**doxygen 주석문법**

이제 doxygen에서 doxygen문서를 만들기 위해서는 doxygen이 지원하는 형태의 주석을 달아야 합니다. 그 문법과 사용 예제에 대한 내용이 있습니다. 예제에 사용되는 주석 블럭은 C 스타일(주석블록 참조)을 따라 작성되었습니다.

doxygen을 이용한 헬프문서를 만들때(1.3.9.1)에는 source/header파일에 doxygen에서 지원하는 문법을 사용하면 좀 더 효과적인 문서를 만들 수 있습니다. 그 사용법중에서 사용빈도가 높은 항목들에 대한 설명입니다. 더 자세한 사항은<http://www.stack.nl/~dimitri/doxygen/manual.html> 에서 확인 하세요.

**주석 블럭**

**주석 블럭 작성하기**

주석 블럭을 사용하는 방법입니다. doxygen은 여러 언어의 주석 블럭을 지원합니다. 기존언어의 주석법에 문자를 하나 더 추가하여 사용하는 방법입니다. **반드시 설명하고자 하는 코드(함수선언/클래스선언/매크로, 상수선언)의 상단에 위치**해야 하며 doxygen최신 버전은 source파일에 대한 주석도 지원합니다.

**C 스타일**

C스타일은 전형적인 /\* \*/의 주석에 JavaDoc[?](http://wiki.rabidus.net/ow.asp?p=JavaDoc" \o "Describe this page" \t "_blank)스타일로 '\*'를 두개 붙여서 사용해야 합니다.

/\*\*
comment
\*/

/\*\* comment \*\*/

**C++ 스타일**

C++스타일은 전형적인 주석 방법인 // 문자에 / 문자나 !문자를 를 하나더 추가 하여 사용해야 합니다. **C++스타일을 추천합니다.**

/// comment 1
/// comment 2
/// comment 3

//! comment 1
//! comment 2
//! comment 3

//! comment //
/// comment //

때로는 구분하기 위해서 블럭표시를 하고 사용하기도 합니다.

////////////////////////////////////////////////////////////////////////////////
/// comment
////////////////////////////////////////////////////////////////////////////////

// ----------------------------------------------------------------------------
/// comment
// ----------------------------------------------------------------------------

**Qt스타일**

Qt스타일은 \*문자뒤에 !문자를 추가하여 사용하는 방법입니다.

/\*!
comment 1
comment 2
\*/

/\*! comment \*/

**주석기술하기**

주석블록에서 이제 주석을 달기 시작할때 해당 주석에 대한 제목이나 요약을 적는 방법을 소개합니다. 주석 설명은 해당 함수/클래스/매크로/상수값등이 무엇인지 설명하는 것입니다. 설명은 반드시 마침표로 끝나야하며 **주석블록에서 처음 항목에 기술되어야 합니다.** 여러줄도 기술이 가능합니다.

/\*\* this class is sample class.
next line.
\*/
class CSampleClass{};

/\*\*
this class is sample class.
next line.
\*/
class CSampleClass{};

**주석명령(comment command)**

기본적으로 위의 사항으로 정보를 출력 할 수 있지만 좀 더 효과적이고 doxygen이 지원하는 기능을 사용하려면 doxygen에서 제공하는 명령을 사용하면 문서가 더 잘 꾸며지게 됩니다. 그 중에서 필요한 몇가지 명령에 대한 설명입니다.

명령(command)를 사용할때는 '@'문자나 '@'문자를 prefix로 가집니다. 작성자 표시를 하는 author의 경우 \author, @author로 사용 할 수 있습니다.

**요약정보 기술하기**

doxygen에서는 문서의 첫부분에 해당 파일의 모든 항목의 리스트와 prototype등을 기술하게 되는데 이때 상세설명(detail decoration)이나 리스트에 설명을 추가 하고 싶을때는 @brife 명령을 사용하여 항목 리스트에 상세설명이나 기록한 설명이 보여지게 할 수 있습니다. **사용은 주석블럭 처음 항목에 @brife를 기술하고 마침표로 끝나는 설명문들 추가하면 됩니다.**

/\*\* @brief description.
detail description line 1
detail description line 2
\*/

이때 brief에 기술한 decoration 문장이 항목 리스트에 보여지고 링크를 따라오면 상세 설명을 볼 수 있습니다.

