MICRO FOCUS

Silk Performer 18.5

ブラウザ駆動型 Web 負荷テスト チュートリアル

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK
http://www.microfocus.com

2017-10-17

目次

ブラウザ駆動型 Web 負荷テスト チュートリアル

このチュートリアルでは、Silk Performer を使用して、Web 2.0 アプリケーション (特に、AJAX 技術に依存するアプリケーション) の負荷テストを実行し、できる限り迅速に稼働させるために、その手順を説明します。本書は、ユーザーが、Silk Performer の操作を理解し、e ビジネスの負荷テストに最適な本ツールの最先端機能を使いこなせるよう、支援します。

ブラウザ駆動型負荷テストの概要

Silk Performer を使用すると、今日の最新の Web アプリケーションのテストをプロトコル レベル (HTTP) で簡単に行えるだけでなく、実際の Web ブラウザ (Internet Explorer) を使用して負荷を生成することができます。 これにより、Web アプリケーションに組み込まれた AJAX ロジックを利用して、複雑な AJAX 動作をテスト中に正確にシミュレートすることができます。 この強力なテスト方法では、レンダリング時間やプロトコル レベルの統計など、エンド ユーザーによる実際のブラウザ動作を反映する結果が提供されます。

特定の AJAX フレームワークのみ (およびコントロールの特定のバージョンまたはサブセットのみ) をサポートする他の負荷テストソリューションと異なり、Silk Performer では、Internet Explorer 用に開発され Internet Explorer でテストされた広範囲にわたる Web アプリケーションがサポートされています。

Internet Explorer には互換性の問題が生じる場合があることに注意してください (Internet Explorer (IE) 9.0 がインストールされている場合、IE7 標準モード (7000) 用の設定だけが正しく機能します。レジストリ設定は、正しく書き込まれますが、Silk Performer 内の IE コントロールによって使用されることはありません。 この問題に対する回避策はありません。ただし、Internet Explorer 8 がインストールされている場合、IE8 と IE7 モードを使用できます。)。

ポップアップ ウィンドウのサポート

Silk Performer のブラウザ駆動型テストでは、ポップアップ ウィンドウ (ログイン ダイアログ ボックスなど) を利用するサイトがサポートされています。 ポップアップ ブラウザ ウィンドウには、メイン ページに戻される値をユーザーが入力する入力フィールドが含まれることがよくあります (ユーザー名やパスワード文字列など)。 複数ブラウザ ウィンドウのサポートは、Web ブラウザ駆動型 (AJAX) タイプの Silk Performer プロジェクトを作成するときにデフォルトで使用できます。

アプリケーションの記録中にポップアップ ウィンドウが生成されるたびに、新しいタブが Browser Application に作成されます。 発生した各ポップアップ ウィンドウが、Browser Application に作成されるタブになります。 記録中に Browser Application のタブをクリックするたびに、BrowserActivateWindow 関数が自動的にスクリプト化されます。

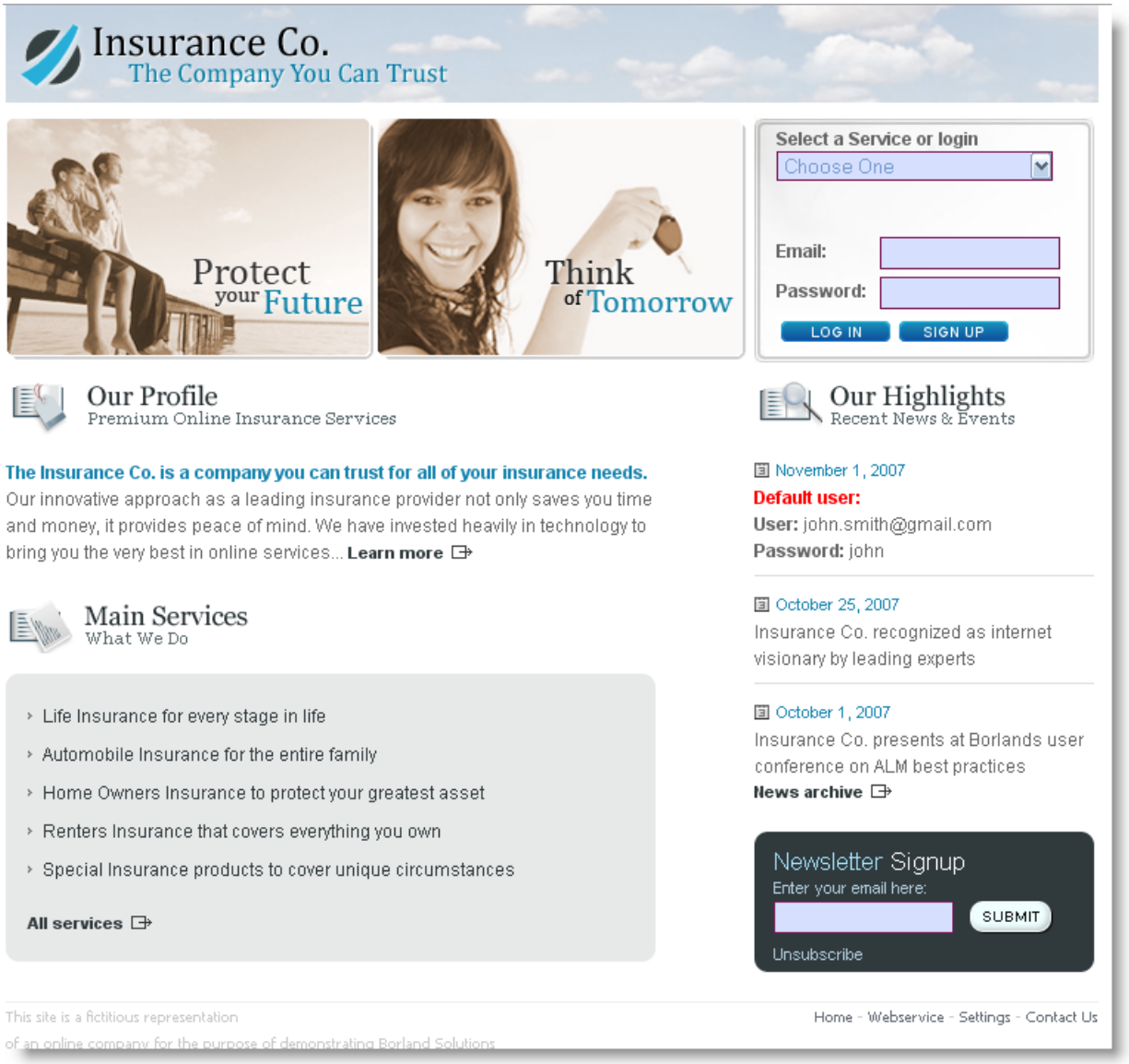
注: 1 つのユーザー アクションによって複数のブラウザ ウィンドウが生成された場合は、最後に生成されたウィンドウが Browser Application によって認識されます。 つまり、最後に作成されたウィンドウによって、BrowserGetActiveWindow 関数がスクリプト化されます。 それ以前に作成されたウィンドウはどれも、Browser Application ではアクセスされません。

注: 記録中、手動で (メニュー バー、コンテキスト メニュー、またはキーボードのショートカット経由で) ウィンドウまたはタブを開くことはできません。

サンプル Web 2.0 アプリケーション

Silk Performer には、Web 2.0 アプリケーションのテストを習得するために使用できる最新のサンプル Web アプリケーションが備えられています。 InsuranceWeb サンプル Web アプリケーションは、ExtJS および JSF フレームワークに基づいて作成され、AJAX 技術を採用し、JSON と XML を介して通信します。

サンプル アプリケーションは、*http://demo.borland.com/InsuranceWebExtJS/* でホストされます。



サンプル アプリケーションのポップアップ ウィンドウ

サンプルの Web 2.0 アプリケーションには、複数のブラウザ ウィンドウの Silk Performer サポートを試すために使用できるポップアップ ウィンドウ機能が含まれています。

1. ポップアップ ウィンドウを生成するには、サンプルの Web 2.0 アプリケーション *http://demo.borland.com/InsuranceWebExtJS/* にアクセスします。
2. **Select a Service or Log in** ドロップリストから **Agent Lookup** を選択します。
3. **Find an Insurance Co. Agent** ページの下部にある **Open in new window** リンクをクリックします。 **Find an Insurance Co. Agent** ページが、Browser Application の新しいタブに表示されます。

ページ下部にある **Close Window** リンクをクリックして、タブを閉じます。

HTML ダイアログ ボックスのサポート

Silk Performer では次のウィンドウ タイプが認識されます。

- 標準的なブラウザ ウィンドウ/タブ (完全サポート)
- JavaScript ダイアログ ボックス (サポート済み。**プリンタ** を除く)
- ダウンロード ダイアログ ボックス (完全サポート)
- モーダル ウィンドウ、モードレス ウィンドウ (HTML ダイアログ ボックス) (完全サポート)ウィンドウとして表示する HTML ダイアログ ボックス
- ドキュメントのレンダリングのための埋め込み Active-X コントロールのあるウィンドウ (未対応)

ネイティブ再生

ネイティブ再生 の概念により、スクリプトの再生の信頼性を高めることができます。これは、頻繁に使用される関数に対して JavaScript イベントの代わりに Windows API レベルのイベントを使用して実現します。

ネイティブ再生のコンテキストでは、2 つの特別な用語、ネイティブ再生 と 従来の入力モード を使用します。ネイティブ再生は、従来の入力モードの対極にあります。従来の入力モードをオンにすると、ネイティブ再生は自動的にオフになります。逆も同様です。この観点から、ネイティブ関数 と 従来の関数 を分類することもできます。

