MICRO FOCUS

Silk Test 19.0

Silk4J ユーザー ガイド

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

2018-06-07

目次

Silk4J 19.0 へようこそ

Silk4J 19.0 へようこそ

Silk4J について
製品スイート

新機能

リリース ノート

主なセクション

Silk4J を使用したベスト プラクティス
テストの作成
特定の環境のテスト

チュートリアルとデモ

クイック スタート チュートリアル

オンライン リソース

Micro Focus のホーム ページ
Micro Focus リソース ページ
YouTube の *Micro Focus* チャンネル
オンライン ドキュメント
Micro Focus SupportLine
Micro Focus 製品アップデート
Silk Test Knowledge Base
Silk Test Forum
Web ベースのトレーニング

フィードバックをお寄せください

Micro Focus へのお問い合わせ（313 ページ）
このヘルプに関するフィードバックを電子メールで送信してください
機能を提案する

ライセンス情報

評価版を使用しているのでない限り、Silk Test はライセンスを必要とします。

注: Silk Test ライセンスは、Silk Test の特定のバージョンに固定されています。たとえば、Silk Test 19.0 には Silk Test 19.0 のライセンスが必要です。

ライセンス モデルは、使用しているクライアントとテストすることができるアプリケーションに基づきます。利用可能なライセンス モードに応じて、次のアプリケーションの種類がサポートされます。

ライセンス モード	アプリケーションの種類

モバイル ネイティブ	• モバイル Web アプリケーション
	  • Android
	  • iOS
	• ネイティブ モバイル アプリケーション
	  • Android
	  • iOS

フル	• Web アプリケーション (以下を含む)
	  • Apache Flex
	  • Java アプレット
	• モバイル Web アプリケーション
	  • Android
	  • iOS
	• Apache Flex
	• Java AWT/Swing (Oracle Forms を含む)
	• Java SWT と Eclipse RCP
	• .NET (Windows Forms および Windows Presentation Foundation (WPF) を含む)
	• Rumba
	• Windows API ベース
	注: ライセンスをフル ライセンスにアップグレードする場合は、*http://www.microfocus.co.jp* に移動します。

プレミアム	フル ライセンスでサポートされるすべてのアプリケーションの種類 + SAP アプリケーション **注:** ライセンスをプレミアム ライセンスにアップグレードする場合は、*http://www.microfocus.co.jp* に移動します。
モバイル ネイティブ アドオン	フル ライセンスとプレミアム ライセンスでサポートされるテクノロジに加えて、モバイル ネイティブ アドオン ライセンスによって、Android と iOS 上でのネイティブ モバイル アプリケーション テストのサポートを提供します。

Silk4J

Silk4J によって、Java プログラム言語を使用して機能テストを作成することができるようになります。Silk4J は、Java ランタイム ライブラリを提供しており、そのライブラリには Silk4J がテストをサポートするすべてのクラスに対するテスト クラスが含まれています。このランタイム ライブラリは、JUnit と互換性があります。つまり、JUnit のインフラストラクチャを利用して Silk4J テストを実行することができます。また、すべての利用可能な Java ライブラリをテスト ケースで使用することもできます。

Silk4J がサポートするテスト環境は次のとおりです。

- モバイル Web アプリケーション
  - Android
  - iOS
- ネイティブ モバイル アプリケーション
  - Android
  - iOS
- Apache Flex
- Java AWT/Swing
- Java SWT
- Rumba
- SAP
- Microsoft Silverlight
- Windows API ベースのクライアント／サーバー (Win32)
- Windows Forms
- Windows Presentation Foundation (WPF)
- xBrowser (Web アプリケーション)

Web アプリケーション テストのサンプル スクリプトは、パブリックのドキュメント フォルダ (%PUBLIC%¥Documents¥SilkTest¥samples¥Silk4J) にあります。

注: Silk4J の起動時に開始画面を表示しないように選択している場合、**ヘルプ** > **製品更新の確認** をクリックして利用可能な更新を確認できます。

