能在Qt UI 界面设置好的都在该界面设置好，在该界面完不成的再用代码解决

它的大部分样式都是在代码里面写的，可以在designer里面改的是有限的

**功能先行，UI其从**

**重功能、轻UI、强进度**

NewCourierItem

Hover

#40C0A0

#A0A0A0

## 界面尺寸集合

1400\*800

100、450、850

850里面margin为25、25，因此，EventItem为800\*100

800里面是150+50+500+100

ui->eventNameLabel->move**(30,12**);

ui->eventDescriptionLabel->move(**30,42**);

EditDialog是500\*800

## 设计构想

做一个设置，以方便用户自定义一些内容，包括清单的“隐藏列表”、“24小时制”、一周开始日期之类的，

右键

快捷键

最后再写事件监听的

## 进度

Spinbox没改变，点了这个月的其它天

两个show，checkdate和今天之间计算差值

点了按钮后checktoday，两个hide

只是spinbox改变的时候，显示今天的按钮

按下的时候把spinbox调回这个月，按钮Hide

Years fly by, but the heart stays in the same place.时光飞逝，但我心依旧。

― Harlan Coben, [Promise Me](https://www.goodreads.com/work/quotes/14596742)

A man who dares to waste one hour of time has not discovered the value of life.敢于浪费一个小时的人，没有发现生命的价值。——达尔文

Your time is limited, so do*n't waste it living som*eone else's life.

QStringList mottos={"Your time is limited, so \n"

" don't waste it living \n"

" someone else's life.",

"A man who dares to waste \n"

" one hour of time has not \n"

" discovered the value of life.",

"Years fly by, but the heart \n"

" stays in the same place.",

"I believe every human has a finite \n"

" number of heartbeats. I don't \n"

" intend to waste any of mine.",

"Every moment is a\n"

" fresh beginning.",

"The people who are crazy enough \n"

" to think they can change the \n"

" world, are the ones that do.",

"When you cease to dream \n"

" you cease to live",

"The best way to predict your \n"

" future is to create it.",

"Make each day your masterpiece.",

"You can’t put a limit on "

" anything. The more you "

" dream, the farther you get."};

QStringList celebrities={"- Steve Jobs",

"- Charles Darwin",

"- Harlan Coben",

"- Neil Armstrong",

"- T.S. Eliot",

"- Steve Jobs",

"- Malcolm Forbes",

"- Abraham Lincoln",

"- John Wooden",

"- Michael Phelps"};



QTest mouseClick()

Parent Widget / Child Widget，在父Widget中的子类Widget就叫Child Widget

# EditDialog

EventBaseWidget只有EventFull的数据，不含EditDialog的指针。它是直接全局访问到EditDialog的。

点击某个EventBaseWidget对象时，该对象把其含有的EventFull数据传给全局的EditDialog

QRect 是Rectangle(int x, int y, int width, int height)

QPoint 是(int x, int y)

对一个QWidget设置setGeometry()可以设置其QRect，geometry()可以拿到其QRect

绝对位置可以用QPoint()来设置

QString中isEmpty() 和isNull() 是一样的，推荐有isEmpty()

Cursor可以设置鼠标移动到widget上呈现出来的样式

## 技术问题

怎么调用Window的通知？

不同的派生类的大小不一样，先测试出最大的派生类，然后用最大的派生类去初始化文件。

## 提示和警告

1. 对象都是在ui里面的，要加上ui->

只在ui界面改动再编译运行是无效的，必须要有代码的改动才会产生新的运行结果。

多用Ctrl B 进行构建来及时debug

需要全局各组件都要访问的变量和对象直接写在全局变量里面（写在一个各文件都包含的基文件中）。

**当涉及到又读又写的时候，在一次读写完成之后要seek(0)，把指针调回0位置。这点很重要！！！**

**对于字体大小，分清楚pt（点大小）和px（像素大小）**

# 综合

TextField是单行，TextArea是多行

QPushButton的stateChanged就是toggled

Toggle之后**自己判断自己是不是check**，**自己判断是不是今天**，再加上eventbool表

void QAbstractButton::**toggled**(bool checked)

This signal is emitted whenever a checkable **button changes its state**. checked is true if the button is checked, or false if the button is unchecked.

saveBtn->setEnabled(true);

saveBtn->setChecked(true); （这个Checked是被选中）

区分清楚Checked和Clicked，toggled的区别

QScrollBar的位置获取和设置

Scrollbar->sliderPosition(); int类型

Scrollbar->setSliderPosition(int);

