CodeScroll Controller Tester

3.2 — Last update: May 21, 2020

Suresofttech

Table of Contents

1. 시작하기 전에

문서의 목적

CodeScroll Controller Tester 제품의 사용법에 관한 정보를 제공합니다.

2. CONTROLLER TESTER 개요

CONTROLLER TESTER 소개

다양한 환경에서 개발 및 수행되는 소프트웨어에 대한 단위 및 통합 테스트를 수행하는 테스트 자동화 솔루션 입니다.

CONTROLLER TESTER 지원 툴체인

Controller Tester는 약 100종의 툴체인을 지원합니다. 자세한 내용은“Controller_Tester_지원_툴체인” 문서를 참고하시기 바랍니다.

CONTROLLER TESTER 특징

Controller Tester는 코드 기반 소프트웨어 단위/통합 테스트 자동화 도구로서 주요 특징은 다음과 같습니다.

테스트 설계

1. 자동으로 테스트 및 테스트 데이터를 생성
2. 직관적인 테스트 설계 인터페이스
3. 다양한 형식의 테스트 데이터 가져오기 기능
4. 복잡하게 중첩된 구조의 통합 테스트 설계 가능

커버리지 목표 달성

1. 구문, 분기, MC/DC, 함수 호출, 함수 커버리지 지원
2. MC/DC 목표 달성을 위한 가이드 제공
3. 소스 코드와 제어 흐름 그래프를 연동하여 커버리지 결과 확인
4. 시스템 커버리지 측정(QualityScroll COVER 제품) 후 커버되지 않은 부분만 테스트하여 빠르게 커버리지 목표 달성 가능

기타

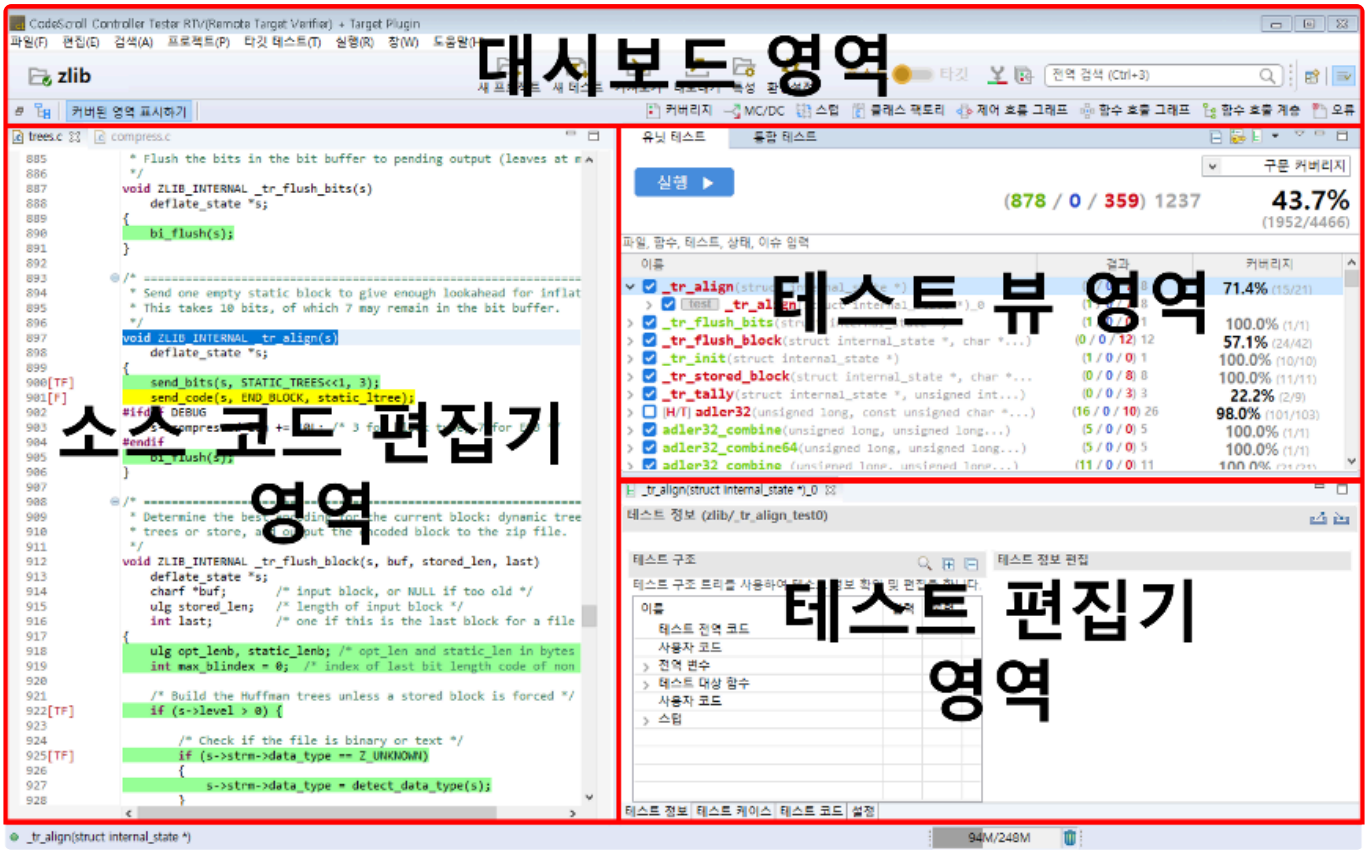
1. 시뮬레이션 및 실제 타깃 환경 테스트 지원
2. DevOps(이슈관리 + 지속통합 + 형상관리)도구와 연동
3. CodeScroll 제품군과 프로젝트 설정 공유: 한 번의 프로젝트 설정으로 코딩 규칙 검사 + 실행시간 오류 검출 + 단위/통합 테스트 가능
4. 다양한 형식의 보고서

표준 인증

1. IEC 61508-3 (전기/전자)
2. ISO 26262-8 (자동차)
3. IEC 60880 (원자력)
4. IEC 62279 / EN 50128 (철도)
5. DO-178C / DO-330 (항공)

CONTROLLER TESTER의 전체 화면

Eclipse RCP로 제작되어 자유도가 높은 창들을 제공합니다. [창] 메뉴와 각 뷰에서 사용자가 설정할 수 있습니다.



3. CONTROLLER TESTER 설치하기

설치 요구 사항

OS	Microsoft Windows 7/10 (32bit/64bit)
RAM	최소 512MB 이상
HDD	약 500MB(GCC Compiler 미설치) 또는 약 1GB(GCC Compiler 설치)의 여유 공간 (테스트가 수행되면 테스트 결과물로 인해 HDD 사용량은 더 늘어납니다.)

윈도우 기반 CONTROLLER TESTER 설치하기

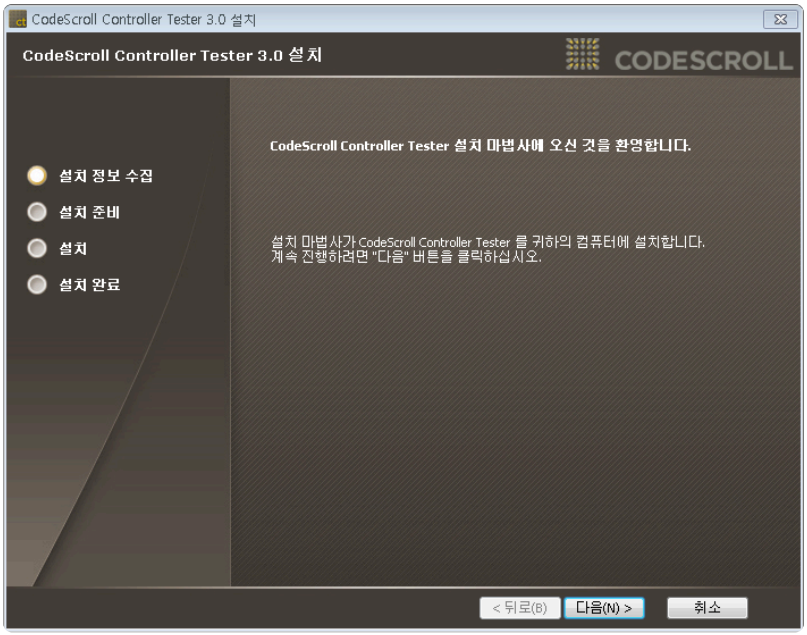
인스톨 패키지를 이용하여 Controller Tester를 설치하는 방법은 다음과 같습니다.

1. setup.exe 파일을 실행시킵니다.

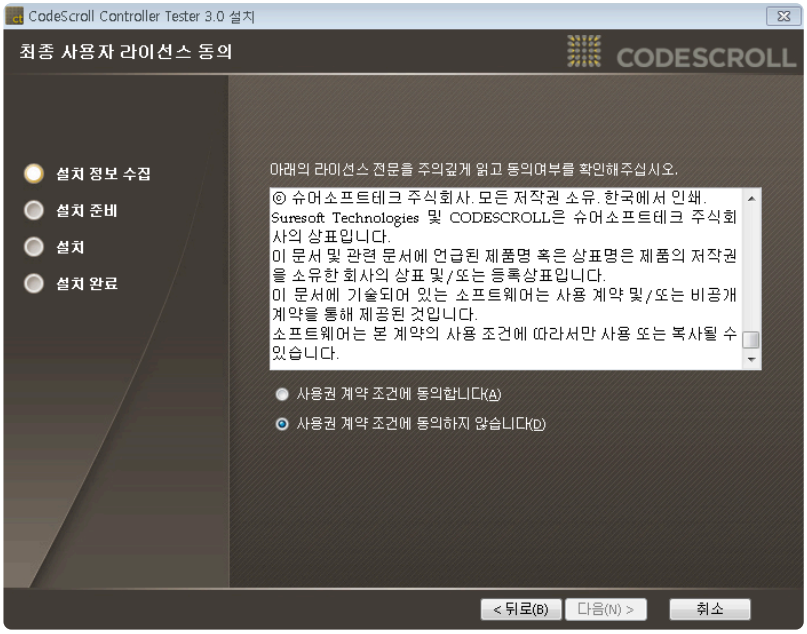
이름	수정한 날짜	유형	크기
CodeScroll Controller Tester_3.0.msi	2017-09-20 오전...	Windows Installer...	636,670KB
setup.exe	2017-09-20 오전...	응용 프로그램	1,349KB

> ! ‘CodeScroll Controller Tester_3.x.msi’로 설치 시 정상적으로 동작하지 않습니다.

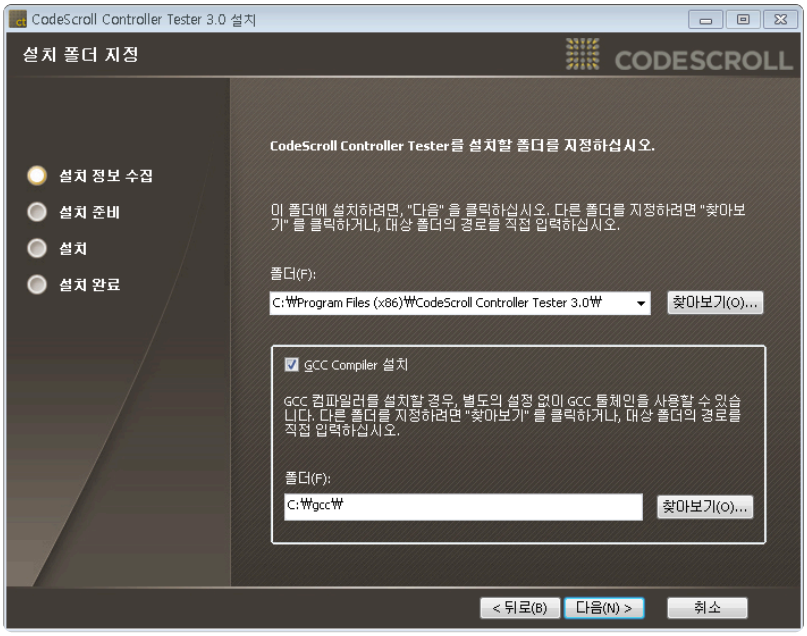
2. 설치 마법사가 실행되고 설치 정보를 수집합니다. [다음]을 클릭합니다.



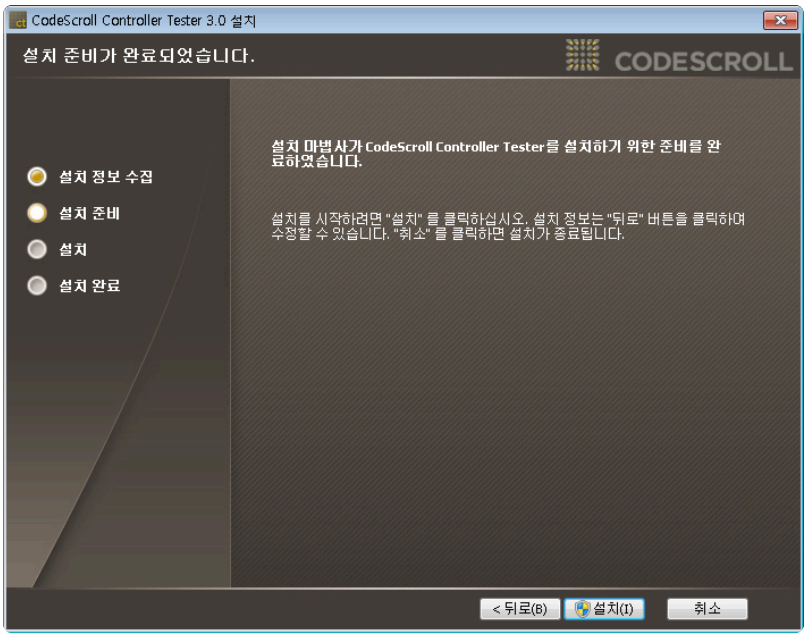
3. 최종 사용자 라이선스를 동의하고 [다음]을 클릭합니다.



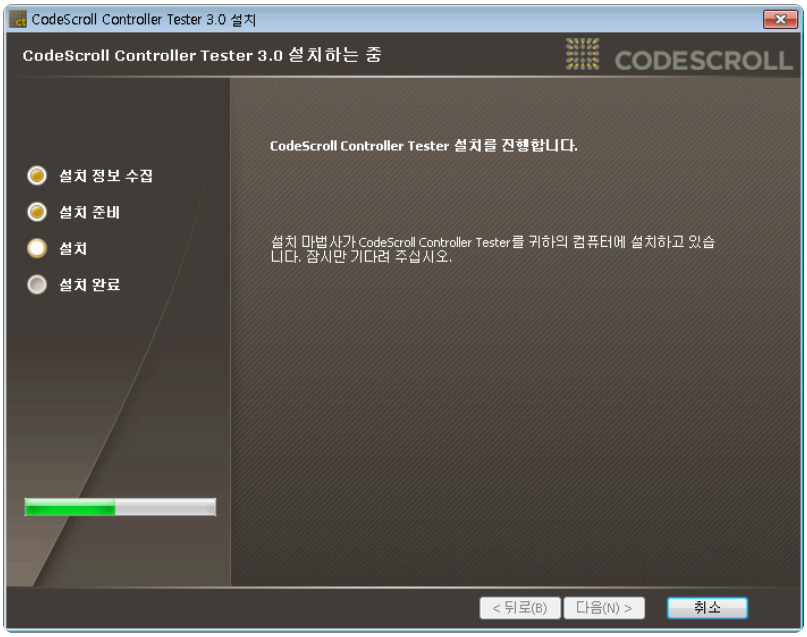
4. Controller Tester를 설치할 경로와 Controller Tester가 내장하고 있는 GCC 컴파일러의 설치 여부 및 경로를 설정하고 [다음]을 클릭합니다.



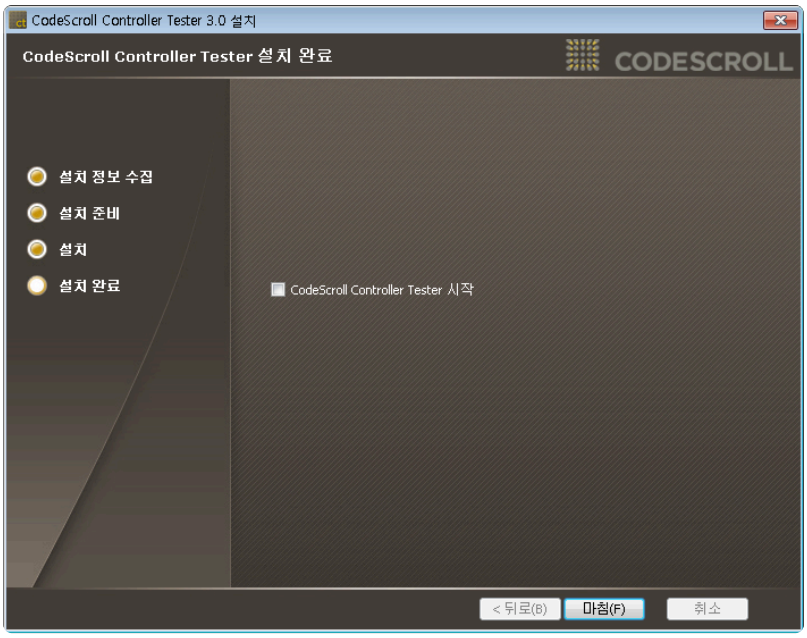
5. 설치에 필요한 정보를 모두 수집하였습니다. [설치]를 클릭합니다.



6. Controller Tester가 설치됩니다.



7. 설치 완료 후 Controller Tester를 바로 실행할 때는 [CodeScroll Controller Tester 시작]을 체크하고 [마침]을 클릭합니다.

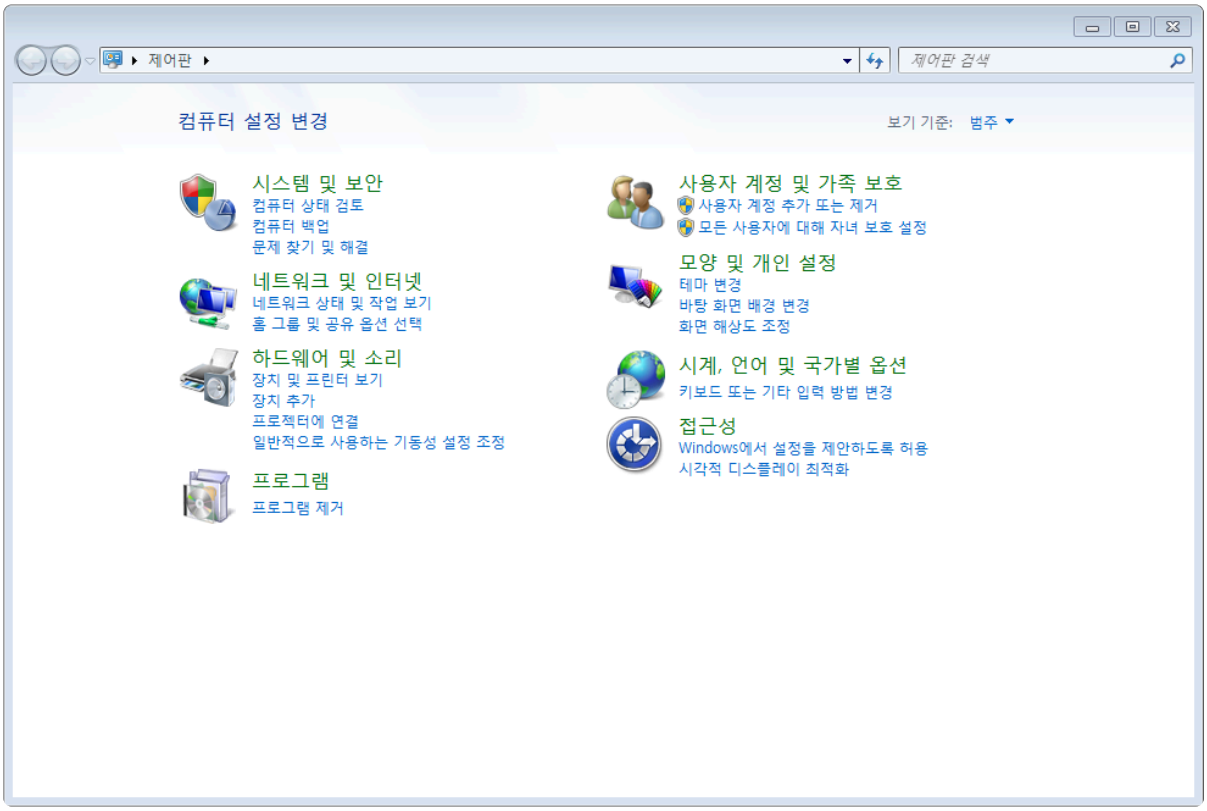

CodeScroll Controller Tester 3.0 설치
CodeScroll Controller Tester 설치 완료
CODESCROLL
설치 정보 수집
설치 준비
설치
설치 완료
CodeScroll Controller Tester 시작
< 뒤로(B)
마침(F)
취소

4. CONTROLLER TESTER 제거하기

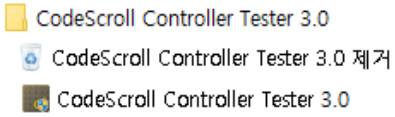
윈도우 기반 Controller Tester 제거하기

Controller Tester를 제거하는 방법은 세 가지가 있습니다.

1. [제어판] -> [프로그램 제거]를 이용한 삭제



2. [시작] -> [모든 프로그램] -> [CodeScroll Controller Tester 3.x] -> [CodeScroll Controller Tester 3.x 제거]를 이용한 삭제



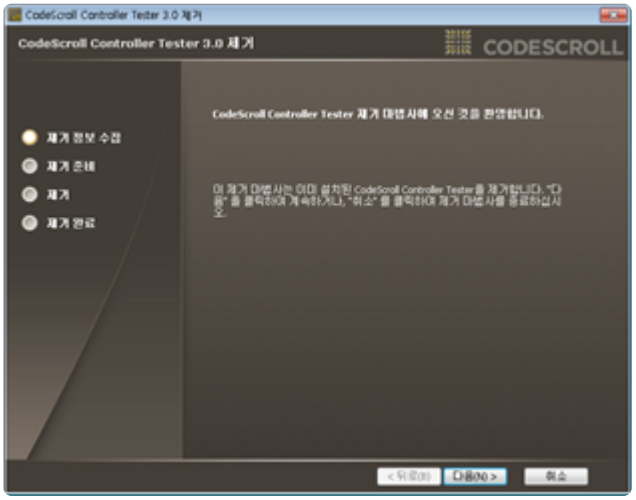
3. [설치 경로] -> [CodeScroll Controller Tester 3.x 제거]를 이용한 삭제

configuration	2017-09-20 오전...	파일 폴더	
features	2017-09-20 오전...	파일 폴더	
jre	2017-09-20 오전...	파일 폴더	
p2	2017-09-20 오전...	파일 폴더	
plugins	2017-09-20 오전...	파일 폴더	
artifacts.xml	2017-09-20 오전...	XML 문서	119KB
CodeScroll Controller Tester 3.0 제거	2017-09-20 오전...	바로 가기	2KB
CodeScroll.exe	2017-09-20 오전...	응용 프로그램	312KB
CodeScroll.exe.manifest	2017-08-18 오후...	MANIFEST 파일	2KB
CodeScroll.ini	2017-09-20 오전...	구성 설정	1KB
eclipsec.exe	2017-09-20 오전...	응용 프로그램	24KB
epl-v10.html	2017-08-18 오후...	Whale HTML Doc...	17KB
notice.html	2017-08-18 오후...	Whale HTML Doc...	9KB

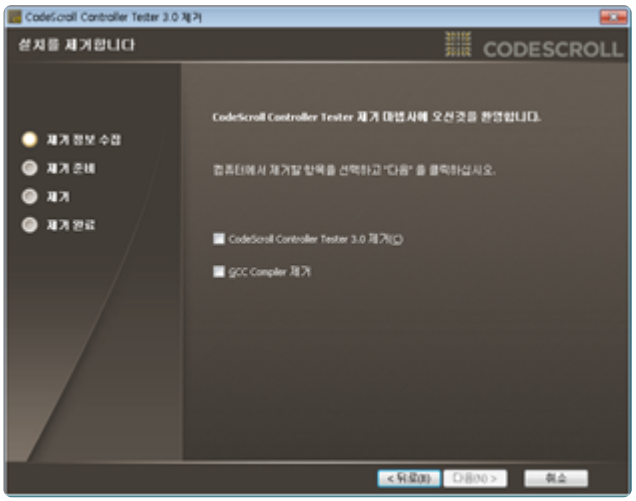
> “Controller Tester 제거하기”는Controller Tester의 동작에 필요한 파일들을 제거합니다. 사용자가 설정한 워크스페이스는 제거되지 않습니다.

[CodeScroll Controller Tester 3.x 제거] 수행 과정입니다.

1. [CodeScroll Controller Tester 3.x 제거]를 실행하면 제거에 필요한 정보를 수집합니다. [다음]을 클릭하여 진행합니다.

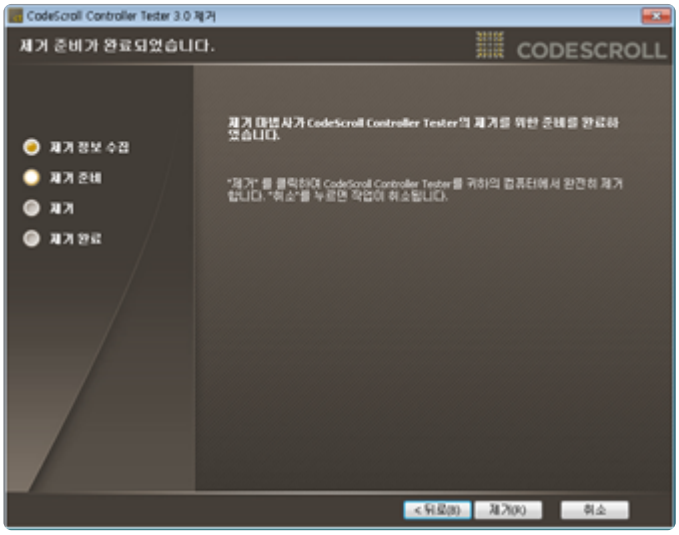


2. [CodeScroll Controller Tester 3.x 제거]와 [GCC Compiler 제거]를 체크하고 [다음]을 클릭합니다.

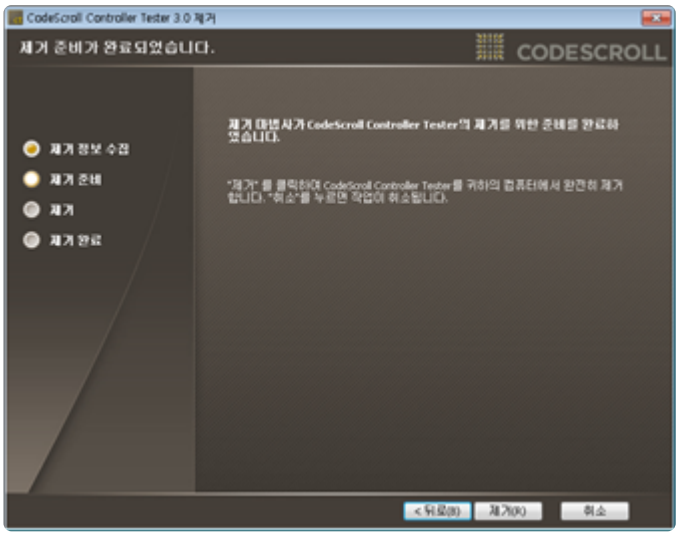


> [GCC Compiler 제거]는 [CodeScroll Controller Tester 3.x 제거]가 체크되어 있을 때 선택적으로 제거할 수 있습니다.
> GCC 컴파일러를 제거하지 않은 경우 Controller Tester 재설치 시 GCC Compiler는 설치할 수 없습니다.

3. 제거에 필요한 정보를 모두 수집하였습니다. [제거]를 클릭합니다.

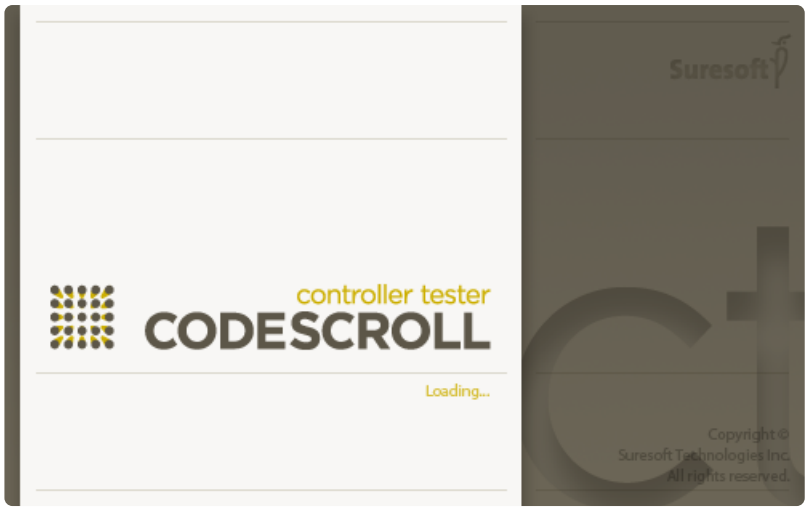


4. Controller Tester가 성공적으로 제거되면 [마침]을 클릭합니다.

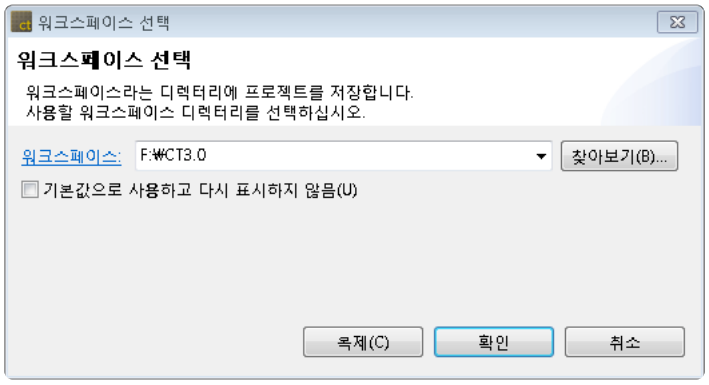

CodeScroll Controller Tester 3.0 제거
제거 준비가 완료되었습니다.
CODESCROLL
제거 정보 수집
제거 준비
제거
제거 완료
제거 마법사가 CodeScroll Controller Tester의 제거를 위한 준비를 완료하였습니다.
"제거"를 클릭하여 CodeScroll Controller Tester를 귀하의 컴퓨터에서 완전히 제거합니다. "취소"를 누르면 작업이 취소됩니다.
< 뒤로(B)
제거(R)
취소

5. CONTROLLER TESTER 실행

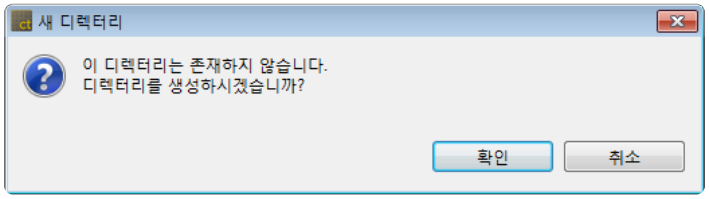
바탕 화면의 단축 아이콘 혹은 [시작] -> [모든 프로그램] -> [CodeScroll Controller Tester 3.x] -> [CodeScroll Controller Tester 3.x]을 선택합니다.



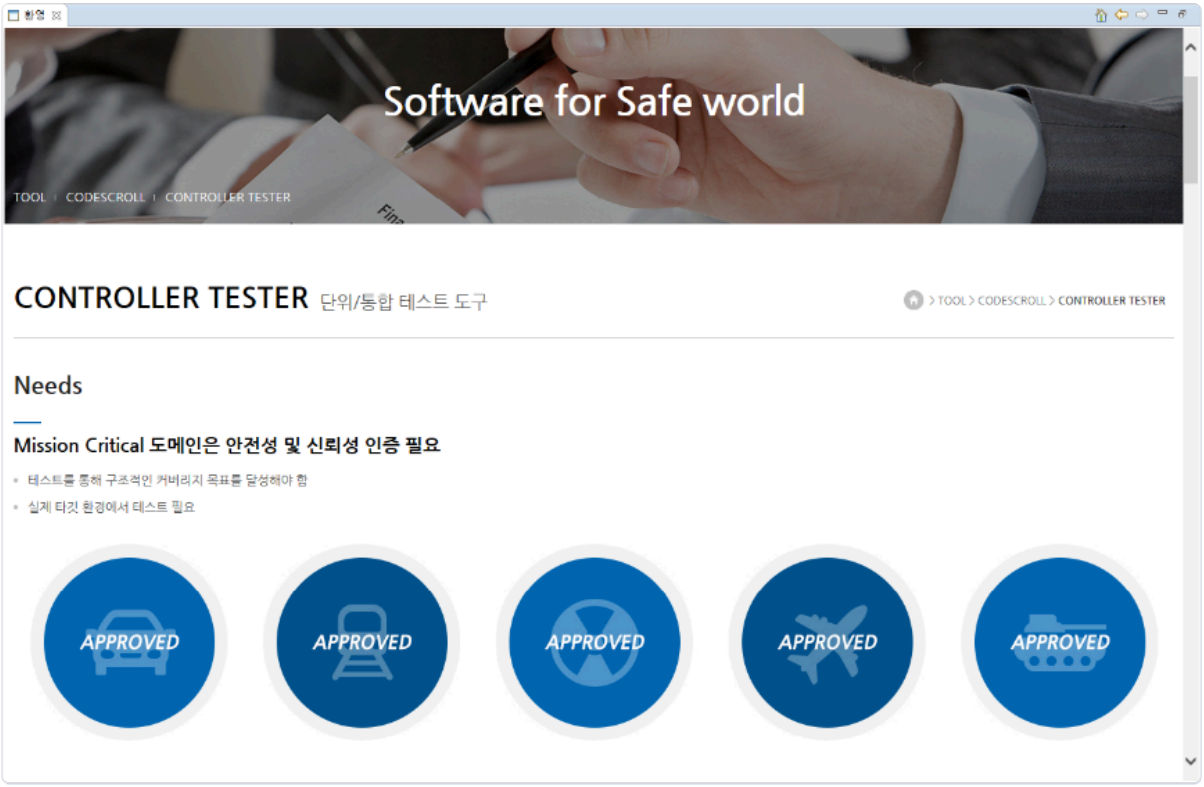
Controller Tester의 스플래시 화면에 이어 워크스페이스(작업 공간)를 선택합니다. 워크스페이스 경로는 이미 존재하는 디렉터리를 선택하거나 새로운 디렉터리를 입력할 수 있습니다. 다음에 실행할 때 워크스페이스를 다시 선택하지 않도록 지금 선택한 워크스페이스를 기본값으로 사용할 수도 있습니다. 선택을 마치면 [확인]버튼을 클릭합니다.



선택한 디렉터리가 존재하지 않을 경우, 사용자에게 디렉터리 생성여부를 확인합니다.



설정한 워크스페이스에 디렉터리를 생성하고 도구의 환영 페이지를 보여 줍니다. 환영 페이지는 워크스페이스를 새롭게 생성했을 경우에만 보여주며, 이전에 워크스페이스로 설정된 경로를 다시 워크스페이스로 선택했을 경우에는 보이지 않습니다.

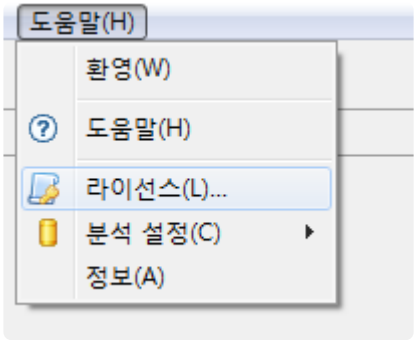


환영 페이지를 닫으면 제품 화면이 나타납니다.

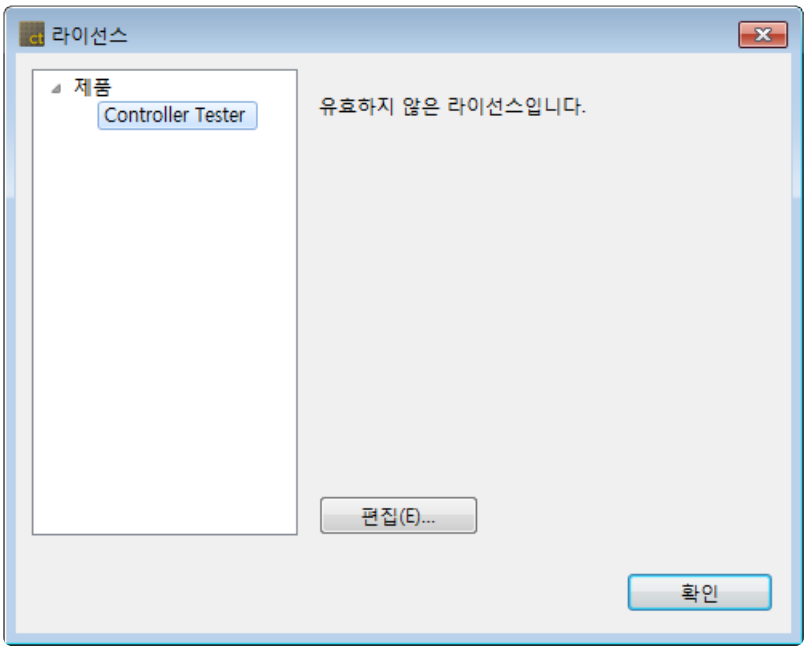
6. CONTROLLER TESTER 라이선스 설정

Controller Tester를 사용하려면 먼저 라이선스를 등록해야 합니다. 라이선스 등록 과정은 다음과 같습니다.

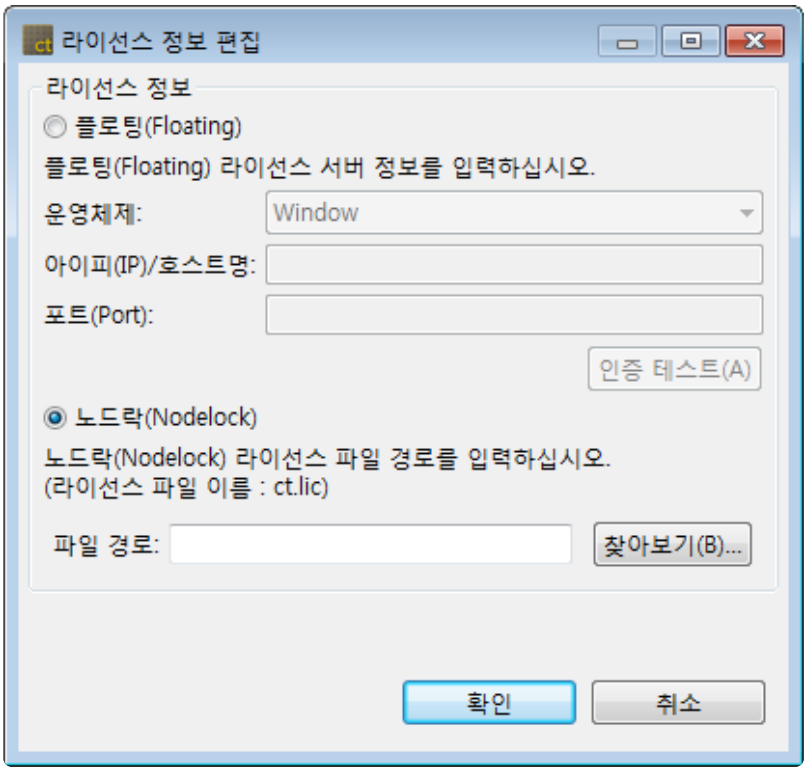
1. 메뉴의 [도움말] -> [라이선스]를 선택합니다.



2. [제품] -> [Controller Tester]를 선택하고 [편집]을 클릭합니다.



3. 라이선스 정보 편집에서 제공받은 라이선스를 입력하고 [확인] 버튼을 클릭하여 라이선스를 등록합니다.

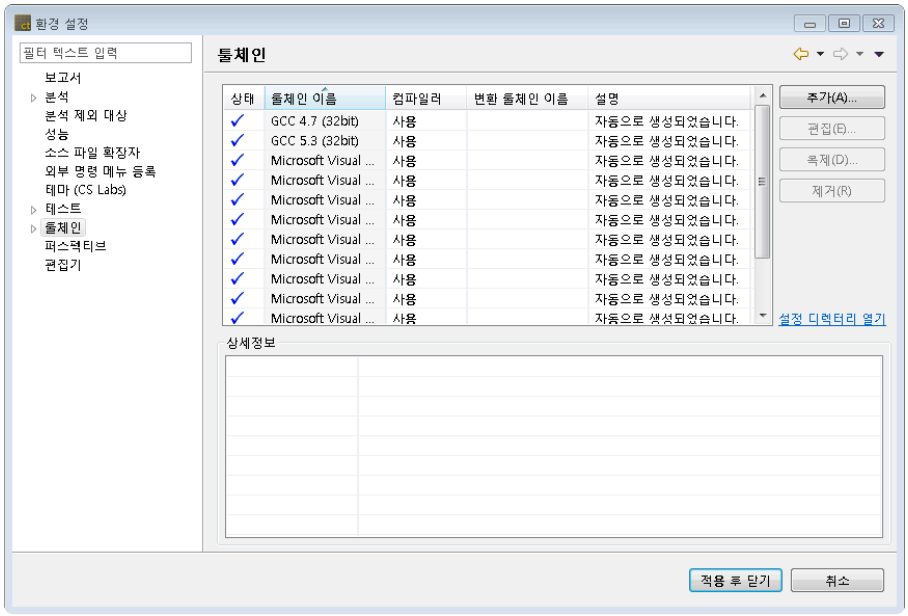


- 플로팅 방식: 라이선스 서버를 통해 등록합니다. 서버 정보(운영 체제, 아이피, 포트)를 입력하고, [인증 테스트] 버튼을 클릭합니다.
- 노드락 방식: 제공받은 라이선스 파일을 등록합니다.

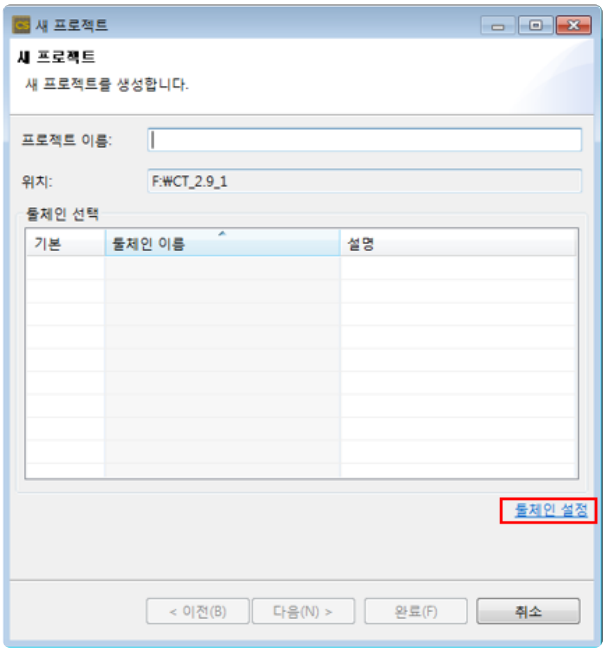
7. 툴체인(분석기) 설정

Controller Tester를 사용하여 테스트를 수행하기 위해서는 툴체인 설정이 필요로 합니다.
테스트 대상 소스에 대한 툴체인(컴파일러 정보)이 있어야 프로젝트를 생성할 수 있습니다.

툴체인은 [창] -> [환경 설정] -> [툴체인]에서 설정할 수 있습니다.



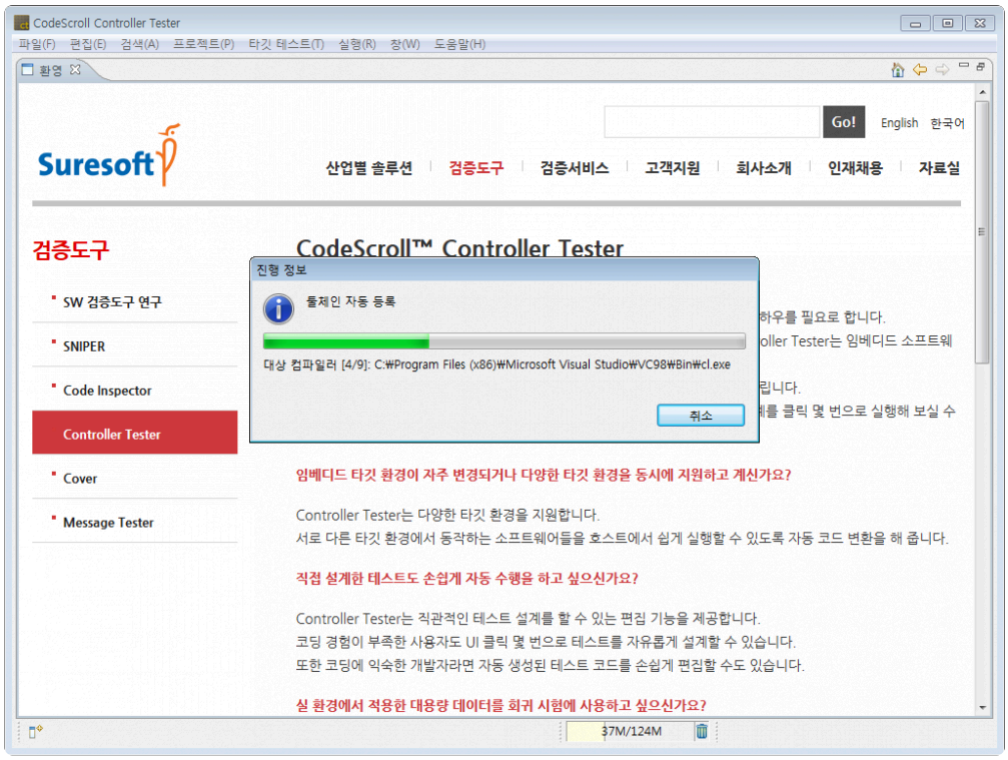
또한 소스 파일 C/C++ 프로젝트 생성 마법사 또는 임베디드 C/C++ 프로젝트 생성 마법사 내의 [툴체인 설정]을 통해서도 설정할 수 있습니다.



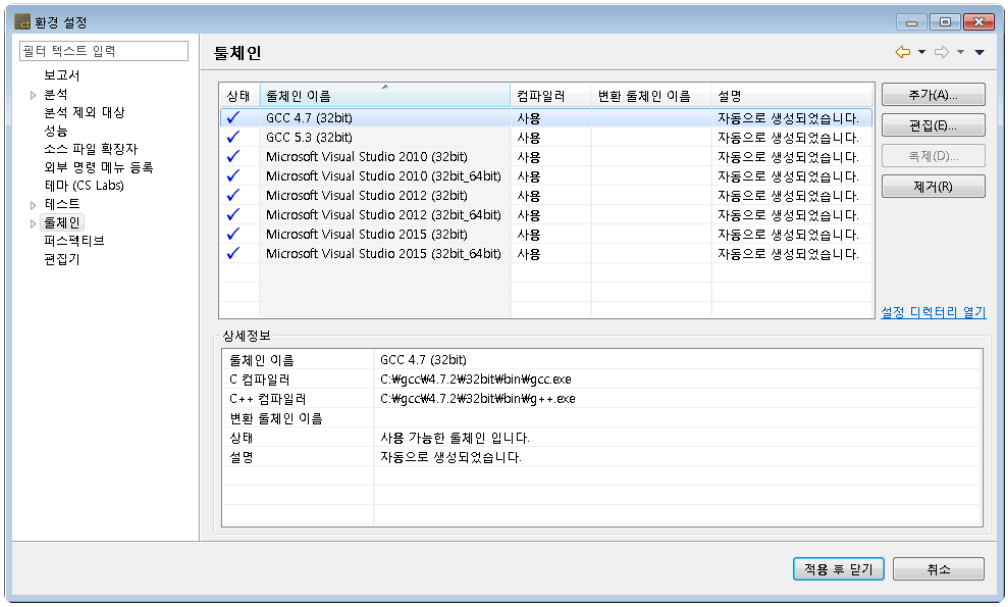
Controller Tester가 제공하는 툴체인 설정 관련 기능은 다음과 같습니다.

7.1. 툴체인 자동 등록

Controller Tester 실행 시, 사용자 PC에 설치된 Visual Studio 컴파일러 정보와 Controller Tester와 함께 설치된 GCC 컴파일러 정보를 추출하여 자동으로 툴체인을 등록합니다(등록된 툴체인은 다시 등록하지 않습니다).

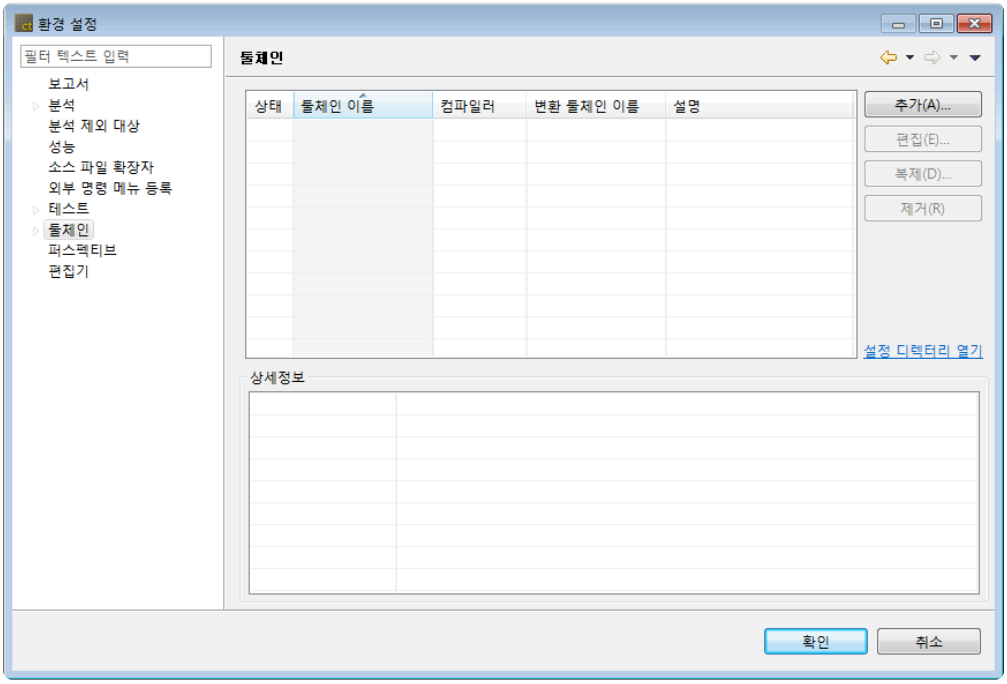


자동 등록된 툴체인은 [창] -> [환경 설정] -> [툴체인] 창에서 확인할 수 있으며, 설명에 “자동으로 생성되었습니다.” 라고 표시됩니다.

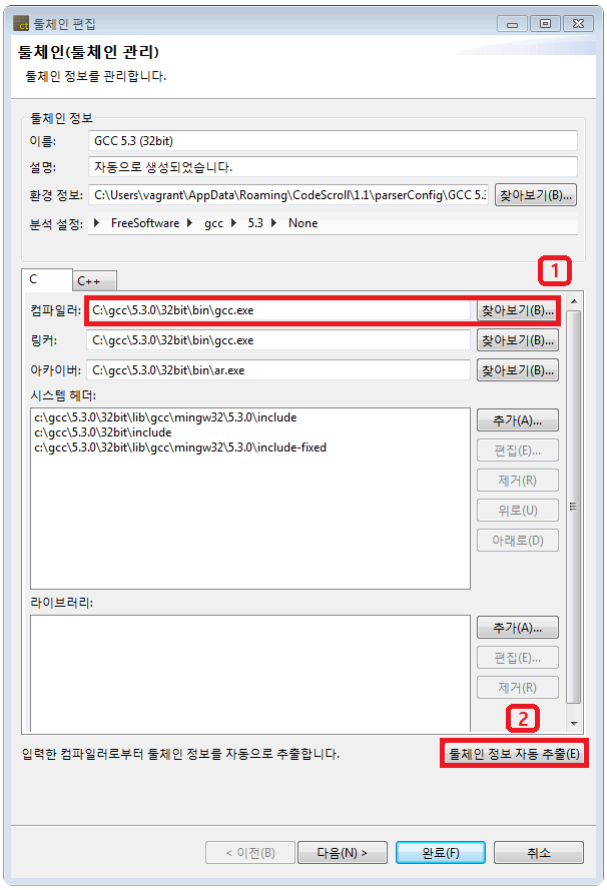


7.2. 툴체인 추가

1. [창] -> [환경 설정] -> [툴체인] 창에서 [추가] 버튼을 클릭합니다.

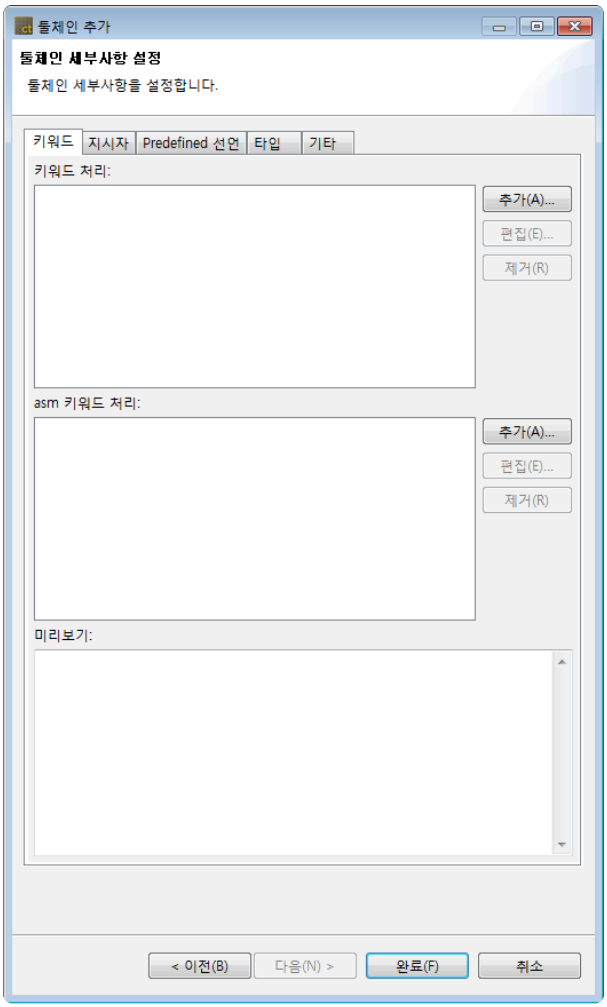


2. [툴체인 추가] 창에서 툴체인 정보를 입력합니다.



- [컴파일러]1 입력 후 [툴체인 정보 자동 추출]2 버튼을 클릭하면 입력한 컴파일러로부터 툴체인 정보를 자동으로 추출합니다.
- [컴파일러]1 란에는 컴파일러 경로를 찾아 입력합니다.
  - 예:
    - 비주얼 스튜디오 기반일 경우:
      C:\program Files (x86)\Microsoft Visual Studio 10.0\VS\bin\cl.exe
    - GCC 기반일 경우:
      C:\MinGW\bin\gcc.exe
    - Freeescale HC(s)12용 CodeWarrior 5.x 이상인 경우:
      C:\Program Files (x86)\Freescale\CWS12v5.1
      \Prog \chc12.exe

3. [이름]에 생성하려는 툴체인 이름을 입력합니다.
4. [설명]에 생서하려는 툴체인에 대한 설명을 입력합니다.
5. [환경 정보], [분석 설정]. [C/C++]의 입력정보들은 컴파일러로부터 정보를 추출하여 자동으로 설정되거나 사용자가 직접 입력할 수 있습니다.
6. 툴체인 정보 입력 후 [다음] 또는 [완료] 버튼을 클릭합니다.

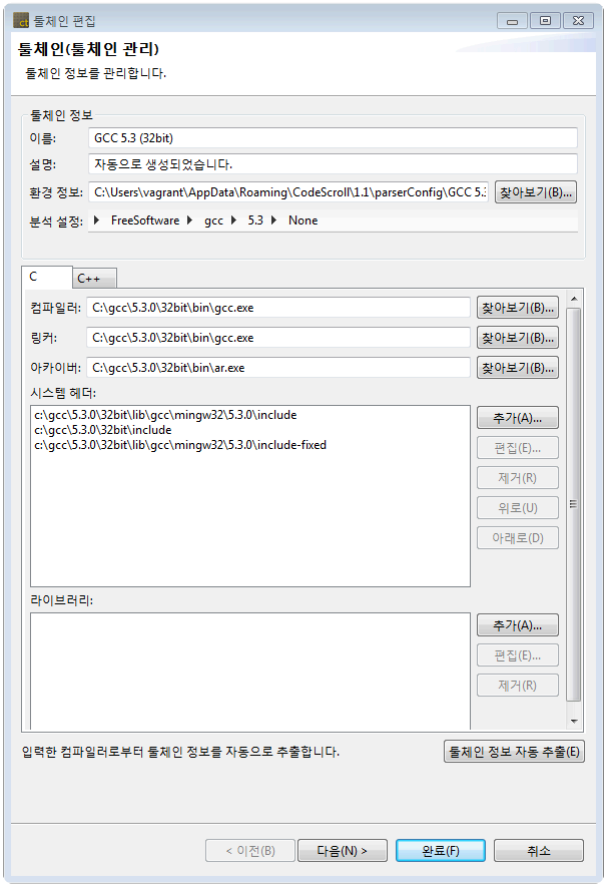


7. [다음] 버튼을 클릭하며 나오는 [툴체인 세부사항 설정] 창에서는 [분석 설정]에서 선택한 분석 설정에 관련된 세부 항목들을 보여줍니다. 각 항목들은 사용자가 변경할 수 있습니다. 변경된 사항은 [미리보기]를 통해 어떻게 적용되지 확인할 수 있습니다.

> 그 외 툴체인 세부사항에 대한 상세 설정 방법은 문제 해결 에 있는 기술지원 연락처로 문의하시기 바랍니다.

7.3. 툴체인 편집

1. 편집할 툴체인을 선택하고 [편집] 버튼을 클릭합니다.



2. 툴체인 정보를 편집합니다.
3. [완료] 버튼을 클릭합니다.

7.4. 툴체인 복제

1. 복제할 툴체인을 선택한 후 [복제] 버튼을 클릭합니다.
2. 변경할 툴체인 정보가 있으면 편집을 합니다.
3. [완료] 버튼을 클릭합니다.

7.5. 툴체인 삭제

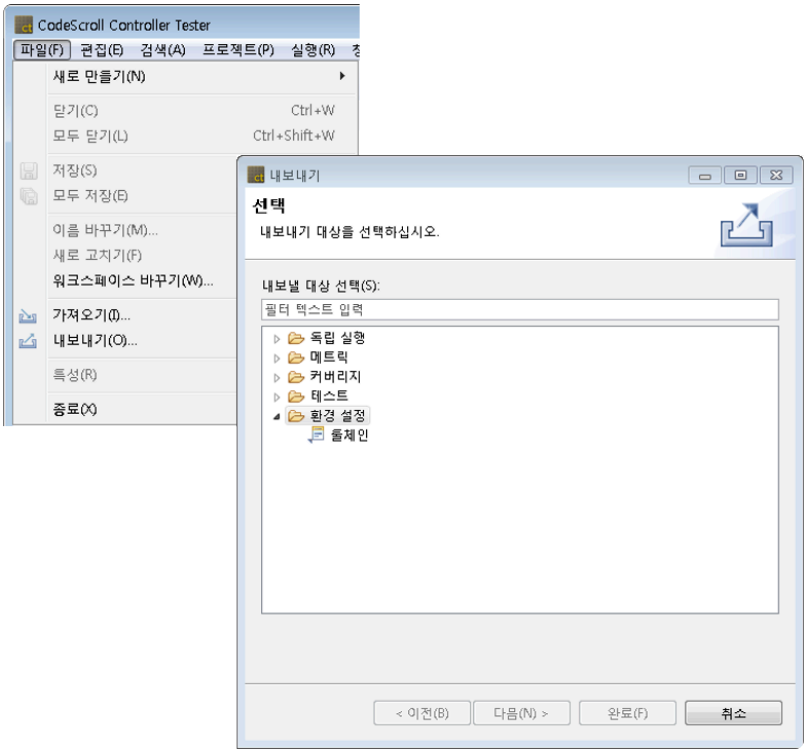
삭제할 툴체인을 선택한 후 [제거] 버튼을 클릭합니다.

> ! 삭제하려는 툴체인을 사용하는 프로젝트가 있을 경우에, 해당 프로젝트가 정상적으로 동작하지 않게 됩니다.

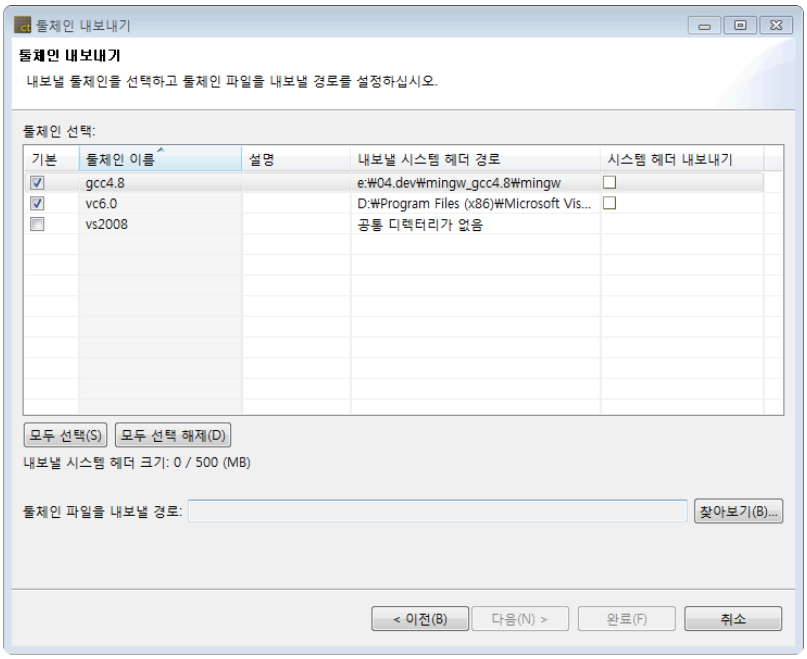
7.6. 툴체인 내보내기

'내보내기' 기능을 통해 툴체인 정보를 내보낼 수 있습니다.

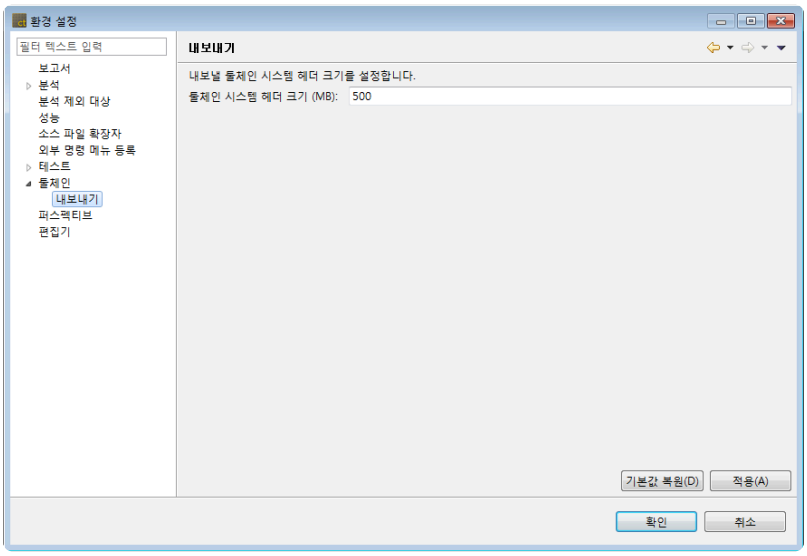
1. 메뉴의 [파일] -> [내보내기]를 클릭하여 [환경설정]의 [툴체인]을 선택합니다.



2. 내보낼 툴체인을 선택하고 [찾아보기] 버튼으로 내보낼 경로를 지정합니다. 툴체인 설정에서 시스템 헤더 파일을 설정하게 되면 [시스템 헤더 내보내기] 컬럼에 내보내기 여부를 선택할 수 있는 체크 박스가 나타납니다.



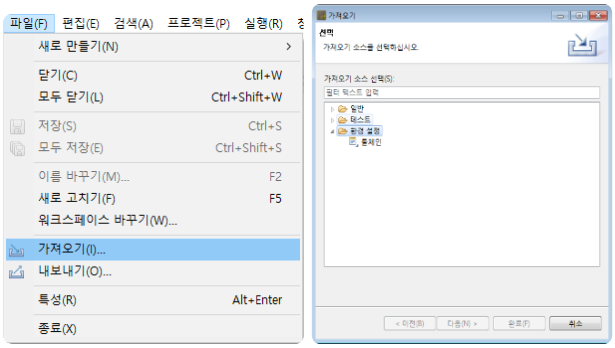
3. 메뉴의 [창] -> [환경 설정] -> [툴체인] -> [내보내기]를 클릭하면 내보낼 시스템 헤더 크기를 설정할 수 있습니다. 설정한 크기보다 큰 시스템 헤더는 내보낼 수 없습니다.



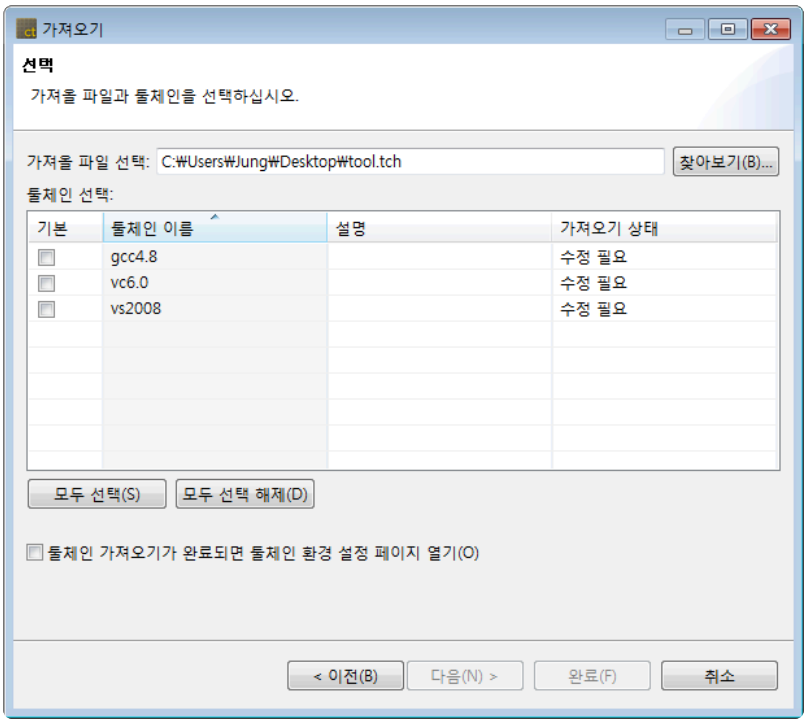
7.7. 툴체인 가져오기

'가져오기' 기능을 통해 내보낸 툴체인을 가져올 수 있습니다.

1. 메뉴의 [파일] -> [가져오기]를 선택하여 [환경 설정]의 [툴체인]을 클릭합니다.



2. 가져올 파일을 선택한 뒤에 툴체인을 선택합니다.



가져올 툴체인 이름이 기존 툴체인 이름과 중복될 경우, 가져오기 상태가 "수정 필요"가 됩니다. [다음] 버튼을 클릭하여 나오는 [문제해결]창에서 가져올 툴체인의 이름을 변경할 수 있습니다.

8. CONTROLLER TESTER 프로젝트 만들기

프로젝트 만들기 전 확인 사항

Controller Tester는 소스 코드를 실제 수행하여 오류를 검출하는 도구이므로, 테스트 대상 소스 코드를 수행 가능한 코드로 만들 수 있어야 합니다. 즉, 빌드가 가능한 소스 코드여야 정상적인 테스트를 진행할 수 있습니다.

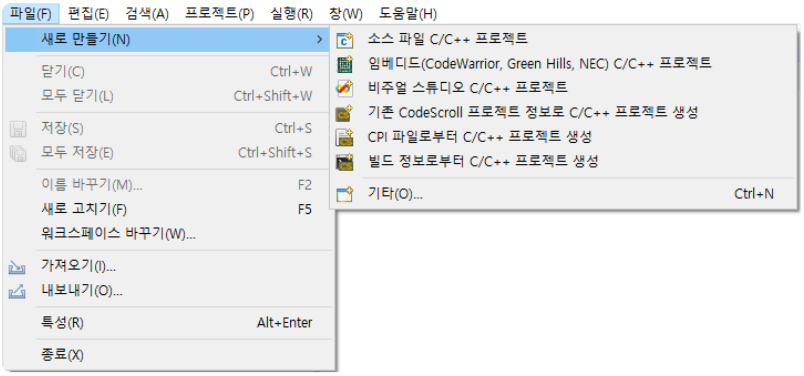
프로젝트를 생성하여 테스트를 진행하기 전에 대상 소프트웨어의 개발 환경을 고려한 '툴체인' 설정 과정이 필요합니다. 자세한 사항은 툴체인 설정 을 참고하시기 바랍니다.

빌드 과정에서 링크할 대상 객체가 존재하지 않을 경우에는 Controller Tester가 자동으로 스텁을 생성합니다. 하지만 자동 테스트 스텁 생성 기능은 테스트 수행에 제약 사항이 있을 수 있으므로, 적절한 테스트 전략을 사전에 준비하는 것이 좋습니다.

프로젝트 생성

Controller Tester 프로젝트를 생성하기 위해 [프로젝트 생성 마법사]를 실행합니다.

[파일] -> [새로 만들기] 또는 해당 단축 아이콘을 클릭합니다.



[프로젝트 생성 마법사]는 다음과 같은 종류의 프로젝트 생성을 제공합니다.

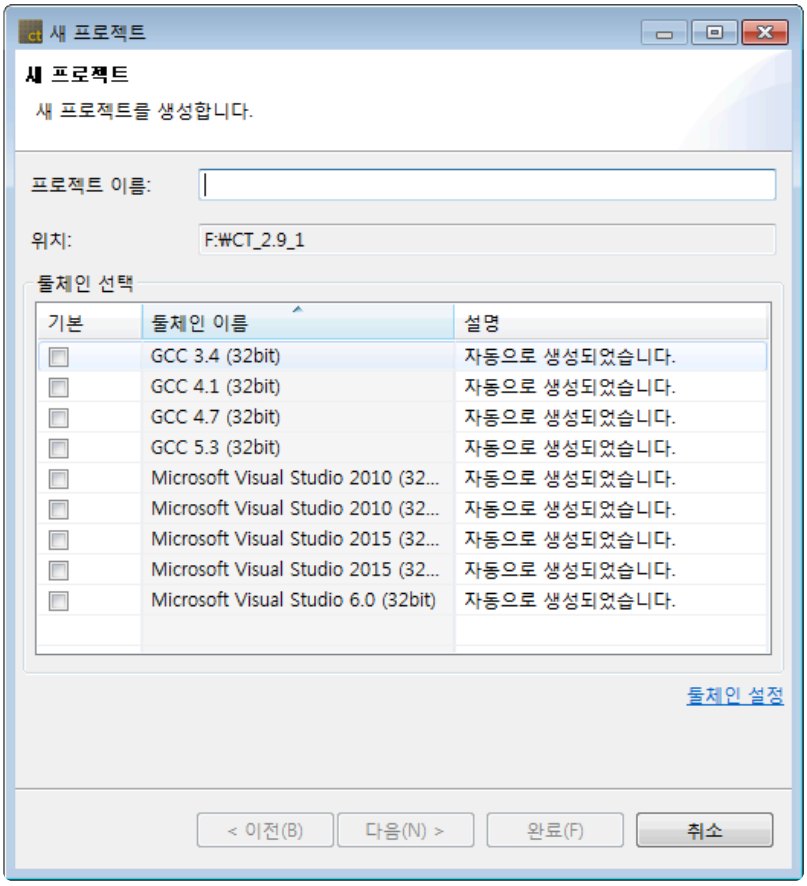
- 소스 파일 C/C++ 프로젝트
  소스 파일을 직접 선택하여 프로젝트 생성
- 비주얼 스튜디오 프로젝트
  비주얼 스튜디오 dsw, sln, vcxproj, vcproj 파일을 이용해서 프로젝트 생성
- 임베디드 C/C++ 프로젝트
  임베디드 프로젝트 파일을 이용해서 프로젝트 생성
- 기존 CodeScroll 프로젝트 정보로 C/C++ 프로젝트 생성
  기존에 생성된 CodeScroll 프로젝트를 이용해서 프로젝트 생성

- CPI 파일로부터 C/C++ 프로젝트 생성
  CPI 파일에 있는 프로젝트 생성 정보를 이용하여 프로젝트 생성
- 빌드 정보로부터 C/C++ 프로젝트 생성
  makefile 및 빌드 스크립트를 이용해서 프로젝트 생성

8.1. 소스 파일 C/C++ 프로젝트

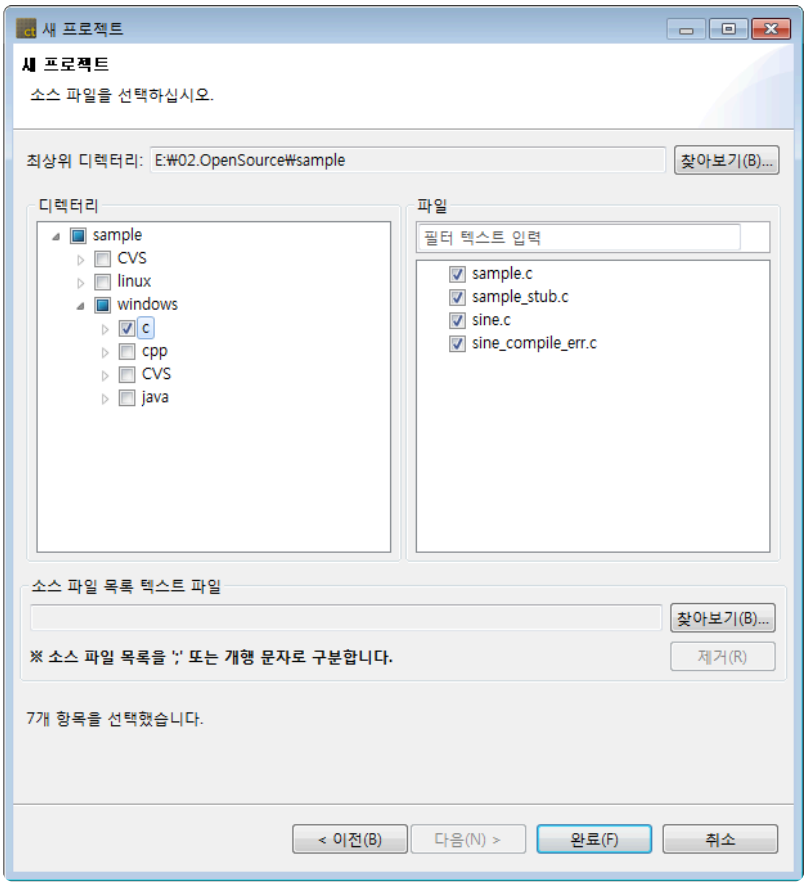
C/C++ 소스 코드 파일만으로 프로젝트를 생성할 경우 선택합니다. 소스 코드 빌드에 필요한 정보를 사용자가 직접 입력해야 합니다.

1. [프로젝트 이름]에 프로젝트 이름을 입력합니다.



2. [툴체인 선택]에서 해당 프로젝트에서 사용할 툴체인을 선택합니다. 생성된 툴체인이 없다면 [툴체인 설정]을 통해 툴체인을 생성합니다. 자세한 사항은 툴체인 설정 을 참고하시기 바랍니다.

3. [다음] 버튼을 클릭하여 다음 화면으로 이동합니다. [완료] 버튼을 클릭 하였을 경우에는 소스 파일이 포함되지 않는 빈 프로젝트가 생성됩니다.



- 소스 파일을 직접 선택하거나, 파일 경로가 쓰인 텍스트 파일을 통해 선택할 수 있습니다.

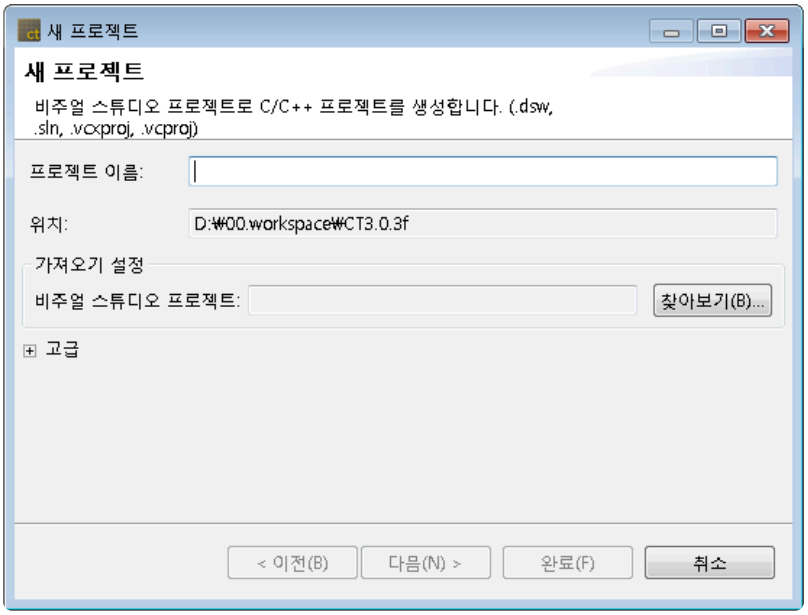
4. 소스 파일을 직접 선택
   a. [찾아보기...]를 클릭하여 아래 디렉터리 화면에 보일 [최상위 디렉터리]를 지정합니다.
      - 최상위 디렉터리는 특별한 경우를 제외하고, 선택하려는 소스 파일을 담고 있는 디렉터리보다 한 단계 높은 수준의 디렉터리를 지정합니다.
   b. 왼쪽에 보이는 [디렉터리] 화면에서 프로젝트 생성에 사용할 소스 파일이 있는 디렉터리를 선택합니다.
   c. 선택한 디렉터리에 있는 파일이 오른쪽에 보이는 [파일] 화면에 나타납니다. 추가할 파일을 체크합니다.

5. 파일 경로가 쓰인 텍스트 파일을 통해 선택
   a. 아래에 보이는 [소스 파일 목록 텍스트 파일] 화면에서 [찾아보기...]를 클릭하여 소스 파일 목록(파일 절대 경로)이 기록된 텍스트 파일을 선택합니다.
   b. [제거] 버튼을 클릭하면 지정한 텍스트 파일과 파일을 통해 선택된 소스 파일을 선택 해제할 수 있습니다.

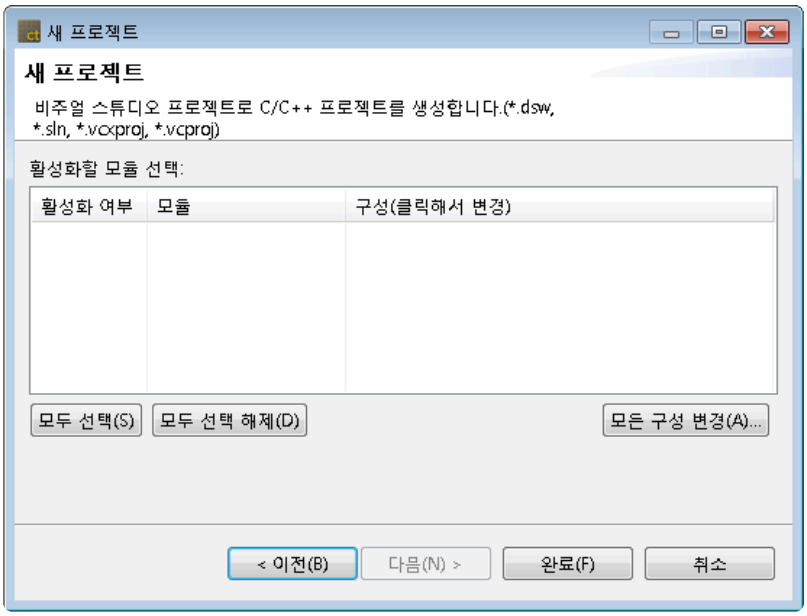
6. 모든 설정이 완료된 후 [완료] 버튼을 클릭하여 프로젝트를 생성합니다.

8.2. 비주얼 스튜디오 프로젝트

마이크로소프트 비주얼 스튜디오 프로젝트 파일(.dsw, .sln, .vcxproj, .vcproj)로 프로젝트를 생성할 때 선택합니다. 특별한 경우를 제외하고, 빌드에 필요한 정보는 자동으로 입력되므로 사용자가 별도로 지정할 내용이 없습니다.



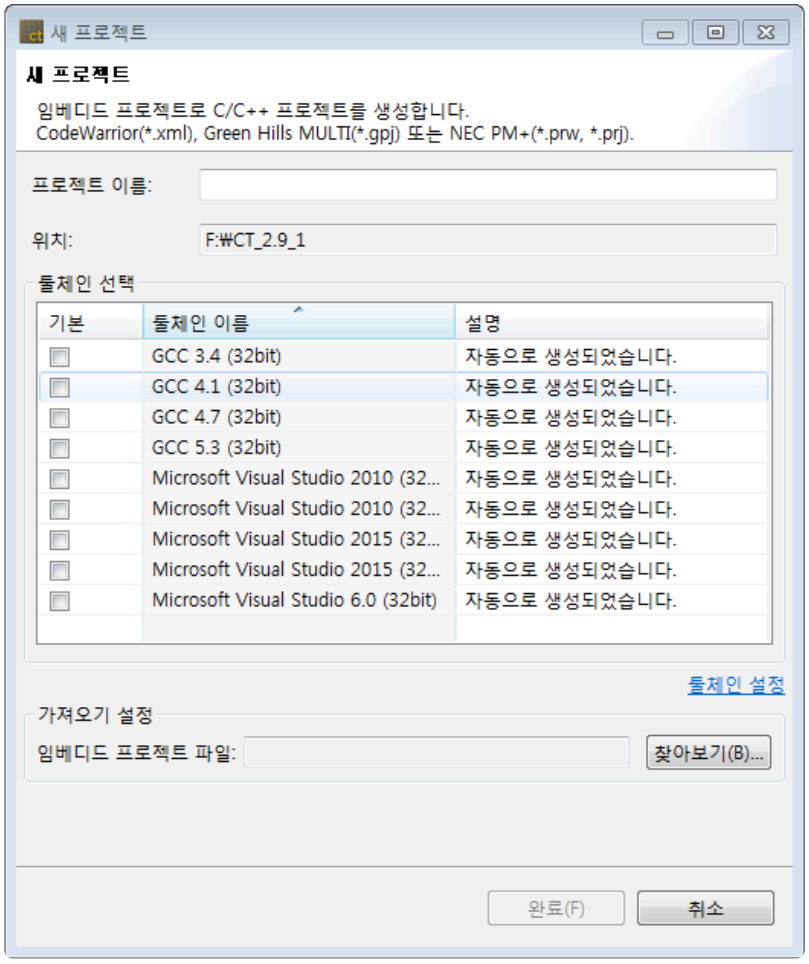
1. [프로젝트 이름]에 프로젝트 이름을 입력합니다.
2. [가져오기 설정] -> [비주얼 스튜디오 프로젝트] 에 가져오기 위한 비주얼 스튜디오 프로젝트 파일을 지정합니다.
3. [고급] -> [환경 설정 스크립트 파일]은 등록하는 툴체인이 사용하는 컴파일러가 특정 환경 변수를 사용하는 경우, 컴파일러를 호출하기 전에 반드시 설정해야 하는 값들이 저장된 스크립트 파일을 설정합니다.
4. [다음]을 클릭하여 다음 화면으로 이동합니다. [완료]를 클릭하면 가져온 비주얼 스튜디오 프로젝트에 대해 모듈 설정은 임의적으로 선택됩니다.



