第6章：Qt高级特性与实战

高等程序设计 - Qt/C++

课程讲义

高等程序设计课程

2025 年 7 月 27 日

目录

Qt多线程概述

Qt多线程特点

- 基于QThread类
- 信号槽跨线程通信
- 线程安全的事件循环
- 自动内存管理
- 与Qt事件系统集成

多线程应用场景

- 耗时计算
- 网络请求
- 文件I/O操作
- 数据处理
- 实时更新UI

Qt多线程示例

```
#include <QThread>
#include <QObject>
#include <QDebug>

// worker thread class
class Worker : public QObject
{
    Q_OBJECT

public slots:
    void doWork() {
        qDebug() << "Worker thread:" << QThread::currentThread();

        // simulate time-consuming work
        for (int i = 0; i < 10; ++i) {
            QThread::msleep(100);
            emit progressUpdated(i * 10);
        }

        emit workFinished();
    }

signals:
    void progressUpdated(int value);
    void workFinished();
};

// main window class
class MainWindow : public QMainWindow
{
    Q_OBJECT

```

Qt绘图系统概述

Qt绘图组件

- QPainter - 绘图引擎
- QPaintDevice - 绘图设备
- QPen - 画笔
- QBrush - 画刷
- QFont - 字体
- QPixmap - 位图

绘图应用

- 自定义控件绘制
- 图表和图形
- 图像处理
- 动画效果
- 游戏开发

Qt绘图示例

```cpp
#include <QWidget>
#include <QPainter>
#include <QPen>
#include <QBrush>

// custom drawing widget
class DrawingWidget : public QWidget
{
    Q_OBJECT

protected:
    void paintEvent(QPaintEvent *event) override {
        QPainter painter(this);
        painter.setRenderHint(QPainter::Antialiasing);

        // set background
        painter.fillRect(rect(), QColor(240, 240, 240));

        // draw rectangle
        QPen pen(Qt::black, 2);
        QBrush brush(QColor(100, 150, 200));
        painter.setPen(pen);
        painter.setBrush(brush);
        painter.drawRect(10, 10, 80, 60);

        // draw ellipse
        pen.setColor(Qt::red);
        brush.setColor(QColor(200, 100, 100));
        painter.setPen(pen);
        painter.setBrush(brush);
        painter.drawEllipse(120, 10, 80, 60);

```

QtCharts概述

QtCharts特性

- 丰富的图表类型
- 交互式图表
- 实时数据更新
- 自定义样式
- 导出功能

图表类型

- 折线图（Line Chart）
- 柱状图（Bar Chart）
- 饼图（Pie Chart）
- 散点图（Scatter Chart）
- 面积图（Area Chart）

QtCharts示例

```
#include <QtCharts/QChartView>
#include <QtCharts/QLineSeries>
#include <QtCharts/QBarSeries>
#include <QtCharts/QBarSet>

QT_CHARTS_USE_NAMESPACE

class ChartDemo : public QMainWindow
{
    Q_OBJECT

public:
    ChartDemo(QWidget *parent = nullptr) : QMainWindow(parent) {
        setupCharts();
    }

private:
    void setupCharts() {
        // create chart
        m\_chart = new QChart();
        m\_chartView = new QChartView(m\_chart);
        m\_chartView->setRenderHint(QPainter::Antialiasing);

        // create data series
        m\_lineSeries = new QLineSeries();

        // add data
        for (int i = 0; i < 10; ++i) {
            m\_lineSeries->append(i, qrand() % 100);
        }

        // show chart
```

Qt网络编程概述

Qt网络模块

- QNetworkAccessManager - 网络访问管理器
- QNetworkRequest - 网络请求
- QNetworkReply - 网络响应
- QNetworkProxy - 网络代理
- QSslSocket - SSL套接字

