Selenium WebDriver

test automation for web applications

Day one

Introduction

Who am I?

Who are you?

Sign the attendant list

Three questions

Close your eyes

Who is comfortable speaking English?

Who is not comfortable speaking English?

Who understood my questions?

Course layout

Block of about two hours

Theroy

first

Hands on laborations

Instructions in PDF on USB

Own work

Pairs

Help each other

Share the keyboard

Alone if the pair doesn't work

Example site

http://selenium.thinkcode.se

Local

Selenium

Browser automation

Can be used for tests

Doesn't have to be used for testing

Tests

Many frameworks

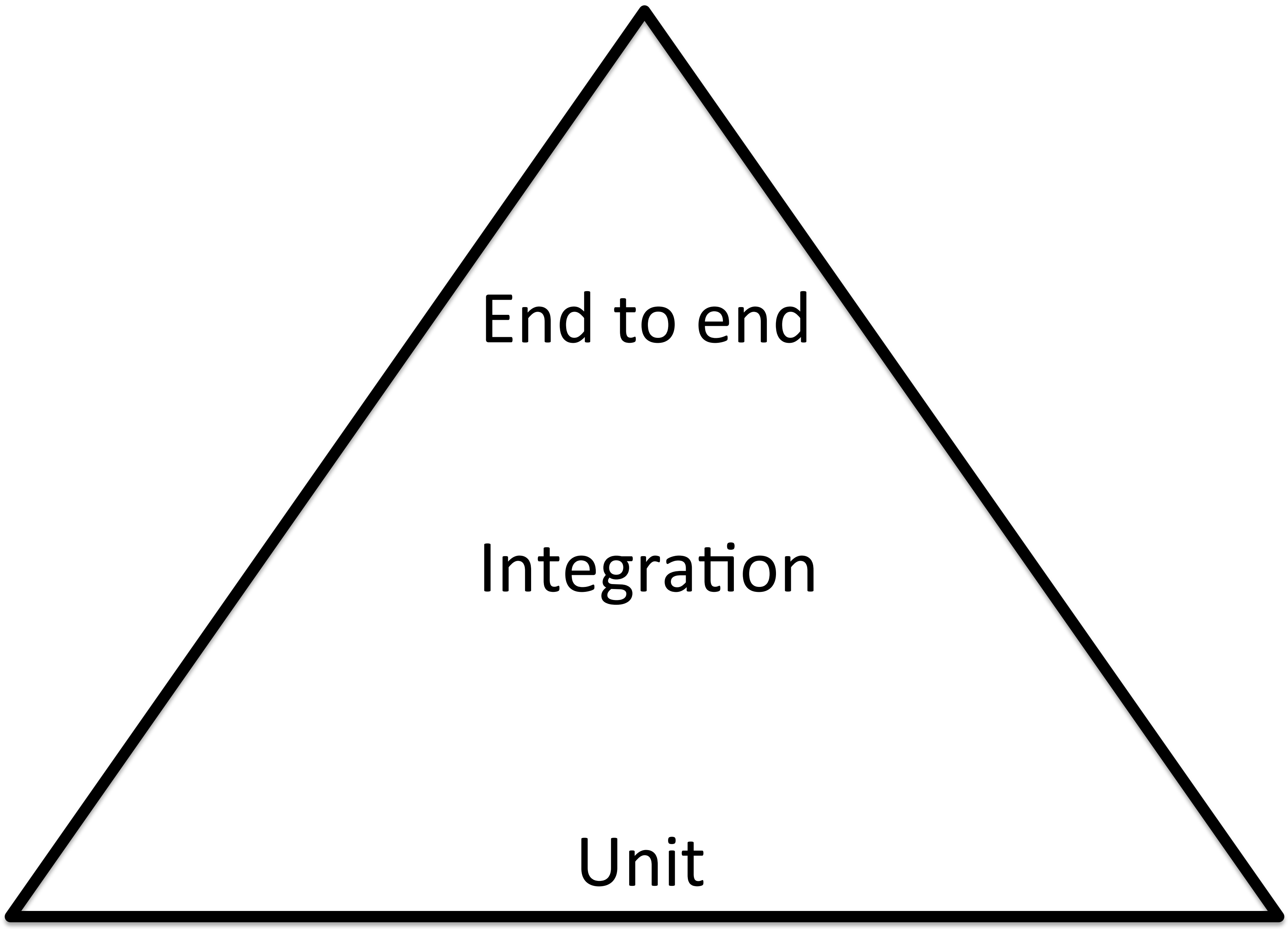
Test NG

Spock

JUnit

Others?

