Unit Testing with Google Test

Dror Helper

@dhelper http://blog.drorhelper.com

Module Overview

Introducing GTest

- Writing unit tests with GTest
- Running unit tests with GTest

xUnit testing framework structure

- Arrange, Act, Assert
- Test setup and test teardown

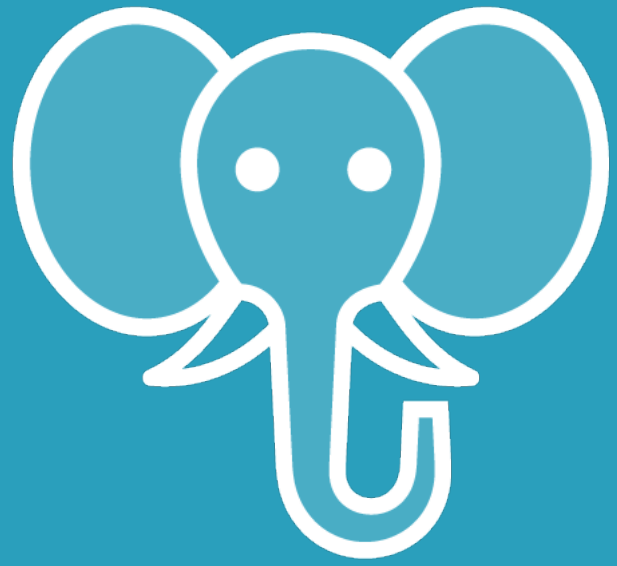
Why use GTest

- Unit testing frameworks comparison

Using Assertions

- Assert and Expect
- Testing for exceptions

Parameterized tests



Why GTest?

Google's C++ Test Framework

Or GTest for short,

Open source, widely used, multi platform, xUnit test framework and part of GMock distribution

Unit Testing with GTest

```
TEST(AddTwoNumbers)
{
    Calculator calc;

    int result = calc.Add(2, 3);

    ASSERT_EQ(5, result);
}
```

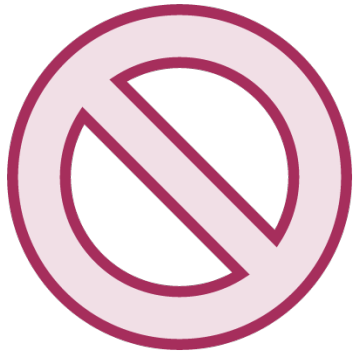
Test's Names Using GTest

Compile

Must be a valid method name

Explain

What is being tested



Ignore

use *DISABLED_* prefix

Running Tests

The tests are compiled into a console app (exe)

Command Line arguments can be used

--help for a full list of available flags

GTest & the Command Line

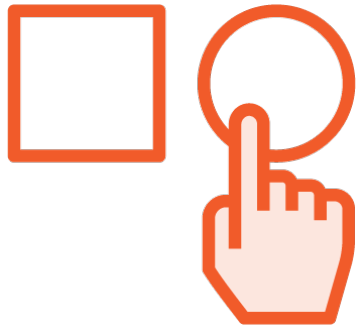
Test Selection

Test Output

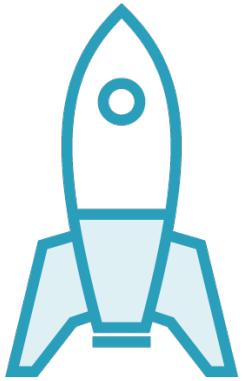
Test Execution

Assertion Behavior

GTest & the Command Line

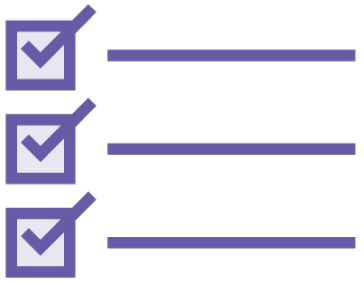


Test Selection

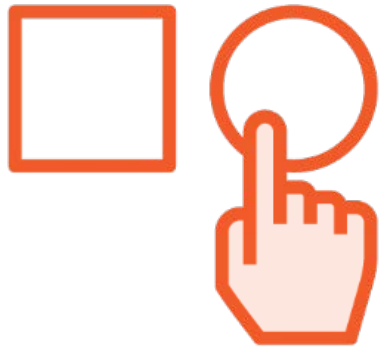


Test Execution

Test Output



Assertion Behavior



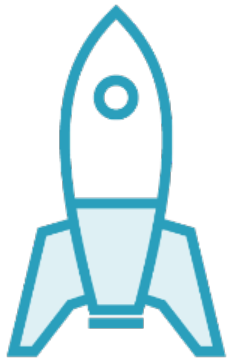
Test Selection

`--gtest_list_tests`

`--gtest_filter`

- Run only specified Tests
  - xyzTest
  - xyz*, *zTest, *yz*
  - xyzTes?
- Run all except specified test
- Separated by ‘:’
  - xyz.*:abc.*-xyz.old

