

Элементы отображения

Элементы отображения не принимают активного участия в действиях пользователя и используются для информирования его о происходящем. Эта информация может носить как текстовый, так и графический характер (картинки, графика).

Надписи

Виджет надписи служит для показа состояния приложения или поясняющего текста и представляет собой текстовое поле, текст которого не подлежит изменению со стороны пользователя. Информация, отображаемая этим виджетом, может изменяться только самим приложением. Таким образом, приложение может сообщить пользователю о своем изменившемся состоянии, но пользователь не может изменить эту информацию в самом виджете. Класс виджета надписи `QLabel` определен в заголовочном файле `QLabel`.

Виджет надписи унаследован от класса `QFrame` и может иметь рамку. Отображаемая им информация может быть текстового, графического или анимационного характера, для передачи ее используются слоты `setText()`, `setPixmap()` и `setMovie()`.

Расположением текста можно управлять при помощи метода `setAlignment()`. Метод использует большое количество флагов, некоторые из них приведены в табл. 1. Обратите внимание, что значения не пересекаются, и это позволяет комбинировать их друг с другом с помощью логической операции `|` (ИЛИ). Наглядным примером служит значение `AlignCenter`, составленное из значений `AlignVCenter` и `AlignHCenter`.

Таблица 1. Значения флагов `Alignment` пространства имен `Qt`

Константа	Значение	Описание
<code>AlignLeft</code>	<code>0x0001</code>	Расположение текста слева
<code>AlignRight</code>	<code>0x0002</code>	Расположение текста справа
<code>AlignHCenter</code>	<code>0x0004</code>	Центровка текста по горизонтали
<code>AlignJustify</code>	<code>0x0008</code>	Растягивание текста по всей ширине
<code>AlignTop</code>	<code>0x0010</code>	Расположение текста вверху
Константа	Значение	Описание
<code>AlignBottom</code>	<code>0x0020</code>	Расположение текста внизу
<code>AlignVCenter</code>	<code>0x0040</code>	Центровка текста по вертикали
<code>AlignCenter</code>	<code>AlignVCenter</code> <code>AlignHCenter</code>	Центровка текста по вертикали и горизонтали

Как видно из рис. 1, виджет надписи может отображать не только обычный текст, но и текстовую информацию в формате HTML (HyperText Markup Language, язык гипертекстовой разметки). В этом примере (листинг 1) использовался HTML для вывода текста, таблицы и растрового изображения.

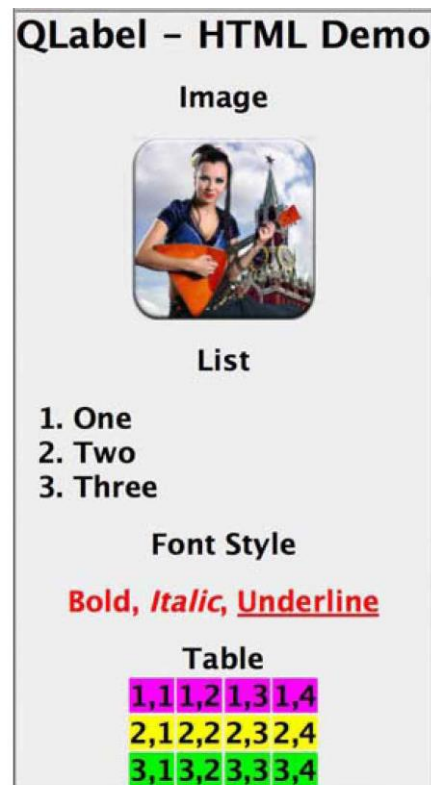


Рис. 1. Отображение информации виджетом надписи в формате HTML

Листинг 1. Файл main.cpp

```
#include <QtWidgets>
int main(int argc, char** argv)
{
    QApplication app(argc, argv);

    QLabel lbl("<H1><CENTER>QLabel - HTML Demo</CENTER></H1>"
        "<H2><CENTER>Image</CENTER><H2>"
        "<CENTER><IMG BORDER=\"0\" SRC=\":/Balalaika.png\"></CENTER>"
        "<H2><CENTER>List</CENTER><H2>"
        "<OL><LI>One</LI>"
        "  <LI>Two</LI>"
        "  <LI>Three</LI>"
        "</OL>"
        "<H2><CENTER>Font Style</CENTER><H2>"
        "<CENTER><FONT COLOR=RED>"
        "  <B>Bold</B>, <I>Italic</I>, <U>Underline</U>"
        "</FONT></CENTER>"
        "<H2><CENTER>Table</CENTER></H2>"
        "<CENTER> <TABLE>"
        "  <TR BGCOLOR=#ff00ff>"
        "    <TD>1,1</TD><TD>1,2</TD><TD>1,3</TD><TD>1,4</TD>"
        "  </TR>"
        "  <TR BGCOLOR=YELLOW>"
        "    <TD>2,1</TD><TD>2,2</TD><TD>2,3</TD><TD>2,4</TD>"
        "  </TR>"
        "  <TR BGCOLOR=#00f000>"
        "    <TD>3,1</TD><TD>3,2</TD><TD>3,3</TD><TD>3,4</TD>"
        "  </TR>"
        "</TABLE>"
        "</CENTER>"
    );
    lbl.show();
    return app.exec();
}
```

```

        " </TR>"
        "</TABLE> </CENTER>"
    );
    lbl.show();
    return app.exec();
}

