特別企画

ブラウザベースで
Webアプリの品質向上をサポート

機能テストの繰り返しを自動化する「Selenium」の実力

重要なビジネスに多くのWebアプリケーションが導入されている昨今、当然のことながらバグの発生はビジネスに支障を来たすものとなる。バグを防止するには、アプリケーションの機能テストを繰り返さなければならないが、これはエンジニアにとって大きな負担となる。この問題を解消するのが、機能テスト自動化ツール「Selenium」だ。本稿では、Seleniumの概要のほか、使い方について基礎から詳しく解説する。

株式会社フルネス
田原 孝 TAHARA, Takashi

機能テストの自動化

設計フェーズや実装フェーズで入念にレビュー（検証）を行なっても、少なからずミスは混入する。そのミスを発見してソフトウェアの品質を向上させるためには、テストが非常に重要となる。

しかし、品質向上のためにテストケースを増やすと、テスト工数が大幅に増えてしまう。さらに、バグ修正や仕様変更などで同じテストを何度も繰り返すと、テスト担当者の精神的負担も大きくなる。

テスト工数や担当者の負担を増やさずに、テストにより品質を向上させるには「テストの自動化」がポイントとなる。繰り返し実施するテストを自動化することで、テスト工数やテスト担当者の精神的負担を軽減できる。また、自動化したテストを定期的に実施することで、バグ修正や仕様変更による品質低下を早期に発見することもできる。

単体テストの自動化は、JavaならJUnit、.NETならNUnitなどといった定番のフレームワークが存在する。Webアプリケーションの機能テストを自動化する場合、自動化するためのスクリプト作成、および保守にコストがかかる。そのため、現場では機能テストの自動化を実施しているケースは少ない。また、有償の自動化ツールは価格が高いため、試しに導入することもなかなか難しいのが現状である。

しかし、このような問題を解決してくれるツールがある。それが、今回紹介する「Selenium」（セレニウム）である。

Seleniumとは

Selenium（http://seleniumhq.org/）は、Webアプリケーション向けの機能テスト自動化ツールである。品質管理のオープンソースソフトウェアを手がける「OpenQA」（http://www.openqa.org/）が拠点となって開発され、Apache 2.0 Licenseのオープンソースソフトウェアとして無償で提供されている。

