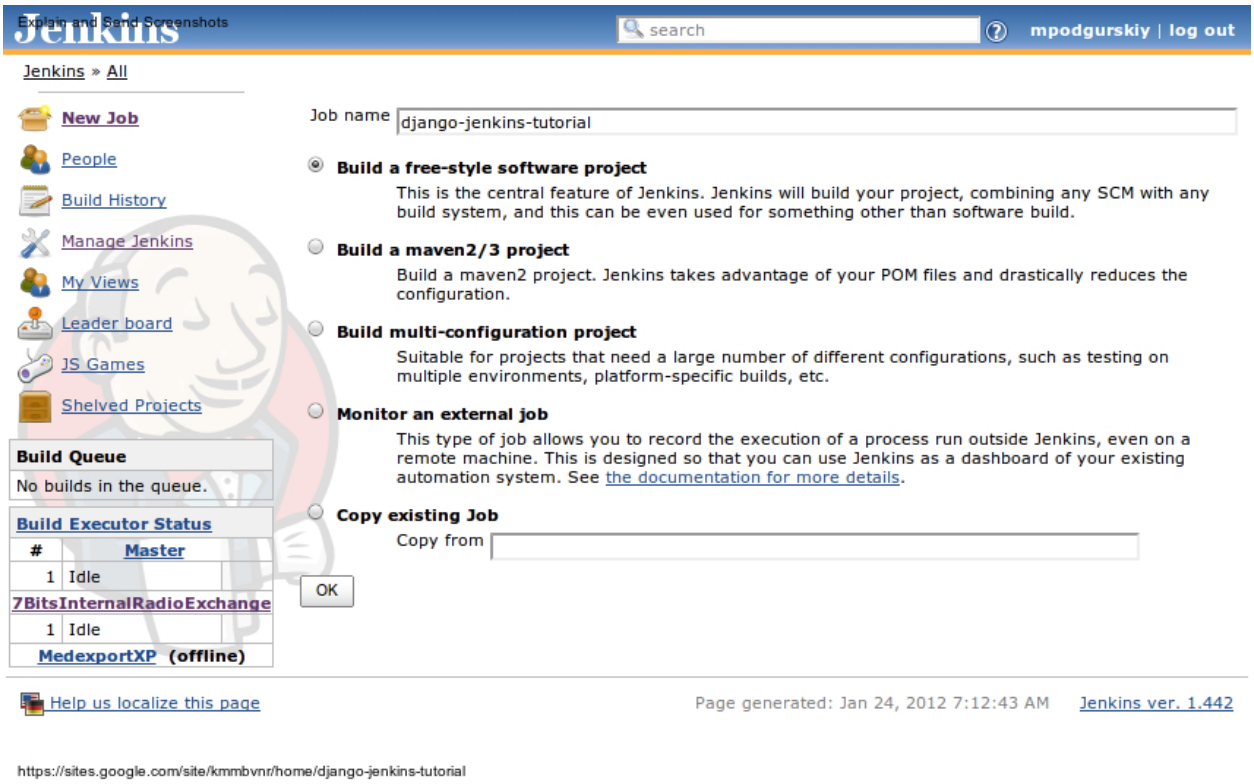
Configuring jenkins

After a fresh Jenkins installation, you'll need to install two required plugins:

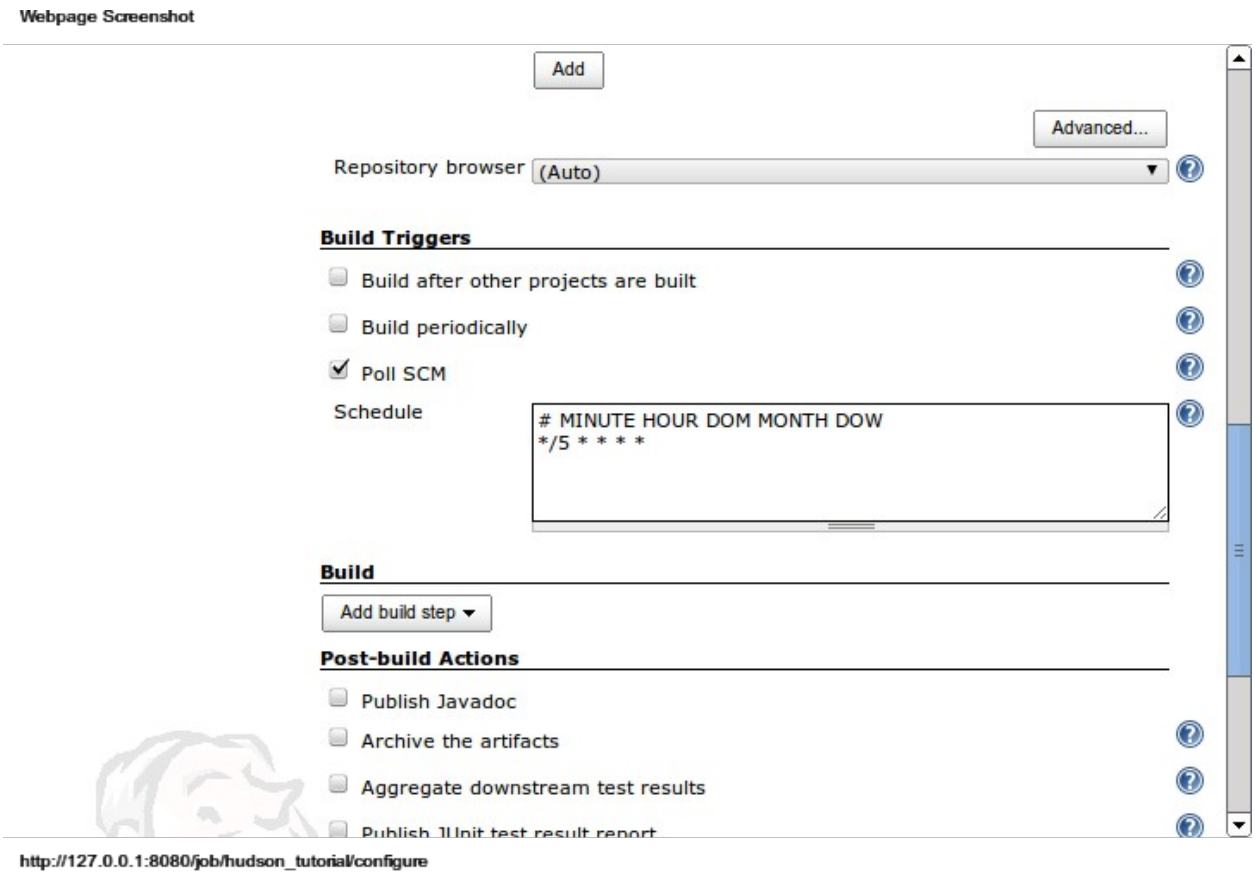
- Violations plugin for parsing `pylint` reports and
- Cobertura Plugin for showing the code coverage.

Install these plugins via `Manage Jenkins -> Manage Plugins -> Available`.

Start new job with creating free-style project:



After configuring the repository access, set up the build trigers. Poll SCM will run the build if any changes in repository are found. The line `*/5 * * * *` means checking repository every 5 minutes.



Select `Add build step -> Execute shell`. Add comands to setup environment and the run tests command

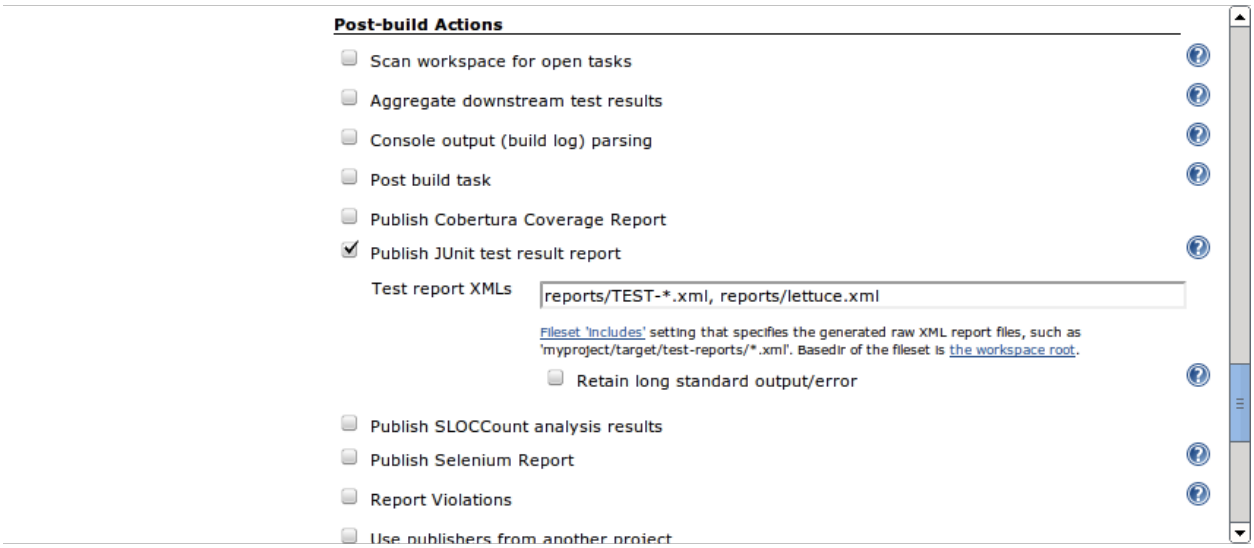
```
$ python manage.py jenkins --enable-coverage

    .. image:: _static/jenkins-2.png
```

