Cricket Documentation

Release 0.2.4.dev20140401003534

Russell Keith-Magee

March 31, 2014

Contents

Cricket is part of the BeeWare suite. The project website is http://pybee.org/cricket.

Cricket is a graphical tool that helps you run your test suites.

Normal unittest test runners dump all output to the console, and provide very little detail while the suite is running. As a result:

- You can't start looking at failures until the test suite has completed running,
- It isn't a very accessible format for identifying patterns in test failures,
- It can be hard (or cumbersome) to re-run any tests that have failed.

Why the name `cricket`? Test Cricket is the most prestigious version of the game of cricket. Games last for up to 5 days... just like running some test suites. The usual approach for making cricket watchable is a generous dose of beer; in programming, Balmer Peak limits come into effect, so something else is required...

CHAPTER 1

Quickstart

At present, Cricket has support for:

- Pre-Django 1.6 project test suites,
- Django 1.6+ project test suites using unittest2-style discovery, and
- unittest project test suites.

In your Django project, install cricket, and then run it:

```
$ pip install cricket
$ cricket-django
```

`cricket-django` will also work in Django's own tests directory – i.e., you can use `cricket-django` to run Django's own test suite (for Django 1.6 or later).

In a unittest project, install cricket, and then run it:

```
$ pip install cricket
$ cricket-unittest
```

This will pop up a GUI window. Hit "Run all", and watch your test suite execute.

1.1 Problems under Ubuntu

Ubuntu's packaging of Python omits the `idlelib` library from it's base package. If you're using Python 2.7 on Ubuntu 13.04, you can install `idlelib` by running:

```
$ sudo apt-get install idle-python2.7
```

For other versions of Python and Ubuntu, you'll need to adjust this as appropriate.

1.2 Problems under Windows

If you're running Cricket in a virtualenv, you'll need to set an environment variable so that Cricket can find the TCL graphics library:

```
$ set TCL_LIBRARY=c:\Python27\tcl\tcl8.5
```

You'll need to adjust the exact path to reflect your local Python install. You may find it helpful to put this line in the `activate.bat` script for your virtual environment so that it is automatically set whenever the virtualenv is activated.

Contents:

1.2.1 Contributing to Cricket

If you experience problems with cricket, log them on GitHub. If you want to contribute code, please fork the code and submit a pull request.

Setting up your development environment

The recommended way of setting up your development envrionment for `cricket` is to install a virtual environment, install the required dependencies and start coding. Assuming that you are using `virtualenvwrapper`, you only have to run:

```
$ git clone git@github.com:pybee/cricket.git
$ cd cricket
$ mkvirtualenv cricket
```

Cricket uses `unittest` (or `unittest2` for Python < 2.7) for its own test suite as well as additional helper modules for testing. To install all the requirements for cricket, you have to run the following commands within your virtual environment:

```
$ pip install -e .
$ pip install -r requirements_dev.txt
```

In case you are running a python version `< 2.7` please use the `requirements_dev_python2.7.txt` instead because `unittest2` is not part of the standard library for these version.

Now you are ready to start hacking! Have fun!

1.2.2 Writing a Cricket backend

Why you might want to do this

A number of test execution environments are not necessarily supported. This includes pytest, GUI test tools, or even custom stuff. The sky is the limit. Or, you might just want to understand the architecture.

Helicopter Overview of the Architecture

The main directory consists of events, executor, model, pipes, view and widgets. The ones which are the concern of the GUI are events, view and widgets. The ones which concern the backend are model, executor and pipes. The one which you need to really understand is pipes, but that's not the best starting point.

The best starting point is either the unittest or django subdirectory. The GUI is first built by the relevant backend, and the backend provides standard callbacks for the GUI.

Layout of a Backend System

A Cricket backend should contain the following 4 files:

- `__main__.py` - The entry point for the user.
- `discoverer.py` - Generates the list of available tests
- `executor.py` - Wraps execution of test functions
- `model.py` - Defines the method for executing the discoverer and executor

Requirements of a backend

Both the Django and the unittest backend take advantage of the unittest module to create and execute test suites. The core file pipes.PipedTestRunner will run unittest-style tests and provide the appropriately well-formed output expected by the GUI. However, it is a valid choice for the executor to produce output of the same for onto stdout itself. The only hard requirement is that the executor function stream onto stdout a series of well-formed outputs. To understand the full detail, examine pipes.py.

The Django and the unittest mechanisms for executing tests are different. The Django backend is a thin hook into the Django test execution machinery. The unittest backend is a slightly less thin hook into the unittest modele. The key requirements of the executor backend are:

1. The ability to stream well-formed output to stdout
2. The ability to limit/target test execution according to supplied labels

At the time of writing, sys.argv[1:] will be the list of dotted-namespaced names of tests which should be run. More complex command-line calls are simply not supported at this stage. A very useful task would be to do some more thinking on this interface.

1.2.3 Cricket Roadmap

Cricket is a new project - we have lots of things that we'd like to do. If you'd like to contribute, providing a patch for one of these features:

- Use a standard protocol (e.g., subunit) for communicating between the executor and the GUI
- Port to Python 3
- Add a pytest backend
- Add a nose backend
- Add a selenium backend, including possibly adding the ability to collect and store screenshots at the end of each test
- Improve GUI interface, including:
  - keyboard shortcuts
  - search
- Integrate with coverage, and use tkreadonly to display coverage stats overlaid on actual code
- Add historical tracking of test results
- Integrate with testr
- Add continuous background testing based on file modifications

1.2.4 Release History

0.2.4 - In development

0.2.3 - September 26, 2013

- Added ability to generate coverage
- Integration with Duvet

0.2.2 - September 17, 2013

- Corrected a problem with starting unittest2 projects.
- Corrected a error raised when unittest2 was manually installed in PYTHONPATH.

0.2.1 - September 14, 2013

- Fixed selection of test modules in unittest2-style suite discovery.
- Added ability to run Django's own test suite.

0.2.0 - August 31, 2013

- Relaunch as a PyBee project.
- Added test and help menus.

0.1.3 - July 29, 2013

- Corrected problem with Django test executor, masked by pyc caching.
- Improved collection of errors raised during test startup.

0.1.2 - July 26, 2013

- Improved handling of button state during test execution.

0.1.1 - July 9, 2013

Release incorporating updates from the PyCon AU 2013 sprints, including:

- Unittest2 support
- Improved handling of errors raised during test execution
- Improved reporting of errors caused on Ubuntu

0.1.0 - June 21, 2013

Initial public release, at PyCon AU 2013

CHAPTER 2

Indices and tables

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