Seleniumの主な特徴は、次の2つである。

- Webブラウザ上の操作を記録してテストケースを作成できる
- 各種のWebブラウザで動作を検証できる

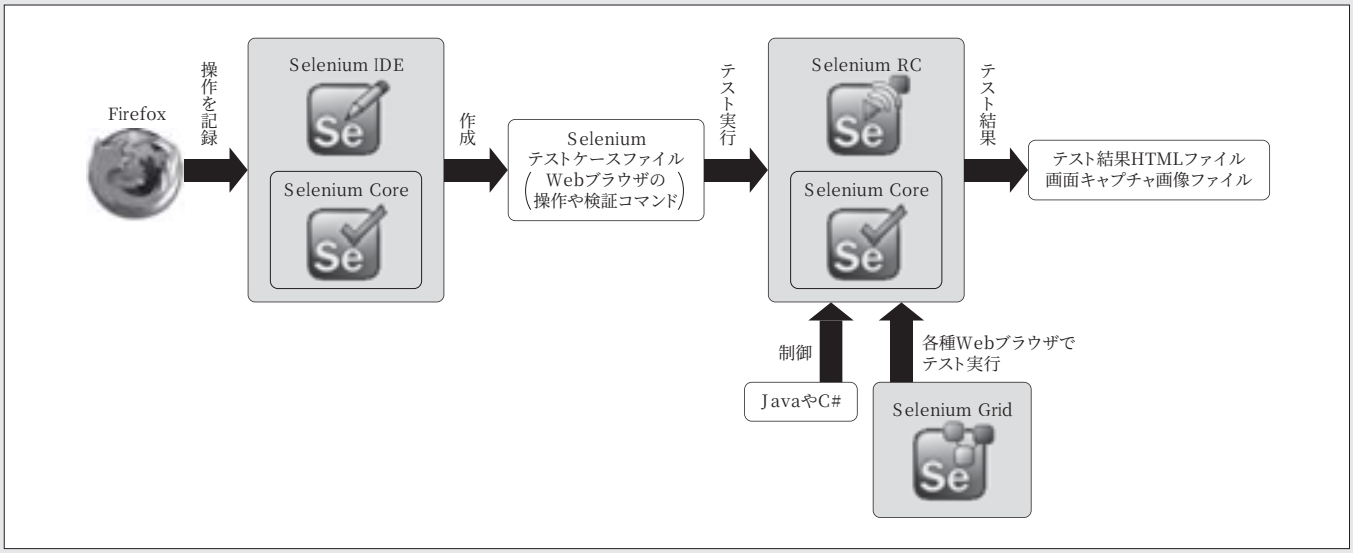


図1：Seleniumプロジェクト

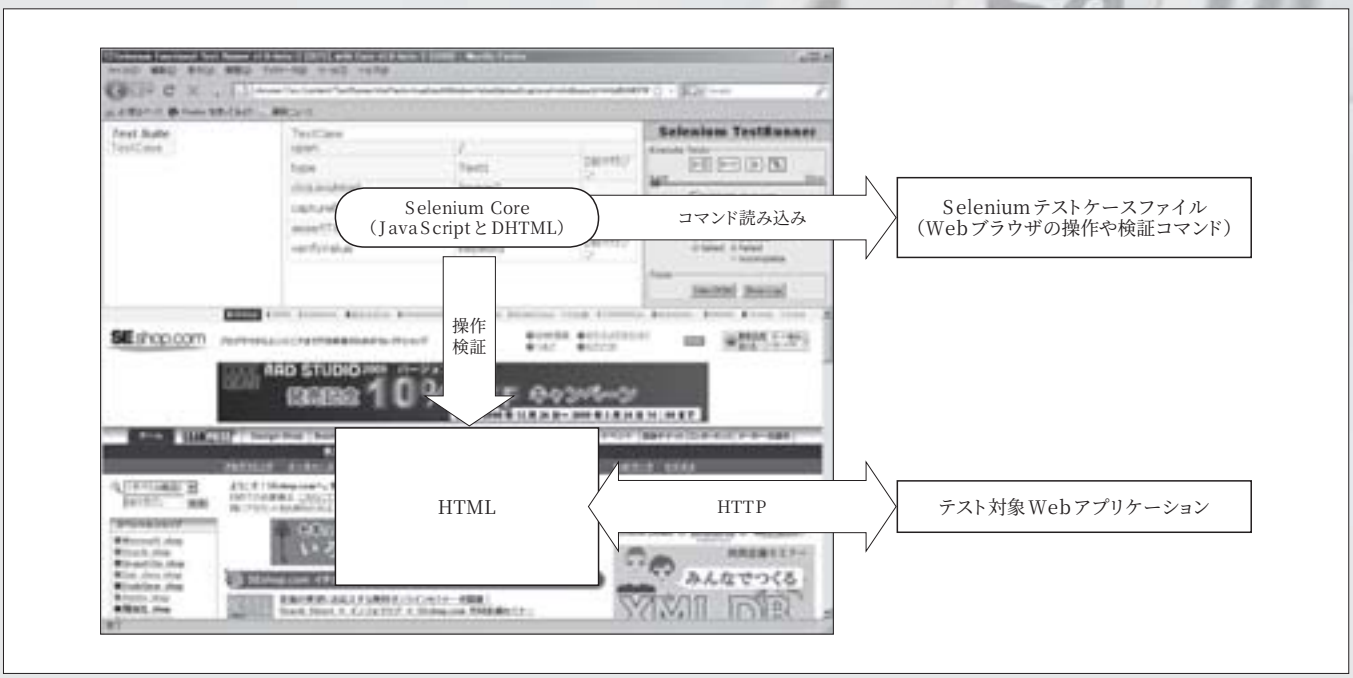


図2：Seleniumの動作原理

Seleniumプロジェクト

Seleniumには、テストの作成や実行を行なうためのさまざまなツールがある（**図1**）。以降で、その概要を簡単に紹介していこう。

Selenium Core

Selenium Coreは、テスト実行環境を提供するプログラムである。JavaScriptとDHTMLで実装されており、Webブラウザ上で実行される。Webブラウザの操作や検証コマンドが格納された「Seleniumテストケース」に従って、テストを実施する。