`--gtest_also_run_disabled_tests`



Test Execution

```
--gtest_repeat=count
--gtest_shuffle
  --gtest_random_seed=number
```

Test Output

`--gtest_color`
- On/Off/Auto

`--gtest_print_time`
- Test execution time or use ‘0’ to disable

`--gtest_output=xml`
- JUnit compatible
- Can specify file/directory

Assertion Behavior

`--gtest_break_on_failure`

- For simple debugging on test fail
- Do not use in CI server

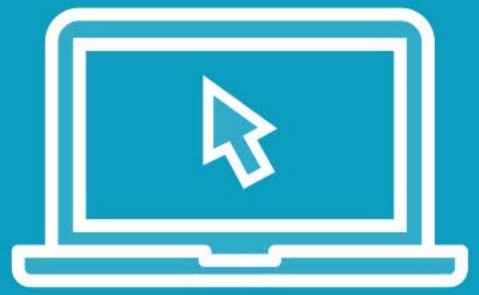
`--gtest_throw_on_failure`

- For servers with limited test support

`--gtest_catch_exceptions`

- Do not report exceptions as failures
- Allow them to crash the program

Demo



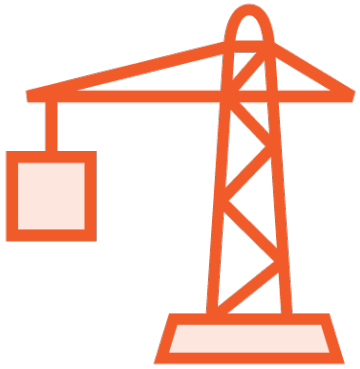
Writing tests using GTest

- Test names
- Disabling tests

Running Gtest from the command line

- List all test names
- Execute tests by test names
- Test result file

The Three Parts of a Unit Test



Arrange

Set-up preconditions and inputs

Act

Invoke the method under test

Assert

Verify test result

```
TEST(WhoWeAreTest) {
    Knight knight;

    ASSERT_EQ(“Ni!”, knight.Say());
}
```

```
 Expected: knight.Say()
      Which is: "ekki-ekki-ekki-pitang-zoom-boing!"
To be equal to: "Ni!"
```

Assert

Multiple Macros for specific checks

Comes at the end of the test

Ideally one per test

Assert and Expect

ASSERT

Fatal failures

Abort current function

EXPECT

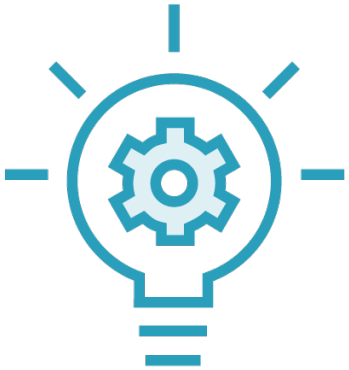
Nonfatal failures

Doesn't abort current function

Basic Assertions

```
ASSERT_TRUE(SomeCondition);
EXPECT_TRUE(SomeCondition);

ASSERT_EQ(2 + 2, 5); // 2 + 2 == 5 → Abort test
ASSERT_NE(2 + 2, 5); // 2 + 2 != 5 → Continue test

ASSERT_LT(val1, val2) // val1 < val2
```

Why Should You Care About Failure Messages?



