XStudio 使用手册

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

目录：

- 简介
- 布局简介
- XStudio 的下载与安装
  - 一、从官网下载
  - 二、从源码编译
- 个人偏好设置
  - 一、外观设置
  - 二、快捷键
  - 三、输出
  - 四、代码编辑器设置
  - 五、项目相关
  - 六、智能感知组件
- 第一个项目
  - 一、从向导新建项目
  - 二、打开已有的项目
  - 三、加载在线代码示例
- 项目属性
  - 一、设置项目生成的类型
  - 二、设置目标路径
  - 三、设置目标可执行文件图标
  - 四、设置依赖库和库目录
  - 五、编译选项
  - 六、调试设置
  - 七、额外的构建步骤
- UI 设计器
- 编辑代码
- 添加文件
- 添加类、接口
- 编译生成
- Native 混合编程
- 调试&运行
- 远程调试
- 包管理器

简介

XStudio 是使用 xlang 语言开发的基于 Qt 的可视化集成开发环境。

目前主要用于开发 xlang 项目，其集成设计、开发、调试为一体，并支持跨主流操作系统和平台，Windows(微软视窗系统)、Linux、Darwin（苹果公司 MacOSX 操作系统），支持的架构有 X86 、X86_64，且具备个性化定制和远程调试功能，自动更新等实用功能。

XStudio 最早立项于 2018 年 6 月初，时值 xlang 语言的开发阶段，由于没有一款适用的开发和调试环境，于是便在 xlang 开发的同步进行 XStudio 开发工作，并于 2018 年 11 月在 Github 开放了全部源代码（仓库地址 https://github.com/ixlang/XStudio），并发布了第一个版本，至今约历时一年，经过不断努力，XStudio 日趋完善，成为官方支持的 xlang 的主要开发工具。

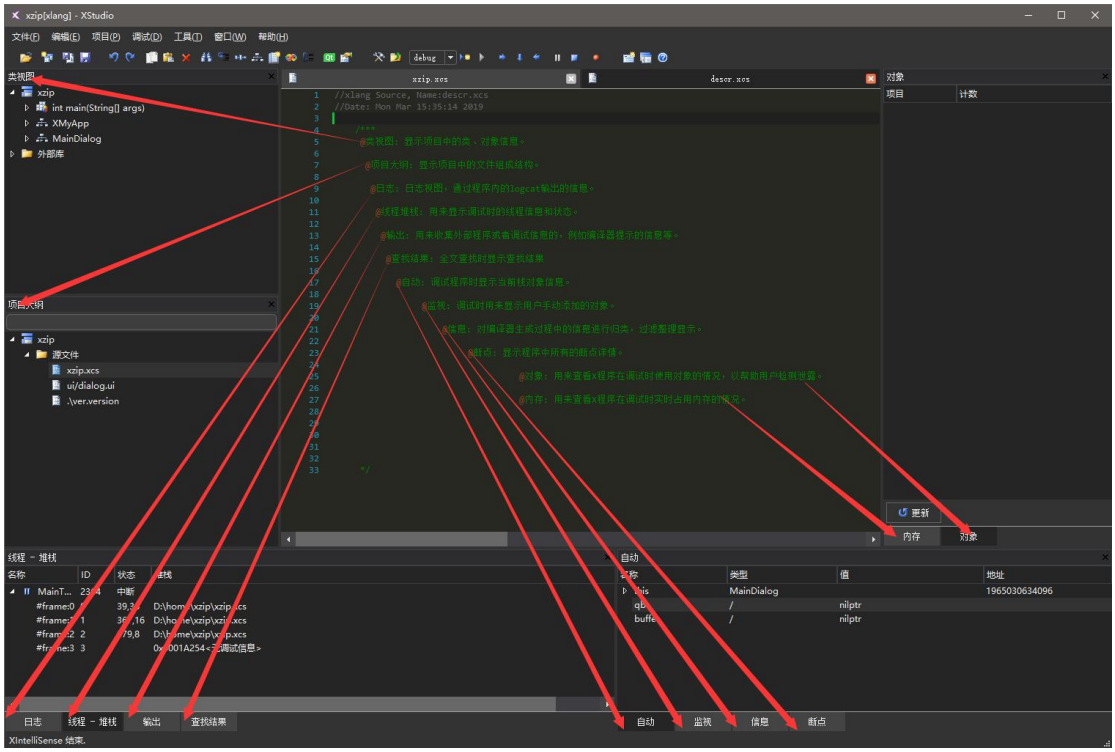
重要部分都被设计为模块化的\插件形式的，同时 XStudio 使用 C++编写的 IntelliSence 智能提示插件，为 xlang 代码编辑带来了智能提示、自动完成等辅助工作，极大提高了编写 xlang 程序的体验，因其预留了大部分的接口，所以可以轻松的进行二次开发。