なお、後述のSelenium IDEやSelenium RCには、テストを実施するためにSelenium Coreが含まれている。

Selenium IDE

Selenium IDEは、Seleniumテストケースを効率良く作成するためのキャプチャリプレイツールである。Firefoxのアドオンとして提供されており、Firefoxの操作を自動で記録することによりSeleniumテストケースを作成できる。

Selenium Remote Control

Selenium Remote Control（以下、Selenium RC）は、Seleniumテストケースの自動実行やJava、C#、およびRubyなどのプログラム言語からSeleniumを制御するためのツールである。

Selenium Grid

Selenium Gridは、各種Webブラウザの動作検証を一括して行なうためのツールである。さまざまなOSやWebブラウザをインストールした複数のマシンに対して、一括してテストを実施することで、各種Webブラウザの動作検証を効率良く実施できる。

なお、原稿執筆時点のバージョンはSelenium Core、Selenium IDE、およびSelenium RCが1.0 beta 2で、Selenium Gridが1.0.3である。

Seleniumの動作原理

すでに紹介したとおり、SeleniumはWebブラウザ上でテストを実施する（**図2**）。ここでは、その動作原理について紹介しよう。

まず、テスト実行環境のSelenium CoreをWebブラウザに読み込ませる。そして、WebブラウザがSelenium CoreのJavaScriptを実行し、Seleniumテストケースに記述されたコマンドに従ってWebアプリケーションの機能テストを実施する。

このような仕組みになっているため、実際のWebブラウザで動作確認できる。また、JavaScriptに対応しているWebブラウザで動作するため、サポートするWebブラウザの種類も豊富だ（**表1**）。さらに、Webブラウザ経由でテストを実施するため、Webアプリケーションの実装に依存しない。各種言語で開発したWebアプリケーションの機能テストが行なえるというわけだ。

Seleniumのテスト手順

Seleniumによるテスト手順は、大別すると次のとおりである。

①Seleniumテストケースファイルを作成する

Seleniumテストケースファイルには、テストを実施するためのWebブラウザの操作や検証コ

マンドがHTMLテーブル形式で格納されている（**LIST1**）。

HTMLテーブル形式なので、テキストエディタなどでも作成できるが、Firefoxの操作からSeleniumテストケースを作成できるSelenium IDEを使うのが効率的だ（使い方は後述）。

②Seleniumテストケースを実行する

次に、作成したSeleniumテストケースファイルを、Selenium RCで実行する。

JavaコマンドやAntなどでSelenium RCを起動すると、WebブラウザにSelenium Coreを読み込ませ、Seleniumテストケースが実行される。

テスト結果のエビデンスは、テスト結果HTMLファイルとして出力される。FirefoxとInternet Explorerに限り、captureEntirePageScreenshotコマンドとアドオンを使うことで、Webブラウザの画面キャプチャを保存することも可能だ（**コラム**「Webブラウザのキャプチャ画面」を参照）。

Seleniumテストケースの作成

ここでは、まずはじめに作成するテストケースについて解説する。

Seleniumテストケースのファイル構成

Seleniumテストケースファイルは、次の2つから構成されている。これらのファイルをテキストエディタやSelenium IDEを利用して作成する。

①Seleniumテストケースファイル

Webブラウザの操作や検証コマンドを格納するファイル。コマンドとコマンド引数をHTMLテーブル形式で記述する（LIST1）。

②Seleniumテストスイートファイル

複数のSeleniumテストケースファイルをまとめるファイル。Seleniumテストケースファイルへのハイパーリンクをテーブル形式で記述する（**LIST2**）。

コマンドの種類

テストを実施するためのWebブラウザの操作や検証は、コマンドで指定する。Selenium Coreが提供する主なコマンドは、次の2つである。

Actionコマンド

Webブラウザを操作するコマンド。主なActionコマンドを**表2**に示す。これらのコマンドを使って、マウスやキーボードの操作、HTML入力項目の操作、ブラウザバックなど、Webブラウザを操作する。

Assertionコマンド

WebページのHTMLの値や状態を検証するためのコマンド（**表3**）。Assertionコマンドは、1種類の検証に対して検証失敗時の動作の違いに対応する複数のコマンドがある。

