Doxygen 配置使用指南

2009-02-24

目录

序言

为代码写注释一直是大多数程序员有些困扰的事情。当前程序员都能接受为了程序的可维护性、可读性编码的同时写注释的说法，但对哪些地方应该写注释，注释如何写，写多少等这些问题，很多程序员仍然没有答案。更头痛的是写文档，以及维护文档的问题，开发人员通常可以忍受编写或者改动代码时，编写或者修改对应的注释，但之后需要修正相应的文档却比较困难。如果能从注释直接转化成文档，对开发人员无疑是一种福音。而 doxygen 就能把遵守某种格式的注释自动转化为对应的文档。

Doxygen 是基于 GPL 的开源项目，是一个非常优秀的文档系统，当前支持在大多数 Unix（包括 Linux），Windows 家族，Mac 系统上运行，完全支持 C++，C，Java，IDL（Corba 和 Microsoft 家族）语言，部分支持 PHP 和 C#语言，输出格式包括 HTML、Latex、RTF、ps、PDF、压缩的 HTML 以及 Unix Manpage。有很多开源项目（如：TinyXML）都使用了 doxygen 文档系统。而国内的开发人员却使用的不多，这里从开发人员使用的角度介绍这个工具，使开发人员用最少的代价尽快掌握这种技术，并结合这个工具探讨如何撰写注释的问题。

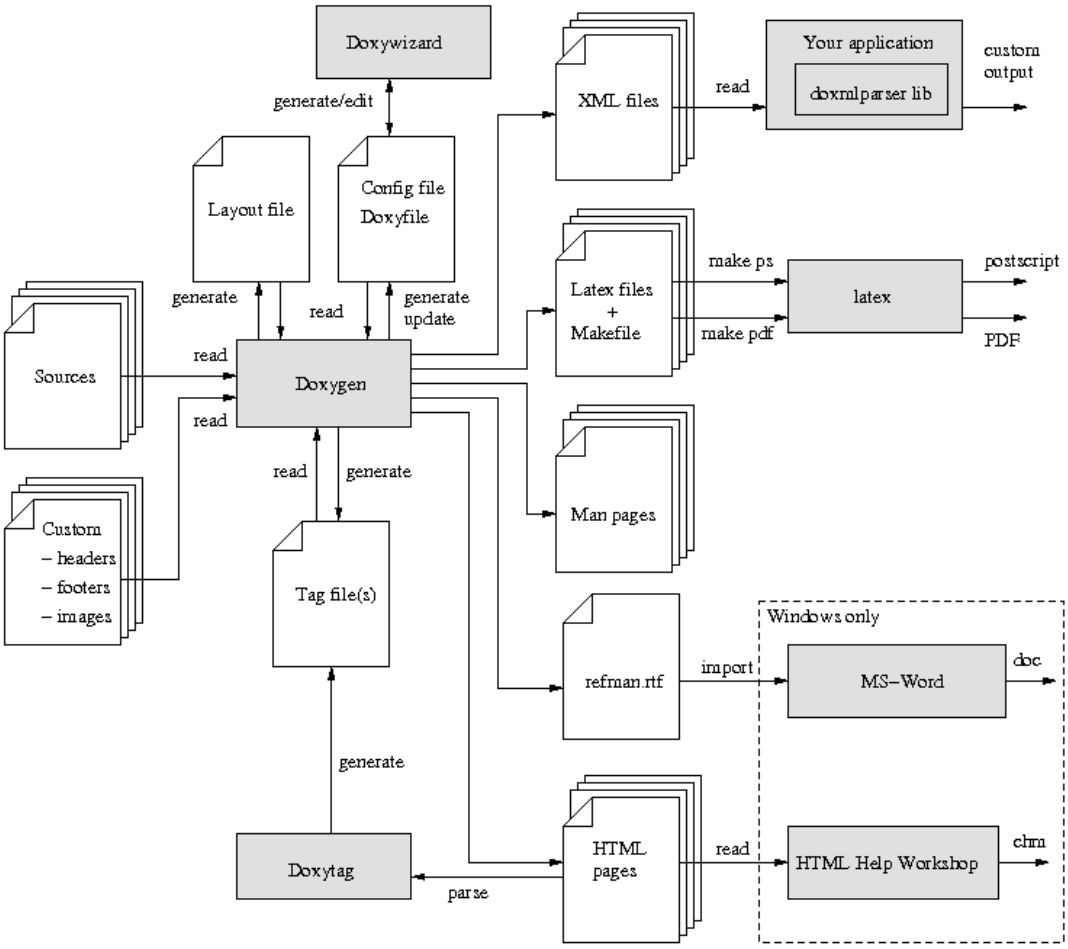
以下讨论基于 Doxygen-1.5.6 版本，分别以 Linux 下的 C 语言和 Windows 下的 C++为例进行介绍。

1．Doxygen 的工作过程

Doxygen 的工作过程可分为三个步骤：

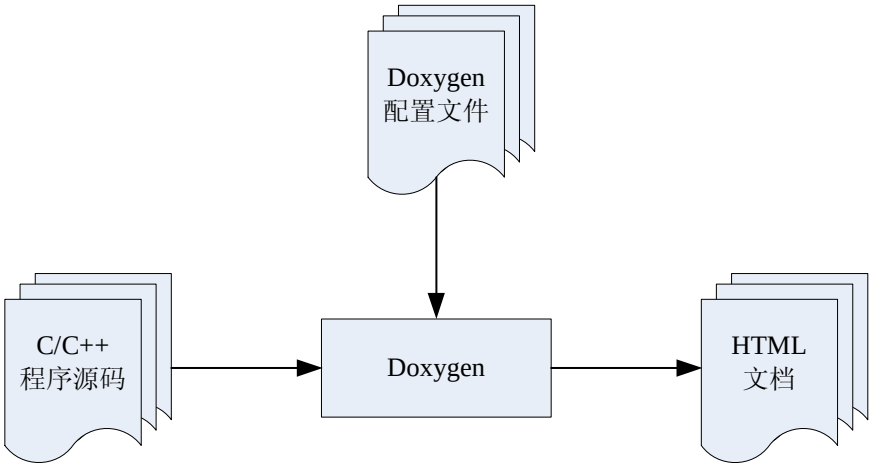
- 安装配置 Doxygen 工作环境，生成 Doxygen 配置文件；
- 在程序源码中添加符合 Doxygen 可解析的注释格式；
- 使用 Doxygen 解析源码，输出格式化文档。