- ネイティブ関数は、Windows API レベルのイベントを使用します。
- 従来の関数は、JavaScript イベントを使用します。

次の関数は従来の関数の呼び出しで、それにはすべて同等なネイティブ関数があります。これらの従来の関数の 1 つが再生時に検出されると、

- BrowserClick は BrowserNativeClick のように再生されます
- BrowserDoubleClick は BrowserNativeDoubleClick のように再生されます
- BrowserSetText は BrowserTypeKeys のように再生されます
- BrowserSetPassword は BrowserTypeKeys のように再生されます
- BrowserMouseMove は BrowserNativeMouseMove のように再生されます

従来の関数の代わりにネイティブ関数を実行することができない場合には、従来の関数が使用され、警告メッセージがログに記録されます。たとえば、要素をクリックするマウスの位置が決定できない場合などがこれに該当します。

ネイティブ再生は、デフォルトで有効になっています。プロファイル設定でネイティブ再生を無効にすることができます。**設定** > **アクティブ プロファイル** > **再生 - Web (ブラウザ駆動)** をクリックします。**全般** タブで、**従来の入力モード** をオンにします。または、スクリプトに BrowserSetOption(BROWSER_OPT_LEGACY_INPUT_MODE, true) 関数を挿入して無効にすることもできます。

従来の入力モードは、Silk Performer バージョン 9.0 以前を使用して作成されたすべてのプロジェクト プロファイルに対してデフォルトでオンになっています。

Web ブラウザの構成設定

安定したテスト実行を維持するには、いくつかのブラウザ設定が重要です。設定を変更しなくても Silk Performer は動作しますが、Internet Explorer のブラウザ設定を変更するにはいくつかの理由があります。

- 再生速度を向上させる

    - ロード速度の遅い Web ページではなく、about:blank をホーム ページとして使用する
- 予期しないブラウザの動作を回避する
    - ポップアップ ウィンドウおよび警告ダイアログ ボックスを無効にする
    - オートコンプリート機能を無効にする
    - パスワード ウィザードを無効にする
    - Silk Performer を Windows Server オペレーティング システムで実行している場合、Internet Explorer Enhanced Security Configuration (IE ESC) を無効にします。
- ブラウザの誤動作を回避する
    - サードパーティのプラグインを無効にする

次の表に、Internet Explorer GUI 内でこれらの設定を確認できる場所を示します。

注: ブラウザ設定は、**ツール** > **インターネット オプション** にあります。

タブ名	オプション	構成	コメント
全般	ホーム ページ	about:blank に設定する	新しいタブの開始時間を最小限に抑えます。
全般	タブ	• 複数のタブを閉じるときの警告を無効にする • 新しいタブの作成時における新しいタブへの移動を有効にする	• 予期しないダイアログ ボックスの表示を回避します。 • このようにしないと、新しいタブを開くリンクが正しく再生されない可能性があります。
プライバシー	ポップアップ ブロック	ポップアップ ブロックを無効にする	Web サイトを新しいウィンドウで確実に開くことができます。
コンテンツ	オートコンプリート	オフにする	• 予期しないダイアログ ボックスの表示を回避します。 • 入力中に予期しないデータ入力を回避します。
プログラム	アドオンの管理	必要なアドオンのみを有効にする	• サードパーティのアドオンには不具合が含まれることがあります。 • サードパーティのアドオンに互換性のない場合があります。
詳細設定	設定	• **Internet Explorer の更新について自動的に確認する** を無効にする • **スクリプトのデバッグを使用しない (Internet Explorer)** を有効にする • **スクリプトのデバッグを使用しない (その他)** を有効にする • **スクリプト エラーごとに通知を表示する** を無効にする	予期しないダイアログ ボックスの表示を回避します。 **注:** ご使用のブラウザのバージョンによっては、利用できない設定もあります。

タブ名	オプション	構成	コメント
		• すべての **...警告する** 設定を無効にする	

複数の仮想ユーザーを実行する

他の負荷テスト ツールと異なり、Silk Performer では、Internet Explorer ActiveX コントロールを使用して仮想ユーザーがシミュレートされます。 Internet Explorer コントロールのデフォルトの動作では、Windows ユーザーごとに 1 つのクッキー データベース、キャッシュ、履歴が保持されます。 仮想ユーザーごとに 1 つのクッキー データベース、キャッシュ、履歴を保持する (正確なシミュレーションを行うための要件) ため、負荷テストでは、Silk Performer によって Internet Explorer コントロールが再構成されます。

仮想ユーザーごとに固有の独立した Internet Explorer コントロール サンドボックスがあるため、プロトコル ベースの Web シミュレーションで使用される方法と同じように、初めてのユーザーおよび 2 回目以降のユーザーの動作を正確にシミュレートすることができます。

テスト スクリプトを作成する

テスト スクリプトを作成する最も簡単な方法は、Silk Performer Recorder を使用することです。 Recorder は、トラフィックを測定して記録し、テスト スクリプトを生成する Silk Performer のエンジンです。

Silk Performer Recorder は、テスト対象のクライアント アプリケーションとサーバー間で移動するトラフィックをキャプチャして記録します。 記録が終了すると、Silk Performer Recorder は、記録されたトラフィックに基づいてテスト スクリプトを自動生成します。 スクリプトは、Silk Performer のスクリプト言語である *Benchmark Description Language (BDL)* で記述されます。

テスト スクリプトを記録する

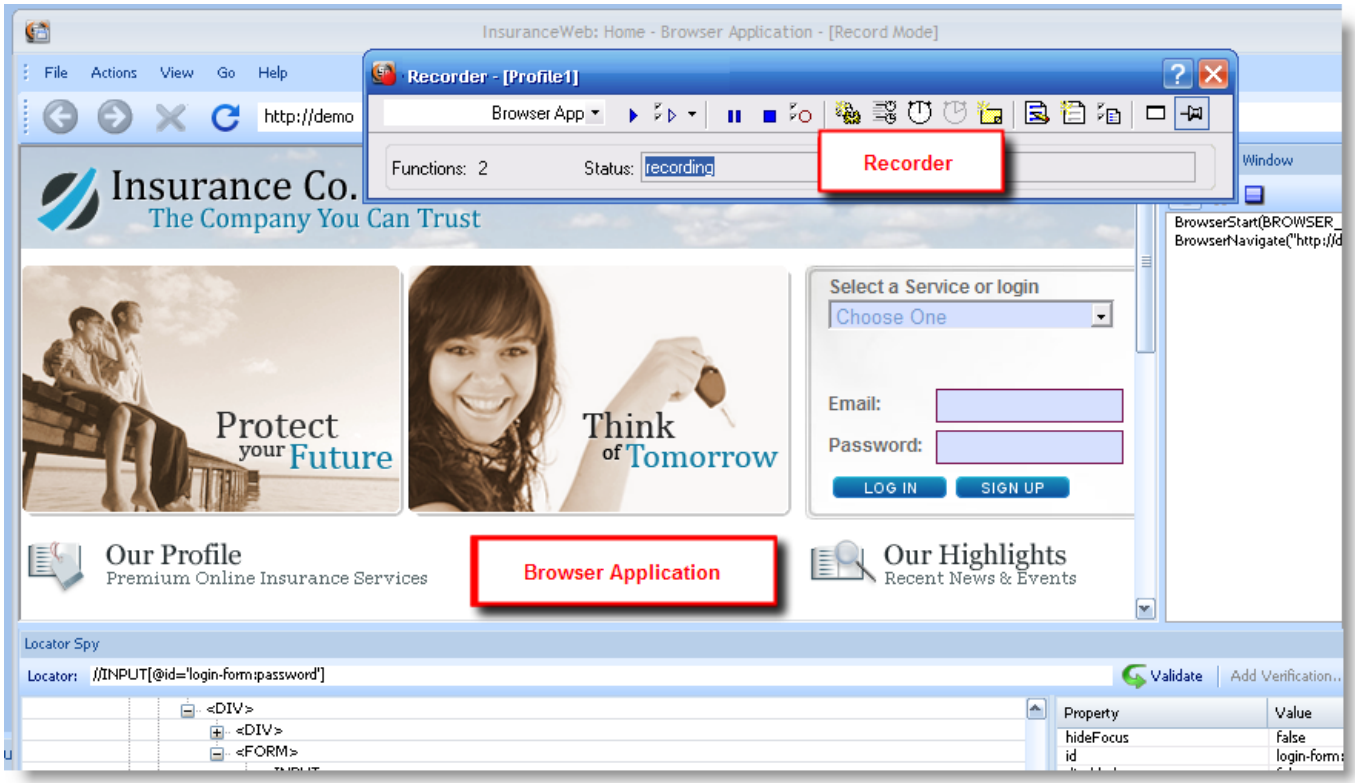
1. ワークフロー バーの **スクリプトの作成** をクリックします。 **ワークフロー - スクリプトの作成** ダイアログ ボックスが表示されます。
2. Silk Performer **Browser Application** を **アプリケーション プロファイル** リストから選択します。
3. **URL** フィールドに、記録する URL を入力します。

   注: InsuranceWeb サンプル Web 2.0 アプリケーションは、*http://demo.borland.com/InsuranceWebExtJS/* から入手できます。 **Select a Service or login** リストでは、Auto Quote および Agent Lookup サービスをテストに使用することができますが、リストに示されたそれ以外のサービスには機能がありません。
4. **記録の開始** をクリックします。

