



1. Qt 项目结构

2. Qt 核心概念

- 信号与槽机制

3. Qt Widgets 基础

- QMainWindow
- QWidget
- QPushButton

4. 布局管理

- QVBoxLayout
- QHBoxLayout
- QGridLayout
- QFormLayout

5. 事件处理

- 鼠标事件
- 键盘事件
- 绘图事件

6. Qt 常用控件

- QLabel
- QLineEdit
- QTextEdit
- QComboBox
- QCheckBox
- QRadioButton

7. Qt 文件对话框

- QFileDialog

8. Qt 定时器

- QTimer

9. Qt 数据模型与视图

- QTableView
- QListView
- QStandardItemModel

10. Qt 绘图系统

- QPainter

11. Qt 多线程

- QThread

12. Qt 数据库

- QSqlDatabase
- QSqlQuery

13. Qt 国际化 (i18n)

- QTranslator

14. Qt 样式与美化

- QStyle
- QSS (Qt Style Sheet)

15. Qt 信号与槽高级用法

- Lambda表达式

16. Qt 资源管理

- qrc文件

17. Qt 动画

- QPropertyAnimation

1. Qt 项目结构


- **.pro文件**: 定义项目结构、依赖库、编译选项等。
- **源代码文件**: `main.cpp`、`mainwindow.cpp`、`mainwindow.h`、`mainwindow.ui`。

2. Qt 核心概念

- 信号与槽机制：

- 信号 (Signal)：对象状态发生变化时发出的通知。由 `emit` 触发。
- 槽 (Slot)：响应信号的函数，可以是任何普通成员函数。
- 连接信号与槽：


cpp

 Copy code

```
connect(senderObject, SIGNAL(signalName()), receiverObject, SLOT(slotName()));
```

- 新语法 (推荐)：

cpp


 Copy code

```
connect(senderObject, &SenderClass::signalName, receiverObject, &ReceiverClass::sl
```

3. Qt Widgets 基础

- **QMainWindow**: 主窗口类, 提供菜单栏、工具栏、状态栏等窗口元素。
- **QWidget**: 所有可视化组件的基类。
- **QPushButton**: 按钮组件, 常用方法:
 - `setText()`: 设置按钮文本。
 - `clicked()`: 按钮点击信号。

cpp


 Copy code

```
QPushButton *button = new QPushButton("Click Me", this);  
connect(button, &QPushButton::clicked, this, &MainWindow::onButtonClicked);
```

4. 布局管理

- **QVBoxLayout**: 垂直布局。
- **QHBoxLayout**: 水平布局。
- **QGridLayout**: 网格布局。
- **QFormLayout**: 表单布局, 适合两列布局 (标签+输入框)。

cpp


 Copy code

```
QVBoxLayout *layout = new QVBoxLayout;  
layout->addWidget(widget1);  
layout->addWidget(widget2);  
setLayout(layout); // 设置布局
```

5. 事件处理

- **事件对象**：继承自 `QEvent` 类。
- **事件重载方法**：
 - `mousePressEvent(QMouseEvent *event)`：鼠标点击事件。
 - `keyPressEvent(QKeyEvent *event)`：键盘按键事件。
 - `paintEvent(QPaintEvent *event)`：绘图事件。

cpp

 Copy code

```
void MainWindow::mousePressEvent(QMouseEvent *event) {  
    // 处理鼠标点击事件  
}
```

6. Qt 常用控件

- **QLabel**: 显示文本或图片的标签。
- **QLineEdit**: 单行文本输入框。
- **QTextEdit**: 多行文本输入框。
- **QComboBox**: 下拉菜单。
- **QCheckBox**: 复选框。
- **QRadioButton**: 单选按钮。

cpp


 Copy code

```
QLabel *label = new QLabel("Label Text", this);  
QLineEdit *lineEdit = new QLineEdit(this);
```

7. Qt 文件对话框

- **QFileDialog**: 文件选择对话框。

cpp


 Copy code

```
QString fileName = QFileDialog::getOpenFileName(this, "Open File", "", "Text Files (*.
```


8. Qt 定时器

- **QTimer**: 用于触发定时事件。

cpp


 Copy code

```
QTimer *timer = new QTimer(this);  
connect(timer, &QTimer::timeout, this, &MainWindow::onTimeout);  
timer->start(1000); // 每秒触发一次
```

9. Qt 数据模型与视图

- **QTableView、QListView**: 数据视图组件, 使用模型驱动。
- **QStandardItemModel**: 标准数据模型。

cpp


 Copy code

```
QStandardItemModel *model = new QStandardItemModel(2, 2, this);  
model->setItem(0, 0, new QStandardItem("Item 1"));  
tableView->setModel(model);
```

10. Qt 绘图系统

- **QPainter**: 用于绘制图形和文本。

cpp


 Copy code

```
void MainWindow::paintEvent(QPaintEvent *event) {  
    QPainter painter(this);  
    painter.drawRect(10, 10, 100, 100); // 绘制矩形  
}
```

11. Qt 多线程

- **QThread**: 线程类, 用于多线程编程。
- 创建新线程:

cpp

 Copy code


```
class Worker : public QObject {
    Q_OBJECT
public slots:
    void doWork() { /* 任务执行代码 */ }
};

QThread *thread = new QThread;
Worker *worker = new Worker;
worker->moveToThread(thread);
connect(thread, &QThread::started, worker, &Worker::doWork);
thread->start();
```

12. Qt 数据库

- QSqlDatabase: 数据库连接类。
- QSqlQuery: SQL查询执行类。

cpp


 Copy code

```
QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");  
db.setDatabaseName("test.db");  
db.open();  
  
QSqlQuery query;  
query.exec("CREATE TABLE people (id INT, name TEXT)");
```

13. Qt 国际化 (i18n)

- **QTranslator**: 翻译类, 用于加载翻译文件。

cpp


 Copy code

```
QTranslator translator;  
translator.load("myapp_zh.qm");  
qApp->installTranslator(&translator);
```

14. Qt 样式与美化

- **QStyle**: 控制控件的外观。
- **QSS (Qt Style Sheets)** : 类似于CSS, 用于控制控件的外观。

cpp


 Copy code

```
button->setStyleSheet("QPushButton { background-color: blue; color: white; });
```

15. Qt 信号与槽高级用法

- Lambda表达式:

cpp


 Copy code

```
connect(button, &QPushButton::clicked, [=]() {  
    qDebug() << "Button clicked!";  
});
```


16. Qt资源管理

- **qrc文件**: Qt资源文件, 用于嵌入图片、图标等资源。

cpp


 Copy code

```
QPixmap pixmap(":/images/logo.png");
```

17. Qt 动画

- **QPropertyAnimation**: 属性动画, 用于动画效果。

cpp

 Copy code

```
QPropertyAnimation *animation = new QPropertyAnimation(button, "geometry");  
animation->setDuration(1000);  
animation->setStartValue(QRect(0, 0, 100, 30));  
animation->setEndValue(QRect(250, 250, 100, 30));  
animation->start();
```