SQLite – PHP SQLite – Python

SQLite - Perl

安装

SQLite3 可使用 Perl DBI 模块与 Perl 进行集成。Perl DBI 模块是 Perl 编程语言的数据库访问模块。它定义了一组提供标准数据库接口的方法、变量及规则。

下面显示了在 Linux/UNIX 机器上安装 DBI 模块的简单步骤：

```
$ wget http://search.cpan.org/CPAN/authors/id/T/TI/TIMB/DBI-1.625.tar.gz
$ tar xvfz DBI-1.625.tar.gz
$ cd DBI-1.625
$ perl Makefile.PL
$ make
$ make install
```

如果您需要为 DBI 安装 SQLite 驱动程序，那么可按照以下步骤进行安装：

```
$ wget http://search.cpan.org/CPAN/authors/id/M/MS/MSERGEANT/DBD-SQLite-1.11.tar.gz
$ tar xvfz DBD-SQLite-1.11.tar.gz
$ cd DBD-SQLite-1.11
$ perl Makefile.PL
$ make
$ make install
```

DBI 接口 API

以下是重要的 DBI 程序，可以满足您在 Perl 程序中使用 SQLite 数据库的需求。如果您需要了解更多细节，请查看 Perl DBI 官方文档。

序号	API & 描述
1	**DBI->connect($data_source, "", "", \%attr)** 建立一个到被请求的 $data_source 的数据库连接或者 session。如果连接成功，则返回一个数据库处理对象。 数据源形式如下所示：**DBI:SQLite:dbname='test.db'**。其中，SQLite 是 SQLite 驱动程序名称，test.db 是 SQLite 数据库文件的名称。如果文件名 *filename* 赋值为 **':memory:'**，那么它将会在 RAM 中创建一个内存数据库，这只会在 session 的有效时间内持续。 如果文件名 filename 为实际的设备文件名称，那么它将使用这个参数值尝试打开数据库文件。如果该名称的文件不存在，那么将创建一个新的命名为该名称的数据库文件。 您可以保留第二个和第三个参数为空白字符串，最后一个参数用于传递各种属性，详见下面的实例讲解。
2	**$dbh->do($sql)**

	该例程准备并执行一个简单的 SQL 语句。返回受影响的行数，如果发生错误则返回 undef。返回值 -1 意味着行数未知，或不适用 ，或不可用。在这里，$dbh 是由 DBI->connect() 调用返回的处理。
3	**$dbh->prepare($sql)** 该例程为数据库引擎后续执行准备一个语句，并返回一个语句处理对象。
4	**$sth->execute()** 该例程执行任何执行预准备的语句需要的处理。如果发生错误则返回 undef。如果成功执行，则无论受影响的行数是多少，总是返回 true。在这里，$sth 是由 $dbh->prepare($sql) 调用返回的语句处理。
5	**$sth->fetchrow_array()** 该例程获取下一行数据，并以包含各字段值的列表形式返回。在该列表中，Null 字段将作为 undef 值返回。
6	**$DBI::err** 这相当于 $h->err。其中，$h 是任何的处理类型，比如 $dbh、$sth 或 $drh。该程序返回最后调用的驱动程序（drive r）方法的数据库引擎错误代码。
7	**$DBI::errstr** 这相当于 $h->errstr。其中，$h 是任何的处理类型，比如 $dbh、$sth 或 $drh。该程序返回最后调用的 DBI 方法的数据库引擎错误消息。
8	**$dbh->disconnect()** 该例程关闭之前调用 DBI->connect() 打开的数据库连接。

连接数据库

下面的 Perl 代码显示了如何连接到一个现有的数据库。如果数据库不存在，那么它就会被创建，最后将返回一个数据库对象。

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "SQLite";
my $database = "test.db";
my $dsn = "DBI:$driver:dbname=$database";
my $userid = "";
my $password = "";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
                      or die $DBI::errstr;

print "Opened database successfully\n";
```

现在，让我们来运行上面的程序，在当前目录中创建我们的数据库 **test.db**。您可以根据需要改变路径。保存上面代码到 sqlite.pl 文件中，并按如下所示执行。如果数据库成功创建，那么会显示下面所示的消息：

