C++ Unit Testing Fundamentals Using Catch2

INTRODUCING CATCH2

Dror Helper

@dhelper http://helpercode.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 technics

- But will cover unit testing best practices

Course Overview

Module 1: Introducing Catch^2

- What is Catch and unit testing
- Setting up Catch^2

Module 2: Organizing tests using Catch^2

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

Module 3: Asserting results using Catch^2

- Using REQUIRE and CHECK
- 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/catchorg/Catch2)

Why Use Catch^2 ?

- Single header deployment
- No external dependencies
- Tests names are free-form strings
- Powerful “Assertions”
- Excellent error messages
- Sections!

Getting started with Catch^2

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 Catch²

```
TEST_CASE(“This is a test name”, “[Tag]”)
{
    MyClass myClass;

    REQUIRE(myClass.MeaningOfLife () == 42);
}
```

T9 Predictive Text Algorithm

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 Catch² test

- Test structure
- Running Catch²
- Test failure

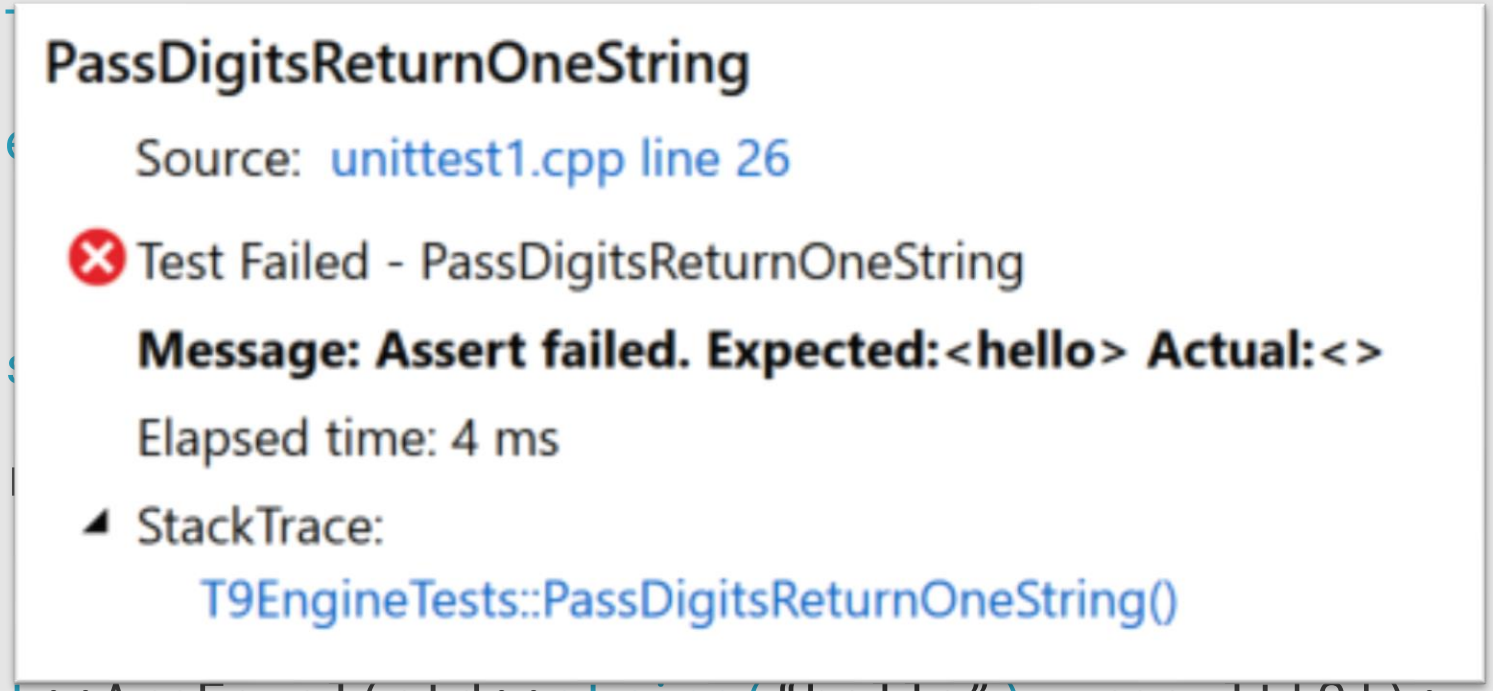
Writing Unit Tests using xUnit test frameworks

```
TEST_METHOD(PassDigitsForHelloReturnCorrectString) {
    Words
    Engine


    Digit

    auto


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



CATCH vs. Traditional xUnit testing frameworks

CATCH	Traditional
Names are strings	Names are valid method names
One REQUIRE	Several methods (Assert class)
Out of the box detailed failure messages	Failure messages depends on assertion

Summary

- Why use unit tests
- How to set up Catch^2
- Writing your first Catch^2 test
- Why use Catch^2