

qt实战之时钟表盘

时钟表盘案例

Dialog

获取当前时间

日期2021/9/18时间14:05

日期时间2021/9/18 14:05

其他格式2021年09月18日 14:05:16

九月 2021

	周一	周二	周三	周四	周五	周六	周日
35	30	31	1	2	3	4	5
36	6	7	8	9	10	11	12
37	13	14	15	16	17	18	19
38	20	21	22	23	24	25	26
39	27	28	29	30	1	2	3
40	4	5	6	7	8	9	10

时钟

开始

停止

定时周期毫秒

14

5

18

案例知识点

- 1. 日期和时间控件
- 2. QDateTime、QDate、QTime类的操作
- 3. QCalendarWidget日历组件
- 4. QTimer定时器

QT中时间和日期类型

时间日期是经常遇到的数据类型，Qt 中时间日期类型的类如下。

- `QTime`: 时间数据类型，仅表示时间，如 15:23:13。
- `QDate`: 日期数据类型，仅表示日期，如 2017-4-5。
- `QDateTime`: 日期时间数据类型，表示日期和时间，如 2017-03-23 08:12:43。

Qt 中有专门用于日期、时间编辑和显示的界面组件，介绍如下。

- `QTimeEdit`: 编辑和显示时间的组件。
- `QDateEdit`: 编辑和显示日期的组件。
- `QDateTimeEdit`: 编辑和显示日期时间的组件。
- `QCalendarWidget`: 一个用日历形式选择日期的组件。

定时器是用来处理周期性事件的一种对象，类似于硬件定时器。例如设置一个定时器的定时周期为 1000 毫秒，那么每 1000 毫秒就会发射定时器的 `timeout()` 信号，在信号关联的槽函数里就可以做相应的处理。Qt 中的定时器类是 `QTimer`，它直接从 `QObject` 类继承而来，不是界面组件类。

QT中时间和日期类型

QDateEdit 和 QTimeEdit 都从 QDateTimeEdit 继承而来，实现针对日期或时间的特定显示功能。实际上，QDateEdit 和 QTimeEdit 的显示功能都可以通过 QDateTimeEdit 实现，只需设置好属性即可。

QDateTimeEdit 类的主要属性的介绍如下。

- `datetime`: 日期时间。
- `date`: 日期，设置 `datetime` 时会自动改变 `date`，同样，设置 `date` 时，也会自动改变 `datetime` 里的日期。
- `time`: 时间，设置 `datetime` 时会自动改变 `time`，同样，设置 `time` 时，也会自动改变 `datetime` 里的时间。
- `maximumDateTime`、`minimumDateTime`: 最大、最小日期时间。
- `maximumDate`、`minimumDate`: 最大、最小日期。
- `maximumTime`、`minimumTime`: 最大、最小时间。
- `currentSection`: 当前输入光标所在的时间日期数据段，是枚举类型 `QDateTimeEdit::Section`。
QDateTimeEdit 显示日期时间数据时分为多个段，单击编辑框右侧的上下按钮可修改当前段的值。如输入光标在 `YearSection` 段，就修改“年”的值。
- `currentSectionIndex`: 用序号表示的输入光标所在的段。
- `calendarPopup`: 是否允许弹出一个日历选择框。当取值为 `true` 时，右侧的输入按钮变成与 `QComboBox` 类似的下拉按钮，单击按钮时出现一个日历选择框，用于在日历上选择日期。对于 `QTimeEdit`，此属性无效。
- `displayFormat`: 显示格式，日期时间数据的显示格式，例如设置为“`yyyy-MM-dd HH:mm:ss`”，一个日期时间数据就显示为“2016-11-02 08:23:46”。