支持的网络协议

- HTTP/HTTPS
- FTP
- WebSocket
- TCP/UDP
- SSL/TLS

Qt网络编程示例

```
#include <QNetworkAccessManager>
#include <QNetworkRequest>
#include <QNetworkReply>
#include <QJsonDocument>
#include <QJsonObject>

class NetworkDemo : public QMainWindow
{
    Q_OBJECT

private slots:
    void sendGetRequest() {
        QString url = "https://httpbin.org/get";

        QNetworkRequest request(QUrl(url));
        request.setHeader(QNetworkRequest::UserAgentHeader,
                          "Qt Network Demo/1.0");

        QNetworkReply *reply = m\_networkManager->get(request);

        connect(reply, &QNetworkReply::finished, [this, reply]() {
            if (reply->error() == QNetworkReply::NoError) {
                QString response = QString::fromUtf8(reply->readAll());
                qDebug() << "Response:" << response;
            } else {
                qDebug() << "Error:" << reply->errorString();
            }
            reply->deleteLater();
        });
    }

    void sendPostRequest() {
```

Qt数据库编程概述

Qt数据库支持

- QSqlDatabase - 数据库连接
- QSqlQuery - SQL查询
- QSqlTableModel - 表格模型
- QSqlRelationalTableModel - 关系表格模型
- QSqlQueryModel - 查询模型

支持的数据库

- SQLite
- MySQL
- PostgreSQL
- Oracle
- Microsoft SQL Server

Qt数据库编程示例

```
#include <QSqlDatabase>
#include <QSqlQuery>
#include <QSqlError>
#include <QSqlTableModel>
#include <QTableView>

class DatabaseDemo : public QMainWindow
{
    Q_OBJECT

public:
    DatabaseDemo(QWidget *parent = nullptr) : QMainWindow(parent) {
        setupDatabase();
        setupModel();
    }

private slots:
    void addRecord() {
        QString name = "New user";
        QString email = "newuserexample.com";int age = 25;QSqlQuery query;query.prepare("INSERT INTO users (name, email, age)
            VALUES (?, ?, ?)");query.addBindValue(name);query.addBindValue(email);query.addBindValue(age);if (query.exec())
            m_model->select(); // refresh modelqDebug() << "Record added successfully"; else qDebug() << "Add failed:" <<
            query.lastError().text();private:void setupDatabase() // create SQLite database connectionQSqlDatabase db =
            QSqlDatabase::addDatabase("QSQLITE");db.setDatabaseName("users.db");if (!db.open()) qDebug() << "Cannot open
            database:" << db.lastError().text();return;// create tableQSqlQuery query;query.exec("CREATE TABLE IF NOT EXISTS
            users (""id INTEGER PRIMARY KEY AUTOINCREMENT,""name TEXT NOT NULL,""email TEXT NOT NULL,""age INTEGER)");// insert
            example dataquery.exec("INSERT OR IGNORE INTO users (name, email, age) VALUES ""('Zhang San',
            'zhangsan@example.com', 25),""('Li Si', 'lisi@example.com', 30)");void setupModel() m_model = new
            QSqlTableModel(this);m_model->setTable("users");m_model->select();// set table headerm_model->setHeaderData(0,
            Qt::Horizontal, "ID");m_model->setHeaderData(1, Qt::Horizontal, "Name");m_model->setHeaderData(2, Qt::Horizontal,
            "Email");m_model->setHeaderData(3, Qt::Horizontal, "Age");m_tableView = new
            QTableView(this);m_tableView->setModel(m_model);setCentralWidget(m_tableView);private:QSqlTableModel
            *m_model;QTableView *m_tableView;;
```

总结

本章要点

- 掌握Qt多线程编程
- 学会使用Qt绘图系统
- 理解QtCharts的使用
- 掌握Qt网络编程
- 学会Qt数据库编程
- 理解Qt高级特性的应用
- 能够开发完整的Qt应用程序

实践建议

- 多练习实际项目开发
- 关注性能优化
- 学习最佳实践
- 参与开源项目