Doxygen 的使用手册里有一个完整的工作流程图，如下所示：



Doxygen information flow

对于我们一般的用户来说，仅需要完成以下流程即可：



2. Doxygen 注释块

Doxygen 注释块其实就是在 C/C++程序注释块的基础添加一些额外标识，使

Doxygen 把它识别出来，并将它组织到生成的文档中去。Doxygen 支持多种注释风格，这里统一使用 C++风格代码注释。

名词解释：

- 行间注释：注释语句不与程序源码出现在同一行，主要用于文件注释、文件中结构体（struct）、枚举（enum）、联合等数据类型注释，以及程序函数接口的功能等注释；
- 行内注释：注释语句与程序源码出现在同一行内，主要用于代码的局部注释。

2.1 文件注释

文件注释主要用于解释当前文件的用途、作者、创建及修改日期等信息。

格式如下：

```
/**
 @file list.h
 @brief 链表头文件

 该文件主要用C语言实现带头节点的链表：包括创建、销毁、查询、插入、判断为空及链表元素倒置等功能。
 @author gomo
 @version 1.0.0.1
 @date 2008-08-19
 */
```

/**　　*/　Doxygen 的 C++注释风格行间注释起始、终止标识。

上面的“@file”，“@brief”，“@author”，“@data”，“@version”都是 Doxygen 注释风格的关键字。

注意：

- 各关键字后面添加内容时，必须有空格隔开；
- 简要描述和详细描述应该与之间隔一个空行。
- /**　　*/之间的“*”是可以省略的。

上面的注释生存 HTML 文档后效果如下：

详细描述

链表头文件

该文件主要用C语言实现带头节点的链表：包括创建、销毁、查询、插入、判断为空及链表元素倒置等功能。

作者: gomo

版本: 1.0.0.1

日期: 2008-08-19

在文件**list.h**中定义。

2.2 函数注释

示例一：

函数注释主要用于解释当前函数的用途、参数、返回值以及使用示例等信息。格式如下：

```
/**
 @brief 创建链表(简要描述)

 这是个创建链表的函数...(详细描述)
 @return 返回创建带头节点的空链表.

 示例:
 @verbatim
   link_t head = dh_create_link();
 @endverbatim
 */
```

下面的注释风格也能有同样的效果：

```
///创建链表(简要描述)
/**
 这是个创建链表的函数...(详细描述)
 @return 返回创建带头节点的空链表.

 示例:
 @verbatim
   link_t head = dh_create_link();
 @endverbatim
 */
```

注意：“@return”等关键字和示例等字符之间至少有一行间隔。

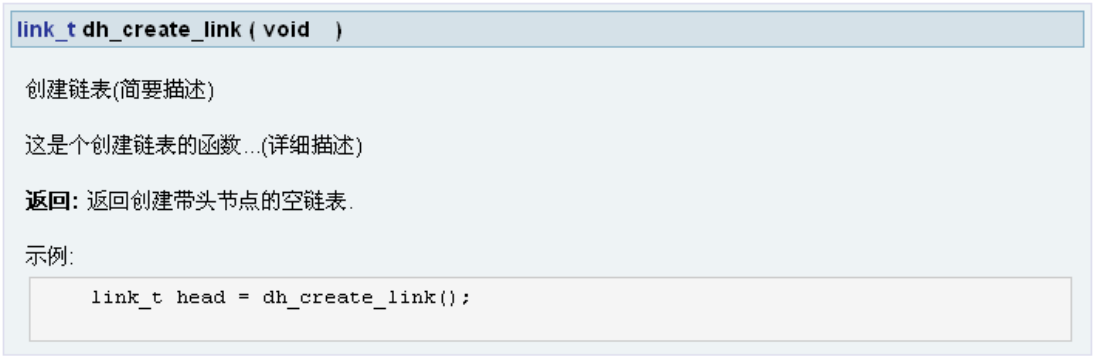
上面的注释生成 HTML 文档后效果如下：

a) 函数列表里面显示**简要描述**内容：

函数

link_t dh_create_link (void)
创建链表(简要描述)

b) 函数定义处显示函数**详细描述**内容：



示例二：

如多参数的函数注释：

```
/**
 在链表尾插入新节点

 @param list 要插入节点的链表
 @param tv 插入链表的节点内容
 @return 0，成功；-1，失败

 示例:
 @verbatim
 struct timeval tmp;
 struct timezone tz;
 gettimeofday(&tmp, &tz);
 if (dh_insert_node(head, &tmp) == -1)
 {
     printf("Insert node failed.\n");
     return -1;
 }
 @endverbatim
 */
dh_link_status dh_insert_node(link_t list, struct timeval *tv);
```

上面的注释生成 HTML 文档后效果如下：

```
dh_link_status dh_insert_node ( link_t              list,
                                struct timeval *    tv
                              )
```

在链表尾插入新节点节点

参数:

list 要插入节点的链表
tv 插入链表的节点内容

返回: 0，成功；-1，失败

示例:

```
struct timeval tmp;
struct timezone tz;
gettimeofday(&tmp, &tz);
if (dh_insert_node(head, &tmp) == -1)
{
     printf("Insert node failed.\n");
     return -1;
}
```

2.3 结构体注释

结构体、枚举以及联合等注释。

格式如下：

```
/**
 @brief    图像数据结构体(简要描述)

 这里是详悉说明,该结构体主要用于.....
 */
typedef struct _Image
{
  int width;            ///< 图像宽
  int height;           ///< 图像高
  size_t size;          ///< 图像大小
  int bpp;              ///< 图像颜色深度
  unsigned char *buffer;  ///< 图像缓冲区数据
}Image;
```

下面的注释风格也能有同样的效果：

```
/// 图像数据结构体(简要描述)
/**
 这里是详悉说明,该结构体主要用于.....
 */
typedef struct _Image
{
  int width;            ///< 图像宽
  int height;           ///< 图像高
  size_t size;          ///< 图像大小
  int bpp;              ///< 图像颜色深度
  unsigned char *buffer;  ///< 图像缓冲区数据
}Image;
```

///<　　//Doxygen 的 C++行间注释风格；

上面的注释生成 HTML 文档后效果如下：

a)　数据结构列表显示**简要描述**内容：

b)　结构参考处列出**详细描述**和**行间注释**

2.4 其它常用注释

如参考关系注释：

```
/**
 @brief    图像数据结构体(简要描述)
 @sa M2_3D_Point

 这里是详悉说明,该结构体主要用于.....
 */
typedef struct _Image
{
  int width;            ///< 图像宽
  int height;           ///< 图像高
  size_t size;          ///< 图像大小
  int bpp;              ///< 图像颜色深度
  unsigned char *buffer;  ///< 图像缓冲区数据
}Image;
```

