Qt学习笔记 汪雄

积土成山，风雨兴嫣；积水成渊，蛟龙生焉

1. 设置QSlider的样式（滑动条的样式）采用样式表



出处：<http://blog.csdn.net/tax10240809163com/article/details/50899023>

源代码：

//首先是设置主体

QSlider{

border-color：#bcbcbc;

}

QSlider::groove:horizontal{

Border:1px solid #999999;

Height:1px;

Margin:0px 0;

Left:5px;

Right:5px;

}

//设置中间那个滑动的键

QString：：handle：horizontal{

Border：0px;

Border-image:ur(:/NickTalk/Images/Setting/volume\_nor.png);

Width:15px;

Margin:-7px -7px -7px -7px;

}

//还没有滑上去的地方

QSlider：：add-page:horizontal{

Background:qlineargradient(spread:pad,x1:0,y1:2,x2:0,stop:0 #bcbcbc,stop:0.25 #bcbcbc,stop:0.5 #bcbcbc,stop:1 #bcbcbc);

}

//已经划过去的地方

QSlider：：sub-page：borizontal{

Background:qlineargradient(spread:pad,x1:0,y1:1,x2:0,y2:0,stop:0 #439cf3,stop:0.25 #439cf3,stop:0.5 #439cf3,stop:1 #439cf3);

}

1. qt样式表选择器语法规则
2. QPushButton[text = “ok”]{

Color:red;

}

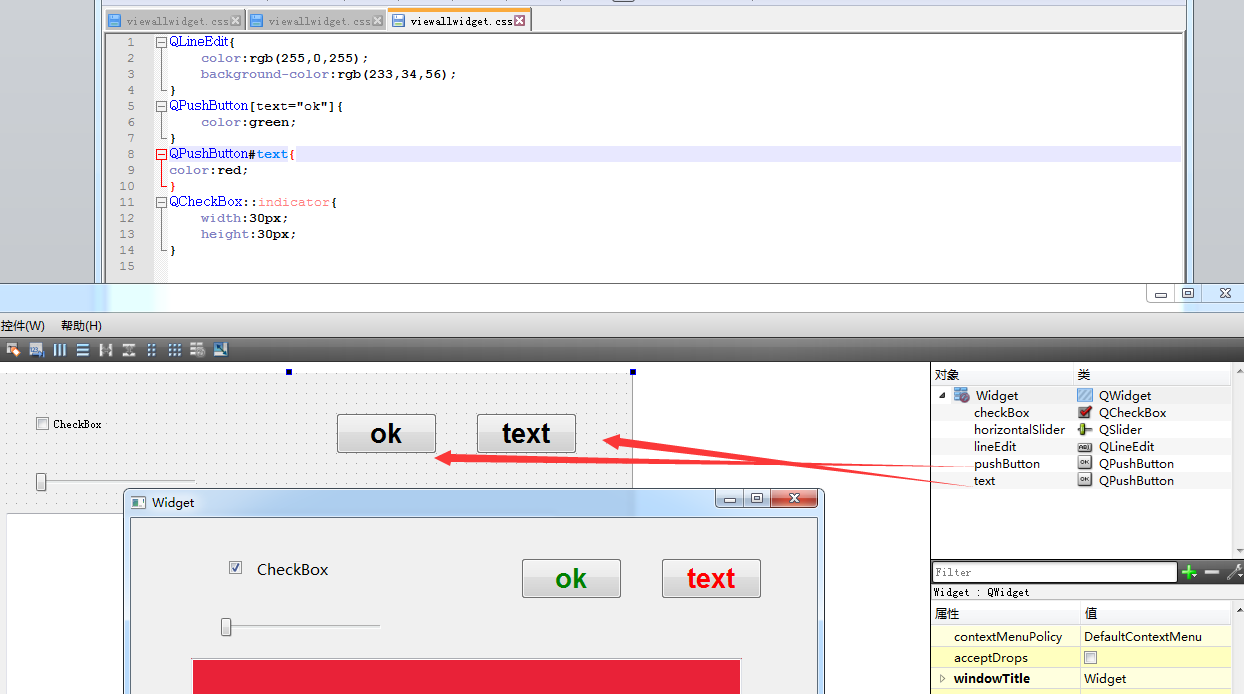
设置QPushButton的变量中，text内容为“ok”的PushButton的字体颜色

1. QPushButton#text{

Color:red;

}

设置QPushButton的变量中，对象名为text的PushButton的字体yanse



1. qt中设置字体

QFont ft;

ft.setFamily(“宋体”);

ft.setPixelSize(14);

