SRA の技術情報誌
GS LetterNeo
Vol.160　2021 年 11 月

Web アプリケーション 自動テスト支援ツール Testablish の紹介

米村　信之

ビジネスイノベーション事業部

テスト自動化はなぜ失敗するか

テスト自動化によってコスト削減や品質の維持管理をしたいと考えるユーザーはたくさんいますが、成功している事例が少ないように思います。

テスト自動化は何回も実行して初めてメリットが生まれます。しかしながら、検討の段階で諦めているケースも多く見受けられます。

また、せっかくテストスクリプトを整備してとりかかったものの途中でやめてしまったり、一回使って終わりといったように定着していない開発現場も多いのではないでしょうか。

これらテスト自動化の普及を阻むよく聞く課題として以下のものが挙げられます。

① **導入のコストが許容できない**
期間もコストも余裕がないため取り組みができないというプロジェクトは多いと思います。

ただ、長期にわたってメンテナンスするようなシステムや、マルチ OS/マルチブラウザをサポートするシステムなどは未来に発生するコストを考慮してもう一度検討しなおしても良いのではないでしょうか。部分的でも自動化しておくことでその分の工数の見返りは少なからずあるはずです。

② **作成した自動テストが動かなくなる**

不安定な物を前提にテストケースが作られている場合に多いと思われます。

- テスト実行する環境のパフォーマンスが一定でない
  - 同時に実行しているタスクや、ネットワーク状態、DB の負荷状況によって応答時間が異なりテストが止まってしまう
- データが変更されて想定するデータセットになっていない
  - データの初期化が不十分であったり、別のテストを並行して実施する場合などが考えられます
- 日付など可変のものを使っている
  - 可変部分は変数を使うなどの工夫が必要です
- DOM の構造が動的に変化する。あるいは操作対象のセレクタがビルドのたびに変わってしまう
  - オブジェクトの ID を固定で定義するとか、動的に変わってしまうセレクタでも規則性を把握して正規表現で指定するなどアプリ実装やテストケースの工夫で回避可能です

テスト自動化のメリット最大化するためには環境の整備やアプリ実装、テストケース設計の工夫が必要になります。

③ **ツールを導入したが効果が出ない**

ツールを導入したがうまく使いこなせず宝の持ち腐れになっており、自動化自体もあきらめてしまったなどで躓いているケースも良く見かけます。

- ツールの機能を使いこなすには習熟が必要
  - 単純なテストであればどのツールでも簡単にテストケースを作成することができると思いますが、複雑なテストを行う場合、ツールの特性を理解して習熟する必要があります。最初のうちは小さい単位のテストから始めて膨らませていくなかで習熟する必要があります。
- テスト設計は必要
  - テスト自動化を支援するツールはたくさん出回っていますが、テストケースの設計自体をサポートする製品はあまり多くありません。どんなテストを実施するかは従来通り人手で考える必要があります。

④ **すべてのテストを自動化したい**

すべてのテストが自動化できるわけではありません。手作業でのテストは少なからず残ることを想定して、”何回も実行する”、”正常ケースだけ一通り流す” などもっとも効果のありそうな部分のテストから自動化しましょう。

⑤ **思ったテストができない**

”画面のレイアウトが崩れていないか”、“このケースの時はこう、こっちはこうなど、判断条件が複雑なテスト”、”サブシステムと正しく連携できているか“　など人間の判断が必要なテストができない場合もあります。

テストツールの機能が足りない場合は回避するか、別のツールやプログラムを組み合わせて実現できるか検討することも必要です。

開発中ならツールがサポートする機能の範囲内でアプリの設計を行うなどテスト自動化しやすい設計にすることも検討してみましょう。

⑥ **テストのメンテナンスができない**

テストを作成した人がプロジェクトから離脱してしまいメンテナンスができないなど属人化してしまう。テストの目的や処理内容をドキュメント化する必要があるため工数がかかるなどの問題があります。

テスト自動化を効果的に実践するためには上記のような課題を認識して割り切れるところとできないところ、費用対効果の考え方を整理したうえで取り組むことが必要です。

この取り組みにおいてテスト自動化をサポートするツールが非常に役に立ちます。次の章では弊社で開発・販売している Testablish を紹介しています。これらの課題解決を支援する機能を提供しておりますのでぜひとも導入をご検討頂ければ幸いです。

