

# Connecting MySQL and Python

O'Reilly MySQL Conference & Expo 2010 - Tue, 13 Apr 2010

Geert Vanderkelen

geert.vanderkelen@sun.com

## 1. Introduction

This handout is supporting the talk *Connecting MySQL and Python* which tries to give an overview of how to connect your *Python* application with *MySQL*. It introduces Python, goes over some MySQL features and dives into the possibilities to setup the connection between the two using drivers and popular frameworks.

## 2. About Python

Python is an object-oriented programming language which is easy to learn and easy to read. It comes with a large collection of build-in functionality but can easily be extended.

Python is used for scripting and web development, but it is also embed it into application or can for creation graphical games for example.

There should be one-- and preferably only one-- obvious way to do it.  
(Part of *The Zen of Python*)

### Software quality

Small example, following C code:

```
int i = 0;
while(i < 1000) {
    if(i < 10)
        foo();
    else {
        bar();
        foo();
    }
    spam();
    i++;
}
```

Lets take a look at how you would do the same in Python. Try to write the following differently (of course, you can add spaces):

```
i = 0
while i < 1000:
    if i < 10:
        foo()
    else:
        bar()
        foo()
    spam()
    i += 1
```

### Portable

Your Python scripts or applications will run on virtually all major platforms where Python is installed. Simply copying it over from a Linux distribution to a Windows machine should work.

### Object-Oriented

Python is an object-oriented language, but you don't have too. Just like C++, if you want to develop in more procedural or structural way: you can.



## All included, and extendible

The distribution of Python comes with a large collection of functionality known as the *standard library*. It includes most tools you need to get you going right away.

There are also lots of 3rd party modules available. The database interfaces for connecting to MySQL are examples.

## Alternative implementations

There are currently three primary implementations of the Python language: CPython, Jython and IronPython.

*CPython* is the most common and standard implementation of Python written using the ANSI C programming language. When you run CPython source it will translate the code to byte code and run it using the Python Virtual Machine (PVM).

CPython is the Python you would download from <http://python.org> and find on most major platforms pre-installed.



Script Java using the Python language.

*Jython* is completely written in and integrates with Java. It allows you to code Java using the Python language and run it through the Java Virtual Machine.

Jython makes it possible to script Java.

Similar to Jython, you have *IronPython* which integrates with *Microsoft's .Net* framework as well as *Mono*.

## Python v3.x

The next generation of Python, v3, is already available, but the adaption is a bit slow. Python v2 and v3 are not compatible and most, if not all code, has to be rewritten. This is quite some work for all 3rd party module developers. Luckily there

is a library and tool called 2to3 which makes easier.

## 3. Database Interfaces

Python defines the portable Database API v2.0 in PEP-249 (Python Enhancement Proposal). This protocol should be implemented by all modules which offer an interface to DBMS.

A script which works using MySQL should also work using Oracle by just replacing the database interface.

### MySQLdb

The most used database interface for MySQL is without doubt *MySQL for Python*, i.e. MySQLdb. It was developed and is maintained by Andy Dustman.

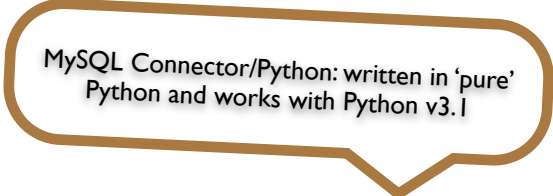
MySQLdb uses the C MySQL Client Libraries and usually needs MySQL installed on the box on which you are running your applications.

Binaries are downloadable for most platforms like Windows and Mac OS X.

Most Linux and BSD distributions also make it available through their packaging system.

### oursql

oursql is a recent development by the Python community. Just like MySQLdb it uses the C MySQL Client Libraries.



MySQL Connector/Python: written in 'pure' Python and works with Python v3.1

## 4. MySQL Connector/Python

MySQL Connector/Python is developed at Sun and implements the MySQL Client/Server Protocol completely in Python. This makes it extremely easy to install and start

coding right away. You don't even need MySQL installed on the machine unless of course you need a local test database.

## Releases

There is no 'stable' or 'GA' release yet, but we try to make regular development packages to make it easy to try it out.

MySQL Connector/Python is released under GPLv2 and can be downloaded from Launchpad (see Links).

## Packaging

Python v2.5 and later, and v3.1 are supported. Both editions come in the same package. and installation takes care to install the correct one.

Make sure you use the correct examples for your Python version: the v3.1 you'll find in the py3k/ subdirectory.

## 5. Things about MySQL

What is important in MySQL for the Connectors in general, and for Python in particular.

### Character set

The world is going unicode, and it might give you less headaches to do the same when you're coding in Python.

Most drivers support the `use_unicode` parameters and can set the character set. MySQL Connector/Python and oursql are for example defaulting to UTF-8.

