25

The Python DB-API

DB-API is a single API for supporting database-independent database access. The MySQL implementation of this API, MySQLdb, can be downloaded from http://dustman.net/andy/python/MySQLdb. It comes with a Redhat RPM Linux installer, a Win32 installer, and a Python script for other platforms. For the "other platforms":

1. Uncompress the .tar.gz file that contains MySQLdb using the commands *gunzip FILENAME.tar.gz* and *tar xf FILENAME.tar*.
2. Change directories into the newly generated MySQLdb directory.
3. Issue the command: *python setup.py install*

The MySQLdb module contains both the standard DB-API methods and attributes as well as several proprietary methods and attributes. Proprietary APIs are marked with asterisks.

Module: MySQLdb

The entry point into the MySQL module is via the `MySQLdb.connect()` method. The return value from this method represents a connection to a MySQL database that you can use for all of your MySQL operations.

Module Attributes

Attribute: apilevel

Synopsis

A string constant storing the version of the DB-API that MySQLdb supports.

Attribute: paramstyle

Synopsis

Defines the type of parameter placeholder in parameterized queries. DB-API supports many valid values for this attribute, but MySQLdb actually supports only "format" and "pyformat". This attribute is largely meaningless to MySQL developers.

Attribute: quote_conv*

Synopsis

Maps Python types to MySQL literals via a dictionary mapping.

Attribute: threadsafety

Synopsis

Specifies the level of thread-safety supported by MySQLdb. Possible values are:

0 - Threads may no share the module

1 - Threads may share the module, but not the connections

2 - Threads may share the module and connections

3 - Threads may share the module, connections, and cursors

Attribute: type_conv*

Synopsis

Maps MySQL types from strings to the desired mapping type using. This value is initialized with the following values:

```
{ FIELD_TYPE.TINY : int,
FIELD_TYPE.SHORT: int,
FIELD_TYPE.LONG: long,
FIELD_TYPE.FLOAT: float,
FIELD_TYPE.DOUBLE: float,
FIELD_TYPE.LONGLONG: long,
FIELD_TYPE.INT24: int,
FIELD_TYPE.YEAR: int }
```

Module Methods

Method: MySQL.connect( )

Signature

```
connection = MySQL.connect(params)
```

Synopsis

Connects to the MySQL database engine represented by the various connection keyword/value parameters. These parameters include:

host

The name of the server on which the MySQL database is running

user

The user ID to use for connecting to MySQL. This user should be allowed by MySQL to make the connection.

passwd

The password to authenticate the user ID for the connection.

db

The MySQL database to which the application is attempting to connect.

port

Directs MySQLdb to connect to a MySQL installation on a custom part. When left unspecified, the method will use the default MySQL port of 3306.

unix_socket

Identifies the location of a socket or named pipe to use if the value of the host allows it.

client_flags

An integer specifying the client connection flags to use. These client connection flags are the same ones enumerated in Chapter 22, *C API* for the `mysql_real_connect()` method.

This method returns a Python object representing a connection to a MySQL database.

Example

```
connection = MySQLdb.connect(host='carthage', user='test',
                             passwd='test', db='test');
```

Connection Attributes

Attribute: db*

Synopsis

A window into the MySQL C API. MySQLdb uses this attribute to make calls to the underlying C API.

Connection Methods

Method: close( )

Signature

```
close()
```

Synopsis

Closes the current connection to the database and releases any associated resources.

Example

```
connection = MySQLdb.connect(host='carthage', user='test',
                             passwd='test', db='test');
connection.close();
```

Method: commit( )

Signature

```
commit()
```

Synopsis

Commits the current transaction by sending a COMMIT to MySQL.

Example

```
connection = MySQLdb.connect(host='carthage', user='test',
                             passwd='test', db='test');
connection._transactional = 1;
cursor = connection.cursor();
cursor.execute("UPDATE TNAME SET COL = 1 WHERE PK = 2045");
cursor.execute("UPDATE TNAME SET COL = 1 WHERE PK = 3200");
connection.commit();
connection.close();
```

Method: cursor( )

Signature

```
cursor = cursor()
```

Synopsis

Creates a cursor associated with this connection. Transactions involving any statements executed by the newly created cursor are governed by this connection.

Example

```
connection = MySQLdb.connect(host='carthage', user='test',
                             passwd='test', db='test');
cursor = connection.cursor();
cursor.execute("UPDATE TNAME SET COL = 1 WHERE PK = 2045");
connection.close();
```

Method: rollback( )

Signature

```
rollback()
```

Synopsis

Rollsback any uncommitted statements. This only works if MySQL is setup for transactional processing in this context.