   Silk Performer Browser Application と共に、Silk Performer Recorder が最小化されて開きます。

   注: 記録時のブラウザ ウィンドウのサイズを指定するには、**表示** > **ブラウザ ウィンドウのサイズ変更** に移動し、**幅** および **高さ** にピクセル値を定義します。

   記録中に発生したアクションのレポートを表示するには、Recorder ツールバーの  をクリックして Recorder ダイアログ ボックスを最大化します。



5. Browser Application Recorder を使用して、テスト中に仮想ユーザーが行う操作 (リンクのクリック、フィールドへのデータ入力、データ送信、ポップアップ ウィンドウを開くなど) と同じ方法でサンプル アプリケーションとやり取りします。 アクションは、Browser Application Recorder によってキャプチャされ、記録されます。

   - 記録を一時的に停止し、再開するには、 (**記録の一時停止/再開**) をクリックします。
   - スクリプトの記録を終了し、スクリプトを保存するには、 (**記録停止**) をクリックします。

6. 作業が完了したら、ブラウザ ウィンドウを閉じ、**記録停止** をクリックします。 **名前を付けて保存** ダイアログ ボックスが表示されます。
7. スクリプトに意味のある名前を入力し、**保存** をクリックします。

   実行したユーザー アクションに基づいた BDL テスト スクリプトが、**スクリプト** ウィンドウに表示されます。

Browser Application と Locator Spy の使用方法

便利な記録/再生を有効にするため、Silk Performer には独自の Browser Application が備えられています。 アプリケーションには、次の機能があります。

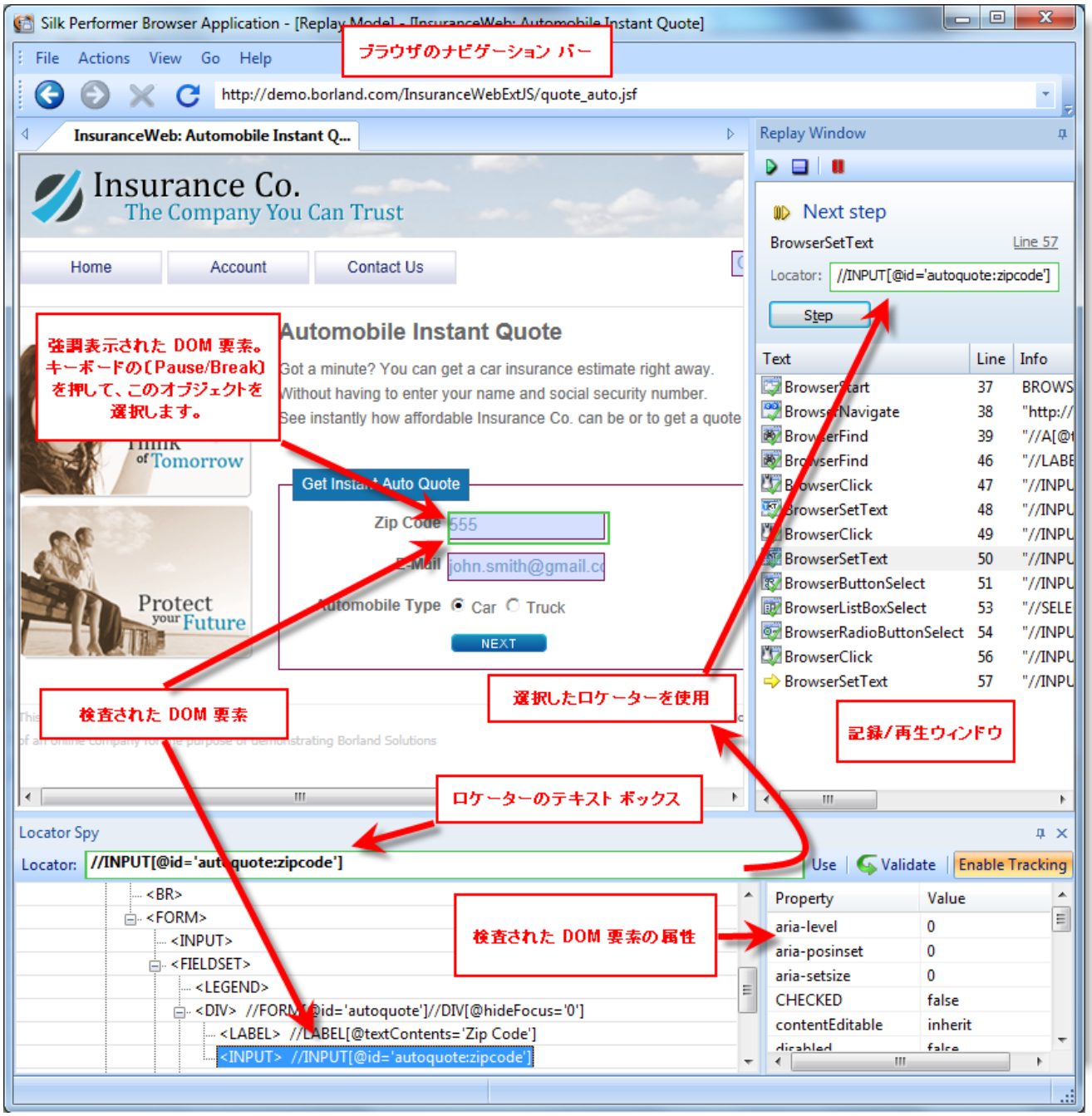
- **ブラウザ ウィンドウ**
- **Locator Spy**
- **記録/再生ウィンドウ**

記録/再生ウィンドウには、記録時と再生時のログ情報が表示されます。 記録の起動/停止と一時停止/再開 (記録モード)、および再生の一時停止/再開 (再生モード) を行うことができます。

次の画像に、再生モードでの Silk Performer Browser Application と、ブラウザ ウィンドウおよび Locator Spy の最も重要な要素を示します。

注: DOM オブジェクトを選択する前に、UI 要素の追跡が有効になっている必要があります。追跡が有効になっている場合、UI 要素の上にカーソルを置くと、要素が緑色の矩形で囲まれます。追跡が有効になっていない場合は **追跡の有効化** をクリックします。

選択した UI オブジェクトのロケータが **ロケータ** テキスト ボックスに表示され、DOM 階層がツリー メニューに表示されます。



ブラウザ ナビゲーション バー

ブラウザ ナビゲーション バー を使用すると、標準のブラウザ ナビゲーションが可能になります。

強調表示された DOM 要素

マウスを移動すると、現在のマウスの位置にある DOM 要素が決まり、DOM 要素の位置が緑色の四角形で示されます。 これにより、現在のページのアーキテクチャと DOM 要素の階層の雰囲気をつかむことができます。

検査された DOM 要素

Pause/Break キーを押すと、次のアクションが起動されます。

- 強調表示された DOM 要素が、検査された DOM 要素になります。
- 検査された DOM 要素の位置が、青色の強調表示で示されます。
- 現在のページの DOM 階層ツリーが決まり、**Locator Spy** に DOM 要素の HTML タグで表示されます。
- 検査された DOM 要素のパスが展開され、検査された DOM 要素が選択されます。
- 検査された DOM 要素の属性が決まり、表示されます。
- 検査された DOM 要素のロケータが決まり、**ロケータ編集フィールド** に表示されます。

選択した DOM オブジェクトで特定のテキストまたは数値文字列を検索するには、キーボードの Ctrl+F を押して、**Locator Spy 内の検索** ダイアログ ボックス (または **アクション** > **DOM ツリー内の検索** を選択) を開きます。 **タグ**、**プロパティ名**、または **プロパティ値** 内の文字列を検索できます。 **次へ** をクリックして、検索文字列のすべてのインスタンスをステップ スルーします。

検査された DOM 要素を変更するには、強調表示された DOM 要素で Pause/Break キーを押すか、DOM 階層ツリー内で別の DOM 要素を選択します。

DOM 階層ツリー内で別の DOM 要素を選択すると、DOM 要素のロケータが決まり、DOM 要素の HTML タグの横に表示されます。 ツリー項目のテキストの更新に加え、**ロケータ** テキスト ボックスが更新され、現在のページの DOM 要素の位置が青色の強調表示で示されます。

Pause/Break キーを押した後で、新しく選択した DOM 要素が見つからないためにページの DOM が無効になると、**ロケータ編集フィールド** の周りに赤色の枠線が表示されます。 Pause/Break キーを押すと、階層ツリーが更新され、現在の DOM オブジェクトが強調表示されます。 DOM 階層ツリー内のロケータ文字列も無効になるため、削除されます。

[ロケータ] テキスト フィールド

ロケータ テキスト フィールドには、現在検査中の DOM 要素のロケータ文字列が表示されます。 検査された DOM 要素が変更されるたびに、ロケータが更新されます。

このテキスト フィールドを使用して、ロケータ文字列を BDL スクリプトなどの別の場所にコピーしたり、現在のページにあるユーザー定義のロケータを検証するためにロケータ文字列を手動で編集することができます。 ロケータ文字列の編集中に、**整合性チェック** をクリックして、ロケータを検証します。 検証が正常に行われると、ロケータ文字列に対応する DOM 要素の位置が緑色で強調表示されます。 検証に失敗すると、**ロケータ** テキスト フィールドの枠線が赤色になります。

スクリプトの試行中に検証を追加する場合は、再生を一時停止して **検証の追加** をクリックします。 スクリプトの試行中における検証の追加は、記録中と完全に同じように機能します。