Understand why the test failed

Reduce debugging time

It's the purpose of the test

String Assertions

```
ASSERT_STREQ(str1, str2);
EXPECT_STREQ(str1, str2);

ASSERT_STRNE(str1, str2);
EXPECT_STRNE(str1, str2);

ASSERT_STRCASEEQ(str1, str2);
EXPECT_STRCASENE(str1, str2);
```

Exception Assertions

```
ASSERT_THROW(statement, exception_type);
EXPECT_THROW(statement, exception_type);

ASSERT_ANY_THROW(statement);
EXPECT_ANY_THROW(statement);

ASSERT_NO_THROW(statement);
EXPECT_NO_THROW(statement);
```

Predicate Assertions

```
ASSERT_PRED1(IsEvenNumber, num);
EXPECT_PRED1(IsEvenNumber, num);

ASSERT_PRED2(MutuallyPrime, a, b);
EXPECT_PRED2(MutuallyPrime, a, b);

// Can improve output message using ASSERT_PRED_FORMAT
```

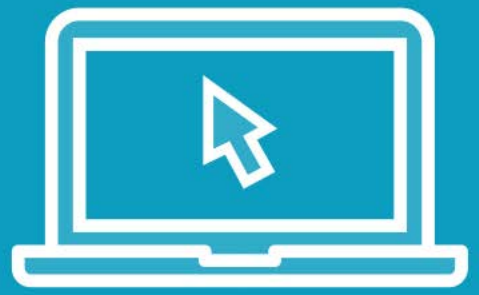
More Assertions

- Floating point
- HRESULT
- Type
- Death Tests

Custom Failure Message

```
ASSERT_EQ(x, y) << “x is not equal to y”;
```

Demo



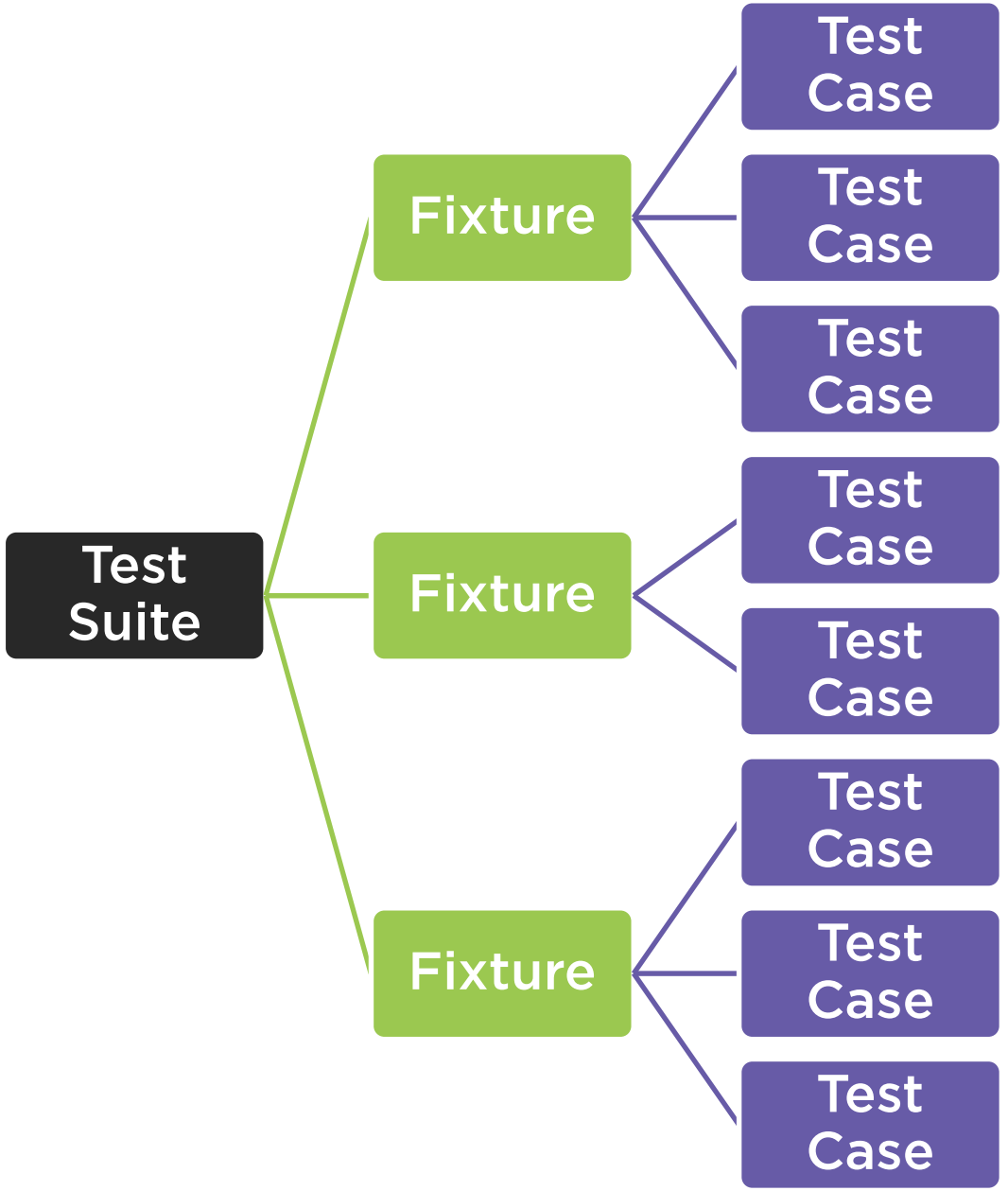
Using Assertions in Tests

- Assert vs. Expect
- Using the right assertion
- Failure messages

Test Suite == exe/dll

Fixture == class

Test Case == method



Creating Test Fixtures

```
class MyFixture : public ::testing::Test {
   virtual void SetUp()

    { // Common setup code }

    virtual void TearDown()

    { // Common teardown code }

};


TEST_F(MyFixture, ThisIsATest) {

    ...

}
```

Parameterized Tests

Avoid writing the same test body for different scenarios

- Value Parameterized Tests
- Type Parameterized Tests

Value Parametrized Tests

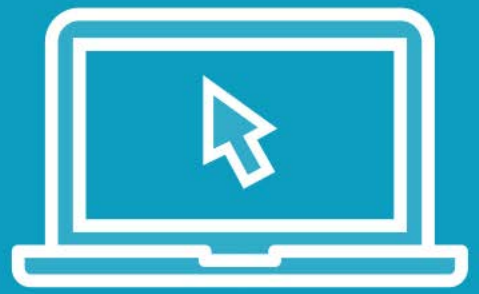
```
class MyFixt

};


TEST_P(MyFix
    Foo foo;
    EXPECT_T
}

INSTANTIATE_
    Instanti                                    4));
```

```
C:\WINDOWS\system32\cmd.exe
[==========] Running 4 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 4 tests from InstantiationName/MyFixture
[ RUN      ] InstantiationName/MyFixture.SomeTest/0
