什么是QT

- QT是一个跨平台的C++图像用户界面应用程序框架
- QT在1991年由奇趣科技开发
- QT的优点
    - 跨平台,几乎支持所有平台
    - 接口简单，容易上手
    - 一定程度上简化了内存回收机制
    - 有很好的社区氛围
    - 可以进行嵌入式开发

QWidget

QT注意事项

- 项目名和路径
    - 名称不能有中文和空格
    - 路径不能有中文
- 类信息
    - 基类
        - QWidget(父类)
        - QMainWindow
        - QDialog
- 命名规范
    - 类名 首字母大写，单词和单词之间首字母大写
    - 函数名 变量名称 首字母小写,单词和单词之间首字母大写
- 快捷键
    - 注释 ctrl + /
    - 运行 ctrl + r
    - 编译 ctrl + b
    - 整行移动 ctrl + shift + ↑或↓
    - 查找 ctrl + f
    - 帮助文档 F1
    - 自动对齐 ctrl + i
    - 同名的.h和.cpp切换 F4

按钮

- 按钮常用API

1. show() 以顶层方式弹出窗口控件
2. setParent() 选择依赖方式
3. setText() 设置文本
4. resize() 重置窗口大小
5. move() 移动

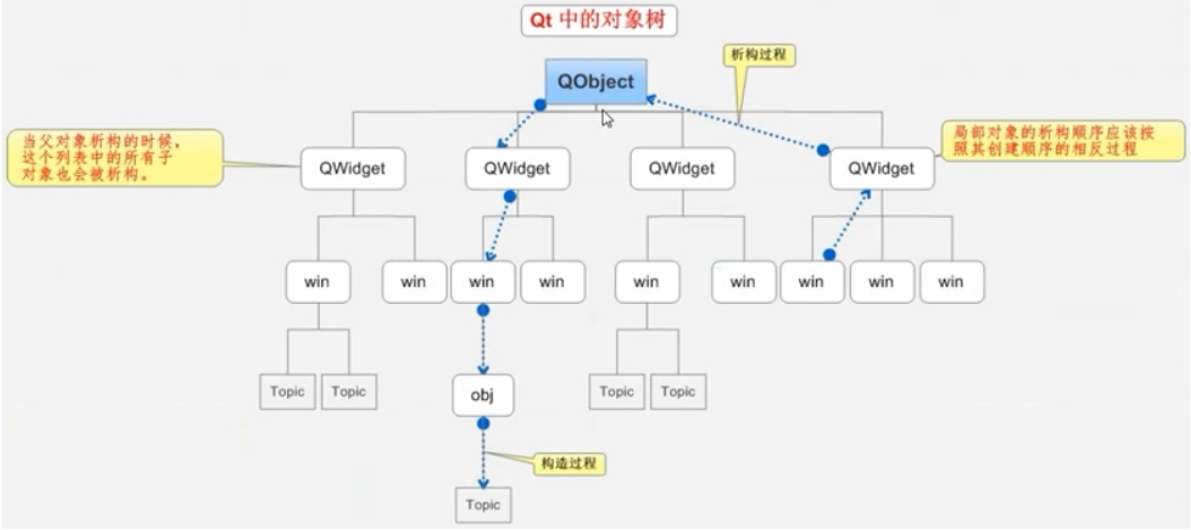
6. setWindowTitle() 设置窗口大小
7. setFixedSize() 设置固定窗口大小

```
MyWidget::MyWidget(QWidget *parent): QWidget(parent)
{
    //按钮控件
    QPushButton * btn = new QPushButton;
    //btn->show(); //以顶层的方式显示

    //如果想显示时候依赖在当前的窗口中 需要设置父窗口
    btn->setParent(this);

    //设置文本
    btn->setText("你好");

    //创建第二个按钮
    QPushButton * btn2 = new QPushButton("你好！",this);
    btn2->resize(100,30); //按钮也可以重置尺寸

    //移动位置
    btn2->move(100,0);

    //设置窗口的尺寸
    //this->resize(600,400);

    //设置固定窗口尺寸
    this->setFixedSize(600,400);

    //设置窗口标题
    this->setWindowTitle("第一个窗口");

}
```

QT中的对象树

1. 当创建的对象在堆区的时候,如果对象的父窗口是QObject 或QObject派生下来的类，
   此对象会放到对象树上，当程序执行完毕，树上的内容会从下往上依次释放。
2. 派生下来的类,可以不需要管理释放操作，会将对象放入对象树一定程度上简化了内存回收机制，

