Ranorex

複数のプラットフォームで画像比較を成功させるための方法

Last updated: 2026.10

目次

内容

目次 **2**

1 はじめに **3**

本書について 3

2 Ranorex の画像比較 **4**

2.1 ContainsImage による画像比較 4

2.2 CompareImage による画像比較 5

3 画像比較が失敗する主な要因 **6**

4 画像比較が失敗する主な要因の解決策 **7**

4.1 マシンの解像度やテスト対象アプリケーションのウィンドウサイズを統一する 8

4.2 Ranorex の標準機能（プリプロセスフィルター・類似度）で差分を許容する **エラー! ブックマークが定義されていません。**

4.2.1 プリプロセスフィルター 11

4.2.2 Similarity(類似度) 15

4.3 他ツール連携により、文字・画像の位置ズレを自動検出して検証する（例：OpenCV 利用）.**エラー! ブックマークが定義されていません。**

4.4 環境ごとに期待画像を切り替えて、テストを実行する ..... **エラー! ブックマークが定義されていません。**

4.5.差分が発生しやすい箇所や動的に変わる箇所をマスキングする 35

1 はじめに

本書について

クライアントマシンやクラウドマシン、リモートデスクトップなど、さまざまな環境で Ranorex の画像比較を実施する場合、環境による表示の微細な差分により、検証が失敗するケースがあります。

Ranorex の画像比較は、ピクセル単位で RGB 値(赤・緑・青の色を組み合わせた数値)を参照した厳密で高精度な検証です。そのため、複数のテスト実行環境でテストする場合に、期待画像と微細な色の違いや文字・グラフィックの位置の微妙なズレなどが発生すると画像検証が失敗しやすくなります。

本書では、テスト実行環境の違いにより、人の目で認識できない微細な差分で失敗する画像比較を成功させるための設定方法について説明します。

本書は、Ranorex の基本機能や操作を理解し、Ranorex の画像検証に関する基礎知識をお持ちの方を対象としています。（弊社で開催している無償ハンズオンセミナーの内容レベル）



メモ

無償ハンズオンセミナーの詳細については、以下のリンクをご確認ください。Ranorex の画像検証の設定方法は、中級コースに含まれます。

- ハンズオンセミナー（初級）
  https://ranorex.techmatrix.jp/handson-details/
- ハンズオンセミナー（中級）
  https://ranorex.techmatrix.jp/handson-intermediate-details/

2 Ranorex の画像比較

テスト実行時に表示された画像が期待画像と一致しているかを判断する指標として、類似度（Similarity)が使用されます。類似度は 0.0 から 1.0 の範囲で表され、1.0 は 100%の一致を意味します。具体的には、比較対象の画像のすべてのピクセルにおいて、赤（R）、緑（G）、青（B）の各色の値が完全に一致している状態を指します。このため、類似度が 1.0 の場合、完全に一致で画像検証の結果は成功と判断されます。

また、Ranorex の画像比較は、Validate アクションで設定でき、ContainsImage と CompareImage の 2 種類があります。

#	Action				
1	Validate	ContainsImage	Screenshot1		RxStudioLogo
2	Validate	CompareImage	Screenshot1		RxStudioLogo

2.1 ContainsImage による画像比較

テスト対象アプリケーションのあるオブジェクト（画像）に対して、期待する画像（画像を比較した際に正となる画像）が含まれているかを検証したい場合は、ContainsImage を使用します。

たとえば、下図のアプリケーション画面において、カレンダー（赤枠）をオブジェクトとして取得し、カレンダー内にある、"xxxx 年 xx 月"の画像部分が含まれているかを検証します。



ContainsImage では、期待する画像で指定した範囲（ピンク色の枠） の画像が、テスト実行時に表示される対象オブジェクトの画像に含まれるかを検証します。選択しない箇所については検証対象に含まれません。

2.2 CompareImage による画像比較

テスト対象アプリケーションのあるオブジェクト（画像）に対して、期待する画像と一致するかを検証したい場合は、CompareImage を使用します。

たとえば、下図のアプリケーション画面において、カレンダー（赤枠）をオブジェクトとして取得し、カレンダー全体の画像が期待する画像と一致するかを検証します。



CompareImage では、期待する画像で指定した範囲（ピンク色の枠） の画像が、テスト実行時に表示される対象オブジェクトの画像と一致するかを検証します。また CompareImage では、画像だけではなく、画像のサイズも比較条件に含まれます。

3 画像比較が失敗する主な要因

Ranorex の画像比較が失敗する主な要因は、以下の通りです。

- **マシンの解像度やウィンドウサイズが統一されていない複数環境でのテスト**

  マシンの解像度やウィンドウサイズが同一でない場合、テスト作成時と画像サイズが異なるため、失敗することがあります。

- **Windows OS の異なるバージョンでのテスト**

  Windows OS のアップデートにより、描画方式などが変わることで、アプリケーションの見た目に差異が生じて失敗する場合があります。たとえば、Windows 10 で作成したテストを Windows11 で実行すると失敗することがあります。

- **クラウド環境でのテスト**

  クライアント環境で作成したテストをクラウド環境でテストする場合、解像度や表示倍率の差異、GPU・グラフィックドライバー（GPU を動かすためのソフトウェア）の種類やバージョン差異などによる描画環境の違いで、画面表示結果に差異が生じると失敗することがあります。

- **リモートデスクトップ接続によるリモートテスト**

  クライアント環境で作成したテストをリモートデスクトップ接続先のマシン環境でテストする場合、環境の違いにより見た目に差異が発生すると失敗することがあります。

- **検証対象の画像内に動的なコンテンツが含まれるテスト**

  テスト環境によって変化するような動的なコンテンツが画像内に含まれている場合、期待画像と差分が出るため失敗します。

4 画像比較が失敗する主な要因の解決策

「3.画像比較が失敗する主な要因」であげた内容は、大きく分けて次の 3 パターンに分類できます。４つの解決策のなかから、それぞれの要因に対応する解決策をお試しください。

- **解像度・ウィンドウサイズの違いが原因の場合**

  「4.1. マシンの解像度やテスト対象アプリケーションのウィンドウサイズを統一する」で対応します。

- **Windows OS バージョン・クラウド環境・リモートデスクトップ環境の違いが原因の場合**

  いずれも「微細な表示差分」として現れるため、「4.2. Ranorex の標準機能（プリプロセスフィルター・類似度）で差分を許容する」、「4.3. 他ツール連携により、文字・画像の位置ズレを自動検出して検証する (例：OpenCV 利用)」、「4.4.環境ごとに期待画像を切り替えて、テストを実行する」の解決策のいずれかで対応します。まずは Ranorex の標準機能のみで完結する「4.2. Ranorex の標準機能（プリプロセスフィルター・類似度）で差分を許容する」を試し、精度が足りない場合や文字・グラフィックの位置ズレを自動検知して補正したい場合は「4.3. 他ツール連携により、文字・画像の位置ズレを自動検出して検証する (例：OpenCV 利用)」をご検討ください。環境ごとに正しい期待画像を用意できる場合は「4.4. 環境ごとに期待画像を切り替えて、テストを実行する」がもっとも確実です。

- **テスト環境によって変化するコンテンツが画面に含まれる場合**

  「4.5. 差分が発生しやすい箇所や動的に変わる箇所をマスキングする」で対応します。

4.1 マシンの解像度やテスト対象アプリケーションのウィンドウサイズを統一する

失敗要因：テスト実行マシンとテスト作成マシンの解像度やテスト対象アプリケーションのウィンドウサイズが異なる場合
効果：解像度やウィンドウサイズを統一することで、画像サイズなどの比率が同じになるため、差異が出にくくなる

テスト実行マシンにて、テスト作成で使用しているマシンの解像度、DPI スケーリング、アプリケーションのウィンドウサイズを統一ください。

マシンの解像度と DPI スケーリングの設定手順

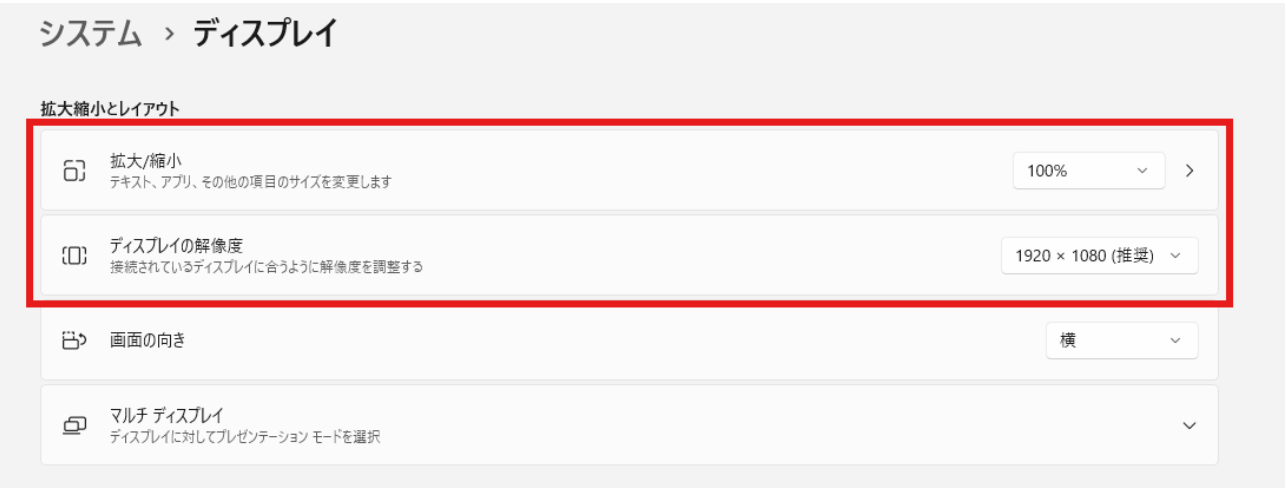
※Windows11 の方法です。

(1) Windows の設定画面を開きます。
スタートメニュー ＞「設定」で開く

(2) Windows の設定画面にて、「システム」＞「ディスプレイ」を選択します。

(3) 「拡大/縮小」(DPI スケーリング)が 100%になっていることをご確認ください。
Ranorex は、100%で利用することを推奨しております。レコーディング時の UI 要素の認識や Ranorex 自体のレイアウト表示に影響があるためです。

(4) 「ディスプレイの解像度」がテスト作成マシンと同様であることをご確認ください。



テスト対象アプリケーションやブラウザーのウィンドウサイズを統一するための設定

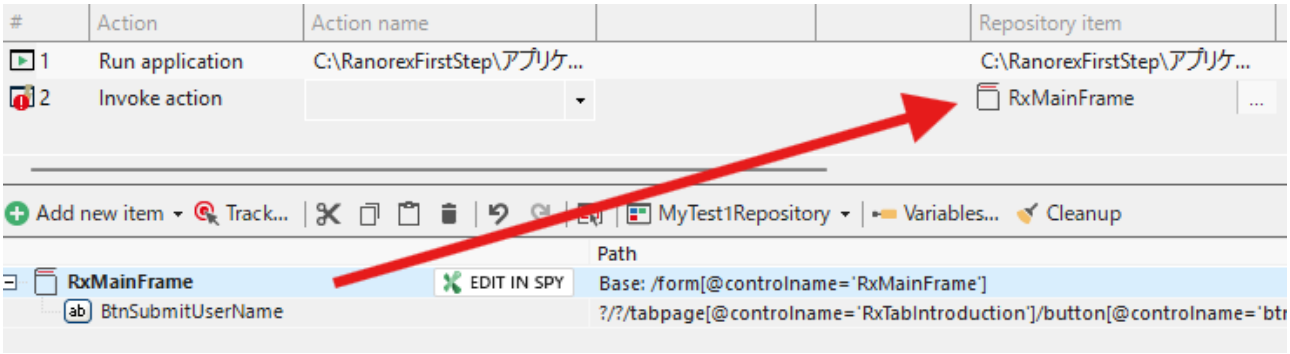
テスト実行時に、Ranorex からテスト対象アプリケーションを起動するときに、常に同じウィンドウサイズで起動することができます。テスト対象アプリケーションの種類により、設定方法は異なります。