5. 비주얼 스튜디오 프로젝트에 포함된 모듈들 중에서 활성화 시킬 모듈을 선택합니다.
6. 각 모듈들이 갖는 구성을 선택합니다.
    - [모든 구성 변경]은 모듈에 대해 구성을 일괄적으로 변경할 수 있습니다.
7. 모든 설정이 완료된 후 [완료]버튼을 클릭하여 프로젝트를 생성합니다.

8.3. 임베디드 C/C++ 프로젝트

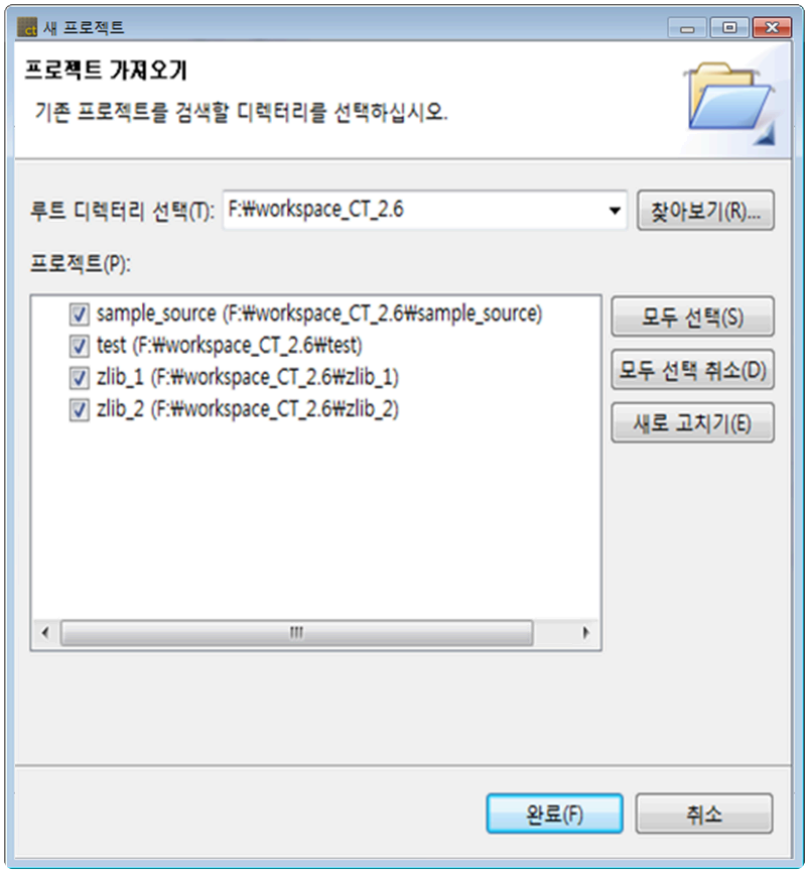
임베디드 프로젝트 파일(.xml, .gpj, .prw, .prj)로 프로젝트를 생성할 때 선택합니다. 특별한 경우를 제외하고, 빌드에 필요한 정보는 자동으로 입력되므로 사용자가 별도로 지정할 내용이 없습니다.



1. [프로젝트 이름]에 프로젝트 이름을 입력합니다.
2. 생성할 임베디드 프로젝트에 맞는 툴체인을 선택합니다.
3. [가져오기 설정] -> [임베디드 프로젝트 파일]에서 시험 대상 임베디드 프로젝트 파일을 지정합니다.
4. 모든 설정을 완료하고 [완료]를 클릭하여 프로젝트를 생성합니다.

8.4. 기존 CodeScroll 프로젝트 정보로 C/C++ 프로젝트 생성

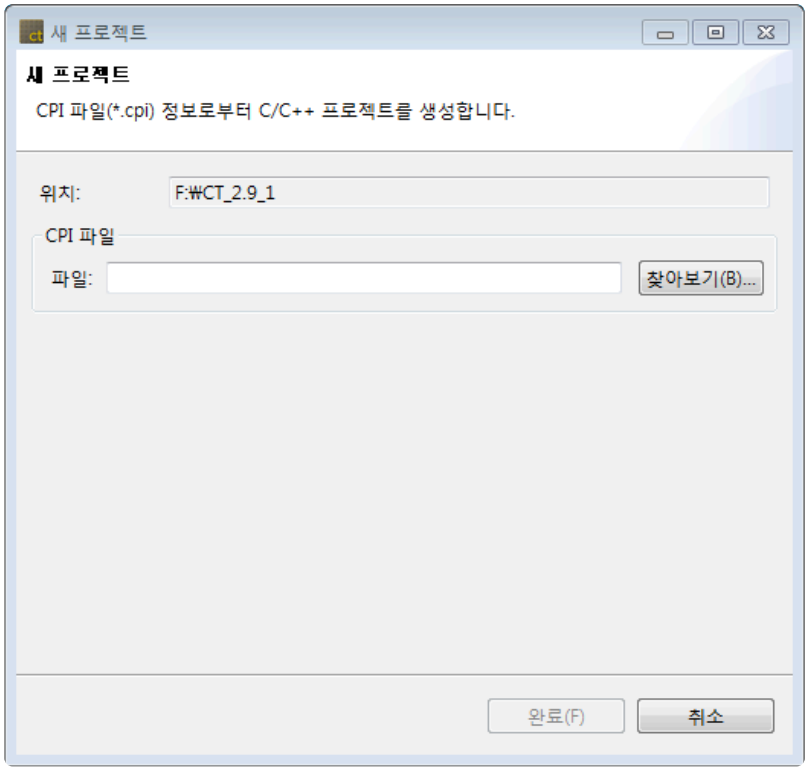
기존 CodeScroll 프로젝트를 임포트하여 해당 정보로 새로운 프로젝트를 생성합니다. 기존 프로젝트에 대한 여러 프로젝트 설정(소스 파일, 분석 제외/포함, 컴파일 플래그)을 그대로 구성하여 같은 정보의 새 프로젝트를 쉽게 생성할 수 있습니다.



1. [찾아보기...] 버튼을 이용하여 CodeScroll 작업공간이나 CodeScroll 프로젝트 경로를 선택합니다.
2. 선택한 경로에 존재하는 프로젝트가 [프로젝트] 목록에 표시됩니다.
    - CodeScroll 작업공간을 선택했을 경우, 하위의 모든 프로젝트들이 표시됩니다.
3. [완료]를 클릭하여 프로젝트를 생성합니다.

8.5. CPI 파일로부터 C/C++ 프로젝트 생성

CPI 파일로부터 프로젝트를 생성할 경우 선택합니다. CPI 파일은 커맨드라인 인터페이스에서 프로젝트 생성을 위한 정보 파일이며, CPI 파일은 사용자가 직접 작성을 해야 합니다.
{설치 경로}\plugins\com.codescroll.gp.cli_x.x.x.x\cpi 폴더에 있는 템플릿 파일로 작성합니다.



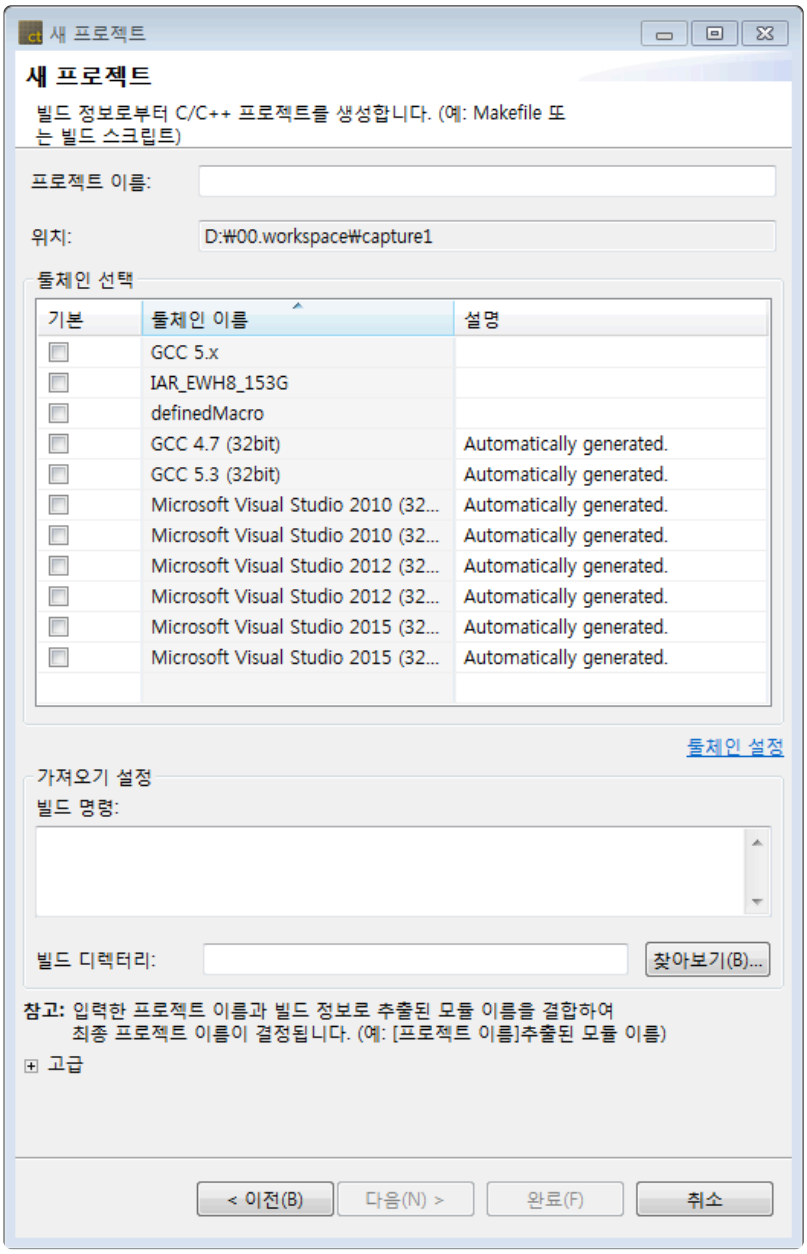
1. [찾아보기...] 버튼을 이용하여 CPI 파일을 선택합니다.
2. [완료] 버튼을 클릭하여 프로젝트를 생성합니다.

테스트 네비게이터에 CPI 파일을 끌어서 놓아 프로젝트를 생성할 수도 있습니다.

8.6. 빌드 정보로부터 C/C++ 프로젝트 생성

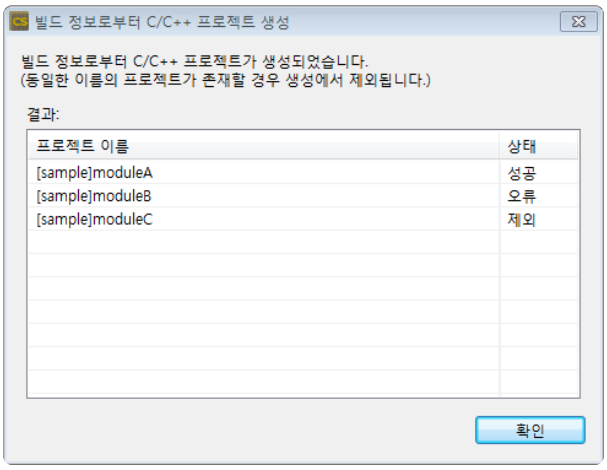
빌드 정보로부터 C/C++ 프로젝트를 생성할 때 선택합니다. 빌드 정보로 추출된 모듈의 수만큼 프로젝트가 생성됩니다.

1. [프로젝트 이름]에 프로젝트 이름을 입력합니다. 입력한 프로젝트의 이름과 빌드 정보로 추출된 모듈 이름을 결합하여 최종 프로젝트 이름이 결정됩니다. (예: [프로젝트 이름]추출된 모듈 이름)



2. 빌드에 사용하는 컴파일러와 동일한 툴체인을 선택합니다.
3. [가져오기 설정] -> [빌드 명령]에 빌드 시 필요한 명령어를 입력합니다.
   - 예) make all / make clean
4. [가져오기 설정] -> [빌드 디렉터리]에서 makefile이 저장되어 있는 경로를 지정해 줍니다.

5. [고급] -> [환경 설정 스크립트 파일]은 등록하는 툴체인이 사용하는 컴파일러가 특정 환경 변수를 사용하거나 환경 설정 스크립트 파일을 필요로 하는 프로젝트일 경우, 컴파일러를 호출하기 전에 반드시 설정해야 하는 값들이 저장된 스크립트 파일을 지정합니다.
6. 모든 설정을 완료하고 [완료]를 클릭하여 프로젝트를 생성합니다.
7. 빌드 후 프로젝트 생성이 완료되면 결과 화면이 나타납니다.



- 성공: 프로젝트 생성이 완료.
- 오류: 프로젝트 생성 중 오류로 인해 생성하지 못한 경우.
- 제외: 워크스페이스 내에 같은 프로젝트 이름이 존재하는 경우.

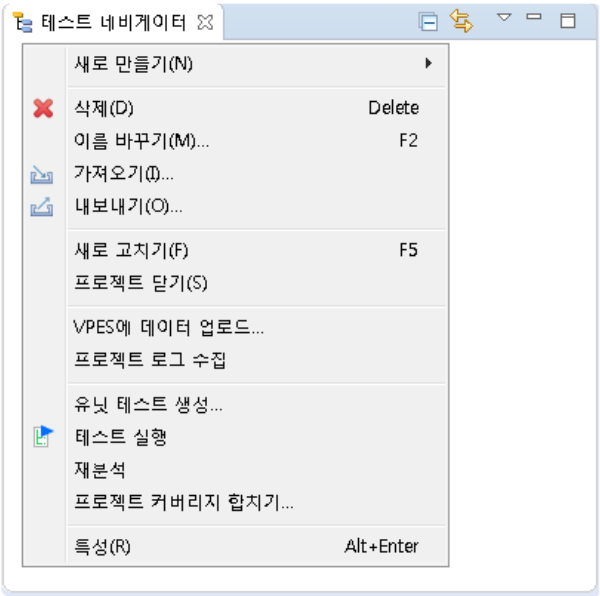
9. 테스트 생성

소스 코드의 함수를 테스트하기 위해서 테스트 데이터와 테스트 코드를 생성합니다. 각 함수 단위로 개별적인 테스트를 만들 수 있습니다.

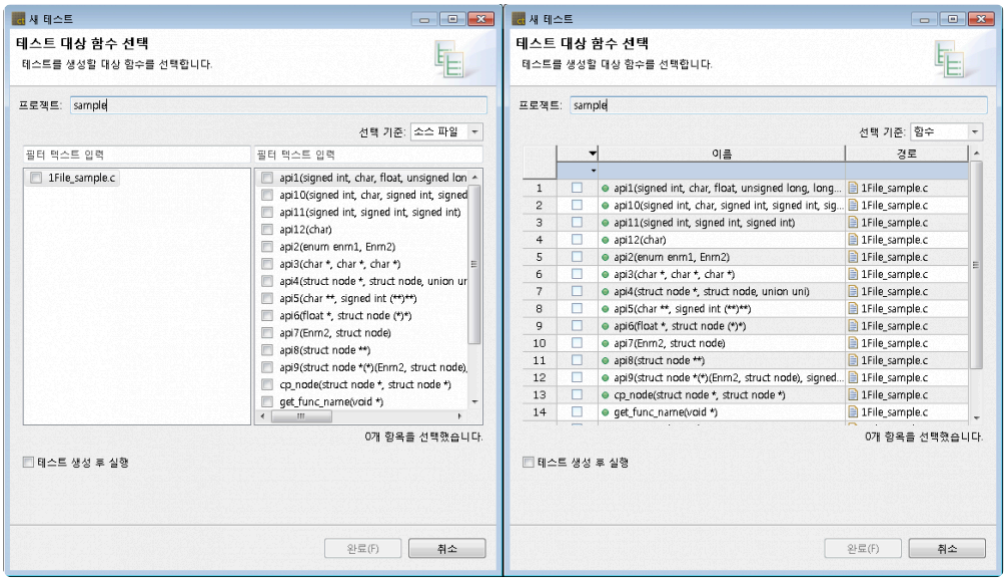
유닛 테스트 생성

사용자 소스의 각 함수에 대해 개별적으로 테스트 데이터와 테스트 코드를 생성합니다.

1. 유닛 테스트를 생성할 프로젝트 또는 프로젝트에 포함된 모듈을 선택하고 우 클릭 후 [유닛 테스트 생성] 메뉴를 클릭합니다.



2. 분석이 끝나면 [테스트 대상 함수 선택]에서 유닛 테스트를 수행할 대상 함수를 선택합니다.



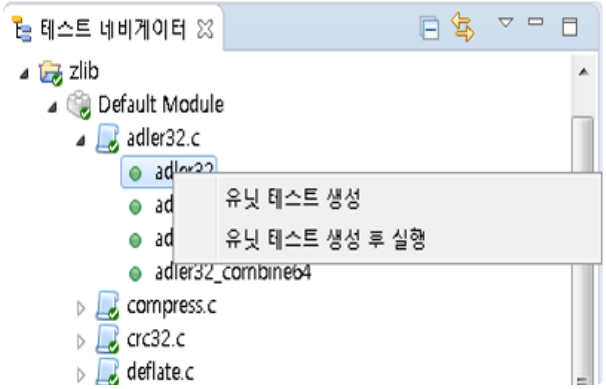
a. [필터 텍스트 입력]을 이용해서 [테스트 대상 함수]에 대해 필터링할 수 있습니다.
b. [선택 기준] 콤보 박스를 이용해서 소스 파일 또는 함수 기준으로 [테스트 대상 함수]를 선택할 수 있습니다.
c. 우 클릭을 이용하여 손쉽게 [테스트 대상 함수]를 전체 선택 및 해제할 수 있습니다.
d. [테스트 생성 후 실행]이 선택되면 테스트 데이터 준비 후 테스트를 실행합니다.

3. [완료]를 클릭하여 유닛 테스트를 생성합니다.

유닛 테스트 생성 후 실행

[유닛 테스트 생성] 기능을 사용하지 않고, 유닛 테스트 생성 및 테스트 수행을 한 번에 진행할 수 있는 [유닛 테스트 생성 후 실행] 기능을 제공합니다.

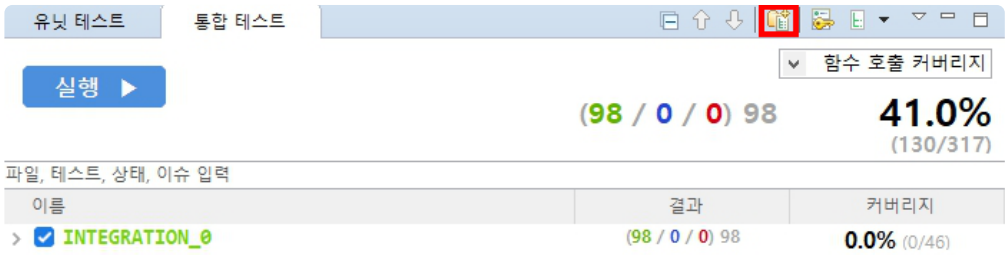
1. 테스트 네비게이터에서 테스트 대상 함수를 선택한 후 우 클릭합니다.



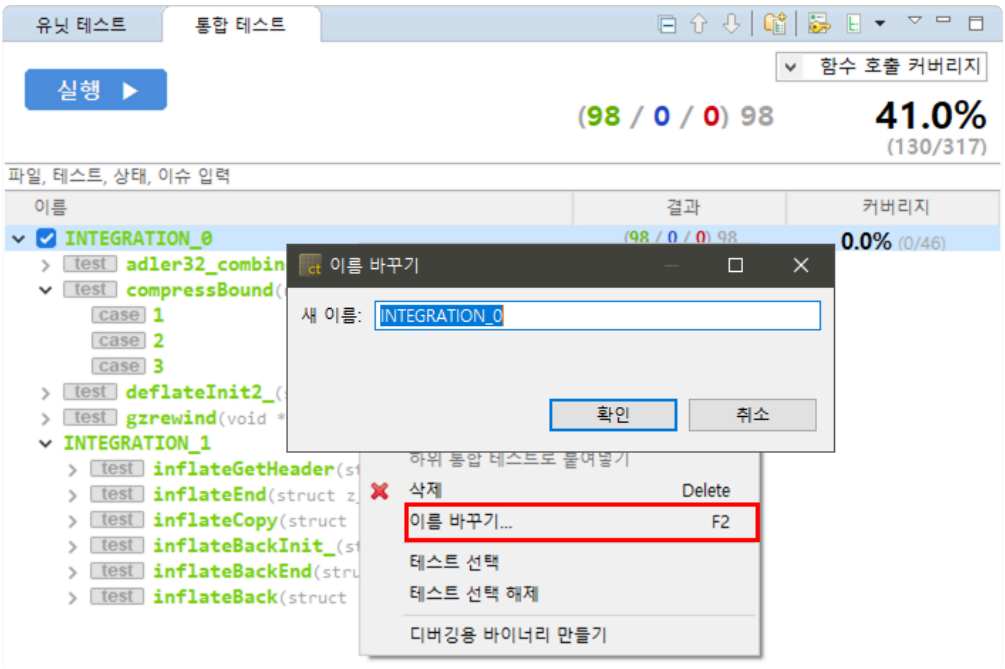
2. [유닛 테스트 생성 후 실행] 메뉴를 선택하면 대상 함수에 대한 테스트를 생성한 후, 테스트를 수행합니다.

통합 테스트 생성

툴바 메뉴에서 통합 테스트를 생성할 수 있습니다.

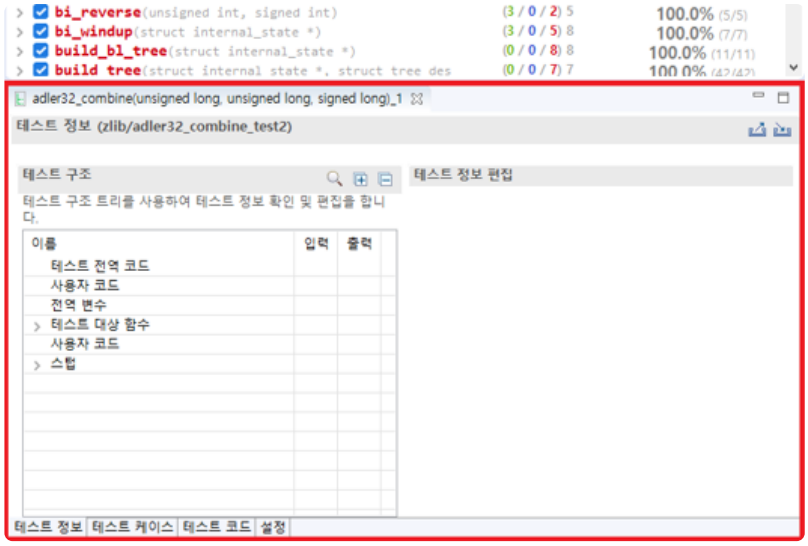


통합 테스트 이름은 자동으로 부여되며 [이름 바꾸기] 컨텍스트 메뉴를 이용해 변경할 수 있습니다.


유닛 테스트
통합 테스트
실행
함수 호출 커버리지
(98 / 0 / 0) 98
41.0%
(130/317)
파일, 테스트, 상태, 이슈 입력
이름
결과
커버리지
INTEGRATION_0
0.0% (0/46)
adler32_combin
compressBound(
case 1
case 2
case 3
deflateInit2_(
gzrewind(void *
INTEGRATION_1
inflateGetHeader(st
inflateEnd(struct z_
inflateCopy(struct
inflateBackInit_(st
inflateBackEnd(stru
inflateBack(struct
이름 바꾸기
새 이름:
INTEGRATION_0
확인
취소
삭제 Delete
이름 바꾸기... F2
테스트 선택
테스트 선택 해제
디버깅용 바이너리 만들기

10. 테스트 편집기

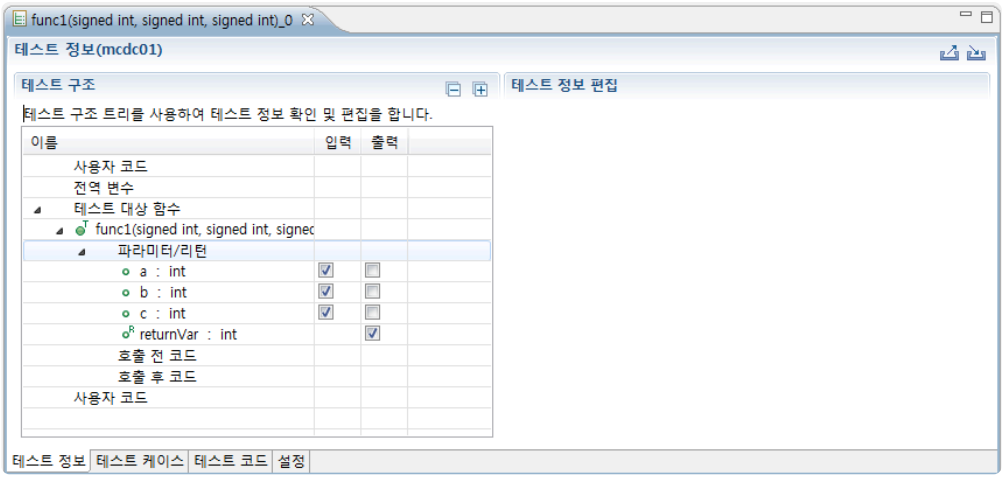
테스트 편집기는 유닛/통합 테스트 뷰 하단에 위치하며, 각 테스트 뷰의 테스트나 테스트 케이스를 더블 클릭하여 열 수 있습니다.



편집기의 위치는 사용자가 변경할 수 있으며, 독립된 창에서도 열 수 있습니다.

10.1. 테스트 정보

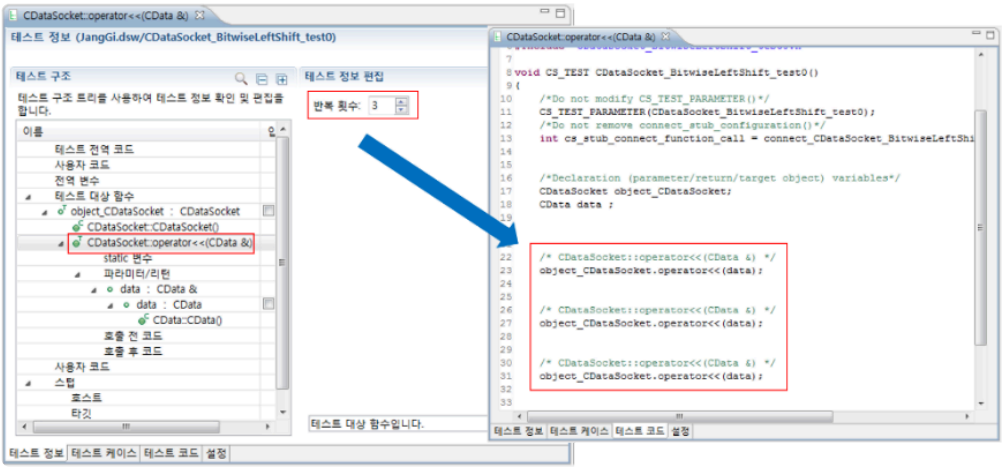
테스트 정보 탭은 테스트 코드를 테스트 구조 트리로 나타내어 사용자가 테스트 코드를 쉽게 수정할 수 있도록 제공하는 기능입니다.



- 사용자 코드를 테스트 코드에 삽입할 수 있습니다.
- 테스트 대상 함수와 관련이 있는 전역변수를 제어할 수 있습니다.
- 함수의 파라미터 및 리턴 값에 대한 입출력을 제어할 수 있습니다.
- 테스트 대상 함수를 호출하기 직전에 수행할 코드를 입력할 수 있습니다.

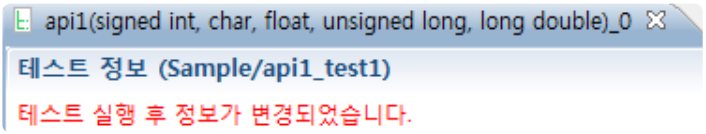
함수 호출 반복

테스트 대상 함수의 반복 횟수를 지정할 수 있습니다.



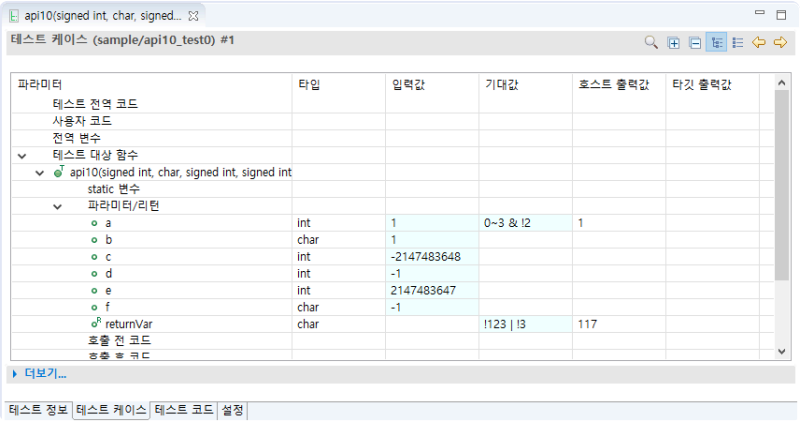
변경 내용 감지

이미 실행이 완료된 테스트 정보를 수정한 뒤 저장하면 상단에 “테스트 실행 후 정보가 변경되었습니다.”라고 표시됩니다.



10.2. 테스트 케이스

테스트 케이스는 테스트 대상 함수의 입력과 실행 결과의 예상값을 편집하는 기능을 제공합니다.



- 테스트 구조 트리에서 입력으로 지정한 변수에 대한 입력값을 입력할 수 있습니다.
    - **입력값이 없는 경우, 해당 변수의 값은 'undefined value'이며, 도구에서는 사용자 편의를 위해 입력 사용으로 체크된 변수 중 아무 값도 입력되지 않은 변수는 0(문자열일 경우 공백)으로 초기화합니다.**
- [트리로 보기], [테이블로 보기] 툴바 메뉴로 테스트 구조를 테이블과 트리로 표현할 수 있습니다.
- [테스트 케이스 일괄 적용] 컨텍스트 메뉴로 해당 테스트 케이스의 입력값과 기대값을 같은 테스트 내 다른 테스트 케이스에 한 번에 적용할 수 있습니다.
- 테스트 구조 트리에서 출력으로 지정한 변수는 테스트를 수행하고 난 후, 결과값을 확인할 수 있습니다.
- 편집 가능한 셀은 옅은 하늘 색으로 표시됩니다.
- 기대값과 호스트/타깃 출력값이 다른 경우 파란색으로 표시됩니다.
- 기대값에 ~(범위)와 논리 연산자 !(not), &(and), |(or)를 사용할 수 있습니다.
- 다음과 같은 단축키를 제공합니다.
    - 행 이동: 방향키(↑,↓)
    - 편집 가능한 셀 포커스: enter
    - 편집 가능한 셀 포커스 아웃: esc
    - 열(편집 가능한 셀)이동: tab(오른쪽), shift + tab(왼쪽)

변경 내용 감지

이미 실행이 완료된 테스트 케이스를 수정하면 상단에 "테스트 실행 후 정보가 변경되었습니다."라고 표시됩니다.

10.3. 테스트 코드

테스트 코드는 테스트 정보 창의 테스트 코드 탭을 클릭하여 확인할 수 있습니다.

func1(signed int, signed int, signed int)_0

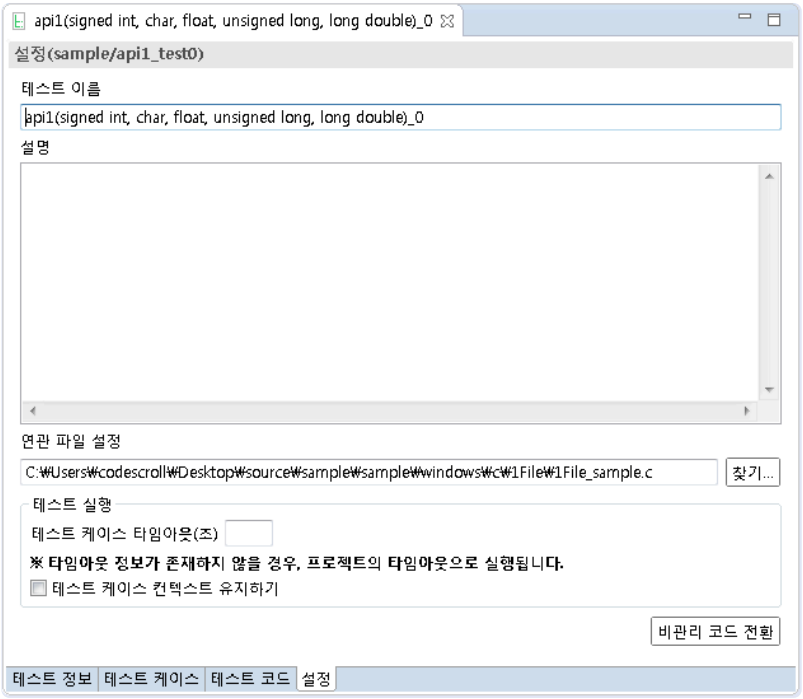
```
1 /* func1_test0.c */
2 #ifndef __func1_test0__
3 #define __func1_test0__
4
5 void CS_TEST func1_test0()
6 {
7     /*Do not modify CS_TEST_PARAMETER()*/
8     CS_TEST_PARAMETER(func1_test0);
9
10    /*Declaration (parameter/return/target object) variables*/
11    int a  = 0;
12    int b  = 0;
13    int c  = 0;
14    int returnVar = 0;
15
16    /*Input*/
17    a = CS_INT_INPUT(int,"a");
18    b = CS_INT_INPUT(int,"b");
19    c = CS_INT_INPUT(int,"c");
20
21
22
23    /* int func1(signed int, signed int, signed int) */
24    returnVar = func1(a,b,c);
25
26
27    /*Output*/
28    CS_INT_OUTPUT(returnVar, "returnVar");
29
30 }
31
32 #endif
```

테스트 정보 | 테스트 케이스 | 테스트 코드 | 설정

- 사용자가 테스트 구조 트리를 편집한 후 저장하면 그 변경 사항은 자동으로 테스트 코드에 반영됩니다.
- 테스트 대상 함수의 반환 타입이 primitive 타입일 경우 반환 값을 출력하는 매크로가 자동으로 생성됩니다.
- 사용자가 직접 테스트 코드를 수정하려면 해당 테스트를 비관리 코드로 전환해야 합니다.
- 테스트 수행 함수의 형식은 반환 타입과 입력 파라미터가 void 타입이어야 합니다.

10.4. 설정

설정 탭에서는 유닛 테스트에 대한 여러 설정을 할 수 있습니다.



- 테스트 이름을 수정한 후 저장하면, 유닛 테스트 뷰에 나타나는 테스트 이름이 변경됩니다.
- 테스트에 대한 설명을 입력할 수 있습니다.
- 연관 파일 설정으로 테스트 대상 함수가 속한 파일을선택할 수 있습니다.
- 테스트에 대한 타임아웃 설정을 할 수 있습니다.
- 테스트에 대한 컨텍스트 유지 여부를 선택할 수 있습니다.
- 사용자가 테스트 코드를 직접 작성하려면 [비관리 코드 전환]을 해야 합니다.

10.5. 테스트 매크로

Controller Tester는 여러 가지 매크로를 제공하여 사용자가 테스트 코드를 보다 쉽게 작성할 수 있도록 도와줍니다. 제공되는 매크로는 에디터 뷰에서 “[Ctrl]+[SPACEBAR]” 단축키를 사용하여 확인 및 작성할 수 있습니다.

ASSERT 매크로

조건식을 검사하여 성공/실패 여부를 테스트 케이스 뷰에 출력합니다.

매크로 형태	입력값
CS_ASSERT(_b)	_B: 조건식
CS_ASSERT_MSG(_B, _msg)	_b: 조건식 _mgs: 조건식이 거짓일 경우 출력할 메시지

출력 매크로

특정 변수의 값을 테스트 케이스 뷰에 출력합니다.

매크로 형태	입력값
CS_INT_OUTPUT(_v, _s)	_v: 정수형 변수 _s: 테스트 데이터 이름
CS_UINT_OUTPUT(_v, _s)	_v: 부호 없는 정수형 변수 _s: 테스트 데이터 이름
CS_FLT_OUTPUT(_v, _s)	_v: 실수형 변수 _s: 테스트 데이터 이름
CS_STR_OUTPUT(_v, _s)	_v: char* 또는 char[]형 변수 _s: 테스트 데이터 이름

입력 매크로

테스트 수행 함수에 테스트 데이터를 전달하는 매크로입니다.

매크로 형태	입력값
CS_INT_INPUT(_t, _s)	_t: 정수형 타입 이름 _s: 테스트 데이터 이름

CS_UINT_INPUT(_t, _s)	_t: 부호 없는 정수형 타입 이름 _s: 테스트 데이터 이름
CS_FLT_INPUT(_t, _s)	_t: 실수형 타입 이름 _s: 테스트 데이터 이름
CS_STR_INPUT(_t, _s)	_t: char* 또는 char[] 형 타입 이름 _s: 테스트 데이터 이름

! 타깃 테스트에 사용할 스텁에는 입력 매크로를 사용할 수 없습니다.

주소 관련 매크로

변환된 소스코드에서 임베디드 상의 주소에 직접 값을 입력하거나 가져오는 부분이 있는 경우 로컬 컴퓨터에서 수행 시 정상 동작을 하지 않을 수 있습니다. 이런 경우 주소 관련 매크로를 사용하여 가상의 주소에 값을 입력하고 출력할 수 있습니다.

매크로 형태	설명
CS_VIRTUAL_ADDR(_b,_e)	주소(_b)부터 주소(_e)까지의 공간을 생성합니다.
CS_ADDR_ASSIGN(_t,_a,_v) CS_ADDR_SET(_t,_a,_v)	주소(_a)에 타입(_t)의 값(_v)을 지정합니다.
CS_ADDR_GET(_t,_a)	주소(_a)에서 타입(_t)의 값을 가져옵니다.
CS_VIRTUAL_ADDR_CLEAR()	생성한 메모리 공간을 제거합니다.

주소 관련 매크로 예제

```
//0xFFE40000U에서부터 100만큼의 가상메모리 공간 생성
CS_VIRTUAL_ADDR(0xFFE40000U,0xFFE40000U+100);

//0xFFE40000U에 int타입의 10할당
CS_ADDR_SET(int,0xFFE40000U,10);
CS_ADDR_ASSIGN(int,0xFFE40000U,10);

//변수 a에 0xFFE40000U의 값을 할당
a = CS_ADDR_GET(int,0xFFE40000U);
```

기타 매크로

매크로 형태	입력값	설명
CS_LOG(_msg)	_msg: 로그 메시지	사용자 로그 출력
CS_TESTCASENO()		현재 수행중인 테스트의 번호 반환

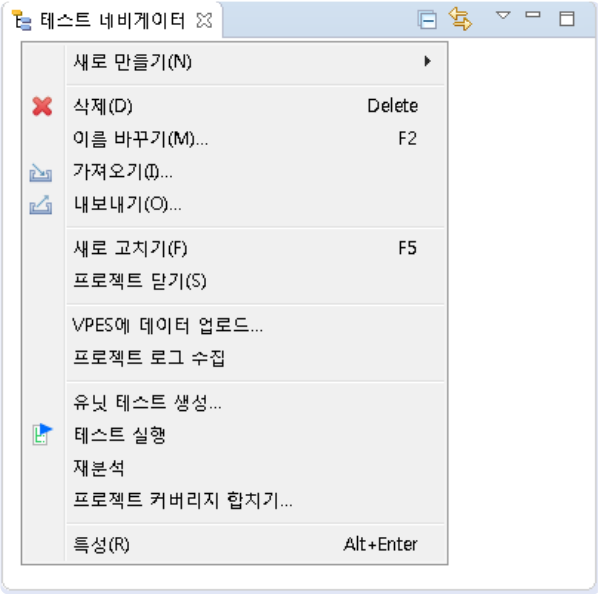
11. 테스트 실행

테스트는 다양한 방법으로 실행 가능합니다.

전체 테스트 실행

[유닛 테스트] 뷰와 [통합 테스트] 뷰에서 선택한 테스트를 실행합니다.

1. 테스트를 실행할 프로젝트를 선택, 우 클릭 후 [테스트 실행]

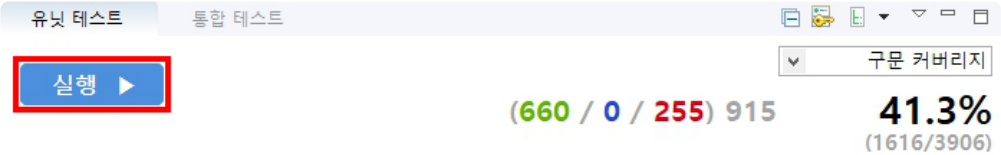


2. 테스트 진행 상태를 확인할 수 있는 진행 정보 창이 열리며, 테스트가 완료되면 각 뷰를 통해 결과를 확인할 수 있습니다.

유닛 테스트 실행

[유닛 테스트] 뷰에서 선택한 테스트를 수행합니다.

1. [유닛 테스트] 뷰의 대시보드에서 [실행] 버튼 클릭



2. 테스트 진행 상태를 확인할 수 있는 진행 정보 창이 열리며, 테스트가 완료되면 각 뷰를 통해 결과를 확인할 수 있습니다.

특정 테스트 케이스만 수행하려면 마우스 우 클릭하여 [테스트 케이스 실행] 메뉴를 클릭합니다.

통합 테스트 실행

[통합 테스트] 뷰에서 선택한 테스트를 수행합니다.

1. [통합 테스트] 뷰의 대시보드에서 [실행] 버튼 클릭

2. 테스트 진행 상태를 확인할 수 있는 진행 정보 창이 열리며, 테스트가 완료되면 각 뷰를 통해 결과를 확인할 수 있습니다.

12. 특성 페이지

Controller Tester는 프로젝트, 모듈, 소스파일(번역 단위)의 특성을 설정할 수 있는 페이지를 제공합니다.

특성 페이지 종류는 다음과 같습니다.

- 프로젝트 특성
- 모듈 특성
- 소스 파일(번역 단위) 특성

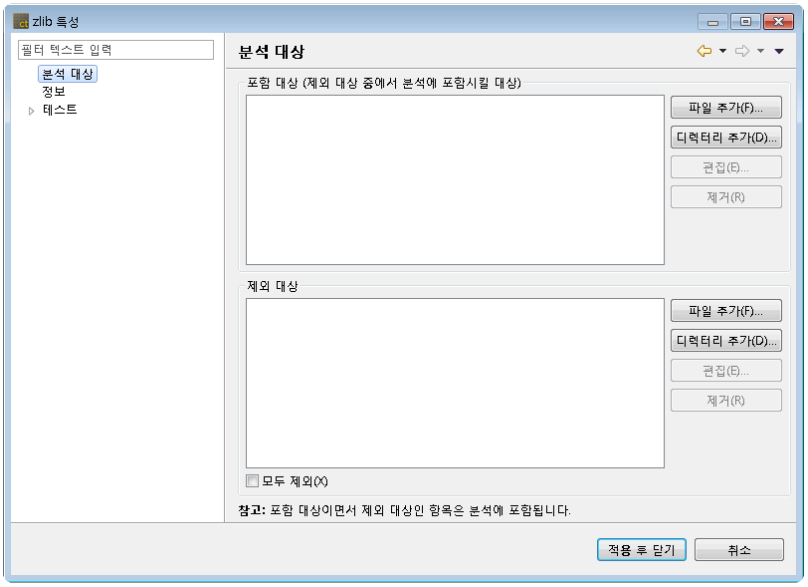
12.1. 프로젝트 특성

생성된 프로젝트의 특성을 변경할 수 있습니다.

분석 대상

해당 프로젝트를 분석할 때 제외 대상과 포함 대상을 설정합니다. 포함 대상은 제외 대상 중에서 분석에 포함시킬 대상을 입력합니다.

> 선택한 디렉터리 또는 파일만 분석하려면, 분석할 대상을 포함 대상에 추가하고 [모두 제외]를 체크합니다.

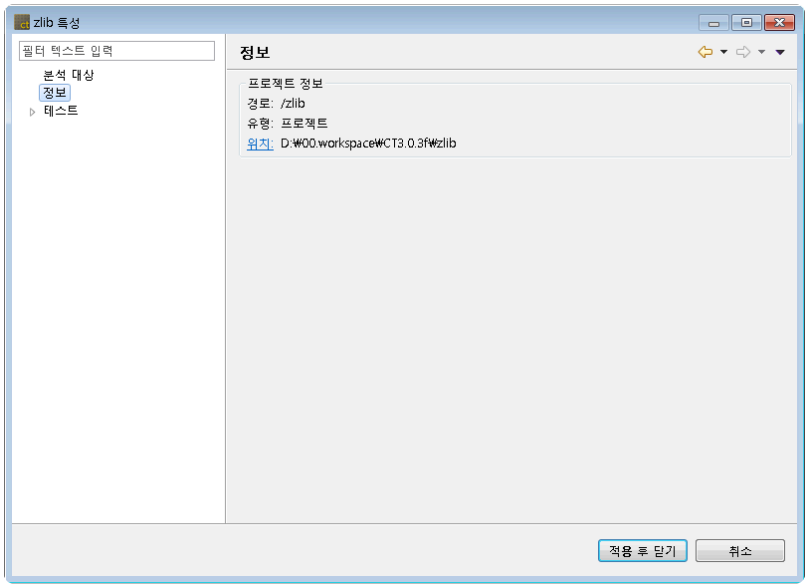


- 분석 포함 대상 추가
  1. [파일 추가] 또는 [디렉터리 추가] 버튼을 클릭합니다.
  2. 분석할 파일 또는 디렉터리를 선택합니다.
  3. [열기] 또는 [확인] 버튼을 클릭합니다.

- 분석 제외 대상 추가
  - [파일 추가] 또는 [디렉터리 추가] 버튼을 클릭합니다.
  - 분석 제외할 파일 또는 디렉터리를 선택합니다.
  - [열기] 또는 [확인] 버튼을 클릭합니다.

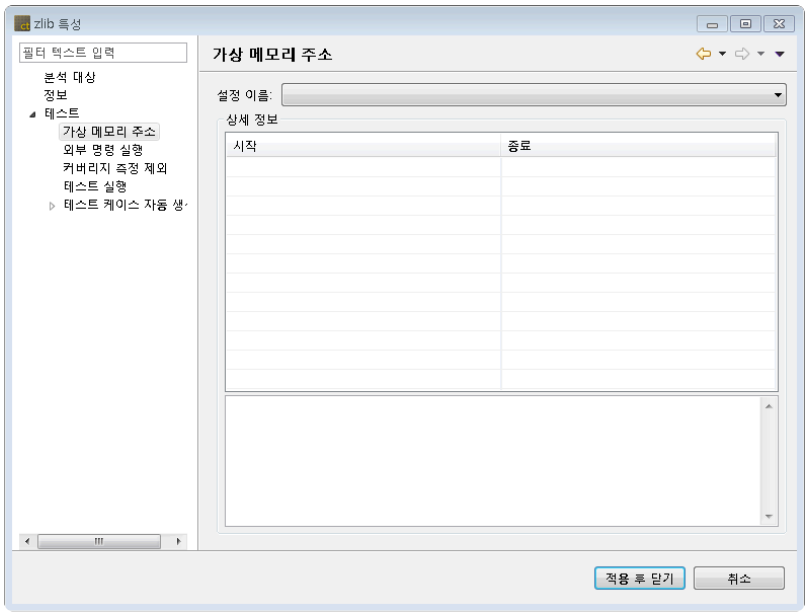
정보

프로젝트에 대한 간략한 정보(경로, 유형, 위치)를 보여줍니다.



가상 메모리 주소

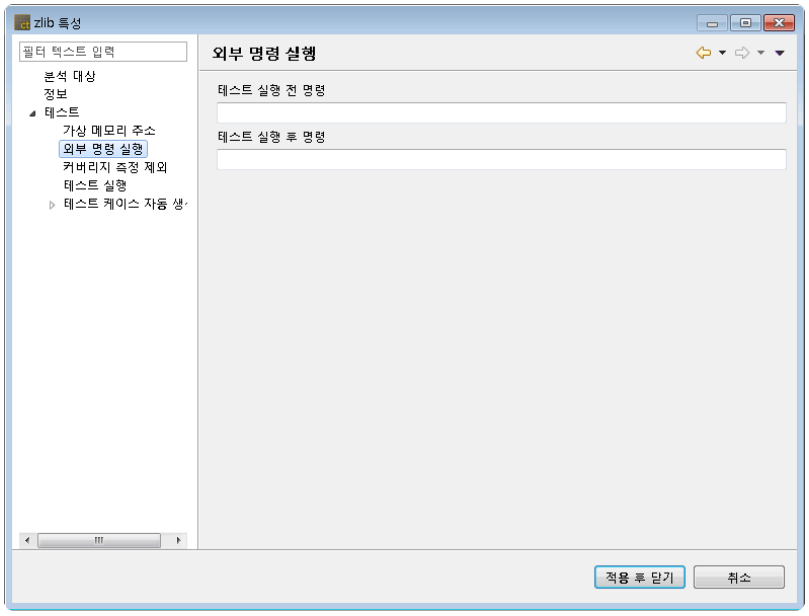
가상 메모리 주소 환경설정에서 관리되는 메모리 설정 정보를 선택할 수 있습니다.



외부 명령 수행

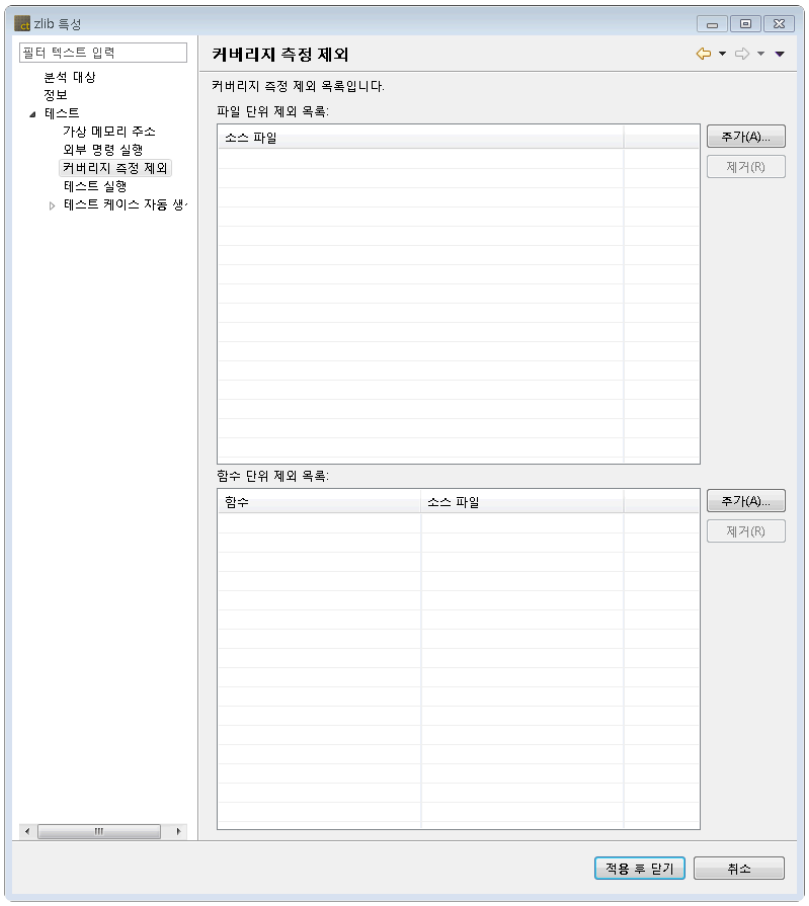
테스트 실행 전, 후 외부 명령어를 입력할 수 있습니다.
[테스트 실행 전 명령]에는 테스트를 수행하기 전 외부 명령을 수행 할 명령어 또는 배치 파일을 입력할 수 있습니

다.
[테스트 실행 후 명령]에는 테스트를 수행한 후 외부 명령을 수행 할 외부명령어 또는 배치 파일을 입력할 수 있습니다.



커버리지 측정 제외

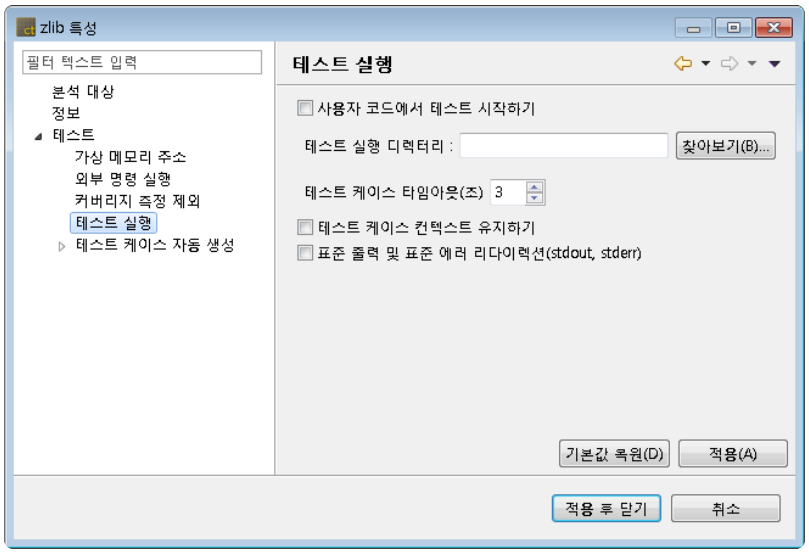
파일 단위 또는 함수 단위로 커버리지 제외 항목을 설정할 수 있습니다.
테스트 실행을 통해 제외된 함수의 커버리지 정보가 측정 대상에서 제외되었음을 확인할 수 있습니다.



테스트 실행

생성된 테스트를 워크스페이스가 아닌 사용자가 지정한 디렉터리에서 수행합니다. 이 경우 해당 테스트 대상이 사용하는 라이브러리나 헤더파일들을 워크스페이스로 복사하거나 링크 설정을 변경하는 작업을 하지 않아도 됩니다.

[사용자 코드에서 테스트 시작하기]의 체크 박스를 클릭합니다.
테스트가 수행될 홈 디렉터리를 선택한 후 [적용]버튼을 클릭합니다.



[테스트 케이스 타임아웃] 테스트 대상 함수가 테스트 케이스를 실행할 때 타임 아웃으로 판정하는 기준이 되는 시간을 설정합니다. 테스트 수행 시간이 테스트 케이스 타임아웃 시간을 초과하면 해당 테스트 케이스의 테스트는 종료되고 결과에 타임아웃으로 리포트 됩니다.

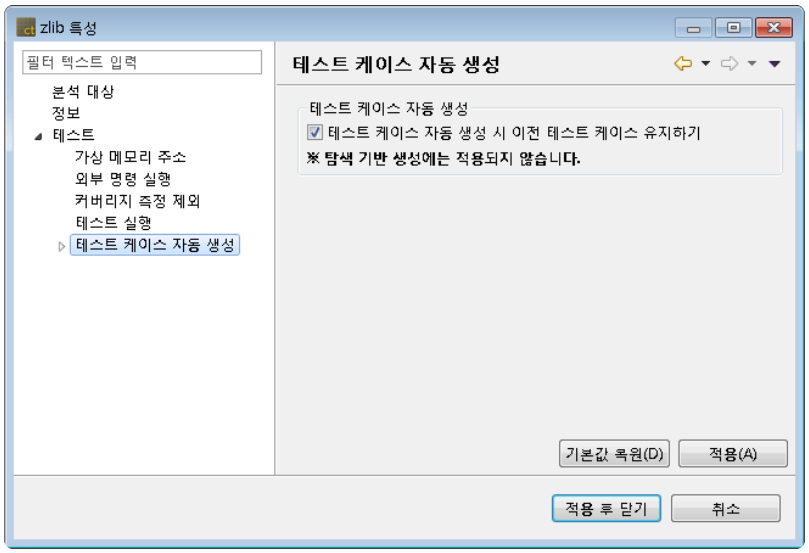
[테스트 케이스 컨텍스트 유지하기]는 하나의 실행기로 모든 테스트 케이스를 수행하는 기능입니다. 이전 테스트 수행 결과(static, 전역변수)가 테스트 수행 중 유지되며 테스트 케이스 결과값은 독립적으로 나타나지 않습니다. 이로 인해 테스트 케이스 결과값의 선, 후 관계가 분명해집니다.

[표준 출력 및 표준 에러 리다이렉션(stdout, stderr)]는 각 테스트 케이스별 로그파일에 표준 출력과 표준 에러를 기록하는 기능 입니다.

테스트 케이스 자동 생성

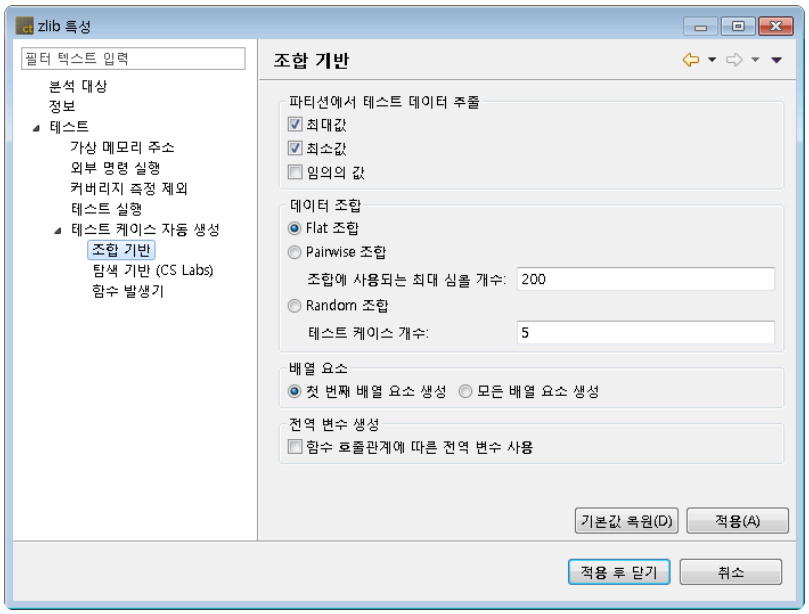
테스트 케이스 자동 생성 시 이전 테스트 케이스 유지하기를 설정할 수 있습니다.

> ! 탐색 기반 생성에는 적용되지 않습니다.



조합 기반

변수 파티션이 범위일 경우 테스트 데이터를 추출하는 방법을 설정할 수 있습니다.



체크박스를 체크하여 원하는 데이터의 추출 방법을 선택합니다. 이때 각 값의 의미는 다음과 같습니다.

추출 방법	설명
최대 값	파티션 구간의 최대 값 선택
최소 값	파티션 구간의 최소 값 선택
임의의 값	파티션 구간 중 임의의 값 선택

데이터 추출의 방법은 복수 선택이 가능합니다.
모두 체크 해제 시 기본으로 최대 값, 최소 값으로 자동 설정됩니다.

데이터 조합은 Flat 조합, Pairwise 조합과 Random 조합을 지원합니다. 조합의 의미는 다음 표와 같습니다.

값	설명
Flat	테스트 데이터 개수가 가장 많은 변수를 기준으로 단순 조합
Pairwise	최소의 개수로 서로소가 되도록 테스트 케이스를 조합하여 제한적이지만 최대의 커버리지 달성이 가능한 방식 조합에 사용되는 최대 심볼 개수 만큼 Pairwise로 수행하고 심볼 개수를 초과하면 Flat으로 생성(기본 200)
Random	입력 파라미터에 해당하는 변수 파티션의 최소값과 최대값 사이의 임의의 값으로 사용자가 입력한 테스트 케이스의 개수만큼 테스트 데이터를 조합(기본 5)

배열 요소는 각 배열 요소에 대한 테스트 데이터의 생성 여부를 선택하는 항목입니다.

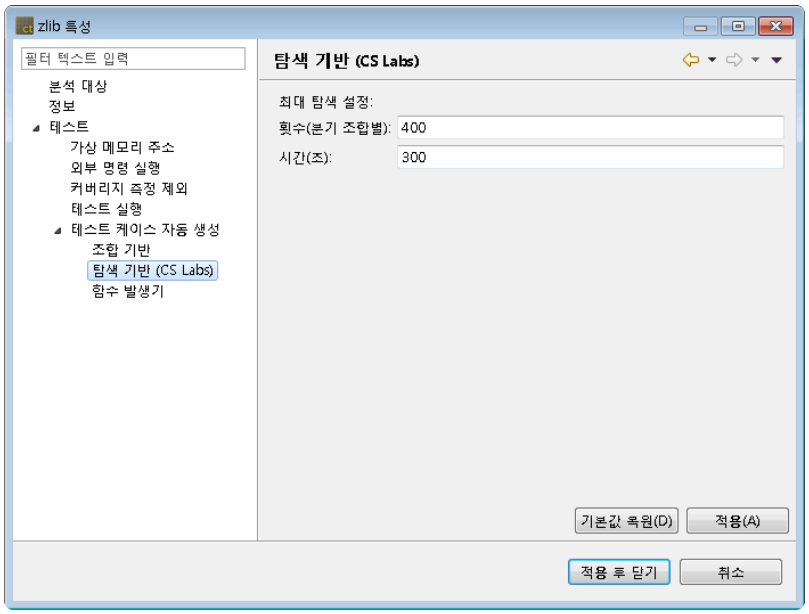
값	설명
첫 번째 배열 요서 생성	배열 사이즈에 상관없이 첫 번째 배열 요소만 생성
모든 배열 요소 생성	배열 사이즈만큼 배열 요소 생성

전역변수 생성은 테스트를 생성할 때 테스트 대상 함수로부터 호출되는 모든 함수들에서 사용하는 전역 변수를 모두 생성하는 옵션입니다.

탐색 기반(CS Labs)

> CS Labs는 CodeScroll 도구에서 향후 릴리즈 될 새로운 기능을 미리 사용해볼 수 있게 해줍니다. 새로운 기능에 대한 피드백을 바탕으로 CS Labs의 기능들은 정식 기능으로 반영되거나 사라질 수 있습니다.

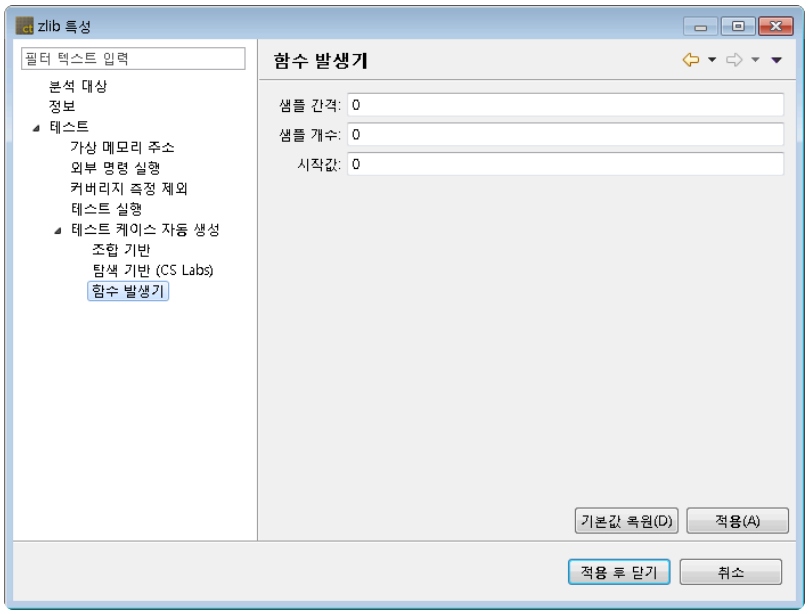
탐색 기반으로 테스트 케이스를 자동 생성하는 경우에 최대 탐색 설정을 할 수 있습니다.



설정	설명
횟수(분기 조합별)	분기 조합별로 입력한 횟수만큼 탐색
시간(초)	입력한 시간 동안 탐색

함수 발생기

함수 발생기를 사용하여 테스트 케이스를 자동으로 생성하는 경우에 각 함수가 공통으로 가지는 공통 설정 값을 지정합니다.

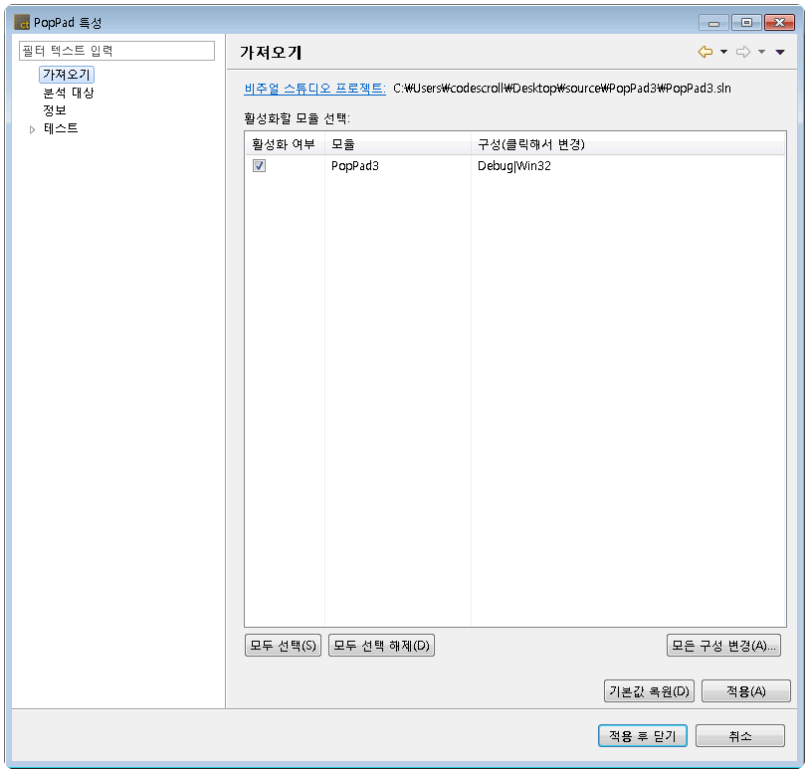


설정	설명
샘플 간격	함수로부터 샘플링할 샘플의 간격

샘플 개수	함수로부터 샘플링할 샘플의 개수 (테스트 케이스 개수)
시작 값	함수가 시작되는 기본 값으로 해당 값을 기준으로 값을 생성

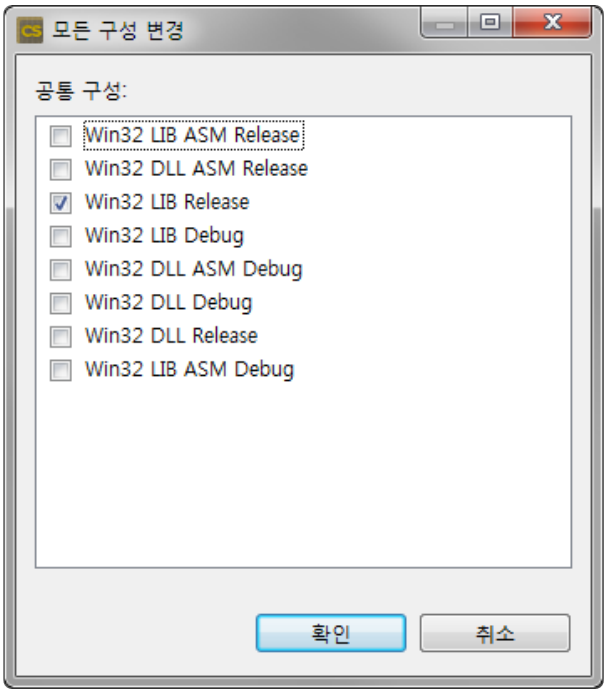
가져오기 설정

비주얼 스튜디오 프로젝트로 생성한 프로젝트



프로젝트를 생성하는데 사용한 비주얼 스튜디오 프로젝트의 경로와 선택된 프로젝트에서 활성화된 모듈을 확인할 수 있습니다.
각 모듈의 활성화 여부와 구성을 변경할 수 있습니다.
[모든 구성 변경] 메뉴를 통해 모든 모듈의 구성을 일괄 변경할 수 있습니다.

1. [모든 구성 변경] 버튼을 클릭하면, 모든 모듈에 공통으로 포함된 구성들 중에서 한가지를 선택할 수 있는 대화 상자가 표시됩니다.

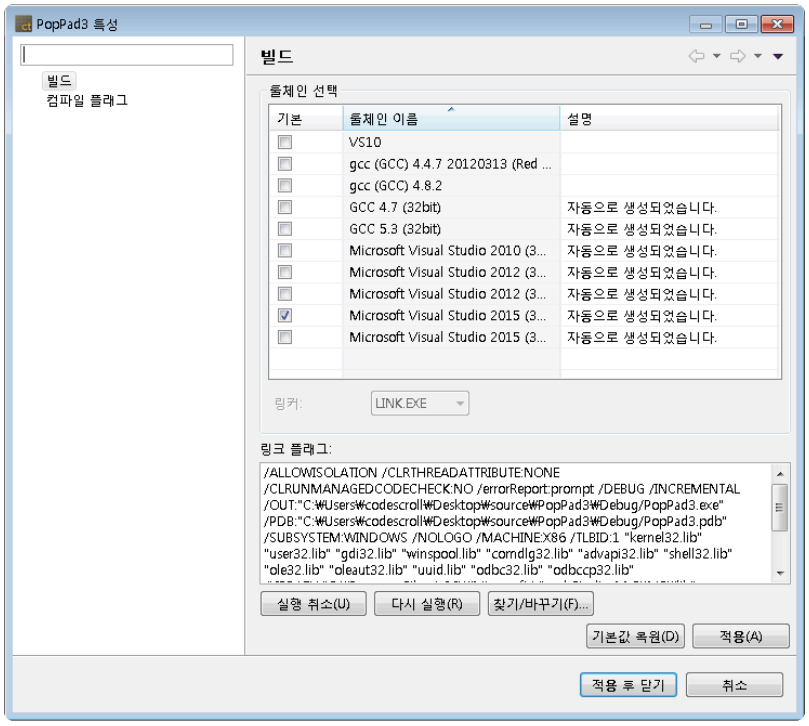


2. 모든 모듈에 공통으로 적용할 구성을 선택한 뒤 [확인] 버튼을 클릭합니다.

12.2. 모듈 특성

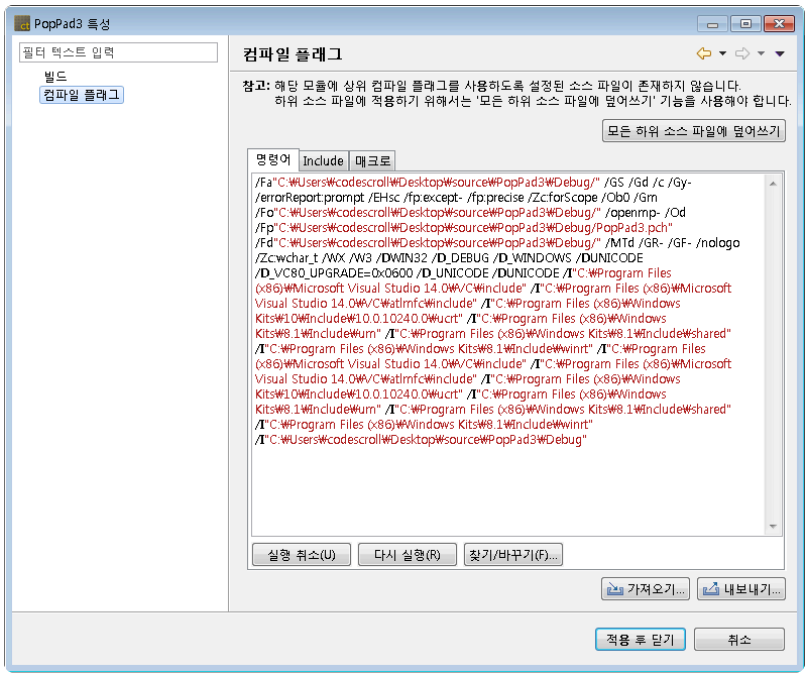
빌드

모듈에 대한 툴체인 관련 정보(툴체인, 링커)와 링크 플래그를 설정합니다. 툴체인을 변경하면 모듈 하위에 있는 모든 소스 파일(번역 단위)에 설정된 툴체인도 변경됩니다.



컴파일 플래그 설정

해당 모듈에 대한 컴파일 플래그 정보를 설정합니다.

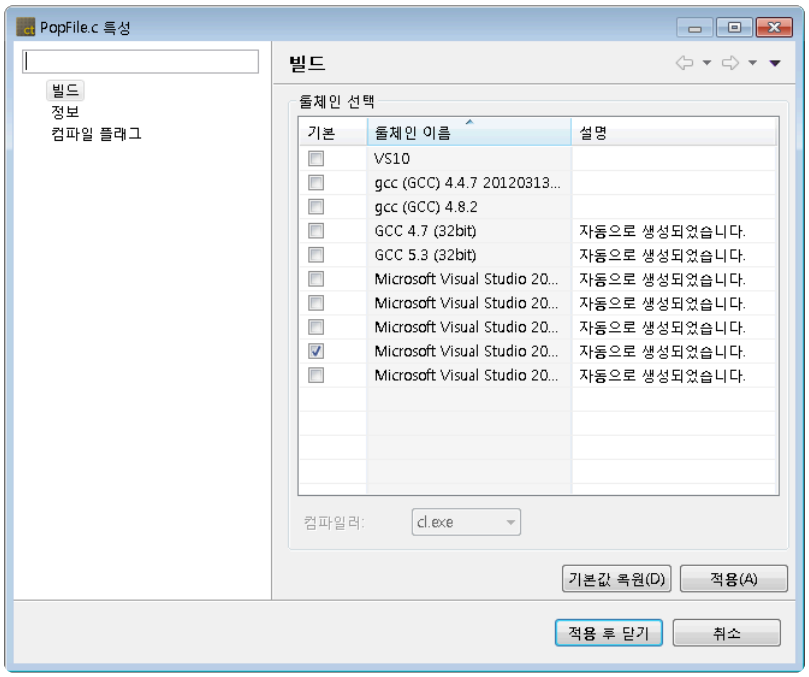


- [모든 하위 소스 파일(번역 단위)에 덮어쓰기]는 입력된 컴파일 플래그 정보가 해당 모듈 하위에 있는 모든 소스 파일(번역 단위)의 컴파일 플래그 정보를 덮어씁니다.
- [가져오기]는 외부에 있는 컴파일 플래그 정보를 가져옵니다. [내보내기]는 현재 컴파일 플래그 정보를 외부에 저장합니다.
  - 가져오기
    1. [가져오기] 버튼을 클릭합니다.
    2. 파일열기 다이얼로그로부터 컴파일 플래그 정보가 입력된 파일(.cf 파일)을 선택합니다.
  - 내보내기
    1. [내보내기] 버튼을 클릭합니다.
    2. 파일저장 다이얼로그로부터 컴파일 플래그 정보가 입력될 위치와 파일명을 입력 후 저장합니다.

12.3. 소스 파일(번역 단위) 특성

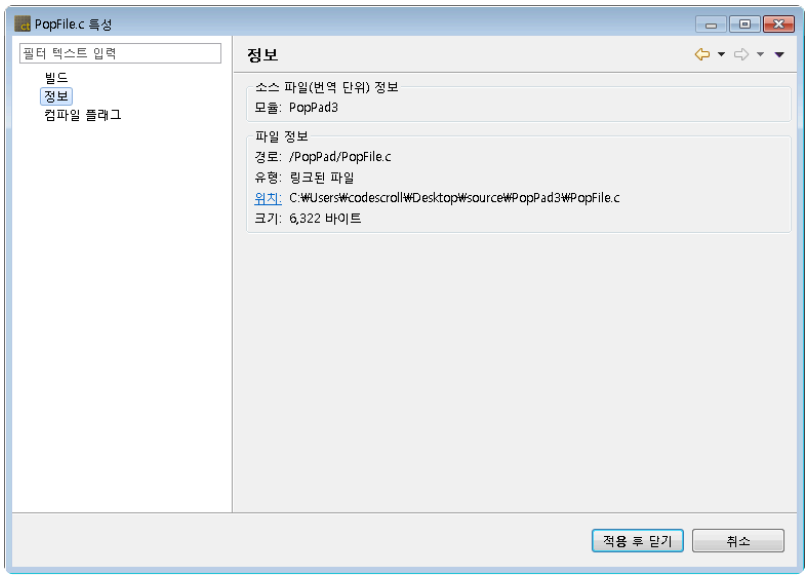
빌드

소스 파일(번역 단위)에 대한 툴체인 관련 정보(툴체인, 컴파일러)를 설정합니다.



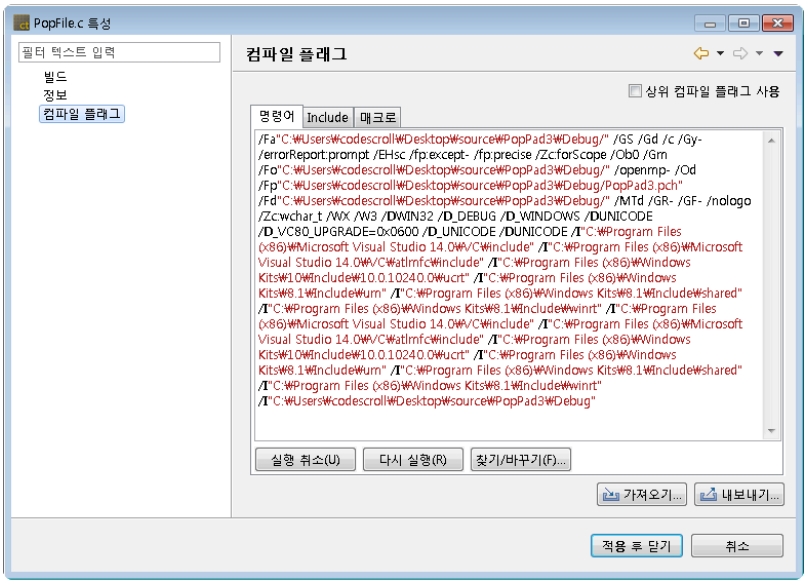
정보

소스 파일(번역 단위)이 포함된 모듈 정보와 파일 정보를 보여줍니다.



컴파일 플래그

소스 파일(번역 단위)에 대한 컴파일 플래그 정보를 설정합니다.



- [상위 컴파일 플래그 사용] 체크버튼을 선택하면 컴파일 플래그 정보를 변경할 수 없고, 상위 모듈의 컴파일 플래그 정보로 덮어쓰게 됩니다. 반대로 선택해제 하면 현재 컴파일 플래그 정보를 변경할 수 있습니다.
- [가져오기]는 외부에 있는 컴파일 플래그 정보를 가져옵니다. [내보내기]는 현재 컴파일 플래그 정보를 외부에 저장합니다.
  - 가져오기
    1. [가져오기] 버튼을 클릭합니다.
    2. 파일열기 다이얼로그로부터 컴파일 플래그 정보가 입력된 파일(.cf 파일)을 선택합니다.
  - 내보내기
    1. [내보내기] 버튼을 클릭합니다.
    2. 파일저장 다이얼로그로부터 컴파일 플래그 정보가 입력될 위치와 파일명을 입력 후 저장합니다.

13. 환경설정

환경 설정 메뉴에서는 도구 전체에 적용되고 있는 현재 설정을 확인하거나 변경할 수 있습니다.

Controller Tester가 제공하는 환경설정의 종류는 다음과 같습니다.

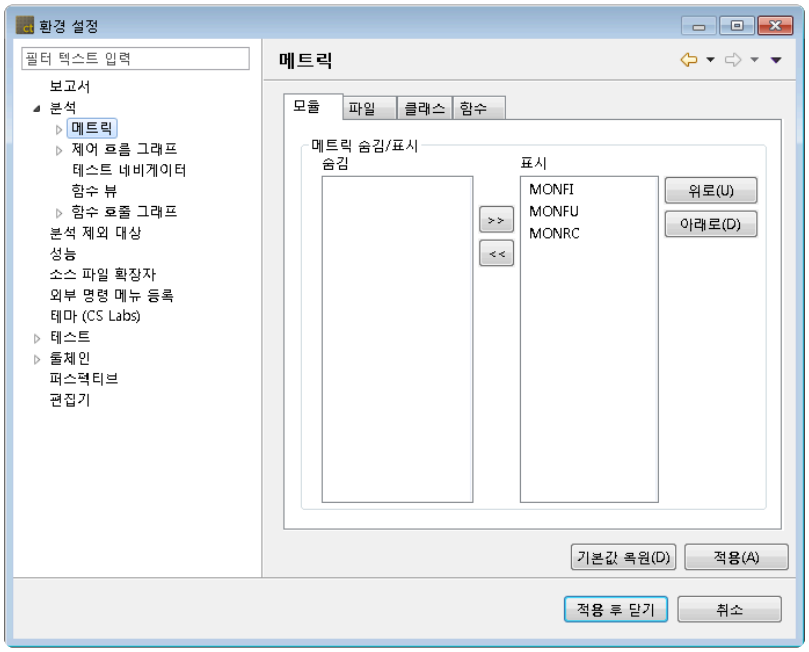
- 보고서
- 분석
- 분석 제외 대상
- 성능
- 소스 파일 확장자 설정
- 테마
- 테스트
- 툴체인 설정
- 퍼스펙티브
- 편집기

13.1. 분석

분석 정보에 관련된 설정을 확인하거나 변경할 수 있습니다.

메트릭

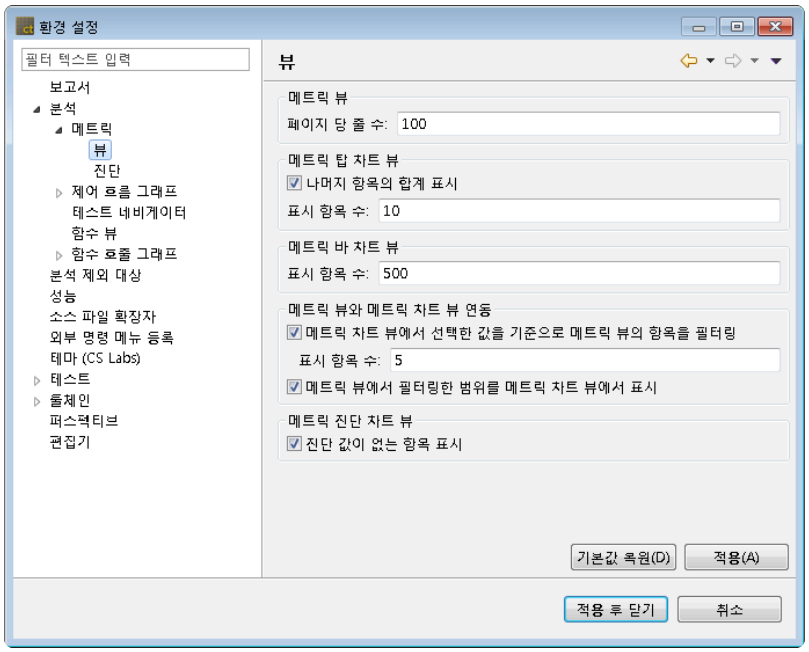
메트릭 데이터를 보여주는 모든 뷰에서 어떤 메트릭을 보이거나 숨길지 여부와 보이는 순서를 설정할 수 있습니다.
툴팁 메시지로 선택한 메트릭에 대한 설명을 볼 수 있습니다.



