本日やること

- Geb

ほかに使うこと

- Groovy
- Spock
- maven

本日やること(2)

- Gebを使ってhtml要素を取得してみる
- PageObjectパターン
  →テストをやってみる
- htmlが変更になった(改修が入った)場合の修正

進め方

- 基本的にはソースを書いてもらいます
- 不明点があれば遠慮なく聞いてください
- 周りにわからなそうな人がいれば教えてあげてください

環境確認

- Groovyコンソール
- JDK
- Eclipse
  →各種プラグイン
- Firefox

Groovyとは

- JVM上で動作する言語です。
- Javaと親和性が高く、Javaのコードがそのまま使えたりします
- Javaより規約がゆるい（主観）です。

詳しくは

http://ja.wikipedia.org/wiki/Groovy

Gebとは

- Webアプリ向けの機能テスト用ライブラリです
- jQueryライクな記述で要素を取得できます
- 同じテスト用APIのseleniumに比べ、記述量はかなり少ないです。
- 手続き的な記述が少なくてすみます。

- ドキュメント

  http://www.gebish.org/manual/0.6.2/index.html

Spockとは

- Gebでのテストのフレームワークです。
- 事前処理、条件、期待される結果を決められた書式で書くことで、簡単に実行できます。

詳しくは

https://spock-framework-reference-documentation-ja.readthedocs.org/ja/latest/

Gebを触ってみる

- Groovyコンソールの起動

  println “hello”

  ※上記入力後、Ctrl + R

Geb資料のディレクトリ

- C:\Temp\handson

  にintroWorkディレクトリが置かれる想定です。

- htmlファイルに対してgroovyファイルが対応しています。

Gebを動かしてみる

- 前提

  go 'URL' でURLにアクセスできます

  assert で値の検証ができます

  4行目（import）より上はおまじないです
- 補足

  実行は Script　→ run もしくは Ctrl + R

  コンソールをきれいにするには

  View → Clear Output もしくは Ctrl + W

Gebを動かしてみる

ブラウザを手動で閉じたりした場合、そのままだと次回の実行でブラウザがあがらなくなります。

その場合

Script → Clear Script Context

を実行してください。

テキストに値を入力してみる

- ID指定には $(“#id名”) が使用できます
- テキストに値を入力したい場合

  .value(“入力したいテキスト”)

  が使用できます。

- 課題

  一番上のテキストボックスに”test”を入れてみてください

テキストに値を入力してみる

$("#text1").value('test') で一番上のテキストに”test”という文字列が入力されます。

- 同様にIDを変更することで、指定したテキストに値を入力することができます。

- 課題

  上から順番に 1,2,3 を入力してみてください

テキストに値を入力してみる

- name指定には$(name:"nameの値")が使用できます。
- テキストに値を入力したい場合

  .value(“入力したいテキスト”)

  が使用できます。

- 課題

  一番下のテキストボックスに”test2”を入れてみてください

テキストに値を入力してみる

- $(name:"text3").value('test2')で一番下のテキストに”test2”という文字列が入力されます。

- 同様にnameを変更することで、指定したテキストに値を入力することができます。

- 課題

  上から順番に 3,2,1 を入力してみてください

ボタンを押下してみる

- ID指定(#)が使用できます。
- .click()を使用するとボタンを押すことができます。
- 課題
  画面のボタンを押してください

ボタンを押下してみる

- $("#ok_button").click()　でボタンを押すことができます。

ラジオボタンを選択してみる

- .nameの値 = valueの値 でラジオボタンがチェックできます

- 課題

  ラジオボタン”なし”にチェックを入れて、その後”あり”にチェックを入れてください

ラジオボタンを選択してみる

```
$("form").on_off_radio = "off"
sleep(1000)
$("form").on_off_radio = "on"
```

でチェックができます。

チェックボックスを選択してみる

- .value(true)でチェックをつけることができます。
- .value(false)でチェックを外すことができます。
- 課題

  チェックボックスにチェックをつけたり外したりしてください

チェックボックスを選択してみる

```
$("#allowed_check").value(true)
sleep(1000)
$("#allowed_check").value(false)
```

でチェックをつけたり外したりできます。

プルダウンから選択してみる

- .value(プルダウンから選択したい値)でプルダウンから値を選択することができます。

- 課題

  プルダウンから”2”を選択してください

プルダウンから選択してみる

$("#head_count").value(2)

で値を取得することができます。

画面のテキストを取得する

- .text()を使用すると画面のテキストを取得することができます。

- 画面上変化がないので、assertを使用して確認してください。

- 課題

  画面の”9000”という値を取得してください

画面のテキストを取得する

```
assert $("span").text() == "9000"
```

で画面に表示されている”9000”が取得できます。

検証は

```
assert $("span").text() == "9000"
```

で行うことができます。

セレクトボックスから選択してみる

- .value()を使用すると値が取得できます
- 複数選択も[値1,値2,...]を使うことで可能
- 課題
  セレクトボックスから”test2”を選択し
  その後”test3”,”test4”を複数選択してください

