用Python实现数据库编程

<用PYTHON进行数据库编程>

用PYTHON语言进行数据库编程, 至少有六种方法可供采用. 我在实际项目中采用,不但功能强大,而且方便快捷.以下是我在工作和学习中经验总结.

方法一:使用DAO (Data Access Objects)

这个第一种方法可能会比较过时啦.不过还是非常有用的. 假设你已经安装好了PYTHONWIN,现在开始跟我上路吧……
找到工具栏上Tools à COM MakePy utilities,你会看到弹出一个Select Library的对话框, 在列表中选择'Microsoft DAO 3.6 Object Library'(或者是你所有的版本).
现在实现对数据的访问:

```
#实例化数据库引擎
import win32com.client
engine = win32com.client.Dispatch("DAO.DBEngine.35")

#实例化数据库对象,建立对数据库的连接
db = engine.OpenDatabase(r"c:\temp\mydb.mdb")
```

现在你有了数据库引擎的连接,也有了数据库对象的实例.现在就可以打开一个recordset了. 假设在数据库中已经有一个表叫做'customers'. 为了打开这个表,对其中数据进行处理,我们使用下面的语法:

```
rs = db.OpenRecordset("customers")
#可以采用SQL语言对数据集进行操纵
rs = db.OpenRecordset("select * from customers where state = 'OH'")
```

你也可以采用DAO的execute方法. 比如这样:

```
db.Execute("delete * from customers where balancetype = 'overdue' and name = 'bill'")
#注意,删除的数据不能复原了J
```

EOF 等属性也是可以访问的, 因此你能写这样的语句:

```
while not rs.EOF:
    print rs.Fields("State").Value
    rs.MoveNext()
```

我最开始采用这个方法,感觉不错.

方法二:使用Python DB API,Python ODBC modules(you can use ODBC API directly, but maybe it is difficult for most beginner.)

为了在Python里面也能有通用的数据库接口,DB-SIG为我们提供了Python数据库.(欲知详情,访问DB-SIG的网站,http://www.python.org/sigs/db-sig/). Mark Hammond的win32扩展PythonWin里面包含了这些API的一个应用-odbc.pyd. 这个数据库API仅仅开放了一些有限的ODBC函数的功能(那不是它的目的),但是它使用起来很简单,而且在win32里面是免费的.

安装odbc.pyd的步骤如下:

1. 安装python软件包:
   http://www.python.org/download/
2. 安装Mark Hammond的最新版本的python win32扩展 - PythonWin:
   http://starship.python.net/crew/mhammond/
3. 安装必要的ODBC驱动程序,用ODBC管理器为你的数据库配置数据源等参数

你的应用程序将需要事先导入两个模块:

dbi.dll - 支持各种各样的SQL数据类型,例如:日期-dates

odbc.pyd – 编译产生的ODBC接口

下面有一个例子:

```
import dbi, odbc    # 导入ODBC模块
import time         # 标准时间模块

dbc = odbc.odbc(    # 打开一个数据库连接
    'sample/monty/spam' # '数据源/用户名/密码'
    )
crsr = dbc.cursor() # 产生一个cursor
crsr.execute(       # 执行SQL语言
    """
    SELECT country_id, name, insert_change_date
```

```
    FROM country
    ORDER BY name
    """
    )
  print 'Column descriptions:' # 显示行描述
  for col in crsr.description:
    print ' ', col
  result = crsr.fetchall()    # 一次取出所有的结果
  print '\nFirst result row:\n ', result[0] # 显示结果的第一行
  print '\nDate conversions:' # 看看dbiDate对象如何?
  date = result[0][-1]
  fmt = '  %-25s%-20s'
  print fmt % ('standard string:', str(date))
  print fmt % ('seconds since epoch:', float(date))
  timeTuple = time.localtime(date)
  print fmt % ('time tuple:', timeTuple)
  print fmt % ('user defined:', time.strftime('%d %B %Y', timeTuple))
```

下面是结果:

```
  --------------------------------输出(output)----------------------------
  Column descriptions:
   ('country_id', 'NUMBER', 12, 10, 10, 0, 0)
   ('name', 'STRING', 45, 45, 0, 0, 0)
   ('insert_change_date', 'DATE', 19, 19, 0, 0, 1)

  First result row:
   (24L, 'ARGENTINA', <DbiDate object at 7f1c80>)

  Date conversions:
   standard string:        Fri Dec 19 01:51:53 1997
   seconds since epoch:    882517913.0
   time tuple:             (1997, 12, 19, 1, 51, 53, 4, 353, 0)
   user defined:           19 December 1997
```

大家也可以去http://www.python.org/windows/win32/odbc.html看看,那儿有两个Hirendra Hindocha写的例子,还不错.