① デスクトップアプリ、Edge Chromium の場合

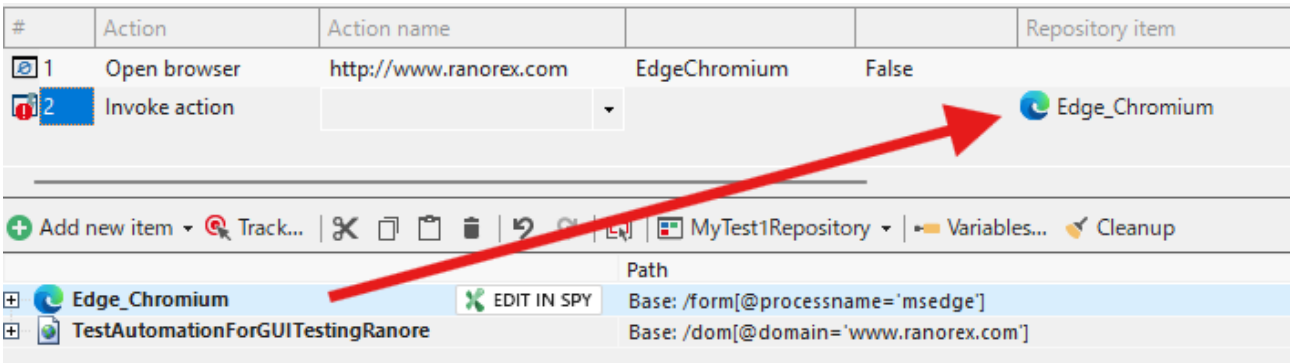
Invoke action アクションの **Maximize** や **Resize(Width,Height)**を使用し、アプリケーションの最上位要素のアプリフォルダーを指定することで、アプリケーションのウィンドウサイズを常に最大化もしくは指定した横と縦のサイズで起動できます。

(1) アプリケーションおよび Edge Chromium ブラウザーの起動後に Invoke action アクションを追加し、Repository item にアプリフォルダーを指定します。

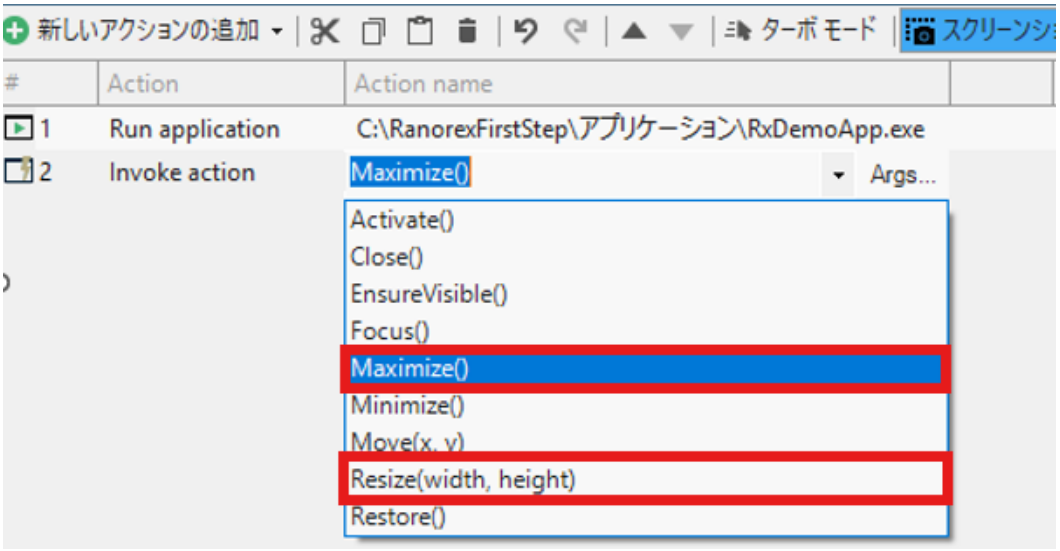
デスクトップアプリの場合：Run application アクションの次に Invoke action アクションを追加し、Repository item にそのアプリケーションの Form 要素を指定します。



Edge Chromium の場合：Open browser アクションの次に Invoke action アクションを追加し、Repository item に Edge ブラウザーの Form 要素を指定します。

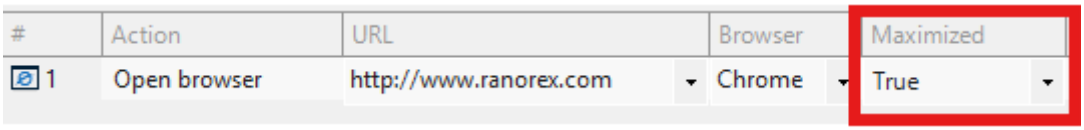


(2) Action name の▼をクリックし、使用するアクションの種類を設定します。
Maximize の場合は、常に最大化して起動、Resize(Width,Height)の場合は、常に指定した横と縦の幅で起動されます。



② Chrome、Firefox、Chromium ブラウザーの場合

(1) OpenBrowser アクションの **Maximized** 設定で、**True** に変更します。この設定でブラウザーを常に最大化して起動できます。



4.2 Ranorex の標準機能（プリプロセスフィルター・類似度）で差分を許容する

失敗要因：Windows OS バージョン・クラウド環境・リモートデスクトップ環境が異なることで微細な差分が発生する場合
効果：Ranorex の設定のみで、人の目では分からない微細な差分を成功とできる

Ranorex の画像比較は、対象となる 2 つの画像を厳密に比較するため、人の目では識別できないような 1 ピクセルの違いに対しても異なる画像としてレポートします。

しかしながら、検証結果として人の目で識別できるレベルや、多少の誤差に対しては成功とさせたい場合において、Ranorex のオプションに用意されているプリプロセスフィルター、または Similarity（類似度）を使用することで、検証内容に合わせた画像比較をおこなうことができます。Ranorex の標準機能のため、コーディング不要で設定できます。

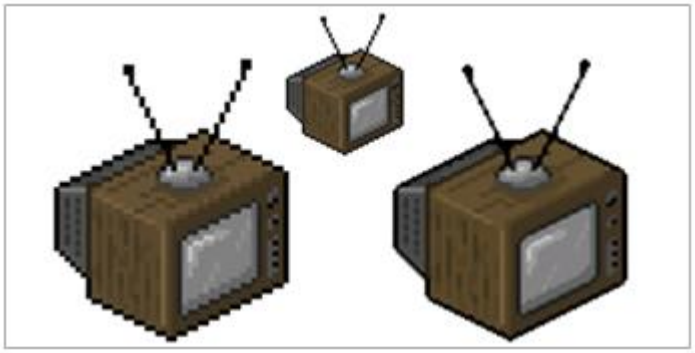
4.2.1 プリプロセスフィルター

プリプロセスフィルダーで画像を加工してから検証することで、画質・解像度の細かい差分や、色味・鮮度の変化に堅牢な画像検証をすることができます。

プリプロセスフィルターの種類

Downsize（ダウンサイジング）

Downsize は、画像の特徴を残しつつスケールを小さくして比較をおこないます。これにより、表面的な違いによってテストが失敗となる可能性が低くなります。また、比較する必要のあるピクセル数が少なくて済むため、テストの実行が速くなります。



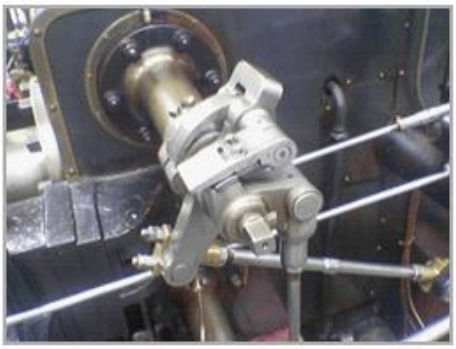
Edges（エッジ）

Edges は、画像の情報量を減らして比較をおこないます。これによりテスト実行を速くします。画像の色や輝度の変化に対して、画像の認識をより堅牢なものにします。



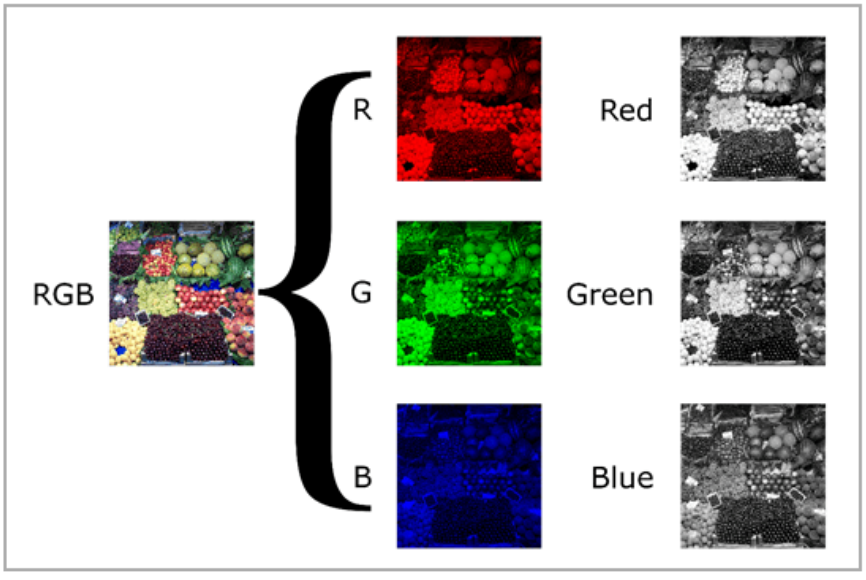

EdgesSobel（エッジソーベル）

EdgesSobel は、画像のエッジ（明るい部分（白）と、暗い部分（黒）の境界）を強調して比較をおこないます。Edges に似ており、色や輝度の変化に対してイメージ認識をより堅牢なものにします。




Grayscale（グレースケール）

Grayscale は、画像の色や輝度の情報を減らして比較をおこないます。（画像をグレースケール 256 階調で表現します）画像の色の変化に対して、イメージ認識をより堅牢なものにします。



Threshold（スレッショルド）

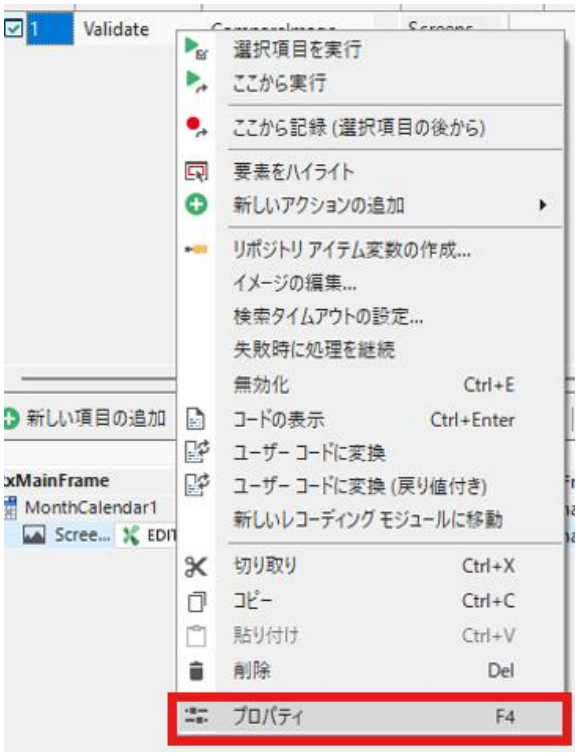
Threshold は、画像を白と黒の 2 色で表して比較をおこないます。たとえば右下の画像では、 暗い木はほぼ完全に黒く、白い雪はほぼ完全に白になります。画像の色や輝度の変化に対して、画像の認識をより堅牢なものにします。