메트릭을 선택한 뒤 [>>] 또는 [<<] 버튼으로 숨김 목록이나 표시 목록으로 이동할 수 있습니다.
표시 목록에 있는 메트릭을 선택한 뒤 [위로] 또는 [아래로] 버튼으로 메트릭이 보이는 순서를 변경할 수 있습니다.

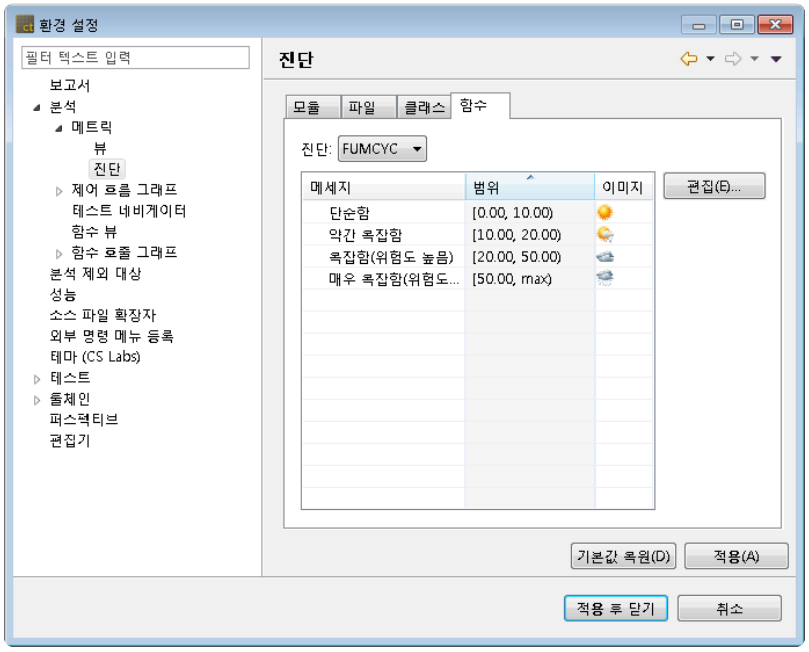
메트릭 뷰

메트릭 데이터를 보여주는 뷰들의 설정을 변경할 수 있습니다.

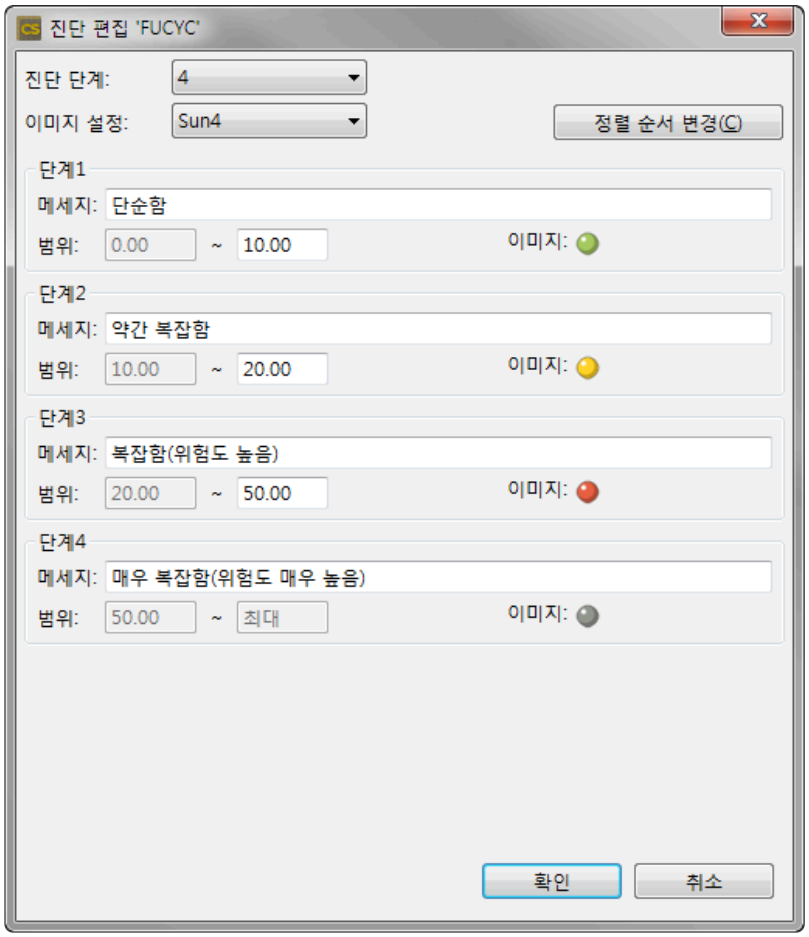


진단

메트릭에 대한 진단 정보를 확인하거나 설정할 수 있습니다.
선택된 메트릭 값이 각각의 범위에 포함될 경우에 대한 진단 메시지와 진단 이미지(메트릭 뷰에 표시)를 보여줍니다.



진단 정보를 수정하려면 [편집] 버튼을 클릭합니다.
선택한 메트릭의 진단 정보를 편집할 수 있는 대화상자가 나타납니다.

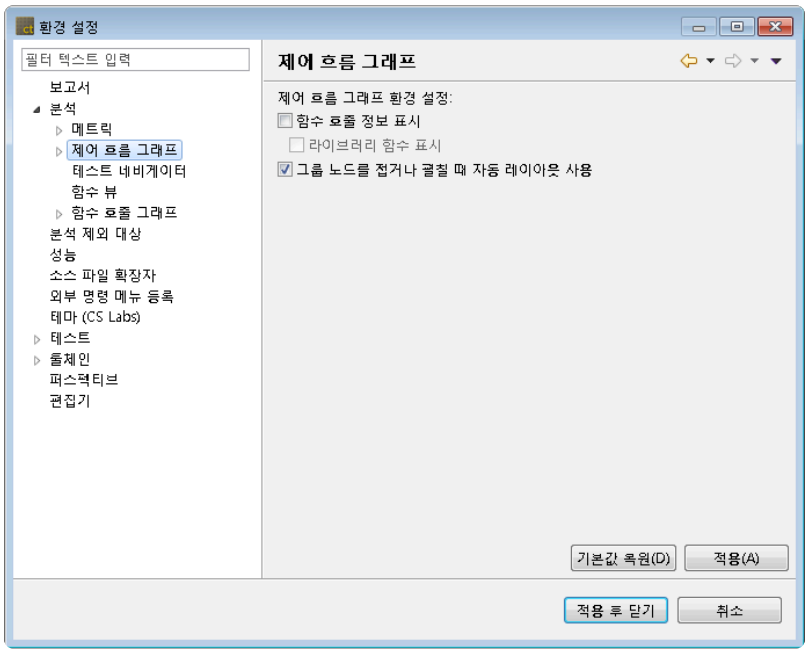


1. 진단 단계(2~5)를 설정합니다.
2. 메트릭 뷰에서 보여줄 진단 이미지를 설정합니다.
3. [정렬 순서 변경] 버튼을 누르면 진단 이미지의 정렬 순서가 변경됩니다.
4. 각 단계별로 진단 메시지와 진단 범위를 입력합니다.
5. [확인] 버튼을 클릭합니다.

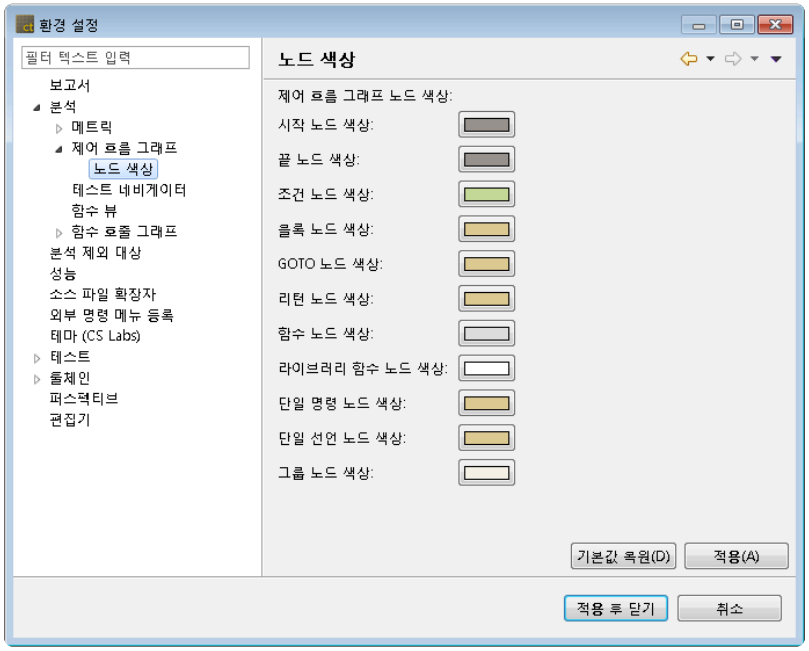
제어 흐름 그래프

제어 흐름 그래프 뷰에서 보이는 제어 흐름 그래프에 대한 설정을 변경할 수 있습니다.

노드 색상

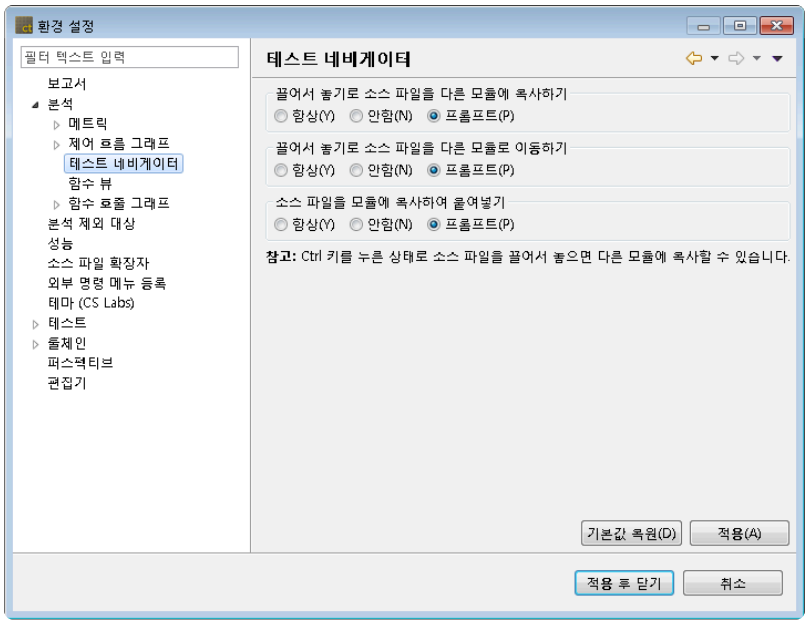


제어 흐름 그래프에 보이는 노드의 색상을 변경할 수 있습니다.

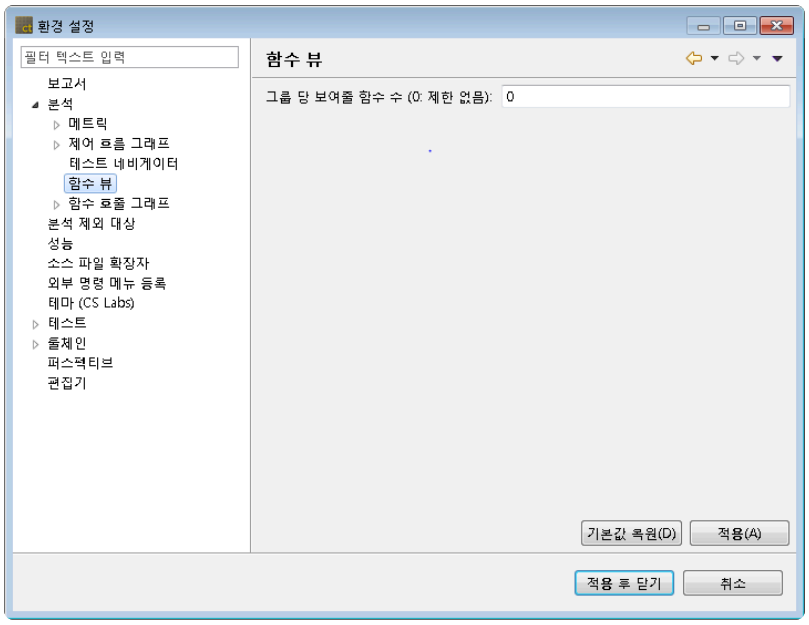


테스트 네비게이터

테스트 네비게이터 뷰에서 소스 파일을 복사하거나 이동하는 각각의 작업에 대해, 계속할지를 항상 묻거나 묻지 않도록 설정할 수 있습니다.

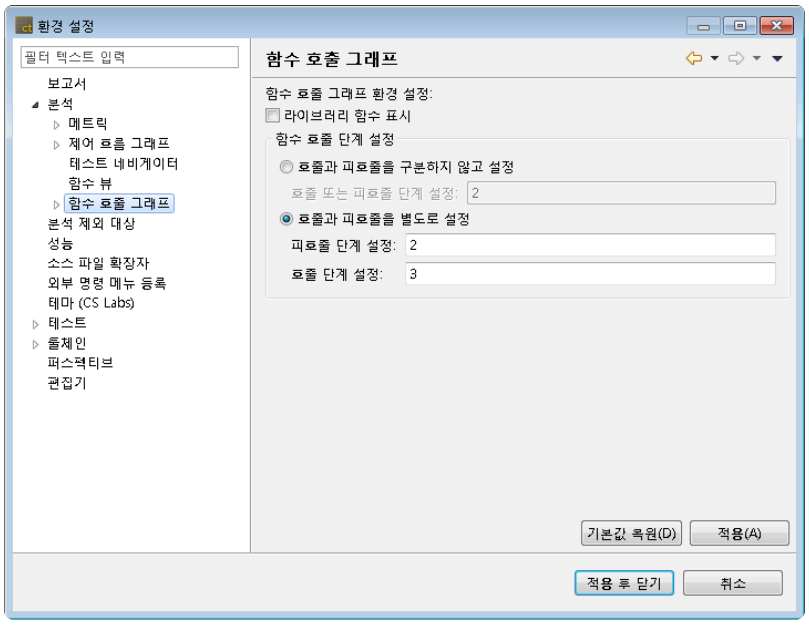


함수 뷰

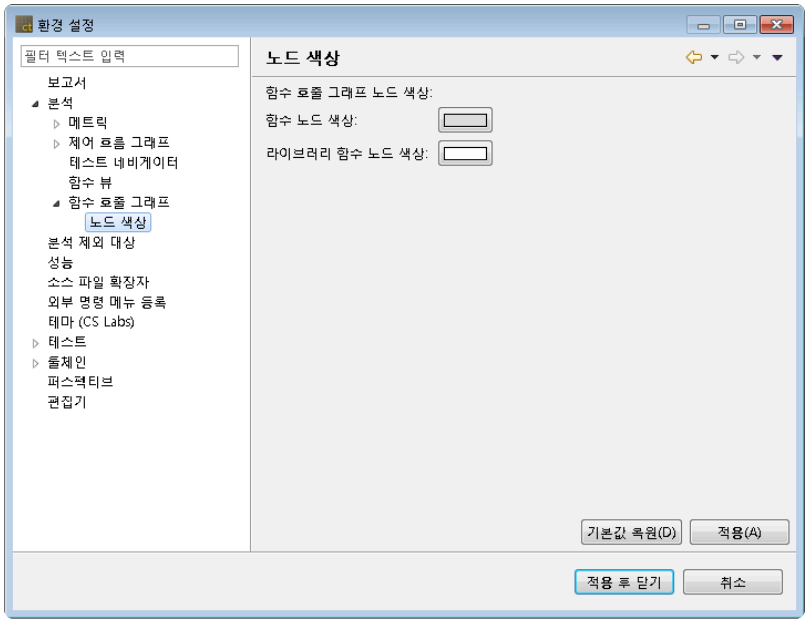


함수 호출 그래프

함수 호출 그래프 뷰에서 보이는 함수 호출 그래프의 설정을 변경할 수 있습니다.

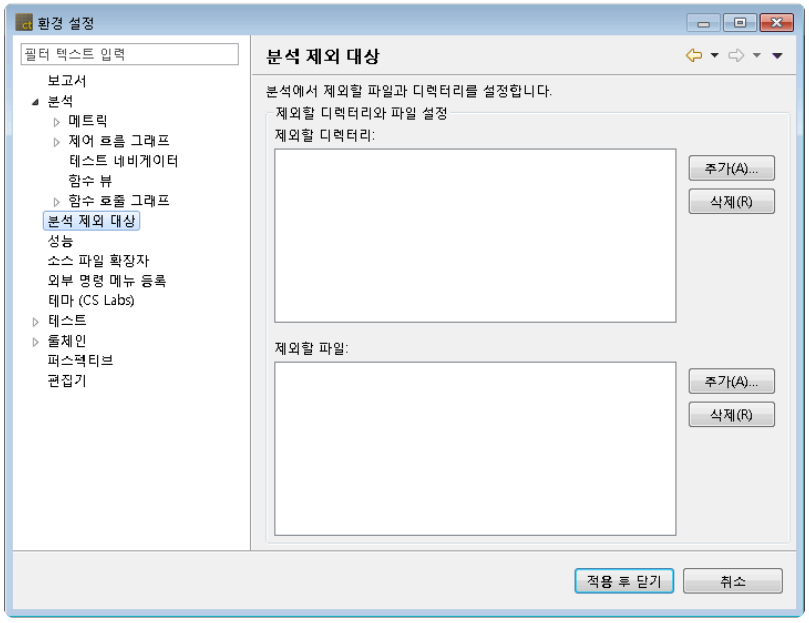


[라이브러리 함수 표시] 체크박스는 라이브러리 함수를 표시할 것인지 아닌지를 설정할 수 있습니다.
함수 호출 단계 설정에서 '호출과 피호출을 구분하지 않고 설정'의 값을 '2'로 설정하면 선택한 함수를 중심으로 호출과 피호출 함수 모두에 대해 한 단계의 호출 정보를 보여줍니다(선택한 함수를 호출하거나 선택한 함수가 호출한 함수까지 보여짐).
만일, 호출과 피호출을 별도로 설정하고자 한다면 하위 옵션을 선택하여 각각의 단계 값을 지정할 수 있습니다. 함수 호출 그래프에 보이는 노드의 색상을 변경할 수 있습니다.



13.2. 분석 제외 대상

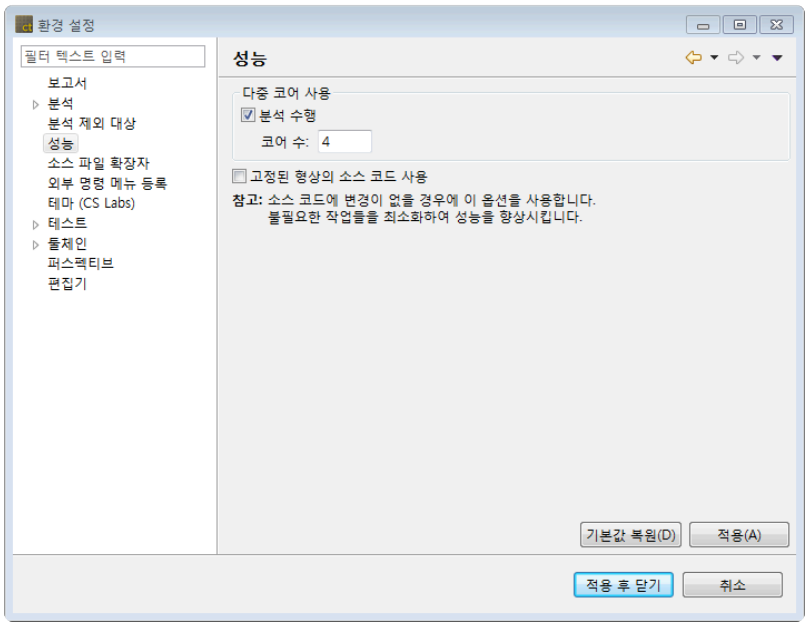
분석에서 제외할 디렉터리와 파일을 설정합니다.



- 분석 제외 대상 추가
  1. [추가] 버튼을 클릭합니다.
  2. 제외할 디렉터리 또는 파일을 선택합니다.
  3. [확인] 또는 [열기] 버튼을 클릭합니다.
- 분석 제외 대상 삭제
  1. 삭제할 디렉터리 또는 파일을 선택합니다.
  2. [삭제] 버튼을 클릭합니다.

13.3. 성능

사용자 환경에 맞게 작업 성능을 향상시킬 수 있는 설정을 제공합니다.



다중 코어 사용

[다중 코어 사용] 옵션은 프로젝트 분석을 수행할 때 다중 코어를 사용하여 성능을 향상시킵니다(CPU 자원이 충분하지 않은 PC에서는 다른 작업의 성능에 영향을 미칠 수 있습니다).

- 분석 수행: 소스 코드를 분석할 때 사용할 코어 수를 입력합니다(입력할 수 있는 코어 수: 2개 ~ 실행 환경의 최대 코어 수).

고정된 형상의 소스 코드 사용

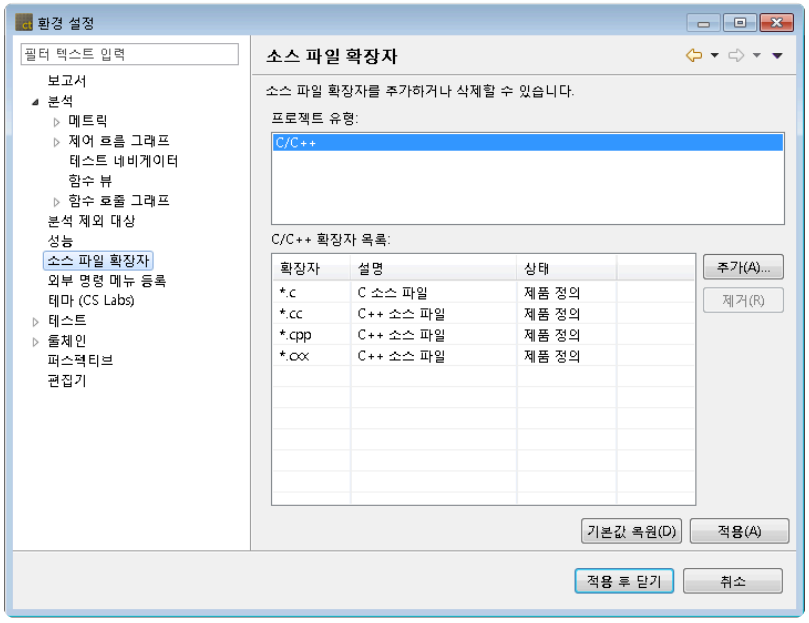
[고정된 형상의 소스 코드 사용] 옵션은 소스 코드 형상에 변경이 없다고 가정하고, 불필요한 작업(소스 코드 수정 감지 및 재분석 등)을 최소화하여 작업 성능을 향상시킵니다.
PC 성능이 매우 느리고, 테스트 수행 중 소스 코드 형상이 절대로 변경되지 않는 경우에만 이 옵션을 사용하시기 바랍니다(옵션이 활성화된 상태에서 소스 코드 또는 분석 설정이 변경되면, 예상하지 못한 문제가 발생할 수 있습니다).

> **편집기 – [수정 감지] 옵션과 차이점**
> 편집기의 [수정 감지] 옵션을 활성화하는 경우 [분석]을 수행하면 수정된 소스 코드로 분석하지만, 분석의 [고정된 형상의 소스 코드 사용] 옵션을 활성화하면 소스 코드를 수정해도 [재분석]하

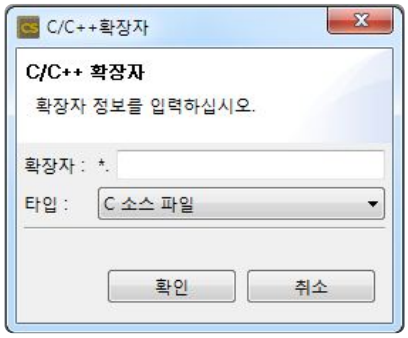
기 전까지 수정된 소스 코드를 분석하지 않습니다. 또한 편집기 – [수정 감지] 옵션은 무시됩니다.

13.4. 소스 파일 확장자 설정

프로젝트 유형별로 소스 파일로 사용할 확장자를 설정합니다.



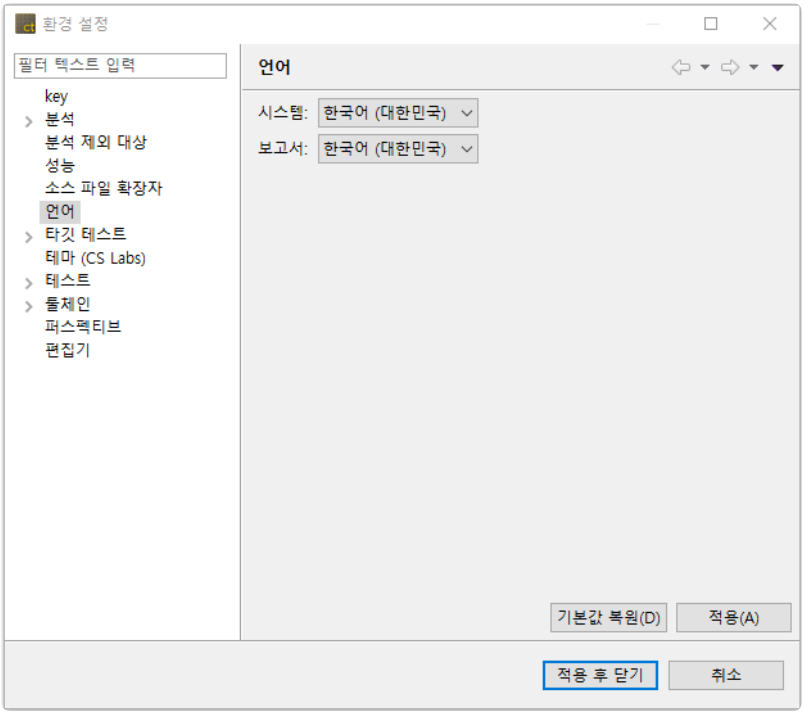
- 소스 파일 확장자 추가
  - 프로젝트 유형을 선택합니다.
  - [추가] 버튼을 클릭합니다.
  - 확장자를 입력하고 타입을 선택한 후 [확인] 버튼을 클릭합니다.



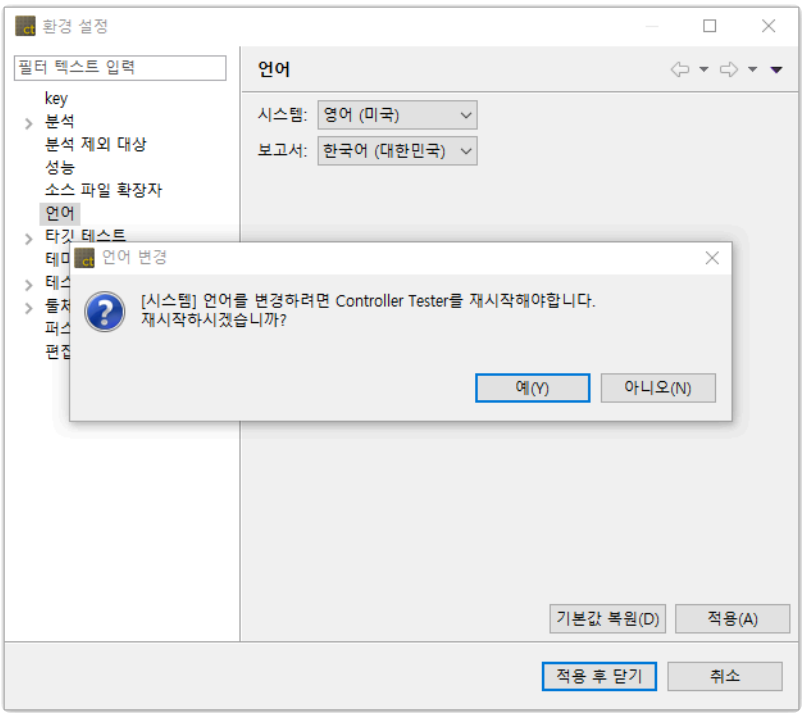
- 소스 파일 확장자 삭제
  - 프로젝트 유형을 선택합니다.
  - 삭제할 확장자를 선택합니다.
  - [삭제] 버튼을 클릭합니다.

13.5. 언어

도구와 보고서에 표시할 언어를 설정합니다.
[도구 언어], [보고서 언어] 각각의 드롭다운 메뉴에서 언어를 선택할 수 있습니다.



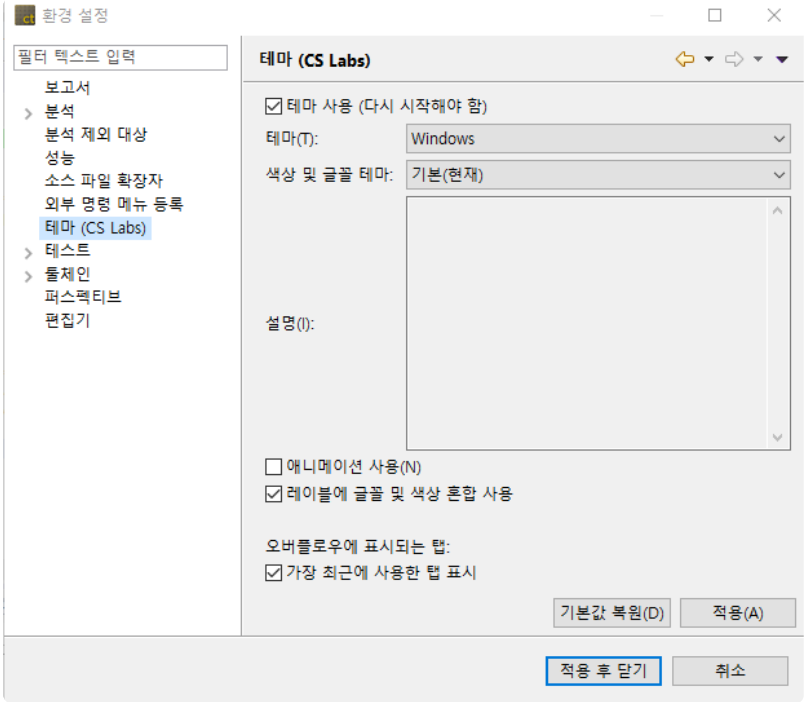
[도구 언어] 변경 후 [적용] 버튼을 클릭하면 Controller Tester을 재시작을 묻는 대화창이 열립니다.



13.6. 테마

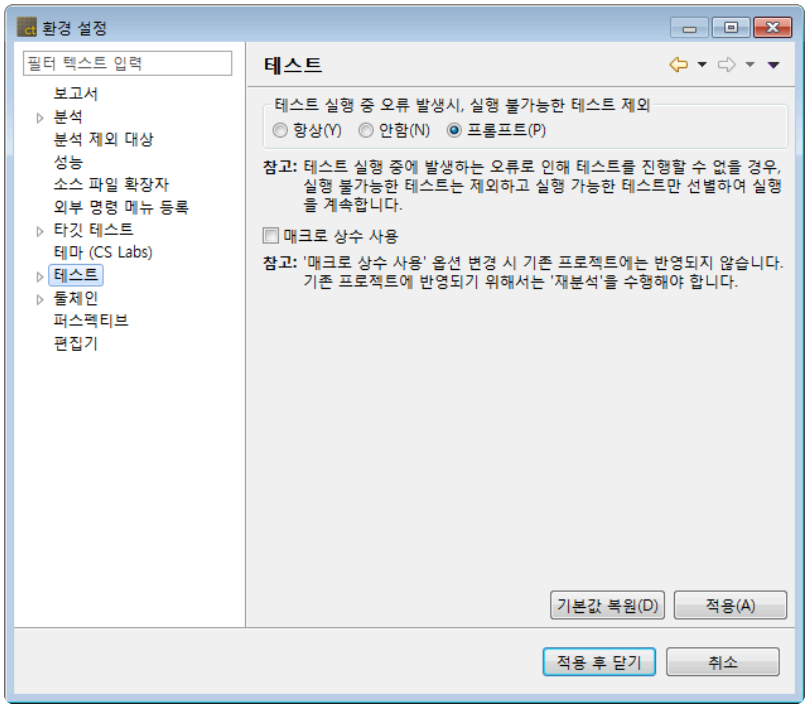
Controller Tester의 테마를 변경할 수 있는 설정을 제공합니다.

> CS Labs는 CodeScroll 도구에서 향후 릴리즈 될 새로운 기능을 미리 사용해볼 수 있게 해줍니다. 새로운 기능에 대한 피드백을 바탕으로 CS Labs의 기능들은 정식 기능으로 반영되거나 사라질 수 있습니다.



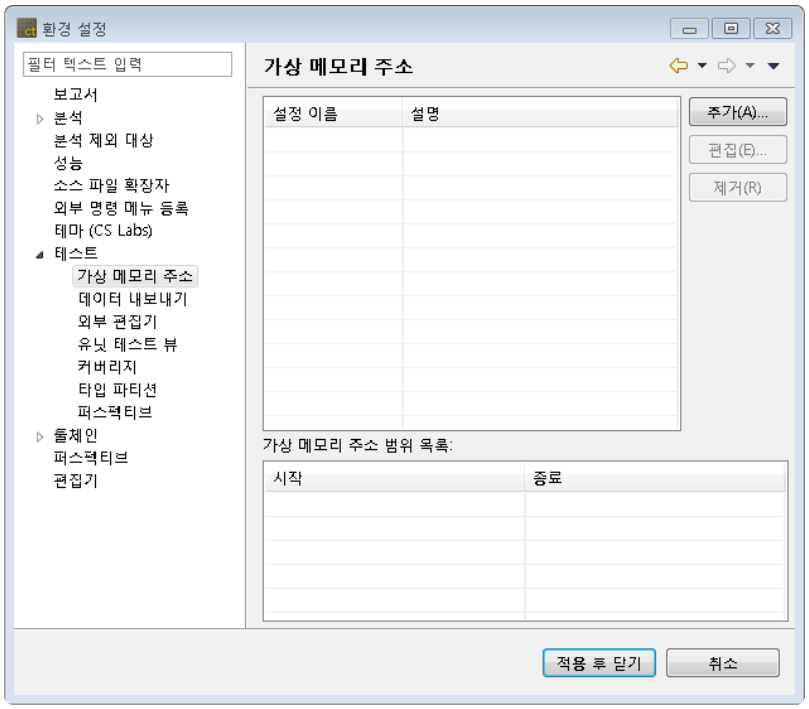
13.7. 테스트

테스트 편집 및 실행 관련 설정을 제공합니다.
[테스트 실행 중 오류 발생시, 실행 불가능한 테스트 제외] 옵션은 테스트 실행 중에 발생하는 오류(예: 테스트 대상 소스 코드가 변경된 경우)로 인해 테스트를 진행할 수 없을 경우, 실행 불가능한 테스트는 제외하고 실행 가능한 테스트만 선별하여 실행할 수 있도록 설정합니다.
[매크로 상수 사용] 옵션은 소스 파일에 정의된 매크로 상수를 이용하여 테스트를 편집(예: 배열 인덱스 및 테스트 케이스의 입력/기대값에 매크로 상수 사용)할 수 있도록 설정합니다(※ 이 설정은 변환 프로젝트인 경우에만 적용됩니다).

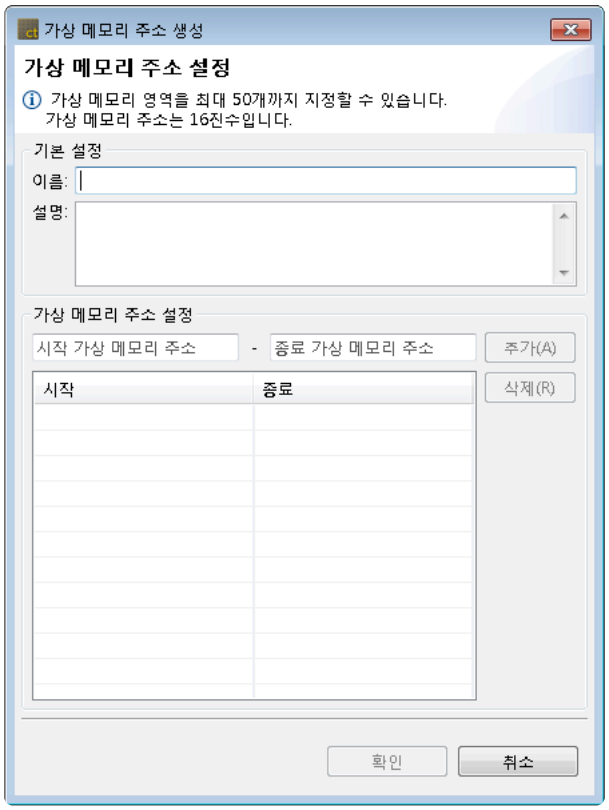


가상 메모리 주소

임베디드 환경 테스트를 위한 메모리 설정을 제공합니다. 프로젝트 특성에서 사용할 메모리 주소 그룹을 관리할 수 있습니다.



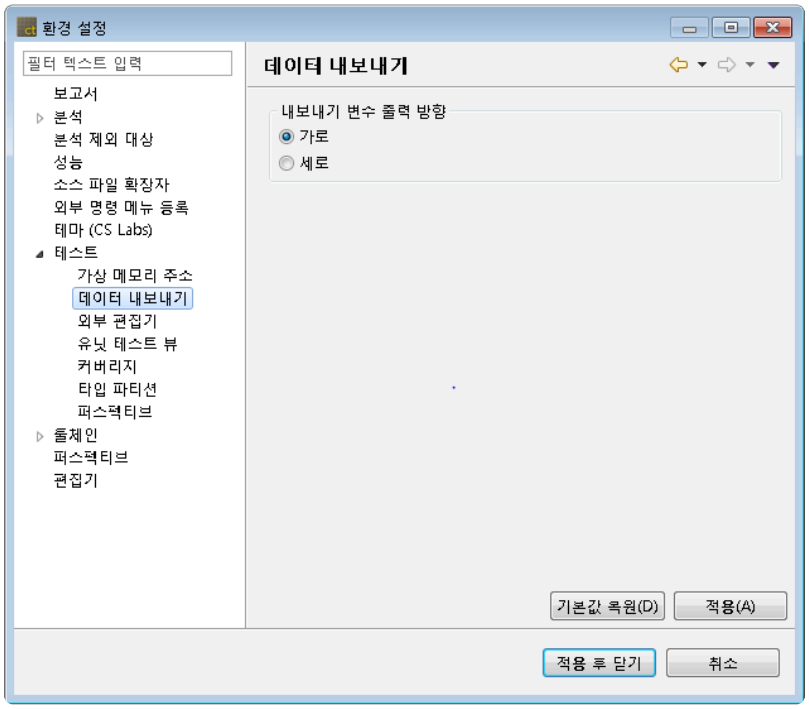
위 화면에서는 설정된 가상 메모리 주소 정보를 확인할 수 있습니다. [추가...] 버튼을 클릭하여 가상 메모리 주소를 추가할 수 있습니다.



가상 메모리 주소 영역은 최대 50개까지 지정할 수 있고, 16진수로 입력할 수 있습니다.

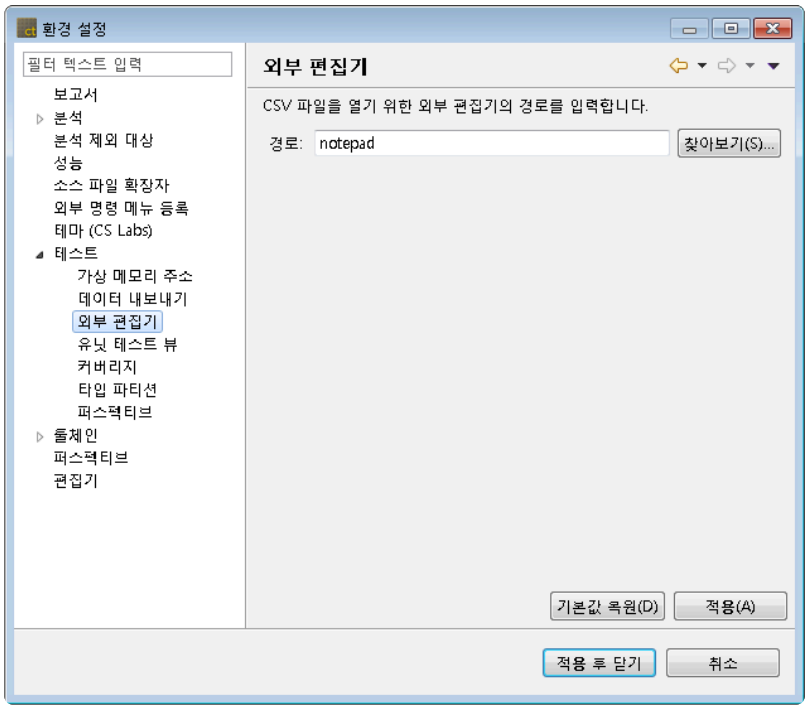
데이터 내보내기

데이터 내보내기 수행 시 변수의 출력 방향을 가로 또는 세로로 설정할 수 있습니다.
내보내기 변수 출력 방향을 선택한 후 [적용] 버튼을 클릭합니다.



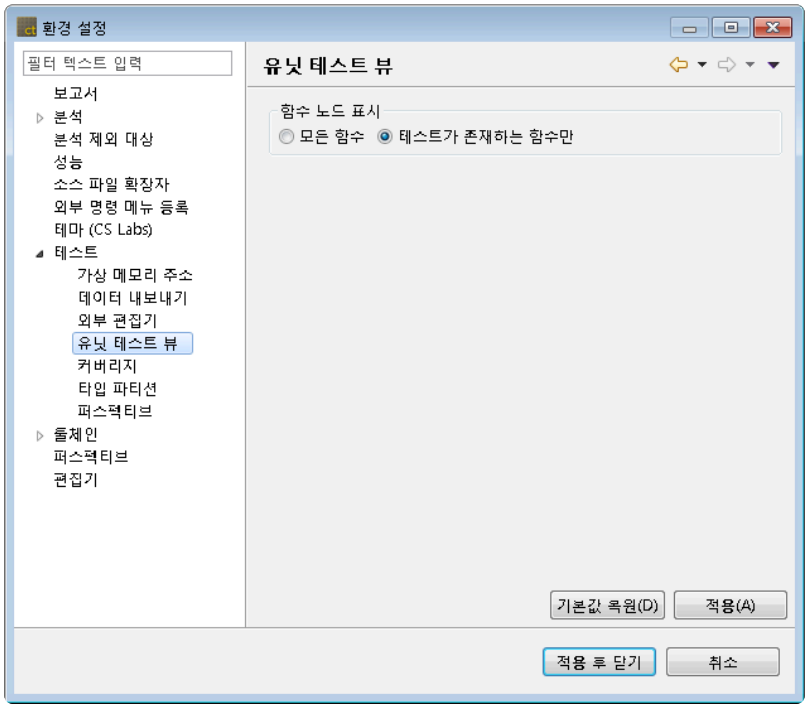
외부 편집기

CSV 파일을 열기 위한 외부 편집기를 설정할 수 있습니다.
외부 편집기 경로를 입력한 후 [확인] 버튼을 클릭합니다.



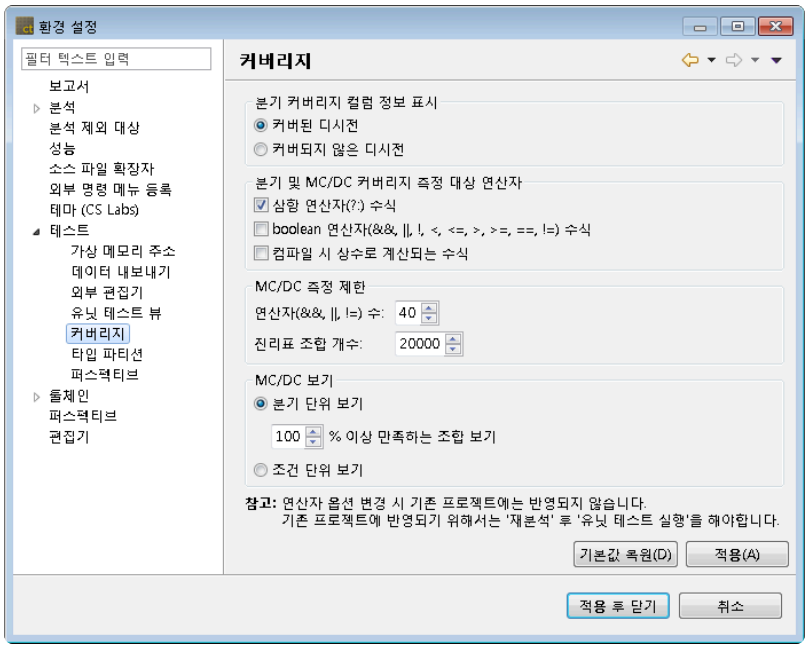
유닛 테스트 뷰

UI/UX가 개선된 테스트 퍼스펙티브에 포함된 유닛 테스트 뷰의 함수 노드 표시에 관해 설정할 수 있습니다.



커버리지

분기 커버리지 측정 및 커버 여부 표시에 관해 설정할 수 있습니다.



[분기 커버리지 컬럼 정보 표시] 라디오 버튼은 에디터 왼쪽 컬럼에 보여질 마커의 정보를 선택할 수 있도록 합니다.

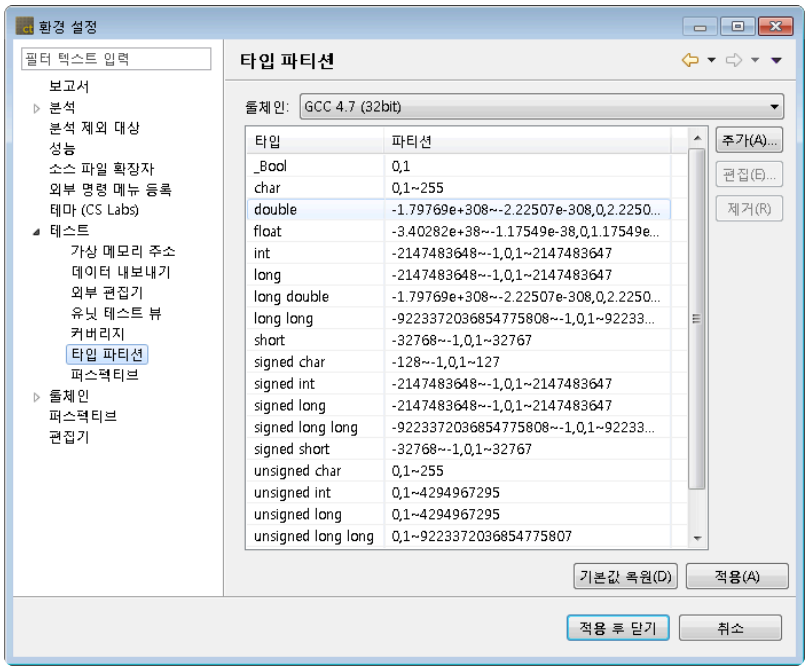
- 소스 코드 에디터에 보여지는 T/F 마커 옵션
- 커버된 디시전: 커버된 디시전의 T/F표시
- 커버되지 않은 디시전: 커버되지 않은 디시전의 T/F표시

[분기 및 MC/DC 커버리지 측정 대상 연산자] 체크 박스 버튼은 분기 커버리지 측정 대상을 설정합니다. 재분석 및 실행을 하면 해당 설정이 적용 됩니다.

- 삼항 연산자(?:) 수식: 삼항 연산자 수식 분기 커버리지 측정
- boolean 연산자(&,||,!,<,<=,>,>=,==,!=) 수식: boolean 연산자 수식 분기 커버리지 측정
- 모두 체크 해제하면 if, for, while, do-while, switch 문만 분기 커버리지 측정

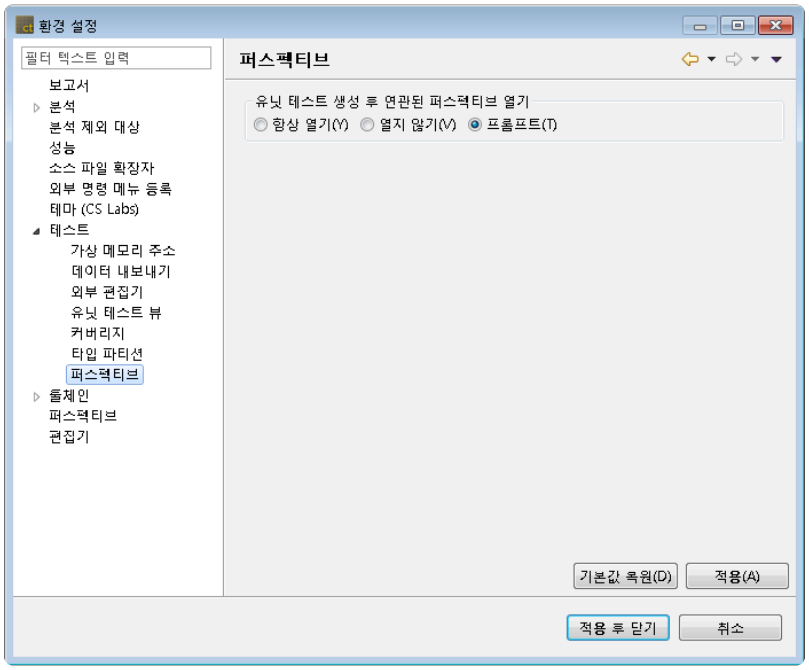
타입 파티션

각 툴체인에 따른 기본 타입 파티션을 편집 및 제공되는 기본 값으로 복원할 수 있습니다.
툴체인 콤보 박스에서 수정할 툴체인을 선택하고 파티션을 편집 후 [적용] 버튼을 클릭합니다.



퍼스펙티브

테스트 생성 후 연관된 퍼스펙티브 열기에 관해 설정할 수 있습니다.
퍼스펙티브 항상 열기, 열지 않기, 프롬프트 중 하나를 선택한 후 [적용] 버튼을 클릭합니다.



13.8. 툴체인 설정

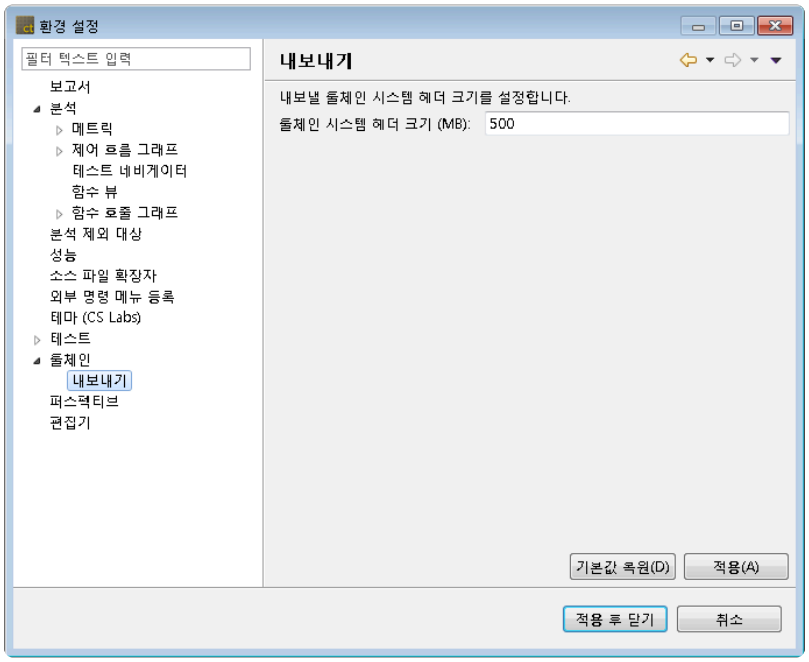
도구에서 사용할 툴체인에 대한 정보를 설정합니다.

테스트 대상 소스에 대한 툴체인(컴파일러 정보)이 있어야 프로젝트를 생성하거나 분석할 수 있습니다.

툴체인 설정에 대한 상세 사용법은 제품 매뉴얼의 “툴체인(분석기)의 설정” 장을 통해 확인할 수 있습니다.

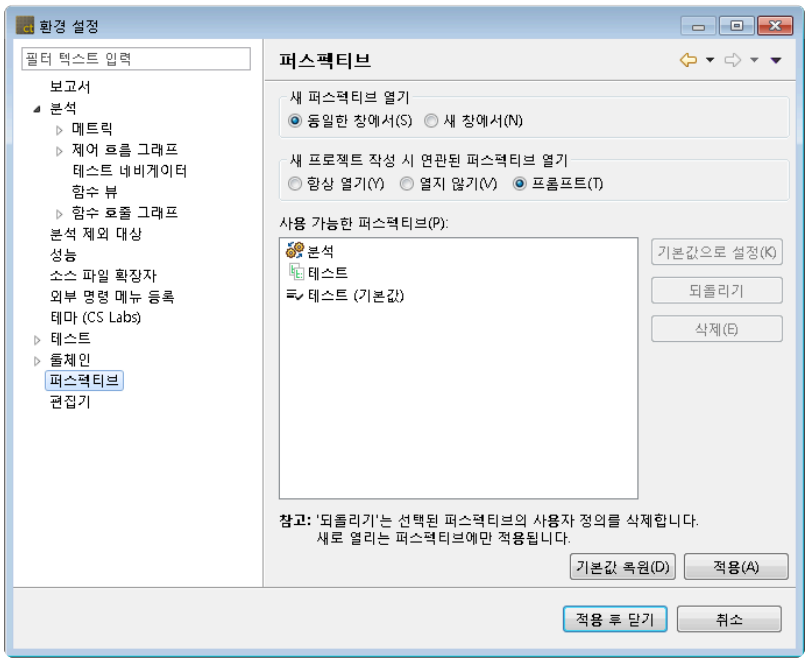
툴체인 내보내기

내보낼 툴체인의 시스템 헤더 크기를 설정합니다. 설정한 크기 보다 큰 시스템 헤더는 내보낼 수 없습니다.



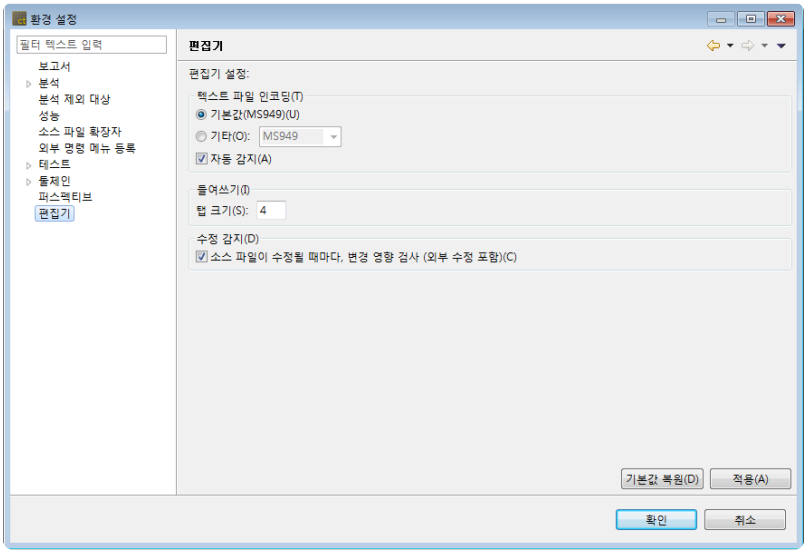
13.9. 퍼스펙티브

퍼스펙티브와 관련된 설정을 변경할 수 있습니다.



13.10. 편집기

편집기와 관련된 설정을 변경할 수 있습니다.



텍스트 파일 인코딩

텍스트 파일을 편집기에서 열 때 사용할 인코딩을 설정할 수 있습니다. '자동 감지' 옵션이 켜져 있으면, 텍스트 파일의 인코딩을 자동으로 감지합니다(자동으로 감지할 수 없는 경우 설정된 인코딩으로 열림).

들여쓰기

탭 크기를 변경하면, 편집기에 보이는 탭 크기가 변경됩니다.

수정 감지

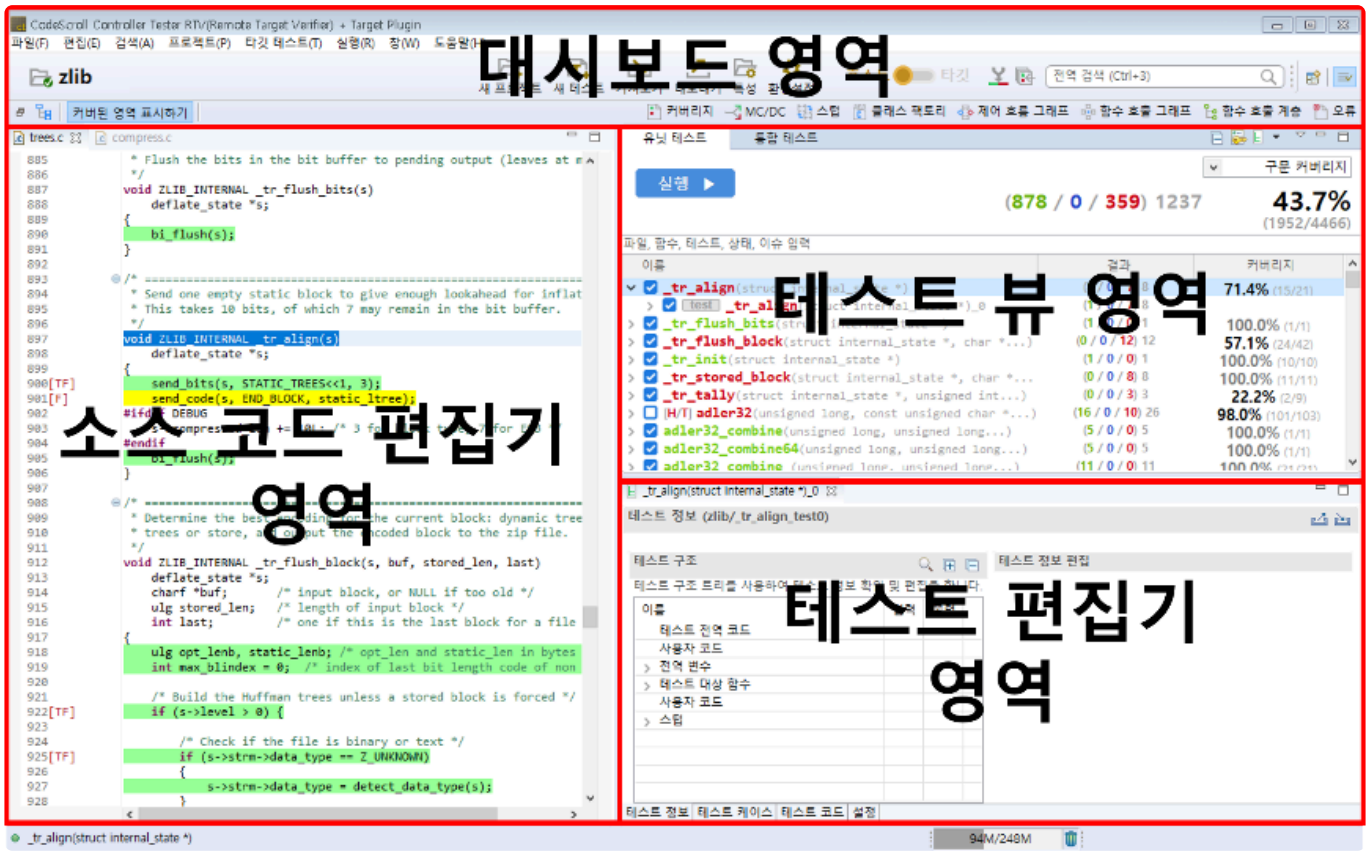
수정 감지 옵션이 켜져 있으면, 소스 파일이 수정될 때마다, 변경 영향을 검사합니다.

> 디스크 속도가 느린 경우, 이 옵션을 해제하면 성능이 향상됩니다.

14. 테스트 퍼스펙티브

테스트 퍼스펙티브는 시험 검증 작업의 흐름을 유지하고 목표에 집중할 수 있도록 필요한 정보만 노출하는 UI를 제공합니다.

Eclipse RCP로 제작되어 자유도가 높은 화면을 제공합니다. [창] 메뉴와 각 뷰에서 사용자가 설정할 수 있습니다.



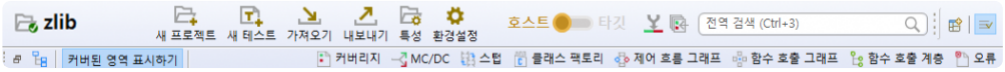
테스트 퍼스펙티브를 구성하는 요소는 다음과 같습니다.

- 대시보드
- 테스트 네비게이터 뷰
- 소스 코드 편집기
- 유닛 테스트 뷰
- 통합 테스트 뷰
- 커버리지 뷰
- MC/DC 뷰
- 스텁 뷰
- 클래스 팩토리 뷰
- 제어 흐름 그래프 뷰
- 함수 호출 그래프 뷰

- 함수 호출 계층 뷰
- 오류 뷰
- 결함 주입 뷰
- 입출력 데이터 그래프 뷰
- 콘솔 뷰

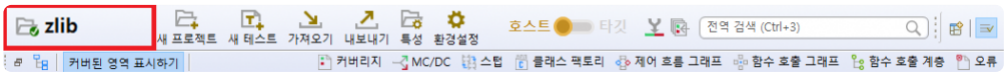
14.1. 대시보드

대시보드는 자주 사용되는 기능이나 현재 진행중인 테스트와 관련된 요약정보를 제공합니다.



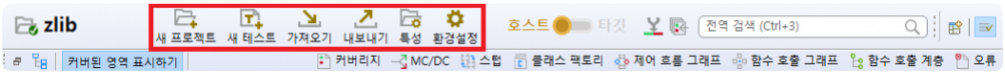
프로젝트 이름

테스트 네비게이터 뷰에서 선택한 프로젝트의 이름을 표시합니다.



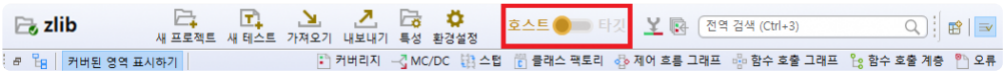
북마크 메뉴

프로젝트 생성, 테스트 생성, 가져오기, 내보내기, 프로젝트 특성, 환경 설정 기능을 사용할 수 있습니다.



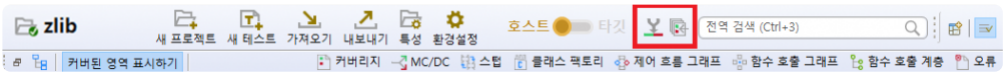
호스트/타깃 설정 메뉴

테스트 환경을 호스트 또는 타깃으로 변경할 수 있습니다.



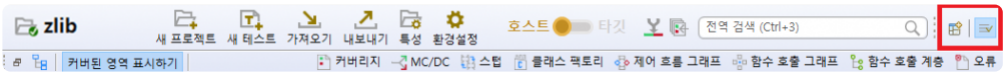
커버리지 관련 메뉴

호스트/타깃 병합 커버리지 보기, 전체 커버리지 보기(외부 커버리지 포함) 기능을 사용할 수 있습니다.



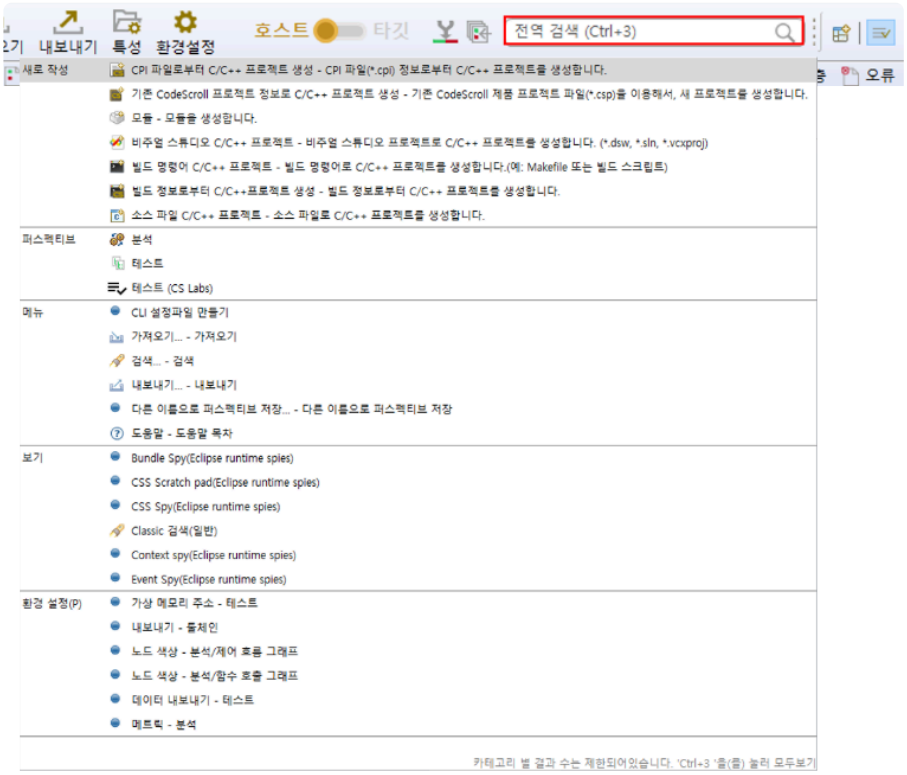
퍼스펙티브 열기 메뉴

Controller Tester에 존재하는 퍼스펙티브를 선택하여 열 수 있습니다.



전역 검색

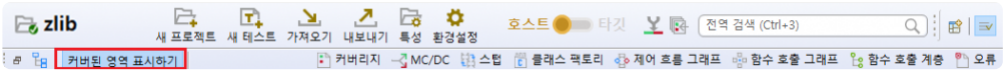
전역 검색은 새로 작성, 보기, 편집, 메뉴, 특성, 환경 설정, 프로젝트 리소스(소스 파일, 함수) 테스트, 스텁, 클래스 팩토리 등을 키워드로 검색할 수 있습니다. 전역 검색 기능은 전역 검색 창을 선택하거나, 단축키(Ctrl+3)를 통해 사용할 수 있습니다.



> 프로젝트의 리소스(소스 파일, 함수 등), 테스트, 스텁, 클래스 팩토리 는 분석된 프로젝트를 선택하면 검색할 수 있습니다.

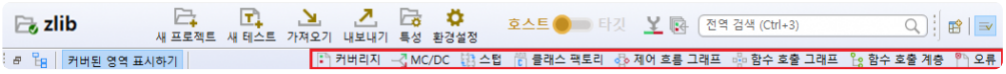
커버된 영역 표시하기

소스 코드 편집기 창에 커버된 영역을 표시합니다.



테스트 관련 뷰 열기/닫기 메뉴

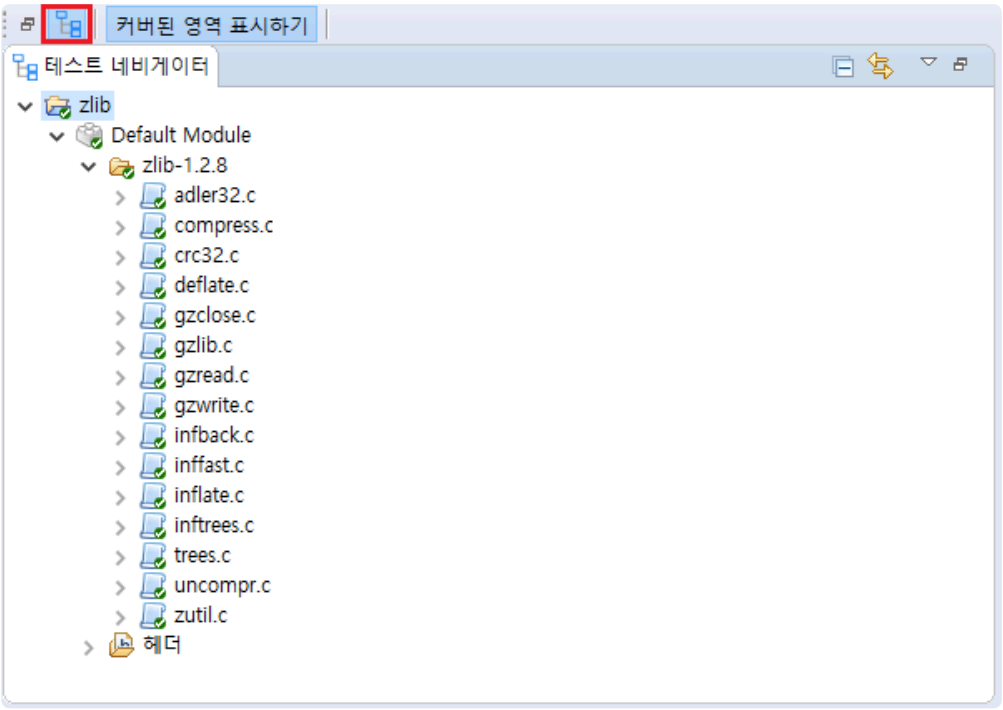
커버리지 뷰, MC/DC뷰, 스텁 뷰, 클래스 팩토리 뷰, 제어 흐름 그래프 뷰, 함수 호출 그래프 뷰, 함수 호출 계층 뷰, 오류 뷰를 열고 닫을 수 있습니다.



14.2. 테스트 네비게이터 뷰

테스트 네비게이터 뷰는 워크스페이스에 속한 프로젝트와 프로젝트 하위 테스트 모델들의 계층구조를 보여줍니다.

영역을 차지하지 않고 최소화 상태를 유지하는 것이 기본 설정입니다. 대시보드의 [테스트 네비게이터] 메뉴 선택 시, 테스트 네비게이터 뷰는 소스 코드 편집기 영역 위에 위치합니다.



아이콘 설명

아이콘	설명(*: 분석 이후 생성됨)
	열린 프로젝트
	닫힌 프로젝트
	모듈
	소스 파일(번역 단위)
	*전역 변수
	*함수
	*헤더 폴더
	*헤더 파일

	*시스템 헤더 파일

툴바 메뉴

메뉴	설명
다른 뷰와 연결	다른 뷰에서 선택한 항목을 강조
모두 접기	모든 트리 노드를 접음

아이콘 오버레이 설명

아이콘 오버레이	설명
	분석됨
	분석 이후 변경됨

소스 파일을 다른 모듈에 복사하기

소스 파일을 다른 모듈에 복사하려면, Ctrl 키를 누른 상태로 복사할 소스 파일을 끌어서 대상 모듈에 놓습니다. 다른 방법은, 복사할 소스 파일을 오른쪽 클릭한 뒤에 [복사]를 선택하고, 대상 모듈을 오른쪽 클릭한 뒤에 [붙여넣기]를 선택하는 것입니다(바로 가기 키인 Ctrl+C 및 Ctrl+V 사용 가능).

소스 파일을 다른 모듈로 이동하기

소스 파일을 다른 모듈로 이동하려면, 이동할 소스 파일을 끌어서 대상 모듈에 놓습니다.

Windows 탐색기에서 끌어서 놓기

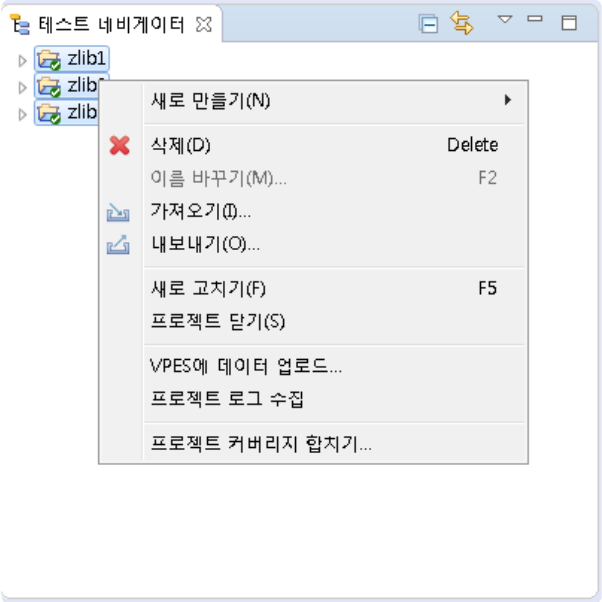
- **비주얼 스튜디오 프로젝트로 프로젝트 생성**
  Windows 탐색기에서 비주얼 스튜디오 프로젝트 파일(.dsw, .sln, .vcxproj, .vcproj)을 끌어서, 테스트 네비게이터 뷰에 놓습니다.
- **파일 시스템에서 소스 파일 가져오기**
  Windows 탐색기에서 소스 파일을 끌어서, 테스트 네비게이터 뷰의 대상 프로젝트 또는 모듈에 놓습니다.

14.2.1. 프로젝트 커버리지 합치기

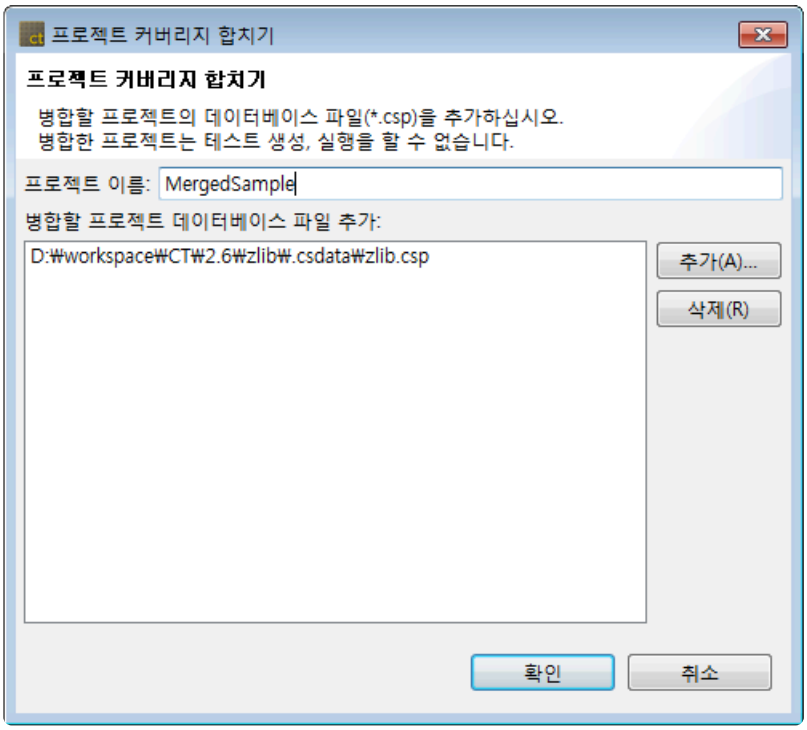
프로젝트 크기가 너무 커서 한 번에 테스트 실행을 하기 어려운 경우, 여러 개의 프로젝트로 나누어 테스트를 실행할 수 있습니다. [프로젝트 커버리지 합치기] 기능은 하나의 프로젝트를 다수의 프로젝트로 나누어 테스트를 실행한 후, 이 프로젝트들의 테스트 실행 결과를 병합하여 커버리지 결과를 전체적으로 보여줍니다.

프로젝트 커버리지 합치기

1. 커버리지 결과를 병합하려는 프로젝트를 모두 선택한 후, 마우스를 우 클릭하여 [프로젝트 커버리지 합치기]를 선택합니다.

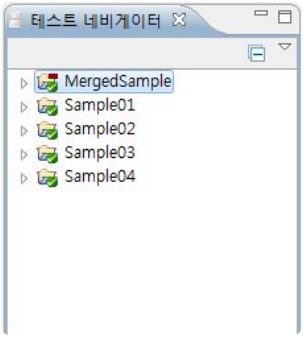


2. [프로젝트 커버리지 합치기] 메뉴를 선택하면 프로젝트의 이름을 작성하는 창이 열립니다. [프로젝트 이름]에 생성하려는 프로젝트 이름을 작성합니다.



- 테스트 네비게이터에 나타나지 않은 프로젝트에 대해서는 외부 프로젝트 추가 기능을 통해 프로젝트 결과 합치기 기능을 수행할 수 있습니다.

3. [확인] 버튼을 클릭하면 테스트 네비게이터 뷰에 프로젝트 결과가 병합된 새로운 프로젝트가 생성된 것을 확인할 수 있습니다. 병합된 프로젝트는 우측 상단에 빨간색 마크가 추가되어 표시됩니다.



4. 프로젝트 결과가 병합된 프로젝트를 선택 후 커버리지 뷰를 확인하면 전체(test01, test02, test03, test04) 커버리지를 확인할 수 있습니다.

커버리지

'Sample01' 프로젝트의 호스트 커버리지 정보를 보여줍니다.

	대상 함수	구문	분기	MC/DC	함수 호출	함수
1	function1()	75.00% (...	50.00% (...	0.00% (0...	N/A	Y
2	function2()	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
3	function3()	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
4	function4()	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
합계		**16.66% (...**	**10.00% (...**	**0.00% (0...**	**N/A**	**25.00...**

타깃 환경에서는 Asm 코드가 포함된 함수의 커버리지를 측정할 수 없습니다.

커버리지

'Sample02' 프로젝트의 호스트 커버리지 정보를 보여줍니다.

	대상 함수	구문	분기	MC/DC	함수 호출	함수
1	function1()	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
2	function2()	83.33% (...	50.00% (...	0.00% (0...	N/A	Y
3	function3()	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
4	function4()	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
합계		**13.88% (...**	**5.00% (1...**	**0.00% (0...**	**N/A**	**25.00...**

타깃 환경에서는 Asm 코드가 포함된 함수의 커버리지를 측정할 수 없습니다.

커버리지

'Sample03' 프로젝트의 호스트 커버리지 정보를 보여줍니다.

	대상 함수	구문	분기	MC/DC	함수 호출	함수
1	function1()	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
2	function2()	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
3	function3()	60.00% (...	33.33% (...	0.00% (0...	N/A	Y
4	function4()	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
합계		**16.66% (...**	**10.00% (...**	**0.00% (0...**	**N/A**	**25.00...**

타깃 환경에서는 Asm 코드가 포함된 함수의 커버리지를 측정할 수 없습니다.

커버리지

'Sample04' 프로젝트의 호스트 커버리지 정보를 보여줍니다.

	대상 함수	구문	분기	MC/DC	함수 호출	함수
1	function1()	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
2	function2()	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
3	function3()	0.00% (0...	0.00% (0...	0.00% (0...	N/A	N
4	function4()	50.00% (...	25.00% (...	0.00% (0...	N/A	Y
합계		**16.66% (...**	**10.00% (...**	**0.00% (0...**	**N/A**	**25.00...**

타깃 환경에서는 Asm 코드가 포함된 함수의 커버리지를 측정할 수 없습니다.

커버리지 오류 검색

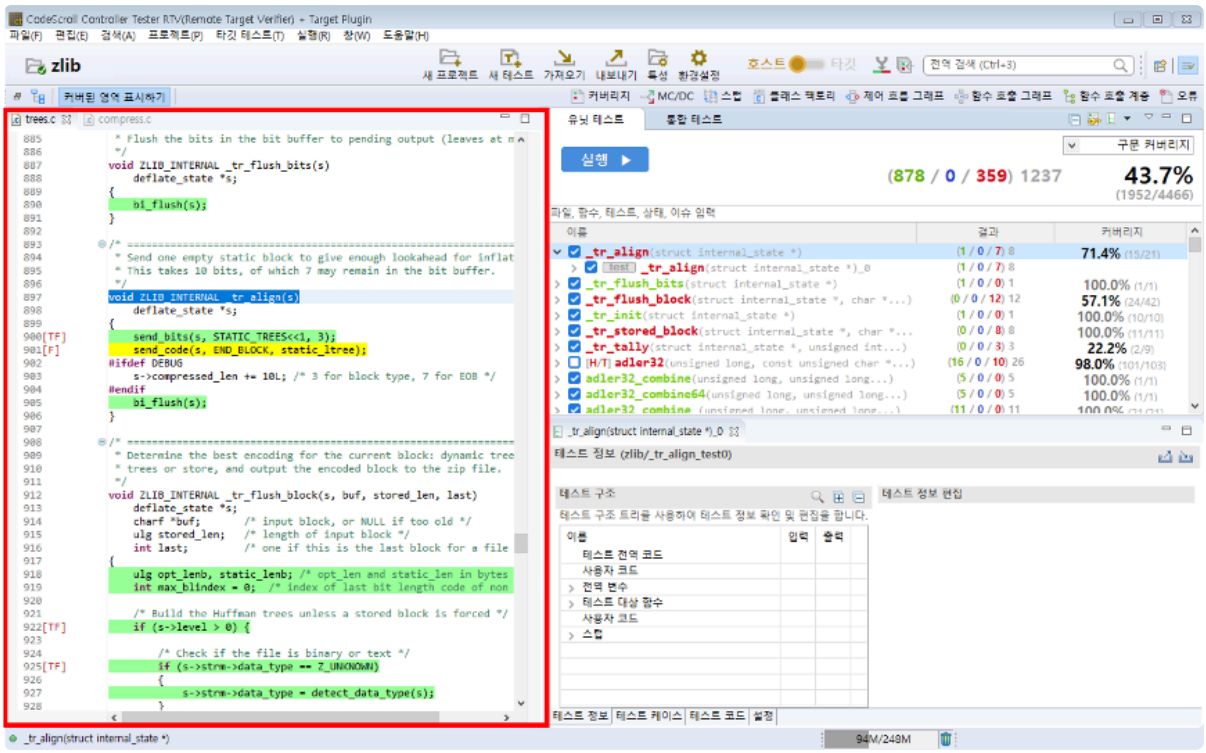
'MergedSample' 프로젝트의 호스트 커버리지 정보를 보여줍니다.

	대상 함수	구문	분기	MC/DC	함수 호출	함수
1	function1()	75.00% (...	50.00% (...	0.00% (0...	N/A	Y
2	function2()	83.33% (...	50.00% (...	0.00% (0...	N/A	Y
3	function3()	60.00% (...	33.33% (...	0.00% (0...	N/A	Y
4	function4()	50.00% (...	25.00% (...	0.00% (0...	N/A	Y
합계		**63.88% (...**	**35.00% (...**	**0.00% (0...**	**N/A**	**100.0...**

타깃 환경에서는 Asm 코드가 포함된 함수의 커버리지를 측정할 수 없습니다.

