Tester24

Version.1.0

操作マニュアル

合同会社　ハイルディングシステム

2021/7/29

https://hyldingsystem.co.jp/

目次

１．はじめに 1
１－１．Tester24（テスター24）について 1
１－２．Tester24 の利用シーン 1
１－３．動作環境 2
１－４．ライセンスおよび免責事項 2
１－５．連絡先について 3
２．インストールとアンインストールについて 4
２－１．インストール 4
２－２．アンインストール 4
３．Tester24 を使ってみる 5
３－１．事前準備 5
３－２．サンプルを実行してみる 6
４．テストケースを作成する 12
４－１．テストケースの共通書式について 12
４－２．テストケース（シナリオ）を作成する 16
４－３．テストケース（マトリクス：横）を作成する 34
４－４．テストケース（マトリクス：縦）を作成する 42
５．スクリプトを作成する 52
５－１．URL を表示する 53
５－２．画面をキャプチャする 54
５－３．クリックする 56
５－４．入力する 58
５－５．属性を設定する 60
５－６．属性を追加する 61
５－７．待機する 62
５－８．値を取得して利用する 63
５－９．値を設定して利用する 65
５－１０．一連の処理を再利用する 67
５－１１．値を判定する 69
６．Tester24 でサポートされていない機能 72

１．はじめに

１－１．Tester24（テスター24）について

Tester24 は、Excel で作成したテストケースを Chrome 互換の組み込みブラウザにて自動で実行するツールです。テストケースに簡単なスクリプト（手順）を記述するだけで、任意の URL へアクセスして画面のキャプチャを行うことや、テキストボックスに値を入力して送信ボタンをクリックするといった操作を自動で行うことが可能です。

１－２．Tester24 の利用シーン

Web システムのテスト自動化と言えば「Selenium」が有名ですが、実行環境を整えるだけでも手間がかかり、スクリプト作成には Java の知識が必要となります。システム開発で既存部分を修正した際に、自動化されたリグレッションテストを実行することで、デグレートを防ぐ効果があることは分かりつつも、必要となるコストの高さや、作成した大量のスクリプトをメンテナンスできない現実を考えるとなかなか導入に踏み切れません。

これらのハードルを下げるべく Tester24 は開発されました。Selenium のような細やかな制御は行えませんが、必要最低限の機能を簡単なスクリプトで記述できるようにしております。その簡単なスクリプトを Excel のテストケースに追記することで、簡単に Web システムのテスト自動化が行える手軽さを備えています。

１－３．動作環境

OS	Microsoft Windows10 の動作する環境
メモリ	1.0GB 以上
HDD	出力する画面キャプチャの画像が格納できる程度
アプリケーションなど	OpenJDK 15.0.1 以上 .NET Core3.1 ランタイム

１－４．ライセンスおよび免責事項

・本ソフトウェアを利用する際は年間ライセンスの購入が必要となります。
ただし、試用期間を 90 日設けておりますので、試用期間中は自由にご利用いただけます。
※年間ライセンスについては以下を参照ください。
http://hyldingsystem.co.jp/製品/Tester24 各プラン料金/

・試用期間を超過後もご利用いただけますが、以下のように画面キャプチャに Tester24 のロゴが挿入されます。

・本ソフトウェアは安全に配慮して作成しておりますが、本ソフトウェア利用に伴う逸失利益、逸失売上もしくはデータの紛失、金銭的損失、または間接損害、特別損害、結果損害もしくは懲罰的損害について責任を負いません。

１－５．連絡先について

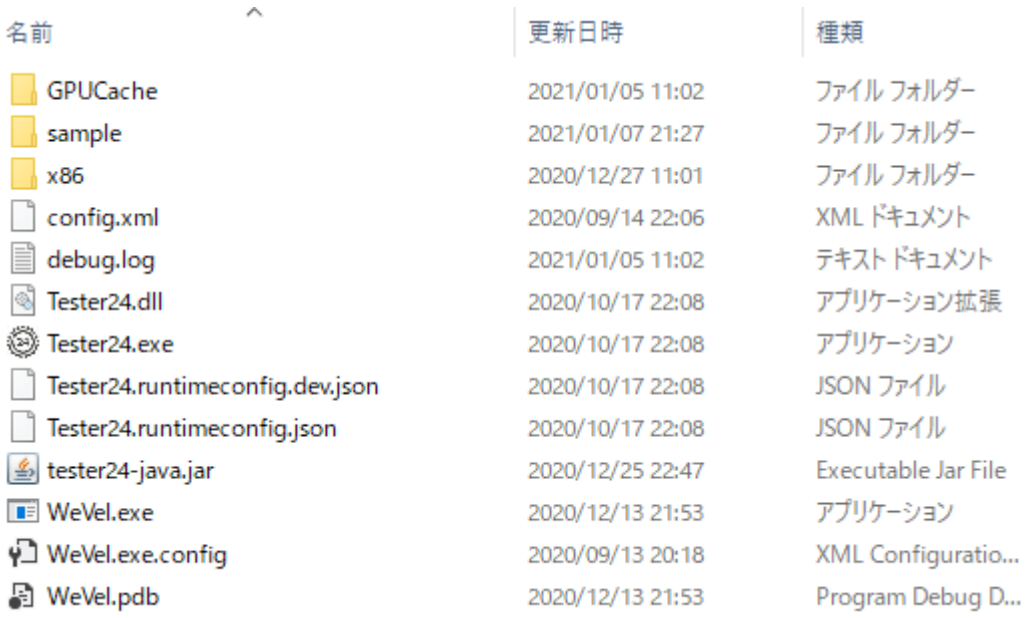
本ソフトウェアについてのご質問などについては、以下のメールアドレスまでお問い合わせください。

support@hyldingsystem.co.jp

２．インストールとアンインストールについて

２－１．インストール

Tester24 の Zip ファイルを解凍すると、以下のような各ファイルが存在します。



名前	更新日時	種類
GPUCache	2021/01/05 11:02	ファイル フォルダー
sample	2021/01/07 21:27	ファイル フォルダー
x86	2020/12/27 11:01	ファイル フォルダー
config.xml	2020/09/14 22:06	XML ドキュメント
debug.log	2021/01/05 11:02	テキスト ドキュメント
Tester24.dll	2020/10/17 22:08	アプリケーション拡張
Tester24.exe	2020/10/17 22:08	アプリケーション
Tester24.runtimeconfig.dev.json	2020/10/17 22:08	JSON ファイル
Tester24.runtimeconfig.json	2020/10/17 22:08	JSON ファイル
tester24-java.jar	2020/12/25 22:47	Executable Jar File
WeVel.exe	2020/12/13 21:53	アプリケーション
WeVel.exe.config	2020/09/13 20:18	XML Configuratio...
WeVel.pdb	2020/12/13 21:53	Program Debug D...

Tester24.exe	Tester24 のプログラム本体
sample	Tester24 で実行できるテストケースのサンプルを格納しております。スクリプトを作成する際の参考にしてください。

２－２．アンインストール

Tester24 のフォルダを削除してください。

３．Tester24 を使ってみる

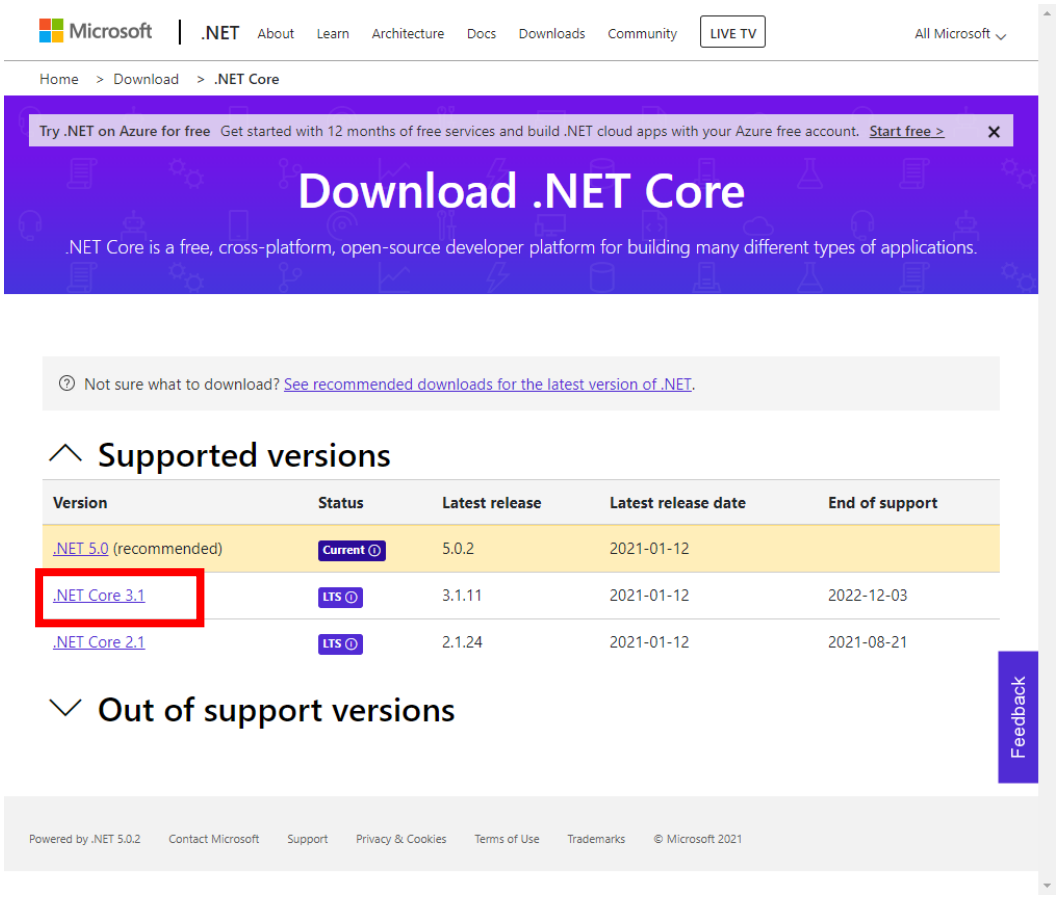
３－１．事前準備

Tester24 を実行するには JDK および.NET Core3.1 ランタイムのインストールが必要になります。お使いのパソコンにインストールされていない場合は、以下の URL から事前にインストールを行ってください。

３－１－１．.Net Core3.1 のインストール

お使いのパソコンに.Net Core3.1 がインストールされていない場合は、以下の URL からダウンロードおよびインストールを行ってください。

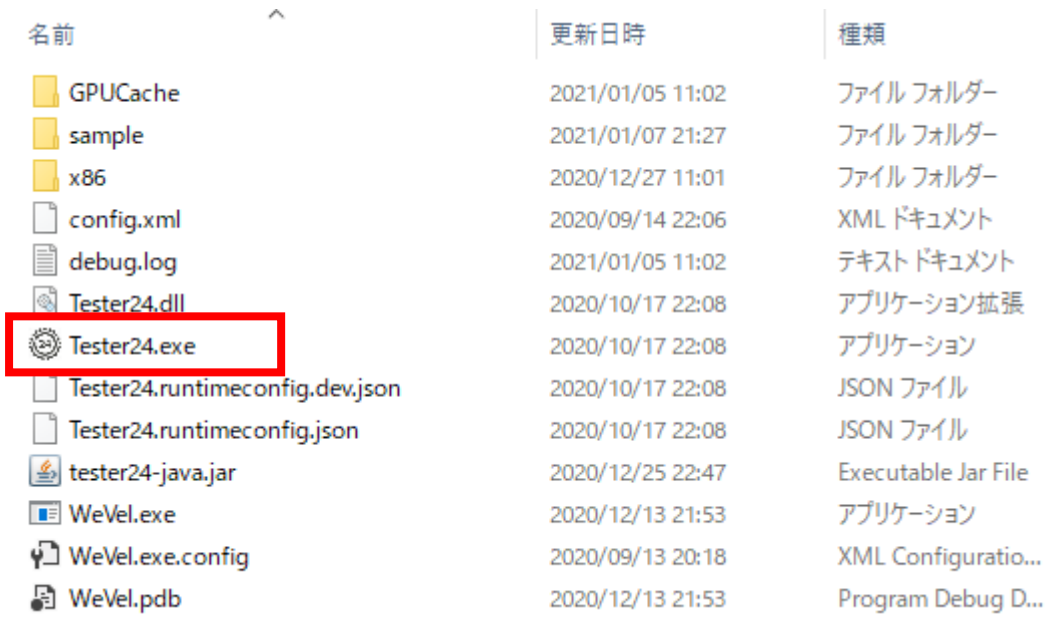
https://dotnet.microsoft.com/download/dotnet-core