XStudio 还支持 xlang 语言与 C++的混合开发，自定义的构建步骤可以在开发 xlang 的同时轻松同步开发 C/C++项目，并结合 Qt 官方的 UI 设计器，在需要编辑 UI 时，可直接使用 qtdesigner。

XStudio 是开源的，其全部代码量约为 3 万行 xlang 代码，其没有任何开源协议限制，可用于任何形式的二次开发和发布。

本文档旨在更好的帮助用户使用 XStudio 以及解决使用 XStudio 中遇到的问题，有任何疑问欢迎加入 QQ 群 591392649 进行讨论。

布局简介



类视图: 显示项目中的类、对象信息。
项目大纲: 显示项目中的文件组成结构。
日志: 日志视图，通过程序内的 logcat 输出的信息。
线程堆栈: 用来显示调试时的线程信息和状态。
输出: 用来收集外部程序或者调试信息的，例如编译器提示的信息等。
查找结果: 全文查找时显示查找结果
自动: 调试程序时显示当前栈对象信息。
监视: 调试时用来显示用户手动添加的对象。
信息: 对编译器生成过程中的信息进行归类，过滤整理显示。
断点: 显示程序中所有的断点详情。
对象: 用来查看 X 程序在调试时使用对象的情况，以帮助用户检测泄露。
内存: 用来查看 X 程序在调试时实时占用内存的情况。

XStudio 的下载与安装

一、从官网下载

使用任意一款浏览器打开 http://xlang.link/ ，下载你操作系统平台对应的 XStudio，目前支持 5 种操作系统和处理器架构的版本。

Windows 系统下载后是 7z 或者 Zip 压缩包，该压缩包无需安装，将压缩包解压到指定目录，找到 XStudio.exe 运行即可，注意不要解压到受写权限保护的目录、以及不要解压到 Program Files 目录等受系统保护的目录下，因为 XStudio 的配置文件在当前目录，如解压到受写权限保护的目录下，则有可能会导致写配置文件失败。

Linux 系统的用户下载到的是一个 tar.gz 压缩包， 可使用命令行 tar -zxvf xlang_xxx.tar.gz 进行解压，到解压后的目录下找到 XStudio 文件，双击运行即可，如不能运行，请检查是否有执行权限，若无执行权限，使用 chmod +x 授予执行权限.

Macos 系统下载之后是 DMG 文件，双击 DMG 文件，按照 MAC 软件常规安装流程，将 XStudio 安装到 Application 目录即可.

需要注意的是，在 MacOSX 下 XStudio 第一次启动会出现一个错误提示并退出，重新运行即可，另外由于 MacOSX 平台上选用的 QT 版本存在 BUG，在向导界面点击**[刷新在线示例]**时可能会导致闪退。

二、从源码编译

从源码编译需要按照以下步骤.

1.先从官网下载并安装 XStudio(参考从官网下载).

2.从 https://github.com/ixlang/XStudio 克隆或者下载 XStudio 源代码.

3.运行第一步下载安装好的 XStudio，打开第二步下载的 XStudio 中的项目文件 XStudio.xprj。

4.将第一步解压后的整个文件夹复制一个副本。

5.点击菜单**[项目]** -> **[属性]**，打开项目属性页，选择 **[项目属性]** ，将输出位置更改为第四步创建的文件夹副本，以使编译输出的目标文件能覆盖目录中原来的 XStudio 可执行文件.