**要有一个全局的思想，一些变量访问不到的，就放到全局里面去**。这是一个解决问题的好办法！！

并不是所有组件在初始化的时候都要用指针去写，有些情况用指针去写反而是糟糕的。比如要设置它的样式表的时候（这是经常需要做的），那么用指针去写就获取不了其对象名而难以对其进行特定地样式设置。

## 给QLabel或QWidget设置背景

ui->eventIcon->setStyleSheet("border-image:url(:/icon/badge\_green.png)");

样式表 **border-image:url()**

## QPushButton

给QPushButton加图标

image:url(":/delete.png");

background-image:url(":/delete.png");

按钮禁用时，设置样式

background-image或者border-image，（前者是按像素显示图片，后者可根据窗体大小来自动缩放图片）

QPushButton:disabled {

/\*设置禁用时按钮的样式\*/

按钮选中时，设置样式 前提是setCheckable(true)

QPushButton:checked {

/\*设置选中时按钮的样式\*/

}

QPushButton:!checked {

/\*设置未选中时按钮的样式\*/

}

按钮选中并且禁用时，设置样式

QPushButton:checked:disabled {

/\*设置选中并且禁用时按钮的样式\*/

}

点击按钮时，设置样式

QPushButton:pressed {

/\*设置点击按钮时按钮的样式\*/

}

鼠标悬浮在按钮上时，设置样式

QPushButton:hover {

}

# QIcon

QString normalStr = "E:/work/xxx/src/creator/resource/images/edit/effect/btn\_text1\_press.png";

QString selectStr = "E:/work/xxx/src/creator/resource/images/edit/effect/btn\_text2\_press.png";

QString activerStr = "E:/work/xxx/src/creator/resource/images/edit/effect/btn\_text6\_press.png";

QString disableStr = "E:/work/xxx/src/creator/resource/images/edit/effect/btn\_text7\_press.png";

funcIcon.addPixmap(QPixmap(normalStr), QIcon::Normal, QIcon::On);

funcIcon.addPixmap(QPixmap(selectStr), QIcon::Selected, QIcon::On);

funcIcon.addPixmap(QPixmap(activerStr), QIcon::Active, QIcon::On);

funcIcon.addPixmap(QPixmap(disableStr), QIcon::Disabled, QIcon::On);

ui->pushButton->setIcon(funcIcon); //no select stauts //item has

toolBtn->setIcon(funcIcon); // no select active status

# QLabel

Tooltip是鼠标停在上面可以看见的

statusTip状态提示就是在状态栏中提示，当鼠标移动到widget上时，在状态栏显示的提示信息。

accessibleDescription和accessibleName属性都是用于残疾人辅助阅读的

selectionChanged

textEdit() 不包括被程序代码造成的改变

textChanged() 包括被程序代码造成的改变

## 文本过长显示省略号的方法

使用QFontMetrics::elidedText函数让文本显示省略号(省略号位置可以在文本开头/中间/末尾)

函数原型：

QString QFontMetrics::elidedText(const QString &text, Qt::TextElideMode mode, int width, int flags = 0) const

第二个参数为文本缩略后，省略号所在的位置，可选：

Qt::ElideLeft 0 省略号应出现在文本的开头。

Qt::ElideRight 1 省略号应出现在文本的末尾。

Qt::ElideMiddle 2 省略号应出现在文本的中间。

Qt::ElideNone 3 省略号不应出现在文本中。

示例代码：

QString str = ui->label->text();

QFontMetrics metrics(ui->label->font());

if (metrics. horizontalAdvance (str) > ui->label->width())

{

str = QFontMetrics(ui->label->font()).elidedText(str, Qt::ElideRight, ui->label->width());

}

ui->label->setText(str);

# focusPolicy : [Qt::FocusPolicy](../qtcore/qt.html#FocusPolicy-enum)

This property holds the way the widget accepts keyboard focus

The policy is [Qt::TabFocus](../qtcore/qt.html#FocusPolicy-enum) if the widget accepts keyboard focus by tabbing, [Qt::ClickFocus](../qtcore/qt.html#FocusPolicy-enum) if the widget accepts focus by clicking, [Qt::StrongFocus](../qtcore/qt.html#FocusPolicy-enum) if it accepts both, and [Qt::NoFocus](../qtcore/qt.html#FocusPolicy-enum) (the default) if it does not accept focus at all.

Focu有分Widget内部的focus，也有分窗口与窗口之间的focus

void QWidget::setFocus()

This is an overloaded function.

Gives the keyboard input focus to this widget (or its focus proxy) if this widget or one of its parents is the **active window**.

void QWidget::**activateWindow**()

Sets the top-level widget containing this widget to be the active window.

An active window is a visible top-level window that has the keyboard input focus.

