TEST DI APPROVAZIONE IN C++

- Yuri Valentini
- Modena
- 20 Maggio 2021
- Meetup C++ (online)

SOMMARIO

- Episodi precedenti
- ApprovalTests.cpp
- Concetti e differenze da Unit Test
- Refactoring kata

YURI VALENTINI

- Sviluppo Windows, Linux, iOS, Android
- C/C++, C#, Python, Dart/Flutter
- Videoconferenza e VOIP
- yuroller [at] gmail.com
- https://github.com/yuroller

PUNTATE PRECEDENTI

- Test del codice con Google Test (15/3/18)
- Mocking & Testing con Google Mock (19/4/18)
- Test basati su proprietà con rapidcheck (9/4/20)

UNIT TEST CON GOOGLE TEST

```
TEST(DeepThoughtTest, ChecksJobIsStarted) {
  // Arrange
  auto computer = DeepThought();

  // Act
  auto state = computer.calculateAnswer();

  // Assert
  EXPECT_EQ("processing", state);
}
```

BDD CON GOOGLE MOCK 1

```
TEST(JobRunnerTest, WaitsForComputationToFinsh) {
  // Create and configure mock
  auto computer = MockComputer();
  ON_CALL(computer, getAnswer())
    .WillByDefault(Return(42));
  EXPECT_CALL(computer, calculateAnswer())
    .WillOnce(Return("complete"));

  // Test JobRunner
  auto runner = JobRunner(computer);
  auto result = runner.waitForAnswer();
  EXPECT_EQ(42, result);
  // call to calculateAnswer() is checked
}
```

BDD CON GOOGLE MOCK 2

```
class Computer {
public:
  virtual ~Computer() {}
  virtual int getAnswer() const = 0;
  virtual std::string calculateAnswer() = 0;
};

class MockComputer : public Computer {
public:
  MOCK_METHOD(int, getAnswer, (), (const, override));
  MOCK_METHOD(std::string, calculateAnswer, (), (override));
};
```

TEST PROPRIETÀ CON RAPIDCHECK

```
void propertyTest() {
  rc::check("double reversal yields the original value",
    [](const std::vector<int> &l0) {
      auto l1 = l0;
      std::reverse(std::begin(l1), std::end(l1));
      std::reverse(std::begin(l1), std::end(l1));
      RC_ASSERT(l0 == l1); // return l0 == l1;
  });
}
```

DIAMOND KATA

```
"  A  "
" B B "
"C   C"
" B B "
"  A  "
```

```
std::vector<std::string> crea_rombo(char l);
```

DIAMOND - PROPRIETÀ

- Numero di righe dispari
- Prima e ultima riga contengono A
- Tutte le righe hanno un contorno simmetrico
- Prima lettera in pattern AB..l..BA
- Altezza == Larghezza
- 2 lettere uguali in ogni riga, non la prima e l'ultima
- 0,1,2.. spazi iniziali nella metà inferiore
- Simmetria orizzontale

TEST DI APPROVAZIONE

- ApprovalTests.cpp
- Clare Macrae al prossimo Italian C++
- Libreria header-only
- Framework di test (es. gtest, catch2)
- Strumento confronto testi (es. WinMerge)

ESEMPIO

```cpp
#include "ApprovalTests.hpp"
#include "catch2/catch.hpp"
using namespace ApprovalTests;
TEST_CASE("DeepToughAnswerFor6And7")
{
  // Arrange
  auto computer = DeepTough(6, 7);
  // Act
  auto answer = computer.getAnswer();
  // Print 'actual' value
  auto received = std::to_string(answer);
  // Verify received with approved ('expected') text
  Approvals::verify(received);
}
```

COSA SUCCEDE?

1. scrive 'received' in

   `[NomeFileCpp].[NomeTest].received.txt`

2. confronta con il testo approvato in

   `[NomeFileCpp].[NomeTest].approved.txt`

   - se differenti → strumento diff e test fallito
   - se uguali → test riuscito

MAIN.CPP

```
#define APPROVALS_CATCH
#include "ApprovalTests.hpp"
#include <memory>

using namespace ApprovalTests;
auto defaultReporterDisposer = // RAII
  Approvals::useAsDefaultReporter(
    std::make_shared<Windows::WinMergeReporter>());
```

DEMOTEST

- flusso di lavoro
- funzioni:

```
verify()
verifyAll()
verifyAllCombinations()
```

- stampa di oggetti
- regex scrubber

NOMENCLATURA

- Approval test
- Golden Master test
- Characterization test
- Snapshot test

Emily Bache post sui nomi

FORMATO TESTCASE

1. Arrange
2. Act
3. ~~Assert~~ Print/Compare

L'output approvato va in git/svn/...

TERMINOLOGIA

- ~~Actual~~ Received
- ~~Expected~~ Approved

CONFRONTO DI COSE

- supporto diretto per testo
- estensibile per
  - oggetti binari (es. protobuf)
  - immagini
  - suoni

Clare Macrae confronta immagini

SCRUBBER

- Utili per confronto di json o xml
- Tolgono dipendenza da formattazione
- Rendono il confronto deterministico

QUANDO USARE APPROVAL? 1/2

- Codice legacy (e funzionante)
- Codice difficile da capire
- Grandi refactoring evitando regressioni
- Codice che produce output testuale:
  - log/trace file
  - output a console

QUANDO USARE APPROVAL? 2/2

- Test nei loop esterni (es. integration test)
- In sostituzione di ASSERT multipli
- Passo intermedio per unit test tradizionale

REFACTORING KATA

- Esercizio di programmazione
- Spesso con codice legacy
- Migliorare il design
- Aggiungere funzionalità
- In modo sicuro

REFACTORING KATA

1. Estrazione di pezzo piccolo di codice
    - presumo pochi comportamenti
    - in genere più grande che per gli unit test
2. Guardare come è fatto
3. Scrivere test con input adatto
4. Approvare output
5. Apportare cambiamenti
6. Verificare stesso output

SUPERMARKET RECEIPT

- Stampa scontrino supermercato (testo)
- Codice legacy (pochi test)
- Prodotti con prezzi e quantità
- Preservare formato di stampa
- Aggiungere gestione offerte bundle

CODE COVERAGE

- OpenCppCoverage
- Quantità di codice esplorato dai test
- Indica dove porre l'attenzione
- Aiuta a trovare codice morto

DOMANDE?

GRAZIE

https://github.com/yuroller