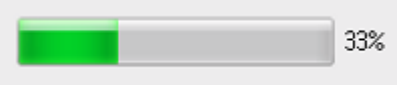
QProgressBar



위의 그림은 각각 macOS와 Windows7의 기본적인 진행 표시줄(progress bar)을 나타냅니다. 진행 표시줄은 시간이 걸리는 작업에 사용되는 위젯입니다.

QProgressBar 위젯은 수평, 수직의 진행 표시줄을 제공합니다. setMinimum()과 setMaximum() 메서드로 진행 표시줄의 최소값과 최대값을 설정할 수 있으며, 또는 setRange() 메서드로 한 번에 범위를 설정할 수도 있습니다. 기본값은 0과 99입니다.

setValue() 메서드로 진행 표시줄의 진행 상태를 특정 값으로 설정할 수 있고, reset() 메서드는 초기 상태로 되돌립니다.

진행 표시줄의 **최소값과 최대값을 모두 0**으로 설정하면, 진행 표시줄은 위의 그림과 같이 항상 진행 중인 상태로 표시됩니다. 이 기능은 다운로드하고 있는 파일의 용량을 알 수 없을 때 유용하게 사용할 수 있습니다.

예제

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QProgressBa
from PyQt5.QtCore import QBasicTimer


class MyApp(QWidget):

    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.pbar = QProgressBar(self)
```

```
        self.pbar.setGeometry(30, 40, 200, 25)

        self.btn = QPushButton('Start', self)
        self.btn.move(40, 80)
        self.btn.clicked.connect(self.doAction)

        self.timer = QBasicTimer()
        self.step = 0

        self.setWindowTitle('QProgressBar')
        self.setGeometry(300, 300, 300, 200)
        self.show()

    def timerEvent(self, e):
        if self.step >= 100:
            self.timer.stop()
            self.btn.setText('Finished')
            return

        self.step = self.step + 1
        self.pbar.setValue(self.step)

    def doAction(self):
        if self.timer.isActive():
            self.timer.stop()
            self.btn.setText('Start')
        else:
            self.timer.start(100, self)
            self.btn.setText('Stop')


if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = MyApp()
    sys.exit(app.exec_())
```

수평의 진행 표시줄과 푸시 버튼을 하나씩 만들었습니다.

푸시 버튼을 통해 진행 표시줄을 시작하고 멈출 수 있습니다.

설명

```
self.pbar = QProgressBar(self)
```

QProgressBar 생성자로 진행 표시줄을 하나 만들어줍니다.

```
self.timer = QBasicTimer()
```

진행 표시줄을 활성화하기 위해, 타이머 객체를 사용합니다.

```
self.timer.start(100, self)
```

타이머 이벤트를 실행하기 위해, start() 메서드를 호출합니다.

이 메서드는 두 개의 매개변수를 갖는데, 첫 번째는 종료시간이고 두 번째는 이벤트가 수행될 객체입니다.

```
def timerEvent(self, e):

    if self.step >= 100:

        self.timer.stop()
        self.btn.setText('Finished')
        return

    self.step = self.step + 1
    self.pbar.setValue(self.step)
```

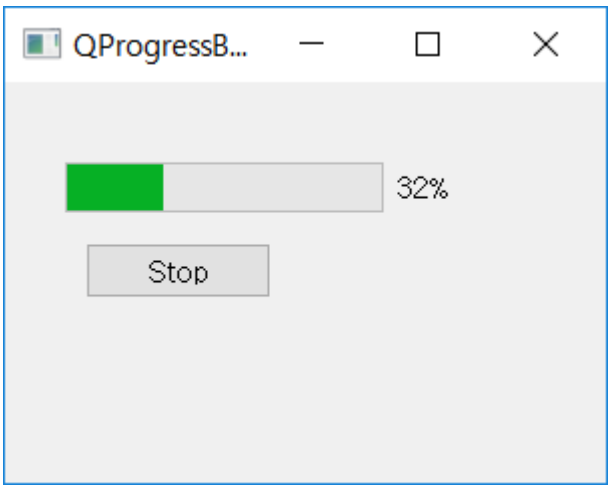
각각의 QObject와 그 자손들은 timerEvent() 이벤트 핸들러를 갖습니다. 타이머 이벤트에 반응하기 위해, 이벤트 핸들러를 재구성해줍니다.

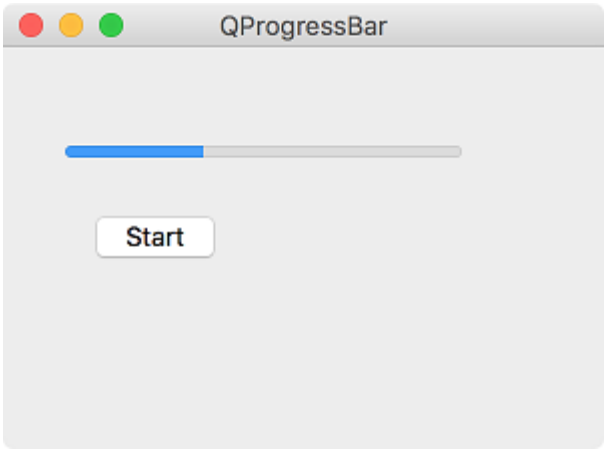
```
def doAction(self):

    if self.timer.isActive():
        self.timer.stop()
        self.btn.setText('Start')
    else:
        self.timer.start(100, self)
        self.btn.setText('Stop')
```

doAction() 메서드 안에서, 타이머를 시작하고 멈추도록 해줍니다.

결과


QProgressB...
32%
Stop


QProgressBar
Start