Locator Spy の右側のウィンドウで、プロパティを右クリックしてプロパティ名、プロパティ値、またはそれら両方をクリップボードにコピーできます。 両方をコピーした場合は、文字列が @name='value' の形式で保存されます。 実世界の例として、@hideFocus='false' があります。 これにより、**ロケーター** フィールドのプロパティを交換する便利な方法を提供します。

検査された DOM 要素の属性

これは、現在検査中の DOM 要素に属する属性 (名前/値のペア) のリストです。 デフォルトで生成されるロケータ文字列が要件を満たしていない場合は、リストに示される属性を使用して、手動で編集したロケータ文字列を構築します。

Browser Application でのロケータの検証

Browser Application には、**再生** ウィンドウでのロケータ情報の分析と操作を簡単にするコマンドが用意されています。 **再生** ウィンドウの任意の API 呼び出しを右クリックして、呼び出しのロケータ情報のコピー、**情報** 列のコンテンツのコピー、および **Locator Spy** DOM 階層ツリーの呼び出しのロケータの表示を行うための状況依存のコマンドにアクセスします。

このようなコマンドは、たとえばロケータの検証または API 呼び出しに失敗したときに使うと便利です。 API 呼び出しのロケータを使用して、**Locator Spy** で呼び出しを探したり、問題をトラブルシューティン

グしたり、それに応じてスクリプトを編集できます。 また、**コピー** コマンドを使用して、API の詳細をコピーして、電子メールや問題レポートに貼り付けることができます。

マウス移動の挿入

特定の要素 (ボタンやメニュー項目など) 上にマウスを使用して移動する場合のみ項目が表示される Web サイトをテストすると、スクリプトの再生中にエラーが発生します。 ホバーリング（マウス移動）イベントは記録されないため、 Silk Performer は項目を検出できません。 このようなケースの例として、 JavaScript で作成されたメニューがあります。 ただし、 Silk Performer では、スクリプトの再生中にこの問題を修正することができます。

エラーが発生した場合は、 **Browser Application** の **問題の解決** ボタンをクリックし、リストから **マウスの移動の挿入** を選択してマウスを UI 要素の上に移動して、キーボードの **Pause/Break** を押し、 **挿入** をクリックして **スクリプトの再実行** をクリックします。 これでスクリプトがエラーなしで実行されるようになります。

検証関数を挿入する

1. Browser Application を使用したブラウザ駆動型のスクリプト記録時に、後でスクリプト再生時に検証する値が含まれる DOM オブジェクトを選択します (キーボードの Pause/Break キーを押して DOM オブジェクトを選択します)。

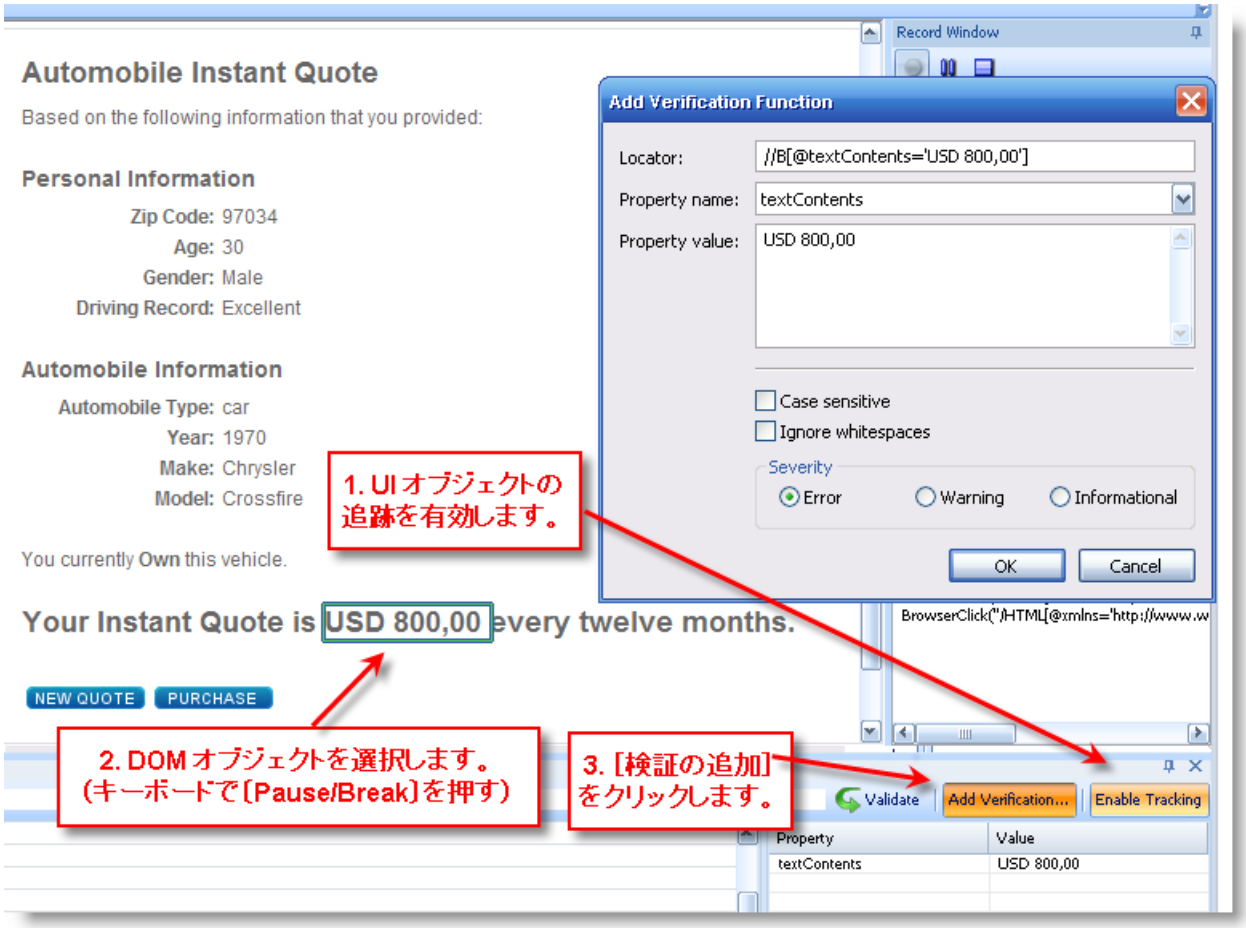
   注: DOM オブジェクトを選択する前に、UI 要素の追跡が有効になっている必要があります。追跡が有効になっている場合、UI 要素の上にカーソルを置くと、要素が緑色の矩形で囲まれます。追跡が有効になっていない場合は **追跡の有効化** をクリックします。

   選択した UI オブジェクトのロケータが **ロケータ** テキスト ボックスに表示され、DOM 階層がツリー メニューに表示されます。
2. **検証の追加** をクリックします。

   検証の追加 ボタンは、**ロケータ** フィールドにロケータの値が表示されたときに有効になります。

   ロケータ フィールドにロケータの値が事前にロードされた状態で **検証関数の追加** ダイアログ ボックスが表示されます。
3. DOM の **プロパティ名** を選択します (たとえば href、class、onmousedown、または textContents)。

   意味のある検証関数にするためには、選択するプロパティ名に、検証可能な **プロパティ値** が必要です。たとえば、プロパティ名 href は、プロパティ値が特定の URL である必要があります。
4. **OK** をクリックして、選択した DOM 要素の BrowserVerifyProperty 検証関数と、対応するプロパティ名と値のペアをスクリプトに挿入します。



検証アクションが **記録ウィンドウ** に記録され、検証関数が BDL スクリプトに挿入されます。

TTI に要素を含める

1. **ファイル** > **プロジェクトの新規作成** をクリックして、新しいブラウザ駆動型プロジェクトを作成します。既存のプロジェクトを使用する場合は、ワークフロー バーから **スクリプトの作成** をクリックして、ステップ 3 に移ります。
2. **Web browser-driven (AJAX)** をツリーから選択し、**名前** と **説明** を入力して **次へ** をクリックします。
3. 記録するアプリケーションの **URL** を入力して **記録の開始** をクリックします。**Browser Application** が起動します。
4. アプリケーションを操作して記録します。要素を TTI 関連要素としてタグ付けするには、要素上にマウスを移動して **Pause/Break** を押し、**TTI に含める** をクリックします。必要な要素をいくつでも含めることができます。Silk Performer は、BrowserTtiIncludeElement() 関数をスクリプトに追加します。

   注目: 含める前に要素をクリックしないように注意してください。このような場合、BrowserTtiIncludeElement() 関数が Click 関数と結び付けられ、再生時に問題が発生する場合があります。
5. **Browser Application** 閉じて記録を停止し、スクリプトを保存します。

スクリプトの試行

テスト スクリプトを生成したら、スクリプトの試行を行って、スクリプトがエラーなしで動作するかどうかを確認します。 スクリプトの試行を行うことによって、Browser Application ベースの Recorder によ

って記録された操作がスクリプトで正確に再現されるかどうかがわかります。 また、スクリプトがエラーなしで動作できるようにするために、パラメータ化を必要とするコンテキスト固有のセッション処理がスクリプトに含まれているかどうかもわかります。

スクリプトの試行では、1 人の仮想ユーザーのみが実行され、トランザクション間の思考時間遅延が発生しない ストレス テスト オプションが有効になります。

注: ブラウザ駆動型のスクリプトの試行のデフォルト オプション設定には、ログ ファイルとレポート ファイルの作成および Browser Application の **再生ウィンドウ** 内での再生は含まれますが、テスト時にダウンロードされたコンテンツのリアルタイム表示 (TrueLog Explorer 経由) は含まれません。