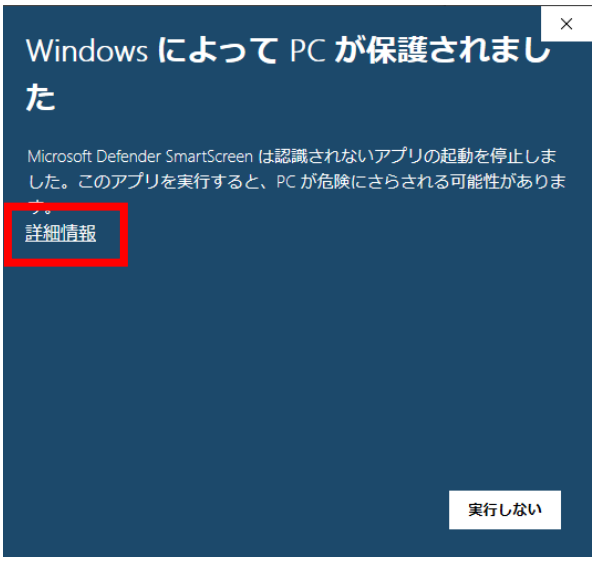
３－２．サンプルを実行してみる

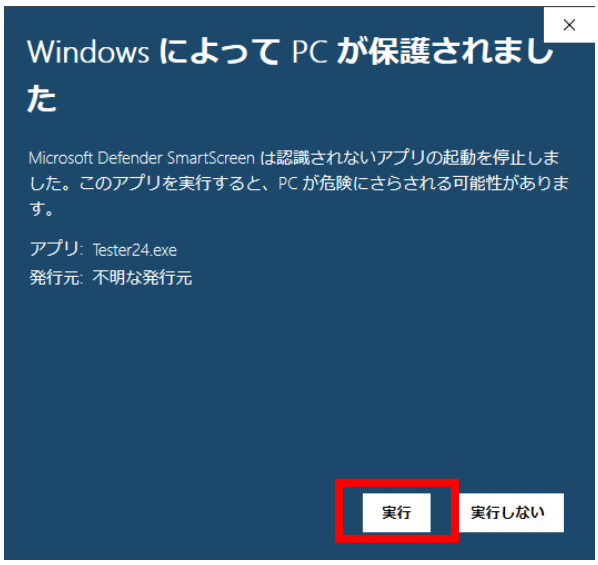
３－２－１．テスター24 を起動する

インストールしたフォルダにある Tester24.exe を実行してください。

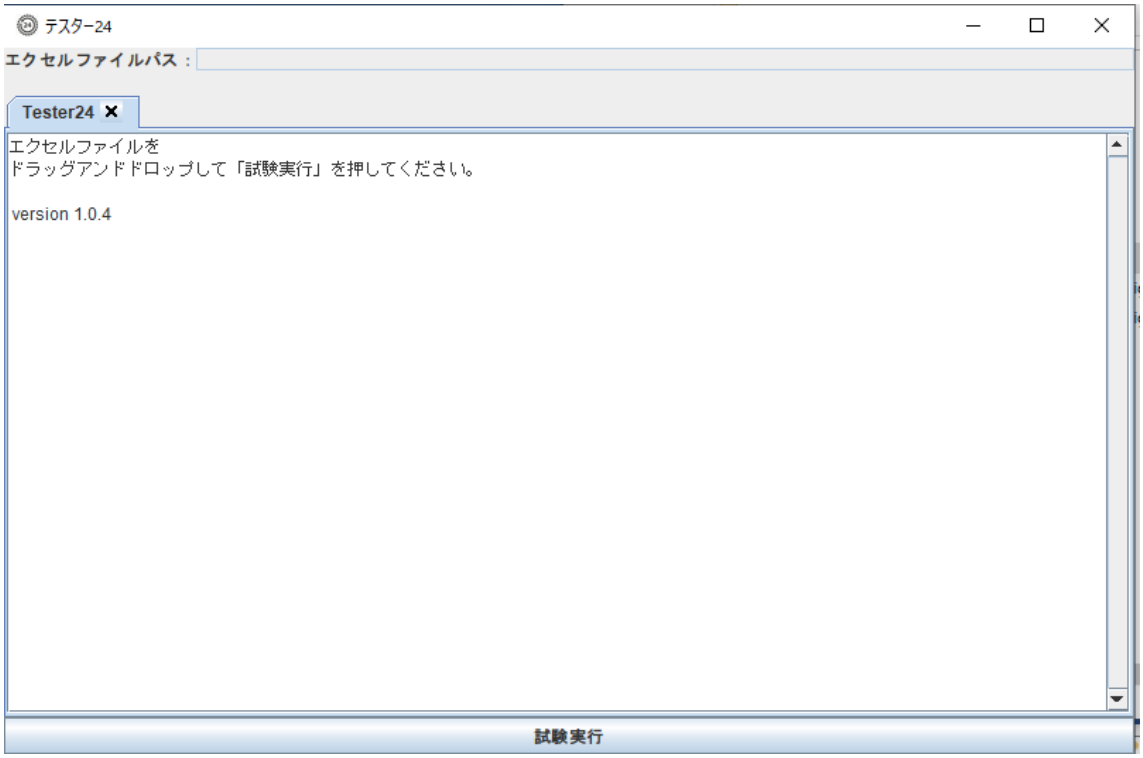


初めて実行する場合には、PC 保護のダイアログが表示されることがあります。
詳細情報をクリックし、実行ボタンをクリックしてください。



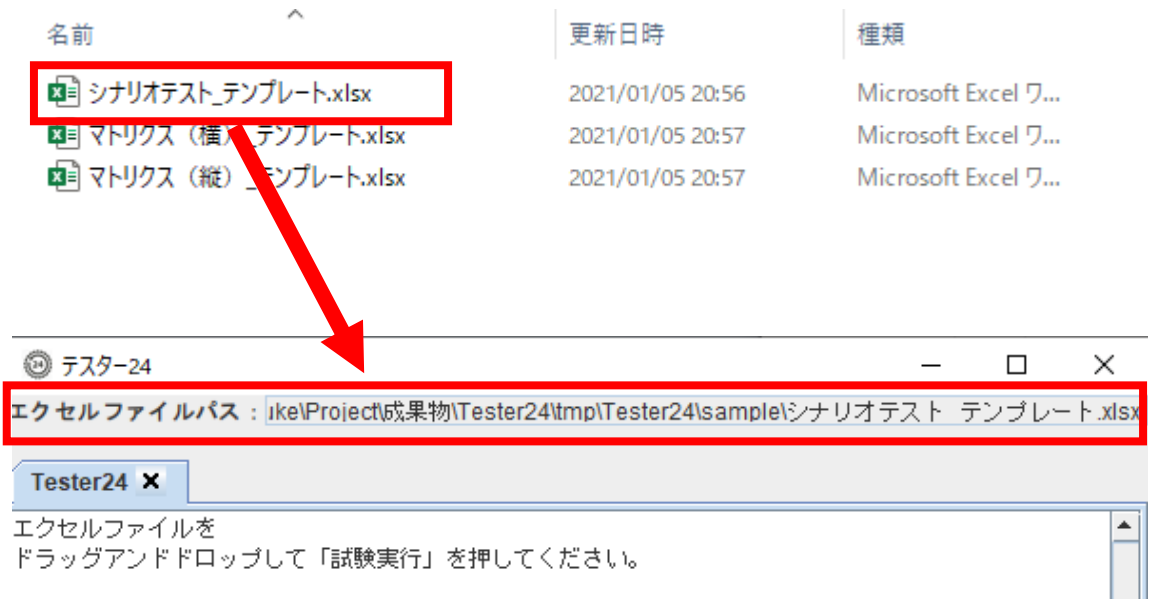


テスター24 のアプリケーションが起動されます。



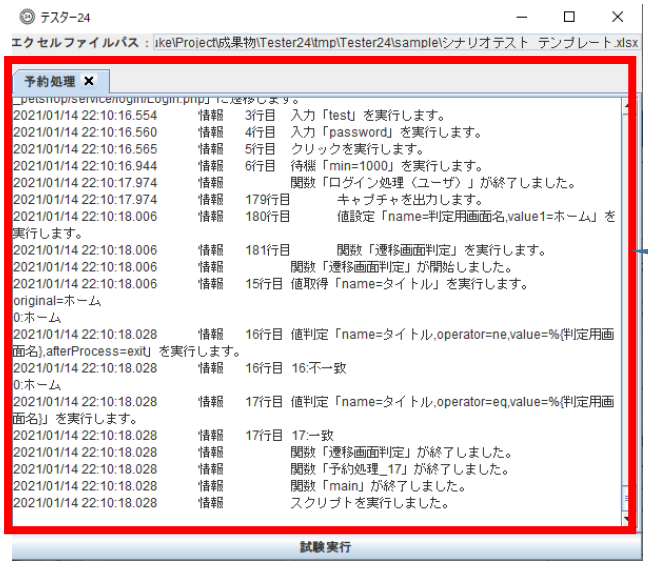
３－２－２．サンプルのテストケースをドラッグ＆ドロップする

sample フォルダに格納されている任意のテストケースを、テスター24 のアプリケーションにドラッグ＆ドロップしてください。Excel で作成したテストケースをツールに読み込ませることでテストの実施、画面イメージの取得、結果判定を自動で行います。



３－２－３．サンプルのファイルを実行する

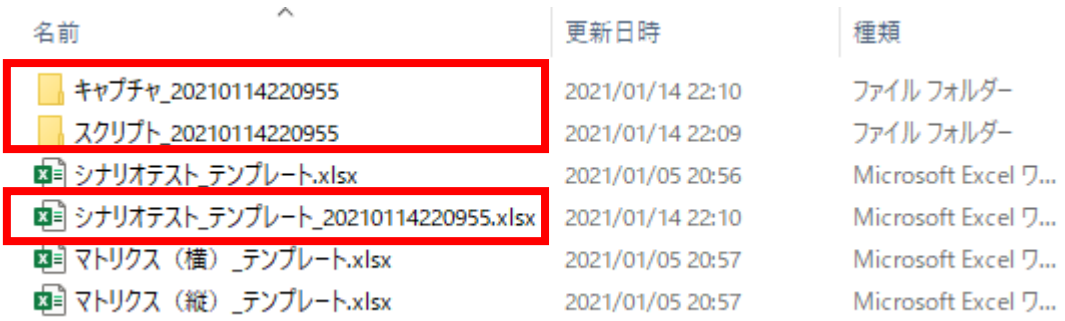
「試験実行」ボタンをクリックすると、画面にログが出力されるので完了するまで待ちます。



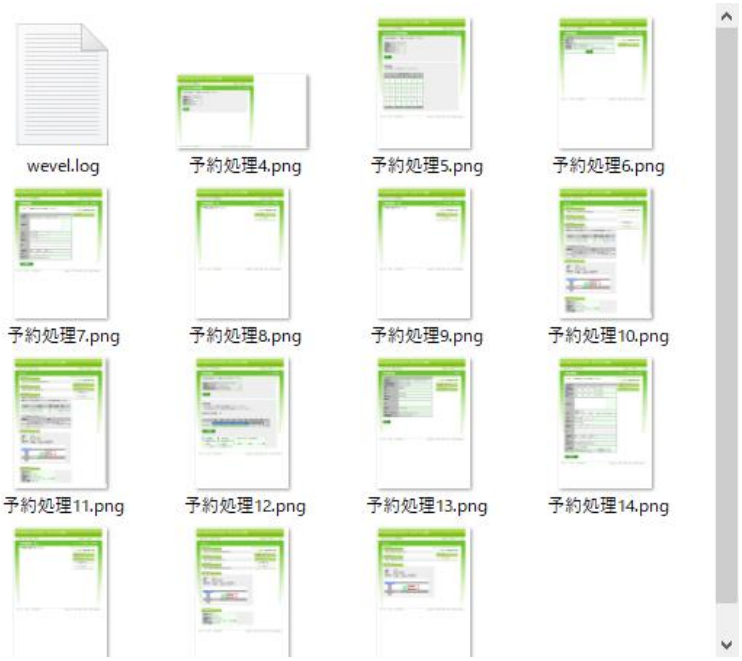
実行時のログが出力されます。

３－２－４．実行結果を確認する

テストケースの実行結果は読み込んだファイルと同じフォルダに出力されます。



キャプチャのフォルダには、テストケースを実行した際に取得した画面イメージがPNG ファイルで出力されます。今回使用したサンプルは、ある Web システムにてログインから登録処理を行うシナリオのテストケースとなっているため、その処理過程の画面イメージを連続して取得しています。



スクリプトファルダには、Excel のテストケースに記載した内容から内部ブラウザにて実行させるために必要となった中間ファイルを出力しています。通常は破棄して構いませんが、意図した動作をしない場合の解析などにご利用ください。

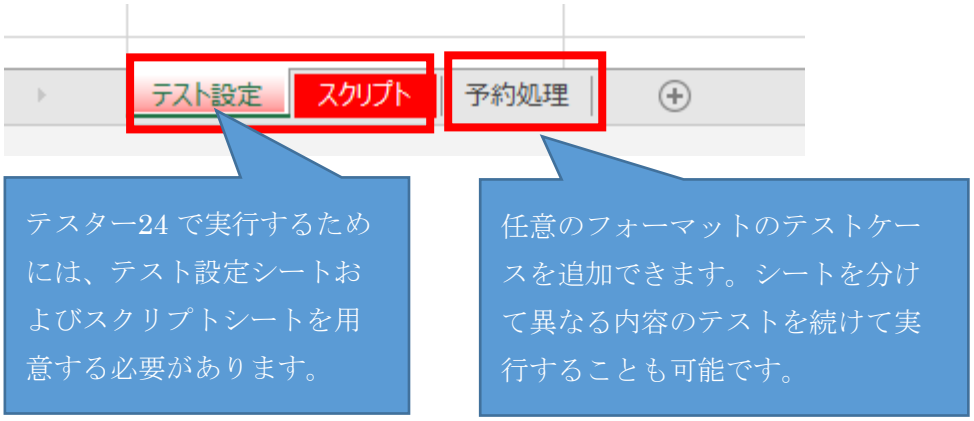
実行したテストケースと同名のタイムスタンプ付きのファイルは、実施結果を埋め込んだファイルになります。試験結果としてご利用ください。

４．テストケースを作成する

テスター24 では、Web システムの結合テストやリグレッションテストに適したシナリオベースのテストと、入力値のエラー検証に適したバリエーションのテストという 2 種類のテスト自動化をサポートします。

４－１．テストケースの共通書式について

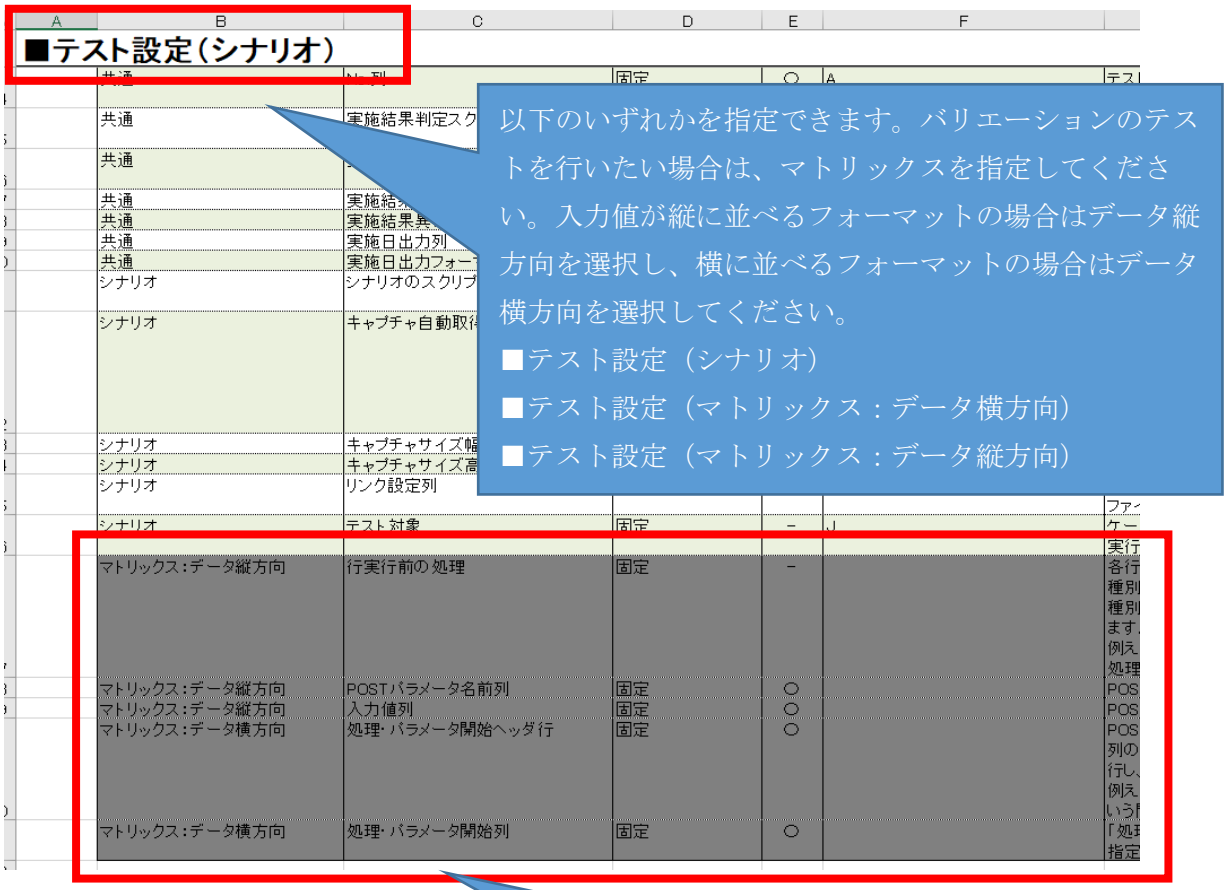
テストケースは Excel ファイルにて作成します。Excel にテスター24 で動作させるための設定用のシート（テスト設定、スクリプト）を用意することで、任意のフォーマットで作成したテストケースのシートを実行させることができます。



４－１－１．テスト設定シートについて

テスター24 にテストケースを読み込ませるための各種設定は、テスト設定シートにて行います。設定できる項目は、シナリオベースのテストとバリエーションのテストとで異なるため、まずは１行目のプルダウンにて選択を行います。

各パターンで設定できない項目はグレーアウトされますので、必要な項目について値を設定してください。各項目の内容については、備考に記載された内容を参照ください。



テスター24 では、内部ブラウザにより Web システムへアクセスします。そのため、Web システムによってはユーザエージェントのチェックにてアクセスが拒否されることが考えられます。

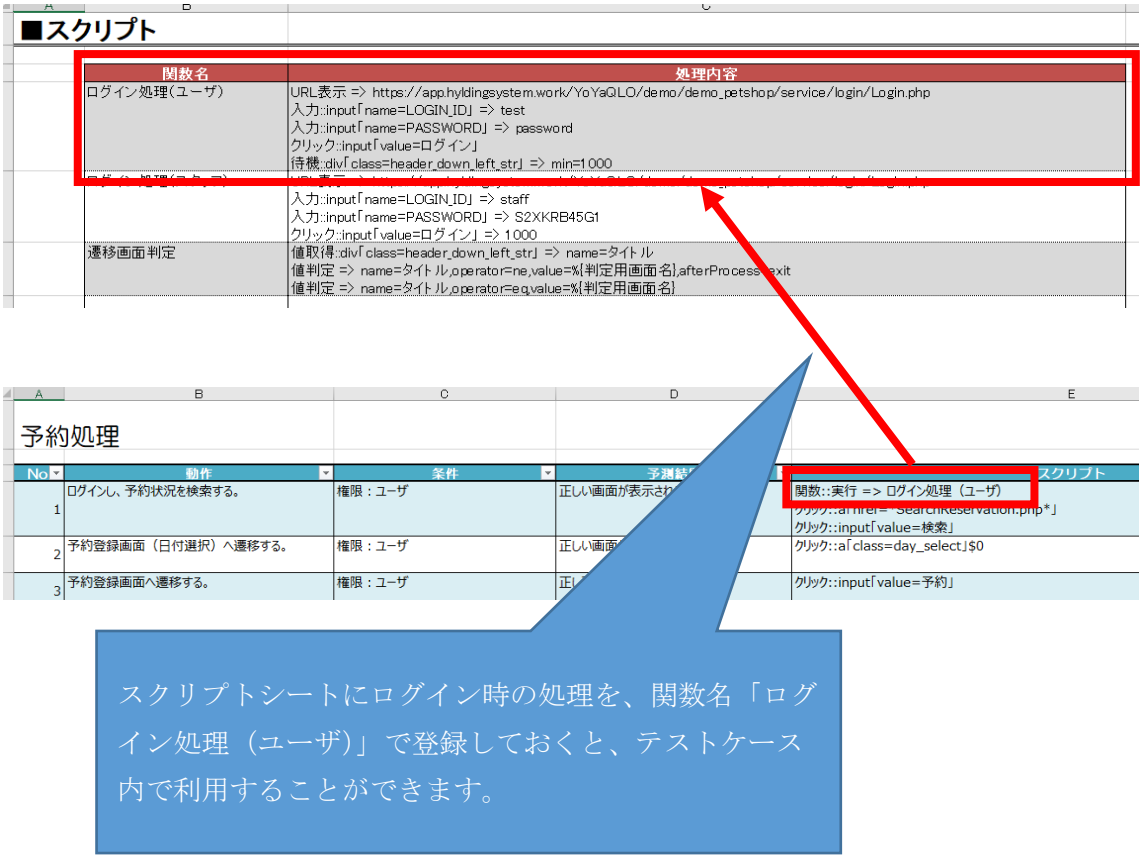
テスト対象の Web システムがユーザエージェントのチェックを行っている場合は、そのシステムが許容するユーザエージェントの文字列をテスト設定シートの「■変数についての情報」の予約語である「ユーザエージェント名」の値に設定してください。

■変数についての情報　※変数は必要に応じて追加ください

カテゴリ	変数名	種別	列	値	備考
予約語	ユーザエージェント名	固定	－	Orignal Browser	ユーザエージェントの文字列を指定してください。
	変数名1	参照(列指定)	－		
	変数名2	参照(行列指定)	－	B3	
	変数名3	固定	－	変数値	

４－１－２．スクリプトシートについて

Web システムのテストを行う際には、ログイン処理を行ったうえで所定の画面まで遷移する必要があります。テスター24 では、テストケースを実施する際に共通的に利用する処理をスクリプトシートに記載して、関数として各テストケースにて利用できます。