**상제정보 기술하기**

상세정보 블럭에서 여러가지 항목을 표현 할 수 있습니다. 이렇게 사용된 명령은 doxygen으로 만들어지 문서에서 정해진 포맷으로 출력됩니다.

@author: 작성자 표시를 합니다.
@date: 작성일등의 날짜 표시를 합니다. 직접 기술 해야 합니다.
@see: 참고하는 함수/클래스등의 표시로 해당 항목(doxygen에서 존재하는)의 링크를 제공합니다.
@return: 함수의 리턴값에 대한 설명
@exception: exception처리에 대한 설명
@throw: throw하는 객체나 변수등에 대한 설명
@version; 버전설명
@param: 파라미터에 대한 설명 ex) @param [파라미터이름] [파라미터설명]
@warning: 경고
@bug: 버그에 대한 기술, 전체 항목의 @bug에 대해서 따로 리스트가 생성됩니다.
@todo: 해야할일들에 대한 기술, 전체 항목의 @todo에 대한 리스트가 생성됩니다.
@param: 함수의 파라미터 대한 정보를 기술 합니다.

다음이 예제입니다. 어떤 함수에 대한 주석을 작성하였습니다.

/\*\* brief brief description of somefunc().
@author rabidus
@date 2004-11-1
@see refer\_doxyfunc();
@param param\_first it is count
@param param\_second it is name
@exception none
@throw none
@bug memory leak
@todo fix bugs
@warning none
\*/
bool somefunc(int param\_first, char\* param\_second);

**변수(파라미터, 데이터 멤버변수, 전역변수)정보 기술하기**

일반적으로 함수, 클래스, 구조체(struct), 공용체(union), 매크로, 상수값에는 위와 같은 방법으로 기술하면 되지만 변수에 대해서는 다른 방법으로도 기술 할 수 있습니다. 기존에 사용하는 주석 블럭 표시에 '<'문자를 붙여서 사용하면 변수에 대한 문서화 작업을 하게 됩니다. 이 방법은 변수 / 데이터 멤버 (변수/함수)에 모두 적용됩니다.

int g\_TotalCount; ///< Total Count of People

/\*\*
@brief brief decsription of CSomeClass.
@author rabidus
@date 2004-11-1
@todo nothing
\*/
class CSomeClass {
public:
CSomeClass();
~CSomeClass();
public:
int Init(); ///< Class data member init
void Reset(); ///< Class data member reset
public:
int m\_DataMember1; ///< data member 1 description
int m\_DataMember2; ///< data member 2 description
};

**항목을 지정하여 설명 달기**

함수 / 클래스 / 구조체 / 변수(멤버, enum, 전역)등의 항목에는 반드시 해당 코드 상위에 코드를 작성하지 않아도 주석 블럭 처음에 해당 항목을 정확히 지시(indicate)해주면 주석을 한곳에 모아서 작성 할 수 있습니다. 하지만 이때 반드시 정확한 정보를 작성해야 하는데 이것은 코드부와 주석이 다른 파일에 존재 할 수도 있기 때문입니다. 예를 들어 클래스의 경우 @class이후에 선언되어 있는 헤더 파일, 문서에 기록될 헤더파일 경로를 수동으로 적어 주어야 합니다. 다음이 그 예제에 따라 작성 하시면 됩니다.

class CIndicatesClass {
public:
enum ClassEnum{enum1, enum2};
};
void IndicatesFunc();
#define INDICATES\_MACRO 1
int g\_IndicateVar;
typedef struct INDICATE\_STRUCT {};

/\*\* @class doxygen.h "doxgen.h"
Indicate that comment of CIndicateClass
\*/

/\*\* @fn void IndicatesFunc()
Indicate that comment of IndicatesFunc
\*/

/\*\* @def INDICATES\_MACRO
Indicate that comment of INDICATES\_MACRO
\*/

/\*\* @var g\_IndicatesVar
Indicate that comment of g\_IndicatesVar
\*/

/\*\* @typedef INDICATE\_STRUCT
Indicate that comment of INDICATE\_STRUCT
\*/

/\*\* @enum CIndicatesClass::ClassEnum
Indicate that comment of CIndicatesClass::ClassEnum
\*/

