

第17篇 (数据库) Qt数据库简单应用

代码地址：<https://github.com/Lornatang/QtStartQuicklyTutorial/tree/main/SQL01>

一、数据库简介

二、数据库驱动

一、数据库简介

Qt中的 `QtSql` 模块提供了对数据库的支持，该模块中的众多类基本上可以分为三层，如下图所示。

QtSql 模块的类分层	
用户接口层	
QSqlQueryModel、QSqlTableModel 和 QSqlRelationalTableModel	
SQL 接口层	
QSqlDatabase、QSqlQuery、QSqlError、QSqlField、QSqlIndex 和 QSqlRecord	
驱动层	
QSqlDriver、QSqlDriverCreator<T>、QSqlDriverCreatorBase、QSqlDriverPlugin 和 QSqlResult	

其中驱动层为具体的数据库和SQL接口层之间提供了底层的桥梁；SQL接口层提供了对数据库的访问，其中的 `QSqlDatabase` 类用来创建连接，`QSqlQuery` 类可以使用SQL语句来实

现与数据库交互，其他几个类对该层提供了支持；用户接口层的几个类实现了将数据库中的数据链接到窗口部件上，这些类是使用前一章的模型/视图框架实现的，它们是更高层次的抽象，即便不熟悉SQL也可以操作数据库。如果要使用 `QtQml` 模块中的这些类，需要在项目文件（.pro文件）中添加 `QT += sql` 这一行代码。对应数据库部分的内容，大家可以在帮助中查看SQL Programming关键字。

二、数据库驱动

`QtSql` 模块使用数据库驱动来和不同的数据库接口进行通信。由于Qt的SQL模型的接口是独立于数据库的，所以所有数据库特定的代码都包含在了这些驱动中。Qt现在支持的数据库驱动如下图所示。

Driver Type	Description
QDB2	IBM DB2
QIBASE	Borland InterBase Driver
QMYSQL	MySQL Driver
QOCI	Oracle Call Interface Driver
QODBC	ODBC Driver (includes Microsoft SQL Server)
QPSQL	PostgreSQL Driver
QSQLITE	SQLite version 3 or above
QSQLITE2	SQLite version 2
QTDS	Sybase Adaptive Server

需要说明的是，由于GPL许可证的兼容性问题，并不是这里列出的所有驱动插件都提供给了Qt的开源版本。下面我们通过程序来查看一下现在版本的Qt中可用的数据库插件。

1. 新建Qt 控制台应用，名称为 `sqldrivers`。
2. 完成后将 `sqldrivers.pro` 项目文件中第一行代码更改为：

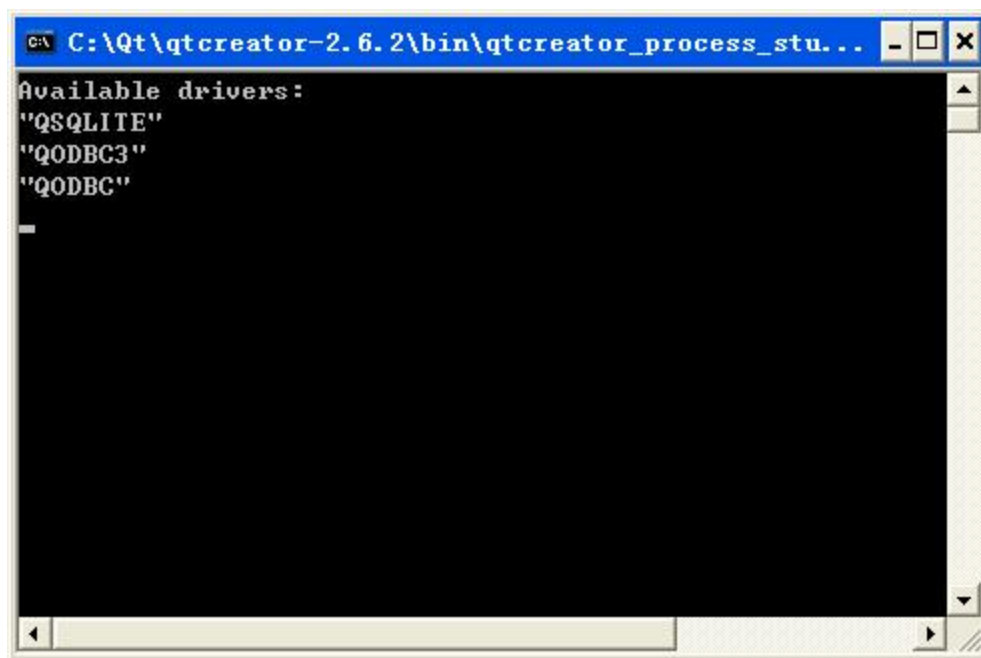
```
QT      += core sql
```

表明使用了 `sql` 模块，然后按下 `Ctrl + S` 快捷键保存该文件。

3. 将 `main.cpp` 文件的内容更改如下：

```
#include <QCoreApplication>#include <QSqlDatabase>#include <QDebug>#include <QStringList>
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    qDebug() << "Available drivers:";
    QStringList drivers = QSqlDatabase::drivers();
    foreach(QString driver, drivers)
        qDebug() << driver;
    return a.exec();
}
```

