Taming Functional Web Testing with Spock and Geb

- Peter Niederwieser, Gradleware
- Creator, Spock
- Contributor, Geb

The Actors

Spock, Geb, Page Objects

Spock

> “Spock is a testing and specification framework for Java and Groovy applications. What makes it stand out from the crowd is its beautiful and highly expressive specification language. Thanks to its JUnit runner, Spock is compatible with most IDEs, build tools, and continuous integration servers. Spock is inspired from JUnit, RSpec, jMock, Mockito, Groovy, Scala, Vulcans, and other fascinating life forms.

Spock (ctd.)

- http://spockframework.org
- ASL2 licence
- Serving mankind since 2008
- Latest releases: 0.7, 1.0-SNAPSHOT
- Java + Groovy
- JUnit compatible
- Loves Geb

Geb

> “Geb is a browser automation solution.
>
> It brings together the power of WebDriver, the elegance of jQuery content selection, the robustness of Page Object modelling and the expressiveness of the Groovy language.
>
> It can be used for scripting, scraping and general automation — or equally as a functional/web/acceptance testing solution via integration with testing frameworks such as Spock, JUnit & TestNG.

Geb (ctd.)

- http://gebish.org
- ASL2 license
- Serving mankind since 2009
- Latest releases: 0.7.2, 1.0-SNAPSHOT
- Java + Groovy
- Use with any test framework
- Loves Spock
- First-class page objects

Page Objects

> “The Page Object pattern represents the screens of your web app as a series of objects.
>
> Within your web app's UI, there are areas that your tests interact with. A Page Object simply models these as objects within the test code. This reduces the amount of duplicated code and means that if the UI changes, the fix need only be applied in one place.

Demo: Testing Google Search

WebDriver

http://seleniumhq.org/projects/webdriver/

WebDriver

Successor to the Selenium project.

Also known as “Selenium 2”.

Sponsored and driven by Google.

Becoming a W3C standard.

http://dvcs.w3.org/hg/webdriver/raw-file/515b648d58ff/webdriver-spec.html

Cross-browser automation

Java based, with many language bindings.

```
import org.openqa.selenium.*;
import org.openqa.selenium.firefox.*;

WebDriver driver = new FirefoxDriver();
driver.get("http://google.com");
WebElement searchBox = driver.findElement(By.name("q"));
searchBox.sendKeys("webdriver");
driver.findElement(By.name("btnK")).click();
```




Mobile Browsers

Rapidly improving.

- iPad
- iPhone
- Android
- Blackberry

Can use real devices or emulators in most cases.

A headless driver based on PhantomJS (called GhostDriver) is in progress.

WebDriver

Pros:

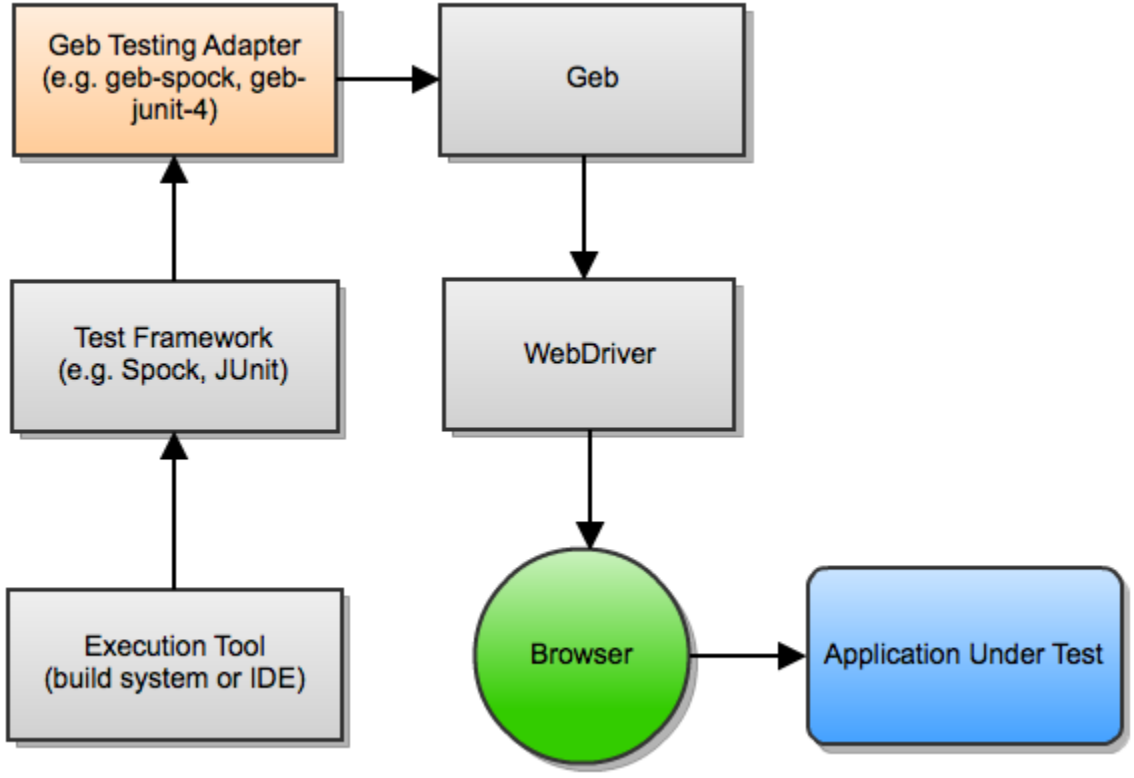
1. Proven solution for cross browser automation
2. Very active development
3. Stable API and feature set

Cons:

1. Verbose
2. Low level
3. Not a complete solution

Geb embraces the pros and solve the cons by building on top.

Architecture



WebDriver API

Geb sits on top of WebDriver.

Can access WebDriver API directly if needed.

Only WebDriver talks to the actual browser.

jQuery

http://jquery.com/

jQuery - write more, do less

jQuery provides an incredibly powerful API for navigating and selecting content.

```
$("div#footer").prev().children();
```

CSS based, a whole lot better than XPath.

Geb's inspiration

Geb features a “Navigator API” that is inspired by jQuery.

```
// This is Geb code, not jQuery JavaScript…

$("div#footer").previous().children()
```

Geb doesn't aim to emulate jQuery, it just borrows ideas.

Groovy

http://groovy-lang.org

Dynamic JVM Language

Groovy is…

- Compiled, never interpreted
- Dynamic, optionally typed
- 95% Java syntax compatible
- Concise, clear and pragmatic
- Great for DSLs

Geb & Groovy

Geb uses Groovy's dynamism to remove boilerplate, to achieve pseudo English code.

```
to GoogleHomePage
searchFor "Wikipedia"
assert resultName(0) == "Wikipedia"
resultLink(0).click()
at WikipediaPage
```

Conciseness = improved clarity & improved maintainability.

Navigator API

jQuery inspired content selection/navigation

The $() method

Returns a Navigator object.

General format:

```
$(«css selector», «index/range», «attribute/text matchers»)
```

Examples:

```
$("div") // all divs
$("div", 0) // first div
$("div", 0..2) // first three divs

// The third section heading with text “Geb”
$("h2", 2, id: "section", text: "Geb")
```

CSS Selectors

Full CSS3 if the target browser supports it.

```
$("div.some-class p:first[title='something']")
$("ul li a")
$("table tr:nth-child(2n+1) td")
$("div#content p:first-child::first-line")
```

CSS lookups are fast.

Attribute/Text matching

Can match on attribute values:

```
//<div foo="bar">
$("div", foo: "bar")
```

The “text” attribute is special:

```
//<div>foo</div>
$("div", text: "foo")
```

Can use Regular Expressions:

```
//<div>foo</div>
$("div", text: ~/f.+/)
```

Predicates

Geb supplies some handy predicates:

```
$("p", text: startsWith("p"))
$("p", class: contains("section"))
$("p", id: endsWith(~/\d/))
```

There are more of these.

Relative Content

`$()` returns a Navigator that allows you to find relative content.

```
$("p").previous()
$("p").prevAll()
$("p").next()
$("p").nextAll()
$("p").parent()
$("p").siblings()
$("div").children()
```

Most of these methods take selectors, indexes and attribute text/matchers too.

```
$("p").nextAll(".listing")
```

Pages and Modules

Content DSL

```
class GoogleResultsPage extends Page {
    static content = {
        results { $("li.g") }
        result { i -> results[i] }
        resultLink { i -> result(i).find("a.l", 0) }
        firstResultLink { resultLink(0) }
    }
}
```

Content definitions can build upon each other.

Optional Content

```
class OptionalPage extends Page {
    static content = {
        errorMsg(required: false) { $("p.errorMsg") }
    }
}
```

By default, Geb will error if the content you select doesn't exist.

The "`required`" option disables this check.

Dynamic Content

```
class DynamicPage extends Page {
    static content = {
        errorMsg(wait: true) { $("p.errorMsg") }
    }
}
```

Geb will wait for some time for this content to appear.

Same semantics as the `waitFor {}` method that can be used anywhere.

Expensive Content

```
class ExpensivePage extends Page {
    static content = {
        results(cache: true) { $("li.results") }
        result { results[it] }
    }
}
```

By default, all content is transient.

The `cache` option instructs Geb to hold on to the content, avoiding redundant lookups.

Use carefully, can cause problems with dynamic pages.

Modules

Modules can be used for reused content fragments.

```
class CartInfoModule extends Module {
    static content = {
        section { $("div.cart-info") }
        totalCost { section.find("span.total-cost").toDouble() }
    }
}

class HomePage extends Page {
    static content = {
        cartInfo { module CartInfoModule }
    }
}
```

Modules

They encapsulate detail.

```
to HomePage
assert cartInfo.totalCost == 10.00
```

And support reuse…

```
class AnotherPage extends Page {
    static content = {
        cartInfo { module CartInfoModule }
    }
}
```

Inheritance

Pages (and modules) can be arranged in inheritance hierarchies.

```
class Footer extends Module {
    static content = {
        copyright { $("p.copyright") }
    }
}
class StandardPage extends Page {
    static content = {
        footer { module Footer }
    }
}
class FrontPage extends StandardPage {}
```

The front page will inherit the “footer” content definition.

Feature Tour!

More info @ http://gebish.org/manual/current

Form Shortcuts

Geb makes it easy to deal with all form controls. Read and write `inputs` like properties.

Finds first `input|select|textarea` with the given name.

```
firstName = "Luke" // write
firstName == "Luke" // read

someCheckbox = true
someSelect = "some option name or value"
someMultiSelect = ["selected1", "selected2"]
```

Unified API for all element types.

Driver Management

Geb caches the `WebDriver` instance (per thread) and shares it across tests.

Manages clearing cookies and is configurable.

This can be disabled and tuned.

Configuration Management

Looks for `GebConfig` class or `GebConfig.groovy` file (or class) on classpath.

```
driver = {
    new FirefoxDriver()
}

waiting {
    timeout = 2
    slow { timeout = 100 }
}

reportsDir = "geb-reports"

environments {
    chrome { driver = "chrome" }
}
```

Select environment with `-Dgeb.env=chrome`.

Adhoc waiting

The `waitFor` method can be used anywhere to wait for any condition to be true.

Groovy's flexible notion of “truth” makes this powerful.

```
waitFor { $("p.errorMsg") }.text() == "Error!"
waitFor { $("p.errorMsg").text() } == "Error!"
```

Waiting options are configurable…

```
waitFor(timeout: 10, retry: 0.1) { … }
waitFor("fast") { … }
```

Defaults and named presets controlled by configuration.

Direct Downloading

Examine binary resources easily, with the session state.

```
go "http://myapp.com/login"

// login
username = "me"
password = "secret"
login().click()
def downloadLink = $("a.pdf-download-link")

// now get the pdf bytes (that requires authentication)
def bytes = downloadBytes(downloadLink.@href)
```

Also `downloadStream()` and `downloadText()`.

JavaScript Interface

```
<script>
    var globalVar;
    function globalFunction(arg1) {
        …
    }
</script>
```

Can access global JS variables and functions with Groovy code…

```
js.globalVar = 1
assert js.globalVar == 1

js.globalFunction("the arg")
```

jQuery Adapter

Geb makes it easy to get a jQuery object for content.

```
$("input").jquery.keydown()
```

Useful for simulating interactions that are difficult/unreliable with the WebDriver API.

The `jquery` property is a proxy for an equivalent jQuery object in the JS runtime.

Not to be abused!

Frame / Window Support

Jumping to frames temporarily…

```
withFrame($('#footer')) { assert $('span') == 'frame text' }
```

Focussing on an open window…

```
withWindow({ title == "Another Window" }) {
    // do some stuff with the window
}
// back to the previous window
```

Managing opening new windows…

```
withNewWindow({ $('a').click() }) {
    // do some stuff with the new window
}
// back to the previous window
```

Interaction DSL (Actions)

DSL for building actions…

```
import static org.openqa.selenium.Keys.*

class SomeClass extends Page {
    static content = { someContent { $("div.someContent") } }
}

interact {
    keyDown SHIFT
    clickAndHold someContent
    moveToElement $("div.dropZone")
    keyUp SHIFT
    release()
}
```

Builds on top of WebDriver's `Actions` support.

Remote Browsers

WebDriver can automate remote browsers. This might be a local Windows VM, or a browser out in the “cloud”.

SauceLabs offer cloud browser infrastructure that Geb can use.

```
def capabilities = DesiredCapabilities.firefox()
capabilities.setCapability("version", "5")
capabilities.setCapability("platform", Platform.XP)

def url = "http://user:pass@ondemand.saucelabs.com:80/wd/hub"
def sauceLabsDriver = new RemoteWebDriver(
   new URL(url), capabilities
)
```

Q&A

Further Spock Links

- Homepage - spockframework.org
- Documentation - docs.spockframework.org
- Source Code — github.spockframework.org
- Forum - forum.spockframework.org

Further Geb Links

- Homepage - www.gebish.org
- Documentation — www.gebish.org/manual/current
- Source Code — github.com/geb/geb
- Mailing List — xircles.codehaus.org/projects/geb/lists

Both Spock and Geb will go **1.0** this year.

Thank You

May no bugs be with you!