

Archived:Using QStateMachine and QState

 Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template `{{ReviewForRemovalFromArchive|user=~~~~~|write your reason here}}`.

Qt Quick should be used for all UI development on mobile devices. The approach described in this article (using C++ for the Qt app UI) is deprecated.

Description

The Qt state machine framework provides classes for creating and executing state graphs in a hierarchical finite state machine. The state machine is represented by `QStateMachine` and the individual states are represented by `QState`. The corresponding transitions (and signals on which transitions occur) are set up with the `QState::addTransition` function. Once all transitions are set, the state machine can be started using `QStateMachine::start()`. The `QState::assignProperty()` function can be used to define assignment for a property of a `QObject`, performed when the state is entered.

In the following example, whenever a state transition occurs, the text property of a `QLabel` is changed to provide information about the new state.

Solution

MainWindow.h

```
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    MainWindow( QWidget *parent = 0 );
    ~MainWindow();

private:
    QStateMachine machine;
};
```

MainWindow.cpp

```
MainWindow::MainWindow( QWidget *parent )
    : QMainWindow( parent )
{
    QWidget* centralWidget = new QWidget(this);

    // Create a layout
    QVBoxLayout *layout = new QVBoxLayout( centralWidget );

    // Create push buttons to control the state machine
    QPushButton* produceButton = new QPushButton( tr("Produce"), centralWidget );
    QPushButton* consumeButton = new QPushButton( tr("Consume"), centralWidget );

    // Create a label which will display text when state change occurs
    QLabel* myLabel = new QLabel( tr("Unknown state"), centralWidget );
```

```
// Add all the widgets to layout
layout->addWidget( produceButton );
layout->addWidget( consumeButton );
layout->addWidget( myLabel );

centralWidget->setLayout( layout );
setCentralWidget( centralWidget );

// Create states and add them to machine
QState *produce = new QState();
QState *consume = new QState();
machine.addState( produce );
machine.addState( consume );

// Add state transitions and assign property to control the to the text label
produce->addTransition( produceButton, SIGNAL(clicked()), consume );
consume->addTransition( consumeButton, SIGNAL(clicked()), produce );
produce->assignProperty(myLabel, "text", tr("Produce Item"));
consume->assignProperty(myLabel, "text", tr("Consume Item"));

// Start the machine
machine.setInitialState( produce );
machine.start();
}
```