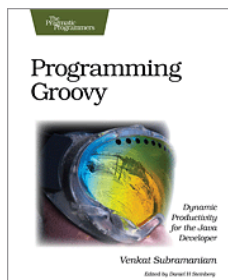


Testing With Dependencies

Venkat Subramaniam
venkats@AgileDeveloper.com

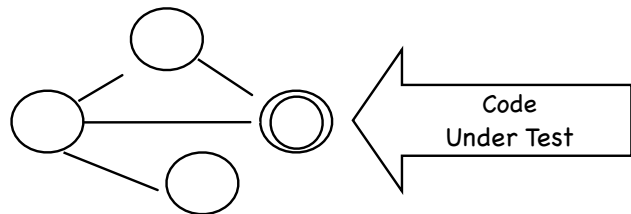


Unit Testing

- Unit Testing has significant benefits
 - Serves as a design tool
 - Helps to make code more robust
 - Serves as a documentation
 - Safety net while refactoring
 - Helps quickly identify problems
 - Gives feedback when code begins to break
 - ...

UT is easy when...

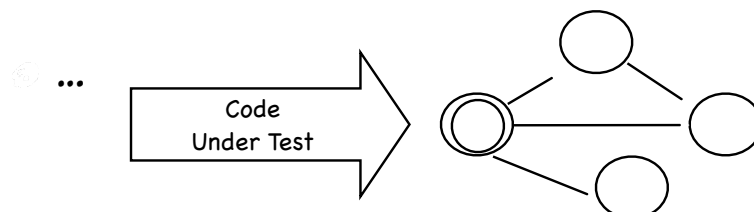
- Unit testing is easy when
 - your code does not depend on any thing
 - No database, file system, web service, socket, ... dependencies
 - your code can be tested fully in isolation
 - Does not require other classes/components, frameworks, etc.
- You quickly create an object, run some tests, and off you go



3

UT is hard when...

- Unit testing is hard when
 - your code had dependencies
 - it needs to fetch that data from the database
 - needs access to the network for some validation
 - needs to validate the credit card number



4

But remember UT means Automation

- ☉ But when we talk about unit testing, we mean automated unit testing
- ☉ We need to be able to run tests quickly
- ☉ Run them as many times as we like
- ☉ You certainly don't want to do either laborious and/or manual setup and tear down
- ☉ You want to check for behavior and ill-behavior of dependent code as well

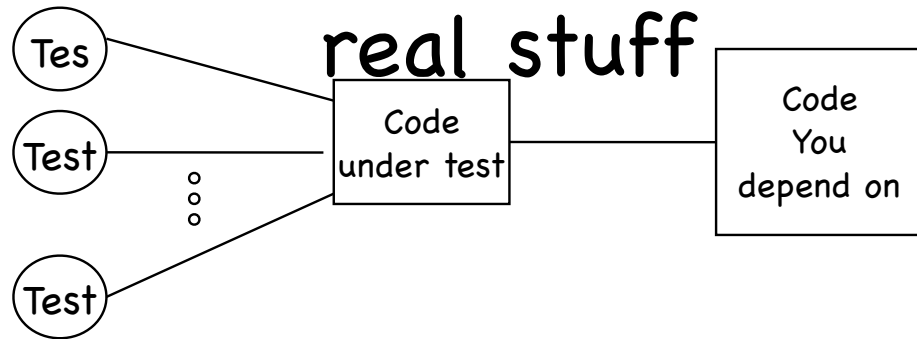
5

What's a Mock?

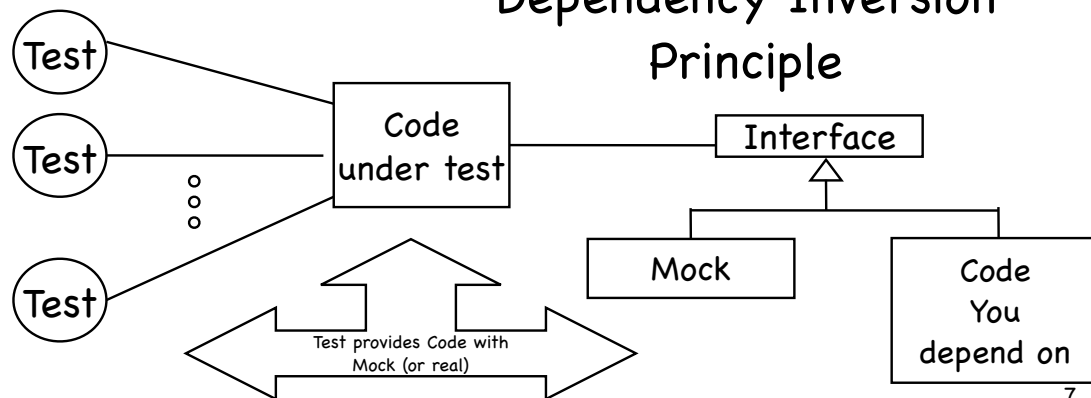
- ☉ A Mock object stands in for the real object
- ☉ It is an object that you first coach... tell him what to say and how to dance, then you set him loose, and he mimics what you coached
- ☉ Simulates the behavior of your dependencies for you

6

But my code depends on real stuff



Dependency Inversion Principle



7

What can a Mock do for you?

