Minimalist Testing Techniques for Enterprise Java Technology-based Applications

Chris Richardson
Author of POJOs in Action
Chris Richardson Consulting, Inc

www.chrisrichardson.net

TS-4439

What You Will Learn…

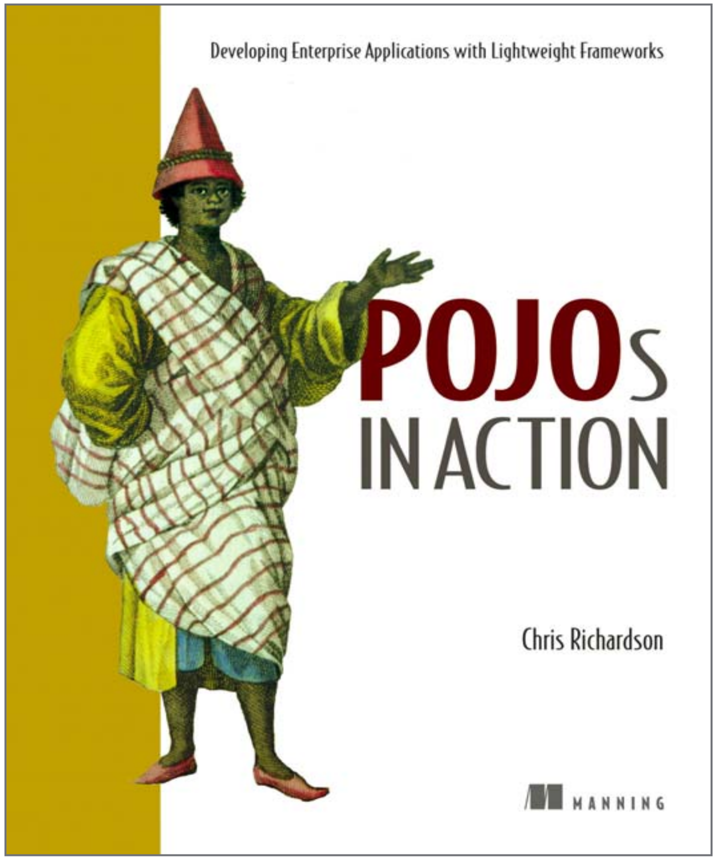
Nasty things can happen to you when you don’t write tests

BUT

It isn’t too difficult to write a few fast running tests

About Chris

- Grew up in England
- Live in Oakland, CA
- 21 years of software development experience
    - OO development since 1986
    - Java™ platform since 1996
    - Java Platform, Enterprise Edition (Java EE) since 1999
- Author of *POJOs in Action*
- Speaker at the JavaOne℠ conference, JavaPolis, NFJS, SD West, JUGs…
- Chair of the eBIG Java SIG in Oakland (www.ebig.org)
- Run a consulting and training company that helps organizations build better software faster


Developing Enterprise Applications with Lightweight Frameworks
POJOs IN ACTION
Chris Richardson
MANNING

Agenda

When Developers Write Tests

Fast Feedback Is Essential

Business Tier Tests

Persistence Tier Tests

Web Tier Tests

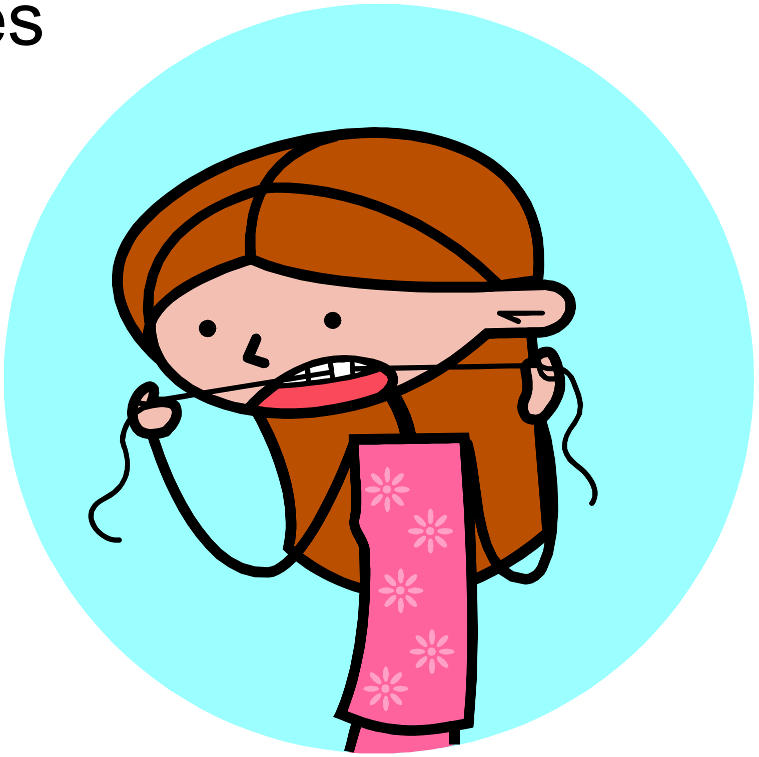
Getting Started

The State of Developer Testing

- (Almost) Everybody agrees that automated tests are good idea

BUT TYPICALLY

- Developers don't write tests
- QA does (manual) testing



Obstacles to Developer Testing

- Cultural obstacles to testing
  - Perceived as extra work that is QA’s responsibility
  - Unnecessary—“My code always works”
  - Not always rewarded—paradox of excellence?
  - Something new to learn
- Technical obstacles to testing
  - Spaghetti code
  - Some frameworks make testing difficult
  - Framework developers must consider testability

Edit and Pray Development

- You can perhaps live with few tests at the start of a project
- But very quickly you need to change existing code, and development slows down
  - No tests—make changes very carefully
  - Lots of manual testing
- More bugs → Long nights, stress…
- Your application decays
  - No one has confidence or time to refactor code
  - Even slower progress
  - Eventually you need to throw it away and start over