上面的注释生成 HTML 文档后效果如下：

Doxygen 的还有很多的注释关键字，上面介绍的这些关键字已经足够我们使用了。如果您还需要其它特殊的关键字，请参见《Doxygen_manal.pdf》

3 Linux 系统 C 语言使用

3.1 安装 Doxygen

网上搜寻最新的Doxygen，目前最新版本为doxygen-1.5.6，笔者找到的最新版下载地址http://www.stack.nl/~dimitri/doxygen/download.html#latestsrc，下载源码文件doxygen-1.5.6.src.tar.gz。

解压安装：

```
tar -zxvf  doxygen-1.5.6.src.tar.gz
cd doxygen-1.5.6
./configure --prefix=/usr    //指定安装路径
make && make install      //编译安装
```

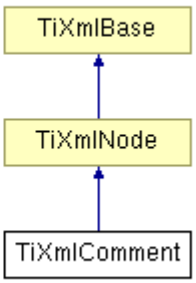
如果需要以图形方式显示函数之间的调用关系，还需要安装graphviz软件包。下载地址：http://www.graphviz.org/Download..php 这里下载的软件包版本为：graphviz-2.20.2.src.rpm

安装办法：

```
rpm -ivh graphviz-2.20.2.src.rpm    //安装源码包
cd /usr/src/redhat/SPECS
rpmbuild -bp graphviz.spec          //解压源码包并打补丁
cd /usr/src/redhat/BUILD
./configure --prefix=/usr           //指定安装路径
make && make install                //编译安装
```

Linux 系统安装这个软件后，生成 XML 文件时，Doxygen 会自动把程序源码里面类、函数、文件的包含/调用关系用图像化方式表示出来。如 TinyXML 中的类文件关系：

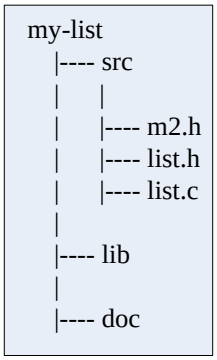
Inheritance diagram for TiXmlComment:



List of all members.

3.2 示例使用说明

这里构造一个简单的项目，目录结构如下图所示



切换至项目目录，使用“doxygen -g”命令生成 Doxygen 配置文件模板。

```
$ cd /home/my_list
$ doxygen -g
```

默认生成的配置文件名为"Doxyfile"，也可以采用"doxygen -g your-cfg-filename"命令格式指定所生成的配置文件名。如无特殊需要，采用默认的配置文件名即可。

Doxyfile 文件内容非常多，大概 1000 多行，不过其中约 4/5 都是注释，每个配置选项都有一段详细的注释。日后，如果对 Doxygen 各配置选项的意义有一定了解，可以在生成配置文件的命令中添加"-s"选项，生成不含注释的配置文件，操作如下：

```
$ doxygen -s -g
```

这里仅仅介绍几项与我们息息相关的项的设置：

```
PROJECT_NAME          = my_list         #项目名称，将作为于所生成的程序文档首页标题

DOXYFILE_ENCODING       =UTF-8   #文件编码格式，可选UTF-8/GB2312

# 文档版本号，可对应于项目版本号，譬如 svn、cvs 所生成的项目版本号
PROJECT_NUMBER      = “1.0.0”

# 程序文档输出目录
OUTPUT_DIRECTORY    = “doc/”
# 程序文档语言环境，可选项：Chinese/English
OUTPUT_LANGUAGE    = Chinese

# 如果是制作 C 程序文档，该选项必须设为 YES，否则默认生成 C++ 文档格式
OPTIMIZE_OUTPUT_FOR_C  = YES

# 对于使用 typedef 定义的结构体、枚举、联合等数据类型，只按照 typedef 定义的类型名进行文档化
TYPEDEF_HIDES_STRUCT   = YES

# 在 C++ 程序文档中，该值可以设置为 NO，而在 C 程序文档中，由于 C 语言没有所谓的域/名字空间这样的概念，
所以此处设置为 YES
HIDE_SCOPE_NAMES       = YES

# 让 doxygen 静悄悄地为你生成文档，只有出现警告或错误时，才在终端输出提示信息
QUIET   = YES

#输入文件编码，即您生成的代码时使用的文件编码，可选：UTF-8/GB2312
INPUT_ENCODING              =UTF-8

# 只对头文件中的文档化信息生成程序文档
FILE_PATTERNS         = *.h

# 递归遍历当前目录的子目录，寻找被文档化的程序源文件
RECURSIVE             = YES

# 示例程序目录
EXAMPLE_PATH          = example/

# 示例程序的头文档 (.h 文件) 与实现文档 (.c 文件) 都作为程序文档化对象
EXAMPLE_PATTERNS       = *.c \
                         *.h

# 递归遍历示例程序目录的子目录，寻找被文档化的程序源文件
EXAMPLE_RECURSIVE     = YES

# 允许程序文档中显示本文档化的函数相互调用关系
REFERENCED_BY_RELATION = YES
REFERENCES_RELATION    = YES
REFERENCES_LINK_SOURCE = YES

# 不生成 latex 格式的程序文档
GENERATE_LATEX        = NO

# 在程序文档中允许以图例形式显示函数调用关系，前提是你已经安装了 graphviz 软件包
HAVE_DOT              = YES
CALL_GRAPH            = YES
CALLER_GRAPH       = YES
```

准备好 Doxygen 的工作环境后，就需要根据 Doxygen 所定义的注释规则，对程序源码进行文档化。换句话说，就是在对程序源码添加注释时，要按照 Doxygen 的游戏规则来进行。其它注释规则也请参见《Doygen_manual.pdf》文档。

在开始生成程序文档，将终端的工作目录定位在 my_list 目录，然后键入：

```
$ doxygen your_cfg_name
```

your-cfg-name 是上面生成的 Doxygen 配置文件名，如果是使用"doxygen -g"生成的配置文件——Doxyfile，那么可以在终端里仅键入 "doxygen" 命令即可生成程序文档。生成的文档位于 my_list/doc/html 目录中，使用浏览器打开该目录中的 index.html 文件，即可看到自己的工作成果。

注意：Linux 系统下生成的都是 UTF-8 编码的 html 文件。浏览器打开 index.html 文件时请使用 UTF-8 编码方式打开。

4. Windows 系统 C++语言使用

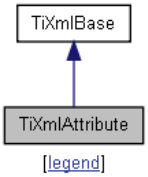
4.1 安装 Doxygen

网上搜寻最新的Doxygen，目前最新版本为doxygen-1.5.6，笔者找到的最新版下载地址http://www.stack.nl/~dimitri/doxygen/download.html#latestsrc，下载源码文件doxygen-1.5.6-setup.exe。双击直接安装即可。

