UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

SENG365 Web Computing Architecture: Testing Web Clients

Austen Rainer
Course Coordinator
austen.rainer@canterbury.ac.nz
202, Erskine Building

Teaching team

Tom Young
The Mastermind
Lecturer, Term 3

Ashley Williams
The Explosives Expert
Tutor

James Watson
The Getaway Driver
Tutor

Austen Rainer
The Enforcer
Course Coordinator &
Lecturer, Term 4

News

- Arrangements for today (Thursday 28 September)
  - 9am – 10am: lecture e.g. on testing
  - 10am – 11am: lab on automated front-end testing
  - (Ensure you have downloaded and completed step 3 BEFORE the lab)
- Compulsory lab on Thursday 19 October
  - The lab on testing is to help you prepare for the lab on 19 October
- Return of Mid-semester test
  - In lab, or at the end of this lecture depending on time

Assignment 2

- Who has started the assignment?
- Who has progressed on the labs?
- Who are learned lessons from Assignment 1?

The story so far…

Lectures

Term 3: What happens on the server side?

Term 4: What happens on the client side?

- Client – Server architecture
- HTML, CSS, & JavaScript
- Single Page Applications & Vue.js

Labs

- Introduction to HTML, CSS, JavaScript
- Introduction to Vue.js

This week

In the lecture

- Testing
- Automated testing of web client
- Selenium

In the labs

- Introduction to Vue.js
- Automated testing using Selenium

Testing

Brief overview to testing

- Lots of different types of test
  - Unit tests, acceptance tests, regression tests etc
- Lots of different kinds of system to test
  - Safety-critical, embedded systems, real-time systems etc
- Particular challenges for testing web applications, for example:
  - Different browsers (FireFox, Chrome)
  - Different versions of browser (IE6 vs IE8)
  - Differences in versions of HTML, ECMAScript (JavaScript)
  - Differences in APIs e.g. how XHRs are handled by different browsers
  - Differences in the handling of the DOM, JavaScript etc.
  - Differences in libraries and versions (e.g. jQuery)

Automated testing

Advantages of automating testing

- Automation means you can offload testing to machines (rather than rely on humans)
- (More) frequent testing e.g. regression testing
- Quick and regular feedback to developers
- Virtually unlimited iterations of test case execution
- Support for Agile and extreme development methodologies
- Disciplined documentation of test cases
- Customized defect reporting
- Finding defects missed by manual testing
  - Humans get tired etc.

When not to automate (and manually test instead)

- If the user interface is rapidly changing
  - e.g. HTML elements are changing
  - Will need to keep changing the tests to match the interface
- Tight timescales
  - Don’t have time to develop the tests

WebDriver API

https://www.w3.org/TR/webdriver/

WebDriver API

- Intended to enable web authors to write tests that automate a user agent from a separate controlling process
  - May also be used to allow in-browser scripts to control a (possibly separate) browser
- Provides a set of interfaces to:
  - discover and manipulate DOM elements in web documents; and
  - to control the behavior of a user agent (a browser).
- Provides a platform-neutral and language-neutral wire protocol as a way for out-of-process programs to remotely instruct the behavior of web browsers.
- Forms part of the W3C Web Testing Activity

Selenium

Selenium-WebDriver | Selenium 2.0

- Replaces Selenium Remote Control
  - Webdriver has a very different approach to testing (see next page)
- Automates a browser
  - Use to automate testing
  - Use to automate routine web tasks e.g. some admin task
- Selenium WebDriver
  - Drive the browser (automatically) the way a user would
    - Automate what you want the user to do
    - Automate what you want the user not to do
      - Automate unintentional behaviour, accidental behaviour, stupid behaviour, risky behaviour (e.g. security attacks?)