But if You Write Tests…

- Fewer bugs that impact customers **and** development
- Write new code more easily
  - Automates what we are doing already—right!?
  - Run fast unit tests instead of slower web application
  - Use TDD to incrementally solve a problem
- Tests are a safety net
  - Confidently change existing code
  - Easier to refactor code to prevent decay
  - The application has longer, healthier life

POJOs Make Testing Easier

- A Plain Old Java Object
  - Does not implement any special interfaces
  - Does not call infrastructure APIs
  - Decouples business logic from infrastructure
- Dependency injection wires components together
  - Simplifies code
  - Promotes loose coupling between components
  - Makes it easy to pass in mocks for testing
- Aspects handle cross-cutting concerns
  - Simplifies code that implements business logic
  - Decouples it from infrastructure

Agenda

When Developers Write Tests

Fast Feedback Is Essential

Business Tier Tests

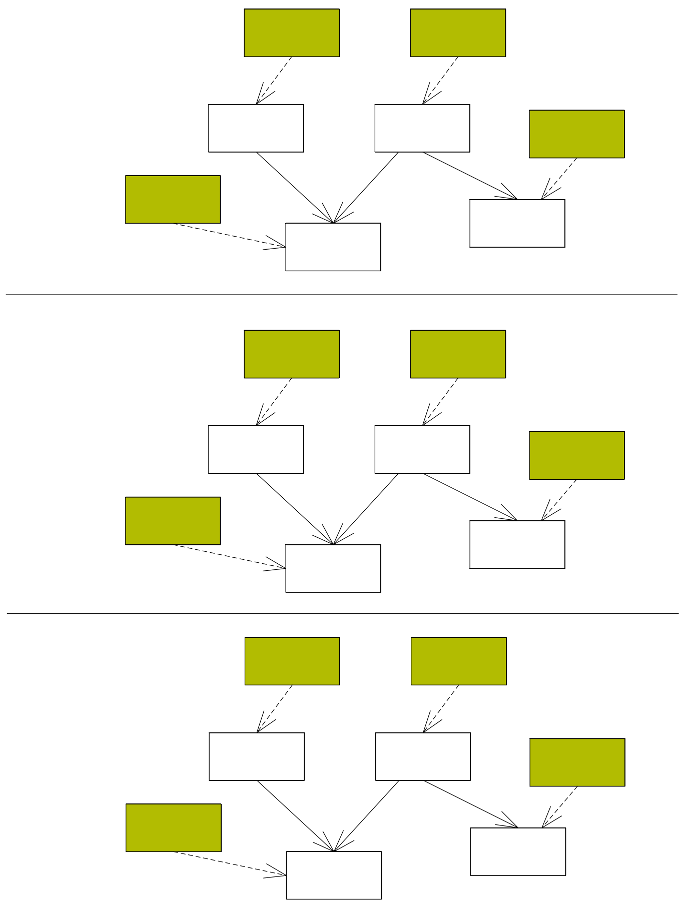
Persistence Tier Tests

Web Tier Tests

Getting Started

Fast Feedback Is Essential

- You change a complex interest calculation
- When do you want to find out whether it works?
  - 15 minutes later after run web tests?
  - 10 seconds later after running a hundred unit tests?
- Write unit tests at every level
  - Fast running
  - Easy to relate test failure with cause

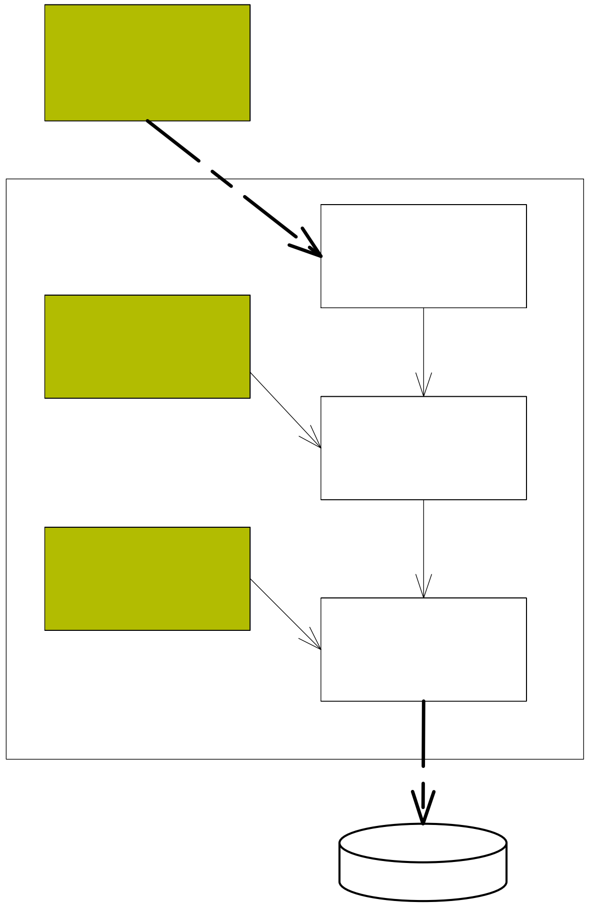


Fast Builds Are Also Essential

- If you are good at writing tests → lots of tests
- Unit tests run very quickly but lots of functional tests can take a long time to run
- Building and testing an application can be slow
  - On some past projects it took 30–50 minutes
  - Yet this had to be done prior to check-in
- Consequences
  - Check-in was a big deal → rarely done
  - Developers didn't run tests → broken builds

One Reason for Slow Tests = Going Outside of the Virtual Machine for the Java Platform (JVM™ Interface)

- Tests that cross JVM interface boundaries are generally slow
- Databases are slow
  - Testing at every layer → hit the database over and over again
- Web tests tend to be slow
  - JavaServer Pages™ (JSP™) technology compilation time
  - Networking



The terms “Java Virtual Machine” and “JVM” mean a Virtual Machine for the Java™ platform

Minimizing Test Times

- Solutions
  - Write lots of fast running tests, i.e., unit tests
  - Run different tests at different times
- Developers should run mainly fast tests
  - During development run mostly unit tests
  - Before check-in run some functional tests
- Continuous integration server
  - Runs slower but more thorough tests
  - But getting fast feedback is also important
  - Consider multiple levels of CI server testing
  - Use a parallelized build server