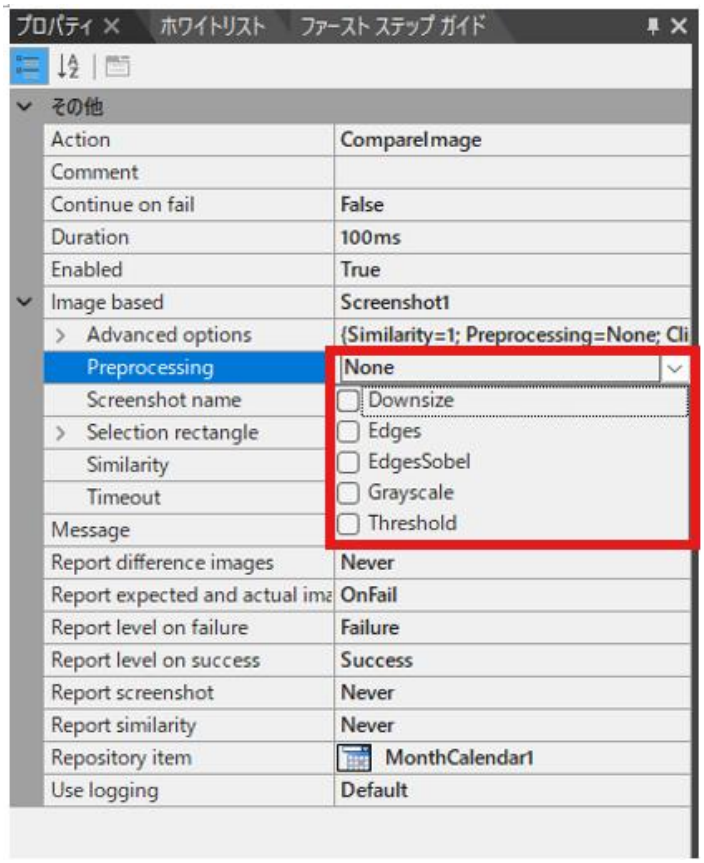


設定手順

(1) 画像検証の Validate アクションを右クリックし、「プロパティ」をクリックします。

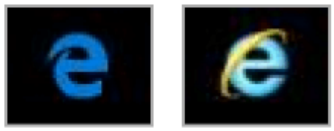


(2) プロパティ画面にて、「Image based」をダブルクリックして展開し、「Preprocessing」にて、対象のプリプロセスフィルターを設定します。



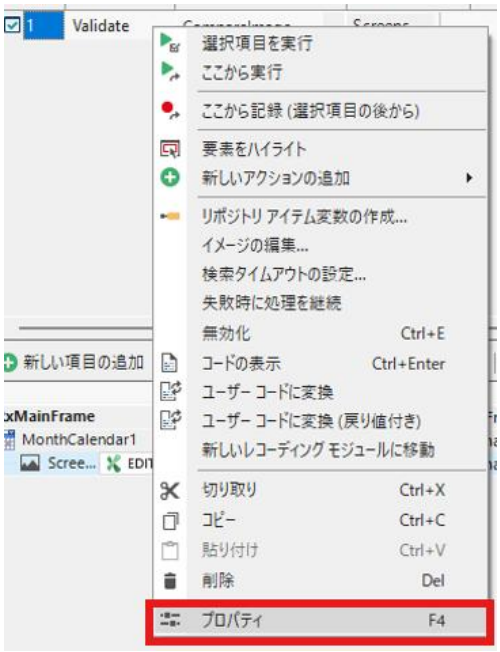
4.2.2 Similarity(類似度)

人の目では認識できない差分がある 2 つの画像の類似度に合わせて Similarity オプションを設定することで、微細な差分を許容し、テストを成功と見なすことができます。類似度は、0.0～1.0 の間で指定され、1.0 = 100%を表します。

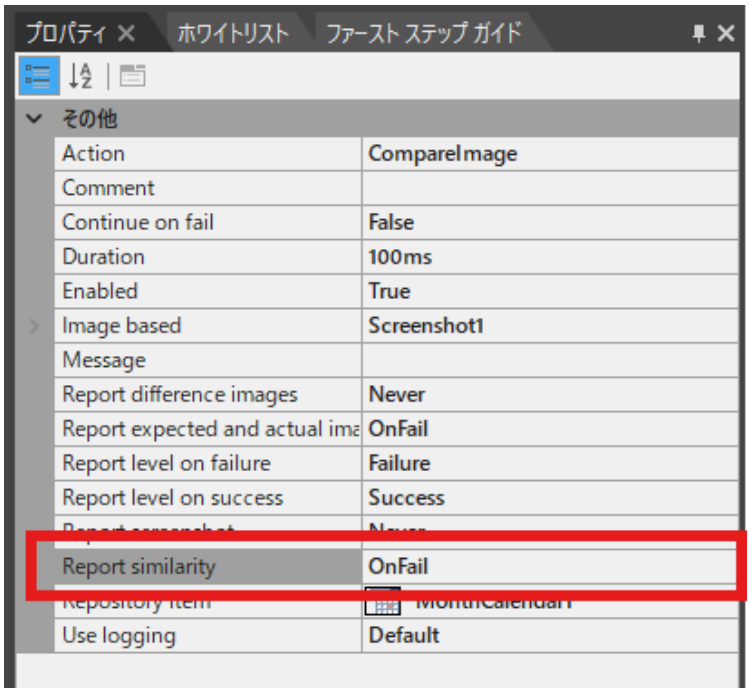
> 注意
>
> 類似度の設定として、0.8 や 0.9 は非常に低い値です。たとえば、以下の Edge および Internet Explorer のアイコンを例に取り上げます。
>
> 
>
> Edge と Internet Explorer のアイコン
>
> それぞれ約 2000 ピクセルであり、その見た目は異なります。類似度 0.9 は、この違いを捉えることができず、同一のものとみなします。これらを異なるものとみなすためには、少なくとも類似度 0.95 以上が必要です。
>
> このため、可能な限り高い類似度の値を常に使用することをお勧めします。1.0 でうまくいかない場合は、通常 0.9999、0.99、0.98 で十分です。0.95 を下回ることはほとんどありません。
>
> 類似度に低い値を設定すると人の目でも判別できるかなり大きな差分も成功となる可能性がありますので、ご注意ください。

設定手順

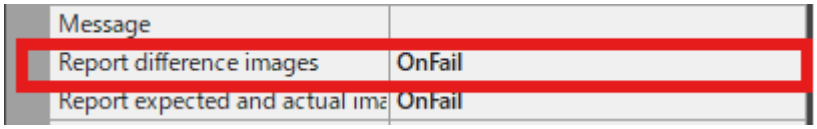
(1) 画像検証の Validate アクションを右クリックし、「プロパティ」を選択します



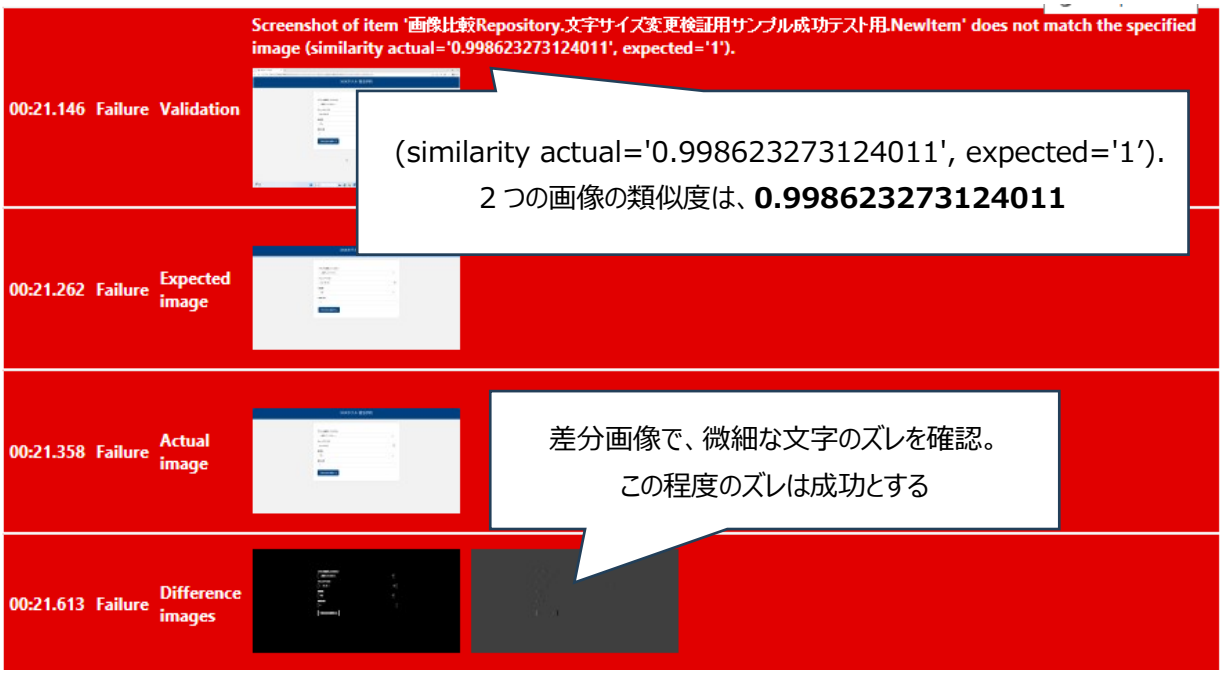
(2) プロパティ画面にて、「Report similarity」を「OnFail」に変更して、画像比較が失敗したときに期待画像と実際の画像の類似度をレポートに出力します。



「Report difference images」を「OnFail」にすることで差分画像もレポートに出力できます。差分画像も確認して成功とさせても問題ないかを確認することをおすすめします。

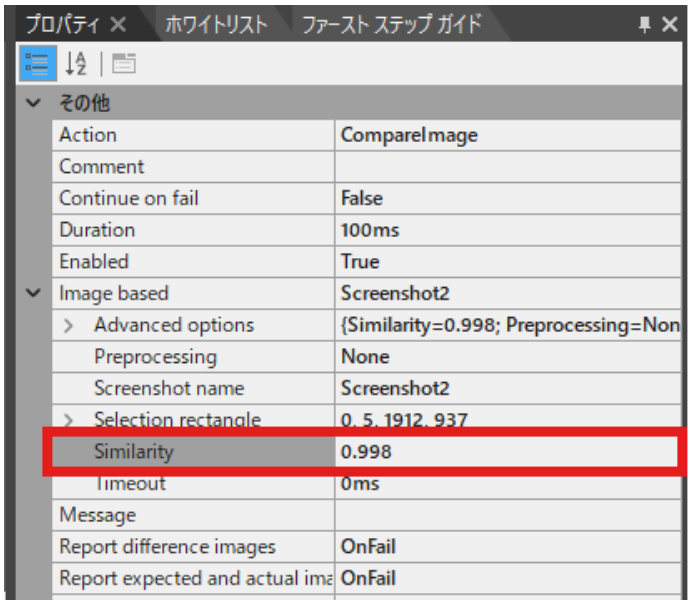


(3) テストを実行して、検証結果を確認します。失敗した場合、2 つの画像の類似度を確認します。



(4) 画像比較の Validate アクションのプロパティを再度開きます。

(5) プロパティ画面にて、「「Image based」」をダブルクリックして展開し、「Similarity」を変更します。手順 3 のレポートで確認した類似度をもとに指定します。



(6) テストを再実行して、画像検証が成功となることを確認します。



> メモ
>
> Ranorex の画像比較オプションのプリプロセスフィルターと Similarity(類似度)については、以下のブログ記事にも記載していますので、ご確認ください。
>
> - イメージ バリデーション（応用）#4
> https://ranorex.techmatrix.jp/blog/2023/03/31/image-validation-application/

4.3 他ツール連携により、文字・画像の位置ズレを自動検出して検証する (例：OpenCV 利用)

失敗要因：Windows OS バージョン・クラウド環境・リモートデスクトップ環境が異なることで微細な差分が発生する場合
効果：コード実装によるカスタマイズで、人の目では分からない微細な差分を自動検知して成功とできる

Ranorex はコーディングしてテストをカスタマイズすることができるため、画像検証をおこなうことができるツールと連携してテストすることもできます。たとえば、OSS ツールである OpenCV と Ranorex を連携し、OpenCV の高度な画像検証アルゴリズムを活用することで、微細な差分を自動的に許容して検証を成功させることができます。