6.按 F7 进行编译组建，F5 调试运行。

注意：如果不能运行请检查输出的目标文件路径是否正确，是否在 XStudio 的目录中。

个人偏好设置

一. 外观设置

点击菜单[工具]->[选项], 选择[环境], 右侧的外观样式可有三种选择,分别为系统默认，浅色，深色，选择自己喜欢的样式，点击关闭即可生效。

二. 快捷键

点击菜单[工具]->[选项], 选择[加速键], 右侧的会出现所有菜单的快捷键设置，第一项为键盘布局，如果你之前是熟悉 VisualStudio 或者 eclipse 的用户，则可以直接在键盘布局中选择自己熟悉的预设，否则可能需要手动逐个更改按键设置.

注意:在 1.0.18.2764 及以前的版本更改快捷键设置需要用键盘手动输入按键名称。

三. 输出

点击菜单[工具]->[选项], 选择[输出], 右侧的会出现三项分别为：

1. 调试时输出线程状态

   说明:在调试时会在输出窗口打印所有线程的活动，如线程的创建、激活、休眠、退出等事件，由于 X 的 timer 是多线程的，所以如果有 timer 运行的话，输出视图会被大量的线程事件输出，此时如果不关心线程事件，则推荐关闭此项。

2. logcat 最大数量

   说明:logcat 数量巨大时会引起响应迟钝，推荐数量不要超过 10000。

3. 调试/运行时重定向 StdOut 到输出窗口

   说明:在 linux 或者 macos 系统上开发的程序在默认调试运行时不会像 Windows 一样有 Console 窗口弹出，所以控制台的输出可能无法查看，所以建议在 Linux 或者 macos 系统下保持此选项打开。

四. 代码编辑器设置

点击菜单[工具]->[选项], 选择[编辑器]， 其选项有括号、引号配对的自动输入，智能缩进开关、替换提示去人和代码编辑器中的行号显示控制。

五. 项目相关

点击菜单[工具]->[选项], 选择[项目]，用户可自行查看选项。

六. 智能感知组件

点击菜单[工具]->[选项], 选择[智能感知]，仅有一个是否启用的开关。

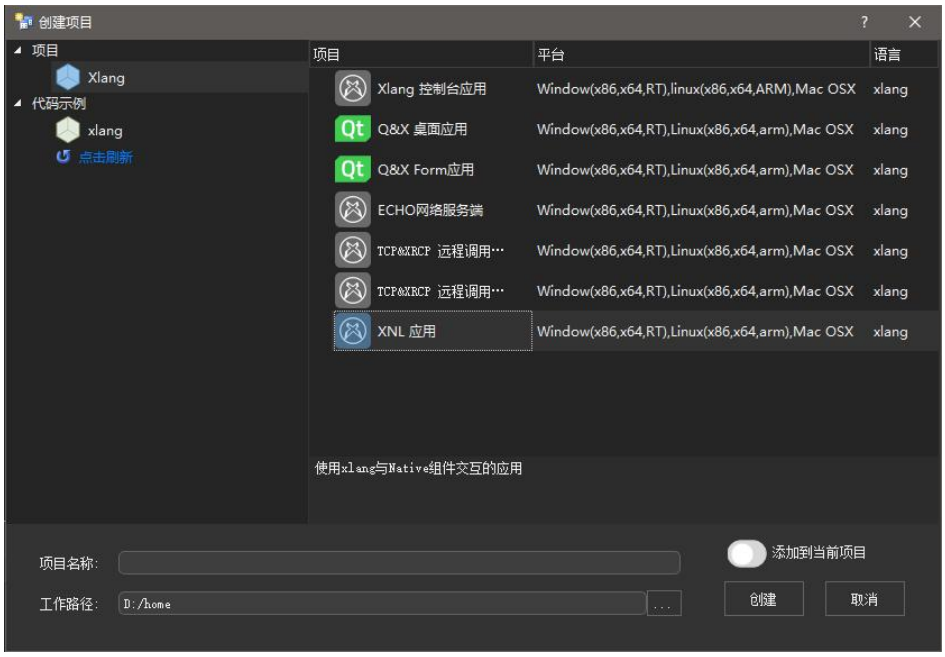
智能感知是为了更加方便的开发，提供包括构建项目大纲视图、提供智能提示、自动完成、重写方法等一切在开发期间的辅助工作，该模块是由插件的形式提供，详情查看文件 XIntelliSense.xcs 和 ProjectPropManager.xcsm。

注意：该插件尚在 beta 阶段，由于该插件在运行时由于占用内存较大，因此系统内存较小或者性能不够的情况下，可选择关闭此功能。

第一个项目

一、从向导新建项目

运行 XStudio，在 XStudio 的起始页点击**[新建]**，或者点击菜单**[文件]->[新建项目]**，打开向导窗口，如下图示:



左侧点击 xlang，并在右侧选择一个项目模板，填写好项目名称后点击加载即可.

