GEB

GROOVIER BROWSER FUNCTIONAL TESTING

Craig Atkinson

Gr8Conf US 2013

July 23, 2013

ABOUT ME

- Principal Consultant at Object Partners, Inc.
- Developed in Groovy 4 years & Grails 3 years
- Created Geb test suites for several Grails applications
- craig.atkinson@objectpartners.com

COVERAGE

- Browser functional testing high-level
- Getting started with Geb
- Different ways to model your Geb tests
- Geb testing across browsers

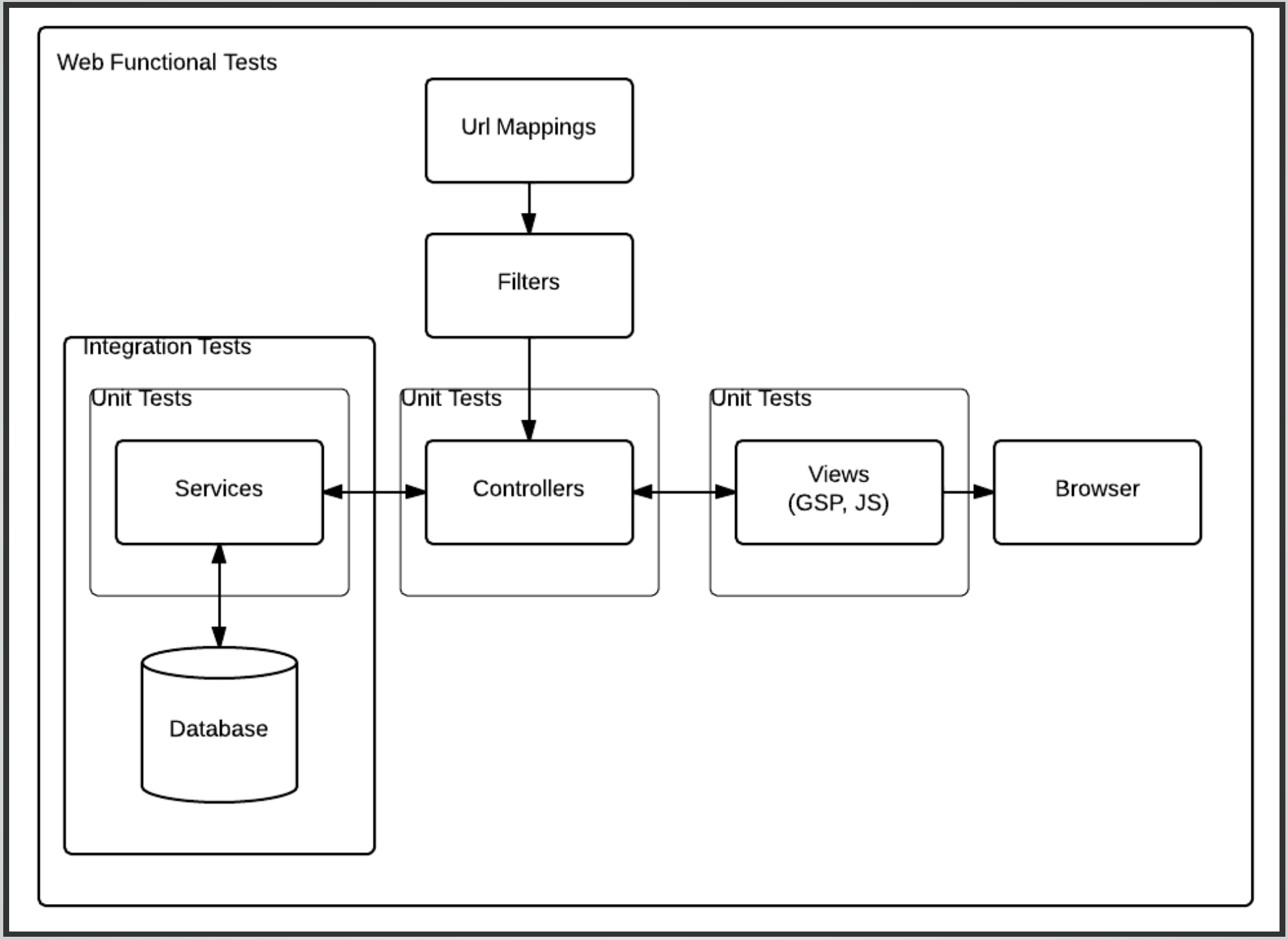
WHAT IS FUNCTIONAL TESTING?

