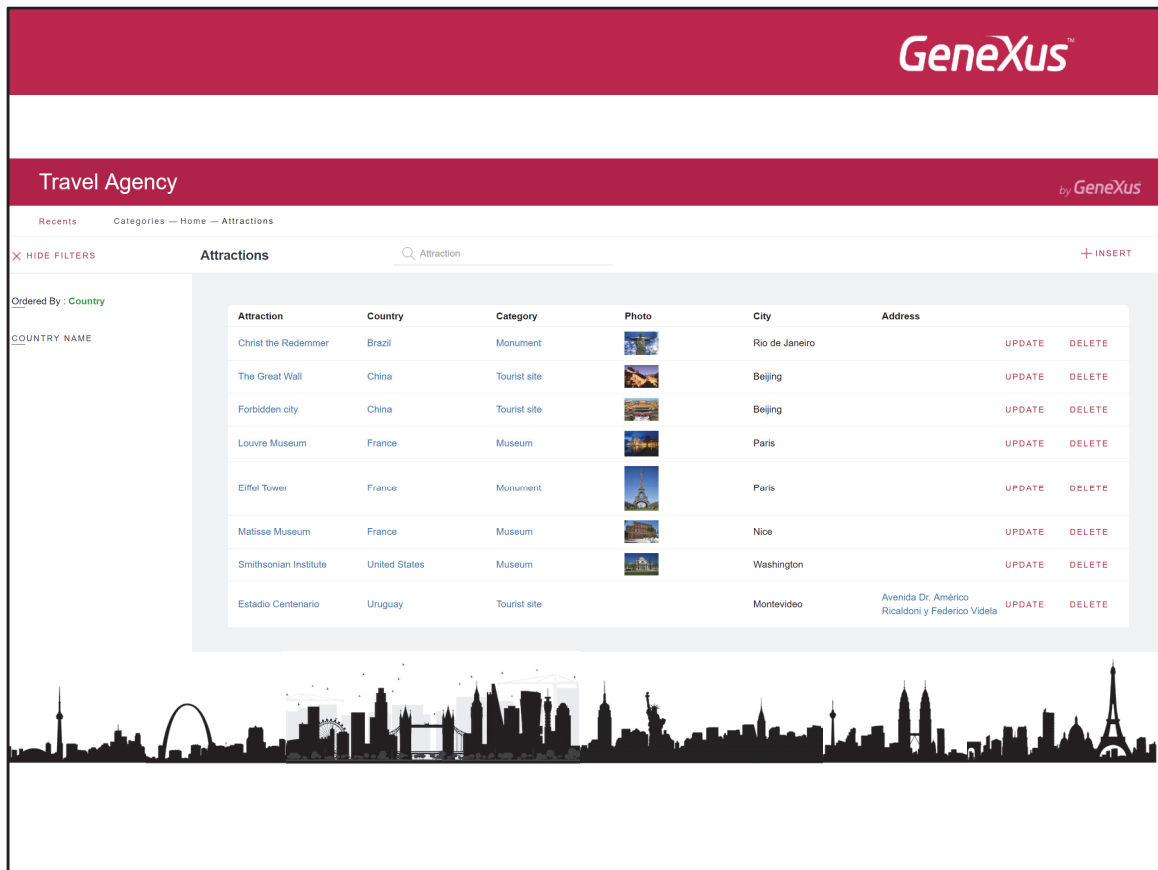


GeneXus での アプリケーションのテスト

ユーザー インターフェース テスト (UI テスト) と
GXtest の概要

GeneXus™



これまで、旅行代理店向けのアプリケーションを拡大するなかで、さまざまな機能を追加したり、前に実装したものに変更を加えたりしてきました。

しかし、アプリケーションを再度テストして、変更前に機能していたものが引き続き正常に動作することを確認するという点については、これまで触れてきませんでした。

このような作業は、大規模なアプリケーションでは非常に手のかかるものとなり、大半は既にテストしたものを繰り返しテストすることになります。これは必要なことではありますが、非常に退屈な作業です。