テスト スクリプトを試行する

1. ワークフロー バーで、**スクリプトの試行** をクリックします。 作成したスクリプトが **スクリプト** リストで選択され、アクティブ プロファイルが **プロファイル** リストで選択されている状態で、**スクリプトの試行** ダイアログ ボックスが表示されます。 [VUser] という仮想ユーザー グループが、**ユーザー グループ** グループ ボックスで選択されています。
2. 設定を次のように構成します。
   a) Browser Application の **再生ウィンドウ** に Web ページのコンテンツが表示されるようにするため、**クライアントの表示** オプションを有効にします。
   アプリケーション状態のスクリーンショットは、各 API 関数呼び出しの前に作成されます。

   注: シミュレーション設定は、Browser Application でスクリプトを再生する場合には適用されません。

   b) **Browser Application** ウィンドウでスクリプトをステップごとに実行するため、**ステップ実行** オプションを有効にします。
3. **実行** をクリックします。

   注: ここでは、実際の負荷テストではなく、スクリプトのデバッグが必要かどうかを確認するために、1 人の仮想ユーザーを使用したテスト実行のみを行います。

   スクリプトの試行が開始されます。 **監視** ウィンドウが開き、実行の進捗についての詳細な情報が表示されます。

スクリプトの試行のステップ再生

スクリプトの試行 ダイアログで **ステップ実行** を有効にすると、スクリプトの試行の再生を 1 ステップずつ進めることができます。

1. 上記の説明に従って [スクリプトの試行] を実行します。
   スクリプトの試行 ダイアログで **ステップ実行** オプションを有効にします。
2. **再生ウィンドウ** のボタンを使用して再生を制御します。

   - 現在の API 呼び出しを実行するには、 (**ステップ再生**) をクリックします。
   - 残りの API 呼び出しを続けて実行するには、 (**再生の実行**) をクリックします。
   - スクリプトの試行を終了するには、 (**再生の停止**) をクリックします。

一般的な再生エラー

以下に、記録後にスクリプトが正しく再生されない一般的な理由を示します。 このような場合には、テスト スクリプトをカスタマイズする必要があります。

- **ステートフルなスクリプト :** 記録されたスクリプトは、テストするアプリケーションが再生時とスクリプトの記録時で同じ状態である場合にのみ動作します。 たとえば、ユーザー ログインを含むスクリプトは、アプリケーションがログアウト状態である場合にのみ正しく実行できます。 この問題を解決するには、ロジックをスクリプトに手動で追加してアプリケーションの状態を設定するか、記録したスクリ

プトによってアプリケーションの状態が変更されないことを最初に確認します (たとえば、スクリプトの記録中にユーザーのログアウトを含めることができます)。

- **一時的に生成される DOM 属性:** 一部の AJAX フレームワークでは、ページがロードされるたびに変更する属性 (ext の x-auto の値など) が生成されます。 ロケータがこのような属性に依存していると、スクリプトの再生に失敗します。 無視する属性リストに属性を追加して、属性が今後記録されないようにする必要があります。
- **マウス移動の欠落 :** 特定の要素 (たとえば、ボタンまたはメニュー項目など) の上にマウスを移動した場合のみ表示される項目がある Web サイトをテストする場合は、スクリプトの再生中にエラーが発生します。 要素の上にマウスを移動する動作は記録されないため、Silk Performer は項目を検出できません。 このようなケースの例として、JavaScript で作成されたメニューがあります。 ただし、Silk Performer では、スクリプトの再生中にこの問題を修正することができます。 エラーが発生した場合は、 **Browser Application** の **問題の解決** ボタンをクリックし、リストから **マウスの移動の挿入** を選択してマウスを UI 要素の上に移動して、キーボードの **Pause/Break** を押し、 **挿入** をクリックして **スクリプトの再実行** をクリックします。 これでスクリプトがエラーなしで実行されるようになります。
- **同期タイムアウトに遭遇する呼び出し :** AJAX 組み込み同期では、ブラウザがアイドル状態になるまで待機してから API 呼び出しが返されます。 これは、AJAX ベースのアプリケーションで信頼できるテストを行ううえで重要な要因となります。 ただし、アイドル状態がない場合もあります (ページでポーリングを使用したり、サーバープッシュ イベント用に接続を開いたままにする場合など)。 このような場合には、タイムアウトが発生するまで同期が待機します。 この問題は、同期モードの設定を一時的に HTML に戻すことによって解決できます。

テスト スクリプトを分析する

Web プロトコルによる負荷テストの方法と異なり、ブラウザ駆動型の Web 負荷テストでは、スクリプトの検証に Browser Application が使用されます。

Browser Application でスクリプト試行を行う利点は、次のとおりです。

- スクリプトの高度な変更/適合を行うための Locator Spy 機能を含め、アプリケーション状態がリアルタイムでブラウザに表示されます。
- ステップ モードでスクリプトを実行できます。
- 各ブラウザ API 呼び出しの前にスクリーンショットがキャプチャされ、以降の分析用に TrueLog に保存されます。

Browser Application でスクリプト試行が正常に行われたら、スクリプト試行の結果を TrueLog Explorer で分析できます。 TrueLog Explorer によるテスト スクリプトの分析には、次のタスクが含まれます。

- 仮想ユーザー要約レポートを表示する
- エラーの検索
- 再生テスト実行と記録テスト実行の比較

TrueLog Explorer による視覚的分析

TrueLog Explorer の最も強力な機能の 1 つは、テスト中のアプリケーションによって表示された Web コンテンツを視覚的にレンダリングする機能です。 実際に、仮想ユーザーがアプリケーションとやり取りをするときに見るものが表示されます。

TrueLog Explorer のインターフェイスは、次のセクションから構成されています。

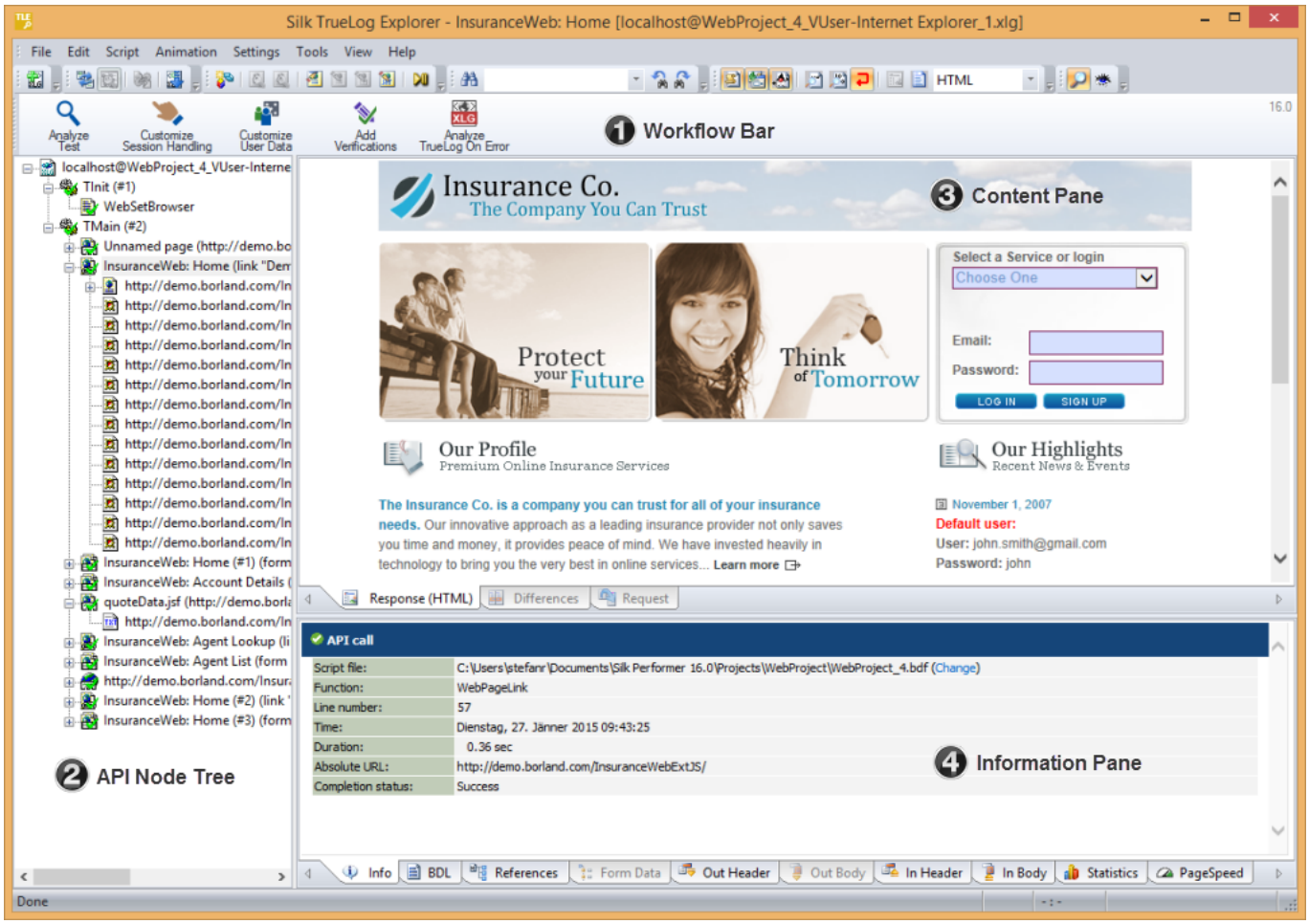
- **ワークフロー バー** は、TrueLog Explorer を扱う際の主要なインターフェイスとなります。 ワークフロー バーは、TrueLog Explorer に組み込まれているテスト方法論を元に、5 つの主要な作業をサポートしています。
- インターフェイスの左側にある **API ノード ツリー** メニューでは、負荷テスト中にダウンロードされた TrueLog データを展開または縮小できます。 ロードされている TrueLog ファイルはそれぞれ、関連す

