Selenium IDE と rspec で簡単自動テスト

Web アプリケーションの自動テストと言えば「Selenium」。Ruby を使用して簡単な操作方法を紹介します。特に、自動化プログラムの中で CSV ファイルを取り込む記述方法については、検索してもなかなか無かったため、是非、参考にしていただければと思います。

1. Tools

Selenium はテスト方法に応じて、幾つかバリエーションが用意されていますが、簡単にテストしたい場合には、Firefox のプラグインの利用がお勧めです。

- Firefox
- Selenium IDE:
  簡単に Selenium 用のスクリプトを作成できる Firefox プラグインです。ブラウザの操作を記録し、Selenium 用のスクリプトを生成する。ただし、Web アプリの動きが複雑な場合、自動作成したスクリプトを修正する必要があります。
- RubyGems
  - http://rubyinstaller.org/downloads から Ruby をインストール
  - http://rubyinstaller.org/downloads から Development DevKit-mingw***-sfx.exe をインストール
- Selenium-webdriver
  作成したスクリプトの実行をブラウザの拡張機能を使用して行うライブラリです。
  Gem を使用して、以下のコマンドで簡単にインストールできます。
  C:¥> gem install selenium-webdriver

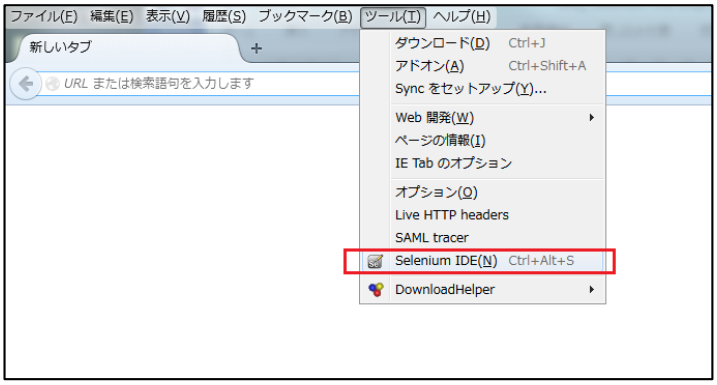
- rspec
  テスト対象の「振舞い」を記述することで単体テストを行うテストフレームワークです。
  Ruby の Domain Specific Language の機能を活用したものだが、言語的には Ruby とは異なる部分もあります。同じく gem でインストールできます。
  C:¥> gem install rspec

今回のテストケースは Web アプリケーション（以下、WebApp（※））へのログインテストを自動化します。以下の図がテストを行うイメージとなります。
※今回、弊社クラウドサービス「SeciossLink」をテストに利用します。