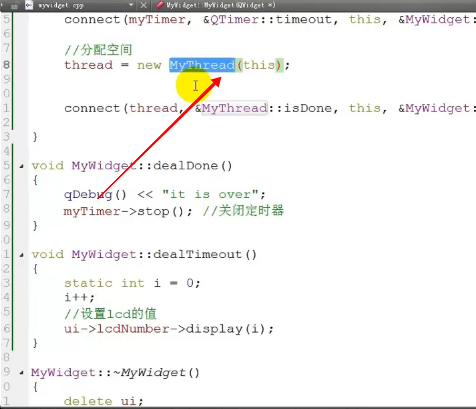
1. Qt中，QGraphicsScene中响应鼠标操作，对图像放大，或缩小
2. qt中三套坐标系统间的关系 QGraphicsItem，QGraphicsSence，QGraphicsView，中坐标关系
3. Qt中QLabel上的图片不现实的问题



解决办法：在创建QLabel用来显示图像的时候，指定父对象，图像就显示出来了。

1. qt4中线程处理函数

只能在虚函数void QThread::run()中定义，而在qt5中可以不一样。



线程关闭，terminal()，很暴力，坚决不要用，用quit()处理完后在关闭。

Void QWidget::stopThread()

{

Thread->quit(); //等待线程处理完手头工作

Thread->wait();

}

Qt5 新

1. 设定一个类，继承自QObject
2. 类中设置一个线程函数（只有一个线程函数）

Class MyThrea:public QObject

{

Void myTimer()

While(1)

{

Emit mySignal();

}

};

1. 创建线程对象（不能指定父对象）

MyThread \*myT；

myT = new MyThread;

1. QThread子线程对象

QThread \*thread = new QThread（this）指定父对象

1. 把自定义线程类，加入到子线程

myT->moveToThread(thread);

1. 启动子线程(只是启动了子线程，并没有启动线程处理函数)

thread.start()

1. 线程函数的启动，必须通过signal-slot来启动。

Connect(this, &startThread, myT, &myTimer);

emit startThread();

1. 子线程退出()

{

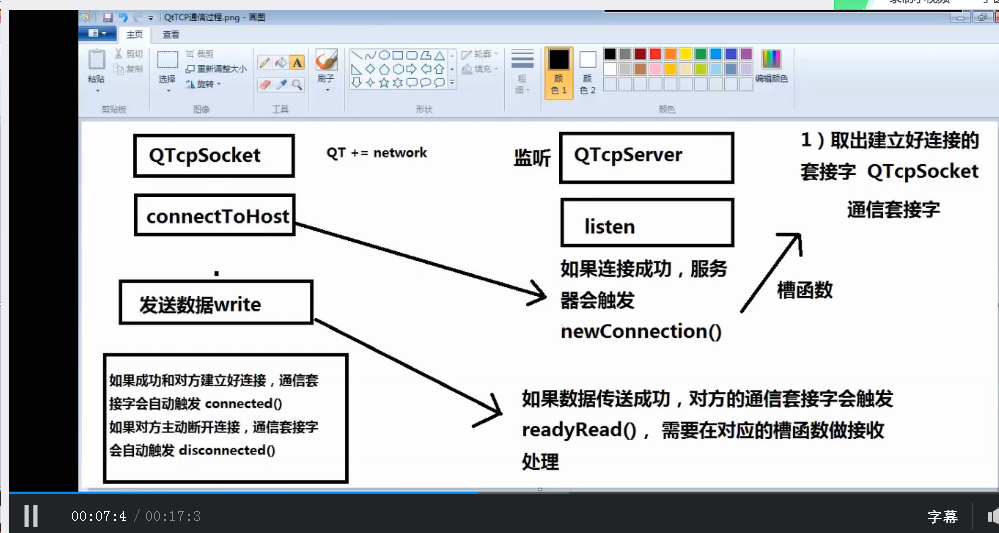
thread->quit();

thread->wait();

Delete thread;

}

1. qt中tcp通信



服务器端：

//监听套接字，制定父对象，让其自动回收空间

m\_tcpServer = new QTcpServer(this);

m\_tcpServer->listen(QHostAddress::Any, 8888);

//监听好

connect(m\_tcpServer, &QTcpServer::newConnection, [=]()

{

//取出建立好的套接字

m\_tcpSocket = m\_tcpServer->nextPendingConnection();

//获取对方的IP和端口

QString ip = m\_tcpSocket->peerAddress().toString();

qint16 port = m\_tcpSocket->peerPort();

QString temp = QString(“[%1:%2]:成功连接”).arg(ip).arg(port);

}

);

1. 数据库
2. **查看支持的数据库类型**

qDebug()<< QSqlDatabase::drivers();

**（2）打开数据库**

QSqlDatabase db = QSqlDatabase::addDatabase(“QMYSQL”);

db.setHostName(“127.0.0.1”);