例えば、Webページのタイトルを検証する「Titleコマンド」には、次のように複数の種類がある。

- assertTitleコマンド
- assertNotTitleコマンド
- verifyTitleコマンド
- verifyNotTitleコマンド
- waitForTitleコマンド
- waitForNotTitleコマンド

assert～は、検証が失敗した際にテストを終了するコマンドだ。画面遷移の検証失敗など、テストを続けても意味がないケースで使う。verify～は、検証が失敗した際に、エラーメッセージを出力してテストを続行する。画像のURLの検証失敗など、その後のテストに影響がないケースで使う。waitFor～は、assert～と同じだが、一定時間（デフォルトは30秒）以内に条件を満たさないと検証失敗になる。

LIST1：Seleniumテストケースファイル（一部抜粋）

```
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">TestCase</td></tr>
</thead><tbody>
<tr>
    <td>コマンド</td>
    <td>第1コマンド引数（target）</td>
    <td>第2コマンド引数（value）</td>
</tr>
<tr>
    <td>type</td>
    <td>Text1</td>
    <td>DBマガジン</td>
</tr>
</tbody></table>
```

表1：Seleniumがサポートする主なブラウザとOS

対応Webブラウザ	Firefox 2以降、Internet Explorer 7、Safari 2以降、Opera 8以降など
対応OS	Windows、Mac OS X、Linux、Solarisなど

LIST2：Seleniumテストスイートファイル（一部抜粋）

```
<table id="suiteTable" cellpadding="1" cellspacing="1"
border="1" class="selenium"><tbody>
<tr><td><b>Test Suite</b></td></tr>
<tr><td><a href=" Seleniumテストケース・ファイルのパス">テストケース名</a></td></tr>
<tr><td><a href="TestCase.html">TestCase</a></td></tr>
</tbody></table>
```

表2：主なActionコマンド

コマンド名	概要	第1引数（target）	第2引数（value）
click	指定したHTML要素をクリック	エレメントロケータ	
doubleClick	指定したHTML要素をダブルクリック	エレメントロケータ	
goBack	ブラウザバック		
keyPress	指定したHTML要素にキーを押下する	エレメントロケータ	キーコード
type	指定されたHTMLの入力要素に文字列を入力	エレメントロケータ	入力する文字列

なお、assertNot～、verifyNot～、およびwaitForNot～は、それぞれのコマンドの検証条件が逆になったコマンドである。

このように、Selenium Coreには多くのコマンドが用意されている。これらのコマンドはJavaScriptで実装されており、JavaScriptで独自のコマンドを追加することもできる。コマンドの詳細やコマンドの追加方法は、次に示すWebページを参照してほしい。

- **Selenium Core Reference**
http://seleniumhq.org/projects/core/reference.html

コマンド引数

続いて、コマンドに渡す引数の種類について解説する。

エレメントロケータ

コマンドの対象となるHTML要素を指定する（**表4**）。例えば、clickコマンドではクリックするHTML要素を指定する。

エレメントロケータでHTML要素を正確に指定できるようにするため、HTML要素にはid属性を設定するようにしてほしい。

文字列マッチングパターン

文字列マッチングパターンとは、Assertionコマンドで文字列を比較するための式である。主な文字列パッチングパターンを**表5**に示す。

Selenium IDEの使い方

すでに、Seleniumテストケースの中身について解説した。これらをテキストエディタなどを使って手書きで記述するのは面倒だ。そこで、Selenium IDEを使ってテストケースを自動で作成してみよう。

Selenium IDEは、Firefoxの操作をコマンドとしてSeleniumテストケースに記録するキャプチャリプレイツールである。Selenium IDEを使えば、Firefoxの操作を自動で記録してSeleniumテストケースを作成できる。

表3：主なAssertionコマンド

コマンド名	概要	第1引数(target)	第2引数(value)
assertTitle verifyTitle waitForTitle	Webページのタイトルを検証	文字列マッチングパターン	
assertElementWidth verifyElementWidth waitForElementWidth	指定したHTML要素の幅を検証	エレメントロケータ	文字列マッチングパターン
assertText verifyText waitForText	指定したHTML要素のテキストを検証	エレメントロケータ	文字列マッチングパターン
assertValue verifyValue waitForValue	指定したHTMLの入力要素の入力値を検証	エレメントロケータ	文字列マッチングパターン

