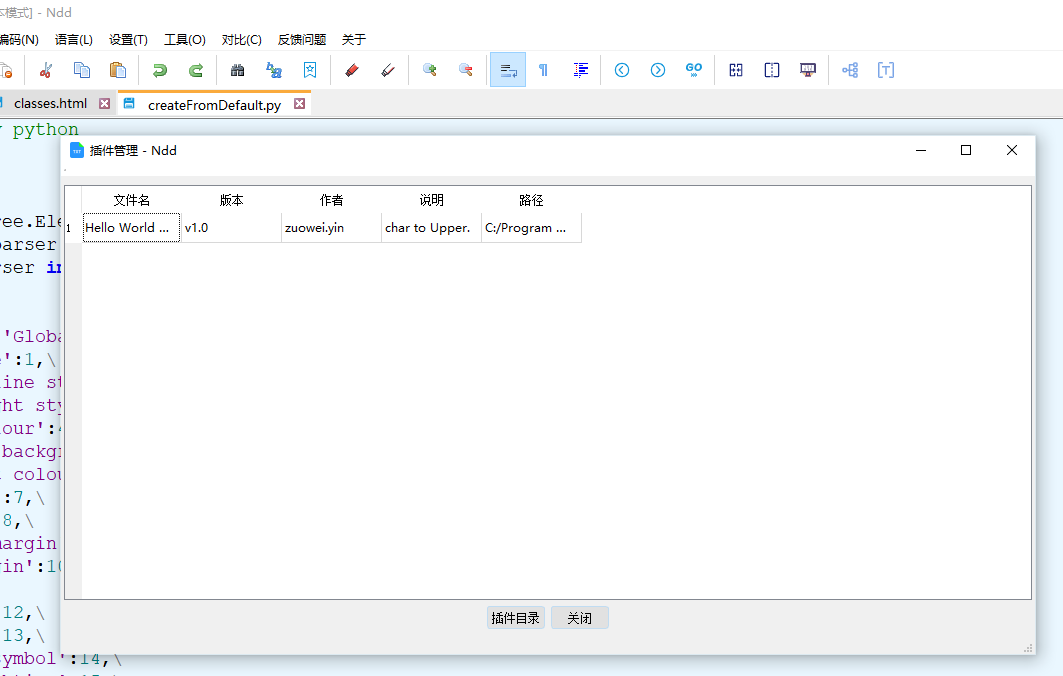
# Ndd 插件编程开发说明

从Notepad-- 1.21.0 开始，ndd提供了一个动态编译的版本，可以支持插件开发。如下图所示：



下面我们介绍，如何进行Ndd的插件开发。

## 什么样的功能，适合使用插件来完成。

请看下面的逻辑图，目前插件主要用来做文字处理工作。插件的流程有如下四步：

1. 第一步：从主程序获取当前编辑框对象。
2. 第二步：从当前编辑框对象读取文本。
3. 第三部：插件对文本进行处理，这里也是插件的主要工作。
4. 第四部：插件把处理后的结果问题，回填入当前编辑框对象。



*基于以上流程，什么样的功能时候插件来做？***主要是针对文字处理的功能。**举例来说，要做一个把繁体转换为简体，或者简体转换为繁体的工作，则比较适合插件来完成。再举例，要做一个把全码转换为半码，或者半码转换为全码的文字处理功能，也比较适合使用插件来完成。总体而言，如果与NDD主程序交互的工作相对简单，只有一个来回的获取文字、回填文字的流程工作，都比较适合使用插件来完成。

对于文字的处理，无论其做多么复杂的工作，都是插件本身来完成。

再举一个例子，经常有人说，要针对xml和json做格式化处理，加密解密处理，那么这个功能虽然比较复杂，但是其本身也非常适合使用插件功能来完成。可以武断的说，几乎百分之90以上的小功能，都可以使用插件工作来完成。