Avoid the Boiled Frog Problem

- More development → more tests → longer test times
- Suddenly, the tests take too long
- But you don't know how to fix it
- Be vigilant! Invest in reducing the build time when necessary

Agenda

When Developers Write Tests

Fast Feedback Is Essential

Business Tier Tests

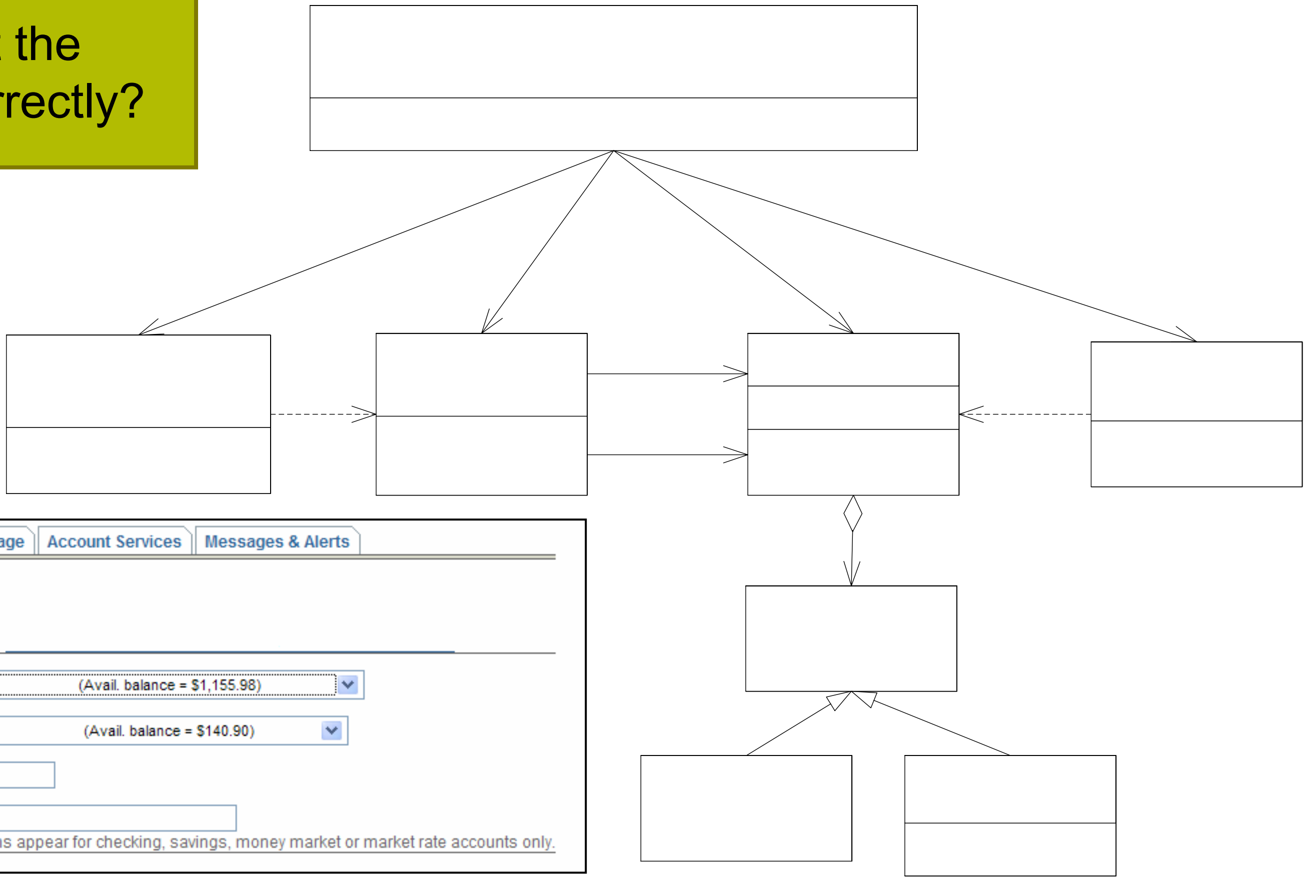
Persistence Tier Tests

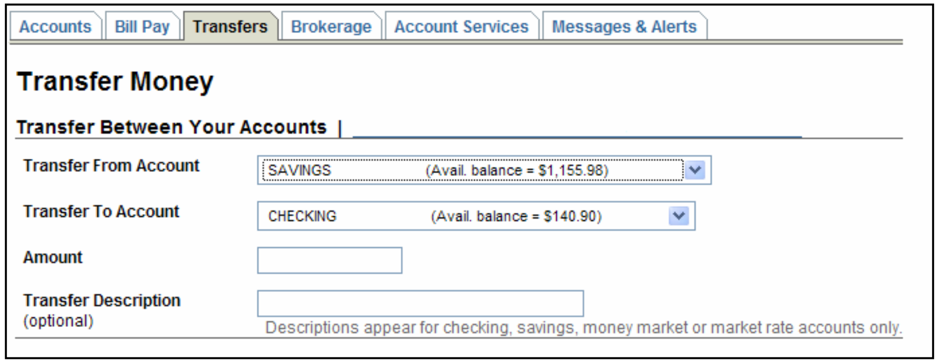
Web Tier Tests

Getting Started

The Example Business Logic

Does it implement the business rules correctly?





Transfer Money

Transfer Between Your Accounts |

Transfer From Account	SAVINGS (Avail. balance = $1,155.98)
Transfer To Account	CHECKING (Avail. balance = $140.90)
Amount	
Transfer Description (optional)	

Descriptions appear for checking, savings, money market or market rate accounts only.

