

Quickly and Effectively Testing Legacy C++ Code with Approval Tests

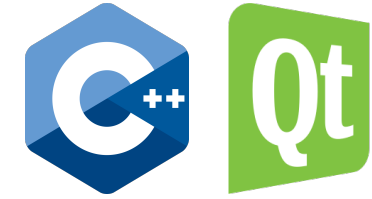
Clare Macrae (She/her)

clare@claremacrae.co.uk

15 July 2020

C++ on Sea

About Me



- **C++** and **Qt** developer since 1999
- **My mission: Sustainable and efficient testing and refactoring of legacy code**
 - Co-author of “**Approval Tests for C++**”
- **Consulting & training**
 - <https://claremacrae.co.uk>
- **All links from this talk via:**
 - github.com/claremacrae/talks

Goal

Share the Power of Approval Tests

Contents

- **Intro** 
- Getting Started
- Legacy Example
- Non-deterministic Output
- Customisability
- Visualising Differences
- Testing GUIs
- Summary



Llewellyn Falco

@LlewellynFalco

As part of expanding my c++ I was thinking of improving [#ApprovalTests](#) and making it work with GoogleTest. Anyone interested in pairing on that?

12:52 PM - 26 Nov 2017



Llewellyn Falco

@LlewellynFalco

As part of expanding my c++ I was thinking of improving [#ApprovalTests](#) and making it work with GoogleTest. Anyone interested in pairing on that?

12:52 PM - 26 Nov 2017



Clare Macrae

@ClareMacraeUK

Replying to [@LlewellynFalco](#)


Would be interested in hearing more.

2:46 PM - 26 Nov 2017

**“Approval Tests allow you to
verify a chunk of output (such as a file)
in one operation
as opposed to
writing test assertions for each element”**

Questions at the end...

Contents

- Intro
- **Getting Started** 
- Legacy Example
- Non-deterministic Output
- Customisability
- Visualising Differences
- Testing GUIs
- Summary

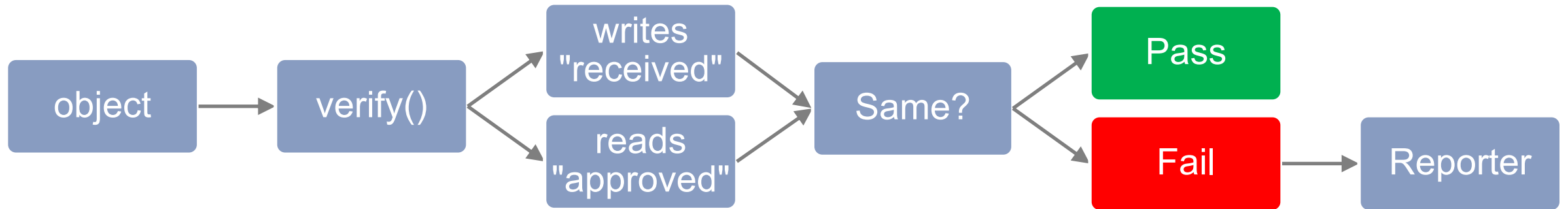
Demo 1:

Hello Approvals

Main Methods

- `Approvals::verify(std::string, Options)`
 - 2 other overloads take template types
- `Approvals::verifyAll()`
 - For use with containers
 - Writes one element at a time
- All are in namespace `ApprovalTests`

Stages of Approvals::verify()



Easy to:

- ... Add tests quickly
- ... Visualise differences
- ... Update output
- Convenience
 - Separates **test data** from **test code**
 - Ignore end-of-line differences

Easy setup

- Single header
 - <https://github.com/approvals/ApprovalTests.cpp/releases>
- Finds diff tool automatically
- Sensible filenames automatically
 - `source_directory/test_filename.test_name.approved.txt`
- “It just works”

Supported Test Frameworks


- Lots to choose from:



[Boost].UT / μ t

- Get to know chosen one well!