Selenium-WebDriver

- Developed to better support automation of **dynamic web pages**,
  - Dynamic pages: elements of a page may change without the page itself being reloaded. (What is a webpage?)
  - Single Page Applications generate dynamic web pages
- Webdriver relies on the browser's built-in (native) support for automation
  - You'll need an update-to-date browser
  - Harder to automatically test older browsers

Cross-browser testing

“When we say “JavaScript” we actually mean “JavaScript and the DOM”. Although the DOM is defined by the W3C each browser has its own quirks and differences in their implementation of the DOM and in how JavaScript interacts with it. HtmlUnit has an impressively complete implementation of the DOM and has good support for using JavaScript, but it is no different from any other browser: it has its own quirks and differences from both the W3C standard and the DOM implementations of the major browsers, despite its ability to mimic other browsers.”

http://www.seleniumhq.org/docs/03_webdriver.jsp#htmlunit-driver

Implication: automate your testing on different browsers

Example from lab on automated testing

```
const webdriver = require('selenium-webdriver'),
    By = webdriver.By,
    Key = webdriver.Key,
    until = webdriver.until;

const driver = new webdriver.Builder().forBrowser('firefox').build();


driver.get('http://google.co.nz')
    .then(_ =>
        driver.findElement(By.name('q')).sendKeys('university of canterbury',
        Key.RETURN))
    .then(_ => driver.wait(until.titleIs('university of canterbury - Google Search'),
    1000))
    .then(_ => driver.quit());
```

Example from lab on automated testing

```
const webdriver = require('selenium-webdriver'),
    By = webdriver.By,
    Key = webdriver.Key,
    until = webdriver.until;

const driver = new webdriver.Builder().forBrowser('firefox').build();


driver.get('http://google.co.nz')
    .then(_ =>
        driver.findElement(By.name('q')).sendKeys('university of canterbury',
        Key.RETURN))
    .then(_ => driver.wait(until.titleIs('university of canterbury - Google Search'),
    1000))
    .then(_ => driver.quit());
```

Summary of what you can do

- Fetch a page:
- Locate a UI (DOM) element
- Get text values
- User input
- Move between windows and frames
- Popup dialogs
- Navigation and history (may be harder in SPA)
- Cookies
- Drag and drop
- Check out the details are:
  - http://www.seleniumhq.org/docs/03_webdriver.jsp#introducing-the-selenium-webdriver-api-by-example

Navigating a page using WebDriver

- To navigate to a page:
- driver.navigate().to(‘http://www.example.com’);
- But SPAs dynamically generate pages

Examples of what you can do

Fetch a page:

- `driver.get('http://www.google.com');`

Locate a UI (DOM) element

- By ID:
  - `var element = driver.findElement(By.id('someID'));`
  - `Compare with JavaScript getElementById();`

WebDriver vs the Selenium-Server

- You can use WebDriver without Selenium Server
  - We're using only WebDriver in the labs (not Selenium Server)
  - Browser and tests will all run on the same machine

There are reasons to use the Selenium-Server with Selenium-WebDriver.

- You are using Selenium-Grid to distribute your tests over multiple machines or virtual machines (VMs).
- You want to connect to a remote machine that has a particular browser version that is not on your current machine.
- You are not using the Java bindings (i.e. Python, C#, or Ruby) and would like to use HtmlUnit Driver

Further information

- http://www.seleniumhq.org/docs/03_webdriver.jsp#chapter03-reference

Next week

In the lecture

- More on SPAs
- TBC, but
  - Reactive programming
  - Progressive web apps
  - Other aspects of client-side applications
    - Session management
  - Performance
  - Design patterns e.g. MVVM

In the labs

- More on Vue.js
- Reactive programming lab

Automated testing lab

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

- Follow the lab instructions, available on Learn
- If there's a slow response on download of selenium-webdriver, use the local registry:
- To set to the local registry

```
npm set registry http://csse-s365.canterbury.ac.nz:4873
```

- To restore to the original registry (at the end of the lab):

```
npm set registry https://registry.npmjs.org
```

- Don't forget the GeckoDriver for FireFox

UNIVERSITY OF CANTERBURY
Te Whare Wānanga o Waitaha
CHRISTCHURCH NEW ZEALAND

SENG365 Web Computing Architecture: Testing Web Clients

Austen Rainer
Course Coordinator
austen.rainer@canterbury.ac.nz
202, Erskine Building