PyMySQL Documentation

Release 0.7.2

Yutaka Matsubara and GitHub contributors

Mar 22, 2017

Contents

1 User Guide **1**
- 1.1 Installation 1
- 1.2 Examples 1
- 1.3 Resources 2
- 1.4 Development 2

2 API Reference **5**
- 2.1 `pymysql.connections` 5
- 2.2 `pymysql.cursors` 5

3 Indices and tables **7**

Python Module Index **9**

CHAPTER 1

User Guide

The PyMySQL user guide explains how to install PyMySQL and how to contribute to the library as a developer.

Installation

The last stable release is available on PyPI and can be installed with `pip`:

```
$ pip install PyMySQL
```

Requirements

- Python – one of the following:
  - CPython >= 2.6 or >= 3.3
  - PyPy >= 4.0
  - IronPython 2.7
- MySQL Server – one of the following:
  - MySQL >= 4.1 (tested with only 5.5~)
  - MariaDB >= 5.1

Examples

CRUD

The following examples make use of a simple table

```
CREATE TABLE `users` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `email` varchar(255) COLLATE utf8_bin NOT NULL,
    `password` varchar(255) COLLATE utf8_bin NOT NULL,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin
AUTO_INCREMENT=1 ;
```

```
import pymysql.cursors

# Connect to the database
connection = pymysql.connect(host='localhost',
                             user='user',
                             password='passwd',
                             db='db',
                             charset='utf8mb4',
                             cursorclass=pymysql.cursors.DictCursor)

try:
    with connection.cursor() as cursor:
        # Create a new record
        sql = "INSERT INTO `users` (`email`, `password`) VALUES (%s, %s)"
        cursor.execute(sql, ('webmaster@python.org', 'very-secret'))

    # connection is not autocommit by default. So you must commit to save
    # your changes.
    connection.commit()

    with connection.cursor() as cursor:
        # Read a single record
        sql = "SELECT `id`, `password` FROM `users` WHERE `email`=%s"
        cursor.execute(sql, ('webmaster@python.org',))
        result = cursor.fetchone()
        print(result)
finally:
    connection.close()
```

This example will print:

```
{'password': 'very-secret', 'id': 1}
```

Resources

DB-API 2.0: http://www.python.org/dev/peps/pep-0249

MySQL Reference Manuals: http://dev.mysql.com/doc/

MySQL client/server protocol: http://dev.mysql.com/doc/internals/en/client-server-protocol.html

PyMySQL mailing list: https://groups.google.com/forum/#!forum/pymysql-users

Development

You can help developing PyMySQL by contributing on GitHub.

Building the documentation

Go to the `docs` directory and run `make html`.

Test Suite

If you would like to run the test suite, create a database for testing like this:

```
mysql -e 'create database test_pymysql  DEFAULT CHARACTER SET utf8 DEFAULT COLLATE␣
↪utf8_general_ci;'
mysql -e 'create database test_pymysql2 DEFAULT CHARACTER SET utf8 DEFAULT COLLATE␣
↪utf8_general_ci;'
```

Then, copy the file `.travis.databases.json` to `pymysql/tests/databases.json` and edit the new file to match your MySQL configuration:

```
$ cp .travis.databases.json pymysql/tests/databases.json
$ $EDITOR pymysql/tests/databases.json
```

To run all the tests, execute the script `runtests.py`:

```
$ python runtests.py
```

A `tox.ini` file is also provided for conveniently running tests on multiple Python versions:

```
$ tox
```

CHAPTER 2

API Reference

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

`pymysql.connections`

class `pymysql.connections.`**`Connection`**(*host=None*, *user=None*, *password=''*, *database=None*, *port=0*, *unix_socket=None*, *charset=''*, *sql_mode=None*, *read_default_file=None*, *conv=None*, *use_unicode=None*, *client_flag=0*, *cursorclass=<class 'pymysql.cursors.Cursor'>*, *init_command=None*, *connect_timeout=10*, *ssl=None*, *read_default_group=None*, *compress=None*, *named_pipe=None*, *no_delay=None*, *autocommit=False*, *db=None*, *passwd=None*, *local_infile=False*, *max_allowed_packet=16777216*, *defer_connect=False*, *auth_plugin_map={}*, *read_timeout=None*, *write_timeout=None*, *bind_address=None*)

Representation of a socket with a mysql server.

The proper way to get an instance of this class is to call connect().

`pymysql.cursors`

class `pymysql.cursors.`**`Cursor`**(*connection*)

This is the object you use to interact with the database.

class `pymysql.cursors.`**`SSCursor`**(*connection*)

Unbuffered Cursor, mainly useful for queries that return a lot of data, or for connections to remote servers over a slow network.

Instead of copying every row of data into a buffer, this will fetch rows as needed. The upside of this is the client uses much less memory, and rows are returned much faster when traveling over a slow network or if the result set is very big.

There are limitations, though. The MySQL protocol doesn't support returning the total number of rows, so the only way to tell how many rows there are is to iterate over every row returned. Also, it currently isn't possible to scroll backwards, as only the current row is held in memory.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

pymysql.connections, 5

pymysql.cursors, 5

Index

C

Connection (class in pymysql.connections), 5
Cursor (class in pymysql.cursors), 5

P

pymysql.connections (module), 5
pymysql.cursors (module), 5

S

SSCursor (class in pymysql.cursors), 5