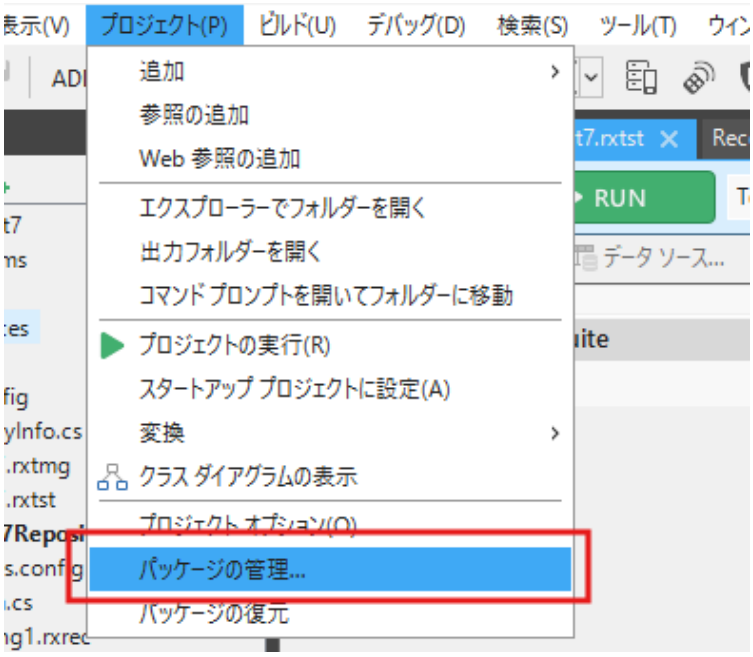
ここでは、OpenCV を利用した画像検証をご紹介します。「2.画像比較のプリプロセスフィルターや類似度を設定する」で紹介した Ranorex の機能の精度で対応できない場合は、この方法をご検討ください。

OpenCV（Open Source Computer Vision Library）は、Intel が開発した画像処理・画像解析・コンピュータビジョン向けのオープンソースライブラリです。**構造類似度指標（SSIM**: Structural Similarity Index Measure）を使用して、画像の明るさ、コントラスト、構造の類似性を評価し、**人間の視覚に近い形で画像の差異を検出**することができます。

この連携により、**文字やグラフィックのわずかなズレや揺れを自動検知・調整**することで、**人間の目では認識できない微細な差分を許容し、**より実用的な判定をおこなうことができます。※画像の複雑さなどで対応できないケースもあります。

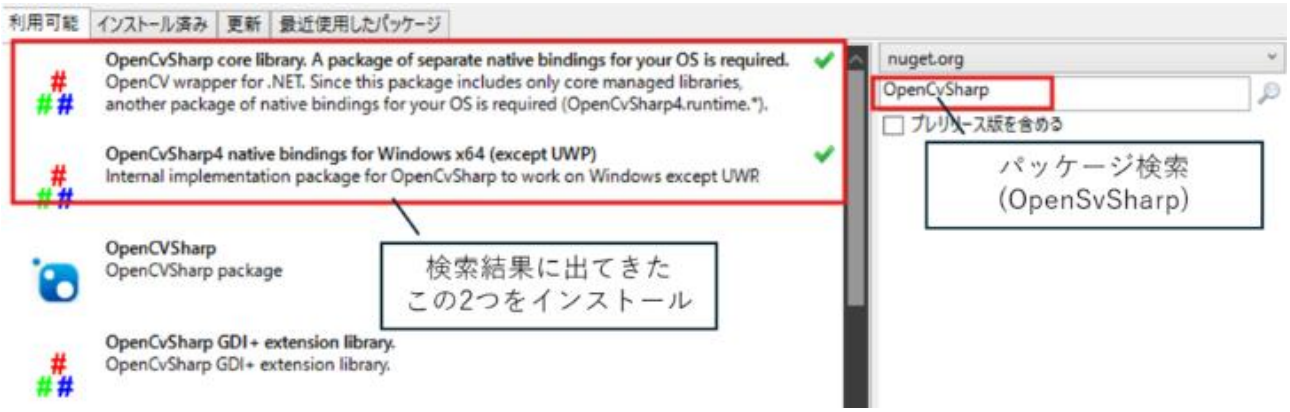
設定手順

(1) Ranorex プロジェクトに OpenCvSharp を追加します。
メニューバーから「プロジェクト」>「パッケージの管理」をクリックします。

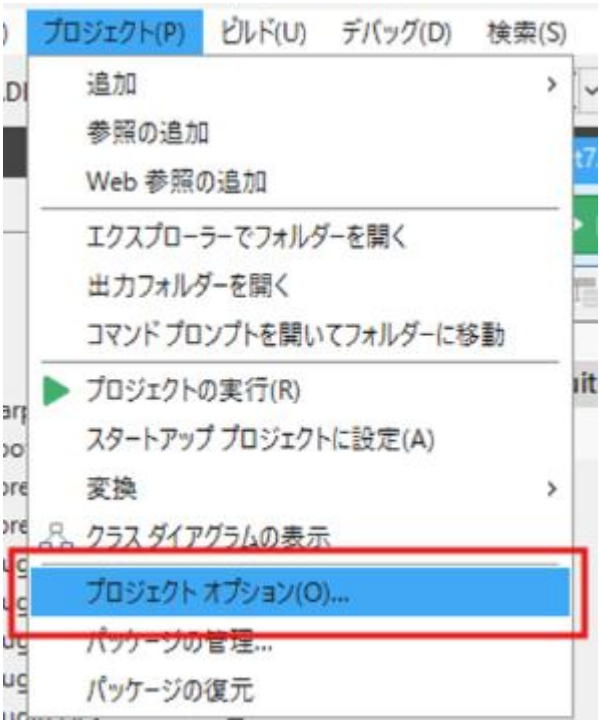


検索ボックスに「OpenCvSharp」を入力し、「OpenCvSharp core library」と「OpenCvSharp4 native bindings for

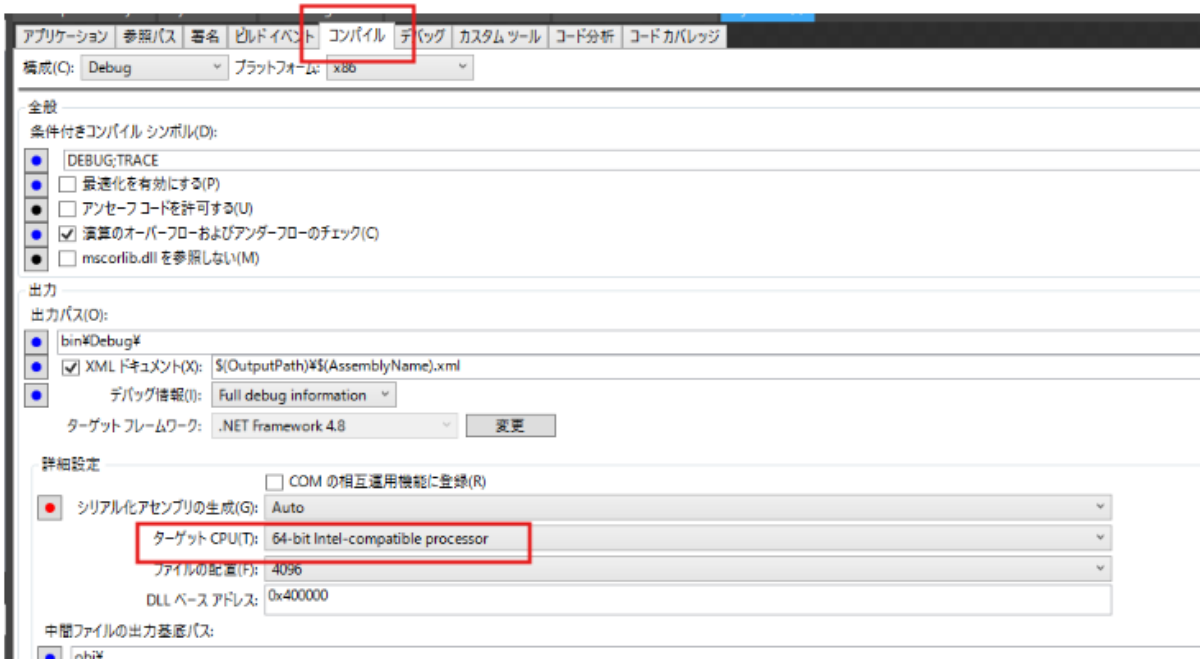
Windows x64」をプロジェクトにインストールする。
※今回、ご紹介するユーザーコードでは StronglyTyped.PixelMatch.net のパッケージも利用しますので、追加が必要です。



(2) ビルド対象（プラットフォームターゲット）を統一します。
Ranorex のプロジェクトで、プロジェクトオプションを開きます。
メニューバーから「プロジェクト」>「プロジェクト オプション」をクリックします。



ターゲット CPU をインストールした OpenCvSharp のビット数に合わせる。上記でインストールしたパッケージは 64bit のため、「64-bit Intel-compatible processor」に設定する。

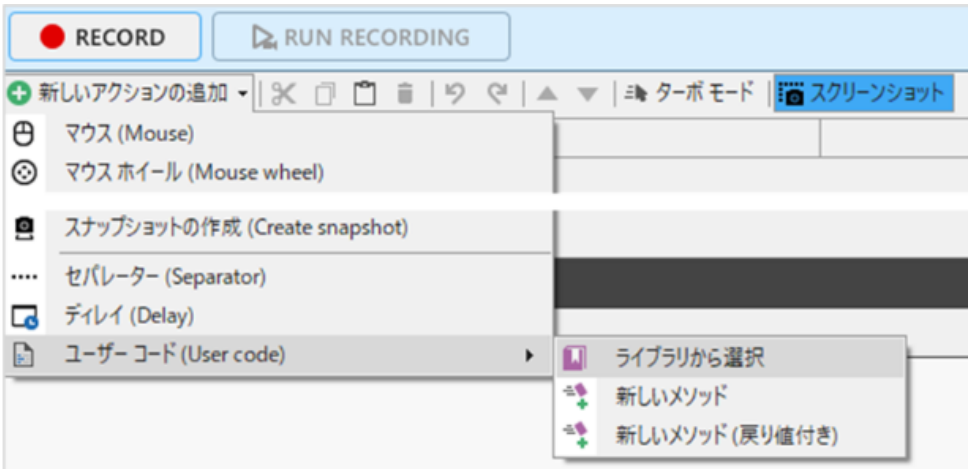


(3) 期待画像のスクリーンショットを用意します。
今回、ご紹介するユーザーコードでは、手動で取得したスクリーンショットを期待画像として設定できますので、期待画像をローカルマシンに保存ください。また、今回の画像形式は PNG にしています。

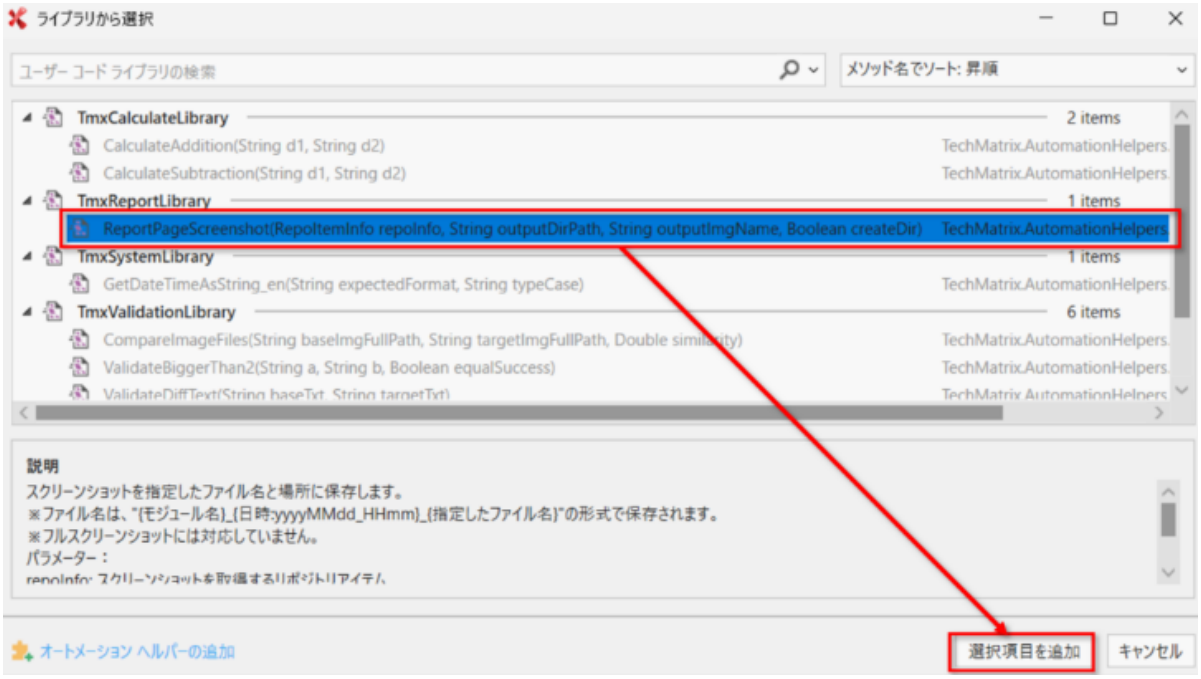
(4) TMX 提供のオートメーションヘルパーの「ReportPageScreenshot」でスクリーンショットの取得アクションを追加します。