Contents

- Intro
- Getting Started
- **Legacy Example** 
- Non-deterministic Output
- Customisability
- Visualising Differences
- Testing GUIs
- Summary

Demo 2:

Legacy Code: Gilded Rose

Approvals and Legacy Code

- Before you start refactoring...
- Achieve good coverage
- `verifyAllCombinations()`
 - Not just for legacy code

Lots of String Conversion Options

- String Conversions

- Pass in a `std::string`
- Use `Approvals::verify(object, lambda)`
- Write custom `operator<<(std::ostream, ...)`
- Specialize `ApprovalTests::StringMaker::toString(...)`
- Use `TApprovals<YourStringConvertingClass>`

- How to Use the Fmt Library To Print Objects

- Use {fmt} library via `FmtApprovals::verify()`

String Design Guidelines


- Objects print their **relevant** data
- The data is consistent between runs
 - (no times, pointers, random)
- The data is easy to read, at a glance:

```
[0] = [x: 4 y: 50 width: 100 height: 61]  
[1] = [x: 50 y: 5200 width: 400 height: 62]  
[2] = [x: 60 y: 3 width: 7 height: 63]
```

```
(x,y,width,height) = (4,50,100,61)  
(x,y,width,height) = (50,5200,400,62)  
(x,y,width,height) = (60,3,7,63)
```

- Advice: [Tips for Designing Strings](#)

Contents

- Intro
- Getting Started
- Legacy Example
- **Non-deterministic Output** 
- Customisability
- Visualising Differences
- Testing GUIs
- Summary


Demo 3:

Log Files

Key Points

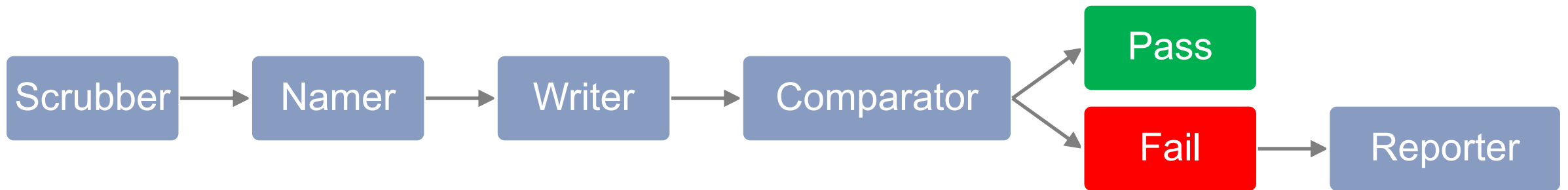
- `Approvals::verifyExistingFile()`
 - For when the file is already written
- Scrubbers for non-deterministic output
 - More scrubber options coming soon...

Contents

- Intro
- Getting Started
- Legacy Example
- Non-deterministic Output
- **Customisability** 
- Visualising Differences
- Testing GUIs
- Summary

Customisation Points Overview

Stages of Approvals::verify()



Everything is customisable

- We take great pride in the docs
- All examples generated from compiled, tested code

github.com/approvals/ApprovalTests.cpp/blob/master/doc/

approvaltestscpp.readthedocs.io/en/latest/

Customising Behaviour

- **Principles:** [Options](#) | [Disposable Objects](#)
- **Customisation points:** [Reporters](#) | [Comparators](#) | [Writers](#) | [Namers](#) | [Scrubbers](#) | [Configuring Approval Tests](#)
- **Summary:** [All Customizations of Approval Tests](#)

Customising Behaviour

- **Principles:** [Options](#) | [Disposable Objects](#)
- **Customisation points:** [Reporters](#) | [Comparators](#) | [Writers](#) | [Namers](#) | [Scrubbers](#) | [Configuring Approval Tests](#)
- **Summary:** [All Customizations of Approval Tests](#)

Contents

- Intro
- Getting Started
- Legacy Example
- Non-deterministic Output
- Customisability
- **Visualising Differences**
- Testing GUIs
- Summary




Demo 4:

SVG Files

Key Points

- Custom Reporters can help understand failures
- Make it easy to visualise differences
- No need to version control .png files

Contents

- Intro
- Getting Started
- Legacy Example
- Non-deterministic Output
- Customisability
- Visualising Differences
- **Testing GUIs** 
- Summary

Demo 5:

