在进行窗口程序的处理过程中，经常要周期性的执行某些操作，或者制作一些动画效果，看似比较复杂的问题使用定时器就可以完美的解决这些问题， Qt中提供了两种定时器方式一种是使用Qt中的事件处理函数这个在后续章节会给大家做细致的讲解，本节主要给大家介绍一下Qt中的定时器类 `QTimer` 的使用方法。

要使用它，只需创建一个QTimer类对象，然后调用其 `start()` 函数开启定时器，此后QTimer对象就会周期性的发出 `timeout()` 信号。我们先来了解一下这个类的相关API。

1. public/slot function

C++

```cpp
// 构造函数
// 如果指定了父对象, 创建的堆内存可以自动析构
QTimer::QTimer(QObject *parent = nullptr);

// 设置定时器时间间隔为 msec 毫秒
// 默认值是0，一旦窗口系统事件队列中的所有事件都已经被处理完，一个时间间隔为0的QTimer就会触发
void QTimer::setInterval(int msec);
// 获取定时器的时间间隔, 返回值单位: 毫秒
int QTimer::interval() const;

// 根据指定的时间间隔启动或者重启定时器, 需要调用 setInterval() 设置时间间隔
[slot] void QTimer::start();
// 启动或重新启动定时器，超时间隔为msec毫秒。
[slot] void QTimer::start(int msec);
// 停止定时器。
[slot] void QTimer::stop();

// 设置定时器精度
/*
参数:
    - Qt::PreciseTimer -> 精确的精度, 毫秒级
    - Qt::CoarseTimer  -> 粗糙的精度, 和1毫秒的误差在5%的范围内, 默认精度
    - Qt::VeryCoarseTimer -> 非常粗糙的精度, 精度在1秒左右
*/
void QTimer::setTimerType(Qt::TimerType atype);
Qt::TimerType QTimer::timerType() const;        // 获取当前定时器的精度

// 如果定时器正在运行，返回true; 否则返回false。
bool QTimer::isActive() const;

// 判断定时器是否只触发一次
bool QTimer::isSingleShot() const;
// 设置定时器是否只触发一次, 参数为true定时器只触发一次, 为false定时器重复触发, 默认为false
void QTimer::setSingleShot(bool singleShot);
```

2. signals

这个类的信号只有一个，当定时器超时时，该信号就会被发射出来。给这个信号通过 `conect()` 关联一个槽函数，就可以在槽函数中处理超时事件了。

C++

```cpp
[signal] void QTimer::timeout();
```

3. static public function

C++

```cpp
// 其他同名重载函数可以自己查阅帮助文档
/*
功能: 在msec毫秒后发射一次信号, 并且只发射一次
参数:
        - msec:     在msec毫秒后发射信号
        - receiver: 接收信号的对象地址
        - method:   槽函数地址
*/
```

```
[static] void QTimer::singleShot(
        int msec, const QObject *receiver,
        PointerToMemberFunction method);
```

4. 定时器使用举例

- 周期性定时器

```C++
// 创建定时器对象
QTimer* timer = new QTimer(this);

// 修改定时器对象的精度
timer->setTimerType(Qt::PreciseTimer);

// 按钮 loopBtn 的点击事件
// 点击按钮启动或者关闭定时器，定时器启动，周期性得到当前时间
connect(ui->loopBtn, &QPushButton::clicked, this, [=]()
{
    // 启动定时器
    if(timer->isActive())
    {
        timer->stop();  // 关闭定时器
        ui->loopBtn->setText("开始");
    }
    else
    {
        ui->loopBtn->setText("关闭");
        timer->start(1000); // 1000ms == 1s
    }
});

connect(timer, &QTimer::timeout, this, [=]()
{
    QTime tm = QTime::currentTime();
    // 格式化当前得到的系统时间
    QString tmstr = tm.toString("hh:mm:ss.zzz");
    // 设置要显示的时间
    ui->curTime->setText(tmstr);
});

```

- 一次性定时器

```C++
// 点击按钮 onceBtn 只发射一次信号
// 点击按钮一次，发射一个信号，得到某一个时间点的时间
connect(ui->onceBtn, &QPushButton::clicked, this, [=]()
{
     // 获取2s以后的系统时间，不创建定时器对象，直接使用类的静态方法
    QTimer::singleShot(2000, this, [=](){
        QTime tm = QTime::currentTime();
        // 格式化当前得到的系统时间
        QString tmstr = tm.toString("hh:mm:ss.zzz");
        // 设置要显示的时间
        ui->onceTime->setText(tmstr);
    });
});

```

5. 视频讲解

以上知识点对应的视频讲解可以关注 B站-爱编程的大丙

视频地址: https://www.bilibili.com/video/BV1Jp4y167R9