/\*\* @var CIndicatesClass::ClassEnum CIndicatesClass::enum1
Indicate that comment of CIndicatesClass::ClassEnum::enum1
\*/

**Text Decoration**

**링크 걸기**

소스가 문서화 되면 각각의 항목들은 링크를 가지게 됩니다. 이 링크를 문서 설명에 강제로 링크를 걸 수 가 있는데 이 때 사용되는 명령이 @ref입니다. @ref이후에 **존재하는 링크**의 이름을 적어주게 되면 그 항목이 페이지든 함수, 클래스등에 대해서 링크가 걸어지게 됩니다.

**글자 효과 주기**

bold, italic, argument문자 표현하기등이 각각의 명령인 @b, @e, @a에 따라 단어가 꾸며집니다.

**예제 코드 달기**

주석을 다는 중에 예제 코드등의 코드를 삽입시킬수가 있는데 시작블럭 표시와 종료블럭 표시를 해야 합니다. 각각 @code, @endcode라는 명령이며 그 안에 있는 내용은 코드 형태로 문서화 됩니다.

다음은 위 세개 항목에 대한 설명입니다.

/\*\* text decoration function.
@b bold, @e italic @a argument. @ref CSomeClass
@code
void main() {
text\_decoration\_function("decoration");
}
@endcode
\*/
void text\_decoration\_function();

**해당페이지를 메인페이지로 만들기**

주석블럭에 @mainpage를 사용하면 그 페이지가 헬프문서의 메인페이지가 된다.

**전체 예제 코드**

다음 예제 코드로 doxygen문서를 만들어 보세요

/// @file this\_file\_name.h
/// @date 2004-11-01, rabidus
/// @author rabidus
/// this file is sample of doxygen command
/// refer from: http://www.stack.nl/~dimitri/doxygen/manual.html
// ----------------------------------------------------------------------------
// section: detail comment and variable comment
// ----------------------------------------------------------------------------
bool refer\_doxyfunc();
// ----------------------------------------------------------------------------
/// @fn somefunc
/// @author rabidus
/// @date 2004-11-1
/// @see refer\_doxyfunc();
/// @param param\_first it is count
/// @param param\_second it is name
/// @exception none
/// @throw none
/// @bug memory leak
/// @todo fix bugs
/// @warning none
// ----------------------------------------------------------------------------
bool somefunc(int param\_first, char\* param\_second);
// ----------------------------------------------------------------------------
/// @brief decsription of CSomeClass.
/// @author rabidus
/// @date 2004-11-1
/// @todo nothing
// ----------------------------------------------------------------------------
class CSomeClass {
public:
CSomeClass();
~CSomeClass();
public:
int Init(); ///< Class data member init
void Reset(); ///< Class data member reset
public:
int m\_DataMember1; ///< data member 1 description
int m\_DataMember2; ///< data member 2 description
};

int g\_TotalCount; ///< Total Count of People

// section: indicates comment sample
// ----------------------------------------------------------------------------
class CIndicatesClass {
public:
enum ClassEnum{enum1, enum2};
};
void IndicatesFunc();
#define INDICATES\_MACRO 1
int g\_IndicateVar;
typedef int uint;

/// @class doxygen.h "doxygen.h"
/// Indicate that comment of CIndicateClass
///

/// @fn void IndicatesFunc()
/// Indicate that comment of IndicatesFunc
///

/// @def INDICATES\_MACRO
///Indicate that comment of INDICATES\_MACRO
///

/// @var g\_IndicateVar
///Indicate that comment of g\_IndicatesVar
///

/// @typedef uint
///Indicate that comment of uint
///

/// @enum CIndicatesClass::ClassEnum
///Indicate that comment of CIndicatesClass::ClassEnum
///

/// @var CIndicatesClass::ClassEnum CIndicatesClass::enum1
///Indicate that comment of CIndicatesClass::ClassEnum::enum1
///

// section: text decoration
// ----------------------------------------------------------------------------
/// text decoration function.
/// @b bold, @e italic @a argument. @ref CSomeClass
/// @code
/// void main() {
/// text\_decoration\_function("decoration");
/// }
/// endcode
void text\_decoration\_function();