るすべての API ノードへのリンクと共に、ここに表示されます。 ノードをクリックして、**画面** ペインにスクリーンショットを表示し、**情報** ビューに履歴詳細を表示することができます。

- **コンテンツ** ペインでは、受信したすべてのデータに対して複数のビューを利用できます。
- **情報** ペインには、テスト スクリプトおよびテスト実行に関するデータが表示されます。読み込まれた TrueLog ファイル、選択された API ノード、 BDL スクリプト、統計に関する一般情報が含まれます。

注: HTTP ヘッダー データは、現時点では使用できません。

注: Silk Performer から TrueLog Explorer を起動するには、**結果** > **TrueLog の検討** を選択します。



テスト実行を分析する

1. スクリプト試行によって TrueLog が TrueLog Explorer に読み込まれた状態で、ワークフロー バー の **テストの分析** ボタンをクリックします。
   テストの分析 ダイアログ ボックスが表示されます。
2. 次のいずれかのオプションで操作を続行します。
   - 仮想ユーザー要約レポートの表示
   - TrueLog 内のエラーの検索
   - 再生テスト実行と記録テスト実行の比較

要約レポートを表示する

仮想ユーザー要約レポートは、個々のスクリプト試行について要約したレポートで、基本情報とタイミング平均を含みます。 それぞれのレポートは、仮想ユーザーごとに記録されており、表形式でデータが表示されます。

仮想ユーザー要約レポートには、以下に関する詳細情報が含まれています。

- 仮想ユーザー
- 検出されたエラー
- テスト スクリプトに定義されているトランザクションごとに追跡したレスポンス時間情報
- ダウンロードされた Web ページごとのページ タイマの測定値
- スクリプトで使用された個別のタイマおよびカウンタ (Measure 関数)

仮想ユーザー要約レポートを表示する

1. スクリプトの試行によって生成された TrueLog が TrueLog Explorer に読み込まれた状態で、**テストの分析** ボタンをクリックします。
2. **仮想ユーザー要約レポートの表示** リンクをクリックします。

要約レポートを有効にする

仮想ユーザー要約レポートの処理にはかなりのリソースを必要とするので、デフォルトでは生成されない設定になっています。 アニメーションを指定したスクリプトの試行が終了したとき (または **API ノード ツリー** メニューで TrueLog ファイルのルート ノードをクリックしたとき) に、自動的に仮想ユーザー レポートが表示されるようにするには、**仮想ユーザー レポートを表示する** オプション (**設定** > **ワークスペース** > **レポート**) を有効にします。

注: 仮想ユーザー レポートは、Silk Performer 内でも表示させることができます。仮想ユーザー名を右クリックし、**仮想ユーザー レポート ファイルの表示** を選択します。

TrueLog 内のエラーを検索する

TrueLog Explorer では、スクリプトの試行の後でエラーをすばやく検索できます。 エラーとなったリクエストを調査して、TrueLog Explorer で必要なカスタマイズを行うことができます。

注: API ノード ツリー メニューで表示すると、再生エラーを含む API ノードには、赤い「X」印が付いています。

1. スクリプトの試行によって生成された TrueLog が TrueLog Explorer に読み込まれた状態で、**テストの分析** ボタンをクリックします。
2. **エラーの検索** リンクをクリックします。 **TrueLog のステップ スルー** ダイアログが **エラー** オプションの選択された状態で開きます。
3. 一度に 1 つのエラーを検索しながら TrueLog 結果ファイル内を移動するには、**次を検索** をクリックします。

ページ統計値を表示する

テスト実行の正確性を検証したら、ページ統計を利用して、「負荷のない」状態のアプリケーションのパフォーマンスを分析できます。

概要ページの詳細：

- アクション時間：ページの総レスポンス時間。ブラウザでの処理およびレンダリングの時間が含まれます。

- ドキュメント時間 : ドキュメント ダウンロード時間 (サーバーのビジー時間を含む)、および埋め込みオブジェクトの受信に要した時間が表示されます。

アクションの詳細な統計値には、個々の Web ページ コンポーネントの正確なレスポンス時間が表示されるため、エラーや、ページのダウンロードが遅くなる根本原因を簡単に突き止めることができます。

スクリプトの試行には、思考時間が含まれないため、TryScript による測定結果は、実世界のパフォーマンスを予測するためには利用できません。

アクションの詳細な統計値には、各ページ コンポーネントの次のデータが含まれます。

- DNS 検索時間
- 接続時間
- ラウンドトリップ時間
- キャッシュ統計値

注: プロトコル ベースの方法と比較すると、ブラウザ駆動型のテスト統計には、特定の低レベル/プロトコル関連の指標は含まれません。

概要ページを表示する

1. API ノードのツリー メニューから、統計を表示する API ノードを選択します。
2. **TrueLog のステップ スルー** ダイアログ ボックスで、**ブラウザ ノード** を選択します。
3. **統計** タブをクリックして、**統計** ビューを開きます。
4. 詳細分析およびページの掘り下げを行うには、[URL] 列に一覧されている中から、特定のコンポーネントを選択します。

記録 Truelog と再生 Truelog を比較する

Web アプリケーションのテストの場合、TrueLog Explorer には、テスト時に受信された実際の Web ページが表示されます。 TrueLog Explorer のアニメーション モードでは、ダウンロードされたデータをリアルタイムで監視することができます。 テスト中に受信されたとおりにデータが表示されます。

スクリプトの開発プロセス中に生成された TrueLog と、当初生成された TrueLog を比較することで、テスト スクリプトが正確に実行されたかどうかを確かめることができます。

1. ワークフロー バーの **テストの分析** ボタンをクリックします。 **ワークフロー - テストの分析** ダイアログ ボックスが表示されます。
2. **テスト実行の比較** をクリックします。
3. 対応する記録 TrueLog が比較ビューに開き、**TrueLog のステップ スルー** ダイアログ ボックスが、**ブラウザ ノード** オプションが選択された状態で表示されます。これにより、TrueLog をノードごとに比較することができます。
4. **次を検索** ボタンをクリックすると、TrueLog 結果ファイル内を 1 ページずつ移動することができます。

注: 再生時のコンテンツを表示しているウィンドウには、左上隅に緑色の三角マークが付いています。 アプリケーションの記録時に元々表示されていたコンテンツを表示しているウィンドウには、左上隅に赤い三角マークが付いています。

プロジェクトのプロファイル設定を構成する

Silk Performer には、ブラウザ駆動型 Web 負荷テストの多様なプロファイル設定が備えられています。 Web (ブラウザ駆動) プロファイル設定は、同期およびオブジェクト ロケータ生成に関係するプロジェクト固有の設定です。 設定内容は、プロジェクトごとに指定されます。

注: このチュートリアルでは、デフォルト設定を変更する必要はありません。

ブラウザ駆動型の記録設定を構成する

1. **プロジェクト** ツリー メニューで **プロファイル** ノードを右クリックして、**アクティブ プロファイルの編集** を選択します。 **シミュレーション** タブ (**再生** カテゴリ) に **プロファイル - [Profile1] - シミュレーション** ダイアログ ボックスが表示されます。
2. **記録** をクリックします。
3. **Web (ブラウザ駆動)** までスクロールして選択します。
4. **記録** タブを選択します。
5. **無視する DOM 属性名** テキスト フィールドに、記録時に無視する DOM 属性名を入力します。 **無視する DOM 属性名** フィールドのどのパターンにも一致する属性名が、記録時に無視されます。
6. **無視する DOM 属性値** テキスト フィールドに、記録時に無視する DOM 属性値を入力します。 **無視する DOM 属性値** フィールドのどのパターンにも一致する属性値が、記録時に無視されます。
7. **優先する DOM 属性名** オプションは、記録されるカスタム属性名を構成します。
8. **OK** をクリックします。

ブラウザ駆動型の再生設定を構成する

1. **プロジェクト** ツリー メニューで **プロファイル** ノードを右クリックして、**アクティブ プロファイルの編集** を選択します。 **シミュレーション** タブに **プロファイル - [Profile1] - シミュレーション** ダイアログ ボックスが表示されます。
2. **再生** カテゴリ ボタンをクリックします。
3. **Web (ブラウザ駆動)** までスクロールして選択します。 **Web (ブラウザ駆動) / 全般** タブが表示されます。
4. **シミュレーション** グループ ボックスを使用して、Web サイトを訪問するユーザーの現実的なシミュレーションのためのオプションを設定します。
   - 初めて Web サイトを訪問したユーザーを実際にシミュレートするには、**初めてのユーザー** オプション ボタンをクリックします。