> メモ
>
> TMX 版オートメーションヘルパーを以下のリンクからダウンロードください。
>
> - TMX 版オートメーションヘルパー
> https://rx_sp:RX969nKOGr@as3.techmatrix.co.jp/se/ranorex/support/TechMatrix%20Automation%20Helpers_v1.0.zip

画像検証を追加したいレコーディングモジュールを開き、アクションテーブルを右クリックし、「新しいアクションの追加」>「ユーザーコードアクション」>「ライブラリから選択」をクリックします。



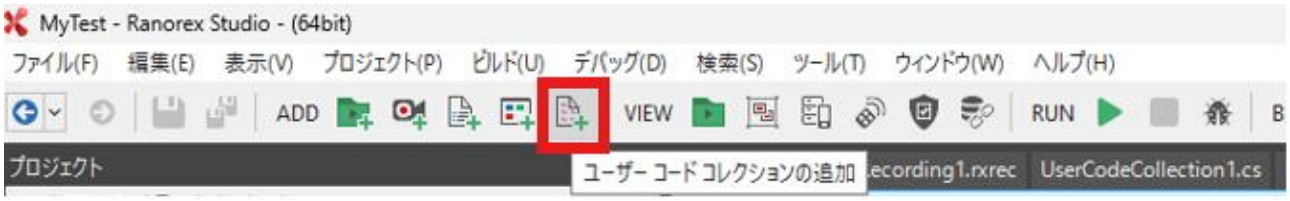
ライブラリから選択　画面にて、「TmxValidationLibrary」 >「 ReportPageScreenshot」メソッド選択し、「選択項目を追加」」ボタンをクリックします。



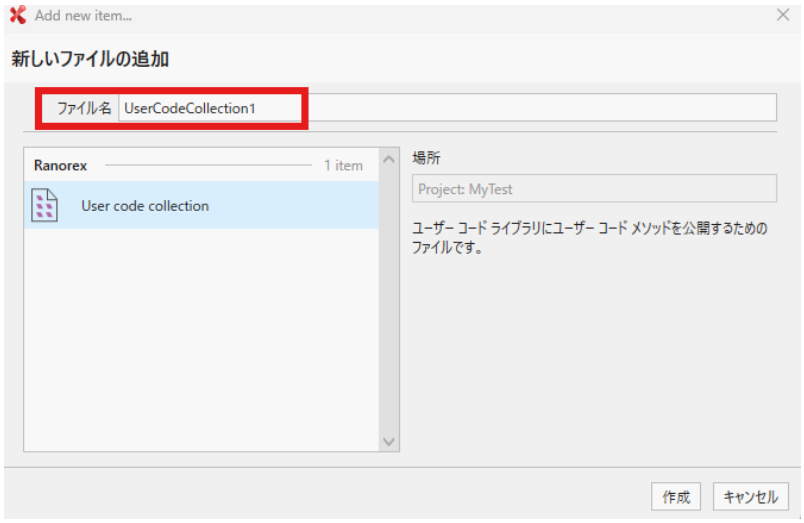
追加された User code　アクションにて、選択したメソッドのパラメーターを設定します。
※画像ファイル名は、以下の設定では「モジュール名_yyyyMMdd_HHmm_出力画像ファイル名.png」で出力されます。



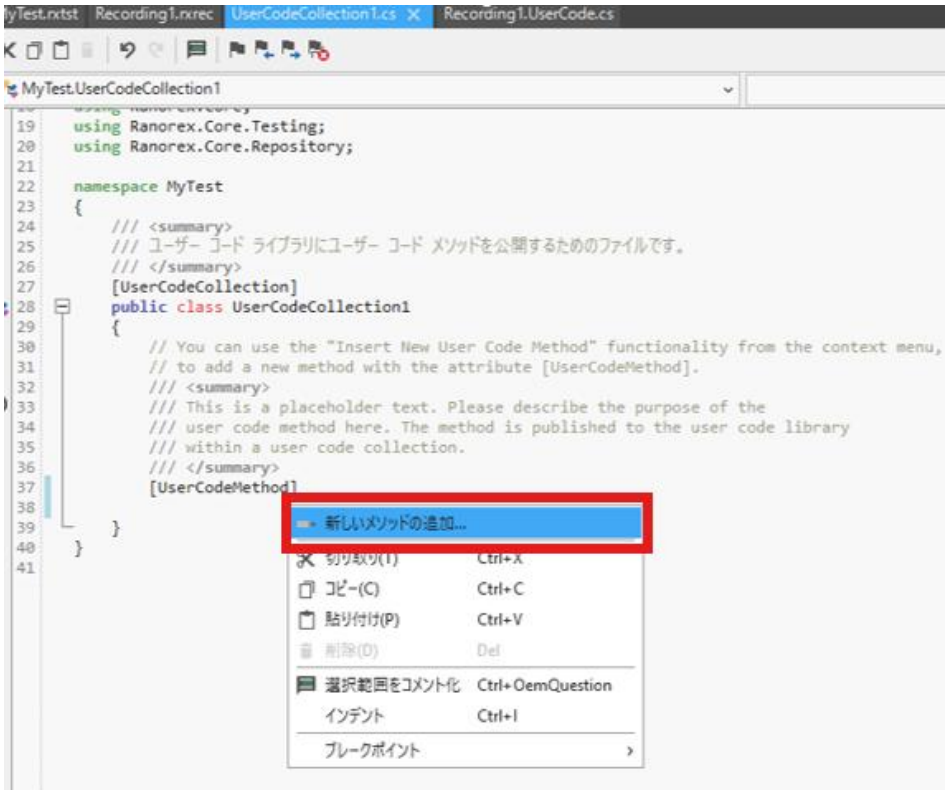
(5) OpenCV を利用して、期待画像と実行時に取得した画像を比較するコードを記述します。
ツールバーから「ユーザーコードレクションの追加」をクリックします。



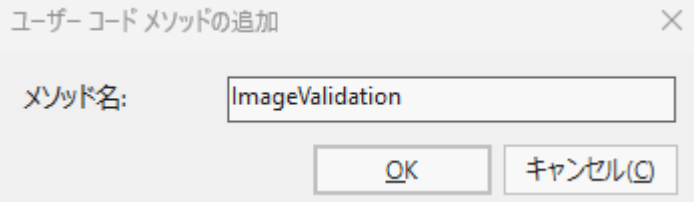
ファイル名は、任意の名前を指定ください。



作成されたユーザーコードコレクションにて、メソッドを追加します。
[UserCodeMethod]の下で右クリックし、「新しいメソッドの追加」をクリックします。



メソッド名は、任意の名前を指定ください。



作成されたユーザーコードコレクションファイルに使用するパッケージを追加する

```
using System.IO;
using OpenCvSharp;
using OpenCvSharp.Quality;
```

サンプルコードは、以下のリンクからソリューションをダウンロードしてご確認ください。

> メモ
>
> ブラウザーで以下のリンクにアクセスして、サンプルソリューションをダウンロードください。
> https://rx_hon:RXBledL5OE@as3.techmatrix.co.jp/se/ranorex/hands-on/画像検証_OpenCV 連携.zip

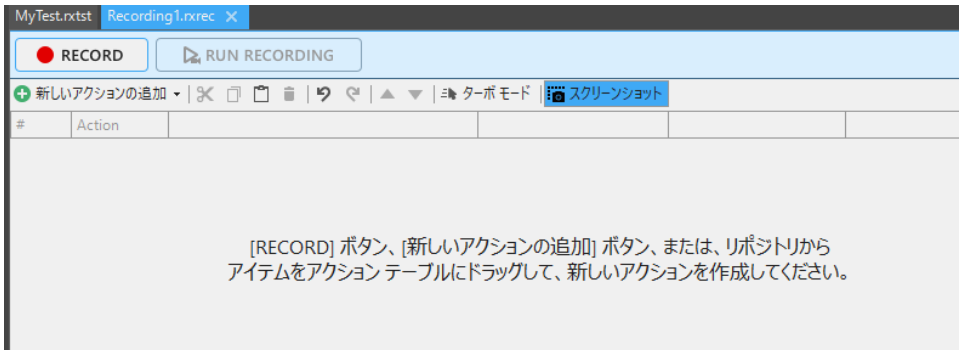
ソリューションの「UserCodeCollection1.cs」ファイルにサンプルコードがありますので、コードをコピーしてご利用ください。

ファイルパス：
画像検証_OpenCV 連携¥Documents¥Ranorex¥RanorexStudio Projects¥画像検証_OpenCV 連携¥画像検証_OpenCV 連携¥UserCodeCollection1.cs

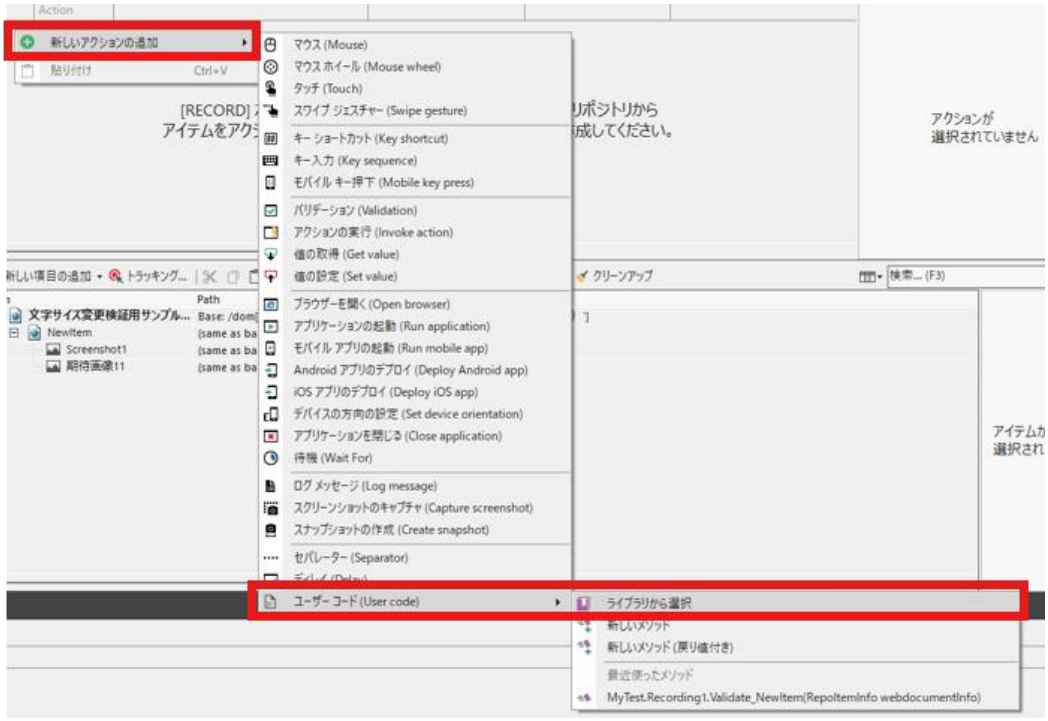
サンプルコード記述内容：

- 文字が存在する領域（最小の四角枠）を自動検出
- 文字領域をテンプレートして、文字領域の位置ズレを補正
- 文字領域のクロージング処理で文字の輪郭の微細なギザギザやノイズを滑らかに補正
- SSIM（構造類似度指数）を用いた画像比較
- 2 つの画像の異なる箇所を赤色でハイライトした差分画像を保存

