

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](#)，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](#)，然后下载 [graphviz-2.18](#)，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：

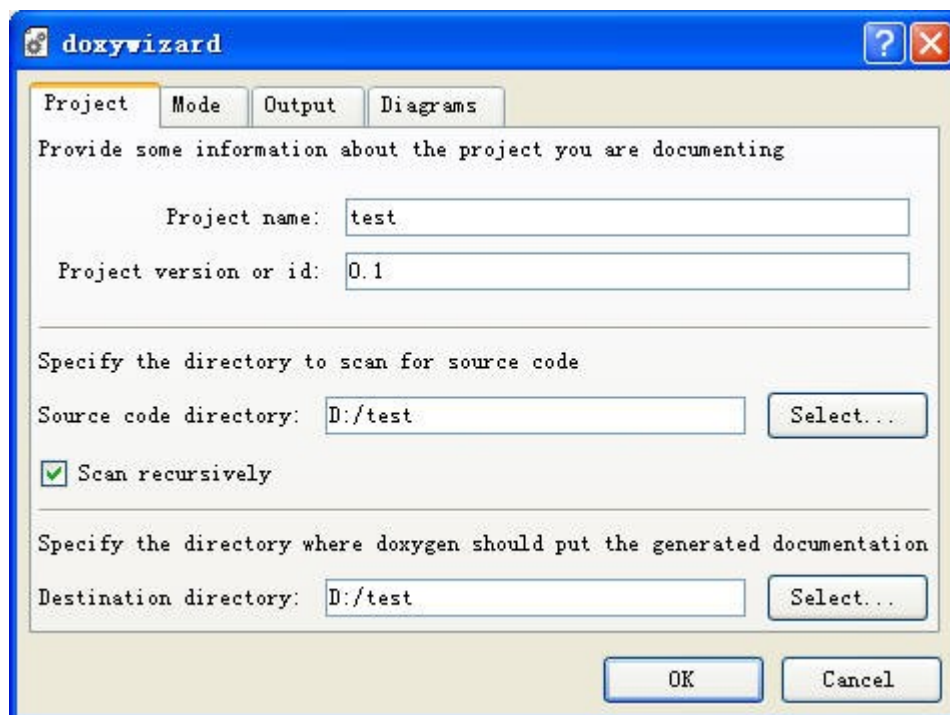


图 1

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：

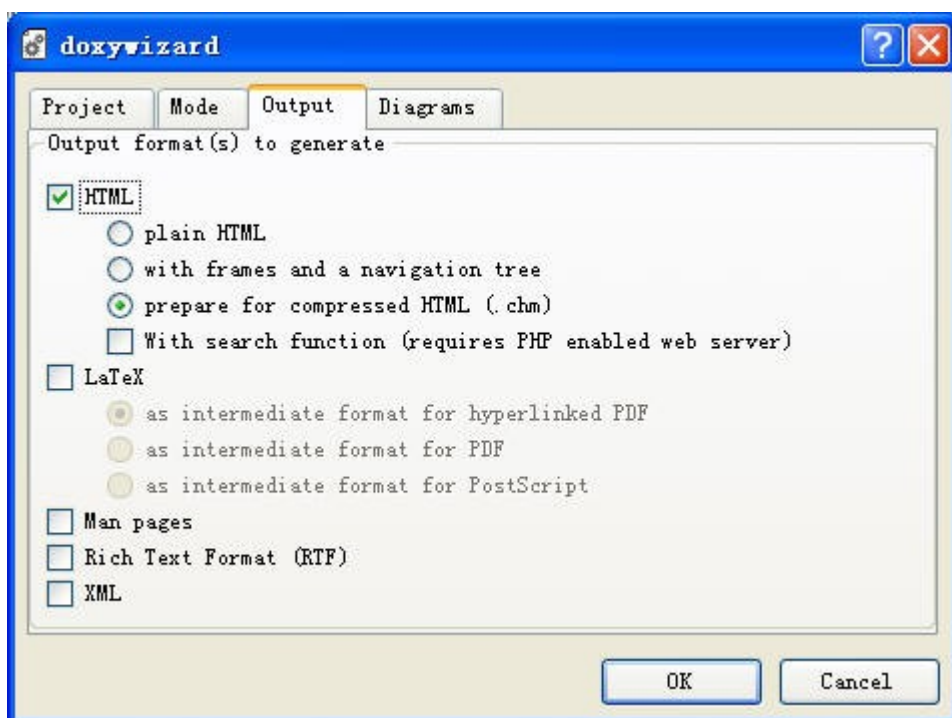


图 2

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：

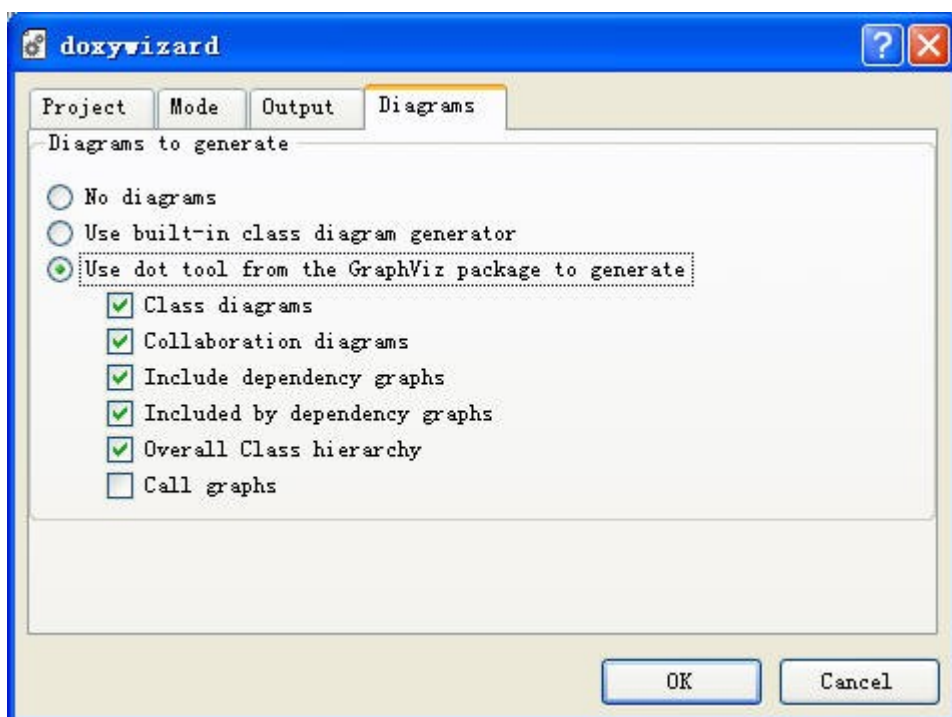