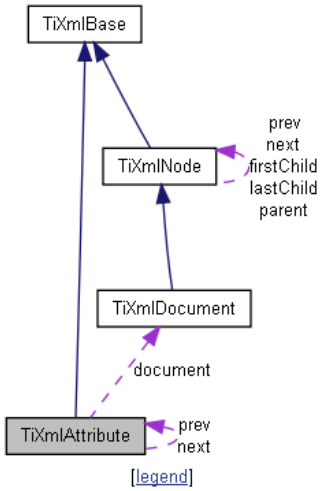
如果需要以图形方式显示函数之间的调用关系，还需要安装graphviz软件包。下载地址：http://www.graphviz.org/Download..php 这里下载的软件包版本为：graphviz-2.20.2.exe。双击直接安装即可。

Windows 系统安装这个软件后，生成 XML 文件时，Doxygen 会自动把程序源码里面类、函数、文件的包含/调用关系用图像化方式表示出来。这是我使用 TinyXML 源码生成的 HTML 时自动画出的关系图：

Inheritance diagram for TiXmlAttribute:



Collaboration diagram for TiXmlAttribute:



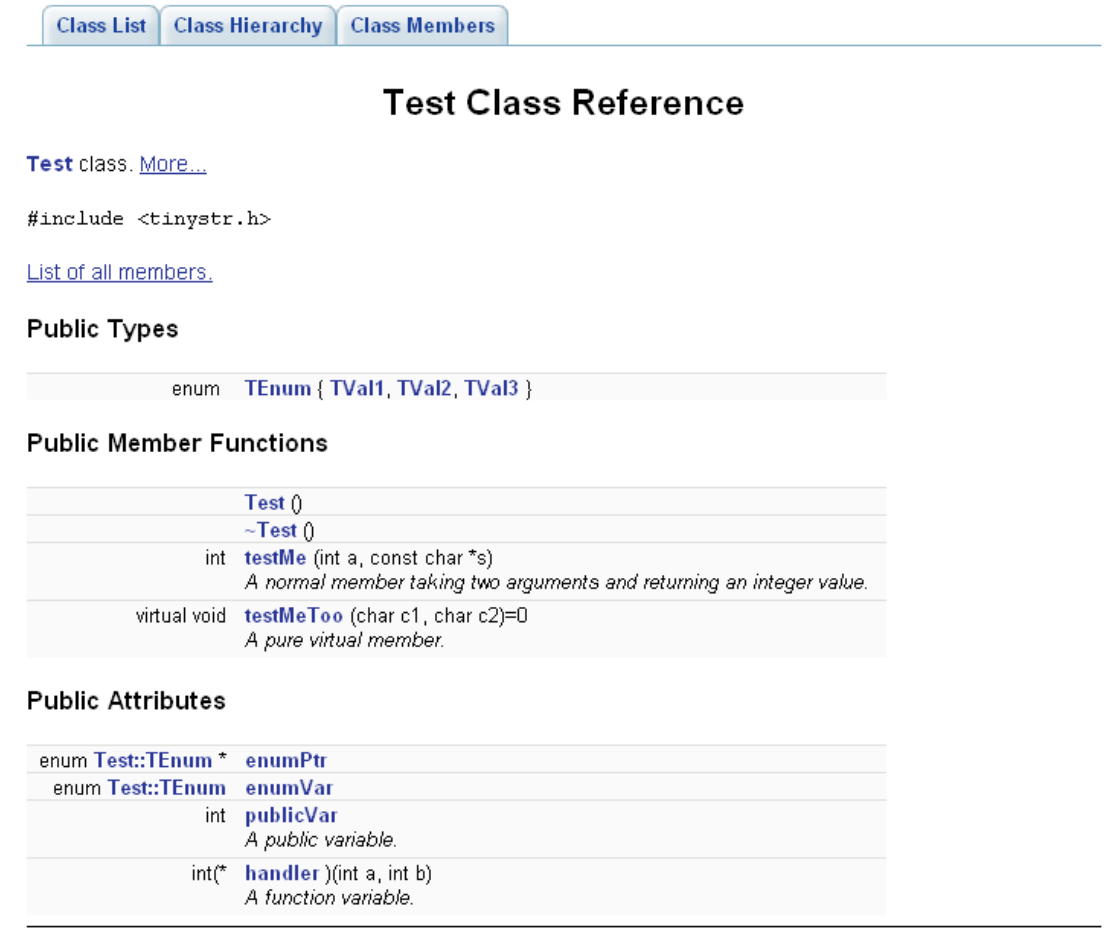
List of all members.

4.2 C++注释介绍

下面是类的定义的注释，包括了常用的 Doygen 注释风格：

```
/** A more elaborate class description.*/
class Test
{
   public:
   /** More detailed enum description. */
        enum TEnum
        {
             TVal1, ///< Enum value TVal1.
             TVal2, ///< Enum value TVal2.
             TVal3 ///< Enum value TVal3.
        }

        /** Details. */
        *enumPtr,
         /** Details. */
        enumVar;

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

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

        /// 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
             @sa Test(), ~Test(), testMeToo() and publicVar()
        */
        int testMe(int a,const char *s);

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

        /// A public variable.
        /**      Details.    */
        int publicVar;

        /// A function variable.
        /**      Details.    */
        int (*handler)(int a,int b);
};
```

上面的 C++类的注释生成 HTML 文档后效果如下：



Class List | Class Hierarchy | Class Members

Test Class Reference

Test class. More...

```
#include <tinystr.h>
```

List of all members.

Public Types

enum	**TEnum** { **TVal1**, **TVal2**, **TVal3** }

Public Member Functions

	Test ()
	~Test ()
int	**testMe** (int a, const char *s) *A normal member taking two arguments and returning an integer value.*
virtual void	**testMeToo** (char c1, char c2)=0 *A pure virtual member.*

Public Attributes

enum **Test::TEnum** *	**enumPtr**
enum **Test::TEnum**	**enumVar**
int	**publicVar** *A public variable.*
int(*	**handler**)(int a, int b) *A function variable.*

Member Function Documentation

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

A normal member taking two arguments and returning an integer value.

Parameters:
a an integer argument.
s a constant character pointer.