```

В листинге 1 при создании виджета надписи lbl первым параметром в конструктор передается текст в формате HTML. Его можно передать и после создания этого виджета при помощи метода-слота setText (). Второй параметр конструктора опущен, а так как по умолчанию он равен 0, то это делает его виджетом верхнего уровня.

Следующий пример (листинг 2), показанный на рис. 2, демонстрирует возможность отображения информации графического характера в виджете надписи без использования формата HTML.



Рис. 2. Отображение графической информации виджетом надписи

Листинг 2. Файл main.cpp

```

#include <QtWidgets>
int main(int argc, char** argv)
{
    QApplication app(argc, argv);
    QPixmap pix;
    pix.load(":/mira.jpg");
    QLabel lbl;
    lbl.resize(pix.size());
    lbl.setPixmap(pix);
    lbl.show();
    return app.exec();
}

```

Как видно из листинга 2, сначала создается объект растрового изображения QPixmap. После этого вызовом метода load() в него загружается из ресурса файл mira.jpg. Следующим шагом является создание самого виджета надписи— объекта lbl класса QLabel. Затем вызовом метода resize () его размеры приводятся в соответствие с размерами растрового изображения. И, наконец, вызов метода setPixmap () устанавливает в виджете само растровое изображение.

При помощи метода setBuddyO виджет надписи может ассоциироваться с любым другим виджетом. Если текст надписи содержит знак & (амперсанд), то символ, перед которым он стоит, будет подчеркнутым. При нажатии этого символа совместно с клавишей <Alt> фокус перейдет к виджету, установленному методом setBuddy(). На рис. 3 показаны такие виджеты, а в листинге 3 приведен текст соответствующей программы.



Рис. 7.3. Использование знака &

Листинг 3. Файл main.cpp

```
#include <QtWidgets>
int main(int argc, char** argv)
{
    QApplication app(argc, argv);

    QWidget wgt;
    QLabel* plblName = new QLabel("&Name:");
    QLineEdit* ptxtName = new QLineEdit;
    plblName->setBuddy(ptxtName);

    QLabel* plblAge = new QLabel("&Age:");
    QSpinBox* pspbAge = new QSpinBox;
    plblAge->setBuddy(pspbAge);

    //Layout setup
    QVBoxLayout* pvbLayout = new QVBoxLayout;
    pvbLayout->addWidget(plblName);
    pvbLayout->addWidget(ptxtName);
    pvbLayout->addWidget(plblAge);
    pvbLayout->addWidget(pspbAge);
    wgt.setLayout(pvbLayout);

    wgt.show();

    return app.exec();
}
```

В листинге 3 виджет wgt класса QWidget является виджетом верхнего уровня, так как по умолчанию его конструктор присваивает указателю на виджет-предок значение 0. Виджеты не обладают способностью самостоятельного размещения виджетов-потомков, поэтому позже в нем осуществляется установка компоновки для вертикального размещения QVBoxLayout. В виджете надписи Name (Имя) в тексте символ N определен как символ для быстрого доступа. Затем создается виджет однострочного текстового поля. Далее, вызов метода setBuddy() связывает виджет надписи с созданным текстовым полем, используя указатель на поле в качестве аргумента. Аналогично происходит создание виджета надписи Age (Возраст), поля для ввода возраста (класса QSpinBox) и их связывание.

Индикатор процесса

Индикатор процесса (progress bar) — это виджет, демонстрирующий процесс выполнения операции и заполняющийся слева направо. Полное заполнение индикатора информирует о завершении операции. Этот виджет необходим в том случае, когда программа выполняет продолжительные действия, — виджет дает пользователю понять, что программа не зависла, а находится в работе. Он также показывает, сколько уже сделано и сколько еще предстоит сделать. Класс QProgressBar виджета индикатора процесса определен

в заголовочном файле `QProgressBar`. Обычно индикаторы процесса располагаются в горизонтальном положении, но это можно изменить, передав в слот `setOrientation()` значение `Qt::vertical`,— после этого он будет расположен вертикально.