- Interact with the system the way a user does
- For web applications, use a browser

```
// User goes to sign up form

// Fills in all required fields on form

// Clicks submit

// Verify user data saved to database
```

WHY FUNCTIONAL TESTS?

- End-to-end verification all pieces of the application work together to meet customer needs
- Views, controllers, services, database, messaging, caching, etc.


Web Functional Tests
Url Mappings
Filters
Integration Tests
Unit Tests
Unit Tests
Unit Tests
Services
Controllers
Views
(GSP, JS)
Browser
Database

BRIEF HISTORY

- Selenium 1 (JavaScript)
- WebDriver (Native)
- Selenium 1 + WebDriver = Selenium 2 (Native)
- Selenium 2 + Groovy = **Geb**

GEB INTRODUCTION

- Created by Luke Daley
- Groovy wrapper around Selenium 2
- Write tests in JUnit or Spock
- Groovy Page Objects

SAMPLE CODE

Sample project available in GitHub

https://www.github.com/craigatk/geb-example

SIMPLE EXAMPLE TEST

```
go "http://www.google.com"

$("input", name: "q").value("Grails")
$("button", name: "btnG").click()

waitFor { $("div", id: "search").displayed }

assert $("div", id: "search").text().contains("grails.org")
```

TEST DESIGN

- Tests with embedded page structure (HTML) are less readable
- Multiple places to update if page structure changes
- Can we abstract page structure out of tests?

PAGE OBJECT PATTERN

- Abstract page-specific details into helper classes (Page Objects)
- Re-used across tests
- Single point of maintenance
- Enhance test readability

EXAMPLE TEST, REVISITED

```
to GoogleHomePage

searchBox = "Grails"
searchButton.click()

assert searchResults.text().contains("grails.org")
```

WHAT'S IN A GEB PAGE OBJECT?

- Elements on page and how to find them
- How to verify currently on page
- URL to go directly to page

CONTENT BLOCK

- Defines elements on the page that tests will interact with
- Includes **selectors** that tell Geb how to find the element on the page

CONTENT SELECTORS

- Many different types of selectors
- **Geb manual - Selectors**

CONTENT BLOCK EXAMPLE

```
class IdeaCreatePage extends geb.Page {
  static content = {
    titleField { $("input#title") }
    descriptionField { $("textarea#description") }

    createButton { $("input#create") }
  }
}
```

MORE SELECTORS

```
static content = {
  createButtonById    { $("input#create") }

  createButtonByClass { $("input.button") }

  createButtonByName  { $("input", name: "create") }

  createButtonByText  { $("input", text: "Create") }
}
```

SELECTOR SPEED

- Selecting by ID is fastest
- Selecting by text is slow
- Nested selectors can help on large pages

NESTED SELECTORS

```
static content = {
  loginForm { $("form#loginForm") }

  submitButton { loginForm.find("input", text: "Submit") }
}
```

ELEMENT WAITING

- Wait for an element to be on the current page
- I put this on (almost) all elements

```
class IdeaCreatePage extends geb.Page {
  static content = {
    titleField(wait: true) { $("input#title") }
    descriptionField(wait: true) { $("textarea#description") }

    createButton(wait: true) { $("input#create") }
  }
}
```

DYNAMIC CONTENT

- Ajax & server-push
- Wait for element displayed, content values, etc.
- **waitFor** method available in tests and page objects

```
waitFor {
  $("div.alert").displayed
}

waitFor {
  $("div.message").text() == "Update successful"
}
```

CHANGE PAGE WHEN CLICKING

- Geb keeps track of the current page
- Can tell Geb when the page is changing

```
static content = {
  createButton(to: IdeaCreatePage) { $("input#create") }
}

void clickCreateButton() {
  createButton.click()

  assert browser.page.class == IdeaCreatePage
}
```

