

# Doxygen



안정적인 실행 확인, 23/01/20, 23/01/24

- ◆ Doxygen이 무엇이나?  
게임 개발 중에 클래스 관계 - 변수가 하는 역할 - 각 파일의 역할 등,  
복잡해지기 쉬운 정리 과정을 자동적으로 해주는 것!

Step 1. Visual Studio 확장에서 “Doxygen Comments”를 검색, VS에 포함시킨다! (VS 확장 업데이트를 위해 Visual Studio를 한번 꺾다 킨다)

Step 2.

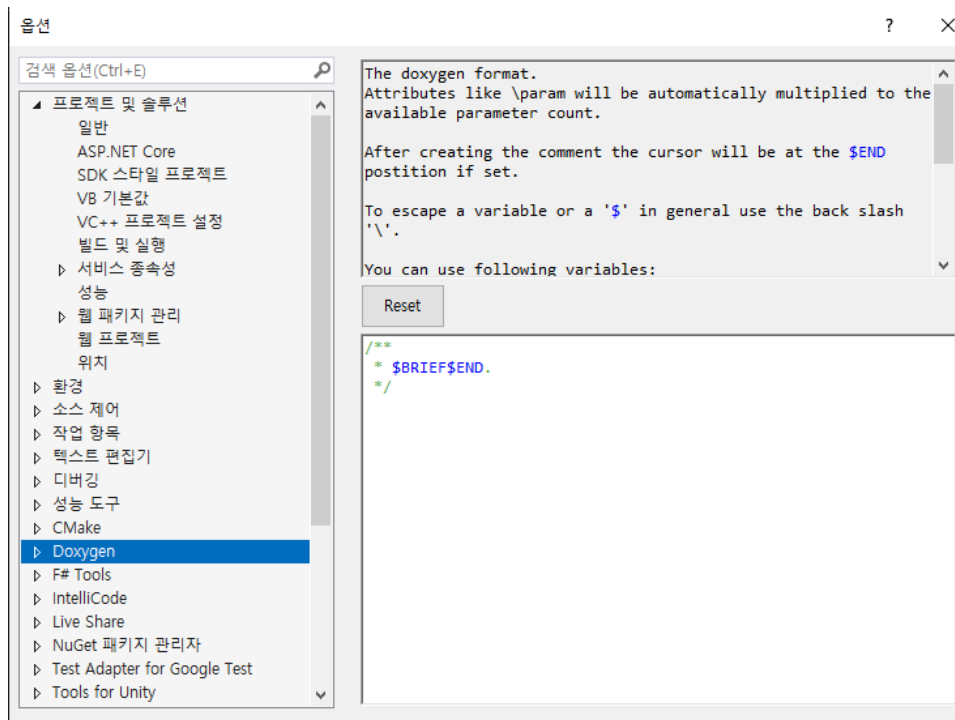
<https://www.doxygen.nl/files/doxygen-1.9.6-setup.exe>

(위 URL 누르면 그냥 설치됨, 컴퓨터 브라우저에서 클릭! 윈도우 기준)

인스톨러 설치, License만 동의한다고 하고 나머지 그대로 냅두면 됨

Step 3.

Visual Studio를 다시 열고, 도구 → 옵션 → Doxygen 들어가자.



Doxygen 토글을 열면, Default/Function/Header가 있다.