Testablish の特徴

Testablish[1]は弊社で開発している Web アプリケーションのテスト自動化ツールです。Web アプリケーションのテストは通常、テスト仕様書を作成してそれに沿って操作を実行してその期待結果を人間が判断して進めますが、Testablish を利用すればこれらを自動実行することができます。

また、前章のような課題も解決できるよう開発しており、以下のような特徴があります。

SELENIUM[2]のコードを自動生成

GUI で編集したテストシナリオから Selenium のテストコードをボタン一つで自動生成します。

テストは Selenium の環境さえあればどこでも実行でき、多重実行環境も構築できます。Selenium はオープンソースですので実行環境のライセンスを購入する必要もありません。

WEB ブラウザで利用可能

Testablish は Node.js で作成されており、Web ブラウザで利用することが可能です。共用サーバーにインストールすれば複数人で利用することが可能です。これによりテスト作成者の PC はそれほど高スペックである必要はありません。

また、一つのテストを複数のユーザーでメンテナンスすることができます。

キャプチャ&リプレイとリポジトリ方式に対応

"キャプチャ"とはブラウザ上の操作を記録することです。記録した操作内容を呼び出して再生することにより自動テストを行います。この機能はいろいろなテストツールでサポートされています。

一方リポジトリ方式はページ内に配置されているオブジェクトを登録しておき、この情報を元にテストを組み立てます。これによりアプリケーションが動いていなくてもテストを作成することができ、ソフトウェア結合の段階ですぐに自動テストが行えるようになります。

Testablish はこれら両方の方式を採用しており、簡単な使い方からより高度な使い方まで幅広く応用することが可能です。

[1] https://www.sra.co.jp/testablish/

[2] ウェブブラウザをプログラムで制御するためのライブラリ https://www.selenium.dev/ja/documentation/

わかりやすいテスト編集画面

GUI でテストの編集が行なえます。画面遷移を確認しながら、どの画面でどの操作を行ってどのようなアサーションを実行するかをマウスの操作で編集できます。

Testablish での課題解決

前述のよくある課題に対して Testablish では以下のような機能を提供しています。

課題	Testablish の対応
作成したテストが動かなくなる	変数や Javascript の埋め込みの機能があり、動的に変わる値にも対応できます。またシステムコマンドの呼び出しもサポートしていますので DB の初期化を行ってからテストを実行することも可能です。
すべてのテストを自動化したい	たくさんのテストケースに対応するために同じシナリオで色々なデータパターンで実行するテストパターンが Excel で作成できる機能を提供しています。
ツールを導入したが効果が出ない	テストケースの編集はウェブ画面の GUI で行いますので、習熟は容易です。
思ったテストができない	上記『**作成したテストが動かなくなる**』の機能を応用することにより様々なテストが可能です。また、作成したテストケースを CI ツールなどから API で連携して夜間自動実行の環境を構築することもできます。
テストのメンテナンスは必要ない	テストケースは GUI 上で編集できますので修正は容易です。マルチユーザで利用できますので別の人が作成したテストケースを修正することもできます。 テストケースは excel,word で可読性のあるドキュメントとして生成できます。

