GEB TIPS & TRICKS

Craig Atkinson | Chief Technologist | Object Partners

OBJECT PARTNERS

ABOUT ME

- Craig Atkinson
- Chief Technologist, Object Partners (OPI)
- Using Geb 4 past years
- Very minor Geb contributor

AGENDA

- Stable tests in CI environments
- Alternative approach to page objects
- Complex UI elements
- Cross-browser testing
- Running tests in parallel
- Remote controls
- Mock 3rd-party services

STABLE TESTS IN CI

- Frustrating when tests pass locally but fail in CI
- Failures tougher to debug

TIMING ISSUES

- CI machines often slower than developer laptops
- Dialogs that open or close slowly, dynamic content rendering, etc.
- Geb has built-in waiting support

GEB WAITING SUPPORT

- Wait for elements to be visible, page content to change, etc.
- Can also wait for data changes (new DB record created, etc.)
- **waitFor** method available in tests and page objects
- Waits for any closure to return true

WAITING EXAMPLES

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

assert $("div.alert").text() == "Error creating user"
```

```
waitFor {
  databaseUtil.userExists('myUsername')
}

loginPage.login('myUsername', 'myPassword')
```

TEST FAILURE SCREENSHOTS

- Usually can't watch browser test run on CI server
- PNG screenshot and HTML dump when test finishes
- Have tests extend **GebReportingSpec**
- Configure CI build to archive screenshots & HTML

SCREENSHOT CONFIGURATION

GebConfig.groovy

```
// Configure where screenshots are stored
reportsDir = 'build/test-reports'

// Optional, less noise when only failed tests generate screenshots
reportOnTestFailureOnly = true
```

JAVASCRIPT ERRORS

- Often have logs for server errors
- More code on front-end with newer frameworks like Angular, React, etc.
- What about Javascript errors?

CAPTURE BROWSER LOGS

```
def cleanup() {
  LogEntries logs = driver.manage().logs().get(LogType.BROWSER)
  List errorLogEntries = logs.filter(Level.SEVERE)

  if (errorLogEntries) {
    println "Browser errors:"

    errorLogEntries.each { logEntry ->
      println("[${logEntry.level}] ${logEntry.message}")
    }
  }
}
```

BROWSER LOG EXAMPLE

```
// Intentional Javascript error for demonstrating capturing browser logs
fakeObject.fakeMethod();
```

```
Browser errors:
[SEVERE] /author/edit/4 59:13 Uncaught ReferenceError: fakeObject is not defined
```

BROWSER LOGGING GOTCHAS

- Not supported by Internet Explorer
- For safety, wrap logging code in try/catch

HEADLESS CI

- Run tests without display attached
- **X virtual frame buffer**

PAGE OBJECTS

PAGE OBJECT PATTERN

- Abstract page-specific details into helper classes (Page Objects)
- Re-used across tests
- Single point of maintenance
- Tests become easier to read

PAGE OBJECT EXAMPLE

```
class LoginPage extends geb.Page {
  static content = {
    usernameField { $("#username") }
    passwordField { $("#password") }

    submitButton(to: DashboardPage) { $("#submit") }
  }
}
```

USING PAGE OBJECTS IN TEST

Standard: Geb delegates method calls to current page object

TEST WITH STANDARD PAGE OBJECTS

```
to HomePage
loginButton.click()

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

// What page is the test on now?

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

ALTERNATIVE APPROACH TO PAGE OBJECTS

STRONGLY-TYPED PAGE OBJECTS

- Test uses instance of current page object
- Page objects have methods that resemble user actions
- Methods that change page return instance of new page
- Tests even easier to read
- IDE autocomplete available on pages in tests

TEST USING TYPED PAGES

```
HomePage homePage = to(HomePage)

LoginPage loginPage = homePage.clickLoginButton()

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

CHAIN PAGE CALLS

```
HomePage homePage = to(HomePage)

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

TYPED HOME PAGE

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

  LoginPage clickLoginButton() {
    loginButton.click()

    return browser.page
  }
}
```

TYPED LOGIN PAGE

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

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

    return browser.page
  }
}
```