- A Mock Object can
 - simulate the expected functionality
 - isolates your code from details that may be filled in later
 - speeds up development of code under test
 - Can simulate both behavior and ill-behavior
 - can be refined incrementally by replacing with actual code

8

A class to create

- Write a Monitor class that will monitor a directory on the file system for any addition or deletion of files
- It raises an alarm or sends an email when a change is detected
- Remember you want to automate your tests
 - Each time you run your test
 - don't expect me to create a file in the directory being monitored
 - No, don't expect me to listen to your stinkin' alarm or be spammed by your emails

9

Mocking Exercise 1

- Mock to stand in for the code that will
 - Represent the file system (or change to it)
 - Code to raise an alarm
 - Code to send out an email

10

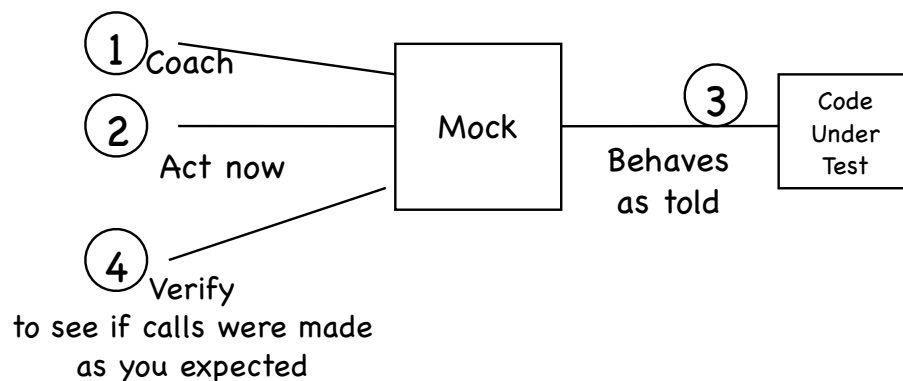
Frameworks to ease your pain

- You can certainly hand toss your mock
- In fact, that is not a bad idea
 - You understand what's going on
 - You control your code
 - You have the flexibility to modify at your will
- But, you may end up creating a lot of classes
 - Why not simply use a mock instead of creating one... this is where the frameworks come in
 - EasyMock, JMock, etc.

11

EasyMock

- EasyMock helps create Mock objects for you on the fly
- Instead of spending time creating a Mock class, you can get ready to use it
- Let's take a look at using EasyMock



12

Mocking Exercise 2

- ☉ Mocking with EasyMock

13

EasyMock is Easy?

- ☉ While the objective is to make it easier to create mock, the reality is it can get a bit tedious setting up the mock's behavior
- ☉ Should you use EasyMock
- ☉ If it is simpler to hand toss a mock, do so
- ☉ If it is easier to use EasyMock, then use it
- ☉ On a project, you may use either approach at different places

14

Switching between Mock and Real

- Your code works with Mock
- Will it work with real object
- When problems creep up, you may want to experiment with mock and real object
- So, how can you switch between these so you can test with mock or real code
- Let's see how...

15

To Mock or not to?

- To
 - Mock only code that code under test directly depends on
 - Mock object that is hard to work with
 - Mock object that takes effort to set up
 - Mock object whose behavior is hard to predict
 - Mock object that requires lots of resources
 - Mock objects that are computationally expensive or very slow to respond
 - Mock objects that your test needs to verify with

16

To Mock or not to?

⌚ Not To

- ⌚ Do not mock for the sake of mocking, ask if you can eliminate the need for mock by refactoring your code
- ⌚ Mock your objects but not resources like database, etc.
- ⌚ When writing the real code is easier than writing mock
- ⌚ Do not mock what will slow down your UT, mocks should be used to speed up UT

17

Other Frameworks

- ⌚ You may hand toss a Mock or use EasyMock/JMock for POJOs
- ⌚ What about something more complex like Servlets, EJBs, JDBC, etc.
- ⌚ Imagine hand tossing a Mock for one of these interfaces?
- ⌚ You may use frameworks like MockRunner to help you with that

18

MockRunner

- To unit test apps in J2EE environment
- Provides mock for Servlet, JDBC, JMS, struts actions, ... without the real container
 - No need for the app server or database
 - Does not provide any in-container testing
 - Can be combined with MockEJB for EJB

19

Dynamic Languages

- Dynamic Languages (like Ruby and Groovy) make Mocking easier
- Duck Typing and Meta Programming eliminates the need for heavy weight frameworks and tools

20

Mocking Exercise 3

- ☉ Mocking Using Dynamic Languages

21

References

- ☉ <http://mockobjects.com>
- ☉ <http://easymock.org>

You can download examples and slides from
<http://www.agiledeveloper.com> - download

22

Thank You!

Please fill in your session evaluations

You can download examples and slides from
<http://www.agiledeveloper.com> - download