Java有好用的JavaDoc文档生成工具，那么C++有没有呢？有，这就是大名鼎鼎的[Doxygen](http://www.stack.nl/~dimitri/doxygen/manual.html)，开源，功能强大，支持非常多的编程语言。

**1.安装和配置**

首先下载[Doxygen1.5.6](ftp://ftp.stack.nl/pub/users/dimitri/doxygen-1.5.6-setup.exe)，然后下载[graphviz-2.18](http://www.graphviz.org/pub/graphviz/ARCHIVE/graphviz-2.18.exe)，安装。

运行Doxywizard，开始配置。

单击Wizard按钮，会弹出对话框，输入项目名，这个名字会作为文档的大标题，输入版本，也会出现在文档中，然后输入源代码的根目录，勾选”Scan recursively”，输入文档输出路径。如图1所示：

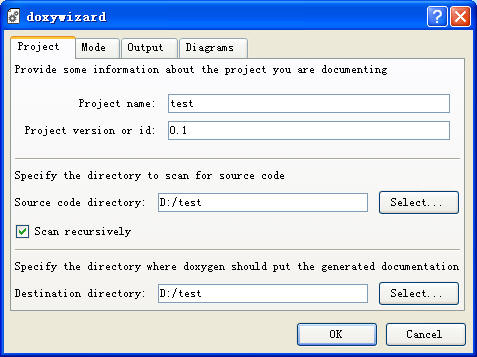


图1

单击Mode标签，不做任何改动，保持默认。

单击Output标签，去掉”LaTex”，选择“prepare for compressed HTML(.chm)”，因为输出chm比较方便，只有一个文件就包含所有文档，不向html会有一堆的文件。如图2所示：

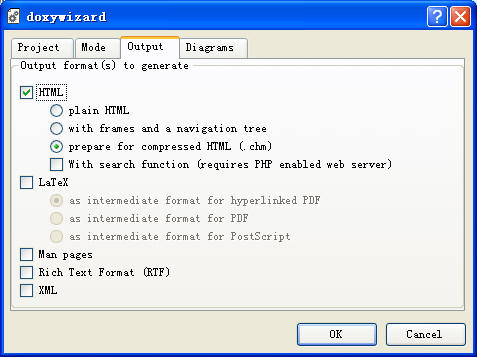


图2

单击Diagrams标签，如果已经安装了GraphViz，则保持默认，如果没安装，则选择“Use built-in class diagram generator”就足够，如图3所示：

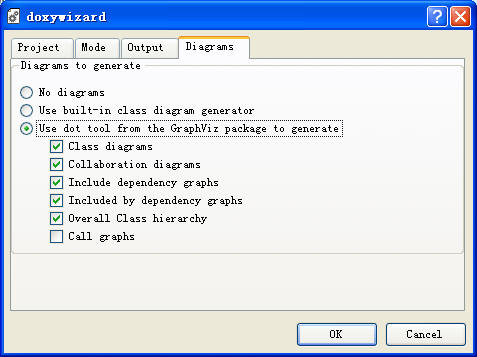


图3

点击OK，返回。

单击Expert按钮，会弹出一个有更多标签页的对话框，在"Project"标签页下，将OUTPUT\_LANGUAGE设置为Chinese，因为我需要生成中文文档，如图4所示：

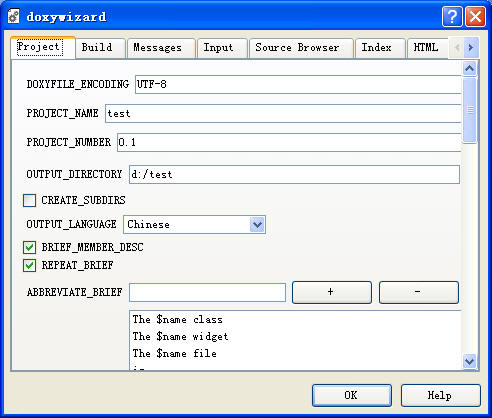


图4

单击"Input"标签，将INPUT\_ENCODING保持默认的utf-8，因为我用的是Visual Studio源代码文件的编码默认就是utf-8。如图5所示：