Many levels



Testing pyramid

End to end

From the user interface

Very slow

It takes forever to start a browser and do something

Very fragile

Lots of reasons for them to break

The application didn't start

The database
is broken

The user interface is missing a vital part

and many more...

Impossible to test all permutations

10 entrances from the UI

5 paths through controllers

7 paths through the model

10 * 5 * 7 = 350 paths

Integration

Verify that components works together

Is the database properly configured

Is the que started?

Slow

Fragile

Many reasons for failing

Unit

Small pieces of functionality at a low level

Very fast

Stable

Only one reason for failing

The failures are easy to understand

Use for verifying all use cases for each implementation

Possible to verify all permutations

Test automation

Why?

Repeatable

Same execution every time

Catch silly mistakes early

Bugs

Fast

Time to market

We want to deploy this site now!

Scalable

Add more hardware

Test in parallell

Fun!

It is programming

Programming is fun

Manual testing

Doesn't scale

Error

prone

Not easily repeatable

Boring

Stupid

Programming

You can't automate anything seriosly without programming

Software development

Good tests

Must be easy to

Understand

Maintain

Change

Extend

Selenium echo system

Selenium IDE

Firefox plugin

Record and replay

Run local in your Firefox browser

WebDriver

A programming framework

Local in any browser you have installed

Open source

Anyone can use it

Many programming languages supported

We will use Java

W3C

standard

Not done, but on its way

Every browser that follow W3C MUST support WebDriver

Selenium hub

Automate a browser remote

Enable us to do cross browser testing

Browser

Operating system

Remote WebDriver

Same API as regular WebDriver

SauceLabs

http://seleniumhq.org

Selenium IDE

Install

Record a scenario

http://selenium.thinkcode.se

Find an element and verify the text

Hello
world

Fill out a form

Change password

What should you test?

Testing theatre

Security theatre

Not aiming for security

Aim to make people feel safe

Security chek on airports

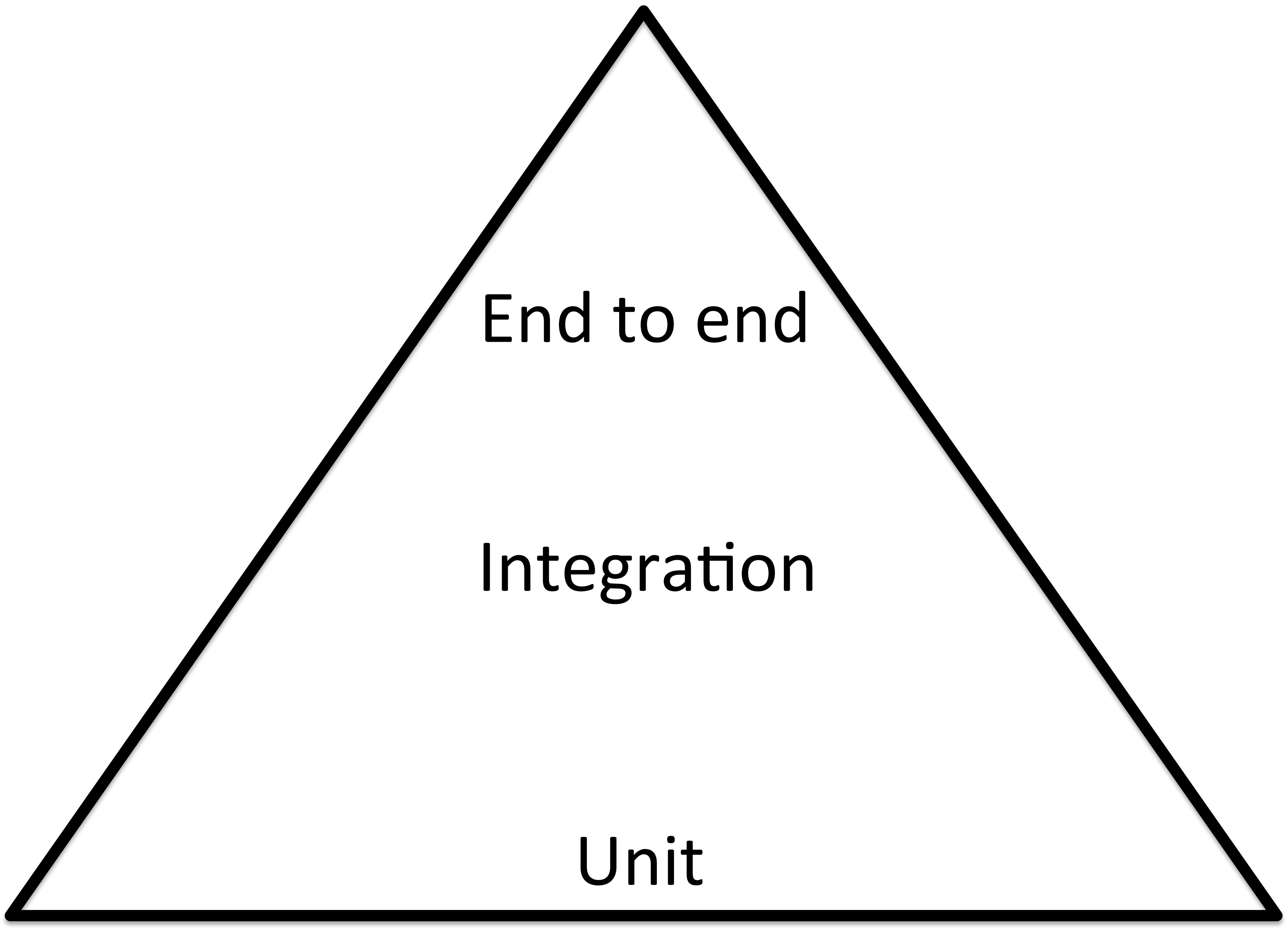
Why do they take you water bottle?