注: Silk4J を使用してキーワード駆動テストを実行する場合は、Eclipse プラットフォームを Java 7 以降で実行する必要があります。

Silk4J の実行に必要な管理者権限

Silk4J のインストールまたは実行には、次の権限が必要です。

- Silk4J のインストールには、ローカル管理者権限が必要です。
- Windows サーバー上への Silk4J のインストールには、ドメイン レベルの管理者権限が必要です。
- Silk4J を実行するには、次のフォルダとそのすべてのサブフォルダに対するフルアクセス権が必要です。
  - C:¥ProgramData¥Silk¥SilkTest
  - %APPDATA%¥Roaming¥Silk¥SilkTest
  - %APPDATA%¥Local¥Silk¥SilkTest
  - %TEMP%

Silk4J 使用法のベスト プラクティス

テスト対象アプリケーションとテスト環境によって、アプリケーションに対して機能テストや回帰テストを実行しようとするときに、さまざまな課題に直面することがあります。 Micro Focus では、次のベスト プラクティスを推奨します。

- Silk4J が提供する機能を最適に使用するには、同じテストで複数のアプリケーションをテストする場合を除き、テストするアプリケーションごとに個別のプロジェクトを作成します。
- 大規模なテスト フレームワークが整っているのであれば、キーワード駆動テスト手法を使用することを検討してください。
- Eclipse に最低 512MB のメモリを割り当ててください。eclipse.ini ファイルで xms の値として 512 以上を設定することで割り当てることができます。 このファイルは、Eclipse インストール ディレクトリにあります。
- テストするマシンの画面上のテキストやその他の項目の大きさをデフォルト値に設定してください。デフォルト以外の設定を使用すると、記録や再生時に座標が一致しない問題が発生する可能性があります。
    - Microsoft Windows 7 を使用している場合は、**スタート** > **コントロール パネル** > **デスクトップのカスタマイズ** > **ディスプレイ** でこの設定を変更できます。
    - Microsoft Windows 10 を使用している場合は、**設定** > **システム** > **ディスプレイ** > **テキスト、アプリ、その他の項目のサイズを変更する** でこの設定を変更できます。

特殊な状況下での自動化 (周辺機器が無い)

製品の基本的な位置付け

Silk4J は GUI テスト製品で、自動化された状況下での有意なテスト結果を得るために、人間のように振舞います。Silk4J が実行したテストは、人間が実行するよりもすばやく実行しますが、同等の価値のあるものです。このことは、人間が同じテストを実行するために必要なテスト環境とできる限り同等なテスト環境を Silk4J が必要とすることを意味します。

物理的な周辺機器

実際のアプリケーション UI の手動テストでは、キーボード、マウス、ディスプレイなどの入出力デバイスが必要です。Silk4J では、テストの再生時に物理的な入力デバイスを必要としません。Silk4J に必要なものは、キーストロークやマウス クリックを実行するオペレーティング システムの機能です。大抵の場合、入力デバイスが接続されていなくても、Silk4J の再生は期待通り動作します。ただし、デバイス ドライバによっては、物理的な入力デバイスが利用可能でないと、Silk4J の再生機構をブロックする場合があります。

同じことが物理的な出力デバイスについても言えます。物理的なディスプレイが接続されている必要はありませんが、機能するビデオ デバイス ドライバがインストールされ、オペレーティング システムが画面にレンダリングできる状態になければなりません。たとえば、スクリーン セーバー モードやセッションがロックされている状況では、レンダリングできません。レンダリングできない場合、低レベルの再生は機能せず、高レベルの再生もテスト対象アプリケーション (AUT) で使用するテクノロジに依存しますが、期待通り機能しない可能性があります。

仮想マシン

Silk4J は仮想化ベンダーを直接サポートしませんが、仮想ゲスト マシンが物理マシンと同等に動作する限り、任意の仮想化手法のもとで動作可能です。標準的な周辺機器は、通常は仮想デバイスとして提供されており、仮想マシンを実行するマシンで使用されている物理デバイスとは無関係です。

クラウド インスタンス

自動化の観点からは、クラウド インスタンスは仮想マシンと変わりありません。ただし、クラウド インスタンスでは、ビデオ レンダリングに特殊な最適化が行われている場合があり、ハードウェア リソースの消費を抑えるために、画面のレンダリングが一時的にオフになる状況があります。これは、ディスプレイを表示しているアクティブなクライアントが無いと、クラウド インスタンスが検知した場合に発生する場合があります。このような場合、回避策として VNC ウィンドウを開くことができます。

