

C++ Unit Testing Fundamentals Using Catch

INTRODUCING CATCH



Dror Helper

@dhelper blog.drorhelper.com



Who is this course for?

C++ Developers

- Learn about unit testing
- Add Catch to the development process
- Beginner level knowledge of C++

Not about TDD or advanced techniques

- But will cover unit testing best practices

Course Overview



Module 1: Introducing Catch

- What is Catch and unit testing
- Setting up Catch

Module 2: Organizing tests using Catch

- Naming tests and using Tags
- Using Catch from Command line

Module 3: Asserting results using Catch

- Using REQUIRE and ASSUME
- Checking for Exceptions
- Getting detailed information from tests

Module 4: Reducing duplicate test code

- Using test fixtures vs. Sections
- Writing BDD style tests

A “Unit Test” is:

A method (Code)

Tests specific functionality

Clear pass/fail criteria

Runs in Isolation

Simple Unit Test

```
TEST_CLASS(MyUnitTest)
{
    public:
        TEST_METHOD(TestMethod1)
        {
            // Your test code here
        }
}
```

Why Write Automated Tests?

Quick **Feedback**



Avoid **Stupid** Bugs



Immune to **Regression**



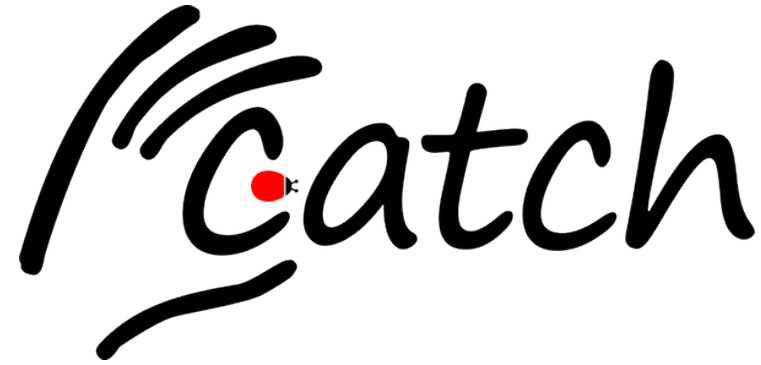
Change Your Code
Without Fear



In Code
Documentation



You're already
testing your code!

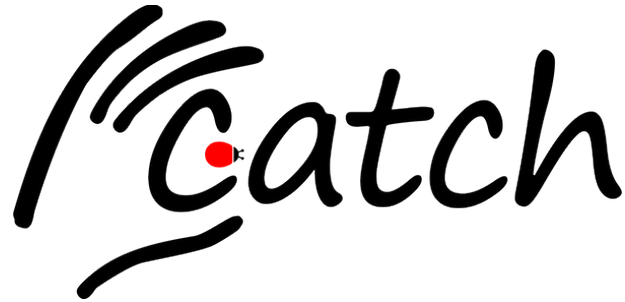


What is CATCH?

C++ Automated Cases in Headers

Open source (<https://github.com/philsquared/Catch>)

Why Use CATCH?



Single header deployment

No external dependencies

Tests names are free-form strings

Powerful “Assertions”

Excellent error messages

Sections!

Getting Started with Catch

Download catch.hpp

```
#define CATCH_CONFIG_MAIN
#include "catch.hpp"

TEST_CASE("This is a test name")
{
    . . . . .
}
```

Demo



How to get `catch.hpp`

Writing an empty test

Writing Tests Using Catch

```
TEST_CASE("This is a test name", "[Tag]")  
{  
    MyClass myClass;  
  
    REQUIRE(myClass.MeaningOfLife () == 42);  
}
```

T9 Predictive Text Algorithm

HELLO

1	2 ABC	3 DEF
4 GHI	5 JKL	6 MNO
7 PQRS	8 TUV	9 WXYZ
*	0	#

Unit testing T9 algorithm

- Used in older cell phones
- Input: a sequence of digits
- Output: suggested words

Examples:

- 843 → “the”
- 4663 → “good”
- 43556 → “hello”

Naïve implementation

Demo



Writing your first CATCH test

- Test structure
- Running CATCH
- Test failure

Writing Unit Tests using xUnit test frameworks

```
TEST_METHOD(PassDigitsForHelloReturnCorrectString) {
```

```
Words
```

```
Engine
```

```
Digits
```

```
auto
```

```
Assert::AreEqual(std::string("hello"), result[0]);
```

```
}
```

PassDigitsReturnOneString

Source: [unittest1.cpp line 26](#)

 Test Failed - PassDigitsReturnOneString

Message: Assert failed. Expected:<hello> Actual:<>

Elapsed time: 4 ms

Stack Trace:

[T9EngineTests::PassDigitsReturnOneString\(\)](#)

CATCH vs. Traditional xUnit testing frameworks

CATCH

Names are strings

One REQUIRE

Out of the box detailed failure
messages

Traditional

Names are valid method names

Several methods (Assert class)

Failure messages depends on assertion

Summary



Why use unit tests

How to set up CATCH

Writing your first CATCH test

Why use CATCH