**是 activate**Window**() ，不是active**

**QPushButton可以直接设置setDisabled让按钮不可用**

# QPoint

//Widget coordinate mapping

QPoint mapToGlobal(const QPoint &) const; //控件内转屏幕

QPoint mapFromGlobal(const QPoint &) const; //屏幕转控件内

QPoint mapToParent(const QPoint &) const; //控件内转父控件

QPoint mapFromParent(const QPoint &) const; //父控件转控件内

QPoint mapTo(const QWidget,const QPoint &) const; //转换至指定控件

QPoint mapFrom(const QWidget,const QPoint &) const; //从指定控件转换为控件内

# 添加字体

int fontID=QFontDatabase::addApplicationFont(":/font/PingFangSC.ttf");

QString PingFangSC=QFontDatabase::applicationFontFamilies ( fontID ).at(0);

QFont font(PingFangSC,10);

QApplication::setFont(font);

方法：

1. 将字体放到目录下并纳入res资源文件中
2. 再加入FontDatabase中

QFontDatabase::addApplicationFont(":/font/AdobeHeitiStd-Regular.otf");

QFontDatabase::addApplicationFont(":/font/AdobeSongStd-Light.otf");

1. 把FontDatabase中的字体打印出来

QFontDatabase fontDataBase;

foreach (const QString &family, fontDataBase.families()) {

qDebug()<<family;

}

1. 根据打印出来的名字加入样式表中

# 为QPushButton添加菜单

QMenu \*menu = new QMenu(); //创建菜单

QAction \*action1 = new QAction("新建"); //创建动作1

QAction \*action2 = new QAction("保存"); //创建动作2

menu->addAction(action1); //添加动作1

menu->addAction(action2); //添加动作2

btn->setMenu(menu); //为按钮添加菜单



# QEvent和QMouseEvent

！！要注意，主要使用的有五个鼠标事件，最常用的是enterEnter(QEvent\* ev), leaveEvent(QEvent\* ev)和mousePressEvent(QMouseEvent\* ev) 这几个有的是QEvent的，有的是QMouseEvent的，注意搞清楚

# CalendarPage技术实现

建两个page，一个CalendarPage，一个CalendarFullPage，用户可以选择默认的页面。页面内可以切换到另外一个页面。CalendarFullPage完全按照滴答清单去做。

## EventlistWIdget

单击日程之后，在日程的下方显示该日程的详细信息。这个详细信息也是一个listWidgetItem形式，写多几种自定义控件（因为不同的日程有不同的详细信息展示形式）。eventItem处设置一个编辑按钮，这个编辑按钮默认是不显示的，只有当鼠标移动到这个evenItem上才会显示。这个图标也是一个圆形按钮，鼠标移动上面会变成点击样式，没有hover样式，（没有点击效果）。

先写好DayEvent吧

MomentEvent

IntervalEvent

800里面是150+50+500+100

ListWidget写border-radius就行了，item不写，itemInfo也不写。ItemInfo的背景色要改一下。

Vector

点击日历的其它天的时候，删除页面的所有item，并读文件根据点击的天重新建新的item。

根据创建的先后给对象创建一个event标识码，这个标识码与年月日、创建顺序有关。先创建对象，再根据标识码vector中的

Vector在中间插入一个元素

v2.insert(v2.begin()+4, L"3"); *//在指定位置，例如在第五个元素前插入一个元素*

插完之后之后遍历当天设置文件的标识码（写在对象的类函数里面），这个标识码通过信号和槽传递给编辑框

在文件中是完全乱序的，就是按照加进去的顺序逐个写，有删除的，覆盖掉删除继续写。

## 文件存储

信号和槽只是建立起一个联系，这个联系不会在函数结束之后而消失。

如果事件没有发生，则槽函数不会执行。

每一次读写之后都要迅速关闭文件。

在各个槽函数里面都写QFile，读写就关掉。

信号和槽也可以装进函数里面简化主代码。

文件通过 [open](https://zhuanlan.zhihu.com/p/264909693/edit#override-virtualbool-qfileopenqiodeviceopenmode-mode)() 打开，通过 [close](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qfiledevice.html%23close)() 关闭，通过 [flush](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qfiledevice.html%23flush)() 刷新。数据通常使用 [QDataStream](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qdatastream.html) or [QTextStream](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qtextstream.html) 读写，但您也可以使用 由 [QIODevice](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qiodevice.html) 的继承函数 [read](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qiodevice.html%23read)(), [readLine](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qiodevice.html%23readLine)(), [readAll](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qiodevice.html%23readAll)(), [write](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qiodevice.html%23write)()。单字符的操作也可以使用 [getChar](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qiodevice.html%23getChar)(), [putChar](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qiodevice.html%23putChar)(), and [ungetChar](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qiodevice.html%23ungetChar)()。[size](https://zhuanlan.zhihu.com/p/264909693/edit" \l "size)() 返回文件大小。您可以通过 [pos](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qfiledevice.html%23pos)() 获取当前文件位置，或通过 [seek](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qfiledevice.html%23seek)() 移动到新的位置（译者注：此句中的“位置”指文件内操作的字节位置）。当您读到文件结尾， [atEnd](https://link.zhihu.com/?target=https%3A//doc.qt.io/qt-5/qfiledevice.html%23atEnd)() 返回 true。