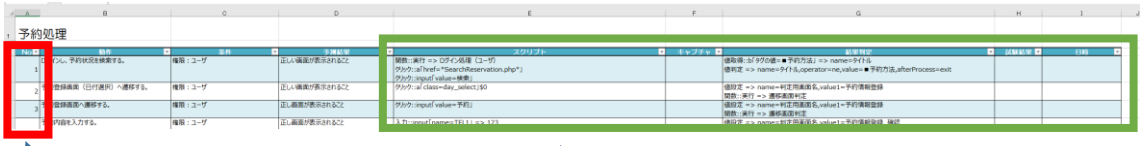
４－２．テストケース（シナリオ）を作成する

同梱されている「シナリオテスト_テンプレート.xlsx」を例に、シナリオベースのテスト自動化の方法について説明します。

このサンプルでは、ペットショップの予約システムに対して利用者が予約し、スタッフが予約を取り消しするまでの一連の処理をテストする内容になっています。利用シーンとしては、仕様変更などでシステムの改修を行った後に、予約システムの主要な機能である予約から取り消しまでの挙動にデグレートが発生していないかの回帰テストを想定します。

４－２－１．テストシートについて

テストを記載するシートには、必須項目と任意項目の列が存在します。列の位置はテスト設定シートにて指定できるため、任意のフォーマットを使用することができます。



No.は必須項目になります。No.に空白が発生するまで、下方向に向かってテストの実施を行います。

スクリプト、キャプチャ、結果判定、試験結果、日時は任意項目になります。ですが、実際の画面操作についてはスクリプトに記載するため、通常は設定が必要になります。

４－２－２．テストシナリオについて

サンプルのペットショップ予約システムでは、以下の手順で予約から取消が行われます。

これらの一連の流れをテストシートのスクリプト列に記載します。また、動作した結果が意図するものかを結果判定列に記載することで、妥当性のチェックも行います。

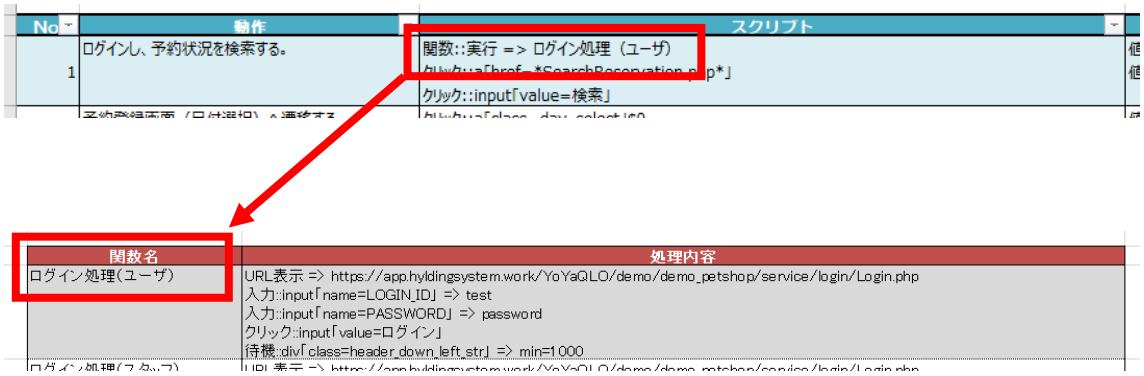
① 利用者がログインし、予約可能日を検索する。
② 利用者が予約したい日を選択する。
③ 利用者が予約するための各種情報を入力する。
④ 利用者が予約を行う。
⑤ スタッフがログインする。
⑥ スタッフが予約内容を確認し、依頼された予約を取り消す。
⑦ スタッフがホーム画面にて予約が削除されたことを確認する。
⑧ 利用者がログインし、ホーム画面で予約が削除されたことを確認する。

４－２－３．テストシナリオからスクリプトを作成する

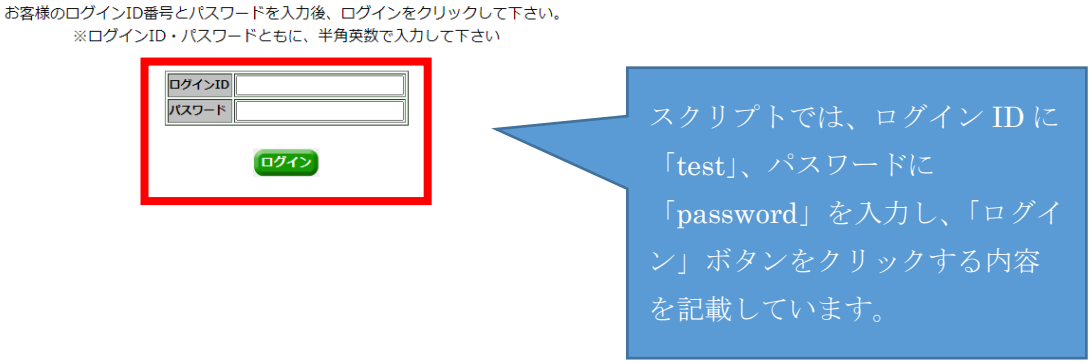
テストシナリオをテスター24 で自動実行させるためには、テスター24 が解析できるスクリプトに変換する必要があります。スクリプトの記載方法については、「５．スクリプトを作成する」を参照ください。

① 利用者がログインし、予約可能日を検索する。

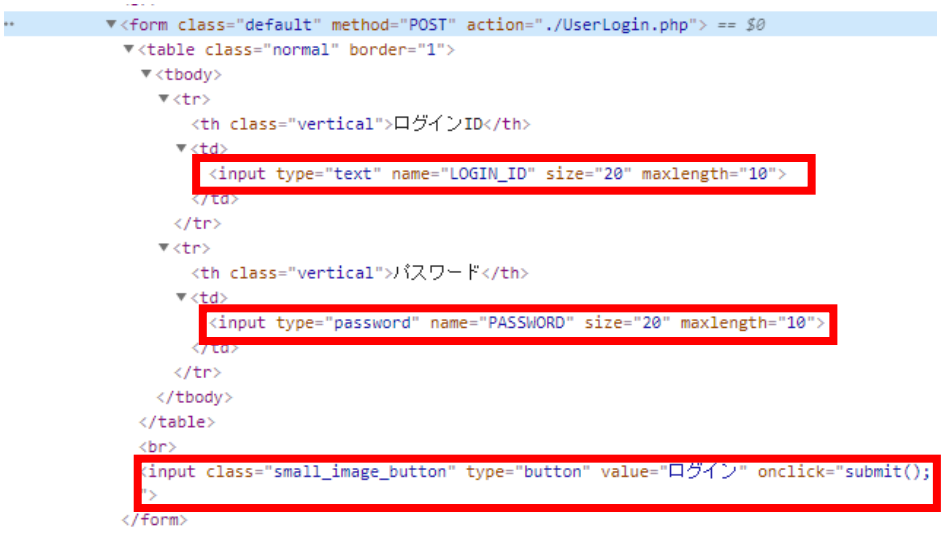
サンプルの予約システムに利用者がログインする必要があるため、スクリプト列に「関数::実行 => ログイン処理（ユーザ）」を記載しています。これにより、スクリプトシートに記載した「ログイン処理（ユーザ）」の共通スクリプトの呼び出しを行っています。



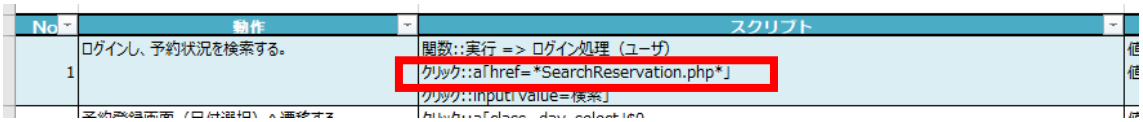
「ログイン処理（ユーザ）」では、予約システムの URL に遷移して、ログイン ID、パスワードを入力したのちに「ログイン」ボタンをクリックし、1 秒の待機を行っています。



値の入力やボタンをクリックする際の指定方法ですが、タグと属性を指定することで操作する箇所を特定します。今回のログイン ID であれば、input タグの name が LOGIN_ID を指定することで特定できます。ログインボタンについては name がないため、input タグの value がログインを指定することで特定できます。

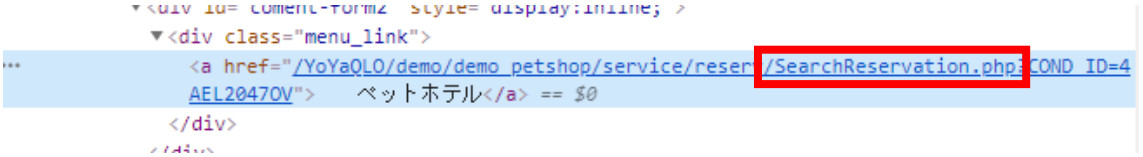


次にログイン後に予約可能日を検索する画面へ遷移する処理を行っています。

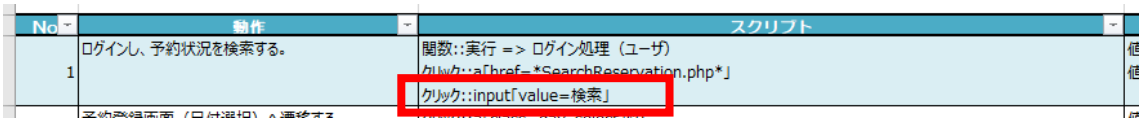


サンプルの予約システムでは、ペットホテルのリンクから予約可能日の検索画面へと遷移できます。

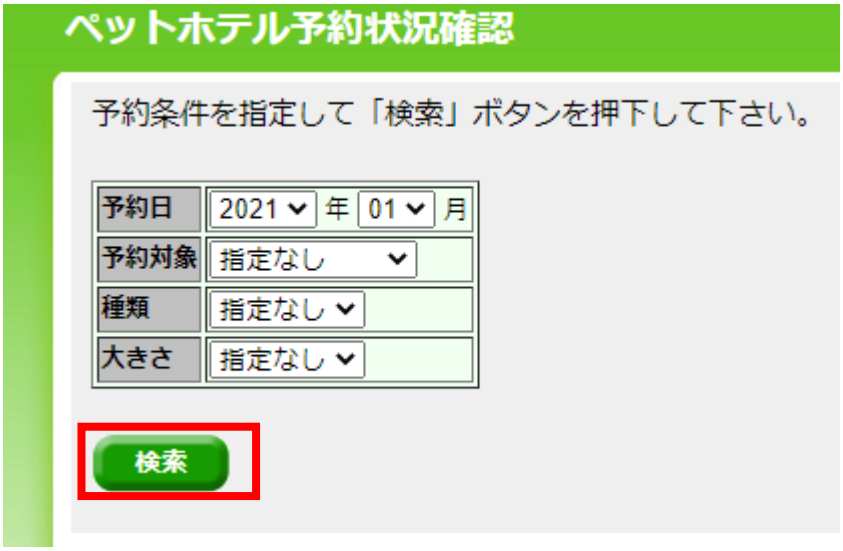
検索画面へのリンクは、a タグの href で特定できます。*を指定することで値の部分一致による検索を行うことができます。



次に予約画面の検索ボタンをクリックして、予約可能日の検索を行っています。



サンプルの予約システムでは、予約条件を満たす部屋と日付を表示してから予約が行えるようになります。



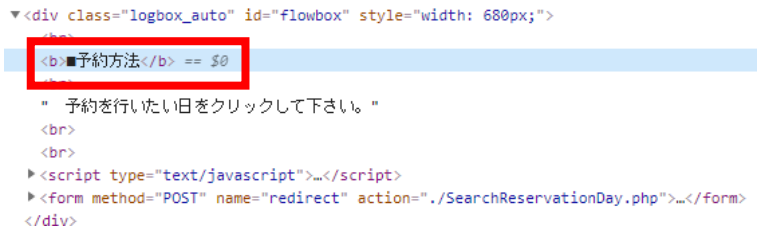
ログインボタンと同様に name がないため、input タグの value が検索を指定することで特定できます。

```
▼<form method="POST" action="./SearchReservation.php">
    <input type="HIDDEN" name="COND_ID" value="4AEL2047OV">
  ▶<table class="normal" border="1">…</table>
    <br>
    <input class="small_image_button" type="button" value="検索" onclick="submit();"> ==
  </form>
```

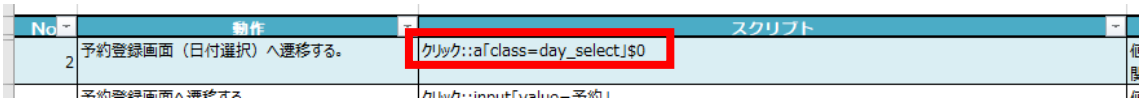
結果判定では、予約状況確認画面にて検索結果が表示されたかを確認し、意図しない画面の場合はテストを中止しています。

No	動作	結果判定
1	ログインし、予約状況を検索する。	値取得::b「タグの値=■予約方法」=> name=タイトル 値判定 => name=タイトル,operator=ne,value=■予約方法,afterProcess=exit

検索に成功すると、「■予約方法」というエリアが表示されます。そのため、スクリプトが完了した後に b タグから「■予約方法」という値を探して、変数にタグの値である「■予約方法」を設定しています。次に値判定にて変数の値が「■予約方法」ではない場合は、正しく処理が行えていないためテストの中止（exit）を行っています。

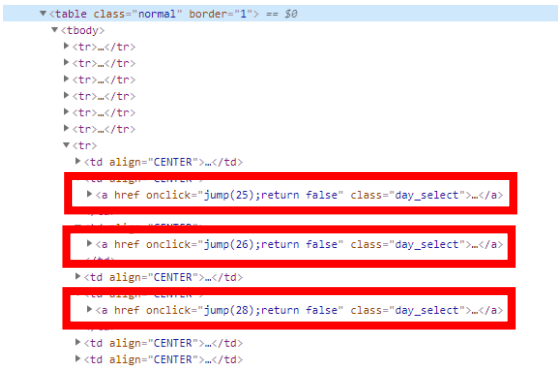


② 利用者が予約したい日を選択する。
カレンダーから予約可能な日付を選択します。

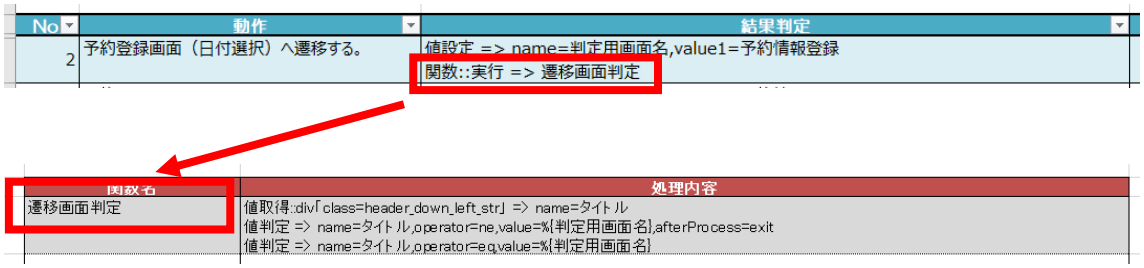


サンプルの予約システムでは、予約可能な日が〇のリンクで表示されます。

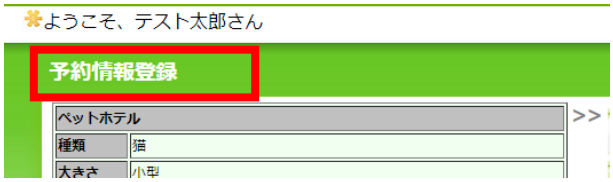
今回は a タグで class の値が同じものが複数存在しています。回帰テストなので、どれか 1 つをクリックできれば良い場合は、$にて何個目に対して操作するかを指定できます。$0 を指定すると最初の項目に対して操作を行います。

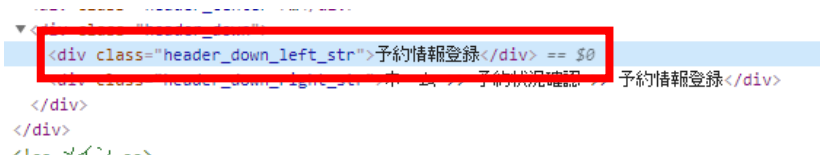


結果判定では、予約情報画面が表示されたかを確認し、意図しない画面の場合はテストを中止しています。スクリプト列に「関数::実行 => 遷移画面判定」を記載しています。これにより、スクリプトシートに記載した「遷移画面判定」の共通スクリプトの呼び出しを行っています。



スクリプトの処理に成功すると、画面遷移が発生します。意図した画面に遷移したことで正常とみなす場合は、上記のように画面を識別する文言を取得して、呼び出し元で設定した文言（上記では「予約情報登録」）と比較します。次に値判定にて変数の値が意図した文言ではない場合は、正しく処理が行えてないためテストの中止（exit）を行っています。No.3 以降も同様の結果判定を行っています。