注意点：内存回收是从下往上回收，但是调用析构函数，是从上往下调用。



3. 测试流程，创建自定义的MyPushButton类(右击工程文件，添加新文件，选择C++ class)，基类选择QWidget控件，生成文件后，修改基类为QPushButton

```
#ifndef MYPUSHBUTTON_H
#define MYPUSHBUTTON_H

#include <QPushButton>

class MyPushButton : public QPushButton
{
    Q_OBJECT
public:
    explicit MyPushButton(QWidget *parent = 0);

    ~MyPushButton();

signals:

public slots:
};

#endif // MYPUSHBUTTON_H
```

```
#include "mypushbutton.h"
#include <QDebug>

MyPushButton::MyPushButton(QWidget *parent) : QPushButton(parent)
{

}


MyPushButton::~MyPushButton()
{
    qDebug() << "MyPushButton析构函数调用";
}
```

```
//自定义我自己的按钮，捕获析构函数
MyPushButton * myBtn = new MyPushButton;
myBtn->setParent(this);
myBtn->move(0,100);
myBtn->setText("我的按钮");
```

QT的窗口坐标系

1. 笛卡尔坐标系[左上角为0,0点]，x轴以左边为正，y轴以下面为正。

QT信号和槽

（day2工程文件）

- connect(信号的发送者（指针） ,信号的具体信息 （地址）, 信号的接受者（指针）,信号的处理[槽函数]（地址））
- QT5版本新语法——`connect(this->yc,&Teacher::hungry,this->st,&Student::treat);`
- QT4版本老语法——使用 `SIGNAL` 和 `SLOT` 宏`connect(teacher, SIGNAL(hungry(int)), student, SLOT(onHungry(int)));`
  - 优点：参数只管，写法简单；
  - 缺点：编译器不检测参数类型；
- 信号槽的优点 松散耦合
  - 信号发送端 和 接收端本身是没有关联的,通过connectl连接,将两者耦合在一起
  - 信号关键字：Signals
    - chlicked(bool) 点击
    - pressed() 按下
    - released() 释放
    - toggled(bool) 切换状态
  - 槽的关键字：Slots
- 自定义信号和槽位函数
  - 自定义信号
    - 写在类的signals下，不需要返回值，返回值为void；可以有参数，支持重载；**只需要声明，不需要实现**
  - 自定义槽函数
    - 不能写在signals下，需要写在public slots[公共的槽函数]下 (5.4版本以后可以写在全局函数、或者public、或者lambda)
    - 返回值也是void；**需要声明，也需要实现**；可以有参数，支持重载。
  - 然后用connect连接信号和槽
  - 触发信号关键字 emit
  - 信号和槽重载，需要函数指针，明确指向函数的地址
  - QString 转char * 使用.toUtf8().data()
  - 信号和槽连接：触发这个信号才能触发槽函数
    - 信号可以连接信号，触发一个信号也能触发另外一个信号
    - 一个信号可以连接多个槽
    - 多个信号也可以连接同一个槽函数
    - 信号和槽的参数类型 必须一一对应

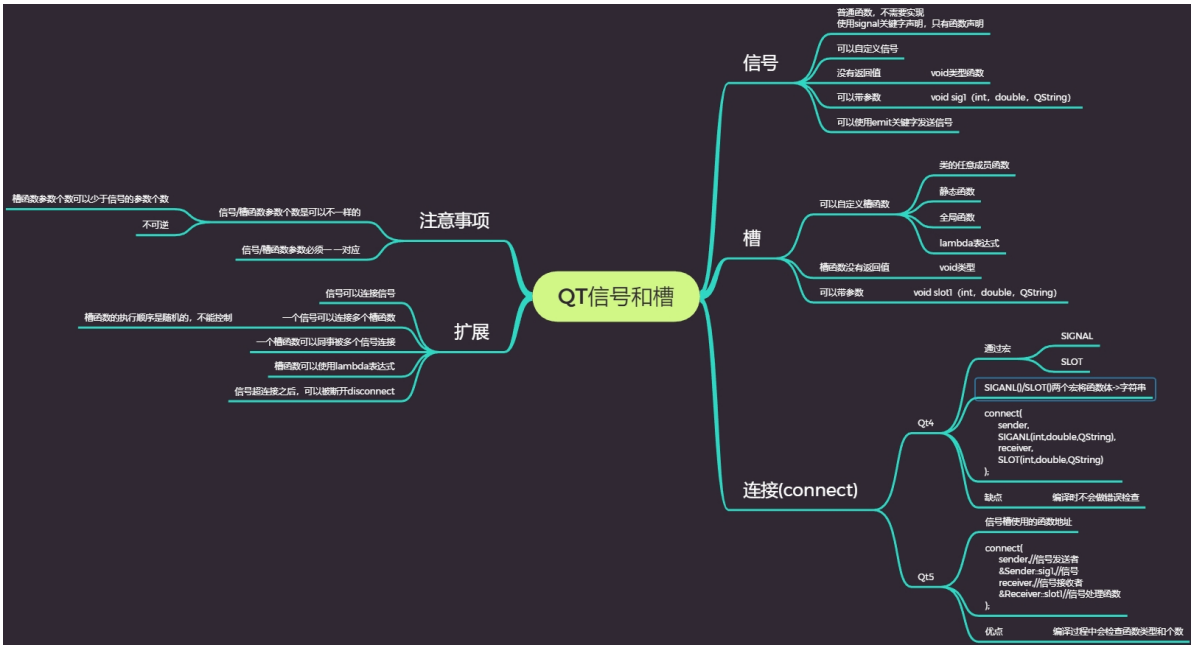
- 信号的参数个数可以多于槽的参数个数，反之不可以，相同个数的参数类型也要一一对应。
- 可以断开信号 disconnect(参数一样，怎么连的，怎么断开)