MULTIPLE POSSIBLE PAGES

- What if one button could take me to different pages?
- Geb uses **at** check to determine current page

```
createButton(to: [IdeaShowPage, IdeaErrorPage]) { $("input#create") }
```

```
static at = { $("div#show-page).displayed }
```

```
static at = { $("div#error-page).displayed }
```

GO DIRECTLY TO A PAGE

- Define **url** in Page Object
- Use **to** method in test
- Speed up test by skipping preliminary pages

```
static url = "idea/create"

IdeaCreatePage ideaCreatePage = to(IdeaCreatePage)
```

PAGE OBJECT INHERITANCE

- **content** blocks from parent classes merged with subclass
- Normal method inheritance as well

RE-USE COMMON CONTENT

- For example, a navigation bar on multiple pages
- Use a Geb Module
- Module's content is defined just like page object

MODULE DEFINITION

```
class NavBarModule extends geb.Module {
  static content = {
    homePageLink(to: HomePage) { $("a#homePageLink") }

    profileLink(to: ProfilePage) { $("a#profileLink") }
  }
}
```

MODULE USAGE

```
class HomePage extends geb.Page {
  static content = {
    navBarModule { module NavBarModule }
  }

  void goToProfilePage() {
    navBarModule.profileLink.click()
  }
}
```

REPEATING CONTENT

- Modules also work well for interacting with structured repeating content
- Tables, lists, etc.

TABLE ROW MODULE

```
class IdeaListPage extends geb.Page {
  static content = {
    ideaListTable {moduleList IdeaListRow, $("table#ideaList tbody tr")
}
  }

  String findDescription(String title) {
    IdeaListRow ideaListRow = ideaListTable.find {
        it.titleCell.text() == title }

    return ideaListRow?.descriptionCell?.text()
  }
}

class IdeaListRow extends geb.Module {
  static content = {
    titleCell(wait: true) { $("td.listTitleTest") }
    descriptionCell(wait: true) { $("td.listDescriptionTest") }
  }
}
```

PAGE OBJECT BUILDER

- Standard: Geb delegates method calls to current page object
- Builder: Test uses reference to current page object

WHY USE PAGE BUILDER?

```
to HomePage
loginButton.click()

username = "user1"
password = "password1"
loginButton.click()

// What page am I on now?

// What fields are on the current page?
```

WITH PAGE BUILDER

```
HomePage homePage = to HomePage

LoginPage loginPage = homePage.clickLoginButton()

DashboardPage dashboardPage = loginPage.login("user1", "password1")
```

HOME PAGE

```
class HomePage extends geb.Page {
  static content = {
    loginButton(to: LoginPage, wait: true) { $("#loginButton") }
  }

  LoginPage clickLoginButton() {
    loginButton.click()

    return browser.page
  }
}
```

LOGIN PAGE

```
class LoginPage extends geb.Page {
  static content = {
    usernameField(wait: true) { $("#username") }
    passwordField(wait: true) { $("#password") }
    submitButton(to: DashboardPage, wait: true) { $("#submit") }
  }

  DashboardPage login(String username, String password) {
    usernameField.value(username)
    passwordField.value(password)

    submitButton.click()

    return browser.page
  }
}
```

CAN CHAIN PAGE CALLS

```
HomePage homePage = to HomePage

DashboardPage dashboardPage = homePage.clickLoginButton().login("user1"
, "password1")
```

JUNIT OR SPOCK

- Geb supports tests written in JUnit or Spock
- Geb test libraries for each test runner
- JUnit and Spock test libraries can coexist in same project

```
def gebVersion = "0.9.0"

test "org.gebish:geb-junit4:${gebVersion}"
test "org.gebish:geb-spock:${gebVersion}"
```

JUNIT TESTS

```
class CreateIdeaFunctionalTests extends geb.junit4.GebReportingTest {
  @Test
  void shouldCreateNewIdea() {
    String ideaTitle = "My new idea"
    String ideaDescription = "It's going to be a good idea."

    IdeaCreatePage ideaCreatePage = to(IdeaCreatePage)

    IdeaShowPage ideaShowPage = ideaCreatePage.createIdea(ideaTitle,
        ideaDescription)

    assert ideaShowPage.titleText == ideaTitle
    assert ideaShowPage.descriptionText == ideaDescription
  }
}
```

SPOCK TESTS

```
class CreateIdeaFunctionalSpec extends geb.spock.GebReportingSpec {
  def "should create a new idea"() {
    given: "the idea creation page"
    IdeaCreatePage ideaCreatePage = to(IdeaCreatePage)

    when: "the user creates a new idea"
    String ideaTitle = "My new idea"
    String ideaDescription = "It's going to be a good idea."

    IdeaShowPage ideaShowPage = ideaCreatePage.createIdea(ideaTitle,
        ideaDescription)

    then: "should show the new idea"
    assert ideaShowPage.titleText == ideaTitle
    assert ideaShowPage.descriptionText == ideaDescription
  }
}
```

SPOCK DATA-DRIVEN TESTS

```
when: "logging in as invalid user"
LoginPage loginPage = loginAsUser(username)

then: "should show error message"
assert loginPage.errorMessage == expectedErrorMessage

where:
username        | expectedErrorMessage
'disabledUser' | 'Sorry, your account is disabled'
'lockedUser'   | 'Sorry, your account is locked'
'invalidUser'  | 'Sorry, we could not find that account'
```

SPOCK CUSTOM TEST REPORTS

- Can create reports with given/when/thens
- Human-readable reports of current application features

SCREENSHOTS AND HTML DUMPS

- Very helpful for debugging test failures, especially in CI
- Extend GebReportingSpec or GebReportingTest
- Screenshot after cleanup, not at line of test failure

CROSS-BROWSER TESTING

- Can use same test code across browsers
- Configure supported browsers in GebConfig.groovy

BROWSER REQUIREMENTS

- Firefox: Selenium driver dependency only
- Chrome: Selenium driver and OS-specific library
- IE: Selenium driver and server library

AUTOMATICALLY DOWNLOAD BROWSER LIBRARIES

- Manually downloading driver libraries on each machine is a pain
- Instead, let's automatically download the library
- Download based on current OS
- See example **GebConfig.groovy**

HEADLESS TESTING

- X Virtual Frame Buffer (xvfb)
- Simulated browsers (HtmlUnit, PhantomJS)

HTMLUNIT

- Uses Rhino JavaScript engine
- Can get into dependency conflict hell

PHANTOMJS

- Used by several libraries, including JavaScript unit testing runners
- Uses WebKit JavaScript engine
- GhostDriver - WebDriver implementation for PhantomJS

SIMULATED VS. REAL BROWSERS

- How many and which browsers do we test with?
- Value, maintenance, speed

SELENIUM VERSION DECOUPLED

- Can update version of Selenium independent of Geb version
- Selenium updated frequently to work with latest browsers
- Don't need new Geb release for each Selenium release

RUNNING GEB TESTS

Grails

```
// Run all functional tests
grails test-app functional:

// Run individual test
grails test-app functional: IndividualGebSpec

// Run all tests in a package (and sub-packages)
grails test-app functional: com.mycompany.auth.**.*
```

SWITCHING BROWSERS

GebConfig.groovy

```
// Default browser
driver = { new FirefoxDriver() }

environments {
  chrome {
    System.setProperty('webdriver.chrome.driver', '/path/to/chromedrive
r')

    driver = { new ChromeDriver() }
  }
}
```

Command line

```
grails -Dgeb.env=chrome test-app functional:
```

GEB WITH GRADLE

- Execute tests against running server
- **build.gradle example**
- Can deploy/start/stop container with **Gradle Cargo plugin**

RESOURCES

- Geb Manual
- Geb Mailing List
- Presentation Examples

SUMMARY

- Automated functional tests: Sleep easier at night
- Geb: Groovier functional testing
- Page Objects: readable and maintainable tests
- Spock: Powerful data-driven and reporting test runner

Q & A