sneeze Documentation

Release 0.0.1

Silas Ray

February 01, 2016

Contents

Sneeze is a pluggable plugin for the nose unit testing framework that logs test activity to a database at test runtime. It also exposes an interface that allows extension of the DB model and an API modeled on the nose plugin API to extend behavior. Finally, it adds a number of command line options to nosetests to configure the database to connect to and to rerun tests from identifiers from the database. It is intended primarily as a base for additional plugins.

CHAPTER 1

Installation and Quickstart

First, install Sneeze (`pip install nose-sneeze`), then run any tests with nose using nosetests from the command line as you would normally, but with the following command line arguments:

- *`--reporting-db-config`* An SQLAlchemy formatted connection string (you can use an SQLite database without any additional setup provided you have a standard Python install)
- *`--test-cycle-name`* A short identifying string for the *Test Cycle* being run
- *`--test-cycle-description`* A longer description of the *Test Cycle* being run

*Test Cycle*s are just collections of test results; they are not unique by name or description, and can contain one or more results for each of any number of tests. To report tests in an existing *Test Cycle*, use *`--test-cycle-id`* with the *Test Cycle* id from the database of the *Test Cycle* you wish to add the results of the tests being run to. Either the *Test Cycle* name/description pair or the id must be provided; if an id is provided, it will always supersede any other *Test Cycle* configurations. The database config can also be stored more permanently by setting it on an environment variable named `sneeze_db_config`.

Be aware that do to nose being in maintenance mode, Sneeze relies on a custom version of nose, nose-for-sneeze. It should work fine with normal nose as long as you don't attempt to use the multiprocess plugin. Links to relevant fork and pull request here:

- nose with worker exit hook
- pull request

CHAPTER 2

Repo

https://github.com/NYTimes/sneeze

CHAPTER 3

Details

3.1 Sneeze

The Sneeze class is the plugin for nose. It provides the core command line options for controlling Sneeze behavior for a given nosetests execution. Providing the *--reporting-db-config* argument to nosetests enables Sneeze (and any of its enabled plugins) for the given nosetests execution. The user must also provide either *--test-cycle-name* and *--test-cycle-description*, or a valid *--test-cycle-id* for the tests to execute successfully, once Sneeze is enabled.

The plugin starts the Tissue and connects the Sneeze API hooks to the nose plugin API. It also loads and initializes the plugin managers for any Sneeze plugins being used, and attaches them to the Tissue.

3.1.1 Options

--reporting-db-config=CONFIG_STRING
SQLAlchemy formated connection string for reporting database.

--test-cycle-name=NAME
Name of the test cycle being run.

--test-cycle-description=DESCRIPTION
Description for the test cycle being run.

--test-cycle-id=CYCLE_ID
id of test cycle to run tests under. Overrides *--test-cycle-name* and *--test-cycle-description*.

--rerun-from-case-execution=EXECUTION_ID_LIST
Case execution id to base rerun upon.

--pocket-change-host=HOST
url of associated pocket change server.

--pocket-change-username=USERNAME
username for pocket change user.

--pocket-change-password=PASSWORD
password for pocket change user.

--pocket-change-token=TOKEN
token for pocket change user.

--pocket-change-environment-envvar=ENVIRONMENT_VAR_NAME
Name of environment variable to record as the test environment value.

3.1.2 Plugin

class sneeze.nose_interface.**Sneeze**
Bases: nose.plugins.base.Plugin

3.1.3 Source

3.2 Plugins

The behavior of Sneeze can be modified and extended by implementing plugins.

3.2.1 Entrypoints

The Sneeze plugin API utilizes 3 distutils entrypoints to add functionality to the system. They share a root nose.plugins.sneeze.plugins.

add_options

nose.plugins.sneeze.plugins.add_options allows a plugin to add new command line arguments to nosetests. It should point at a callable that expects an options parser and an environment string; return values are ignored.

managers

nose.plugins.sneeze.plugins.managers allows a plugin to define a *Plugin Manager* to execute the behavior of the plugin during test exection. It should point to a class that utilizes the Sneeze plugin API.

add_models

nose.plugins.sneeze.plugins.add_models allows a plugin to extend the DB schema by adding new models to the ORM. It should point to a callable that expects a SQLAlchemy declarative base object. It should return a dictionary containing key value pairs where the key is a string of the model name and the value is the model itself for all "public" models implemented.

3.2.2 API

The Sneeze API generally assumes that *Plugin Manager*s will use the Tissue instance from initialization to retrieve any needed state, so the calls generally avoid passing any information as arguments that could be obtained through the Tissue.

enter_test_cycle()
Called once, after the Tissue has been initialized, before any *Case Execution*s are started.

before_enter_case()
Called before entering each case. Note that this is called when entering a *Default Case* as well.

after_case(*case*, *description*)
Called after a case has been entered, but before it is run. Receives a *Case* object for case and a *string* description.

handle_pass()
Called from nose API addSuccess(), when a test has passed.

handle_fail(*error*)
Called from nose API addFailure() and addError() in cases where the test errored (generally, raised an uncaught exception). Receives a *string* as error.

handle_skip(*error*)
Called from nose API addError() in cases where the test was skipped or is deprecated, receives a *string* as error containing any message from the skip or deprecation exception.

before_exit_case(*result*)
Called before a *Case Execution* result is recorded, after it has completed and :func:handle_failor :func:handle_pass have been called. Receives a *string* as result. Note that this is not called for *Case Executions* of the *Default Case*.

after_exit_case(*result*)
Called after the result has been recorded for a *Case Execution*, but before entering an execution of the *Default Case*. Note that this is not called for *Case Executions* of the *Default Case*.

exit_test_cycle()
Called when the Tissue exits, after all tests in the executor have been completed and recorded.

3.3 Glossary

Case Execution The record of a specific run of a *Test Case*. The execution has a description, start time, end time, result, and a reference to the *Test Case* that it is for. Linked through the test_cycle_test_case_execution table to one or more *Test Cycle*s. Case Executions also have meta information attached to them through *Execution Batch*s. Stored in the test_case_execution table, represented by the CaseExecution model.

Default Case Each *Execution Batch* has a default *Test Case* associated with it. This case is used to generate *Case Execution* records that act as the current *Test Case* while the executor is outside of a test scope. This is primarily provided so that plugins have an entity to associate data to (such as log messages) when a test is not currently executing.

Execution Batch A collection of meta information about a set of tests run together in a single execution process. Provides data on batch start and end time, execution host, environment, and nosetests command line options. Stored in the execution_batch table, represented by the ExecutionBatch model.

Plugin Manager An object registered via the managers entrypoint that implements Sneeze plugin runtime behaviors via the plugin API.

Test Case A particular set of operations and validations defined by a nose test. Test Cases have a label and id. Stored in the test_case table, represented by the Case model.

Test Cycle A collection of *Case Execution*s with a name, label, and id. Stored in the test_cycle table, represented by the TestCycle model.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

S

Symbols

A

B

C

D

E

H

P

S

T