# Every Flavour Beans

"The time has come…to talk of many [technologies]." –Lewis Carroll('The Walrus and the Carpenter')
Development Tools. Web Frameworks. GNU/Linux. Nokia N800. Video Encoding.



## November 26, 2007

### Unit Testing C++ Programs using CppUnit in Eclipse IDE on Windows

Filed under: C++, Eclipse — tabrez @ 10:47 pm



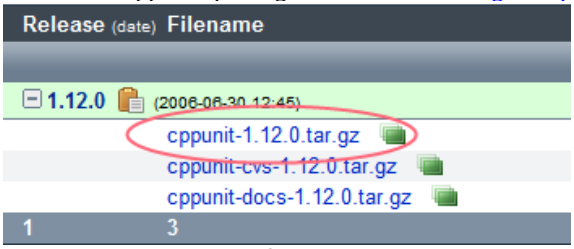
CppUnit is the most popular unit testing framework available for the C++ language today. So integrating it with one of the most popular IDEs available for the C++ language should be very appealing indeed. If you have not yet configured *CppUnit* library to work with the Eclipse IDE yet, here is a step-by-step procedure to do the same. By the end of it you will be able to create C++ classes and functions, write unit tests for them, run them and see the results, all from with-in the Eclipse IDE. This is for the Windows users; can be adapted for GNU/Linux users but there are shorter procedures for them.

You can either use *Eclipse Europa for C++* as the IDE or other IDEs based on it like EasyEclipse, *Wascana* etc. And of course MinGW or someother C++ toolchain must already be installed and configured with your Eclipse IDE; if not, read the MinGW and EasyEclipse configuration post for more information; it's for older version of the Eclipse and configuration is much simpler for Eclipse Europa.

**Downloading and Preparing the CppUnit Package**

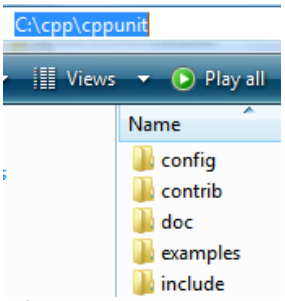
You can skip this section and the next one if you already have *CppUnit* library built/installed on your system and you know where its include and library files reside. Jump over to the last section in that case.

1. Download CppUnit package from its sourceforge.net page.




Extract it to a directory of your choice. I will assume that it is extracted to *'c:\cpp\cppunit'*. This directory should look like

this:




(If the downloaded archive file gets extracted to a directory named *cppunit-1.12.0* then just rename it to *cppunit*.)

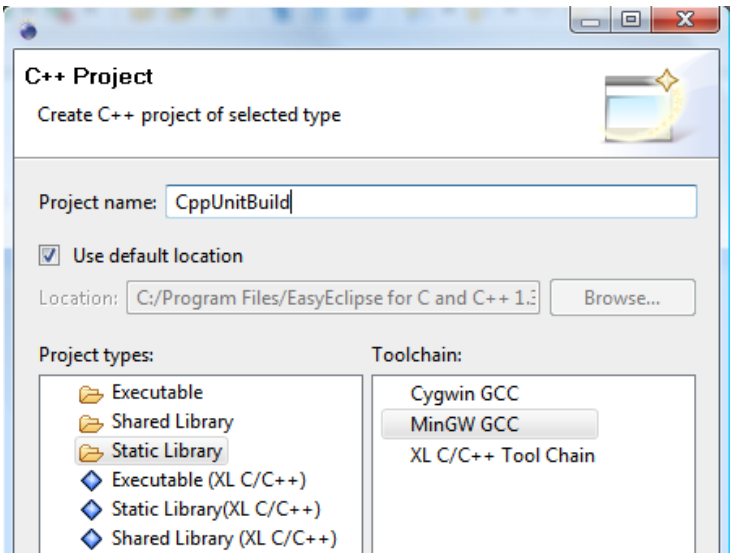
2. What we downloaded in the above step is the *CppUnit* source code. We need to build this source code to create a library file which we can then use in the C++ programs. *CppUnit* can be built from sources either from the command line or from the Eclipse IDE itself. To build it from the Eclipse IDE, we still need to generate at least one file from the MSYS command line. Go to your MSYS installation directory and click on *msys.bat* file in it. In the MSYS console window, change to the CppUnit directory(*c:\cpp\cppunit* in our example) and run the *./configure* command.

```
$ cd C:/cpp/cppunit

tabrez@TABREZ-VISTA /c/cpp/cppunit
$ pwd
/c/cpp/cppunit

tabrez@TABREZ-VISTA /c/cpp/cppunit
$ ./configure
```

This will create the *cppunit/config-auto.h* file that we need. Close the MSYS command window.