日期类型转换字符串

将 curDateTime 表示的日期时间数据转换为字符串，然后在 LineEdit 编辑框上显示。时间日期转换为字符串使用了 QDateTime 的 toString()函数，分别用不同的格式显示时间、日期、日期时间。

```
ui->editTime->setText(curDateTime.toString("hh:mm:ss"));
ui->editDate->setText(curDateTime.toString("yyyy-MM-dd"));
ui->editDateTime->setText(curDateTime.toString("yyyy-MM-dd hh:mm:ss"));
```

QDateTime::toString()函数的函数原型是：

```
QString QDateTime::toString(const QString &format) const
```

它将日期时间数据按照 format 指定的格式转换为字符串。format 是一个字符串，包含一些特定的字符，表示日期或时间的各个部分，表 4-2 是用于日期时间显示的常用格式符。

表 4-2 用于日期显示的格式符及其意义

字符	意义
d	天，不补零显示，1-31
dd	天，补零显示，01-31
M	月，不补零显示，1-12
MM	月，补零显示，01-12
yy	年，两位显示，00-99
yyyy	年，4 位数字显示，如 2016
h	小时，不补零，0-23 或 1-12（如果显示 AM/PM）
hh	小时，补零 2 位显示，00-23 或 01-12（如果显示 AM/PM）
H	小时，不补零，0-23（即使显示 AM/PM）
HH	小时，补零显示，00-23（即使显示 AM/PM）
m	分钟，不补零，0-59
mm	分钟，补零显示，00-59
z	毫秒，不补零，0-999
zzz	毫秒，补零 3 位显示，000-999
AP 或 A	使用 AM/pm 显示

字符串转换日期类型

同样的，也可以将字符串转换为 QTime、QDate 或 QDateTime 类型，使用静态函数 QDateTime::fromString()，其函数原型为：

```
QDateTime QDateTime::fromString(const QString &string, const QString &format)
```

定时器的使用

Qt 中的定时器类是 QTimer。QTimer 不是一个可见的界面组件，在 UI 设计器的组件面板里找不到它。实例程序实现了一个计时器的功能，就是计算定时器开始到停止持续的时间长度，计时器是 QTimer 类。

QTimer 主要的属性是 interval，是定时中断的周期，单位毫秒。QTimer 主要的信号是 timeout()，在定时中断时发射此信号，要想在定时中断里做出响应，这就需要编写 timeout()信号的槽函数。下面是窗口类中增加的定义（省略了其他不相关的定义）：

```
class Dialog : public QDialog
{
private:
    QTimer *fTimer; //定时器
    QTimer fTimeCounter;//计时器
private slots:
    void on_timer_timeout(); //定时溢出处理槽函数
};
```


定时器的使用

这里定义了一个定时器 fTimer，一个计时器 fTimeCounter。还定义了一个槽函数 on_timer_timeout()，作为定时器的 timeout()信号的响应槽函数。

需要在窗口类的构造函数里创建定时器，并进行信号与槽的关联。代码如下：

```
Dialog::Dialog(QWidget *parent) :    QDialog(parent),    ui(new Ui::Dialog)
{
    ui->setupUi(this);
    fTimer=new QTimer(this);
    fTimer->stop();
    fTimer->setInterval(1000); //设置定时周期，单位：毫秒
    connect(fTimer,SIGNAL(timeout()),this,SLOT(on_timer_timeout()));
}
```

槽函数 on_timer_timeout()的实现代码如下：

```
void Dialog::on_timer_timeout()
{ //定时器中断响应
    QTime curTime=QTime::currentTime(); //获取当前时间
    ui->LCDHour->display(curTime.hour()); //显示 小时
    ui->LCDMin->display(curTime.minute()); //显示 分钟
    ui->LCDSec->display(curTime.second()); //显示 秒
    int va=ui->progressBar->value();
    va++;
    if (va>100)
        va=0;
    ui->progressBar->setValue(va);
}
```