二、打开已有的项目

运行 XStudio，在 XStudio 的起始页点击**[打开...]**，或者点击菜单**[文件]->[打开]**，选择要打开的项目中的*.xprj 文件即可，注意 xlang 的项目文件扩展名为.xprj；

三、加载在线代码示例

运行 XStudio，在 XStudio 的起始页点击**[新建]**，或者点击菜单**[文件]->[新建项目]**，打开向导窗口，如上图示，左侧选择代码示例下面的的任何一个选项，或者选择[点击刷新]来重新加载在线的代码示例，加载之后在右侧选择一个示例项目，点击[创建]即可，若第一次加载会从 github 上下载对应项目。

注意:由于所有的代码示例都是存放于 github 之上，国内链接 github 速度可能较慢，有条件的用户可选择在浏览器中打开 github 代码仓库 https://github.com/ixlang/examples，下载 update.json 和 zip 文件之后放到 XStudio 的 examples 的目录下并将 update.json 更名为 project.list 即可。

项目属性

一、设置项目生成的类型

XStudio 载入项目以后，点击菜单[项目]->[属性]，左侧选择[项目属性]，右侧的[项目类型]有以下几项可供选择：

1. xlang 可执行文件(-c)

   编译为 xlang 的字节码程序，该程序无法独立运行，需要在系统中安装 x 运行环境才可以运行。

2. 可执行文件(-ce)

   编译为能够独立运行于操作系统的可执行文件(PE，ELF，MACHO)。

   该项可支持交叉生成，需要选择目标操作系统与处理器架构。

   目前支持的操作系统有 Windows（微软视窗系列），Linux，以及 Darwin（苹果公司的 MacOSX）。

3. xlang 静态库(-lix -c)

   编译为可复用的静态库，注意编译静态库时编译器只检查语法，而不做逻辑检查，因此需要先测试再编译为静态库，否则存在错误的话将会在链接时被编译器报告。

二、设置目标路径

XStudio 载入项目以后，点击菜单[项目]->[属性]，左侧选择[项目属性]，编辑右侧的输出位置，默认为$(ProjectDir)/$(Arch)/$(Configure)，更改此路径可改变编译输出目录。

三、设置目标可执行文件图标

点击菜单[项目]->[属性]，左侧选择[项目属性]，可见应用程序图标的设置选项，该位置直接填写文件路径即可，支持的文件格式为.ico 或者 png。

注意:由于 linux 和 macos 系统不支持为可执行文件设置图标。

因此若项目类型为**[可执行文件(-ce)]**时，则该选项仅影响 Windows 下的可执行文件图标。

若项目类型为**[xlang 可执行文件(-c)]**时，需要安装 X 运行环境才能正确显示图标。

四、设置依赖库和库目录

点击菜单[项目]->[属性]，左侧选择[项目设置]，库目录为编译器查找静态库的默认路径，多个路径使用;（分号）相隔，依赖库则为静态库(.lix)格式的文件.

引用目录为查找 include 或者 require 文件的目录。

五、编译选项

点击菜单[项目]->[属性]，左侧选择[编译选项]。

忽略所有警告：打开此项目则编译器会禁止所有警告信息输出。

生成调试信息：对需要调试的程序一定要打开此项，否则调试时无法找到对应到源文件。

生成可调试版本：该选项仅对类型为**[可执行文件(-ce)]的项目有效，**若需要开启调试功能的项目必须选择此项，否则将造成目标程序不可调试。

版本随构建次数增加：项目中需要有版本文件，对于已载入的项目可以使用菜单**[文件]->[新建项目]**功能打开向导，选择 xlang 并选择版本信息添加到当前项目中，则该项目每编译一次，版本文件中的版本号会增加 1。

六、调试设置

点击菜单[项目]->[属性]，选择[调试];