**Building CppUnit Package from the Source Code in Eclipse IDE**

1. Start the Eclipse IDE and create a new C++ project in it by going to *File -> New -> C++ Project*, enter a name(say, "CppUnitBuild") in the **Project Name** text box, select "Static Library" from the **Project Types** pane and "MinGW GCC" from the **Toolchain** pane.

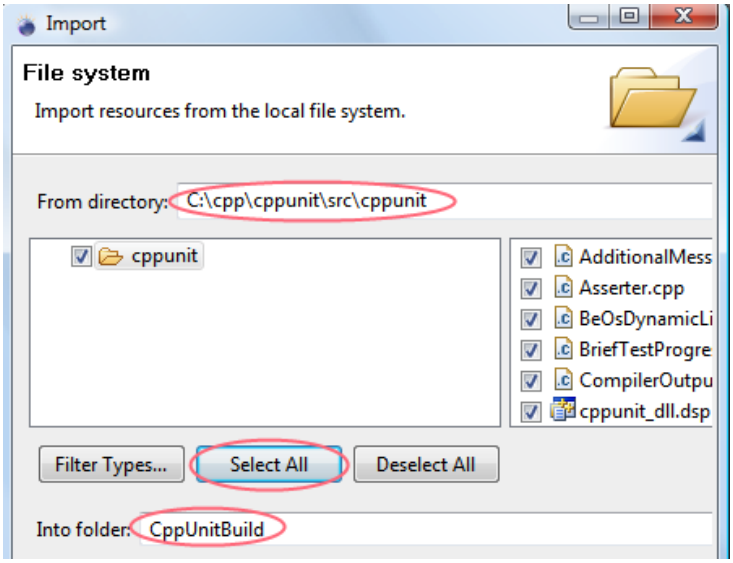



You can also choose to select "Shared Library" if you want to build *CppUnit* as a shared library. Similary, select "Cygwin GCC" if that is the toolchain you prefer.

Click **Finish** when done.

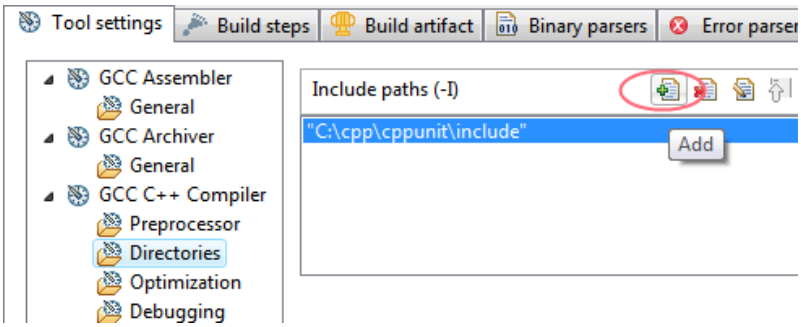
2. Now we need to import the entire CppUnit source code into this project. Right-click on the newly created project (CppUnitBuild) and select the **"Import"** menu item. In the **"Import"** dialog box, expand the **"General"** tab, select **"File**

**System"** and click the **Next** button. Click **"Browse"** and browse to the *c:\cppunit\src\cppunit* directory and click the **"Select All"** button and click **Finish**.




This will select and import all the source(*.cpp*) and header(*.h*) files of the CppUnit package into the *CppUnitBuild* project.

3. Next step is to add the CppUnit include directory to the compiler's include path. Right-click on the project(CppUnitBuild), select **"Properties"** and go to *C/C++ Build -> Settings node*. In the right pane, go to *Tool Settings -> GCC C++ Compiler -> Directories*. Click the **"Add"** button located near the **"Include Paths (-I)"** text box(see the screenshot below), click **"File System…"** button and browse to *c:\cpp\cppunit\include* directory.




4. Finally, build the project by pressing **Ctrl-B** or by right-clicking on the project and selecting **"Build Project."**
At the end of the build process, a static *CppUnit* library(*libCppUnitBuild.a*) should be built in the *Debug* subdirectory of the project directory in your Eclipse workspace(If you build using the *Release* configuration, the library will be generated in *Release* subdirectory instead). If you had opted for a shared library earlier, then [1] suggests that you define **CPPUNIT_DLL_BUILD** variable by going to *Project -> Properties -> C/C++ Build -> Settings -> GCC C++ Compiler -> Preprocessor -> Define Symbols (-D)*. You will see a file named *libCppUnitBuild.dll* generated in this case.

   Let us test it with a sample C++ project.