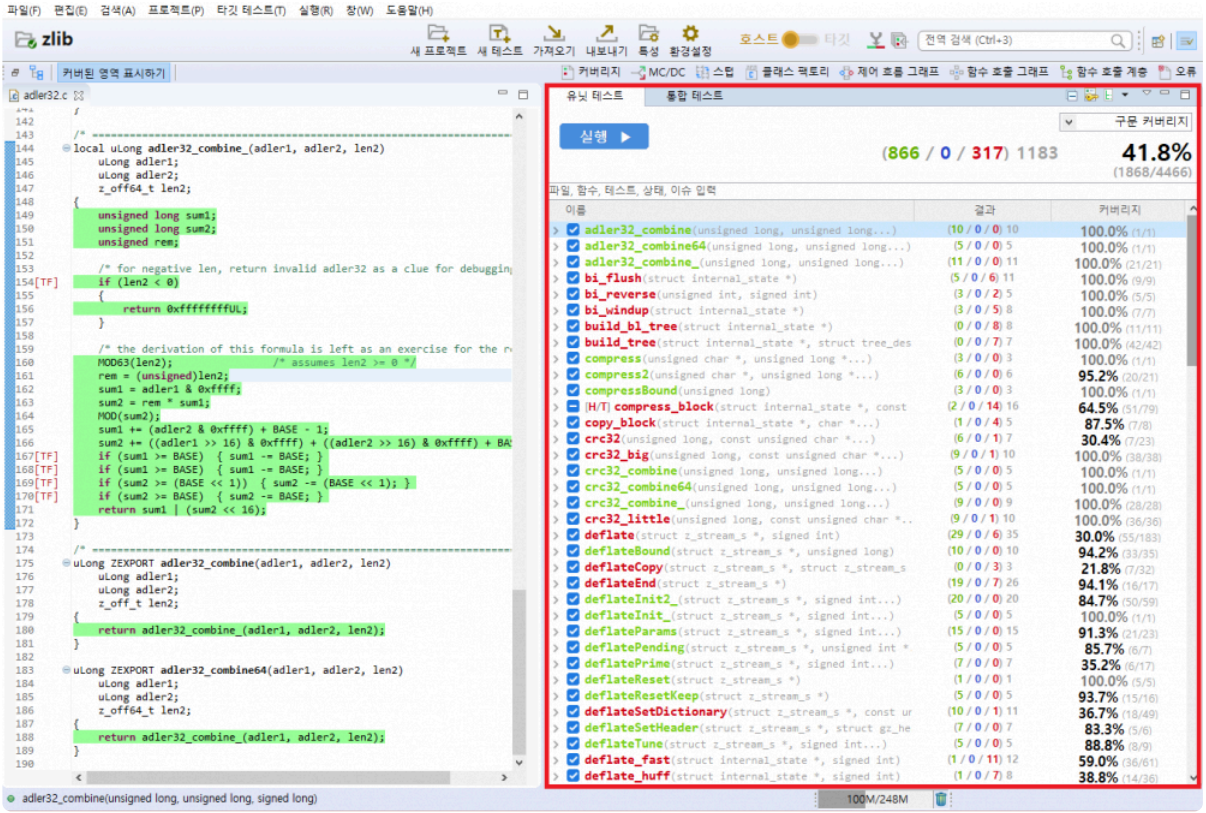
14.3. 소스 코드 편집기

소스 코드 편집기는 기본으로 화면 왼쪽에 위치합니다.



14.4. 유닛 테스트 뷰

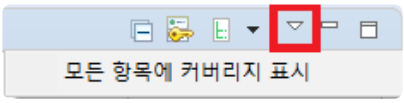
유닛 테스트 뷰는 기본으로 화면 오른쪽에 위치합니다. 유닛 테스트 뷰는 테스트와 커버리지 정보를 통합하여 하나의 뷰로 제공합니다.



유닛 테스트 뷰의 툴바 메뉴

툴바 아이콘	설명		
모두 접기	테스트 뷰 트리 모두 접기		
고유한 테스트 이름으로 보기	고유한 테스트 이름으로 보기	합계	전체 테스트 보기
실패/오류	테스트 결과가 실패 또는 오류인 테스트만 보기		
실패	테스트 결과가 실패인 테스트만 보기		
오류	테스트 결과가 오류인 테스트만 보기		
성공	테스트 결과가 성공인 테스트만 보기		
함수 변경	테스트 대상 함수가 변경된 테스트만 보기		
실행 보장 안됨	테스트 실행 보장 안되는 테스트만 보기		
호스트/타깃 결과 다름	호스트/타깃 결과가 다른 테스트만 보기		

유닛 테스트 뷰의 풀다운 메뉴(▽)



메뉴 이름	설명
모든 항목에 커버리지 표시	모든 항목(함수, 테스트, 테스트 케이스)에 커버리지 표시

유닛 테스트 대시보드

메뉴	설명
실행 ▶	유닛 테스트 실행 버튼
(878 / 0 / 359) 1237	테스트 케이스의의 성공, 실패, 오류, 총합 표현
구문 커버리지	유닛 테스트 뷰에 표시할 커버리지 종류(구문, 분기, MC/DC, 함수 호출, 함수) 선택
43.7% (1952/4466)	선택한 종류의 전체 커버리지 표현

검색

유닛 테스트 뷰에서는 파일, 함수, 테스트, 상태(성공, 실패, 오류), 이슈의 이름을 통해 함수, 테스트, 테스트 케이스를 검색할 수 있습니다.

상태 검색 키워드

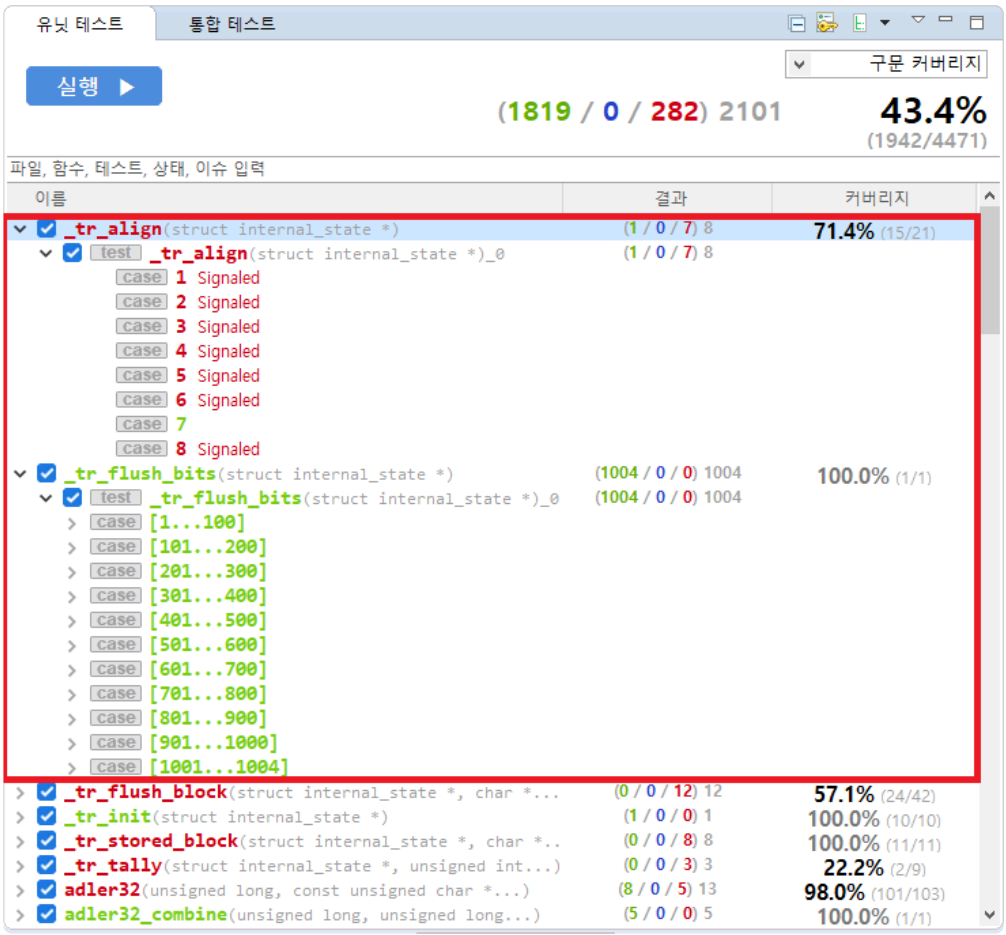
키워드	설명
%TEST_SUCCESS%	테스트 성공

%TEST_FAILURE% %TEST_ERROR%	테스트 실패/오류
%TEST_FAILURE%	테스트 실패
%TEST_ERROR%	테스트 오류
%TEST_HAS_NOT_FUNCTION%	함수 변경
%TEST_NOT_GUARANTEE%	실행 보장 안됨
%TEST_RESULT_DIFFERENT%	호스트/타깃 결과 다름

'실행 보장 안됨'은 테스트 코드를 생성했지만 내부적으로 정상 실행이 될 지 여부를 보장할 수 없는 경우입니다. 특정하기 어려운 타입을 파라미터나 반환값으로 사용하는 경우입니다.

유닛 테스트 뷰의 구조

유닛 테스트 뷰는 [함수]-[테스트]-[테스트 케이스]의 계층 구조로 표현합니다. 하나의 테스트에 100개 이상의 테스트 케이스가 존재할 경우, 100개씩 묶어서 그룹으로 표현합니다.



유닛 테스트 뷰의 항목 아이콘

항목 아아이콘	설명
아이콘 없음	함수
test	테스트
case	테스트 케이스, 테스트 케이스 그룹

유닛 테스트 뷰의 항목 상태 표현

유닛 테스트 뷰의 함수, 테스트, 테스트 케이스는 테스트 실행 여부 및 실행 결과를 색상으로 표현합니다.

색상	설명
녹색	함수/테스트: 하위에 모든 테스트 케이스의 실행 결과가 성공인 경우 테스트 케이스: 실행 결과가 성공인 경우
파란색	함수/테스트: 하위에 실행 결과가 실패인 테스트 케이스가 존재하고 오류인 테스트 케이스는 존재하지 않음 테스트 케이스: 실행 결과가 실패인 경우
붉은색	함수/테스트: 하위에 실행 결과가 오류인 테스트 케이스가 존재함 테스트 케이스: 실행 결과가 오류인 경우
주황색	함수/테스트: 하위에 모든 테스트 케이스를 실행하지 않았고 실행 보장 안 됨 테스트 케이스가 존재함 테스트 케이스: 실행 보장 안 됨
검정색	함수/테스트: 하위에 모든 테스트 케이스를 실행하지 않음 테스트 케이스: 실행하지 않음

함수 노드

함수 노드는 함수의 이름, 커버리지, 테스트 케이스 수행 결과(성공, 실패,오류, 합계)를 제공합니다.

> 각 함수 노드의 커버리지는 함수 간 호출 관계에 따라 다른 함수 노드의 테스트 수행 결과가 반영될 수 있습니다.

함수 노드를 더블 클릭하여 해당 함수의 위치를 소스 코드 편집기 창에서 확인할 수 있습니다.

유닛 테스트 통합 테스트

실행 ▶

구문 커버리지

(1819 / 0 / 282) 2101 43.4% (1942/4471)

파일, 함수, 테스트, 상태, 이슈 입력

이름	결과	커버리지
_tr_align(struct internal_state *)	(1 / 0 / 7) 8	71.4% (15/21)
test _tr_align(struct internal_state *)_0	(1 / 0 / 7) 8	
case 1 Signaled		
case 2 Signaled		
case 3 Signaled		
case 4 Signaled		
case 5 Signaled		
case 6 Signaled		
case 7		
case 8 Signaled		
_tr_flush_bits(struct internal_state *)	(1004 / 0 / 0) 1004	100.0% (1/1)
test _tr_flush_bits(struct internal_state *)_0	(1004 / 0 / 0) 1004	
case [1...100]		
case [101...200]		
case [201...300]		
case [301...400]		
case [401...500]		
case [501...600]		
case [601...700]		
case [701...800]		
case [801...900]		
case [901...1000]		
case [1001...1004]		
_tr_flush_block(struct internal_state *, char *...	(0 / 0 / 12) 12	57.1% (24/42)
_tr_init(struct internal_state *)	(1 / 0 / 0) 1	100.0% (10/10)
_tr_stored_block(struct internal_state *, char *..	(0 / 0 / 8) 8	100.0% (11/11)
_tr_tally(struct internal_state *, unsigned int...)	(0 / 0 / 3) 3	22.2% (2/9)
adler32(unsigned long, const unsigned char *...)	(8 / 0 / 5) 13	98.0% (101/103)
adler32_combine(unsigned long, unsigned long...)	(5 / 0 / 0) 5	100.0% (1/1)

함수 노드의 컨텍스트 메뉴

컨텍스트 메뉴	설명
테스트 생성	선택된 함수의 테스트 생성
테스트 복사	선택된 함수의 테스트 복사
삭제	선택된 함수의 테스트 및 케이스 삭제
함수 호출 그래프 보기	선택된 함수의 함수 호출 그래프 보기
제어 흐름 그래프 보기	선택된 함수의 제어 흐름 그래프 보기
MC/DC 보기	선택된 함수의 MC/DC 보기
호스트 출력값 -> 기대값	기대값에 호스트 출력값을 붙여넣기
타깃 출력값 -> 기대값	기대값에 타깃 출력값을 붙여넣기
연관 파일 설정	해당 테스트가 속한 파일 선택
연관 이슈 설정	선택된 테스트를 이슈 관리 도구의 이슈와 연관
테스트 데이터 가져오기	로컬에 저장된 테스트 데이터 가져오기
테스트 데이터 내보내기	로컬로 테스트 데이터 내보내기
기존 스텁 추가	선택된 테스트에 스텁 추가
함수 선택	마우스로 선택된 모든 함수의 실행 체크박스를 선택으로 변경

함수 선택 해제	마우스로 선택된 모든 함수의 실행 체크박스를 해제로 변경
선택한 테스트 커버리지 내보내기	선택한 테스트 커버리지를 선택한 경로로 내보내는 기능

테스트 노드

테스트 노드는 커버리지, 테스트 케이스 수행 결과(성공, 실패,오류, 합계)를 제공합니다.
테스트 노드를 더블 클릭하여 해당 테스트 편집기 를 열 수 있습니다.

테스트 노드의 컨텍스트 메뉴

컨텍스트 메뉴	설명
테스트 케이스 추가	선택된 유닛 테스트에 테스트 케이스 추가
테스트 케이스 자동 생성	다양한 방법으로 테스트 케이스를 생성
호스트 출력값 -> 기대값	기대값에 호스트 출력값을 붙여넣기
타깃 출력값 -> 기대값	기대값에 타깃 출력값을 붙여넣기
테스트 케이스 초기화	모든 테스트 케이스 삭제

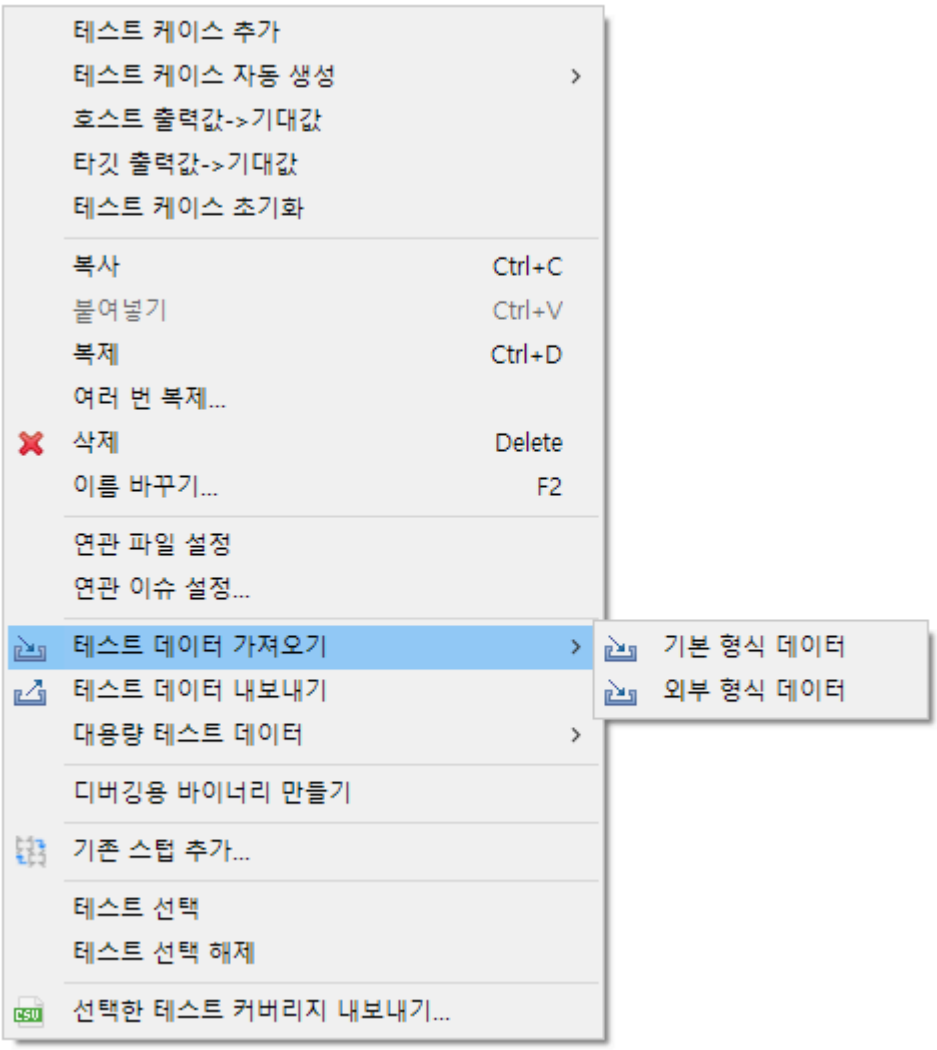
복사	테스트 및 케이스 복사
붙여넣기	테스트 및 케이스 붙여넣기
복제	테스트 및 케이스 복제
여러 번 복제	테스트 및 케이스 입력한 수만큼 복제
삭제	테스트 및 케이스 삭제
이름 바꾸기	테스트 이름 변경
연관 파일 설정	해당 테스트가 속한 파일 선택
연관 이슈 설정	선택된 테스트를 이슈 관리 도구의 이슈와 연관
테스트 데이터 가져오기	로컬에 저장된 테스트 데이터 가져오기
테스트 데이터 내보내기	로컬로 테스트 데이터 내보내기
대용량 테스트 데이터	로컬로 내보내거나 사용자가 작성한 파일을 대상 테스트에 테스트 케이스로 등록하는 기능
기존 스텁 추가	선택된 테스트에 스텁 추가
테스트 선택	마우스로 선택된 모든 테스트의 실행 체크박스를 선택으로 변경
테스트 선택 해제	마우스로 선택된 모든 테스트의 실행 체크박스를 해제로 변경
선택한 테스트 커버리지 내보내기	선택한 테스트 커버리지를 선택한 경로로 내보내는 기능

- 테스트 데이터 내보내기 규칙
  테스트 데이터를 내보내기 하려는 곳에 이전에 내보내기 했던 같은 테스트의 테스트 데이터가 있을 경우 자동으로 파일에 넘버링을 하여 테스트 데이터를 생성합니다.
  넘버링 규칙은 “테스트 이름_#No.” 입니다.
  예) test_1_test0.csv, test_1_test0_0.csv, test_1_test1.csv ……

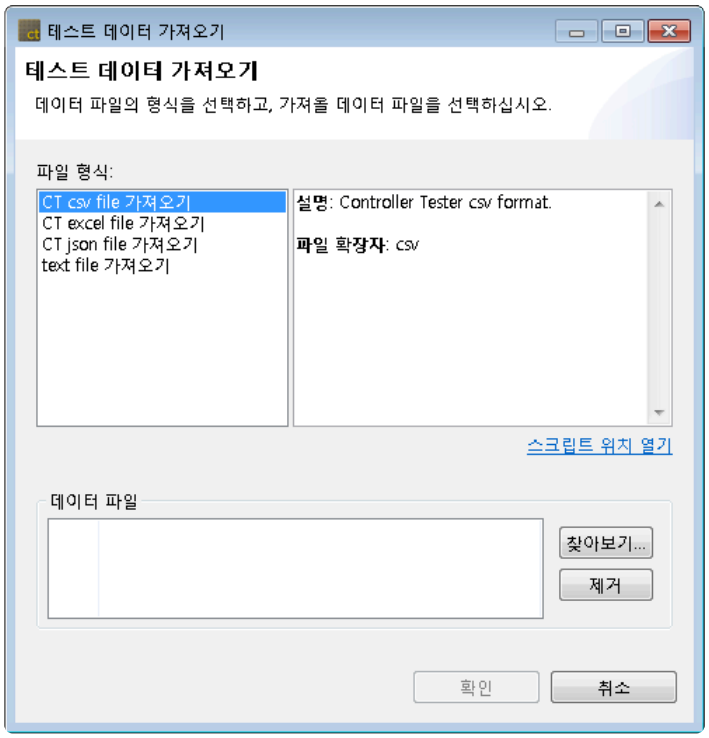
- 테스트 데이터 가져오기 규칙
  테스트 데이터를 가져올 때 해당 테스트의 이름을 가지고 넘버링 된 다수의 테스트 데이터를 선택하면 해당 파일을 병합하여 가져옵니다.

테스트 데이터 가져오기

다양한 형식(csv, xlsx, txt, json)의 테스트 데이터를 가져올 수 있습니다.

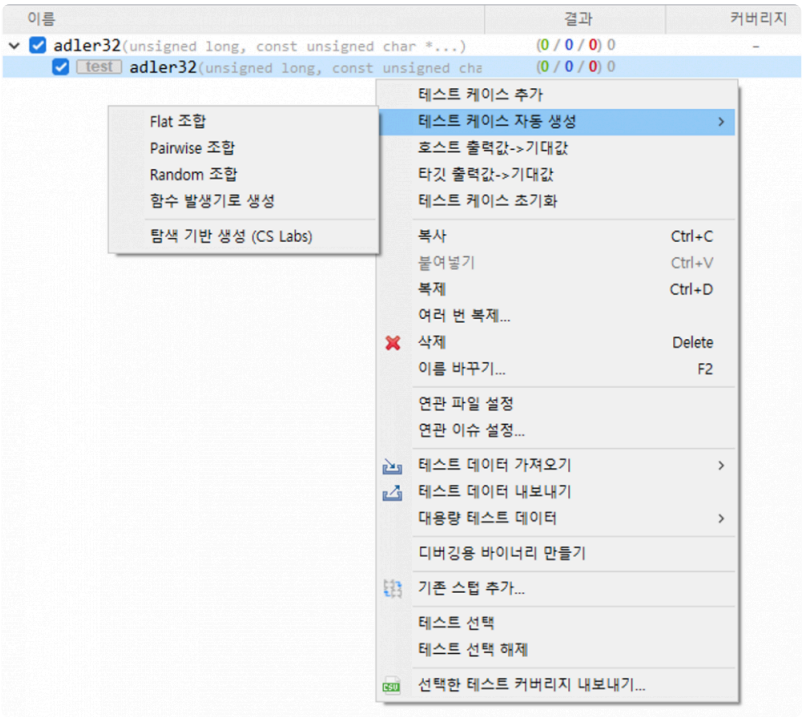


1. 유닛 테스트 뷰의 컨텍스트 메뉴 중 [테스트 데이터 가져오기]를 클릭 후, [기본 형식 데이터], [외부 형식 데이터] 중 하나를 선택합니다.
2. [기본 형식 데이터]를 선택할 경우, Controller Tester에서 내보낸 형식의 테스트 데이터를 가져옵니다.
3. [외부 형식 데이터]를 선택할 경우, 다양한 형식(csv, xlsx, txt, json)의 파일로부터 테스트 데이터를 가져옵니다.



테스트 케이스 자동 생성

Flat/Pairwise/Random 조합, 함수 발생기로 생성, 탐색 기반 생성 (CS Labs)를 사용하여 테스트 케이스를 생성합니다.

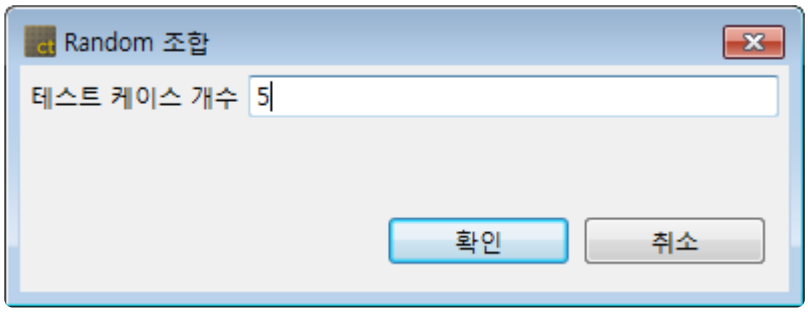


1. 유닛 또는 통합 테스트 뷰의 컨텍스트 메뉴 중 [테스트 케이스 자동 생성]을 클릭 후, [Flat 조합], [Pairwise 조합], [Random 조합], [함수 발생기로 생성], [탐색 기반 생성 (CS Labs)] 중 한 가지 테스트 케이스 자동

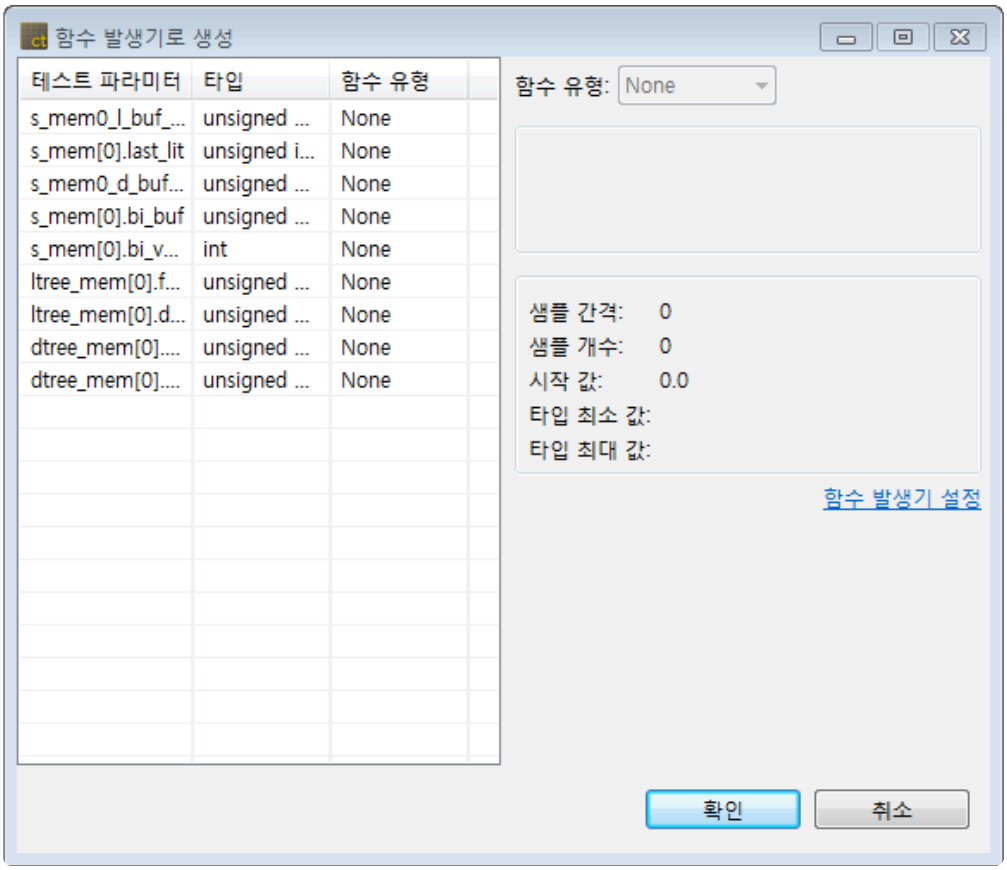
생성 방식을 선택합니다.

방법	설명
FLAT	테스트 데이터 개수가 가장 많은 변수를 기준으로 단순 조합
PAIR WISE	각 선출된 파라미터의 데이터가 자신 이외의 파라미터 데이터와 적어도 한 번씩은 쌍을 이루도록 조합 ※ 변수 파티션 개수는 2보다 크고 52보다 작아야 함
RANDOM	입력 파라미터의 변수 파티션의 최소값과 최대값 사이의 임의의 값을 사용자가 정의한 테스트 케이스 수만큼 테스트 데이터 조합
함수 발생기로 생성	6가지 함수 유형(Ramps, Random, Range, Sine, Toggle, Single Value)을 사용하여 테스트 케이스 생성
탐색 기반 생성 (CS Labs)	국지 탐색(Local search) 알고리즘인 AVM (Alternating Variable Method) 기법을 통해 테스트 케이스를 생성 ※ 현재는 C 언어만 지원

2. [Random 조합]을 선택할할 경우, 사용자로부터 테스트 케이스 개수를 입력 받을 수 있도록 Random 조합 창을 보여줍니다.



3. [함수 발생기로 생성]을 선택할 경우, 함수 유형{Ramps, Sine, Random, Toggle, Range, Single value, None)을 선택할 수 있고, 함수 발생기 설정을 통해 선택된 함수의 설정 값을 변경할 수 있습니다. [프로젝트 특서 설정]에서 함수 발생기 정보를 설정할 수 있습니다.



공통 설정 값

각 함수가 공통으로 가지는 설정 값으로, [프로젝트] -> [특성]에서 값을 변경할 수 있습니다. [프로젝트 특성 설정]에서 함수 발생기 정보를 설정할 수 있습니다.

설정	설명
샘플 간격	함수로부터 샘플링 할 샘플의 간격
샘플 개수	함수로부터 샘플링 할 셈플의 개수(테스트 케이스 개수)
시작 값	함수가 시작되는 기본 값으로 해당 값을 기준으로 테스트 데이터 생성
타입 최소 값	변수 파파티션의 최소 값(해당 함수의 반환 값이 타입 최소 값보다 작은 경우 타입 최소 값 반환)
타입 최대 값	변수 파파티션의 최대 값(해당 함수의 반환 값이 타입 최대 값보다 큰 경우 타입 최대 값 반환)

Ramps 함수

Pre, Post, Hold 값을 사용하여 펄스를 생성하는 함수입니다. 샘플 개수가 함수의 주기보다 큰 경우는 함수가 재귀적으로 호출됩니다.

설정	설명
Pre delay	Pre/Post 샘플 값이 지속되는 시간
Rise samples	Pre/Post 샘플 값에서 Hold 샘플 값으로 상승하는 시간

Hold samples	Hold 샘플 값이 지속되는 시간
Fall samples	Hold 샘플 값에서 Pre/Post 샘플 값으로 떨어지는 시간
Post delay	Pre/Post 샘플 값이 지속되는 시간
Pre/Post delay value	Pre/Post delay 샘플 값
Hold value	Hold 샘플 값

Random 함수

최소 값(Min)과 최대 값(Max) 사이에서 랜덤한 값을 생성하는 함수입니다.

설정	설명
Min	랜덤 범위 최소 값
Max	랜덤 범위 최대 값

Range 함수

타입 최소 값과 최대 값 사이에 일정한 간격(Step Size)만큼 증가 혹은 감소하는 값을 생성하는 함수입니다.

설정	설명
Step size	증가 혹은 감소 값 크기
Hold	Step size가 유지되는 횟수
Rising	Step size만큼 상승되는 함수 유형 예) step size가 30일 경우, 0,30,60……
Falling Step	size만큼 감소되는 함수 유형 예) step size가 30일 경우, 100,70,40……
Alternate	함수의 결과값이 타입 최대 최소 값을 만난 경우 상승 혹은 감소로 바뀌는 함수 유형 예) step size가 30이고 최대 최소 값이 0~100인 경우, 0,30,60,90,60,30,0……

Sine 함수

Sine값을 생성하는 함수 입니다. 샘플 개수가 함수의 주기보다 큰 경우는 함수가 재귀적으로 호출됩니다.

설정	설명
Amplitude	Sine 함수의 진폭
Period	Sine 함수의 주기
Phase	Sine 함수의 위상
Offset	Sine 함수의 offset

Toggle 함수

FirstValue와 SecondValue를 반복적으로 생성하는 함수입니다.

설정	설명
First value	Toggle함수에서 반복되는 첫 번째 값
Second value	Toggle함수에서 반복되는 두 번째 값

SingleValue 함수

일정한 단일 값만 반환하는 함수입니다.

설정	설명
Value	생성할 값

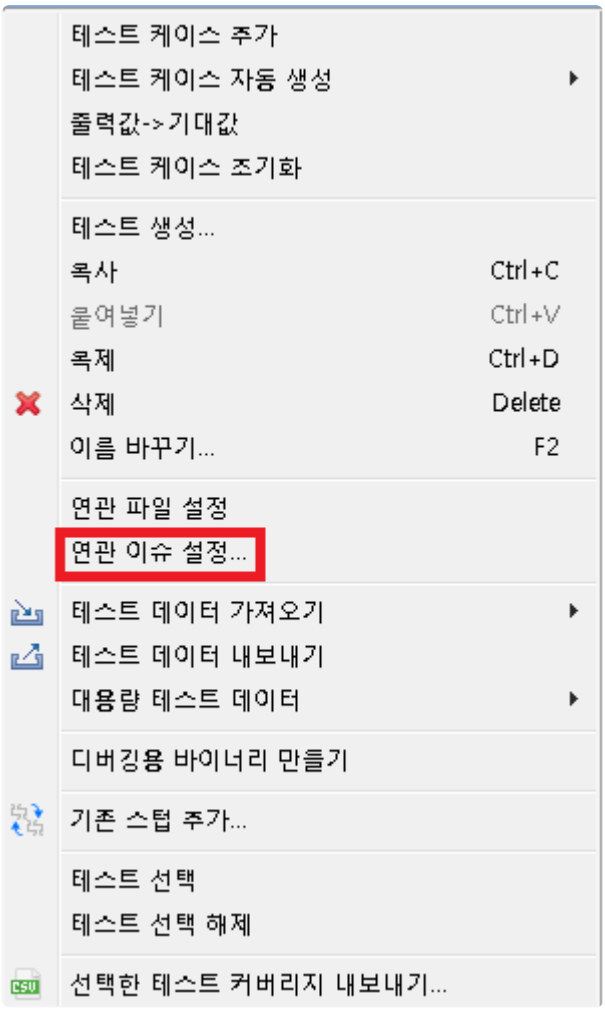
None

함수를 생성하지 않습니다.

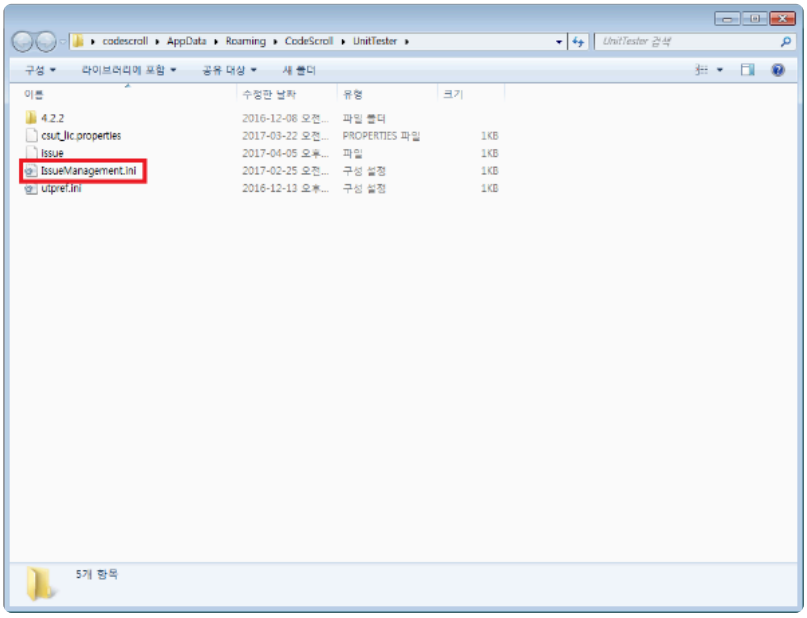
연관 이슈 설정

선택된 테스트를 등록된 이슈 관리 도구의 이슈와 연관시키는 기능을 제공합니다. Controller Tester에서 지원하는 이슈 관리 도구의 종류는 JIRA, Trac, Redmine, Mantis, Bugzilla 입니다.

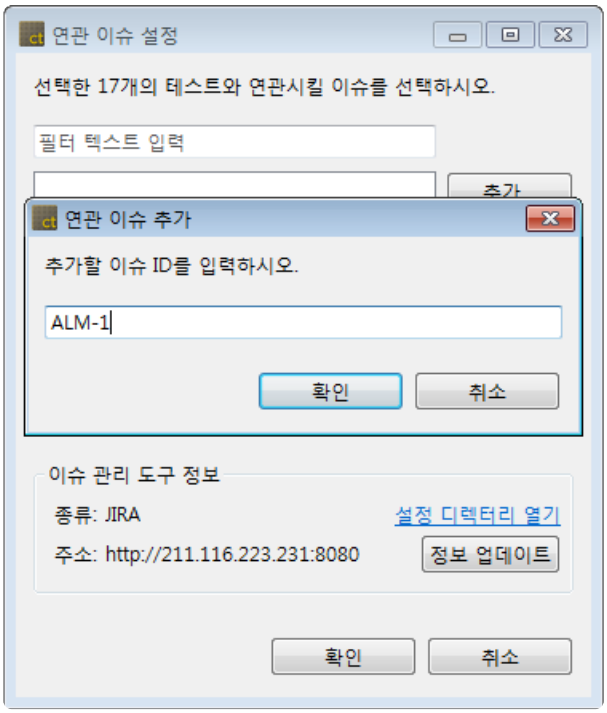
1. 유닛 테스트 뷰의 컨텍스트 메뉴 중 [연관 이슈 설정]을 클릭합니다.



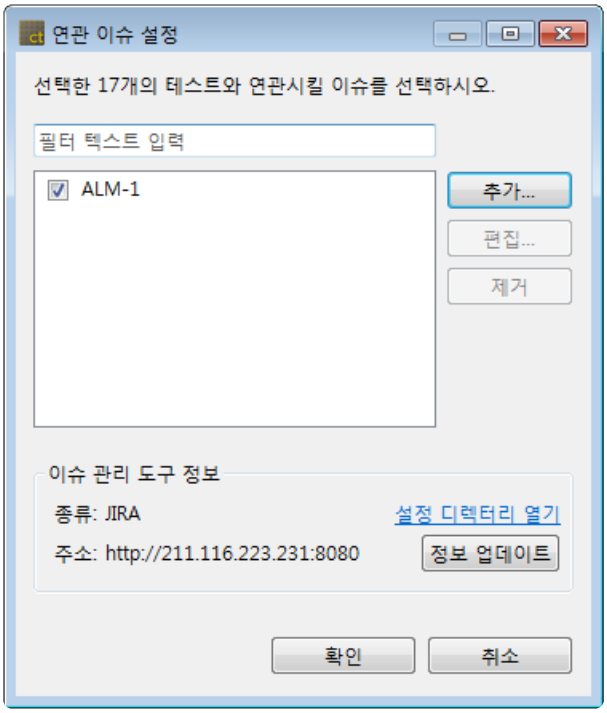
2. 이슈 관리 도구 정보를 설정 파일(IssueManagement.ini)에 입력합니다.



3. 테슽스트와 연관 시키려는 이슈를 추가합니다.

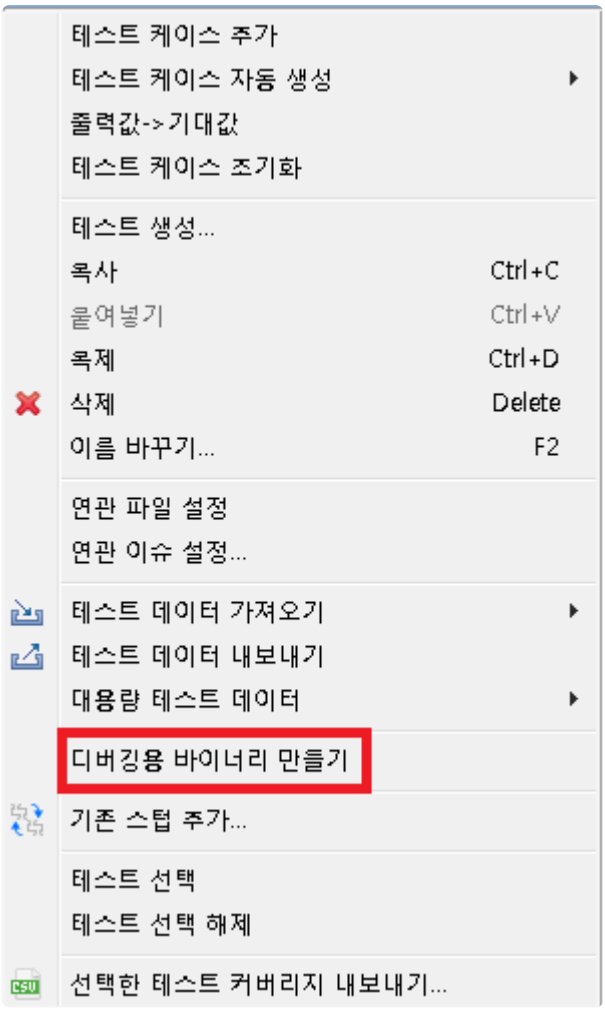


4. 체크된 이슈 목록을 확인한 뒤, [확인]을 클릭합니다.

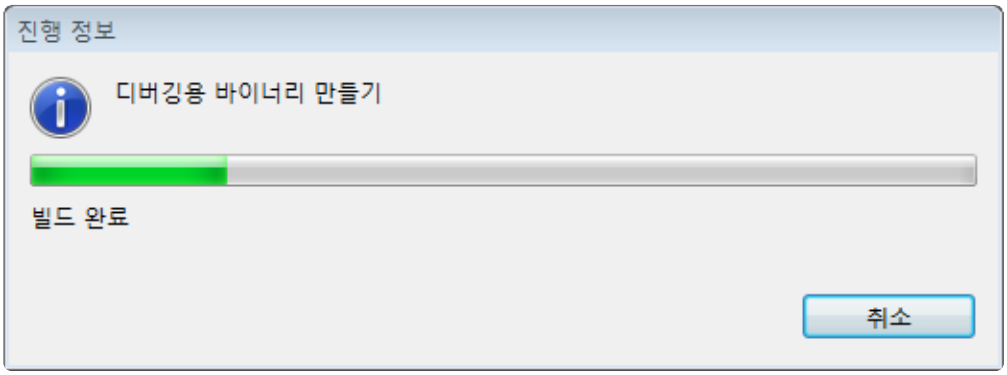


디버깅용 바이너리 만들기

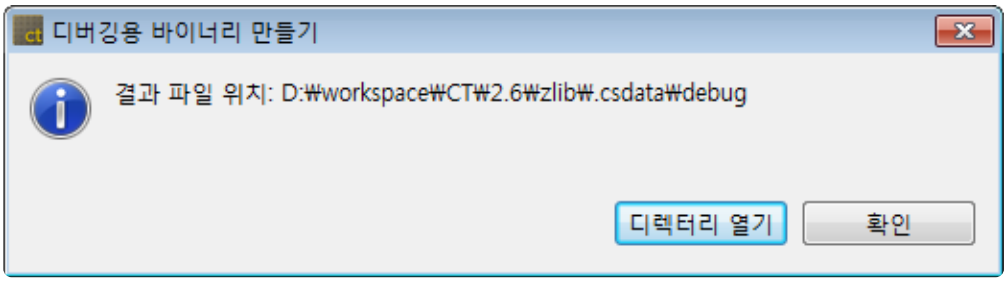
1. 유닛 또는 통합 테스트 뷰에서 테스트 선택 후 마우스 우 클릭합니다. 컨텍스트 메뉴에서 [디버깅용 바이너리 만들기] 메뉴를 선택하면 디버깅용 바이너리 만들기 기능이 수행됩니다.



2. [확인] 버튼을 클릭하면 디버깅용 바이너리 만들기 진행 정보를 확인할 수 있습니다. 또한, 결과 파일의 위치를 알려주는 창이 열립니다.



3. [디렉터리 열기] 버튼을 클릭하여 아래의 디버깅용 바이너리 만들기 결과물이 저장된 디렉터리로 이동하거나 또는 [확인] 버튼을 클릭하여 디버깅용 바이너리 만들기를 종료합니다.

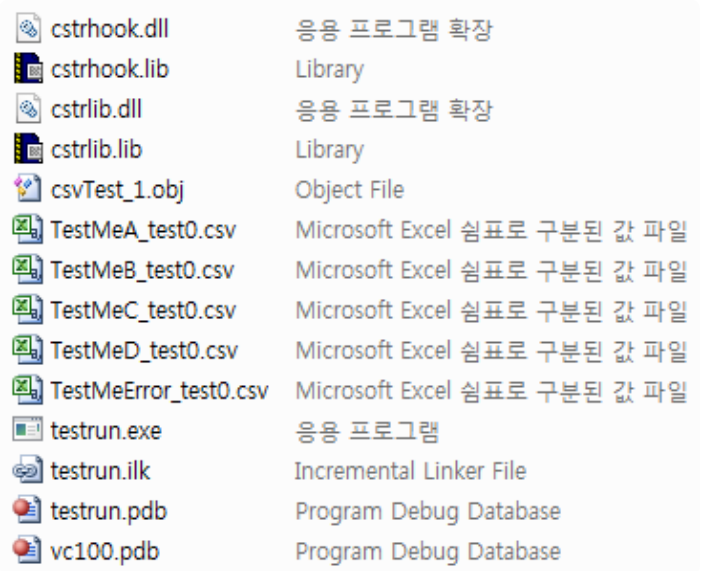


- 해당 프로젝트가 “Visual studio CL compiler”를 사용한다면 “vsjitdebugger”를 자동으로 실행 시킵니다.

디버깅용 바이너리 만들기 결과물

[디버깅용 바이너리 만들기]기능은 사용자 환경에 맞는 디버깅용 바이너리 (testrun.exe)를 생성해줍니다. 사용자는 대상 프로젝트의 툴체인에 맞는 디버깅 도구를 통해 testrun.exe 파일을 직접 디버깅할 수 있습니다.

[디버깅용 바이너리 만들기]기능 수행 시 생성되는 파일 목록은 아래 그림과 같습니다.

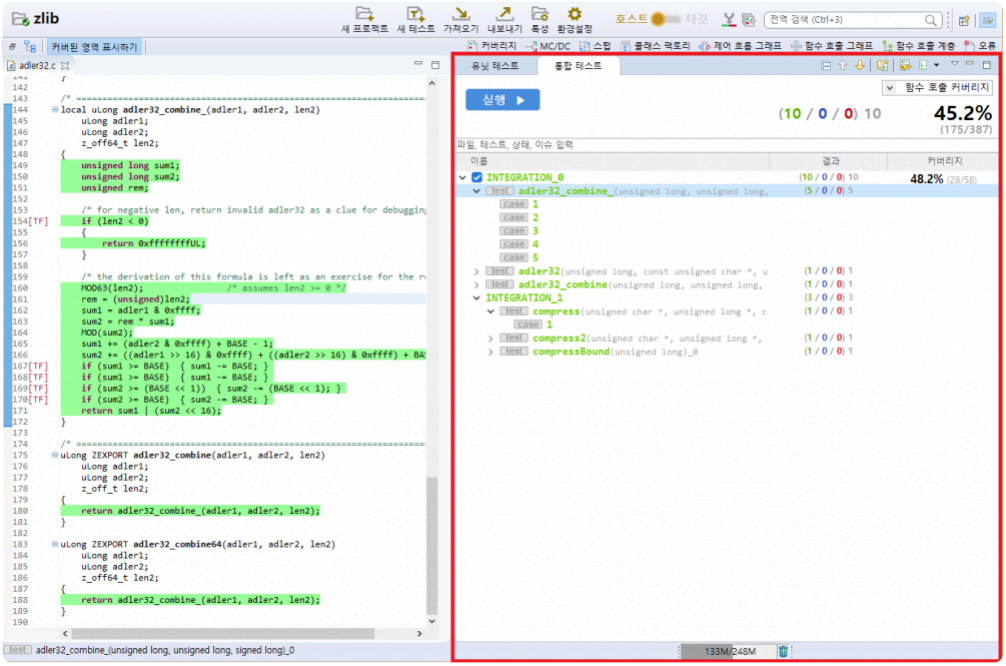


cstrhook.dll	응용 프로그램 확장
cstrhook.lib	Library
cstrlib.dll	응용 프로그램 확장
cstrlib.lib	Library
csvTest_1.obj	Object File
TestMeA_test0.csv	Microsoft Excel 쉼표로 구분된 값 파일
TestMeB_test0.csv	Microsoft Excel 쉼표로 구분된 값 파일
TestMeC_test0.csv	Microsoft Excel 쉼표로 구분된 값 파일
TestMeD_test0.csv	Microsoft Excel 쉼표로 구분된 값 파일
TestMeError_test0.csv	Microsoft Excel 쉼표로 구분된 값 파일
testrun.exe	응용 프로그램
testrun.ilk	Incremental Linker File
testrun.pdb	Program Debug Database
vc100.pdb	Program Debug Database

- Visual Studio CL컴파일러를 사용했을 때의 파일 목록
- testrun.exe: 디버깅용 바이너리
- *.csv: 테스트의 입력값

14.5. 통합 테스트 뷰

통합 테스트 뷰에서는 여러 유닛 테스트를 묶을 수 있는 단위인 [통합 테스트]를 제공합니다. 통합 테스트로 묶인 유닛 테스트들은 컨텍스트 유지 하여 순서대로 실행합니다.

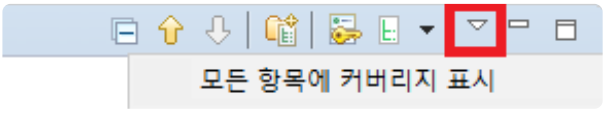


통합 테스트 뷰의 툴바 메뉴

툴바 아이콘	설명
모두 접기	테스트 뷰 트리 모두 접기
테스트 위로 이동	선택한 테스트의 순서를 위로 이동
테스트 아래로 이동	선택한 테스트의 순서를 아래로 이동
생성	통합 테스트 생성
고유한 테스트 이름으로 보기	고유한 테스트 이름으로 보기
합계	전체 테스트 보기
실패/오류	테스트 결과가 실패 또는 오류인 테스트만 보기
실패	테스트 결과가 실패인 테스트만 보기
오류	테스트 결과가 오류인 테스트만 보기
성공	테스트 결과가 성공인 테스트만 보기
함수 변경	테스트 대상 함수가 변경된 테스트만 보기

실행 보장 안됨	테스트 실행 보장 안되는 테스트만 보기
호스트/타깃 결과 다름	호스트/타깃 결과가 다른 테스트만 보기

통합 테스트 뷰의 풀다운 메뉴(▽)



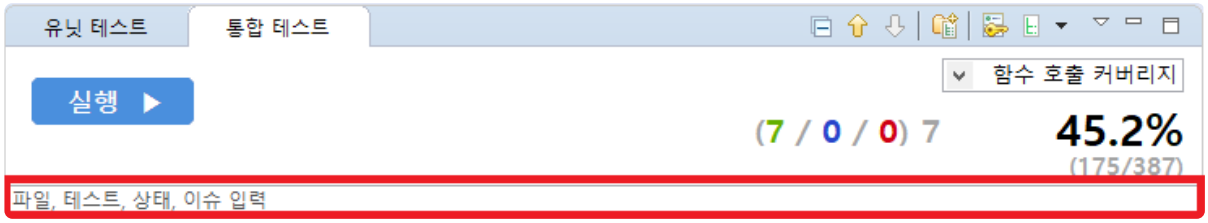
메뉴 이름	설명
모든 하항목에 커버리지 표시	모든 항목(통합 테스트, 테스트, 테스트 케이스)에 커버리지 표시

통합 테스트 대시보드

메뉴	설명
실행 ▶	통합 테스트 실행 버튼
(878 / 0 / 359) 1237	테스트 케이스의의 성공, 실패, 오류, 총합 표현
구문 커버리지	통합 테스트 뷰에 표시할 커버리지 종류(구문, 분기, MC/DC, 함수 호출, 함수) 선택
43.7% (1952/4466)	선택한 종류의 전체 커버리지 표현

검색

통합 테스트 뷰는 파일, 함수, 테스트, 상태(성공, 실패, 오류), 이슈의 이름을 통해 통합 테스트, 테스트, 테스트 케이스를 검색할 수 있습니다.

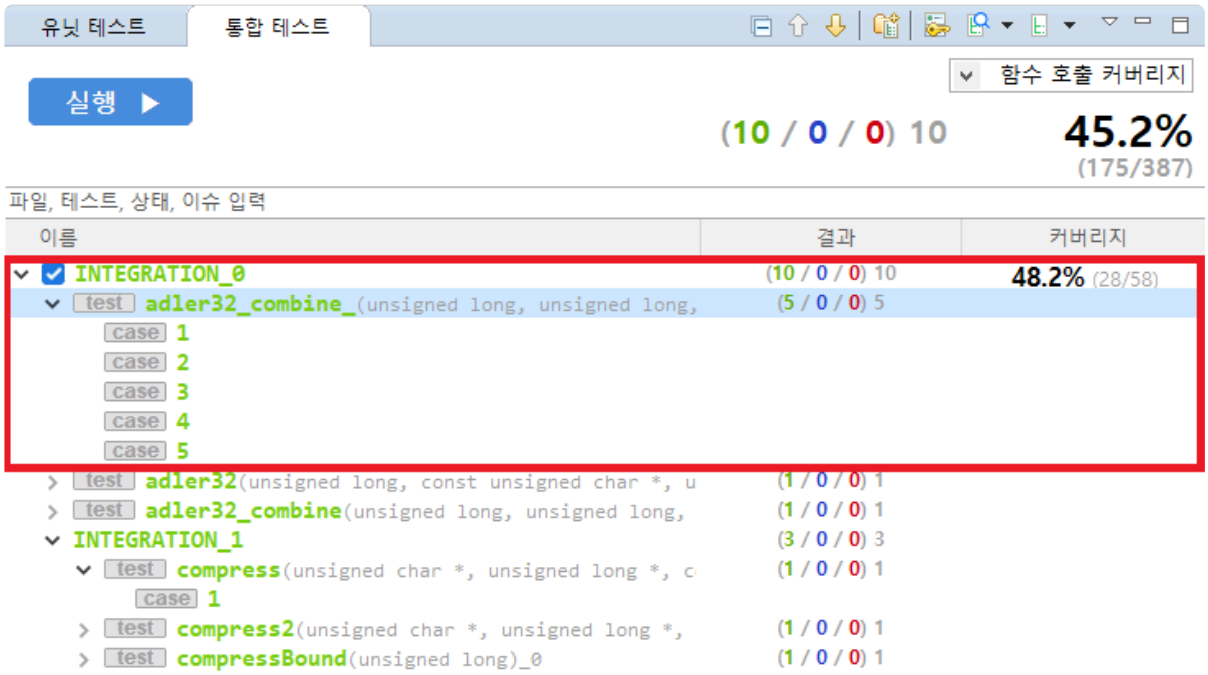


상태 검색 키워드

키워드	설명
%TEST_SUCCESS%	테스트 성공
%TEST_FAILURE% %TEST_ERROR%	테스트 실패/오류
%TEST_FAILURE%	테스트 실패
%TEST_ERROR%	테스트 오류
%TEST_HAS_NOT_FUNCTION%	함수 변경
%TEST_NOT_GUARANTEE%	실행 보장 안됨
%TEST_RESULT_DIFFERENT%	호스트/타깃 결과 다름

통합 테스트 뷰의 구조

통합 테스트 뷰는 [통합 테스트]-[테스트]-[테스트 케이스]의 계층 구조로 표현합니다.



통합 테스트 뷰의 항목 아이콘

항목 아아이콘	설명
아이콘 없음	통합 테스트
test	테스트
test	전역 변수 테스트
case	테스트 케이스, 테스트 케이스 그룹

통합 테스트 뷰의 항목 상태 표현

통합 테스트 뷰의 통합 테스트, 테스트, 테스트 케이스는 테스트 실행 여부 및 실행 결과를 색상으로 표현합니다.

색상	설명
녹색	통합 테스트/테스트: 하위에 모든 테스트 케이스의 실행 결과가 성공인 경우 테스트 케이스: 실행 결과가 성공인 경우
파란색	통합 테스트/테스트 : 하위에 실행 결과가 실패인 테스트 케이스가 존재하고 오류인 테스트 케이스는 존재하지 않음 테스트 케이스: 실행 결과가 실패인 경우
붉은색	통합 테스트/테스트: 하위에 실행 결과가 오류인 테스트 케이스가 존재함 테스트 케이스: 실행 결과가 오류인 경우
주황색	통합 테스트/테스트: 하위에 모든 테스트 케이스를 실행하지 않았고 실행 보장 안 됨 테스트 케이스가 존재함 테스트 케이스: 실행 보장 안 됨
검정색	통합 테스트/테스트: 하위에 모든 테스트 케이스를 실행하지 않음 테스트 케이스: 실행하지 않음

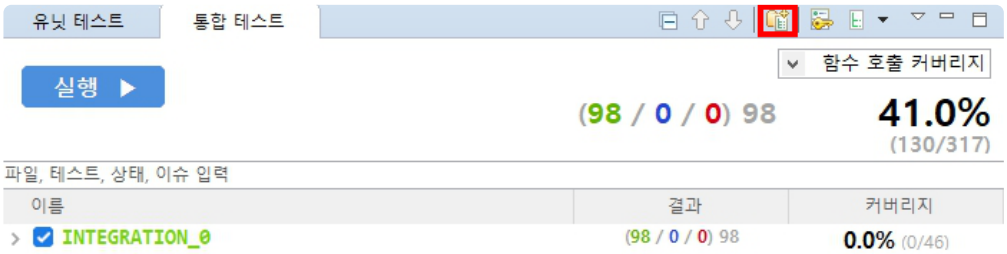
통합 테스트 컨텍스트 메뉴

컨텍스트 메뉴	설명
테스트 생성	테스트 생성 마법사 열기
하위 통합 테스트 생성	하위 통합 테스트 생성
복사	테스트 및 테스트 케이스 복사
붙여넣기	복사한 테스트 및 테스트 케이스 붙여넣기
복제	테스트 및 테스트 케이스 복제
여러번 복제	테스트 및 테스트 케이스를 입력한 수만큼 복제
하위 통합 테스트로 붙여넣기	복사한 통합 테스트를 선택한 통합 테스트의 하위 통합 테스트로 붙여넣기
삭제	테스트 및 테스트 케이스 삭제
이름 바꾸기	테스트 이름 변경
테스트 선택	선택된 모든 테스트의 실행 체크박스 상태를 체크로 변경
테스트 선택 해제	선택된 모든 테스트의 실행 체크박스 상태를 체크 해제로 변경

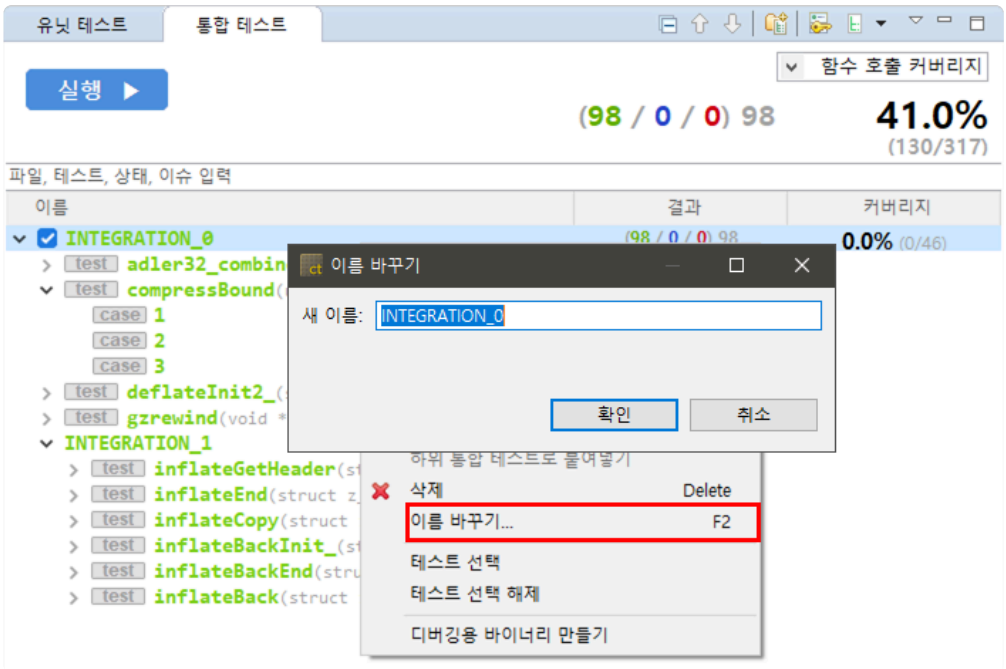
통합 테스트 생성 및 이름 변경

툴바 메뉴에서 통합 테스트를 생성할 수 있습니다. 통합 테스트 이름은 생성 시 자동으로 부여되며, 오른 쪽 클릭 후 [이름 바꾸기]를 선택하여 변경할 수 있습니다.

- 생성

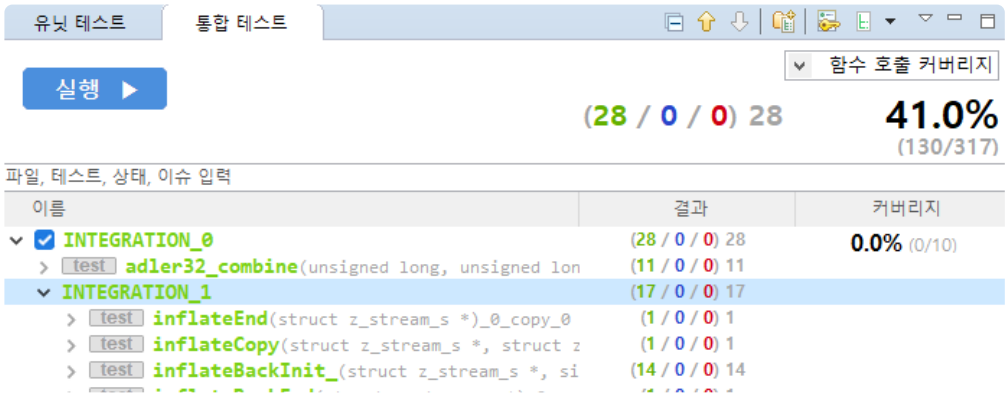


- 이름 바꾸기



하위 통합 테스트

통합 테스트에 하위 통합 테스트를 추가할 수 있습니다. [하위 통합 테스트 생성] 컨텍스트 메뉴를 이용해 추가할 수도 있고, 이미 만들어진 통합 테스트를 [하위 통합 테스트 붙여넣기] 컨텍스트 메뉴를 이용하여 붙여넣을 수도 있습니다.



테스트 추가 및 실행 순서 변경

통합 테스트에 테스트를 추가하는 방법은 네 가지 방법이 있습니다.

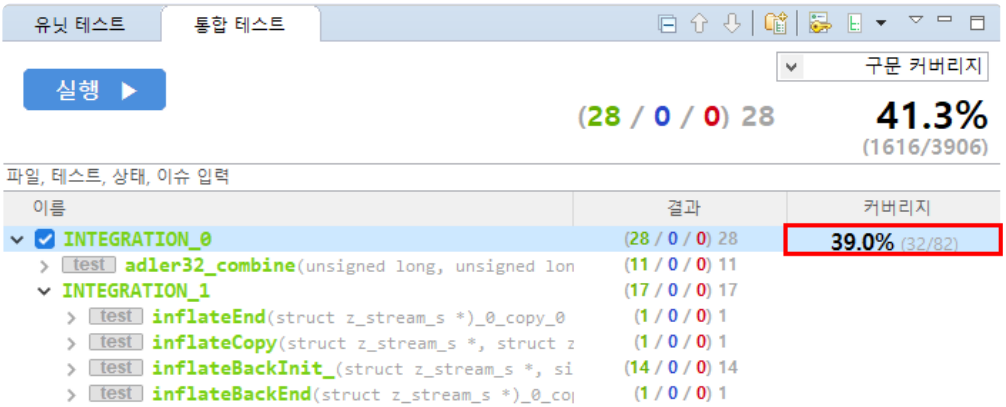
1. 통합 테스트를 선택 후 [테스트 생성] 컨텍스트 메뉴를 이용하여 생성할 수 있습니다.
2. 유닛 테스트 뷰의 테스트를 끌어서 놓아 추가할 수 있습니다.
3. 통합 테스트 뷰의 테스트를 선택 후 컨텍스트 메뉴를 이용하여 복사/붙여넣기 할 수 있습니다.
4. 유닛 테스트 뷰의 테스트를 선택 후 컨텍스트 메뉴를 이용하여 복사/붙여넣기 할 수 있습니다.

테스트 실행 순서를 변경하는 방법은 두 가지 방법이 있습니다.

1. 테스트 또는 테스트 케이스를 선택 후 마우스로 끌어서 순서를 변경할 수 있습니다.
2. 테스트 또는 테스트 케이스를 선택 후 툴바에 있는 “테스트 위로 이동”, “테스트 아래로 이동”을 선택하여 변경할 수 있습니다.

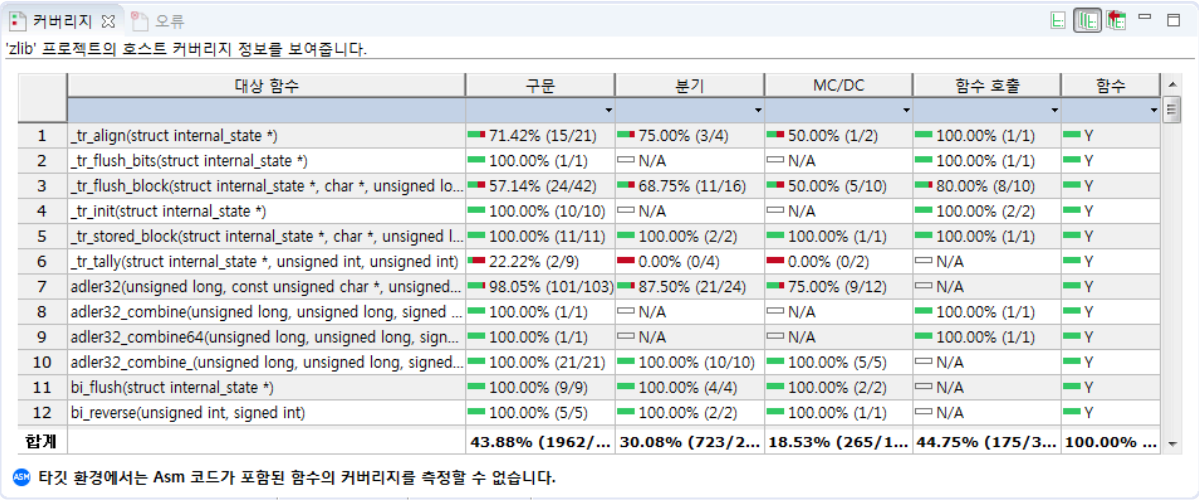
통합 테스트 커버리지 결과 보기

통합 테스트를 선택하면 커버리지 뷰에서 결과를 확인할 수 있습니다.



14.6. 커버리지 뷰

테스트 수행 후 해당 테스트 대상의 함수가 얼마만큼을 커버했는지 백분율로 보여줍니다.



	대상 함수	구문	분기	MC/DC	함수 호출	함수
1	_tr_align(struct internal_state *)	71.42% (15/21)	75.00% (3/4)	50.00% (1/2)	100.00% (1/1)	Y
2	_tr_flush_bits(struct internal_state *)	100.00% (1/1)	N/A	N/A	100.00% (1/1)	Y
3	_tr_flush_block(struct internal_state *, char *, unsigned lo...	57.14% (24/42)	68.75% (11/16)	50.00% (5/10)	80.00% (8/10)	Y
4	_tr_init(struct internal_state *)	100.00% (10/10)	N/A	N/A	100.00% (2/2)	Y
5	_tr_stored_block(struct internal_state *, char *, unsigned l...	100.00% (11/11)	100.00% (2/2)	100.00% (1/1)	100.00% (1/1)	Y
6	_tr_tally(struct internal_state *, unsigned int, unsigned int)	22.22% (2/9)	0.00% (0/4)	0.00% (0/2)	N/A	Y
7	adler32(unsigned long, const unsigned char *, unsigned...	98.05% (101/103)	87.50% (21/24)	75.00% (9/12)	N/A	Y
8	adler32_combine(unsigned long, unsigned long, signed ...	100.00% (1/1)	N/A	N/A	100.00% (1/1)	Y
9	adler32_combine64(unsigned long, unsigned long, sign...	100.00% (1/1)	N/A	N/A	100.00% (1/1)	Y
10	adler32_combine_(unsigned long, unsigned long, signed...	100.00% (21/21)	100.00% (10/10)	100.00% (5/5)	N/A	Y
11	bi_flush(struct internal_state *)	100.00% (9/9)	100.00% (4/4)	100.00% (2/2)	N/A	Y
12	bi_reverse(unsigned int, signed int)	100.00% (5/5)	100.00% (2/2)	100.00% (1/1)	N/A	Y
합계		**43.88% (1962/...**	**30.08% (723/2...**	**18.53% (265/1...**	**44.75% (175/3...**	**100.00% ...**

Controller Tester에서 제공하는 커버리지 종류

커버리지	설명
구문	테스트에 의해 수행되는 소스코드 구문의 백분율
분기	테스트에 의해 수행된 분기의 백분율 100% 분기 커버리지는 100%의 결정 커버리지와 100% 구문 커버리지를 포함
MC/DC	조건 커버리지와 결정 커버리지를 복합적으로 고려한 커버리지 (MC/DC 뷰에서 자세히 설명)
함수 호출	전체 함수 중 테스트에 의해 호출되는 함수의 백분율
함수	전체 함수 중 한 번이라도 호출된 함수의 백분율

커버리지 뷰 레이블 아이콘

아이콘	설명
	변경된 함수
	Asm 코드가 포함된 함수

커버리지 뷰 툴바 메뉴

툴바 아이콘	설명
	테스트별 커버리지 보기

	전체 커버리지 보기
	전체 커버리지 보기(외부 커버리지 포함)

커버리지 보기

커버리지 뷰에서는 유닛 테스트 뷰와 통합 테스트 뷰에서 선택한 테스트 또는 테스트 케이스의 커버리지 정보를 보여줍니다. [전체 커버리지 보기]를 클릭하면, 전체 테스트의 커버리지 정보를 병합해서 볼 수 있습니다. 그리고 [전체 커버리지 보기(외부 커버리지 포함)]를 클릭하면, 외부에서 가져온 커버리지까지 병합해서 보여줍니다.

커버리지 뷰 하단에는 트리에서 선택한 항목의 커버리지가 표시됩니다.

합계		**43.88% (1962/...**	**30.08% (723/2...**	**18.53% (265/1...**	**44.75% (175/3...**	**100.00% ...**

커버리지 뷰에는 유닛 또는 통합 테스트 뷰에서 선택한 항목에 대하여 각 함수에 대한 커버리지가 표시됩니다.

커버리지 오류

'zlib' 프로젝트의 호스트 커버리지 정보를 보여줍니다.

	대상 함수	구문	분기	MC/DC	함수 호출	함수
1	_tr_align(struct internal_state *)	71.42% (15/21)	75.00% (3/4)	50.00% (1/2)	100.00% (1/1)	Y
2	_tr_flush_bits(struct internal_state *)	100.00% (1/1)	N/A	N/A	100.00% (1/1)	Y
3	_tr_flush_block(struct internal_state *, char *, unsigned lo...	57.14% (24/42)	68.75% (11/16)	50.00% (5/10)	80.00% (8/10)	Y
4	_tr_init(struct internal_state *)	100.00% (10/10)	N/A	N/A	100.00% (2/2)	Y
5	_tr_stored_block(struct internal_state *, char *, unsigned l...	100.00% (11/11)	100.00% (2/2)	100.00% (1/1)	100.00% (1/1)	Y
6	_tr_tally(struct internal_state *, unsigned int, unsigned int)	22.22% (2/9)	0.00% (0/4)	0.00% (0/2)	N/A	Y
7	adler32(unsigned long, const unsigned char *, unsigned...	98.05% (101/103)	87.50% (21/24)	75.00% (9/12)	N/A	Y
8	adler32_combine(unsigned long, unsigned long, signed ...	100.00% (1/1)	N/A	N/A	100.00% (1/1)	Y
9	adler32_combine64(unsigned long, unsigned long, sign...	100.00% (1/1)	N/A	N/A	100.00% (1/1)	Y
10	adler32_combine_(unsigned long, unsigned long, signed...	100.00% (21/21)	100.00% (10/10)	100.00% (5/5)	N/A	Y
11	bi_flush(struct internal_state *)	100.00% (9/9)	100.00% (4/4)	100.00% (2/2)	N/A	Y
12	bi_reverse(unsigned int, signed int)	100.00% (5/5)	100.00% (2/2)	100.00% (1/1)	N/A	Y
합계		**43.88% (1962/...**	**30.08% (723/2...**	**18.53% (265/1...**	**44.75% (175/3...**	**100.00% ...**

타깃 환경에서는 Asm 코드가 포함된 함수의 커버리지를 측정할 수 없습니다.

커버리지 뷰 테이블에서 컨텍스트 메뉴를 통하여 각 함수의 제어 흐름 그래프, 함수 호출 그래프, MC/DC를 확인할 수 있습니다.

커버리지

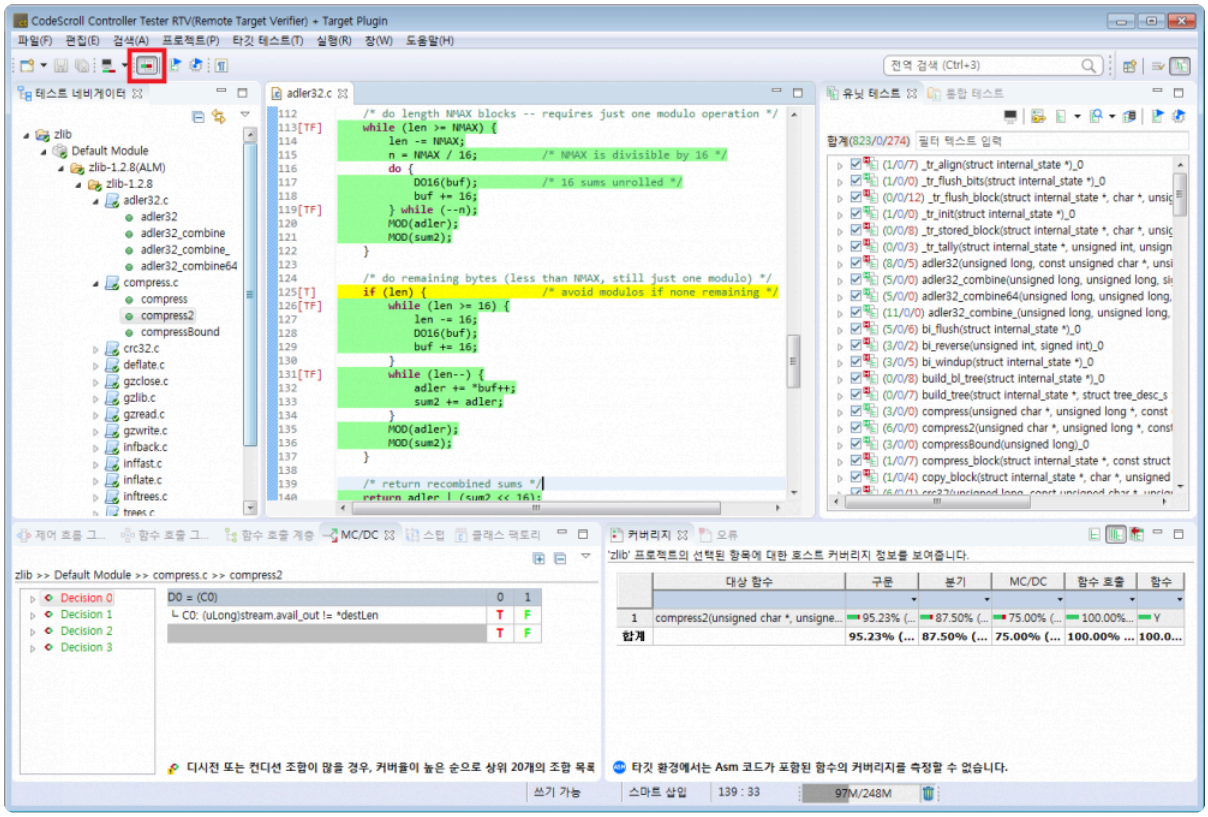
'zlib' 프로젝트의 호스트 커버리지 정보를 보여줍니다.

	대상 함수	구문	분기	MC/DC	함수 호출	함수
1	_tr_align(struct internal_state *)	72.72% (32...	70.00% (7/...	4		Y
2	_tr_flush_block(struct internal_state *, char *, unsign...	57.14% (24/42)	68.75% (11/16)	5		Y
3	_tr_init(struct internal_state *)	100.00% (11/...	N/A	N		Y
4	_tr_stored_block(struct internal_state *, char *, unsig...	100.00% (11/...	100.00% (2/2)	1		Y
5	_tr_tally(struct internal_state *, unsigned int, unsign...	22.22% (2/9)	0.00% (0/4)	0.00% (0/2)	N/A	Y
6	adler32(unsigned long, const unsigned char *, unsi...	97.08% (100/...	87.50% (21/24)	75.00% (9/12)	N/A	Y
7	adler32_combine(unsigned long, unsigned long, si...	100.00% (18/...	100.00% (8/8)	100.00% (4/4)	N/A	Y
8	adler32_combine_symbols(unsigned long, unsigned...	100.00% (18/...	87.50% (7/8)	75.00% (3/4)	N/A	Y
9	bi_flush(struct internal_state *)	100.00% (9/9)	100.00% (4/4)	100.00% (2/2)	N/A	Y
10	bi_reverse(unsigned int, signed int)	100.00% (5/5)	100.00% (2/2)	100.00% (1/1)	N/A	Y
11	bi_windup(struct internal_state *)	100.00% (7/7)	100.00% (4/4)	100.00% (2/2)	N/A	Y
12	build_bl_tree(struct internal_state *)	100.00% (11/	75.00% (3/4)	50.00% (1/2)	100.00% (3/3)	Y
합계		**41.37% (1616...**	**27.21% (563/...**	**18.02% (221/...**	**41.00% (130/...**	**100.00%...**

제어 흐름 그래프 보기
함수 호출 그래프 보기
MC/DC 보기

타깃 환경에서는 Asm 코드가 포함된 함수의 커버리지를 측정할 수 없습니다.

메인 툴바의 [커버 여부 보여주기] 아이콘을 활성화 시킨 후 테스트 케이스 뷰에서 원하는 테스트 케이스를 더블 클릭하면 해당 소스코드의 어느 부분이 커버 되었는지 색상 정보로 표시해 줍니다. 초록색으로 표시가 된 부분은 코드가 실행되고 모든 분기가 커버되었을 경우, 붉은색으로 표시가 된 부분은 실행되지 않는 코드를 의미합니다. 노란색으로 표시가 된 부분은 분기중 일부만 커버된 경우입니다.
에디터 왼쪽 세로 컬럼에서 커버된 분기문의 T/F를 확인 할 수 있습니다.



14.7. MC/DC 뷰

MC/DC 뷰에서는 변경 조건/결정 커버리지(Modified Condition/Decision Coverage, MC/DC)의 정보를 보여 줍니다.

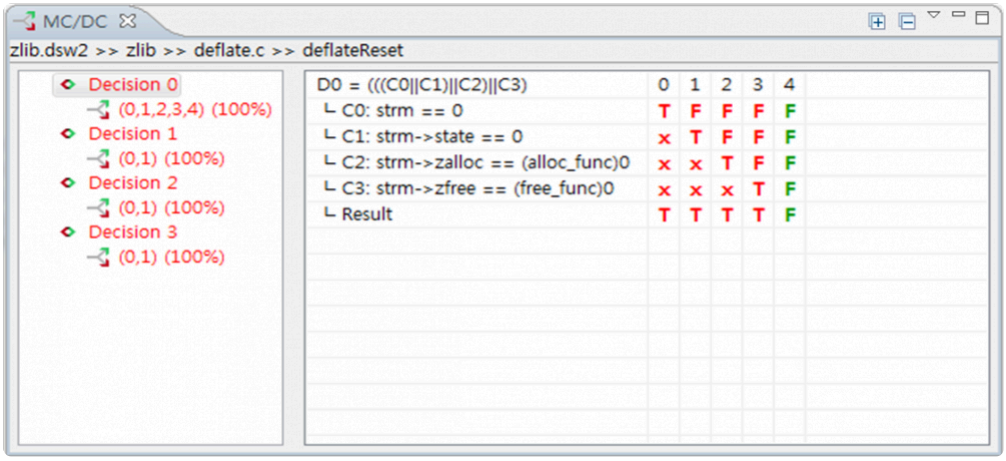
MC/DC 커버리지

MC/DC란 각 개별 조건식이 다른 개별 조건식에 영향을 받지 않고 전체 조건식의 결과에 독립적으로 영향을 주도록 함으로써 조건/결정 커버리지를 향상시킨 것으로 결정 커버리지, 조건/결정 커버리지 보다 강력합니다.
다른 상태(condition)들의 변동이 없고 자신의 상태가 변경 되었을 때 결과값에 영향을 미치는 경우 해당 상태는 MC/DC를 만족한다 할 수 있고 이때 해당 진리표를 만드는 조건은 다음과 같습니다.

- 해당 디시전(결정문)의 상태는 모든 가능한 결과(참, 거짓)를 적어도 한 번씩 만족해야 한다.
- 디시전에 속한 모든 개별 조건식은 모든 가능한 결과(참, 거짓)를 적어도 한 번씩 만족해야 한다.
- 디시전에 속한 각각의 개별 조건식은 다른 개별 조건식에 영향을 받지 않고 자신이 속한 디시전의 결과값에 독립적으로 영향을 준다.

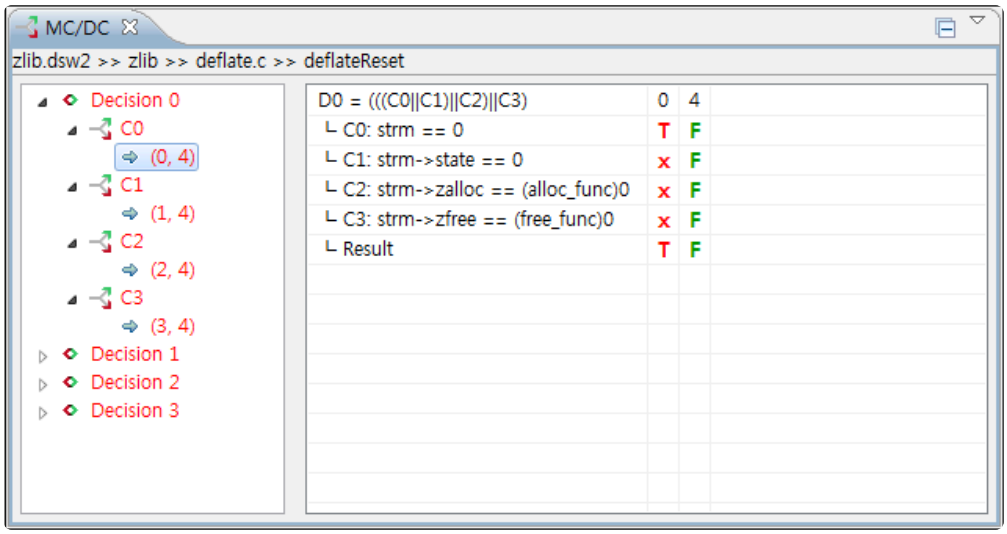
MC/DC 뷰 테이블 설명

- 목표 커버리지를 만족하는 조합 목록 보기
  아래 그림은 목표 커버리지를 달성하기 위해 커버해야 하는 조합을 쉽게 확인할 수 있는 뷰입니다.
  좌측에는 선택한 함수에 대한 Decision 목록이 있고, 그 하위에는 목표 커버리지를 만족하는 조합 목록이 나타납니다.
  우측에서는 선택한 Decision 또는 조합의 진리표와 커버 여부를 보여줍니다. 커버된 조합은 녹색, 커버되지 않은 조합은 빨간색으로 표시됩니다.
  환경 설정 페이지에서 목표 커버리지율과 화면 보기 모드를 변경할 수 있습니다. 그리고 진리표를 전체 선택하여 'Ctrl + C' 또는 컨텍스트 메뉴로 클립보드 복사 기능을 사용할 수 있습니다.