哪些工作不适合使用插件来完成？反之，如果您的工作重点不在于文本处理，而在于用户交互，则可能不适合插件来完成，或者可以实现，但是会比较麻烦。

## 二、如何编程开发插件

### 2.1 加载工程库

从gitee下载代码后，分别使用vs或qtcreator打开如下两个pro工程：

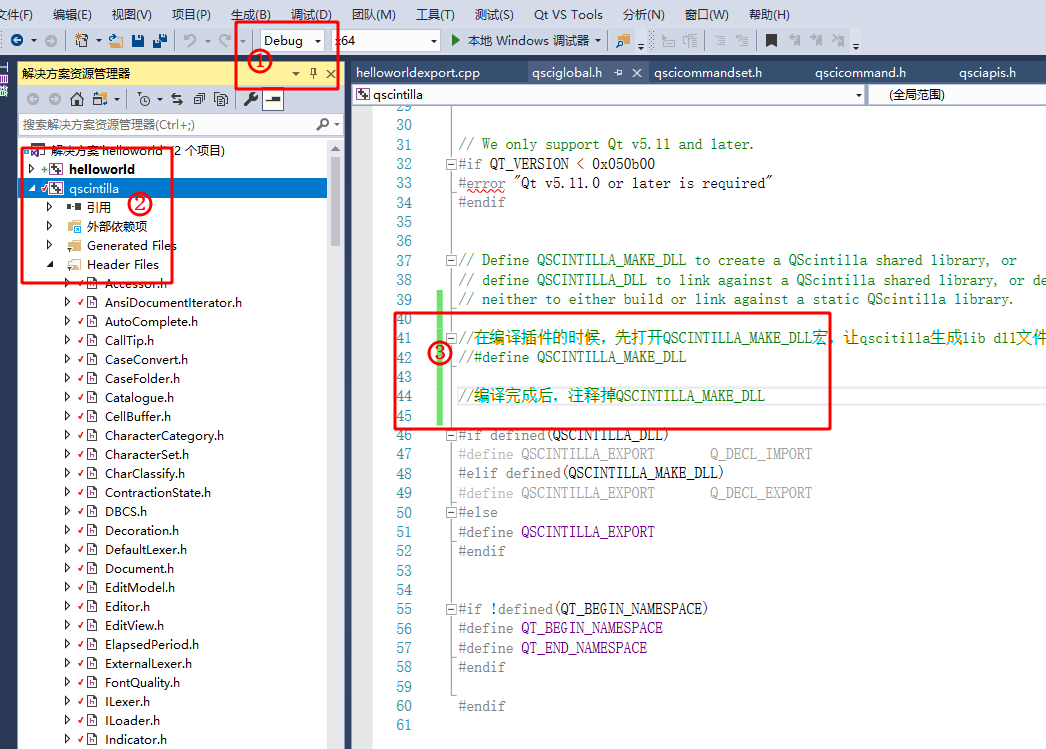
1)notepad--\src\qscint\src\qscintilla.pro

2) notepad--\src\plugin\helloworld\helloworld.pro

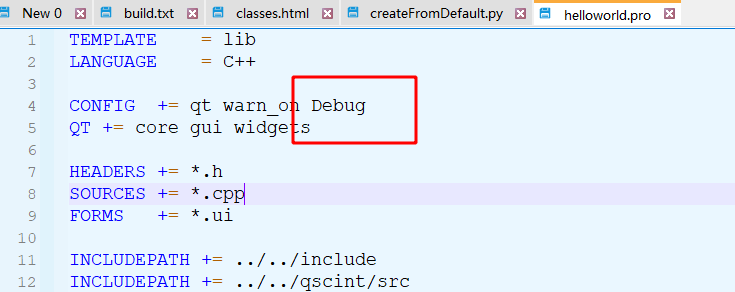
插件只依赖qscintilla库，不依赖ndd的库。

这一步，我们假定你已经对qt的配置环境比较熟悉，对qt中的基本配置环境，比如qtcreator中qt环境的配置，或者vs qtvstool中qt的配置，已经比较熟悉。如果您不熟悉这一步，建议还是自己先写一些qt基本的hello world方面的例子，把这一步补上再说。

下面给出vs加载两个工程后的样子。



按照上图，我们在Debug模式下做开发。如果是release模块，则pro中对应的如下配置，要修改为Release，注意qscintilla.pro 和helloworld.pro两个pro都要修改。我们下面的举例，都是假定在Debug模式下面做开发。



### 推荐的开发环境：

推荐vs2019 + qt512.2。

目前我本是就是使用的vs2019和qt515.2。QT官方到今天（20230107），能够下载的最新的QT5就是QT515.2。而据我所知，有些QT的bug在qt515.7中做了解决，但是官方没有提供qt515.7的在线安装包。如果你也使用QT515.7制作插件，则您的插件，大概率的不用做任何修改，可以直接加载到NDD官方发布的NDD主程序中。