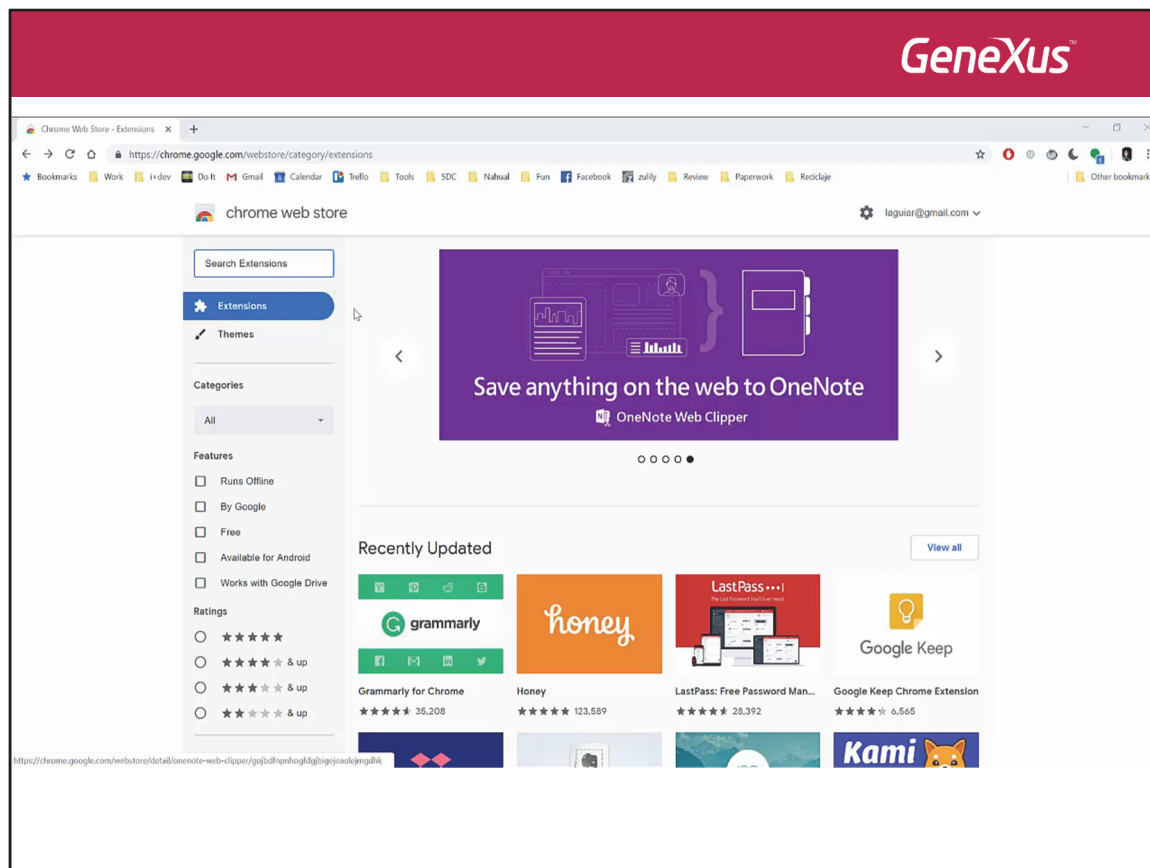


GeneXus は、GXtest というツールでそのようなテストの自動化を支援します。

GXtest を使用してテストを定義すると、後で自動的に再現できるようになります。システムが適切に機能しているかどうかについて、人間がデータを入力して検証しているかのようなテストが可能です。

これを例を使って説明します。

旅行代理店にとって、観光名所の管理、つまり、観光名所の登録、変更、削除が重要な作業であるとして。その場合、観光名所を管理するプロセスでエラーが生じると、ビジネスに大きな影響があります。そこで、このプロセスをプログラミングして適切に機能していることを検証できたら、その検証をキャプチャして、将来はリグレッションテストの形態の 1 つとして自動的に実施できるようにします。



[デモ: <https://youtu.be/K2ZeWgYyjdk>]

テストを自動化するために必要なものとして、まず GXtest Recorder があります。このツールを使うと、手動での実行をキャプチャして、GeneXus に取り込むことができます。GXtest Recorder は Google Chrome の拡張機能であり、Chrome ウェブストアから無料でダウンロードできます。

ほかの拡張機能と同様に、この拡張機能をインストールして、Chrome で有効にします。拡張機能をインストールしたら、Chrome のツールバーから実行できます。GXtest Recorder を開き、カメラボタンを押して録画を開始します。