Testing a POJO Domain Object

```
public class Account {

private String accountId;
private double balance;
private OverdraftPolicy
        overdraftPolicy;

public double getBalance() {
  return balance;
}

public void debit(double amount)
{ … }

public void credit(double
       amount) { … }
```

Relatively easy to write tests that run blindingly fast

```
public class AccountTests
          extends TestCase {

private Account account;

public void setUp() {
  account = AccountMother
              .makeAccount(10.0);
}

public void test_normal() {
  assertMoneyEquals(10.0,
               account.getBalance());
  account.debit(5);
  assertMoneyEquals(5.0,
               account.getBalance());
  account.credit(10);
  assertMoneyEquals(15.0,
               account.getBalance());
}

…
```

But How to Test a Service?

```
public class MoneyTransferServiceImpl implements MoneyTransferService {

  private final AccountRepository accountRepository;
  private final BankingTransactionRepository bankingTransactionRepository;

  public MoneyTransferServiceImpl(AccountRepository accountRepository,
      BankingTransactionRepository bankingTransactionRepository) {
    this.accountRepository = accountRepository;
    this.bankingTransactionRepository = bankingTransactionRepository;
  }

  public BankingTransaction transfer(String fromAccountId,
      String toAccountId, double amount) throws MoneyTransferException {
    Account fromAccount = accountRepository.findAccount(fromAccountId);
    Account toAccount = accountRepository.findAccount(toAccountId);
    fromAccount.debit(amount);
    toAccount.credit(amount);
    TransferTransaction txn = new TransferTransaction(fromAccount,
        toAccount, amount, new Date());
    bankingTransactionRepository.addTransaction(txn);
    return txn;
}

…
```

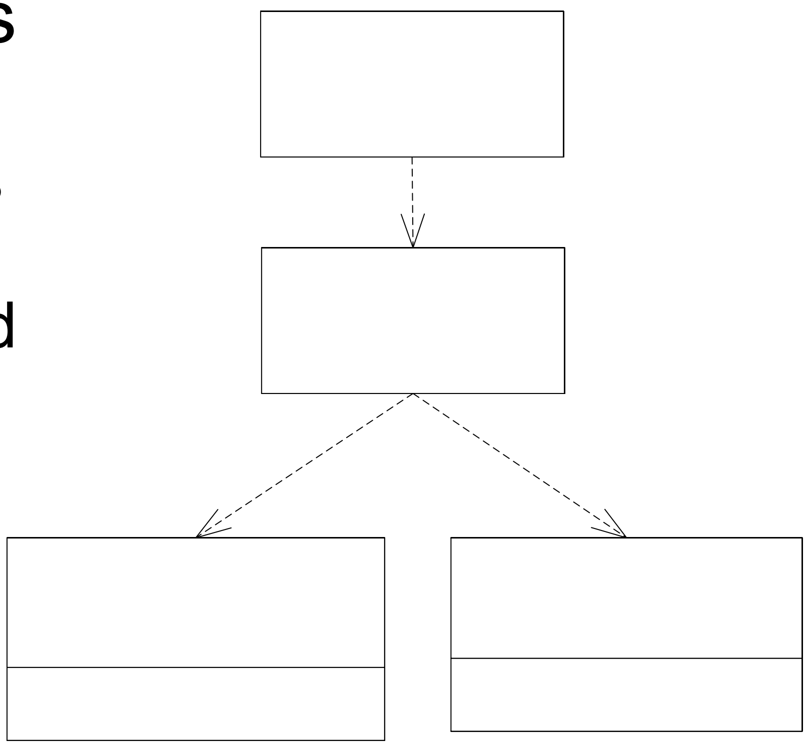
Database access

The Slow Way: Write Integration Tests

- Each test
  - Initializes the database
  - Calls the service
  - Verifies state of database
- But lots of database accesses → slow test
- Require lots of setup
  - Setting up the database
  - Initializing the database
  - …
- Integration tests are valuable but…

Faster Testing With Mock Objects

- A mock object simulates the real object
  - Returns values or throws exceptions
  - Verifies that the expected methods are called
- Using mocks
  - Simplifies tests
  - Speeds up tests
  - Enables an object to be tested in isolation
  - Enables top-down development



Creating Mocks

- Write your own mocks
  - Simple for interfaces but it becomes tedious
  - How to mock concrete classes?
- Use a mock object framework
  - jMOCK, EasyMock
- Create and configure mock object
  - Specify expected method and arguments
  - Define method behavior: return value or throw exception

Mock Objects Example: Part 1

```
public class MoneyTransferServiceTests extends TestCase {

protected void setUp() throws Exception {
    super.setUp();
    accountRepository = createMock(AccountRepository.class);
    bankingTransactionRepository =
         createMock(BankingTransactionRepository.class);

    service = new MoneyTransferServiceImpl(accountRepository,
        bankingTransactionRepository);

    fromAccount = AccountMother.makeAccount(100);
    toAccount = AccountMother.makeAccount(200);
    fromAccountId = fromAccount.getAccountId();
    toAccountId = toAccount.getAccountId();

}
```

Create service with mocks

Mock Objects Example: Part 2

```
public class MoneyTransferServiceTests extends TestCase {

  public void testTransfer_normal() {

    expect(accountRepository.findAccount(fromAccountId)).andReturn(fromAccount);
    expect(accountRepository.findAccount(toAccountId)).andReturn(toAccount);

    bankingTransactionRepository.addTransaction(isA(BankingTransaction.class));

    replay(accountRepository, bankingTransactionRepository);

    BankingTransaction result = service.transfer(fromAccountId, toAccountId, 50);

    assertNotNull(result);

    assertMoneyEquals(50.0, fromAccount.getBalance());
    assertMoneyEquals(250.0, toAccount.getBalance());

    verify(accountRepository, bankingTransactionRepository);

  }
```

Downsides of Mocks

- Testing the collaboration of objects = white box testing
- Tests can be brittle
  - Change design without changing what it does → failing tests
  - Discourages developers from writing tests
- Fortunately, many collaborations are stable
  - e.g., between services and repositories
- Mock selectively!

