Testing Spring

Beginner to Guru

Introduction to Testing

Why Write Tests?

- Writing tests takes a lot of time. Why do it?
  - Improves your software quality
  - Prove’s that your code is doing what you think it should be doing
  - End the fix one thing, break another!
  - Widely accepted as an industry best practice
  - Change existing (and working) code with confidence

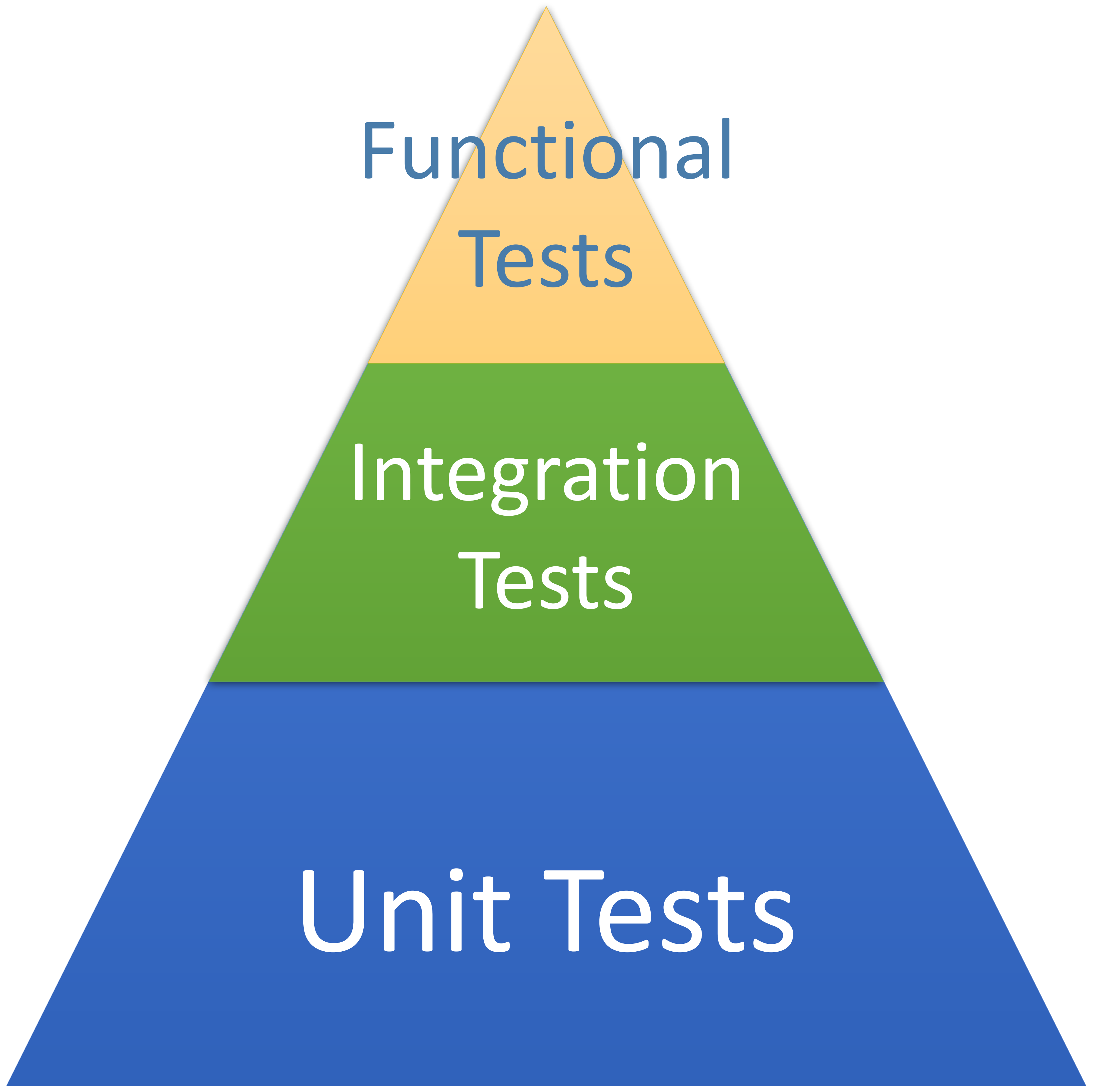
Testing Terminology

- **Code Under Test** - This is the code (or application) you are testing
- **Test Fixture** - "A test fixture is a fixed state of a set of objects used as a baseline for running tests. The purpose of a test fixture is to ensure that there is a well known and fixed environment in which tests are run so that results are repeatable." - JUnit Doc
  - Includes: input data, mock objects, loading database with known data, etc

Testing Terminology

- **Unit Tests / Unit Testing** - Code written to test code under test
  - Designed to test specific sections of code
  - Percentage of lines of code tested is code coverage
  - Ideal coverage is in the 70-80% range
  - Should be 'unity' and execute very fast
  - Should have no external dependencies
  - ie no database, no Spring context, etc

- **Integration Tests** - Designed to test behaviors between objects and parts of the overall system
  - Much larger scope
  - Can include the Spring Context, database, and message brokers
  - Will run much slower than unit tests
- **Functional Tests** - Typically means you are testing the running application
  - Application is live, likely deployed in a known environment
  - Functional touch points are tested - (i.e. Using a web driver, calling web services, sending / receiving messages, etc)



- All three types of tests play important roles for software quality
- The majority of tests should be **Unit Tests**
  - Small, fast, light weight tests
  - Very detailed and specific
- **Integration Tests** should be next largest category
- **Functional Tests** are smallest and least detailed of the categories.

Importance of Clean Code

- But my 2,000 line method tested just fine!
  - Impossible
- Quality code starts with - **QUALITY CODE!!!**
- Follow good coding practices:
  - S.O.L.I.D. OOP, Gang of Four Design Patterns
  - Clean Code - Robert "Uncle Bob" Martin
- Test coverage cannot overcome poor coding practices

Agile Testing Methods

- **TDD** - Test Driven Development
  - Write tests firsts, code to ‘fix’ tests, refactor code to cleanup, improve etc.
- **BDD** - Behavior Driven Development
  - Very similar to TDD
  - Describes the expected behavior of software
    - Often expressed as: when / then; given / when / then
- Which is better to use?
  - Use both!!

Testing Components

- **Mocks** - A fake implementation of a class used for testing
  - A test double for dependent objects - like a datasource
  - Can provide expected responses
  - Can verify expected interactions
- **Spy** - Like a mock, but real object is used
  - Mocks completely replace expected object
  - Spys are wrappers, but with real object inside

SPRING FRAMEWORK

GURU