     トランザクションごとに、永続的な接続は閉じられ、Web ブラウザのエミュレーションがリセットされます。また、ドキュメント キャッシュ、ドキュメント履歴、Cookie データベース、認証データベース、および SSL コンテキスト キャッシュもクリアされます。このオプションを選択した場合、Silk Performer は、サーバーから全サイト(すべてのファイルを含む)をダウンロードします。
   - Web サイトを再訪問したユーザーを実際にシミュレートするには、**2 回目以降のユーザー** オプション ボタンをクリックします。トランザクションごとに、非永続的なセッションは閉じられますが、ドキュメント履歴、永続 Cookie データベース、およびコンテキスト キャッシュはクリアされません。このような場合、ドキュメント キャッシュにあるページはダウンロードされません。
   - ユーザーの Web ブラウザで自動再生を表示するときに Internet Explorer (IE) が使用するレンダリング モードを定義するには、**IE 互換モード** を選択します。異なる HTTP ヘッダーをサーバーに送信し、Web コンテンツをレンダリングするように IE のバージョンを構成できます。たとえば、IE9 は IE7 として IE7 ヘッダーを送信し、IE7 としてレンダリングします。**デフォルト** 値は、ユーザーの Internet Explorer ブラウザ バージョンによって異なります。

   注: シミュレーション設定は、Browser Application でスクリプトを再生する場合には適用されません。ただし、Internet Explorer の インターネット オプション内で構成するすべてのキャッシュ設定は、データ駆動型テストに適用されます。
5. **再生時の動作** を選択して、ロケーターが生成される方法を定義します。このオプションを設定すると、互換性の問題を避けることができるため、新しいバージョンの Silk Performer で古いスクリプトを問題なく再生できます。BrowserSetReplayBehavior 関数を手動でスクリプトに記述することによって、すべての単一のスクリプトに対して再生時の動作を定義することもできます。

6. **従来の入力モード** 設定が無効になっていることを確認します。
7. **同期** タブを選択します。
8. 必要に応じて、**同期** 設定を構成します。
   - **同期モード** オプションは、ブラウザの起動呼び出しの準備状態を待機するために使用されるアルゴリズムを構成します (呼び出しの前後)。
   - **同期タイムアウト** オプションは、オブジェクトが準備完了になるまで待機する最大時間 (ミリ秒単位) を構成します (呼び出しの前後)。
   - **同期から除外する URL** テキスト ボックスに、除外するサービスまたは Web ページの URL 全体あるいは URL の一部を入力します。AJAX フレームワークやブラウザによっては、サーバーから非同期にデータを取得するために、特殊な HTTP 要求を継続して出し続けるものがあります。これらの要求により、指定した同期タイムアウトの期限が切れるまで同期がハングすることがあります。この状態を回避するには、HTML 同期モードを使用するか、問題が発生する要求の URL をここで指定します。複数のエントリをカンマで区切って指定します。
   - **オブジェクト解決タイムアウト** オプションは、再生中にオブジェクトが解決されるまで待機する最大時間 (ミリ秒単位) を構成します。
   - **オブジェクト解決再試行間隔** オプションは、解決されないオブジェクトの後で別の再生が試行するまでの時間 (ミリ秒単位) を構成します。
9. **OK** をクリックします。

ブラウザ駆動型テストの高度な概念

記録時のブラウザ ウィンドウのサイズを定義する

ブラウザ駆動型負荷テストで使用される Browser Application を起動します。

注: ブラウザのサイズは、スクリプトの記録時にのみ定義できます。

1. 記録時の特定のブラウザ ウィンドウのサイズを定義するには、**表示** > **ブラウザ ウィンドウのサイズ変更** に移動します。**ブラウザ ウィンドウのサイズ変更** ダイアログ ボックスが表示されます。
2. **幅** にピクセル値を指定します。
3. **高さ** にピクセル値を指定します。
4. **OK** をクリックします。

システム コードページ以外の文字を使用した Web サイトのテスト

Silk Performer は、マルチバイト文字セット (MBCS) ベースのアプリケーションです。ブラウザ駆動型負荷テストを使用する際に、正しいシステム コードページを設定しなければなりません。これによって、Web サイトで表示される文字が正しく処理されることを保障します。

表示可能でない文字を使用した Web サイトの 負荷テストをサポートするために、Silk Performer は、これらの文字を変換します。例：次の文字列は、数値のセットに変換されます。

русский ⟶ [raw[440 443 441 441 43a 438 439]]

数値は、各文字の Unicode 値を 16 進表記で表します。スクリプトを再生すると、Silk Performer は、文字列を逆変換し、ブラウザを操作する際にそれを使用します。

注: Silk Performer は、ブラウザ駆動型 API 呼び出しにのみ、変換を適用します。他の API 呼び出しに表示可能でない文字を使用しないでください。

ブラウザ駆動型の負荷テストのトラブルシューティング

実際のユーザー アカウントを使用した perfrun 処理の開始方法、クライアント証明書の処理方法、および AJAX 同期からの特定の URL の除外方法について説明します。

注: ブラウザ駆動型の負荷テストは、Internet Explorer 10、11 でサポートされています。

リモート エージェント上のブラウザ駆動の仮想ユーザー

システム アカウントではなく実際のユーザー アカウントでリモート エージェントを開始すると (デフォルト)、ブラウザ駆動の仮想ユーザーに大きな違いが生じます。 各仮想ユーザーは、自分の Internet Explorer インスタンスを使用しますが、このインスタンスには Microsoft Windows ユーザーのプロファイルに保存されている設定が読み込まれます。

システム アカウントの元では、Internet Explorer にはユーザー アカウントの場合とは異なる設定が読み込まれます。 通常、Internet Explorer では、ユーザー アカウントの場合よりも数が少ない別のヘッダーが使用されます。 記録されたトラフィックが生成されたトラフィックと異なる問題を回避するには、リモート エージェントをユーザー アカウントで実行することをお勧めします。

注: 指定されたユーザー アカウントが、リモート エージェントで Remote Desktop Users Windows グループのメンバになっていることを確認します。

必要なアカウント設定は、**アプリケーション** タブの System Configuration Manager で設定でき、すべてのリモート エージェントを同一のユーザー アカウントで実行する場合のユーザー アカウントは **システム設定** > **エージェント** > **の 詳細設定** タブで設定できます。

クライアント証明書を処理する

スクリプトの記録中にクライアント証明書を選択できます。クライアント証明書によって、特定の Web サイトに対する認証が容易になります。Microsoft の証明書ストアから入手した証明書をインポートしたり削除するための API が利用できるようになりました。この API は、Internet Explorer および Silk Performer のブラウザ駆動型負荷テスト機能で使用されます。

証明書 API は、Microsoft Windows 7 以降または Microsoft Windows Server 2008 R2 以降、かつ Internet Explorer 8 以降で利用できます。

ブラウザベースの Web 負荷テストでの証明書の処理は、プロトコルベースの Web テストでの証明書の処理とは無関係に行われます。そのため、証明書は手動で Internet Explorer の **インターネット オプション** メニュー エントリ (または管理コンソールの snap-in certmgr.msc) からインポートする必要があります。認証が Internet Explorer 8 で機能する場合は、ブラウザベースの負荷テストでも機能します。

1. 次の手順に従って、証明書をインポートするときに、強力な秘密キーの保護を無効にします。
   a) **証明書のインポート** ウィザードの **パスワード** ページで、**秘密キーの保護を強力にする** チェックボックスをオフにします。
2. サーバー証明書の失効を無効にします。
   a) Internet Explorer の **ツール** メニューから **インターネット オプション** を選択します。**インターネット オプション** オプション ダイアログが開きます。
   b) **詳細設定** タブをクリックします。
   c) **サーバー証明書の取り消しを確認する*** チェックボックスをオフにします。
   d) **OK** をクリックします。
3. 次の手順に従って、クライアント証明書の選択ダイアログ ボックスを有効にします。
   a) Internet Explorer の **ツール** メニューから **インターネット オプション** を選択します。**インターネット オプション** オプション ダイアログが開きます。
   b) **セキュリティ** タブをクリックします。
   c) **レベルのカスタマイズ...** をクリックします。**セキュリティの設定** ページが開きます。

d) **既存のクライアント証明書が 1 つ、または存在しない場合の証明書の選択** までスクロール ダウンし、**無効にする** オプション ボックスを選択します。
e) **OK** をクリックします。
f) Internet Explorer を再起動します。

証明書エラーの除去

記録中に、Web ページに この Web サイトのセキュリティ証明には問題があります という内容のメッセージが表示される場合があります。 また、このサイトの閲覧を続行する (推奨されません)。 リンクが機能しなくなります。 証明書エラーにはいくつかの原因が考えられ、ブラウザ駆動の Web サイトを記録可能にする前に、証明書エラーを解決する必要があります。 証明書エラーに関する詳細は、証明書エラーについて を参照してください。

非常によくある問題の 1 つはアドレスの不一致です。 アドレス不一致の警告を無効にするには、以下を行います。

1. Internet Explorer の **ツール** メニューから **インターネット オプション** を選択します。 **インターネット オプション** オプション ダイアログが開きます。
2. **詳細設定** タブをクリックします。
3. **証明書のアドレスの不一致について警告する*** チェックボックスをオフにします。
4. **OK** をクリックします。
5. Internet Explorer を再起動します。

AJAX 同期から URL を除外する

AJAX ベースの Web アプリケーションのテストが容易になるよう、特定の URL をブラウザの同期から除外できます。

この価値を理解するには、サーバーからデータをポーリングすることでアプリケーションにサーバー時間を表示することを想像してみてください。このサービスを利用するには、クライアントとサーバーの間のトラフィックのストリームが一定である必要があります。この機能拡張は、アプリケーションがアイドル状態に一切ならないため、AJAX 同期への挑戦といえます。同期からこのサービスを除外することで、別のサービスを使用する他のアプリケーションの処理を正確にテストできるようになります。