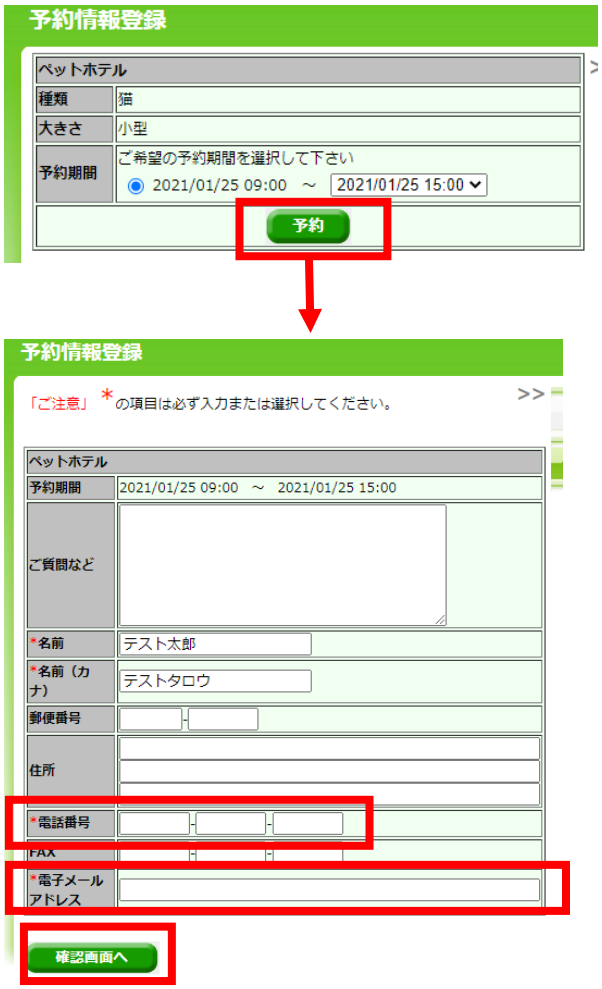




③ 利用者が予約するための各種情報を入力する。
予約する日を選択した後、予約内容を入力します。

No	動作	スクリプト
3	予約登録画面へ遷移する。	クリック::input「value=予約」
4	予約内容を入力する。	入力::input「name=TEL1」=> 123 入力::input「name=TEL2」=> 456 入力::input「name=TEL3」=> 789 入力::input「name=MAIL」=> testxxx@hyldingsystem.co.jp クリック::input「value=確認画面へ」

サンプルの予約システムでは、予約期間の終了を指定した後に連絡先を入力して確認画面へと遷移します。スクリプトでは必須項目のみを入力して確認画面へボタンをクリックしています。



④ 利用者が予約を行う。

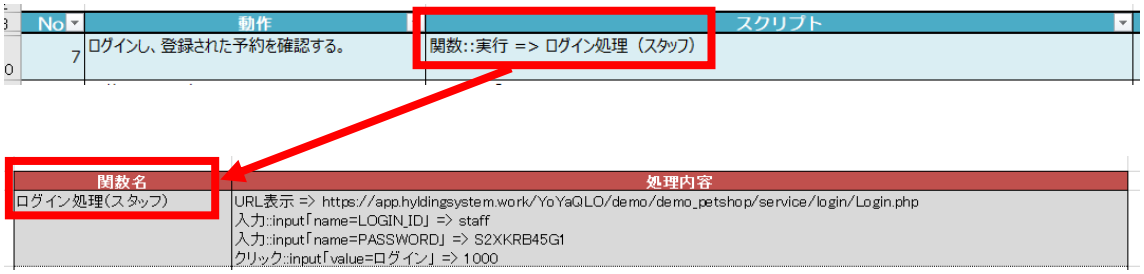
予約を行った後にホーム画面にて予約状況を確認します。

No	動作	スクリプト
5	予約内容を確認する。	クリック::input「value=予約する」
6	予約登録を完了する。	クリック::a「href=*index.php」

サンプルの予約システムでは、利用者が予約処理を行うと仮予約済となります。予約状況はホーム画面にて確認できます。

⑤ スタッフがログインする。

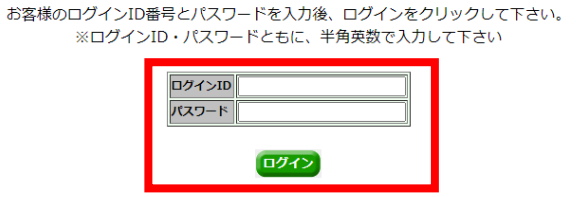
サンプルの予約システムにスタッフがログインする必要があるため、スクリプト列に「関数::実行 => ログイン処理（スタッフ）」を記載しています。これにより、スクリプトシートに記載した「ログイン処理（スタッフ）」の共通スクリプトの呼び出しを行っています。



No	動作	スクリプト
7	ログインし、登録された予約を確認する。	関数::実行 => ログイン処理（スタッフ）

関数名	処理内容
ログイン処理(スタッフ)	URL表示 => https://app.hyldingsystem.work/YoYaQLO/demo/demo_petshop/service/login/Login.php 入力::input「name=LOGIN_ID」 => staff 入力::input「name=PASSWORD」 => S2XKRB45G1 クリック::input「value=ログイン」 => 1.000

「ログイン処理（スタッフ）」では、予約システムの URL に遷移して、ログイン ID、パスワードを入力した後に「ログイン」ボタンをクリックし、1 秒の待機を行っています。ユーザでログインした際と、ログイン ID とパスワードの値が異なるだけで同様の処理になります。

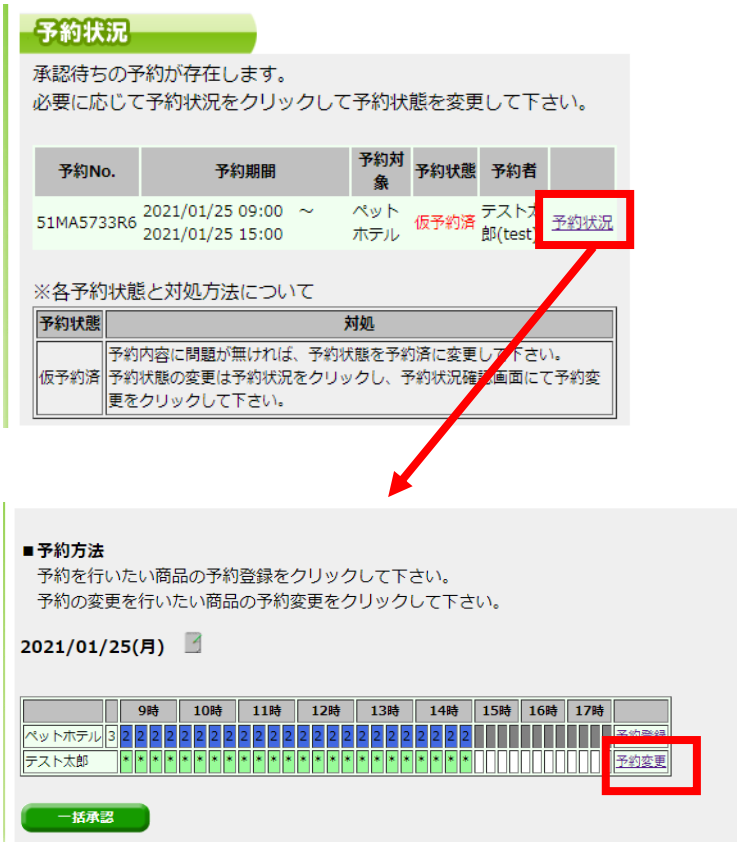


⑥ スタッフが予約内容を確認し、依頼された予約を取り消す。

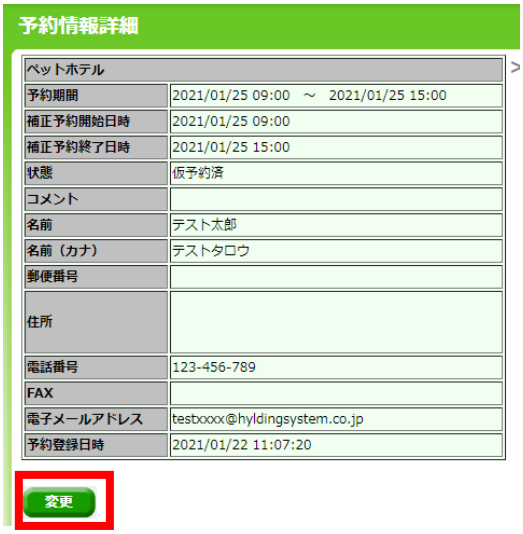
利用者が登録した予約を確認して、予約の状態を取消済に更新します。

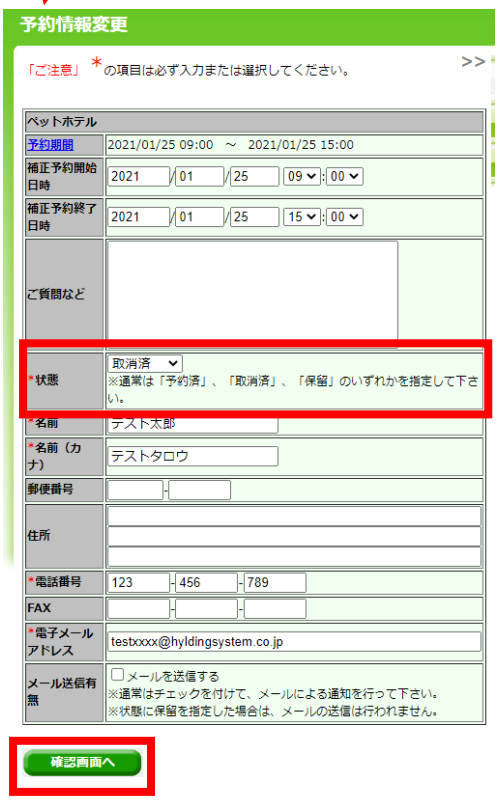
No	動作	スクリプト
8	予約状況を検索する。	クリック::a「href=*SearchReservation.php*」$0
9	予約内容を確認する。	クリック::a「onclick=jumpChangeReserv*」$0
10	予約内容を編集する。	クリック::input「value=変更」
11	ステータスを取消済みに変更し、予約内容を確認する。	入力::select「name=STATUS」=> 4 クリック::input「value=確認画面へ」
12	予約内容を変更する。	クリック::input「value=予約する」

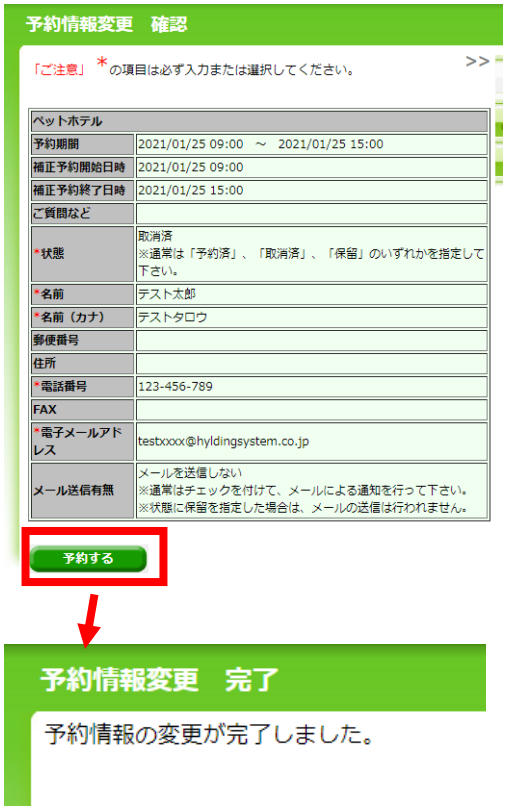
サンプルの予約システムでは、利用者からの予約はホーム画面の予約状況に表示され、予約状況のリンクをクリックすると編集画面へと遷移できます。



次に予約情報の状態を取消済に更新します。






予約情報変更　確認
「ご注意」 * の項目は必ず入力または選択してください。
>>
ペットホテル
予約期間
2021/01/25 09:00 ～ 2021/01/25 15:00
補正予約開始日時
2021/01/25 09:00
補正予約終了日時
2021/01/25 15:00
ご質問など
*状態
取消済
※通常は「予約済」、「取消済」、「保留」のいずれかを指定して下さい。
*名前
テスト太郎
*名前（カナ）
テストタロウ
郵便番号
住所
*電話番号
123-456-789
FAX
*電子メールアドレス
testxxxx@hyldingsystem.co.jp
メール送信有無
メールを送信しない
※通常はチェックを付けて、メールによる通知を行って下さい。
※状態に保留を指定した場合は、メールの送信は行われません。
予約する
予約情報変更　完了
予約情報の変更が完了しました。

⑦　スタッフがホーム画面にて予約が削除されたことを確認する。

　ホーム画面にて利用者が登録した予約が削除されたことを確認します。

No	動作	スクリプト
13	ホーム画面を表示し、予約が取り消されたことを確認する。	クリック::a「href=*index.php」

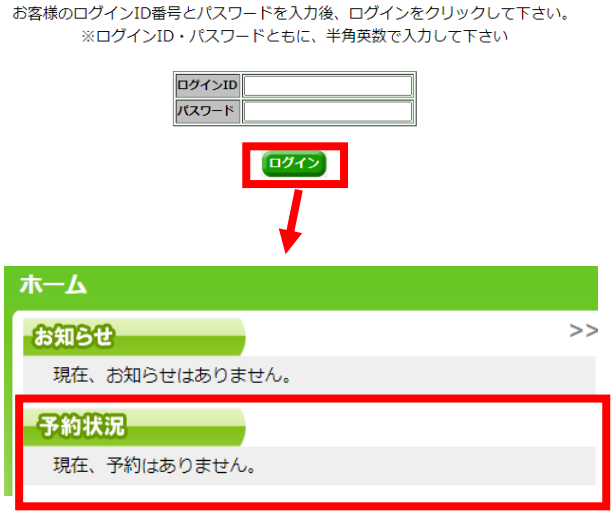
サンプルの予約システムでは、ホーム画面にて予約状況の確認が行えます。

⑧ 利用者がログインし、ホーム画面で予約が削除されたことを確認する。

利用者でログインし直し、ホーム画面にて予約状況を確認します。

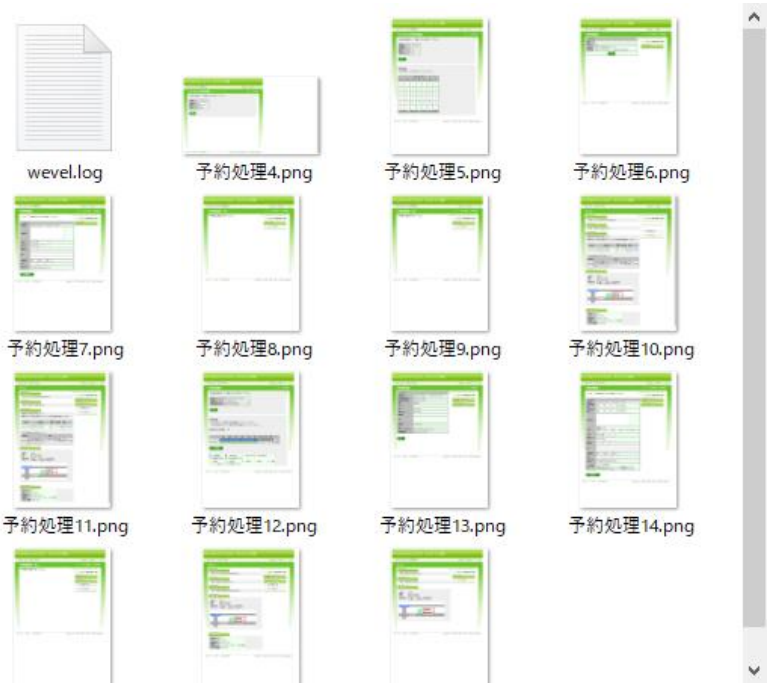
No	動作	スクリプト
14	ログインし、ホーム画面を表示し、予約が取り消されたことを確認する。	関数::実行 => ログイン処理（ユーザ）

サンプルの予約システムでは、ログイン後の初期画面がホーム画面となっています。



４－２－４．実行結果について

シナリオベースのテストケースでは、各行のスクリプトを実行すると現在の画面のイメージを自動で出力します。もし、スクリプトの途中で入力値を入れた直後の画面イメージも取得したい場合は、スクリプト列に「キャプチャ」を記載することで任意のタイミングにて画面イメージの出力が行えます。



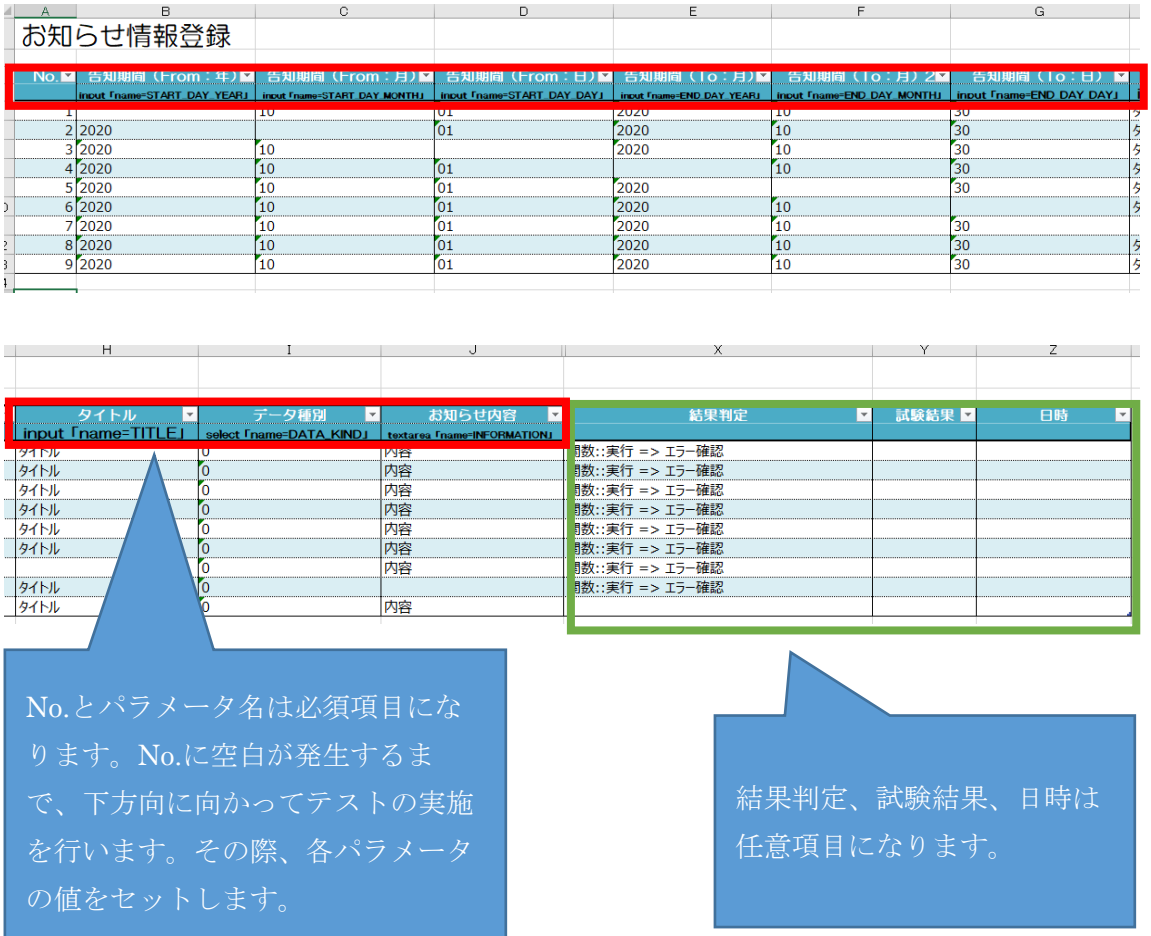
４－３．テストケース（マトリクス：横）を作成する

同梱されている「マトリクス（横）_テンプレート.xlsx」を例に、シナリオベースのテスト自動化の方法について説明します。

このサンプルでは、ペットショップの予約システムにて管理者がお知らせを登録する際の入力チェックをテストする内容になっています。利用シーンとしては、単体試験などでの入力値のバリエーションテストを想定します。