不需要用tr()，这是用来翻译的

写saveBtn

## QListWidget

[**currentItem**](qlistwidget.html#currentItem)()

[**currentRow**](qlistwidget.html#currentRow-prop)()

item和其eventBaseWidget的关系

## QListWidgetItem

设置Item的hint，hover的时候显示（没有备注就不显示），并且有备注的话单独再显示一个图标

Click之后弹出编辑框

在MainWidget里面写一个读写数据的static函数（用一个EventFull对象作为参数）

从自定义Dialog调用这个static函数完成对数据的写入

用定时器（在MainWidget里写一个connect，Dialog/QPushButton的按下）去完成对freshList()

hover的话listWidget可以实现，但是点击必须要PushButton

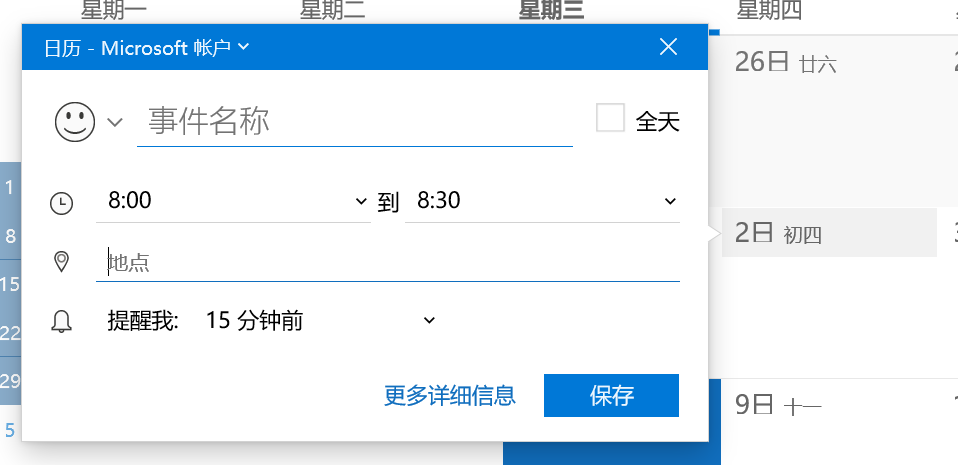
widget有hover，没有clicked，但是自定义控件只能是Widget，所以必须要事件监听

## QT-子线程或自定义类操作访问主界面UI控件的几种方法

<https://blog.csdn.net/u011555996/article/details/122153240>

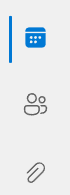
## 新建日程对话框

要检查输入



# 侧边栏

hover图标的时候改变图标（图标反色）、背景色改变

点击之后有类似于这样的变化，或是整个背景色保持hover 的背景色，要建一个buttonGroup并实现不能多个按下，且有默认的

## 要广泛地使用QString和QStringList

QList<QString> **arrHuayu**;

for (int **i** = 0; i<5;i++) {

arrHuayu.append(QString::asprintf("./images/pictureMusic/huayu%d.png",i));

}

QString **style** = "QWidget#picTitle{border-image:url("+arrHuayu[0]+");}";

ui->picTitle->setStyleSheet(style);

style = "QWidget#widMus1{border-image:url("+arrHuayu[1]+");}";

ui->widMus1->setStyleSheet(style);

style = "QWidget{border-image:url("+arrHuayu[2]+");}";

ui->widMus2->setStyleSheet(style);

style = "QWidget{border-image:url("+arrHuayu[3]+");}";

ui->widMus3->setStyleSheet(style);

style = "QWidget{border-image:url("+arrHuayu[4]+");}";

ui->widMus4->setStyleSheet(style);

## 巧妙地设置图片路径

QStringList pictureList;

for(int i = 0; i<11; i++)

{

QString str = QString::asprintf("./images/message/per**%d**.png",**i**);

pictureList.append(str);

