Giacomo Strangolino

Elettra – Sincrotrone Trieste

QTWatcher and QTWriter classes

mailto: giacomo.strangolino@elettra.trieste.it

QTWatcher

- Reads tango variables using Qtango;
- QObject or base types can be *attached()*;
- on new data, a SLOT can be invoked on the QObject;
- the *data type* is guessed from the QObject SLOT input parameter
- *auto configuration* possible if QObject has suitable slots (e.g. *QProgressBar setMinimum()*)
- On read error, *slots aren't invoked and variables aren't updated*!

QTWatcher with QObjects

```
QProgressBar *pbar = new QProgressBar(this);
QTWatcher *pbarWatcher = new QTWatcher(this);

pbarWatcher->attach(pbar, SLOT(setValue(int)));

// configure maximum and minimum values when available
pbarWatcher->setAutoConfSlot(QTWatcher::Min, SLOT(setMinimum(int)));
pbarWatcher->setAutoConfSlot(QTWatcher::Max, SLOT(setMaximum(int)));

pbarWatcher->setSource("$1/short_scalar_ro");
```

QTWatcher with simple data types

```
short int var;
QTWatcher *intWatcher = new QTWatcher(this);

pbarWatcher->attach(&var);

pbarWatcher->setSource("$1/short_scalar_ro");
```

- var is always up to date;
- tango reads are performed in another thread;
- it is safe to access var in any moment inside your thread.

QTWatcher: signals

- attributeAutoConfigured(const TangoConfigurationParameters *);
- connectionFailed();
- connectionOk(bool);
- connectionErrorMessage(const QString &);
- readOk(bool);
- newData(int), newData(double), … , newData(const QString&); (QTango 4.2.1)

QTWatcher: filter the updated value

- Modify the value read before invoking your slot or using your variable (***TValueFilter*** class)

```
class PlotLevelFilter : public TValueFilter
{
public:
   PlotLevelFilter(short int *imgDepth) :
      TValueFilter(),
      imageDepth(imgDepth)
      {};

   void filter(const TVariant& variant, int &intValue,
      bool read, State updateState)
      {
              if (*imageDepth == 16)
                      intValue = round(intValue/16);
      }

      short int* imageDepth;
};
```

QTWatcher: filter the updated value (II)

- install the implementation of ***TValueFilter***

```
QSlider *color_Slider = new QSlider(this);
QTWatcher *plotLevelWatcher = new QTWatcher(this);
plotLevelWatcher->attach(color_Slider, SLOT(setValue(int)));

PlotLevelFilter *plotLevelFilter = new
PlotLevelFilter(&imageDepth);
plotLevelWatcher->installRefreshFilter(plotLevelFilter);

 plotLevelWatcher->setSource(“a/b/c/PlotLevel");
```

QTWriter

- Write an attribute or give a command from any QObject or Qwidget;
- a *signal* of the QObject is connected to a compatible *execute()* method implemented in QTWriter;
- a *set point slot* can be provided to initialize the object with the current value at auto configuration time;
- data type automatically detected from the *signal* specified!

QTWriter

```
QLineEdit *lineEdit = new QlineEdit(this);
QTWriter *lineEditWriter = new QTWriter(this);

lineEditWriter->attach(lineEdit,
        SIGNAL(textChanged(const QString&)),
        SLOT(setText(const Qstring&)));

lineEditWriter->setTargets(“test/device/1/string_scalar”);
```

QTWatcher and QTWriter combined

- Create a QComboBox *reader/writer* in a few steps:

```
QComboBox *comboBox = new QComboBox(this);
QTWatcher *comboWatcher = new QTWatcher(this);
comboWatcher->attach(ui.comboBox,
    SLOT(setCurrentIndex(int)));
comboWatcher->setSource("$1/string_scalar");

QTWriter *comboWriter = new QTWriter(this);
comboWriter->attach(ui.comboBox,
SIGNAL(activated(int)));
comboWriter->setTargets(comboWatcher->source());
```