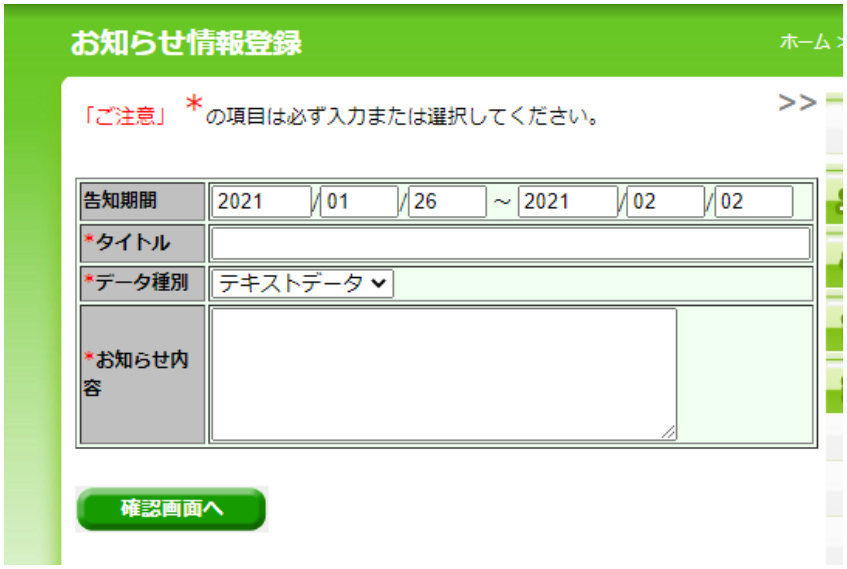
４－３－１．テストシートについて

テストを記載するシートには、必須項目と任意項目の列が存在します。列の位置はテスト設定シートにて指定できるため、任意のフォーマットを使用することができます。



４－３－２．バリエーションテストのシナリオについて

サンプルのペットショップ予約システムでは、お知らせ情報を登録する際に告知期間（From）、告知期間（To）、タイトル、データ種別、お知らせ内容を設定することができます。告知期間は一部未入力を許容せず、タイトル、お知らせ内容は必須項目であることから、それらの入力値のバリエーションテストを行います。



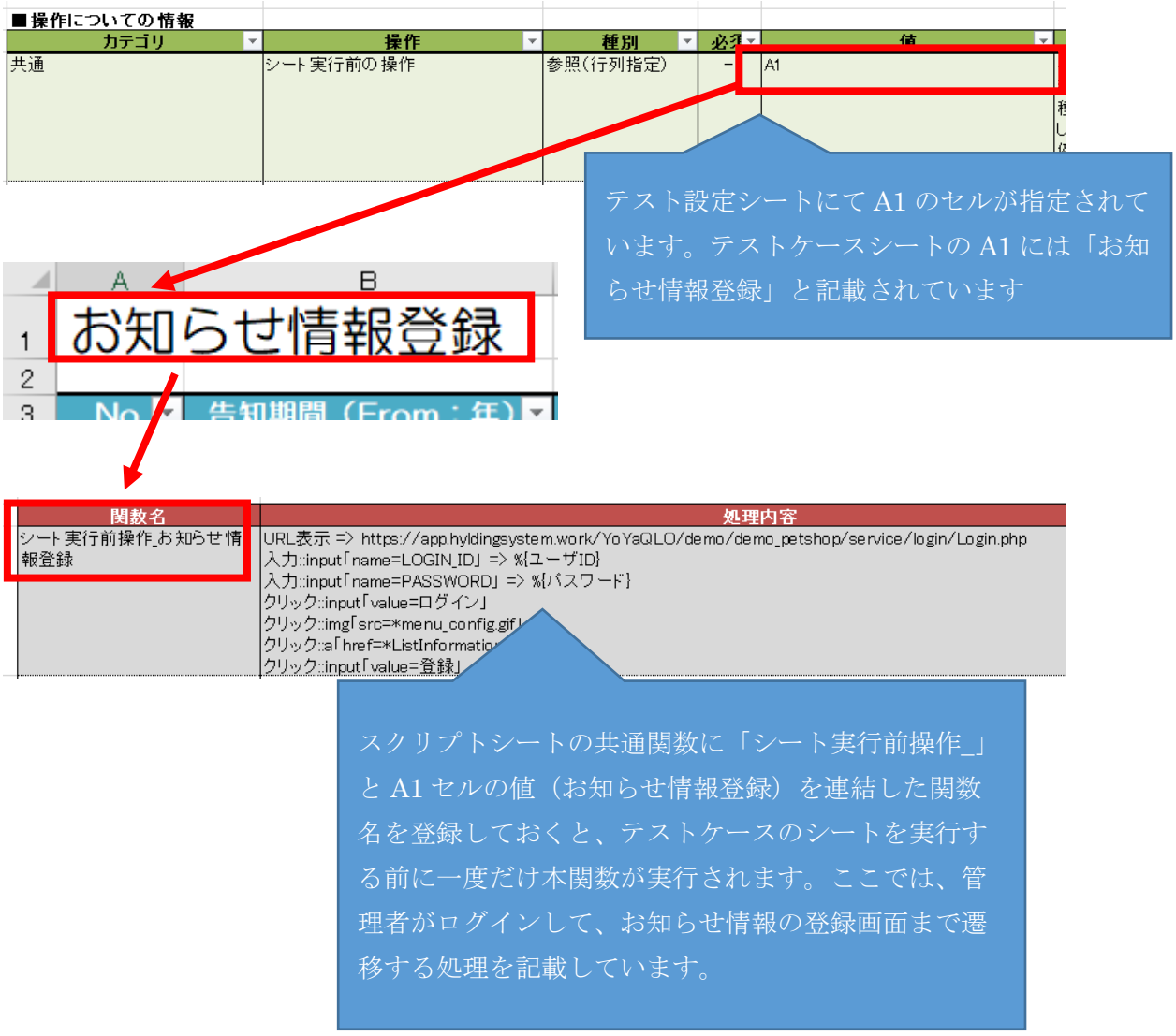
① 管理者がログインし、お知らせ情報登録画面を表示する。
② 以下の入力値でエラーが発生することを確認する。
・告知期間（From）が一部未入力
・告知期間（To）が一部未入力
・タイトルが未入力
・お知らせ内容が未入力
③ 以下の入力値でエラーが発生しないことを確認する。
・全てを入力

４－３－３．バリエーションテストのパターンからスクリプトを作成する

バリエーションテストをテスター24 で自動実行させるためには、テスター24 が解析できるスクリプトに変換する必要があります。スクリプトの記載方法については、「５．スクリプトを作成する」を参照ください。

① 管理者がログインし、お知らせ情報登録画面を表示する。

先ずは管理者でログインを行い、お知らせ情報の登録画面まで遷移する必要があります。テスト設定シートのシート実行前の操作に参照するセルを記載することで、各テストケースシートの指定セルの値と同名の関数をシートのテストを実行する前に一度だけ実行します。



② 入力値でエラーが発生することを確認する。

今回は、告知期間（From）が一部未入力、告知期間（To）が一部未入力、タイトルが未入力、データ種別が未入力、お知らせ内容が未入力のパターンについて確認を行います。

告知期間（From）が一部未入力のパターンは、以下のように年月日がケース毎に空白となる記載を行います。この 3 ケースで年が未入力、月が未入力、日が未入力の入力パターンのテストケースを確認できます。

No.	告知期間（From：年）	告知期間（From：月）	告知期間（From：日）
	input「name=START_DAY_YEAR」	input「name=START_DAY_MONTH」	input「name=START_DAY_DAY」
1		10	01
2	2020		01
3	2020	10	

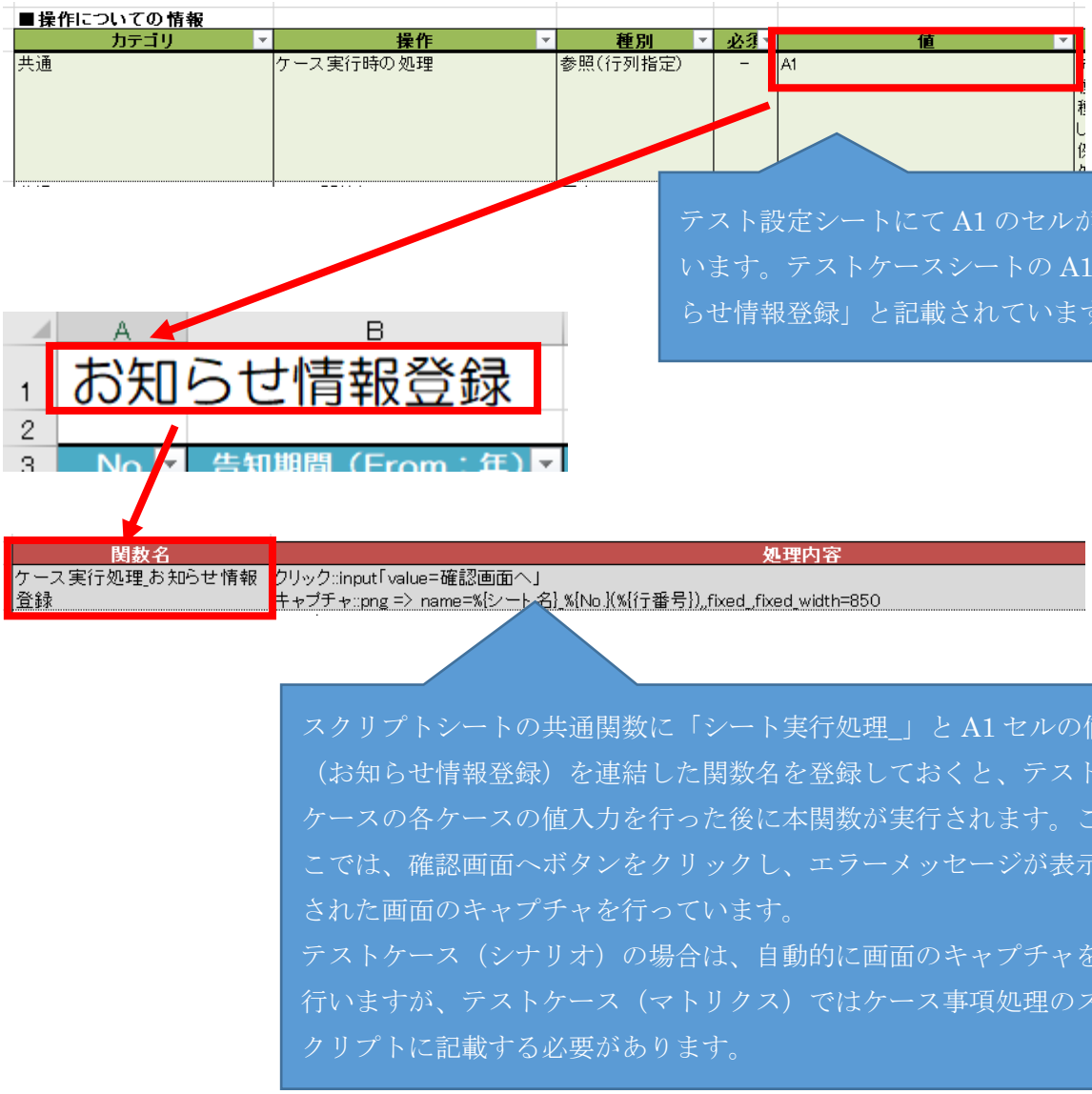
告知期間（To）が一部未入力のパターンは、以下のように年月日がケース毎に空白となる記載を行います。この 3 ケースで年が未入力、月が未入力、日が未入力の入力パターンのテストケースを確認できます。

No.	告知期間（To：月）	告知期間（To：月）2	告知期間（To：日）
	input「name=END_DAY_YEAR」	input「name=END_DAY_MONTH」	input「name=END_DAY_DAY」
4		10	30
5	2020		30
6	2020	10	

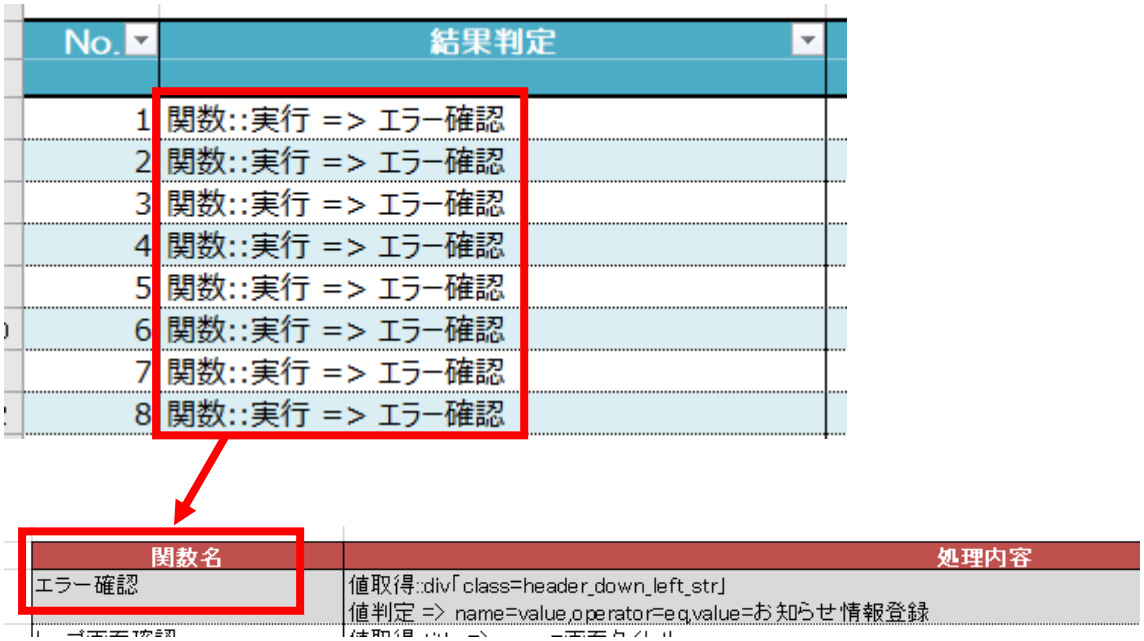
タイトル、お知らせ内容が未入力のパターンは、以下のようにケース毎に空白となる記載を行います。この 2 ケースでタイトルが未入力、データ種別が未入力、お知らせ内容が未入力の入力パターンのテストケースを確認できます。

No.	タイトル	データ種別	お知らせ内容
	input「name=TITLE」	select「name=DATA_KIND」	textarea「name=INFORMATION」
7		0	内容
8	タイトル	0	

上記の設定にて No.毎に各項目の入力設定までを実行しますが、入力チェックのテストとしては確認ボタンをクリックする必要があります。入力値を設定した後にどのような操作を行うかは、テスト設定シートとスクリプトシートに記載します。



結果判定ではエラー確認の関数を呼び出して、確認画面へ遷移していないことを確認しています。



No.	結果判定
1	関数::実行 => エラー確認
2	関数::実行 => エラー確認
3	関数::実行 => エラー確認
4	関数::実行 => エラー確認
5	関数::実行 => エラー確認
6	関数::実行 => エラー確認
7	関数::実行 => エラー確認
8	関数::実行 => エラー確認

関数名	処理内容
エラー確認	値取得::div「class=header_down_left_str」 値判定 => name=value,operator=eq,value=お知らせ情報登録

③ 全てを入力してエラーが発生しないことを確認する。

全ての項目に値を設定して確認画面へ遷移することを確認します。

No.	告知期間（From：年） input「name=START_DAY_YEAR」	告知期間（From：月） input「name=START_DAY_MONTH」	告知期間（From：日） input「name=START_DAY_DAY」	告知期間（To：月） input「name=END_DAY_YEAR」
9	2020	10	01	2020

告知期間（To：月）2 input「name=END_DAY_MONTH」	告知期間（To：日） input「name=END_DAY_DAY」	タイトル input「name=TITLE」	データ種別 select「name=DATA_KIND」	お知らせ内容 textarea「name=INFORMATION」
10	30	タイトル	0	内容

