CI/CD
Lecture 9-Testing

Omkarendra Tiwari

October 14, 2022

Testing

Why

- Proof of working code
- Ensure code keeps working
- Enable refactoring
- Documentation
- Release faster
- Feedbacks
- Code Coverage

Testing

Which one describes best

1. Process of demonstrating the absence of errors
2. To show that a program performs its functions correctly
3. A process of executing a program with intent to find errors

Testing

Psychology at play

- Human being tends to be highly goal oriented
- If the goal is to demonstrate that program is error free then...
- Same is applicable for otherwise

Another look

1. Process of demonstrating the absence of errors
2. To show that a program performs its functions correctly
3. A process of executing a program with intent to find errors

Testing

Economical Aspect

- White Box Testing
- Black Box Testing
  - Test on exhaustive set of valid inputs
  - Test on exhaustive set of invalid inputs
  - Example: Testing a compiler would require
    - Virtually infinite correct programs
    - Infinite incorrect programs

Testing

A few Principles/Guidelines

- Definition of the output is a necessary part of the test case
- A programmer/organization should avoid testing their own code
- Test case must be written for invalid/unexpected inputs as well as valid/expected inputs
- Test the program for if it does not do what it is supposed to do as well as it does what it is not supposed to do

Testing Stages

- Unit Testing
- Integration Testing
- Functional/Acceptance/End-to-End Testing
- Performance Testing
- Smoke Testing

Unit Testing

Unit Testing

What

- Objective is to test the functionality of a piece of code

How

- For a fixed and verified Input, and fixed Output
- If Output is not known; Don't attempt to Unit Test
- Find a fixed Input and know the fixed Output for it
- For example, clustering problem.

Unit Testing: Best Practices

Test Distribution

- Each Test tests exactly on thing
- One model-class may be tested by multiple test-classes
- Tests must not be dependent on other tests

Unit Testing: Best Practices

Example: JUnit

- In JUnit it is one test-method
- One assert per test-method
- Write multiple tests rather than multiple asserts per test-method
- In multiple assert, if first fails, other asserts can't provide the information

Unit Testing: Best Practices

Speed and Time

- Keep individual tests runtime in seconds
- Run quick tests first; fail early
- Long tests should be towards the end

Unit Testing: Best Practices

Bugs and Feedbacks

- Tests should be focused
- For each bug exactly one test should fail
- If a test passed; Keep it quiet; Close to none output
- If a test failed; Output all hints/help that will make bug finding quick

Unit Tests

Flakiness

- Rotate values for inputs such as 3, 8, 99, -11
- Avoid Input sources that are not-controlled, such as Network, File System (sometimes), system clock, gravity
- Multi-threaded, Math.random()
- Different OS
- Floating point round off
- Integer width
- Path separator

Unit Testing

Write a failing Test, when you

- A bug is found
- Notified for the presence of a bug
- Going refactor

Integration Testing

- Test multiple modules work together
- Multiple modules may comprise of whole system or a sub-system
- Its costly as multiple modules need to be up together
- Examples:
  - Test if Database connection is establishing
  - Payment Services integrated with Order Placement modules

Acceptance/Functional Test

Testing from user's perspective

- Whole application need to be up and running
- Test could be as simple as logging window up
- Or, if a particular button is clicked a sequence of actions are carried out

Smoke Test

- A quick test for verifying correctness of a functionality
- Can be used as a preliminary test for big events
  - Heavy and expensive tests
  - Gaining confidence for newly deployed applications

Code Coverage

Code Coverage

What

- A metric for assessing part of code covered under Tests

Kinds

- Function coverage: Percent of defined functions called
- Statement coverage: Statements executed
- Branches coverage: Branches executed
- Condition coverage: Boolean sub-expression tested for *true*/*false*
- Line coverage : Lines tested

TDD

Red-Green Refactor

- Write a test that fails; RED
- Write bare minimal code to make test pass
- Test again; GREEN

Further Readings

Highly Recommended

- https://www.atlassian.com/continuous-delivery/software-testing
- https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing
- https://martinfowler.com/bliki/UnitTest.html
- https://continuousdelivery.com/foundations/test-automation/
- https://martinfowler.com/bliki/IntegrationTest.html
- https://martinfowler.com/articles/practical-test-pyramid.html

Valuable Resources

- www.jamesshore.com/v2/books/aoad1/test_driven_development
- http://www.jamesshore.com/
- http://www.jamesshore.com/v2/blog/2006/quality-with-a-name
- https://www.youtube.com/watch?v=nlGSDUuK7C4