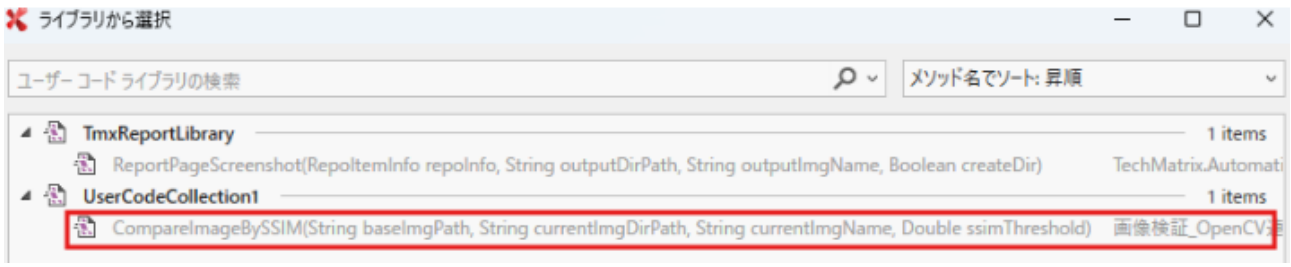
(6) 画像検証を追加したいレコーディングモジュールを開きます。



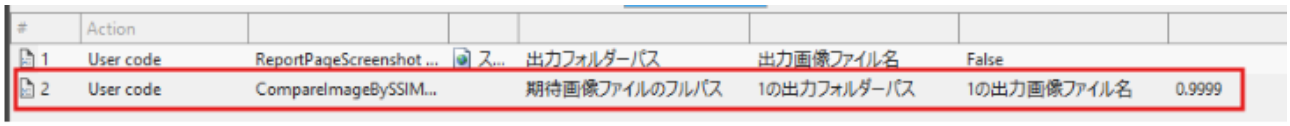
(7) アクションテーブルを右クリックし、「新しいアクションの追加」>「ユーザーコードアクション」>「ライブラリから選択」をクリックします。



(8) ライブラリから作成したメソッドを選択します。

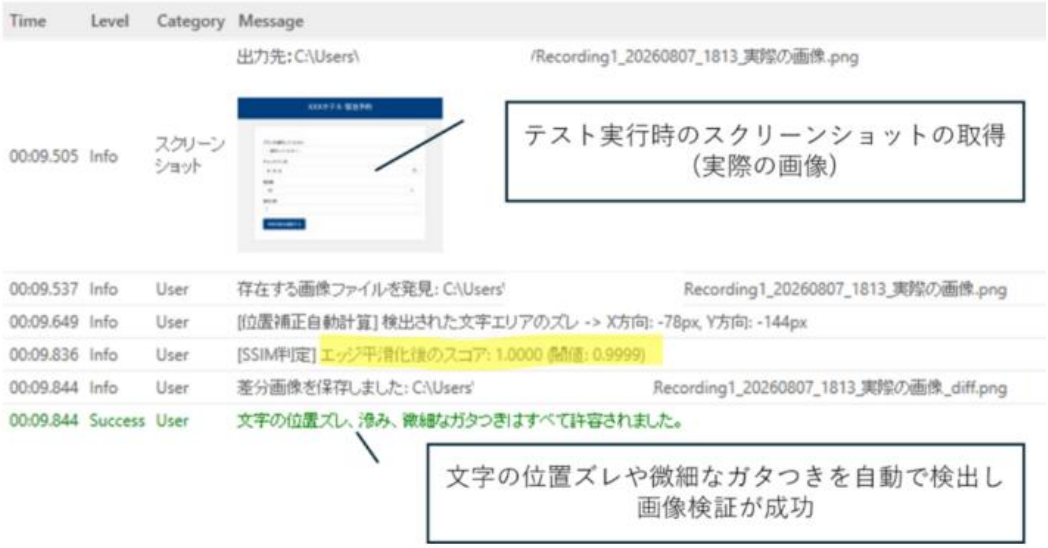


アクションを追加すると以下のような設定ができます。閾値は 1 に近い数値を設定することで、人の目では分からない微差のみを成功とできます。



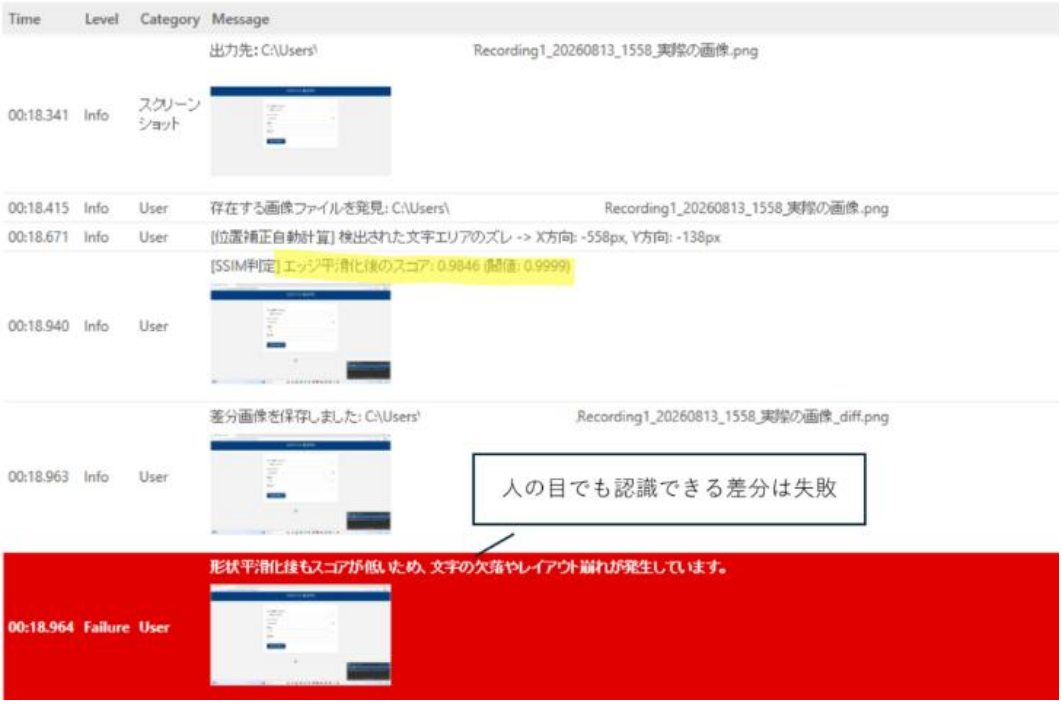
この設定でテスト実行することで、文字の位置ズレや微細なガタつきを自動検出して、期待画像と実際の画像の検証をおこなうことができます。

成功パターンのテスト結果



失敗パターンのテスト結果

人の目でも認識できる差分がある場合は、画像検証が失敗します。



差分画像は出力先(currentImgDirPath)に保存されます。
以下のように差分がある部分は赤く表示されます。



4.4 環境ごとに期待画像を切り替えて、テストを実行する

失敗要因：Windows OS バージョン・クラウド環境・リモートデスクトップ環境が異なることで微細な差分が発生する場合
効果：環境ごとの期待画像で画像比較ができるため、環境依存の影響を受けづらくなる

環境ごとに期待画像を取得する作業ができる場合、Ranorex のユーザーコード機能を使用して、環境ごとに期待画像を切り替える処理をご自身でコーディングすることで、対応できます。