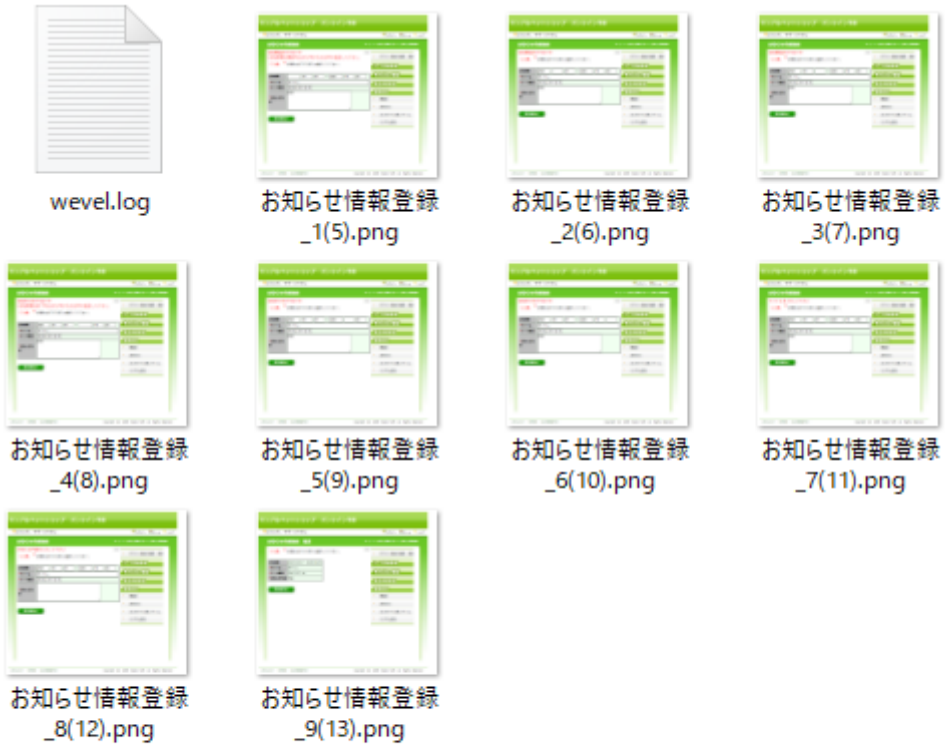
結果判定では正常確認の関数を呼び出して、確認画面へ遷移していることを確認しています。

No.	結果判定
9	関数::実行 => 正常確認

関数名	処理内容
正常確認	値取得::div「class=header_down_left_str」 値判定 => name=value,operator=eq,value=お知らせ情報登録　確認

４－３－４．実行結果について

マトリクスベースのテストケースでは、シート実行処理のスクリプトに「キャプチャ」を記載することで、各行のスクリプトを実行すると現在の画面のイメージを出力します。もし、スクリプトの途中で入力値を入れた直後の画面イメージも取得したい場合は、シート実行前操作のスクリプトに「キャプチャ」を記載することで任意のタイミングにて画面イメージの出力が行えます。



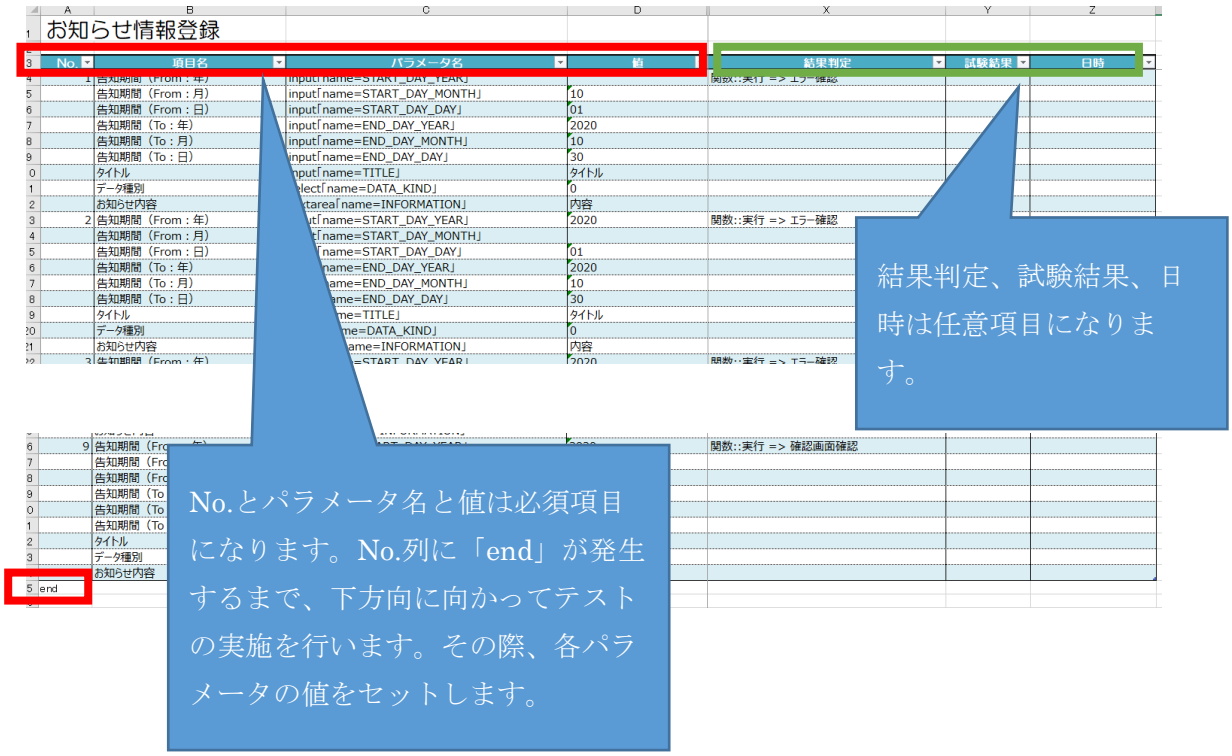
４－４．テストケース（マトリクス：縦）を作成する

同梱されている「マトリクス（縦）_テンプレート.xlsx」を例に、シナリオベースのテスト自動化の方法について説明します。

このサンプルでは、ペットショップの予約システムにて管理者がお知らせを登録する際の入力チェックをテストする内容になっています。利用シーンとしては、単体試験などでの入力値のバリエーションテストを想定します。内容はマトリクス（横）と同様ですが、入力値を縦に並べるフォーマットでのサンプルとなります。

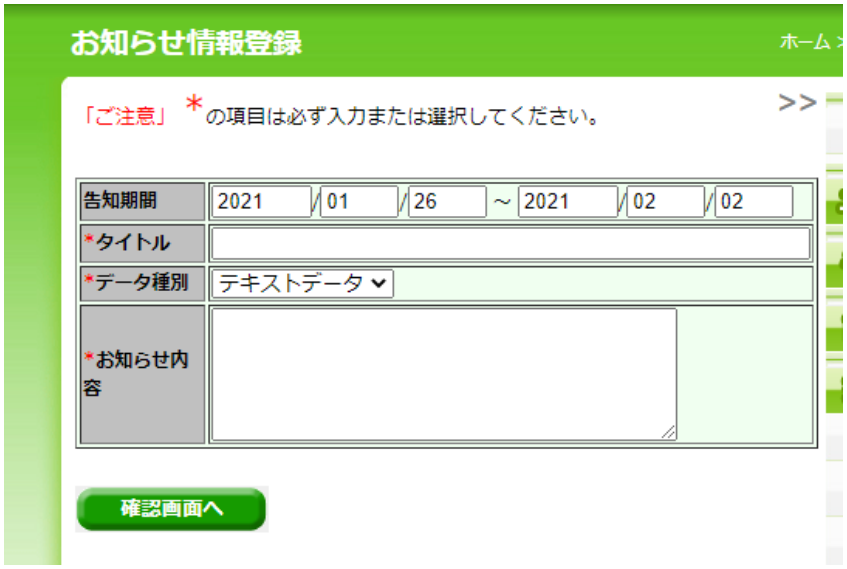
４－３－１．テストシートについて

テストを記載するシートには、必須項目と任意項目の列が存在します。列の位置はテスト設定シートにて指定できるため、任意のフォーマットを使用することができます。



４－４－２．バリエーションテストのシナリオについて

サンプルのペットショップ予約システムでは、お知らせ情報を登録する際に告知期間（From）、告知期間（To）、タイトル、データ種別、お知らせ内容を設定することができます。告知期間は一部未入力を許容せず、タイトル、お知らせ内容は必須項目であることから、それらの入力値のバリエーションテストを行います。



④ 管理者がログインし、お知らせ情報登録画面を表示する。

⑤ 以下の入力値でエラーが発生することを確認する。

・告知期間（From）が一部未入力

・告知期間（To）が一部未入力

・タイトルが未入力

・お知らせ内容が未入力

⑥ 以下の入力値でエラーが発生しないことを確認する。

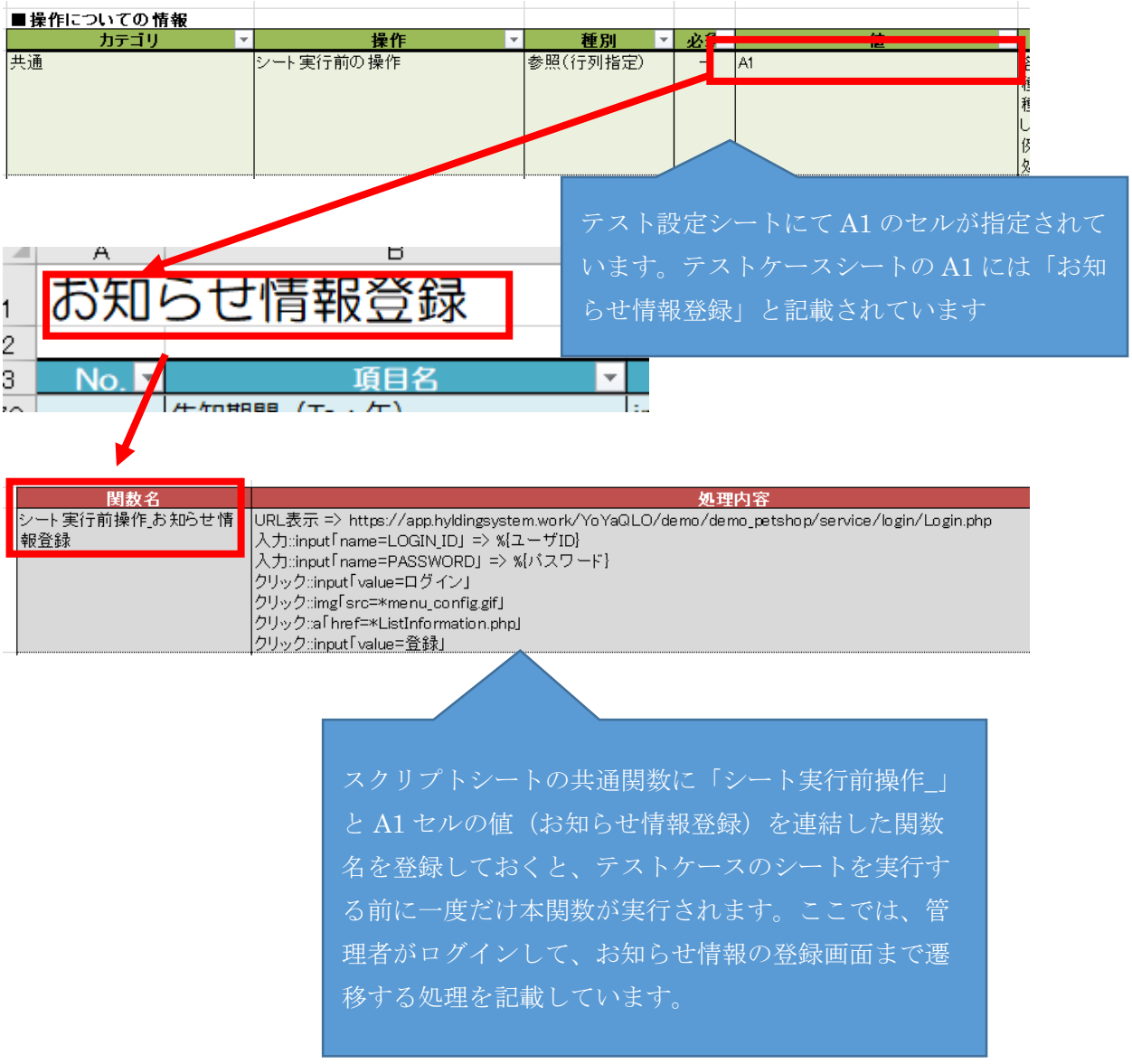
・全てを入力

４－４－３．バリエーションテストのパターンからスクリプトを作成する

バリエーションテストをテスター24 で自動実行させるためには、テスター24 が解析できるスクリプトに変換する必要があります。スクリプトの記載方法については、「５．スクリプトを作成する」を参照ください。

① 管理者がログインし、お知らせ情報登録画面を表示する。

先ずは管理者でログインを行い、お知らせ情報の登録画面まで遷移する必要があります。テスト設定シートのシート実行前の操作に参照するセルを記載することで、各テストケースシートの指定セルの値と同名の関数をシートのテストを実行する前に一度だけ実行します。



② 入力値でエラーが発生することを確認する。

今回は、告知期間（From）が一部未入力、告知期間（To）が一部未入力、タイトルが未入力、データ種別が未入力、お知らせ内容が未入力のパターンについて確認を行います。

告知期間（From）が一部未入力のパターンは、以下のように年月日がケース毎に空白となる記載を行います。この 3 ケースで年が未入力、月が未入力、日が未入力の入力パターンのテストケースを確認できます。

No.	項目名	パラメータ名	値
1	告知期間（From：年）	input「name=START_DAY_YEAR」	
	告知期間（From：月）	input「name=START_DAY_MONTH」	10
	告知期間（From：日）	input「name=START_DAY_DAY」	01
	告知期間（To：年）	input「name=END_DAY_YEAR」	2020
	告知期間（To：月）	input「name=END_DAY_MONTH」	10
	告知期間（To：日）	input「name=END_DAY_DAY」	30
	タイトル	input「name=TITLE」	タイトル
	データ種別	select「name=DATA_KIND」	0
	お知らせ内容	textarea「name=INFORMATION」	内容
2	告知期間（From：年）	input「name=START_DAY_YEAR」	2020
	告知期間（From：月）	input「name=START_DAY_MONTH」	
	告知期間（From：日）	input「name=START_DAY_DAY」	01
	告知期間（To：年）	input「name=END_DAY_YEAR」	2020
	告知期間（To：月）	input「name=END_DAY_MONTH」	10
	告知期間（To：日）	input「name=END_DAY_DAY」	30
	タイトル	input「name=TITLE」	タイトル
	データ種別	select「name=DATA_KIND」	0
	お知らせ内容	textarea「name=INFORMATION」	内容
3	告知期間（From：年）	input「name=START_DAY_YEAR」	2020
	告知期間（From：月）	input「name=START_DAY_MONTH」	10
	告知期間（From：日）	input「name=START_DAY_DAY」	
	告知期間（To：年）	input「name=END_DAY_YEAR」	2020
	告知期間（To：月）	input「name=END_DAY_MONTH」	10
	告知期間（To：日）	input「name=END_DAY_DAY」	30
	タイトル	input「name=TITLE」	タイトル
	データ種別	select「name=DATA_KIND」	0
	お知らせ内容	textarea「name=INFORMATION」	内容

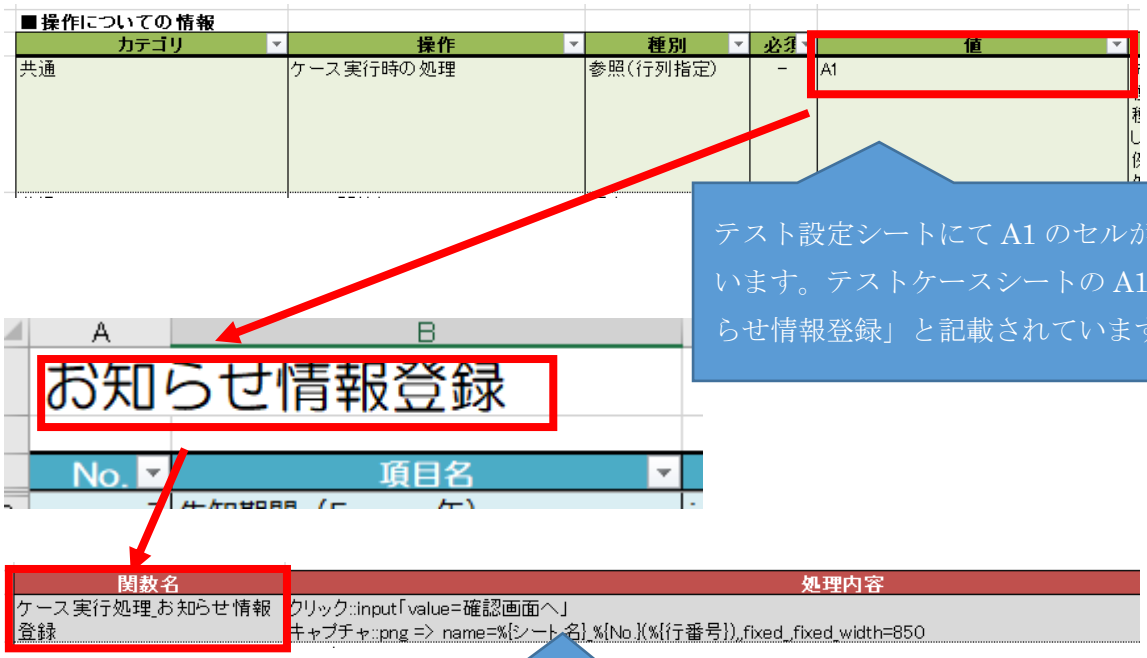
告知期間（To）が一部未入力のパターンは、以下のように年月日がケース毎に空白となる記載を行います。この 3 ケースで年が未入力、月が未入力、日が未入力の入力パターンのテストケースを確認できます。