- 조건 별 커버리지를 만족하는 조합 목록 보기

아래 그림은 하나의 조건을 만족하는 한 쌍(Pair)의 조합을 확인할 수 있는 뷰입니다. 좌측 화면에는 각 조건 별 커버리지를 만족하는 조합 목록을 보여줍니다. 우측 화면에는 선택한 조합에 대한 진리표와 커버 여부 정보를 보여줍니다.

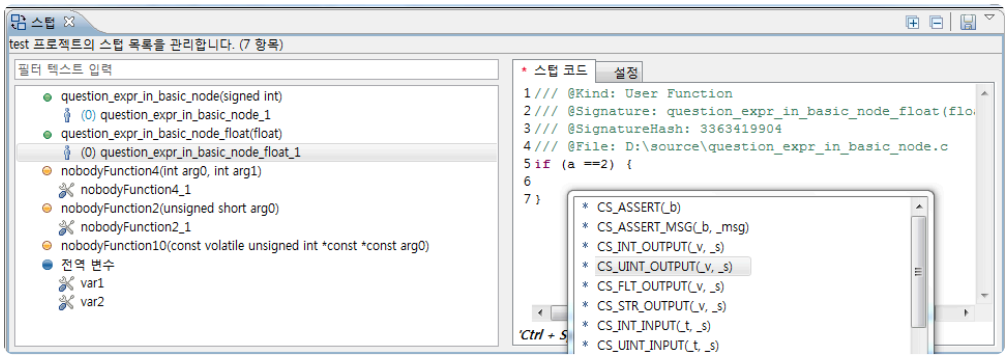


14.8. 스텁 뷰

Controller Tester에서는 테스트 수행 시 대상 함수에서 사용하는 라이브러리가 없거나 아직 개발되지 않은 경우 혹은 선언되어 있지 않은 변수를 사용하는 경우, 이를 대신하는 스텁을 자동으로 생성합니다. 이때 만들어진 스텁은 테스트 수행 시 생성되며 함수 내부는 구현되어 있지 않습니다.

Controller Tester가 자동으로 생성한 스텁 외에 사용자가 스텁을 직접 추가할 수 있습니다. 동일한 함수에 대하여 여러 개의 스텁을 생성할 수 있으며 생성된 스텁은 각각의 테스트에 연결할 수 있습니다.

스텁 내부에서 사용자 매크로를 사용하여 각 스텁의 동작을 제어할 수 있습니다. 사용자가 입력한 매크로에서 사용하는 값은 연결된 테스트의 테스트 케이스 편집 탭에서 제어할 수 있습니다.



함수 타입

타입	설명
함수	정의가 없는 함수
함수	정의가 있는 함수
변수	전역 변수

스텁 타입

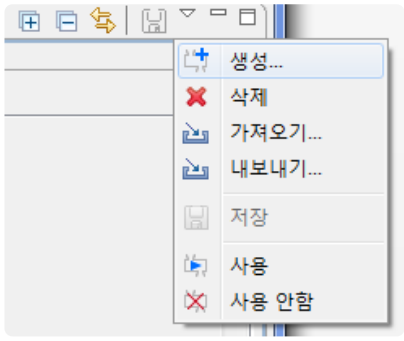
타입	설명
사용자 스텁	사용자 정의 스텁
빌드 스텁	자동 생성된 스텁
구 버전 스텁	2.3 이하 버전의 스텁

메뉴

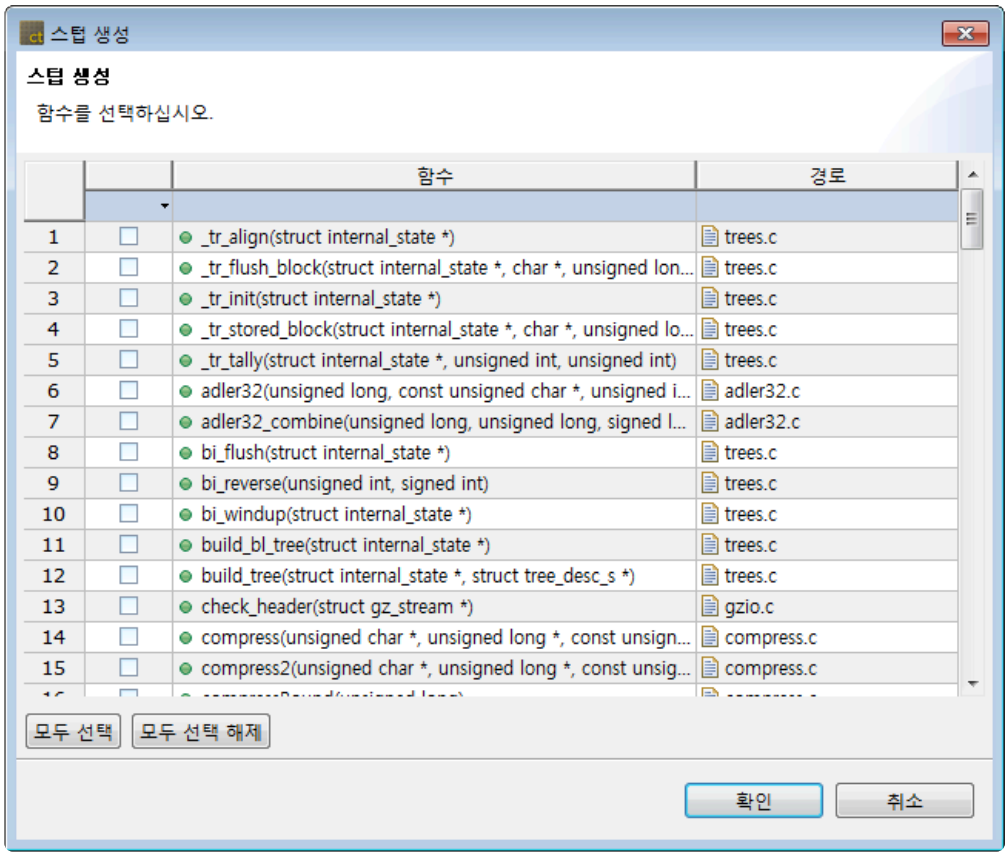
메뉴	설명
생성	스텁 생성
삭제	스텁 삭제
가져오기	내보낸 스텁 파일 가져오기
내보내기	스텁 파일 내보내기
저장	변경된 스텁 정보 저장하기
사용	스텁 활성화
사용 안함	스텁 비활성화

스텁 생성

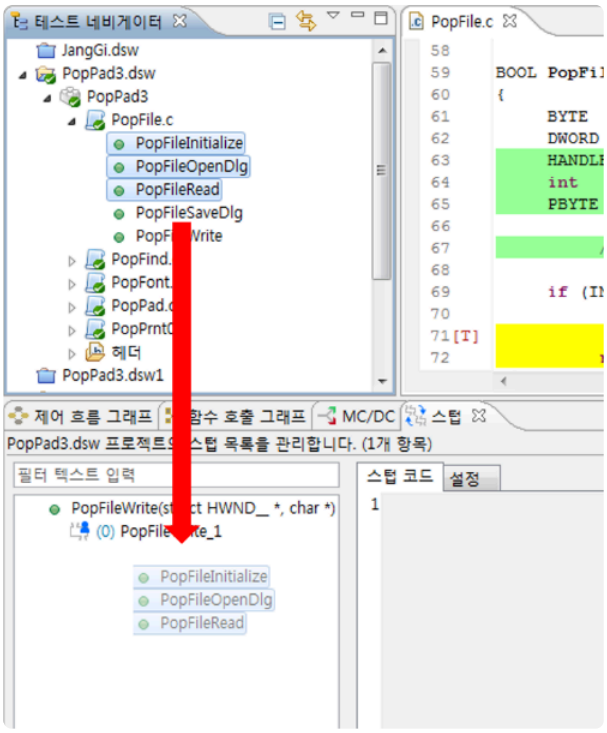
- 스텁 뷰에서 생성하기
  1. 스텁 뷰의 풀다운 메뉴(▽)를 클릭하여 [생성] 메뉴를 클릭합니다.



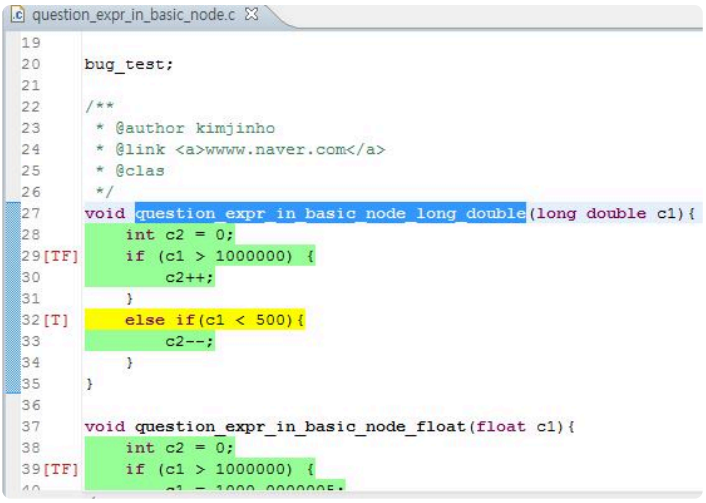
  2. 스텁으로 생성할 함수를 체크한 후 [확인] 버튼을 선택합니다.



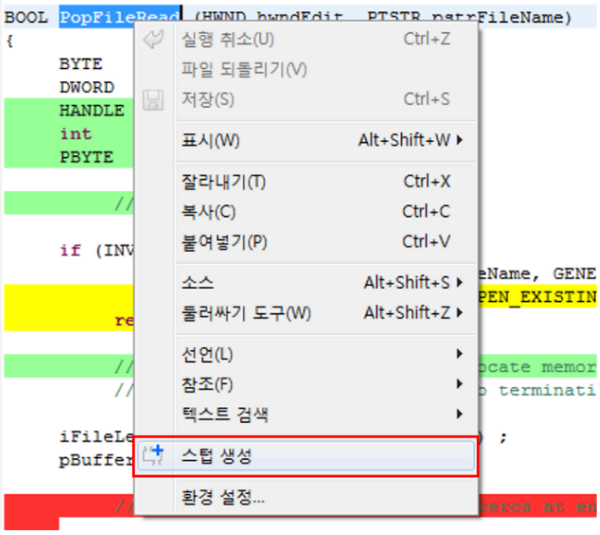
- 테스트 네비게이터에서 스텁 뷰로끌어 놓아 생성하기



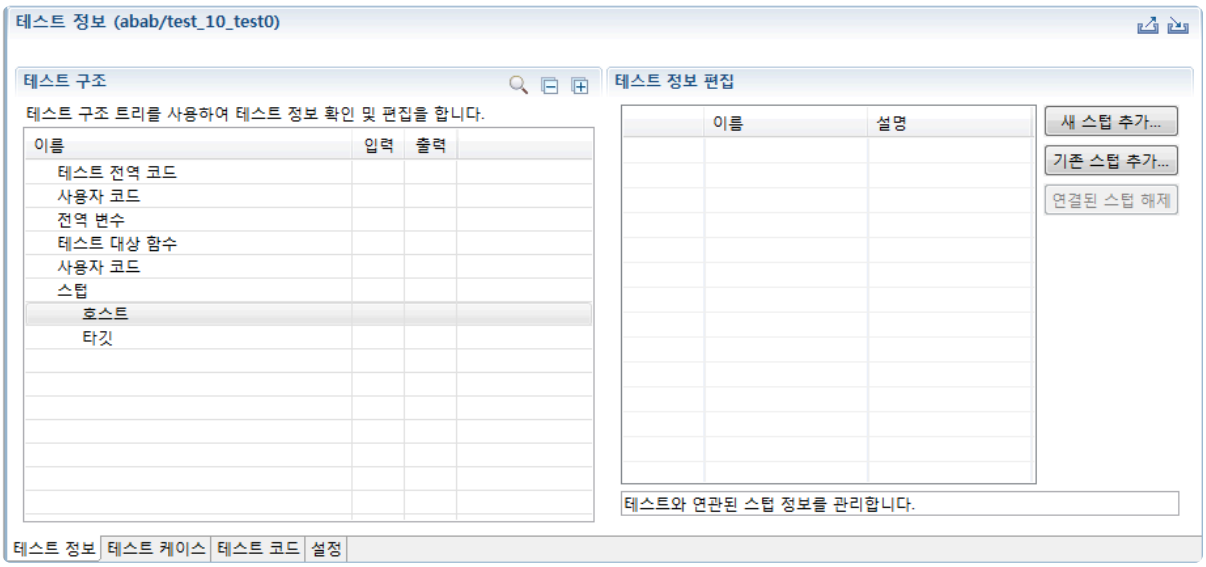
- 소스 파일 편집기에서 생성하기
  1. 스텁으로 생성할 함수를 선택합니다.



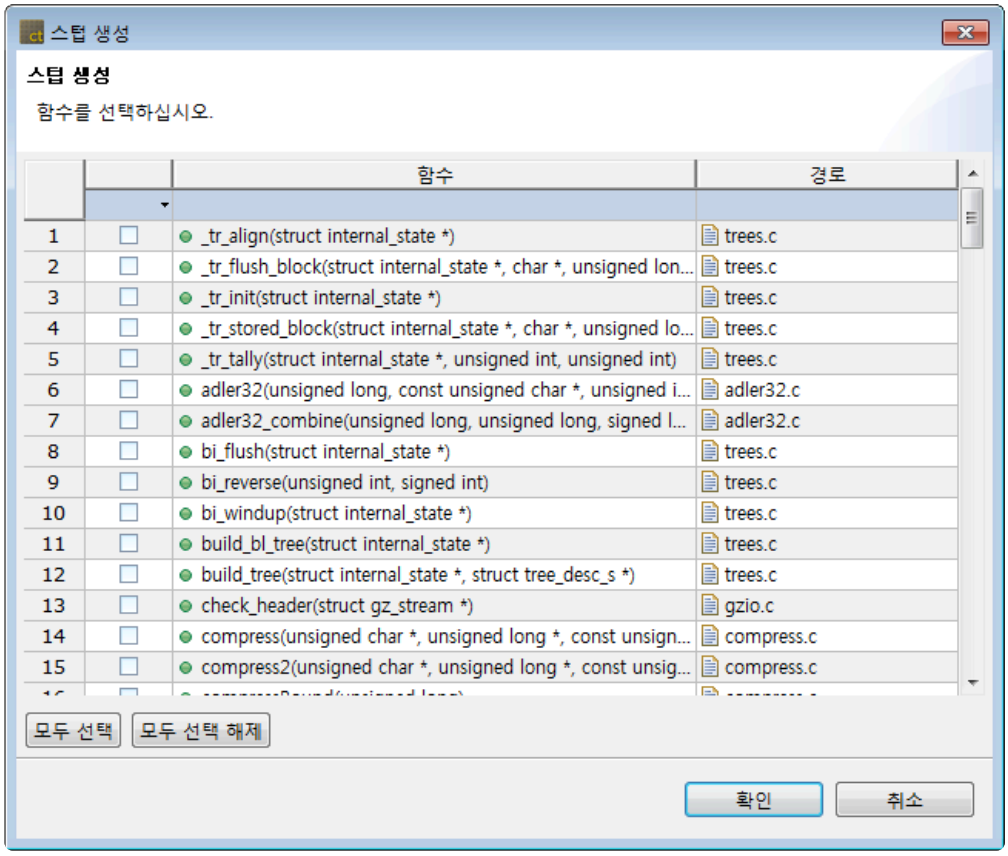
2. 우 클릭 후 [스텝 생성] 컨텍스트 메뉴를 선택합니다.



- 테스트 에디터에서 생성 후 연결하기
  1. 테스트 에디터의 테스트 구조에서 스텝 또는 하위 노드를 선택합니다.
  2. 우측 화면에 있는 [새 스텝 추가...] 버튼을 선택합니다.

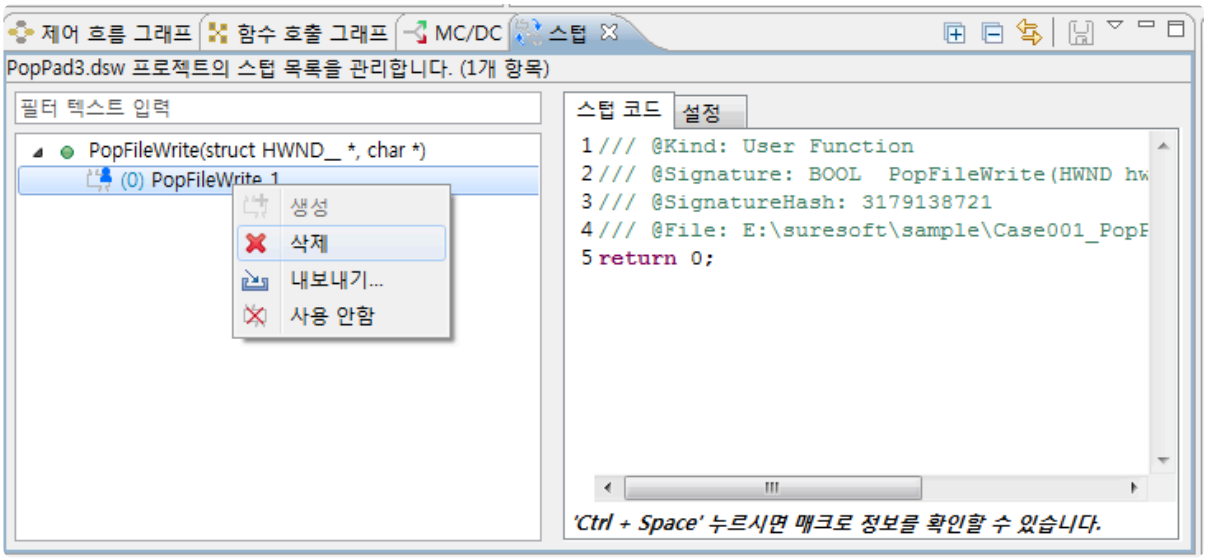


3. 스텁으로 생성할 함수를 체크한 후 [확인] 버튼을 선택합니다.



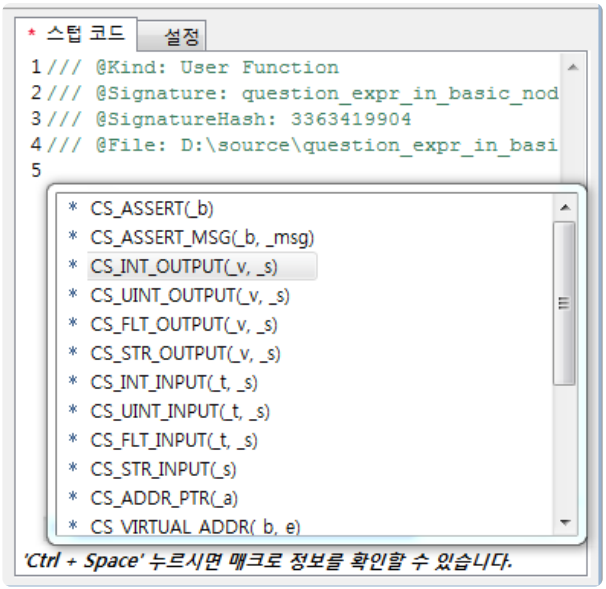
스텁 삭제

- 불필요한 스텁을 선택하고 우 클릭하여 [삭제] 메뉴를 선택합니다.

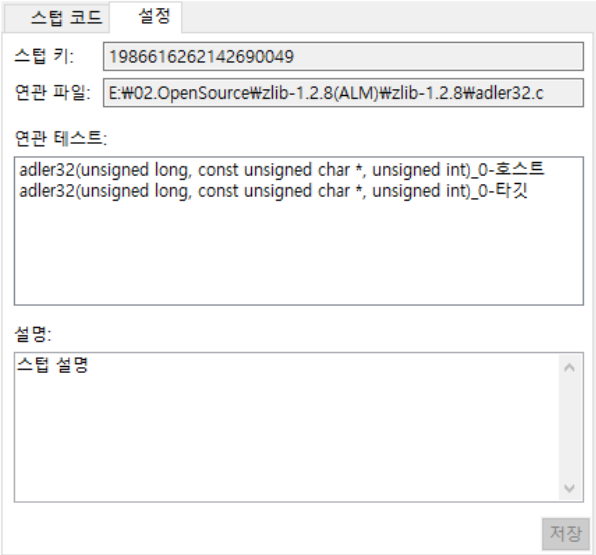


스텁 편집

- 스텁 코드 화면에서 'Ctrl + Space' 누르면 매크로 정보를 확인할 수 있습니다. 스텁 코드를 편집 후 'Ctrl + S'를 누르거나, [저장] 툴바 버튼을 누르면 저장됩니다.



- 스텁 설정 화면에서는 설명을 편집할 수 있습니다.

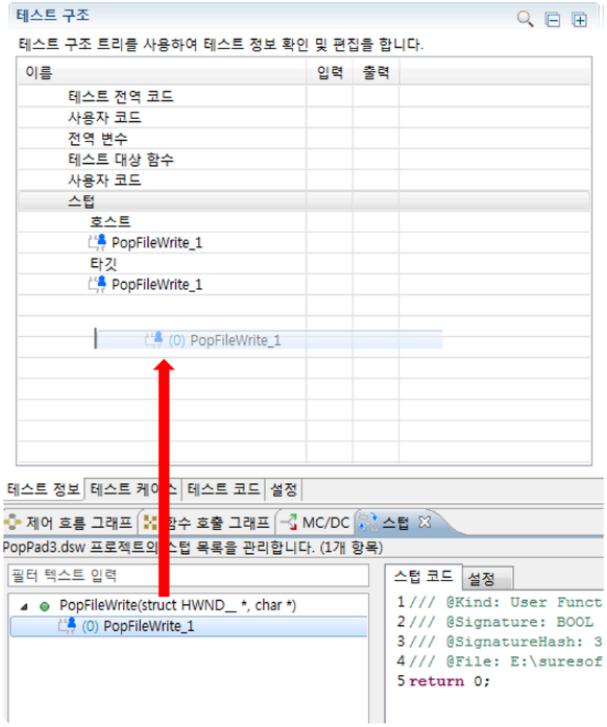


스텁과 테스트 연결하기

각 스텁은 호스트/타깃 테스트에 연결 가능합니다.
테스트 구조 편집기의 “스텁”을 선택 하면 호스트와 타깃에 속한 모든 스텁을 편집할 수 있습니다.

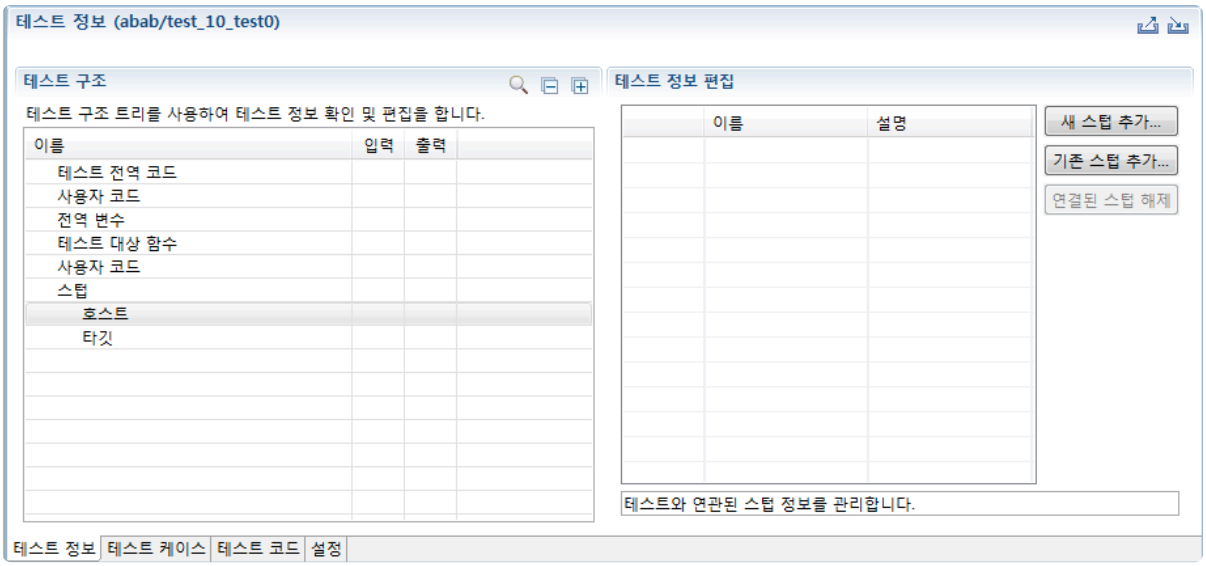
“호스트” 또는 “타깃”을 선택하면 선택된 시험 환경의 스텁만 편집할 수 있습니다.

- 스텝 뷰의 스텝을 테스트 에디터에 끌어서 놓기로 연결하기

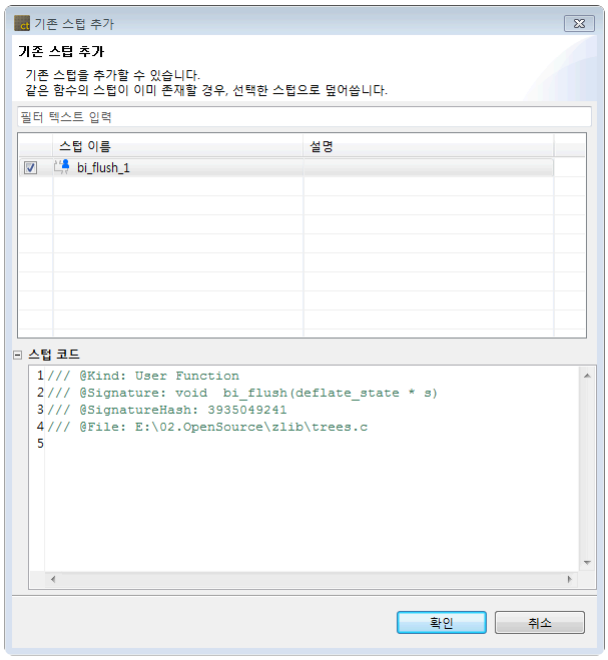


> 끌어서 놓기로 연결 시 타깃과 호스트 전부에 연결이 됩니다.

- 테스트 에디터에서 [기존 스텝 추가...] 버튼으로 연결하기
  1. 테스트 에디터의 테스트 구조에서 스텝을 선택합니다.
  2. 우측 화면에 있는 [기존 스텝 추가...] 버튼을 선택합니다.

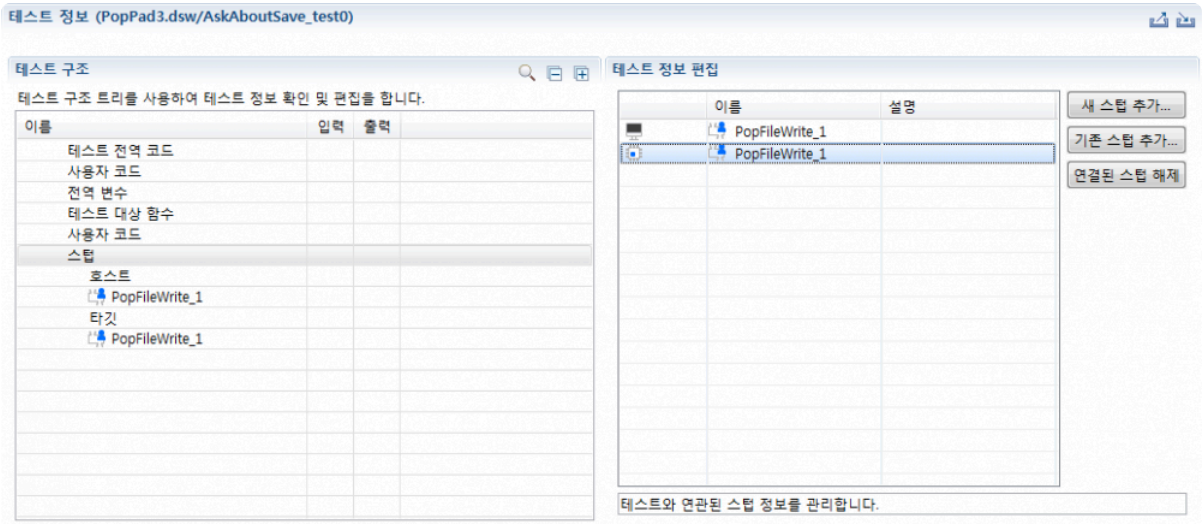


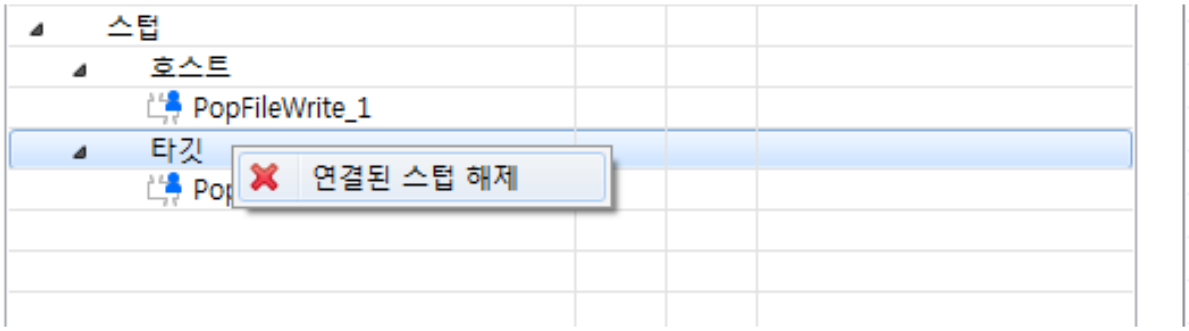
  3. 기존 스텝 목록 중 연결할 스텝을 체크한 후 [확인] 버튼을 선택합니다.



스텁과 테스트 연결 해제하기

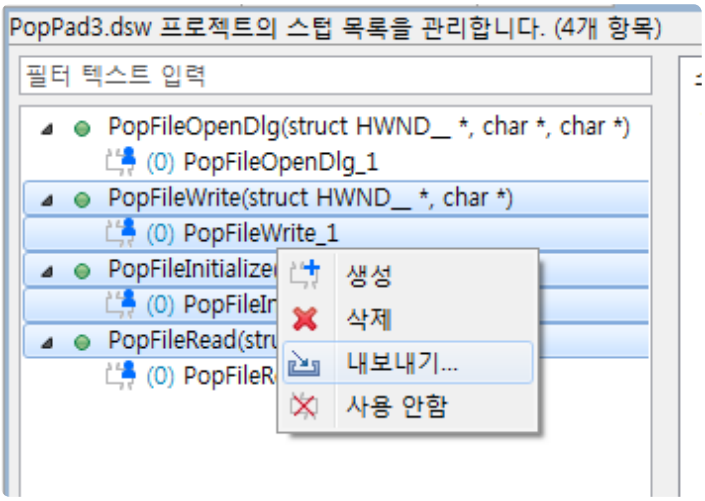
테스트 편집기에서 연결되어 있는 스텁을 선택한 후 [연결 스텁 해제] 버튼을 선택을 하거나, 테스트 구조 트리에서 스텁을 선택한 후 [Delete] 키 또는 우 클릭하여 연결 해제를 할 수 있습니다.



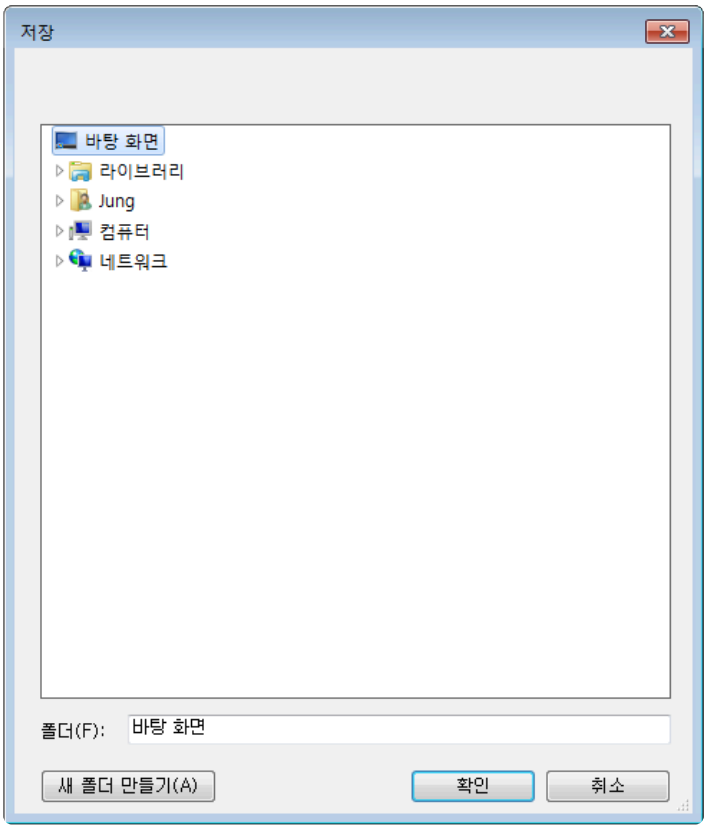


스텁 내보내기

1. 내보낼 스텁을 선택한 후 우 클릭으로 [내보내기...] 메뉴를 선택합니다.

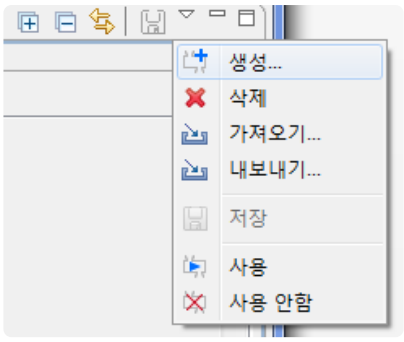


2. 디렉터리 다이얼로그에서 스텁 파일의 저장 위치를 지정한 후 [확인] 버튼을 선택합니다.

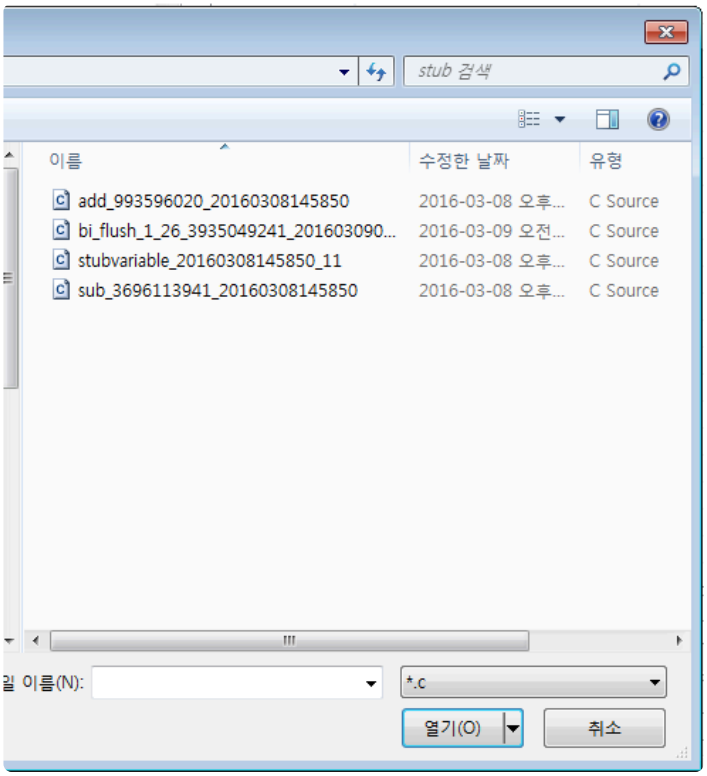


스텁 가져오기

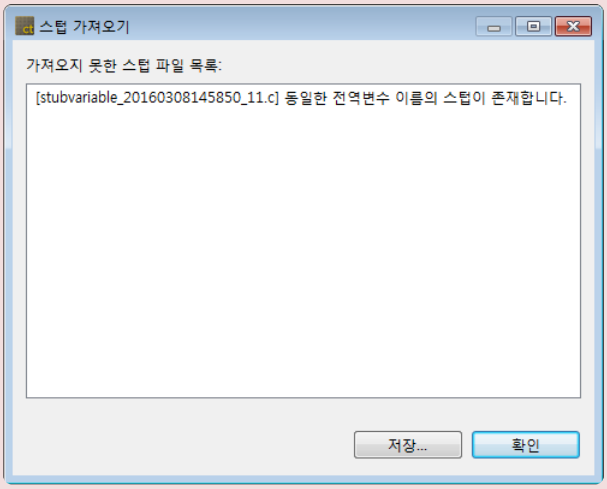
1. 스텁 뷰의 풀다운 메뉴(▽)에서 [가져오기...] 메뉴를 선택합니다.



2. 가져올 스텁 파일을 선택한 후 [열기] 버튼을 선택합니다.



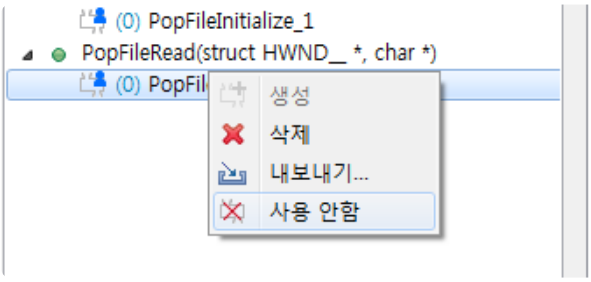
! 가져오려는 스텁과 동일한 구 버전 스텁 또는 빌드 스텁이 프로젝트에 존재할 경우, 가져올 수 없습니다.



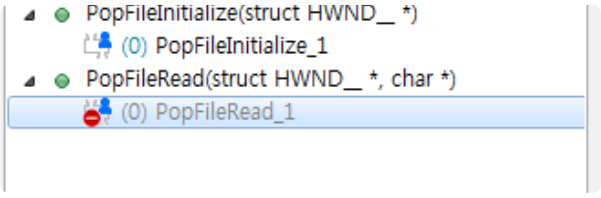
스텁 사용, 사용 안함

생성된 스텁 사용 여부를 설정할 수 있습니다.

1. 스텁을 선택한 후 우 클릭으로 [사용 안함] 메뉴를 선택합니다.

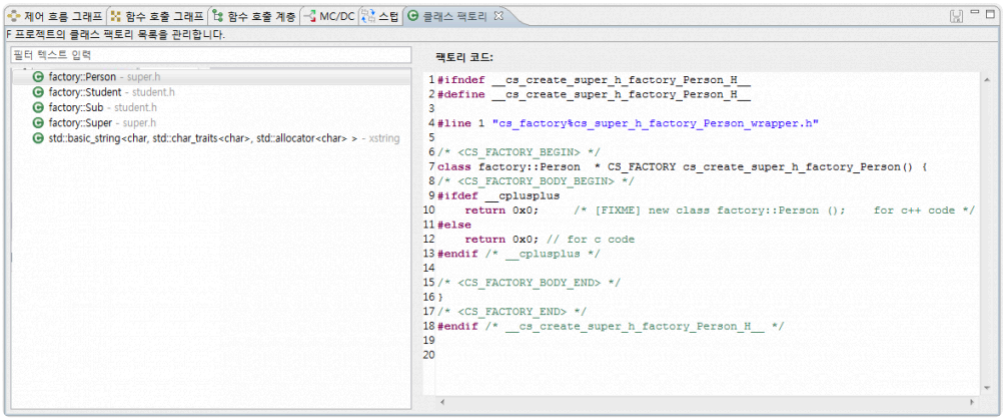


2. 아래 그림과 같이 스텁 이미지 데코레이터가 추가되고 스텁 이름의 색이 회색으로 변경됩니다.

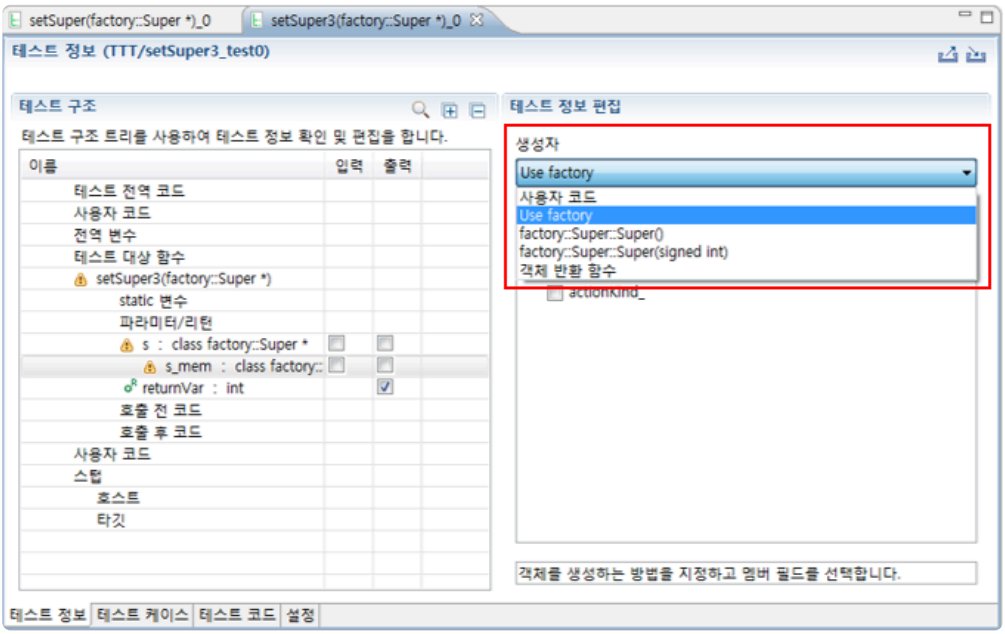


14.9. 클래스 팩토리 뷰

클래스 팩토리 뷰는 프로젝트에 속한 클래스의 목록을 보여줍니다. 자동 생성된 테스트 코드는 클래스 팩토리 뷰에 있는 클래스 생성 함수를 사용하여 객체를 생성합니다. 클래스 팩토리 뷰의 객체 생성 코드를 수정하면 해당 팩토리를 사용하는 모든 테스트의 객체 생성에 적용 됩니다.

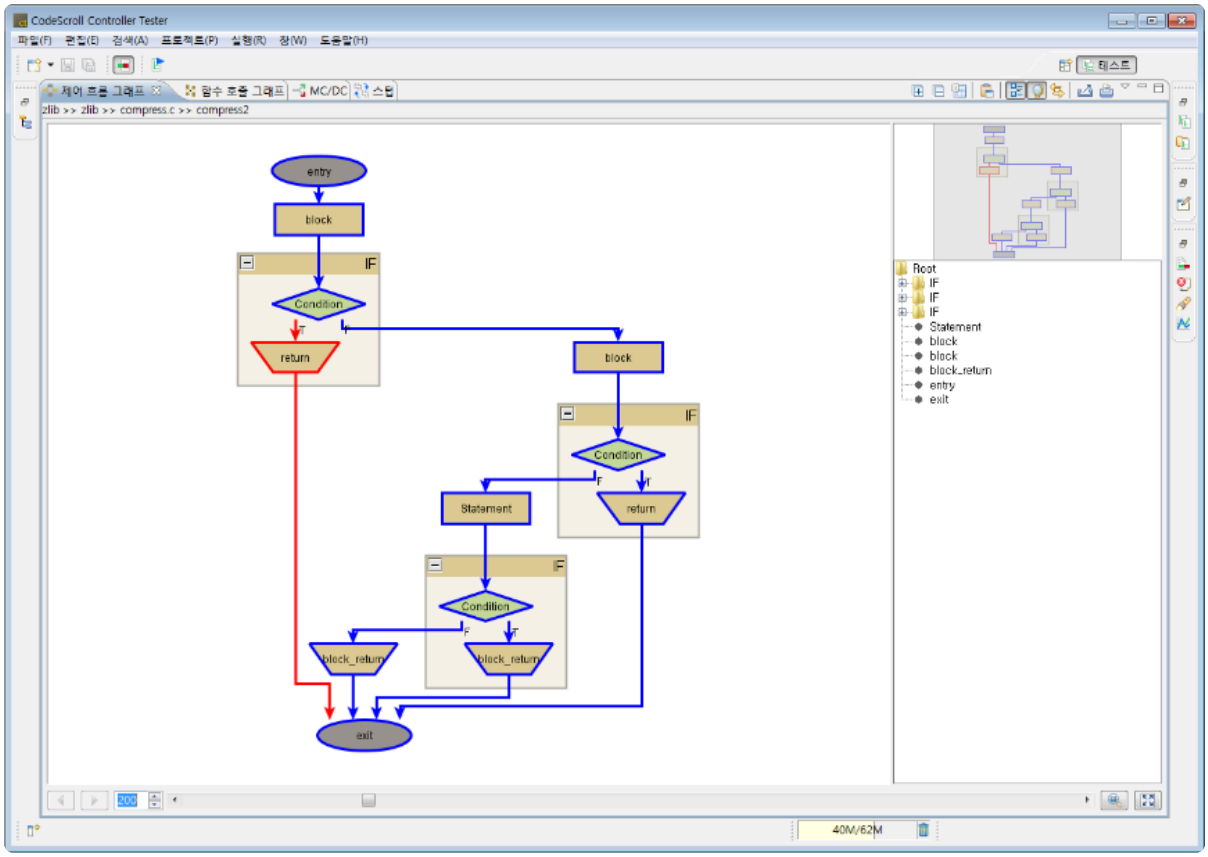


클래스 팩토리 사용을 원하지 않는 경우 테스트 정보 편집기에서 사용할 생성자를 직접 선택하여 변경 가능합니다.



14.10. 제어 흐름 그래프 뷰

제어 흐름 그래프 뷰는 선택된 함수의 제어 흐름 정보를 그래프로 보여줍니다.
메인 툴바의 [커버 여부 보여주기] 버튼을 클릭하면 커버된 영역은 파란색, 커버되지 않은 영역은 붉은 색으로 표시 됩니다.



툴바 메뉴

메뉴	설명
모두 펼치기	모든 그룹 노드를 펼쳐 보임
모두 접기	모든 그룹 노드를 접어 보임
범례 표시	범례(현재 보이는 그래프 상의 노드 종류)를 표시
시스템 클립보드에 복사	현재 화면에 보이는 그래프를 클립보드에 복사
아웃라인 표시	그래프를 트리 형태로 표시
오버뷰 표시	그래프의 오버뷰 표시
편집기와 연결	그래프에서 선택된 항목을 원클릭으로 편집기에 표시

뷰 내용 내보내기	뷰의 내용을 리포트로 내보내기
뷰 내용을 인쇄	뷰의 내용을 인쇄

풀다운 메뉴

메뉴	설명
모두 펼치기	모든 그룹 노드를 펼쳐 보임
모두 접기	모든 그룹 노드를 접어 보임
함수 호출 그래프 보기	현재 뷰에 선택된 함수를 중심으로 함수 호출 그래프 보기
범례 표시	범례(현재 보이는 그래프 상의 노드 종류)를 표시
아웃라인 표시	그래프를 트리 형태로 표시
오버뷰 표시	그래프의 오버뷰 표시
편집기와 연결	그래프에서 선택된 항목을 원클릭으로 편집기에 표시
뷰 내용 내보내기	뷰의 내용을 리포트로 내보내기
그래프 포맷으로 저장	현재 화면에 보이는 그래프에 대한 그래프 모델 파일 생성 네 사지 포맷 지원: • Graph Modeling Language XML (*.xgml) • Graph Modeling Language(*.gml) • yWorks Binary Graph Format(.ygf) • Trivial Graph Format(*.tgf)
이미지 포맷으로 저장	현재 화면에 보이는 그래프를 이미지 포맷 파일(jpg, gif)로 저장
시스템 클립보드에 복사	현재 화면에 보이는 그래프를 클립보드에 복사
환경 설정	환경 설정 열기

노드 팝업 메뉴

메뉴	설명
함수 호출 그래프 보기	현재 뷰에 선택된 함수를 중심으로 함수 호출 그래프 보기

히스토리 기능

뷰 왼쪽 하단의 화살표 버튼(뒤로 가기, 앞으로 가기)으로 현재 그래프에서 선택했던 노드를 다시 볼 수 있습니다.

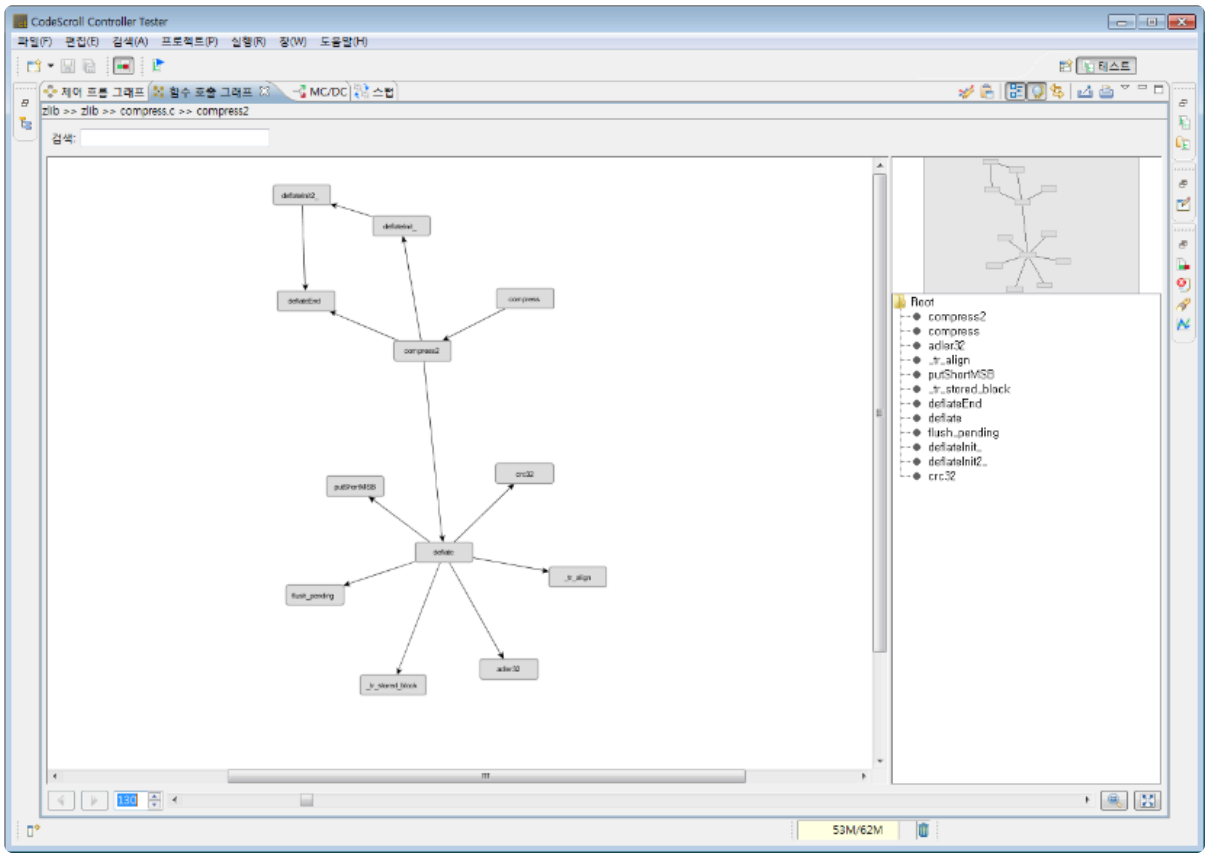
확대/축소 기능

뷰 하단에 숫자를 입력하거나 슬라이더를 조정해서 확대/축소 비율을 변경할 수 있습니다.

뷰 오른쪽 하단의 [확대/축소 비율 초기화] 버튼으로 확대/축소 비율을 초기화할 수 있고, [뷰 크기에 맞추기] 버튼으로 뷰의 크기에 맞게 확대/축소 비율을 변경할 수 있습니다.

14.11. 함수 호출 그래프 뷰

함수 호출 그래프 뷰는 선택된 함수를 중심으로 함수의 호출 정보를 그래프로 보여줍니다.
(예: 함수 'A'에서 함수 'B'를 호출했을 때, 노드 'A'에서 노드 'B'로의 에지로 표현하며, 여러 번 호출하더라도 하나의 에지로 표현)
노드(함수)를 선택하면 해당 노드를 중심으로 함수 호출 정보를 보여주게 됩니다.



툴바 메뉴

메뉴	설명
레이아웃 편집	그래프의 노드 위치를 수정할 수 있게 변경(그래프가 갱신되면 자동으로 편집 불가능)
시스템 클립보드에 복사	현재 화면에 보이는 그래프를 클립보드에 복사
아웃라인 표시	그래프를 트리 형태로 표시
오버뷰 표시	그래프의 오버뷰 표시
편집기와 연결	그래프에서 선택된 항목을 원클릭으로 편집기에 표시
뷰 내용 내보내기	뷰의 내용을 리포트로 내보내기
뷰 내용을 인쇄	뷰의 내용을 인쇄

풀다운 메뉴

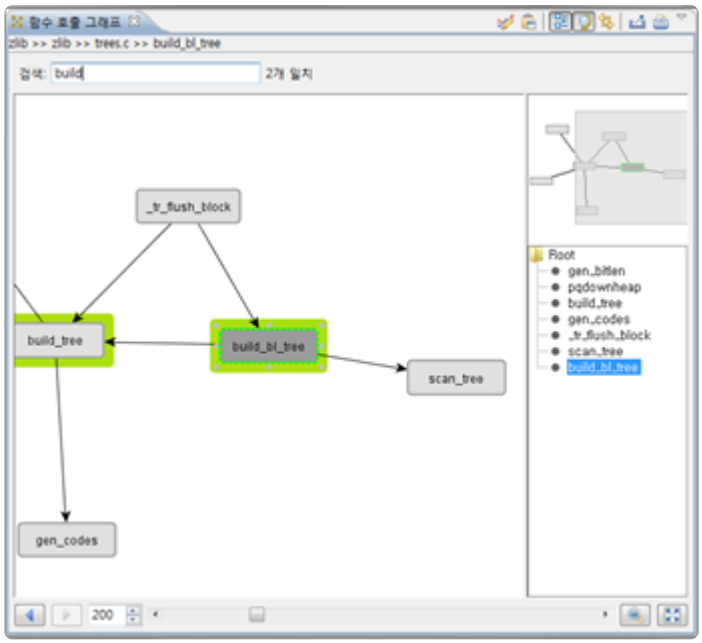
메뉴	설명
제어 흐름 그래프 보기	현재 뷰에 선택된 함수에 대한 제어 흐름 그래프 보기
아웃라인 표시	그래프를 트리 형태로 표시
오버뷰 표시	그래프의 오버뷰 표시
편집기와 연결	그래프에서 선택된 항목을 원클릭으로 편집기에 표시
그래프 포맷으로 저장	현재 화면에 보이는 그래프에 대한 그래프 모델 파일 생성 네 사지 포맷 지원: • Graph Modeling Language XML (*.xgml) • Graph Modeling Language(*.gml) • yWorks Binary Graph Format(.ygf) • Trivial Graph Format(*.tgf)
이미지 포맷으로 저장	현재 화면에 보이는 그래프를 이미지 포맷 파일(jpg, gif)로 저장
시스템 클립보드에 복사	현재 화면에 보이는 그래프를 클립보드에 복사
환경 설정	환경 설정 열기

노드 팝업 메뉴

메뉴	설명
제어 흐름 그래프 보기	현재 뷰에 선택된 함수에 대한 제어 흐름 그래프 보기
펼치기	선택한 함수 이후의 호출 관계를 한 단계 더 표시
접기	선택한 함수 이후의 호출 관계를 숨김
시작 노드로 설정	2개 함수의 호출 경로를 보기 위한 시작 함수 지정
종료 노드로 설정	2개 함수의 호출 경로를 보기 위한 종료 함수 지정, 현 그래프 상에서 가능한 모든 경로를 표시

검색 기능

뷰 상단에 검색어를 입력하면, 입력된 검색어를 포함하는 이름을 가진 함수들이 강조되어 보여집니다.

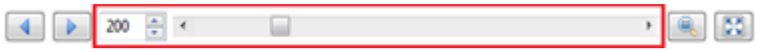


히스토리 기능

뷰 왼쪽 하단의 [화살표(뒤로 가기, 앞으로 가기)] 버튼으로 현재 그래프에서 선택했던 노드를 다시 볼 수 있습니다.

확대/축소 기능

뷰 하단에 숫자를 입력하거나 슬라이더를 조정해서 확대/축소 비율을 변경할 수 있습니다.



뷰 오른쪽 하단의 [확대/축소 비율 초기화] 버튼으로 확대/축소 비율을 초기화할 수 있고, [뷰 크기에 맞추기] 버튼으로 뷰의 크기에 맞게 확대/축소 비율을 변경할 수 있습니다.

펼치기/접기 기능

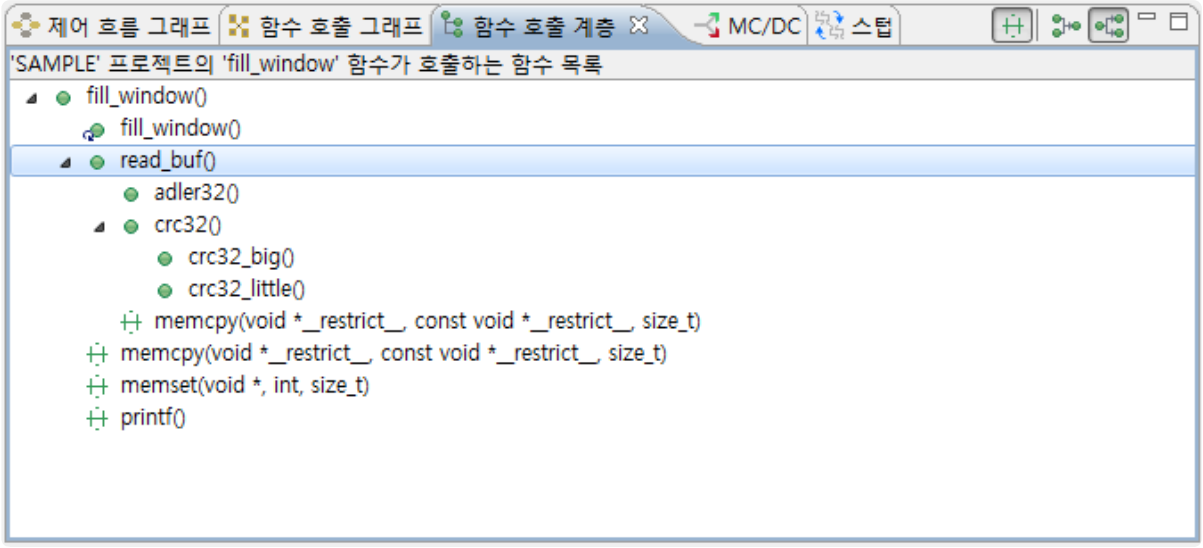
노드를 오른쪽 클릭하면 나타나는 메뉴를 통해 해당 노드에서 나가는 에지를 접거나 펼칠 수 있습니다. [접기]를 선택하면 선택한 노드에 해당하는 함수가 호출하는 모든 함수의 호출 관계를 숨깁니다. 함수 호출 정보가 없는 노드의 접기 메뉴는 비활성화됩니다. [펼치기]를 선택하면 선택된 노드에 해당하는 함수가 호출하는 함수들을 한 단계 더 보여줍니다. 마찬가지로 더 이상의 호출 관계가 없다면 펼치기 메뉴는 비활성화됩니다.

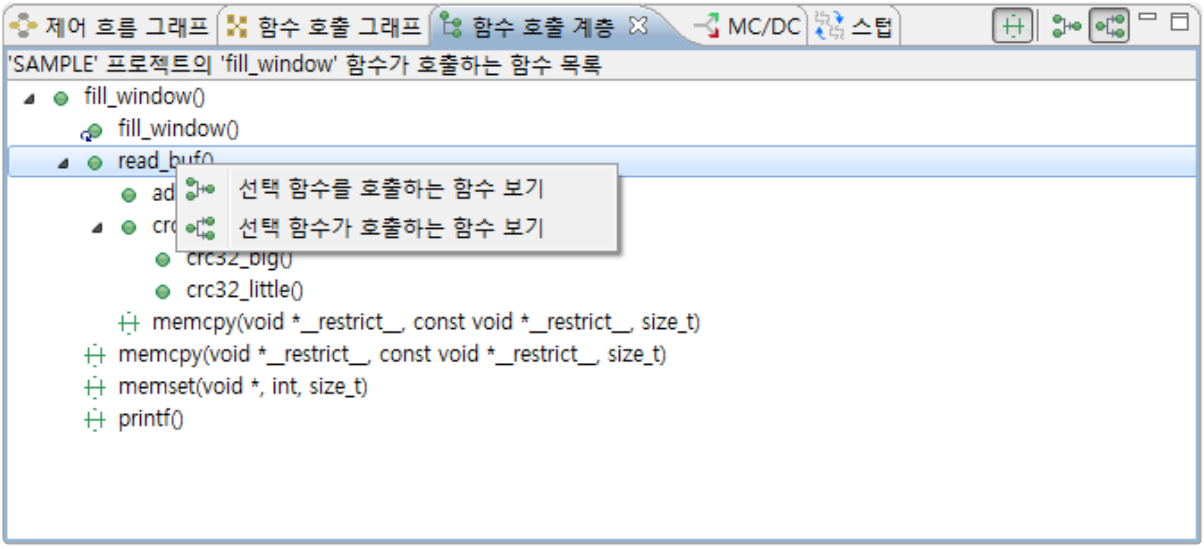
노드 사이 경로 강조 기능

시작 노드와 종료 노드를 선택하면, 두 노드 사이에 존재하는 모든 경로를 강조해서 보여줌으로써 복잡한 함수 호출 관계에서 관심 있는 부분을 쉽게 볼 수 있습니다.

14.12. 함수 호출 계층 뷰

함수 호출 계층 뷰는 선택된 함수가 호출하는 또는 호출된 함수 정보를 계층 구조로 보여줍니다.
계층 구조에 표시되는 함수를 선택하고 우 클릭하였을 때 나타나는 컨텍스트 메뉴를 이용하여 선택한 함수의 호출 정보로 변경할 수 있으며, 더블 클릭을 하면 함수가 정의된 소스 파일의 편집기가 열리고 선택한 함수의 위치로 이동합니다. (단, 시스템 함수인 경우에는 포함되지 않습니다.)

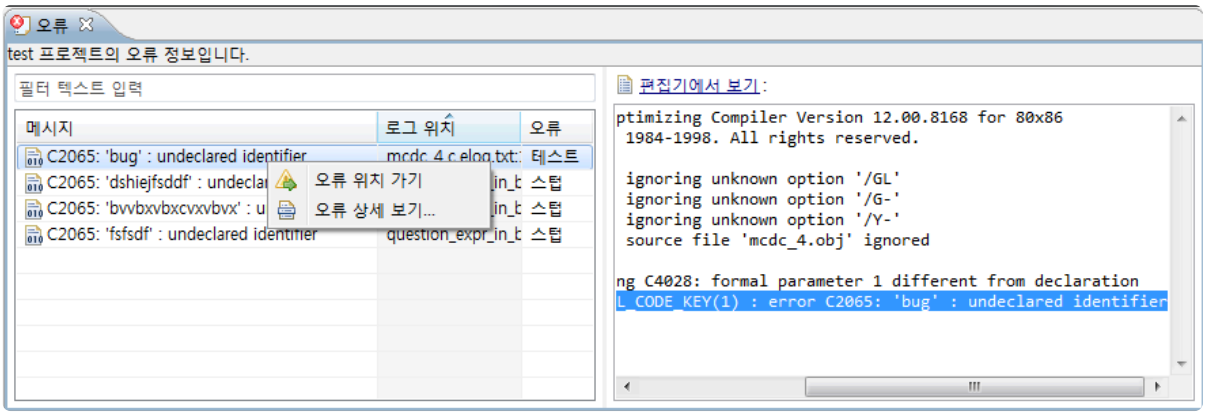




메뉴	설명
시스템 함수 보기	시스템 함수 호출 정보를 계층구조에 표시여부 설정
(선택한 함수를 호출하는 함수 보기)	선택한 함수를 호출하는 함수들을 계층 구조로 표시
(선택한 함수가 호출하는 함수 보기)	선택한 함수가 호출하는 함수들을 계층 구조로 표시

14.13. 오류 뷰

프로젝트의 분석 오류 정보를 상세하게 확인할 수 있습니다.



로그 타입

타입	설명
빌드	빌드 중 발생한 오류
구문 분석	구문 분석 중 발생한 오류
링크	링킹 중 발생한 오류
전처리	전처리 중 발생한 오류

오류 타입

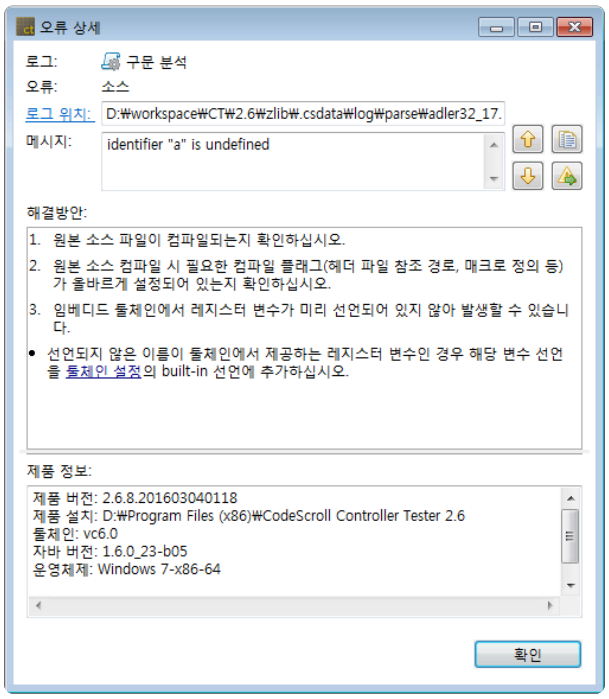
타입	설명
소스	소스 파일에서 발생한 오류
스텁	스텁 파일에서 발생한 오류
테스트	테스트 에디터에서 발생한 오류

오류 위치 가기

컨텍스트 메뉴에서 [오류 위치 가기]를 선택하면, 오류가 발생한 위치로 이동하여 쉽게 오류를 수정할 수 있습니다.

오류 상세 보기

오류 정보를 더블클릭하거나 컨텍스트 메뉴에서 [오류 상세 보기...]를 선택하면, 오류에 대한 상세 정보를 확인할 수 있습니다.



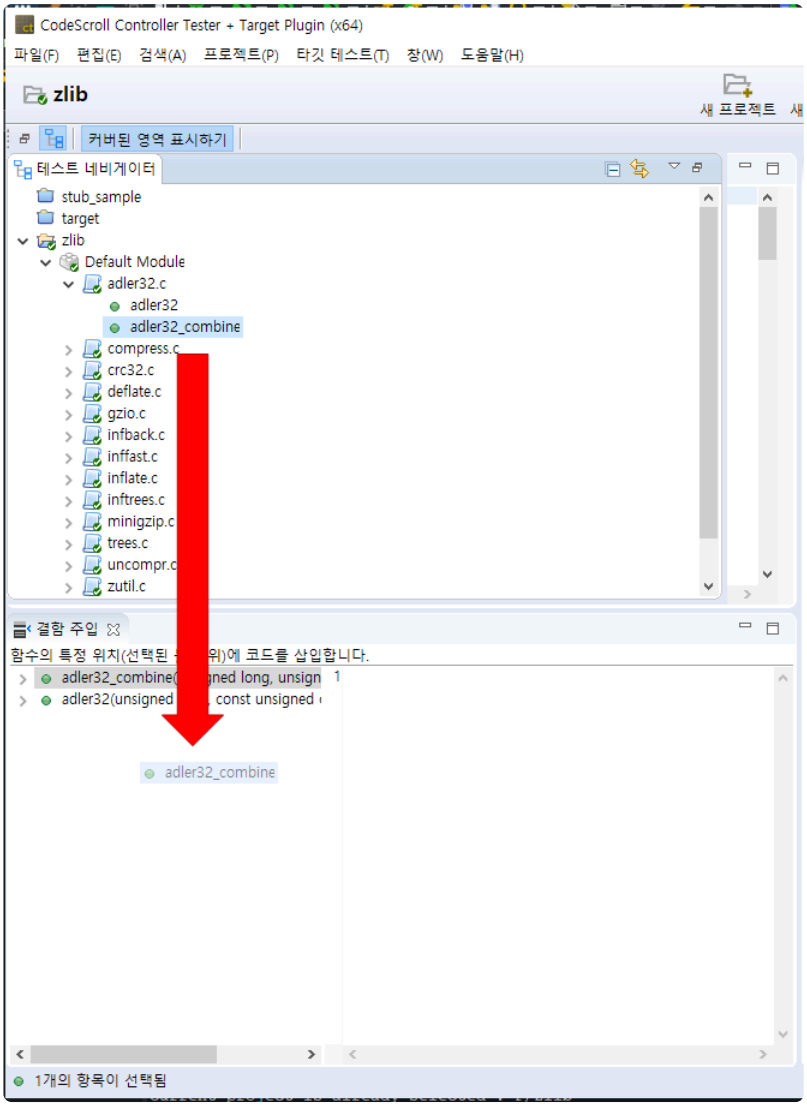
버튼 설명

버튼	설명
이전	이전 오류 정보 보기
다음	다음 오류 정보 보기
복사	오류 내용 복사하기
오류 위치 가기	오류 위치 가기

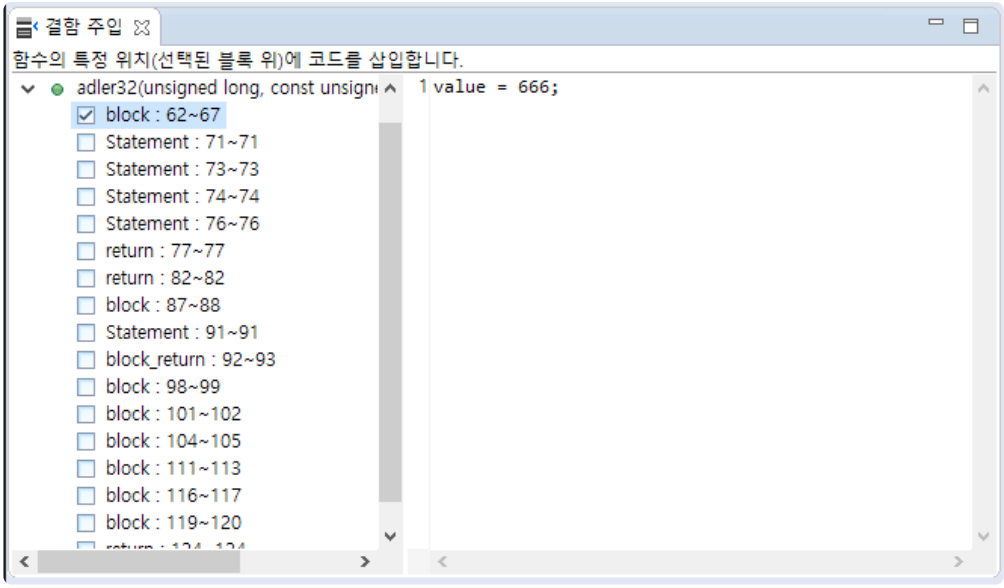
14.14. 결함 주입 뷰

결함 주입 뷰는 테스트 대상 함수의 특정영역에 사용자가 테스트에 필요한 결함을 추가로 삽입할 수 있는 기능을 제공하는 뷰입니다.

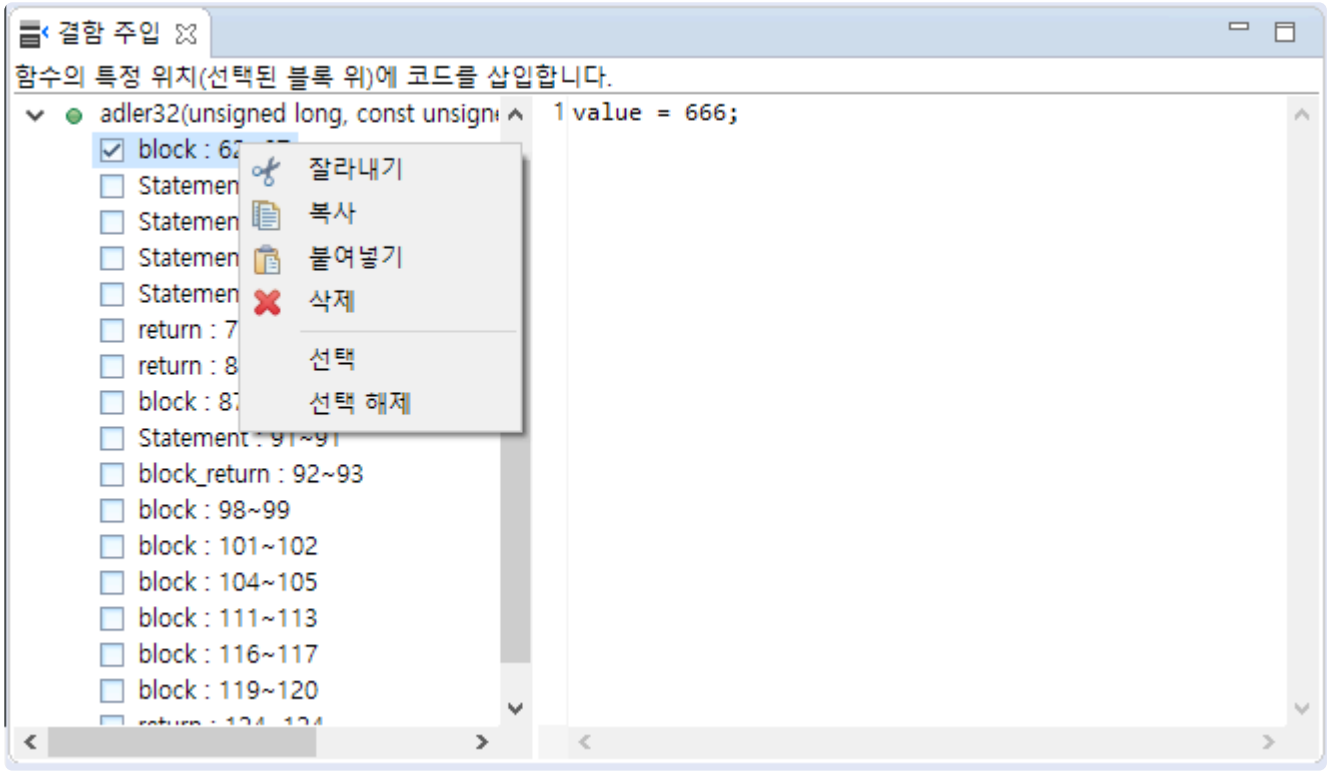
1. [테스트 네비게이터]에서 결함을 삽입할 함수를 [결함 주입 뷰]로 끌어서 놓습니다.



2. 뷰의 왼쪽 트리 구조에서 결함을 삽입 할 노드에 체크박스를 클릭한 후, 우측 편집 창에 사용자 코드를 입력합니다.
3. [유닛 테스트 실행] 버튼을 클릭하여 사용자가 삽입한 결함을 반영한 테스트를 수행할 수 있습니다.



결함 주입 뷰 컨텍스트 메뉴

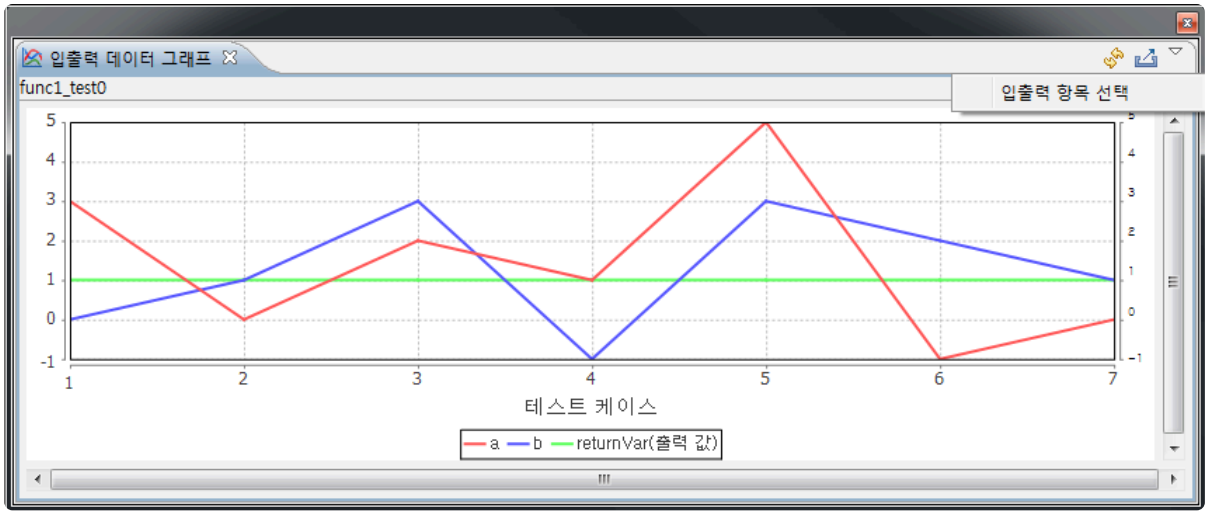


메뉴 이름	설명
잘라내기	사용자가 삽입한 코드를 잘라내기
복사	사용자가 삽입한 코드를 복사
붙여넣기	사용자가 삽입한 코드를 붙여넣기
삭제	사용자가 삽입한 코드를 삭제

선택	선택된 노드의 체크박스를 체크하기
선택 해제	선택된 노드의 체크박스를 체크 해제하기

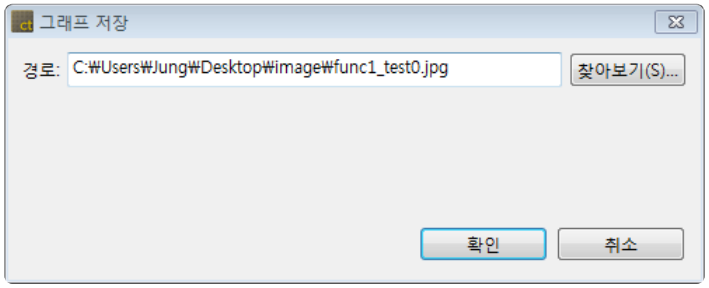
14.15. 입출력 데이터 그래프 뷰

Controller Tester에서는 입력값/기대값/출력값 데이터를 그래프 형식으로 제공합니다.
그래프의 가로축은 테스트 케이스 번호, 세로축은 테스트 케이스 데이터를 의미합니다.



툴바 아이콘	설명
새로고치기	입출력 데이터 새로 고치기
저장	입출력 데이터 그래프 저장

[그래프 저장]을 클릭하면 그래프 저장 경로를 입력할 수 있는 알림 창을 출력합니다. 그래프가 저장될 경로, 파일 이름 및 파일 형식을 입력하고 [확인] 버튼을 클릭하면 그래프 저장이 완료됩니다.

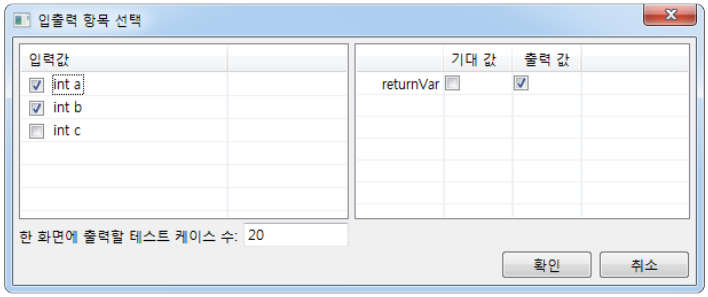


그래프 설정

풀다운 메뉴(▽)에서 [입출력 항목 선택]을 선택하면 그래프 설정 창을 출력합니다.

- 체크박스에 선택 된 항목에 대해서 [입력값], [출력값], [기대값]이 입출력 데이터 그래프에 출력이 됩니다. 단, 기대값을 ~, &, |, ! 등으로 지정 시 입출력 데이터 그래프 뷰에 기대값이 반영되지 않습니다.
- [한 화면에 출력할 테스트 케이스 수]에는 사용자가 한 화면에 출력할 테스트 케이스 개수를 지정할 수 있습

니다.


입출력 항목 선택
입력값
int a
int b
int c
기대 값
출력 값
returnVar
한 화면에 출력할 테스트 케이스 수:
20
확인
취소

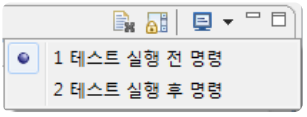
14.16. 콘솔 뷰

[프로젝트] -> [특성] -> [유닛 테스트] -> [외부 명령 실행]에 [테스트 실행 전 명령], [테스트 실행 후 명령]에 명령어 또는 배치 파일을 입력하고, 유닛 테스트를 실행하면 콘솔 뷰에 테스트 실행 전, 후 외부 명령 수행 결과가 나타납니다.