And place it behind their back?

If it dangerous, remove it far away so it can't hurt you

Not everything

Enough to fell safe



Testing pyramid

Do not test everything through the user interface

Test at the right level

Can you login

End to end

Can you fail to log in?

End to end

Can you do a purchase?

At least place something in the shopping bag

End to end

Connect to the database

Integration

Password algorithm

Unit level

You must be in control

Collaborators

Mock or Stub

Database

Know about the content

Decide what the content should be

This makes End to end complicated

What should you automate?

Any test that should be repeated

But at the right level...

What should you not automate?

Expensive tests

Buying using a credit card

Probably enough to do one manual test

Setup a fake server that mimics the payment service

http://wiremock.org

One off

If it truly exists

Automate it the second time

Java environment

Get a sample project up and running

GitHub

Clone

USB

Use zip

Maven

Must be installed

Gradle

Can used without installation

Editor

Notepad

IntelliJ IDEA

Community edition

Available
on USB

Eclipse if you know how

Run the included unit test from a command line

Change it
and see it fail

Write your own JUnit test

A test that calls a method and verifies the result

A Hello world

Run the test site local

Executable jar

```
java -jar web-samples-1.0.0.jar
```

http://localhost:8080

USB

Clone and build

```
mvn clean package
```

First WebDriver test

Export the code from Selenium IDE

Use the exported code in your unit test

The code
is ok

Not pretty

Ok to start with

Good for finding elements

Browser tools

Firebug

Firepath

Other favorites?

First handmade WebDriver test

Hello world in the example app

Without page object

With page object

Handmade is better

You have better control

You know what you did

Slower?

Maybe, but not in the long run

Page objects

Separation of navigation logic and test

Only need to change the page object when the page changes

The tests are the same as long as the logic is the same

Lazy

Hard to solve a hard problem

Better to solve many small problems

One class per page

Hides all the page functionality

Hide the page navigation

Simplifies the tests

The test focus on WHAT the application should do, not HOW

Supply the browser

Through the constructor

Allow you to use the same page object with different browsers

Verify that the page is the right page

Check
title

Something else that can assert that you are at the right page

Throw an exception if you can't verify the page

asserThat

Signals through exception

Fill out a form and verify the result

Reset a password

WebDriver API

Small

13
methods

close()
findElement(By by)
findElements(By by)
get(java.lang.String url)
getCurrentUrl()
getPageSource()
getTitle()
getWindowHandle()
getWindowHandles()
manage()
navigate()
quit()
switchTo()

Javadoc

http://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/WebDriver.html close()

Close the current window, quitting the browser if it's the last window currently open.