Lambda表达式

(day2工程文件)

- C++11版本特性 ,低版本qt需要在pro文件下加一个"CONFIG += c++11" 匿名函数对象
  - Lambda表达式函数声明 {}
    - []标识一个Lambda的开始，这部分必须存在不能省略。
    - [=] **用的最多**允许使用lambda所在作用范围内所有局部变量，并且是值传递方式
    - [&a] 允许使用引用传递变量a，一般不用，connect里会把函数指针设置为只读，所以通过引用修改变量会报错
    - [this]允许使用lamdba所在类中的成员变量
    - [变量] 允许变量使用值传递
    - mutable 可以省略，可修改值传递进来的拷贝，注意是能修改拷贝，而不是值本身
      - `[m]()mutable{m+=100;打印};` 不加mutable会报错
    - ->类型 带返回值
      - `int num = []()->int{return 1000}();`
  - Lambda表达式函数调用 `[](){}();`
  - 最常用的 `[=](){}`

day2总结



MainWindow

(day3示例)

QMainWindow是一个为用户提供主窗口程序的类，包含一个菜单栏（menu bar）、多个工具栏(tool bars)、多个锚接部件(dock widgets)、一个状态栏(status bar)及一个中心部件(central widget)，是许多应用程序的基础，如文本编辑器，图片编辑器等。

菜单栏 QMenuBar

```
//1、菜单栏 只能有一个
QMenuBar * bar = menuBar();
//菜单栏设置到窗口中
this->setMenuBar(bar);
//添加菜单
QMenu * fileMenu = bar->addMenu("文件");
QMenu * editMenu = bar->addMenu("编辑");
//添加菜单项
QAction * newAction = fileMenu->addAction("新建");
//添加分割线
fileMenu->addSeparator();
QAction * openAction = fileMenu->addAction("打开");
//菜单项中添加子菜单
QMenu * subMenu = new QMenu;
subMenu->addAction("子菜单1");
subMenu->addAction("子菜单2");
newAction->setMenu(subMenu);
```

工具栏 QToolBar

```
//2、工具栏 可以多个
    QToolBar * toolBar = new QToolBar(this);
    //将工具栏 设置到窗口中
    addToolBar(Qt::LeftToolBarArea,toolBar);
    //设置只允许左右停靠
    toolBar->setAllowedAreas(Qt::LeftToolBarArea | Qt::RightToolBarArea);
    //设置浮动
    toolBar->setFloatable(false);
    //设置移动
    toolBar->setMovable(false);
    //添加菜单项"新建"
    toolBar->addAction(newAction);
    //添加分割线
    toolBar->addSeparator();
    //添加菜单项"打开"
    toolBar->addAction(openAction);
```

状态栏 QStatusBar

```
    //3、状态栏 只能有一个
    QStatusBar * sBar = statusBar();
    //将状态栏 设置到窗口中
    setStatusBar(sBar);
    //添加一个标签信息
    QLabel * label1 = new QLabel("信息1",this);
    //标签加到左侧
    sBar->addWidget(label1);
    //标签加到右侧
    QLabel * label2 = new QLabel("信息2",this);
    sBar->addPermanentWidget(label2);
    //中间插一个信息 传入(位置+标签)
    QLabel * label3 = new QLabel("信息3",this);
    sBar->insertWidget(0,label3);
```

铆接部件 QDockWidget

```
    //4、铆接部件，浮动窗口，可以多个
    QDockWidget *dock = new QDockWidget("浮动窗口",this);
    //设置到父窗口中，必须要穿初始的位置再哪，帮助里有枚举值
    addDockWidget(Qt::BottomDockWidgetArea,dock);
    //设置只允许上和下停靠
    dock->setAllowedAreas(Qt::TopDockWidgetArea | Qt::BottomDockWidgetArea);
```

中心部件

```
    //5、核心部件，中心部件，只能有一个
    QTextEdit * textEdit = new QTextEdit("中心部件",this);
    //设置到父窗口中
    setCentralWidget(textEdit);
```

小总结

```
只能有一个的是set 可以允许多个是add
```

资源文件

(day4示例)