Returns:
The test results

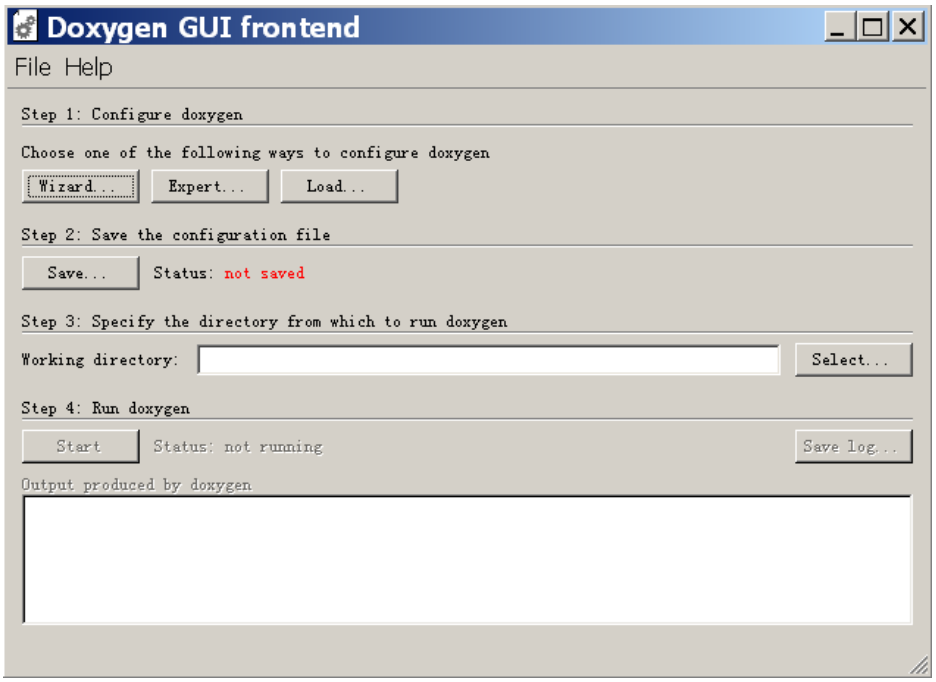
See also:
Test(), **~Test()**, **testMeToo()** and **publicVar()**