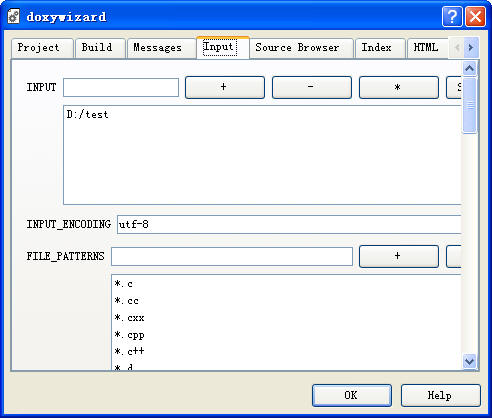


图5

如果你有洁癖，你可以耐心的将FILE\_PATTERNS下的后缀一个一个删掉（用记事本打开配置文件，搜索”FILE\_PATTERNS”，一下可以删除一片，免去你点鼠标点到食指抽筋之苦），只留下\*.h、\*.hpp、\*.c、和\*.cpp等，意思是只扫描C++头文件和源文件，如图6所示：

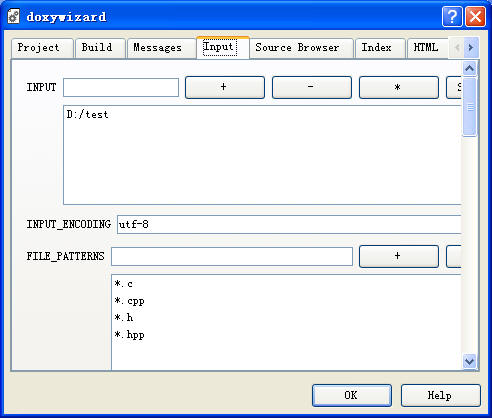


图6

下拉滚动条，会有EXCLUDE和EXCLUDE\_PATTERNS表示不要进行解析的目录和文件，即工程目录下有的目录不需要进行文档化（比如测试代码），就用这两个排除掉。

单击“Source Browser”标签，勾选“SOURCE\_BROUSER”，这样文档中就会附加一份源码，方便随时查阅，如图7所示：

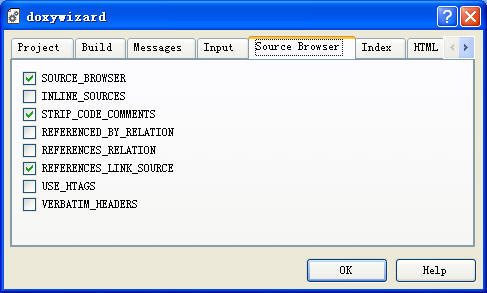


图7

单击"HTML"标签，勾选“HTML\_DYNAMIC\_SECTION”，表示要输出chm文件，同时在CHM\_FILE输入文件名作为要最终生成的chm文件名，旁边的那个"File.."按钮其实没用。同时点击“HHC\_LOCATION”右边的按钮找到chm编译器hhc.exe。如图8所示：

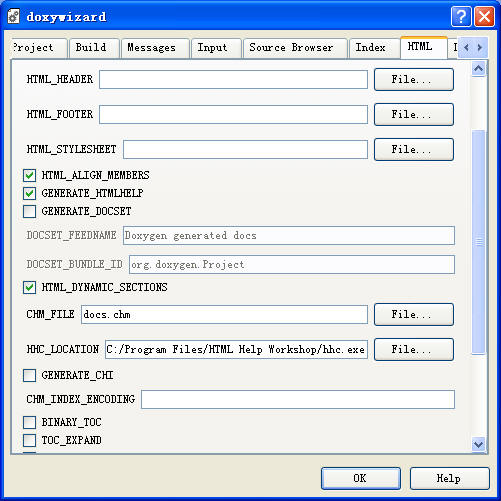


图8

单击OK返回，接下来按“Save...”按钮保存配置文件，文件名随意，如图9所示：

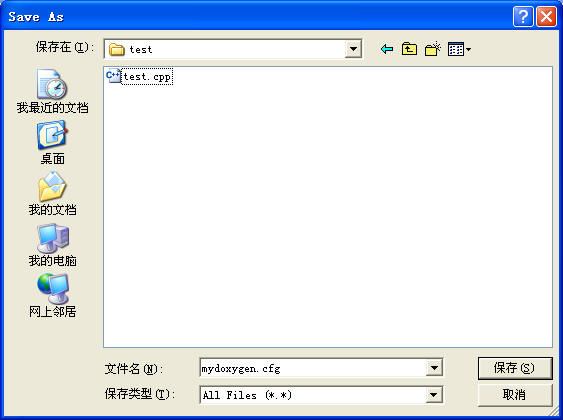


图9

这个配置好的文件以后可以重复利用，每次点”Load…”装载进来，然后点击”Wizard…”，根据不同的工程，修改工程名字，版本，源代码根目录，文档输出目录就可以了，不用再重复上述配置。