```
$ chmod +x sqlite.pl
$ ./sqlite.pl
Open database successfully
```

创建表

下面的 Perl 代码段将用于在先前创建的数据库中创建一个表：

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "SQLite";
my $database = "test.db";
my $dsn = "DBI:$driver:dbname=$database";
my $userid = "";
my $password = "";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
                      or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(CREATE TABLE COMPANY
      (ID INT PRIMARY KEY     NOT NULL,
       NAME           TEXT    NOT NULL,
       AGE            INT     NOT NULL,
       ADDRESS        CHAR(50),
       SALARY         REAL););
my $rv = $dbh->do($stmt);
if($rv < 0){
   print $DBI::errstr;
} else {
   print "Table created successfully\n";
}
$dbh->disconnect();
```

上述程序执行时，它会在 **test.db** 中创建 COMPANY 表，并显示下面所示的消息：

```
Opened database successfully
Table created successfully
```

注意：如果您在任何操作中遇到了下面的错误： in case you see following error in any of the operation:

```
DBD::SQLite::st execute failed: not an error(21) at dbdimp.c line 398
```

在这种情况下，您已经在 DBD-SQLite 安装中打开了可用的 dbdimp.c 文件，找到 **sqlite3_prepare()** 函数，并把它的第三个参数 0 改为 -1。最后使用 **make** 和 **make install** 安装 DBD::SQLite，即可解决问题。 in this case you will have open dbdimp.c file available in DBD-SQLite installation and find out **sqlite3_prepare()** function and change its third argument to -1 instead of 0. Finally install DBD::SQLite using **make** and do **make install** to resolve the problem.

INSERT 操作

下面的 Perl 程序显示了如何在上面创建的 COMPANY 表中创建记录：

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "SQLite";
my $database = "test.db";
my $dsn = "DBI:$driver:dbname=$database";
my $userid = "";
my $password = "";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
                      or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
      VALUES (1, 'Paul', 32, 'California', 20000.00 ));
my $rv = $dbh->do($stmt) or die $DBI::errstr;

$stmt = qq(INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
      VALUES (2, 'Allen', 25, 'Texas', 15000.00 ));
$rv = $dbh->do($stmt) or die $DBI::errstr;

$stmt = qq(INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
      VALUES (3, 'Teddy', 23, 'Norway', 20000.00 ));
$rv = $dbh->do($stmt) or die $DBI::errstr;

$stmt = qq(INSERT INTO COMPANY (ID,NAME,AGE,ADDRESS,SALARY)
      VALUES (4, 'Mark', 25, 'Rich-Mond ', 65000.00 ););
$rv = $dbh->do($stmt) or die $DBI::errstr;

print "Records created successfully\n";
$dbh->disconnect();
```

上述程序执行时，它会在 COMPANY 表中创建给定记录，并会显示以下两行：

```
Opened database successfully
Records created successfully
```

SELECT 操作

下面的 Perl 程序显示了如何从前面创建的 COMPANY 表中获取并显示记录：

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "SQLite";
my $database = "test.db";
my $dsn = "DBI:$driver:dbname=$database";
my $userid = "";
my $password = "";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
                      or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(SELECT id, name, address, salary  from COMPANY;);
my $sth = $dbh->prepare( $stmt );
my $rv = $sth->execute() or die $DBI::errstr;
if($rv < 0){
   print $DBI::errstr;
}
while(my @row = $sth->fetchrow_array()) {
      print "ID = ". $row[0] . "\n";
      print "NAME = ". $row[1] ."\n";
      print "ADDRESS = ". $row[2] ."\n";
      print "SALARY =  ". $row[3] ."\n\n";
}
print "Operation done successfully\n";
$dbh->disconnect();
```