图 3

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：

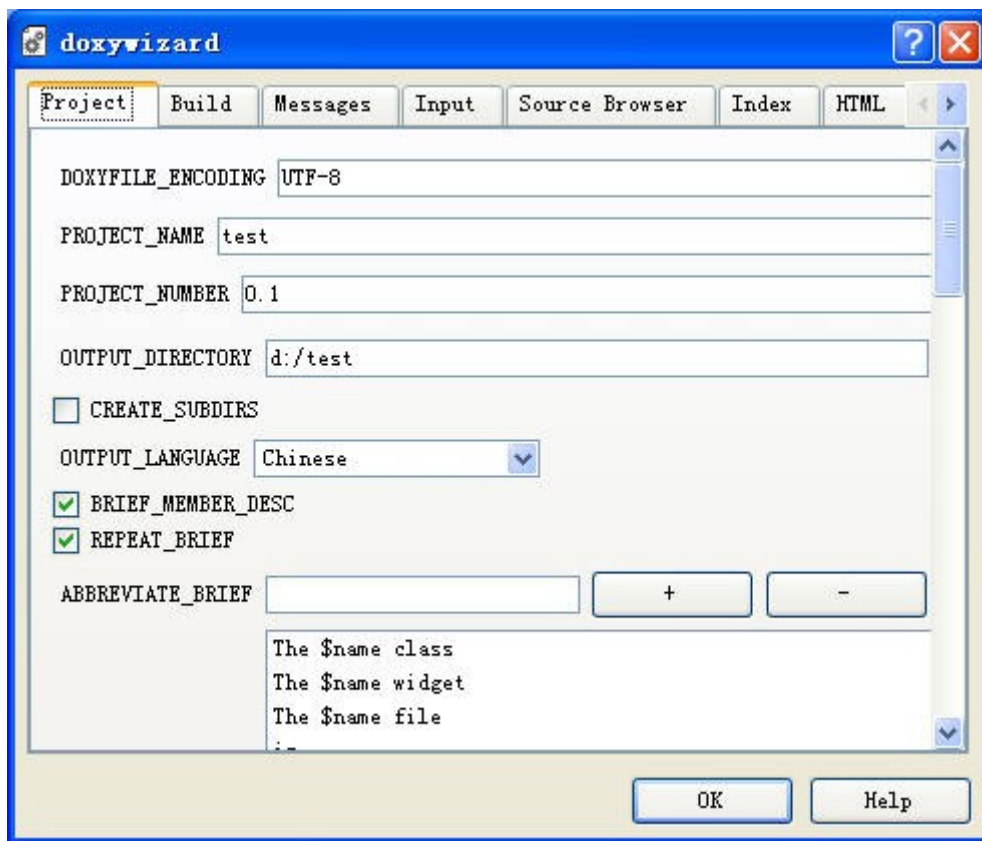


图 4

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：

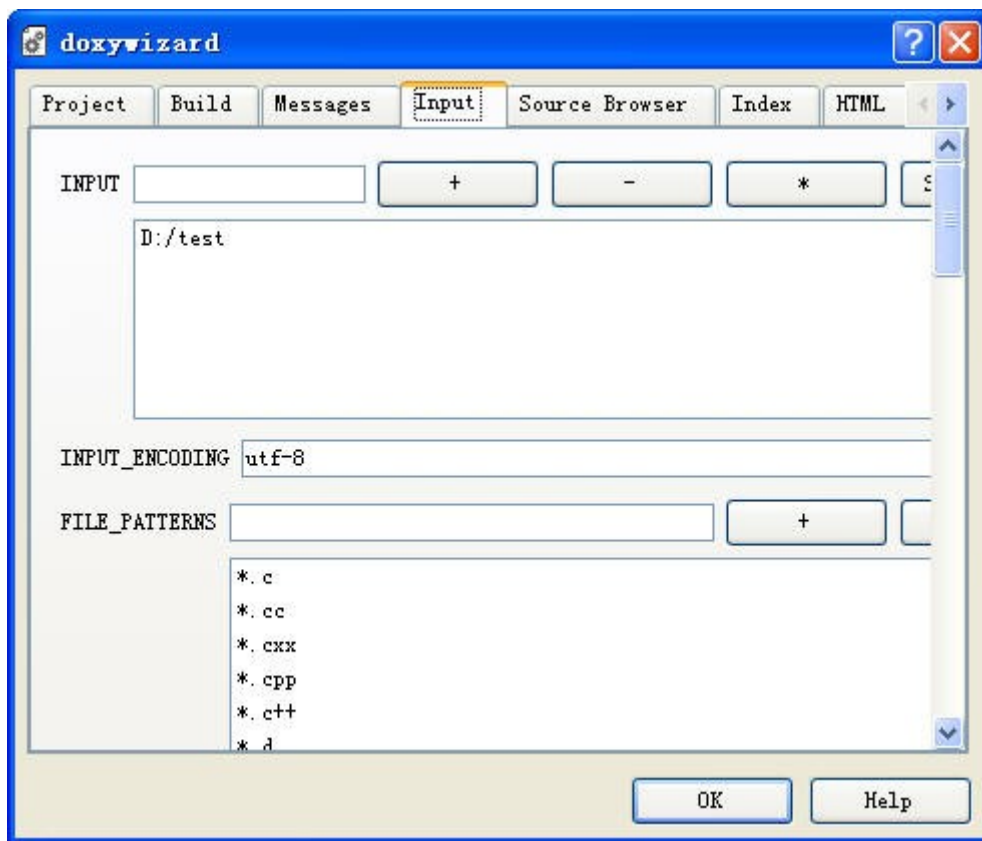