TYPED PAGE OBJECTS SUMMARY

- Some additional work to write page objects
- Save much more time when writing (and reading) test cases

COMPLEX UI ELEMENTS

MOUSE INTERACTIONS

Click and drag slider with mouse

```
static content = {
  ratingSliderHandle { $(".ui-slider-handle") }
}

void moveRatingSlider(Integer rating) {
  // Slider is 400 pixels wide and starts at 1,
  // so each notch above 1 is 100 pixels apart
  Integer numPixelsX = (rating - 1) * 100

  interact {
    clickAndHold(ratingSliderHandle)
    moveByOffset(numPixelsX, 0)
    release()
  }
}
```

MOUSE INTERACTION DEMO

USE KEYBOARD

Send keystrokes with left-shift operator <<

```
$("#myInputField") << "value"
```

```
$(".ui-slider-handle") << Keys.ARROW_RIGHT
```

KEYBOARD DEMO

EXECUTE RAW JAVASCRIPT

- Last-ditch effort with complex UI controls
- All Navigator elements have a 'jquery' field to run JS

SHOW HIDDEN ELEMENT

```
static content = {
  hiddenLink { $("#hiddenLink") }
}

void clickHiddenLink() {
  hiddenLink.jquery.show()

  hiddenLink.click()
}
```

CROSS-BROWSER TESTING

AVAILABLE BROWSERS

- Any browser with a Selenium/Webdriver library
- Real: Firefox, Chrome, IE, Safari, etc.
- Simulated: HtmlUnit, PhantomJS

CONFIGURING BROWSER IN GEB

- Browser Selenium driver dependency
- Additional OS-specific browser driver
- Section in GebConfig.groovy
- Pass environment parameter to Geb

SELENIUM DRIVER DEPENDENCIES

```
def seleniumVersion = "2.53.1"

testCompile "org.seleniumhq.selenium:selenium-chrome-driver:${seleniumVersion}"
testCompile "org.seleniumhq.selenium:selenium-firefox-driver:${seleniumVersion}"
testCompile "org.seleniumhq.selenium:selenium-ie-driver:${seleniumVersion}"
```

OPERATING SYSTEM DRIVER

- Chrome & Internet Explorer need local driver executable
- Firefox <= 47 doesn't need one, Firefox >= 48 does
- Manually download, install, and configure
- Or ...

AUTOMATIC DOWNLOAD

- **WebdriverManager library** by Boni Garcia
- Downloads and configures driver for your OS

WEBDRIVERMANAGER DEPENDENCY

```
testCompile("io.github.bonigarcia:webdrivermanager:1.4.1")
```

CHROME GEBCONFIG EXAMPLE

```
import io.github.bonigarcia.wdm.ChromeDriverManager
import org.openqa.selenium.chrome.ChromeDriver

environments {
  chrome {
    ChromeDriverManager.getInstance().setup()

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

WEBDRIVERMANAGER CONFIGURATION

- Drivers downloaded to ~/.m2/repository/webdriver by default
- Download location and many other parameters are **configurable** via config file or system properties

EXAMPLE CONFIG FILE

```
# src/test/resources/application.properties

wdm.targetPath=build/webdriver
```

PASS GEB.ENV PARAMETER

Grails 2

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

Grails 3

```
gradle -Dgeb.env=chrome integrationTest
```

Spring Boot

```
gradle -Dgeb.env=chrome test
```

TWEAK GRAILS 3 GRADLE FILE

```
// Pass system properties through to the integrationTest task
// so we can pass in the 'geb.env' property
configure(integrationTest) {
  systemProperties System.properties
}
```

TWEAK SPRING BOOT GRADLE FILE

```
// Pass system properties through to the test task
// so we can pass in the 'geb.env' property
configure(test) {
  systemProperties System.properties
}
```

PARALLEL TESTING

PARALLEL TESTING REQUIREMENTS

- Build tool that supports parallel testing (Gradle, etc.)
- Tests can't modify any shared data
- Safest to have each test set up its own data

PARALLEL TESTING EXAMPLE

- Using Gradle
- Start app using Grails wrapper
- Run 2 tests simultaneously
- Shut down Grails app
- Demo

GROOVY REMOTE CONTROL

WHAT ARE REMOTE CONTROLS USEFUL FOR?

- When tests run in separate JVM from application
- Setting up data specific to a test
- Grabbing data after test for verification
- Retrieving & setting up data in mock services (stay tuned)

HOW DOES IT WORK?

- Lets you execute code inside the running app from a test
- **Remote Control plugin docs**

DATA SETUP EXAMPLE