Testablish のシステム構成

Testablish のシステム構成は以下のようになります。

多種のオープンソースを利用して開発されており、構成管理ツールや CI/CD ツールとの相性も抜群です。

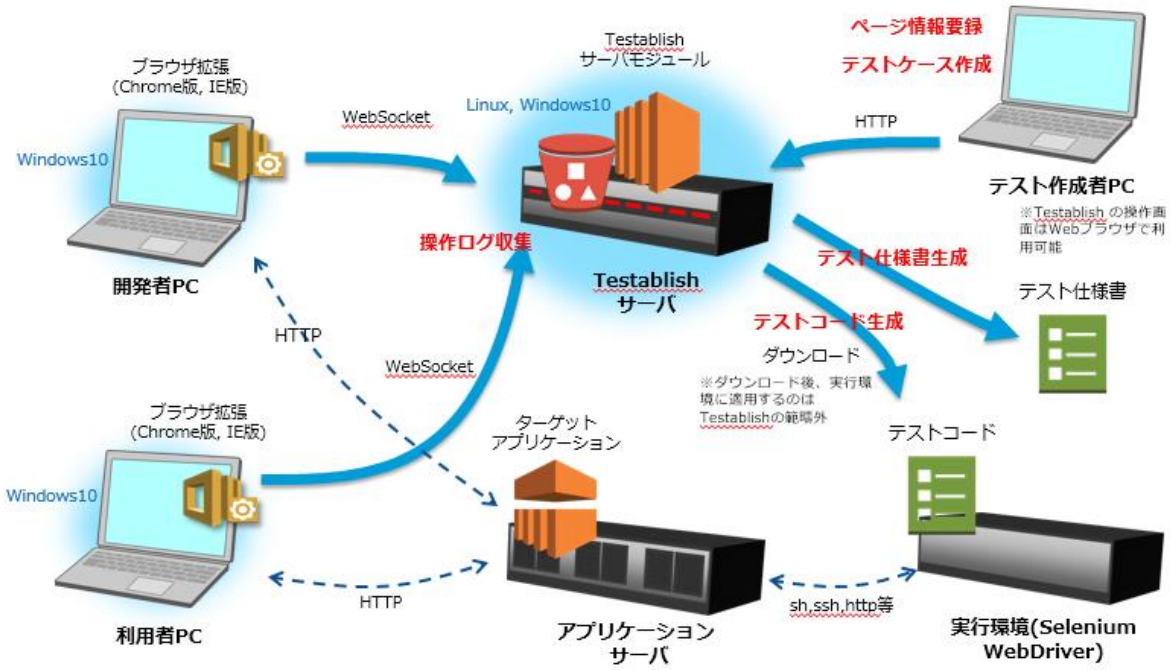


図 *1.Testablish* のシステム構成図

TESTABLISH サーバー

Testablish はオンプレミスで準備したサーバーにインストールする形になります。サーバーOS は Windows10, CentOS, RHEL での稼働実績があります。

Testablish サーバーは複数のオープンソースのモジュールを利用しており、それぞれ Docker[3]コンテナとして配置されます。サーバーのモジュールは以下の図のような構成で配置されます。