这里先使用 `drivers()` 函数获取了现在可用的数据库驱动，然后分别进行了输出。运行程序，结果如下图所示。



可以发现，现在只支持三个数据库。这里要重点提一下SQLite数据库，它是一款轻型的文件型数据库，主要应用于嵌入式领域，支持跨平台，而且Qt对它提供了很好的默认支持，所以在本章后面的内容中，我们将使用该数据库作为例子来进行讲解。

三、简单的数据库应用

下面使用SQLite数据库来进行一个简单的演示，创建一个数据库表，然后查找其中的数据并进行输出。我们更改 `main.cpp` 文件的内容如下：

```
#include <QCoreApplication>#include <QSqlDatabase>#include <QDebug>#include <QSqlQuery>
int main(int argc, char *argv[])
```

```

{
    QApplication a(argc, argv);

    //添加数据库驱动
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    //设置数据库名称
    db.setDatabaseName(":memory:");
    //打开数据库
    if(!db.open())
    {
        return false;
    }

    //以下执行相关sql语句
    QSqlQuery query;

    //新建student表, id设置为主键, 还有一个name项
    query.exec("create table student(id int primary key,name varchar)");

    //向表中插入3条记录
    query.exec("insert into student values(1,'xiaogang')");
    query.exec("insert into student values(2,'xiaoming')");
    query.exec("insert into student values(3,'xiaohong')");

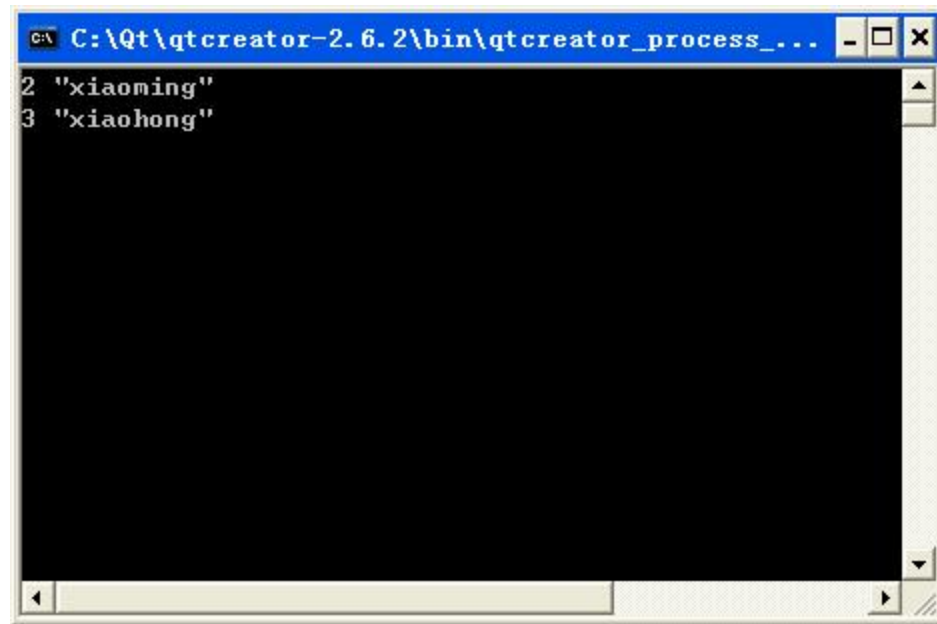
    //查找表中id >=2 的记录的id项和name项的值
    query.exec("select id,name from student where id >= 2");

    //query.next()指向查找到的第一条记录, 然后每次后移一条记录
    while(query.next())
    {
        //query.value(0)是id的值, 将其转换为int型
        int value0 = query.value(0).toInt();
        QString value1 = query.value(1).toString();
        //输出两个值
        qDebug() << value0 << value1 ;
    }

    return a.exec();
}

```

这里使用了SQLite数据库, 数据库名为 `:memory:` 表示这是建立在内存中的数据库, 也就是说该数据库只在程序运行期间有效。如果需要保存该数据库文件, 我们可以将它更改为实际的文件路径。程序中使用到的 `QSqlQuery` 类, 将在后面的内容中讲到。运行程序, 结果如下图所示。



```
C:\Qt\qtcreator-2.6.2\bin\qtcreator_process_...  
2 "xiaoming"  
3 "xiaohong"
```