Doesn't always quit the browser

findElement(By by)

Find the first WebElement using the given method.

Return WebElement

findElements(By by)

Find all elements within the current page using the given mechanism.

Return java.util.List<WebElement>

get(java.lang.String url)

Load a new web page in the current browser window.

Doesn't work with relative urls

getCurrentUrl()

Get a string representing the current URL that the browser is looking at.

getPageSource()

Get the source of the last loaded page.

getTitle()

The title of the current page.

getWindowHandle()

Return an opaque handle to this window that uniquely identifies it within this driver instance.

getWindowHandles()

Return a set of window handles which can be used to iterate over all open windows of this WebDriver instance by passing them to switchTo().WebDriver.Options.window()

manage()

Gets the Option interface

Managing stuff you would do in a browser menu

navigate()

An abstraction allowing the driver to access the browser's history and to navigate to a given URL.

```
quit()
```

Quits this driver, closing every associated window.

switchTo()

Send future commands to a different frame or window.

Locators

The tool to locate an element on a page

Static methods in

By

8
methods

By.ByClassName
By.ByCssSelector
By.ById
By.ByLinkText
By.ByName
By.ByPartialLinkText
By.ByTagName
By.ByXPath

Javadoc

http://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/By.html

By.id(java.lang.String id)

The preferred locater if the IDs are unique

By.name(java.lang.String name)

By.xpath(java.lang.String xpathExpression)

By.cssSelector(java.lang.String selector)

Finds elements via the driver's underlying W3 Selector engine.

By.linkText(java.lang.String linkText)

Complicated if your site supports many langugaes

By.partialLinkText(java.lang.String linkText)

By.tagName(java.lang.String name)

By.className(java.lang.String className)

Finds elements based on the value of the "class" attribute.

findElement(SearchContext context)

Find a single element.

Return WebElement

You can search in many steps by nesting calls

findElements(SearchContext context)

Find many elements.

Return java.util.List<WebElement>

WebElement

The representation of every element on a page

Largest

http://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/WebElement.html clear()

If this element is a text entry element, this will clear the value.

click()

Click this element.

findElement(By by)

Find the first WebElement using the given method.

findElements(By by)

Find all elements within the current context using the given mechanism.

getAttribute(java.lang.String name)

Get the value of a the given attribute of the element.

getCssValue(java.lang.String propertyName)

Get the value of a given CSS property.

getLocation()

Where on the page is the top left-hand corner of the rendered element?

getSize()

What is the width and height of the rendered element?

getTagName()

Get the tag name of this element.

getText()

Get the visible (i.e. not hidden by CSS) innerText of this element, including sub-elements, without any leading or trailing whitespace.

isDisplayed()

Is this element displayed or not? This method avoids the problem of having to parse an element's "style" attribute.

isEnabled()

Is the element currently enabled or not? This will generally return true for everything but disabled input elements.

isSelected()

Determine whether or not this element is selected or not.

sendKeys(java.lang.CharSequence... keysToSend)

Use this method to simulate typing into an element, which may set its value.

```
submit()
```

If this current element is a form, or an element within a form, then this will be submitted to the remote server.

Check boxes

http://selenium.thinkcode.se/selectColor.html

Radio buttons

http://selenium.thinkcode.se/selectBeverage.html

Looking forward

Four rules of simple design

WedDriver
util

Select

Datadriven tests

Slow elements

Examination

Work on your own and verify a test page

Cross browser

Test your own site

Continuous integration

Simpler specifications

Executable specifications

Retrospective

Did you miss anything today?

Do you want more of something?

Was the tempo ok?

What are your expectations for tomorrow?

Anything else?