1. 将图片文件文件夹拷贝到项目下
2. 右键项目->添加新文件->Qt->Qt recourse File
3. res 生成 res.qrc
4. 右键res.qrc->open in editor 编辑资源
5. 添加前缀 添加文件
6. 使用 ":+ 前缀名 + 文件名"

对话框

```
//对话框
//点击新建，弹出对话框
connect (ui->actionNew_2,&QAction::triggered,[=](){
    //对话框分类

    //模态对话框 不可以对其他窗口进行操作      阻塞
    QDialog dlg(this);
    dlg.resize(200,50);
    dlg.exec();

    //非模态对话框      可以对其他窗口进行操作 不会阻塞
    QDialog * dlg1 = new QDialog(this);
    dlg1->resize(200,50);
    dlg1->show();
    //new在堆区开辟了内存就是在对象树上，关掉对话框不会释放内存，所以要设置属性，当对话框关闭的时候，要释放掉堆区的内存
    //设置属性   55号
    dlg->setAttribute(Qt::WA_DeleteOnClose);
    qDebug()<<"弹出对话框";


    //QMessageBox 消息对话框
    //错误提示
    QMessageBox::critical(this,"critical","错误！");

    //信息提示
    QMessageBox::information(this,"info","信息提示！");
```

```
    //询问提示
    //参数1 父窗口
    //参数2    标题
    //参数3    中间显示文本
    //参数4    按键类型
    //参数5    关联回车默认值
    QMessageBox::question(this,"ques","询问！",QMessageBox::Save |
QMessageBox::Cancel,QMessageBox::Cancel);

    if(QMessageBox::Save == QMessageBox::question(this,"ques","询
问！",QMessageBox::Save | QMessageBox::Cancel,QMessageBox::Cancel))
    {
        qDebug<<"选择的是保存";
    }
    else
    {
        qDebug<<"选择的是取消";
    }

    //警告提示
    QMessageBox::warning(this,"warning","警告！");

    //颜色对话框
    QColor = QColorDialog::getColor(QColor(255,0,0));
    qDebug()<<color.red() <<color.green() <<color.blue();

    //字体对话框
    bool ok;
    QFont font= QFontDialog::getFont(&ok,QFont("华文彩云",36));
    qDebug()<<"字体："<<font.family() << "字号："<< font.pointSize()
            <<"加粗："<<font.bold() <<"倾斜："<< font.italtic();

    //文件对话框
    QString str = QFileDialog::getOpenFileName(this,"打开文
件","C:\\Users\\11411\\Desktop","(*.txt *.doc)");
    qDebug()<<str;
```

- 模态对话框 不可以对其他窗口进行操作
    - QDialog dlg(this);
    - dlg.exec();
    - 消息对话框
        - 错误对话框 QMessageBox::critical(this,"critical","错误");
        - 信息对话框 information
        - 提问对话框 question
        - 警告对话框warning
        - 颜色对话框
            - QColor a = QColorDialog::getColor(QColor(255,0,0));
        - 文件对话框 最后一个是过滤

        - QString str = QFileDialog::getOpenFileName(this,"打开文件","./","(*.cpp)");
    - 字体对话框
        - bool flag;
        - QFont font = QFontDialog::getFont(&flag,QFont("华文彩云",12));
        - setFont(font);//设置字体
- 非模态对话框 可以对其他窗口进行操作
    - QDialog *dlg2 = new QDialog(this); //为了确保不释放,开在堆上
    - dlg2->show();
    - dlg2->setAttribute(Qt::WA_DeleteOnClose);//55号 用于按关闭键自动释放[QWidge的对象树是在关闭总的窗口才会全部释放]