Testing GUIs

Key Points

- ApprovalTests.Cpp.Qt
 - Integrates Catch2 test framework with Qt event loop
 - Very early days!
- Multiple output files per test
 - Make filenames unique
- SeparateApprovedAndReceivedDirectoriesNamer
 - Worth being aware of
- AutoApproveReporter
 - **Caution!** Only if approved files already in **version control**

Annoyance

/Users/clare/demo/demos/05QtWidgets/approval_tests/WidgetsTest.It_approves_QImages.approved.png

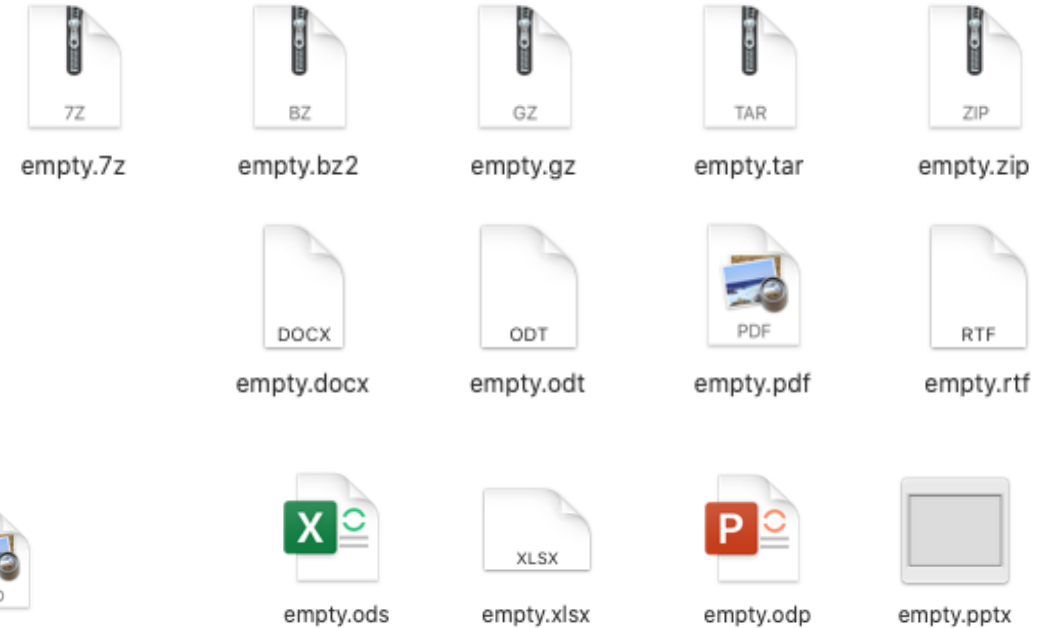
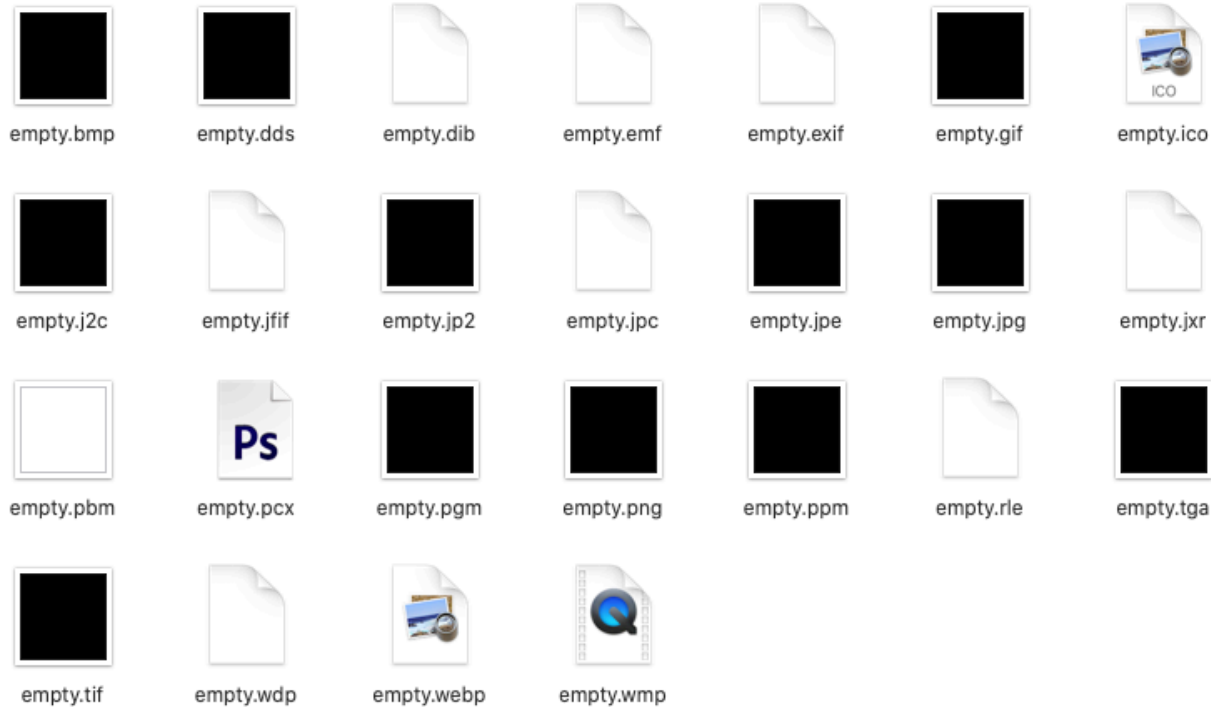