上述程序执行时，它会产生以下结果：

```
Opened database successfully
ID = 1
NAME = Paul
ADDRESS = California
SALARY =  20000

ID = 2
NAME = Allen
```

```
ADDRESS = Texas
SALARY =  15000

ID = 3
NAME = Teddy
ADDRESS = Norway
SALARY =  20000

ID = 4
NAME = Mark
ADDRESS = Rich-Mond
SALARY =  65000

Operation done successfully
```

UPDATE 操作

下面的 Perl 代码显示了如何使用 UPDATE 语句来更新任何记录，然后从 COMPANY 表中获取并显示更新的记录：

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "SQLite";
my $database = "test.db";
my $dsn = "DBI:$driver:dbname=$database";
my $userid = "";
my $password = "";
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
                      or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(UPDATE COMPANY set SALARY = 25000.00 where ID=1;);
my $rv = $dbh->do($stmt) or die $DBI::errstr;
if( $rv < 0 ){
   print $DBI::errstr;
}else{
   print "Total number of rows updated : $rv\n";
}
$stmt = qq(SELECT id, name, address, salary  from COMPANY;);
my $sth = $dbh->prepare( $stmt );
$rv = $sth->execute() or die $DBI::errstr;
if($rv < 0){
   print $DBI::errstr;
}
while(my @row = $sth->fetchrow_array()) {
      print "ID = ". $row[0] . "\n";
```

```
      print "NAME = ". $row[1] ."\n";
      print "ADDRESS = ". $row[2] ."\n";
      print "SALARY =  ". $row[3] ."\n\n";
}
print "Operation done successfully\n";
$dbh->disconnect();
```

上述程序执行时，它会产生以下结果：

```
Opened database successfully
Total number of rows updated : 1
ID = 1
NAME = Paul
ADDRESS = California
SALARY =  25000

ID = 2
NAME = Allen
ADDRESS = Texas
SALARY =  15000

ID = 3
NAME = Teddy
ADDRESS = Norway
SALARY =  20000

ID = 4
NAME = Mark
ADDRESS = Rich-Mond
SALARY =  65000

Operation done successfully
```

DELETE 操作

下面的 Perl 代码显示了如何使用 DELETE 语句删除任何记录，然后从 COMPANY 表中获取并显示剩余的记录：

```
#!/usr/bin/perl

use DBI;
use strict;

my $driver   = "SQLite";
my $database = "test.db";
my $dsn = "DBI:$driver:dbname=$database";
my $userid = "";
my $password = "";
```

```
my $dbh = DBI->connect($dsn, $userid, $password, { RaiseError => 1 })
                      or die $DBI::errstr;
print "Opened database successfully\n";

my $stmt = qq(DELETE from COMPANY where ID=2;);
my $rv = $dbh->do($stmt) or die $DBI::errstr;
if( $rv < 0 ){
   print $DBI::errstr;
}else{
   print "Total number of rows deleted : $rv\n";
}
$stmt = qq(SELECT id, name, address, salary  from COMPANY;);
my $sth = $dbh->prepare( $stmt );
$rv = $sth->execute() or die $DBI::errstr;
if($rv < 0){
   print $DBI::errstr;
}
while(my @row = $sth->fetchrow_array()) {
      print "ID = ". $row[0] . "\n";
      print "NAME = ". $row[1] ."\n";
      print "ADDRESS = ". $row[2] ."\n";
      print "SALARY =  ". $row[3] ."\n\n";
}
print "Operation done successfully\n";
$dbh->disconnect();
```

上述程序执行时，它会产生以下结果：

```
Opened database successfully
Total number of rows deleted : 1
ID = 1
NAME = Paul
ADDRESS = California
SALARY =  25000

ID = 3
NAME = Teddy
ADDRESS = Norway
SALARY =  20000

ID = 4
NAME = Mark
ADDRESS = Rich-Mond
SALARY =  65000

Operation done successfully
```

SQLite – PHP SQLite – Python

点我分享笔记