表4：エレメントロケータの書式

書式	例
id=「HTML要素のid属性」	id=value1、またはvalue1
name=「HTML要素のname属性」	name=calcform
dom=DOMで指定	dom=document.calcform.value1、またはdocument.calcform.value1
xpath=XPathで指定	xpath=//input[@value='計算']、または //input[@value='計算']
link=リンク文字列	link=Topへ戻る

表5：文字列マッチングパターン

比較方法	書式	例
“*”や“？”といったワイルドカードが使用可能	glob:比較文字列	glob:*結果*、または*結果*
完全一致（ワイルドカード使用不可）	exact:比較文字列	exact:計算結果
JavaScript正規表現	regexp:正規表現	regexp:\d*

Selenium IDEのインストール

Selenium IDE（執筆時点のバージョンは1.0 beta 2）を利用するには、まずFirefoxにSelenium IDEをインストールしなければならない。

これには、まずFirefoxから次のURLにアクセスする。

http://seleniumhq.org/download/

続いて、アクセスすると表示されるページにあるSelenium IDEの「Download」をクリックする。後は、指示に従って操作してFirefoxを再起動すれば、インストール完了である。

次に、Webブラウザの画面キャプチャを保存するために「Screengrab!」（原稿執筆時点のバージョンは0.95）をFirefoxにインストールする。

Firefoxから次のURLにアクセスする。

https://addons.mozilla.org/en-US/firefox/addon/1146/

続いて、アクセスすると表示されるページにある［Download Now］ボタンを押す。後は、指示に従って操作しFirefoxを再起動すれば、インストール完了である。

Selenium IDEでSeleniumテストケースを作成

それでは、早速Seleniumテストケースを作成してみよう。Selenium IDEによるSeleniumテストケースの作成手順は、次のとおりである。

手順1 Actionコマンドを追加する
手順2 検証コマンドを追加する
手順3 完成したSeleniumテストケースを実行する
手順4 Seleniumテストケースをファイルに保存する

ここでは、オンラインブックショップ「SE.shop.com」の機能テストを行なう。テスト項目を**表6**に示す。

表6：サンプルのテスト項目

操作	期待する結果
●http://www.seshop.com/にアクセスする ●検索キーワードに「DBマガジン」と入力し、[検索]ボタンを押下する	●商品検索画面へ遷移し、タイトルが「SE Shop.com／検索」となること ●検索する言葉の入力ボックスに「DBマガジン」と格納されていること

手順1 Actionコマンドを追加する

まず、Firefoxのメニューから[ツール]－[Selenium IDE]を選択する。すると、Selenium IDE(**画面1**)が起動する。

次に、Selenium IDEの記録ボタンが押下されていることを確認する。記録ボタンが押下されていたら、Webブラウザの操作をActionコマンドとしてSeleniumテストケースに追加する。

ここでは、Firefox上で以下の操作を行なう。

- http://www.seshop.com/にアクセスする
- 検索キーワードに「DBマガジン」と入力し、[検索]ボタンを押下する

すると、上記の操作はSeleniumテストケースにActionコマンドとして追加される(**画面2**)。

手順2 検証コマンドを追加する

Selenium IDEの「コマンド」と「対象」に次の値を入力する。

コマンド：captureEntirePageScreenshot
対象：c:\selenium\screen.png

captureEntirePageScreenshotは、Webブラウザの画面キャプチャを指定ファイルに保存するコマンドである。

続いて、Selenium IDEの「コマンド」と「対象」に次の値を入力する。

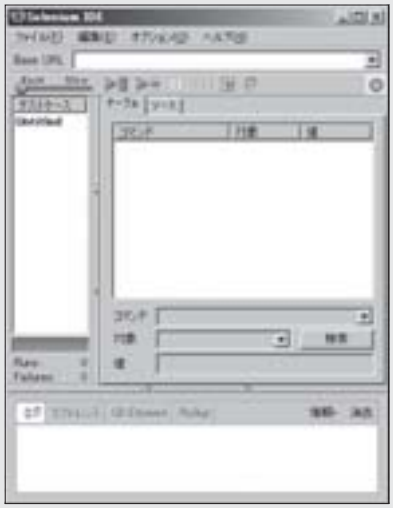
コマンド：assertTitle
対象：SE Shop.com／検索

assertTitleは、タイトルの文字列をチェックする検証コマンドである。assert～コマンドなので、検証失敗(タイトルが違った)の場合はテストが終了する。

最後に、FirefoxからSE.shop.comの「検索する言葉」入力ボックス上で右クリックし、[verifyValue keyword DBマガジン]メニューを選択する(**画面3**)。