No.	項目名	パラメータ名	値
4	告知期間（From：年）	input「name=START_DAY_YEAR」	2020
	告知期間（From：月）	input「name=START_DAY_MONTH」	10
	告知期間（From：日）	input「name=START_DAY_DAY」	01
	告知期間（To：年）	input「name=END_DAY_YEAR」	
	告知期間（To：月）	input「name=END_DAY_MONTH」	10
	告知期間（To：日）	input「name=END_DAY_DAY」	30
	タイトル	input「name=TITLE」	タイトル
	データ種別	select「name=DATA_KIND」	0
	お知らせ内容	textarea「name=INFORMATION」	内容
5	告知期間（From：年）	input「name=START_DAY_YEAR」	2020
	告知期間（From：月）	input「name=START_DAY_MONTH」	10
	告知期間（From：日）	input「name=START_DAY_DAY」	01
	告知期間（To：年）	input「name=END_DAY_YEAR」	2020
	告知期間（To：月）	input「name=END_DAY_MONTH」	
	告知期間（To：日）	input「name=END_DAY_DAY」	30
	タイトル	input「name=TITLE」	タイトル
	データ種別	select「name=DATA_KIND」	0
	お知らせ内容	textarea「name=INFORMATION」	内容
6	告知期間（From：年）	input「name=START_DAY_YEAR」	2020
	告知期間（From：月）	input「name=START_DAY_MONTH」	10
	告知期間（From：日）	input「name=START_DAY_DAY」	01
	告知期間（To：年）	input「name=END_DAY_YEAR」	2020
	告知期間（To：月）	input「name=END_DAY_MONTH」	10
	告知期間（To：日）	input「name=END_DAY_DAY」	
	タイトル	input「name=TITLE」	タイトル
	データ種別	select「name=DATA_KIND」	0
	お知らせ内容	textarea「name=INFORMATION」	内容

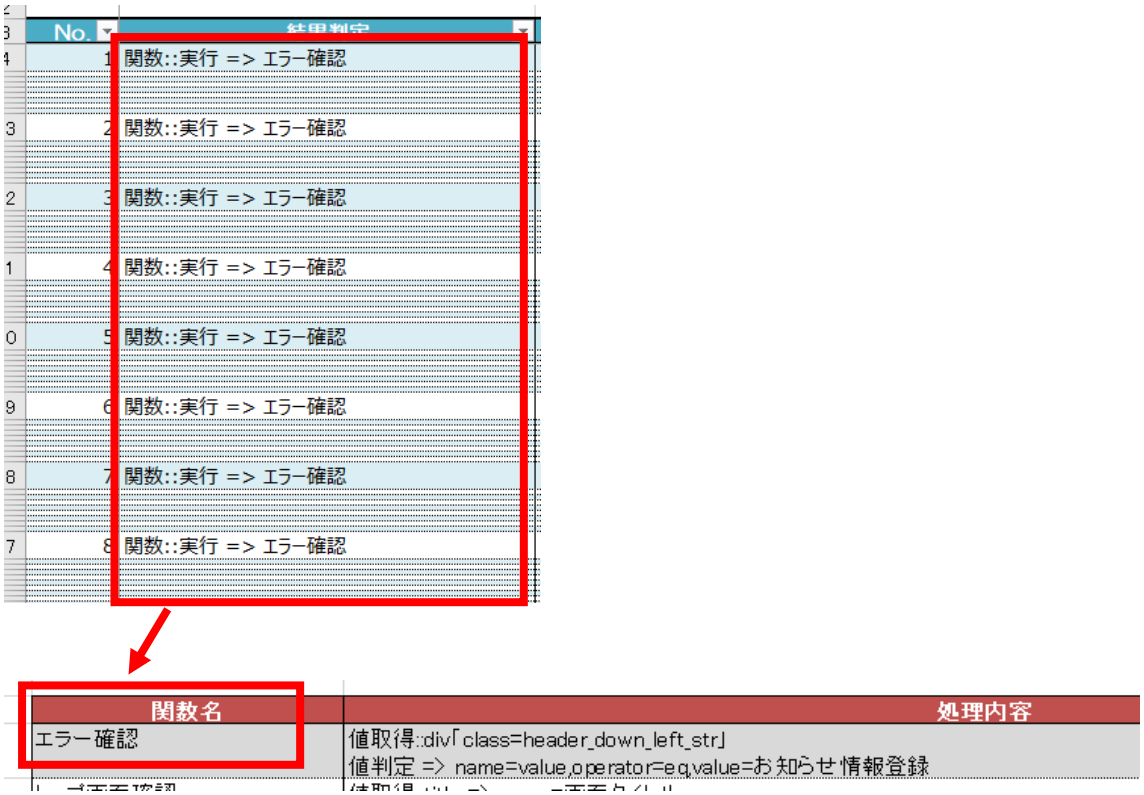
タイトル、お知らせ内容が未入力のパターンは、以下のようにケース毎に空白となる記載を行います。この 2 ケースでタイトルが未入力、データ種別が未入力、お知らせ内容が未入力の入力パターンのテストケースを確認できます。

No.	項目名	パラメータ名	値
7	告知期間（From：年）	input「name=START_DAY_YEAR」	2020
	告知期間（From：月）	input「name=START_DAY_MONTH」	10
	告知期間（From：日）	input「name=START_DAY_DAY」	01
	告知期間（To：年）	input「name=END_DAY_YEAR」	2020
	告知期間（To：月）	input「name=END_DAY_MONTH」	10
	告知期間（To：日）	input「name=END_DAY_DAY」	30
	タイトル	input「name=TITLE」	
	データ種別	select「name=DATA_KIND」	0
	お知らせ内容	textarea「name=INFORMATION」	内容
8	告知期間（From：年）	input「name=START_DAY_YEAR」	2020
	告知期間（From：月）	input「name=START_DAY_MONTH」	10
	告知期間（From：日）	input「name=START_DAY_DAY」	01
	告知期間（To：年）	input「name=END_DAY_YEAR」	2020
	告知期間（To：月）	input「name=END_DAY_MONTH」	10
	告知期間（To：日）	input「name=END_DAY_DAY」	30
	タイトル	input「name=TITLE」	タイトル
	データ種別	select「name=DATA_KIND」	0
	お知らせ内容	textarea「name=INFORMATION」	

上記の設定にて No.毎に各項目の入力設定までを実行しますが、入力チェックのテストとしては確認ボタンをクリックする必要があります。入力値を設定した後にどのような操作を行うかは、テスト設定シートとスクリプトシートに記載します。



結果判定ではエラー確認の関数を呼び出して、確認画面へ遷移していないことを確認しています。



③ 全てを入力してエラーが発生しないことを確認する。

全ての項目に値を設定して確認画面へ遷移することを確認します。

No.	項目名	パラメータ名	値
9	告知期間（From：年）	input「name=START_DAY_YEAR」	2020
	告知期間（From：月）	input「name=START_DAY_MONTH」	10
	告知期間（From：日）	input「name=START_DAY_DAY」	01
	告知期間（To：年）	input「name=END_DAY_YEAR」	2020
	告知期間（To：月）	input「name=END_DAY_MONTH」	10
	告知期間（To：日）	input「name=END_DAY_DAY」	30
	タイトル	input「name=TITLE」	タイトル
	データ種別	select「name=DATA_KIND」	0
	お知らせ内容	textarea「name=INFORMATION」	内容

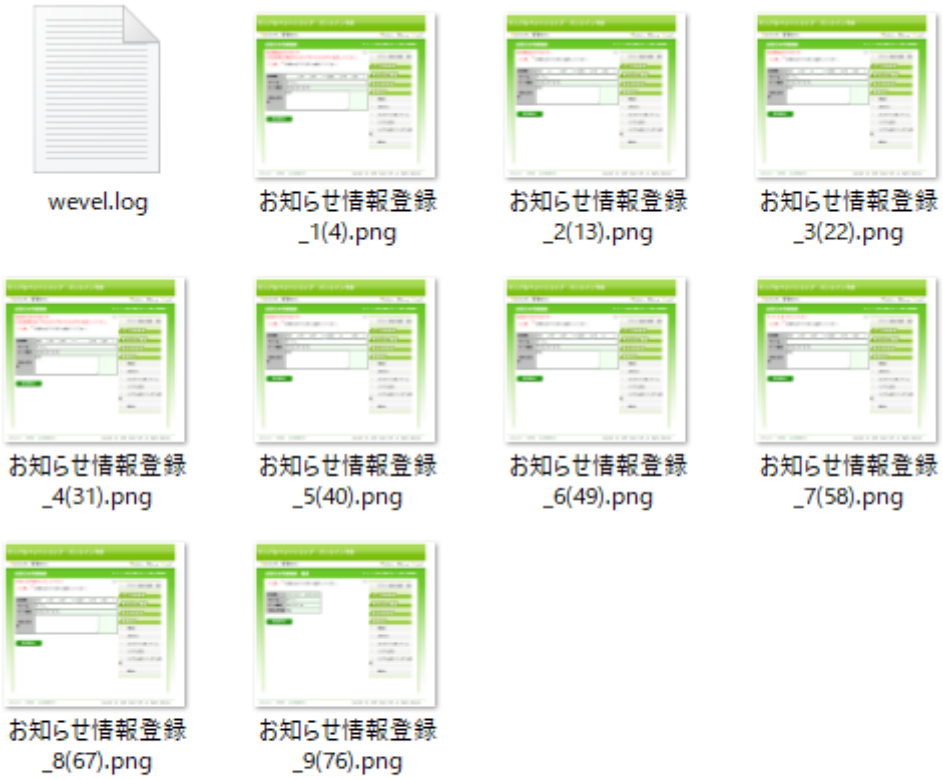
結果判定では正常確認の関数を呼び出して、確認画面へ遷移していることを確認しています。

No.	結果判定
9	関数::実行 => 正常確認

関数名	処理内容
正常確認	値取得::div「class=header_down_left_str」 値判定 => name=value,operator=eq,value=お知らせ情報登録　確認

４－４－４．実行結果について

マトリクスベースのテストケースでは、シート実行処理のスクリプトに「キャプチャ」を記載することで、各行のスクリプトを実行すると現在の画面のイメージを出力します。もし、スクリプトの途中で入力値を入れた直後の画面イメージも取得したい場合は、シート実行前操作のスクリプトに「キャプチャ」を記載することで任意のタイミングにて画面イメージの出力が行えます。



５．スクリプトを作成する

Tester24 で利用可能なスクリプトのフォーマットと用途について、サンプルを交えて説明します。

スクリプト内にコメントを記載したい場合は、#を先頭に記載してください。

なお、スクリプト中の改行は許容していないので、スクリプトは１行で記載ください。各種サンプルスクリプトは sample フォルダのスクリプトサンプル.xlsx を参考ください。

５－１．URL を表示する

■概要

内部ブラウザで指定した URL を表示します。

■フォーマット

URL 表示 => URL

項目	必須	設定内容
URL	○	ブラウザに表示したい URL を記載する。

■記載サンプル

No	動作	スクリプト
1	5-1.URLを表示する	URL表示 => http://hyldingsystem.co.jp/ キャプチャ::png

■スクリプト実行後のキャプチャ出力結果

指定した URL の Web 画面がキャプチャされる。

５－２．画面をキャプチャする

■概要

内部ブラウザで表示中の画面をキャプチャして画像ファイルに出力します。

■フォーマット

キャプチャ::ファイル形式 => name=ファイル名,adjust_height=高さ補正,adjust_width=幅補正,fixed_height=高さ固定,fixed_width=幅固定

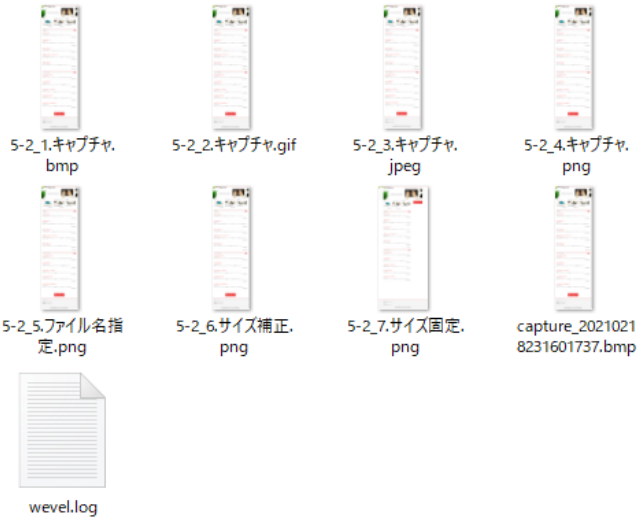
項目	必須	設定内容
ファイル形式		出力するファイル形式を指定する。bmp、gif、jpeg、png のいずれかを指定できる。省略した場合は png で出力する。
ファイル名		出力するファイル名を指定する。拡張子はファイル形式に従って自動的に付与される。 省略した場合はタイムスタンプでファイル名を自動生成して出力する。
高さ補正		キャプチャ出力時の高さは、ブラウザが自動で判定したサイズで出力されるが、想定と異なる高さで出力された場合に、本パラメータにて高さを調整する。
幅補正		キャプチャ出力時の幅は、ブラウザが自動で判定したサイズで出力されるが、想定と異なる幅で出力された場合に、本パラメータにて幅を調整する。
高さ固定		キャプチャ出力時の高さは、ブラウザが自動で判定したサイズで出力されるが、想定と異なる高さで出力された場合に、本パラメータにて高さを固定する。高さ固定を指定した場合、高さ補正による指定は無視される。
幅固定		キャプチャ出力時の幅は、ブラウザが自動で判定したサイズで出力されるが、想定と異なる幅で出力された場合に、本パラメータにて幅を固定する。幅固定を指定した場合、幅補正による指定は無視される。

■記載サンプル

No	動作	スクリプト
2	5-2.画面をキャプチャする	URL表示 => http://hyldingsystem.co.jp/ #パラメータ未指定 キャプチャ #ファイル形式を指定 キャプチャ::bmp => name=5-2_1.キャプチャ キャプチャ::gif => name=5-2_2.キャプチャ キャプチャ::jpeg => name=5-2_3.キャプチャ キャプチャ::png => name=5-2_4.キャプチャ #ファイル名を指定 キャプチャ::png => name=5-2_5.ファイル名指定 #高さ補正、幅補正を指定 キャプチャ::png => name=5-2_6.サイズ補正, adjust_height=100, adjust_width=200 #高さ固定、幅固定を指定 キャプチャ::png => name=5-2_7.サイズ固定, fixed_height=3000, fixed_width=1200

■スクリプト実行後のキャプチャ出力結果

様々なフォーマットやサイズで出力される。



■その他

ページによっては、ファイル形式が PNG 以外だと正しく取得できないことがあるのでご注意ください。

５－３．クリックする

■概要

内部ブラウザで表示中画面のボタンやリンクをクリックする。

■フォーマット

クリック::検索条件 => 待機時間（ミリ秒）

項目	必須	設定内容
検索条件	○	検索条件には ID を指定する方法とタグを指定する方法がある。 ID で指定する場合は、先頭に#を指定する。 例） クリック::#id タグで指定する場合は、タグ名と属性を指定する。複数の要素が一致した場合、$にてインデックスを指定することが可能。インデックスが未指定の場合は、最初に見つかった要素を対象とする。 例） クリック::a「class=Tab__link bdA5」$3 また、先頭と末尾にはアスタリスクを指定することで部分一致を行うことが可能。 例） クリック::a「class=*Tab__link*」$3
待機時間（ミリ秒）		クリックを行った後にページ読み込みが発生している場合は、読み込みが完了するか指定した待機時間（ミリ秒）まで処理を行わない。最大で 60 秒まで指定可能で、省略した場合は 1 秒を設定したものとみなす。

■記載サンプル

No	動作	スクリプト

	5-3.クリックする	# タグ指定でリンクをクリックする
		# Aタグのタグの値が「サービス＆ソリューション」をクリック
		URL表示 => http://hyldingsystem.co.jp/
		クリック::a「タグの値=*サービス＆ソリューション*」
		キャプチャ::png => name=5-3_1.サービス＆ソリューション

3		# タグ指定でリンクをクリックする
		# Aタグのhrefが「http://hyldingsystem.co.jp/?page_id=」の1個目をクリック
		URL表示 => http://hyldingsystem.co.jp/
		クリック::a「href=http://hyldingsystem.co.jp/?page_id*」$0
		キャプチャ::png => name=5-3_2.1つ目のリンク

■スクリプト実行後のキャプチャ出力結果

条件に一致したリンク名をクリックしてキャプチャされる。

条件に一致した 1 つ目の href をクリックしてキャプチャされる。

５－４．入力する

■概要

内部ブラウザで表示中画面のテキストボックス、テキストエリア、ラジオボタン、チェックボックス、プルダウンに任意の値を入力する。

■フォーマット

入力::検索条件 => 任意の値

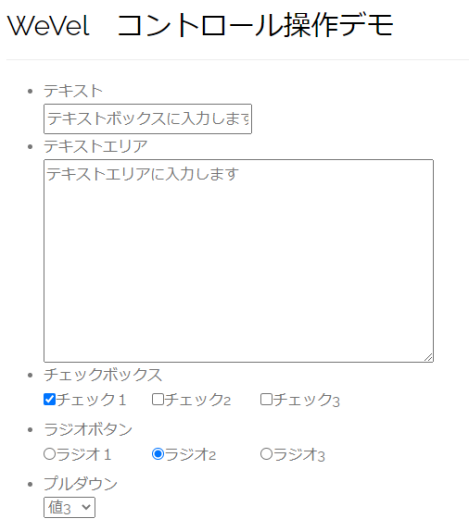
項目	必須	設定内容
検索条件	○	「５－３．クリックする」と同様
任意の値	○	テキストボックス、テキストエリアの場合は、入力したい任意の文字列を指定する。 ラジオボタン、チェックボックス、プルダウンの場合は、入力したい任意の値（value）を指定する。