ui窗口自布局

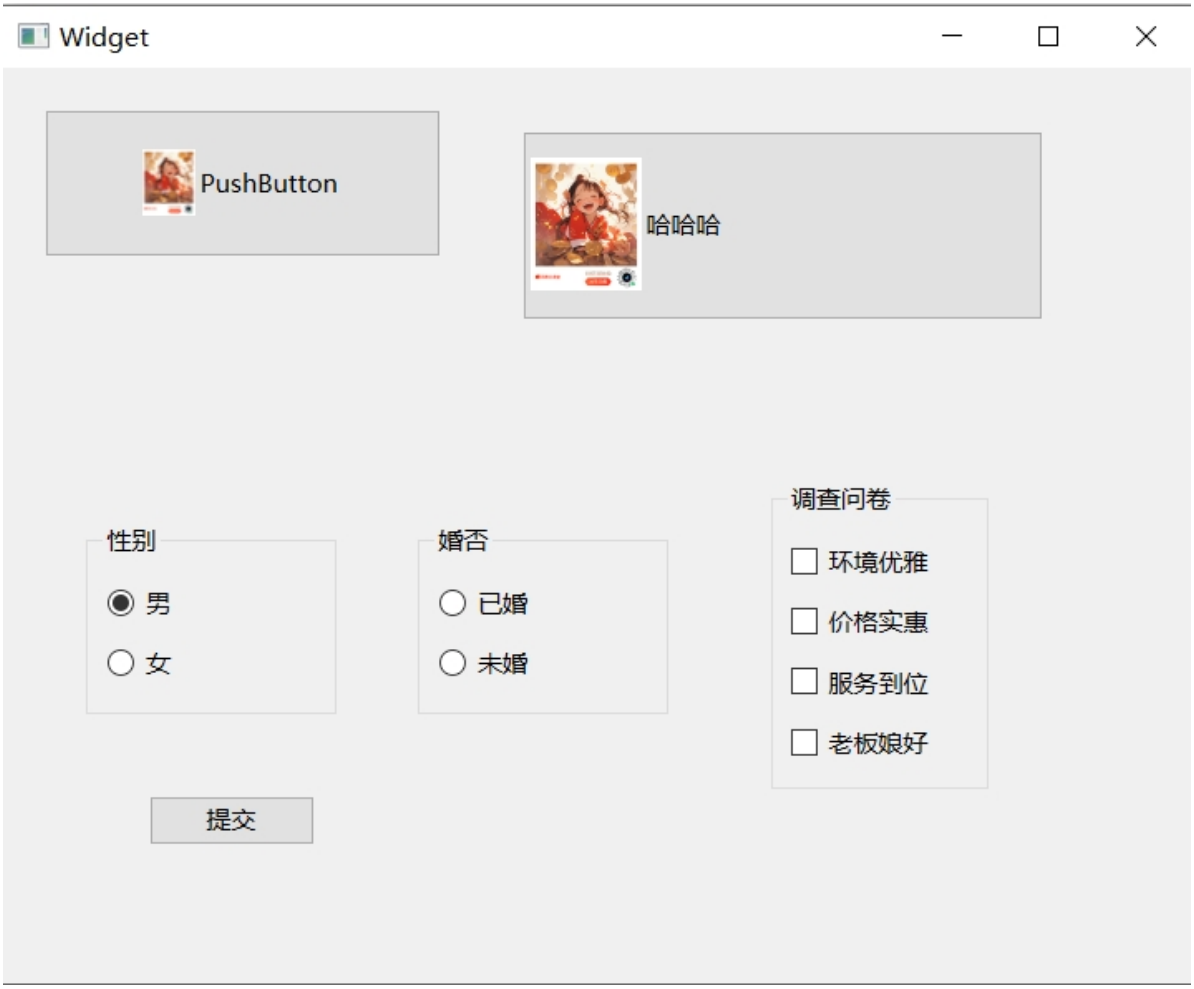
(day5)

1. 利用Widget 做布局：水平、垂直、栅格
2. Spacers 弹簧 进行设置对齐和间隔
3. widget窗口可以设置最大最小窗口来固定窗口不允许拖拽变化
4. 主窗口设置垂直布局后可以在sizePolicy->垂直策略->Fixed来使组件高度合适

常用的按钮控件

(day6)

图标直接选中icon就可以，然后toolbutton可以选择风格，还有透明选中突起的效果等




```
//默认男被选中
```

```
//ui->rbt_man->setChecked(true);
//    //监听是否选中了女的
//    connect(ui->rbt_woman,&QRadioButton::clicked,this,[=](){
//        qDebug()<<"选择了女的";

//    });
    //选择了男
    connect(ui->rbt_man,&QRadioButton::clicked,this,[=](){
        this->sex=1;//sex变量需要在widget.h头文件的类属性里面声明
});

//选择了女
connect(ui->rbt_woman,&QRadioButton::clicked,this,[=](){
    this->sex=0;

});

//提交打印结果
connect(ui->pbt_tj,&QPushButton::clicked,this,[=](){
    if(this->sex==0){
        qDebug()<<"选择了女";
    }
    else
    {
        qDebug()<<"选择了男";
    }

});

//复选框
//判断选中不是clicked是stateChanged
ui->checkBox->setTristate(true);//半选状态,也可以在ui界面调整找到tristate属性
connect(ui->checkBox,&QCheckBox::stateChanged,[=](int state){
    qDebug()<<state;//选中是2，半选是1，不选是0
});
```

列表控件 listWidget

(day7)