このように、Selenium IDEはコマンド入力だけでなく、HTMLの項目を選択して適切な検証コマンドを追加することも可能だ。

手順1と2までの作業を終えたら、Seleniumテストケースは完成となる。Selenium IDEの記録ボタンを押下し、操作記録を停止する。


画面1：Selenium IDEの画面構成


画面2：追加されたActionコマンド


画面3：verifyValueコマンドを追加

手順3 完成したSeleniumテストケースを実行する

それでは、Seleniumテストケースを実行してみよう。

これには、Selenium IDEの「現在のテストケースを実行」ボタンを押下する。すると、Seleniumテストケースに出力されたコマンドに従って、テストが実施される。

テストが成功すると、Selenium IDEのSeleniumテストケースが緑色で表示される。さらに、captureEntirePageScreenshotコマンドによる画面キャプチャの画像ファイルも出力される。

手順4 Seleniumテストケースをファイルに保存する

Selenium IDEで作成したSeleniumテストケースファイルと、Seleniumテストスイートファイルをファイルに保存する。

Seleniumテストケースファイルを保存するには、Selenium IDEの［ファイル］－［テストケースを保存］メニューを選択する。一方のSeleniumテストスイートファイルを保存するには、Selenium IDEの［ファイル］－［テストスイートを保存］メニューを選択する。

このようにSelenium IDEを使えば、簡単にSeleniumテストケースを作成することが可能だ。

テストの実施

続いて、作成したSeleniumテストケースを使って機能テストを行なってみよう。これには、Selenium RCを利用する。

すでに紹介したとおり、Selenium RCとは、Seleniumテストケースやプログラム言語から自動テストを実施するためのプログラムである。

Selenium RCの構成は、**図3**のような構成となっている。以降では、Selenium RCの核となる「Selenium Server」と「Client Driver」について紹介しよう。

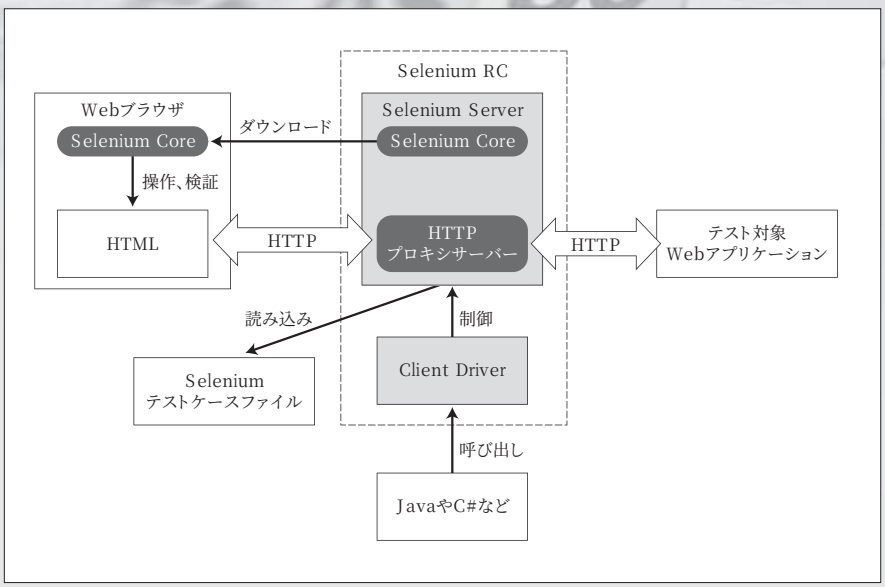
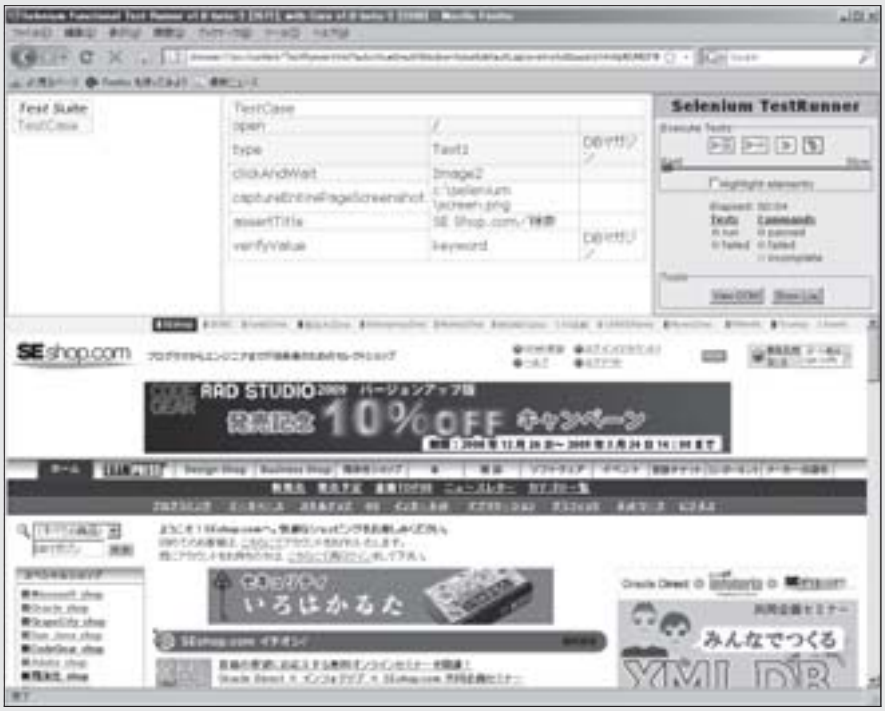