图 5

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：

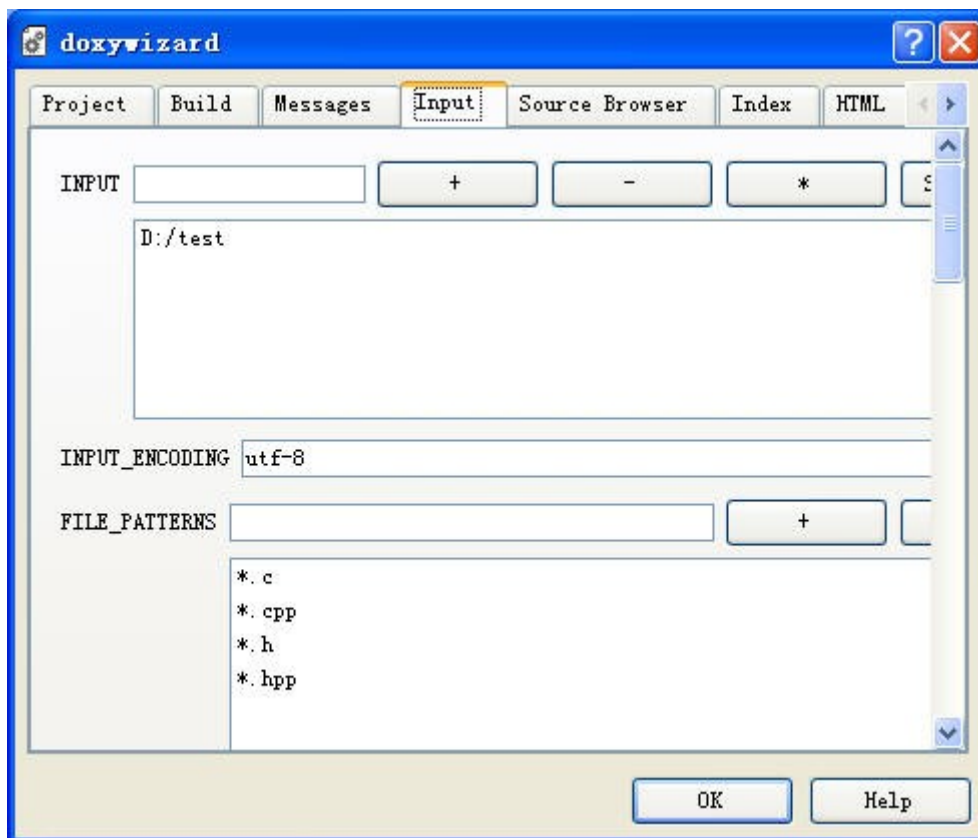


图 6

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_BROWSER”，这样文档中就会附加一份源码，方便随时查阅，如图 7 所示：