案例说明-UI设计

GroupBox

获取当前时间

日期 2000/1/1 时间 0:00

日期时间 2000/1/1 0:00

其他格式

九月 2021

周一 周二 周三 周四 周五 周六 周日

35	30	31	1	2	3	4	5
36	6	7	8	9	10	11	12
37	13	14	15	16	17	18	19
38	20	21	22	23	24	25	26
39	27	28	29	30	1	2	3
40	4	5	6	7	8	9	10

时钟

开始 停止 定时周期毫秒

0 0 0

对象	类
Dialog	QDialog
groupBox	QGroupBox
btCurrentDateTime	QPushButton
calendarWidget	QCalendarWidget
dateEdit	QDateEdit
dateTimeEdit	QDateTimeEdit
label	QLabel
label_2	QLabel
label_3	QLabel
label_4	QLabel
timeEdit	QTimeEdit
txtlineEdit	QLineEdit
groupBox_2	QGroupBox
label_5	QLabel
lnHour	QLCDNumber
lnMinutes	QLCDNumber

Filter

calendarWidget : QCalendarWidget

属性	值
> QObject	
> QWidget	
> QCalendarWidget	
selectedDate	2021/9/18

案例代码

信号和槽

定时器

```
1  #ifndef DIALOG_H
2  #define DIALOG_H
3
4  #include <QDialog>
5  #include <QTimer>
6
7  namespace Ui {
8  class Dialog;
9  }
10
11  class Dialog : public QDialog
12  {
13      Q_OBJECT
14
15  public:
16      explicit Dialog(QWidget *parent = nullptr);
17      ~Dialog();
18
19  private slots:
20      void on_btCurrentDateTime_clicked();
21
22      void on_calendarWidget_selectionChanged();
23
24      void on_timer_timeout();
25
26      void on_pbstart_clicked();
27
28      void on_pbstop_clicked();
29
30  private:
31      Ui::Dialog *ui;
32      QTimer *ftimer;
33  };
34
35  #endif // DIALOG_H
36
```

案例代码

```
dialog.cpp Dialog::on_timer_timeout() -> void
#include "dialog.h"
#include "ui_dialog.h"

Dialog::Dialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::Dialog)
{
    ui->setupUi(this);
    ftimer=new QTimer(this);
    ftimer->stop();
    ftimer->setInterval(1000);

    QObject::connect(ftimer, SIGNAL(timeout()), this, SLOT(on_timer_timeout()));
}

Dialog::~Dialog()
{
    delete ui;
}
```

定时器设置
和定时函数关联

案例代码

```
void Dialog::on_timer_timeout()
{
    QTime curTime = QTime::currentTime();
    ui->lnHour->display(curTime.hour());
    ui->lnMinutes->display(curTime.minute());
    ui->lnSecond->display(curTime.second());
}
```

定时器定时触发的函数内容

```
void Dialog::on_pbstart_clicked()
{
    ftimer->start();
    ui->pbstart->setEnabled(false);
    ui->pbstop->setEnabled(true);
}
```

定时器开始触发

```
void Dialog::on_pbstop_clicked()
{
    ftimer->stop();
    ui->pbstart->setEnabled(true);
    ui->pbstop->setEnabled(false);
}
```

定时器停止触发

案例代码

```
void Dialog::on_btCurrentDateTime_clicked()
{
    QDateTime currentDateTime = QDateTime::currentDateTime();
    ui->dateEdit->setDate(currentDateTime.date());
    ui->timeEdit->setTime(currentDateTime.time());
    ui->dateTimeEdit->setDateTime(currentDateTime);
    ui->txtlineEdit->setText(currentDateTime.toString("yyyy年MM月dd日 hh:mm:ss"));
}
```

text: F1

```
void Dialog::on_calendarWidget_selectionChanged()
{
    QDate selectedDate = ui->calendarWidget->selectedDate();
    ui->dateEdit->setDate(selectedDate);
    ui->txtlineEdit->setText(selectedDate.toString("yyyy年MM月dd日"));
}
```

获得当前时间显示

从日历控件选择日期显示