중요! Default는 냅두고,  
Function에는 원본 내용 날리고 밑 내용 Ctrl-C + Ctrl-V,

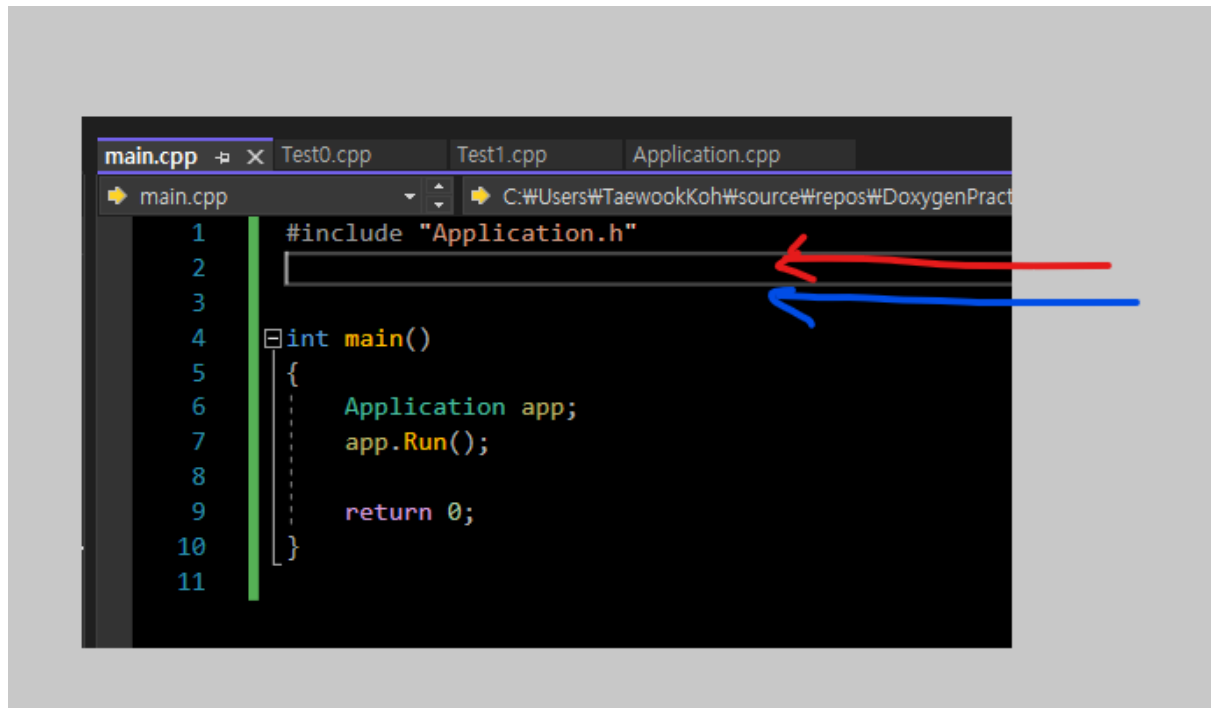
```
/**
 * $BRIEF$END.
 *
 * \details $DETAILS
 * \param $PARAMS
 * \return $RETURN
 * \pre $PRE
 * \todo $TODO
 * \note $NOTE
 * \warning $WARNING
 * \bug $BUG
 */
```

Header에는 원본 내용 날리고 밑 내용 Ctrl-C + Ctrl-V.

```
/**
 * \file $FILENAME
 * \brief $BRIEF$END
 *
 * \author $USERNAME
 * \date $MONTHNAME_EN $YEAR
 * \see $SEE
 */
```

## ■ 테스트 예제에 쓰인 코드 (1).

이제, 재미있는 일이 펼쳐짐! 밑 화살표 별로, 벌어지는 일이 다르다.



빨간 줄에서 /\*\*를 입력할 경우 (코드 파일의 1,2번째 줄에서 /\*\* 입력)