}

## 新建listItem

for (int i = 0; i <11; i++) {

QListWidgetItem \*item = new QListWidgetItem(ui->listMessage); //

MessageItem \*messItem = new MessageItem(); //

messItem->setRed(flog[i]);

messItem->setStringPic(pictureList[i]);

messItem->setName(perNamList[i]);

messItem->setTime(timerList[i]);

messItem->setInformMessage(messageInformList[i]);

ui->listMessage->setItemWidget(item,messItem); //

}

QListWidgetItem \***item4** = new QListWidgetItem(ui->listAll);

NewCourierItem \* **nitem4** = new NewCourierItem();

nitem4->setNum(04); //设置编号

nitem4->setPicture(":/images/NewMusic/20.png");

nitem4->setMusicName("starting over");

nitem4->setLogPic();

nitem4->setMusicAuthor("三代目");

nitem4->setMusicAlbum("BEST BROTHERS");

nitem4->setMusicTimer("04:20");

ui->**listAll->setItemWidget(item4,nitem4);**

item是listWidget的列表项目

messItem是我的自定义组件

先把两个都new出来，其中listWidgetItem在new的时候要和listWidget建立联系

最后再从listWidget处setItemWidget，把item设置成messItem

## 案例剖析

外面是一个widget，里面两个是QPushButton

对Widget（名为widNewNus）

ui->widNewMus->setStyleSheet("*QWidget#widNewMus*{**background:rgb(255,255,255)**;

border:1px solid rgb(187,187,187);

border-radius:16px;}");

对于内部的两个QPushButton（名为ptnNew和ptnNew2）

ui->ptnNew->setParent(*ui*->*widNewMus*);

ui->ptnNew2->setParent(*ui*->*widNewMus*);

ui->ptnNew->setCheckable(true);

ui->ptnNew2->setCheckable(true);

以上应该都可以在ui界面设置

***最关键的是样式表***

ui->ptnNew->setStyleSheet("QPushButton{color:rgb(55,55,55);

border:0px; background:rgb(255,255,255); border-radius:14px;}"

//color是字体的颜色，也是黑，只是有点不太一样而已

//虽然border设置为0px了，但是设置border-radius是很重要的意义的

"QPushButton:**hover**{border:0px;background:rgb(244,244,244);

border-radius:14px;}"

//仍然要再写一次border=0px和border-radius

"QPushButton:**checked**{**color:rgb(255,255,255)**;border:0px;

background:rgb(187,187,187);border-radius:14px;}");

//关键在于color和background颜色反转

(两者设置了相同的样式)

ui->ptnNew2->setStyleSheet("QPushButton{color:rgb(55,55,55);

border:0px;background:rgb(255,255,255);border-radius:14px;}"

"QPushButton:hover{border:0px;background:rgb(244,244,244);

border-radius:14px;}"

"QPushButton:checked{color:rgb(255,255,255);border:0px;

background:rgb(187,187,187);border-radius:14px;}");

还有特别重要的一点

**QButtonGroup** \* btnGroup = new QButtonGroup(this);

btnGroup->addButton(ui->ptnNew,0);

btnGroup->addButton(ui->ptnNew2,1);

btnGroup->**setExclusive(true)**;

btnGroup->button(0)->**setChecked**(true);

建一个ButtonGroup把两个button放进去，设置Exclusive和设置默认checked的button

看不明白的连接槽

connect(btnGroup, QOverload<int>::of(&QButtonGroup::buttonClicked),

[=](int id){

//这里处理对应界面

setShowWidget(id);

});

## setStyleSheet的一个案例

ui->widGroupBut->setStyleSheet("QWidget#widGroupBut

{background:rgb(255,255,255); //背景设置为白色

border:1px solid rgb(187,187,187); border-radius:15px;}");



## 设置多个button不能同时按下

QButtonGroup \* btnGroup = new QButtonGroup(this);

btnGroup->addButton(ui->ptnPer,0);

btnGroup->addButton(ui->ptnComment,1);

btnGroup->addButton(ui->ptnMe,2);

btnGroup->addButton(ui->ptnInform,3);

**btnGroup->setExclusive(true);**  其实关键在这一步

btnGroup->button(0)->setChecked(true); 默认选中第0个按钮

# 样式表

样式表的使用必须极其谨慎！！UI的关键就在样式表！样式表的使用必须有严格的审查！

样式表要放在几个文件的头部以方便审查。

作用的对象必须非常明确，否则会直接牵连所有子类的样式。

最好在Qt Designer那里去设置，后续如果需要代码再把这些代码复制出来小心测试。

## 样式表设置的规则

**！！当一个样式表是无效的时候，系统会恢复原始的样式表**。

**！！当一个样式表加在前一个样式表之上时，前一个样式表全部作废**。即便后一个样式表的语句与前样式表无关。

**！！样式表一次设置一直有效，直到它被改变**。

**！！样式表的设置有区域性**

ui->***sideBar***->setStyleSheet("QWidget{background:red}"); 这只会限于sideBar的区域

样式表作用的规则是： sideBar 范围内的QWidget均采取background:red 该规则，并且前面sideBar 设置的样式表作废。

！！***样式表设置的时候必须加上对象自己的名称以防止干涉其它对象的样式表***。（特殊情况除外）如

ui->sidebar-> setStyleSheet("QWidget# sidebar {background:red}");