Unable to load the image file. It might not be one of the supported image formats.

Excellent Idea

- C# “Empty Files” project by Simon Cropp

- github.com/SimonCropp/EmptyFiles




GUI-testing

- What to test
 - Try to test smaller units of code, such as individual widgets
 - Test behaviors, not appearance (mainly)
 - Keep application logic separate from GUI code
- **ApprovalTests.Cpp.Qt** feedback welcome!
 - github.com/approvals/ApprovalTests.cpp.Qt

See Also ImageApprovals

- github.com/p-podsiadly/ImageApprovals
- Built on ApprovalTests.cpp
- Not yet released
- Recommended over ApprovalTests.cpp.Qt

Contents

- Intro
- Getting Started
- Legacy Example
- Non-deterministic Output
- Customisability
- Visualising Differences
- Testing GUIs
- **Summary** 

Summary

**“Approval Tests allow you to
verify a chunk of output (such as a file)
in one operation
as opposed to
writing test assertions for each element”**

Approval Tests for C++

- Easy to use
- Convenient
- Powerful
- Flexible
- It just works!

Testing Legacy Code effectively with Approval Tests

Clare Macrae & Llewellyn Falco

22nd July (Wednesday)



Quickly & Effectively Test Legacy C++ Code with Approval Tests

References

- All links from this talk, and more, via:
 - github.com/claremacrae/talks
- Sustainable and efficient testing and refactoring of legacy code
- Consulting & Training
 - claremacrae.co.uk
 - clare@claremacrae.co.uk

Any Questions?

Contents

- Intro
- Getting Started
- Legacy Example
- Non-deterministic Output
- Customisability
- Visualising Differences
- Testing GUIs
- Summary
- **Appendix: Challenges** 

Appendix: Common Challenges

External Resources

- I/O, Networking, System API calls...
- Be creative!
- Can you intercept and log calls?
 - And save them as “approved” results?
 - Then intercept again during later runs, and compare results?
- You can even use this data as inputs to later tests!
- Excellent example from Angie Jones:
 - <https://angiejones.tech/verifying-entire-api-responses/>

Embedded Systems

- Separate as much logic as possible
- Test the logic on desktop machines and CI
- Then if you get a failure on device, you've got less to test
- It may become possible to run Approval Tests from cross-built code

Output is OS-specific

- Include the OS in the file-names
- So verify() compares against output from same platform
- For example
 - TestQtDialog.loginScreen.on**MacOSX**.approved.png
 - TestQtDialog.loginScreen.on**Windows**.approved.png
 - TestQtDialog.loginScreen.on**Linux**.approved.png
- See [Multiple output files per test](#)

Approving is fiddly or takes too long

- Approving images?
- Approving hundreds of files?
- Try one of:
 - `AutoApproveIfMissingReporter`
 - `AutoApproveReporter` (and compare the diffs in version control!)