빨간색 화살표 줄 입력 경우, 자동으로

```

cpp C:\Users\TaewookKoh\source\repos\DoxygenPractice\Doxyg
1  #include "Application.h"
2  /*****
3   * \file    main.cpp
4   * \brief
5   *
6   * \author  TaewookKoh
7   * \date    January 2023
8   * \see    $SEE
9   *****/
10
11  int main()
12  {
13      Application app;
14      app.Run();
15
16      return 0;
17  }

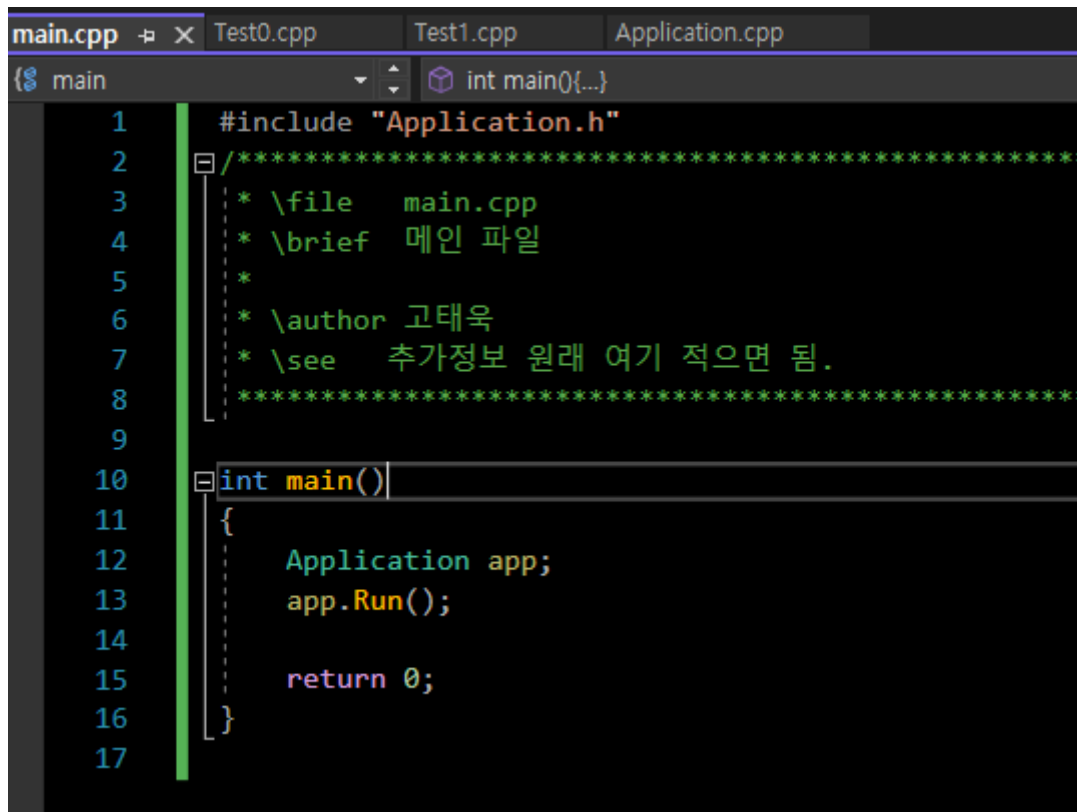
```

이 주석이 쳐진다!

- \file : 파일 이름
- \brief : 간략 설명
- \author : 작성자
- \date : 날짜
- \see : 추가 정보

여기서 필요없는 요소는 삭제하고, 뒤에 내용을 적으면 됨.

내 예시:



```
1  #include "Application.h"
2  /**
3   * \file    main.cpp
4   * \brief   메인 파일
5   *
6   * \author  고태욱
7   * \see    추가정보 원래 여기 적으면 됨.
8   */
9
10 int main()
11 {
12     Application app;
13     app.Run();
14
15     return 0;
16 }
17
```

Test0.h로 이동, 비슷한 처리를 해줬다.

파일 전체에 대한 Doxygen 주석을 봤으니,  
클래스/변수 별 등, 작은 단위의 주석은 어떻게 봐야 하는가?

주석을 쓰고 싶은 대상 바로 윗줄에 /\*\*를 입력하면, 이에 대한 주석이 작성됨.

**변수의 경우 →**

**그냥 설명하고자 하는 변수 윗줄에, /\*\*쓰고, 바로 Space 누르고, 쓰면 됨**

이렇게 써주었다.

**함수의 경우 →**

**그냥 설명하고자 하는 함수 윗줄에, /\*\*쓰고, 바로 Space 누르고, 쓰면 됨**

근데, 이렇게 된다면 조금 다른 결과가 나올 것이다.

이런 내용이 뜨는데, 마찬가지로 생략해줄 것 생략해주고,  
내용을 채워주면 된다.

- \details : 구체적 정보
- \pre : 함수 실행의 전제조건
- \todo : 할 일
- \note : 사용자에게 알려주고 싶은 내용

- \warning : 경고
- \bug : 버그

마찬가지로, 입맛대로 바꿔주면 된다.

내 예시:

```
/**
 * 키 값을 출력하는 함수.
 *
 * \details 아무튼 키 값을 출력함.
 * \todo 키 값 출력하기 전 ==> 화살표를 하나 그려주자.
 * \note 사용자에게의 메세지
 * \warning 경고!
 */
void PrintKey();
```

Test1.h로 이동한다. 비슷한 예시, 이번에는 모든 내용을 다 살려서 써볼 것.

```

class Test1

#pragma once
/*****
 * \file   Test1.h
 * \brief  테스트 일
 *
 * \author 고태욱
 * \date   23/01/24
 * \see    스펠링키 하고 싶다
 *****/

class Test1
{
public:
    /**
     * Test1의 생성자.
     *
     * \details 엄청난 디테일
     * \pre 엄청난 전제조건
     * \todo 엄청난 할일
     * \note 엄청난 노트
     * \warning 엄청난 경고
     * \bug 엄청난 버그
     */
    Test1();
    ~Test1();
    /**
     * 값을 나타내는 변수.
     */
    int value; //예시니까 그냥 public
    int Add(int _a, int _b);
    int Divide(int _a, int _b);
    void CountUpOne();
};

```

일단 이렇게 빠르게 바꾸어 놓았다.