[3] https://www.docker.com/

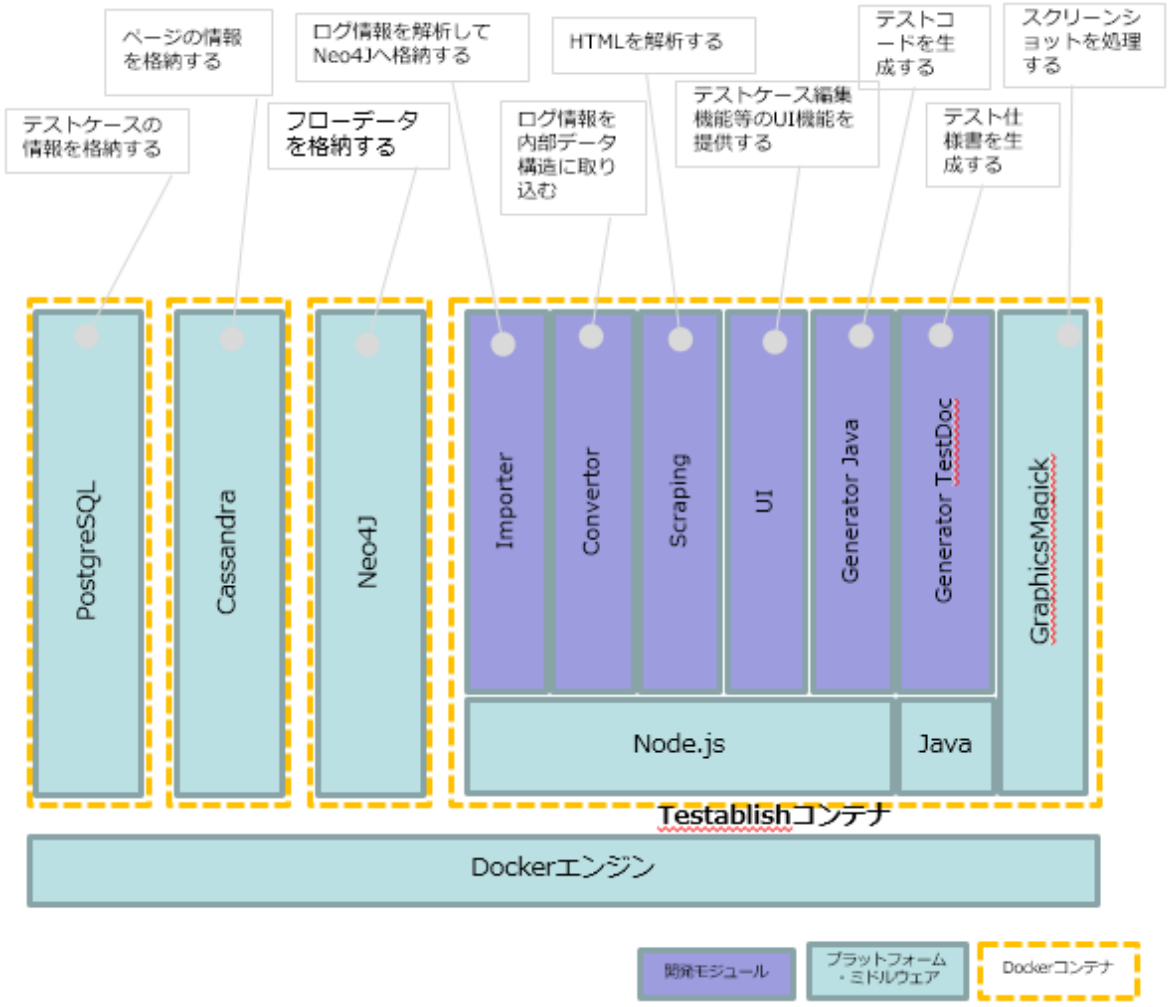


図 2.サーバーのモジュール構成

テスト作成者 PC

Testablish サーバーにアクセスしてテストケースを編集します。Web ブラウザで利用できますので特別なソフトウェアは必要ありません。

利用者・開発者 PC

ブラウザ上の操作を記録してテストケースを作成する場合、ブラウザの機能拡張モジュール(WebExtension)をインストールします。この WebExtension によりブラウザ上の操作をサーバーに収集し、その情報をもとにテストケースを作成することができます。

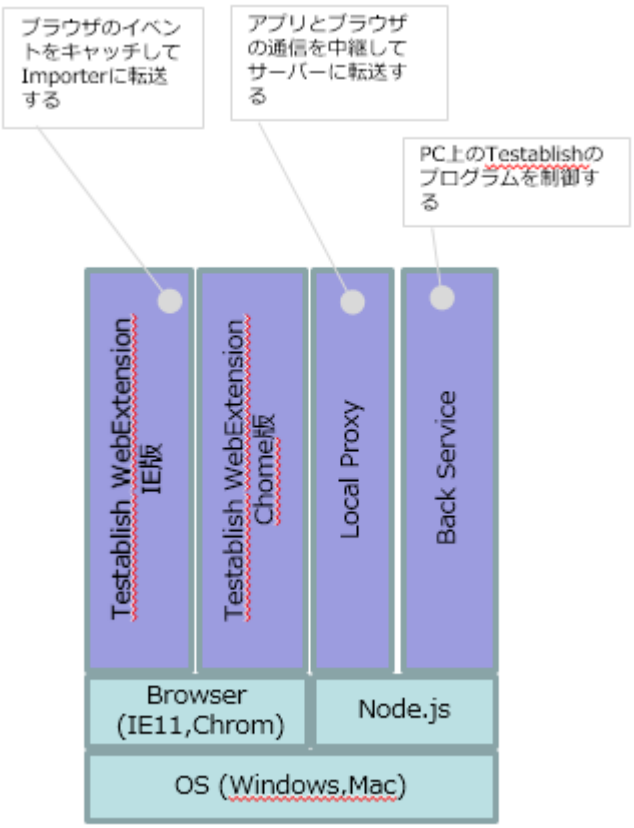


図 *3.WebExtension* 構成

実行環境

Testablish サーバーから出力されるテストコードを実行する環境です。テストで利用するブラウザに対応した Selenium の WebDriver をインストールしてテストコードを実行します。Windows, MacOS, Linux など WedDriver が対応する環境で動作します。