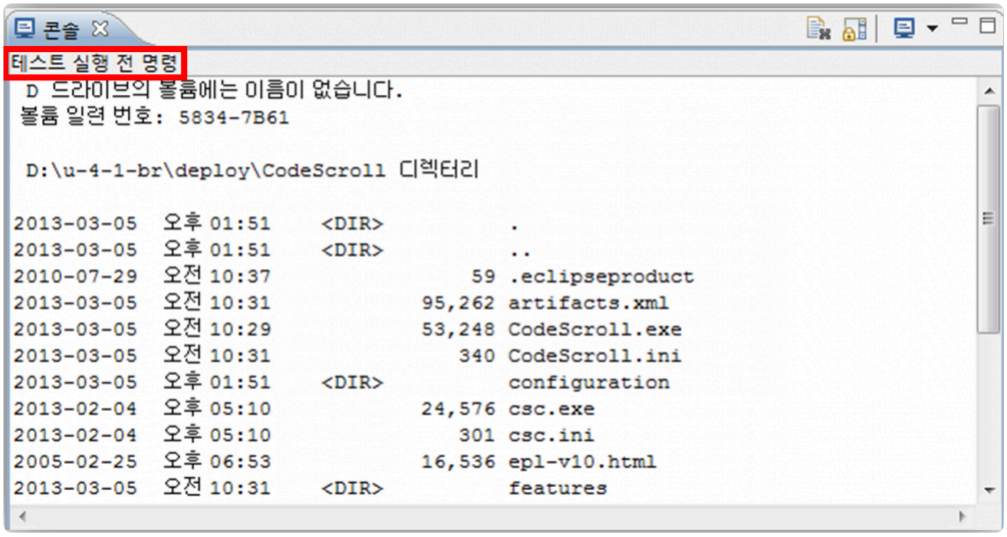
콘솔 뷰 툴바 아이콘

툴바 아이콘	설명
콘솔 지우기	콘솔 내용 지우기
화면 이동 잠금	콘솔 화면 이동 잠금
선택된 콘솔 표시	선택된 콘솔 표시

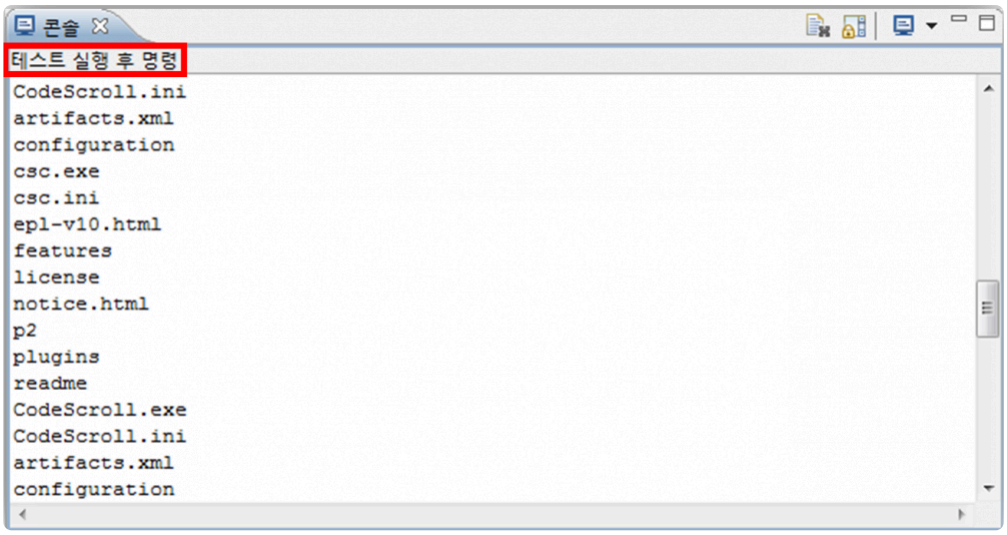
[선택된 콘솔 표시]를 선택하면 [테스트 실행 전 명령]과 [테스트 실행 후 명령] 2개의 메뉴가 나타납니다.



[테스트 실행 전 명령]을 선택하면 테스트 실행 전 외부 명령 수행 결과가 콘솔 뷰에 출력됩니다.



[테스트 실행 후 명령]을 선택하면 테스트 실행 후 외부 명령 수행 결과가 콘솔 뷰에 출력됩니다.


콘솔
테스트 실행 후 명령
CodeScroll.ini
artifacts.xml
configuration
csc.exe
csc.ini
epl-v10.html
features
license
notice.html
p2
plugins
readme
CodeScroll.exe
CodeScroll.ini
artifacts.xml
configuration

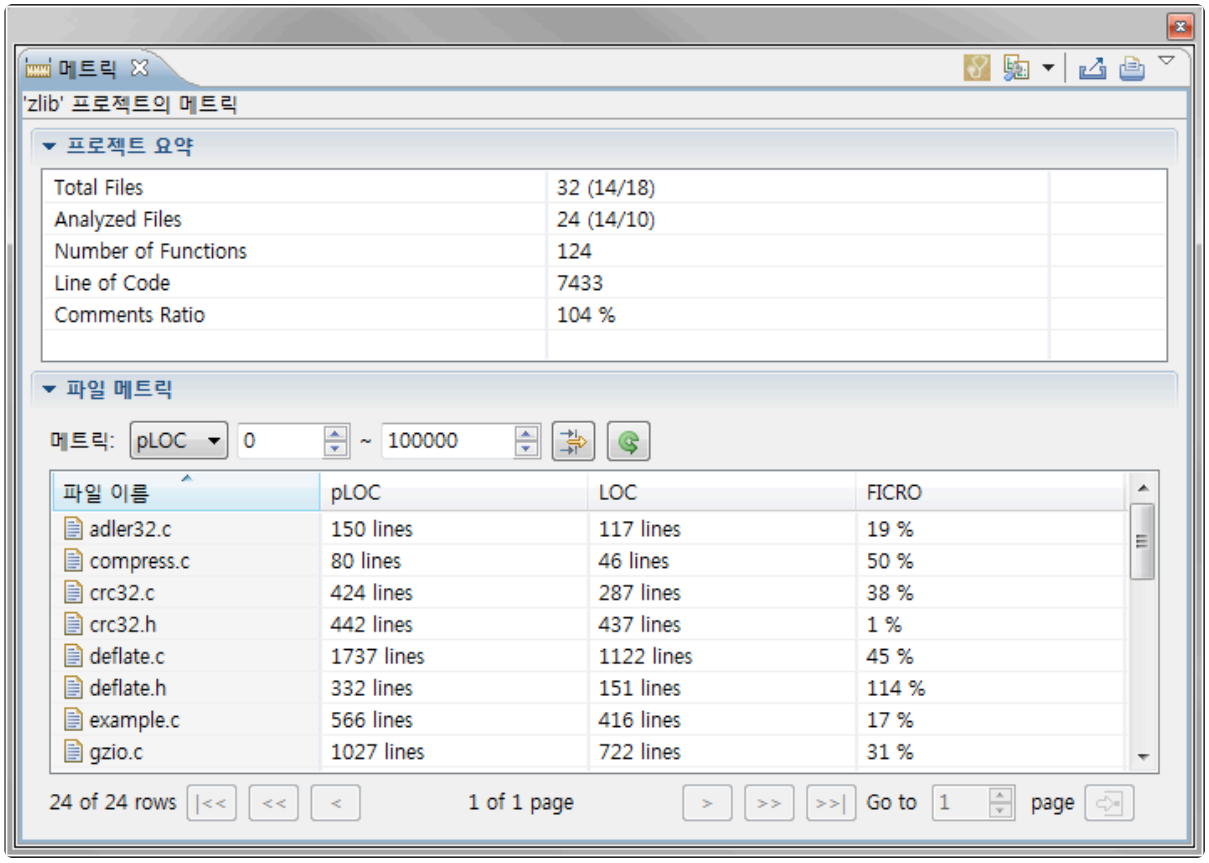
15. 분석 퍼스펙티브

분석 퍼스펙티브를 구성하는 요소는 다음과 같습니다.

- 메트릭 뷰
- 메트릭 차트 뷰
- 메트릭 탑 차트 뷰
- 메트릭 바 차트 뷰
- 메트릭 진단 차트 뷰
- 제어 흐름 그래프 뷰
- 함수 호출 그래프 뷰
- 미사용 함수 뷰
- 소스 헤더 연관 뷰
- 전역 변수 연관 뷰
- 테스트 네비게이터 뷰

15.1. 메트릭 뷰

메트릭 뷰는 프로젝트의 메트릭을 측정한 결과를 보여줍니다. 프로젝트 전체 메트릭 요약 정보와, 프로젝트에 속한 각 항목들(모듈, 파일, 클래스, 함수)의 메트릭 정보를 보여줍니다.



아이콘 설명

아이콘	설명
	모듈(프로젝트를 기능별로 분할한 논리적인 단위)
	파일(프로젝트에 속한 소스 파일과 헤더 파일)
	클래스
	함수

툴바 메뉴

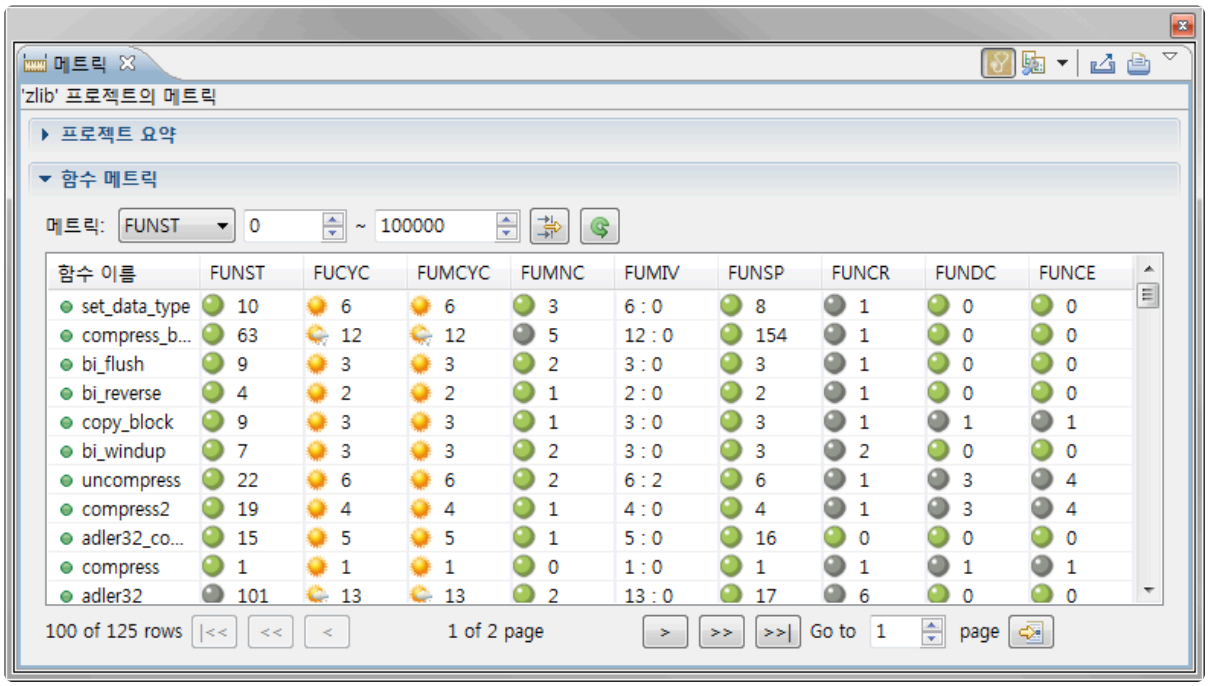
메뉴	설명
진단 표시	메트릭 값의 좌측에 해당 값에 대한 진단 이미지 표시

모드 변경	보기 모드를 변경 (모듈, 파일, 클래스, 함수)
뷰 내용 내보내기	뷰의 내용을 리포트로 내보내기
뷰 내용 인쇄	뷰의 내용을 인쇄

풀다운 메뉴

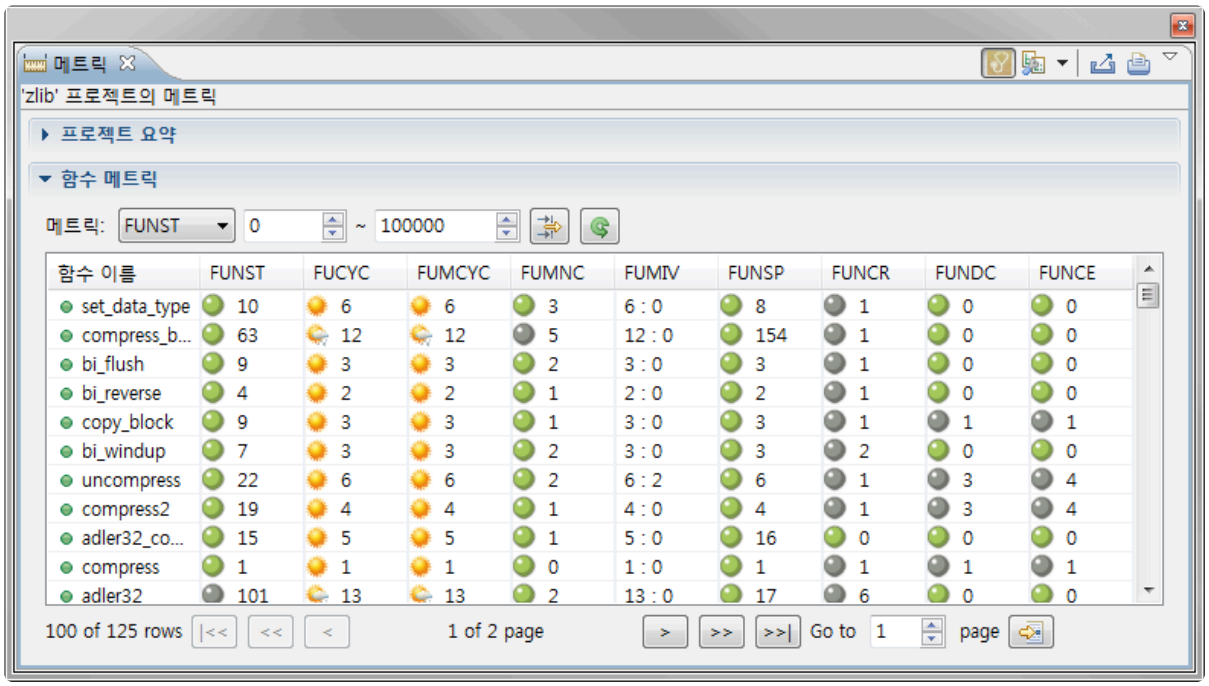
메뉴	설명
정렬	선택한 항목을 기준으로 정렬(테이블의 열 이름을 클릭해도 동일하게 정렬됨)
열	열의 순서와 너비를 변경(테이블의 열 이름 부분을 끌어서 놓기로 순서 변경 가능)
진단 정보 구성	진단 환경 설정 페이지를 띄움(각각의 메트릭에 대한 진단 단계와 범위 및 이미지 설정 가능)
메트릭 숨김/표시 설정	메트릭 표시 여부 설정
메트릭 뷰 옵션	메트릭 뷰의 '페이지 당 줄 수'를 설정

필터링



필터링 기능으로 특정 메트릭에 대하여 원하는 범위에 있는 항목들만 볼 수 있습니다.
메트릭을 선택하고 범위 값을 넣은 뒤에 우측의 [필터링] 버튼을 클릭하면, 입력한 정보에 해당하는 값들만 보여집니다.
필터링 이전 상태로 돌아가고 싶다면 [필터링] 버튼 우측의 [필터링 기준 초기화] 버튼을 클릭합니다.

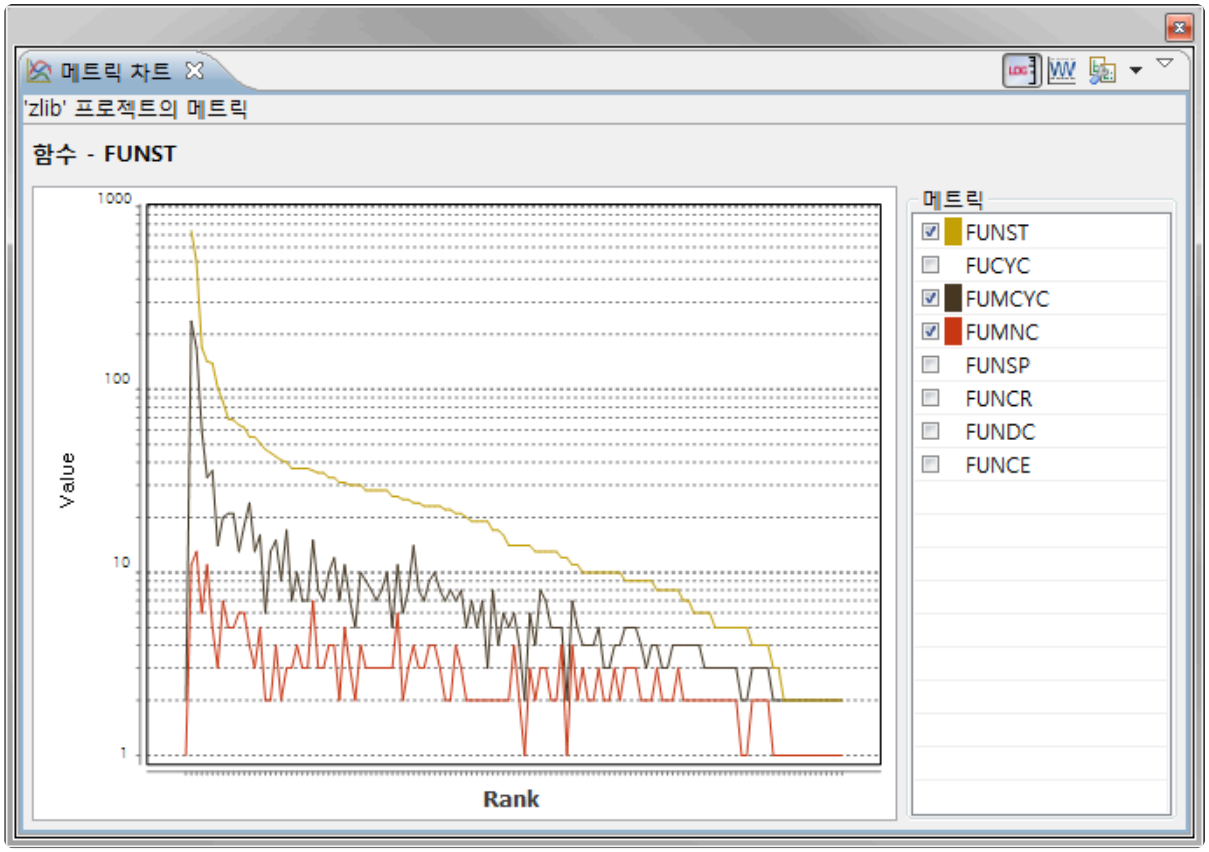
페이징



페이징 기능으로 많은 항목들을 편리하게 볼 수 있습니다.

15.2. 메트릭 차트 뷰

메트릭 차트 뷰는 메트릭 결과를 꺾은선 차트로 보여줍니다. x 축은 각각의 항목들이고, y축은 항목의 메트릭 값입니다. 프로젝트에서 서로 다른 메트릭 사이의 값의 분포를 비교할 수 있습니다.



툴바 메뉴

메뉴	설명
로그 스케일 표시	차트의 y축을 로그 단위로 변경
표준화된 메트릭 표시	표준화된 메트릭 값을 표시
모드 변경	보기 모드를 변경 (모듈, 파일, 클래스, 함수)

풀다운 메뉴

메뉴	설명
정렬	선택한 항목을 기준으로 정렬(테이블에서 메트릭을 선택해도 동일하게 정렬됨)
메트릭 숨김/표시 설정	메트릭 표시 여부 설정

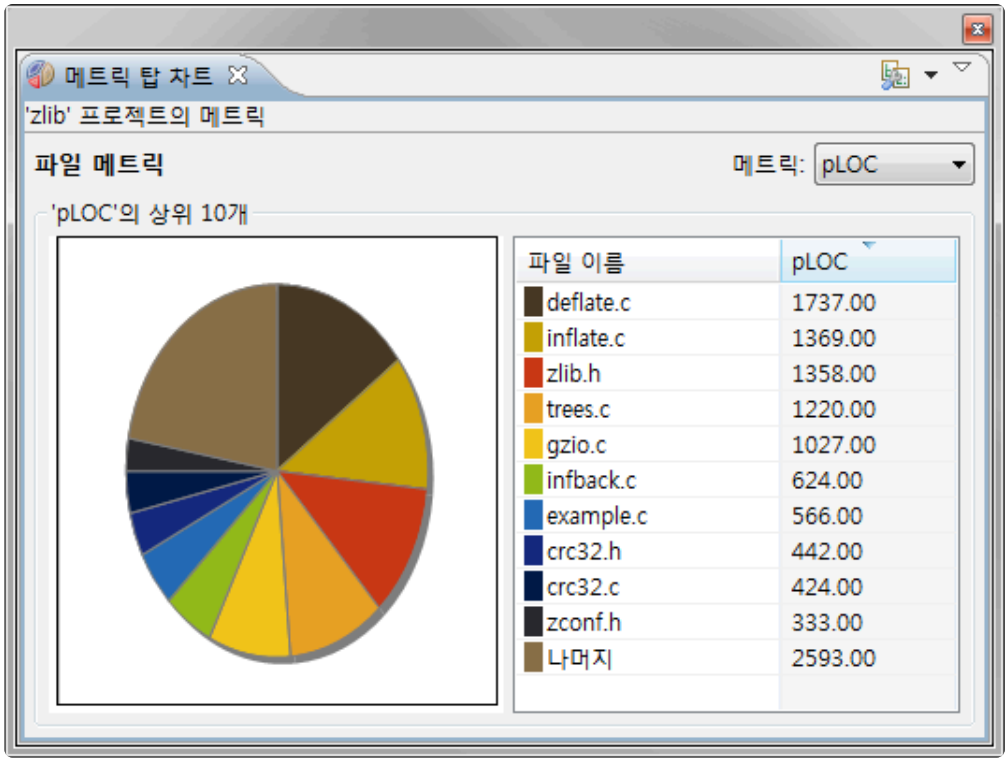
메트릭 뷰 옵션	메트릭 뷰와 연동 기능에 대한 설정

메트릭 선택

우측의 테이블에서 메트릭 이름 좌측의 체크 박스를 체크하면 체크된 메트릭이 차트에 보여집니다. 체크된 메트릭 중에서 메트릭 이름 부분을 클릭해서 선택하면, 해당 메트릭 값을 기준으로 항목들이 정렬됩니다.

15.3. 메트릭 탑 차트 뷰

메트릭 탑 차트 뷰는 메트릭 값의 상위 항목들을 파이 차트로 보여줍니다. 전체에서 상위 항목들이 차지하는 비중을 볼 수 있습니다.



메뉴	설명
모드 변경	보기 모드를 변경 (모듈, 파일, 클래스, 함수)

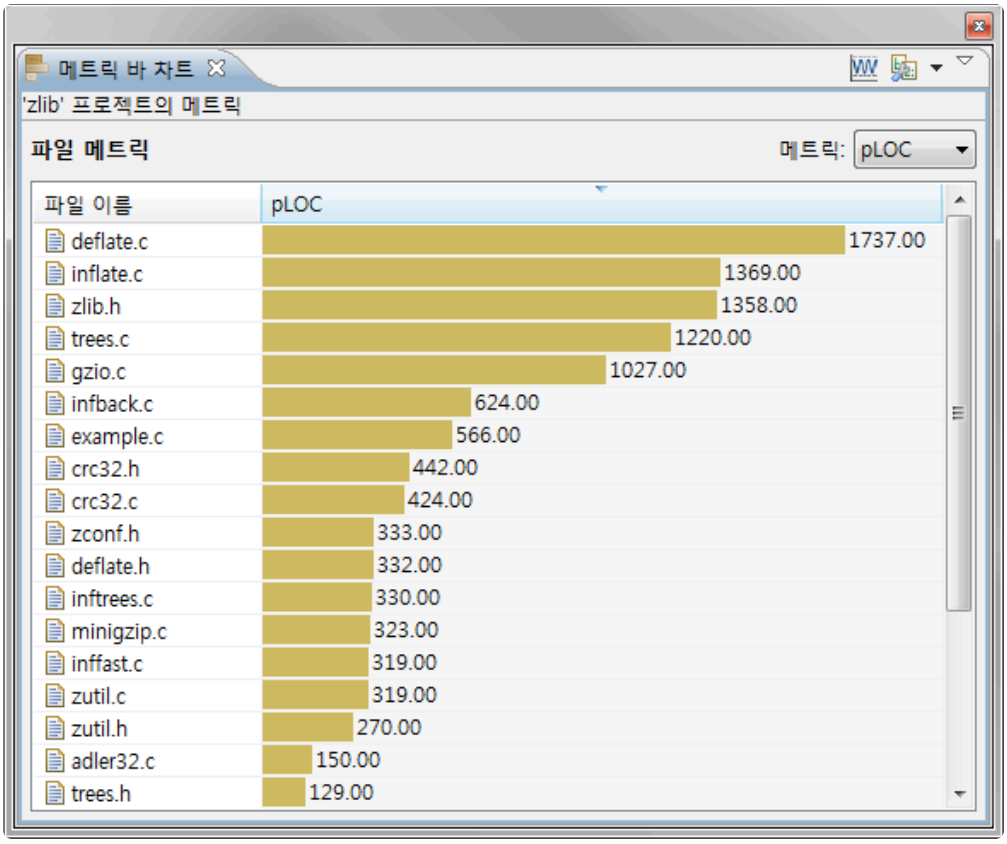
풀다운 메뉴

메뉴	설명
메트릭 숨김/표시 설정	메트릭을 숨길지 또는 표시할지 설정
메트릭 뷰 옵션	표시할 최상위 항목 개수와 나머지 항목 표시 여부 설정

파이 차트의 선택과 테이블의 선택된 항목이 서로 연동됩니다.

15.4. 메트릭 바 차트 뷰

메트릭 바 차트 뷰는 메트릭 값을 바 차트로 보여줍니다. 각 항목들의 값의 크기를 바 차트로 확인할 수 있습니다.



아이콘 설명

아이콘	설명
	모듈(프로젝트를 기능별로 분할한 논리적인 단위)
	파일(프로젝트에 속한 소스 파일과 헤더 파일)
	클래스
	함수

툴바 메뉴

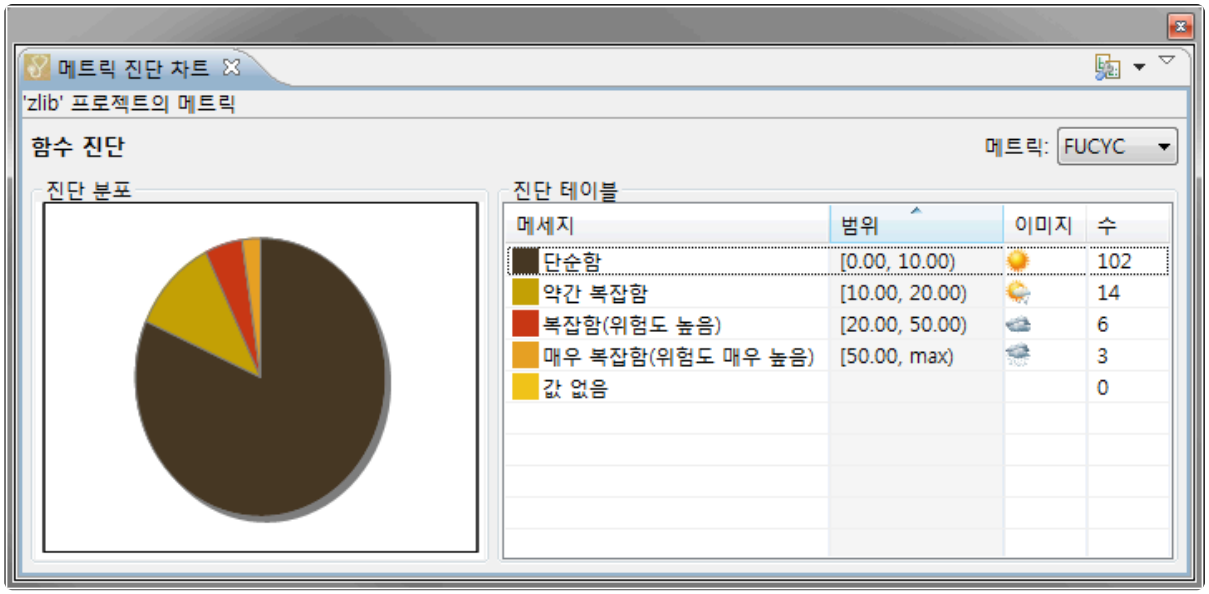
메뉴	설명
표준화된 메트릭 표시	표준화된 메트릭 값을 표시
모드 변경	보기 모드를 변경 (모듈, 파일, 클래스, 함수)

풀다운 메뉴

메뉴	설명
메트릭 숨김/표시 설정	메트릭 표시여부 설정
메트릭 뷰 옵션	표시할 항목의 개수 설정

15.5. 메트릭 진단 차트 뷰

메트릭 진단 차트 뷰는 진단 범위에 속하는 항목의 수를 파이 차트로 보여줍니다. 프로젝트 내에서 각 진단 범위에 항목들의 수를 확인할 수 있습니다.



툴바 메뉴

메뉴	설명
모드 변경	보기 모드를 변경 (모듈, 파일, 클래스, 함수)

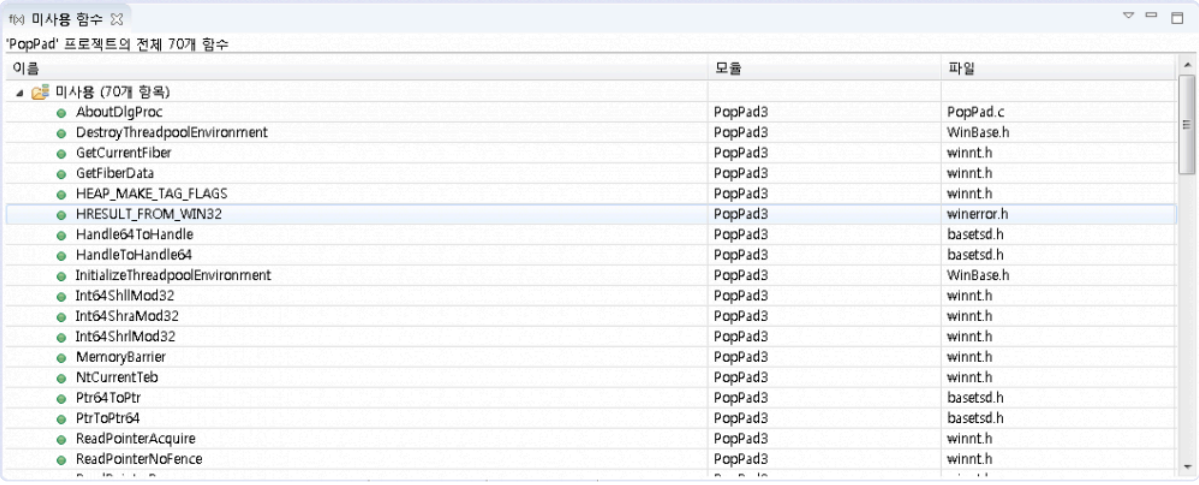
풀다운 메뉴

메뉴	설명
진단 정보 구성	진단 환경 설정 페이지를 띄움(각각의 메트릭에 대한 진단 단계와 범위 및 이미지 설정 가능)
메트릭 숨김/표시 설정	메트릭 표시 여부 설정
메트릭 뷰 옵션	진단 값이 없는 항목 표시 여부 설정

파이 차트의 선택과 테이블의 선택된 항목이 서로 연동됩니다.

15.6. 미사용 함수 뷰

미사용 함수 뷰는 선택된 프로젝트에 속한 함수들을 정해진 기준으로 분류하여 보여줍니다. 현재 미사용 함수들을 보여주고 있으며, 분류하는 기준은 추가될 수 있습니다.



아이콘 설명

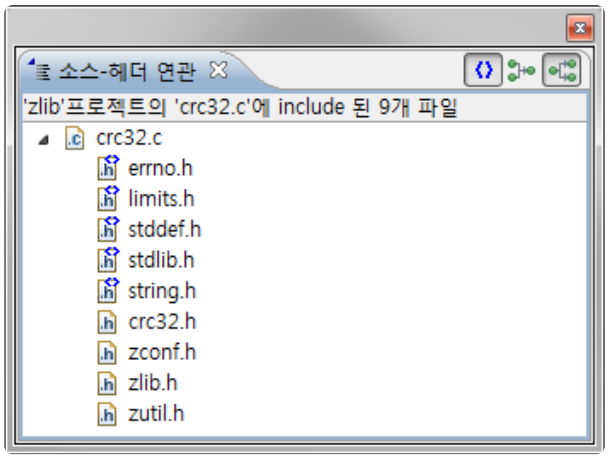
아이콘	설명
	분류 기준
	함수 그룹
	함수

풀다운 메뉴

메뉴	설명
함수 뷰 옵션	그룹별로 보여줄 함수의 수 설정

15.7. 소스 헤더 연관 뷰

소스-헤더 연관 뷰는 소스 파일과 헤더 파일 사이의 `<include>` 와 `#include` 관계를 보여줍니다.



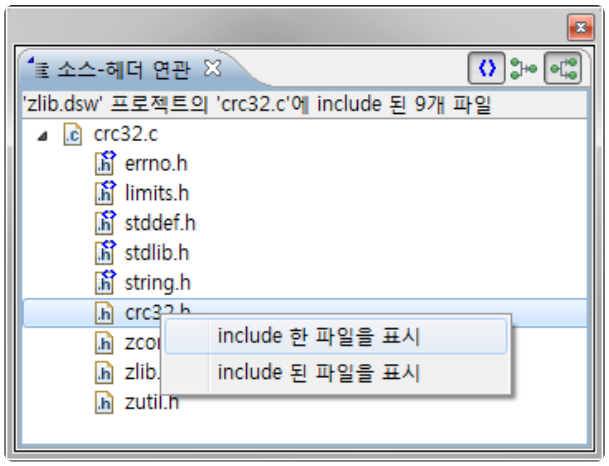
아이콘 설명

아이콘	설명
	소스 파일
	헤더 파일
	시스템 헤더 파일

툴바 메뉴

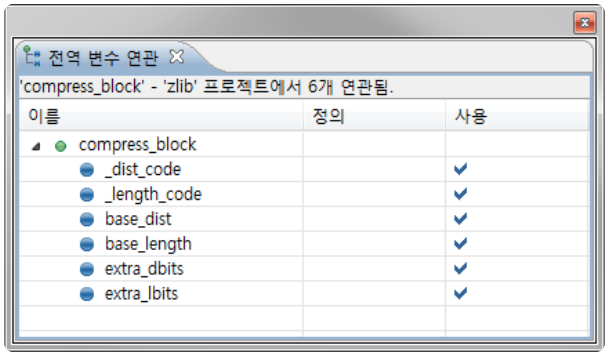
메뉴	설명
시스템 헤더 표시	시스템 헤더를 표시
include한 파일 표시	선택한 파일을 include한 파일을 표시
include된 파일 표시	선택한 파일에 include된 파일을 표시

include 한 파일을 표시, include 된 파일을 표시
뷰에서 파일을 오른쪽 클릭하면 나타나는 메뉴를 통해, 선택한 파일을 include 한 파일을 표시하거나, 선택한 파일이 include 된 파일을 표시할 수 있습니다.


소스-헤더 연관
'zlib.dsw' 프로젝트의 'crc32.c'에 include 된 9개 파일
crc32.c
errno.h
limits.h
stddef.h
stdlib.h
string.h
include 한 파일을 표시
include 된 파일을 표시
zutil.h

15.8. 전역 변수 연관 뷰

전역 변수 연관 뷰는 전역 변수가 어떤 함수에서 정의되고 사용되었는지를 보여줍니다. 함수가 선택되면 해당 함수에서 정의되거나 사용된 전역 변수를 보여주고, 전역 변수가 선택되면 해당 전역 변수가 정의되거나 사용된 함수를 보여줍니다.

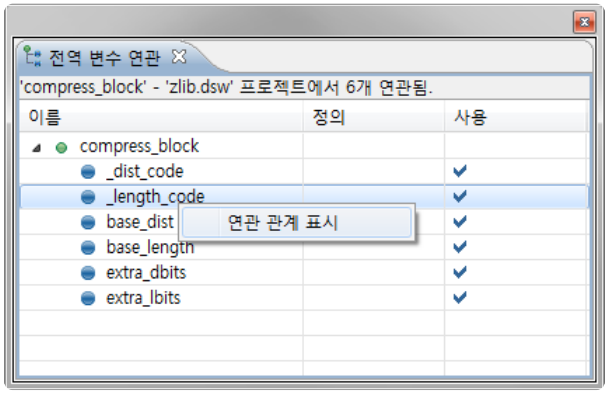


아이콘 설명

아이콘	설명
	함수
	전역 변수

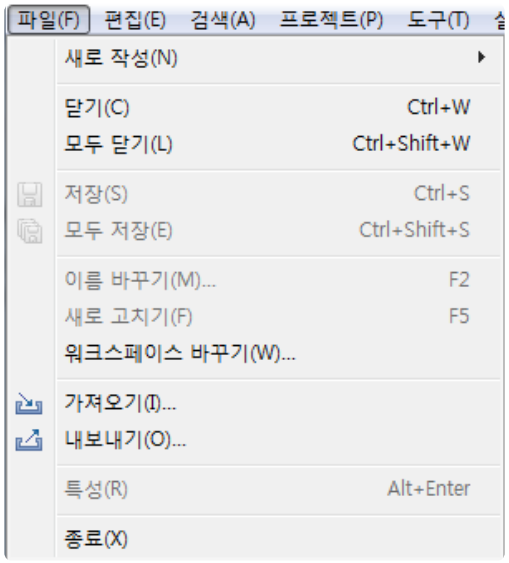
연관 관계 표시

뷰에서 연관된 항목을 오른쪽 클릭하면 나타나는 메뉴를 통해, 선택한 항목을 중심으로 다시 연관 관계를 볼 수 있습니다.



16. 파일 메뉴

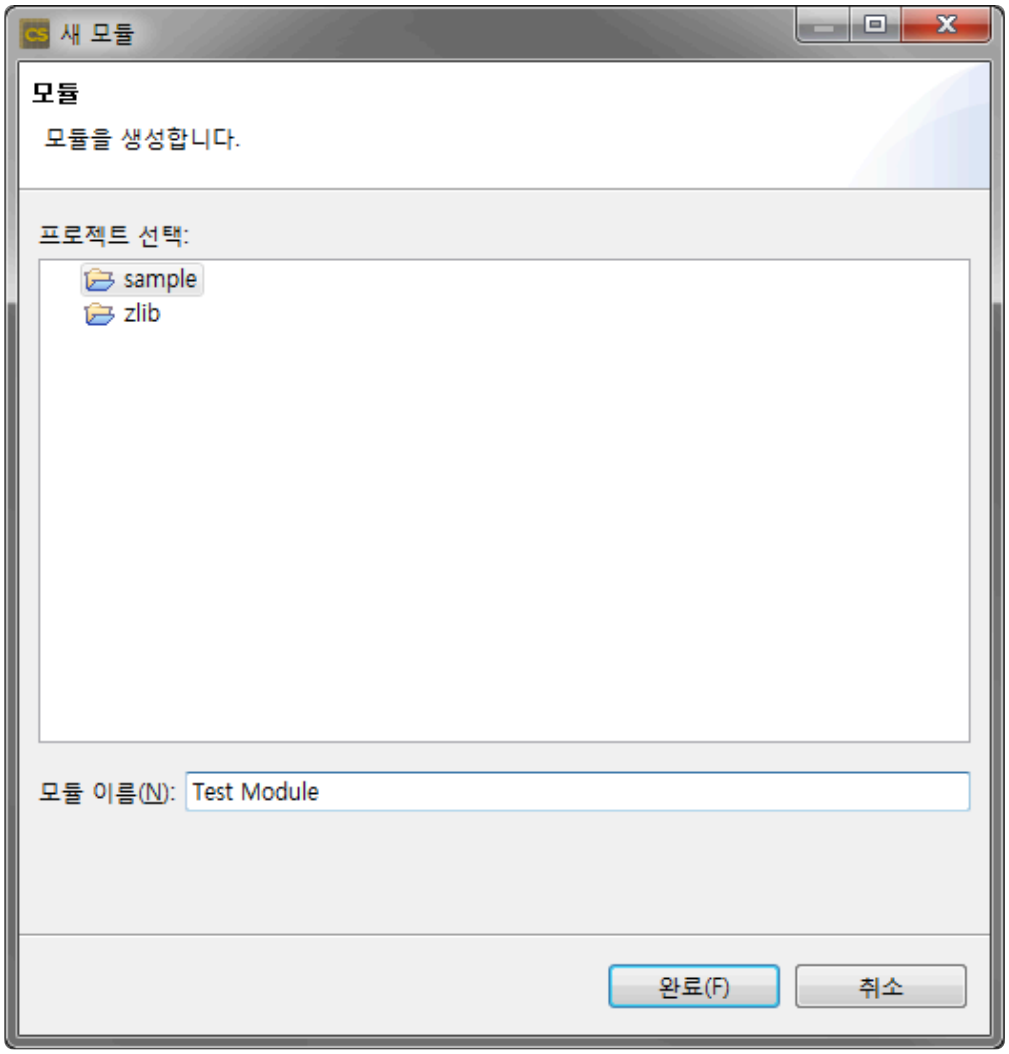
파일 메뉴에서는 새 프로젝트 또는 모듈 생성, 편집기 관련 작업, 새로 고치기, 워크스페이스 바꾸기, 가져오기와 내보내기, 선택된 항목의 특성 보기, 도구 종료 등을 수행할 수 있습니다.



- 새로 작성:
  - 프로젝트 생성
  - 모듈 생성
- 닫기/모두 닫기/저장/모두 저장: 편집기 관련 메뉴입니다.
- 이름 바꾸기: 프로젝트 또는 모듈의 이름을 변경할 수 있습니다.
- 워크스페이스 바꾸기
- 내보내기
- 가져오기
- 종료: 도구를 종료합니다.

16.1. 모듈 생성

프로젝트에 새 모듈을 생성할 수 있습니다. 모듈 이름을 제외한 다른 정보는 대상 프로젝트에 이미 존재하는 모듈과 같게 생성됩니다.

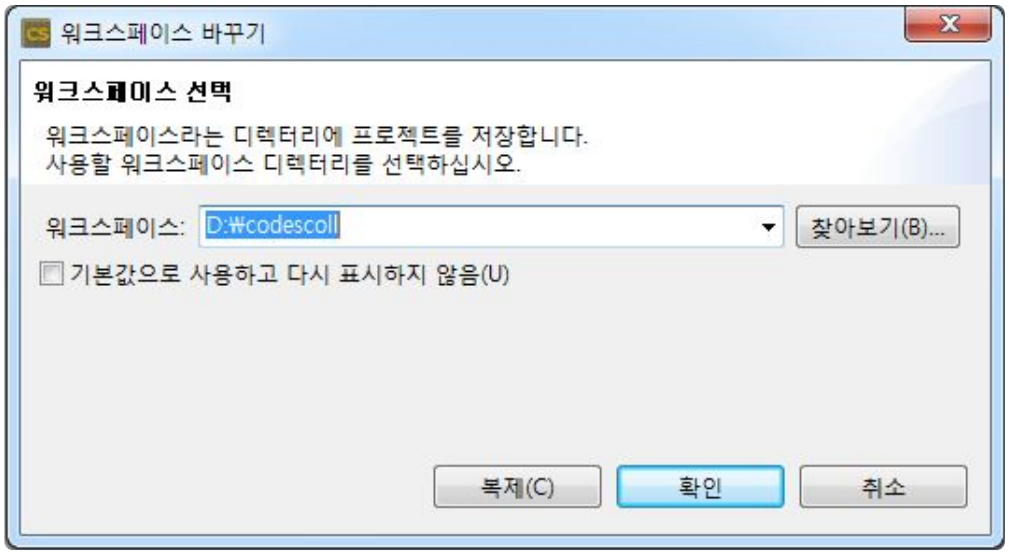


1. [새로 작성] 메뉴에서 [기타...]를 클릭한 뒤에, [기타] 분류의 [모듈]을 선택하고 [다음]을 클릭합니다.
2. 새 모듈을 생성할 프로젝트를 선택합니다.
3. 모듈 이름을 입력합니다. 이미 존재하는 모듈의 이름은 사용할 수 없습니다.
4. [완료]를 클릭하면 모듈이 생성됩니다.

테스트 네비게이터 뷰에서 프로젝트를 오른쪽 클릭하면 나타나는 [새로 작성] 메뉴를 통해서도 모듈을 생성할 수 있습니다.

16.2. 워크스페이스 바꾸기

워크스페이스 디렉터리를 변경할 수 있습니다. 워크스페이스를 변경하면 변경된 워크스페이스로 도구가 다시 시작됩니다.



[찾아보기]를 클릭하여 변경할 워크스페이스 디렉터리를 선택하거나, 수동으로 입력할 수 있습니다. [▼]를 클릭하면 나타나는 리스트에서 이전에 선택했던 워크스페이스 디렉터리를 선택할 수 있습니다.
[기본값으로 사용하고 다시 표시하지 않음]을 선택하면 다음에 도구를 실행할 때 워크스페이스 디렉터리 선택을 다시 묻지 않습니다.
[복제]를 클릭하면 선택한 워크스페이스를 다른 디렉터리로 복제합니다.

16.3. 내보내기

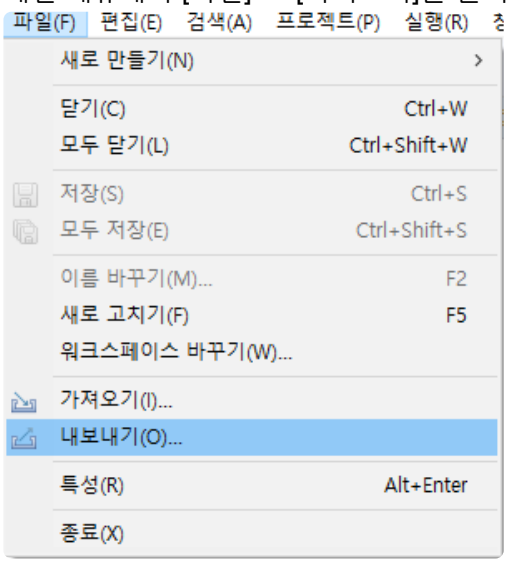
프로젝트 내보내기

프로젝트 설정과 테스트를 포함하여 프로젝트를 내보낼 수 있습니다.

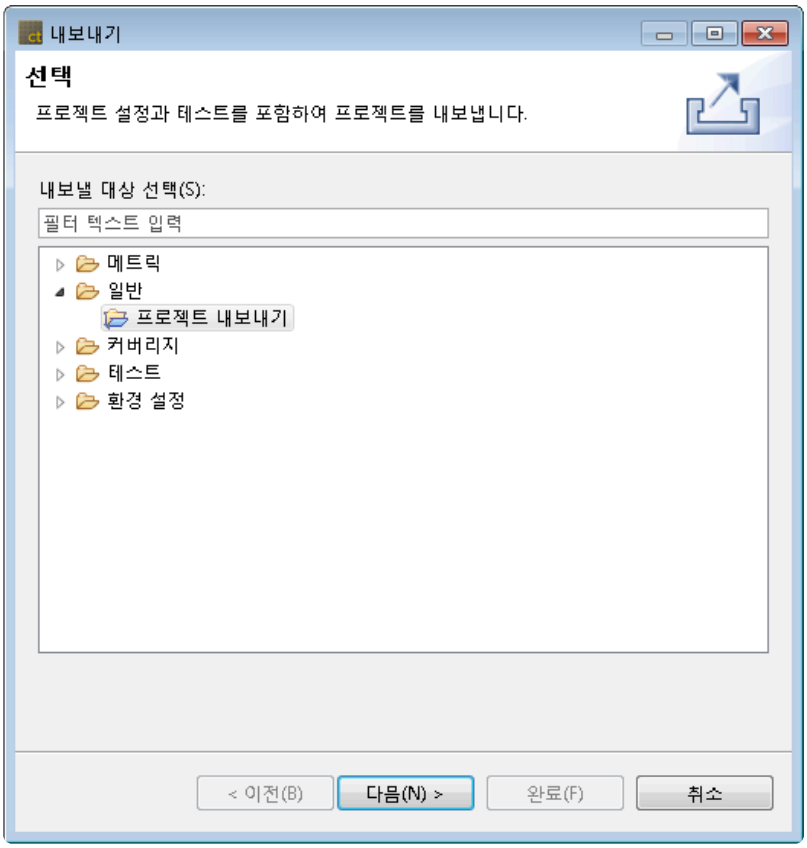
내보낸 프로젝트는 projectIE.exe를 이용하여 CLI 환경에서 프로젝트를 가져올 수 있습니다.

> 자세한 사항은 projectIE 문서를 참고하시기 바랍니다.

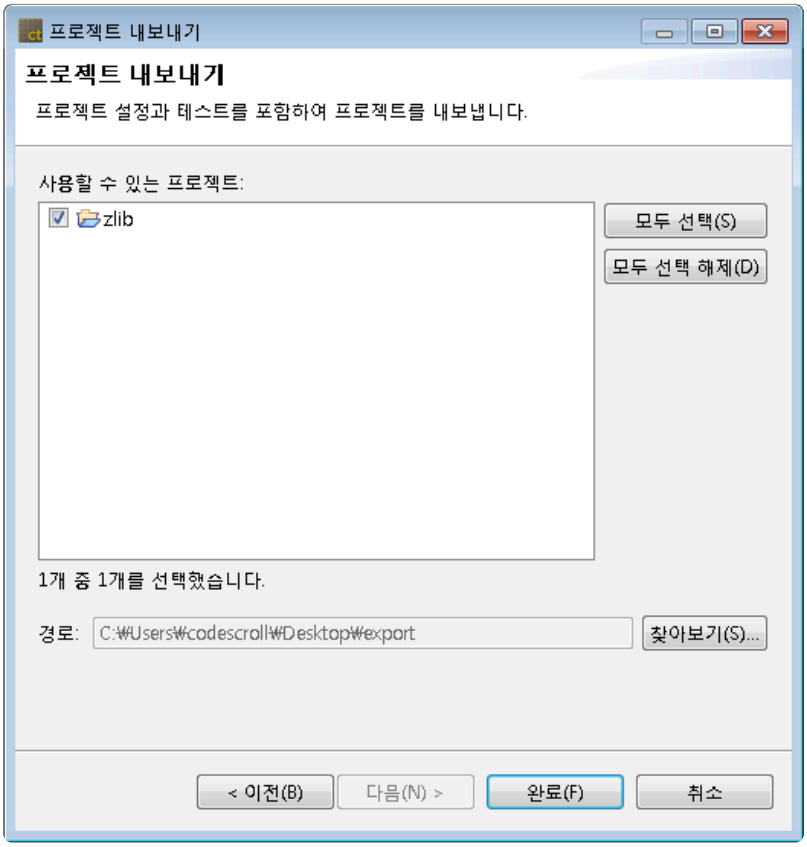
1. 메인 메뉴에서 [파일] -> [가져오기]를 클릭합니다. 내보내기 마법사가 열립니다.



2. [일반] -> [프로젝트 내보내기]를 클릭합니다.



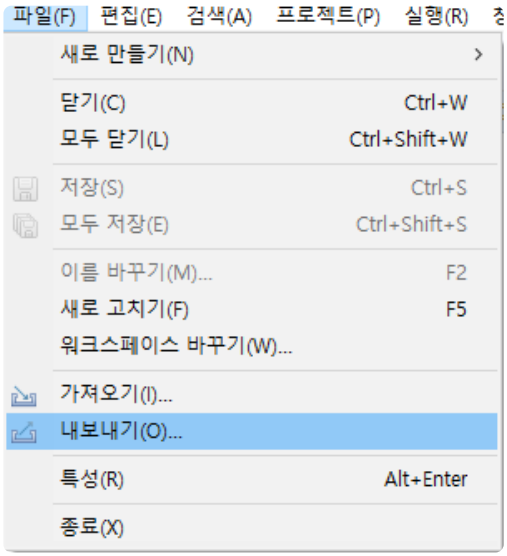
3. 내보내기 대상 프로젝트와 내보낼 경로를 선택한 후, [완료] 버튼을 클릭합니다.



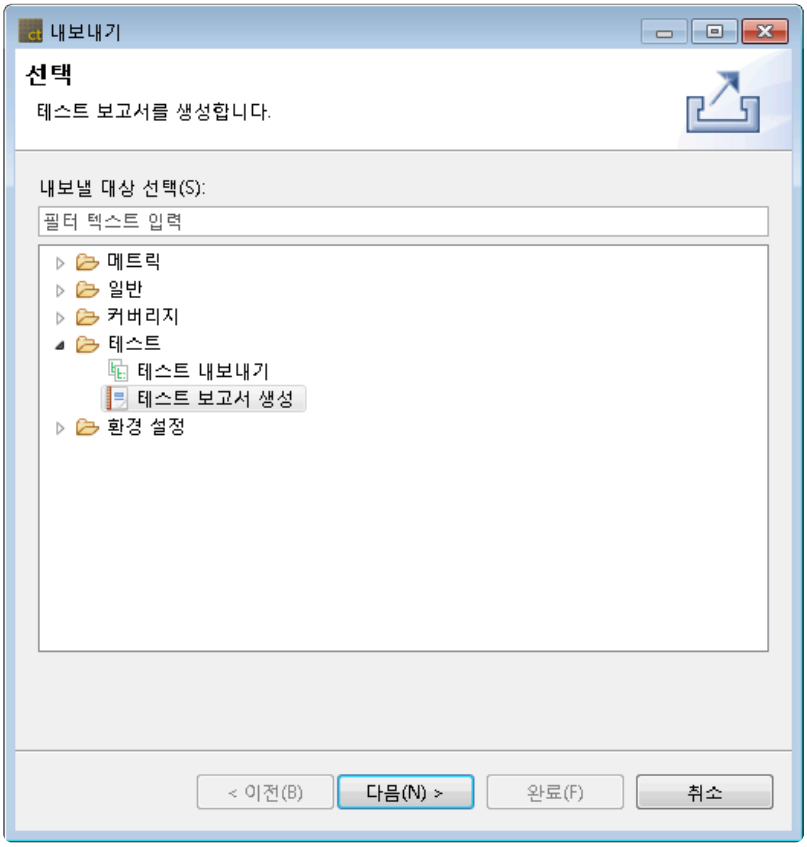
테스트 결과 보고서 생성

내보내기 기능을 통하여 테스트의 수행 결과 보고서를 다양한 형태의 문서로 생성할 수 있습니다.

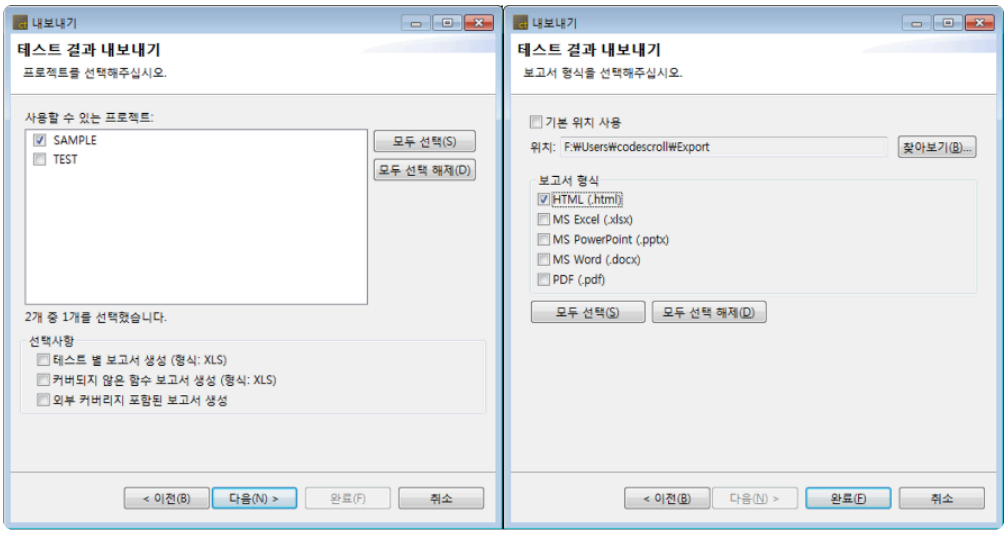
1. 메인 메뉴에서 [파일] -> [가져오기]를 클릭합니다. 내보내기 마법사가 열립니다.



2. [테스트] -> [테스트 보고서 생성]을 클릭합니다.

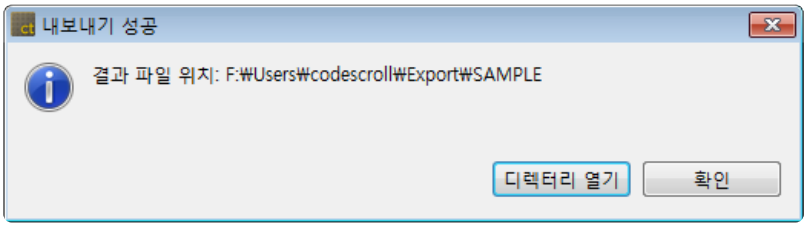


3. 보고서를 생성할 프로젝트와 옵션으로 테스트 별 보고서 생성, 커버되지 않은 함수 보고서 생성, 외부 커버리지 포함된 보고서 생성을 선택한 후 보고서 생성 경로와 보고서 확장자를 선택합니다.



a. [프로젝트]는 보고서를 생성하고자 하는 프로젝트를 선택합니다.
b. [선택사항] 그룹에서 [테스트 별 보고서 생성], [커버되지 않은 함수 보고서 생성], [외부 커버리지 포함된 보고서 생성] 내보내기 여부를 각각 체크합니다.
   - **테스트 별 보고서 생성**: 선택한 프로젝트에서 테스트를 실행한 테스트 별로 보고서를 생성합니다.
   - **커버되지 않은 함수 보고서 생성**: 선택한 프로젝트에서 커버되지 않은 함수 보고서를 생성합니다.
   - **외부 커버리지 포함된 보고서 생성**: 선택한 프로젝트에서 외부 커버리지를 포함한 보고서를 생성합니다.
c. [기본 위치 사용]은 기본값으로 “사용자계정\Export” 디렉터리를 지정합니다.
d. [보고서 형식]은 보고서의 파일 형식으로 html, pdf, docx, pptx, xlsx를 지원합니다(※ doc, ppt, xls로 보고서 생성이 필요할 시에 사업 지원팀에 문의). 보고서의 파일 확장자는 중복 선택이 가능합니다.

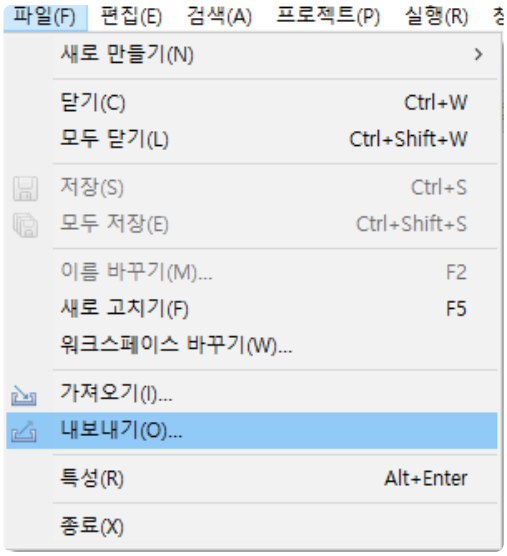
4. 보고서 생성 완료 메시지 창에서 [디렉터리 열기] 버튼을 클릭하여 보고서가 저장된 폴더에서 보고서를 확인하거나 [확인] 버튼을 클릭하여 내보내기를 종료합니다.



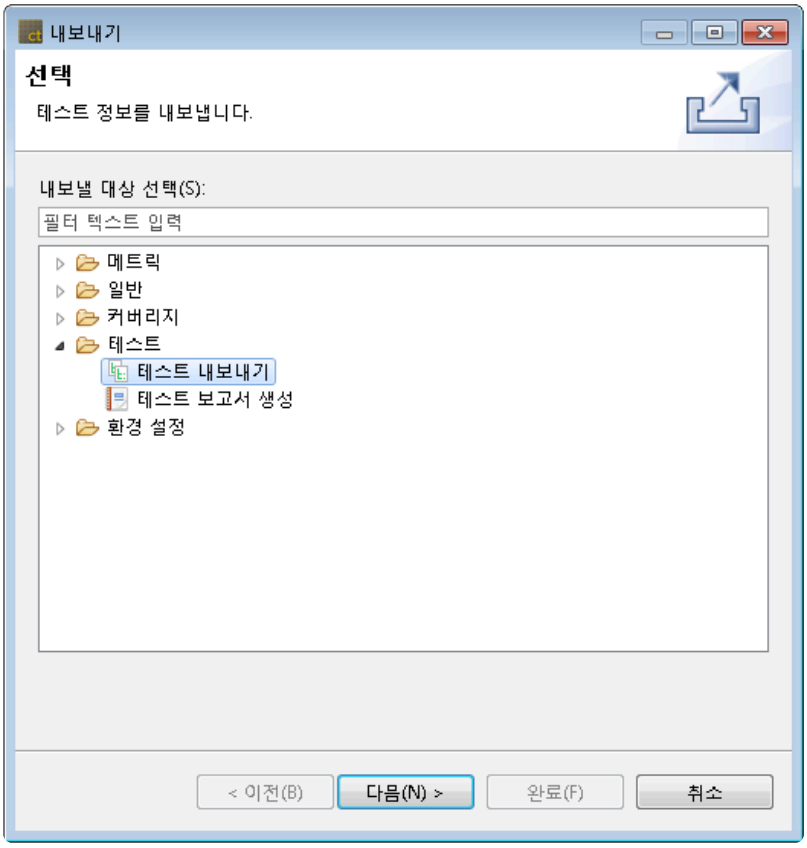
테스트 내보내기

프로젝트에서 생성한 테스트 정보를 한 번에 내보낼 수 있습니다. 프로젝트 분석 전이나 생성된 테스트 정보(유닛/통합 테스트, 스텁)가 없는 경우는 내보내기를 수행할 수 없습니다.

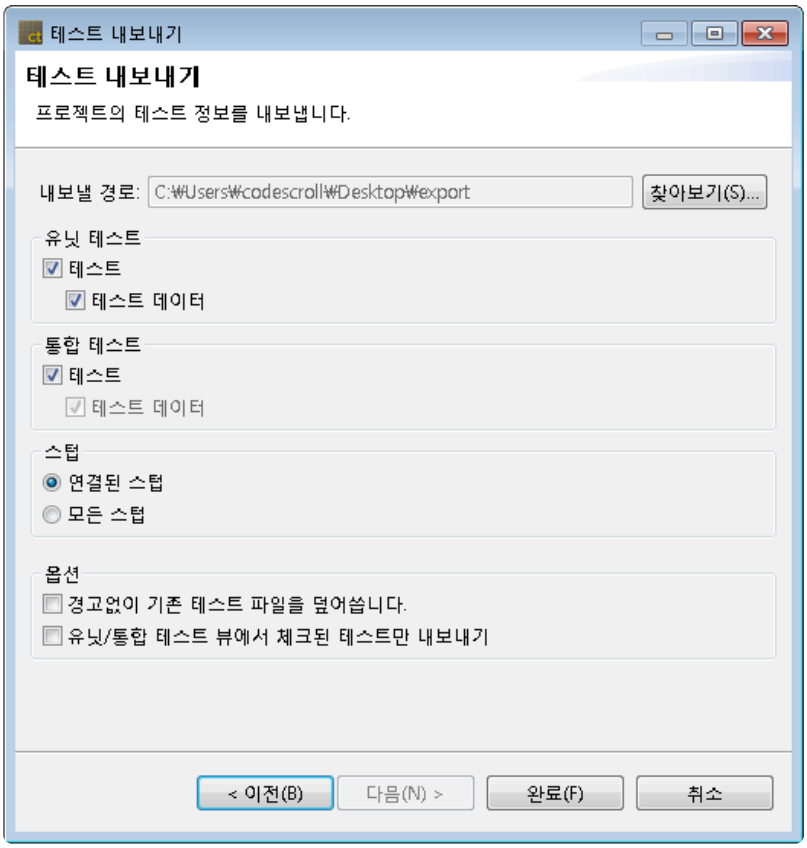
1. 메인 메뉴에서 [파일] -> [내보내기]를 클릭합니다. 내보내기 마법사가 열립니다.



2. [테스트] -> [테스트 내보내기]를 클릭합니다.

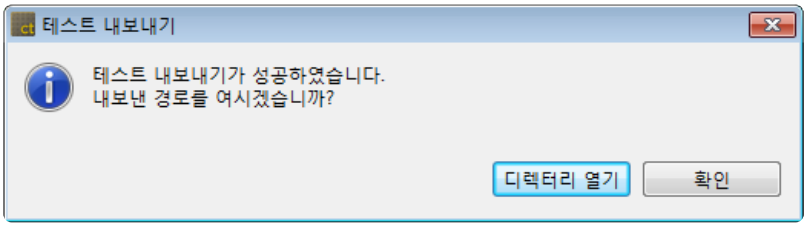


3. 내보낼 경로, 유닛 테스트 항목, 통합 테스트 항목, 스텁 항목과 옵션을 선택 후 [완료] 버튼을 선택합니다.



a. [내보낼 경로]에 테스트를 내보낼 경로를 입력합니다.
b. [유닛 테스트] 그룹에서 [테스트]와 [테스트 데이터] 내보내기 여부를 각각 체크합니다.
c. [통합 테스트] 그룹에서 [테스트] 내보내기 여부를 체크합니다.
    - 통합 테스트를 내보낼 때는 테스트 데이터와 세트로 내보내집니다.
d. [스텁] 그룹에서 [연결된 스텁]과 [모든 스텁] 중 내보낼 스텁을 선택합니다.
    - **연결된 스텁**: 내보낼 테스트와 연결되어 있는 스텁. 테스트와 연결이 없는 스텁 혹은 내보낼 테스트가 선택 되어 있지 않은 경우 스텁을 내보내지 않습니다.
    - **모든 스텁**: 프로젝트에 생성된 모든 스텁을 내보냅니다.
e. [옵션] 그룹에서 기존 테스트 파일이 존재 시 덮어쓸지 여부 또는 유닛/통합 테스트 뷰에서 체크된 테스트만 내보내기를 선택합니다.

4. 테스트 내보내기 성공 메시지 창에서 [디렉터리 열기] 버튼을 선택하여 테스트가 저장된 폴더에서 내보내진 테스트를 확인하거나 [확인] 버튼을 선택하여 내보내기를 종료합니다.

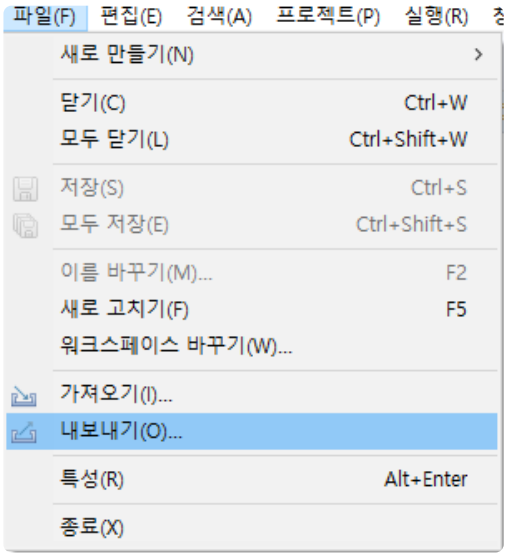


2.6.14 이후의 버전부터 테스트 내보내기 시, 내보낸 프로젝트 디렉터리에 **testinfo.export** 파일이 생성됩니다.
(testinfo.export: 내보낸 정보를 기록하고 있는 파일)

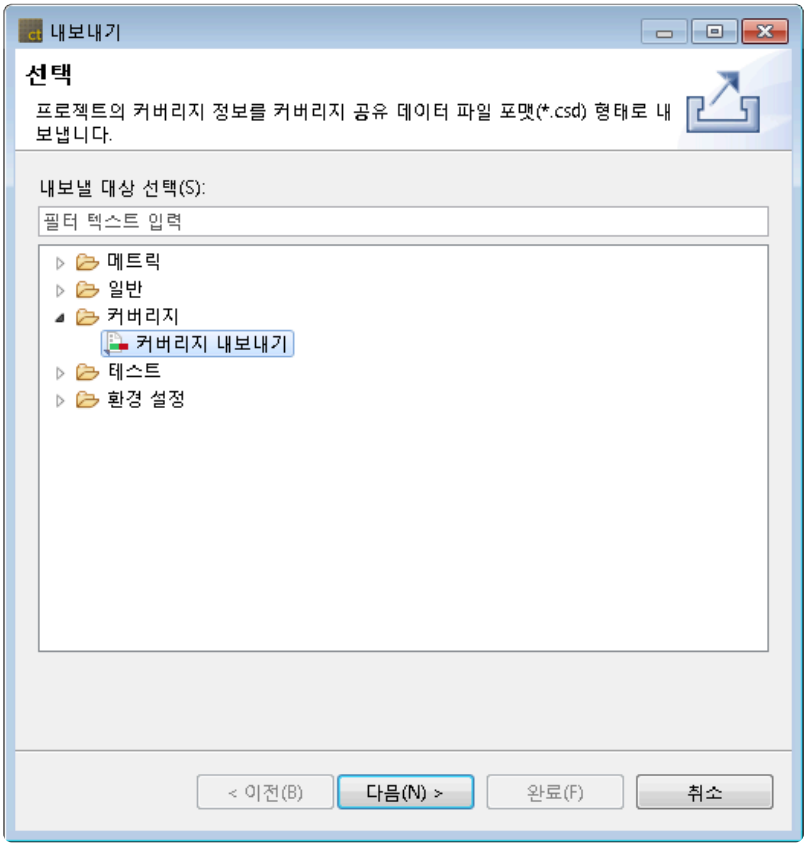
커버리지 내보내기

프로젝트에서 측정한 커버리지 정보를 커버리지 공유 데이터(*.csd) 형식의 파일로 내보낼 수 있습니다.

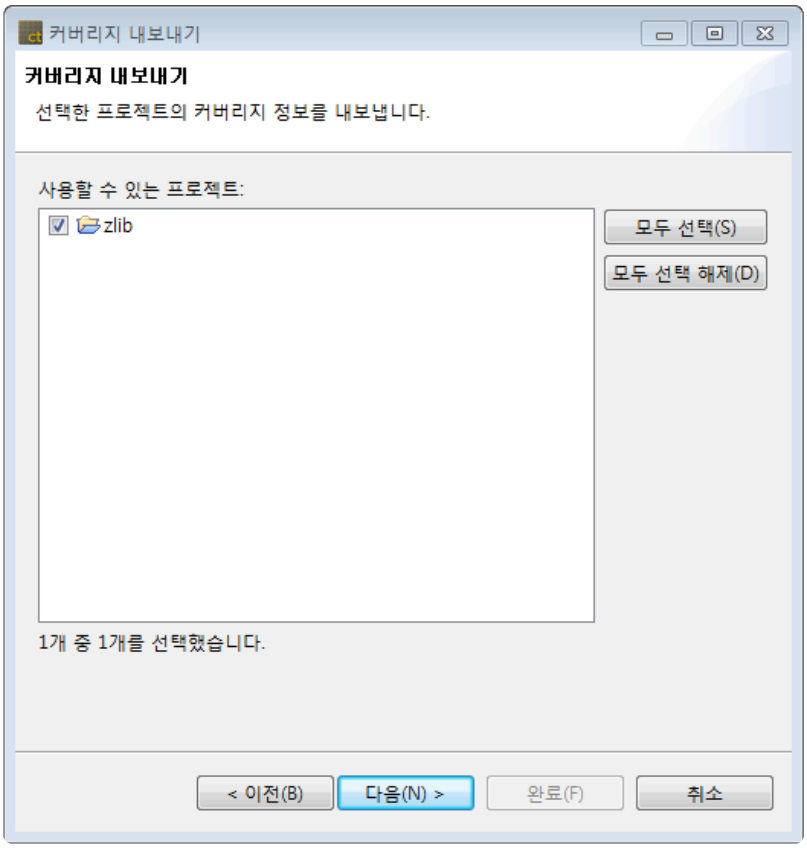
1. 메인 메뉴에서 [파일] -> [내보내기]를 클릭합니다. 내보내기 마법사가 열립니다.



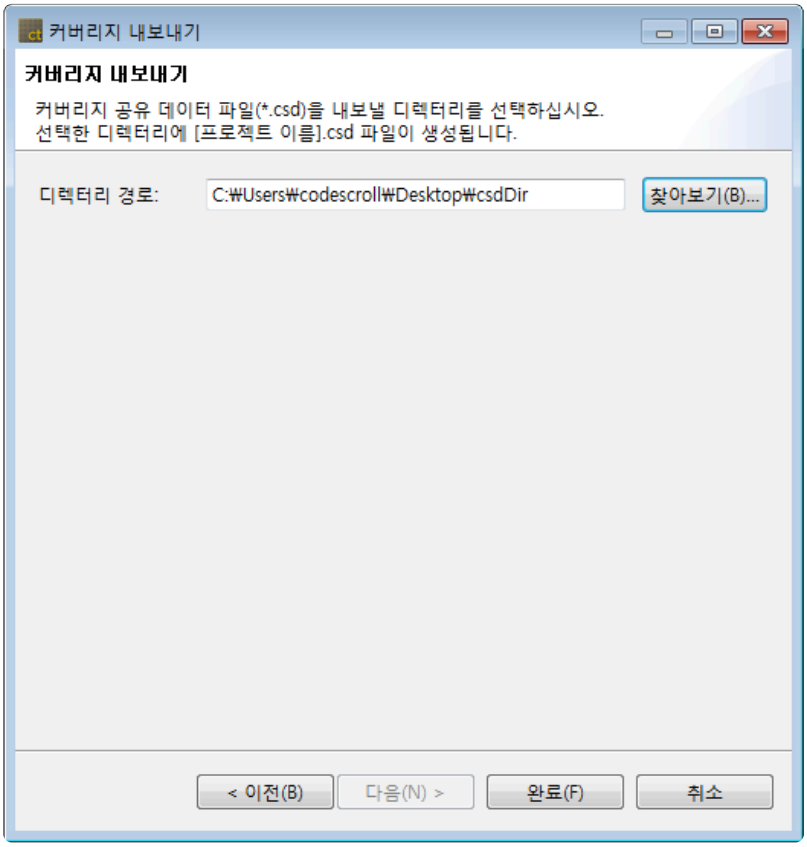
2. [커버리지] -> [커버리지 내보내기]를 선택하고 다음을 클릭합니다.



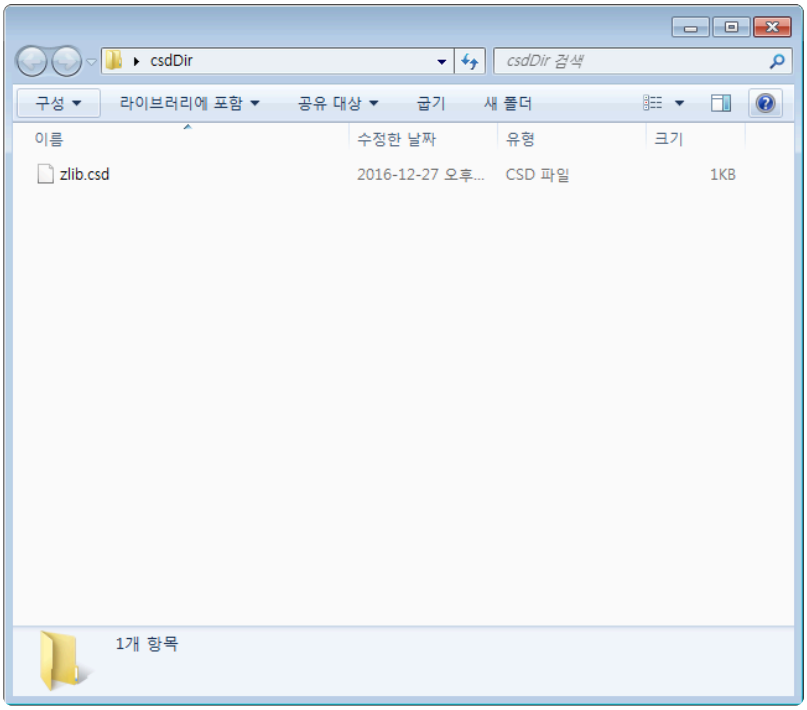
3. 커버리지 정보를 내보낼 프로젝트를 선택하고 다음을 클릭합니다.



4. 내보낼 디렉터리 경로를 입력 후 [완료] 버튼을 클릭합니다.



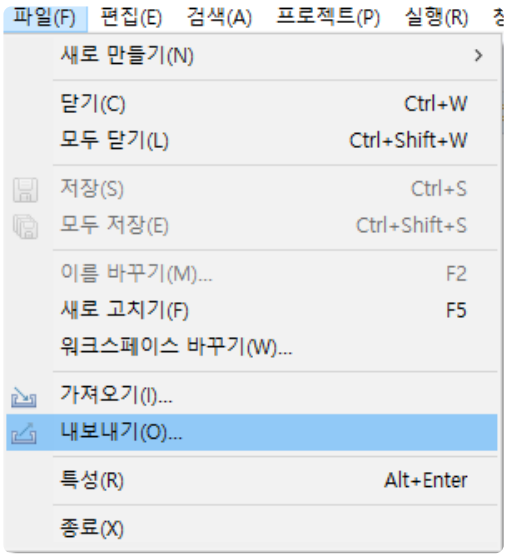
5. 프로젝트 이름과 동일한 이름의 파일이 생성됩니다.



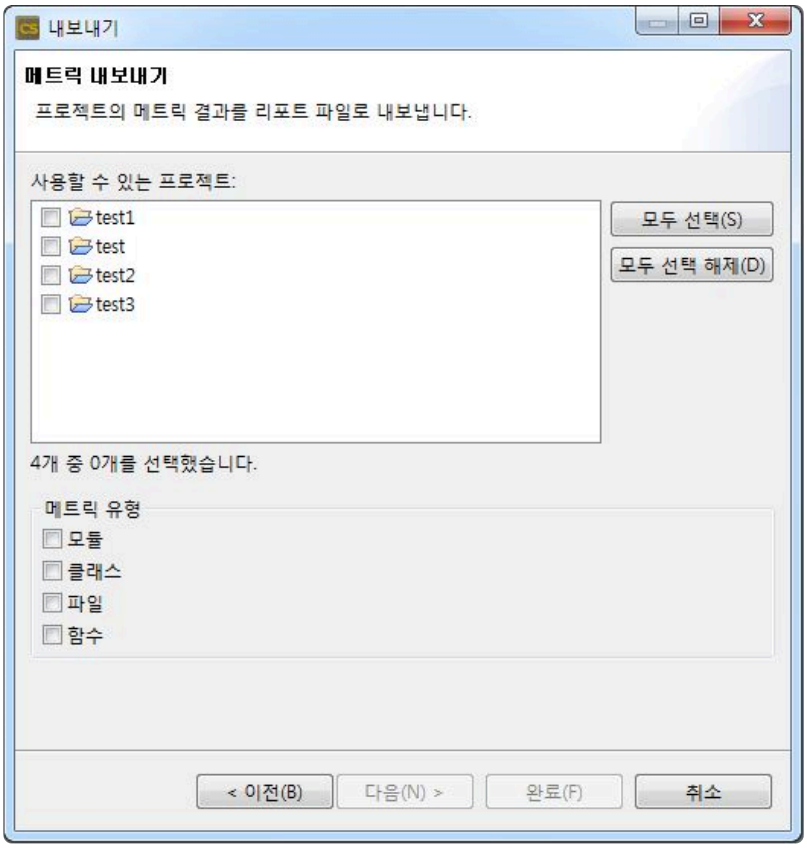
메트릭

프로젝트의 메트릭 결과를 리포트 파일로 내보냅니다.

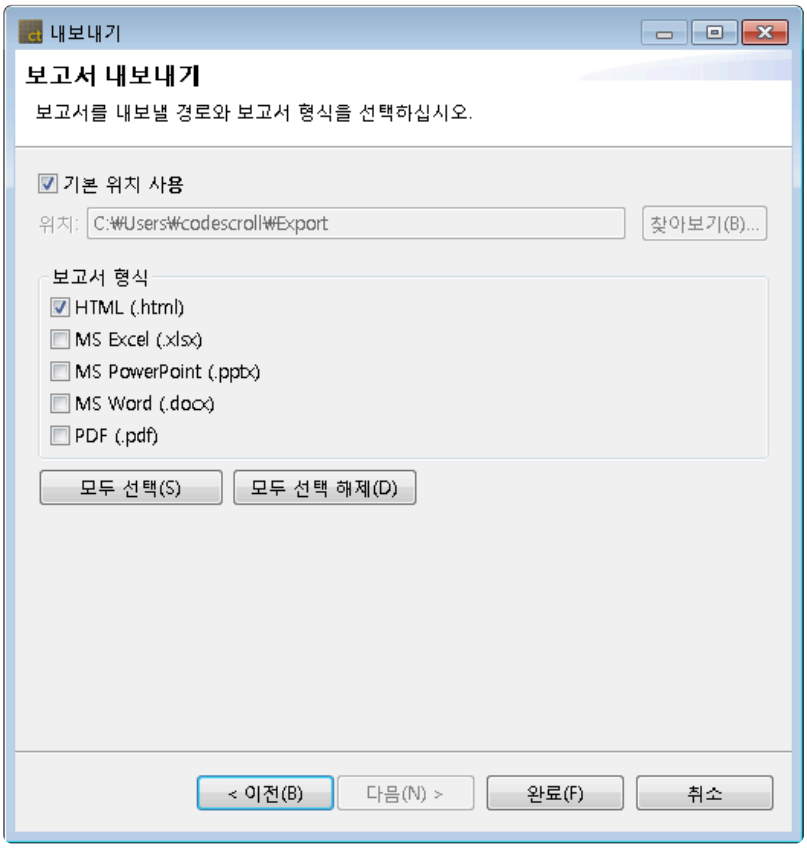
6. [메트릭] -> [메트릭]을 클릭한 후 [다음] 버튼을 클릭합니다.



7. 메트릭 정보를 내보낼 프로젝트와 메트릭 유형을 선택합니다.



1. [다음] 버튼을 클릭합니다.
2. 보고서를 내보낼 경로와 보고서 형식을 선택합니다.

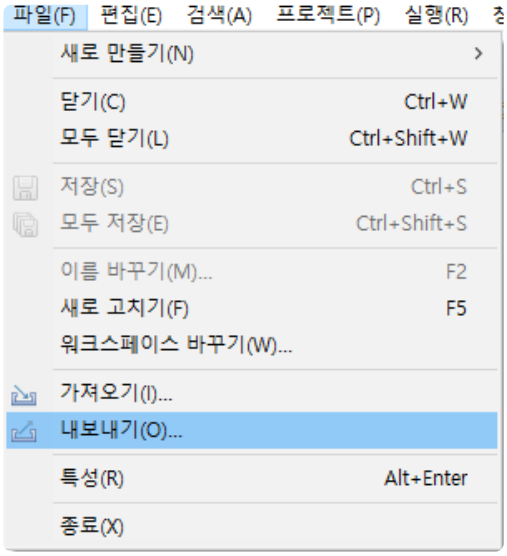


3. [완료] 버튼을 클릭합니다.

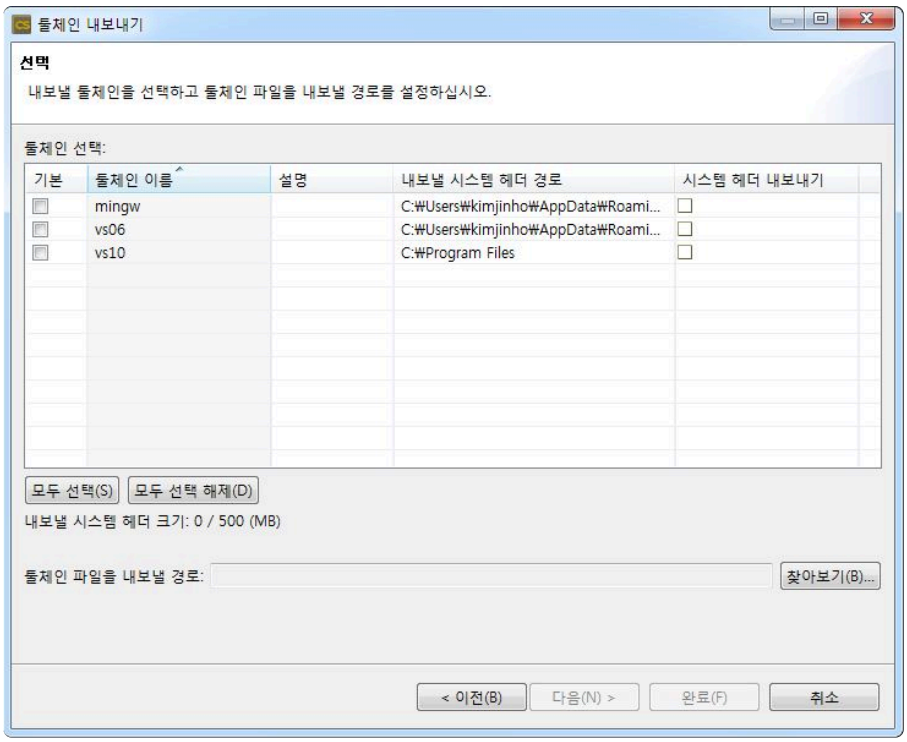
툴체인

생성된 툴체인 정보를 내보낼 수 있습니다.