하지만, 궁금증들이 생긴다. Ex:

- 포인터/참조자 같이, 함수 내부 내용에 따라 외부까지 영향 미치는 것은 어떻게 구분?
- 두괄식으로 설명을 추가하고 싶다면?

---

첫 번째 예시: 봐보자. *Test1.h*의 맨 밑에, *void PlusOut();* 추가.

이렇게 추가되었다.

이 경우에는, 값 // 참조, 포인터를 구분할 수 있을까?

이처럼, \param[in] , \param[out] 으로 구분하면 된다!

\param[in] : 함수 내부 변화 밖 영향 안끼침 (값 변수)

\param[out] : 함수 내부 변화 밖에도 영향 끼침 (참조/포인터 변수)

달라진 값을 “빼낸다고” 생각하자

이렇게 주석을 적어주었다.

---

다른 주제 ⇒ 두괄식으로 적어주고 싶은 경우,

\\i 내용1

\\i 내용2

이런 방식으로 이어나갈 수 있음!

---

결과물:

- 내용1
  - 내용2
- 

이것까지 변형해서 PlusOut을 변경해주었다.

이제 그만! 완성된 예시를 볼 차례.

**도구 → Update Doxygen Comments를 누르던가,  
Ctrl+Shift+R을 누르자.**

---

하지만 ⇒ 당장 아무 변화도 보이지 않을 것!

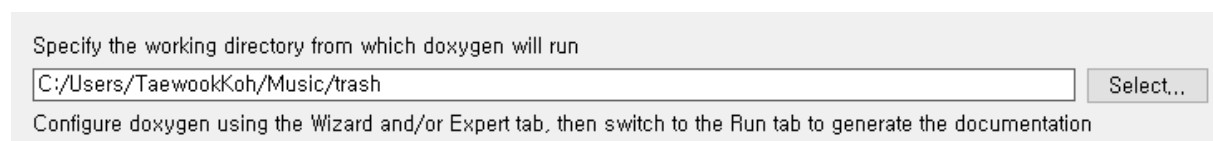
걱정 ㄴ! 당연하다.

Doxygen 프로그램을 돌려야 결과가 나옴.

