Module 2 - Lecture 8

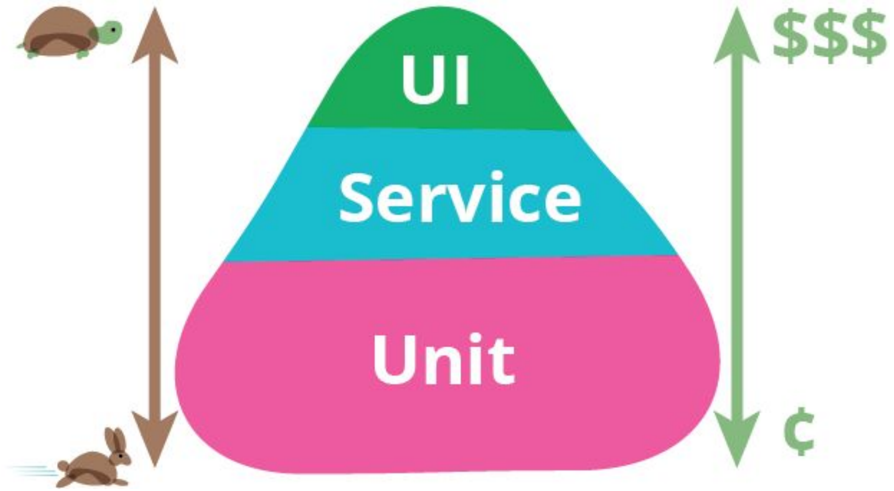
Integration Testing

Review

- JDBC
- DAO Pattern

Testing overview

- **Unit** -> **Integration (Service)** -> **End-to-end (UI)**
- Runtime increases from left to right
- Maintenance and troubleshooting increases from left to right



Integration Testing

- **Integration Testing** is a broad category of tests that validate the integration between units of code or code and outside dependencies, such as databases or network resources.

- **Integration Tests**
  - Often use the same tools as unit tests (i.e. JUnit)
  - Usually slower than unit tests
  - More complex to write, maintain, and debug
  - May have dependencies on outside resources like files or a database

Tests should be...

- **Repeatable**: If the test passes/fails on first execution, it should pass/fail on second execution if no code has changed.

- **Independent**: A test should be able to be run on it's own, independently of other tests, OR together with other tests and have the same result either way.

- **Obvious**: When a test fails, it should be as obvious as possible why it failed.

How do we handle the data?

- If we run a bunch of INSERT, UPDATE, or DELETE statements in a test, won’t that screw up the repeatable and independent parts?

- How do we solve for this?

Transactions!

Recall that a **transaction** is a single unit of work. When it is successful, it should be committed. Transactions can also be rolled back. We will do exactly that to prevent our changes from being realized!

Where does the database we test against reside?

Remotely

A database hosted on a remote server shared by multiple developers on a team.

Pros

- Easy setup, often already exists
- Production-like software and, possibly, hardware

Cons

- Unreliable and brittle
- Temptation of relying on existing data

Locally

A database hosted on a developer’s computer. (This is approach we will use)

Pros

- Reliable (local control)
- Isolation
- Production-like software

Cons

- Requires local hardware resources
- RDBMS needs to be installed and managed

In-memory

An in-memory, embedded database server is started and managed by the test code while running the tests.

Pros

- Reliable
- Isolation
- Consistent across machines (stored in source control)
- Lightweight

Cons

- Not the same software used in production
- Cannot use proprietary features of RDBMS

LET’S CODE!

QUESTIONS?