注意, 这个例子中,结果值被转化为Python对象了.时间被转化为一个dbiDate对象.这里会有一点限制,因为dbiDate只能表示UNIX时间(1 Jan 1970 00:00:00 GMT)之后的时间.如果你想获得一个更早的时间,可能会出现乱码甚至引起系统崩溃.*_*

方法三: 使用 calldll模块

(Using this module, you can use ODBC API directly. But now the python version is 2.1, and I don’t know if other version is compatible with it. 老巫:-)

Sam Rushing的calldll模块可以让Python调用任何动态连接库里面的任何函数,厉害吧?哈.其实,你能够通过直接调用odbc32.dll里面的函数操作ODBC.Sam提供了一个包装模块odbc.py,就是来做这个事情的.也有代码来管理数据源,安装ODBC,实现和维护数据库引擎(Microsoft Access).在那些演示和例子代码中,还有一些让人侧目的好东东,比如cbdemo.py,有一个信息循环和窗口过程的Python函数!

[你可以到Sam’s Python Software去找到calldll的相关连接,那儿还有其他好多有趣的东西]

下面是安装CALLDLL包的步骤:

1. 安装PYTHON软件包(到现在为止最多支持2.1版本)
2. 下载calldll-2001-05-20.zip:
   ftp://squirl.nightmare.com/pub/python/python-ext/calldll-2001-05-20.zip
3. 在LIB路径下面创建一个新路径比如说:
   c:\Program Files\Python\lib\caldll\
4. 在原目录下解压calldll.zip
5. 移动calldll\lib\中所有的文件到上面一个父目录(calldll)里面,删除子目录(lib)
6. 在CALL目录里面生成一个file __init__.py文件,象这样:
   # File to allow this directory to be treated as a python 1.5
package.
7. 编辑calldll\odbc.py:
   在"get_info_word"和"get_info_long"里面,改变"calldll.membuf"为"windll.membuf"

下面是一个怎么使用calldll的例子:

```
from calldll import odbc

dbc = odbc.environment().connection()   # create connection
dbc.connect('sample', 'monty', 'spam')  # connect to db
```

```
# alternatively, use full connect string:
# dbc.driver_connect('DSN=sample;UID=monty;PWD=spam')
print 'DBMS: %s %s\n' % (    # show DB information
    dbc.get_info(odbc.SQL_DBMS_NAME),
    dbc.get_info(odbc.SQL_DBMS_VER)
    )
result = dbc.query(         # execute query & return results
    """
    SELECT country_id, name, insert_change_date
    FROM country
    ORDER BY name
    """
    )
print 'Column descriptions:' # show column descriptions
for col in result[0]:
    print ' ', col
print '\nFirst result row:\n ', result[1]  # show first result row
```

--------output(输出)--------

```
DBMS: Oracle 07.30.0000

Column descriptions:
 ('COUNTRY_ID', 3, 10, 0, 0)
 ('NAME', 12, 45, 0, 0)
 ('INSERT_CHANGE_DATE', 11, 19, 0, 1)

First result row:
 ['24', 'ARGENTINA', '1997-12-19 01:51:53']
```

方法四: 使用ActiveX Data Object(ADO)

现在给出一个通过Microsoft's ActiveX Data Objects (ADO)来连接MS Access 2000数据库的实例.使用ADO有以下几个好处: 首先,与DAO相比,它能更快地连接数据库;其次,对于其他各种数据库(SQL Server, Oracle, MySQL, etc.)来说,ADO都是非常有效而方便的;再

有,它能用于XML和文本文件和几乎其他所有数据,因此微软也将支持它比DAO久一些.

第一件事是运行makepy.尽管这不是必须的,但是它对于提高速度有帮助的.而且在PYTHONWIN里面运行它非常简单: 找到工具栏上Tools à COM MakePy utilities,你会看到弹出一个Select Library的对话框, 在列表中选择’ Microsoft ActiveX Data Objects 2.5 Library ‘(或者是你所有的版本).

然后你需要一个数据源名*Data Source Name* [DSN] 和一个连接对象. [我比较喜欢使用DSN-Less 连接字符串 (与系统数据源名相比, 它更能提高性能且优化代码)]

就MS Access来说,你只需要复制下面的DSN即可.对于其他数据库,或者象密码设置这些高级的功能来说,你需要去 [Control Panel控制面板 | 管理工具Administrative Tools | 数据源Data Sources (ODBC)]. 在那里,你可以设置一个系统数据源DSN. 你能够用它作为一个系统数据源名,或者复制它到一个字符串里面,来产生一个DSN-Less 的连接字符串. 你可以在网上搜索DSN-Less 连接字符串的相关资料. 好了,这里有一些不同数据库的DSN-Less连接字符串的例子:SQL Server, Access, FoxPro, Oracle , Oracle, Access, SQL Server, 最后是 MySQL.

```
>>> import win32com.client
>>> conn = win32com.client.Dispatch(r'ADODB.Connection')
>>> DSN = 'PROVIDER=Microsoft.Jet.OLEDB.4.0;DATA SOURCE=C:/MyDB.mdb;'
>>> conn.Open(DSN)
```

经过上面的设置之后,就可以直接连接数据库了:

首要的任务是打开一个数据集/数据表

```
>>> rs = win32com.client.Dispatch(r'ADODB.Recordset')
>>> rs_name = 'MyRecordset'
>>> rs.Open('[' + rs_name + ']', conn, 1, 3)
```

[1和3是常数.代表adOpenKeyset 和adLockOptimistic