- QListWidgetItem * item = new QListWidgetItem("锄禾日当午");
- ui->listWidget->addItem(item); //添加进去
- item->setTextAlignment(Qt::AlignCenter); //居中

```
    //listwidget使用

//    QListWidgetItem * item =new QListWidgetItem("锄禾日当午");
//    ui->listWidget->addItem(item);
//    //垂直和水平居中
//    item->setTextAlignment(Qt::AlignVCenter | Qt::AlignHCenter);
    //如果没有对齐的需求，可以使用下面容器的方式，将所有的内容添加进去
    //QStringlist -> QList<QSring> -> list<string> 容器
    QStringList list;
    list<<"锄禾日当午"<<"汗滴禾下土"<<"谁知盘中餐"<<"粒粒皆辛苦";
    ui->listWidget->addItems(list);
```

树控件TreeWidget

(day8)

```
    //treeWidget 树控件
    //1、设置头的标签
    ui->treeWidget->setHeaderLabels(QStringList()<<"英雄"<<"英雄介绍");
    //2、Item创建
    QTreeWidgetItem * liItem = new QTreeWidgetItem(QStringList()<<"力量");
    ui->treeWidget->addTopLevelItem(liItem);

    QTreeWidgetItem * minItem = new QTreeWidgetItem(QStringList()<<"敏捷");
    ui->treeWidget->addTopLevelItem(minItem);

    QTreeWidgetItem * zhiItem = new QTreeWidgetItem(QStringList()<<"智力");
    ui->treeWidget->addTopLevelItem(zhiItem);

    //3、创建子Item 挂载到顶层德Item上
    QStringList l1;
    l1<<"洛克萨斯之手"<<"前排战士，会使用无情铁手和洛克萨斯断头台";
    QTreeWidgetItem * L1 = new QTreeWidgetItem(l1);
    liItem->addChild(L1);
```

表格控件


```
    //tableWidget的使用
    //1、设置列数
    ui->tableWidget->setColumnCount(3);
    //2、设置水平表头的标签
    ui->tableWidget->setHorizontalHeaderLabels(QStringList()<<"姓名"<<"性别"<<"年
龄");
    //3、设置行数
    ui->tableWidget->setRowCount(5);
    //4、设置每个Item里的数据(0,0)是表格第一列第一行，参数有提示
//    ui->tableWidget->setItem(0,0,new QTableWidgetItem("亚瑟"));
    //for循环传进去
    QStringList nameList;
    nameList <<"亚瑟"<<"妲己"<<"韩信"<<"孙悟空"<<"梦琪";

    QList<QString> sexList;
    sexList<<"男"<<"女"<<"男"<<"男"<<"女";

    for(int i = 0; i<5 ; i++)
    {
        int col = 0;
        ui->tableWidget->setItem(i,col++,new QTableWidgetItem(nameList[i]));
        ui->tableWidget->setItem(i,col++,new QTableWidgetItem(sexList.at(i)));
        //int 转 QString QString::number(int)
        ui->tableWidget->setItem(i,col++,new
QTableWidgetItem(QString::number(18+i)));
```

```
    }

    //5添加和删除
    //点击添加赵云，实现添加
    connect(ui->btn_add,&QPushButton::clicked,this,[=](){
        //如果有赵云了，就报警，不添加
        bool isEmpty = ui->tableWidget->findItems("赵
云",Qt::MatchExactly).isEmpty();
        if(!isEmpty)
        {
            QMessageBox::warning(this,"警告","已经有赵云了，添加失败");
        }
        else
        {
            ui->tableWidget->insertRow(0);//添加一行
            ui->tableWidget->setItem(0,0,new QTableWidgetItem("赵云"));
            ui->tableWidget->setItem(0,1,new QTableWidgetItem("男"));
            ui->tableWidget->setItem(0,2,new QTableWidgetItem("29"));
        }
    });

    //点击删除赵云，实现删除
    connect(ui->btn_del,&QPushButton::clicked,this,[=](){
        //如果有赵云了，就报警，不添加
        bool isEmpty = ui->tableWidget->findItems("赵
云",Qt::MatchExactly).isEmpty();
        if(isEmpty)
        {
            QMessageBox::warning(this,"警告","没有赵云，删除失败");
        }
        else
        {
            int rowNum = ui->tableWidget->findItems("赵
云",Qt::MatchExactly).first()->row();
            ui->tableWidget->removeRow(rowNum);
        }
    });
```

其他控件

1. stacked widget

```
Widget::Widget(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Widget)
{
    ui->setupUi(this);

    //stacked widget

    //设置默认选中第一个
    ui->stackedWidget->setCurrentIndex(0);
    connect(ui->btn_Scroll,&QPushButton::clicked,[=](){
        //设置当前索引
        ui->stackedWidget->setCurrentIndex(0);
    });

    connect(ui->btn_Tool,&QPushButton::clicked,[=](){
        ui->stackedWidget->setCurrentIndex(1);
    });

    connect(ui->btn_Tab,&QPushButton::clicked,[=](){
        ui->stackedWidget->setCurrentIndex(2);
    });
}
```