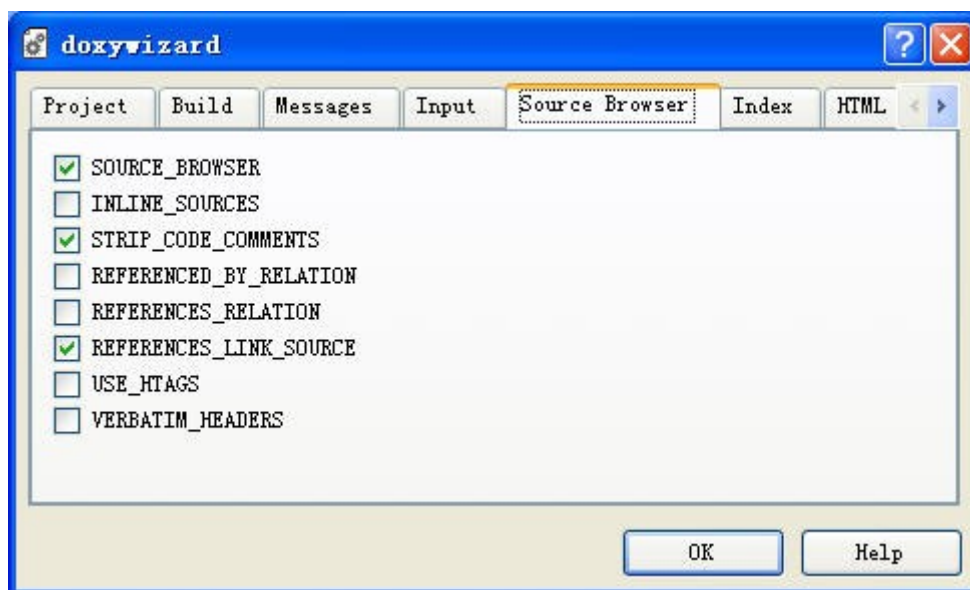


图 7

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：

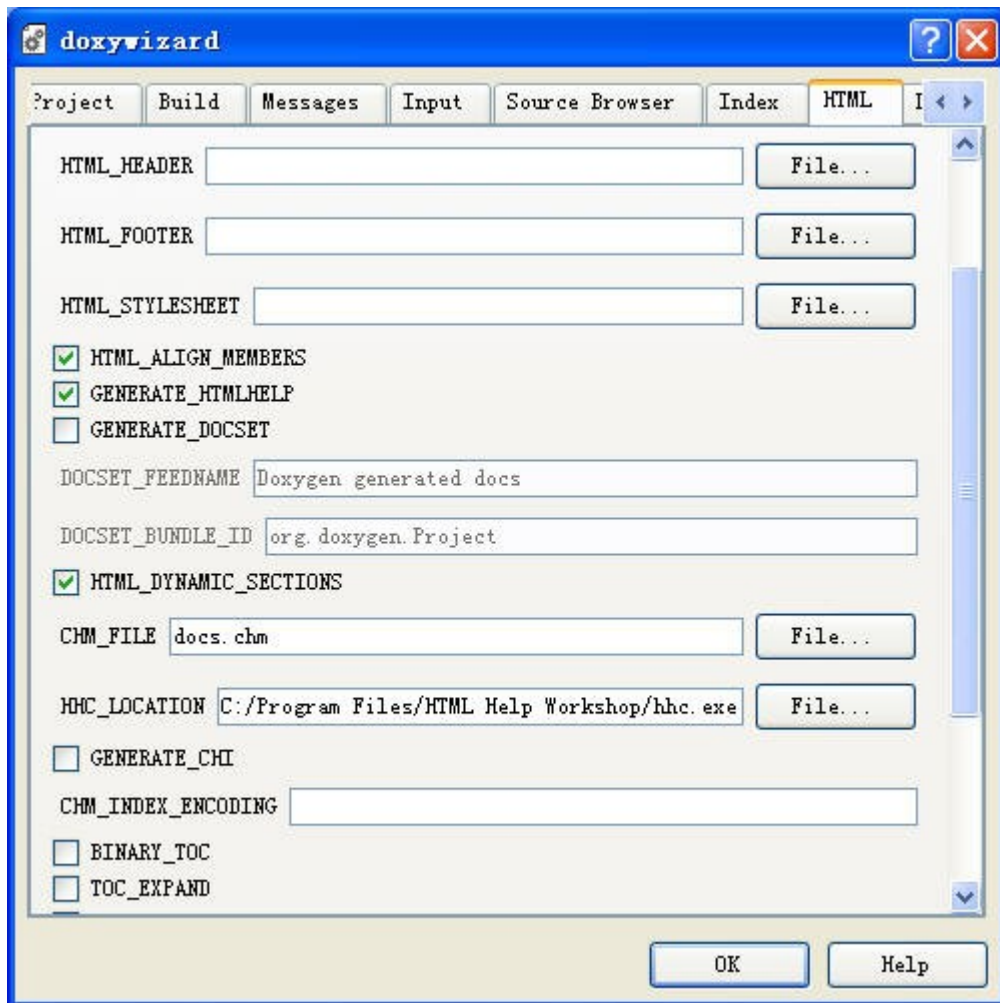


图 8

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：

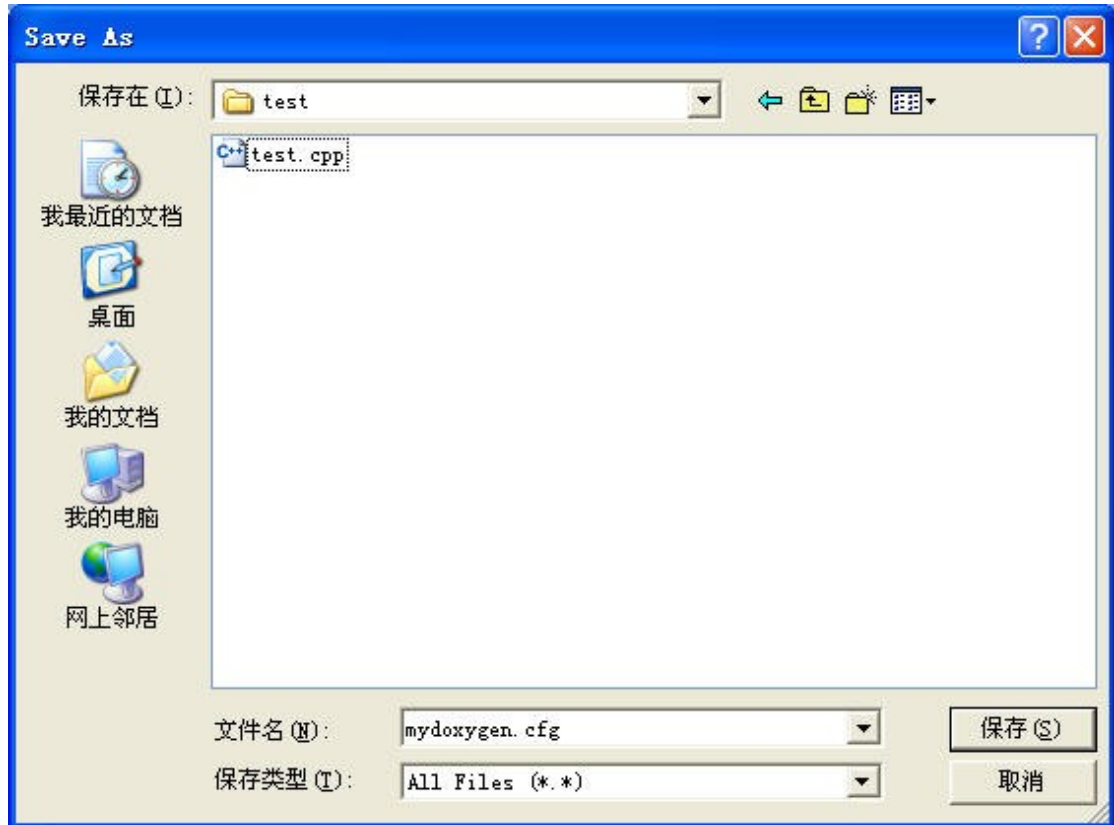


图 9

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：


```
C:\Documents and Settings\Administrator>cd\

C:\>"C:\Program Files\HTML Help Workshop\hhc.exe" D:\test\html\index.hhp
Microsoft HTML Help Compiler 4.74.8702

Compiling d:\test\html\docs.chm

Compile time: 0 minutes, 1 second
26      Topics
188     Local links
10      Internet links
2       Graphics

Created d:\test\html\docs.chm, 42,400 bytes
Compression decreased file by 40,823 bytes.

C:\>
```

