A Dozen Ways to Get the Testing Bug

Mike Clark
www.clarkware.com

Rocky Mountain Software Symposium

May 21-23, 2004

http://nofluffjuststuff.com

Learning Was Painful

Why Even Try?

- I don't have time to test
- Testing is boring
- I'm afraid changes will break something
- My designs don't always work
- I have a tendency to over-engineer

Why I'm Now Hooked

Testing helps me write better code, *faster*!

What I Mean By Testing

- Tests that support programming
  - Clarify thinking
  - (Find bugs)
  - What programmers do
- **Not** tests that verify a product
  - Uncover errors and omissions
  - What "QA" does

1. Let Computers Do Boring Stuff

- Replace visual inspection with automated checking
- Automation isn't testing, but it gives us time to test
- Computers are bored
- Go after low-hanging fruit first

Manual Testing

```
public static void main(String args[]) {

   Spreadsheet sheet = new Spreadsheet();

   System.out.println("Cell reference:");
   sheet.put("A1", "5");
   sheet.put("A2", "=A1");
   System.out.println("A2 = " + sheet.get("A2"));

   System.out.println("\nFormula calculation:");
   sheet.put("A1", "5");
   sheet.put("A2", "2");
   sheet.put("B1", "=A1*(A1-A2)+A2/3");
   System.out.println("B1 = " + sheet.get("B1"));
}
```

```
Cell reference:
A2 = 5

Formula calculation:
B1 = 15
```

JUnit

- A computer's taskmaster
- De facto unit-testing tool for Java
- Tests are self-checking and unambiguous
- Simple to use

Computer-Checked Assertions

```
assertTrue(boolean condition)
assertFalse(boolean condition)

assertEquals(Object expected, Object actual)

assertEquals(float expected, float actual, float delta)

assertSame(Object expected, Object actual)
assertNotSame(Object expected, Object actual)

assertNull(Object o)
assertNotNull(Object o)

fail([String message])
```

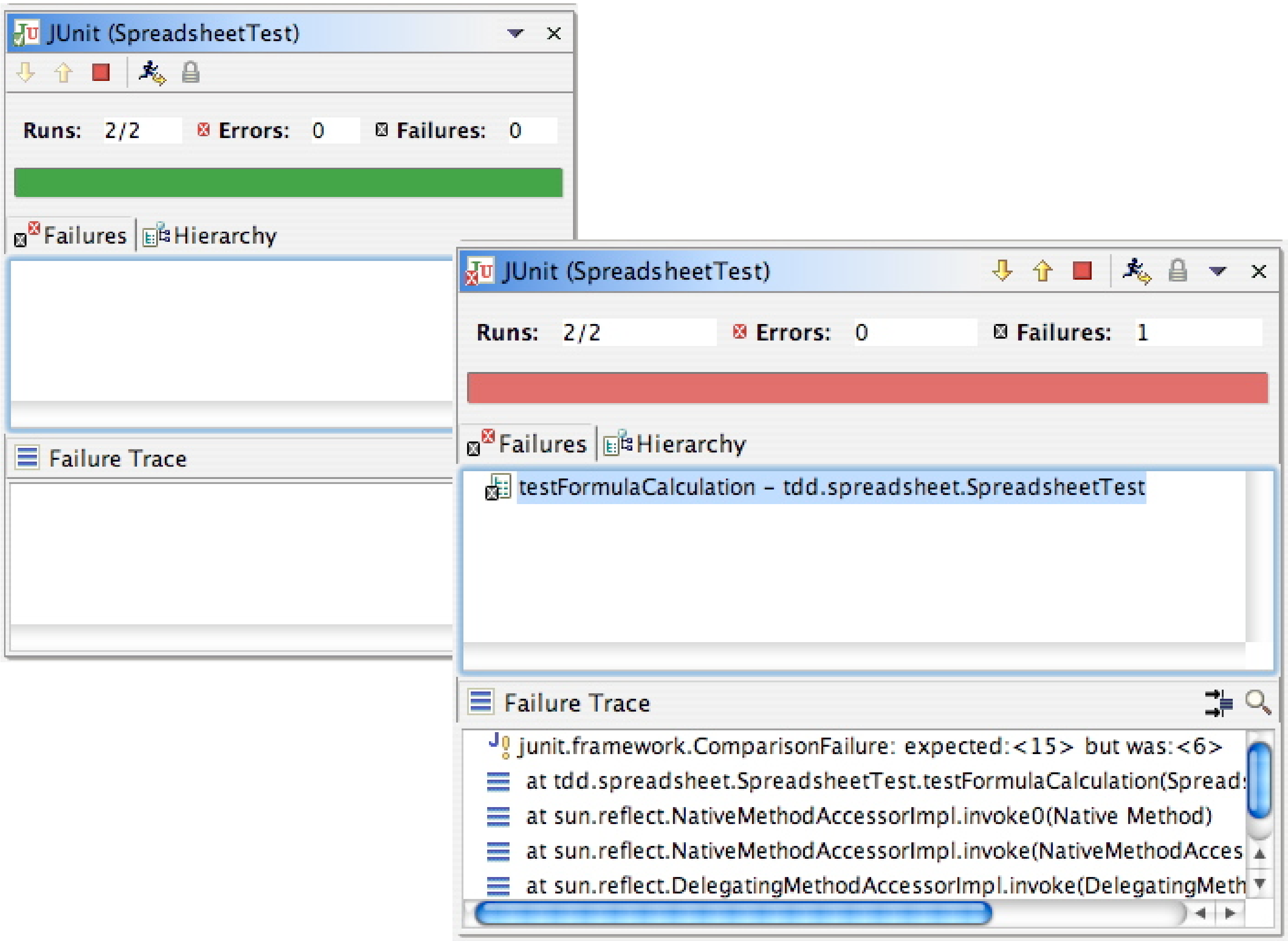
Automated Testing

```
public class SpreadsheetTest extends TestCase {

  public void testCellReference() {
    Spreadsheet sheet = new Spreadsheet();
    sheet.put("A1", "5");
    sheet.put("A2", "=A1");
    assertEquals("5", sheet.get("A2"));
  }

  public void testFormulaCalculation() {
    Spreadsheet sheet = new Spreadsheet();
    sheet.put("A1", "5");
    sheet.put("A2", "2");
    sheet.put("B1", "=A1*(A1-A2)+A2/3");
    assertEquals("15", sheet.get("B1"));
  }
}
```

Green Is Good!



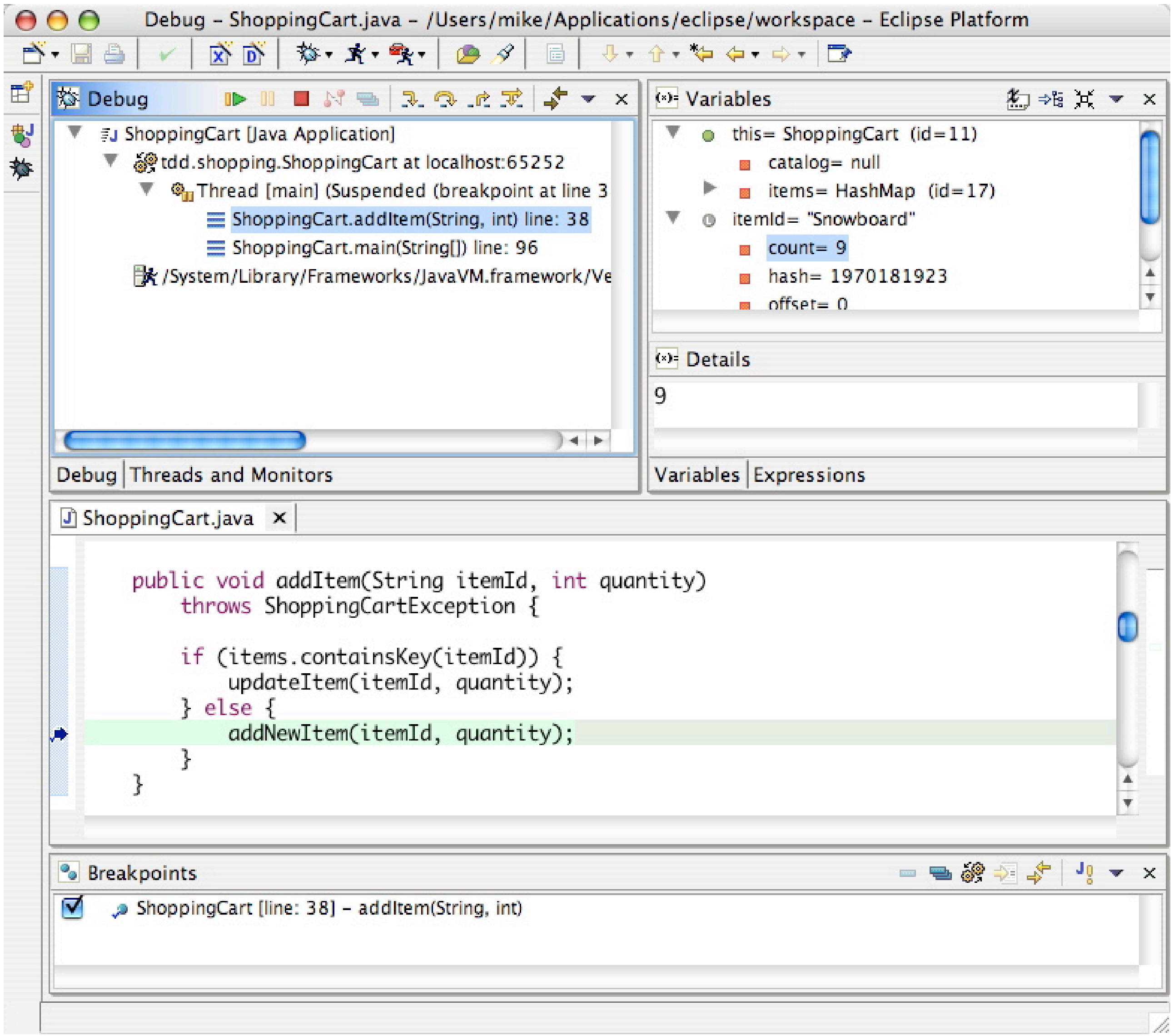
Benefits of Automated Tests

- Increase in value over time
- Automated change detectors
- Refactoring courage
- Executable documentation

2. Stop Debugger Testing

- It's not a regression testing tool
- Mental assertions are error-prone and boring
- Debugger time isn't recyclable
- Debugger sessions aren't reusable by others

Is This Time Well Spent?



Automated Tests Scale

- Codify debugger assertions in an automated test
- And listen to the pain in trying to do so
  - It's really trying to tell you something
- Then run the tests at the push of a button

3. Assert Your Expectations

- Test-driven development recipe:
  1. Write new code only after an automated test has failed
  2. Refactor to keep the code clean
-
- Don't you already have a mental test?
- TDD formalizes and refines that thought process

Put A Stake in the Ground

```
import junit.framework.TestCase;

public class ShoppingCartTest extends TestCase {

    public void testAddItems() {
        ShoppingCart cart = new ShoppingCart();
        cart.addItems("Snowboard", 1);
        cart.addItems("Lift Ticket", 2);
        cart.addItems("Snowboard", 1);
        assertEquals(4, cart.itemCount());
    }
}
```

Code Toward the Goal

- Now make the test pass
- Simplest possible solution
- Then refactor and re-run the test

- Writing tests afterwards isn't nearly as fun!
  - And it's usually more difficult

Question?

Doesn't that mean I spend a lot of time writing tests?

Yes!

But the rest of the time you're making those tests pass.

4. Think of It as Design

- TDD is a design technique
  - Clarifies your thinking
  - Validates design decisions
  - Encourages loose coupling

- Pay careful attention to difficulties writing tests
- If code is difficult to test, it's difficult to use

Design Decision

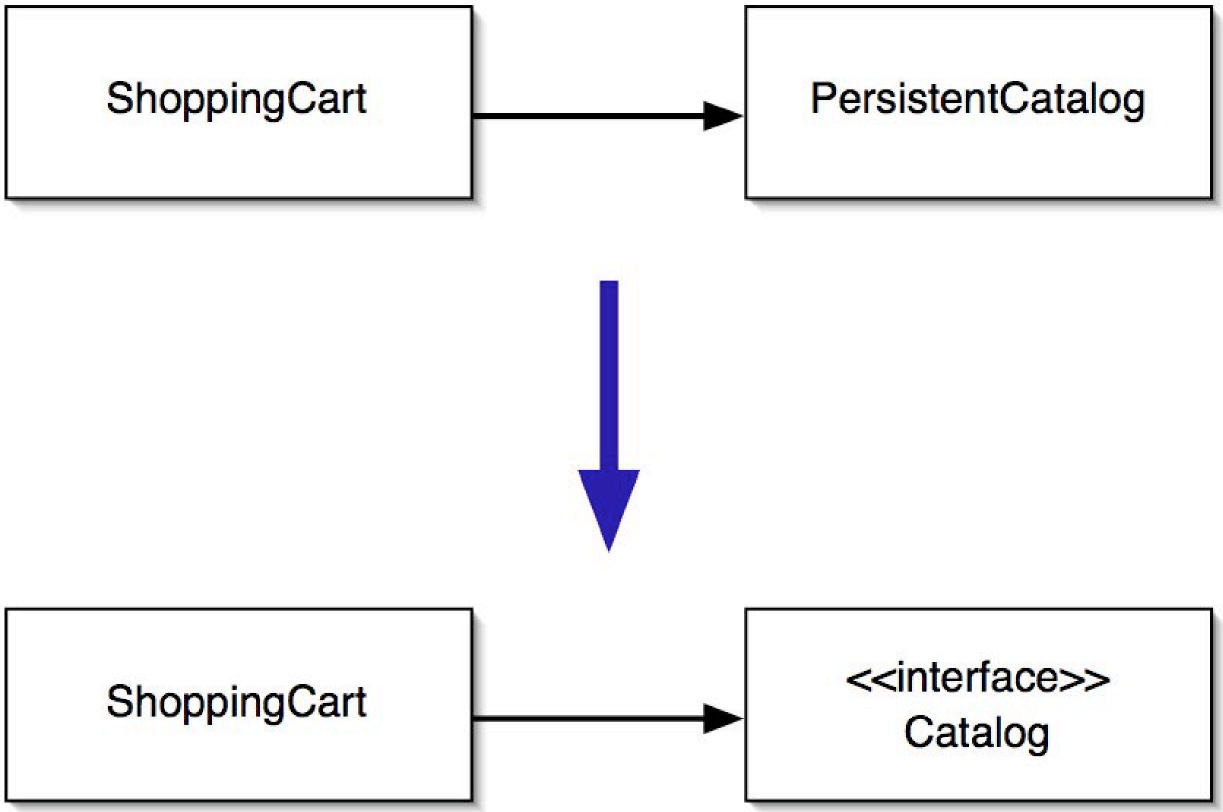
```
public void testGetItem() {

  ShoppingCart cart = new ShoppingCart();
  cart.addItems("ISBN123", 1);

  Iterator items = cart.items();
  Product item = (Product)items.next();

  assertEquals("Confessions of an OO Hired Gun",
    item.getDescription());
  assertEquals(9.95, item.getUnitCost(), 0.0);
  assertEquals(1, item.getQuantity());
}
```

Breaking Dependencies



Design By Interface

```
public interface Catalog {

    public void addProduct(String key, Product p);

    public Product getProduct(String key);

}
```

Decouple By Composition

```
public void testGetItem() {

  Catalog catalog = new InMemoryCatalog();
  catalog.addProduct("ISBN123",
    new Product("Confessions of an OO Hired Gun", 9.95));

  ShoppingCart cart = new ShoppingCart(catalog);
  cart.addItems("ISBN123", 1);

  Iterator items = cart.items();
  Product item = (Product)items.next();

  assertEquals("Confessions of an OO Hired Gun",
    item.getDescription());
  assertEquals(9.95, item.getUnitCost(), 0.0);
  assertEquals(1, item.getQuantity());
}
```

Green

5. Build Safety Nets

- Legacy code without tests is a liability
- Be pragmatic about scope
- Use existing safety nets
- Makes refactoring safe

6. Learn by Checked Example

- Write checked examples to learn APIs
- Safe context for learning
- Builds up a knowledge base
- Regression suite for new releases

Lucene Search Test

```
public class LuceneLearningTest extends TestCase {

  public void testIndexedSearch() throws Exception {

    Directory indexDirectory = new RAMDirectory();
    IndexWriter writer =
      new IndexWriter(indexDirectory, new StandardAnalyzer(), true);

    Document document = new Document();
    document.add(Field.Text("contents", "Learning tests build confidence!"));
    writer.addDocument(document);
    writer.close();

    IndexSearcher searcher = new IndexSearcher(indexDirectory);
    Query query = new TermQuery(new Term("contents", "confidence"));

    Hits hits = searcher.search(query);
    assertEquals(1, hits.length());
  }
}
```

Lucene Knowledge Base

```
public class LuceneLearningTest extends TestCase {

  private IndexSearcher searcher;

  public void setUp() throws Exception {
    Directory indexDirectory = new RAMDirectory();
    IndexWriter writer = new IndexWriter(indexDirectory, new StandardAnalyzer(), true);
    Document document = new Document();
    document.add(Field.Text("contents", "Learning tests build confidence!"));
    writer.addDocument(document);
    writer.close();
    searcher = new IndexSearcher(indexDirectory);
  }

  public void testSingleTermQuery() throws Exception {
    Query query = new TermQuery(new Term("contents", "confidence"));
    Hits hits = searcher.search(query);
    assertEquals(1, hits.length());
  }

  public void testBooleanQuery() throws Exception {
    Query query = QueryParser.parse("tests AND confidence", "contents", new StandardAnalyzer());
    Hits hits = searcher.search(query);
    assertEquals(1, hits.length());
  }
}
```

Ruby Knowledge Base

```
class RubyArrayTest < Test::Unit::TestCase

  def testPushPopShift
    a = Array.new
    a.push("A")
    a.push("B")
    a.push("C")
    assert_equal(["A", "B", "C"], a)
    assert_equal("A", a.shift)
    assert_equal("C", a.pop)
    assert_equal("B", a.pop)
    assert_equal(nil, a.pop)
  end

  def testCollect
      a = ["H", "A", "L"]
      collected = a.collect { |element| element.succ }
      assert_equal(["I", "B", "M"], collected)
  end

end
```

7. Corner Bugs

- Before you can fix a bug, you must find it
- You have expectations for how the code *should* work
- The bug is fixed when the test passes
- And it's cornered for life

8. Expand Your Toolbox

- Be creatively lazy about building testing tools
- The open source world is teeming
- Pick the right tool for the job

How Do I Test This?

```
public class ShoppingServlet extends HttpServlet {

  ...

  public void
  addRequestedItem(HttpServletRequest request, ShoppingCart cart) {

    String itemId = request.getParameter("item");
    String quantity = request.getParameter("qty");
    cart.addItem(itemId, new Integer(quantity).intValue());
  }

  ...
}
```

Mock Objects Framework

- http://mockobjects.com
- Core framework with expectation classes
- Specialized JDK and J2EE frameworks (e.g. servlets)

Mocking Servlets

```
public void testAddRequestedItem() throws Exception {

  ShoppingServlet servlet = new ShoppingServlet();

  MockHttpServletRequest request = new MockHttpServletRequest();
  request.setupAddParameter("item", "Snowboard");
  request.setupAddParameter("qty", "1");

  ShoppingCart cart = new ShoppingCart();
  servlet.addRequestedItem(request, cart);

  assertEquals(1, cart.getItems().size());
  Product item = cart.getItem("Snowboard");
  assertEquals("Snowboard", item.getId());
  assertEquals(1, item.getQuantity());
}
```

Green

Don't Stop There!

- JUnit is a framework, not an application

- NUnit
- CppUnit
- TestUnit (Ruby)
- HttpUnit
- XMLUnit

- Google is a programmer's best friend

9. Make It Part of Your Build Process

- Capitalize on the testing investment
- Build should fail if any test fails
- Instills confidence in the build

Batch Testing with Ant

```
<target name="test" depends="compile-tests“>

  <junit haltonfailure=“true">

    <batchtest>
      <fileset dir="${build.dir}"
        includes="**/*Test.class" />
    </batchtest>

    <formatter type="plain" usefile="false" />
    <classpath refid=“test.classpath" />

 </junit>

</target>
```

Here's Your Build Process

ant test

Radiate Confidence

Unit Test Results.

Home

Packages

tdd.shopping
tdd.shopping.web

Classes

CactusShoppingServletTest
HtmlUnitTest
HttpUnitLinkTest
HttpUnitTest
JWebUnitTest
MockShoppingServletTest
ShoppingCartTest

Unit Test Results

Designed for use with JUnit and Ant.

Summary

Tests	Failures	Errors	Success rate	Time
30	**1**	**0**	**96.67%**	**19.255**

Note: *failures* are anticipated and checked for with assertions while *errors* are unanticipated.

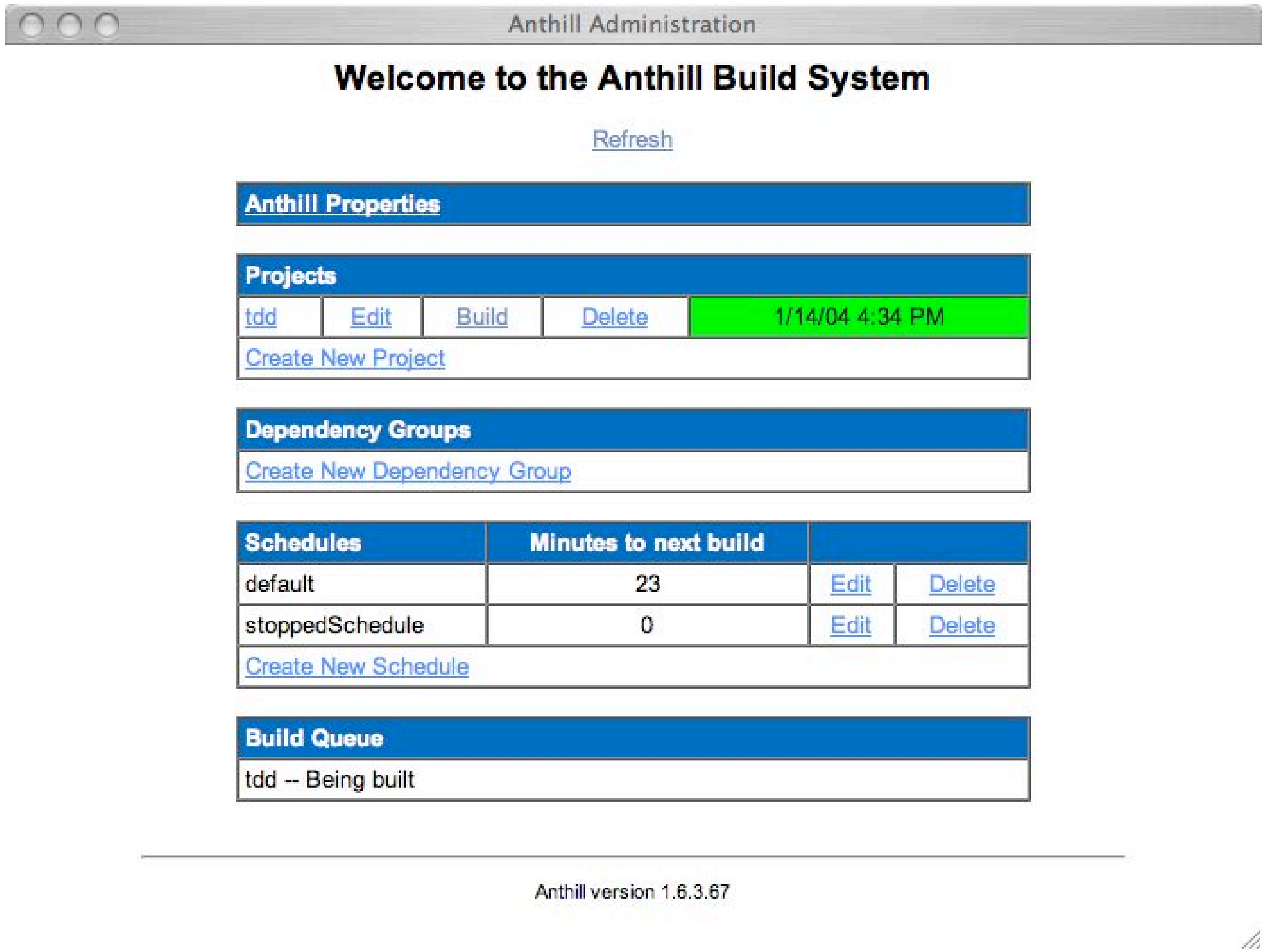
Packages

Name	Tests	Errors	Failures	Time(s)
tdd.shopping	**6**	**0**	**1**	**0.788**
tdd.shopping.web	24	0	0	18.467

Schedule The Build

- CruiseControl or Anthill
- Prevents integration hell
- Flushes out dependencies ("works on my machine")

Build Status



10. Buddy Up

- It’s more fun that way
- You can cover more ground
- Keep each other accountable
- Share experiences

11. Travel With a Guide

- You and your buddy may stumble into same pitfalls
- A guide can keep you from getting bogged down
- Customized training is best
- Find a good mentor

12. Practice, Practice, Practice

- Don't expect it to be easy
- Don't expect to test everything

- Write one good automated test a day
  - Next week you'll have 5!

- Run all your tests when you change code

And Before Long...

Summary

- Testing helps me write better software, faster
- You're already doing all this, manually
- Testing will improve your design skills
- What are you waiting for?

Travel With A Guide

Customized Training and Mentoring

http://clarkware.com
mike@clarkware.com

Articles and Books

- A Dozen Ways to Get the Testing Bug in the New Year
  http://today.java.net/pub/a/today/2004/01/22/DozenWays.html
- JUnit Primer
  http://clarkware.com/articles/JUnitPrimer.html
- Pragmatic Unit Testing
  by Dave Thomas and Andy Hunt
  http://pragmaticprogrammer.com
- Test-Driven Development By Example
  by Kent Beck (Addison-Wesley, 2002)
- Test-Driven Development: A Practical Guide
  by Dave Astels (Prentice Hall, 2003)
- Java Development With Ant
  by Erik Hatcher (Manning, 2002)

Discussion

Thanks and have fun!

(Your input is valuable to me.)