4.2 Windows 示例使用介绍

这里使用 TinyXML 的文件头为例，主要包括头文件 tinyxml.h，tinystr.h。并在 tinystr.h 中添加了上面所列举的 test 类。

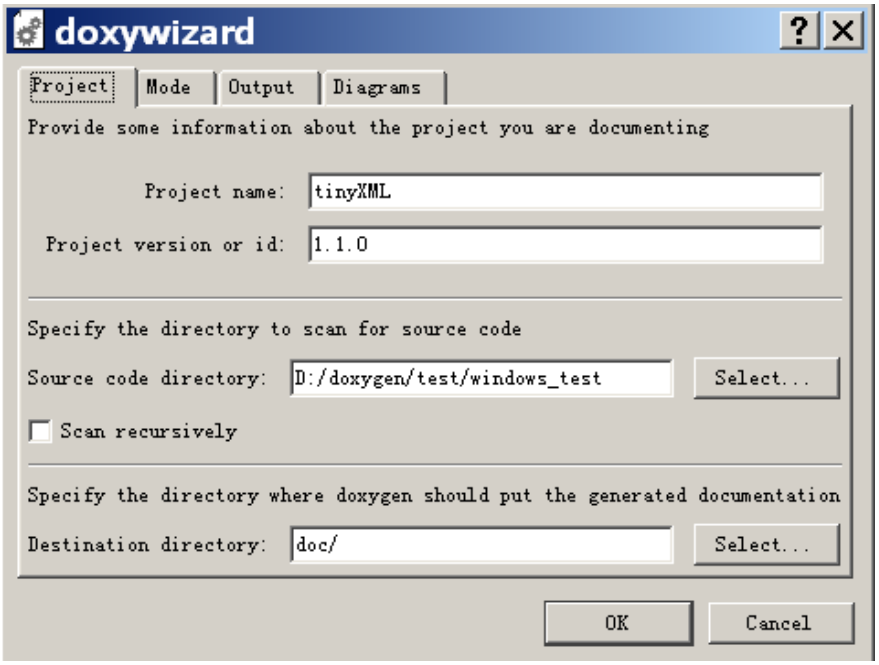
4.2.1 运行 Doxygen

选择开始→程序→doxygen→Doxywizard，运行 Doxywizard 创建配置文件，弹出界面如下图所示：



4.2.2 设置工程

Step1 下的按钮“Load…”按钮，导入已经配置好的 Doxygen 文件；选择 Step1 下的按钮“Wizard…”,弹出工程设置图

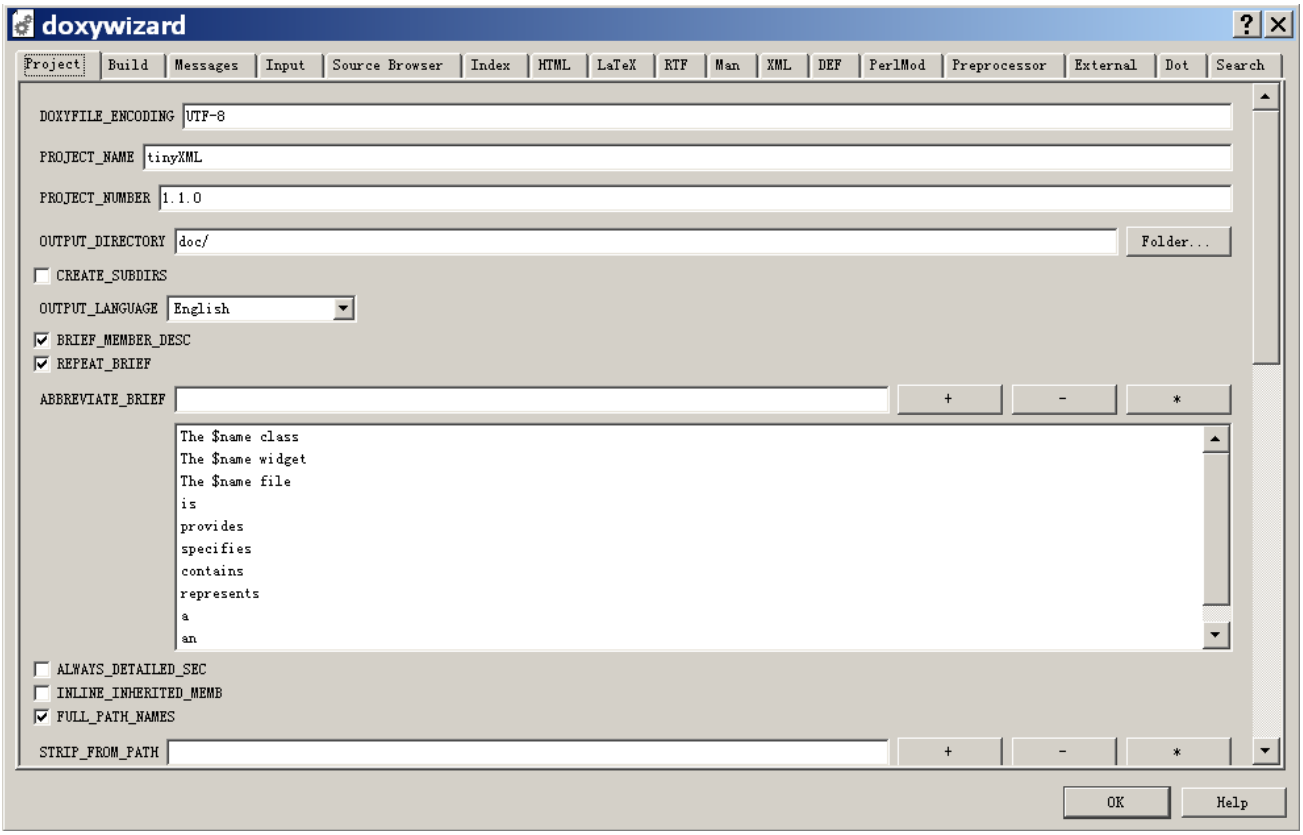


各选项意义：

a) Project：设置好工程名、版本、工程源码路径、以及生成文档的路径；
b) Mode：选择需要优化的编程语言（C++）；
c) Output：选择 HTML 文档格式输出；
d) Diagrams：选择使用 Graphviz 生成图，并选上“Call Graphs”选项。

4.2.3 参数设置

选择 Step1 下的“Expert…”按钮，弹出参数设置图：



各选项意义：

a) Project：这里主要选择输出语言“OUTPUT_LANGUAGE”，可支持中文文档输出。注意：选者中文语言输出文档时，其输入文档（一般是头文件）的编码格式必须与“Input”选项中设置的格式一致，否者头文件中的中文字符在输出文档中会显示为乱码;
b) Message：“WARN_LOGFILE”保存日志；
c) Input：选择源码路径，如：“D://doxygen/test/windows_test/src”，“INPUT_ENCODING”里输入文件编码格式：UTF-8 或 GB2312；
d) Source Browse：选上“Source Browse”，则生成的文档里面包含源码；
e) HTML：选上“GENERATE_HTML”，“HTML_OUTPUT“里可选 HTML 文件输出目录；

f) Dot：选上“HAVE_DOT”，“CLASS_GRAPH”，“CALLER_GRAPH”