テストコードは Java のプログラムとして出力されるので Java のインストールも必要です。

Windows10 の環境であれば付属のインストーラーによって実行環境をセットアップすることができます。

利用イメージ

Testablish を利用した自動テストの実行イメージは以下のようになります。

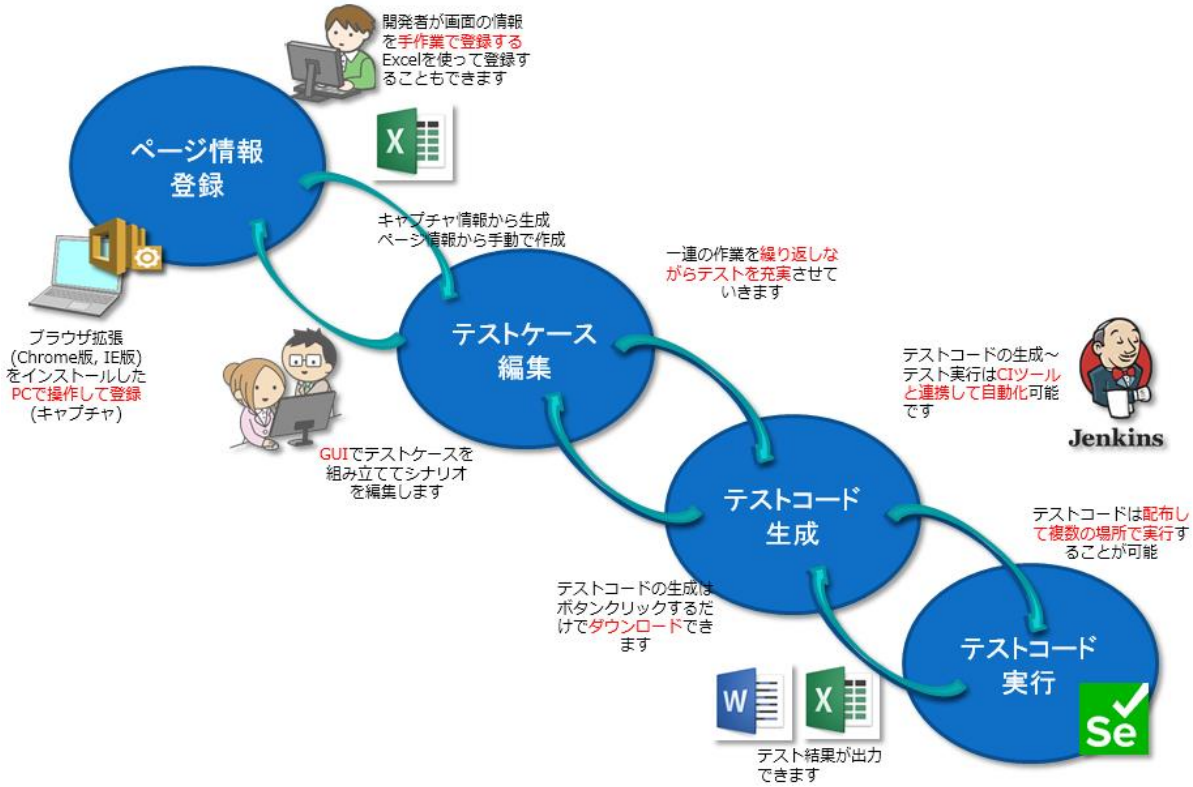


図 *4.Testablish* 利用イメージ

ページ情報登録

ページ情報の登録は 2 通りの方法があります。これらの両方の使い方ができるのは Testablish の特徴の一つでもあります。

1. WebExtension で操作を記録する方法
   所謂キャプチャ&リプレイ方式のキャプチャに当たります。テスト対象のアプリケーションを実際に操作している内容を記録してそれをテストケースにする方法です。Testablish の WebExtension により記録した内容はページ情報としてサーバー上に蓄積されます。

2. 手作業で登録する方法
   Web ページ上のオブジェクト(HTML エレメント)を手作業で登録できます。上記キャプチャの方式と異なり、アプリケーションが動作していなくても画面設計書などからオブジェクトや操作の定義情報を登録して、テストケースを作成することにより実装と並行して自動テストの準備を進めることができます。

