[**Python 连接Postgresql psycopg2 【转】**](http://www.cnblogs.com/leajon-first/archive/2012/11/23/2784339.html)

原文：<http://blog.csdn.net/dajianshi/article/details/763902>2

由于要在python访问PostgreSQL数据库，需要一个符合DB-API的连接库。通过搜索，锁定两个候选库：一个是[py-postgresql](http://python.projects.postgresql.org/)，一个是[Psycopg2](http://initd.org/psycopg/)。简单的看了一下py-postgresql的文档，发现它并非基于DB-api接口的，而是使用postgresql的API，这可不是我想要的，只好排除。剩下就是这个名字很怪异的Psycopg，看了下，它遵循DB-API规范，好了就选它了，下载安装，试用，老规矩，记录下来要点备忘。

一、安装

由于我开发环境在windows下，所以我下载的是window下的一键安装版本，安装过程非常容易没啥可说的。

二、使用

面的简单使用流程是官网的文档中的例子，看上去是这么的熟悉，是个不错的开始。

[?](http://www.cnblogs.com/leajon-first/archive/2012/11/23/2784339.html)

|  |
| --- |
| # 引入psycopg2库  >>> import psycopg2    # 连接到数据库test  >>> conn = psycopg2.connect("dbname=test user=postgres")    # 建立Cursor对象  >>> cur = conn.cursor()    # 执行sql命令：创建一个新表test  >>> cur.execute("CREATE TABLE test (id serial PRIMARY KEY, num integer, data varchar);")    # 传递数据到语句中形参中，让Psycopg处理正确的转换问题（可以避免sql注入）  >>> cur.execute("INSERT INTO test (num, data) VALUES (%s, %s)",  ...      (100, "abc'def"))    # 插叙数据，并获取结果  >>> cur.execute("SELECT \* FROM test;")  >>> cur.fetchone()  (1, 100, "abc'def")    # 提交数据改变  >>> conn.commit()    # 关闭Cursor对象和连接对象  >>> cur.close()  >>> conn.close() |

下面就要看psycopg是否能完全满足我的需求：

（-）建立数据库连接

psycopg2中建立数据库连接需要使用connect方法，方法形式如下：

1. psycopg2.connect(dsn **or** params [, connection\_factory] [, async=0])

返回值是一个Connection对象。

1，可以使用DSN连接数据库，也就是数据源名称字符串，例如

1. conn = psycopg2.connect("dbname=test user=postgres password=secret")

2，也可以使用命名参数进行连接。

1. conn = psycopg2.connect(database="test", user="postgres", password="secret")

基本的连接参数有:

* dbname – 数据库名（仅在DSN中使用有效）
* database – 数据库名（仅在使用命名参数时可用）
* user – 连接数据所用的数据库用户名
* password – 用户的密码
* host – 数据库所在主机名或IP地址（默认为本机）
* port – 端口号（默认为5432 ）

还有其他一些参数，请参见PostgreSQL官方文档 [list of supported parameters](http://www.postgresql.org/docs/current/static/libpq-connect.html#LIBPQ-PQCONNECTDBPARAMS)。另外需要注意的是，同样的参数也可以通过环境变量的形式传递到客户端库。

3，使用*connection\_factory*参数，可以指定自定义的Connection连接类，详情查看官方文档 [*Connection and cursor factories*](http://initd.org/psycopg/docs/advanced.html#subclassing-connection) 。

4，设置参数*async*=1，则会创建一个异步的连接对象。具体参见官方文档[*Asynchronous support*](http://initd.org/psycopg/docs/advanced.html#async-support) 了解异步的优缺点。

（二）创建字典形式的Cursor对象

简单的创建Cursor对象十分简单，但是这不能满足我的需求。DictCursor这个才是我所需要的，也就是在创建Cursor的时候给出参数cursor\_factory

1. <p>**import** psycopg2</p><p>#为了使用DictCursor
2. </p><p>**import** psycopg2.extras</p>cursor = conn.cursor(cursor\_factory=psycopg2.extras.DictCursor)

使用过dbapi编程的，这个好处你懂的  
（三）使用命名参数传递数据  
我的另一个需求就是，执行sql语句时，最好使用命名参数传递数据，幸运的是psycopg2也是支持命名参数的！psycopg中命名参数的形式是：【%(argname)s】，注意在中括号中的内容就是命名参数要求的形式，其中argname是你起的参数名称，其他的百分号小括号以及小括号后面的s，照写就是。  
使用命名参数的好处就是，你不用考虑传递数据的顺序，可以使用字典传递数据（注意字典的key，应该等于上面所说的argname）：

1. >>> cur.execute(
2. ...     """INSERT INTO some\_table (an\_int, a\_date, another\_date, a\_string)
3. ...         VALUES (%(int)s, %(date)s, %(date)s, %(str)s);""",
4. ...     {'int': 10, 'str': "O'Reilly", 'date': datetime.date(2005, 11, 18)})

（四）其他

psycopg有一个怪癖，执行sql时必须在sql语句末尾加分号，不能省略！不过这个咱可以克服^\_^