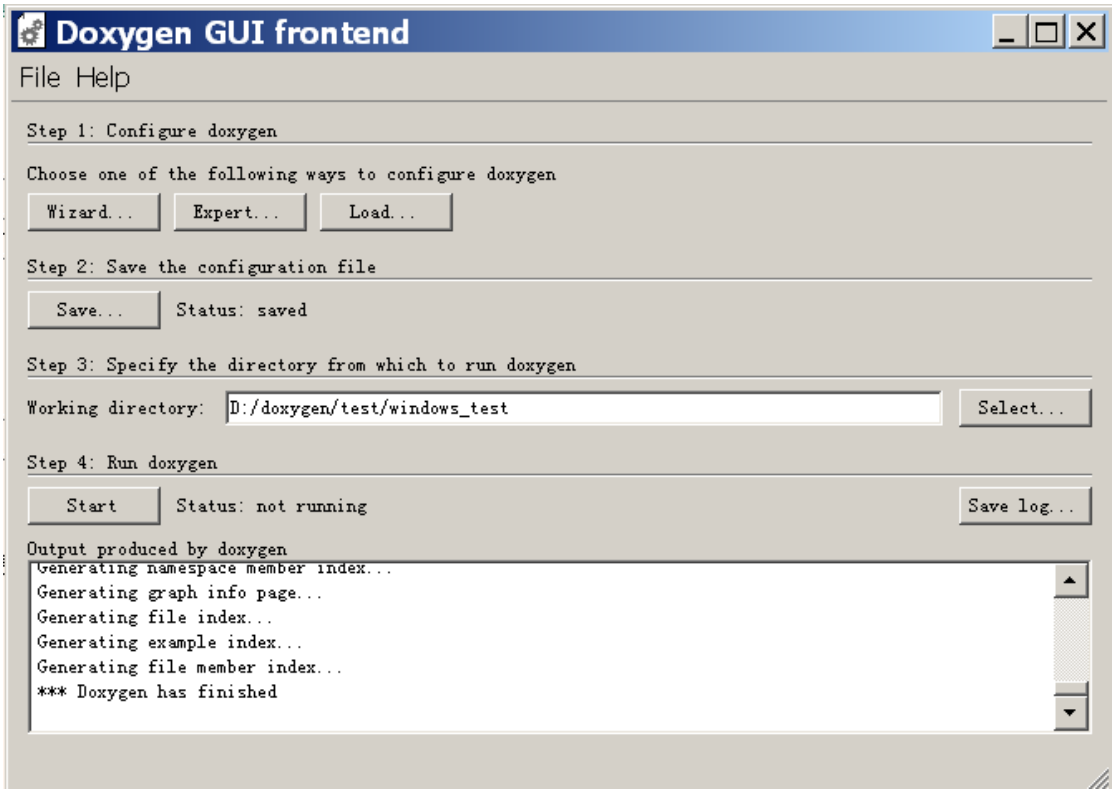
g) LaTeX、RTF、Man、XML、DEF 等都不需要设置。

4.2.4 运行 Doxygen

选择 Step2 下的按钮“Save…”，选择配置文件存放位置，默认的文件名为“Doxygen”。

在 Step3 输入运行 Doxygen 的路径；

选择 Step4 下的按钮“Start”，即可生成 HTML 文件；

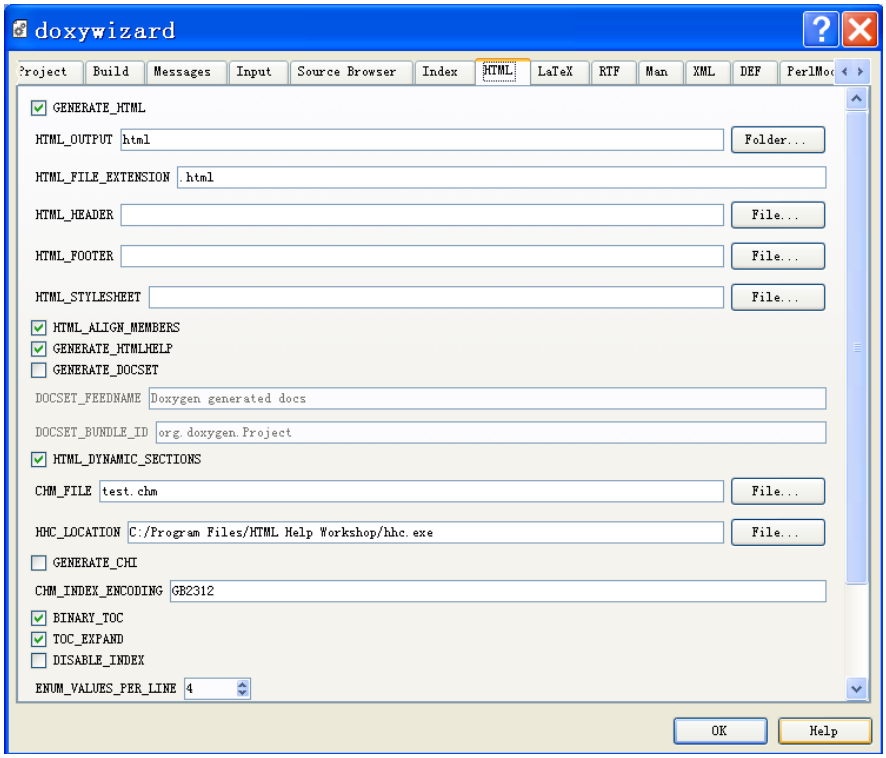


如果生成时没有错误提示，那么就可以在目录“./doc”下找到生成的 html 文档。doc 目录下会有很多中间文件“*.map”、“*.md5”；发布时把这些中间文件删除即可。

4.2.3 生存 CHM 文档方式设置

如果要让doxygen支持输入chm格式的文档，还要安装htmlhelp.EXE，下载网址：http://go.microsoft.com/fwlink/?LinkId=14188 配置doxygen：

要生成 chm 文档，需要设置如下：



1) 选中 GENERATE_HTMLHELP、BINARY_TOC、TOC_EXPAND；
2) 在 HHC_LOCATION 找到您安装 htmlhelp 的安装目录里面的 hhc.exe 文件（如上：C:/Program Files/HTML Help Workshop/hhc.exe）；
3) 在 CHM_FILE 中输入您想要输出的 chm 文件名称（如：test.chm）；

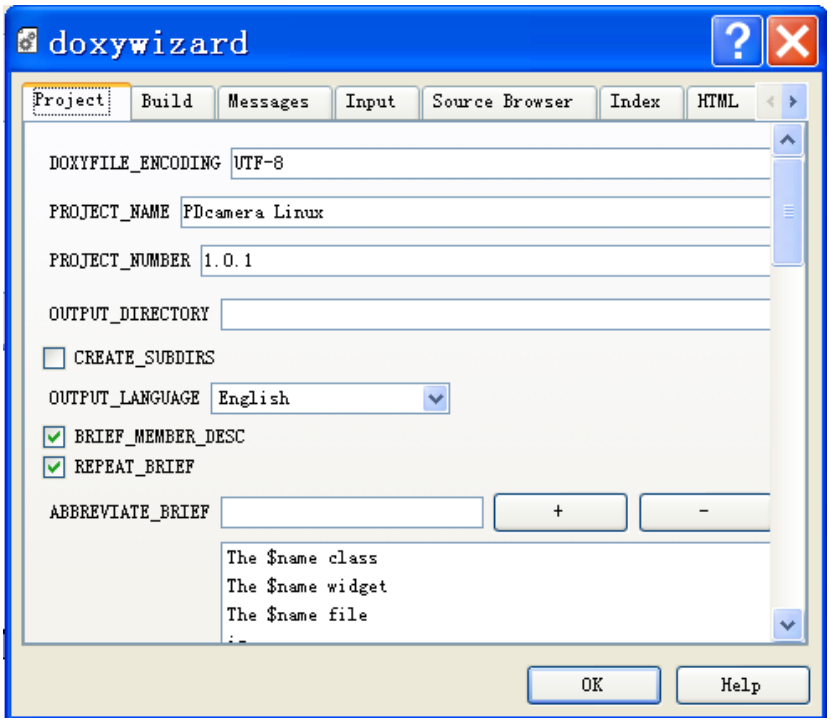
4.2.4 中英文切换

为了方便起见，需要注释能输出中文帮助和英文帮助，因此需要修改注释方式和内容，条件定义字符“English”可以自由设定：

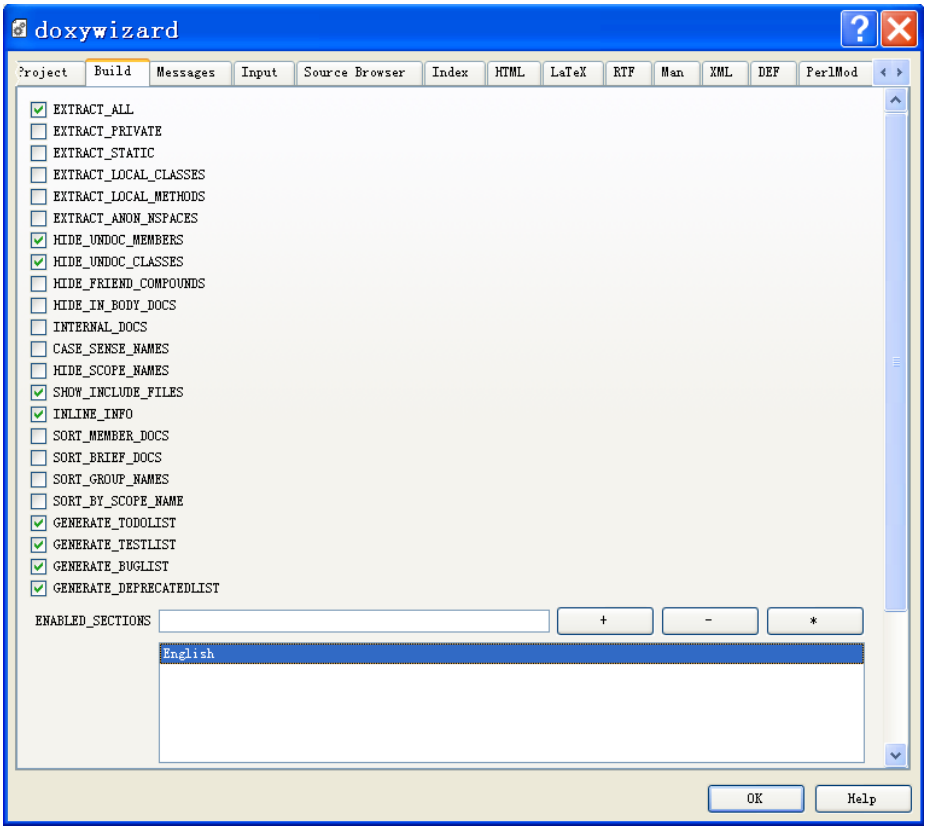
```
/**
@if English
 @brief Find camera

 This function find all the PD-Camera on the net, and return camera list and cameras.
 @param device  Pointer to dh_device_t array, store the founded camera's info;@sa dh_device_t
 @param count Pointer to a num, store the number of found camera.
 @return success, DH_STATUS_SUCCESS;
       failed, other. see dhcam_status.h
@else
 @brief 查找设备

 该函数负责查找局域网内的相机，返回所有相机列表和相机个数。注意：如果您再次调用该函数查找设备时候，返回列表中的相机顺序就有可能和上一次查找的不一致。
 @param device 返回相机列表数组，存放所有找到的相机信息；@sa dh_device_t
 @param count 返回找到的相机个数。
 @return 成功返回状态码, DH_STATUS_SUCCESS;
       失败返回其它状态码。参见 dhcam_status.h
@endif
*/
```