```
Idea createIdea(String title, String description) {
  RemoteControl remote = new RemoteControl()

  remote {
    Idea idea = new Idea(
        title: title,
        description: description
    )

    idea.save()
  }
}
```

```
List ideas = (1..5).collect { i ->
  ideaRemoteControl.createIdea("Title ${i}", "Description ${i}")
}
```

DATA VERIFICATION EXAMPLE

```
Idea findByTitle(String title) {
  RemoteControl remote = new RemoteControl()

  remote {
    Idea.findByTitle(title)
  }
}
```

GRAILS 3 INJECTION

Tests and app run in same JVM

```
import grails.test.mixin.integration.Integration
import org.springframework.beans.factory.annotation.Autowired

@Integration
class AuthorGebSpec extends GebReportingSpec {

    @Autowired
    AuthorDataUtil authorDataUtil

    void "should create Author"() {
      ...

      then:
      assert authorDataUtil.findByLastName('Last')?.firstName == 'First'
    }
}
```

MOCK THIRD-PARTY SERVICES

PROLIFERATION OF THIRD-PARTY SERVICES

- Email, payment processing, storage, address verification, etc.
- Great for productivity, but can complicate testing
- Don't want our tests dependent on services out of our control

SOLUTION?

- Mock request/response from external services
- Our tests not dependent on external service
- Tests have tight control over service responses

TEST-SPECIFIC DEPENDENCY INJECTION

- When using DI framework like Spring, Guice, etc.
- Create mock version of code that calls external service
- Use DI to replace code with mock version during functional tests
- Setup and verify data from mock services in tests

EXAMPLE SERVICE TO MOCK

PatentService sends ideas to patent office

```
class PatentService {

  def sendToPatentOffice(Idea idea) {
    // Send the idea to the real patent office

  }

}
```

MOCK PATENT SERVICE

```
class PatentServiceMock extends PatentService {
  List ideasSentToPatentOffice = []

  @Override
  def sendToPatentOffice(Idea idea) {
    ideasSentToPatentOffice << idea
  }
}
```

USE MOCK PATENT SERVICE

- Replace PatentService instance with mock version
- In Grails, grails-app/conf/spring/resources.groovy

```
beans = {

  if (Environment.current == Environment.TEST) {
    // Override the PatentService with our mock version when running tests
    patentService(PatentServiceMock)

  }
}
```

TEST THAT HITS MOCK SERVICE

```
@Autowired
PatentServiceMock patentService

def 'should submit idea to patent office'() {
  given:
  Idea idea = ideaRemoteControl.findByTitle('Patentable Idea')

  IdeaShowPage ideaShowPage = to([id: idea.id], IdeaShowPage)

  when:
  ideaShowPage = ideaShowPage.submitIdeaToPatentOffice()

  then:
  List ideasSubmittedToPatentOffice = patentService.ideasSentToPatentOffice

  assert ideasSubmittedToPatentOffice*.id.contains(idea.id)
}
```

WRAPPING UP

SUMMARY

- Stable tests in CI environments
- Alternative approach to page objects
- Complex UI elements
- Cross-browser testing
- Running tests in parallel
- Data access with remote controls
- Mock 3rd-party services

RESOURCES

- Geb manual
- Geb User mailing list
- Official Geb/Grails example
- Official Geb/Gradle example
- Geb/Spring Boot example
- Geb/Grails 3 example
- Geb/Grails 2 example
- Geb/Gradle example

CONTRIBUTING TO GEB

- Answer questions on **mailing list**
- Contribute to **Geb manual**
- Tackle a **Geb issue**

Q&A

- @craigatk1
- craig.atkinson@objectpartners.com