Следующий пример демонстрирует использование индикатора процесса. При нажатии кнопки Step (Шаг) выполняется увеличение значения индикатора на один шаг. Нажатие кнопки Reset (Сбросить) сбрасывает значение индикатора. В основной программе, приведенной в листинге 4, создается виджет, показанный на рис. 4.

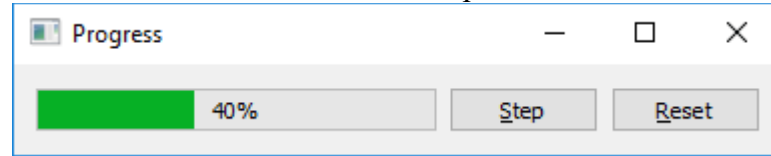


Рис. 4. Пример индикатора процесса

Листинг 4. Файл `main.cpp`

```
#include <QApplication>
#include "Progress.h"
int main (int argc, char** argv)
{
    QApplication app(argc, argv);
    Progress progress;
    progress.show();
    return app.exec();
}
```

В листинге 5 приведен файл `Progress.h`, который содержит определение класса `Progress`, унаследованного от `QWidget`. Класс содержит два атрибута: указатель на виджет индикатора процесса и целое значение, представляющее номер шага. В классе определены два слота: `slotstep ()` и `slotReset ()`. Первый предназначен для наращивания шага на единицу, а второй — для установки индикатора процесса в нулевое положение.

Листинг 5. Файл `Progress.h`

```
#include <QWidget>
class QProgressBar;
class Progress : public QWidget
{
    Q_OBJECT
private:
    QProgressBar* m_pprb;
    int m_nStep;

public:
    Progress(QWidget* pobj = 0);

public slots:
    void slotStep ();
    void slotReset();
};
```

В конструкторе класса (листинг 6) атрибуту `m_nstep` присваивается значение 0. После создания объекта индикатора `m_pprb` вызовом метода `setRange ()` задается количество шагов, равное 5, а метод `setMinimumwidth ()` устанавливает минимальную длину виджета индикации процесса, — в нашем случае мы запрещаем ему иметь длину менее двухсот пикселей. Вызов метода `setAlignment ()` с параметром `Qt::AlignCenter` переводит индикатор в режим

отображения процентов в центре (см. табл. 1). Затем создаются две кнопки: Step (Шаг) и Reset (Сбросить), которые соединяются со слотами slotStep() и slotReset(). В слоте slotStep() значение атрибута m_nstep увеличивается на 1 и передается в слот QProgressBar::setValue() объекта индикатора процесса. Слот slotReset() устанавливает значение атрибута m_nstep равным 0 и, вызвав слот QProgressBar::reset(), возвращает индикатор в исходное состояние. Для размещения виджетов-потомков горизонтально и слева направо необходимо установить в виджете Progress объект класса компоновки QVBoxLayout, предварительно добавив в него, в нужной очередности, виджеты-потомки.

Листинг 6. Файл Progress.cpp

```
#include <QtWidgets>
#include "Progress.h"
Progress::Progress(QWidget* pwgt/*= 0*/)
    : QWidget(pwgt)
    , m_nStep(0)
{
    m_pprb = new QProgressBar;
    m_pprb->setRange(0, 5);
    m_pprb->setMinimumWidth(200);
    m_pprb->setAlignment(Qt::AlignCenter);

    QPushButton* pcmdStep = new QPushButton("&Step");
    QPushButton* pcmdReset = new QPushButton("&Reset");

    QObject::connect(pcmdStep, SIGNAL(clicked()), SLOT(slotStep()));
    QObject::connect(pcmdReset, SIGNAL(clicked()), SLOT(slotReset()));

    //Layout setup
    QHBoxLayout* phbxLayout = new QHBoxLayout;
    phbxLayout->addWidget(m_pprb);
    phbxLayout->addWidget(pcmdStep);
    phbxLayout->addWidget(pcmdReset);
    setLayout(phbxLayout);
}

void Progress::slotStep()
{
    m_pprb->setValue(++m_nStep);
}

void Progress::slotReset()
{
    m_nStep = 0;
    m_pprb->reset();
}
```




Электронный индикатор

Класс QLCDNumber виджета электронного индикатора определен в заголовочном файле QLCDNumber. По внешнему виду этот виджет представляет собой набор сегментных указателей — как, например, на электронных часах. С помощью виджета электронного индикатора отображаются целые числа. Допускается использование точки, которую можно отображать между позициями сегментов или как отдельный символ, вызывая метод setSmallDecimalPoint() и передавая в него true или false соответственно. Количество отображаемых сегментов можно задать в конструкторе или с помощью метода setNumDigits(). В том случае, когда для отображения числа не хватает сегментов индикатора, отсылается

сигнал overflow ().