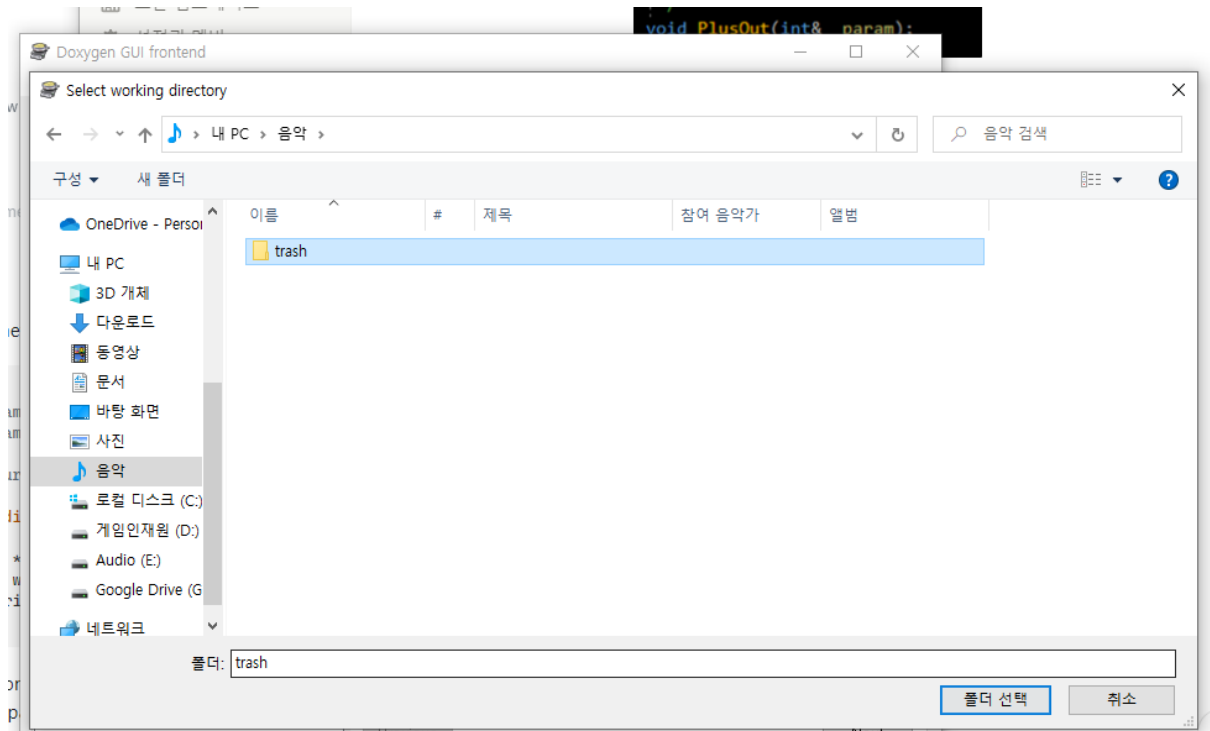
---

Doxywizard라는 앱을 킨다. 설치되어 있을 것이다.

맨 먼저 보이는 요거는 그냥 Doxygen이 내부적인 일하고 내용 없앨 폴더. 아무 폴더나 경로에 설정해주면 됨.







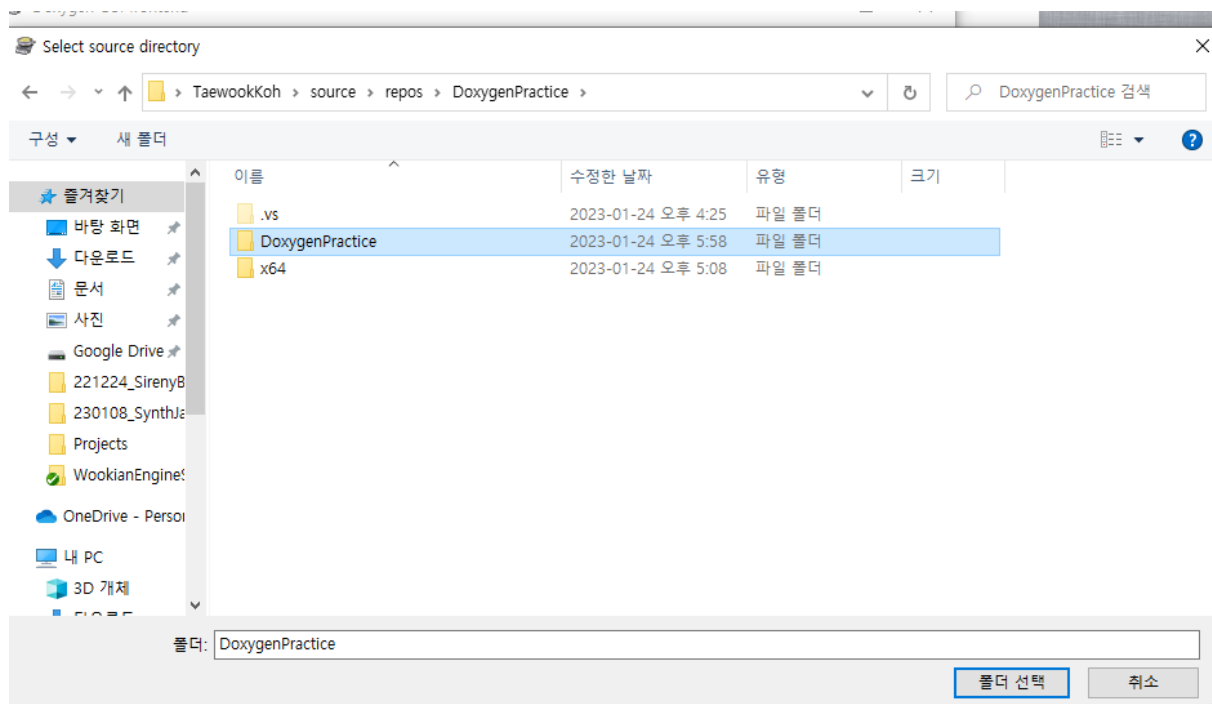
Wizard란의 Projects로 이동.