特殊な状況

ウィンドウが無く起動されるアプリケーション (ヘッドレス)

このようなアプリケーションは、Silk4J を使ってテストできません。Silk4J は、対象のアプリケーション プロセスにフックして、対話操作する必要があります。ウィンドウが表示されないプロセスをフックすることはできません。このような場合は、システム コマンドの実行のみ可能です。

リモート デスクトップ、ターミナル サービス、リモート アプリケーション (すべてのベンダー)

Silk4J がリモート デスクトップ セッション側に存在し、操作する場合、完全に期待通りの操作が行われます。

注: フル ユーザー セッションが必要で、リモート表示ウィンドウは最大化されている必要があります。リモート表示ウィンドウが何らかの理由で表示されていない場合 (ネットワーク上の問題など)、Silk4J は再生を続けますが、使用されているリモート表示技術によっては予期しない結果を生じる可能性があります。たとえば、リモート デスクトップ セッションが失われると、ビデオ レンダリングに悪影響を与えますが、他のリモート表示手法では、一度表示されたウィンドウが失われても、問題なく表示されるものもあります。

Silk4J がリモート デスクトップ、リモート ビュー、リモート アプリ ウィンドウなどとの対話操作に使用される場合は、Silk4J が見ることができるのはリモート マシンのスクリーンショットだけであるため、低レベルな技術だけが使用できます。リモート表示技術によっては、セキュリティ上の制約により、低レベル操作でさえできないものもあります。たとえば、リモート アプリケーション ウィンドウにキーストロークを送信できない場合があります。

既知の自動化の障壁

Silk4J では、ログオンした対話的なフル ユーザー セッションが必要です。スクリーン セーバー、休止状態、スリープ モードなどのセッションをロックするものは無効化してください。組織の方針などで、これができない場合は、キープ アライブ 操作 (定期的にあるいは各テスト ケースの終わりにマウスを動かすなど) を追加することによって、このような問題を回避できます。

注: 実際のテスト環境の構成や AUT、仮想化、ターミナル サービスで使用される技術によっては、テストの自動化プロセスにおいて、さらなる問題や制約に直面する可能性があります。