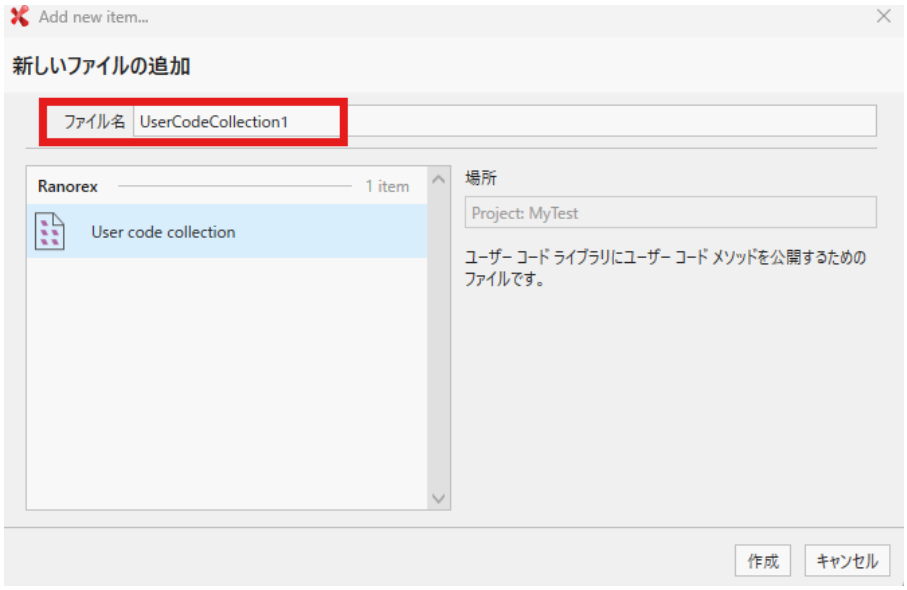
設定手順

(1) 環境ごとに期待画像を取得して、マシンのローカルに保存しておきます。環境ごとにフォルダーを用意して、各フォルダーに同名のスクリーンショットを保存ください。

(2) ユーザーコードコレクションを作成します。
ツールバーから「ユーザーコードレクションの追加」をクリックします。

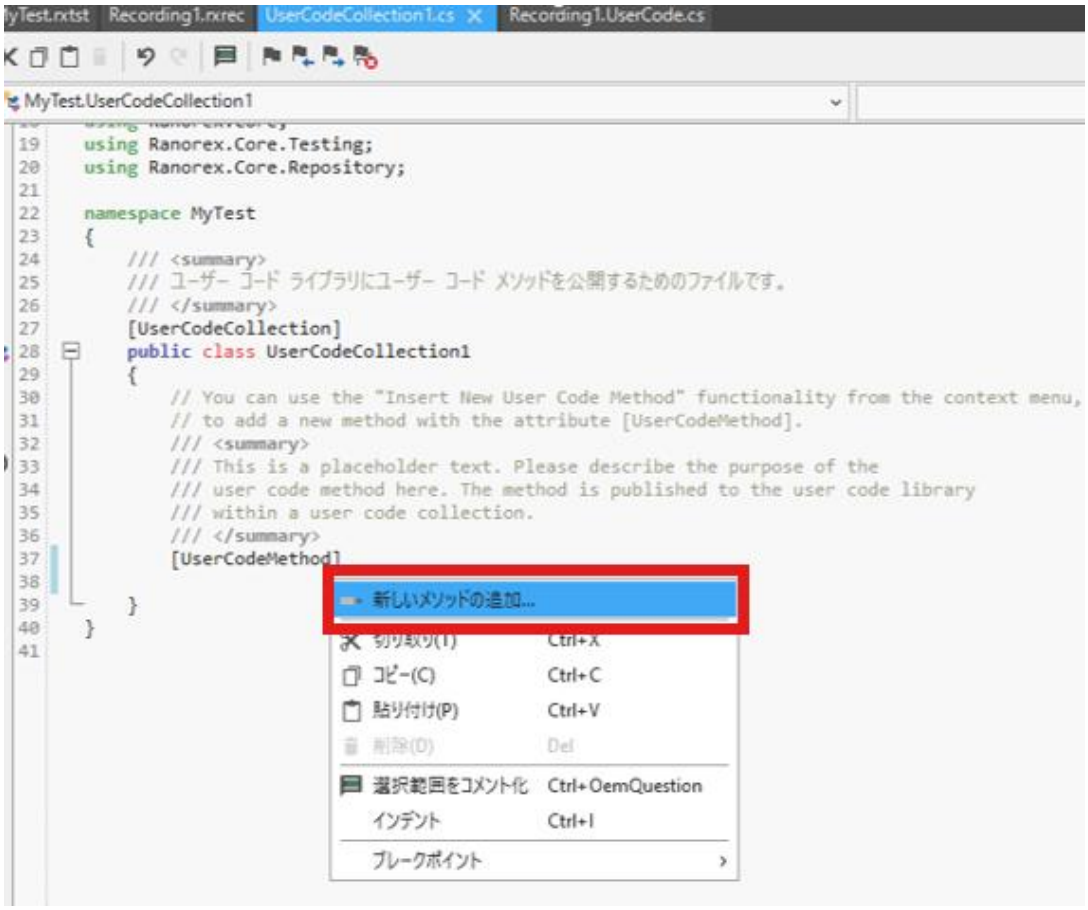


ファイル名は、任意の名前を指定ください。



(3) 作成されたユーザーコードコレクションにて、メソッドを追加します。

[UserCodeMethod]の下で右クリックし、「新しいメソッドの追加」をクリックします。



メソッド名は、任意の名前を指定ください。

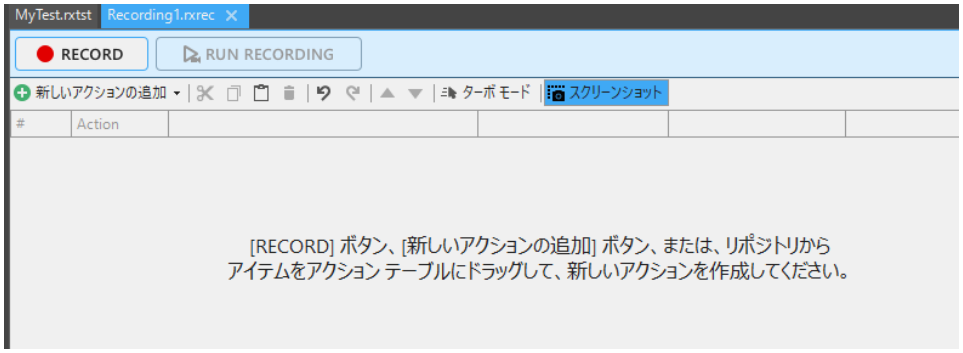


(4) 作成されたメソッドに、以下の処理を記述します。

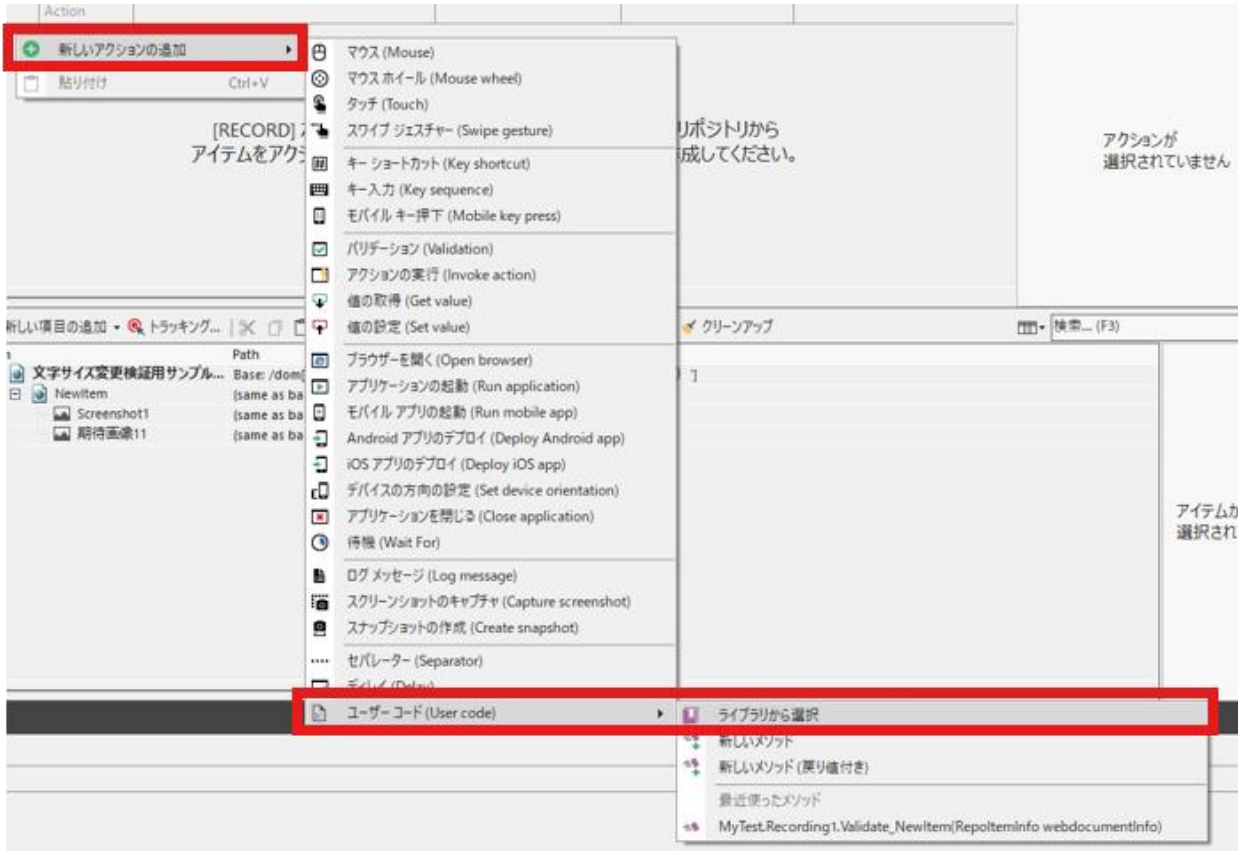
```
using Ranorex.Core.Repository; //追加

public static void ImageValidation(RepoItemInfo pictureInfo, string expectedImgPath, string
expectedImgName)
{
Imaging.FindOptions Screenshot_Options = Imaging.FindOptions.Default;
Report.Log(ReportLevel.Info, "Validation", "Validating CompareImage (Screenshot: 'Screenshot1' with
region {X=0,Y=0,Width=402,Height=263}) on item 'pictureInfo'.", pictureInfo);
// 期待値となる画像ファイルの読み込み
Bitmap expectedImg = Ranorex.Imaging.Load( expectedImgPath + "/" + expectedImgName);
Validate.ContainsImage(pictureInfo, expectedImg, Screenshot_Options);
}
```

(5) 画像検証を追加したいレコーディングモジュールを開きます。



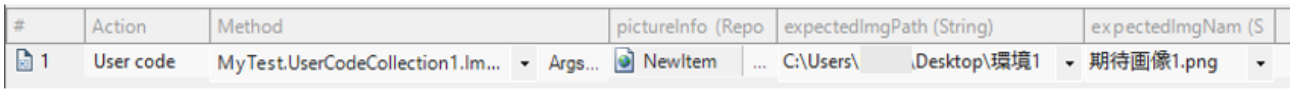
(6) アクションテーブルを右クリックし、「新しいアクションの追加」>「ユーザーコードアクション」>「ライブラリから選択」をクリックします。



(7) ライブラリから作成したメソッドを選択します。

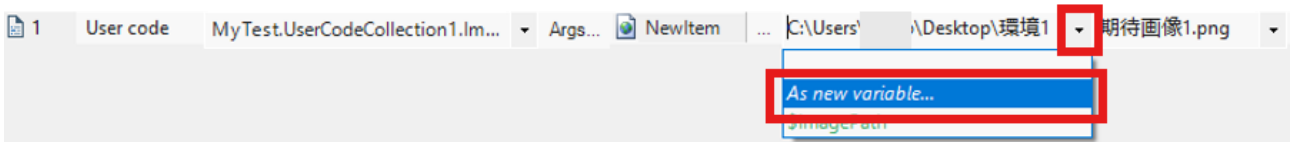


(8) 「pictureInfo」に画像検証対象のオブジェクトを設定し、「expectedImg」に期待画像のフォルダーを設定し、「expectedImgName」に期待画像のファイル名を設定します。

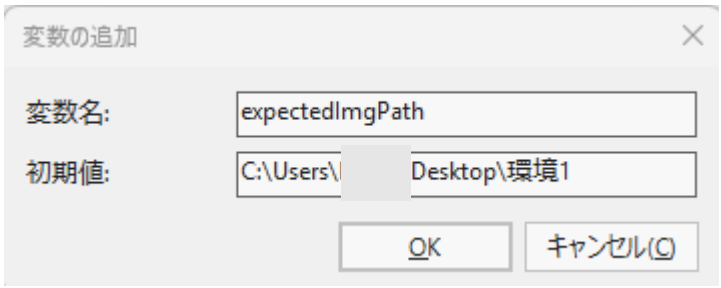


(9) 「expectedImgPath」を変数化します。

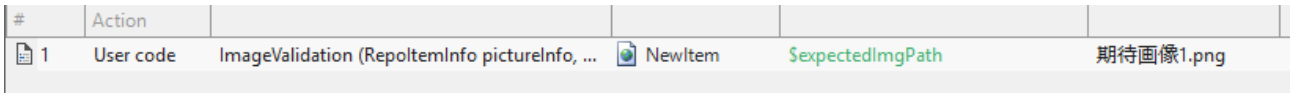
▼をクリックし、「As new variable」をクリックします。



変数名は任意の名前を指定ください。

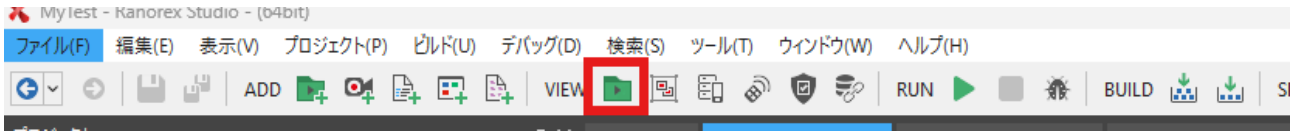


変数を設定すると、先頭に$マークが付き、緑色の文字になります。

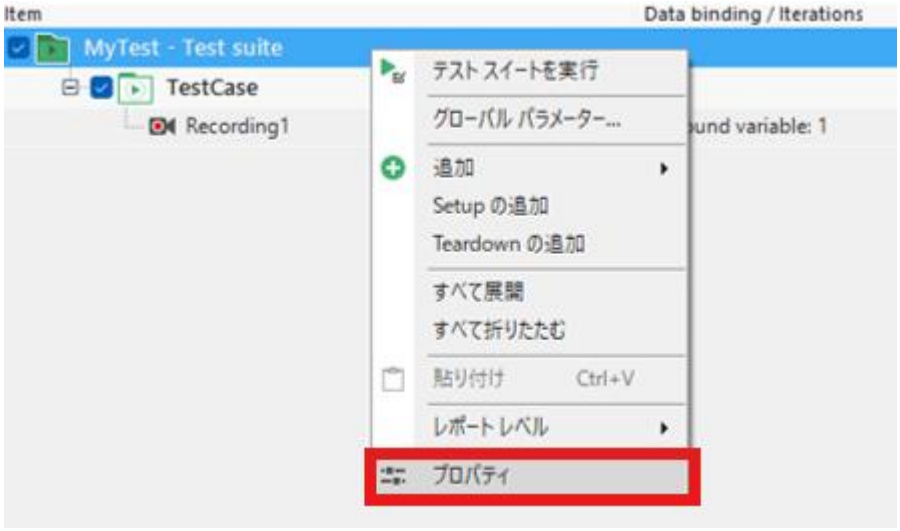


(10) グローバルパラメーターを設定します。

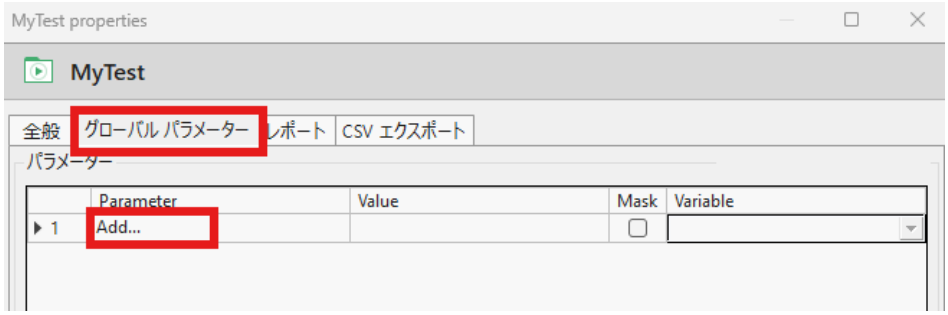
ツールバーから「テストスイートの表示」をクリックします。



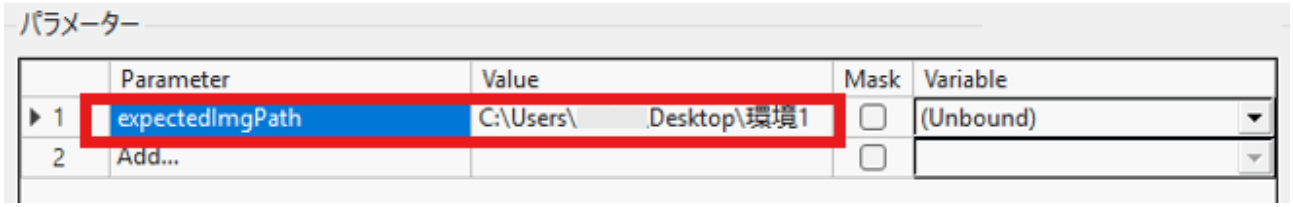
テスト構成から、最上位のテストスイートを右クリックし、「プロパティ」をクリックします。



