# SQLdb Tutorial4 编辑



**Databases portal**

References:

- General info
- Libraries
- Field types
- Controls
- FAQ
- SQL how-to
- Working With TSQLQuery
- In-memory database applications

Tutorials/practical articles:

- Overview
- 0 - Database set-up
- 1 - Getting started
- 2 - Editing
- 3 - Queries
- 4 - Data modules
- SQLdb Programming Reference

Databases

> Advantage - MySQL - MSSQL - Postgres - Interbase - Firebird - Oracle - ODBC - Paradox - SQLite - dBASE - MS Access - Zeos

## Introduction

This tutorial is an attempt to demonstrate the use of Lazarus Data Modules to isolate the data access components of a project from the program logic associated with the access. Such isolation makes program maintenance and debugging easier.

The tutorial was developed using Windows 7 and SQLite3 as the database, with Lazarus 1.0.8 with FPC 2.6.2; however it should work with earlier versions. Similarly, other DBMS and Operating Systems should require minimal change, if any.

## Why use datamodules?

Simple - after following the Lazarus Tutorials:

- SQLdb Tutorial0

- SQLdb Tutorial1
- SQLdb Tutorial2
- SQLdb Tutorial3

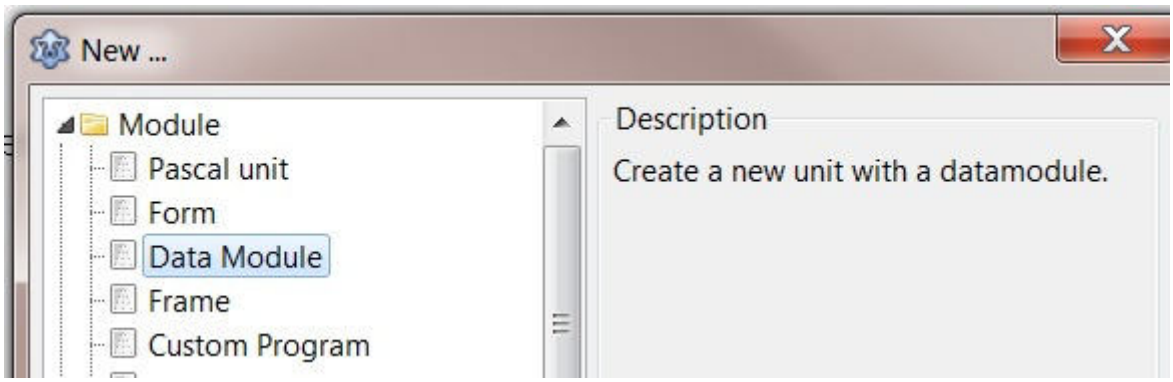
developing a 'real' application becomes harder. 'Form1' grows to an exponential size handling the different Events and Database Queries.

Isolating each table access into a single datamodule makes debugging and maintenance so much easier. An application may have any number of datamodules - a small application with just one or 2 tables can probably suffice with just one Datamodule - a larger application could probably benefit from having a data module for each table or view.

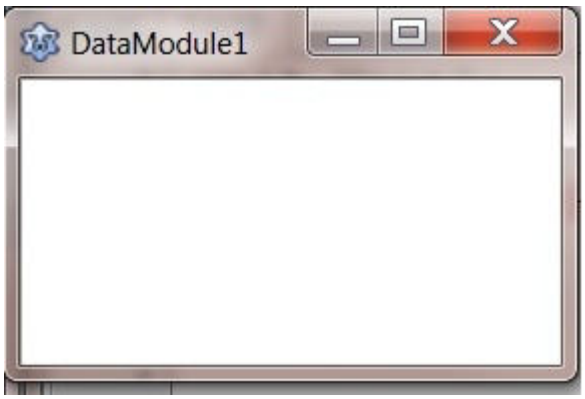
The sample shown here uses just 2 tables with simple queries, but it can be expanded to include more options with each table.

## Getting started

In the Lazarus IDE, create a new Application and then click File --> New --> Data Module




You will be presented with a window as if you selected 'New Form':

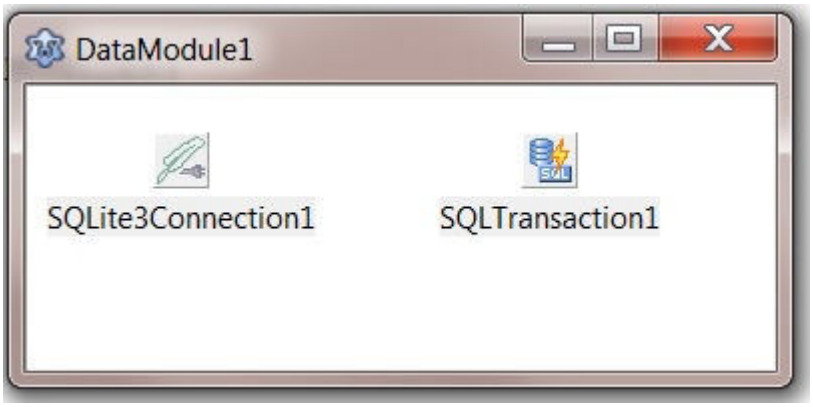



The difference is this window/form will only accept non-visual components. In fact, look at your Component Palette: it has been greatly reduced to only allow selection of ONLY non-visual components.

## Everyone has their own way

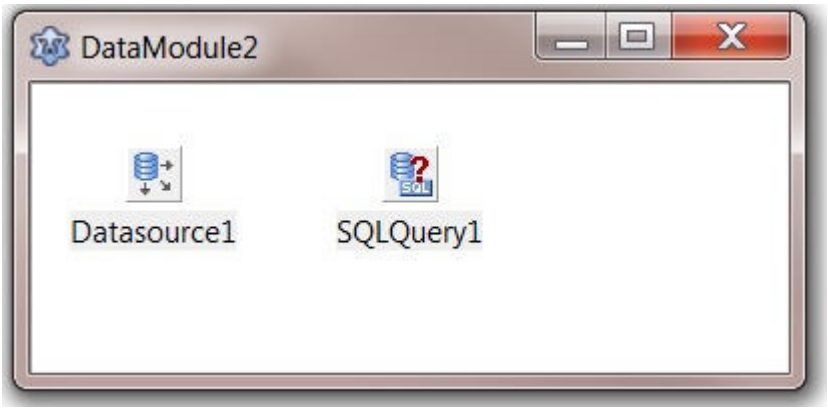
Your use of data modules will vary to suit your own needs. but as an example, I have at least 2 data modules:

Unit: DataModule1 On this module I 'drop' a *T*Connection* and a *TSQLTransaction*.




DataModule1 is used as the connection for all queries.

I then create a DataModuleN for each table or view.




Each DataModuleN will need to have the DataModule1 unit (unit2 in this example) added to the USES clause to connect to the database.

From here everything is the same as stated in SQLdb Tutorial1. The components are connected in the same way and access is identical.

# More uses of data modules

## Additional components

In this example, DataModule1 had nothing more than a Connection and Transaction, but in a 'real' application, this container would typically also hold global non-visual components to be used by the application.

For example, Load and Save INI settings, *TOpenDialog*, *TSaveDialog*, etc. The concept here is to isolate data access from the business logic of an application. A change in data source for any application is never a minimal task, but having the datasources isolated will make the change much easier.

## Debugging

Debugging a program is also a difficult task. By separating data access and business logic, the code to be viewed is halved. Data access and business logic can be tested separately to at least halve the

problem.

The importance of the *DataModule* will become even more obvious when developing other applications using the same database and tables. The data module can of course be reused in the new application.