Day two

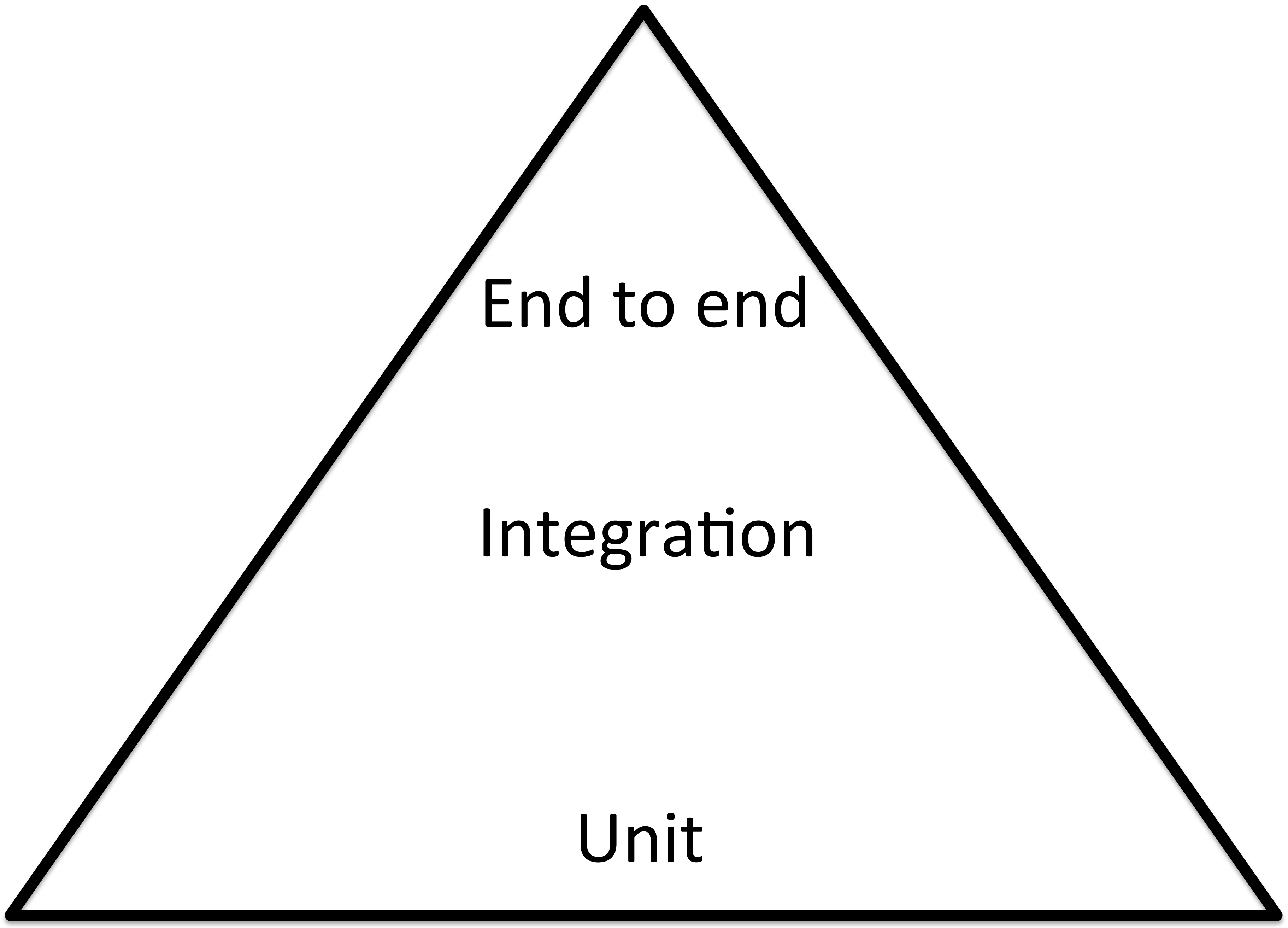
Sign the attendant list

Recap yesterday

Selenium echo system

- Ide
  - Firefox plugin
- WebDriver
  - Programming api
- Selenium hub
  - Remote
  - Different browsers
  - Different operating systems

Testing pyramid



Test things at the right level

End to end

- Verify that the system is alive
- Most crucial flow
  - Buy a product
  - Book a trip
- Very slow
- Lots of reasons to fail

Integration

- Slow
- Many reasons to fail
- Doesn’t scale

Unit

- Fast
- Only one reason to fail
- Algorithms
- Password

Testing theater

- Verifying important parts in the system
- Not for satisfying a managers metric

WebDriver

- close()
- findElement(By by)
- findElements(By by)
- get(java.lang.String url)
- getCurrentUrl()
- getPageSource()
- getTitle()
- getWindowHandle()
- getWindowHandles()
- manage()
- navigate()
- quit()
- switchTo()

