使用 QT 快速搭建跨平台 APP

文档库地址 https://xlang.link/documents/index.html

本文档讲解如何使用 X 和第三方的 QT 库进行快速的 QT 应用开发。

创建使用 QT 的 X 项目有两种方法：
1：使用向导创建 QT 项目，运行 XStudio 点击新建，左侧项目选择 xlang，右侧列表中选择 Q&X 桌面应用或者 Q&X Form 应用；

注： Q&X 桌面应用是使用动态创建控件的项目；
Q&X Form 应用是使用 QT 界面资源 form 文件的项目。

2：已有的项目通过包管理引入 QT 功能模块，在 XStudio 中载入项目，点击菜单[工具]->[包管理]，在列表中选择 QT5.9.1，然后点击[添加到当前项目]。

X 中使用 QT 的有着和 C++相同的开发步骤：

1).需要先实例化一个 QXApplication 对象

如下代码:

```
// new 一个 QXApplication 对象, 或者自定义一个对象继承自 QXApplication 来实现
QXApplication app = new QXApplication();

//需要判断返回值是否为 true
app.createQXApplication();
```

Application 创建之后就可以执行各类窗口和组件的初始化工作

2).载入 ui 界面文件。

当 app.createQXApplication();成功以后

```
// new 一个 Dialog 对象,
QXDialog newDlg = new QXDialog();
// 使用 dialog 的 load 方法来加载 ui 文件
if (newDlg.load("ui/dialog.ui") == false){
    return false;
}
```

注意:载入的文件路径如果不在程序目录或者当前目录需要指定完整路径:

UI 文件内容:

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Dialog</class>
 <widget class="QDialog" name="Dialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>458</width>
    <height>365</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Dialog</string>
  </property>
  <widget class="QPushButton" name="btnOk">
   <property name="geometry">
    <rect>
     <x>320</x>
     <y>300</y>
     <width>101</width>
     <height>41</height>
    </rect>
   </property>
   <property name="text">
    <string>Ok</string>
   </property>
  </widget>
 </widget>
 <resources/>
 <connections/>
</ui>
```

3).初始化结束待，所有初始化工作完成后，需要调用 app .run(); 保持整体框架的消息循环.

因此完整代码为:

```
// new 一个 QXApplication 对象, 或者自定义一个对象继承自 QXApplication 来实现
QXApplication app = new QXApplication();

//需要判断返回值是否为 true
if (app.createQXApplication()){
    // new 一个 Dialog 对象,
    QXDialog newDlg = new QXDialog();
    // 然后使用 dialog 的 load 方法来加载 ui 文件
    if (newDlg.load("ui/dialog.ui") == false){
        // 载入 UI 文件失败
        return false;
    } else{

    }
    // 进入消息循环
    app.run();
}
```

4).进行控件事件交互.

前面的三步， 程序已经可以显示一个窗口，接下来还需要对窗口上的按钮事件响应.

ui 文件中有一个名为 btnOk 的 QPushButton，要响应按钮事件，需要在程序初始化时对按钮的事件进行设置。

1.获取控件.

在 Dialog.load(ui 文件) 成功之后， 通过 ui 文件中定义的控件名获取控件:（黄底代码).

```
....
    // 然后使用 dialog 的 load 方法来加载 ui 文件
    if (newDlg.load("ui/dialog.ui") == false){
        // 载入 UI 文件失败
        return false;
    }
    // 获取按钮控件
    QXPushButton btnOk = (QXPushButton)attachByName(new QXPushButton(), "btnOk");
....
```

获取 btnOk 对象， 若发生异常或者 btnOk 为空，则表明类型或者名称与 ui 文件中的定义不符;

2. 为控件设置事件响应.

这里需要用到 QXPushButton 的 setOnClickListener 方法， 该方法为控件设置点击事件的接收装置.

接口原型:

```
static class onClickListener{
    // 点击事件
    void onClick(QXObject obj, bool checked){
    }
    // 触发事件
    void onToggle(QXObject obj, bool checked){
    }
    // 按下
    void onPress(QXObject obj){
    }
    //松开事件
    void onRelease(QXObject obj){
    }
};
```

为了方便，这里使用一个临时类来设置控件响应事件,如下(黄色底)代码:

```
btnOk.setOnClickListener(new onClickListener(){
    void onClick(QXObject obj, bool checked)override{
        _system_.output("on click");
    }
});
```

在 setOnClickListener 的参数中 new 一个从 onClickListener 派生的临时类， 并重写其方法 onClick 。

注意：重写的方法推荐加上 override 关键字，override 关键字表明这是个重写方法，编译器会强制检查重写关系，没有 override 关键字时，若重写的方法与原型不一致时，编译器会认为这是一个新的方法。

因此本节完整代码为:

```
// new 一个 QXApplication 对象, 或者自定义一个对象继承自 QXApplication 来实现
QXApplication app = new QXApplication();

//需要判断返回值是否为 true
if (app.createQXApplication()){
    // new 一个 Dialog 对象,
    QXDialog newDlg = new QXDialog();
    // 然后使用 dialog 的 load 方法来加载 ui 文件
    if (newDlg.load("ui/dialog.ui") == false){
        //其他处理
        return false;
    } else{
        QXPushButton btnOk = (QXPushButton)attachByName(new QXPushButton(), "btnOk");
        btnOk.setOnClickListener(new onClickListener(){
            void onClick(QXObject obj, bool checked)override{
                _system_.output("on click");
            }
        });
    }
    // 进入消息循环
    app.run();
}
```

要点说明:

使用临时类:

```
btnOk.setOnClickListener(new onClickListener(){
    void onClick(QXObject obj, bool checked)override{
    }
});
```

5).多线程交互与事件通知

QT 界面操作只能在 UI 线程中执行，并且在大多数场景下，需要使用多线程，这里主要来介绍如何使用 X 原生的多线程来完成与 QT 的交互；

我们需要在点击按钮的时候进行 IO 相关操作， 例如 Socket 连接远程服务器的操作，该操作可能会导致线程阻塞，使界面停止响应，因此不能在 UI 线程中执行，而是需要放到新线程中处理:

例:我们需要在按钮的 onClick 事件中使用 TCP 连接 ip 为 192.168.0.110 的 8088 端口,需要将上面代码修改如下（黄色底部分代码）：

```
....
btnOk.setOnClickListener(new onClickListener(){
  void onClick(QXObject obj, bool checked)override{
      new Thread(){
        void run()override{
         StreamSocket socket = new StreamSocket();
         bool succ = socket.connect("192.168.0.110", 8088);
        }
      }
    }.start();
});
....
```

代码说明： new 一个从 Thread 派生的临时类，并实现其 run 接口方法，并调用 start 方法运行线程；在 run 中的代码将在新线程中被执行。

要点说明：

创建新线程:

```
new Thread(){
      void   run()   override{
      }
}.start();
```

接下来将 Socket 连接的结果反馈到界面上，而工作线程无法直接操作 UI，因此需要将这部分代码运行在 UI 线程中，这里使用 Q&X 封装的 runOnUi 来将代码执行在 UI 线程，btnOk 是在外层作用域定义的，但是由于 X 默认支持闭包操作（无需声明捕获），因此 btnOk 可以在 Runnable 的临时类成员方法中直接使用，如下：

```
....
new Thread(){
    void run()override{
        StreamSocket socket = new StreamSocket();
        bool succ = socket.connect("127.0.0.1", 80);
        runOnUi(new Runnable(){
            void run()override{
                // 将结果显示在按钮上
                btnOk.setText(succ ? "OK" : "FAILED");
            }
        });
    }
}.start();
....
```

要点说明：

1.工作线程与 UI 线程的任务交互；

```
runOnUi(new Runnable(){
            void run()override{
                // 将结果显示在按钮上
                btnOk.setText(succ ? "OK" : "FAILED");
            }
        });
```

注意：runOnU 是 QXWidget 的成员方法，因此如果不在 QXWidget 或者其派生类的作用域内使用，需要指定对象，例如 MyDialog.runOnUi(...);

2.闭包操作；

支持跨作用域， 跨线程， 并且无需担心生命周期.

有任何疑问请进入 

QQ 群：591392649，进行交流探讨.