Agenda

When Developers Write Tests

Fast Feedback Is Essential

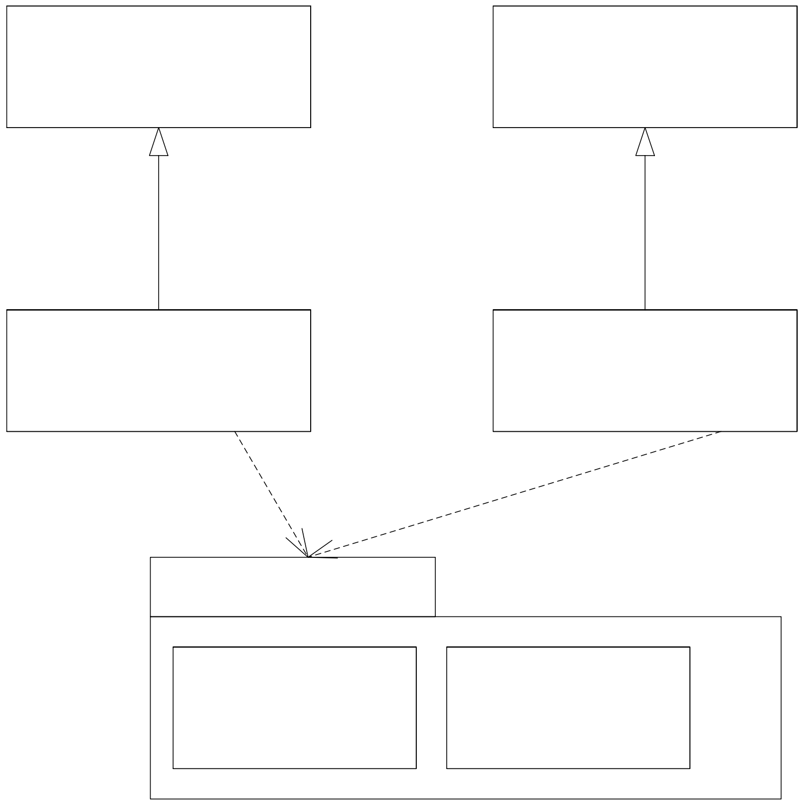
Business Tier Tests

Persistence Tier Tests

Web Tier Tests

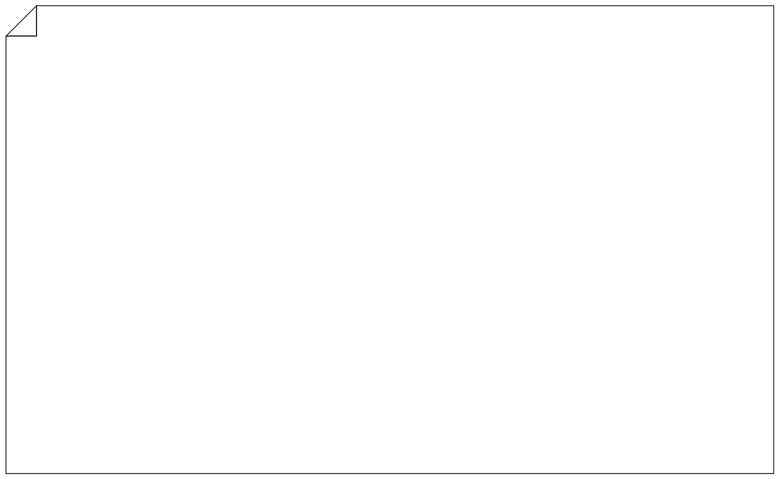
Getting Started

Persistence Tier Components



Can my application create, find, update and delete persistent objects?

Does the data end up in the correct table/column?

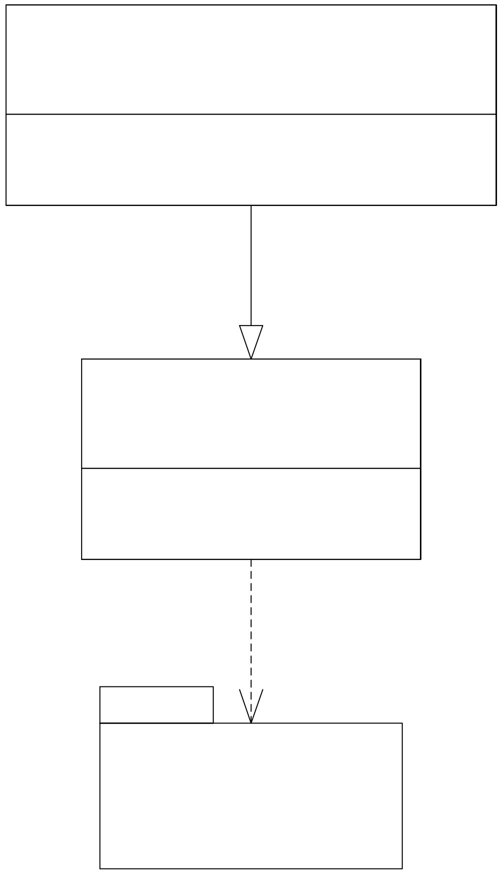


The Slow Way to Test

- Write lots of tests that bang against the database
  - Initialize the database
  - Load an object
  - Save an object
  - Verify the state of the database
- Drawbacks
  - Lots of database access → slow
  - Need to initialize the database → difficult to write
- We still need some tests like this but we can do better…

Avoid the DB #1: Mock the ORM Framework

- Problem
  - Bugs in the logic of the DAOs
  - Testing against the database is slow
- Solution
  - Mock the ORM framework APIs



Avoid the DB #2: Test the O/R Mapping

- Problem
  - Incorrectly defined mapping,
  - e.g., forgetting to map a field is a common bug
  - But tests that save objects and check the contents of the DB are slow to execute and tedious to write
- Solution
  - Read XML O/RM and make assertions about it
  - ORMUnit framework makes this easy to do

```
class BankingMappingTests extends HibernateMappingTests {

 public void testAccount() {
   assertClassMapping(Account.class, "BANK_ACCOUNT");
   assertAllFieldsMapped();
 }
…
```

Faster Tests #1: Use an In-Memory DB

- For example: HSQLDB
- Typically much faster than a traditional DB
  - Committing transactions
  - Recreating the schema
- No install—it's just a Java Archive (JAR) file
- O/RM = DB portability → makes this easy
- Issues
  - Difficult to do if using hand-coded SQL
  - Some incompatibilities: e.g., time precision