接下来在输入Working Directory中一般也输入源代码的根目录，主要是因为配置的一些选项中有的可以用相对路径，这个就可以作为相对路径的参照点。

最后单击“Start”按钮开始生成文件，到D:/test下查看，发现多了个html文件夹，进去一看，有很多HTML和一个chm文件，chm文件就是我们所要的文档，不过还不行，chm的左边导航目录是乱码，还需要一些步骤。

首先用一个文本编辑工具（我用VS2008打开，可以显示中文，以gb2312另存的，可是VS2005貌似打开是乱码）打开index.hhc文件，因为这个文件就是目录，然后另存为gb2312编码的文件，覆盖原来的index.hcc。

然后用hhc编译器重新编译，把chm工程文件传给hhc.exe即可，如图10所示：

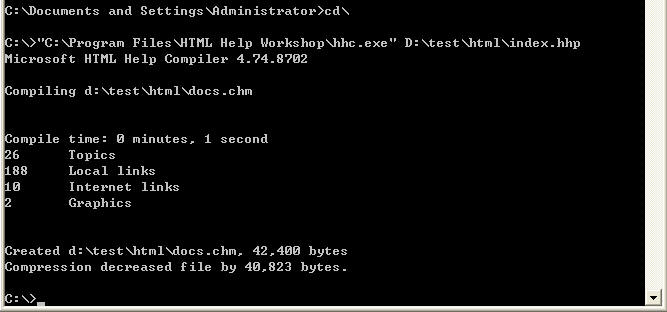


图10

打开docs.chm，目录是中文的了！

**2.常用注释语法**

注释写在对应的函数或变量前面。

简要注释和详细注释：

/\*\*

\* @brief Brief Description.

\*

\* Detailed Description

\*/

简要注释遇到一个空行或新的命令会结束，后面的就表示是详细注释。

JavaDoc风格下，自动会把第一个句号前的文本作为简要注释，后面的为详细注释。你也可以用空行把简要注释和详细注释分开。注意要设置JAVADOC\_AUTOBRIEF设为YES。

为了注释一个类中的member，首先要对该类作注释。同样的问题上升到namespace。要注释一个全局的function，typedef，enum或preprocessor定义，你需要首先定义（只能用@file，因为文件不再任何东西里面，就只能用特殊命令实现了，而不像类、函数等，既可以在上方放注释，也可以用@class、@fn进行注释）包含它的文件。

（1）文件头注释

/\*\* @file [file-name]

\*@brief brief description

\* @author <list of authors>

\* [@author <authors description>]

\* @date <date>

\* @version <version number>

\* @note

\* detailed description

\*/

一般@file后我们空着，Doxygen会默认为是@file所在文件的文件名。

[]表示可选，{}表示重复0到N次，<>表示必须参数。@author 表示作者，@data表示日期，@version表示版本号。

（2）类注释

/\*\*

\* @class <class-name> [header-file] [<header-name]

\* @brief brief description

\* @author <list of authors>

\* @note

\* detailed description

\*/

header-file是类声明所在的头文件名字，header-name是要显示的链接文字，一般为头文件的真实路径。

（3）函数注释

/\*\*

\* @brief brief description

\* @author <list of authors>

\* {@param[in|out] <parameter-name> <parameter description>}

\* @return <description of the return value>

\* {@return <description of the return value>}

\* @note

\* detailed description

\* @remarks <remark text>

\* {@remarks <remark text>}

\* [@deprecated <description>]

\* [@since when(time or version)]

\* [@see references{,references}]

\*/

@可用\代替，但我倾向于用@。

@param[in|out] 参数名及其解释

@return 对函数返回值做解释

@note 表示注解，暴露给源码阅读者的文档

@remark 表示评论，暴露给客户程序员的文档

@since 表示从那个版本起开始有了这个函数

@deprecated 引起不推荐使用的警告

@see 表示交叉参考

函数的详细注释用@note代替详细注释，因为详细注释要空行隔开，容易忘记。

（4）成员注释

/\*\*< 或//<用来注释成员，放在成员后面，格式如下：

int var; /\*\*< Detailed description after the member \*/

int var; ///< Brief description after the member

此语法对函数成员也适用。

（5）枚举类型注释

/\*\* @brief Another enum, with inline docs \*/

enum AnotherEnum

{

V1, /\*\*< value 1 \*/

V2 /\*\*< value 2 \*/

};