図3：Selenium RCの構成



画面4：テスト実施中の画面

Selenium Server

Selenium Serverは、テスト実行環境のSelenium CoreをWebブラウザに読み込ませるためのHTTPプロキシサーバーである。

Selenium Serverが起動すると、Webブラウザが起動され、WebブラウザにSelenium Coreを読み込ませる。Selenium CoreがSeleniumテストケースファイルを読み込んで、テストを実施する（**画面4**）。

なお、Selenium Serverはselenium-server.jarというJARファイルで提供され、実行にはJRE5.0以降が必要だ。

Client Driver

Client Driverは、Seleniumテストケースファイルの代わりにプログラム言語からSelenium ServerやWebブラウザを制御して、テストを実施するためのライブラリだ。

対応言語は、Java、C#など.NET、Perl、PHP、Ruby、Pythonとなっている。例えば、Javaから実行する場合はselenium-java-client-driver.jarをクラスパスに定義する（**LIST3**）。

Selenium RCのインストール

それでは、以下のURLにアクセスしてSelenium RCをダウンロードしてみよう。

http://seleniumhq.org/download/

ダウンロードしたZIPファイルを解凍し、selenium-server.jarをSeleniumテストケースファイルと同じディレクトリにコピーする。

selenium-server.jarには実行に必要なファイルがすべて含まれているため、ほかに必要なJARファイルはない。

LIST3：Client DriverのJava実装例

```
package com.example.tests;

import com.thoughtworks.selenium.*;
import java.util.regex.Pattern;

public class NewTest extends SeleneseTestCase {
  public void setUp() throws Exception {
    setUp("http://change-this-to-the-site-you-are-testing/", "*chrome");
  }
  public void testNew() throws Exception {
    selenium.open("/");
    selenium.type("Text1", "DBマガジン");
    selenium.click("Image2");
    selenium.waitForPageToLoad("30000");
    selenium.captureEntirePageScreenshot("c:¥¥Selenium¥¥screen.png", "");
    assertEquals("SE Shop.com／検索", selenium.getTitle());
    verifyEquals("DBマガジン", selenium.getValue("keyword"));
  }
}
```

画面5：テスト結果HTMLファイル

表7：テスト対象ブラウザの指定

値	起動するブラウザ	複数ドメインにまたがるテスト
*iexploreproxy	Internet Explorer	×
*piiexplore	Internet Explorer	○
*iexplore *iehta	Internet ExplorerをHTMLアプリケーションモードで起動	○
*firefoxproxy	Firefox	×
*pifirefox	Firefox	○
*firefox *chrome	FirefoxをChromeモードで起動	○
*googlechrome	Google Chrome	×
*safari	Safari	×
*opera	Opera	×