Faster Tests #2: Rollback Transactions

- Execute entire test in a transaction, which is rolled back
- Tests run faster
- Leaves the database unchanged
- Issues to consider
  - Single transaction → Single Session/EntityManager → potentially very different behavior
  - Commit-time constraints not checked
  - Code in a different JVM interface can’t see the changes

DEMO

Walkthrough Persistence Tests Code

Agenda

When Developers Write Tests

Fast Feedback Is Essential

Business Tier Tests

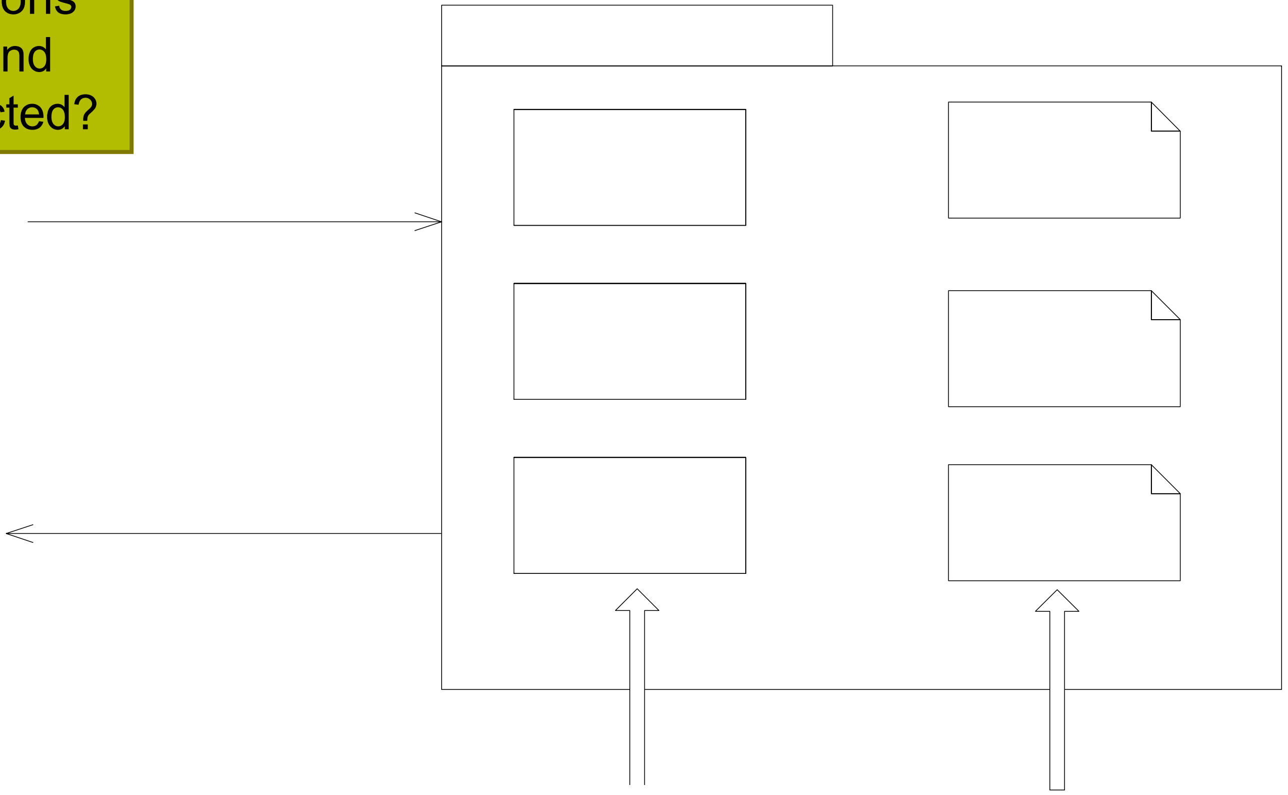
Persistence Tier Tests

Web Tier Tests

Getting Started

Web Tier Design

Do the links, buttons and fields exist and behave as expected?



Unit Test Web Components

- Simulate HTTP request
  - Request parameters
  - Cookies
  - Session state
  - …
- Use mocks for services
- Verify
  - Service invocation
  - View selection
  - Data passed to view

Web Application Testing

- Simulate a user clicking and typing in a browser
- Superficial tests
  - Test happy paths
  - Easy way to test basic functionality
- More thorough tests
  - Test lots of different scenarios
  - Lots of work

Web Testing With Selenium

- Selenium
  - Open source web application testing tool
  - Tests run in a real browser (IE/Firefox/…)
- Three components
  - Core = JavaScript™ technology library that runs in the browser
  - IDE = Firefox plug-in for recording and executing tests
  - Remote Control (RC) = framework for writing automated tests in Java/.NET/Ruby/…

Selenium RC—Code Example

```
public WebTest extends … {

  public void setUp() throws Exception {
    …
    selServer = new SeleniumServer();
    selServer.start();

    selenium = new DefaultSelenium("localhost", selServer.getPort(),
                                   "*iexplore",
                                   "http://localhost:8080");
    selenium.start();
  }

  public void testCreateProject(Selenium selenium) {
    selenium.open("/ptrack/acegilogin.jsp");
    selenium.type("j_username", "proj_mgr");
    selenium.type("j_password", "faces");
    selenium.click("Login");
    …
  }
```

Starting and Stopping the Web Container

- Testing with an embedded web container
  - e.g., Jetty
  - Avoids having to build a WAR
  - Typically starts up faster
- Testing with a web container in a separate JVM interface
  - Typically slower
  - Requires a WAR to be built
- Use the Cargo open-source framework
  - Installs/starts/stops web containers
  - Deploys/un-deploys web applications
  - Java API, Maven Plug-in, Ant tasks, IDE plug-ins