Specify the location of test reports - `reports/TEST-*.xml` and `reports/lettuce.xml` (in case you are using lettuce tests) files.

CHANGED in 0.13.0:: test reports from `TEST-*.xml` now stored in one file: `junit.xml`.

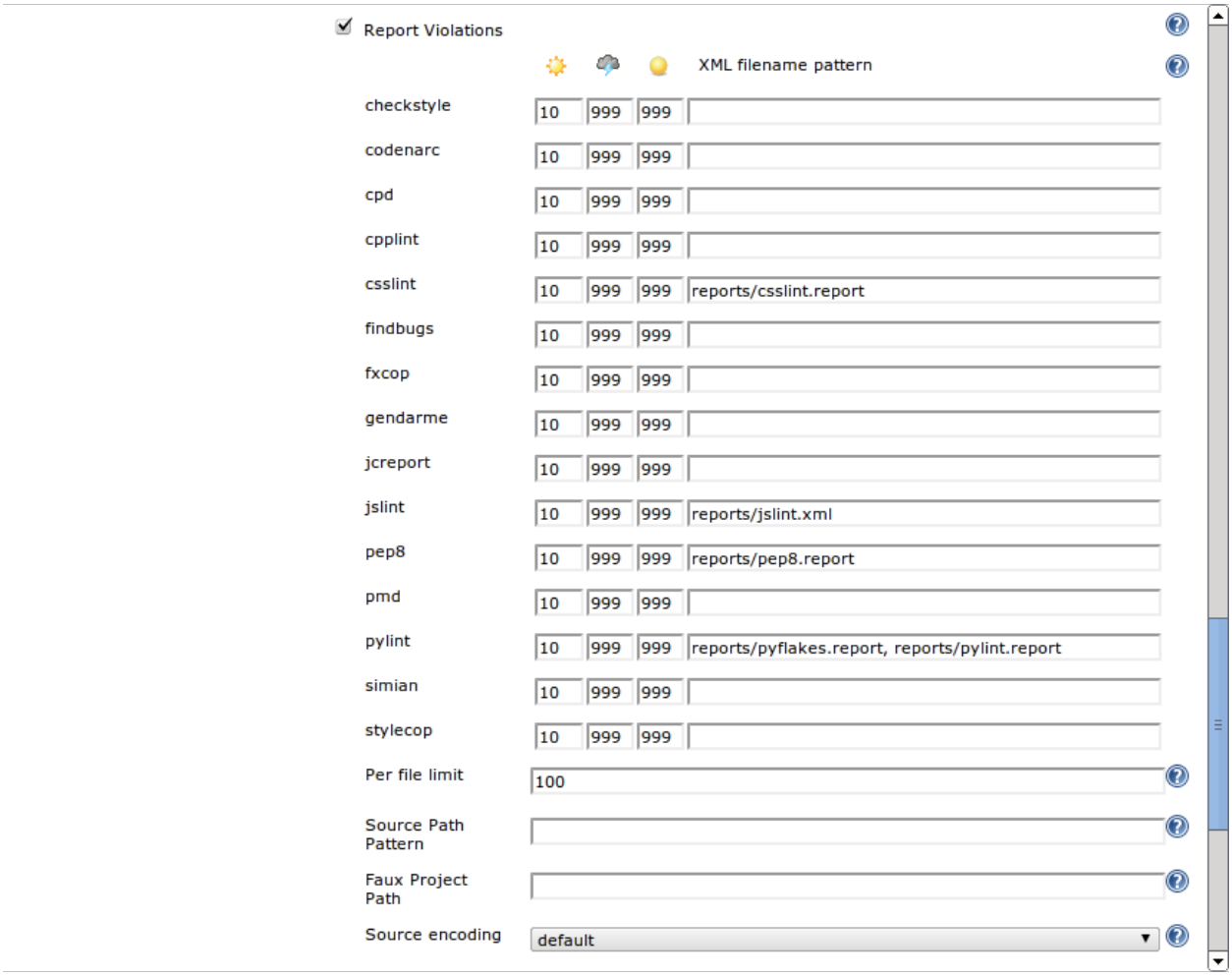
Webpage Screenshot



http://build.fairdevteam.com/job/django-jenkins-tutorial/configure

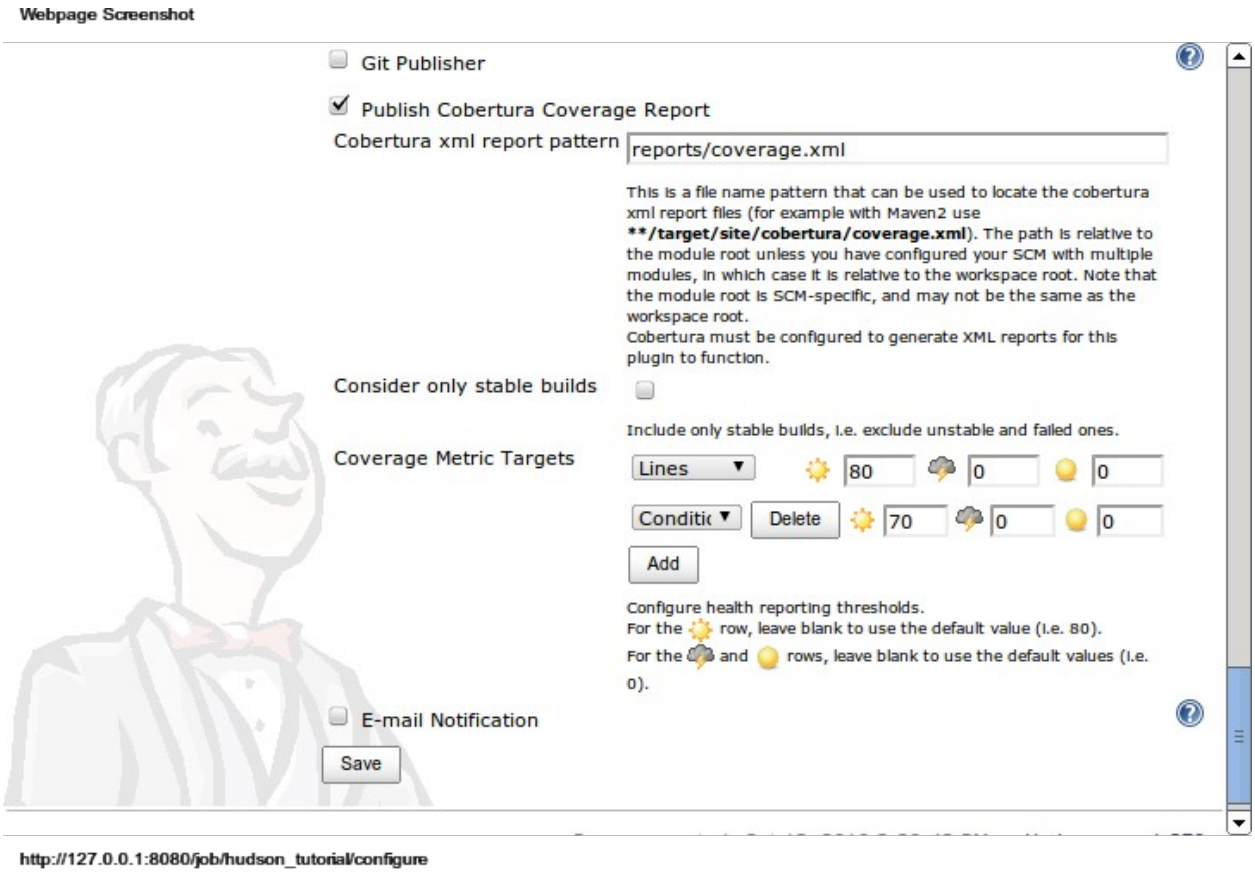
Configure locations of violations reports:

Webpage Screenshot



http://build.fairdevteam.com/job/django-jenkins-tutorial/configure

Test coverage reports are in `reports/coverage.xml`



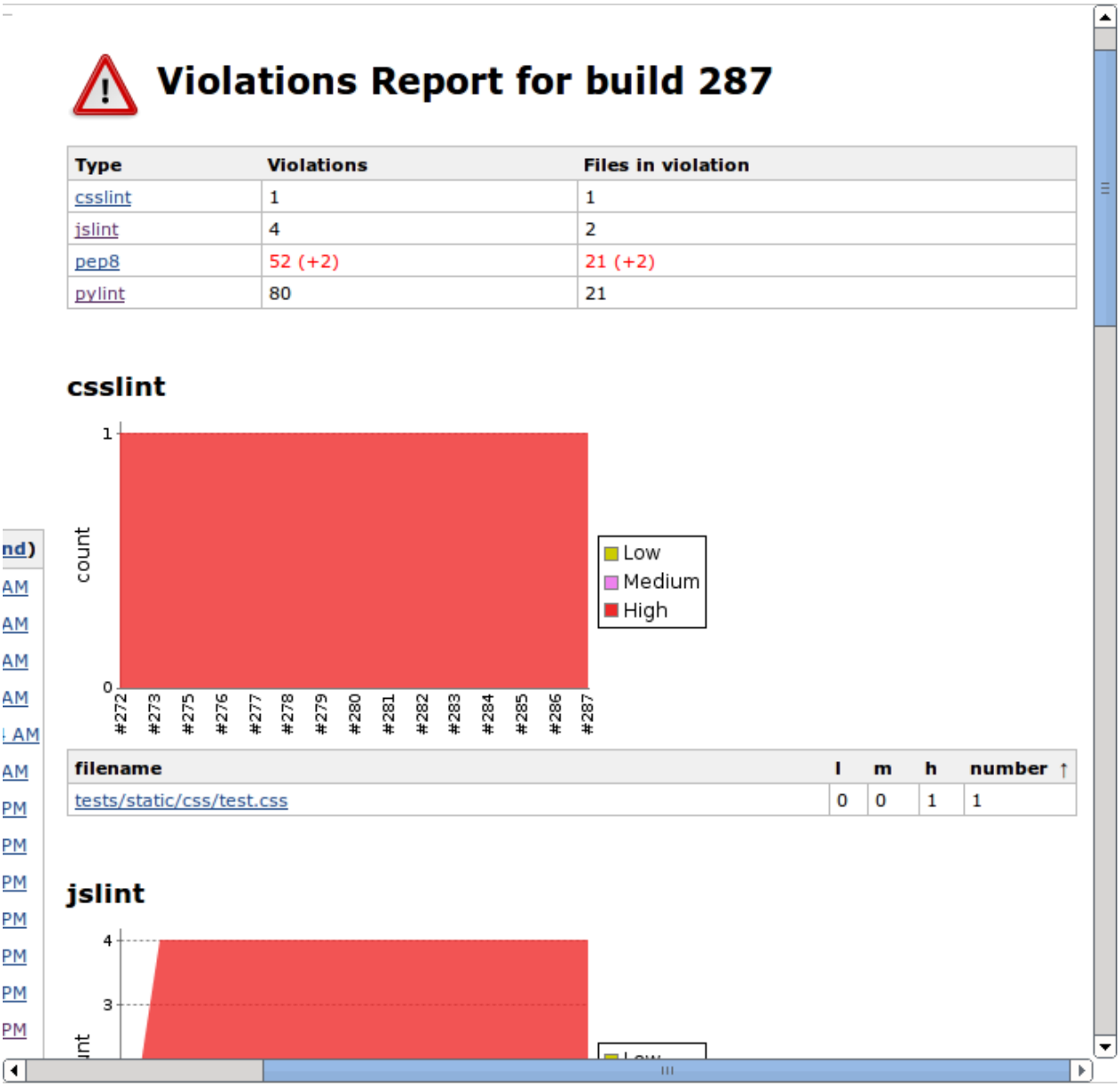
That's all!

Results

Press `Build Now` and see what you've got:

`Pylint` and other lint tools reports for each builds, shows what warning/errors are new for each build.