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
 */
```

简要注释和详细注释用一个空行隔开。

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>
 *
 * detailed description for test.cpp
 */
```

一般 @file 后我们空着，Doxygen 会默认认为是 @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

```
/**
 * @brief brief description.
 *
 *detailed description
 */
```

(3) 函数注释

```
/**
 * @brief brief description.
 *
 * {@param <parameter-name> <parameter description>}
 * @exception <exception-object> <exception description>
 * {@exception <exception-object> <exception description>}
 * @return <description of the return value>
 * {@return <description of the return value>}
 * @remarks <remark text>
 * {@remarks <remark text>}
 * [deprecated <description>]
 * [since when(time or version)]
 * [see references{,references}]
 *
 * detailed description
 */
```

@可用\代替，但我倾向于用@。

@param 参数名及其解释

@return 对函数返回值做解释

@remark 做评论

@exception 用来说明异常类及抛出条件

@deprecated 引起不推荐使用的警告

@see 表示交叉参考

(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,必须添加注释,格式如下:

```
/** @file
 * A brief file description.
 *
 * A more elaborated file description.
 */
```

(2) 每个类的声明上方,必须添加注释,格式如下:

```
/** @class Test test.h "inc/test.h"
 * @brief This is a test class.
 *
 * Some details about the Test class
 */
class Test
{
    ...
};
```

@class 格式如下:

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

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) 全局变量、全局函数、全局宏和类的数据成员必须要有注释, 如果注释较短, 则必须在上方用单行进行简短的详细注释:

```
/** Some details about the function GlobalFunc*/
void GlobalFunc();
```

(4) 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:

```
/**
 * @param a a interger
 * @return 0
 * @par
 * detailed description
 *
 */
```

```
int Test::m_test(int a)
{
    Return 0;
```

```
}
```

(5) 8. 每个枚举定义必须添加注释, 格式如下:

```
/** Another enum, with inline docs */
```

```
enum AnotherEnum
```

```
{
```

```
    V1, ///< value 1
```

```
    V2  ///< value 2
```

```
};
```

下面是一个简单的例子, 不完全符合约定:

```
/** @file
```

```
 * @brief a brief description for the file.
```

```
 * @author soulmachine
```

```
 * @date 2008/07/02
```

```
 * @version 0.1
```

```
 *
```

```
 * detailed description for test.cpp
```

```
 */
```

```
/** global function, no details */
```

```
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.

 * A more elaborate description of the constructor.
 */
Test();

/**
 * @brief A destructor.

 * A more elaborate description of the destructor.
 */
~Test();

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

 * @param a an integer argument.
 * @param s a constant character pointer.
 * @return The test results
 * @see Test()
 * @see ~Test()
 * @see testMeToo()
 * @see publicVar()
 */
int testMe(int a,const char *s);

/**
 * @brief A pure virtual member.
 *
 * @param c1 the first argument.
 * @param c2 the second argument.
 * @see testMe()
 */
virtual void testMeToo(char c1,char c2) = 0;

/**
 * @brief a public variable.

 * Details.
 */

```

```

        int publicVar;

    /**
     * @brief a function variable.

     * Details.
     */
    int (*handler)(int a,int b);

    /** @brief brief before delaration */
    int m_func(int a);
};

/**
 * @param a a interger
 * @return 0
 * @remarks remarks,important
 * @since 1.0
 * @see testMeToo

 * detailed description
 */

int Test::m_func(int a)
{
    /** inner details */
    return 0;
}

```

参考资料:

[Doxygen 简单经验谈。](#)

[C++ 程序文档生成器介绍\(doxygen\)](#)

[Doxygen 总结](#)

[Doxygen 使用笔记](#)

[DoxyGen 生成的 html 制作成 CHM 后目录为乱码的问题](#)

[Doxygen 注释常用标记](#)

[使用 doxygen 为 C/C++ 程序生成中文文档 \(上\)](#)

[Doxygen 文档系列](#)