■記載サンプル

No	動作	スクリプト
4	5-4.入力する	URL表示 => https://app.hyldingsystem.work/captureService/capture/control # テキストボックスに入力する。 入力::#text_test => テキストボックスに入力します # テキストエリアに入力する。 入力::#area_test => テキストエリアに入力します # チェックボックスを選択する。 入力::#check_1 => true # ラジオボタンを選択する。 入力::#radio_2 => true # プルダウンを選択する。 入力::#select_test => value3 キャプチャ::png => name=5-4_キャプチャ

■スクリプト実行後のキャプチャ出力結果

各コントロールに値を設定後にキャプチャされる。



５－５．属性を設定する

■概要

内部ブラウザで表示中画面の任意の要素に対して、指定した属性を設定します。既に属性値が設定されていた場合は上書きされます。既存の値に追加したい場合は、属性追加を利用してください。

■フォーマット

属性設定::検索条件 => 属性名=属性値, …

※属性名と属性値はカンマ（,）で複数指定することが可能です。

項目	必須	設定内容
検索条件	○	「５－３．クリックする」と同様
属性名	○	設定したい属性の名前を指定する。
属性値	○	設定したい属性の値を指定する。

■記載サンプル

No	動作	スクリプト
5	5-5.属性を設定する	URL表示 => http://hyldingsystem.co.jp/ # h2タグの文字列をセンタリングし、文字を太字の青色に変更する。 属性設定::h2 => align=center, style=color: blue; font-weight: bold キャプチャ::png => name=5-5_キャプチャ

■スクリプト実行後のキャプチャ出力結果

H2 タグの文字列を青色の太字に変更した状態でキャプチャされる。

５－６．属性を追加する

■概要

内部ブラウザで表示中画面の任意の要素に対して、指定した属性を追加します。既に属性値が設定されていた場合は半角スペースを付与して値が連結されます。既存の値をクリアしたい場合は、属性設定を利用してください。

■フォーマット

属性追加::検索条件 => 属性名=属性値, …

※属性名と属性値はカンマ（,）で複数指定することが可能です。

項目	必須	設定内容
検索条件	○	「５－３．クリックする」と同様
属性名	○	設定したい属性の名前を指定する。
属性値	○	追加したい属性の値を指定する。

■記載サンプル

No	動作	スクリプト
6	5-6.属性を追加する	URL表示 => http://hyldingsystem.co.jp/ # h2タグの文字列をセンタリングし、文字を太字の青色に変更する。 属性設定::h2 => align=center, style=color: blue; font-weight: bold; # 上記に点線を追加する。 属性追加::h2 => style=border-bottom: dotted 2px; キャプチャ::png => name=5-6_キャプチャ

■スクリプト実行後のキャプチャ出力結果

H2 タグに罫線を追加した状態でキャプチャされる。

５－７．待機する

■概要

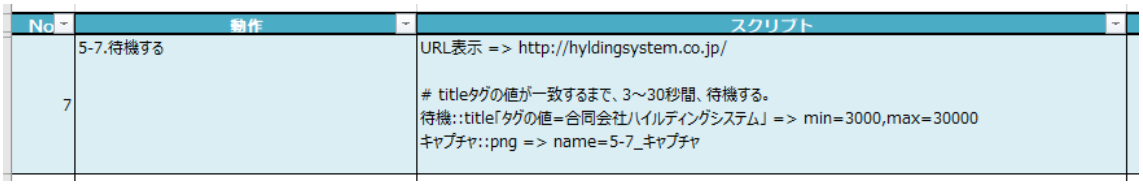
内部ブラウザで表示中画面に指定した条件と一致する要素が出現するまで待機します。ただし、最大待機時間を超過した場合は処理を続行します。
スクリプトでリンクやボタンをクリックして画面遷移を行う際に、正しく読み込みが完了できない場合があります。そういった場合に一致するタイトルが表示されるまで待機することで回避できることがあります。

■フォーマット

待機::検索条件 => min=最小待機時間（ミリ秒）,max=最大待機時間（ミリ秒）

項目	必須	設定内容
検索条件	○	「５－３．クリックする」と同様
最小待機時間（ミリ秒）		検索条件が一致するまで待機する最小の時間を指定する。省略した場合は、100 ミリ秒を設定したものとみなす。
最大待機時間（ミリ秒）		検索条件が一致するまで待機する最大の時間を指定する。最大で 300 秒まで指定可能で、省略した場合は 5 秒を設定したものとみなす。

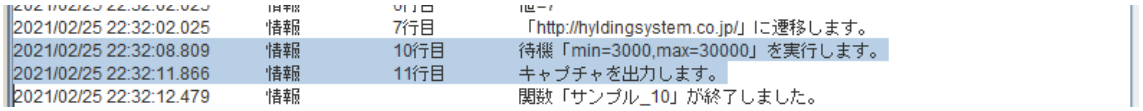
■記載サンプル



No	動作	スクリプト
7	5-7.待機する	URL表示 => http://hyldingsystem.co.jp/ # titleタグの値が一致するまで、3～30秒間、待機する。 待機::title「タグの値=合同会社ハイルディングシステム」=> min=3000,max=30000 キャプチャ::png => name=5-7_キャプチャ

■スクリプト実行後のキャプチャ出力結果

キャプチャ処理まで 3 秒の待機が行われる。



2021/02/25 22:32:02.025	情報	7行目	「http://hyldingsystem.co.jp/」に遷移します。
2021/02/25 22:32:08.809	情報	10行目	待機「min=3000,max=30000」を実行します。
2021/02/25 22:32:11.866	情報	11行目	キャプチャを出力します。
2021/02/25 22:32:12.479	情報		関数「サンプル_10」が終了しました。

５－８．値を取得して利用する

■概要

内部ブラウザで表示中画面から指定した条件と一致する値（タグの値/属性値）を取得します。取得した値は、各種スクリプトにて「%{変数名}」という記載を行うことで利用することができます。既に同一の変数名が存在した場合は上書きされます。

■フォーマット

値取得::検索条件 => name=変数名,regex=正規表現,prefix=接頭辞,suffix=接尾辞,targetAttribute=取得対象の属性名

※正規表現にカンマを使用する場合は、「@COMMA@」と記載する。

項目	必須	設定内容
検索条件	○	「５－３．クリックする」と同様
変数名		取得した値を格納する変数名を指定する。未指定の場合、value という変数名で格納する。
正規表現		取得した値に対して、正規表現を利用して最初にマッチングした部分だけを抜き出して、変数に格納する。 未指定の場合は取得した値をそのまま変数に格納する。
接頭辞		取得した値に接頭辞を付与したい場合に指定する。
接尾辞		取得した値に接尾辞を付与したい場合に指定する。
取得対象の属性名		条件に一致したタグの値ではなく、属性値を変数に格納したい場合は、取得したい属性名を指定する。未指定の場合はタグの値を取得する。

■記載サンプル

No	動作	スクリプト

8	5-8.値を取得して利用する	# Web画面から2番目のh3タグの値を取得する。
		URL表示 => http://hyldingsystem.co.jp/%E8%A3%BD%E5%93%81/
		値取得::h3$1 => name=h3の値
		キャプチャ::png => name=5-8_1_キャプチャ
		
		# 上記で取得した値（h3の値）をYahoo!の検索エリアに入力する。
		URL表示 => https://www.yahoo.co.jp/
		入力::input「type=search」=> %{h3の値}
		キャプチャ::png => name=5-8_2_キャプチャ

■スクリプト実行後のキャプチャ出力結果

他の画面から取得した値を検索エリアに設定した状態がキャプチャされる。

５－９．値を設定して利用する

■概要

指定した変数に任意の値を設定します。設定元の値は変数も指定することができるので、「値取得」にて取得した値に対して四則演算を行うことが可能です。

■フォーマット

値設定 => name=変数名,value1=値 1,value2=値 2,operator=演算子,digits=小数桁数

項目	必須	設定内容
変数名		取得した値を格納する変数名を指定する。未指定の場合、value という変数名で格納する。
値 1		変数に設定する値を指定する。
値 2		値 1 に対して四則演算を行う場合に、対象となる数値を指定する。
演算子		四則演算を行う場合に指定する。+-/*が指定可能。
小数桁数		値が数値の場合に四捨五入する小数点以下桁数を指定する。小数点第一位で四捨五入する場合は 0 を指定し、小数点第二位以上で四捨五入する場合は 1 以上を指定する。

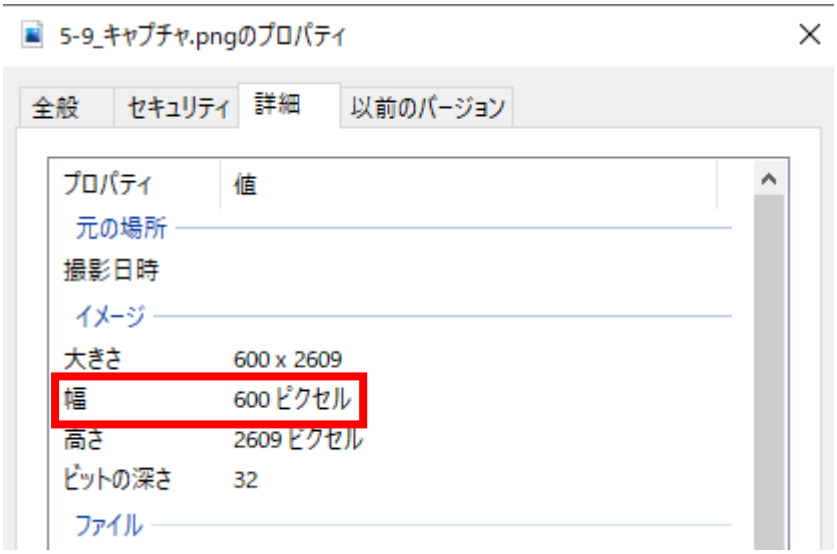
■記載サンプル

No	動作	スクリプト

9	5-9.値を設定して利用する	#幅を計算(400 + 200)して利用する
		値設定 => name=幅,value1=400,value2=200,operator=+
		URL表示 => http://hyldingsystem.co.jp/
		ｷｬﾌﾟﾁｬ::png => name=5-9_ｷｬﾌﾟﾁｬ,fixed_width=%{幅}

■サンプルスクリプト

値設定した 600px にてキャプチャされる。



５－１０．一連の処理を再利用する

■概要

作成した一連の処理を他の場所でも使用したい場合に、関数として再利用可能な状態にすることができます。関数内で関数を利用することも可能ですが、記述方法によっては無限ループに陥る恐れがあるため、10 階層以上の関数利用を検知した場合には処理を中断します。

■フォーマット

関数::処理区分 => 関数名

項目	必須	設定内容
処理区分	○	処理区分には開始、終了、実行の何れかを指定します。開始は一連の処理の開始位置を表します。終了は一連の処理の終了位置を表します。実行は定義した関数の実行を行います。
関数名	○	変数に設定する値を指定する。 処理区分に開始を指定した場合に、関数名を指定する必要があります。

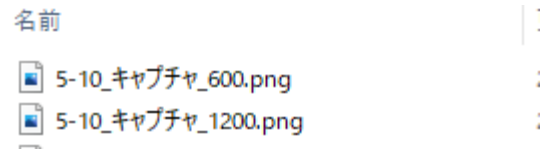
■記載サンプル

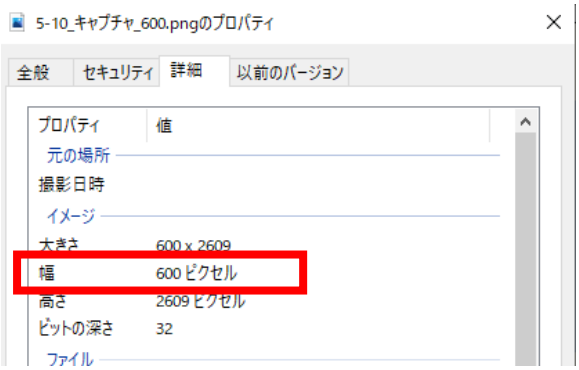
No	動作	スクリプト

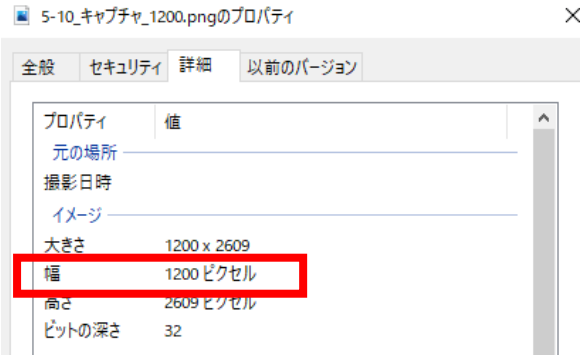
10	5-10.一連の処理を再利用する	# 関数の実行
		値設定 => name=幅,value1=600
		関数::実行 => 会社のHPをキャプチャ
		
		# サイズを変えて実行
		値設定 => name=幅,value1=1200
		関数::実行 => 会社のHPをキャプチャ
		
		# 関数を定義する。
		関数::開始 => 会社のHPをキャプチャ
		URL表示 => http://hyldingsystem.co.jp/
		キャプチャ::png => name=5-10_キャプチャ_%{幅},fixed_width=%{幅}
		関数::終了

■サンプルスクリプト

再利用できる関数を 2 回実行することで、サイズの異なる 2 つのキャプチャが出力される。







５－１１．値を判定する

■概要

変数の値を判定し、その結果に従って以降の動作を制御します。値が一致した場合に関数にて処理を行うことや、スクリプトの終了や処理中の関数を中断することができます。

■フォーマット

値判定 => name=変数名,operator=演算子,value=判定値,resultName=判定結果名,matchStr=一致時の文言,umMatchStr=不一致時の文言,afterFunction=一致時の処理,afterProcess=一致時の挙動

項目	必須	設定内容
変数名	○	判定する対象となる変数名を指定します。
演算子	○	変数の値と判定値の一致条件を指定します。変数の値が数値の場合は、一致（eq）、不一致（ne）、以上（ge）、以下（le）、大きい（gt）、未満（lt）の何れかを指定します。変数の値が文字の場合は、一致（eq）、不一致（ne）、正規表現による一致（regex）、正規表現による不一致（notRegex）の何れかを指定します。
判定値	○	判定する対象となる判定値を指定します。
判定結果名		指定した変数名に判定結果を変数に格納することができます。ただし、変数名には「@値判定.」という接頭辞が自動的に付与されるので、利用する場合は接頭辞も含めて変数名を指定する必要があります。
一致時の文言		値判定で一致した場合に、判定結果名に設定する文言を指定します。省略した場合は、「一致」という文言を設定します。
不一致時の文言		値判定で不一致の場合に、判定結果名に設定する文言を指定します。省略した場合は、「不一致」という文言を設定します。

項目	必須	設定内容
一致時の処理		値判定で一致した際に、定義してある関数に処理を行わせたい場合に関数名を指定します。関数の処理が完了すると値判定の次の行から処理を再開します。
一致時の挙動		値判定で一致した際に、以降の挙動を指定します。スクリプトを終了する場合は、exit を指定します。値判定が関数内での処理の場合に、現在の関数の以降の処理を行いたくない場合は、return を指定します。省略した場合は処理が継続されます。 一致時の処理に関数を指定した場合、関数を実行した後に一致時の挙動に指定された動作を行います。

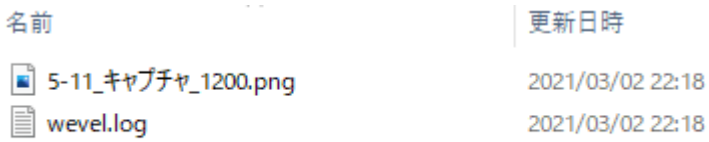
■記載サンプル

No	動作	スクリプト

11	5-11.値を判定する	# 関数の実行（関数が中断される値）
		値設定 => name=幅,value1=600
		関数::実行 => 会社のHPをキャプチャ
		
		# 関数の実行（関数が継続される値）
		値設定 => name=幅,value1=1200
		関数::実行 => 会社のHPをキャプチャ
		
		# 関数の実行（スクリプトが終了する値）
		値設定 => name=幅,value1=2000
		関数::実行 => 会社のHPをキャプチャ
		
		# スクリプトが終了するため、到達しない設定
		キャプチャ::png => name=到達しない,fixed_width=%{幅}
		
		# 関数を定義する。
		関数::開始 => 会社のHPをキャプチャ
		　# 幅の値が1000以下の場合は関数を中断する。
		　値判定 => name=幅,operator=lt,value=1000,afterProcess=return
		
		　# 幅の値が2000以上の場合はスクリプトを終了する。
		　値判定 => name=幅,operator=ge,value=2000,afterProcess=exit
		
		　URL表示 => http://hyldingsystem.co.jp/
		　キャプチャ::png => name=5-11_キャプチャ_%{幅},fixed_width=%{幅}
		関数::終了

■サンプルスクリプト

　値設定された内容を参照し、条件を満たした場合だけキャプチャされる。上記では幅の値が 1000～2000 の場合に条件を満たすため、1 つだけキャプチャされる。



６．Tester24 でサポートされていない機能

本バージョンの Tester24 では以下の機能はサポートされておりません。今後の利用者からのリクエスト状況を見て、技術的に解決可能な事象から対応を進めていく予定です。

・ファイルのダウンロード
・ファイルのアップロード
・ポップアップ画面の操作

変更履歴

バージョン	日付	内容
1.0	2021/07/29	1.0 版リリース