Locators

- By.ByClassName
- By.ByCssSelector
- By.ById
- By.ByLinkText
- By.ByName
- By.ByPartialLinkText
- By.ByTagName
- By.ByXPath

WebElement

- clear()
- click()
- findElement(By by)
- findElements(By by)
- getAttribute(java.lang.String name)
- getCssValue(java.lang.String propertyName)
- getLocation()
- getSize()
- getTagName()
- getText()
- isDisplayed()
- isEnabled()
- isSelected()
- sendKeys(java.lang.CharSequence... keysToSend)
- submit()

Page objects

- Separate navigation and logic
- Clearer and easier tests
- Supply the browser
- Verify the correct page

The four rules of simple design

- Test should always pass
- Express intent
- No duplication
- Small

Kent Beck,

Extreme Programming Explained, 1999

Duplication

- In every test
  - Define the browser
  - Define the baseUrl
- Create a test helper
  - Static methods

Drop down

- Select condiment
  - Sugar
  - Milk
  - Sugar & Milk
- org.openqa.selenium.support.ui.Select
- http://selenium.googlecode.com/git/docs/api/java/org/openqa/selenium/support/ui/Select.html
- **http://selenium.thinkcode.se/selectCondiment.html**

Test reports

- Maven
  - mvn surefire-report:report
  - file:///Users/tsu/tmp/selenium/target/site/surefire-report.html
- Gradle
  - Part of the regular build
  - file:///Users/tsu/tmp/selenium/build/reports/tests/index.html

Data driven tests

- Change the password for many persons
- A list of persons
  - Loop over the list
- Parameterized JUnit
  - A list of persons
  - Run a new test for each person

- Bad example!
  - This test should be done at unit level

Screen shoot on failure

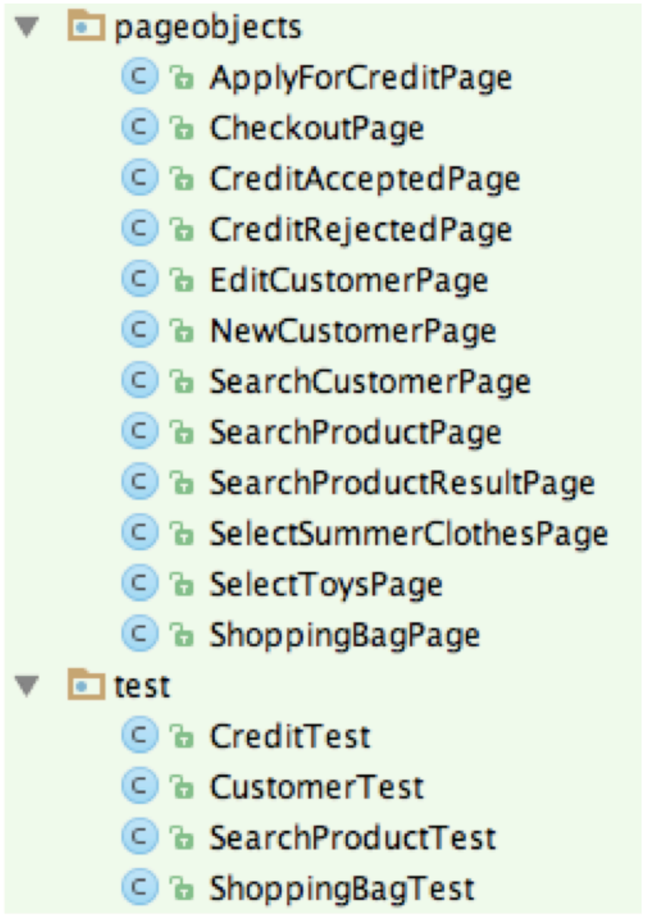
- Save an image when a test fails
- JUnit rule

Slow response

- Thread.Sleep
  - Will always sleep
  - Too slow
- Wait
  - WebDriverWait
  - FluentWait