대충 이름 지어주고

소스코드 경로를 정하라는 밑의 칸에는

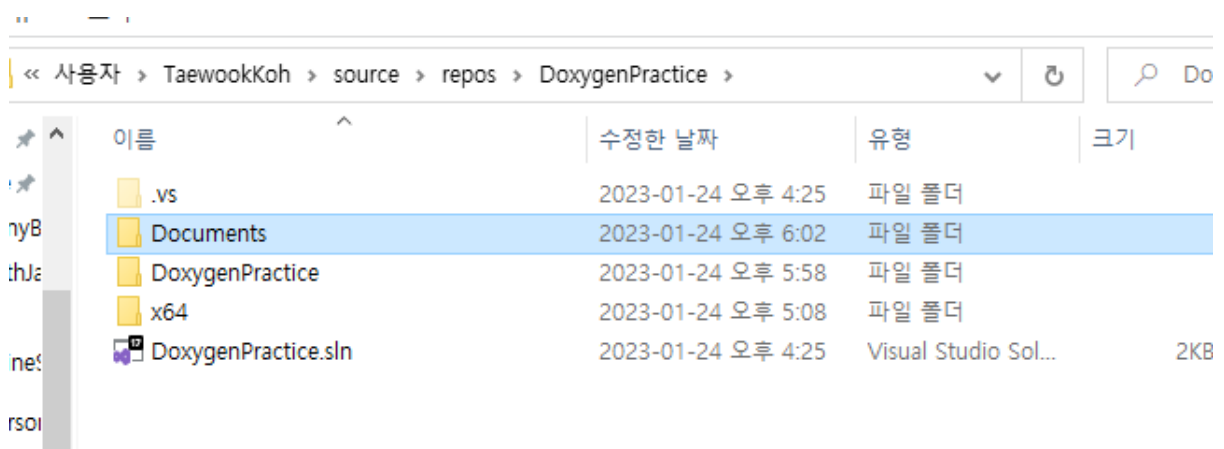
소스 코드들이 속해 있는 경로 복붙 (프로젝트 경로)

<< TaewookKoh > source > repos > DoxygenPractice > DoxygenPractice						
	이름	수정한 날짜	유형	크기		
	x64	2023-01-24 오후 5:08	파일 폴더			
	Application.cpp	2023-01-24 오후 5:46	C++ Source			
	Application.h	2023-01-24 오후 5:17	C/C++ Header			
	DoxygenPractice.vcxproj	2023-01-24 오후 5:08	VC++ Project			
	DoxygenPractice.vcxproj.filters	2023-01-24 오후 5:08	VC++ Project Filt...			
	DoxygenPractice.vcxproj.user	2023-01-24 오후 4:25	Per-User Project ...			
	main.cpp	2023-01-24 오후 5:20	C++ Source			
	Test0.cpp	2023-01-24 오후 5:15	C++ Source			
	Test0.h	2023-01-24 오후 5:38	C/C++ Header			
	Test1.cpp	2023-01-24 오후 5:54	C++ Source			
	Test1.h	2023-01-24 오후 5:58	C/C++ Header			



그 밑에 애는 완성된 HTML 파일이 어느 경로로 갈 것인가에 대한 정보.