启动命令: 为将要运行的可执行文件路径。

命令行参数: 程序运行时的命令行，注意:该命令行包含 arg[0]，所以一般情况下第一个为可执行文件路径或者名称。

工作目录：设置调试运行时的工作目录。

七、额外的构建步骤

点击菜单[项目]->[属性]，选择[构建步骤];

1. makefile 构建参数

   当 XStudio 在构建时检测到项目中存在 makefile，则会调用 make 对其进行编译，构建参数为 make 的参数，如可以在这里填写 -j4 ，则 make 以多线程编译。

2. makefile 清理参数

   当 XStudio 使用清理命令时检测到项目中存在 makefile，则会调用 make clean 对其进行清理，构建参数为 make 的参数。

3. 构建前动作

   XStudio 在构建前运行的批处理脚本，支持的文件为 bat (Windows)， sh (linux 和 maxos), 若存在此文件，则 XStudio 在构建前会自动调用，并将结果打印在输出视图。

4. 构建后动作

   XStudio 在构建结束时运行的批处理脚本，支持的文件为 bat (Windows)， sh (linux 和 maxos), 若存在此文件，则 XStudio 在构建结束之后会自动调用，并将结果打印在输出视图。

5. 清理前动作

   XStudio 在使用菜单[项目]->[清理]时运行的批处理脚本，支持的文件为 bat (Windows)，sh (linux 和 maxos), 若存在此文件，则 XStudio 在执行清理之前会自动调用，并将结果打印在输出视图。

6. 清理后动作

   XStudio 在清理动作完成之后运行的批处理脚本，支持的文件为 bat (Windows)， sh (linux 和 maxos), 若存在此文件，则 XStudio 在执行清理之后会自动调用，并将结果打印在输出视图。

UI 设计器

XStudio 的项目大纲中选择.ui 文件并单击鼠标右键，会出现**[在 UI 设计器中打开]**的选项，点击此选项则可以打开 UI 设计器。

点击菜单**[工具] -> [QT Designer(UI 设计器)]** 同样可以运行设计器。

xlang 使用了第三方界面库 qt，因此界面设计器为 qtdesigner，详情查阅 Qt 官方文档或者 qtdesigner 手册。

编辑代码

在**[项目大纲]**视图或者**[类视图]**中选择要编辑的文件或者对象，即可打开源码/文本编辑器，编辑器根据 xlang 的语法特性和结构对代码进行了高亮显示。