！！同对象的样式表只能有一个，有了新的，旧的就会作废。

！！如果是子对象的样式表，新的会覆盖掉父对象作用于它的样式表。

**！！样式表只能设置一次，前面写的会被后面写的覆盖**。

**样式表只能设置一次，前面写的会被后面写的覆盖**。

**样式表只能设置一次，前面写的会被后面写的覆盖**。

！！**失效就说明：这个样式表的语法不对，或者新写的样式表语法不对**、或是新的样式表覆盖了旧样式表

是以块{}为单位的，如果一个语句块的语法有问题，只会使该语句块的样式设置失效，并不会影响其他语句块

！！！尤其是 ：（冒号） 特别容易写成等号=导致语法错误而致使整个组件样式失效！！！

Widget是有:hover的

设置的时候考虑这几个变量background, border, border-radius, color, :hover, :checked

**::Item** **是用height**，不是用width

QPushButton 有pressed, flat, default

样式表可以到Qt Designer那里去测试。可以直接在ui那里应用，免得代码被改变。

QListWidget{border:0px; border-radius:16px;}

QListWidget**::Item**{height:40px; border:1px solid rgb(200,200,200);

border-radius:32px;}

QListWidget**::Item:hover**{background:rgb(0,0,0);}

## 样式表使用学习

class names, [object names](../qtcore/qobject.html#objectName-prop), and Qt property names are case sensitive.

Several selectors can be specified for the same declaration, using commas (,) to separate the selectors. For example, the rule

QPushButton, QLineEdit, QComboBox { color: red }

is equivalent to this sequence of three rules:

QPushButton { color: red }

QLineEdit { color: red }

QComboBox { color: red }

The declaration part of a style rule is a list of property: value pairs, enclosed in braces ({}) and separated with semicolons. For example:

QPushButton { color: red; background-color: white }

**List of Properties**

The table below lists all the properties supported by Qt Style Sheets. Which values can be given to an property depend on the [property's type](stylesheet-reference.html#list-of-property-types).

Universal Selector **\***  Matches all widgets.

Type Selector **QPushButton** Matches instances of QPushButton and of its subclasses.

Property Selector QPushButton[flat="false"]

Class Selector .QPushButton Matches instances of [QPushButton](qpushbutton.html), but not of its subclasses.

ID Selector **QPushButton#okButton** Matches all [QPushButton](qpushbutton.html) instances whose [object name](../qtcore/qobject.html#objectName-prop) is okButton.

Descendant Selector QDialog QPushButton Matches all instances of [QPushButton](qpushbutton.html) that are descendants (children, grandchildren, etc.) of a [QDialog](qdialog.html).

Child Selector QDialog > QPushButton Matches all instances of [QPushButton](qpushbutton.html) that are direct children of a [QDialog](qdialog.html)

### Sub-Control

[**QComboBox**](qcombobox.html)**::drop-down { image: url(dropdown.png) }**

::icon

::scroller 滚动条

::text

::title

QLabel 不支持 :hover

:pressed The item is being pressed using the mouse.

[**QMenu**](qmenu.html)**::item:selected**

[**QMenu**](qmenu.html)**::icon:checked**

[**QMenu**](qmenu.html)**::separator**

[**QMenu**](qmenu.html)**::indicator**

[**QTabBar**](file:///C:\Users\Redmibook_PC\Desktop\qtabbar.html)**::tab:!selected**