1. 메인 메뉴에서 [파일] -> [내보내기]를 클릭합니다. 내보내기 마법사가 열립니다.



2. [환경 설정] -> [툴체인]을 클릭한 후 [다음] 버튼을 클릭합니다.
3. 현재 등록되어 있는 툴체인 목록을 보여줍니다.
4. 내보낼 툴체인과 시스템 헤더를 체크한 후 툴체인 정보를 내보낼 경로를 입력합니다.



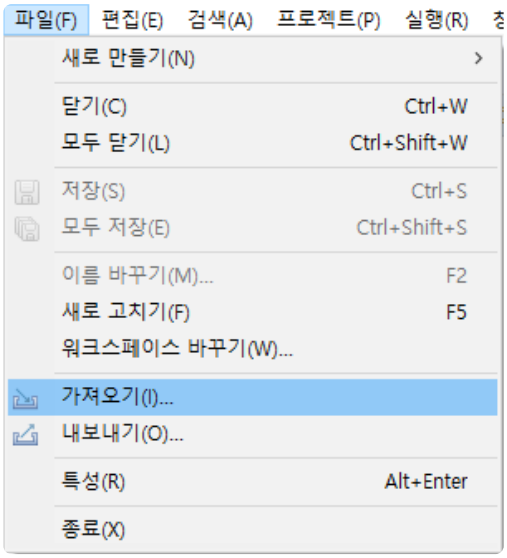
5. [완료] 버튼을 클릭합니다.

16.4. 가져오기

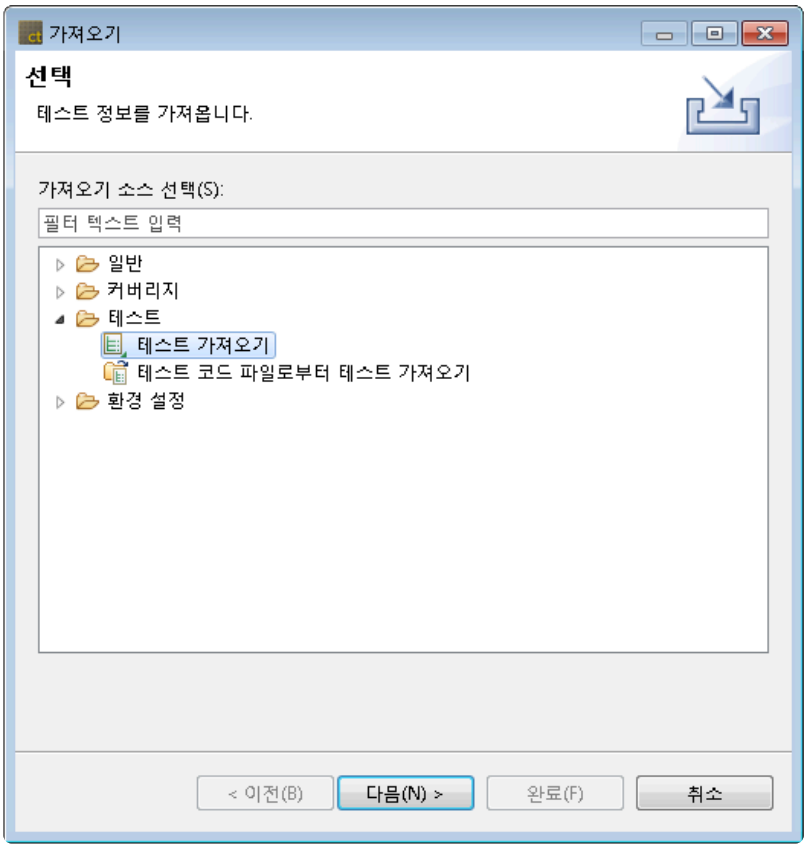
Controller Tester는 하위 버전의 Controller Tester에서 생성한 프로젝트 및 데이터를 사용할 수 있습니다.

테스트 가져오기

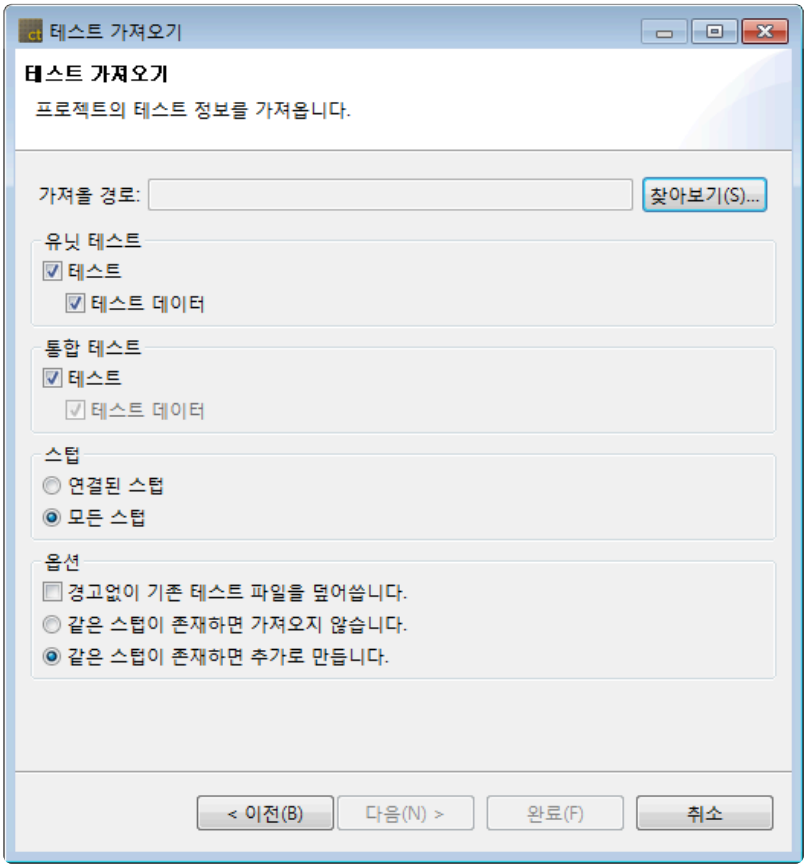
1. 메인 메뉴에서 [파일] -> [가져오기]를 클릭합니다. 가져오기 마법사가 열립니다.



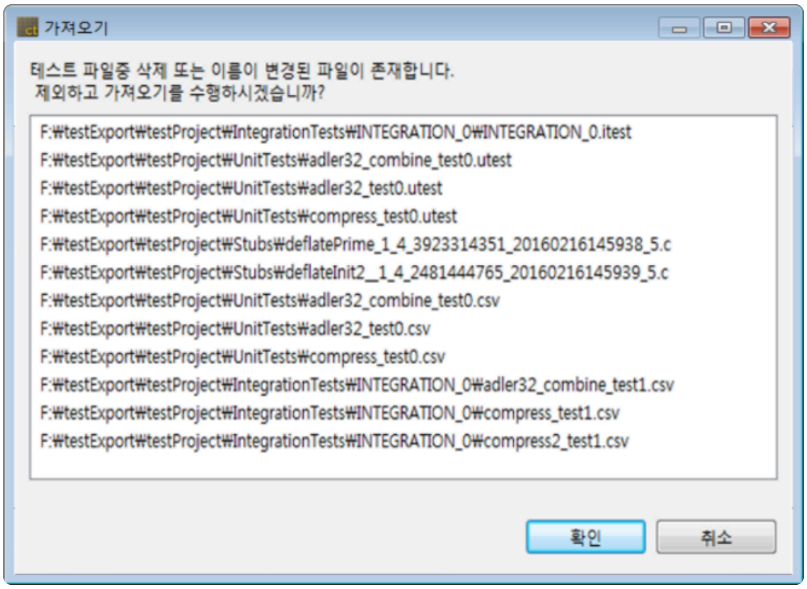
2. [테스트] -> [테스트 가져오기]를 선택하고 다음을 클릭합니다.



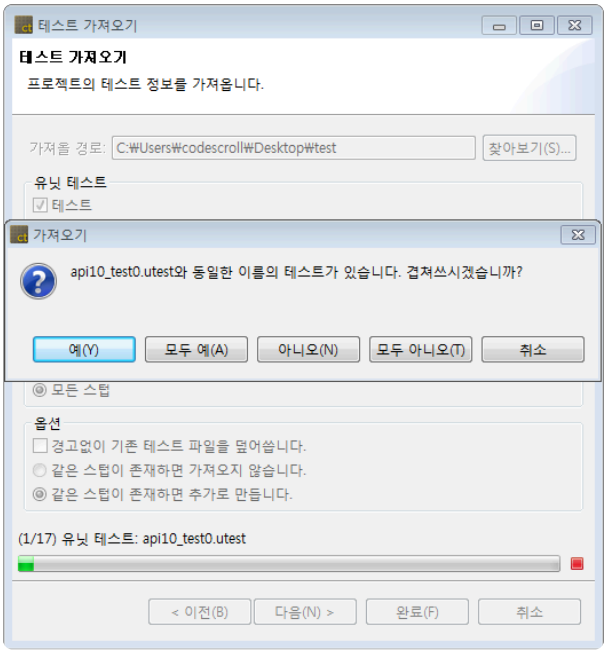
3. 가져올 테스트에 대한 정보를 입력 후 [완료] 버튼을 선택합니다.



4. 내보낸 테스트 정보 중에 가져오기를 수행할 때 이름이 변경되었거나, 파일이 제거된 항목에 대해서 제외하고 가져오기를 수행할 것인지를 선택할 수 있는 알림 창이 나타난다. 이 알림 창은 2.6.14 이후의 버전에서 한 번에 내보낸 정보일 경우에만 나타나며, 다른 메뉴 및 이전 버전에서 내보낸 정보일 경우에는 알림이 나타나지 않습니다.

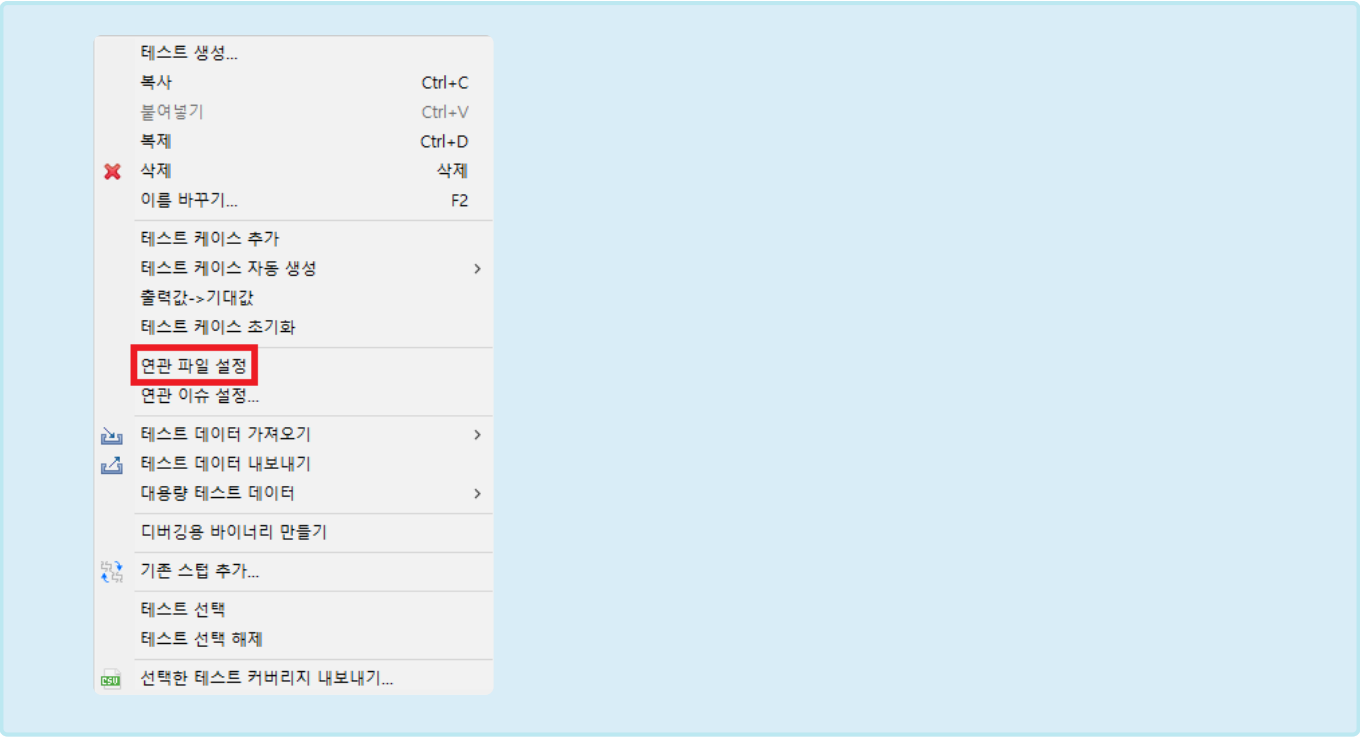


5. 프로젝트에 동일한 테스트가 존재하는 경우 테스트를 덮어 쓸지 여부를 확인하고, [예]를 선택하면 덮어쓰고 [아니오]를 선택하면 가져오기를 수행하지 않습니다.



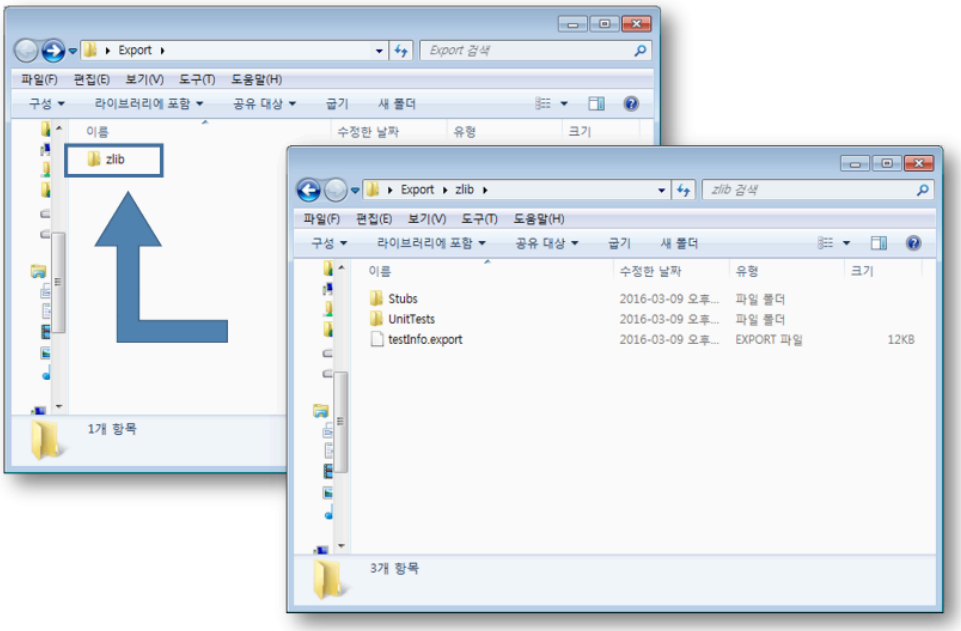
6. 가져오기 완료 후 선택한 프로젝트에 적용됩니다.

> ✻ '테스트 대상 함수를 테스트 대상 소스 파일에서 찾을 수 없는 테스트' 상태 아이콘( )이 나타나면 [연관 파일 설정] 컨텍스트 메뉴를 이용하여 연관 파일을 설정할 수 있습니다.



2.6.14 이하 버전에서 내보낸 정보 가져오기

1. [가져올 경로]에 테스트를 가져올 경로를 입력합니다.



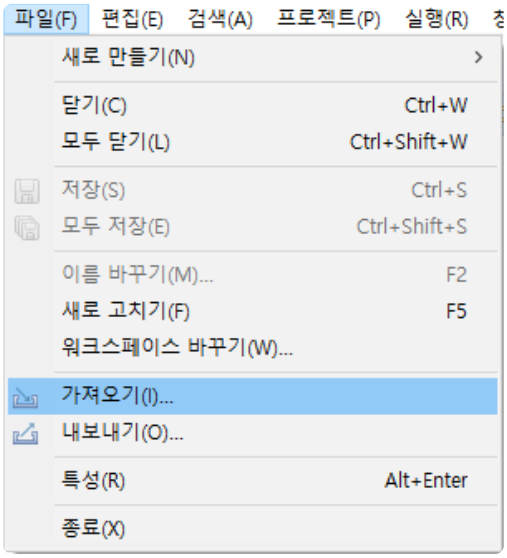
   - 가져올 테스트 경로에 [testinfo.export]파일이 존재한다면 파일이 포함된 디렉터리를 선택합니다. 2.6.14 이하의 버전에서 각각 내보낸 정보를 가져오기 할 경우에는 내보낸 정보가 있는 상위의 디렉터리를 선택합니다.
2. [유닛 테스트] 그룹에서 [테스트]와 [테스트 데이터] 가져오기 여부를 각각 체크합니다.
3. [통합 테스트] 그룹에서 [테스트] 가져오기 여부를 체크합니다.
   - 테스트와 데이터는 항상 같은 디렉터리에 있어야 하며, 같은 디렉터리에 없으면 데이터를 가져오지

않습니다.

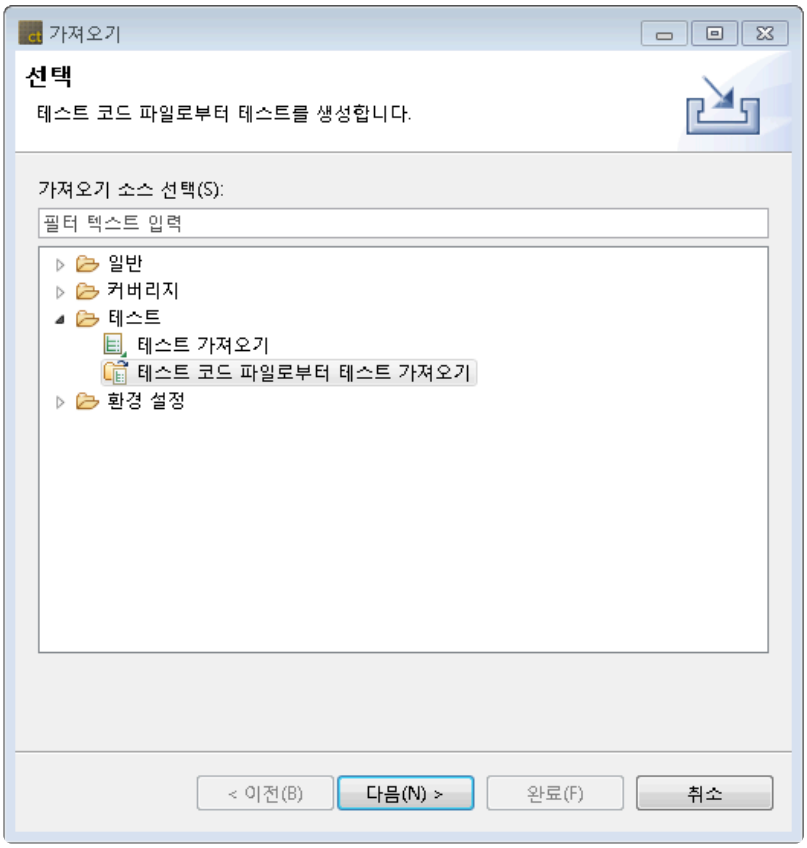
4. [스텁] 그룹에서 [연결된 스텁]과 [모든 스텁] 중 내보낼 스텁을 선택합니다.
    - 2.3버전에서 생성한 스텁 또는 2.6으로 마이그레이션된 스텁의 경우 프로젝트에 동일한 스텁이 존재하는 경우에 가져오기를 수행할 수 없습니다.
    - 2.6에서 생성한 스텁의 경우 동일한 스텁이 존재하는 경우 선택적으로 가져오기를 수행하지 않을 수도 있고, 새로운 스텁으로 추가 생성 할 수 있습니다.
        - **스텁과 연결된 테스트를 가져오는 경우, 예제가 수록된 부록 A를 참고하십시오.**
    - **2.3버전에는 테스트에 스텁을 연결하는 기능이 존재하지 않아, 2.6 이후의 버전으로 마이그레이션 수행 시 테스트에 직접 연결해야 합니다.**
5. [옵션] 그룹에서 기존 테스트 파일이 존재 시 덮어쓸지 여부, 같은 스텁 존재 시 처리 여부를 선택합니다.

테스트 코드 파일로부터 테스트 가져오기

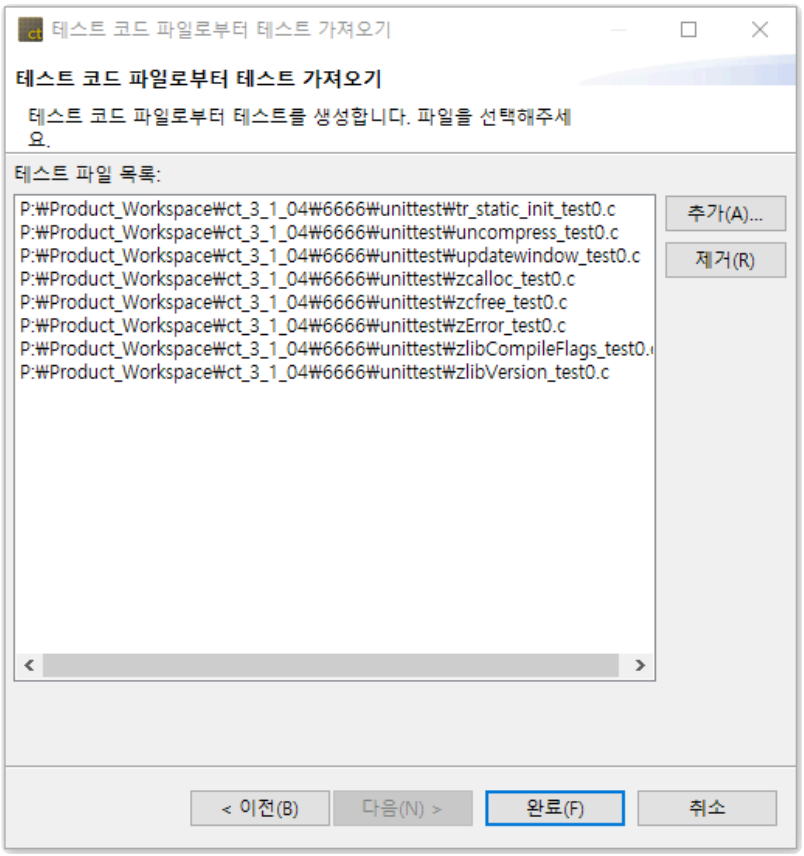
1. 메인 메뉴에서 [파일] -> [가져오기]를 클릭합니다. 가져오기 마법사가 열립니다.



2. [테스트] -> [테스트 코드 파일로부터 테스트 가져오기]를 선택하고 다음을 클릭합니다.

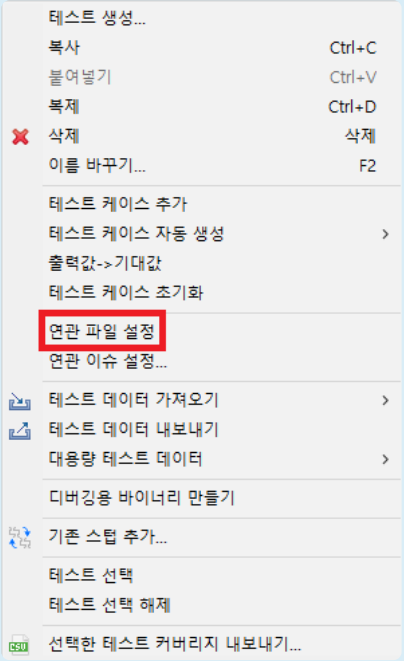


3. [추가]를 클릭한 후 가져올 테스트를 선택합니다.



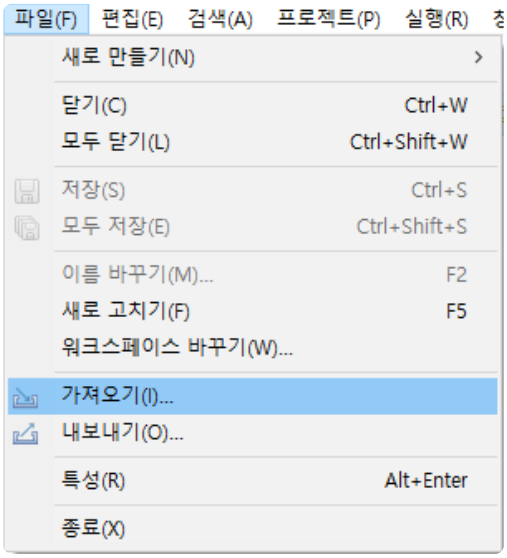
4. [완료]를 클릭하여 선택한 테스트를 선택한 프로젝트에 적용합니다.

'테스트 대상 함수를 테스트 대상 소스 파일에서 찾을 수 없는 테스트' 상태 아이콘( )이 나타나면 [연관 파일 설정] 컨텍스트 메뉴를 이용하여 연관 파일을 설정할 수 있습니다.

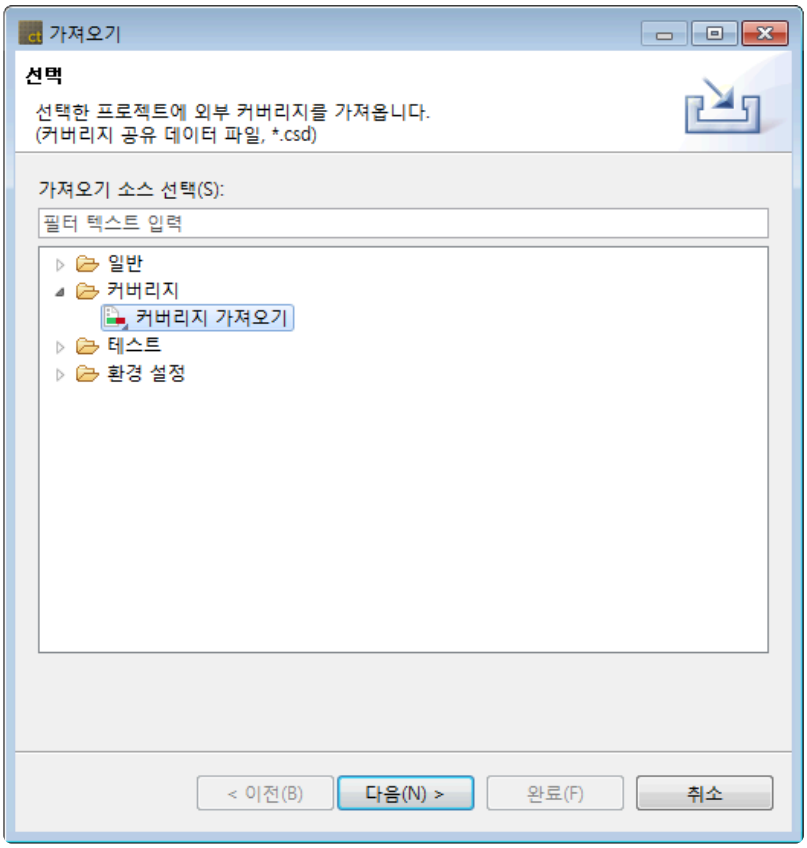


커버리지 가져오기

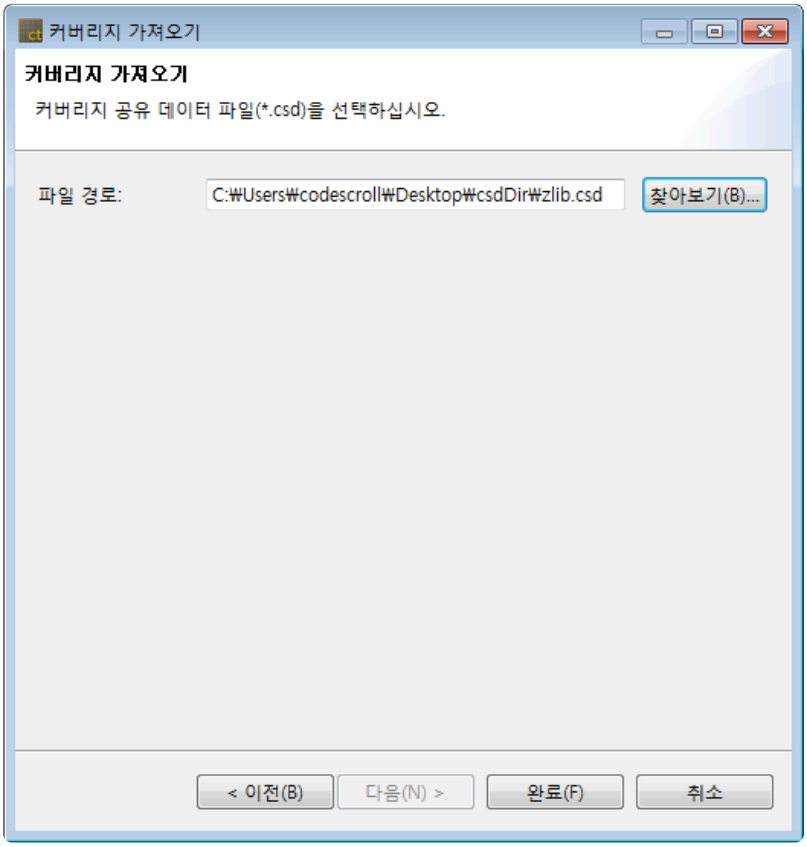
1. 메인 메뉴에서 [파일] -> [가져오기]를 클릭합니다. 가져오기 마법사가 열립니다.



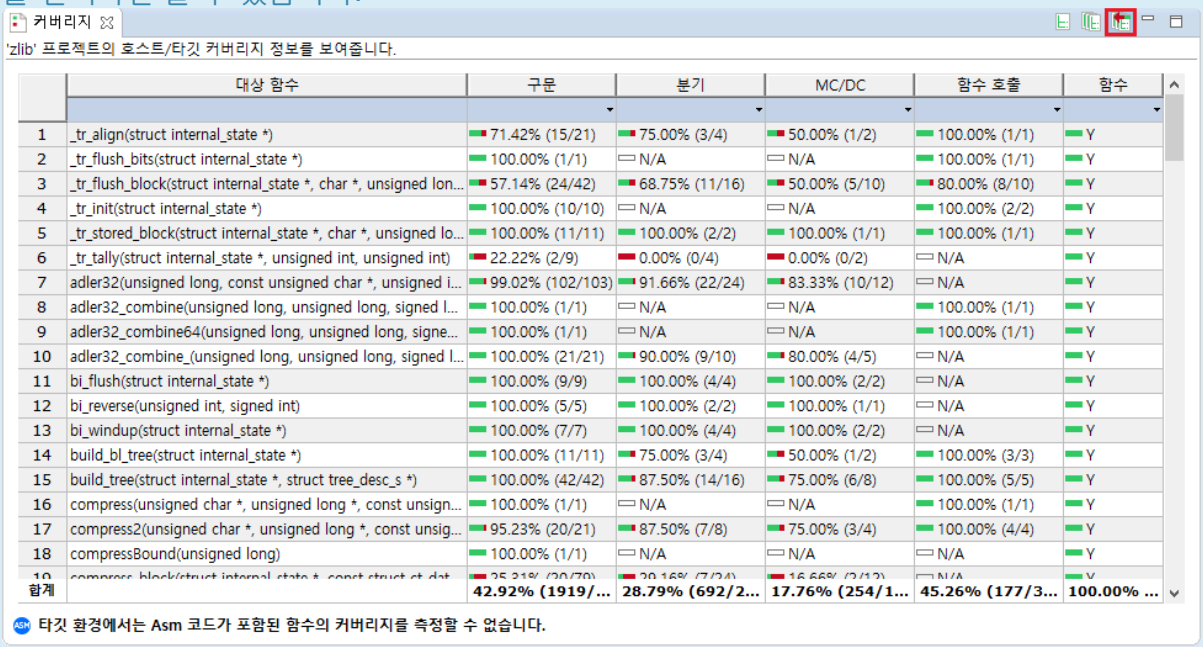
2. [커버리지] -> [커버리지 가져오기]를 선택하고 다음을 클릭합니다.



3. 가져올 커버리지 파일 경로를 입력 후 [완료] 버튼을 클릭합니다.



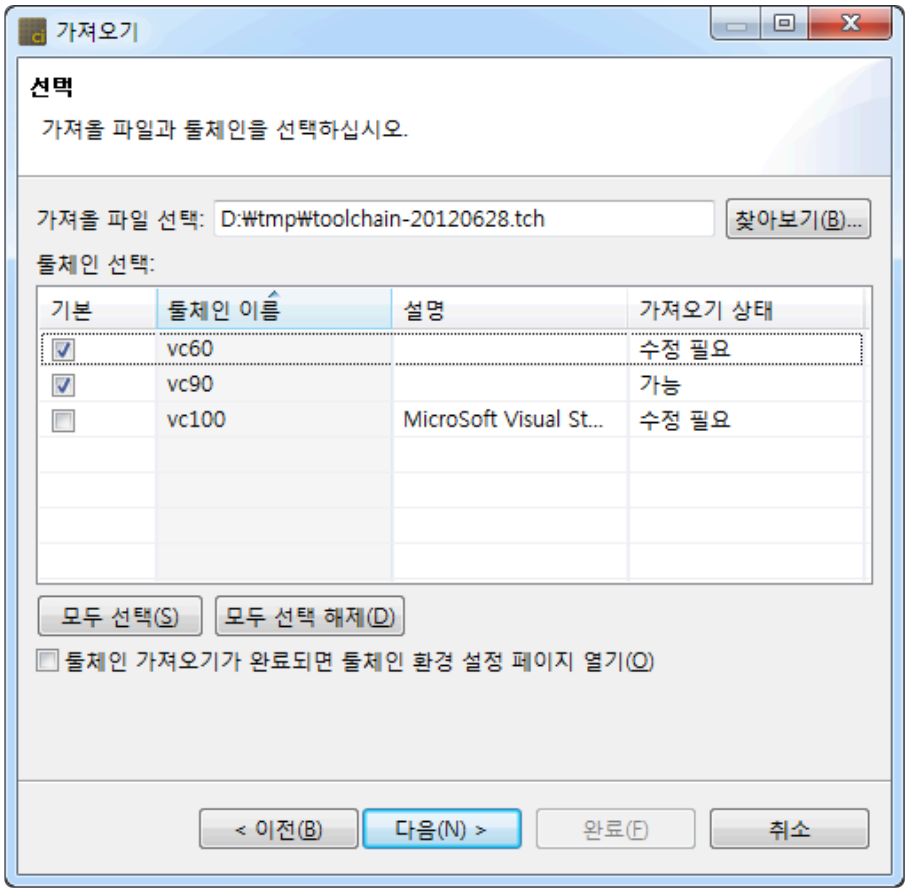
가져온 커버리지 정보는 커버리지 뷰의 툴바 메뉴에서 [전체 커버리지 보기(외부 커버리지 포함)]를 선택하면 볼 수 있습니다.



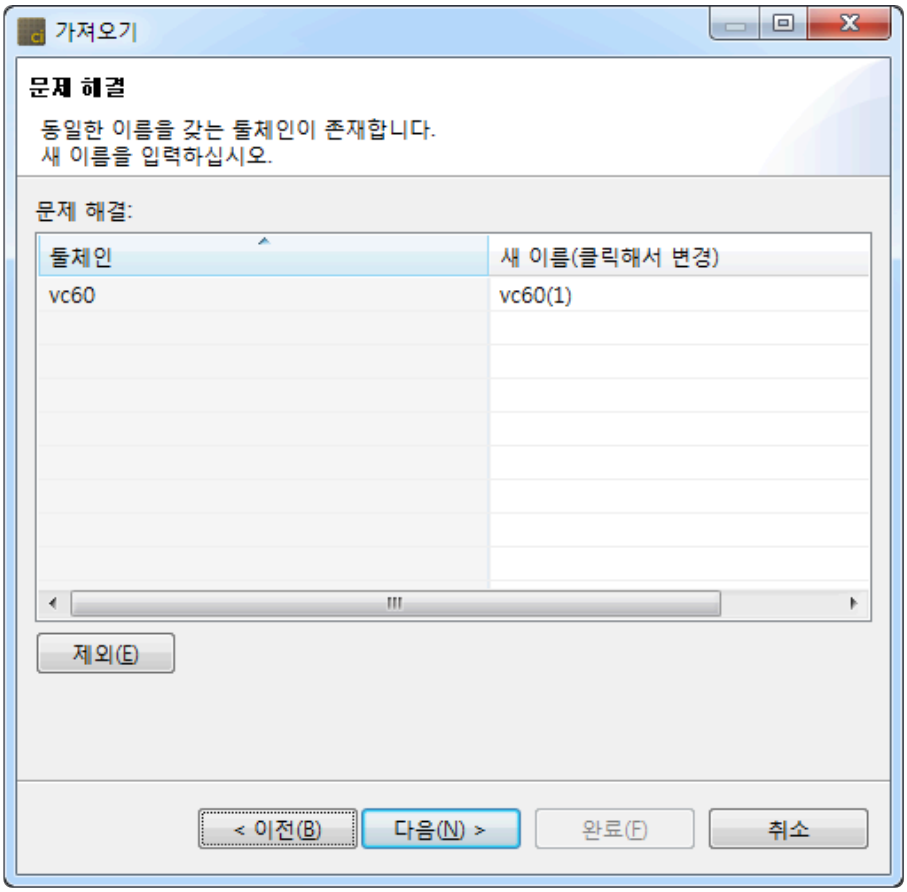
	대상 함수	구문	분기	MC/DC	함수 호출	함수
1	_tr_align(struct internal_state *)	71.42% (15/21)	75.00% (3/4)	50.00% (1/2)	100.00% (1/1)	Y
2	_tr_flush_bits(struct internal_state *)	100.00% (1/1)	N/A	N/A	100.00% (1/1)	Y
3	_tr_flush_block(struct internal_state *, char *, unsigned lon...	57.14% (24/42)	68.75% (11/16)	50.00% (5/10)	80.00% (8/10)	Y
4	_tr_init(struct internal_state *)	100.00% (10/10)	N/A	N/A	100.00% (2/2)	Y
5	_tr_stored_block(struct internal_state *, char *, unsigned lo...	100.00% (11/11)	100.00% (2/2)	100.00% (1/1)	100.00% (1/1)	Y
6	_tr_tally(struct internal_state *, unsigned int, unsigned int)	22.22% (2/9)	0.00% (0/4)	0.00% (0/2)	N/A	Y
7	adler32(unsigned long, const unsigned char *, unsigned i...	99.02% (102/103)	91.66% (22/24)	83.33% (10/12)	N/A	Y
8	adler32_combine(unsigned long, unsigned long, signed l...	100.00% (1/1)	N/A	N/A	100.00% (1/1)	Y
9	adler32_combine64(unsigned long, unsigned long, signe...	100.00% (1/1)	N/A	N/A	100.00% (1/1)	Y
10	adler32_combine_(unsigned long, unsigned long, signed l...	100.00% (21/21)	90.00% (9/10)	80.00% (4/5)	N/A	Y
11	bi_flush(struct internal_state *)	100.00% (9/9)	100.00% (4/4)	100.00% (2/2)	N/A	Y
12	bi_reverse(unsigned int, signed int)	100.00% (5/5)	100.00% (2/2)	100.00% (1/1)	N/A	Y
13	bi_windup(struct internal_state *)	100.00% (7/7)	100.00% (4/4)	100.00% (2/2)	N/A	Y
14	build_bl_tree(struct internal_state *)	100.00% (11/11)	75.00% (3/4)	50.00% (1/2)	100.00% (3/3)	Y
15	build_tree(struct internal_state *, struct tree_desc_s *)	100.00% (42/42)	87.50% (14/16)	75.00% (6/8)	100.00% (5/5)	Y
16	compress(unsigned char *, unsigned long *, const unsign...	100.00% (1/1)	N/A	N/A	100.00% (1/1)	Y
17	compress2(unsigned char *, unsigned long *, const unsig...	95.23% (20/21)	87.50% (7/8)	75.00% (3/4)	100.00% (4/4)	Y
18	compressBound(unsigned long)	100.00% (1/1)	N/A	N/A	N/A	Y
19	compress_block(struct internal_state *, const struct ct_dat...	25.31% (20/79)	29.16% (7/24)	16.66% (2/12)	N/A	Y
합계		42.92% (1919/...	28.79% (692/2...	17.76% (254/1...	45.26% (177/3...	100.00% ...

툴체인

가져오기 기능을 이용하여 내보낸 툴체인 정보를 가져올 수 있습니다.



1. 메인 메뉴에서 [파일] -> [가져오기]를 클릭합니다. 가져오기 마법사가 열립니다.
2. [환경 설정] -> [툴체인]을 클릭한 후 [다음] 버튼을 클릭합니다.
3. 가져올 파일을(*.tch) 선택하면, 선택한 파일에 포함된 툴체인 목록이 나타납니다.
4. 툴체인 목록에서 가져올 툴체인을 선택합니다.
5. 가져올 툴체인이 기존 툴체인과 동일한 컴파일러를 사용하거나 툴체인 이름이 기존 툴체인 이름과 중복될 경우, 가져오기 상태가 “수정 필요”가 됩니다. [다음>] 버튼을 클릭하여 나오는 [문제해결] 창에서 가져올 툴체인의 이름을 변경할 수 있습니다.

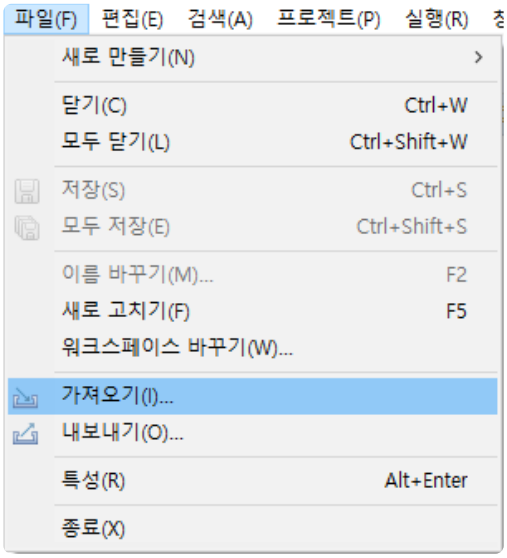


6. [완료] 버튼을 클릭합니다.

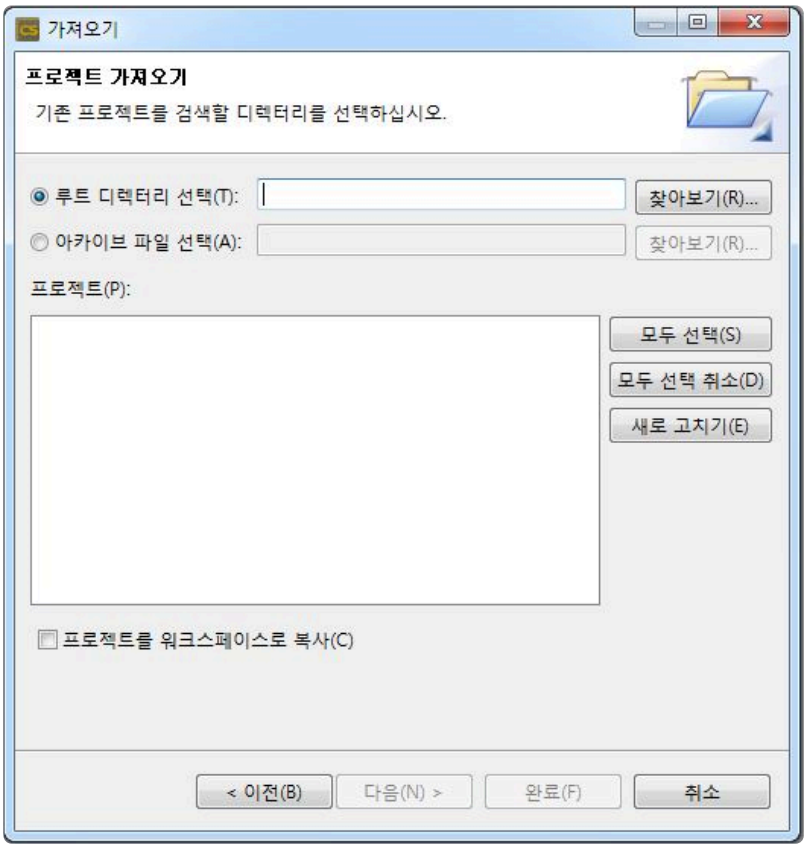
기존 프로젝트를 워크스페이스로

기존 프로젝트를 워크스페이스로 가져오기 기능을 이용하여, 워크스페이스에 존재하지 않는 프로젝트를 워크스페이스로 가져올 수 있습니다.

1. 메인 메뉴에서 [파일] -> [가져오기]를 클릭합니다. 가져오기 마법사가 열립니다.



2. [일반] -> [기존 프로젝트를 워크스페이스로]를 클릭한 후 [다음] 버튼을 클릭합니다.
3. 루트 디렉터리 또는 아카이브 파일을 선택하고 [찾아보기] 버튼을 클릭하여 프로젝트를 포함하는 디렉터리 또는 파일을 찾습니다.
4. 선택한 루트 디렉터리 또는 아카이브 파일에 포함된 프로젝트 목록에서 가져오려는 프로젝트를 선택합니다.



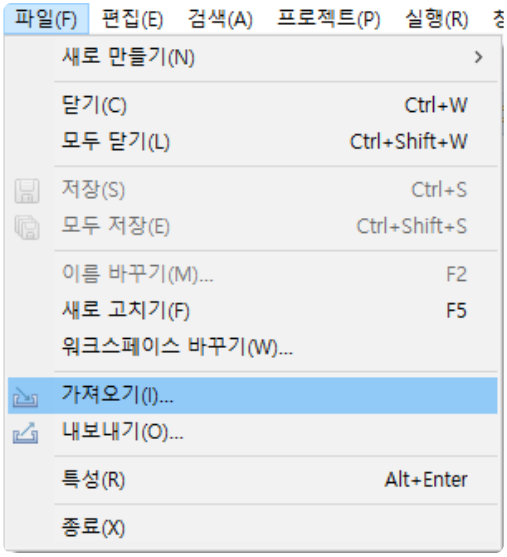
5. [완료]버튼을 클릭합니다.

> [프로젝트를 워크스페이스로 복사]를 선택하면 가져오려는 프로젝트를 워크스페이스 디렉터리로 복사합니다. 선택하지 않으면 가져오려는 프로젝트를 워크스페이스에 연결만 합니다.

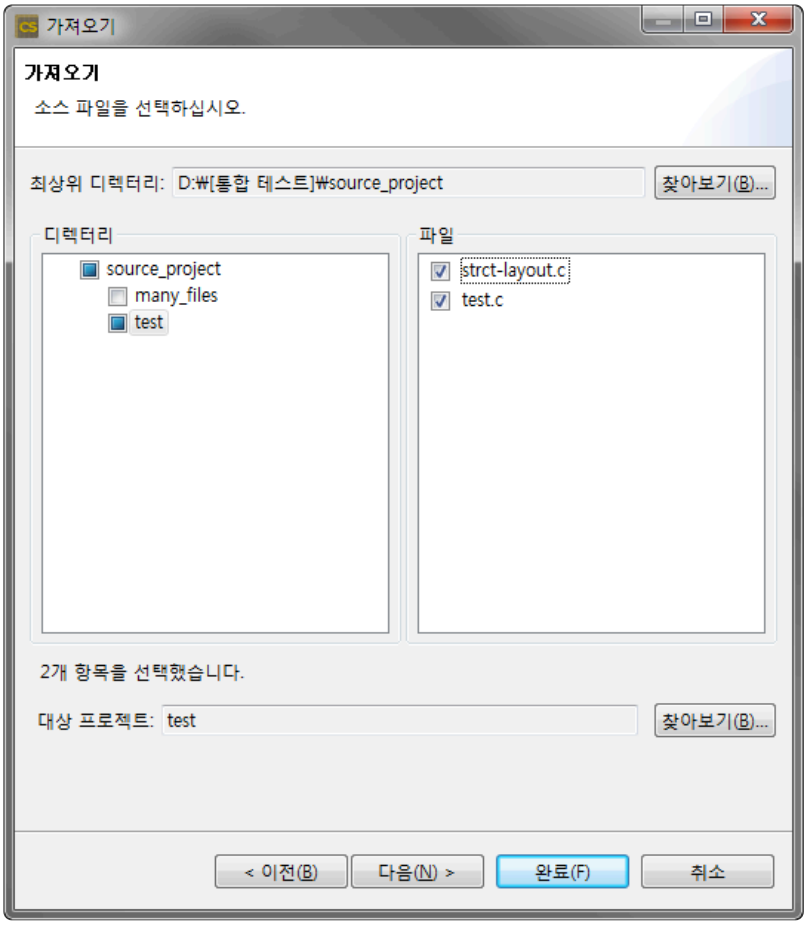
파일 시스템

파일 시스템 가져오기 기능을 이용하여, 프로젝트에 소스 파일을 추가할 수 있습니다.

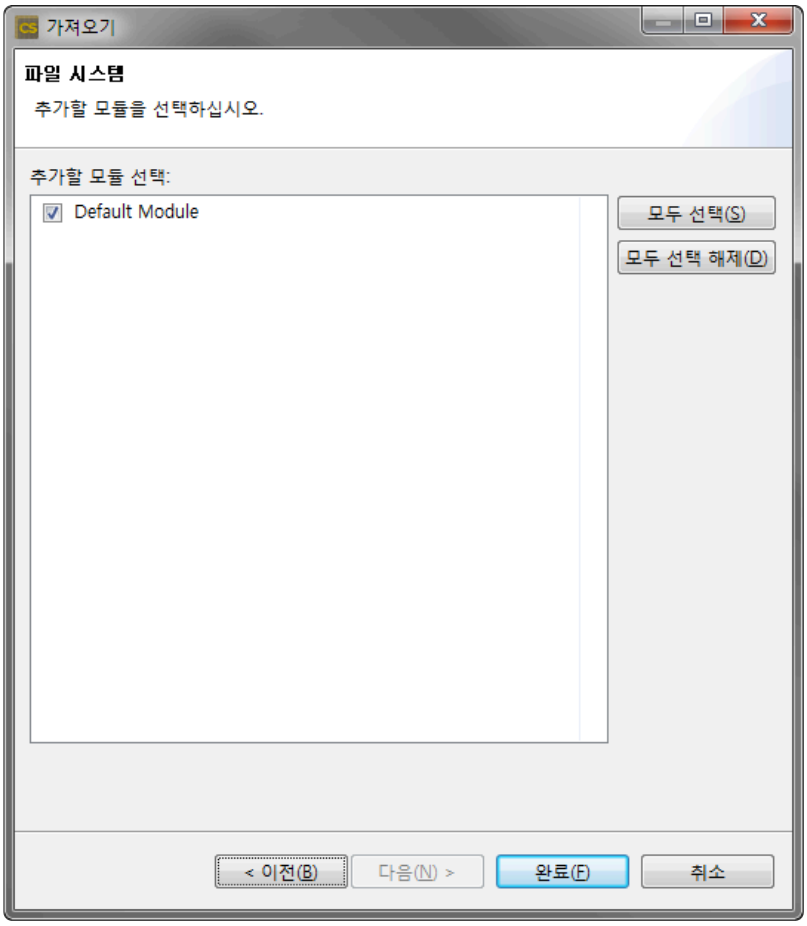
1. 메인 메뉴에서 [파일] -> [가져오기]를 클릭합니다. 가져오기 마법사가 열립니다.



2. [일반] -> [파일 시스템]을 클릭한 후 [다음] 버튼을 클릭합니다.
3. 추가할 소스 파일을 포함하는 최상위 디렉터리 경로를 [찾아보기] 버튼을 클릭하여 선택합니다.
4. 추가할 소스 파일을 선택합니다.
5. 선택한 소스 파일이 추가될 대상 프로젝트를 [찾아보기] 버튼을 클릭하여 선택합니다.



6. [다음] 버튼을 클릭합니다.
7. 프로젝트에 포함된 모듈 목록에서 소스 파일을 추가할 모듈을 선택합니다.



8. [완료] 버튼을 클릭합니다.

> Windows 탐색기에서 소스 파일을 끌어서 테스트 네비게이터 뷰의 대상 프로젝트 또는 모듈에 놓는 방법으로도 소스 파일을 추가할 수 있습니다.

17. 편집 메뉴

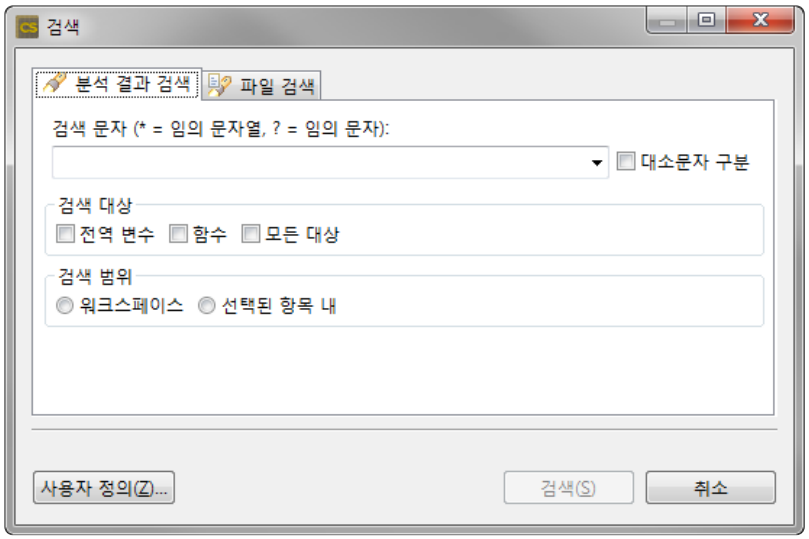
편집 메뉴에서는 뷰 또는 편집기에서 선택된 항목에 대한 잘라내기, 복사, 붙여넣기, 삭제, 찾기/바꾸기 기능을 수행하거나, 마지막으로 수행된 액션을 실행 취소하거나 다시 실행할 수 있습니다.

선택된 뷰 또는 편집기에 따라서 메뉴의 구성이 달라질 수 있습니다.

18. 검색 메뉴

분석 결과 검색

분석 후 생성된 결과를 검색할 수 있습니다.



검색 문자

검색할 문자를 입력합니다.
사용 가능한 와일드 카드는 검색 대화 상자에 표시됩니다.

- “*” 임의의 문자열: 빈 문자열을 포함한 문자의 집합과 일치
- “?” 임의의 문자: 모든 문자에 대해 일치

검색 대상

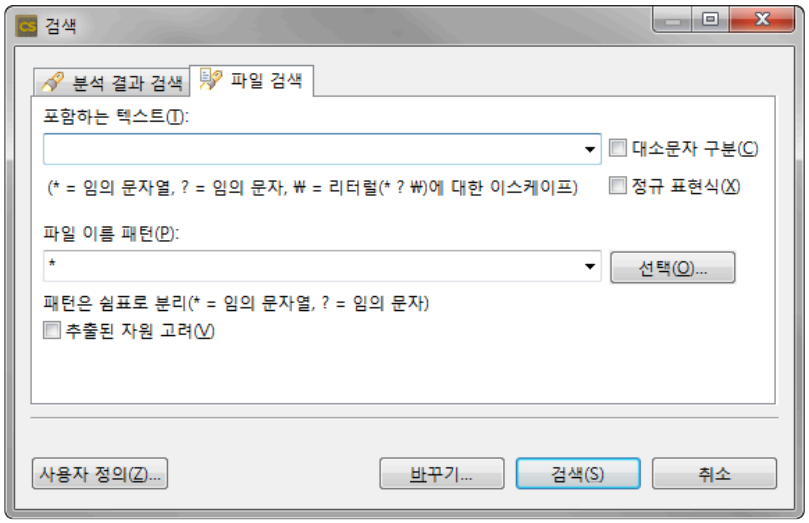
검색 대상을 선택합니다. 전역변수, 함수 또는 모든 대상을 검색할 수 있습니다.

검색 범위

검색의 범위를 선택합니다. 워크스페이스 또는 선택된 항목 범위에서 검색할 수 있습니다.

파일 검색

파일을 검색할 수 있습니다.



포함하는 텍스트

검색할 문자를 입력합니다. 파일을 검색하려면 필드를 비워둬야 합니다.
[▼]를 클릭하면 최근에 검색한 문자를 선택할 수 있습니다.
사용 가능한 와일드 카드는 검색 대화 상자에 표시됩니다.

- “*” 임의의 문자열: 빈 문자열을 포함한 문자의 집합과 일치
- “?” 임의의 문자: 모든 문자에 대해 일치
- “*”, “?”, 또는 “\“문자를 검색하려면 “\ *” 와일드 카드로 이러한 문자를 사용하지 않는 것을 나타내기 위해, 문자 앞에 백슬래시를 입력합니다. (예, “\?“또는 “\ \”)

파일 이름 패턴

특정 표현을 통해 검색할 파일에 대한 모든 파일 이름 패턴을 입력합니다.
파일 이름 패턴을 사용할 수 있는 와일드 카드는 검색 대화 상자에 표시됩니다.

- “*” 임의의 문자열: 빈 문자열을 포함한 문자의 집합과 일치
- “?” 임의의 문자: 모든 문자에 대해 일치

19. 프로젝트 메뉴

프로젝트 메뉴는 프로젝트에 대한 작업(열기, 닫기, 초기화)을 수행할 수 있습니다.

프로젝트 열기

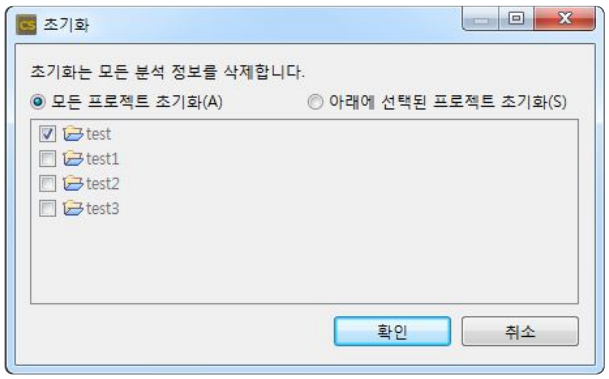
테스트 네비게이터 뷰에서 선택된 닫혀있는 프로젝트를 열 수 있습니다.

프로젝트 닫기

테스트 네비게이터 뷰에서 선택된 열려있는 프로젝트를 닫을 수 있습니다.

프로젝트 초기화

워크스페이스에 열려있는 모든 프로젝트 또는 아래의 프로젝트 목록 중에서 선택한 프로젝트를 초기화합니다. 프로젝트를 초기화하면 분석된 모든 결과가 사라집니다.



CLI 설정파일 만들기

테스트 네비게이터 뷰에서 선택한 프로젝트 정보를 바탕으로 CLI 환경에서 Controller Tester를 수행하는데 필요한 설정파일을 생성할 수 있습니다.

프로젝트 로그 수집

테스트 네비게이터 뷰에서 선택한 프로젝트의 로그 파일을 지정한 경로에 내보낼 수 있습니다.

특성

테스트 네비게이터 뷰에서 선택된 프로젝트에 대한 정보를 보거나 설정을 변경할 수 있습니다.
자세한 설명은 특성 페이지 에서 확인하실 수 있습니다.

20. 창 메뉴

창 메뉴에서는 활성화된 편집기와 동일한 새 편집기를 열거나, 퍼스펙티브 또는 새 뷰를 열고, 환경 설정을 볼 수 있습니다.

새 편집기

선택한 소스코드 편집기 또는 테스트 편집기를 복제합니다.

퍼스펙티브 열기

Controller Tester에 등록된 퍼스펙티브를 선택하여 열 수 있습니다.

뷰 표시

Controller Tester에 등록된 뷰를 선택하여 열 수 있습니다.

다른 이름으로 퍼스펙티브 저장

사용자가 재구성한 퍼스펙티브를 다른 이름으로 저장할 수 있습니다.

퍼스펙티브 재설정

퍼스펙티브를 초기 상태로 재설정할 수 있습니다.

퍼스펙티브 닫기

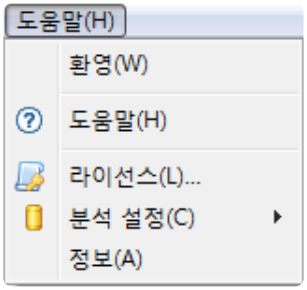
열려있는 퍼스펙티브를 닫을 수 있습니다.

환경설정

현재 도구에 적용된 설정을 확인하거나 변경할 수 있습니다.
자세한 내용은 환경설정 에서 확인하실 수 있습니다.

21. 도움말 메뉴

도움말 메뉴에서는 라이선스 설정, 분석 설정 변경 및 되돌리기 그리고 설치된 제품의 정보를 확인할 수 있습니다.



환영

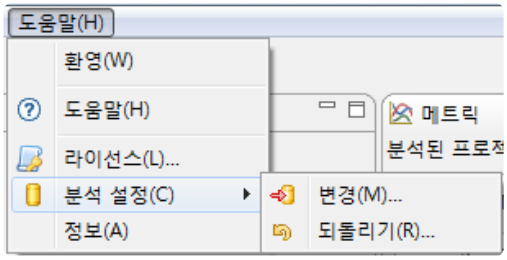
도구가 처음 실행됐을 때 보이는 환영 페이지를 다시 볼 수 있습니다.

분석 설정 변경

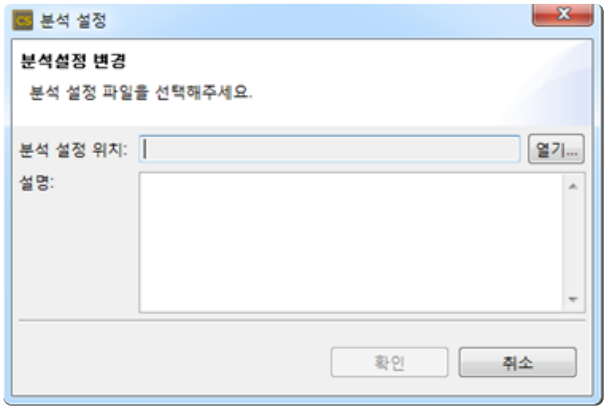
슈어소프트테크㈜에서 제공한 압축 파일을 입력하여 분석 설정 정보를 변경 할 수 있습니다.

사용방법

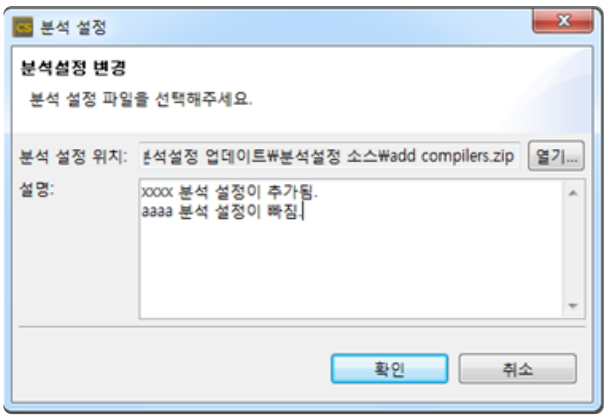
1. [도움말] -> [분석 설정] -> [변경] 메뉴를 선택합니다.



2. [열기...] 버튼을 클릭하여 압축된 분석 설정 파일을 선택하고 옵션으로 분석 설정 변경 사항을 [설명] 텍스트에 입력합니다.

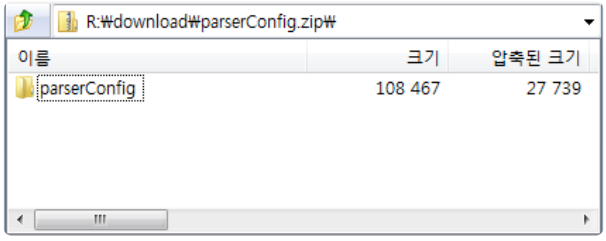


3. [확인] 버튼을 선택합니다.



압축된 분석 설정 파일 형식

- 아래 그림과 같이 압축파일을 열면 바로 [parserConfig] 디렉터리가 존재해야 합니다.
- 확장자는 “zip” 입니다.

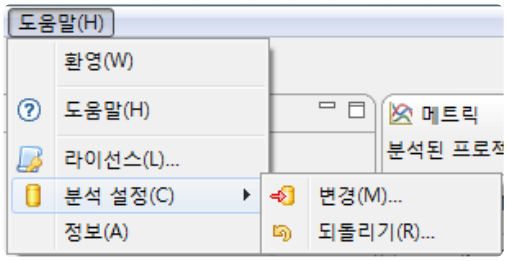


분석 설정 되돌리기

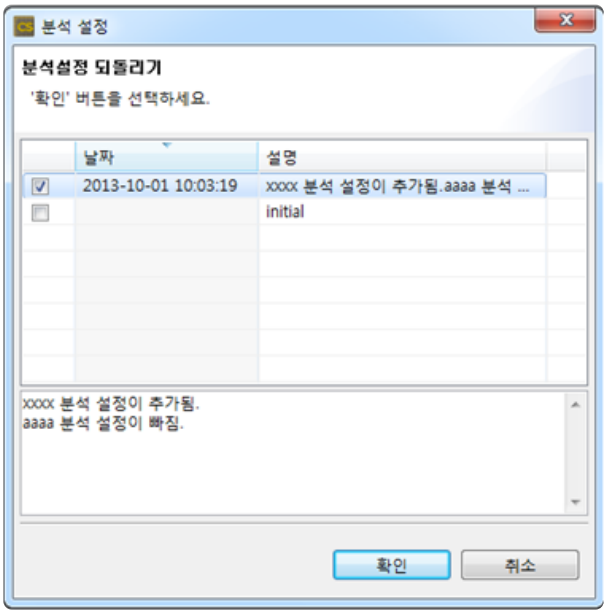
이전 분석 설정 정보 중 하나를 선택하여 되돌릴 수 있습니다.

사용 방법

1. [도움말] -> [분석 설정] -> [되돌리기] 메뉴를 선택합니다.



2. 기존 분석 설정 목록 중 되돌릴 분석 설정 정보를 체크하고, [확인] 버튼을 선택합니다.



정보

설치된 제품의 정보와 버전을 확인할 수 있습니다.

[설치 정보] 버튼을 클릭하면 설치된 기능과 플러그인을 확인할 수 있습니다.

22. 문제 해결

설치 시 보안 프로그램과의 충돌

일부 보안프로그램에서 도구 설치 패키지 실행을 중지시킵니다. 설치 과정에서 오류가 발생한다면, 일시적으로 보안프로그램 동작을 중지하고 설치를 다시 진행해주십시오.

제품 실행 시 작업공간 지정 후 제품이 구동되지 않는 경우

제품 최초 실행 시 CodeScroll의 전역 데이터가 \APPDATA\ 경로에 저장되어 있습니다. 이 경로에 대한 접근에 제약이 있을 경우 제품이 구동되지 않을 수 있습니다.
이를 해결하기 위한 방법으로 전역 데이터 경로를 설정할 수 있는 기능을 사용하시기 바랍니다.
설정 방법: CodeScroll.ini 파일의 –g default(기본값) 대신에 설정할 전역 데이터 경로를 입력합니다(-g 또는 –global, csc.ini 파일도 동일)
-g 옵션이 주어지지 않으면, 제품 최초 실행 시에 임의의 전역 데이터 경로를 설정하는 다이얼로그가 나타납니다. 이때 취소하면 기본 경로로 설정됩니다.

Windows Vista/7 에서 설치 및 실행

User Access Control(UAC) 기능이 비활성화 된 상태에서만 실행할 수 있습니다.
UAC를 비활성화하지 않고 사용하려면, Program Files 이하에 설치하지 않도록 설치 과정에서 디렉터리를 변경하십시오.
또한, 설치 및 실행 Windows 계정이 동일해야 하며, 반드시 관리자 계정으로 설치해야 합니다.

분석 실패 또는 Compile(전처리) 오류

테스트 수행이 불가능한 소스 코드 상태입니다. 컴파일/링크 플래그와 헤더파일/라이브러리 설정을 확인해주십시오.
설정 내용 확인 방법은 메뉴의 [환경 설정] -> [툴체인 설정]에서 가능합니다.
만약, 모든 설정 내용이 정상일 경우(실제 개발 환경(IDE 등)에서 문제 없이 Build가 가능할 경우)에도 오류가 발생하면, 슈어소프트테크㈜로 문의하시기 바랍니다.

모든 테스트 케이스 수행이 실패한 경우

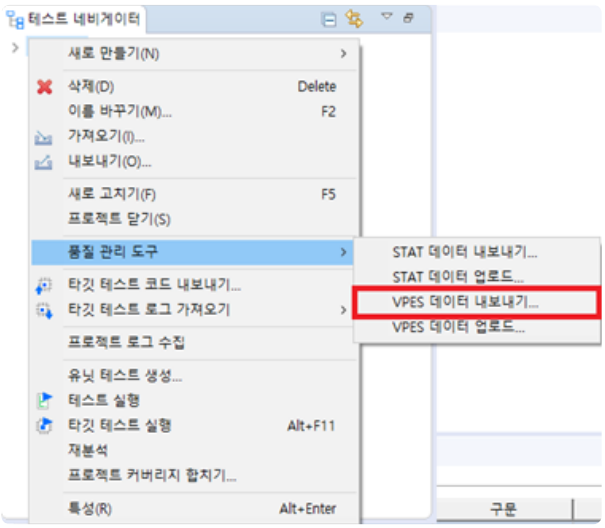
방화벽에 의해 테스트 엔진 동작이 차단되어서 발생하는 현상입니다. 테스트 수행 시 메모리에 로드 되는 테스트 엔진을 방화벽 설정의 예외 사항으로 추가하여야 정상적인 수행이 가능합니다.

툴체인 자동 등록이 실패하는 경우

PC에 설치된 툴체인 정보에 문제가 있어 자동 등록이 실패하는 경우, 툴체인 자동 등록 기능을 끌 수 있습니다. (`--disableAutoToolchain` 옵션)

- CodeScroll.ini 파일에 `--disableAutoToolchain` 옵션을 추가

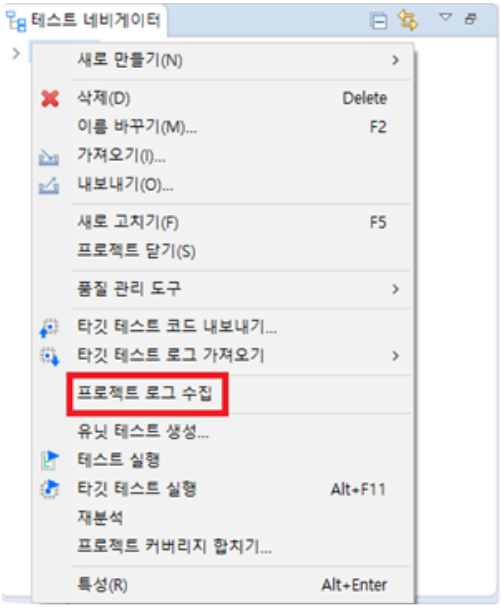
VPES에 데이터 업로드



프로젝트를 선택하고 우 클릭하면 메뉴에 [품질 관리 도구] -> [VPES에 데이터 업로드...] 기능이 있습니다. 이 기능에 대해서는 기술지원 연락처를 통해 문의하시기 바랍니다.

프로젝트 로그 수집

Controller Tester 사용 중 남기는 로그를 프로젝트 단위로 일괄 수집하여
내보내는 기능입니다.
Controller Tester 발생하는 오류로 인해 기술지원에 문의할 때 로그가
수집된 압축파일을 같이 보내주시면 문제를 더욱 빨리 해결할 수 있습니다.



프로젝트를 선택하고 우 클릭하면 메뉴에 [프로젝트 로그 수집] 기능이 있습니다.

기술지원

문제 발견 시 아래 기술지원 연락처로 문의하시기 바랍니다.

- support@suresofttech.com
- +82-2-6472-2800

23. CONTROLLER TESTER Target Plug-in

소개

Controller Tester Target Plug-in은 기본 Controller Tester의 기능을 보완합니다.
Controller Tester를 사용하면 자체 호스트 환경에서 테스트를 자동화할 수 있습니다. 즉, 소프트웨어 개발 환경(일반 PC)에서 테스트를 실행합니다. Controller Tester Target Plug-in을 사용하면 실제 임베디드 타깃 환경에서 Controller Tester의 테스트케이스를 실행할 수 있습니다. 이를 통해 호스트와 타깃 환경에서 테스트 결과가 같은지 쉽게 확인할 수 있고, 타깃 환경에 종속적인 요소 때문에 호스트 환경에서 실행할 수 없는 테스트를 실행할 수 있습니다.

23.1. 타깃 환경 설정하기

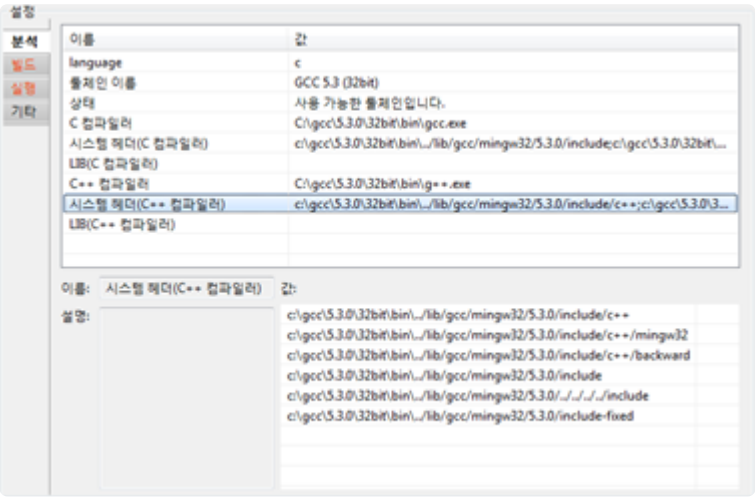
Controller Tester Target Plug-in를 사용하여 타깃 환경에서 테스트를 실행하려면 타깃 환경에 대한 정보를 입력해야 합니다.
Controller Tester Target Plug-in은 사용자가 입력한 타깃 환경에 대한 정보를 사용하여 테스트 하네스를 빌드하고 타깃 환경에서 실행한 결과를 자동으로 가져옵니다.

타깃 환경은 프로젝트 특성 페이지 또는 타깃 테스트 프로젝트 생성 마법사에서 설정할 수 있습니다.

타깃 환경 설정 화면

타깃 환경 설정은 기본 정보와 상세 설정으로 나뉩니다.

기본 정보를 입력하면, 입력이 필요한 상세 설정이 표시됩니다.



타깃 환경 상세 설정

타깃 환경 상세 설정은 분석, 빌드, 실행 그리고 기타로 나뉩니다.

카테고리	설명
분석	툴체인 정보가 표시되며, 타깃 테스트 프로젝트인 경우에는 분석을 위해 필요한 타깃 컴파일러 관련 설정이 표시됩니다.
빌드	테스트 소프트웨어를 빌드하기 위한 설정이 표시됩니다.
실행	타깃 환경에서 테스트를 실행하고 결과를 가져오기 위한 설정이 표시됩니다.
기타	그 외 설정이 표시됩니다. (프로그램 entry point 등)

각 카테고리별 필수 설정은 붉은색으로 표시됩니다. 카테고리별 필수 설정의 입력 여부에 따라 테스트 실행 버튼을 클릭했을 때의 동작이 아래와 같이 달라집니다.

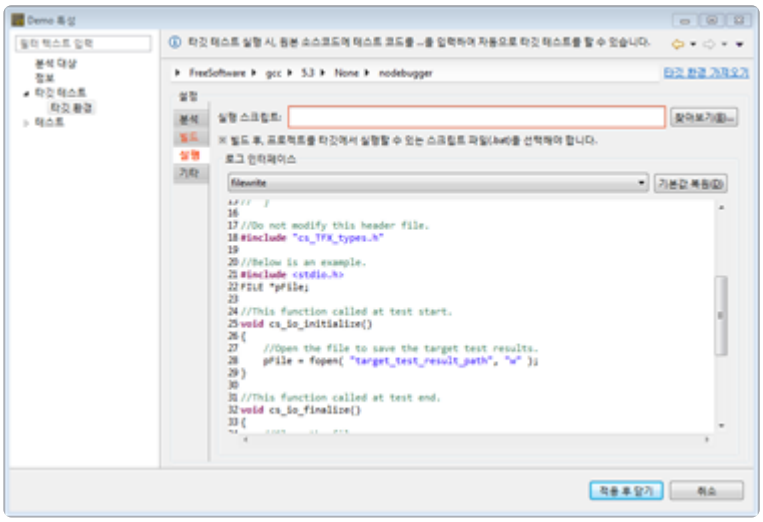
필수 설정 완료된 카테고리	설명
없음	테스트 실행 불가능
분석	테스트를 실행하면 원본 소스코드에 테스트 코드를 덮어씁니다. 테스트를 수행하기 위해서는 사용자가 수동으로 테스트를 빌드하고 실행해야 합니다.
분석, 빌드	테스트를 실행하면 테스트 코드를 빌드합니다. 테스트를 수행하기 위해서는 사용자가 수동으로 테스트를 타깃에서 실행햐 합니다.
분석, 빌드, 실행	타깃 환경에서 자동으로 테스트를 실행합니다.

타깃 테스트 결과 가져오기

Controller Tester Target Plug-in은 타깃 환경에서 테스트를 실행한 결과를 로그 형태로 저장하고 가져옵니다. 디버거를 사용하지 않을 경우에는 타깃 실행 결과를 저장하기 위한 설정(로그 인터페이스)과 가져오기 위한 설정(타깃 로그 수집기 설정 – 환경 설정)이 필요합니다.

로그 인터페이스

로그 인터페이스는 타깃 환경에서 테스트를 실행한 결과를 저장하기 위한 설정입니다. 로그 인터페이스는 타깃 환경에서 실제 수행되는 소스 코드의 형태로 작성합니다.



로그 인터페이스 구조

함수	설명
`viod cs_io_initialize()`	전송을 위한 초기 함수
`void_cs_io_finalize()`	전송 종료 함수

`void cs_io_flush()`	남은 데이터 전송함수
`void cs_io_putbyte(codescroll_byte v)`	1바이트 데이터 전송함수

타깃 로그 자동으로 가져오기

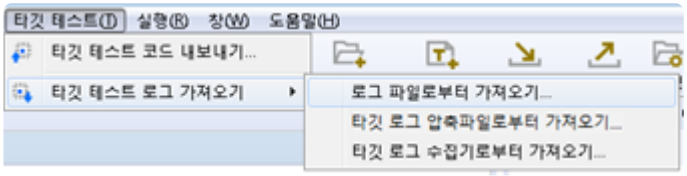
테스트 실행 결과인 타깃 로그는 타깃 로그 수집기를 통해 자동으로 가져올 수 있습니다. 타깃 로그 수집기 설정은 타깃 로그 수집기 와 타깃 테스트 환경 설정 의 [타깃 로그 수집기] 내용을 참고하시기 바랍니다.

타깃 로그 수동으로 가져오기

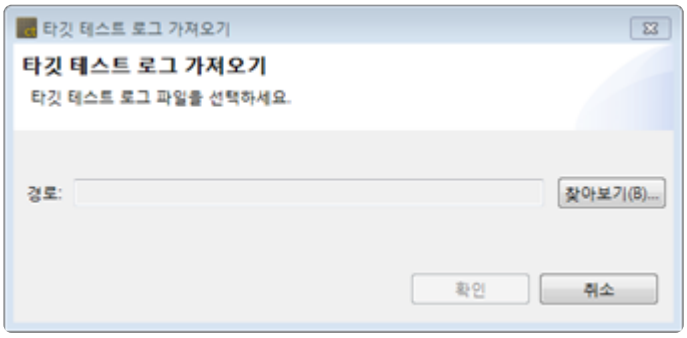
타깃 로그를 자동으로 가져올 수 없을 경우, 타깃 로그를 수동으로 가져올 수 있습니다.

로그 파일로부터 가져오기

1. 분석한 프로젝트를 선택한 후 전역 메뉴에 있는 [타깃 테스트] -> [타깃 테스트 로그 가져오기] -> [로그 파일로부터 가져오기...]를 클릭합니다.

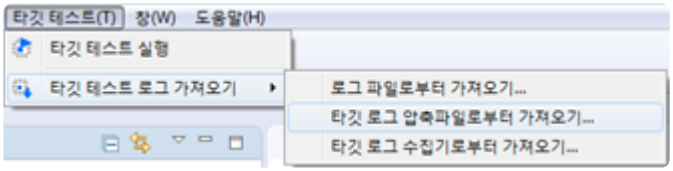


2. 타깃 테스트 로그 파일을 선택한 후 [확인] 버튼을 클릭합니다.

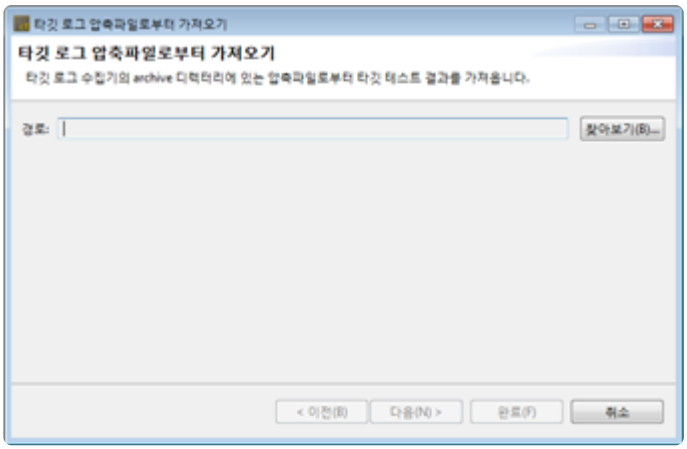


타깃 로그 압축파일로부터 가져오기

1. 분석한 프로젝트를 선택한 후 전역 메뉴에 있는 [타깃 테스트] -> [타깃 테스트 로그 가져오기] -> [타깃 로그 압축파일로부터 가져오기...]를 클릭합니다.



2. 타깃 로그 압축파일을 선택한 후 [다음] 버튼을 클릭합니다.

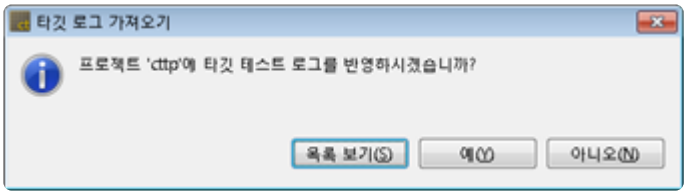


3. 가져올 타깃 테스트 로그를 체크하고 [완료] 버튼을 클릭합니다.
   !(zoom)!(zoom){img-import-target-log-zip-dlg-complete}!!

