Signal und Slots mit QT4

Signal und Slots:

Das "Signal und Slots" Konzept ist eines der wichtigsten Besonderheiten in QT.
Man versteht darunter im Grunde genommen das diverse Widgets Signale senden können
und die bei einen anderen Widget (oder auch dem gleichen) in einem sogenante Slots (Schnitstellen) aufgenommen werden, das heist wenn man beide mit einander verbunden hat.
Dabei gilt das man ein jedes Signal grundsätzlich mit jedem Slot verbinden kann solange der Slot das Argument erwartet was das Signal sendet.
Die meisten Widgets haben von Haus aus eine menge Signal und Slots vordefiniert, man kann aber auch ohne weiteres eigene Signal und Slots erstellen.

Schauen wir uns zuerst einmal einen Einfache Signal und Slot verbindung an.
Bei unseren Hello World Programm (aus der QT-Einführung) haben wir bereits Bekanntschaft mit diesem Konzept gehabt.

Quellcode

```
1. QObject::connect(&beenden, SIGNAL(clicked()), &app, SLOT(qu
```

Hier haben wir das Signal "clicked()" mit dem Slot "quit()" verbunden.
Das Signal clicked wird uns von Widget "QPushbutton" zu Verfügung gestellt und dem Slot vom "QApplication" Widget übergeben.
Wenn man sich die diversen Signale und Slots der Widget anschauen möchte dann sollte man die QT-Dokumentation benutzen.
Beim Verbinden von Signal und Slots muss man das Argument welches unter Umständen übergeben wird berücksichtigen.
Bei unseren kleinen Beispiel hier oben waren keinen Argument vorhanden, da das Signal "clicked()" keine Argument übergibt und auch
"quit()" kein Argument erwartet.

Bei diesen Beispiel wird ein String dem Slot geschickt,

Quellcode

```
1. QObject::connect(eingabe, SIGNAL(textChanged(const QString&)), anzeige, SLOT(setText(const QString&)));
```

Man kann wie schon erwähnt immer nur passende Signal and Slots verbinden.

Natürlich kann man auch ohne weiteres eigene Signale und Slots erstellen, dabei sind aber ein paar Besonderheiten zu beachten.
Ganz wichtig ist das Makro Q_OBJECT, wenn das vergessen wird gibt es zwar keinen Probleme beim kompilieren, aber es passiert ganz einfach nach her nichts.
Slots werden als Funktion implementiert und Signale werden mittels den Befehl "emit" gesendet.

Hier ist ein simples Beispiel für das erstellen von Signal und Slots.

signalandslots.cpp:

Quellcode

```
1. # include "gui.h"
2. int main(int argc, char *argv[])
```

```
{
// Hier wird das QT_Programm erschaffen
QApplication app(argc, argv);
Gui window;
window.show();
return app.exec();
}
```

Alles anzeigen

gui.h:

Quellcode

```
# ifndef GUI_H
# define GUI_H
#include <QApplication>
#include <QFont>
#include <QPushButton>
#include <QWidget>
#include <QLabel>
#include <iostream>
using namespace std;
class Gui : public QWidget
{
Q_OBJECT // Wichitg fuer das erstellen eigener Signal and Slots
public:
Gui(QWidget *parent = 0);
private:
QPushButton *beenden,
*hallo;
// Signale und Slots definieren
signals:
void fertig(void);
private slots:
void slotHallo(void);
};
# endif
```

Alles anzeigen

gui.cpp:

Quellcode

```
# ifndef GUI_CPP
# define GUI_CPP
# include "gui.h"
Gui::Gui(QWidget *parent)
: QWidget(parent)
{
// Die Einzelenen Objekte erzeugen
QPushButton *beenden = new QPushButton("Beenden", this);
QPushButton *hallo = new QPushButton("Hallo + Beneden", this);
//setGeometry(PosH, PosV, Breite, Hoehe)
hallo ->setGeometry(50, 60, 200, 30);
```

```
15.  beenden->setGeometry(50, 20, 200, 30);
16.
17.  // Schriftgroesse und schriftart aendern
18.
19.  beenden->setFont(QFont("Times", 12, QFont::Bold));
20.  hallo ->setFont(QFont("Times", 12, QFont::Bold));
21.
22.  // Signale mit Slots verbinden
23.  connect(beenden, SIGNAL(clicked()), qApp, SLOT(quit()));
24.  connect(hallo, SIGNAL(clicked()), this, SLOT(slotHallo()));
25.  connect(this, SIGNAL(fertig()), qApp, SLOT(quit()));
26.
27.  // Den Fenstertitel abspeichern
28.  setWindowTitle("Signal und Slots");
29.
30.  // Minimum Breite und Höhe fuer das Fenster
31.  setMinimumSize(300,100);
32.  }
33.
34.  void Gui::slotHallo(void)
35.  {
36.  // Hallo auf der Kommandozeile ausgeben
37.  cout<<endl<<" ! ! ! H A L L O ! ! !"<<endl;
38.
39.  // 3 Sekunden warten
40.  sleep(3);
41.
42.  cout<<endl<<" Jetzt wird das Signal ferig() gesendet"<<endl;
43.
44.  // Das Signal fertig senden
45.  emit fertig();
46.  }
47.
48.  # endif
```

Alles anzeigen