

Inhaltsverzeichnis

1. QT-Programmierung: Tipps.....	1
1.1. Quellen.....	1
1.2. Konvertierungen: QString.toInt() u. QString.setNum().....	1
1.3. QString und arg.....	2
1.4. Window Größen festlegen.....	2
1.5. Umlaute.....	2
1.6. Fonts.....	2
1.7. Layout.....	2
1.8. QDebug.....	3
1.9. QMessageBox.....	3
1.10. QInputDialog.....	3
1.11. QTimer.....	4
1.12. Qt KeyEvent.....	4
1.13. Qt PaintEvent.....	4
1.14. Qt und CSS.....	5
1.15. QPushButton/QLabel Farbe setzen (QPalette).....	5
1.16. QLabel Bild laden (QPixmap).....	6
1.17. QRadioButton, QComboBox, QCheckBox, QListWidget,QList.....	6
1.18. QFileDialog,QFile,QTextStream.....	7
1.19. QListView und QStringList, QStringListModel.....	7
1.20. externe Libraries hinzufügen.....	8

1. QT-Programmierung: Tipps

1.1. Quellen

- ☐ <http://zetcode.com/gui/qt5/> (SUPER)
- ☐ <https://www.youtube.com/playlist?list=PLS1QulWo1RIZiBcTr5urECberTITj7gjA> (SUPER)
- ☐ http://www.bogotobogo.com/Qt/Qt5_GridLayout.php (SUPER)
- ☐ <http://doc.qt.io/qt-5/qtexamplesandtutorials.html>
- ☐ <https://wiki.qt.io/Category:HowTo> (SUPER)

1.2. Konvertierungen: QString.toInt() u. QString.setNum()

QString → int

```
QString str("1234");  
  
int wert= str.toInt();
```

int → QString

```
QString str;  
str.setNum(1234);      // str == "1234"
```

int/double in ein Label,... schreiben

muss nicht extra in einen QString konvertiert werden, sondern:

```
ui->labelZahl->setNum(wert);
```

1.3. QString und arg

```
QString line = tr("<b>%1</b> says: <i>%2</i>").arg(nick).arg(message);
```

1.4. Window Größen festlegen

Wir wollen diesmal auch die Maximale und minimale Größe definieren können.

```
setFixedSize(200, 120);  
  
setGeometry(62, 40, 75, 30);
```

Oder den gesamten Bildschirm nutzen

```
#include <QScreen>  
  
resize(QApplication::primaryScreen()->availableSize());
```

1.5. Umlaute

Wenn der Editor mit UTF-8 arbeitet:

```
QString::fromUtf8( "überhaupt nicht" )
```

1.6. Fonts

```
quit->setFont(QFont("Times", 18, QFont::Bold));
```

1.7. Layout

widget.h (Auszug)

```
#include <QtGui/QVBoxLayout>  
#include <QtGui/QHBoxLayout>
```

widget.cpp (Auszug)

```
// 2. Layout festlegen
vlayout = new QVBoxLayout();
hlayout = new QHBoxLayout();

// 2.1. die Buttons horizontal
hlayout->addWidget(qButtonHello);
hlayout->addWidget(qButtonClear);
hlayout->addWidget(qButtonQuit);

// 2.2. das Label und die Buttons vertikal
vlayout->addWidget(qLabelHello);
vlayout->addLayout(hlayout);

// 2.3. ins Formular damit
this->setLayout(vlayout);
```

1.8. QDebug

```
#include <QDebug>

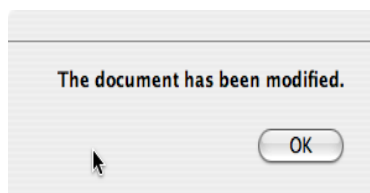
QDebug() << "The document has been modified.";
```

1.9. QMessageBox

```
QMessageBox msgBox;

msgBox.setText("The document has been modified.");

msgBox.exec();
```



1.10. QInputDialog

```
bool ok;
int i = QInputDialog::getInt(this, tr("Input Dialog"),
                             tr("Percentage:"), 25, 0, 100, 1, &ok);
if (ok)
    integerLabel->setText(tr("%1%").arg(i));
```

1.11. QTimer

```
// Timer -----  
QTimer* timer= new QTimer();  
timer->setInterval(1000);  
timer->start();  
  
QObject::connect(timer, SIGNAL(timeout()),  
                 this, SLOT(on_timer_timeout()));
```

in der cpp-Datei:

```
void Widget::on_timer_timeout(){  
    ...  
}
```

in der h-Datei:

```
public slots:  
    void on_timer_timeout();
```

1.12. Qt KeyEvent

```
#include <QWidget>  
#include <QKeyEvent>  
  
class MyClass : public QWidget  
{  
    Q_OBJECT  
public:  
    explicit MyClass(QWidget *parent = 0):QWidget(parent){}  
  
protected:  
    void keyPressedEvent(QKeyEvent *event) override  
    {  
        if(event->key() == Qt::Key_R)  
        {  
            //Do something when 'R' key is pressed  
        }  
    }  
};
```