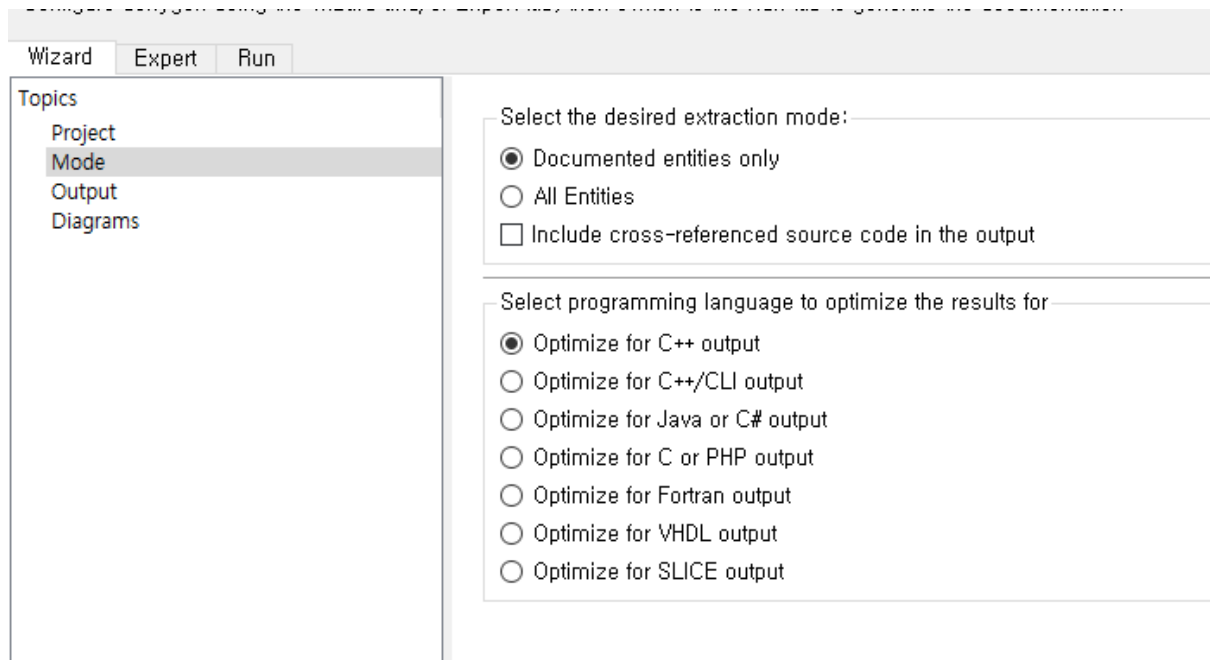
아무데나 (나는 그냥 솔루션 경로 내부 파일로 만들) 연결해준다.



Specify the directory where doxygen should put the generated documentation

Destination directory:

Mode는 그대로!



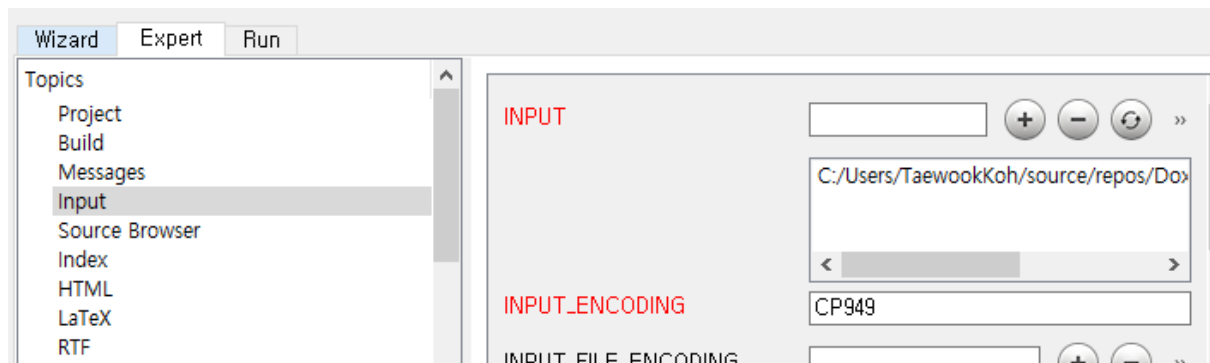
Output, Diagram도 일단 그대로. *Expert로 넘어가자!*