**一般约定：**

（1）每个.h和.cpp文件的头部，必须要有简要注释和详细注释，习惯用法如下：

/\*\* @file

\*@brief brief description

\* @author <list of authors>

\* @date <date>

\* @version <version number>

\* @note

\* detailed description

\*/

（2）每个类的声明上方，必须要有简要注释和详细注释，习惯用法如下：

/\*\*

\* @class

\* @brief brief description

\* @author <list of authors>

\* @note

\* detailed description

\*/

（3）全局变量和全局宏必须要有注释。

如果注释较短，则可以在上方用

/\*\* @brief some brief description \*/或右方用

///< some brief description。

进行简要注释。

（4）任何函数都必须要有简要注释和详细注释，习惯用法如下：

/\*\*

\* @brief brief description

\* @author <list of authors>

\* @param[in|out] <parameter-name> <parameter description>

\* @return <description of the return value>

\* @note

\* detailed description

\* @remarks <remark text>

\*/

对于类的函数成员，在头文件的定义处进行简要注释，放在上方：

class Test

{

public:

/\*\* @brief brief description \*/

int m\_test(int a);

}

而在实现出给出详细注释：

/\*\*

\* @author <list of authors>

\* @param [in|out] <parameter-name> <parameter description>

\* @return <description of the return value>

\* @note

\* detailed description

\* @remarks <remark text>

\*/

int Test::m\_test(int a)

{

Return 0;

}

纯虚函数由于没有实现则简要注释和详细注释不需分开。

对于类的数据成员，只在头文件的定义处进行简要注释，不要详细注释。可以在上方用/\*\* @brief some brief description \*/或右方用///< some brief description。

（5）每个枚举定义必须添加注释,格式如下:

/\*\* @brief Another enum, with inline docs \*/

enum AnotherEnum

{

V1, ///< value 1

V2 ///< value 2

};

下面是一个简单的例子，完全符合约定：

/\*\* @file

\* @brief a brief description for the file.

\* @author soulmachine

\* @date 2008/07/02

\* @version 0.1

\*

\* detailed description for test.cpp

\*/

/\*\* @brief global function, no details

\* @note some details about global function

\*/

void global\_test();

/\*\* @class Test test.h "inc/test.h"

\* @brief A test class.

\*

\* A more elaborate class description.

\*/

class Test

{

public:

/\*\* @brief A enum, with inline docs \*/

enum TEnum {

TVal1, /\*\*< enum value TVal1. \*/

TVal2, /\*\*< enum value TVal2. \*/

TVal3 /\*\*< enum value TVal3. \*/

}

//这里Doxygen对enumPtr的处理有点问题

\*enumPtr, ///< enum pointer.

enumVar; ///< enum variable.

/\*\* @brief A constructor. \*/

Test();

/\*\* @brief A destructor. \*/

~Test();

/\*\* @brief a normal member taking two arguments and returning an integer value. \*/

int testMe(int a,const char \*s);

/\*\* @brief A pure virtual member.

\* @param[in] c1 the first argument.

\* @param[in] c2 the second argument.

\* @see testMe()

\*/

virtual void testMeToo(char c1,char c2) = 0;

int publicVar;//< a public variable.

/\*\* @brief a function variable, note Details. \*/

int (\*handler)(int a,int b);

/\*\* @brief brief before delaration \*/

int m\_func(int a);

};

/\*\* A more elaborate description of the constructor. \*/

Test::Test()

{

}

/\*\* A more elaborate description of the destructor. \*/

Test::~Test()

{

}

/\*\*

\* @param[in] a an integer argument.

\* @param[in] s a constant character pointer.

\* @return The test results

\* @note Details.

\* @par

\* Another detail.

\* @see Test()

\* @see ~Test()

\* @see testMeToo()

\* @see publicVar()

\*/

int Test::testMe(int a,const char \*s)

{

return 0;

}

/\*\*

\* @param[in] a a interger

\* @return 0

\* @note detailed description

\* @remarks remarks,important

\* @since 1.0

\* @see testMeToo

\*/

int Test::m\_func(int a)

{

return 0;

}

参考资料：

[Doxygen简单经验谈。。。](http://www.cnblogs.com/duguguiyu/archive/2008/06/29/1231852.html)

[C++ 程序文档生成器介绍(doxygen)](http://ly4cn.cnblogs.com/archive/2005/11/23/282637.html)

[Doxygen总结](http://blog.csdn.net/smartduck/archive/2005/08/25/464720.aspx)

[Doxygen 使用笔记](http://www.speedradiosity.com/chinese/?p=73)

[DoxyGen生成的html制作成CHM后目录为乱码的问题](http://www.cppblog.com/tx7do/archive/2008/04/21/47731.html)

[Doxygen注释常用标记](http://blog.csdn.net/freestorm/archive/2005/01/17/256237.aspx)

[使用doxygen为C/C++程序生成中文文档（上）](http://blog.csdn.net/fmddlmyy/archive/2007/06/23/1663898.aspx)

[Doxygen文档系列](http://blog.csdn.net/SigmaSys/archive/2003/12.aspx)