GXtest Recorder をアクティブな状態にして、アプリケーションを通常どおりに、手動で検証を行うときと同じように実行します。たとえば、まず、[追加] ([INSERT]) ボタンをクリックして新しい観光名所を追加します。画面の右下隅のポップアップで、実行したアクションが GXtest Recorder によってキャプチャされたことを示すメッセージが表示されます。アクションを実行するたびにこのようなメッセージが表示されます。

新しい観光名所を作成し、既存の国にある既存の観光名所の名前を使います。この場合、検証によってエラーが表示されることが期待されます。この検証をキャプチャするよう指示します。テキストが表示されることを期待するので、AssertText を作成します。

このアサーションを作成するということは、このエラーメッセージが表示されるのを期待することです。つまり、ウルグアイにセンテナリオ競技場という観光名所を作成したときにエラーメッセージが表示されなかったら、テストは失敗ということになります。

テストを続けて、新たに別の名前を入力すれば観光名所を登録できることを示します。新しい観光名所の値を入力し、[実行] ([CONFIRM]) をクリックします。

GXtest Recorder には、実行した個々のアクションが表示されます。クリックした場所、値を入力した場所なども記録されます。これがテストのスクリプトであり、後ほど自動的に実行されます。

次のテストとして、追加した観光名所を検索し、その値が表示されることを期待す

るアサーションを作成します。

テストを完全なサイクルにするために、変更を加えます。エントリーを登録して変更し、削除します。名前を「Estadio-Centenario-3」に変えて [実行] をクリックします。「Estadio Centenario 3」を検索して、この値が表示されることを期待するアサーションを作成します。すべてが問題なく動作し、編集できていれば、「Estadio Centenario 3」という名前がここに表示されるはずです。最後に削除を行い、確認のメッセージの表示に対するアサーションを作成します。[実行] をクリックすると、完全なサイクルのテストが完了します。

--

テストが完了したら、記録をオフにしてキャプチャを完了します。テストのスク립トが記述されます。スク립トを保存すると、何度でも実行することができます。

[再生] ボタンをクリックして、テストが期待どおりに再現されることを確認します。スク립トを実行すると、成功した手順 (コントロールが期待どおりにクリックされた、期待どおりの値が入力された、アサーションで指定したことが実際に発生したなど) が緑で表示されます。

GXtest Recorder では、テストを記録して実行するほかに、スク립トを編集したり、不要な手順を削除したりできます。テストの実行にあたり、不要なクリックが入っている場合は、その手順を削除できます。また、検証を追加することもできます。テストを検証して期待どおりのものになったら、テストを GeneXus に取り込みます。

GXtest Recorder → GeneXus

テストの自動統合が行われていなくても、簡単にナレッジベースに取り込むことができます。[ダウンロード] ([DOWNLOAD]) ボタンをクリックしてエクスポートすると、スク립トがテキストファイルで生成されます。この内容は GeneXus コードです。

そのコードをコピーし、GeneXus で特別なオブジェクトに貼り付けます。

ナレッジベースに移動し、UI Test タイプの新しいオブジェクトを「Test_Attractions」という名前で作成します。そこに GXtest Recorder でキャプチャしたコードを貼り付けます。

UI Test

UI Test オブジェクトは、GeneXus の今回のバージョンで追加された新しいオブジェクトです。使用するには GXtest ライセンスが必要です。ナレッジベース内にテストがあると、ナレッジベースの一部として、テストのバージョン管理、共有、および拡張が可能になります。

UI Test オブジェクトは、GXtest の機能を使い、さまざまなコマンドを使ってブラウザを制御できるようにします。たとえば、ブラウザで特定の URL を開いたり、各種のコントロールを選択またはアクティベートしたり、画面上でのアサーションによる検証を行ったりすることができます。

GXtest Recorder は、コードを自動的に記述することで作業を容易にします。GeneXus では、テストの拡張が可能です。また、テスト用のデータをデータプロバイダーから取得したり、データベースに対する検証を追加したりできます (たとえば、画面上でレコードを削除した後、そのレコードがデータベースに存在しないことを検証するとします。テストにそのアサーションが含まれていない場合、アサーションを追加して、そのレコードがデータベースに存在しないことを検証できます)。

スク립トは、ユニットテストと同じ方法で実行できます。右クリックして [このテストを実行] を選択するか、[テストエクスプローラー] で対象となるインターフェースのテストを選択します。