2. テスト実行

2.1 Selenium IDE 記録開始。

ツールバーから SeleniumIDE を選択し、IDE は記録開始した状態で起動します。

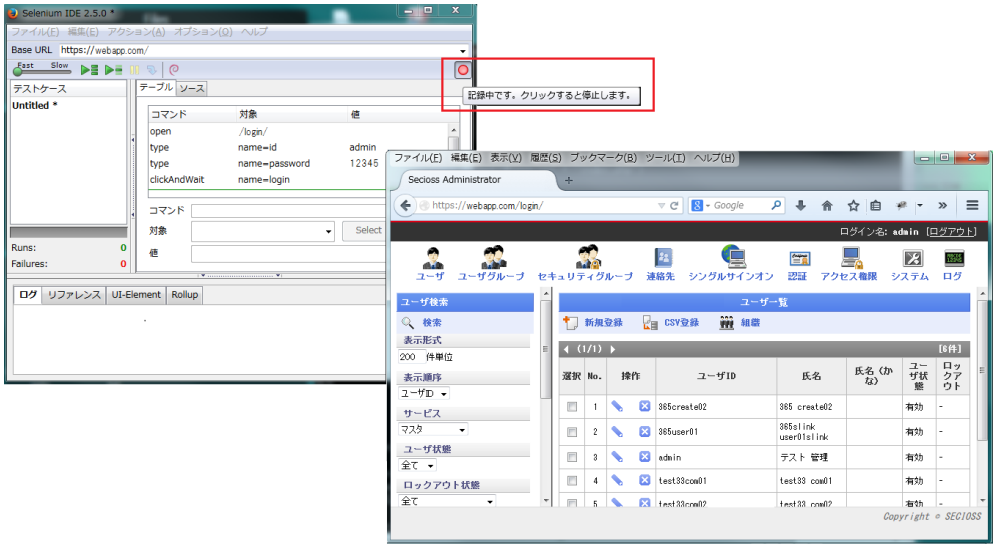


2.2 Web App 正常系実行

Selenium IDE の Base URL に WebApp の base としたい URL を設定し、WebApp に通常のログインをします。

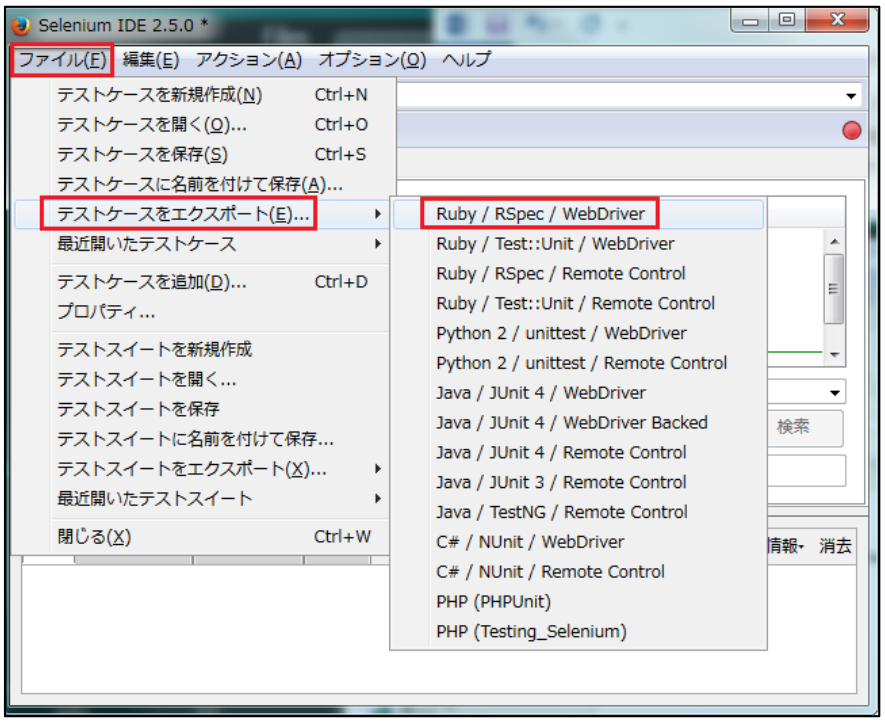
2.3 Selenium IDE 記録停止

WebApp にログインする際、入力値などの情報が IDE に表示されます。ログイン完了後、記録を停止します。



2.4 テストケースエクスポート

IDE の「ファイル」からテストケースを「Ruby/Rspec/WebDriver」でエクスポートします。



エクスポートしたファイルを拡張子“.rb”にして保存します。内容は以下のように編集します。

```
require "json"
require "selenium-webdriver"
require "rspec"
include RSpec::Expectations

describe "SeleniumTest" do

  before(:each) do
    @driver = Selenium::WebDriver.for :firefox
    @base_url = "https://webapp.com"                    ←web アプリのホスト名
    @accept_next_alert = true
    @driver.manage.timeouts.implicit_wait = 30
    @verification_errors = []
  end

  after(:each) do
    @driver.quit                                ←web アプリのテストが終了後、ブラウザを自動に閉じる
    @verification_errors.should == []
  end

  it "test_selenium" do              ←web アプリのテスト処理開始
    @driver.get(@base_url + "/login/")                          ←web アプリ login ページを get
    @driver.find_element(:name, "id").clear
    @driver.find_element(:name, "id").send_keys "admin"         ←login ページにある id を get,
                                                                  そこにログイン名"admin"を入力する
    @driver.find_element(:name, "password").clear
    @driver.find_element(:name, "password").send_keys "12345"←password 見つけて、"12345"入力
    @driver.find_element(:name, "login").click          ←login ボタンをクリック
  end                                 ←web アプリのテスト処理終了

  def element_present?(how, what)
    ${receiver}.find_element(how, what)
    true
  rescue Selenium::WebDriver::Error::NoSuchElementError
    false
  end

  def alert_present?()
    ${receiver}.switch_to.alert
    true
  rescue Selenium::WebDriver::Error::NoAlertPresentError
    false
  end

  def verify(&blk)
    yield
  rescue ExpectationNotMetError => ex
    @verification_errors << ex
  end

  def close_alert_and_get_its_text(how, what)
    alert = ${receiver}.switch_to().alert()
    alert_text = alert.text
    if (@accept_next_alert) then
      alert.accept()
    else
      alert.dismiss()
    end
    alert_text
  ensure
    @accept_next_alert = true
  end
end
```

ページにログインするだけですので簡単にテスト実行できますが、複雑な動きをする場合、WebApp のページソースを見ながら、ターゲットの入力する id や name を取得する修正が必要になります。