[**QMenu**](qmenu.html)**::indicator*:non-exclusive*:unchecked*:selected* {**

**image: url(:/images/checkbox\_unchecked\_hover.png);**

**}**

[**QTabBar**](qtabbar.html)**::tab:first:selected {**

**margin-left: 0; /\* the first selected tab has nothing to overlap with on the left \*/**

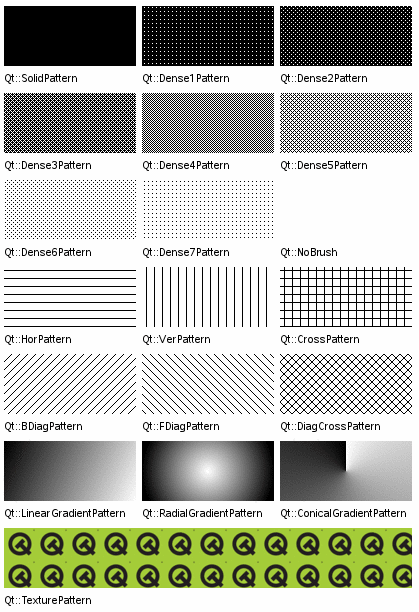
**}**

[**QPushButton**](qpushbutton.html)**:pressed**

[**QPushButton**](qpushbutton.html)**:flat**

[**QPushButton**](qpushbutton.html)**:default**

### background

Shorthand notation for setting the background. Equivalent to specifying background-color, **background-image**, background-repeat, and/or background-position.

[**QLabel**](qlabel.html) **{**

**background-image: url(dense6pattern.png);**

background-repeat: repeat-xy;

**}**

可以设置background:transparent;

**background: #EEEEEE;**

#### background-position

The alignment of the background image within the background-origin rectangle.

If this property is not specified, the alignment is top left.

Example:

[**QFrame**](qframe.html) **{**

**background: url(:/images/footer.png);**

**background-position: bottom left;**

**}**

[**QLabel**](qlabel.html) **{ background-color: yellow }**

[**QLineEdit**](qlineedit.html) **{ background-color: rgb(255, 0, 0) }**

[**QFrame**](qframe.html) **{ background-image: url(:/images/hydro.png) }**

#### background: qlineargradient

**background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,**

**stop: 0 #FAFBFE, stop: 1 #DCDEF1);**

**background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,**

**stop: 0 #E1E1E1, stop: 0.4 #DDDDDD,**

**stop: 0.5 #D8D8D8, stop: 1.0 #D3D3D3);**

**Background-clip 和 background-origin 不明白**

#### background-attachment: fixed;

背景图片不会随着滚动条而滚动

### 滚动条

The [QScrollBar](qscrollbar.html) can be styled using its subcontrols like [handle](stylesheet-reference.html#handle-sub), [add-line](stylesheet-reference.html#add-line-sub), [sub-line](stylesheet-reference.html#sub-line-sub), and so on.

### Border

border-top，border-bottom，border-right，border-left

可以设置border的宽度、颜色

border-top-left-radius，border-top-right-radius，

border-bottom-right-radius，border-bottom-left-radius，

radius是独立出来的，可以设置两个参数（椭圆），一般设置一个就行了

### font

font-family, font-size, font-style, and/or font-weight.

**font: bold italic large "Times New Roman"**

font-family: Cambria;//可以写中文，Times New Roman也行

font-size: 12px;

font-weight: normal|bold|100|200|…

font-style：italic;

### 其它属性

**height: 10px；**

icon-size：12px；

**image: url(:/images/spindown.png)；**

### 文本对齐（只有QPushButton支持）

**text-align: center;**

top, bottom, left, right, center (Alignment)

[**QTextEdit**](qtextedit.html) **{ background-position: bottom center }**

### 实例

**border: 1px solid #6a6ea9;**

**border: none;**

**width: 10px; /\* when vertical \*/**

**height: 10px; /\* when horizontal \*/**

**border-color: darkblue;//可以以直接用border写**

**background: rgba(100, 100, 100, 150);**

**支持rgba色彩，a是透明度.darkblue, lightblue都可以用**

**image: url(:/images/sizegrip.png);**

**image: url(:/images/up\_arrow\_disabled.png);**

**font: italic;**

**opacity: 200;**

**颜色有darkblue, lightblue, navy, cyan，magenta**

## ListWidget和Item的样式设置

ui->listWidget->setStyleSheet("QListWidget{border:1px;background:white;border-radius:5px;}"

"QListWidget**::Item**{ background:white; **height:60px**;}" 给Item设置宽度

"QListWidget**::Item:hover**{background:rgb(246,246,247);}" 当鼠标hover的时候背景颜色发生改变

);

这些设置是有效的！

## 滚动条样式

QScrollBar\* sv = ui->listMessage->verticalScrollBar();

sv->setStyleSheet("QScrollBar:vertical{background:transparent; width:6px;}"

"QScrollBar::handle:vertical{min-height:10px;max-height:20px;background:rgb(224,224,224); width:6px; border-radius:3px;}"

"QScrollBar::up-arrow:vertical{border:none;}"

"QScrollBar::sub-line:vertical{background:transparent;}"

"QScrollBar::add-line:vertical{background:transparent;}");

