

## CppUnit을 활용한 Unit Test 코드 작성법 (4page)

---

작성자 : 이병준 (bjlee@etri.re.kr)

```
#include <stdio.h>
#include <cppunit/extensions/HelperMacros.h>
#include <cppunit/extensions/TestFactoryRegistry.h>
#include <cppunit/ui/text/TestRunner.h>
#include <iostream>

#include "mysql.h" // 테스트해야 할 소스 코드의 헤더 파일

using namespace std;

// 테스트 클래스의 정의.
class CMySQLTest : public CppUnit::TestFixture {

    BJLEE::CMySQL* mysql;

    // 아래 매크로 코드는, 본 클래스가 테스트를 위해 작성된 코드이며,
    // 본 클래스 내의 test1, test2, test3 메소드가 각각 테스트 코드가
    // 들어 있는 코드라는 것을 명시하기 위한 코드입니다.
    CPPUNIT_TEST_SUITE( CMySQLTest );
    CPPUNIT_TEST( test1 );
    CPPUNIT_TEST( test2 );
    CPPUNIT_TEST( test3 );
    CPPUNIT_TEST_SUITE_END();

public :

    // 이 함수는 테스트 메소드들이 수행되기 전, 사전 준비작업을 하기 위한
    // 코드입니다.
    void setUp() {
        try {
            mysql = new BJLEE::CMySQL("129.254.173.94", 3306, "root", "");
        }
        catch ( BJLEE::CMySQLException e ) {
            cout << e << endl;
            CPPUNIT_ASSERT( false );
        }
    }
}
```

```

// 이 함수는 테스트 메소드들이 수행되고 난 후, 사후 정리작업을 하기 위해
// 구현하여야 하는 함수입니다.
void tearDown() {
    delete mysql;
}

// 테스트 메소드입니다. CPPUNIT_ASSERT등의 함수를 사용하여,
// 테스트 조건이 만족되었는지를 테스트 해 볼 수 있습니다.
void test1() {
    using namespace BJLEE;
    cout << endl;
    try {
        mysql->select_database("webnmsdb");
        mysql->query("select * from lspqosprofile");
        CMySQLResult result = mysql->get_result();
        unsigned int num_fields = result.get_num_fields();

        while ( result.next() ) {
            for ( unsigned int i = 0; i < num_fields; ++i ) {
                unsigned long field_length = result.get_length(i);
                char* buffer = new char[field_length + 1];
                result.get_field(buffer, field_length + 1, i);
                cout << buffer << " ";
            }
            cout << endl;
        }
    }
    catch ( CMySQLException e ) {
        cout << e << endl;
        CPPUNIT_ASSERT( false );
    }
}

// 테스트 함수입니다. 미구현 상태입니다.
void test2() {
}

// 테스트 함수입니다. 미구현 상태입니다.
void test3() {
}
};

```

```
// 테스트 수트에 테스트 클래스를 등록해주는 매크로 함수입니다.  
// 테스트 해야 할 클래스가 여러개인 경우, 아래의 매크로를  
// 계속적으로 사용해주면 됩니다.
```

```
CPPUNIT_TEST_SUITE_REGISTRATION( CMySQLTest );
```

```
// 아래의 코드는 거의 변경없이 그대로 사용하면 됩니다.
```

```
int main() {  
    CppUnit::TextUi::TestRunner runner;  
    CppUnit::TestFactoryRegistry &registry =  
        CppUnit::TestFactoryRegistry::getRegistry();  
  
    runner.addTest( registry.makeTest() );  
    bool result = runner.run();  
  
    return result;  
}
```

아래는 본 코드를 컴파일하는 데 사용된 Makefile입니다.

```
.SUFFIXES = .ec .cpp .c .o .y .l
```

```
CXX = g++
```

```
LD = g++
```

```
INCLUDEDIR = -I. -I./include -I/usr/include/mysql -I/usr/local/include
```

```
LIBDIR = -L. -L/usr/lib/mysql
```

```
CXXFLAGS = -O2 -Wall
```

```
YACCSRCS =
```

```
FLEXSRCS =
```

```
LIB_INSTALL=/home/bjlee/lib
```

```
HDR_INSTALL=/home/bjlee/include
```

```
HDRS = cmysql.h
```

```
SRCS = cmysql.cpp cmysqltest.cpp
```

```
OBJS = $(SRCS:.cpp=.o)
```

```
all : libjlee-mysql.a cmysqltest
```

```
libjlee-mysql.a : $(OBJS)
```

```
ar cr libjlee-mysql.a $(OBJS)
```

```
cmysqltest : $(OBJS)
```

```
$(CXX) $(CXXFLAGS) $(LIBDIR) -o $@ $(OBJS) -ljlee-mysql -lcppunit -ldl  
-lmysqlclient
```

```
%o : %.cpp
```

```
$(CXX) $(CXXFLAGS) $(INCLUDEDIR) -c $< -o $@
```

```
install:
```

```
Wcp -f *.a $(LIB_INSTALL)
```

```
Wcp -f $(HDRS) $(HDR_INSTALL)
```

```
clean :
```

```
Wrm -f *.o *.a
```