1.13. Qt PaintEvent

```
class Widget : public QWidget{  
    Q_OBJECT
```

```
public:
    Widget(QWidget *parent = 0);
    ~Widget();

protected:
    // wird immer aufgerufen, wenn das Fenster/Widget
    // neu gezeichnet werden soll
    void paintEvent(QPaintEvent* e);

};
```

```
void Widget::paintEvent(QPaintEvent* e){

    QPainter painter(this);

    painter.setBrush(Qt::blue);
    painter.setPen(Qt::darkRed);
    painter.drawRect(0, 0, 100,100);

    painter.setBrush(Qt::NoBrush);
    painter.setPen(Qt::darkRed);
    painter.drawRect(40, 40, 100, 100);
}
```

1.14. Qt und CSS

- Hintergrundfarbe eines Widgets ändern.

```
this->setStyleSheet("background-color: green;");
```

1.15. QPushButton/QLabel Farbe setzen (QPalette)

```
//Für einen Button:
// neuen Button anlegen
QPushButton *button = new QPushButton();

// Palette des Buttons holen
QPalette palette = button->palette();

// gewünschte Werte ändern
palette.setColor(QPalette::Button,QColor(255,255,0));

// Palette setzen
button->setPalette(palette);
```

```
// Für eine Label:  
// neues Label anlegen  
QLabel *label = new QLabel;  
// Palette des Labels holen  
QPalette palette = label->palette();  
// gewünschte Werte ändern  
palette.setColor(QPalette::Background, QColor(255,255,0));  
// Palette setzen  
label->setPalette(palette);
```

Designer:

QWidget::setAutoFillBackground muß enabled sein

QWidget::Palette setzen

1.16. QLabel Bild laden (QPixmap)

```
QPixmap p("ubuntu.png");  
  
ui->labelBild->setPixmap(p);  
  
ui->labelBild->setVisible(true);
```

1.17. QRadioButton, QComboBox, QCheckBox, QListWidget, QList

```
radioButtonGROSS->setChecked(true);  
  
checkBoxOLIVEN->setChecked(true);  
  
listWidgetZUSATZ->setSelectionMode(QAbstractItemView::MultiSelection);  
listWidgetZUSATZ->addItem(QString::fromUtf8( "Fanta" ));  
listWidgetZUSATZ->addItem("Cola");  
listWidgetZUSATZ->addItem("Clausthaler");  
  
comboBoxBEZAHLUNG->addItem(QString::fromUtf8( "Bar" ));  
comboBoxBEZAHLUNG->addItem(QString::fromUtf8( "überhaupt nicht" ));  
comboBoxBEZAHLUNG->setCurrentIndex(1);
```

```
QString s;  
  
if (ui->radioButtonGROSS->isChecked()) ...  
if (ui->checkBoxOLIVEN->isChecked()) ...  
  
QList<QListWidgetItem*> zusaetze;  
zusaetze= ui->listWidgetZUSATZ->selectedItems();  
  
for (int i=0; i < zusaetze.size(); i++){
```

```
        s+= "    " + zusaetze.at(i)->text() + "\n";  
    }
```

1.18. QFileDialog, QFile, QTextStream

```
void MainWindow::on_pushButton_open_clicked()  
{  
    QString fileName = QFileDialog::getOpenFileName(this,  
        tr("Open Textfile"), ".", tr("Image Files (*.txt *.doc)"));  
    if (!fileName.isEmpty()){  
        QFile file(fileName);  
        if (!file.open(QIODevice::ReadOnly | QIODevice::Text))  
            return;  
        ui->textEdit_Original->setText(file.readAll());  
        file.close();  
    }  
}  
void MainWindow::on_pushButton_save_clicked()  
{  
    QString fileName = QFileDialog::getSaveFileName(this,  
        tr("Save Textfile"), ".", tr("Image Files (*.txt *.doc)"));  
    if (!fileName.isEmpty()){  
        QFile file(fileName);  
        if (!file.open(QIODevice::WriteOnly | QIODevice::Text))  
            return;  
        QString txt= ui->textEdit_Original->toPlainText();  
        QTextStream out(&file);  
        out<< txt;  
        out.flush();  
        file.close();  
    }  
}
```

1.19. QListView und QStringList, QStringListModel

```
// 1. View erstellen  
QListView* listView = new QListView();  
  
//2. Model erstellen und Model u. View verbinden  
QStringListModel* model = new QStringListModel(listView);  
listView->setModel(model);  
  
//3. Daten erzeugen  
QStringList list;  
  
list << "eins" << "zwei"<<"drei";  
  
//4. Daten anzeigen  
model->setStringList(list);
```

1.20. externe Libraries hinzufügen

<http://doc.qt.io/qt-5/third-party-libraries.html>