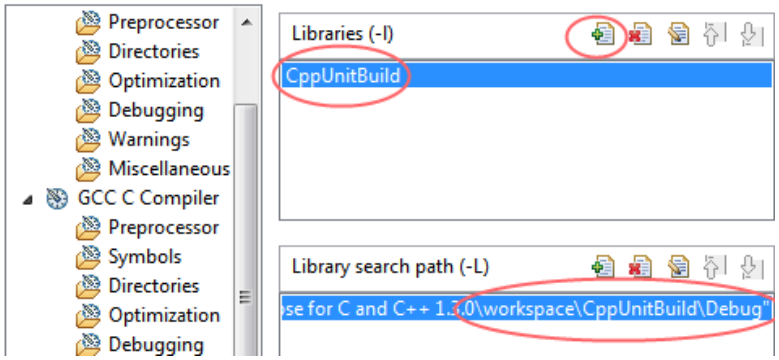
**Unit Testing C++ Programs using CppUnit in Eclipse IDE**

1. From the Eclipse IDE, create a new project by going to *File -> New -> C++ Project*, enter a name(say "CppUnitDemo") in the **Project Name** text field, select *Executable -> Empty Project* from the **Project Types** pane and "MinGW GCC" from the **Toolchain** pane and click **Finish**.

2. Now you can create a sample C++ class and write unit tests for it. A faster way would be to download this sample zip file that contains all the files required for testing. Download and extract it to some directory and then import all its contents to the *CppUnitDemo* project(right-click on project name and select *Import*, just like we did in **Step 2** of second section). You can also drag and drop these files on the project name in Eclipse.

3. Add CppUnit include directory to the project's include path just like we did in **Step 3** of second section. But remember to add the path to the *CppUnitDemo* project, not to the *CppUnitBuild* project! We also need to add the CppUnit library file that we have generated in the second part of this post to our *CppUnitDemo* project.

   Right-click on the project name, select *Properties*, select *C/C++ Build -> Settings* node, select the **Tools Settings** tab, select *MinGW C++ Linker -> Libraries* node, click the *Add* button near **"Libraries (-l)"** text box and enter *CppUnitBuild* in the popped-up dialog box. Similary, click the **Add** button near **"Library search path (-L)"** text box, click **"File System…"** button and browse to the path where the *CppUnit* library file was generated(*[your-eclipse-workspace]\CppUnitBuild\Debug*; The generated file might be in the *Release* sub-directory if you had chosen a *Release* build configuration to build the *CppUnitBuild* project earlier).




   If you had built CppUnit as a shared library earlier, then you need to define **CPPUNIT_DLL** at this stage; see **Step 6** for more details.

4. Right-click on the project name and select **"Build Project"** to build the project and right-click on the project name and select *Run As -> Local C/C++ Application*, select **'gdb debugger'** and click **OK** to run the project. You should see two dots in the output to represent that the two unit tests present in the sample files ran successfully. You can now create your C++ classes, write unit tests for them and then run the tests. Need help in getting started with writing unit tests using CppUnit? Here's a CppUnit cookbook for you and you can also download the CppUnit documentation.

Next up is how to integrate CppUnit Qt GUI test runner in Eclipse IDE. Then I will write about how to integrate *CxxTest* unit testing framework with the Eclipse IDE. *Doxygen* integration may follow, so hang tight ;)

[1] http://cppunit.sourceforge.net/cppunit-wiki/CppUnitWithEclipse - the sample zip file and other help for creating this tutorial were taken from this wiki page.



**Related Posts:**

- Wascana is Eclipse Based Standalone C++ IDE for MS Windows
- Build C++ Programs With SCons in Eclipse Using SConsBuilder Plugin(MS Windows)
- Java on Gentoo
- Installing C++ Boost on Microsoft Windows for Visual Studio .NET 2003/2005/Orcas
- Develop Ruby Applications Using Eclipse IDE
- C++ Development Environment on Windows using Eclipse Ganymede and Nuwen MinGW
- Develop Ruby Applications Using SciTE Editor

**Readers who viewed this page, also viewed:**

- Build C++ Programs With SCons in Eclipse Using SConsBuilder Plugin(MS Windows)
- Wascana is Eclipse Based Standalone C++ IDE for MS Windows
- C++ Development Environment on Windows using Eclipse Ganymede and Nuwen MinGW
- Six Popular IDEs For Developing Software in C/C++ on Windows Platform
- Setting Up C++ Development Environment on Windows with EasyEclipse and MinGW

## 9 Comments »

1. Thanks a lot, great article. Only question is how would one create a makefile project using cppunit instead of a executable?

   Quote

   *Comment by John — March 29, 2008 @ 10:27 pm*

2. Thanks for a great article. This article is very similar to the one one cppunit website but this has screenshots which were quite helpful. I just wish that someone get to the task of building cppunit integration for Eclipse. Visual Studio 2008 ships with its own Unit test feature built in and fully integrated.