源代码中的注释分为三种：

1. 单行注释
   以 // 开始 ，遇到 \n 终止。

2. 多行注释
   以/* 开始 ，遇到 */ 终止

3. 文档注释
   以/** 开始， 遇到 */ 终止

文档注释与多行注释的不同是，文档注释可以高亮显示注释中的关键字，例如参数说明、返回值说明和需要特别注意事项。

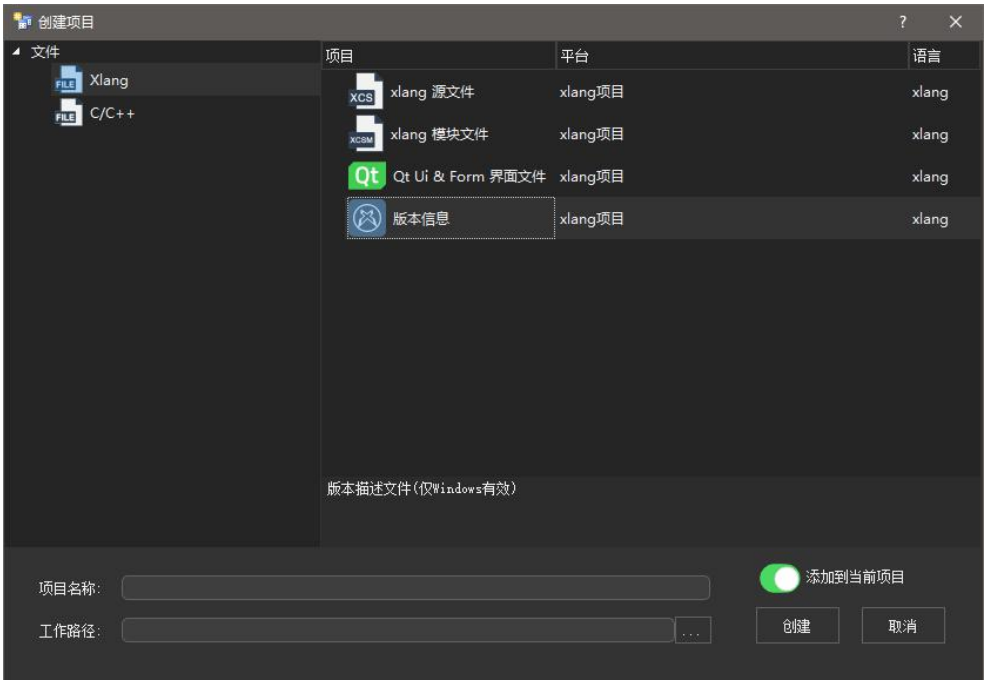
```
1  // xlang
2
3  using{ FileSystem; FileStream;};
4
5
6  class MainDialog : QXDialog{
7      /* 多行注释
8          主窗口
9      */
10     QXPushButton btnDir, btnComp, btnFile, btnDecomp;
11     QXLineEdit directory, file;
12     QXTreeView filecontent = new QXTreeView();
13     QXProgressBar progressBar;
14     QXLabel label;
15
16
17     //单行注释#########################################################################
18
19     /*** 文档注释
20     @caution 为程序设置一个出错回调，
21         程序中可能出现的没有处理的异常会导致程序终止，
22         这个接口将在程序终止前被调用， 用于记录异常信息
23     */
24     class CrashReport : ICrashHandler{
25         void onCrash(String message)override{
26             try{
27                 new FileOutputStream("crash.log").write(message.getBytes());
28             }catch(Exception e){
29
30             }
```

提示：智能感知功能开启以后，在编辑器中的输入都将得到辅助。

添加文件

1.添加新文件:

XStudio 载入项目以后可使用菜单[文件]->[新建项目] 来添加一个新文件。



2.添加已有的文件：

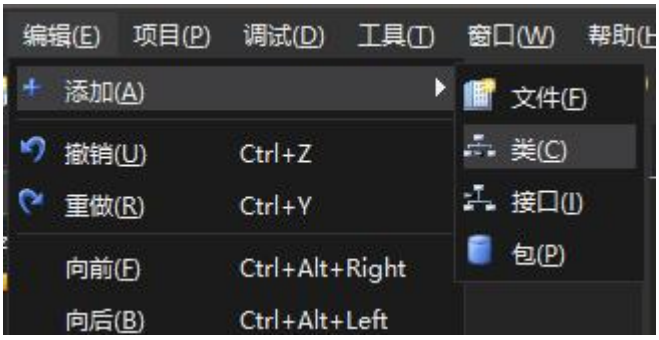
在[项目大纲]视图中右键点击任意项，或者点击菜单[编辑]->[添加]->[文件]，在弹出菜单中选择[添加项]浏览现有文件并添加到项目中。

若要删除项目中的文件，则在[项目大纲]视图中右键点击要删除的项， 再点击移除或者删除，移除为仅从项目中移除而不删除文件，删除为从项目中移除并删除磁盘上的文件，具体操作是会有确认提示。