Test organization

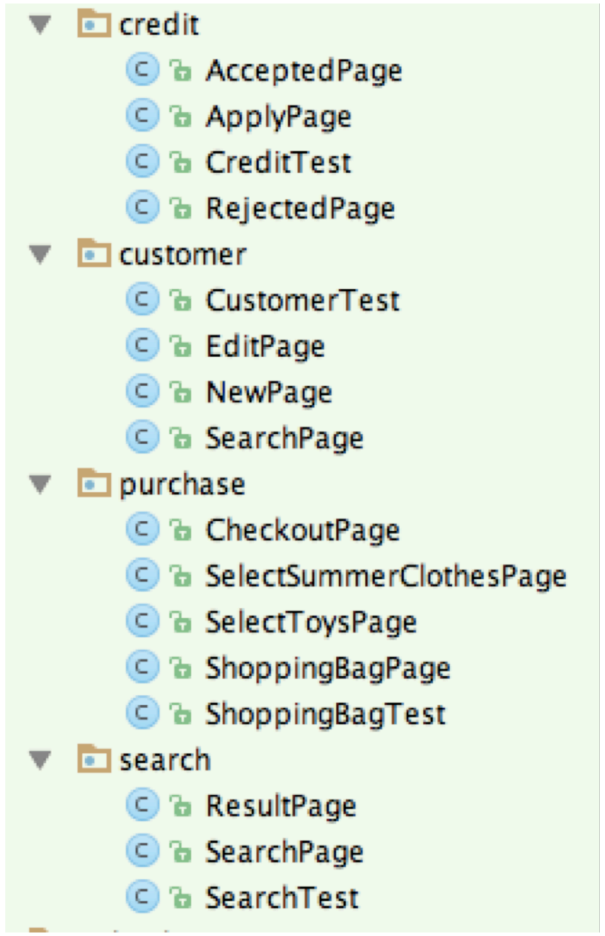
By technology



pageobjects
- ApplyForCreditPage
- CheckoutPage
- CreditAcceptedPage
- CreditRejectedPage
- EditCustomerPage
- NewCustomerPage
- SearchCustomerPage
- SearchProductPage
- SearchProductResultPage
- SelectSummerClothesPage
- SelectToysPage
- ShoppingBagPage

test
- CreditTest
- CustomerTest
- SearchProductTest
- ShoppingBagTest

By functional area



credit
- AcceptedPage
- ApplyPage
- CreditTest
- RejectedPage

customer
- CustomerTest
- EditPage
- NewPage
- SearchPage

purchase
- CheckoutPage
- SelectSummerClothesPage
- SelectToysPage
- ShoppingBagPage
- ShoppingBagTest

search
- ResultPage
- SearchPage
- SearchTest

Test organization

- Separate on functional areas
- Do not separate on technology
  - A new developer want to find the relevant test and page objects fast
- Page objects in one package
- Test in another package
- All tests and page object for a certain area should live in the same package

Examination

- Alone
- Buy currency
- http://selenium.thinkcode.se/buyCurrency.html
- Must use a Page Object
- Do it manually first
  - Can't automate anything you can't do manually

Evaluation

- Divide into small groups of three
- Discuss each solution in the small group
- Was a page object used?
- Did the page object verify that it was on the right page?
- How did the page object get hold of the browser?
- Was the test easy or hard to read?
- Why?

Evaluation

- Present the solutions for the whole group
- Good things
- Things to improve
- A few minutes per group

- I will not give any marks

Time for this: ~45 minutes

Cross browser

- Parameterized JUnit
  - Serial
- Hello world
- Reused the page object
- SauceLabs
  - Serial
  - Parallel

Other usage

- Semantic monitoring
  - Is your site alive?
  - Can you do a purchase?
- Automate tasks
  - Added users in Hybris at H&M
  - Crashed a poll

Course evaluation

- What did you learn?
- How will you be able to use your new knowledge?
- Feedback
  - Course material
  - Lectures
- Did you miss some content?
- Was something too
  - Easy?
  - hard?
- Anything that must be changed?
- Would you recommend this course?
- Would you recommend Mozaicworks?

Thomas Sundberg

Stockholm, Sweden

Think Code AB

thomas@thinkcode.se

@thomassundberg

Blog: https://thomassundberg.wordpress.com/

Code:

https://github.com/tsundberg/selenium-test-automation/tree/timisoara-june-2015