Db.setUserName(“root”);

Db.setPassword(“”);//密码

Db.setDatabaseName(“”);//使用哪个数据库名称

If(db.open())

{

//打开数据库失败

QMessageBox::warning(this,”错误”,db.lastError().text() );

}

QSqlQuery query;

query.exec(“create table student(id int primary key ) ...”); //创建表

当需要对多个数据库进行操作的时候(区分不同的 db)

QSqlDatabase db1= QSqlDatabase：：addDatabase(“QMYSQL”,”a”);

QSqlQuery query(db1);

query.exec(“insert ... ”);

**（3）批量插入**

//obdc风格

query.prepare(“insert into student(name, age, score) values(?,?,?)”);

//给字段绑定特定值

QVariantList nameList；

nameList <<”小明”<<”晓龙”<<”小江”;

QVariantList ageList;

ageList << 11<<22<<33;

QVariantList scoreList;

scoreList << 59<< 69<<79;

//给定字段绑定相应的值，按顺序绑定

Query.addBindValue(nameList);

Query.addBindValue(ageList);

Query.addBindValue(scoreList);

Query.execBatch();

//oracle 风格

Query.prepare(“insert into student(name, age, score) values(:name, :age,:score)”);

Query.bindValue(“:name”, nameList);

Query.bindValue(“:score”,scoreList);

Query.bindValue(“:age”,ageList);

//执行预处理命令

Query.execBatch();

1. 数据库删除

// 获取行编辑内容

QString name = ui->lineEdit->text();

//开启一个事务

QSalDatabase::database().transaction();

QString str = QString

1. QSqlLite数据库

//添加Sqlite数据库

QSqlDatabase db = QSqlDatabase::addDatabase(“QSQLITE”);

db.setDatabaseName(“../info.db”);

If( !db.open() )

{

QMessageBox::waring(this, “错误”, db.lastError().text());

Return ;

}

QSqlQuery query;

Query.exec(“create table student(name, age, score)”);

//数据库遍历

Query.exec(“select \* from student”);

While( query.next())

{

qDebug() << query.value(0).toInt()

<< query.value(1).toString()

<< query.value(“age”).toInt()

<< query.value(“score”).toInt();

}

1. 数据库的可视化

//设置模型

QSqlTableModel \*model = new QSqlTableModel(this); //这个是如何和特定的db绑定的？

Model->setTable(“student”);// 指定使用哪个表

Ui->tableView->setModel(model);

Model->select();

Model->setHeaderData(0, Qt::Horizontal,”学号” );

Model->setEditStrategy(QSqlTableModel::OnManualSubmit);//手动提交修改

Ui->tableView->SetEditTriggers(QAbstractItemView::NoEditTriggers);//设置view中的数据不允许修改

//插入记录

#include<QSqlRecord>

QSqlRecord record = model->record();//获取空记录

Int row = model->rowCount();//获取行号

Model->insertRecord(row, record);

//此时在View中可以手动增加记录

//提交

Model->submitAll();

//取消

Model->reverAll();//取消所有动作

Model->submit();//

//ModelView中删除

//获取选中的模型

QItemSelectModel \*sModel = Ui->tableView->selectionModel();

//取出模型中的索引

QModelIndexList = sModel->selectedRows();

//删除所有选中的行

For(int i = 0; i < sModel.size(); i++)

{

Model->removeRow(list.at(i).row());

}

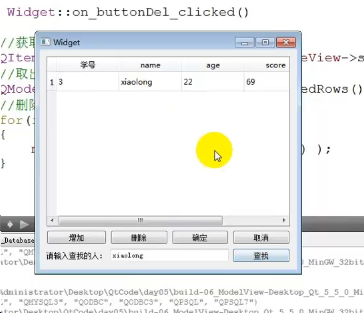
//ModelView 中选择

QString name = ui->lineEdit->text();

QString str = QString(“name = ‘%1’”).arg(name);

Model->setFilter(str);

Model->select();



1. Qt库QtPropertyBrowser库的编译

从官网上下载源代码，官网上没有找到，就去csdn上下载的。

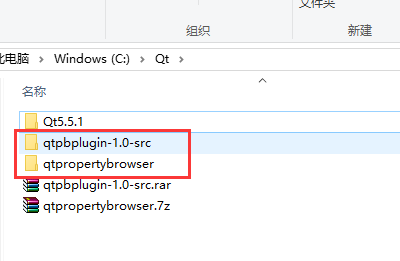
参考网友：<https://blog.csdn.net/yizhou2010/article/details/52994134>