プロパティ画面にて、「グローバルパラメーター」タブをクリックし、「Add...」をクリックします。



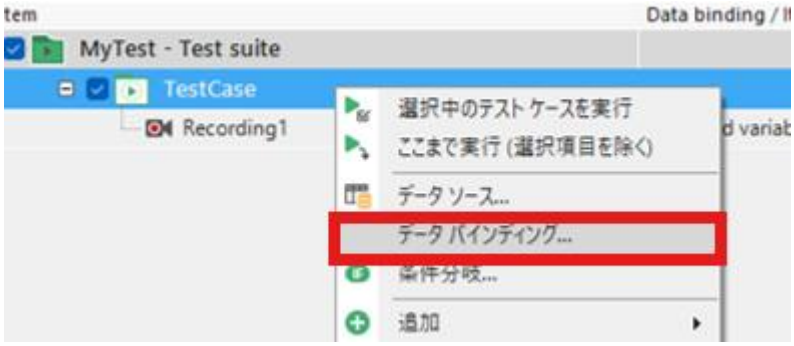
「Parameter」にグローバルパラメーター名を設定し、「Value」に期待画像が含まれるフォルダーパスを設定します。



(11)グローバルパラメーターと手順 8 で作成した変数を連携します。

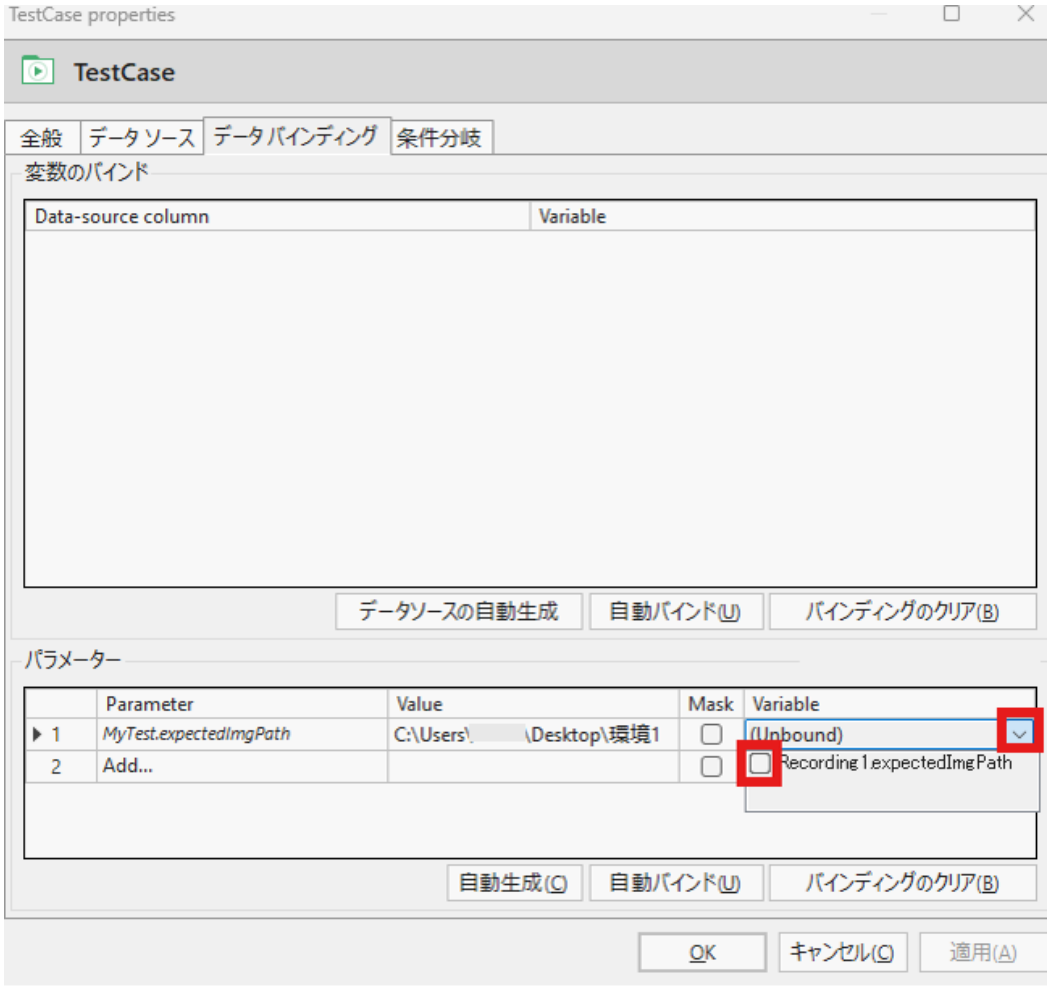
変数を設定したレコーディングモジュールを格納しているテストケースを右クリックし、「データバインディング」をクリックします。

※Recording1 で画像検証のメソッドが使われており、変数が設定されています。



プロパティ画面の下部にある「パラメーター」にて、作成したグローバルパラメーターに変数を連携します。

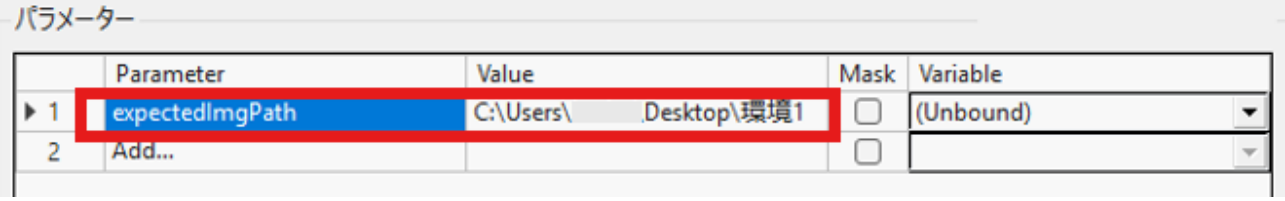
「Variable」項目の「(Unbound)」をクリックし、連携させる変数(レコーディングモジュール名.変数名)にチェックを入れます。



(12)この設定でテスト実行することで、環境ごとに用意したフォルダー内の期待画像と実際の画像の検証をおこなうことができます。



テスト実行環境が変わる場合、グローバルパラメーターのフォルダーパスを変更することで、簡単に期待画像を入れ替えることができます。※環境ごとにフォルダーを分けて期待画像を管理しており、スクリーンショット名が同じ場合にこの方法が使えます。



メモ

環境ごとに期待画像を入れ替える方法については、以下のブログ記事にも記載していますので、ご確認ください。

- ユーザー コード リファレンス：クロスプラットフォームにおけるイメージ比較
  https://ranorex.techmatrix.jp/blog/2020/10/16/user-code-reference_image_validation/

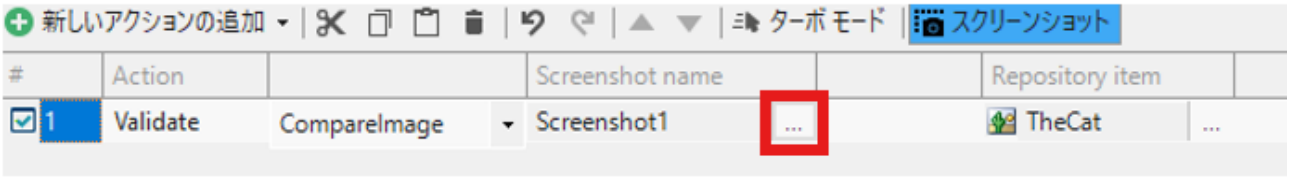
4.5.差分が発生しやすい箇所や動的に変わる箇所をマスキングする

失敗要因：テスト環境により動的に変化するコンテンツが画面に含まれる場合
効果：動的な箇所による画像検証の失敗を排除できる

動的なコンテンツが画像内に含まれている場合、画像検証は失敗しやすくなります。こうした動的な値や画像の一部にマスクをかけて検証することで、動的な部分の影響を排除して検証することが可能になります。この設定は、Validate アクションの CompareImage で対応しています。

設定手順

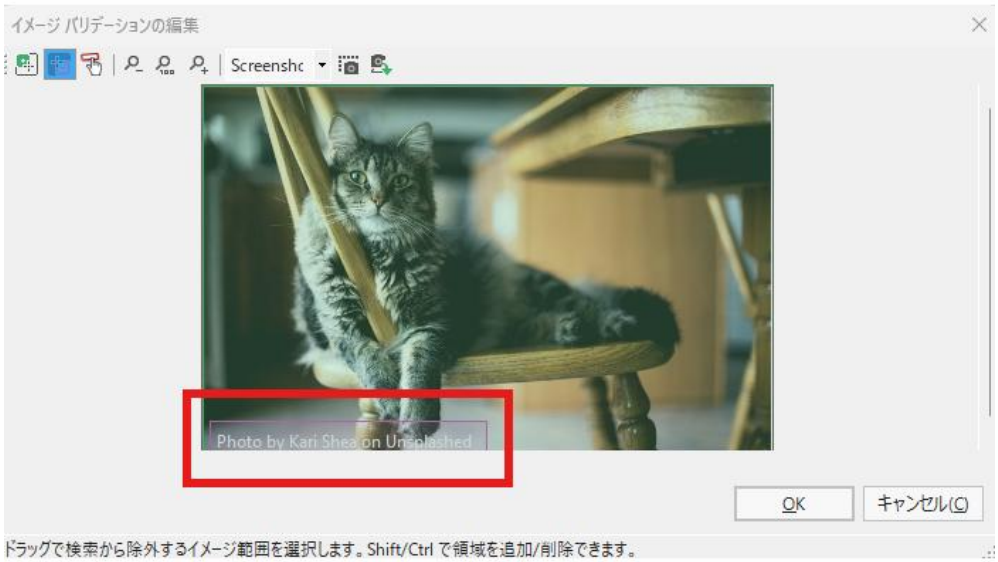
(1) 画像検証の Validate アクションをクリックし、「Screenshot name」の「...」ボタンをクリックします。



(2) イメージエディターのツールバーから「イメージ除外領域の選択」をクリックします。



(3) 除外したい領域をドラッグ&ドロップで範囲選択します。紫の箇所が除外されます。
複数範囲選択：「Shift」キーを押下しながらドラッグ&ドロップ、範囲選択解除：「Ctrl」キーを押下しながら削除したい領域をクリック



(4) この設定で、動的な箇所は除外されて画像検証されるため、除外領域以外に差分がない場合は、テストが成功します。

Time	Level	Category	Message
00:07.603	Info	Validation	Validating CompareImage (Screenshot: 'Screenshot1' with region {X=0,Y=0,Width=402,Height=263}) on item 'RxMainFrame.TheCat'.
00:10.147	Success	Validation	Screenshot of item 'MyTestRepository.RxMainFrame.TheCat' does match the specified image.

注意

Validate アクションの ContainsImage で除外領域を選択領域内に設定すると「選択領域のオーバーラップ」の警告が表示されます。



ContaimsImage で設定する「イメージ除外領域」は、画面内で期待する画像が存在してほしくない箇所を指定するためのものであるため、この領域は、必ずイメージ領域の外側に設定する必要があります。そのため、イメージ除外領域はイメージ領域と重なってはいけません。イメージ領域の内部で除外したい部分がある場合には、Validate アクションの CompareImage をご利用ください。



テクマトリックス株式会社　ソフトウェアエンジニアリング事業部
Ranorex テクニカルサポートセンター

E-MAIL ranorex-support@techmatrix.co.jp