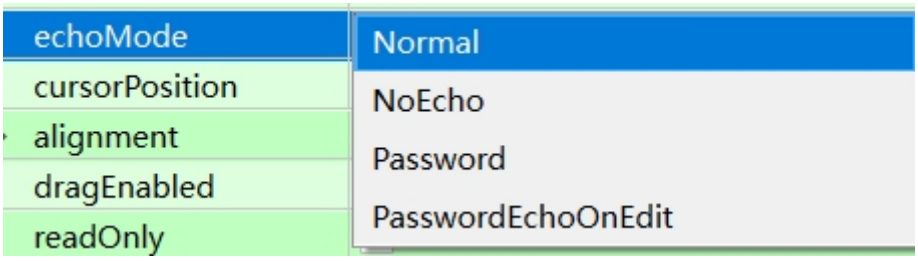
2. scroll area 滚动条
3. tool box 类似qq好友列表分类
4. tab widget 类似网页切换
5. frame 可以有自己边框的样式
6. combo box 下拉框

```
//combo box下拉框
ui->comboBox->addItem("奔驰");
ui->comboBox->addItem("宝马");
ui->comboBox->addItem("保时捷");
//点击保时捷 定位到对应的选项
connect(ui->btn_select,&QPushButton::clicked,[=](){
    //ui->comboBox->setCurrentIndex(2);
    ui->comboBox->setCurrentText("保时捷");
});
```

```
//可以通过currentInde()和currentText()成员函数，查看下拉框具体选中了哪一个选项的索引和文本
connect(ui->cBox_connection,&QComboBox::currentIndexChanged,this,[=](){
        int index = ui->cBox_connection->currentIndex(); // 获取当前选中项的索引
        QString text = ui->cBox_connection->currentText(); // 获取当前选中项的文本

        qDebug() << "选中的索引：" << index;
        qDebug() << "选中的文本：" << text;
    });
```

7. font combo box 字体下拉框
8. line edit 单行文本框

   密码输入可以设置echomode属性为linux类似的无文字，和密码的形式



9. text edit 和 plain text edit 多行文本，第一个可以设置字体属性，第二个就是纯文本，只用第一个，第二个连setText()都没有

```
connect(ui->btn_clear,&QPushButton::clicked,this,[=](){
    ui->textEdit_send->setText("数据发送：\n");
    ui->textEdit_receive->setText("数据接收：\n");
});
```

10. spin box 和 double spin box 可以输入数字外加上下按钮，double的是小数
11. time edit 和 date edit 和 date/time edit 时间日期的控件
12. horizontal scroll bar 和vertical scroll bar 和 horizontal slider 和 vertical slider 水平和垂直滑动条
13. label标签可以显示图片，还可以显示动图，属性里没有icon选项，所以需要代码设置

```
//利用QLabel显示图片
QPixmap pix;
pix.load(":/Image/butterfly.png");
ui->label_Img->setPixmap(pix);
ui->label_Img->setFixedSize(pix.width(),pix.height());

//利用QLabel显示gif动图
QMovie * movie = new QMovie(":/Image/mario.gif");
ui->label_movie->setMovie(movie);
movie->start();

movie->setSpeed(300);

connect(movie,&QMovie::frameChanged,[=](int frameId){
    if(frameId == movie->frameCount() -1)
    {
        movie->stop();
    }
});
```

需要添加头文件movie

这个是老师发现的可以播放一边自动停止的代码

加速播放300是3倍速

主要逻辑是：获取播放的帧为最后一帧时就停止播放

自定义控件的封装

QT事件 QEvent

-
  - 鼠标事件

    事件是虚函数,可以进行重载

```
//鼠标进入事件
virtual void enterEvent(QEvent *event);
//鼠标离开事件
virtual void leaveEvent(QEvent *event);
//鼠标按下
virtual void mouseReleaseEvent(QMouseEvent *ev);
//鼠标释放
virtual void mousePressEvent(QMouseEvent *ev);
//鼠标移动
virtual void mouseMoveEvent(QMouseEvent *ev);
```

- 定时器 QTimeEvent
  - 利用事件实现定时器
    - startTimer(1000); 启动定时器，单位毫秒,返回一个唯一定时器id
    - void timerEvent(QTimerEvent * ev)
      - 定时器函数,可以通过ev->timerId()== id1来判断当前是哪个id进来的
  - 定时器类QTimer
    -

```
//通过定时器类
QTimer * timer = new QTimer(this);
//启动定时器 每隔500秒发一个信号
timer->start(500);
//连接信号
connect(timer,&QTimer::timeout,中括号小括号{
static int num = 1;
ui->label_5->setText(QString::number(num++));
});
```

