# 信号传递自定义结构

例子一：

1. 在sender头文件中声明新类型

typedef QMap<QString,QSharedPointer<QCPDataContainer<QCPGraphData > > > myMap;

1. 在sender类中添加信号

void SigClickOK(myMap vec);

1. 在Sender的构造函数中添加如下代码

qRegisterMetaType<myMap>("myMap");

1. 在sender触发事件时添加如下代码

QMap<QString,QSharedPointer<QCPDataContainer<QCPGraphData > > > mapReturn;

…

myMap mymap(mapReturn);

emit SigClickOK(mymap);

1. 在Receiver中声明接收的结构成员变量

QMap<QString,QSharedPointer<QCPDataContainer<QCPGraphData > > > mainMap;

6） 在Receiver中适当位置连接信号槽 connect(pSamEditDlg,SIGNAL(SigClickOK(myMap)),this,SLOT(OnSampleEditDlgAccept(myMap)));

7） 写Receiver的槽函数

void CBlockSamplePlotDialog::OnSampleEditDlgAccept(myMap curMap)

{

mainMap=curMap;

}

例子二：

1. 在sender头文件中声明新类型

typedef QVector<QVector<QStringList> > myVec;

typedef QPair<QString, double> PAIR;

1. 在sender类中添加信号

void finished1(myVec rec);

QVector<PAIR> vecIDConf;

1. 在Sender的构造函数中添加如下代码

qRegisterMetaType<myVec>("myVec");

1. 在sender触发事件时添加如下代码

…

myVec vec(vecShpRecords);

emit finished1(vec);

1. 在Receiver中声明接收的结构成员变量

QVector<QVector<QStringList> > vecShpRecords;

1. 在Receiver中适当位置连接信号槽connect(task,SIGNAL(finished1(myVec)),this,SLOT(finished1(myVec)),Qt::QueuedConnection);

7） 写Receiver的槽函数

void CQuaChkDialog::finished1(myVec rec)

{

vecShpRecords=rec;

}

# 多线程

1. 任务基类plasyntask.h

#ifndef PLASynTask\_H

#define PLASynTask\_H

#include <QObject>

class APP\_EXPORT PLASynTask : public QObject

{

Q\_OBJECT

public:

PLASynTask(QObject \*parent);

~PLASynTask();

//

// protected slots:

// virtual void doWork() = 0;

public slots:

virtual void execute();

signals:

void progressUpdate(QString msg, int maxValue, int curValue);

void finished();

};

#endif // PLASynTask\_H

1. 异步基类

#ifndef PLASynWorker\_H

#define PLASynWorker\_H

#include <QtCore>

class PLASynTask;

class APP\_EXPORT PLASynWorker : public QObject

{

Q\_OBJECT

public:

PLASynWorker(QObject \* parent = 0);

~PLASynWorker();

void stop();

void launchWorker(PLASynTask \*task);

private:

QThread\* mWorkThread;

signals:

};

#endif // PLASynWorker\_H

1. 实例化任务类头文件

#include "plasyntask.h"

#include "QgsProgressListener.h"

#include "qvector.h"

#include "qgsvectorlayer.h"

#include "ui\_QuaChkDialog.h"

#include "qmap.h"

#include "qvector.h"

//#include "qgisinterface.h"

#include "qmetatype.h"

#include "qmetaobject.h"

#include "qvariant.h"

#include "qpair.h"

typedef QVector<QVector<QStringList> > myVec;

typedef QPair<QString, double> PAIR;

class CPLATSCheckTask:public PLASynTask,public QgsProgressListener

{

Q\_OBJECT

public:

CPLATSCheckTask(QVector<QVector<QStringList> > vecRec,QgsVectorLayer\* pLayer,Ui::QuaChkDialog\* u,QMap<QString,int> map,QObject\* parent=0);

~CPLATSCheckTask(void);

virtual void updateProgress(int maxvalue, int curValue, QString msg = QString());

public slots:

virtual void execute();

void receiveProcess(int maxvalue, int curValue, QString msg = QString());

private:

QVector<QVector<QStringList> > vecShpRecords;

QgsVectorLayer\* pBlockLayer;

Ui::QuaChkDialog\* ui;

QMap<QString,int> mapNameID;

QMap<QString,double> mapIDConf;

QMap<QString,QString> mapIDType;

QVector<PAIR> vecIDConf;

signals:

void finished1(myVec rec);

void finished2(int a);

};

1. 实例化任务类源文件

#include "PLATSCheckTask.h"

#include <QThread>

#include <QtAlgorithms>

#include "qmessagebox.h"

CPLATSCheckTask::CPLATSCheckTask(QVector<QVector<QStringList> > vecRec,QgsVectorLayer\* pLayer,Ui::QuaChkDialog\* u,QMap<QString,int> map,QObject\* parent)

:PLASynTask(parent)

,vecShpRecords(vecRec)

,pBlockLayer(pLayer)

,ui(u)

,mapNameID(map)

{

qRegisterMetaType<myVec>("myVec");

}

struct CmpByValue

{

bool operator()(const PAIR& lhs, const PAIR& rhs)

{

return lhs.second < rhs.second;

}

};

CPLATSCheckTask::~CPLATSCheckTask(void)

{

}

void CPLATSCheckTask::execute()

{

emit progressUpdate(tr("准备查询..."), 0, 0);

myVec vec(vecShpRecords);

emit progressUpdate(tr("正在查询..."), 100,100);

emit finished1(vec);

emit finished();

}

void CPLATSCheckTask::updateProgress(int maxvalue, int curValue, QString msg)

{

emit progressUpdate(msg, maxvalue, curValue);

}

void CPLATSCheckTask::receiveProcess(int maxvalue, int curValue, QString msg)

{

emit progressUpdate(msg, maxvalue, curValue);

}

1. 调用类头文件

声明：PLASynWorker\* mWorker;

Private：

void showMessage(QString msg, int level = 0);

public slots:

void progressUpdate(QString msg, int maxValue, int curValue);

1. 调用类源文件

mWorker = new PLASynWorker(0);

CPLATSCheckTask\* task = new CPLATSCheckTask(vecShpRecords,pBlockLayer,ui,mapNameID);

connect(task, SIGNAL(progressUpdate(QString, int, int)), this, SLOT(progressUpdate(QString, int, int)));

connect(task, SIGNAL(finished1(myVec)), this, SLOT(finished1(myVec)),Qt::QueuedConnection);

connect(task,SIGNAL(finished()),this,SLOT(finished()));

mWorker->launchWorker(task);

void CQuaChkDialog::progressUpdate(QString msg, int maxValue, int curValue)

{

ui->progressBar->setMaximum(maxValue);

ui->progressBar->setValue(curValue);

showMessage(msg);

}

void CQuaChkDialog::showMessage(QString msg, int level)

{

if(level > 0)

{

QPalette pa;

pa.setColor(QPalette::WindowText,Qt::red);

ui->labelMessage->setPalette(pa);

msg = tr("错误:") + msg;

}

else

{

QPalette pa;

pa.setColor(QPalette::WindowText,Qt::black);

ui->labelMessage->setPalette(pa);

msg = tr("信息:") + msg;

}

ui->labelMessage->setText(msg);

}