セレクトボックスから選択してみる

```
$("select" , name:"tests").value("2")
sleep(3000)
("select" , name:"tests").value([3,4])
```

で選択できる。

セレクタで遊んでみる

- 例えばclassタグが入れ子になっているようなhtmlの場合、目的の要素をとるにはどうしたらよいか？

```
<div class="test1">
    <p class="test2">test1.test2</p>
</div>
<div class="test2">test2</div>
```

セレクタで遊んでみる

find() や filter()が使えます。

$("div").find(".test2").text()
→ pタグに囲まれている "test1.test2"を取得

$("div").filter(".test2").text()
→pタグに囲まれていない"test2"を取得

セレクタで遊んでみる

not() や has() も使えます。

$(".test2").not("p").text()
→ pタグに囲まれていない"test2"を取得

$("div").has("p").text()
→pタグに囲まれている "test1.test2"を取得

セレクタで遊んでみる

- 要素の複数指定もできます

  $($("要素A"), $("要素B"))といった記述で
  要素Aと要素Bが取得できます。

正規表現も使えます

- 要素の指定に正規表現を使用することができます

  . → 任意の1文字

  [a-z] → 小文字のアルファベット1文字 など

  例

  ~/^[a-z]*/

  →先頭が小文字のアルファベットで始まる要素

ショートカットパターンメソッド

- 特定の要素とマッチングさせるために使用できるメソッドが存在します。

例

startWith(“文字列”)

→文字列で指定した値から始まる要素とマッチング

要素の最大・最小を求めたりもできます

.max や .min で取得した要素中の最大・最小が求められたりもします。

まとめ

- 少ない記述量で必要な要素を取得できる
  →気軽にテストを書ける

- 要素を取得するための機能も色々ある
  →必要な機能で必要な要素を取得できる

要素の取得方法がわかったところで
Webアプリを使ってテストを作成してみましょう

ページオブジェクトパターン

- 1ページに対して1つのページを定義するファイルを作成
- 作成したファイルにページの定義を行う

レイアウトの変更時、最低限のソース修正で済むようになる

ページオブジェクトパターン

- テスト用のhtmlに対し、ページオブジェクトパターンをあてはめる

- 3ページのhtmlそれぞれのページオブジェクトを作成
- 各ページ内の要素を定義
- テストを記述

テスト用クラス

- GebSpecクラスを継承したファイルでSpockを利用したテストクラスを作成する。
- Def メソッド名 () でテストを定義できる。
- メソッド名を””でくくると日本語名のメソッドも定義可能

Spockのブロック

- setup,given
  →データの初期化等の事前処理
- when
  →テスト条件
- then
  →条件によって予想される結果
- expect
  →条件と予想結果を同時に判定

Spockのブロック

その他 where や cleanupもある。

データテーブル等も使えるが本題ではないので割愛

- ひとまずコメントに合わせてやることを書きましょう

実行

- ファイルを右クリック→実行→JUnitテスト で実行
- JUnitタブに実行結果が表示される
- エラーになった場合、トレースを追って修正しましょう

レポート自動出力

- 設定ファイルを使用することで、画面のキャプチャを自動で出力できます
- GebConfig.groovy をresourcesフォルダ以下に格納

  reportsDir = "target/geb-reports"

  reportOnTestFailureOnly = false

  を設定する
- プロジェクトフォルダ内の target/geb-reports 以下にキャプチャが保管されます

クロスブラウザテスト

- ドライバをダウンロードすることでクロスブラウザテストができます

- プロジェクトフォルダ直下にdriverフォルダを追加
- GebConfig.groovy に以下の設定を追加

```
driver = "chrome"

System.setProperty("webdriver.chrome.driver",
"driver/chromedriver.exe")
```

クロスブラウザテスト

- ドライバがあれば他のブラウザでも試験可能です。
- デフォルトはfirefoxを使用します

共通機能の定義

- ページオブジェクトにメソッドを定義することで共通化ができます
- 例えば、入力画面の入力を1メソッドに共通化してみましょう。

- 入力画面のページオブジェクトにメソッド追加

```
void testInput(){
   //各要素に値を入力するコードを記述
}
```

モジュール

- 各ページで共通の機能がある場合、モジュールに定義をしておくと、各ページオブジェクトで定義を使いまわせます。

- 例えば、違うページに同じ名前のボタンがあるような場合はボタン取得の定義をモジュールに記載しておくことで、コード重複を無くすことができます

UI変更後の改修をしてみる

- テスト用htmlの画面が改修された場合に修正をしましょう
- StarHotel_Renew プロジェクトのページオブジェクト・テストを実装してください
- 基本コピペでいいですが、一部動作しない箇所があります

UI変更後の改修をしてみる

- 基本的にはページオブジェクトを改修すれば動くようになります。

- コード重複を作らないようにすることで、
  改修を楽に実施することができます。

参考ページを記載

終わり