- event事件分发器
  - bool event(QEvent * ev)
    - 返回值是bool类型，如果返回true，代表用户要处理这个事件,不向下分发事件了[类似于钩子]
  - 事件枚举QEvent
    - ev.type();
    - 拦截后使用子类的操作可以使用静态类型转换
      - QMouseEvent *ev = static_cast<QMouseEvent *>(QEvent中行参);
  - 但是尽量别拦截
- 事件过滤器
  - 在app到事件分发器前还能做个过滤
  - 使用方式

        - 给控件安装时间过滤器
            - installEventFilter(this);
        - 重写eventfilter事件

绘图 QPainter

- 绘图事件 void paintEvent(QPaintEvent *)
- 画家类 QPainter(构图的设备)
    - 拿起笔 .setPen(笔)
    - 拿起刷子 .setBrush(刷子)
- 画笔类 QPen(笔的颜色)
- 画刷类 QBrush(笔的颜色)
- 高级操作
    - 效率降低的抗锯齿
        - painter.setRenderHint()
    - 改变画家位置
        - painter.save();保存当前位置
        - painter.restore(); 还原到保存的位置
        - painter.translate(); 移动画家
    - 画家绘制图片drawPixmap

绘图设备

- QPixmap 专门对图像显示做了优化
- QBitmap 色深限定为1
- QImage 专门为图像的像素级访问做了优化
- QPicture 可以记录和重视画家的QPainter的各类命令
    - 自定义绘图操作

文件读写 QFile

- file.open(打开方式) QtODevice::readOnly
- 全部读取 file.readAll() 按行读 file.readLine() 判断文件末尾atend()
- QFile默认支持的是utf-8 指定格式 QTextCodec
    - QTextCodec *codec = QTextCodec::codecForName("gbk");
    - ui->textEdit->setText(codec->toUnicode(array));
- 关闭文件对象 file.close();

文件信息 QFileInfo

- QFileInfo info(path);
- 后缀名 info.suffix()
- 创建日期 info.birthTime().toString("yyyy/MM/dd hh:mm:ss");
- 修改日期 info.lastModified().toString("yyyy/MM/dd hh:mm:ss");

Qss 前端人狂喜

- #myButton 这里的id实际上就是objectName指定的值
- 伪状态
    - :active 当小部件驻留在活动窗口中时，将设置此状态
    - :checked　该控件被选中时候的状态
    - :hover 鼠标在控件上方
    - :pressed 该控件被按下时的状态
    - :disabled　该控件禁用时的状态
    - :first　该控件是第一个（列表中）
    - :focus 该控件有输入焦点时

动画 QPropertyAnimation

```
//winLabel 你要对那个组件使用动画　geometry几何结构
QPropertyAnimation * an = new QPropertyAnimation(winLabel,"geometry");
//动画时间
an->setDuration(1000);
//动画开始
an->setStartValue(QRect(winLabel->x(),winLabel->y(),winLabel->width(),winLabel-
>height()));
//动画结束
an->setEndValue(QRect(winLabel->x(),winLabel->y() + 300,winLabel-
>width(),winLabel->height()));
//动画方式
an->setEasingCurve(QEasingCurve::OutBounce);
an->start();
```

背景音乐 QSound

- qmake: QT += multimedia
- QSound * startSound = new QSound(":/res/TapButtonSound.wav",this); 载入音效
- startSound->play(); 播放
- startSound->setLoops(-1); -1循环次数无限

打包发布

- debug->release
- 运行 运行失败添加环境变量D:\QT\5.12.3\mingw73_64\lib
- 把 Goldreverse.exe 单独丢到一个文件夹下
- cmd中路径后windeployqt .\Goldreverse.exe 运行
- 此时已经可以使用了
- 深入打包[hm nis edit][https://www.bilibili.com/video/BV1g4411H78N?p=63&spm_id_from=pageDriver]
- HM NIS Edit 和 NSIS

案例:翻金币

- 收获
    1. 删除资源文件后需要删除debug文件,不然会报错
    2. 界面的切换可以使用信号和槽 即其它界面emit发送一个信号,主界面接收
        - 当然也可以选择记录父类指针,但是必须要在构造函数中多传个参数，而不是使用默认的parent
    3. 在按钮上方有其他组件，可以使用label->setAttribute(Qt::WA_TransparentForMouseEvents);让其可以点到按钮[51号属性]
    4. 界面翻转金币 本质上是个按钮
        - 人点击后
        - 金币触发翻转
        - 定时器每隔30ms发送一次信号给金币
        - 金币触发图片重新放置,到最大值或者最小值的时候关闭定时器
        - 金币中有坐标i 和 j 以及一个flag 来确定该金币在页面中的位置
    5. 锁定窗口 m_chooseScence->setGeometry(this->geometry()); 每次进入或者退出都锁定他的位置
- 延时器
  QTimer::singleShot(毫秒,拉姆达表达式);

[https://www.bilibili.com/video/BV1g4411H78N?p=63&spm_id_from=pageDriver]: