Fit & FitNesse

Mark Windholtz
ObjectWind Software Ltd

Agenda

- Testing Approaches
- FitNesse tour
- Fit Overview
- Standard Fixtures
- FitNesse
- FitNesse Macros & Features

Manual Testing

- Most expensive testing approach
  - Test takes time and personnel to execute
  - Valid only until the next environmental or code change occurs. And they occur constantly!
  - Error prone as testers get fatigued
  - Not executed often enough to help in development
  - Instructions get outdated and are misunderstood

Acceptance Testing

- Test Features at the end of production
- Pro
  - Bad defects don't ship, fix or delay (both cost)
- Cons
  - Minor defects too late to fix, so they ship
  - Too late to help during programming
  - Expensive and often Manual
  - Takes a long time

xUnit Testing

- jUnit (xUnit for Java) is industry standard
- Pros
  - Test-first design is very powerful
  - Allows for changing requirements
  - Produces better modular design
  - Reduces technical defects
- Cons
  - Most defects are in communication about specs
  - xUnit helps the trees but not the forest

Fit & FitNesse

- Fit
  - Write tests for features directly from the requirements
  - Test high-level and written as html tables
  - Domain Expert can specify fitness criteria
  - Domain Experts can verify fitness criteria
- FitNesse
  - Tool to write and organize fit tests
  - Wiki based for Collaborative web building

Event Scheduling Calendar

- An Event occurs on a date
- An Event belongs to a group
- Add an event to the calendar
  - At startup, should be zero events
  - Create an event
  - Check that there is now one event

CalendarTests.

AddEventTest

[.CalendarTests] [.FrontPage] [.RecentChanges]

fitnesse.FitFilter

Add an event with a group name.

fit.ActionFixture		
start	xpcinci.ECalendarFixture	
check	countOfEvents	0
enter	eventName	xp-cinci
enter	eventGroup	programming
enter	eventDate	04Mar2003
press	create	
check	countOfEvents	1
check	countOfGroups	1
check	groupsTop	programming

Add another event with same group name

Green

AddEventTest

Note: Ouput from Standard Error was captured during execution. You may view it by visiting the ErrorLog
[.CalendarTests] [.FrontPage] [.RecentChanges]
fitnesse.FitFilter

Add an event with a group name.

fit.ActionFixture		
start	xpcinci.ECalendarFixture	
check	countOfEvents	0
enter	eventName	xp-cinci
enter	eventGroup	programming
enter	eventDate	04Mar2003
press	create	
check	countOfEvents	1
check	countOfGroups	1
check	groupsTop	programming

Add another event with same group name

Fail

AddEventTest

Note: Ouput from Standard Error was captured during execution. You may view it by visiting the ErrorLog
[.CalendarTests] [.FrontPage] [.RecentChanges]
fitnesse.FitFilter

Add an event with a group name.

fit.ActionFixture		
start	xpcinci.ECalendarFixture	
check	countOfEvents	0
enter	eventName	xp-cinci
enter	eventGroup	programming
enter	eventDate	04Mar2003
press	create	
check	countOfEvents	42 *expected* 1 *actual*
check	countOfGroups	1
check	groupsTop	programming

suite

CalendarTests

Note: Ouput from Standard Error was captured during execution. You may view it by visiting the ErrorLog

fitnesse.fixtures.RecursiveAllFiles	
AddEventTest	8 right, 1 wrong, 0 ignored, 0 exceptions
QueryByDateTest	6 right, 0 wrong, 0 ignored, 0 exceptions
UiCalendarTest	0 right, 0 wrong, 0 ignored, 4 exceptions

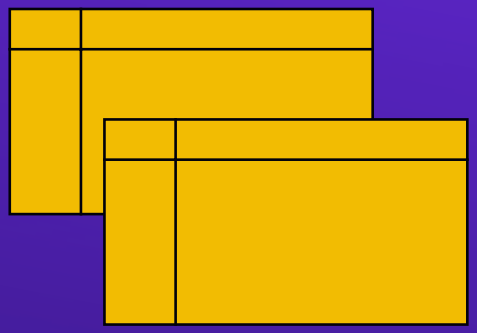
fit.Summary	
counts	1 right, 2 wrong, 0 ignored, 0 exceptions
counts run	14 right, 1 wrong, 0 ignored, 4 exceptions
run date	Fri Mar 14 15:42:42 PST 2003
run elapsed time	0:01.42

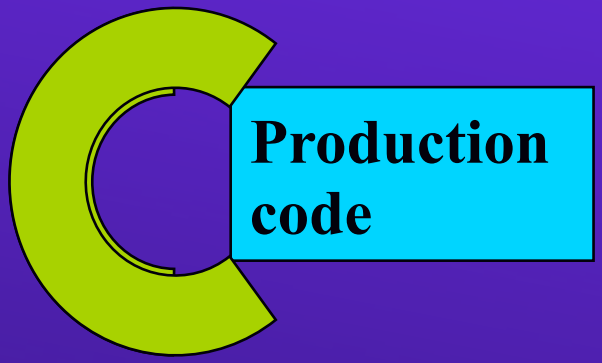
Contents

Fit: Overview

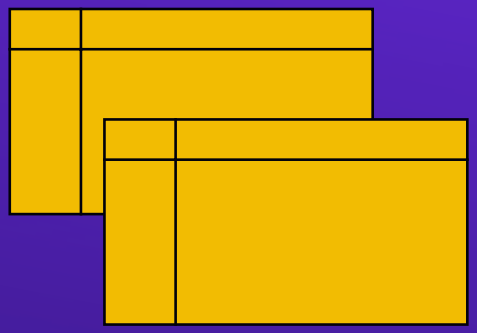
Tests As Html Tables

Fixture to "hold" real code





Web UI Testing



Testing the UI

- GUI is most unstable component
- Minimize Logic in the UI with solid OO design
- Minimize things that can break
- Target UI tests on only what can break
- Test business logic directly elsewhere
- Screen transitions?
- Java Script? (see above)
- Good OO design minimizes Logic in UI
- All that's left is the aesthetics

UI Testing libraries

- HtmlFixture
  - Link: TestUiCalendarDayNames
- IeFixture
  - Uses COM to direct a real IE browser
- HttpUnit
  - Allows low level control
- HtmlUnit
  - Page level control
- jWebUnit / jWebUnitFixture
  - (under development) high level control

Testing The Business Model

Inventory

Party

Location

Address

Address

Vendor

Destination

Trip

Carrier

Load

Carrier

Load

Palette

Item

Item

Item

Item

Palette

Produce

Produce

Produce

Produce

Palette

Item

Item

Item

Item

Business Domain Testing

- Business language
  - Not tech talk
- Focus on testing the business rules
  - Not loop counters and exception catching
  - Low level stuff is better tested by unit tests
- Scalable
  - Tests scale when they remain atomic
  - No dependencies between test order
  - UI driven testing is more Fragile & Expensive
    - Need to keep windows in sequence.
    - Full Database set-up

Standard Fixtures

- Column Fixture
  - *Each row loads a data structure and then invokes functions upon it.*
- Row Fixture
  - *Each row is a query into an array of objects*
- Action Fixture
  - *Write a script that emulates a user interface.*
- Summary Fixture
  - *Report of all tests on a page*

EventBuild is a Column Fixture

- Column Fixture
  - Operates on a Domain Objects directly
  - Each row loads a data structure and then invokes functions upon it.
  - Uses Domain methods and instance variables
  - Often used for test object creation

Add an event in Feb and March

ft.domain.EventBuild			
name	group	when	created()
xp-cinci	prog	time (daynum 09)(month 01)(year 2004)	true
cinjug	prog	time (daynum 13)(month 02)(year 2004)	true

Eg.Division

- Make sure division that works with positive and negative numbers

eg.Division		
numerator	denominator	quotient()
1000	10	100.0000
-1000	10	-100.0000
1000	7	142.85715
1000	.00001	100000000
4195835	3145729	1.3338196

```
public class Division extends ColumnFixture
{
        public float numerator;
        public float denominator;
        public float quotient() {
        return numerator / denominator;
        }
 }
```

← **Famous Pentium Bug**

Display is a Row Fixture

- Row Fixture
  - Operates on a group of Domain Objects
  - Each row is the data of an object
  - Often used to display results of a query
    - Query either built into the fixture
    - Or preserved in a known static location

Display last Query

ft.domain.Display		
name	groupName()	date()
xp-cinci	prog	09Jan2004
non-group *missing*	prog	09Jan2004

ValueObjects

- An Important Idea in Object Oriented Modeling
- like numbers, dates, monies
- Small objects which are used widely

Row Fixture Example

eg.music.Display					
title	artist	album	year	time()	track()
Scarlet Woman	Weather Report	Mysterious Traveller	1974	5.72	6 of 7
American Tango	Weather Report	Mysterious Traveller	1974	3.70	2 of 7

ActionFixture

- Like a Control panel
- *press* buttons that have particular names
- *enter* values into registers that have certain names.
- *check* the values of named meters

fit.ActionFixture		
start	fitnesse.fixtures.CountFixture	
check	counter	0
press	count	
check	counter	1
press	count	
check	counter	2
enter	counter	5
press	count	
check	counter	6

ECalAction is an ActionFixture

- Action Fixture
  - Write a script that emulates a user interface
  - Add a subclass and Use it directly
  - Define your commands in subclass