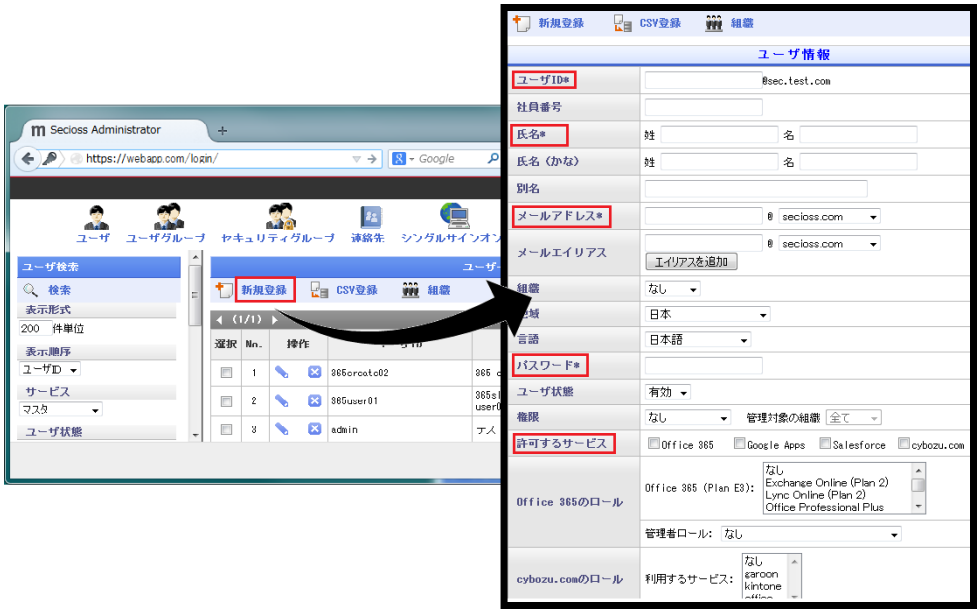
2.5 rspec テストケース実行

保存したテストケースファイルコマンドプロンプトで以下実行すれば、Firefox が自動に立ち上がり、WebApp を開き、テストケースに記載したユーザでログインを始めます。

3. CSV によるデータ投入

ログインテストで生成されたテストケースに、CSV ファイルからアカウントデータを取り込む記述を追加してみます。Web アプリケーションの作りによって異なりますが、弊社クラウドサービス「SeciossLink」では、例えば、Office365 との接続テストを行う場合には、テストユーザの作成が必要です。この操作を自動化できると、テストデータ作成がとても楽になります。

手動でアカウントを作成する場合、図のように「新規登録」をクリックして、ユーザ登録を行います。



Selenium IDE から出力したテストケースに CSV ファイルを取り込む記述を追加します。ソースが長くなりますので、CSV ファイル取り込みの部分だけ紹介します。また、取り込む CSV ファイルのフォーマットは弊社サービスの場合、以下となります。

uid	snkanji	gnkanji	Mail	password	allowedservice
sele01	sele	test01	seletest01	Passwd01	office365
sele02	sele	test02	seletest02	Passwd02	office365
sele03	sele	test03	seletest03	Passwd03	office365

```
require "json"
require "selenium-webdriver"
require "rspec"
require "csv"
include RSpec::Expectations
--省略--
csvfile = "C:¥add_User.csv"                    ←csv ファイル指定
describe "seciosslink user add" do
   before(:all) do
        --省略--
      @data = Hash::new                        ←csv データを取り込む
      reader = CSV.open(csvfile, "r")
      header = reader.take(1)[0]
      header.each do |attr|
          @data[attr.strip] = []
      end
      reader.each do |row|
          i = 0
          row.each do |item|
              @data[header[i].strip].push(item.strip)
              i = i + 1
          end
      end
      @data_length = @data[header[0].strip].size - 1 ←csv ファイルレコード数を計算
      reader.close
   end
     --省略--
   it "User add" do
     --省略--
        for n in 0..@data_length do     ←複数データを処理
          Creat_User(@data["uid".strip][n], @data["snkanji".strip][n],
@data["gnkanji".strip][n], @data["mail".strip][n], @data["password".strip][n],
@data["allowedservice".strip][n])
        end
        @driver.get(@base_url + "index.php")
   end
     --省略--
   def Creat_User( uid,empnum,snkanji,gnkanji,snkana,gnkana,aliasname,mail,mailalias,
slang,password,status,userrole,allowedservice,noticemail)
      @driver.find_element(:id, "create_link").click
      @driver.find_element(:id, "attr_newid").clear
      @driver.find_element(:id, "attr_newid").send_keys uid     ←ユーザ ID
      @driver.find_element(:id, "attr_newsnkanji").clear
      @driver.find_element(:id, "attr_newsnkanji").send_keys snkanji     ←姓
      @driver.find_element(:id, "attr_newgnkanji").clear
      @driver.find_element(:id, "attr_newgnkanji").send_keys gnkanji     ←名
      @driver.find_element(:id, "attr_newmailuser").clear
      @driver.find_element(:id, "attr_newmailuser").send_keys mail     ←メールアドレス
      @driver.find_element(:id, "attr_newpassword").clear
      @driver.find_element(:id, "attr_newpassword").send_keys password ←パスワード
      if (allowedservice == "office365") then
      @driver.find_element(:id, 'newsync_1').click       ←許可サービス
      end
      @driver.find_element(:id, "create_button").click     ←繰り返しユーザ登録
      (close_alert_and_get_its_text()).should == "ユーザを登録します。よろしいですか？"
      sleep 2
   end


--省略--
```

編集した「.rb」ファイルを rspec コマンドで実行します。当然、取り込む CSV ファイルは指定のフォルダに用意しておいてください。

```
C:¥> rspec　実行ファイル名.rb
```

コマンドを実行すると、Firefox が起動→指定 URL へアクセス→ログイン→CSV ファイルを読み込んでアカウント作成、という流れが自動的に実行されます。

Selenium を利用するとテストの自動化だけではなく、テストデータの作成や、いつも使うアプリに自動的にログインする、なんていう事にも利用できます。

以上。