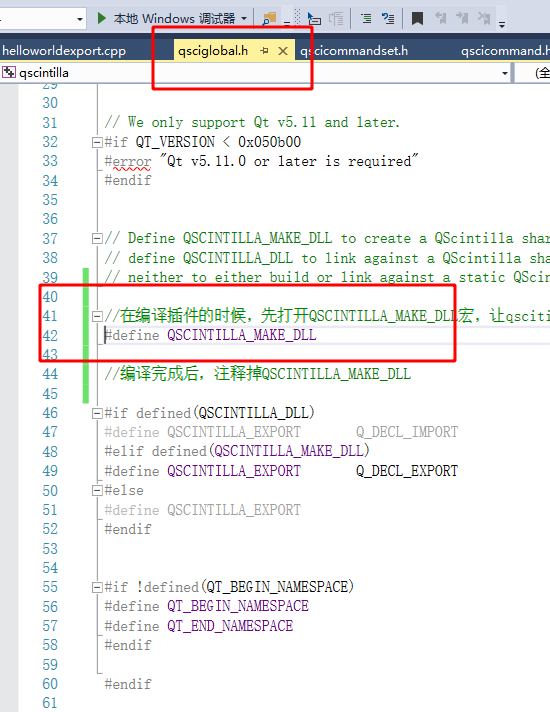
如果你使用的其他QT版本，问题也不大。您只需要手动自己编译NDD程序，即可加载您的插件。只要您的插件开源，而且用户反响较大，我们会在发布NDD的时候，主动打包发布您的插件。所以其实不用太担心QT 版本兼容的问题。

### 编译过程

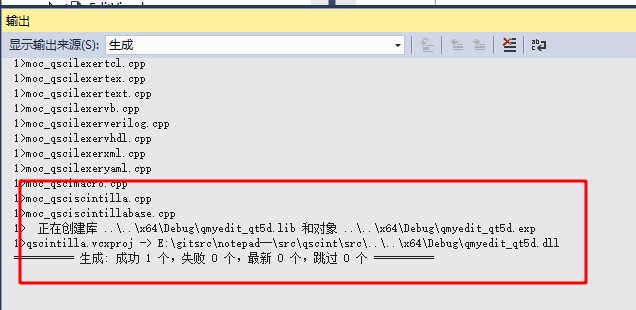
#### 第一步：先编译notepad--\src\qscint\src\qscintilla.pro