下载src: <https://download.csdn.net/download/yizhou2010/9669754>

下载plugin: <https://download.csdn.net/download/yizhou2010/9669759>

编译过程：

1. 将下载得到的压缩包解压到，比如我这里解压到 C:\\Qt目录下



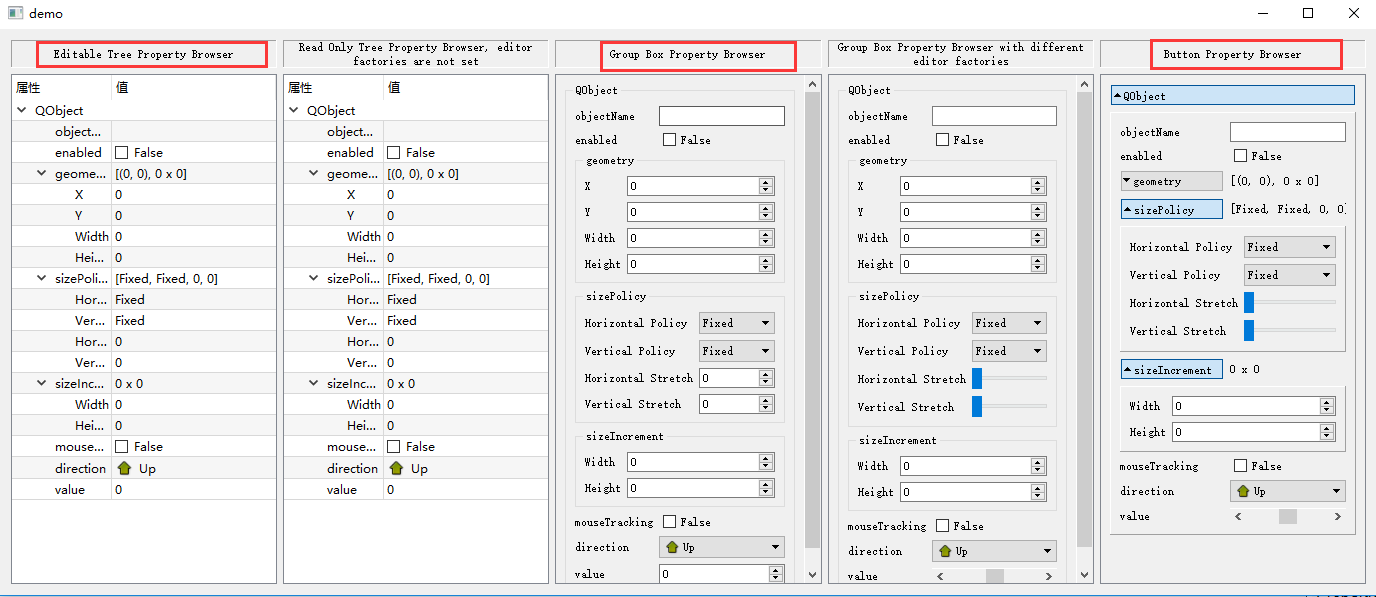
1. 进入到qtpropertybrowser这个文件夹下，用QtCreateor打开qtpropertybrowser.pro这个文件。在这个文件中加上这样的一句话

include("C:/Qt/qtpropertybrowser/src/qtpropertybrowser.pri")

就是将源代码的头文件和cpp文件加载进来

1. 进行qmake一下，就会在C:\Qt\qtpropertybrowser\lib下生成debug和release的lib和dll文件
2. 将这些lib和dll作为第三方库引用就可以了
3. QtPropertyBrowser类解析
4. 三种不同属性树的展现形式

TreeProperty、GroupBoxProperty、ButtonProperty的展现形式如下：



1. 往属性树种两种不同的方式添加item
2. 采用统一方式创建，并添加

添加 int 型

item = variantManager->addProperty(QVariant::Int, QString::number(i++) + QLatin1String(" Int Property"));

item->setValue(20);

item->setAttribute(QLatin1String("minimum"), 0);

item->setAttribute(QLatin1String("maximum"), 100);

item->setAttribute(QLatin1String("singleStep"), 10);

topItem->addSubProperty(item);

添加枚举类型

item = m\_pVarManager->addProperty( QtVariantPropertyManager::enumTypeId(), QStringLiteral("枚举类型"));

QStringList enumNames;

enumNames<<"Enum0"<<"Enum1"<<"Enum2";

item->setAttribute( QLatin1String("enumNames"), enumNames );

item->setValue(1);

groupItem->addSubProperty(item);

参考网友<https://www.cnblogs.com/foohack/p/4553802.html>

1. 采用子类化的方式创建

添加Int类型：

QtIntPropertyManager \*intManager = new QtIntPropertyManager(w);

QtProperty \*item9 = intManager->addProperty("value");

intManager->setRange(item9, -100, 100);

item0->addSubProperty(item9);

参考QtPropertyBrowse自带的demo