### Default Storage Engine

It might be good to set the default storage engine to make sure tables are created correctly by frameworks. For example, you can set it to InnoDB so all your tables are transactional.

```
SET @@global.storage_engine = InnoDB
```

### SQL Mode

If you don't like MySQL to insert to long strings in to small columns; if you don't like invalid dates to be accepted; if you don't

want warnings, but instead would like an error: configure MySQL to use the TRADITIONAL SQL Mode.

```
SET @@session.SQL_MODE = TRADITIONAL
```

You can set it per session, or globally for each connection to the server.

## 6. Frameworks and ORMs

### Django

Django is a web application framework which makes creating database-driven websites a piece of cake. It comes with everything included and works great with MySQL.

It currently comes only with support for MySQLdb, but attempts to make it work with MySQL Connector/Python has been successful.

### TurboGears

Another great web application framework, TurboGears consist of several other projects such as *Pylons*, *Genshi* and *SQLAlchemy*.

### SQLAlchemy

SQLAlchemy is an SQL Toolkit and Object-Relational Mapper.

The latest v0.6 should work out of the box with MySQLdb, oursql and MySQL Connector/Python.

### Twisted

Twisted is a event-driven networking framework. It comes with 'adbapi' to elevate the problem of your application blocking when waiting for a result from the database. It offers an asynchronous mapping to the underlying database interfaces using Python's DB-API v2.0. Using this, you can define connection pools, send a query and let a callback handle the results when it is ready.



## 7. Examples

### Simply getting current date & time

```
#import MySQLdb as db
import mysql.connector as db
#import oursqldb as db

cnx = db.connect(user='root',db='test')
cur = cnx.cursor()
cur.execute("SELECT NOW()")
print(cur.fetchall())
cur.close()
cnx.close()
```

### A Warning turns into an Error

```
import mysql.connector as db

cnx = db.connect(user='root',db='test')
cur = cnx.cursor()
cur.execute("SET SQL_MODE = 'TRADITIONAL'")
cur.execute("CREATE TABLE t1 (c1 CHAR(5))")
s = "INSERT INTO t1 (c1) VALUES (%s)"

try:
    cur.execute(s, ('wrong length',))
except db.Error, e:
    print "Data is to long!"
cur.close()
cnx.close()
```

### Use format codes using execute()

```
cnx = db.connect(user='root',db='test')
cur = cnx.cursor()
ins = "INSERT INTO t1 (dayofbirth) VALUES (%s)"
cur.execute(ins, (datetime(1977,6,14),))
cnx.close()
```

## Jython using MySQL Connector/J

```
import java
from java.lang import Class
from java.sql import DriverManager

class PlayMySQL(java.lang.Object):

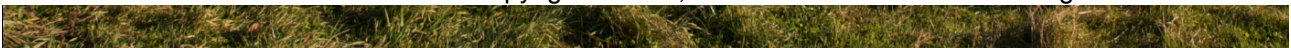
    def __init__(self):
        cs = "jdbc:mysql://%s/%s?user=%s" % (
            'localhost','test','root')
        try:
            c = DriverManager.getConnection(cs)
        except:
            raise
        self.conn = c

    def show_engines(self):
        try:
            stmt = self.conn.createStatement()
            rs = stmt.executeQuery("SHOW ENGINES")
            while rs.next():
                print rs.getString("Engine")
        except:
            raise

if __name__ == '__main__':
    drv = "com.mysql.jdbc.Driver"
    try:
        Class.forName(drv).newInstance();
    except:
        raise
    p = PlayMySQL()
```

## 8. Some links

- Cython  
<http://cython.org>
- Django  
<http://djangoproject.com>
- IronPython  
<http://ironpython.com>
- Jython  
<http://jython.org>
- MySQL  
<http://dev.mysql.com>
- MySQL Connector/Python  
<http://launchpad.net/myconnpy>
- MySQLdb  
<http://mysql-python.sourceforge.net/>
- oursqldb  
<https://launchpad.net/oursqldb>
- Python  
<http://python.org>
- SQLAlchemy  
<http://sqlalchemy.org/>



- TurboGears  
<http://turbogears.com/>
- Twisted  
<http://twistedmatrix.com>

## 9.About

### The Author

Geert Vanderkelen is a member of the MySQL Support Team at Sun Microsystems. He is based in Germany and has worked for MySQL AB since April, 2005. Before joining MySQL he worked as developer, DBA and SysAdmin for various companies in Belgium and Germany. Today Geert specializes in MySQL Cluster and works together with colleagues around the world to ensure continued support for both customers and community. He is also the maintainer of Sun's MySQL Connector/Python.

### Contact

Geert Vanderkelen  
[geert.vanderkelen@sun.com](mailto:geert.vanderkelen@sun.com)