1. **プロジェクト** メニュー ツリーでプロファイルを右クリックして、**プロファイルの編集** を選択します。 **プロファイル - シミュレーション** ウィンドウが開きます。
2. **再生** グループ ボックスで下向き矢印をクリックして、画面をスクロールします。**Web (ブラウザ駆動型)** をクリックします。
3. **同期** タブを選択します。
4. 除外する URL を **同期から除外する URL** テキスト フィールドに入力します。
5. **OK** をクリックします。

注: 単一のサービス内で複数の処理が実行されているために URL の除外が実行できない場合は、AJAX 同期を無効にし、HTML モードに切り替える必要があります。

テスト後にインターネット一時ファイルを削除する

ブラウザ駆動型負荷テストを中断したり、負荷テストの実行中に実行中の仮想ユーザーを強制終了した場合、収集されたインターネット一時ファイルはハード ディスクから削除されません。このような方法で、長い間、負荷テストを定期的に停止していると、ディスク上のデータが非常に大量になっていることがあります。このような場合、一時ファイルのフォルダから手動でデータを削除してください。

停止ボタンを使って負荷テストを終了した場合は、すべての一時ファイルが自動的に削除されます。

ブラウザ駆動型の Web 負荷テスト プロジェクトを定義する

1. Silk Performer ワークフロー バーの **ここから開始する** をクリックします。

   注: 別のプロジェクトが既に開いている場合に、メニュー バーから **ファイル** > **プロジェクトの新規作成** を選択すると、現在開いているプロジェクトを閉じるように確認メッセージが表示されます。

   ワークフロー - プロジェクトの概要設定 ダイアログ ボックスが開きます。
2. **名前** テキスト ボックスに、プロジェクトの名前を入力します。
3. オプションのプロジェクト説明を **説明** に入力します。
4. **種類** メニュー ツリーで、**Web browser-driven (AJAX)** を選択します。
5. **次へ** をクリックし、設定に基づいてプロジェクトを作成します。

ワークフロー - スクリプトの作成 ダイアログ ボックスが表示されます。

操作開始時間

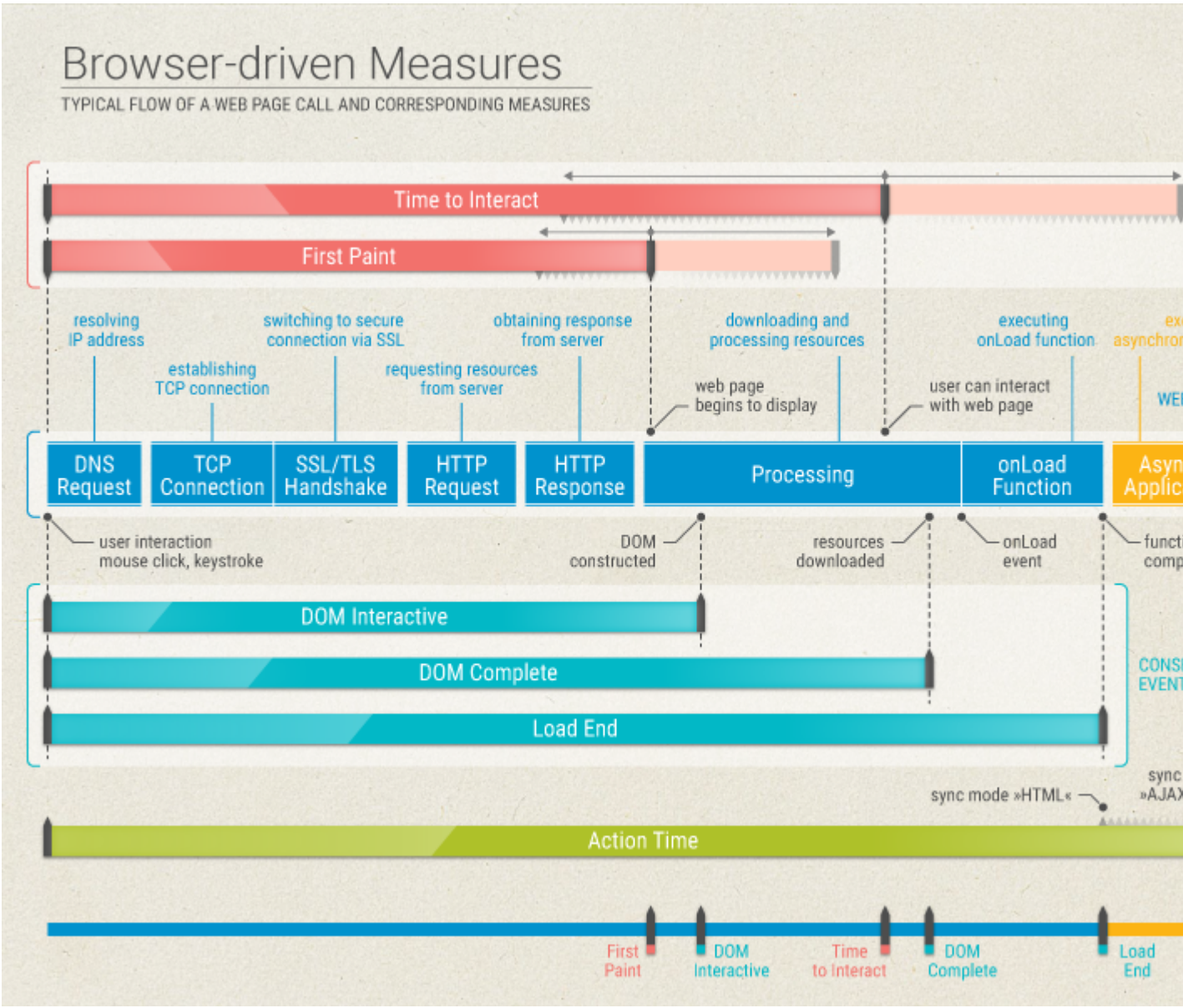
AJAX Web サイトのテストの課題

ブラウザーが提供するタイミングで AJAX Web サイトのユーザー エクスペリエンスを測定することは、非常に困難になります。ユーザーが Web ページが準備できていると感じたときでも、バックグラウンドでまだ処理が完了していない場合もあります。また、処理が完了しても、その時点では Web ページがユーザーに対してまだ準備ができていないこともあります。重要なページ要素が *onLoad* 関数（*onLoad Function*）フェーズの後や 非同期アプリケーション ロジック（*Asynchronous Application Logic*） フェーズに非同期にロードされることがあります。このような場合、Web ページの感覚的なロード時間が、測定したロード時間とかなり異なる場合があります。このような理由から、Silk Performer ではいわゆる 操作開始時間（TTI： Time to Interact）を導入しました。

操作開始時間

Silk Performer の用語では、操作開始時間 を、ユーザーの操作（URL への移動やリンクのクリックなど）からユーザがページを操作するのに必要なすべての関連要素が準備完了になるまでの時間（ページのロードがまだ完了していない場合でも）として定義します。ページの関連要素の識別は、ユースケースやその要素が関連していると感じる観点に強く依存するもので、単純に自動化できません。例 : ある Web ショップでは、運営会社側の観点と、顧客側の観点という、異なる観点から Web サイトをテストできます。運営会社側の観点では、おすすめ商品を提示する要素を関連していると感じるかもしれません。しかし、顧客側の観点では、単に検索フィールドだけを関連要素であると感じるかもしれません。

このため、パフォーマンス エンジニアは、記録時にすべての TTI 関連要素にタグ付けする必要があります。Recorder は、すべてのタグ付けされた要素に対して BrowserTtiIncludeElement() 関数を生成します。再生時に Silk Performer は TTI 関連要素のそれぞれのロードにかかる時間を測定し、その最大値を 操作開始時間 としてレポートします。



ブラウザ駆動型の測定値

ブラウザ駆動型テストでは、Silk Performer は次の測定値を提供します。

測定値	説明
DOM インタラクティブ（DOM Interactive）	リクエストからブラウザがすべての HTML 要素を解析し、DOM を構築するまでの期間を表します。
DOM 完了（DOM Complete）	リクエストからブラウザがすべてのリソース（画像、スタイルシート、スクリプトなど）をダウンロードし、処理を完了するまでの期間を表します。
ロード終了（Load End）	リクエストからブラウザが onLoad 関数の実行を完了するまでの期間を表します。すべてのページのロード処理の最後のステップとして、ブラウザーは onLoad 関数をトリガーする *onLoad* エベントを送信します。onLoad 関数が実行されると、さらにアプリケーションのロジックが実行される場合があります。

測定値	説明
描画開始（First Paint）	リクエストからページの表示が始まるまでの期間を表します。
操作開始時間（Time to Interact）	リクエストからすべての TTI 関連要素がページ上で利用可能になるまでの期間を表します。この時点で、ユーザーは Web ページを操作できるようになります。
アクション時間（Action Time）	リクエストからブラウザがすべてのリソースをダウンロードし、処理を完了するまでの期間を表します。この期間の終了は、定義された同期モードによって異なります。同期モードが HTML の場合、onLoad 関数の完了時にアクション時間が終了します。同期モードが AJAX の場合、非同期アプリケーション ロジック フェーズ中にアクション時間が終了します。

注: DOM インタラクティブ、**DOM 完了**、**ロード終了** は連続したイベントになります。これらのイベントはすべて、処理（*Processing*） フェーズ中にに終了します。一方、**描画開始** と **操作開始時間** は完全に Web サイトに依存しており、*HTTP* レスポンス（*HTTP Response*） フェーズ、処理 フェーズ、非同期アプリケーション ロジック フェーズのどこかで終了します。

索引

A

H

T

U

W

あ

え

く

し

せ

そ

た

と