说明，添加文件之后构建项目时会自动保存项目。

添加类、接口

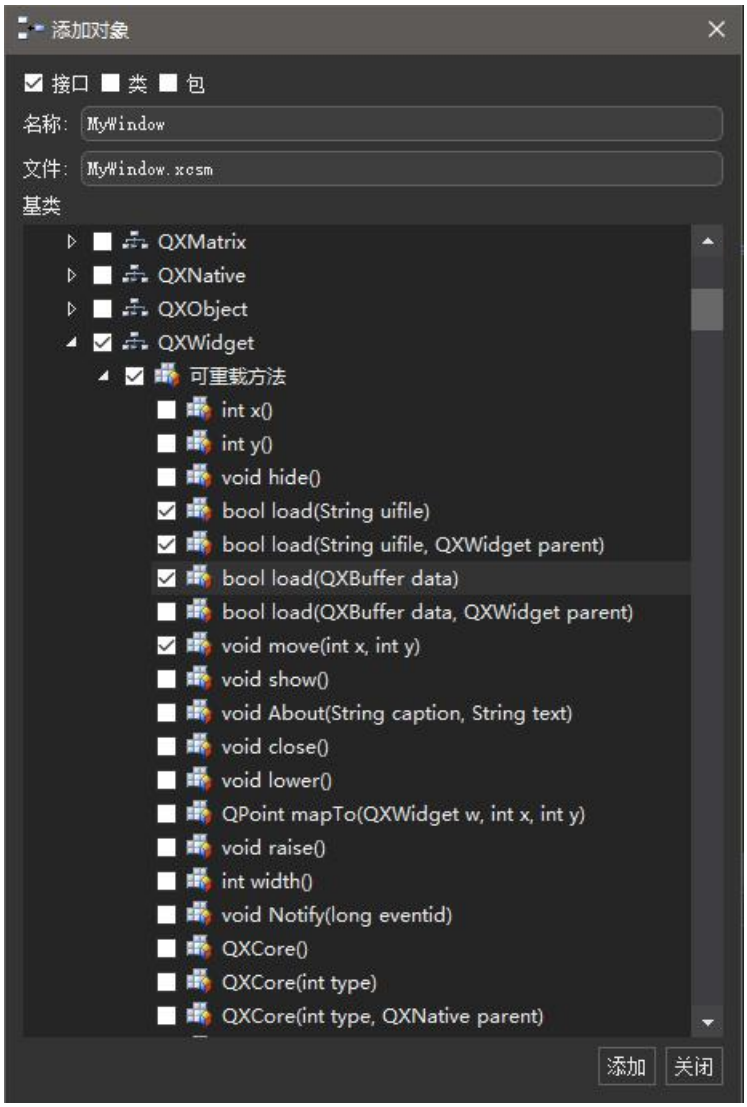
选择菜单[编辑]->[添加]即可选择添加类，接口包等。



点击添加任一项即可打开添加对象窗口。

填写对象名称和生成的文件名\路径

在基类的列表中勾选一项作为基类，并勾选需要重写的方法即可生成对象并添加到项目中。



编译生成

择使用菜单[项目]->[组建项目] 来为当前配置生成目标文件。
注意，在菜单 [调试] -> [调试]或者[运行时]， 若目标文件不存在，将会自动编译生成。

暂不支持批量生成功能.

Native 混合编程

当 xlang 中存在 native 方法时，使用[菜单]->[项目]->[生成 Native 文件] 即可生成 x 代码中对应的调用 C 语言的方法代码结构，手动编辑该文件内容，并编写构建步骤脚本，即可使 C 项目和 x 保持同步编译。