Cargo Example

```
public WebTest extends … {

  public void setUp() throws Exception {
     ZipURLInstaller installer = new ZipURLInstaller(
        new URL("http://apache.tradebit.com/.../jakarta-tomcat-5.0.28.zip"),
        new File(tempDir, "tomcat-install"));
    installer.install();

    Tomcat5xStandaloneLocalConfiguration config = new
                   Tomcat5xStandaloneLocalConfiguration(
                                   new File(tempDir, "tomcat-deploy"));
    config.setProperty(ServletPropertySet.PORT, "8080"));

    WAR war = new WAR(locateWAR("webapp/target/ptrack.war"));
    config.addDeployable(war);

    container = new Tomcat5xInstalledLocalContainer(config);
    File home = installer.getHome();
    container.setHome(home);

    container.start();
    …
  }
```

DEMO

Review and Run Selenium/Cargo Test Code

Speeding Up Web Tests

- Web tests can be slow
  - Lots of inter-process communication
  - Database access
  - JSP technology page compilation
- Minimize start and stops of browser and web application
  - JUnit Decorator that starts browser/server once for a set of tests
  - TestNG @BeforeClass
- Don’t run all the web tests on developer’s desktop, e.g.,
  - Only run embedded web container tests
  - Only run a subset of the tests

Agenda

When Developers Write Tests

Fast Feedback Is Essential

Business Tier Tests

Persistence Tier Tests

Web Tier Tests

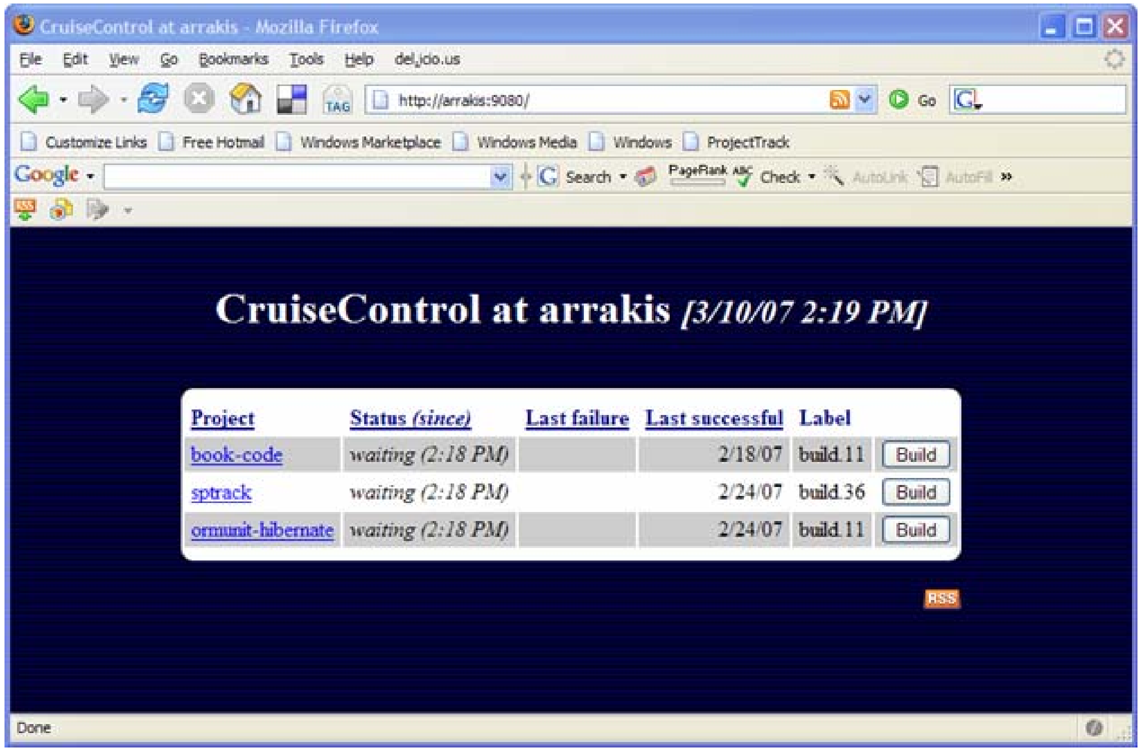
Getting Started

Getting Started Incrementally

- Existing application = lots of code
- Impractical to stop and create tests for everything
- Need an incremental strategy

Install Continuous Integration Server (If You Haven't Already)

e.g., CruiseControl


CruiseControl at arrakis - Mozilla Firefox
File Edit View Go Bookmarks Tools Help del.icio.us
http://arrakis:9080/
Customize Links Free Hotmail Windows Marketplace Windows Media Windows ProjectTrack
Done

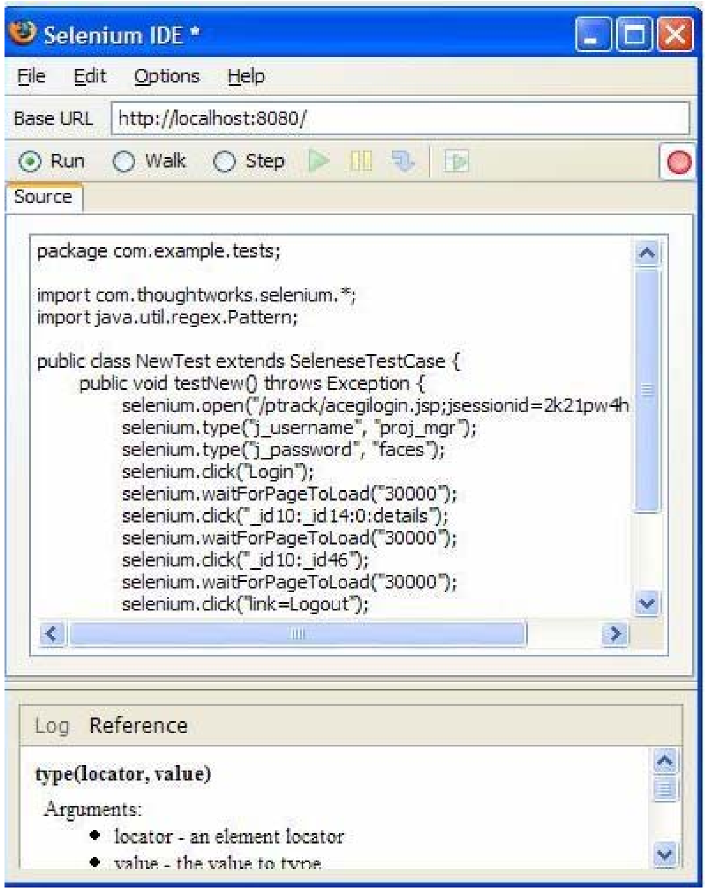
CruiseControl at arrakis *[3/10/07 2:19 PM]*

