Creating Flexible Tests That Allow You to Refactor Your Code

Rusty Divine
SOFTWARE CONSULTANT

@CornerPosts cornerpostsoftware.com

Core Concepts

Keep tests ignorant of the details when possible

Test the unit of work

Minimize the use of mocks

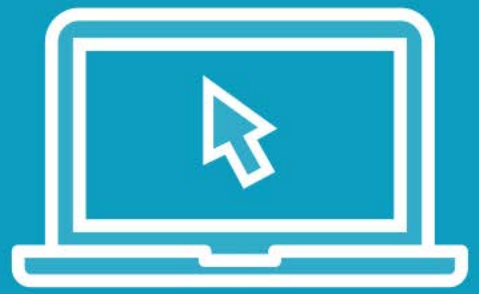
Brittle Test Types

Change detector

Private inspector

Business rule blinder

Demo



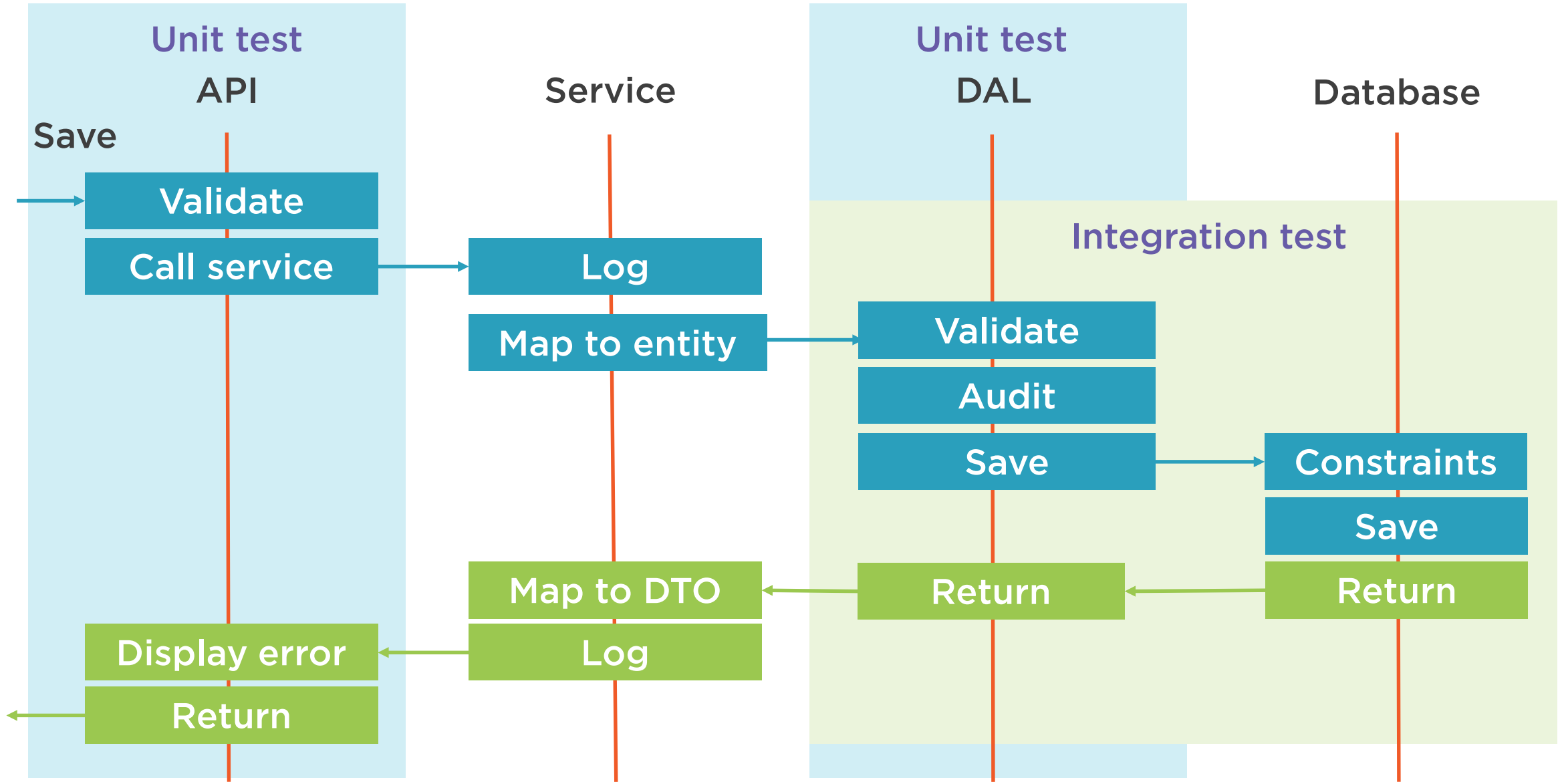
Tests that impede refactoring

Testing the Unit of Work

Unit of Work

Everything that happens from invoking a public method to it returning the results after it's finished; it's the work done along the path you see the debugger take through your code.

Unit of Work


Unit test
API
Service
Unit test
DAL
Database
Save
Validate
Call service
Log
Map to entity
Integration test
Validate
Audit
Save
Constraints
Save
Map to DTO
Return
Return
Display error
Log
Return

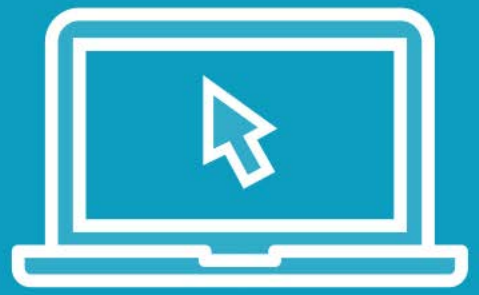
Scenario-driven Testing the API

How it's used

Happy & sad paths

Cross-class

Demo



Testing through the public API

Tips for Flexible Tests

- Maximum of one mock per test
- Fewer than 10% of tests with mocks
- Rarely directly test a private method
- Test by scenario not by method

Summary

Favor testing scenarios over methods

Keep your unit tests in the dark

Limit the use of mocks