调试&运行

使用菜单[调试]->[调试]可对生成的文件进行调试运行，在调试运行期间，可随时在代码中切换断点，当有线程触发断点时，调试器则收到中断信号。

单步步入： 使线程运行一步，如果当前行存在子函数调用，则进入子函数中。

单步步过： 使线程运行一步，跳入下一行。

单步步出： 使线程运行到外层（调用者代码）方法中。

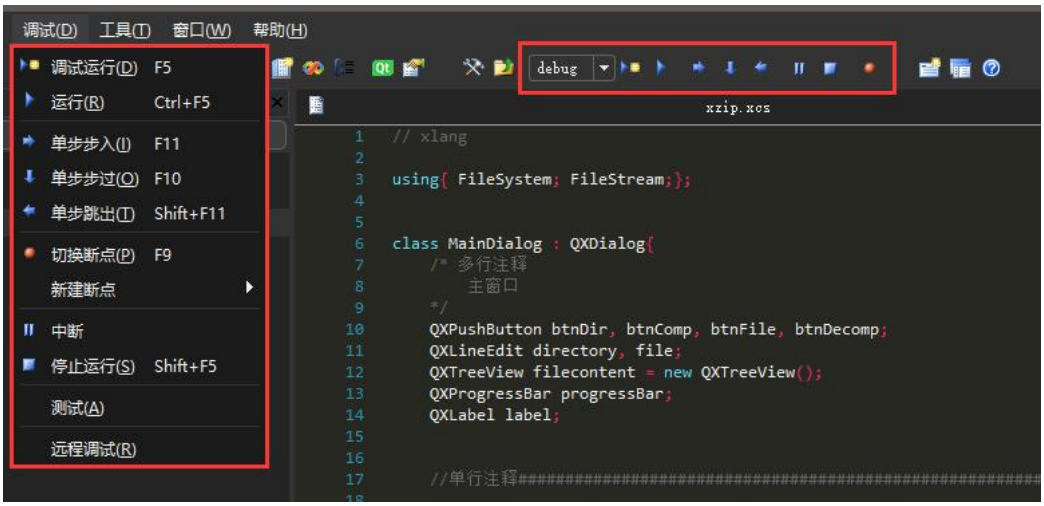
中断：中断当前线程， 需要在线程栈视图中选中要中断的线程。

停止运行： 结束进程或者分离调试器。

在调试过程中需要通过自动窗口或者监视窗口来了解程序运行的动态情况。

自动窗口中显示了当前栈中的对象。
监视窗口中支持用户手动添加对象名称来查看对象状态。

注意：若目标文件未开启可调试功能，则调试器会报告不可调试的错误， 此时需要更改项目设置并重新生成。



远程调试

用菜单[调试]->[远程调试]， 输入 IP 和端口， 如 127.0.0.1:23326。

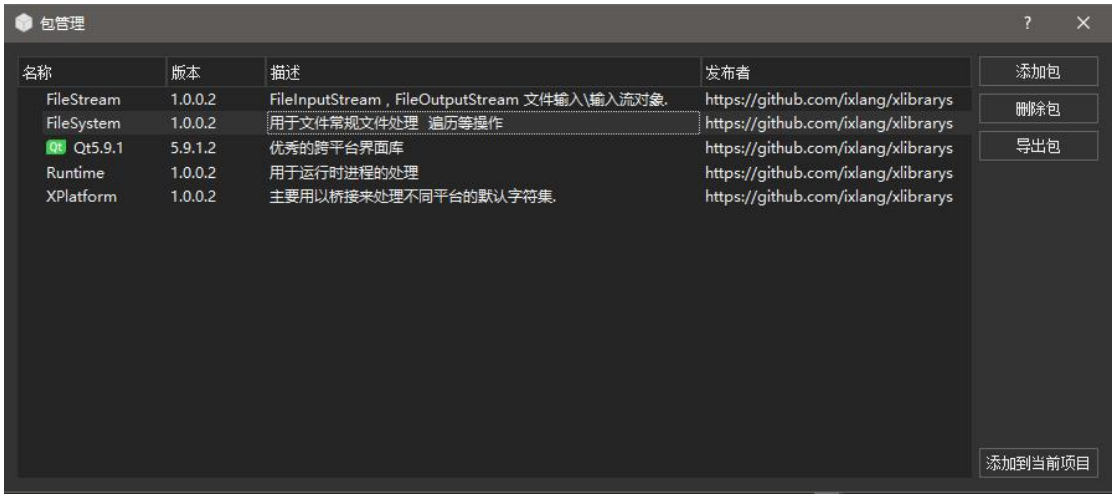
注意：被调试端要以调试参数 -xdbn:端口 运行， 例如 远程调试名为 a.exe 的程序， 需要在被调试机上运行 a.exe -xdbn:23326, 该程序会在被启动前挂起，等待调试器连接， 此时使用 XStudio 的远程调试功能，输入被调试机的 IP 和调试参数中输入的端口号，即可调试，调试器附加以后，程序将在入口点中断，等待调试器的下一步指示。

远程调试时，若源文件不在调试机上，XStudio 会给予提示，此时需要手动选择对应的源文件。

注意: 目前 XStudio 的远程调试支持任何具有 TCP 连接能力的操作系统和（含嵌入式）设备.

包管理器

点击 XStudio 的菜单 [工具] -> [包管理]，打开包管理器:



1.添加包，将*.xp 文件添加到包管理器中。
2.删除包，删除选择的包。
3.导出包，将选择的包导出为*.xp 文件。
4.添加到当前项目，将选择的包添加到当前项目中。

X 包是 lix 静态库的扩展发布形态，X 包中不仅可以存在 lix 静态库，还可以将不同平台的 native 动态库同时打包，同时 X 包中还包含图标、版本，发布者和 README 文档说明。

XStudio 中有统一的 X 包管理器，项目在需要使用某个功能的包时，只需打开包管理器选择哪些包添加到当前项目即可.

有任何疑问请进入 

QQ 群：591392649，进行交流探讨.