Project	Status *(since)*	Last failure	Last successful	Label	
book-code	*waiting (2:18 PM)*		2/18/07	build.11	Build
sptrack	*waiting (2:18 PM)*		2/24/07	build.36	Build
ormunit-hibernate	*waiting (2:18 PM)*		2/24/07	build.11	Build

RSS

Create Some Web UI Tests

- Use Selenium IDE to create basic tests
  - Push buttons
  - Click links
  - …
- Run
  - Before check-in
  - With CruiseControl



Base URL http://localhost:8080/

```
package com.example.tests;

import com.thoughtworks.selenium.*;
import java.util.regex.Pattern;

public class NewTest extends SeleneseTestCase {
	public void testNew() throws Exception {
		selenium.open("/ptrack/acegilogin.jsp;jsessionid=2k21pw4h
		selenium.type("j_username", "proj_mgr");
		selenium.type("j_password", "faces");
		selenium.click("Login");
		selenium.waitForPageToLoad("30000");
		selenium.click("_id10:_id14:0:details");
		selenium.waitForPageToLoad("30000");
		selenium.click("_id10:_id46");
		selenium.waitForPageToLoad("30000");
		selenium.click("link=Logout");
```

type(locator, value)

Arguments:
- locator - an element locator
- value - the value to type

DEMO

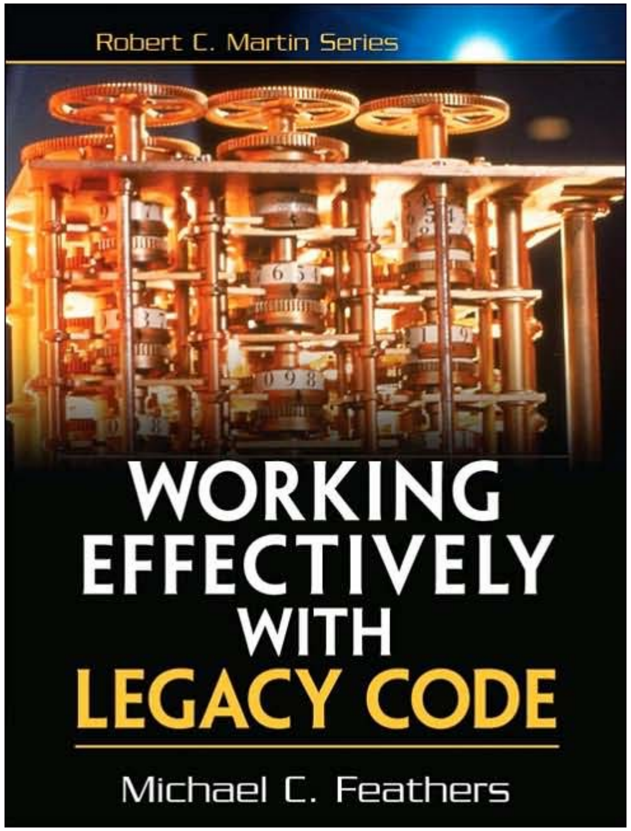
Recording a Web Test With Selenium IDE

Write More Tests When…

- Fixing a bug
  - Write a functional web UI test
  - Write a low-level unit test
- Working on a component
  - Write characterization tests for existing behavior
  - Write some tests for the new behavior
  - Make the tests pass

If Your Code = Big Ball of Mud

- This won’t stop you writing functional tests
- But it’s difficult to write unit tests

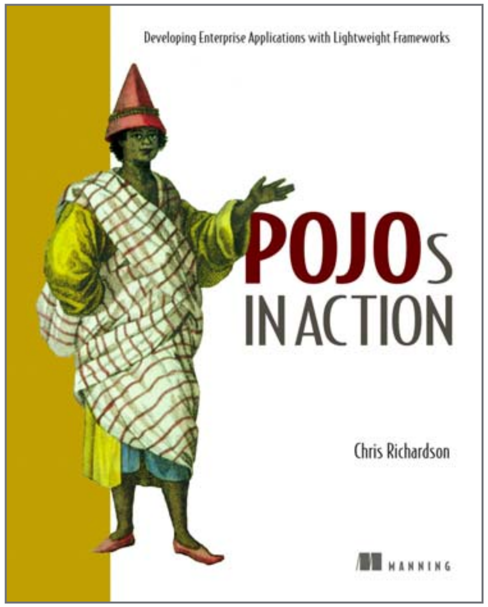


Summary

- Bad things can happen without tests
  - Development slows down
  - The application decays
  - It can be a downward spiral
- Writing some basic tests isn’t that difficult
  - Write tests for the POJO business logic
  - Test the OR/M mapping metadata
  - Use Selenium for web testing
  - Incrementally write tests for existing code

  - Just do it!

For More Information


Developing Enterprise Applications with Lightweight Frameworks
POJOs IN ACTION
Chris Richardson
MANNING

- Send e-mail
  - chris@chrisrichardson.net
- ORMUnit website
  - http://code.google.com/p/ormunit/
- ProjectTrack Sample Code
  - http://code.google.com/p/projecttrack/
- My website for other resources
  - www.chrisrichardson.net
- Other sessions
  - TS-7082—Building JavaServer Faces Applications with Spring and Hibernate
  - BOF-7846—The Long-Tail Treasure Trove
  - BOF-6825—Testing Web 2.0 Features, Using Real-World Applications
  - TS-4588—Advanced Enterprise Debugging Techniques

Q&A

Minimalist Testing Techniques for Enterprise Java Technology-based Applications

Chris Richardson
Author of POJOs in Action
Chris Richardson Consulting, Inc.

www.chrisrichardson.net

TS-4439