Column

Webブラウザの画面キャプチャ

Selenium RC 1.0 beta 2では、FirefoxとInternet Explorerに限り、Webブラウザの画面キャプチャを画像ファイルに保存できる。画面キャプチャされる範囲は、画面に表示されている部分だけではなく、スクロールで隠れている部分も含めてHTML全体が取り込まれる。

Firefoxの場合、Firefoxアドオンの Screengrab!をインストールして、selenium-server.jarのテスト対象ブラウザに「*chrome」を指定する。一方のInternet Explorerの場合、ActiveXコントロールの「snapsIE」(http://sourceforge.net/projects/snapsie/)をインストールして、Internet Explorerのセキュリティ設定で「スクリプトを実行しても安全だとマークされていないActiveXコントロールの初期化とスクリプトの実行」を有効にし、selenium-server.jarのテスト対象ブラウザに「*iexploreproxy」または「*piiexplore」を指定する。

なお、Seleniumテストケースファイルからテストを実施する場合、Client Driverは不要だ。

Selenium Serverでテスト実施

Selenium Serverを起動するには、selenium-server.jarを実行するだけだ。selenium-server.jarの実行は、javaコマンド、AntまたはMavenなどから行なう。

javaコマンドから実行する場合の書式は、次のとおりだ。

```
java -jar selenium-server.
jar -htmlSuite ⇒
「テスト対象ブラウザ」 ⇒
「テスト対象Webアプリのドメイン名」⇒
「Seleniumテストスイートファイルの絶対パス」⇒
「テスト結果HTMLファイル」
```

※誌面の都合により⇒で改行

上記の書式にある「テスト対象ブラウザ」に指定できる値は、**表7**のとおりだ。

Firefoxの場合

まず、Firefoxでテストを実施してみよう。

```
java -jar selenium-server.
jar -htmlSuite ⇒
"*chrome" "http://www.seshop.com/" ⇒
"c:¥selenium¥TestSuite.html" ⇒
"result_Firefox.html"
```

Firefoxでテストする場合、通常は「*firefoxproxy」と指定するが、captureEntirePageScreenshotコマンドを使うので「*chrome」と指定する。

javaコマンドを実行すると、Firefoxが起動してテストが自動的に実施される。テストが完了すると、テスト結果のHTMLファイル(**画面5**)と、captureEntirePageScreenshotコマンドによる画面キャプチャの画像ファイル(**画面6**)が作成される。

Internet Explorerの場合

次に、同じSeleniumテストケースをInternet Explorerで実施してみよう。Internet Explorerでテストする場合、ポップアップブロックを解除しておく必要がある。

画面6：画面キャプチャした画像ファイル

```
java -jar selenium-server.
jar -htmlSuite ⇒
"*piiexplore" ⇒
"http://www.seshop.com/" ⇒
"c:¥selenium¥TestSuite.html" ⇒
"result_Firefox.html"
```

上記javaコマンドを実行すると、Internet Explorerが起動してテストが実施される。テスト実行後、Firefoxと同様にテスト結果HTMLファイルと画面キャプチャした画像ファイルが作成される。

このようにSelenium RCを使えば、特別な設定をしなくても機能テストの自動化が行なえる。さらに、Selenium Gridを使えば、WindowsやMacOSなど各種プラットフォームのWebブラウザの動作検証を一括して行なうことが可能だ。

まとめ

本稿では、Seleniumの基本的な使い方について紹介した。

テストの自動化を成功させるポイントは、「テストケースの作成と保守コストを抑える」ことである。Seleniumならテストケースの作成を手軽に行なえるため、作成コストを抑えることができる。しかし、大量のテストケースを作成すると、仕様変更時にテストケースの保守コストがかかってしまう。保守コストを抑えるために、繰り返し実施するテストケースだけをSeleniumで自動化すると良い。みなさんもSeleniumを活用して、ソフトウェアの品質向上に役立ててほしい。 DBM

本誌CD-ROMに、本稿のサンプルコードを収録しています。またDBマガジンWebサイト(http://www.shoeisha.com/mag/dbm)からもダウンロードできます。

田原 孝(たはら たかし)
計測器メーカーや外資系ツールベンダーにて、ファームウェア開発、Webアプリケーション開発、開発ツールサポートなど数多くの現場を経験。冬になるとオーロラを見にカナダに出かけることが多い。