テストケース編集

登録したページ情報を使ってテストケースの編集を行います。テストケースの編集はどのページでどのような操作、アサーション(検査)を行うかのシーケンスを GUI 上で組み立てていきます。スクリーンショットを撮る設定やエレメントが現れるのを待つ時間などを各ステップで設定できます。

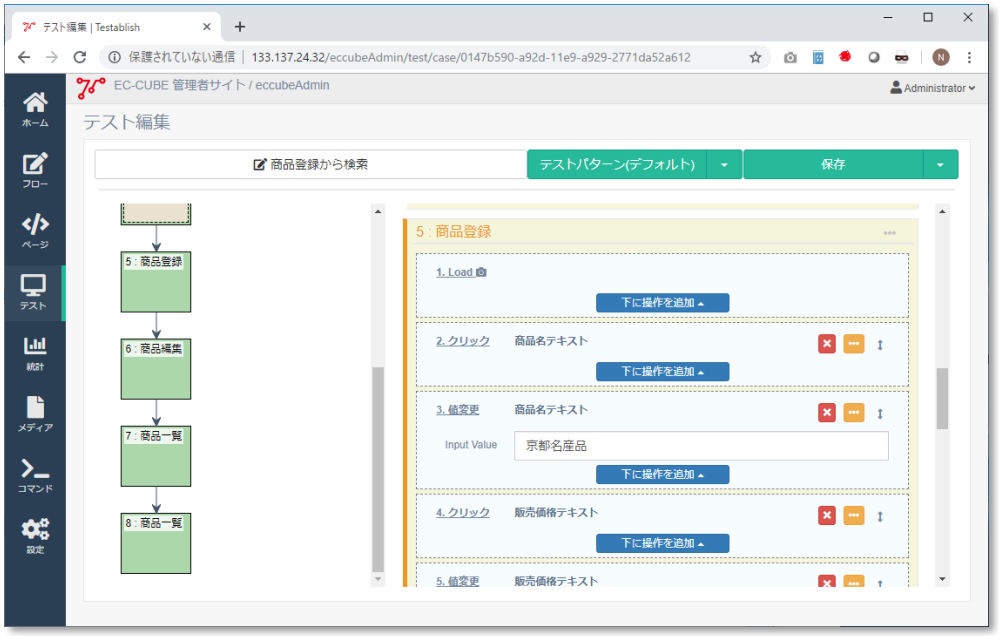


図 5.テストケース編集画面

Testablish では外部コマンドや Javascript の処理もシーケンスの一部として呼び出すことができます。DB を初期化するためのコマンドなどをテスト開始時に呼び出して常に同じ条件で実行したり、Javascript によってページ内のエレメントに介入して、画面上の操作では実行できないようなパラメータを送出するテストも実施することができます。

テストコード生成

テストケースを作成したら、保存してメニューから「テストコードの出力」を選択すると、テストコードとそれを実行するためのスクリプトファイルが入った zip ファイルがダウンロードされます。

図 *6.*テストコード生成メニュー

事前に設定しておけばテストコードと一緒に Microsoft Word, Excel 形式のテスト仕様書も生成することができます。このテスト仕様書はカスタマイズすることができますので、プロジェクト内のドキュメント様式に合わせることも可能です。

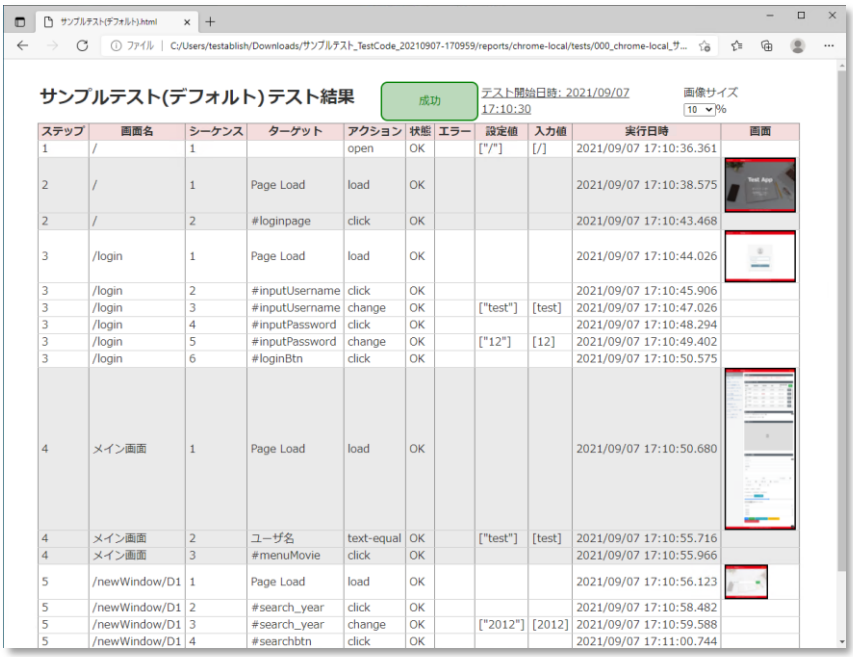
Testablish では同じ処理内容のテストをデータパターンを変えて実行することができます。パターンは画面上だけでなく Excel の帳票でも編集することができます。