c:\projects\pluralsight\gmocksample\gmocksample.cpp(36): error: Value of: foo.Blah(GetParam())
  Actual: false
Expected: true
[  FAILED  ] InstantiationName/MyFixture.SomeTest/0, where GetParam() = 1 (1 ms)
[ RUN      ] InstantiationName/MyFixture.SomeTest/1
[       OK ] InstantiationName/MyFixture.SomeTest/1 (0 ms)
[ RUN      ] InstantiationName/MyFixture.SomeTest/2
c:\projects\pluralsight\gmocksample\gmocksample.cpp(36): error: Value of: foo.Blah(GetParam())
  Actual: false
Expected: true
[  FAILED  ] InstantiationName/MyFixture.SomeTest/2, where GetParam() = 3 (1 ms)
[ RUN      ] InstantiationName/MyFixture.SomeTest/3
[       OK ] InstantiationName/MyFixture.SomeTest/3 (0 ms)
[----------] 4 tests from InstantiationName/MyFixture (6 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 1 test case ran. (8 ms total)
[  PASSED  ] 2 tests.
[  FAILED  ] 2 tests, listed below:
[  FAILED  ] InstantiationName/MyFixture.SomeTest/0, where GetParam() = 1
[  FAILED  ] InstantiationName/MyFixture.SomeTest/2, where GetParam() = 3

 2 FAILED TESTS
Press any key to continue . . .
```

Parameter Generators

```
Range(begin, end[, step])
Values(v1, v2, ..., vN)
ValuesIn(container)
ValuesIn(begin, end)
Bool()
Combine(g1, g2, ..., gN)
```

Demo



Unit Testing Framework Comparison

- MSTest Native
- Catch

Unit Testing Comparison

Google Test	MSTest (Native)	Catch
Test Names are valid methods	Test Names are valid methods	Test Names are strings
Many Asserts + Customization	Few Assert Methods	One REQUIRE fits all
Error message by assertion	Error message by assertion	Detailed failure message
Build & Deploy	Bundled with Visual Studio	Single Header file
Multi platform	Windows only	Multi platform
Fixtures	Classes	Fixtures and Sections

Summary

Writing Unit Tests using GTest

- Test names
- Using Assertions
- Test Fixtures
- Parameterized Tests

Running GTest from the command line

Other C++ Unit Testing Frameworks

- MSTest
- Catch