代码注释如上所示，如果需要生成**英文文档**，配置文件需要修改以下内容：

1) Project 选项，OUTPUT_LANGUAGE 选中为 English；

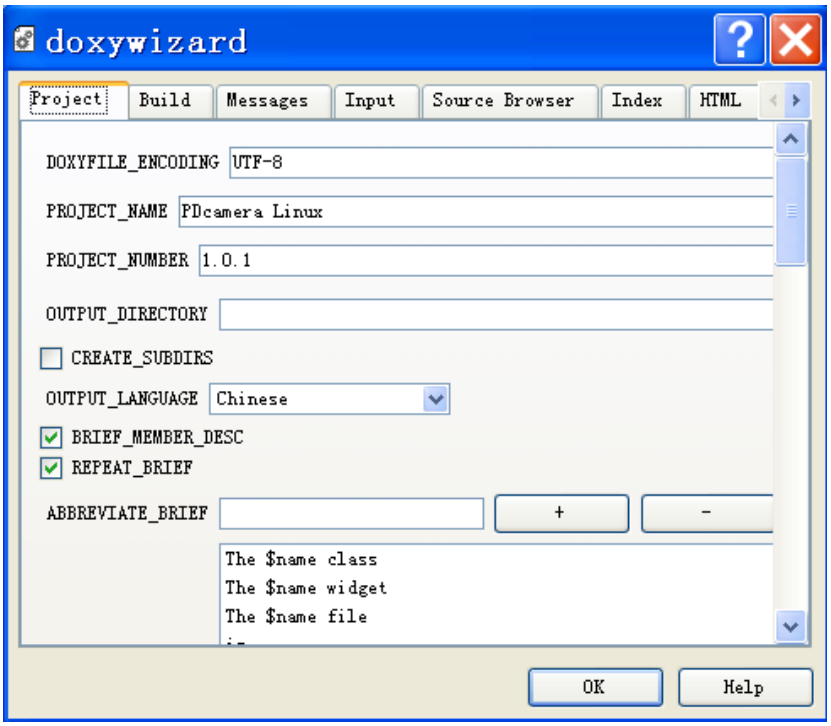


2) Build 选项，ENABLED_SECTIONS 项加入代码注释中的条件注释字符“English”。

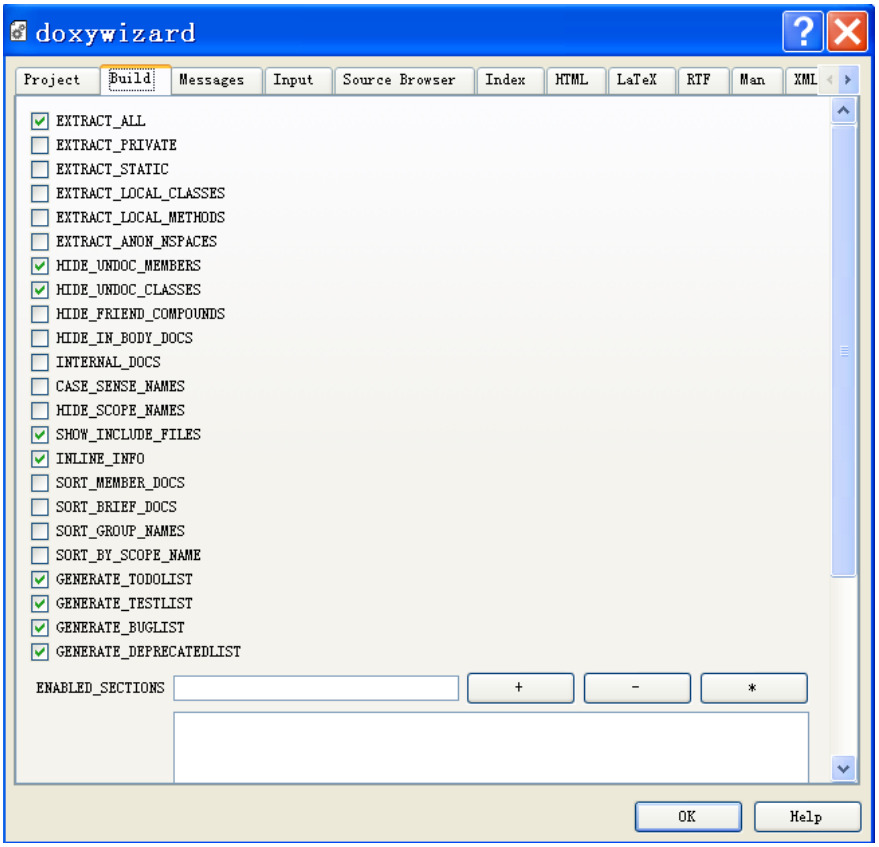


代码注释如上所示，如果需要生成**中文文档**，配置文件需要修改以下内容：

1) Project 选项，OUTPUT_LANGUAGE 选中为 Chinese；



2) Build 选项，ENABLED_SECTIONS 什么也不加入。



版本历史

版本	时间	备注
1.0.0	2008-08-22	初始版本
1.0.1	2009-02-24	增加了生成 chm 文档配置，中英文注释解释