# Using QDataWidgetMapper to show data from a database in Qt

## Overview

This example shows you how to map a UI component to a database table using `QSqlTableModel` and `QDataWidgetMapper` in Qt. Searching and updating data can be done without using any SQL statements.

The table 'person' has the following columns:

- `id (integer primary key)`, **this is an autoincrement field**
- `firstname (varchar(20))`
- `lastname (varchar(30))`
- `age (integer)`

## Preconditions

For Maemo SQLite development, the following packages must be installed:

- libqt4-sql
- libqt4-sql-sqlite
- libsqlite3-0
- libsqlite3-dev

## Header

Dialog for showing and updating a person's last name:

```
#include <QtGui>
#include <QtSql>

class MyDialog : public QDialog
    {
    Q_OBJECT

    public:
        MyDialog(int id = 0, QWidget *parent = 0);
        ~MyDialog();

    public slots:
        void save();

    private:
        // "person" table mode class
        QSqlTableModel*             tableModel;

        // Maps QLineEdit ui control into "person"
        // table's lastname field
        QDataWidgetMapper*          dataMapper;

        QVBoxLayout*                vboxlayout;
        QLineEdit*                  nameLineEdit;
        QPushButton*                saveButton;
    };
```


(C) Copyright Nokia 2014. All rights reserved.

## Source

```
MyDialog::MyDialog(int id, QWidget *parent)
    : QDialog(parent)
    {
    vboxlayout = new QVBoxLayout(this);

    nameLineEdit = new QLineEdit(this);
    nameLineEdit->setContextMenuPolicy(Qt::NoContextMenu);
    nameLineEdit->setAlignment(Qt::AlignTop);
    vboxlayout->addWidget(nameLineEdit);

    saveButton = new QPushButton("Save",this);
    saveButton->setFixedHeight(60);
    QObject::connect(saveButton, SIGNAL(clicked()), this, SLOT(save()));
    vboxlayout->addWidget(saveButton);

    // 'person' table model
    tableModel = new QSqlTableModel(this);
    tableModel->setTable("person");
    // Set where clause
    // QLineEdit shows person lastname that id is given parameter
    tableModel->setFilter(QString("id=%1").arg(id));
    tableModel->select();

    // Maps QLineEdit ui control into "person" table's lastname field
    dataMapper = new QDataWidgetMapper(this);
    dataMapper->setModel(tableModel);
    // We want that data is stored only if we call QDataWidgetMapper::submit()
    dataMapper->setSubmitPolicy(QDataWidgetMapper::ManualSubmit);
    dataMapper->setItemDelegate(new QSqlRelationalDelegate(dataMapper));
    dataMapper->addMapping(nameLineEdit, tableModel->fieldIndex("lastname"));
    dataMapper->toFirst();

    setLayout(vboxlayout);
    }

MyDialog::~MyDialog()
    {
    }


void MyDialog::save()
    {
    // Stores data from QLineEdit to database using QDataWidgetMapper
    dataMapper->submit();
    }
```

## Postconditions

The UI controls get data from the database and the data can be updated via controls without using any SQL statements.


(C) Copyright Nokia 2014. All rights reserved.

## See also

- Creating an SQLite database in Qt
- Creating a database table in Qt
- Inserting a row into a database in Qt
- Searching for data in a database in Qt
- Deleting data from a database in Qt
- Selecting data from a database without using SQL statements in Qt
- QDataWidgetMapper
- QtSql module


(C) Copyright Nokia 2014. All rights reserved.