**Project** →

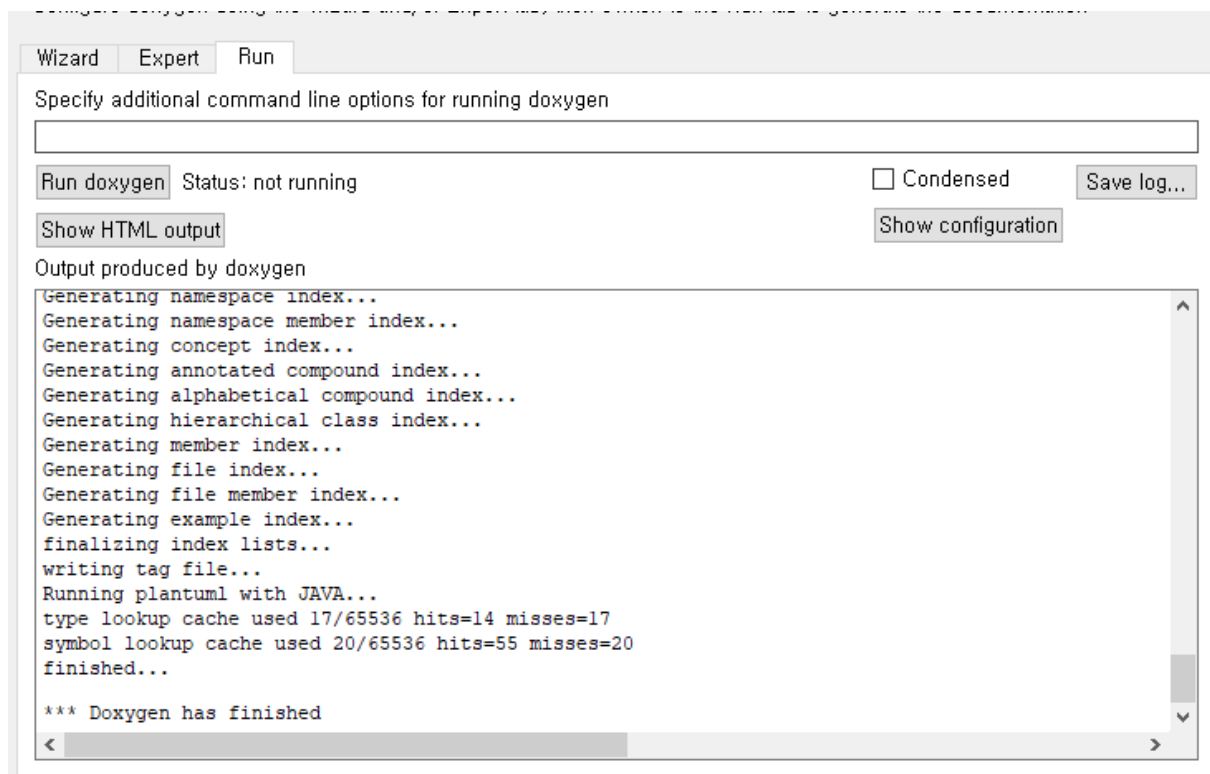
DOXYFILE\_ENCODING에서 EUC-KR,

Output Languages에서 Korean

Expert → Input → INPUT\_ENCODING을 CP949로 바꾼다.



세팅 끝! 이제, Run 탭으로 가고, Run Doxygen을 눌러준다!



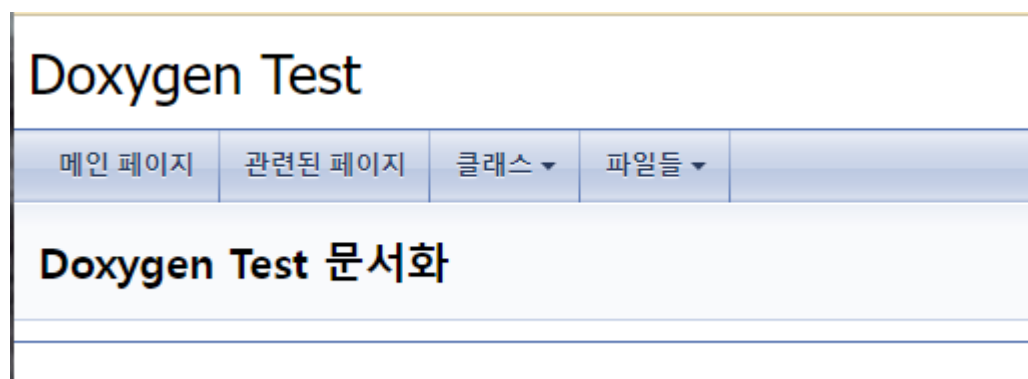
또 다시 개고생하는 것을 막기 위해, 설정을 저장하자!!!!

이제, 결과를 보러가야 하는데,

아까 만들었던 Documents 폴더로 다시 가자.

(아니면, 바로 옆에 Show HTML output 눌러도 됨)

결과:



결과:

◆ Test1()

Test1::Test1 ( )

Test1의 생성자.  
임정난 디테일

전제조건

임정난 전제조건

참고:

임정난 참고

주의

임정난 노트

경고

임정난 경고

버그:

임정난 버그

## 멤버 함수 문서화

◆ PlusOut()

void Test1::PlusOut (int & \_param )

유용한 예시.

- 하나
- 둘

조심!!

매개변수  
[out] 참조형 매개변수

..etc..