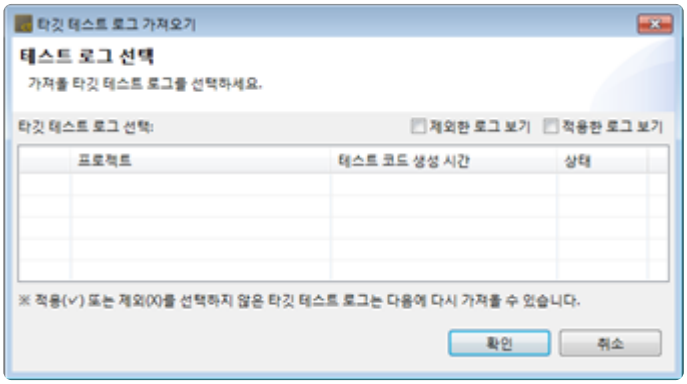
타깃 로그 수집기로부터 가져오기

1. 타깃 로그 수집기로부터 로그를 가져오기 위해서는 [환경설정] -> [타깃 테스트] -> [타깃 로그 수집기]에서 '기본 타깃 로그 수집기 사용'을 설정해야 합니다.
2. 분석한 프로젝트를 선택한 후 전역 메뉴에 있는 [타깃 테스트] -> [타깃 테스트 로그 가져오기] -> [타깃 로그 수집기로부터 가져오기...]를 클릭합니다.

3. [예] 버튼을 클릭하여 신규 타깃 테스트 로그를 모두 반영하거나, [목록 보기] 버튼을 클릭합니다.

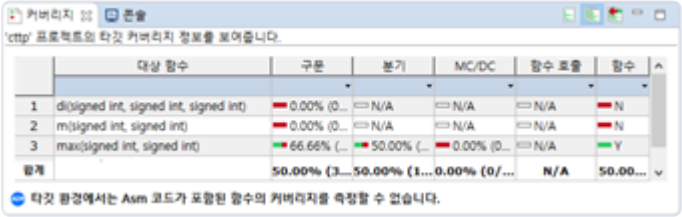


4. [목록 보기] 버튼을 클릭한 경우, 가져올 타깃 테스트 로그를 체크하고 [확인] 버튼을 클릭합니다.



결과

가져오기 완료 후 커버리지 뷰에서 타깃 커버리지 정보를 확인할 수 있습니다.



> 타깃 테스트 수행 결과 로그를 가져오기 전, 호스트의 테스트 케이스, 소스 파일, 함수 정보 등 변경 되었을 경우, 타깃 테스트 수행 정보와 호스트의 테스트 정보가 유효하지 않아 [타깃 테스트 로그 가져오기] 가 실패할 수 있습니다.

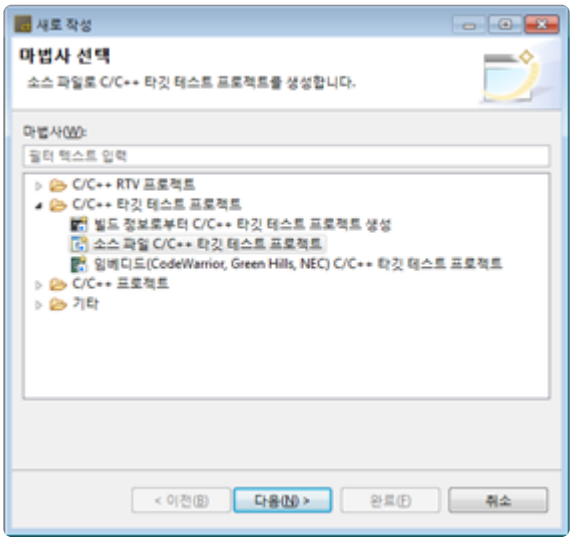
23.2. C/C++ 타깃 테스트 프로젝트

C/C++ 타깃 테스트 프로젝트(이하 타깃 테스트 프로젝트)는 타깃 환경 시험을 위한 프로젝트입니다. 타깃 테스트 프로젝트는 호스트 시험을 지원하지 않습니다.

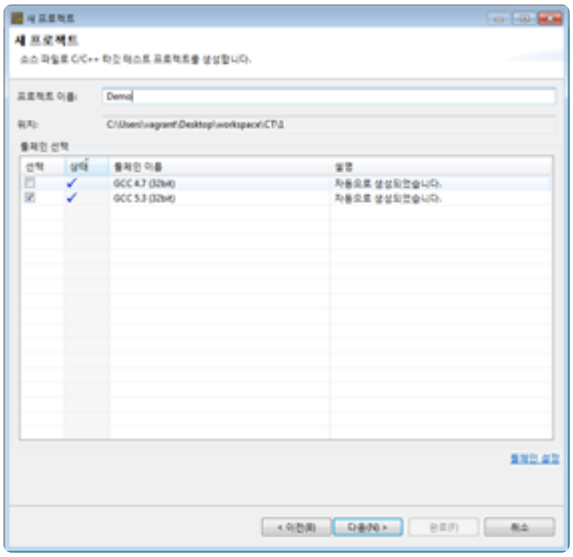
타깃 테스트 프로젝트 생성하기

소스 파일 C/C++ 타깃 테스트 프로젝트

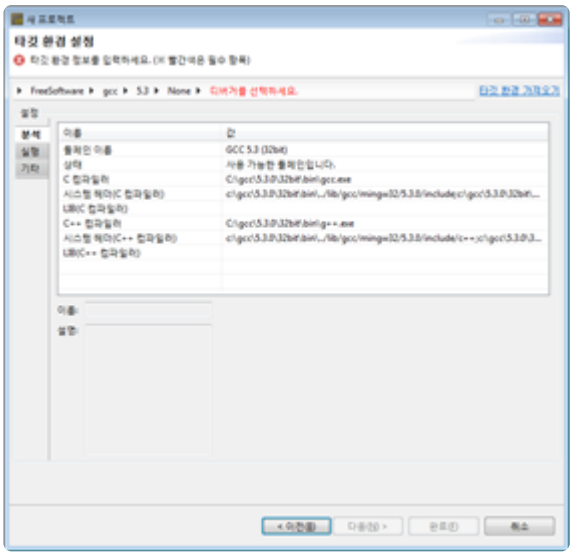
1. 전역 메뉴에 있는 [파일] -> [새로 만들기] -> [기타…]를 선택하고 [소스 파일 C/C++ 타깃 테스트 프로젝트]를 클릭하여 프로젝트를 생성합니다.



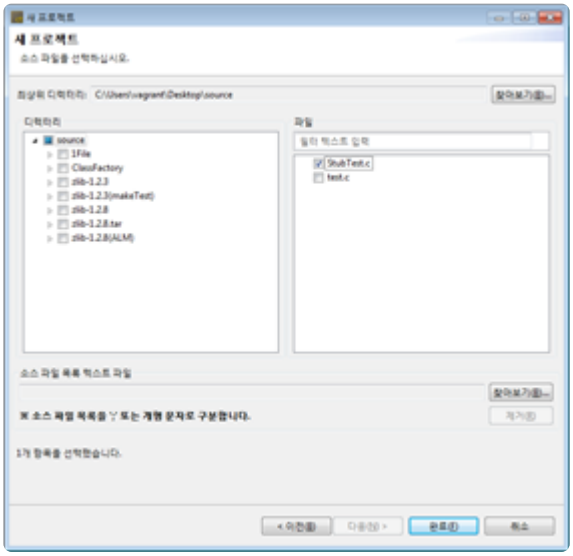
2. 프로젝트 이름을 입력하고 사용할 툴체인을 선택합니다.



3. 타깃 환경의 기본 정보를 선택한 후 각 카테고리별 상세 설정을 입력한 후 [다음] 버튼을 클릭합니다.

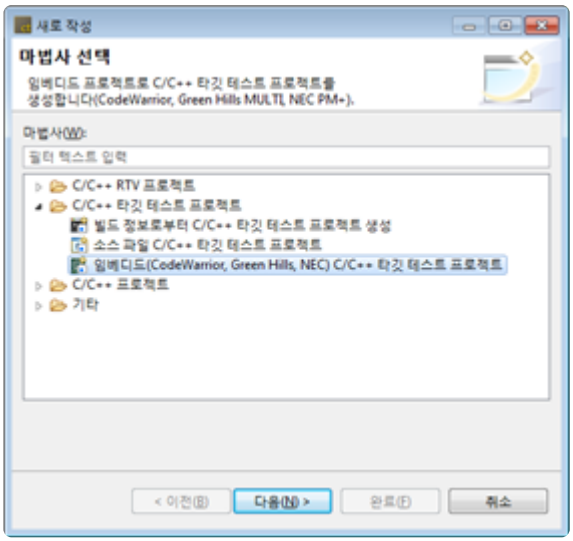


4. 테스트 대상 소스 파일을 선택하고 [완료] 버튼을 클릭합니다.

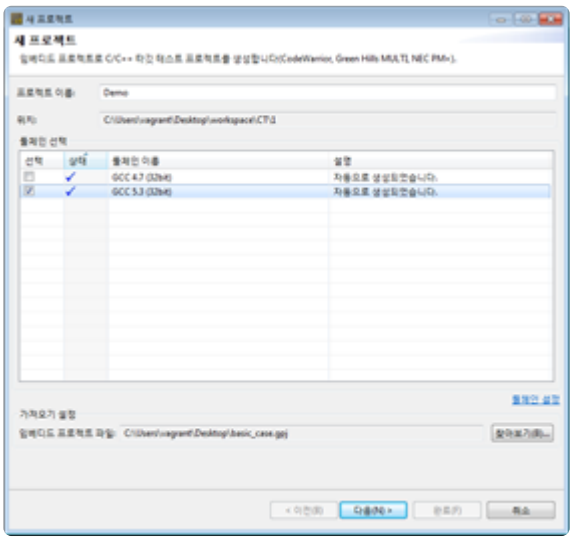


임베디드(CodeWarrior, Green Hills, NEC) C/C++ 타깃 테스트 프로젝트

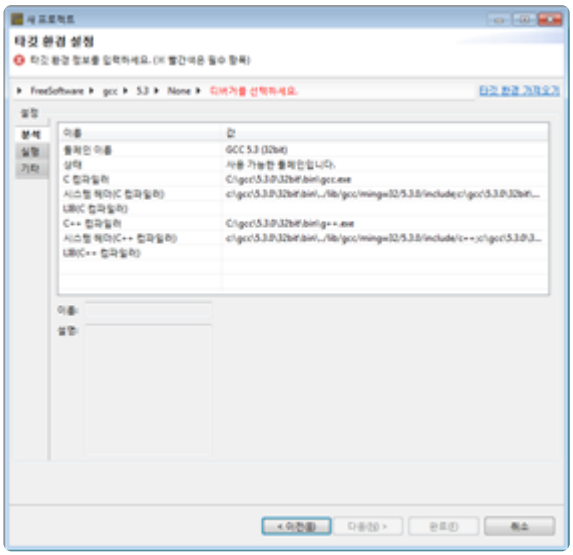
1. 전역 메뉴에 있는 [파일] -> [새로 만들기] -> [기타…]를 선택하고 [임베디드(CodeWarrior, Green Hills, NEC) C/C++ 타깃 테스트 프로젝트]를 클릭하여 프로젝트를 생성합니다.



2. 프로젝트 이름을 작성하고 사용할 툴체인을 선택합니다. 툴체인을 선택하고 난 후, 테스트할 임베디드 프로젝트 파일을 가져옵니다. 현재 Controller Tester에서 지원하는 임베디드 프로젝트는 CodeWarrior, Green Hills, NEC 입니다.

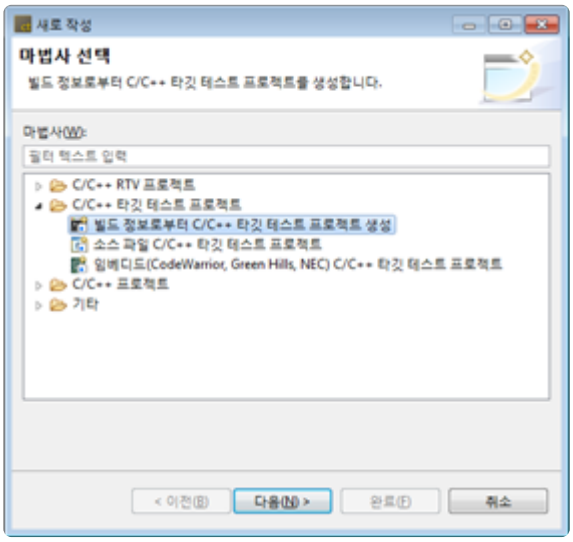


3. 타깃 환경의 기본 정보를 선택한 후 각 카테고리별 상세 설정을 입력합니다.

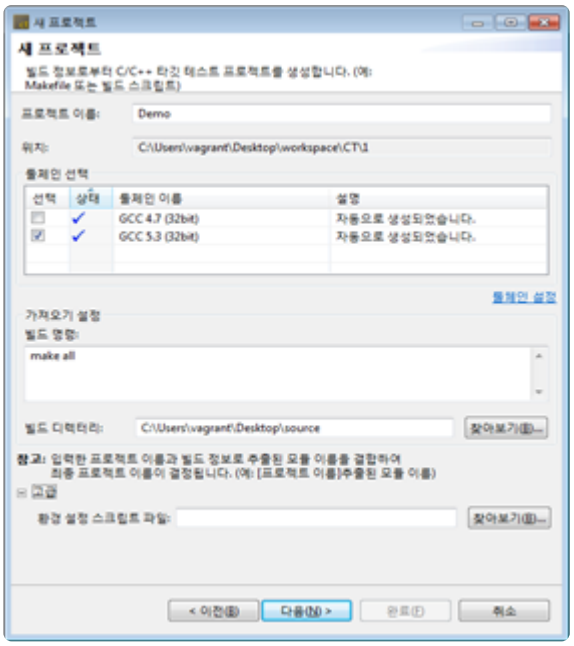


빌드 정보로부터 C/C++ 타깃 테스트 프로젝트 생성

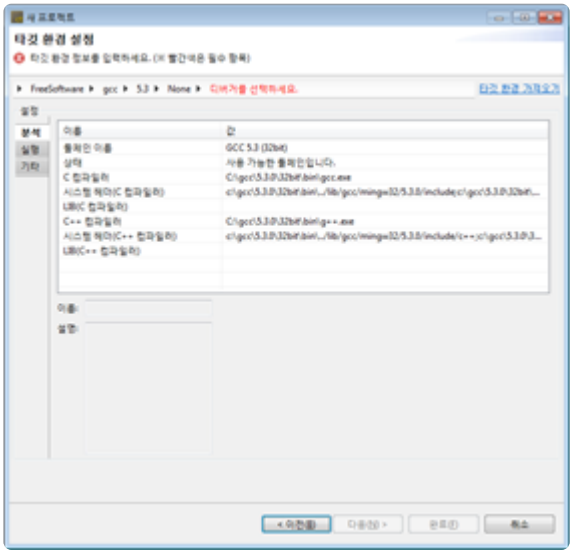
1. 전역 메뉴에 있는 [파일] -> [새로 만들기] -> [기타…]를 선택하고 [빌드 정보로부터 C/C++ 타깃 테스트 프로젝트 생성]을 클릭하여 프로젝트를 생성합니다.



2. 프로젝트 이름을 작성하고 사용할 툴체인을 선택합니다. 툴체인을 선택하고 난 후, 작성된 Makefile 및 빌드 스크립트 경로를 빌드 디렉터리로 지정하고 빌드 명령을 작성합니다. [고급] 메뉴에서 빌드 환경을 설정하는 스크립트 파일을 선택할 수 있습니다.



3. 타깃 환경의 기본 정보를 선택한 후 각 카테고리별 상세 설정을 입력합니다.



> ! 컴파일러 경로가 유효하지 않은 툴체인은 타깃 테스트 프로젝트 생성 시 선택할 수 없습니다.

23.3. 타깃 테스트 실행 설정하기

타깃 테스트 실행 관련 설정은 [프로젝트] -> [특성]의 타깃 테스트 페이지에서 확인할 수 있습니다.

커버리지 종류

타깃 테스트 수행 시 측정할 커버리지 종류(구문, 분기, MC/DC)를 선택할 수 있습니다.

커버리지 측정 대상

테스트 대상 함수 외에 추가로 커버리지를 측정할 함수를 선택할 수 있습니다.

테스트 대상 함수가 호출하는 모든 함수	테스트 대상 함수가 호출하는 모든 함수에 대해 커버리지를 측정합니다.
이전 테스트 실행 시 호출된 함수	이전 테스트 수행 시 커버리지가 측정된 함수들을 포함합니다.

테스트 케이스 실행

테스트 케이스 실행 방식을 선택할 수 있습니다.

한 번에 실행하기	모든 테스트케이스를 한 번에 타깃에 로드하고 실행합니다. 테스트케이스 실행이 이전 테스트케이스 실행 결과에 영향을 받습니다.
각각 실행하기	테스트케이스 하나씩 타깃에 로드하고 실행하기를 반복합니다. 한 번에 실행하는 것보다 시간이 오래 걸리지만, 적은 메모리를 사용합니다.

23.4. 타깃 로그 수집기

타깃 로그 수집기는 타깃 테스트 결과를 수집하여 Controller Tester로 전송하는 서버입니다. 타깃 테스트 결과 수집 방법은 크게 통신과 파일 스캔으로 나뉘며, 지원하는 통신 프로토콜과 스캔 가능한 파일 형식은 다음과 같습니다.

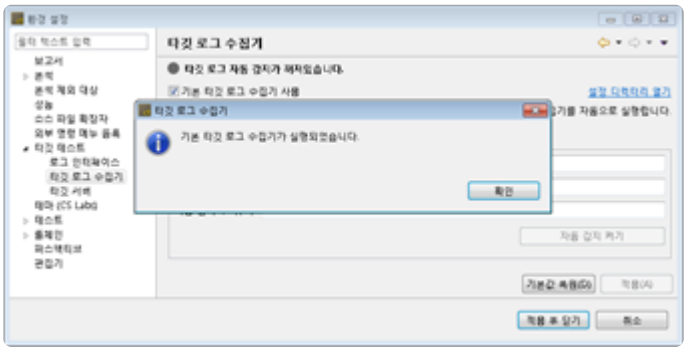
통신 프로토콜	TCP, UDP, UART
파일 형식	타깃 로그(text), 메모리 덤프(Hex, Intel HEX)

설치

Controller Tester 설치 시 타깃 로그 수집기가 자동으로 설치됩니다.

실행

[환경 설정] –> [타깃 테스트] -> [타깃 로그 수집기]에서 [기본 타깃 로그 수집기 사용]을 체크하면 타깃 로그 수집기가 타깃 테스트 결과 수집을 시작합니다.
TargetLogCollector.zip을 압축 해제하여 사용할 경우, 압축 해제 디렉터리안에 TargetLogCollector.exe를 windows 명령프롬프트에서 실행합니다.



설정

타깃 로그 수집기 설치 경로에 ‘settings.ini’ 파일을 통해 설정할 수 있으며, 옵션은 다음과 같습니다.

[LogReceiveServer]	타깃 테스트 결과 수신 관련 설정
port	TCP, UDP 통신 포트
protocol	타깃 테스트 결과 수신을 위한 통신 프로토콜
timeout	연결 타임아웃
lastString	데이터의 마지막을 나타내는 문자열

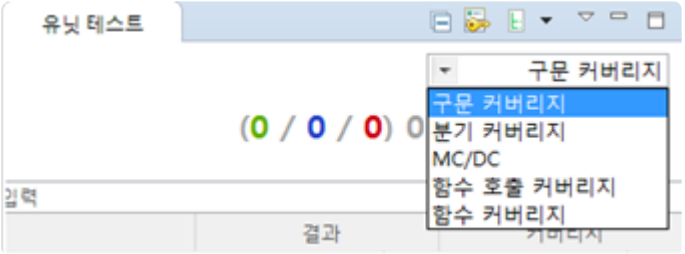
serialPort	Serial 통신 포트 (Windows: COM#, Linux: /dev/ttyS#)
baudRate	Serial 통신 설정
dataBits	Serial 통신 설정
stopBits	Serial 통신 설정
parity	Serial 통신 설정
flowControl	Serial 통신 설정
[ScanLog]	파일 스캔 관련 설정
dir	스캔 대상 디렉터리
fileExtension	스캔 대상 파일 확장자(공백일 경우 모든 파일 스캔)
[LogSendServer]	Controller Tester와 통신 관련 설정
port	타깃 로그 파일 전송 포트

timeout: 마지막 수신으로부터 지정한 시간(초) 동안 데이터 전송이 없는 경우, 수신한 데이터를 저장합니다.
dir: 지정한 디렉터리에 타깃 로그 파일(text, hex)을 입력하면 타깃 로그 수집기가 인식합니다.

23.5. 타깃 테스트 결과 확인

커버리지

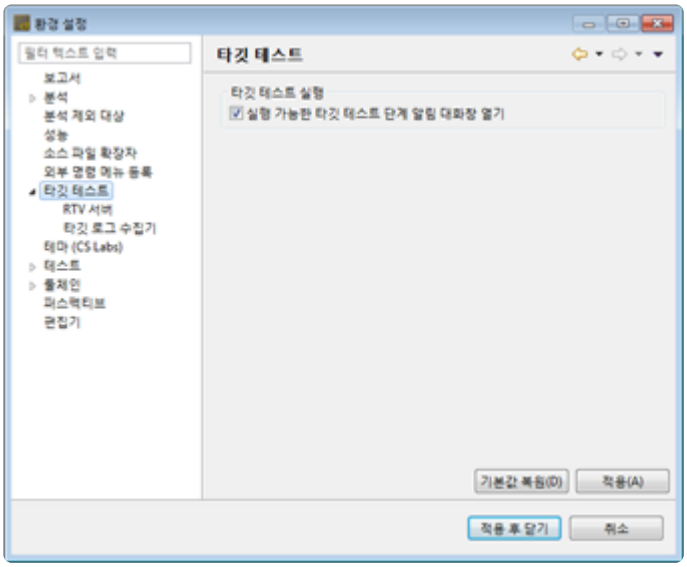
[유닛 테스트] 뷰의 커버리지 메뉴에서 표시할 커버리지 종류를 변경할 수 있습니다.



23.6. 환경 설정

타깃 테스트

타깃 테스트 실행 시 실행 가능한 타깃 테스트 단계 알림 대화창 열기 여부를 설정할 수 있습니다.



타깃 로그 수집기

사용할 타깃 로그 수집기 정보와 타깃 테스트 결과 자동 감지 여부를 설정합니다. [자동 감지 켜기] 버튼을 클릭하여 타깃 로그 자동 감지를 켤 경우, 지정한 자동 감지 주기마다 타깃 로그 수집기로부터 새로운 타깃 테스트 결과를 받아옵니다.

타깃 로그 수집기 설정	설명
자동 감지 주기(초)	새로운 타깃 테스트 결과 확인 주기
아이피(IP)	타깃 로그 수집기가 실행 중인 서버 또는 PC의 아이피
포트(Port)	타깃 로그 수집기의 통신 포트

! 기본 타깃 로그 수집기를 사용할 경우, 자동 감지 주기(초)만 설정합니다.

24. CODESCROLL RTV(Remote Target Verifier)

CODESCROLL RTV 소개

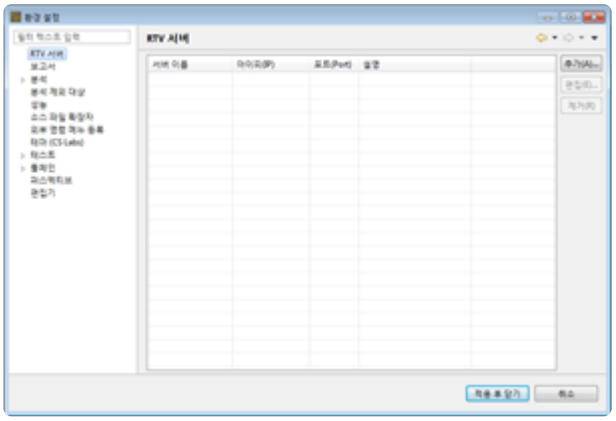
Controller Tester RTV는 RTV 환경에서 자동화된 테스트를 수행할 수 있도록 도와줍니다.
RTV 환경에 Controller Tester RTV 서버를 설치하고 기본 설정을 마치면, Controller Tester RTV 클라이언트를 사용하여 호스트 환경에서 테스트를 수행하는 것처럼 쉽게 RTV 테스트를 수행할 수 있습니다.
RTV 환경에 Controller Tester RTV 서버를 설치하는 방법은 본 문서 2페이지에 있는 기술지원 연락처를 통해 문의해주시기 바랍니다.

CODESCROLL RTV 지원 OS

서버: Linux (RHEL, Ubuntu, Debian, Fedora 기반), 그 외 문의
클라이언트: Windows 7 / 10

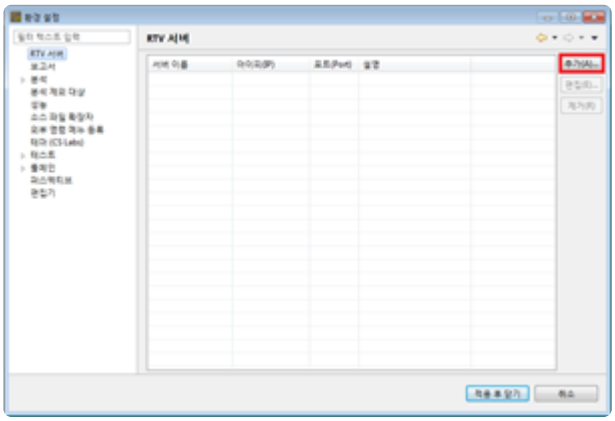
24.1. RTV 서버 설정

Controller Tester RTV의 기능을 사용하기 위해서는 Controller Tester RTV 서버가 설치된 RTV 서버를 추가해야 합니다.
RTV 서버는 [창] -> [환경 설정] -> [RTV 서버]에서 추가할 수 있습니다.



RTV 서버 추가

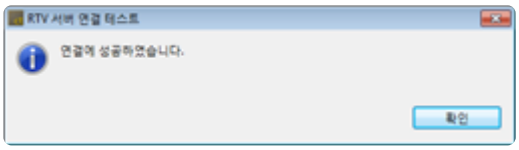
1. [창] -> [환경 설정] -> [RTV 서버]에서 [추가] 버튼을 클릭합니다.



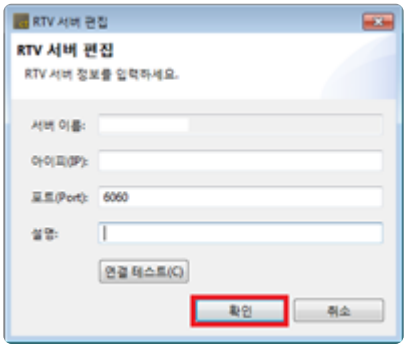
2. [RTV 서버 추가] 창에서 RTV 서버 정보를 입력합니다.

> ! 서버 이름은 Windows의 파일 이름으로 사용할 수 있는 문자만 허용하며, RTV 서버를 추가한 뒤에 수정할 수 없습니다.

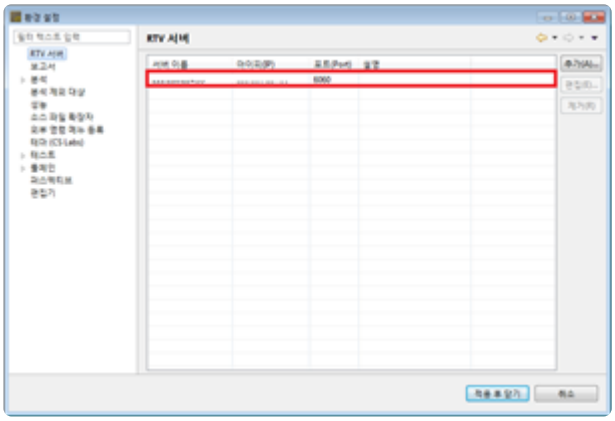
3. [연결 테스트] 버튼을 클릭하여 입력한 RTV 서버와의 연결 상태를 확인합니다.



4. RTV 서버와 연결 테스트에 성공했다면, [확인] 버튼을 클릭하여 RTV 서버를 추가합니다.

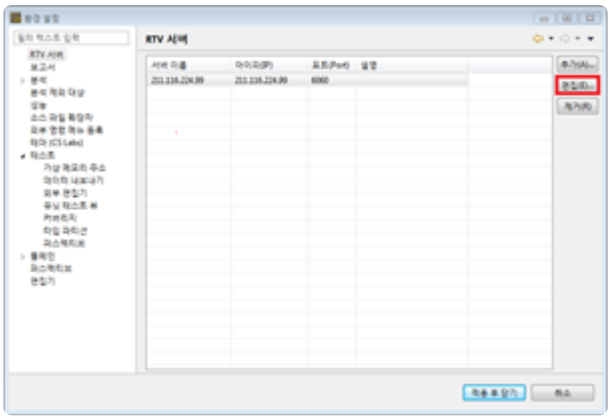


5. RTV 서버가 추가된 것을 확인할 수 있습니다.

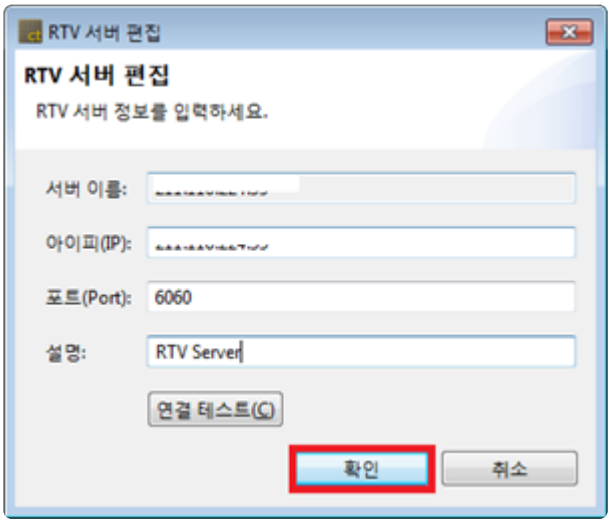


RTV 서버 편집

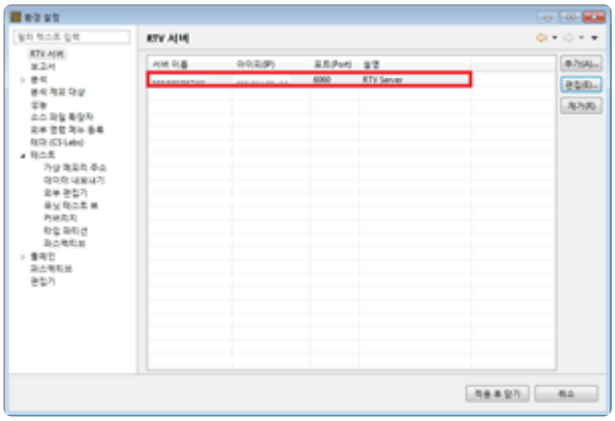
1. RTV 서버 목록에서 서버를 더블 클릭하거나, 서버를 선택한 뒤에 [편집] 버튼을 클릭합니다.



2. [RTV 서버 편집] 창에서 서버 정보를 편집한 뒤에, [확인]을 클릭합니다.



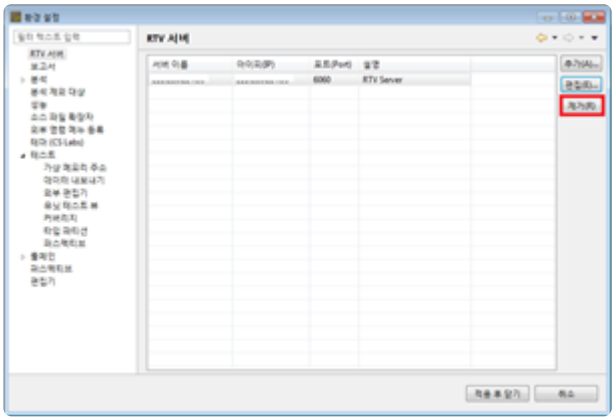
3. RTV 서버 정보가 편집된 것을 확인할 수 있습니다.



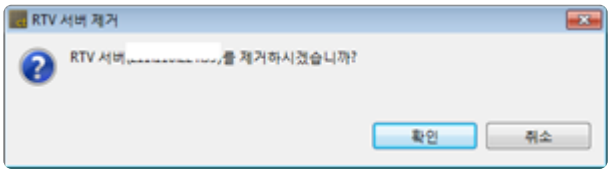
RTV 서버 제거

RTV 서버를 더 이상 사용하지 않을 경우, 제거할 수 있습니다.
RTV 서버를 제거하면, RTV 서버와 연관된 툴체인 또는 프로젝트의 리소스가 클라이언트 환경에서 삭제됩니다 (해당 RTV 서버로부터 추가한 RTV 툴체인이 존재하면 삭제할 수 없음).

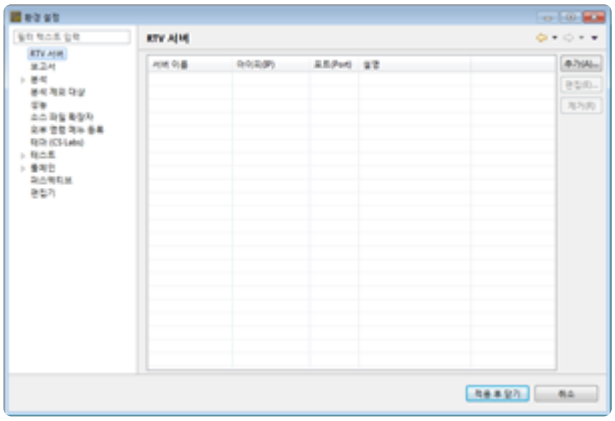
1. RTV 서버 목록에서 서버를 선택한 뒤에 [제거] 버튼을 클릭합니다.



2. [RTV 서버 제거] 창에서 [확인] 버튼을 클릭합니다(RTV 서버 리소스 크기에 따라 1~2분 정도 시간이 소요될 수 있습니다).



3. RTV 서버가 제거된 것을 확인할 수 있습니다.



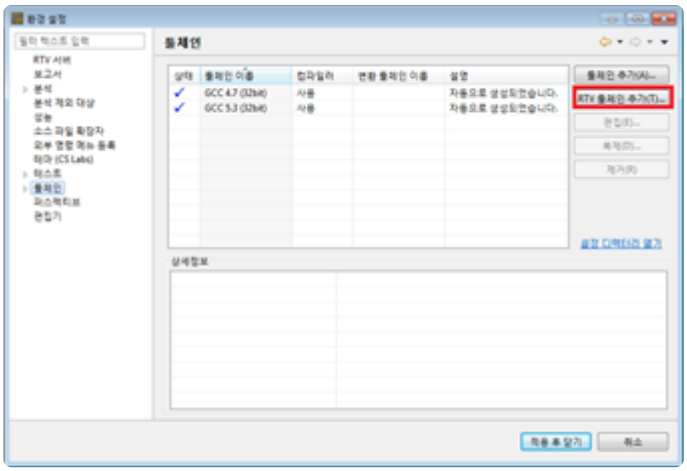
24.2. RTV 툴체인 설정

RTV 툴체인

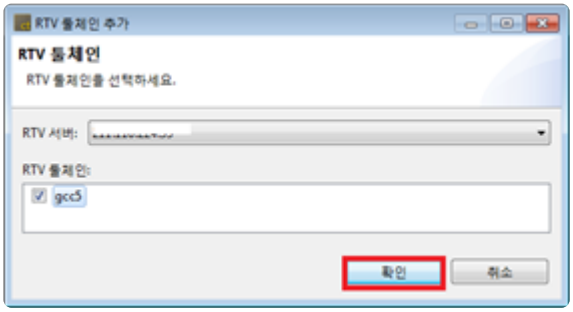
Controller Tester RTV로 RTV 환경에서 테스트를 빌드하고 실행하려면, 테스트 대상 소스 파일의 툴체인(컴파일러) 정보를 설정해야 합니다.
이 장에서는 Controller Tester RTV 서버가 설치된 RTV 환경에서 미리 추출해놓은 RTV 툴체인 정보를 클라이언트에서 추가, 편집, 삭제하는 방법을 설명합니다.
(RTV 환경에서 툴체인 정보를 추출하는 방법은 문제 해결 에 기술지원 연락처를 통해 문의해주시기 바랍니다.)

RTV 툴체인 추가

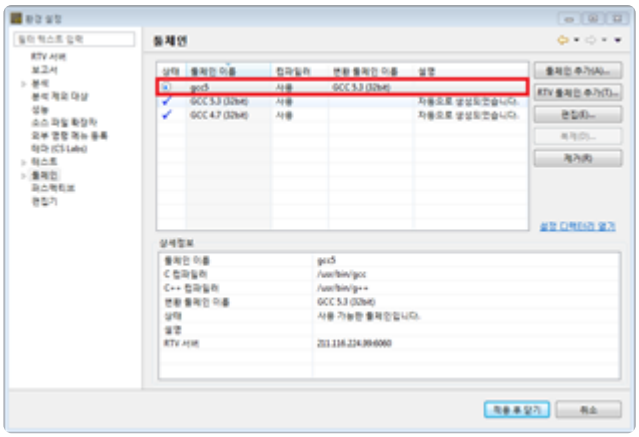
1. [창] -> [환경 설정] -> [툴체인] 창에서 [RTV 툴체인 추가] 버튼을 클릭합니다. **(RTV 서버가 없다면, [RTV 서버 추가] 창이 먼저 나타납니다.)**



2. [RTV 툴체인 추가] 창에서 RTV 서버를 선택하면 서버에 추출되어 있는 RTV 툴체인 목록을 확인할 수 있습니다. 추가하려는 툴체인을 선택한 뒤에 [확인] 버튼을 클릭합니다.



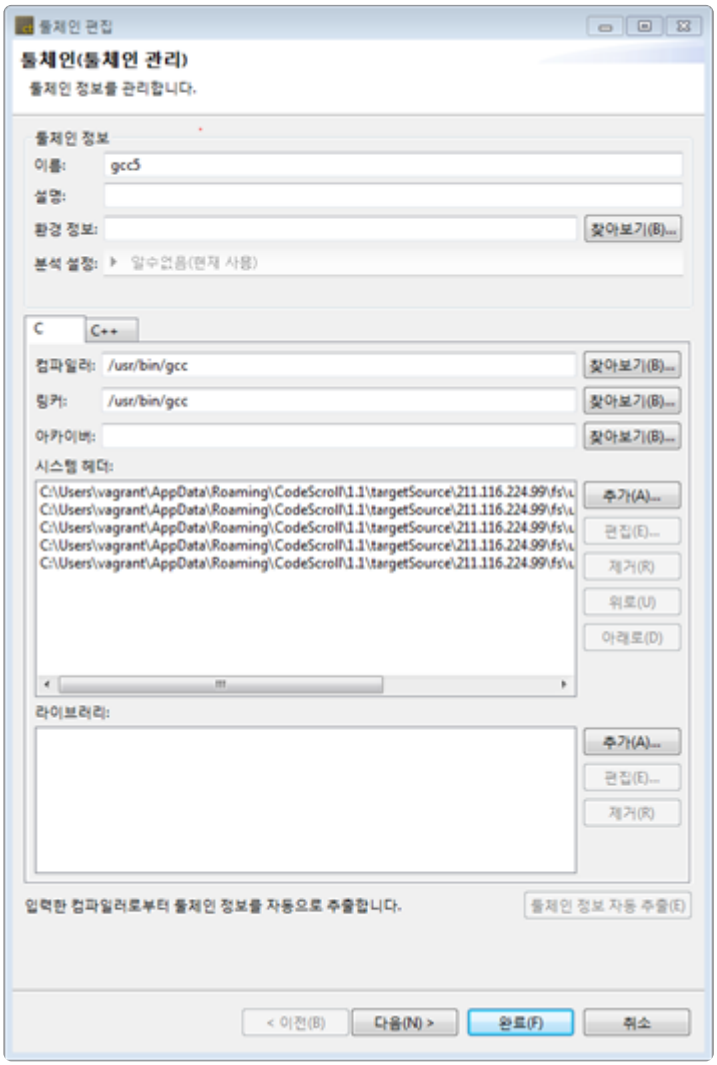
3. RTV 서버로부터 툴체인 정보를 가져옵니다(시스템 헤더 크기에 따라 1~2분 정도 시간이 소요될 수 있습니다). 작업이 완료되면 RTV 툴체인이 추가된 것을 확인할 수 있습니다.



4. 추가된 RTV 툴체인을 선택하면 상세정보 화면에서 RTV 서버의 정보가 표시되는 것을 확인할 수 있습니다.

RTV 툴체인 편집

1. 편집할 툴체인을 선택하고 [편집] 버튼을 클릭합니다.



2. RTV 툴체인 정보를 편집합니다.
3. [완료] 버튼을 클릭합니다.

RTV 툴체인 복제

RTV 툴체인은 복제 기능을 제공하지 않습니다.

RTV 툴체인 삭제

삭제할 툴체인을 선택한 후 [삭제] 버튼을 클릭합니다.
(삭제하려는 툴체인과 연관된 프로젝트가 있다면, 삭제 이후에 해당 프로젝트를 정상적으로 사용할 수 없습니다. 프로젝트의 모듈 특성 페이지에서 툴체인을 다시 설정해야 합니다.)

RTV 툴체인 가져오기/내보내기

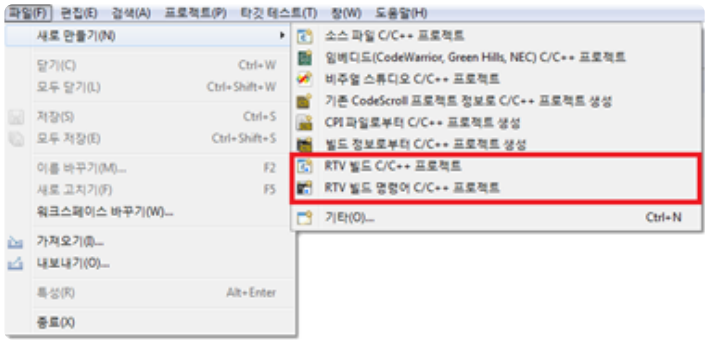
RTV 툴체인은 RTV 서버를 통해서 추가할 수 있기 때문에 가져오기/내보내기 기능은 제공하지 않습니다.

24.3. RTV 프로젝트 만들기

RTV 환경에 존재하는 소스 파일을 테스트하기 위해 RTV 프로젝트를 생성합니다.

RTV 환경에서 소스 파일을 빌드할 때, RTV 프로젝트 생성에 필요한 정보를 추출할 수 있습니다.
RTV 환경에 미리 추출해놓은 소스 파일 빌드 정보(Controller Tester RTV 서버 유틸리티 사용)를 이용하거나, 대상 소스 파일을 빌드하는 동시에 정보를 추출하여 RTV 프로젝트를 생성하는 방법을 설명합니다.

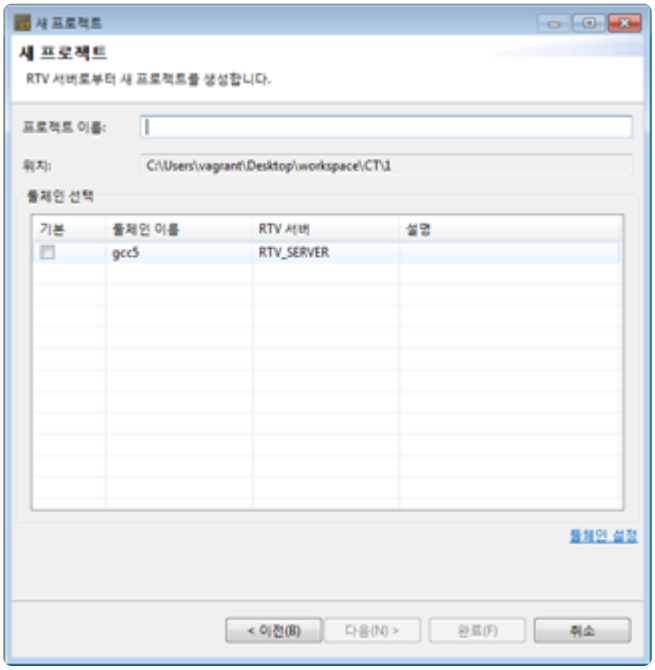
RTV 프로젝트를 생성하려면 [프로젝트 생성 마법사]를 실행합니다.
[파일] -> [새로 만들기] 또는 단축 아이콘을 클릭합니다.



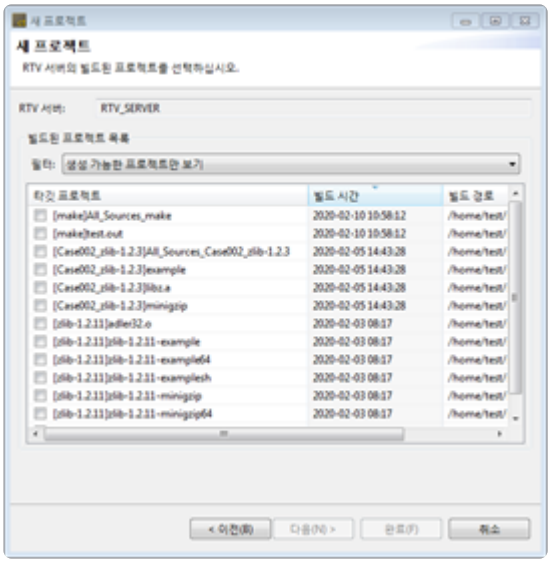
RTV 빌드 C/C++ 프로젝트

RTV 환경에 미리 추출해놓은 소스 파일 빌드 정보(Controller Tester RTV 서버 유틸리티 사용)를 이용하여 RTV 프로젝트를 생성합니다.

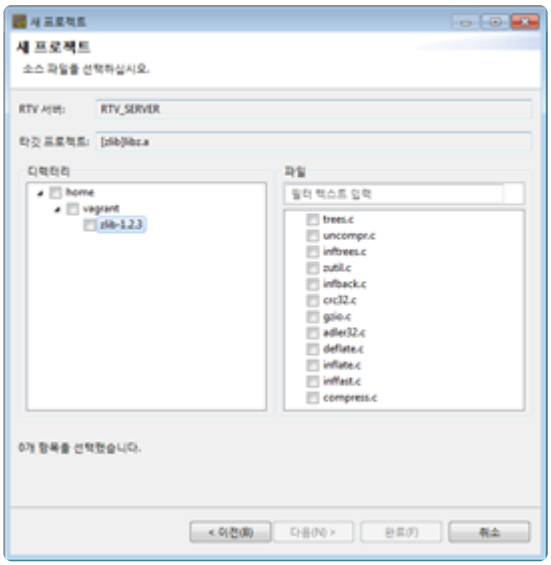
1. [프로젝트 이름]에 프로젝트 이름을 입력합니다.



2. 툴체인 목록에 RTV 툴체인 목록이 나타납니다. 사용할 RTV 툴체인을 선택합니다.
3. 툴체인을 선택한 뒤에 [다음] 버튼을 클릭합니다. 선택한 툴체인을 가져온 RTV 서버에 미리 추출되어 있는 소스 파일 빌드 목록을 확인할 수 있습니다(※ 생성된 프로젝트 중 'All_Sources_프로젝트명'는 csbuild를 통해 빌드 캡쳐 된 모듈의 모든 소스파일을 합친 프로젝트 입니다. 이 프로젝트는 빌드 여부를 고려하지 않고 합쳐진 프로젝트이기 때문에 정상적으로 테스트 수행이 되지 않을 수 있습니다).



4. 빌드된 목록에서 RTV 프로젝트를 생성할 항목을 선택한 뒤에 [다음] 버튼을 클릭합니다.

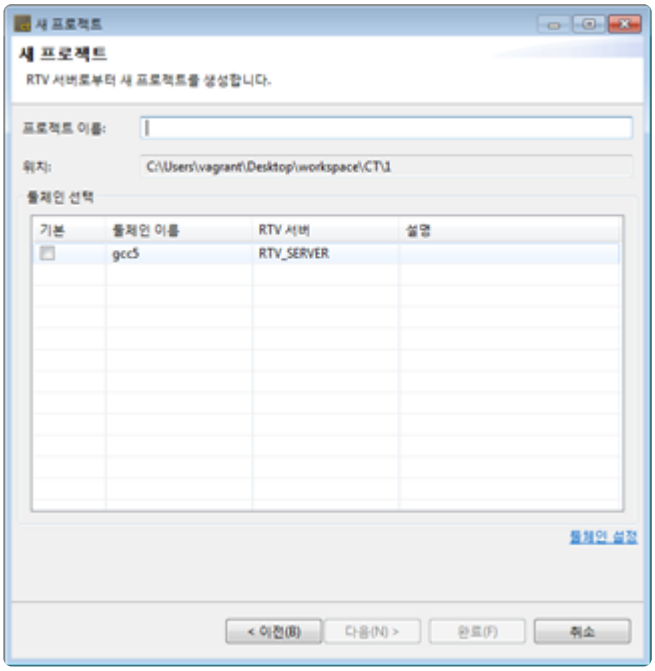


5. RTV 프로젝트를 생성할 때 포함할 소스 파일(테스트 대상)을 선택한 뒤에 [완료] 버튼을 클릭하여 프로젝트를 생성합니다.

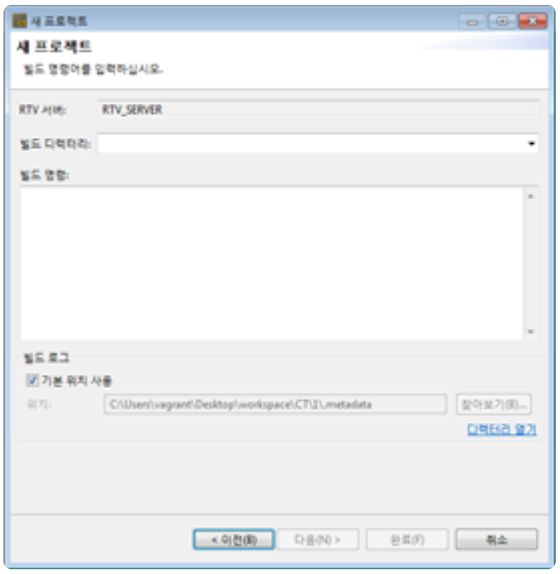
RTV 빌드 명령어 C/C++ 프로젝트

테스트 대상 소스 파일을 빌드하는 동시에 정보를 추출하여 RTV 프로젝트를 생성합니다. 빌드 명령어를 입력하여 RTV 서버에서 소스 파일을 빌드하면서 RTV 프로젝트를 생성합니다.

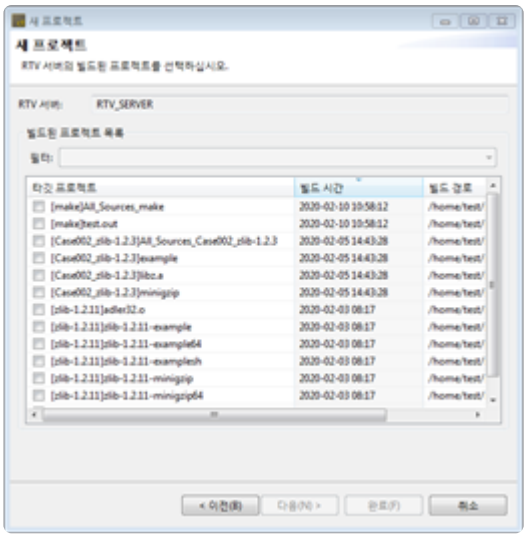
1. [프로젝트 이름]에 프로젝트 이름을 입력합니다.



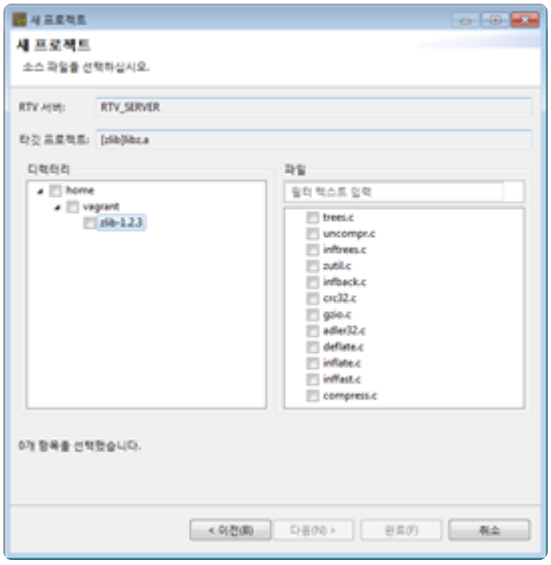
2. 툴체인 목록에 RTV 툴체인 목록이 나타납니다. 사용할 툴체인을 선택합니다.
3. 툴체인 선택한 뒤에 [다음] 버튼을 클릭합니다. 선택한 툴체인을 가져온 RTV 서버에서 빌드를 수행하게 됩니다.



4. [빌드 디렉터리]에 RTV 서버에서 빌드를 수행할 경로(예: make파일이 있는 디렉터리)를 입력하고, 빌드 명령어를 입력합니다.
5. [빌드 로그]에 빌드 결과 파일을 생성할 경로를 지정합니다.
6. 빌드에 필요한 정보를 입력한 뒤에 [다음] 버튼을 클릭합니다. 입력된 빌드 명령어로 RTV 서버에서 소스 파일을 빌드합니다(빌드 대상에 따라 많은 시간이 소요될 수 있습니다).
7. 빌드가 완료되면, 빌드된 프로젝트 목록이 나타납니다(※ 생성된 프로젝트 중 'All_Sources_프로젝트명'는 csbuild를 통해 빌드 캡쳐 된 모듈의 모든 소스파일을 합친 프로젝트 입니다. 이 프로젝트는 빌드 여부를 고려하지 않고 합쳐진 프로젝트이기 때문에 정상적으로 테스트 수행이 되지 않을 수 있습니다).



8. 빌드된 목록에서 RTV 프로젝트를 생성할 항목을 선택한 뒤에 [다음] 버튼을 클릭합니다.



9. RTV 프로젝트를 생성할 때 포함할 소스 파일(테스트 대상)을 선택한 뒤에 [완료] 버튼을 클릭하여 프로젝트를 생성합니다.

RTV 빌드 명령어

빌드 정보로부터 프로젝트를 생성하려면, 빌드 명령어에 csbuild 명령어를 사용해야 합니다.

csbuild 명령어

csbuild는 소프트웨어의 빌드를 수행할 때, 빌드되는 소프트웨어의 프로젝트 구성 정보를 자동으로 추출하여 Controller Tester의 프로젝트를 구성해주는 유틸리티 프로그램입니다. Shell에서 make와 같은 빌드 시스템을 이용하는 경우, 프로젝트를 자동으로 설정합니다.

csbuild 사용법

기존의 빌드 명령어 앞에 csbuild 명령어를 붙입니다.

- 예) `make` → `csbuild capture make`

csbuild 옵션

new : 빌드 명령어 추출 전 필요한 정보를 생성합니다. `--toolchain` 옵션으로 툴체인을 지정하고 `--project` 옵션으로 프로젝트 명을 지정합니다.

- 예) `csbuid new --toolchain=gcc5.4 --project=PRJ`

> ! `--toolchain` 옵션은 필수 입력 사항 입니다.

capture: 빌드 명령어 추출을 수행하여 RTV 프로젝트를 생성합니다.

- 예) `csbuild capture make`

convert: STATIC 프로젝트를 RTV 프로젝트로 변환합니다.

- 예) `csbuild convert`

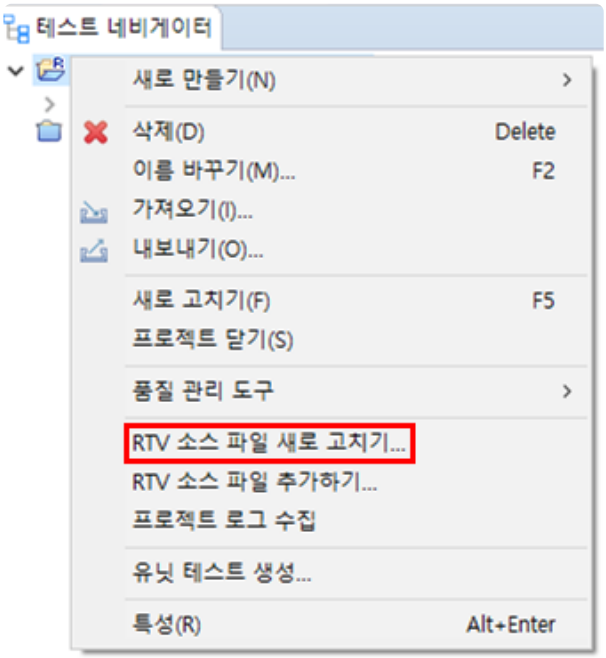
24.4. RTV 소스 파일 새로 고치기/추가하기

RTV 소스 파일 새로 고치기

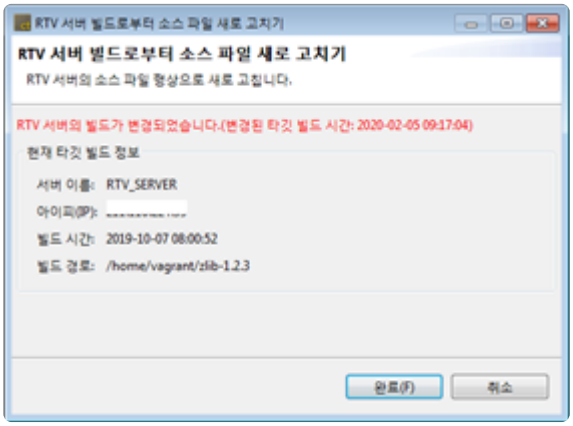
RTV 서버에서 소스 파일이 수정되고 빌드된 경우, [RTV 소스 파일 새로 고치기] 기능을 통해, 수정된 내용을 RTV 프로젝트에 반영할 수 있습니다.

> ! 소스 파일이 수정되었지만, 다시 빌드하지 않은 경우에는 변경된 내용이 반영되지 않습니다.
> RTV 서버의 소스 파일을 수정 후, 빌드하는 경우에는 풀 빌드를 수행해야 합니다.

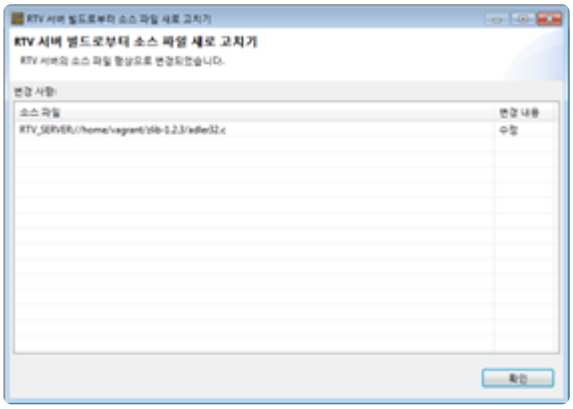
1. [테스트 네비게이터] 뷰의 [RTV 소스 파일 새로 고치기...] 컨텍스트 메뉴를 클릭합니다.



2. RTV 프로젝트를 생성할 때 선택한 소스 파일에 대한 최신 빌드 정보를 확인할 수 있습니다(빌드가 변경된 경우에 변경 알림 메시지 표시). [완료] 버튼을 클릭하여 변경사항을 확인합니다.



3. 변경사항을 확인한 뒤에 [확인] 버튼을 클릭합니다.

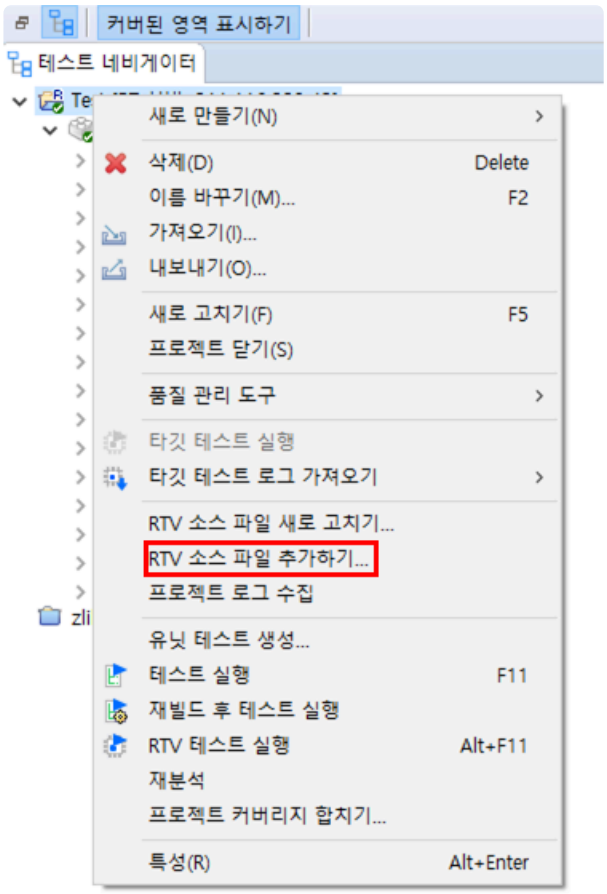


소스 파일 추가하기

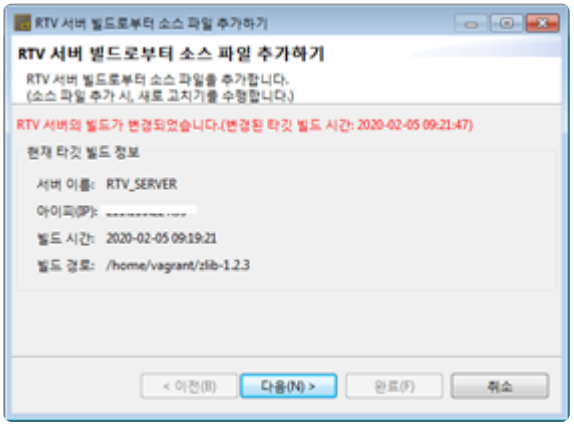
RTV 서버에서 새로운 소스 파일이 추가되고 빌드된 경우 또는 RTV 프로젝트를 생성할 때 일부 소스 파일만 선택한 경우에 [RTV 소스 파일 추가하기] 기능을 통해, 소스 파일을 RTV 프로젝트에 추가할 수 있습니다.

> ! 소스 파일이 추가되었지만, 다시 빌드하지 않은 경우에는 변경된 내용이 반영되지 않습니다.
> RTV 서버의 소스 파일을 추가 후, 빌드하는 경우에는 풀 빌드를 수행해야 합니다.

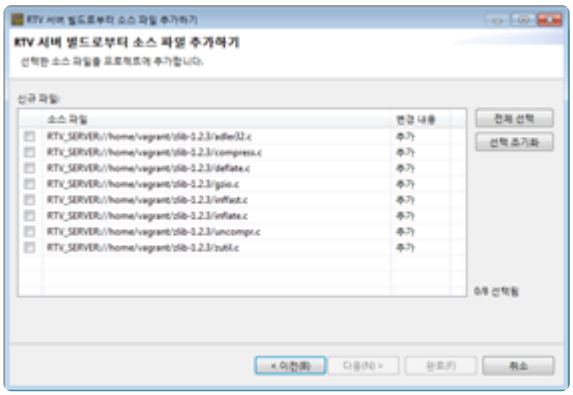
1. [테스트 네비게이터] 뷰의 [RTV 소스 파일 추가하기...] 컨텍스트 메뉴를 클릭합니다.



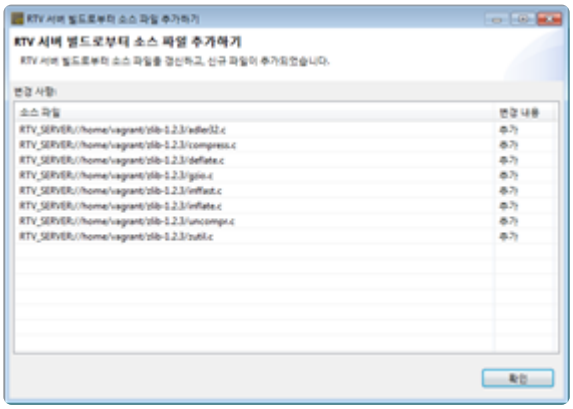
2. RTV 프로젝트를 생성할 때 선택한 소스 파일에 대한 최신 빌드 정보를 확인할 수 있습니다(빌드가 변경된 경우에 변경 알림 메시지 표시). [다음] 버튼을 클릭하여 신규 파일 목록을 확인합니다.



3. 신규 파일 목록에서 새로 추가할 파일을 선택한 뒤에 [완료] 버튼을 클릭합니다.



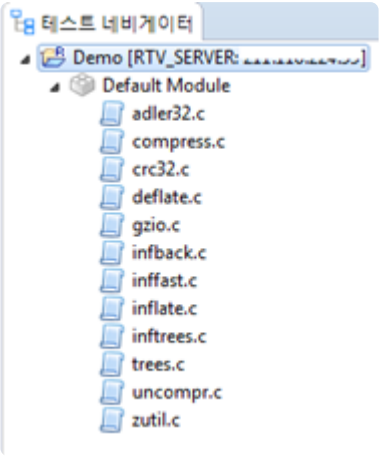
4. 변경사항을 확인한 뒤에 [확인] 버튼을 클릭합니다.



24.5. RTV 프로젝트 정보

테스트 네비게이터

1. RTV 프로젝트의 아이콘은 으로 표시됩니다.
2. RTV 프로젝트의 이름 뒤에 RTV 서버의 정보([서버명: IP])가 표시됩니다.

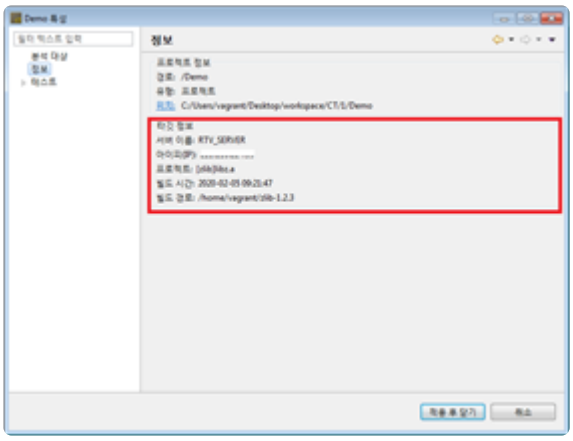


3. RTV 프로젝트의 소스 파일은 수정할 수 없습니다(RTV 프로젝트의 소스 파일 편집기는 읽기 전용).

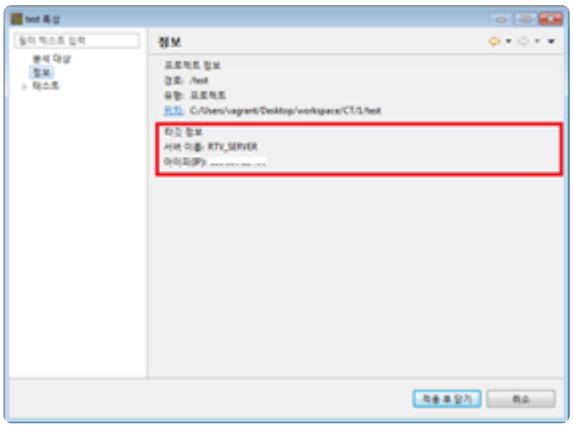
프로젝트 특성

정보 페이지

1. RTV 프로젝트는 RTV 서버와 프로젝트 정보가 표시됩니다.



2. 일반 프로젝트는 툴체인을 RTV 툴체인으로 설정한 경우, RTV 툴체인을 가져온 RTV 서버의 정보가 표시됩니다.



모듈/파일 특성

빌드 페이지

1. RTV 프로젝트는 프로젝트를 생성할 때 설정한 RTV 툴체인을 변경할 수 없습니다.
2. 단, 최초 설정된 RTV 툴체인이 삭제된 경우에는 다른 툴체인으로 변경할 수 있습니다.
3. 일반 프로젝트는 툴체인을 RTV 툴체인으로 변경할 수 있습니다. RTV 툴체인으로 변경하면 프로젝트 하위 모든 툴체인 설정(모듈과 소스 파일)이 변경되고, RTV 테스트를 실행할 수 있게 됩니다.
4. RTV 툴체인이 설정된 프로젝트는 소스 파일의 컴파일러 설정을 변경할 수 없습니다(모듈의 링커 설정은 변경 가능).

24.6. RTV 테스트 실행

RTV 환경에서 테스트를 실행합니다.
RTV 프로젝트는 별도의 설정 없이 RTV 테스트를 실행할 수 있습니다. 일반 프로젝트를 RTV 환경에서 테스트하려면 해당 프로젝트의 툴체인을 테스트 대상 RTV 서버로부터 생성한 RTV 툴체인으로 설정해야 합니다.

전체 RTV 테스트 실행

[유닛 테스트] 뷰와 [통합 테스트] 뷰에서 선택한 테스트를 실행합니다.

RTV 테스트는 다음과 같은 방법으로 실행할 수 있습니다.

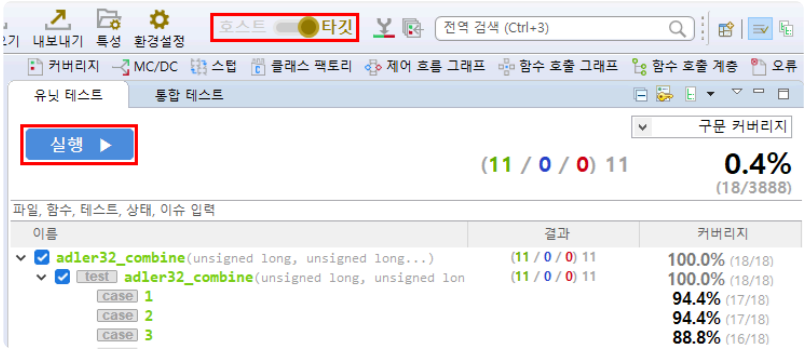
1. [테스트 네비게이터] 뷰의 [RTV 테스트 실행] 컨텍스트 메뉴

RTV 유닛 테스트 실행

[유닛 테스트] 뷰에서 선택한 테스트를 실행합니다.

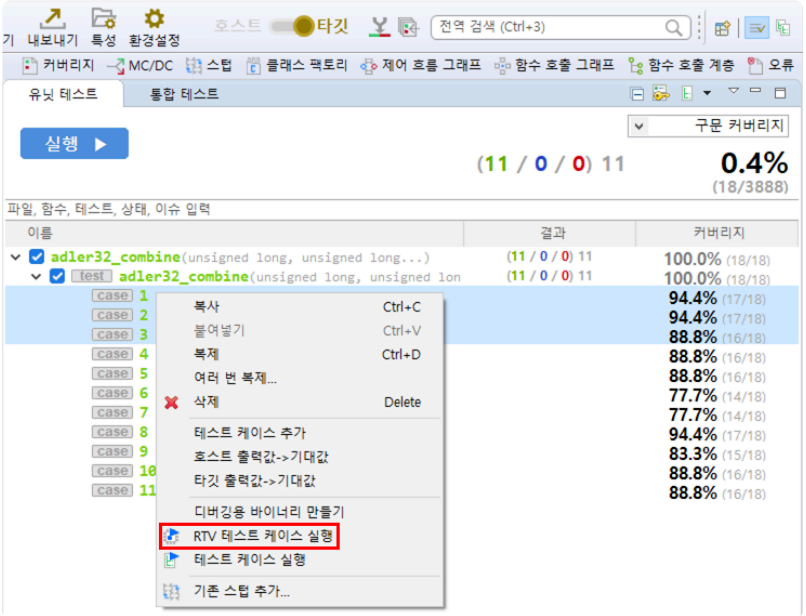
테스트 실행

[유닛 테스트] 뷰의 [실행] 버튼



테스트 케이스 실행

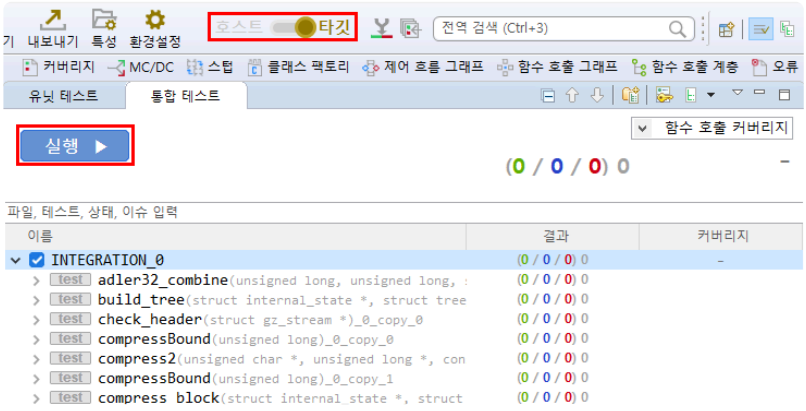
[유닛 테스트] 뷰의 [RTV 테스트 케이스 실행] 컨텍스트 메뉴(다중 선택 가능)



RTV 통합 테스트 실행

[통합 테스트] 뷰에서 선택한 테스트를 실행합니다.

[통합 테스트] 뷰의 [실행] 버튼



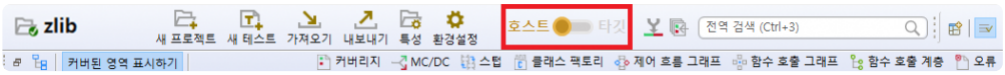
24.7. RTV 테스트 결과 확인

커버리지

최상위 툴바 메뉴의 커버리지 보기 메뉴를 통해 각 뷰와 편집기에서 표시해주는 커버리지 정보(호스트, 타깃, 호스트/타깃 병합)를 변경할 수 있습니다.

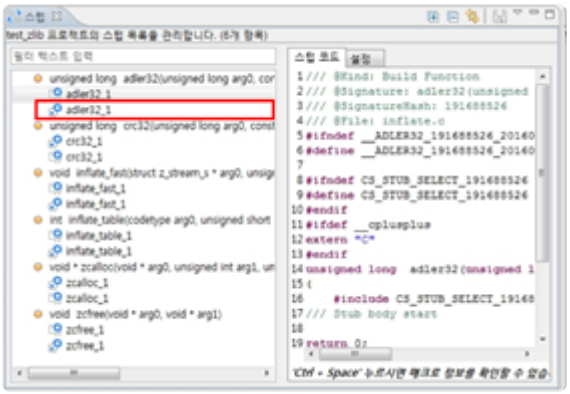
테스트 결과

대시보드의 호스트/타깃 설정 메뉴를 통해 테스트 결과를 변경할 수 있습니다.



스텁

RTV 테스트 실행 중에 생성되는 빌드 스텁은 호스트 테스트 실행 중에 생성되는 빌드 스텁과 구분됩니다.



타입	설명
타깃 빌드 스텁	RTV 실행 중 자동 생성된 스텁

25. EULA(End-User License Agreement)

소프트웨어 최종 사용자 라이선스 계약

중요: 본 최종 사용자 라이선스 계약("EULA")은 사용자(개인 또는 단체)와 본 EULA에 부속된 제품 인식 카드 혹은 인식 라벨로 식별되는 소프트웨어 제품("소프트웨어 제품" 또는 "소프트웨어") 제조업체인 슈어소프트테크 주식회사("슈어소프트테크") 사이의 법적 계약입니다. 소프트웨어 제품에는 컴퓨터 소프트웨어, 관련 미디어, 관련 인쇄물 및 "온라인" 또는 전자 설명서가 포함되어 있습니다. 소프트웨어 제품을 설치, 복사, 다운로드, 백업, 액세스하거나 사용하는 경우 본 EULA의 사용 조건에 동의하여야 하며, 본 EULA의 사용 조건에 동의하지 않으면 소프트웨어 제품에 대한 라이선스를 부여 받을 수 없습니다. 소프트웨어 제품에 대한 라이선스를 부여 받지 않는 경우, 본 소프트웨어 제품을 설치, 복사, 다운로드, 백업, 액세스하거나 사용할 수 없습니다.

소프트웨어 제품 라이선스

소프트웨어 제품은 지적 재산 보호 관련법뿐만 아니라, 저작권, 프로그램저작권 및 국제 저작권 규정에 따라 보호됩니다. 본 계약에서 사용되는 "컴퓨터"라는 용어는 단일 컴퓨터 시스템을 의미하며, 단일 컴퓨터 시스템의 범위에 대해서는 아래에 상술되어 있습니다.

1. 라이선스 부여.

본 EULA는 사용자에게 아래 권한을 부여합니다.

소프트웨어 설치 및 사용. 본 EULA에 명백히 설명한 것을 제외하고, 컴퓨터에 소프트웨어 제품 복사본을 하나만 설치, 사용, 액세스, 실행 또는 상호작용("실행")할 수 있습니다. 두 대 이상의 컴퓨터에서 소프트웨어 제품을 설치, 사용 또는 실행하기 위해서는, 각 컴퓨터 상에서 사용 또는 실행되는 소프트웨어의 상호작용 여부와 무관하게 컴퓨터의 대수에 해당되는 적법한 라이선스를(들을) 사용자가 부여 받아야 합니다.
복사본 백업. 컴퓨터에 포함된 소프트웨어 제품 백업 본이 없는 경우, 사용자가 소프트웨어 제품의 컴퓨터 소프트웨어 부분에 해당하는 단일 백업 본을 작성할 수 있습니다. 기록의 목적으로 백업 복사본을 사용할 수 있습니다. 단일 백업 본을 작성하기 위해 백업 유틸리티를 사용할 수 있습니다. 본 EULA에 명백히 제공된 것을 제외하고, 소프트웨어 제품에 동봉된 설명서를 포함한 소프트웨어 제품의 복사본을 작성할 수 없습니다.

2. 기타 권리 및 제한사항 설명

컴퓨터 혹은 단일 컴퓨터 시스템. 소프트웨어 제품의 라이선스는 단일 컴퓨터 시스템에 적용됩니다. 단일 컴퓨터 시스템이 다수의 중앙처리장치(CPU)를 갖는 경우, 이 중앙처리장치들이 하나의 회로기판 상에 부착되어 있고 하나의 시스템 버스를 공유하는 대칭적 중앙처리 구조를 지니는 경우에 한하여 단일 컴퓨터 시스템으로 간주됩니다.

언어 버전 선택. 소프트웨어 제품이 한 개 이상의 언어 버전에 포함되어 있는 경우, 제공된 언어 버전 중 하나만 사용할 수 있도록 허가됩니다.
분해모방 제한사항. 디컴파일 및 분해. 소프트웨어 제품을 분해모방을 이용하여 디컴파일하거나 분해할 수 없습니

다.
구성요소 분리. 소프트웨어 제품은 단일 제품으로 허가되었습니다. 한 대 이상의 컴퓨터에서 사용하기 위해 구성요소를 분리할 수 없습니다.
임대. 소프트웨어 제품을 임대, 리스 혹은 대여할 수 없습니다.
소프트웨어 제품 권리 이전. 본 EULA에 따라 하드웨어의 판매 또는 권리 이전의 일부분으로 모든 권한을 영구적으로 권리 이전할 수 있으며, 수령인은 본 EULA의 사용 조건에 동의하는 것으로 간주됩니다.
만료. 사용자가 본 EULA의 조항과 조건을 위반한 경우, 다른 권한의 침해 없이, 슈어소프트테크는 본 EULA의 기간 및 조건에 따라 권한을 만료시킬 수 있습니다. 이런 경우, 이미 설치된 소프트웨어 제품을 포함한 모든 소프트웨어 제품의 복사본 및 구성 부품을 폐기시켜야 합니다.
등록 상표. 본 EULA는 소프트웨어 제품 제조업체, 컴퓨터 제조업체, 기타 관련 공급업체의 등록 상표 또는 서비스 상표에 관련된 권한을 부여하지 않습니다.
업그레이드. 본 소프트웨어 제품이 슈어소프트테크에 의해 업그레이드되는 경우, 본 EULA의 사용 조건은 업그레이드된 소프트웨어에 그대로 적용되며 한 대 이상의 컴퓨터에서 사용하기 위해 분리될 수 없습니다.

3. 저작권.

소프트웨어 제품의 모든 권리와 지적소유권(소프트웨어 제품에 내장된 모든 이미지, 사진, 애니메이션, 비디오, 오디오, 음악, 텍스트 및 애플릿을 제한 없이 포함), 동봉된 설명서 및 모든 소프트웨어 제품의 복사본은 소프트웨어 제품 제조업체나 공급업체의 소유입니다. 소프트웨어 제품에 내장되어 있지는 않지만 소프트웨어 제품의 사용을 통해 액세스할 수 있는 내용에 대한 모든 권리와 지적 소유권은 해당 회사의 소유이며, 본 EULA는 이와 같은 내용의 사용에 대한 권한을 부여하지 않습니다. 소프트웨어 제품에 설명서가 파일로 포함되어 있는 경우, 해당 설명서 파일의 복사본을 한 부 인쇄할 수 있습니다. 소프트웨어 제품에 동봉된 인쇄물은 복사할 수 없습니다. 본 EULA에서 명백히 부여하지 않은 모든 권한은 해당 소프트웨어 제품 제조업체와 공급업체의 소유입니다.

4. 이중 미디어 소프트웨어 제품.

한 개 이상의 미디어로 소프트웨어 제품을 전달받을 수 있습니다. 받은 미디어의 종류 또는 크기와 관계없이, 컴퓨터에 해당 미디어를 하나만 사용할 수 있습니다. 다른 컴퓨터에서 다른 미디어를 실행할 수 없습니다. 소프트웨어 제품의 영구적인 권리 이전(위의 설명대로)을 제외하고 다른 사용자에게 다른 미디어를 임대, 리스, 대여 또는 권리 이전할 수 없습니다.

5. 보증.

소프트웨어 제품의 품질에 대한 보증은 첨부된 보증서에 따르며, 별도의 보증서가 첨부되지 않은 경우 슈어소프트테크는 소프트웨어 제품을 구입일로부터 1년간 정상적으로 사용할 경우 소프트웨어 제조공정상의 결함이 발생하지 않음을 보증합니다. 이는 소프트웨어 제품이 오류를 갖지 않는 소프트웨어임을 의미하지 않으며, 사용자가 작업 도중 의도하지 않았던 결과를 초래할 가능성을 배제하지 않습니다. 별도의 보증계약에 의해 명백히 기술된 경우를 제외하고, 슈어소프트테크는 소프트웨어 제품의 부정확성, 오류 또는 손실에 대해 책임지지 않으며 이로 인하여 발생하는 손상 및 손해에 대해서도 책임지지 않습니다. 슈어소프트테크는 제3자가 제기한 배상 청구나 혹은 제3자를 위해 사용자가 제기한 배상 청구에 대해 책임지지 않습니다. 본 EULA에 언급된 보증제한 규정은 사용자가 상기의 가능성에 대해 슈어소프트테크 또는 슈어소프트테크의 공인 대리업체에 미리 통고한 경우에도 유효합니다.

6. 수출 제한사항.

소프트웨어 제품의 권리 이전은 대한민국 내에서만 가능하며, 타 국가로 수출될 수 없습니다.

7. JAVA 지원 참고.

소프트웨어 제품에는 JAVA로 작성된 프로그램에 대한 지원이 포함되어 있습니다. JAVA 기술은 오류를 허용하지 않으며, JAVA 기술 오류로 인하여 사망, 부상 또는 심각한 물리적, 환경적 손상을 유발할 수 있는 핵 설비, 항공기 탐색이나 통신 시스템, 항공 제어, 의약 또는 무기 시스템과 같은 오류 방지 성능이 필요한 위험한 환경에서 온라인 제어 장치로 설계되거나 제조 또는 사용, 판매되도록 제작되지 않았습니다.

주의사항