nosedbreport Documentation

Release 0.3.0

Ali-Akber Saifee

August 04, 2015

Contents

CHAPTER 1

Introduction

nosedbreport exposes a single plugin that can front various backend databases to store the result of a nose test execution. Having the results of your tests, whether they are part of a continuous integration system or not, allows you to ask interesting questions about your project such as

- What were the test suites that ran in the last five minutes
- What is the average time to run test case 'x'
- What is the standard time to failure for test suite 'y'
- and so on...

These questions also allow you to build reporting, and monitoring tools based on automated functional tests that you may be running against your development, staging or production systems, such as heartbeat or availability pages.

CHAPTER 2

Installation

- with easy_install

```
sudo easy_install nosedbreport
```

- with pip

```
sudo pip install nosedbreport
```

- from source (git repository):

```
hg clone http://github.com/alisaifee/nosedbreport
cd nosedbreport
python setup.py build
sudo python setup.py install
```

CHAPTER 3

Usage

- The most basic use case is to report the results of a test run into a mysql database, which can be achieved by adding the following options to your nose execution:

```
nosetests --dbreport-dbtype=mysql --dbreport-host=your.mysql.com\
 --dbreport-username=ali --dbreport-password=some-pass --dbreport-db=nosereport
```

- To create the appropriate schema in your mysql database:

```
nosetests --dbreport-dbtype=mysql --dbreport-host=your.mysql.com\
 --dbreport-username=root  --dbreport-password=your-root-pass\
 --dbreport-db=nosereport --dbreport-create-schema
```

- For detailed usage refer to read the docs

3.1 Development

3.1.1 Contributing

The source is maintained in a bitbucket repository. Feel free to fork it to modify/extend the plugin. Currently, the plugin is only backed by a MySQL & SQLite connector, but it can be easily extended to support other databases.

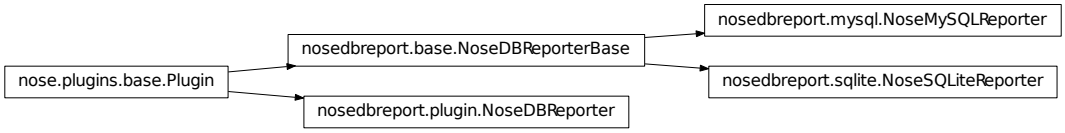
To add a new database connector, you will need to:

- add a new class that extends *NoseDBReporterBase*
- implement the `configure()`, `startTest()`, `report()` and `construct_schema()` methods. (For an example see the MySQL implementation in *NoseMySQLReporter*)
- add your new class to the *connectors* in *plugin* in the following way:

```
import newconnector
connectors = {
              "newconnector" : newconnector.NoseNewConnectorReporter,
              "mysql" : mysql.NoseMySQLReporter
            }
```

- this will make the newconnector available with the command line option –dbreport_dbtype=newconnector

3.1.2 Class Structure



3.1.3 Source Documentation

class `nosedbreport.plugin.`**NoseDBReporter**
The main plugin that is loaded by `nose.plugin.PluginManager`

`configure`(*options*, *conf*)
Configure plugin. Plugin is disabled by default.

`connectors` **= {'sqlite': <class 'nosedbreport.sqlite.NoseSQLiteReporter'>, 'mysql': <class 'nosedbreport.mysql.NoseM**
list of db connectors available for use when specifying db_type.

`options`(*parser*, *env*)
Register commandline options

class `nosedbreport.base.`**NoseDBReporterBase**
Base class for Nose plugins that stash test results into a database.

`addError`(*test*, *err*, *capt=None*)
sets the status of the `test` to either 'skipped' or 'error', collects the trace and time taken to execute.

`addFailure`(*test*, *err*, *capt=None*, *tb_info=None*)
sets the status of the `test` to 'fail', collects the trace and time taken to execute.

`addSuccess`(*test*, *capt=None*)
sets the status of the `test` to 'pass', and sets the time taken to execute.

`get_full_doc`(*test*)
via various nasty inspection methods, return the full docstring of the `test` being executed now.

`startTest`(*test*)
collect information about a `test` before it begins, and initialize a timer to record time taken.

`test_case_results` **= None**
dictionary to keep track of individual test case executions, including status, time taken and tracebacks.

`test_suites` **= None**
dictionary to keep track of the overall suite results

class `nosedbreport.mysql.`**NoseMySQLReporter**
MySQL Connector. Reports the results of each test run into the tables `testcase`, `testsuite`, `testcaseexecution` and `testsuiteexecution`

`configure`(*options*, *conf*)
sets up the MySQL database connection based on the options provided on the command line.

`construct_schema`()
called when the *–dbreport_create_schema* command option is passed to the plugin to create the mysql table schema.

report(*stream*)
After successful completion of a nose run, perform the final reporting of the test results to the MySQL database.

startTest(*test*)
record initiation of a test case. Update the last start time of the test suite & test case.

class nosedbreport.sqlite.**NoseSQLiteReporter**
SQLLite Connector. Reports the results of each test run into the tables `testcase`, `testsuite`,‘‘testcaseexecution‘‘ and `testsuiteexecution`

configure(*options*, *conf*)
sets up the sqlite database connection

construct_schema()
called when the *–dbreport_create_schema* command option is passed to the plugin to create the sqlite table schema.

report(*stream*)
After successful completion of a nose run, perform the final reporting of the test results to the sqlite database.

startTest(*test*)
record initiation of a test case (`test`). Update the last start time of the test suite & test case.

3.2 Data Structures

3.2.1 Overview

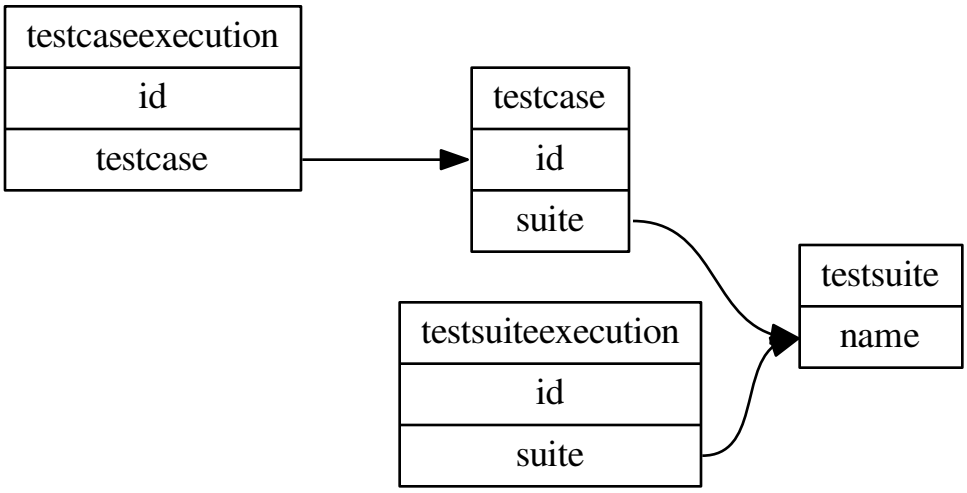
By extending *NoseDBReporterBase* you essentially have access to two relevant dictionaries, *test_suites* and *test_case_results*.

3.2.2 MySQL Example

The *NoseMySQLReporter* backend translates the data structures of the plugin into mysql tables using 4 tables:

- testsuite
- testsuiteexecution
- testcase
- testcaseexecution

The MySQL tables are roughly related as follows:


testcaseexecution
id
testcase
testcase
id
suite
testsuite
name
testsuiteexecution
id
suite

n

A

C

G

N

O

R

S

T