Example

```
connection = MySQLdb.connect(host='carthage', user='test',
                              passwd='test', db='test');
connection._transactional = 1;
cursor = connection.cursor();
cursor.execute("UPDATE TNAME SET COL = 1 WHERE PK = 2045");
try:
    cursor.execute("UPDATE TNAME SET COL = 1 WHERE PK = 3200");
    connection.commit();
except:
    connection.rollback();
connection.close();
```

Cursor Attributes

Attribute: arraysize

Synopsis

Specifies the number of rows to fetch at a time with the `fetchmany()` method call. By default, this value is set to 1. In other words, `fetchmany()` fetches one row at a time by default.

Attribute: description

Synopsis

Describes a result column as a read-only sequence of seven-item sequences. Each sequence contains the following values: `name`, `type_code`, `display_size`, `internal_size`, `precision`, `scale`, `null_ok`.

Attribute: rowcount

Synopsis

Provides the number of rows returned through the last `executeXXX()` call. This attribute is naturally read-only and has a value of -1 when no `executeXXX()` call has been made or the last operation does not provide a row count.

Cursor Methods

Method: callproc()

Signature

```
callproc(procname [,parameters])
```

Synopsis

This method is not supported by MySQL.

Method: close()

Signature

```
close()
```

Synopsis

Closes the cursor explicitly. Once closed, a cursor will throw an `ProgrammingError` will be thrown if any operation is attempted on the cursor.

Example

```
cursor = connection.cursor();
cursor.close();
```

Method: execute()

Signature

```
cursor = execute(sql [,parameters])
```

Synopsis

Sends arbitrary SQL to MySQL for execution. If the SQL specified is parameterized, then the optional second argument is a sequence or mapping containing parameter values for the SQL. Any results or other information generated by the SQL can then be accessed through the cursor.

The parameters of this method may also be lists of tuples to enable you to perform multiple operations at once. This usage is considered depricated as of the DB-API 2.0 specification. You should instead use the `executemany()` method.

Example

```
connection = MySQLdb.connect(host='carthage', user='test',
                             passwd='test', db='test');
cursor = connection.cursor();
cursor.execute('SELECT * FROM TNAME');
```

Method: executemany()

Signature

```
cursor.executemany(sql,parameters)
```

Synopsis

Prepares a SQL statement and sends it to MySQL for execution against all parameter sequences or mappings in the `parameters` sequence.

Example

```
connection = MySQLdb.connect(host='carthage', user='test',
                             passwd='test', db='test');
cursor = connection.cursor();
cursor.executemany("INSERT INTO COLOR ( COLOR, ABBREV ) VALUES (%s, %s )",
                   (("BLUE", "BL"), ("PURPLE", "PPL"), ("ORANGE", "ORN")));
```

Method: fetchall()

Signature

```
rows = cursor.fetchmany()
```

Synopsis

Fetches all remaining rows of a query result as a sequence of sequences.

Example

```
connection = MySQLdb.connect(host='carthage', user='test',
                             passwd='test', db='test');
cursor = connection.cursor();
cursor.execute("SELECT * FROM TNAME");
for row in cursor.fetchall():
        # process row
```

Method: fetchmany()

Signature

rows = cursor.fetchmany(*[size]*)

Synopsis

Fetches the next set of rows of a result set as a sequence of sequences. If no more rows are available, this method returns an empty sequence.

If specified, the `size` parameter dictates how many rows should be fetched. The default value for this parameter is the cursor's `arraysize` value. If the `size` parameter is larger than the number of rows left, then the resulting sequence will contain all remaining rows.

Example

```
connection = MySQLdb.connect(host='carthage', user='test',
                             passwd='test', db='test');
cursor = connection.cursor();
cursor.execute("SELECT * FROM TNAME");
rows = cursor.fetchmany(5);
```

Method: fetchone()

Signature

row = cursor.fetchone()

Synopsis

Fetches the next row of a result set returned by a query as a single sequence. This method will return `None` when no more results exist. It will throw an error should the SQL executed not be a query.

Example

```
connection = MySQLdb.connect(host='carthage', user='test',
                             passwd='test', db='test');
cursor = connection.cursor();
cursor.execute("SELECT * FROM TNAME");
row = cursor.fetchone();
print "Key: ", row[0];
print "Value: ", row[1];
```

Method: insert_id()*

Signature

id = cursor.insert_id()

Synopsis

Returns the last inserted ID from the most recent `INSERT` on an `AUTO_INCREMENT` field.

Example

```
connection = MySQLdb.connect(host='carthage', user='test',
                             passwd='test', db='test');
cursor = connection.cursor();
cursor.execute("INSERT INTO TNAME (COL) VALUES (1)");
id = cursor.insert_id();
```

Method: nextset()

Signature

```
cursor.nextset()
```

Synopsis

This method always returns `None` for MySQL.

Method: setinputsizes()

Signature

```
cursor.setinputsizes(sizes)
```

Synopsis

This method does nothing in MySQL.

Method: setoutputsize()

Signature

```
cursor.setoutputsize(size [,column])
```

Synopsis

This method does nothing in MySQL.