По умолчанию стиль электронного индикатора соответствует стилю Outline, но его можно изменить, передав методу `setsegmentstyle()` одно из следующих значений: `QLCDNumber::Outline`, `QLCDNumber::Filled` ИЛИ `QLCDNumber::Flat`. В табл. 2 показан внешний вид виджета для каждого из перечисленных стилей.

Таблица 2. Стили электронного индикатора

Константа	Внешний вид
Outline	
Flat	
Filled	

Электронный индикатор можно включать в режиме отображения двоичной, восьмеричной, десятичной или шестнадцатеричной систем счисления. Режим отображения изменяется с помощью метода `setMode()`, в который передается одно из следующих значений:

`QLCDNumber::Bin` (двоичная), `QLCDNumber::Oct` (восьмеричная), `QLCDNumber::Dec` (десятичная) или `QLCDNumber::Hex` (шестнадцатеричная). Также для смены режима отображения можно воспользоваться слотами `setBinMode()`, `setOctMode()`, `setDecModeO` и `setHexModeO` соответственно.

Следующий пример (листинг 7) демонстрирует электронный индикатор, отображающий шестнадцатеричные значения (рис. 5).

В листинге 7 создаются виджеты электронного индикатора (указатель `plcd`) и счетчика (указатель `pspb`). Затем вызовом метода `setsegmentstyle()` изменяется стиль сегментных указателей. Электронный индикатор вызовом метода `setMode()` и с параметром `QLCDNumber::Hex` переключается в режим шестнадцатеричного отображения. У элемента счетчика методом `setFixedHeight()` устанавливается неизменная высота, равная 30. После этого сигнал `valueChanged()` виджета счетчика соединяется со слотом `display()` электронного индикатора.



Рис. 5. Пример электронного индикатора

Листинг 7. Файл `main.cpp`

```
#include <QtWidgets>
int main (int argc, char** argv)
{
    QApplication app(argc, argv);
    QWidget wgt;
    QLCDNumber* plcd = new QLCDNumber;
    QSpinBox* pspb = new QSpinBox;

    plcd->setSegmentStyle(QLCDNumber::Filled);
    plcd->setMode(QLCDNumber::Hex);
```

```

pspb->setFixedHeight(30);

QObject::connect(pspb, SIGNAL(valueChanged(int)),
                plcd, SLOT(display(int))
                );

//Layout setup
QVBoxLayout* pvbxLayout = new QVBoxLayout;
pvbxLayout->addWidget(plcd);
pvbxLayout->addWidget(pspb);
wgt.setLayout(pvbxLayout);

wgt.resize(250, 150);
wgt.show();

return app.exec();
}

```

Выводы:

Мы познакомились с элементами отображения. Виджет надписи содержит информацию, предназначенную только для просмотра и не подлежащую изменению со стороны пользователя. Этот виджет способен отображать не только простой текст, но и текст в формате HTML. Кроме текстовой может отображаться и графическая информация.

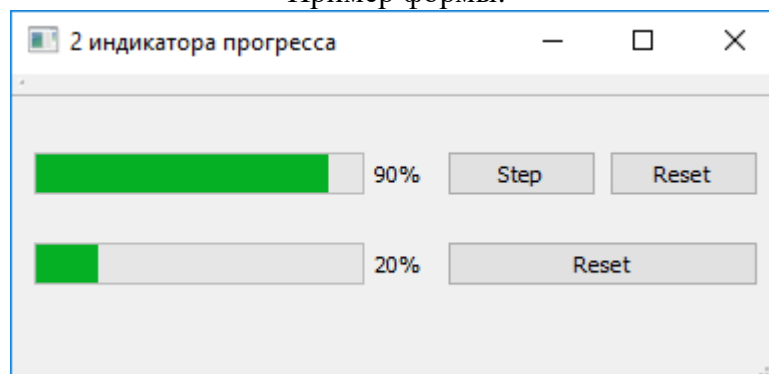
Виджет индикатора процесса — прекрасный элемент для оповещения пользователя о режиме работы приложения. Он незаменим для иллюстрации процесса выполнения продолжительных операций.

Виджет электронного индикатора по внешнему виду представляет собой набор сегментных указателей, как на электронных часах. Стиль виджета можно изменять и использовать различные режимы отображения чисел: в двоичной, десятичной, шестнадцатеричной и восьмеричной системах счисления.

Задания для самостоятельной работы:

1. Создать надпись, которая содержит HTML-разметку с заголовком H2 «Имя Фамилия», заголовком H3 «Группа».
2. Создать надпись, которая отображает значение индикатора прогресса.
3. Создать два индикатора прогресса. Первый от 0 до 10, второй увеличивается после достижения первым значения 10 (максимум также 10).

Пример формы:



4. К третьей задаче добавить электронный индикатор, который отобразит общее количество итераций.