テストコード実行

テスト実施者はダウンロードした zip ファイルを実行環境に配置し、解凍してテスト実行を行います。この操作は CI ツールなどと連携して自動実行させることができます。

解凍した Zip ファイルには Windows 用、Linux 用のスクリプトファイルが用意されているので、様々な環境でのテストが同じ zip ファイルで実行することができます。

テスト実行後、結果が HTML で出力されます。Microsoft Word,Excel のテスト仕様書も出力した場合はテスト結果内容が反映されてテスト結果書も出力されます。



サンプルテスト(デフォルト)テスト結果

成功

テスト開始日時: 2021/09/07 17:10:30

画像サイズ 10 %

ステップ	画面名	シーケンス	ターゲット	アクション	状態	エラー	設定値	入力値	実行日時	画面
1	/	1		open	OK		["/"]	[/]	2021/09/07 17:10:36.361	
2	/	1	Page Load	load	OK				2021/09/07 17:10:38.575	
2	/	2	#loginpage	click	OK				2021/09/07 17:10:43.468	
3	/login	1	Page Load	load	OK				2021/09/07 17:10:44.026	
3	/login	2	#inputUsername	click	OK				2021/09/07 17:10:45.906	
3	/login	3	#inputUsername	change	OK		["test"]	[test]	2021/09/07 17:10:47.026	
3	/login	4	#inputPassword	click	OK				2021/09/07 17:10:48.294	
3	/login	5	#inputPassword	change	OK		["12"]	[12]	2021/09/07 17:10:49.402	
3	/login	6	#loginBtn	click	OK				2021/09/07 17:10:50.575	
4	メイン画面	1	Page Load	load	OK				2021/09/07 17:10:50.680	
4	メイン画面	2	ユーザ名	text-equal	OK		["test"]	[test]	2021/09/07 17:10:55.716	
4	メイン画面	3	#menuMovie	click	OK				2021/09/07 17:10:55.966	
5	/newWindow/D1	1	Page Load	load	OK				2021/09/07 17:10:56.123	
5	/newWindow/D1	2	#search_year	click	OK				2021/09/07 17:10:58.482	
5	/newWindow/D1	3	#search_year	change	OK		["2012"]	[2012]	2021/09/07 17:10:59.588	
5	/newWindow/D1	4	#searchbtn	click	OK				2021/09/07 17:11:00.744	

図 7.テスト結果 *HTML*

最後に

Testablish を利用すれば Selenium によりテスト自動化は実現できますが、そのテストケース作成にはまだまだ相応の工数を掛ける必要があります。しかし、テストの実施に関しては無人で自動実施という計り知れないメリットがあり、昨今の DX を目指す取り組みにおいてテスト関連のイノベーションは間違いなく重要でチャレンジするべきテーマであると考えています。Testablish はその取り組みに寄与できるよう、ユーザー様の意見をいただきながら成長させていきたいと思います。

GSLetterNeo Vol.160
2021 年 11 月 20 日発行
発行者　株式会社 SRA 先端技術研究所

編集者　熊澤努　方 学芬
バックナンバー https://www.sra.co.jp/public/sra/gsletter/
お問い合わせ　gsneo@sra.co.jp

株式会社SRA
〒171-8513 東京都豊島区南池袋 2-32-8

夢を。Yawaraka Innovation
やわらかいのべーしょん