   Quote

   *Comment by Dat Chu — April 21, 2008 @ 10:24 am*

3. This is really beneficial artical who dont know more bout Cpp unit..Thanx a lot such a gud artical

   Quote

   *Comment by divya — September 15, 2008 @ 5:17 pm*

4. Great article but the sample zip file link is broken.

   Quote

   *Comment by Jespr — October 29, 2008 @ 7:12 pm*

5. this is a horribly useless article, and has little to nothing to do with eclipse in reality. Creating a seperate project to do the unit testing in defeats the whole point of it.. This article only would have been worthwhile if it covered the usage / setup that integrated c/cpp unit into eclipse. otherwise theres nothing special that wouldn't be better described generically for the command line…

   Quote

   *Comment by joe — November 19, 2008 @ 3:58 am*

6. Thanks for the great article.
   I don't have MSYS and I don't find it on the net (maybe there is a more advanced replacement?). I don't run Vista but Windows 2000 (my other system, at home, is ubuntu Hardy :-> ). So I could not generate config-auto.h file and this probably is why the build step fails with 100 errors.
   It must be another way of getting config-auto.h…?
   Thanks so much.
   JK

   Quote

   *Comment by Joshua Klein — December 4, 2008 @ 2:59 pm*

7. @joe I don't understand what you are saying but note that you need to complete instructions in "Building CppUnit Package from the Source Code in Eclipse IDE" section only once, not for every C++ project you create in Eclipse.

   @Joshua http://www.mingw.org/wiki/msys :)

   Quote

   *Comment by tabrez — December 16, 2008 @ 9:29 pm*

8. I tried with the same. CppUnitBuild library is made successfully. But when I tried to link this library with one example itäs

giving error.

Like I have taken one example present in CPPUnitTese source.Have kept in some other path. And done the same as mentioned, like set include and library path as described. When I tried to build, it gives error , like present in CPPUnitBuild source code as shown below.

C:/CPP/cppunit/include/cppunit/portability/Stream.h: undefined reference to `(anonymous namespace)::allocator::~allocator()' C:/CPP/cppunit/include/cppunit/portability/Stream.h: undefined reference to `(anonymous namespace)::allocator::allocator()'

C:/Documents and Settings/satapal/workspace/CppUnitBuild/Debug/../Asserter.cpp undefined reference to `(anonymous namespace)::allocator::~allocator()'
C:/Documents and Settings/satapal/workspace/CppUnitBuild/Debug/../Asserter.cpp undefined reference to `(anonymous namespace)::allocator::allocator()'

Pleaze help me regarding this.

Quote

*Comment by Satarupa Pal — December 22, 2008 @ 2:59 pm*

9. JK - I was having the same problem!! I found that MSYS is part of the minGW suite, but is a separate install - http://www.mingw.org/wiki/MSYS just run the MSYS-1.0.10.exe and you will have it on your system - then follow the instructions in step 2. Very Good "How To", much better than the one on the CPPUnit site!!!

   Quote

   *Comment by Brad — January 26, 2009 @ 6:34 am*

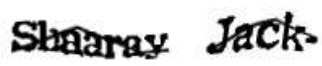
RSS feed for comments on this post. TrackBack URI

## Leave a comment

Name

Mail (will not be published)

Website



Type the two words:

Submit Comment

Notify me of follow up comments via e-mail

Subscribe without commenting
E-Mail: Subscribe

Web This site
Google Search

458 listeners
BY FEEDBURNER

Subscribe

Subscribe through email:

Subscribe



**More from BlogHer**
SXSW: Blackboards or Backchannels: The Techno-Induced Classroom of Tomorrow
Show Related Posts with LinkWithin

- An RDFa extension for

Dreamweaver
Midlife apocalypse:
Oprah's plugging
Facebook

**More from iVillage**

- Test your nutrition IQ

- Categories:
  - C++ (7)
  - C++ Boost (13)
  - Chumby (2)
  - Eclipse (7)
  - General (55)
  - Gentoo (14)
  - GNU/Linux (57)
  - Groovy/Grails (4)
  - Identica (3)
  - Mono (5)
  - N800 (16)
  - Netbeans (2)
  - Ruby/Rails (13)
  - Struts 2 (5)
  - Ubuntu (14)
  - Video (11)
  - Web (29)
  - Wordpress (9)
- Recent Posts:
  - Installing Sun Java SE 6, Apache Maven 2 and Tomcat 5.5 on Ubuntu GNU/Linux
  - "Hello, World" Java Web Application using Java SE 6 + Tomcat 5.5 + Maven 2
  - “Hello, World” Web Application in Ruby on Rails using console
  - Setting Up Rails Development Environment on Fedora GNU/Linux
  - Setting Up Ruby on Rails Projects with Git and Github
  - Installing Sun Java SE 6, Maven 2 and Tomcat 5.5 on Fedora GNU/Linux
  - Installing amazon-ecs/hpricot RubyGem on Windows Operating System
- **My Blogroll**
  - artima developer
  - C-puter
  - Digital Ramble
  - distrowatch
  - Garry’s Bit Patterns
  - redemption in a blog
  - tux machines

---



---