# QDateTime相关

## daysTo

QDateTime startDate(QDate(2012, 7, 6), QTime(8, 30, 0));

QDateTime endDate(QDate(2012, 7, 7), QTime(16, 30, 0));

qDebug() << "Days from startDate to endDate: " << startDate.daysTo(endDate);

startDate = QDateTime(QDate(2012, 7, 6), QTime(23, 55, 0));

endDate = QDateTime(QDate(2012, 7, 7), QTime(0, 5, 0));

qDebug() << "Days from startDate to endDate: " << startDate.daysTo(endDate);

qSwap(startDate, endDate); // Make endDate before startDate.

qDebug() << "Days from startDate to endDate: " << startDate.daysTo(endDate);

# 英文术语

flat 按压下去的

# 报错与解决方案/技术难题

## 找不到ui\_widget.h

Widget没有布局

要建一个Widget才能提升为，不能对主窗口提升为

## 类名与软件内置的类冲突

类名要设置的特殊一点，不要用什么Time、Event，这种必然冲突的

## Layout报错

Break所有布局

## Stylesheet报错

严格检查报错的stylesheet，看有没有括号打漏

## 直接crash

改正第三方库、删除冗余冲突的文件

## 不能用信号和槽去传输文件流的引用

直接在lambda表达式里面写QFile

## 查看实例的源码

双击之后返回代码界面，左下角配置

QString::number() 和 str.toInt() 是融洽的，要一起使用，不能只用一个

虽然 str=int 不会报错，但是这种str.toInt()是得不到原来的 int的，必须要使用QString::number()

## 添加环境变量

我的电脑->属性->高级系统设置->环境变量->系统变量里找到path在最后添加";"号.然后

把Qt的bin目录添加在最后面.保存.

值得注意的是Qt安装目录里面一般有两个bin目录,一个是Qt Creator的bin目录添加这个是木有用的.正确的bin目录在一般是目录嵌套层数比较大的那个.

## QVector out of range

在初始化events时要设置events(500)；

## 相互包含报错

用前置声明

class EditDialog;

## multiple definition of XXX 报错

将所有的全局变量放入一个头文件 global.h (名字随意起，但要加条件编译)中，每一个变量前面加extern，声明一下这些变量将在其它文件中定义。 然后建立一个和头文件名字对应的.c or .cpp文件 如global.c。在里面声明所有的全局变量。然后，让涉及到全局变量的文件include ”global.h“。这样编译时，会先对global.c编译生成一个global.o ，然后再和其它文件的.o链接生成可执行文件。

## 继承于QWidget的自定义控件样式失效问题

在使用Qt的界面时，经常会自定义控件，当我们自定义QWidget控件，即写一个MyWidget，继承自QWidget，在MyWidget中设置样式表，实际运行程序，会发现样式表设置没有效果。

原因：qt的样式表需要通过paintEvent函数来实现，而继承自QWidget的控件，QWidget的paintEvent函数实现为空，也就导致了设置样式表是无法生效的。

解决方法：在继承的类中，重写paintEvent函数

void MyWidget::paintEvent(QPaintEvent\*)

{

QStyleOption opt;

opt.init(this);

QPainter p(this);

style()->drawPrimitive(QStyle::PE\_Widget, &opt, &p, this);

}

还要包含头文件

#include<QPainter>

#include<QStyleOption>

## 模拟鼠标点击（特别困难才解决的一个问题）

QTest mouseClick()

用QTest反复报错

//这是相对于点击作用对象的坐标

QMouseEvent clickPopup(QEvent::MouseButtonPress, **QPoint(140,20**),

Qt::LeftButton, Qt::LeftButton, Qt::NoModifier);

**QApplication::sendEvent(ui->startDateEdit, &clickPopup);**

**//作用对象和点击事件**

## 将cpp/h和ui文件连接起来

最重要的一步是在ui文件中改掉最顶部组件的对象名（改成类名）

## 不报错的crash

必然是调用了空指针的函数

必然是调用了空指针的函数

必然是调用了空指针的函数

有一些指针只是声明了，**还没有赋值**，就会出现这种情况

### 悬垂指针报错

貌似CalendarWidget的初始化比MainWidget还早，我不得不将CheckDate（一个QDateEdit）的初始化放在CalendarWidget的构造函数中以防止程序崩溃

因为指针并没有生成并指向有效对象，因此调用其函数时程序直接崩溃

## Widget无法伸缩的话看下ui里面是不是进行了相关的设置

# 发布程序

1. 把debug换成release
2. 找到目录下的release拷贝出来
3. 打开Qt套件命令行窗口
4. 输入 C:
5. cd 路径
6. windeployqt Scheduling.exe