Query January Events

fit.ActionFixture		
start	ft.domain.ECalAction	
enter	week	2004,02

Example ActionFixture

fit.ActionFixture		
start	eg.music.Browser	
enter	library	Source/eg/music/Music.txt
check	total songs	37

fit.ActionFixture		
enter	select	1
check	title	Akila
check	artist	Toure Kunda
enter	select	2
check	title	American Tango
check	artist	Weather Report
check	album	Mysterious Traveller
check	year	1974
check	time	3.70
check	track	2 of 7

Summary Fixture

- Displays summary of results of Page
- Add it to TearDown and forget it.

  | fit.Summary |

fit.Summary	
counts	8 right, 0 wrong, 0 ignored, 0 exceptions
run date	Sun Oct 19 14:25:39 PDT 2003
run elapsed time	0:00.32

Putting it all together

- BOC: Build-Operate-Check Pattern
  - Link to locally running FitNesse server
  - Testing the eCal application

  CalendarTests.TestQueryByWeekNumbers

FitNesse File Structure

- FrontPage
  - FitNesse
    - UsersGuide
    - (Self tests)
  - CalendarPaths (macros defined here)
    - ClassPath (classpath set here)
    - CalendarTests (root of all tests)
      - PageFooter
      - PageHeader
      - SetUp
      - TearDown
      - TestMoveScheduler
      - TestMoveSchedulerYearBoundry

Test Suite

- Executes all tests in the Sub-Wiki (Tree of Pages)
- **SetUp** and **TearDown** pages invoked for each page of the suite.
- To wrap an entire suite, define the operations on pages named **SuiteSetUp** and **SuiteTearDown**.

```
SuiteSetUp
      SetUp   TestOne     TearDown
      SetUp   TestTwo     TearDown
      SetUp   TestThree   TearDown
SuiteTearDown
```

Editing FitNesse Pages

- Simple syntax
- Tables can be created very easily.
- Vertical stroke as the first character of the line, and separate each table cell with it:

```
|Alpha|
|Beta|gamma|Delta|
|1|2|3|
```

Alpha		
Beta	gamma	Delta
1	2	3

Link: FitNesse.MarkupTable

Headers

Headers are created by prefixing a line with !1 or !2 or !3

Markup Text	Displayed as
!1 Title	**Title**
!2 Header	**Header**
!3 Second Header	**Second Header**

FitNesse Macro Language

- !contents
  - Expands into links to sub pages
- !include
  - Include other pages
  - Handy when building test objects
- !path
  - Used in ClassPath page to define a path to class and jar files
- !define
  - Defines macros that can be referred to in sub pages

Example of Macros

```
!2 Macros defined on this page (click Edit)
!define xpcinci (http://localhost:9090/ECal)
 * defined xpcinci (${xpcinci})
!define eCalDir {d:\eclipse21\workspace\ecal}
 * defined eCalDir ${eCalDir}
!define I {\}
 * defined path separator: ${I}
   * "'\ for Windows,  / for Unix'"

!2 Contents
!contents

Link:  CalendarPaths
```

Macros defined on this page (clicl

- defined xpcinci (http://localhost:9090/ECal)
- defined eCalDir d:\eclipse21\workspace\ecal
- defined path separator: \
  - \ **for Windows, / for Unix**

Contents

- CalendarTests
- ClassPath
- ConfigNotes

Installation

- Download from www.Fitnesse.org
- Unzip into a directory
- Cd to directory and type:

  C:> run

- Open a web browser on:

  http://localhost

Note: You can pick a different port with the *-p* command line option

Command line or Ant script

```
<exec executable="java"
      resultproperty="fit.result">

  <arg line="-cp ${lib}/fitnesse.jar
       fitnesse.TestRunner
       http://localhost/CalendarTests?suite="/>

</exec>
```

Additional FitNesse Features

- Page Versioning
- Recent Changes
- Search for keyword
- Search for page references
- Refactor (rename & delete)
- Virtual Wiki
  - Run local code still under development
  - Using a central set of shared Test pages
  - Helpful in testing code before check-in

Big Picture Stuff

- Plays well with other Best Practices
  - Automated build systems
  - OO Design Patterns
  - jUnit
  - Domain Driven Design
  - Iterative development

Summary

- FitNesse allows for Testing of UIs
  - However, UI testing is fragile and late
- FitNesse also allows Domain Object Testing which is not fragile
- Provides migration path from UI to Domain testing
- FitNesse allows human readable comments to be mixed in with test tables
- And allows easy interpretation by domain experts

Getting Started

1. FitNesse as a Wiki
   - Excellent group communication device
2. FitNesse testing of new project
   - Concurrent with requirements gathering
   - And Domain design
3. FitNesse testing existing projects
   - UI driven with IeFixture and other UI techniques and fixtures
4. FitNesse 2-day Workshop or Mentoring

Perfection: Fitness Testing