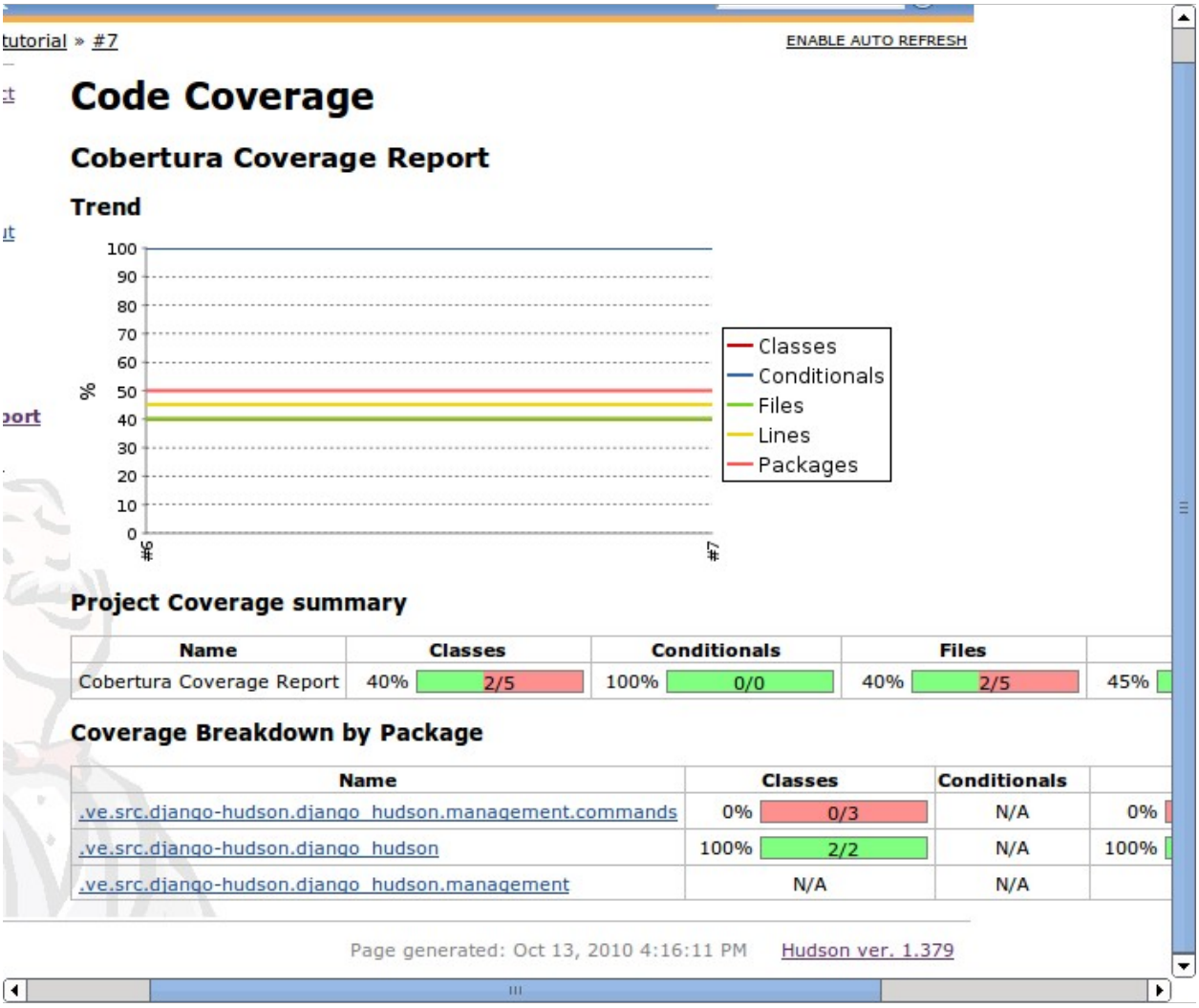
Webpage Screenshot



http://build.fairdevteam.com/job/django_jenkins_python26/violations/#pylint

Nice code coverage reports, showing the total coverage, and colored file listing.

Webpage Screenshot



http://127.0.0.1:8080/job/hudson_tutorial/7/cobertura/?

And of course the test results and output.

Webpage Screenshot

Jenkins search mpodgurskiy | log out

Jenkins » django_jenkins_python26 » #169 » Test Results ENABLE AUTO REFRESH

Back to Project
Status
Changes
Console Output
Edit Build Information
History
Polling Log
Git Build Data
Test Result
Coverage Report
Violations
SLOCCount
Score card
Previous Build
Next Build

Test Result

0 failures (±0)

115 tests (±0)
Took 5 min 52 sec.
add description

All Tests

Package	Duration	Fail	(diff)	Skip	(diff)	Total	(diff)
django.contrib.sessions.tests	5 min 40 sec	0		0		111	
django_jenkins.tests	3 sec	0		0		1	
test_app.tests	3 sec	0		0		1	
test_app.wmtests	6.1 sec	0		0		2	

Help us localize this page Page generated: Jan 24, 2012 7:40:19 AM Jenkins ver. 1.442

http://build.fairdevteam.com/job/django_jenkins_python26/169/testReport/