テストを実行するブラウザーを変更することもできます。既定のブラウザーである Chrome の代わりに Firefox を指定して、Firefox でテストを実行することもできます。



一連のテストを作成し、機能することをローカルで確認したら、テストを GeneXus IDE ではなく別の場所で行います。そうすれば、テストの対象とするすべてのブラウザとオペレーティングシステムを自分のコンピューターにインストールする必要はなくなります。

Sauce Labs などのツールを使って、テストをクラウドで実行するように GXtest に指示することもできます。Sauce Labs などのクラウド環境では、さまざまなオペレーティングシステムやブラウザで機能テストを実行できます。

詳しくは、次の URL で Wiki を参照してください。

<http://wiki.genexus.jp/hwikibypageid.aspx?41131>



継続的インテグレーション

これまで見てきたようなインターフェーステストは、ビジネスで失敗が許されない部分の機能サイクル全体を検証するために、広く使われています。

これらのテストによって、ユーザーがガイドラインに従い、なにか奇妙なことをしない限りは異常が起きることはなく、ビジネスにとって重要な機能を使い続けることができ、フローが円滑に進んでエラーが発生しないことが分かります。

このような最も重要なフローから自動化することで、アプリケーションに変更を加えても重要なフローは必ず機能するようにします。

これまで、どのようにテストが自動化されるかを見てきました。しかし、その実行は手動で行っていました。

これらのテストは、自動的に実行する必要があります。



これらのテストの実行は、Jenkins による継続的インテグレーションのための自動化されたプロセスの一部となります。

詳細はこのコースでは扱いませんが、簡単に触れておくと、Jenkins は継続的インテグレーションのエンジンであり、アプリケーションのビルドのセットアップに関わるさまざまな手順を自動的に実行できるようにします。

GeneXus™

Jenkins

2

search

jenkins admin

log out

Jenkins > Pipeline View >

Build Pipeline: KB Update, Build and Test.

This is a pipeline view to show jobs dependencies

Run

History

Configure

Add Step

Delete

Manage

Update KB (on demand) → Build KB → Run Unit Tests → Run UI Automation

Pipeline #7

#7 Update KB (on demand)
Oct 18, 2018 12:25:18 AM
10 sec
admin

#9 Build KB
Oct 18, 2018 12:25:34 AM
1 min 35 sec

#14 Run Unit Tests
Oct 18, 2018 12:27:19 AM
1 min 7 sec

#3 Run UI Automation
Oct 18, 2018 12:28:34 AM
2 min 25 sec

<http://wiki.genexus.jp/hwikibypageid.aspx?40737>

Jenkins におけるパイプラインの一般的な例を説明します。これは、アプリケーションの新しいビルドをセットアップするために実行する必要がある一連の手順です。

1. 最初に、アプリケーションをセットアップするナレッジベースを更新します。前回の更新以降に変更されたすべてのオブジェクトを GeneXus Server から取得します。
2. 次に、GeneXus でビルドを実行します。GeneXus Server からダウンロードした、変更があったオブジェクトを再編成し、仕様解析して、コンパイルします。
3. 3 番目の手順でユニットテストを実行します。
4. そこで問題がなければ、最後に UI テストを実行します。

その後にアプリケーションのビルドを本番環境にリリースできるか、あるいはテストターに送って新機能を手動でテストするかは、テストカバレッジによって決まります。

テストターの出番が訪れた時点で、少なくともすべてのリグレッションテストと、新機能のユニットテストが完了していることは分かっています。残りは統合テスト、つまりリリースされる新機能の完全なユーザーサイクルです。

継続的インテグレーションとテストの自動的な実行の詳細については、Wiki を参照してください: <http://wiki.genexus.jp/hwikibypageid.aspx?40737>



<https://www.genexus.jp/gxtest-v4-jp>

GXtest は、自動的なテストを簡単に生成し、実行できるため、アプリケーションの信頼性を確保しながらテストの時間を短縮するのに役立ちます。

また、継続的インテグレーションのプロセスに必要な自動化の基本的なツールの 1 つでもあります。GXtest を活用することで、ソフトウェアのアジャイル開発を行い、品質を維持できます。

このツールの詳細については、スライドに表示される URL を参照してください。