修改下面文件，打开宏#define QSCINTILLA\_MAKE\_DLL，然后编译qscintilla.pro

如果你不打开这个宏，则无法得到编译的LIB文件，只有dll文件。

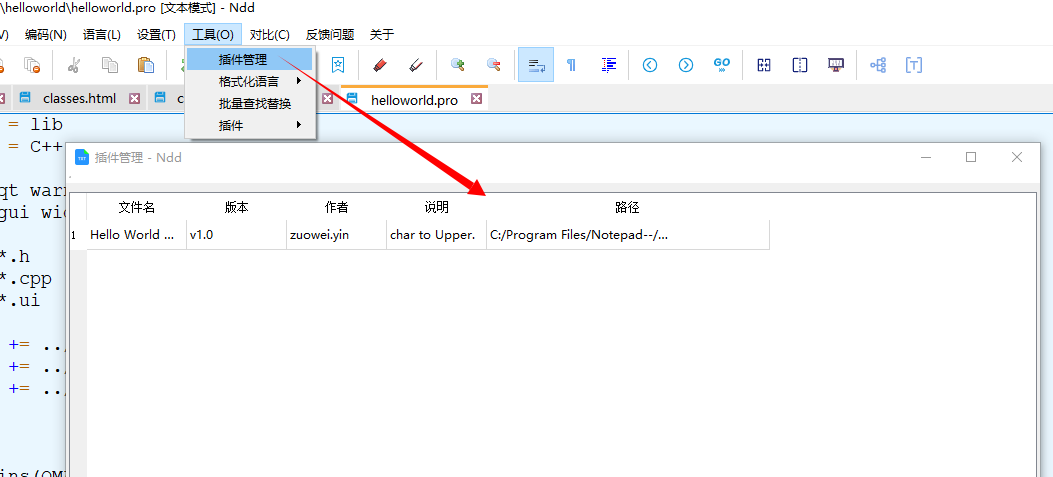


编译完成后，将在如下目录生成出对应的lib 和 dll 库。请到该目录下检查，是否有对应的生成库，如果没有，则说明哪里配置没有对。



### 第二步：编译notepad--\src\plugin\helloworld\helloworld.pro

这个helloworld.pro的功能很简单，就是把当前编辑框的内容，做一个大小写的转换。其加载在NDD中的界面如下：

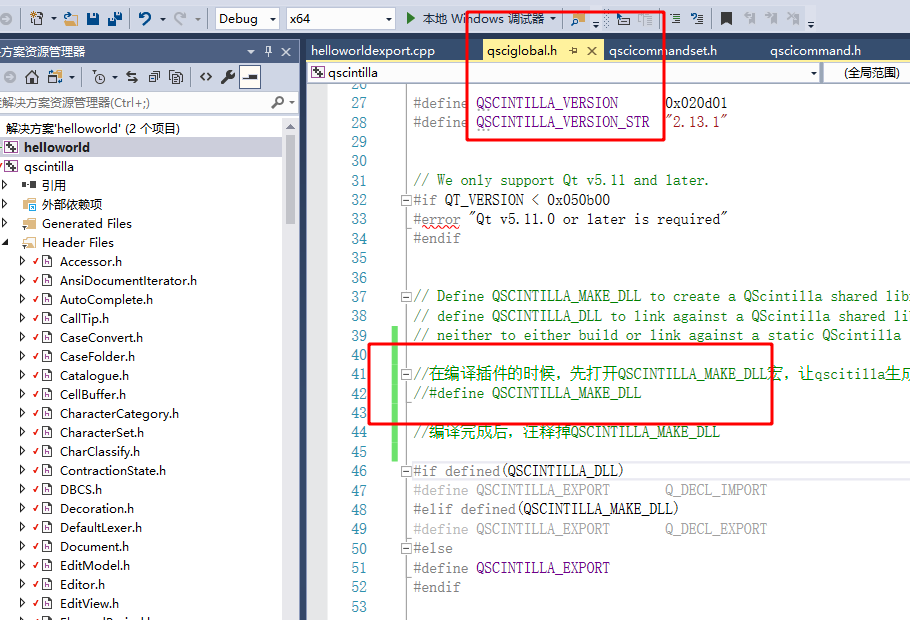


**注意：在编译helloworld.pro，务必要关闭如下文件中的宏//#define QSCINTILLA\_MAKE\_DLL。**

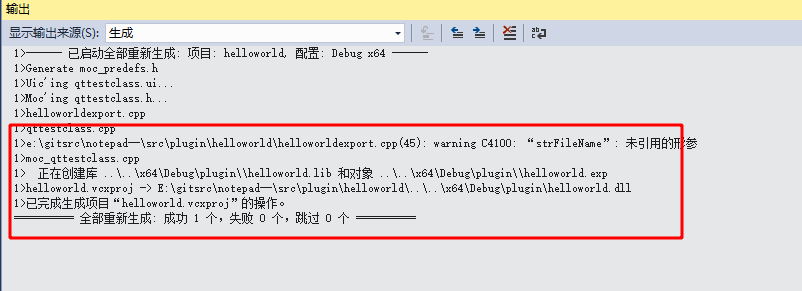
这个宏在编译QSCINTILLA的时候起作用，告诉编译器，要生成导出的LIB文件。

但是其它第三库依赖该头文件的时候，要注释该宏，因为其他库是把该库做一个导入DLL使用的。

***如果不注释，则可能报错。如下图所示：这一步千万不能忘记！***



Helloworld.dll编译成功后，会在如下目录生成对应的dll文件。此时例子插件就已经编译出来了。打开你编译的NDD主程序，则可以加载到该插件。如何编译ndd主程序，在另外一个文档“编译说明”中有讲解。我们建议都在Debug模式下做开发，最后发布的时候，统一切换到Release模式。



如果您熟悉了以上例子，则自己制作新功能的时候，只需要把helloworld的例子拷贝一份，然后改名，则可以在其基础上制作新的插件了。

## 三、对插件代码的分析和讲解

见例子helloword插件的插件文件helloworldexport.cpp中的主函数：

//插件的入口点函数

int NDD\_PROC\_MAIN(QWidget\* pNotepad, const QString &strFileName, std::function<QsciScintilla\*()>getCurEdit)

{

//下面的getCurEdit函数是一个仿函数，主程序会把该函数传递进来。

QsciScintilla\* pEdit = getCurEdit();

//新做一个QtTestClass功能类，在其中做一个简单的转大写的操作

QtTestClass\* p = new QtTestClass(pNotepad,pEdit);

//主窗口关闭时，子窗口也关闭。避免空指针操作。我们建议这样保持不动。

//因为例子QtTestClass是一个QWidget界面，只有给它Qt::Window属性，其才会保持独立存在，而不是//嵌套在主界面中的一个子界面。主界面关闭的时候，会自动关闭插件的界面。这样可以避免主界面//关闭了，而插件界面还存在，还可能去操作主界面函数的异常发生。

p->setWindowFlag(Qt::Window);

p->show();

return 0;

}

而qttestclass.cpp中的代码非常简单，就是一个大小写转换，然后回填编辑器窗口的过程，我们就不再详细讲解，贴图如下：