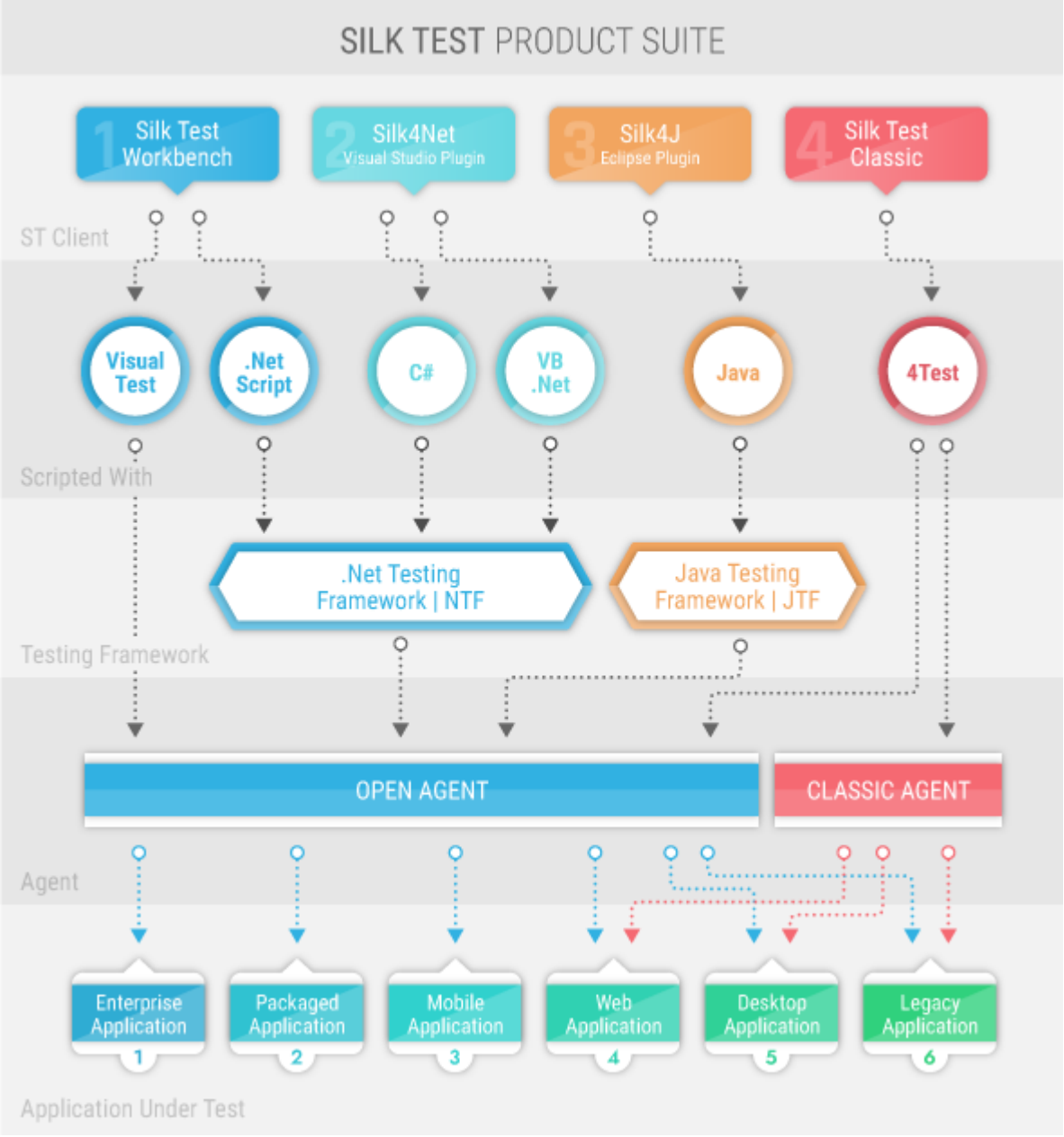
Silk Test 製品スイート

Silk Test は、高速で信頼性の高い機能テストと回帰テストを行うための自動テスト ツールです。Silk Test は、高品質のソフトウェアをすばやくリリースするために、開発チーム、品質管理チーム、ビジネス アナリストを支援します。Silk Test を使用すると、アプリケーションが意図したとおりに動作することを確実にするために、複数のプラットフォームとデバイス上でテストを記録/再生することができます。

Silk Test 製品スイートには、以下のコンポーネントが含まれています。

- Silk Test Workbench： Silk Test Workbench は、品質テスト環境です。上級者用の .NET スクリプトと、より幅広い利用者がテストを行えるようにする使いやすいビジュアル テストが提供されます。
- Silk4NET： Silk4NET Visual Studio プラグインを使用すると、Visual Studio で直接 VB.NET または C# のテスト スクリプトを作成できます。
- Silk4J： Silk4J Eclipse プラグインを使用すると、Eclipse 環境で直接 Java ベースのテスト スクリプトを作成できます。

- Silk Test Classic： Silk Test Classic は、4Test を使用したスクリプトを作成できる Silk Test クライアントです。
- Silk Test Agent： Silk Test Agent は、テストのコマンドを GUI 固有のコマンドに変換するソフトウェア プロセスです。つまり、テストするアプリケーションをエージェントが動かし、監視しています。ホストマシン上で 1 つのエージェントをローカルに実行できます。ネットワーク環境では、任意の数のエージェントをリモート マシン上で実行できます。



上の画像の個々のボックスのサイズは、記述上の理由で異なっているだけで、含まれる機能を反映しているわけではありません。

インストールする製品スイートによって、使用できるコンポーネントが決まります。すべてのコンポーネントをインストールするには、完全インストール オプションを選択します。Silk Test Classic を除くすべてのコンポーネントをインストールするには、標準インストール オプションを選択します。

Silk4J の新機能

Silk4J では、以下の新しい機能がサポートされています。

Mac 上でのクロス ブラウザー テスト

macOS 上の Mozilla Firefox または Google Chrome に対して、Web アプリケーションのテストを記録、再生できるようになりました。

テストの Docker 化

Silk4J テストを Docker で実行できる機能が追加され、継続的インテグレーション (CI) や継続的デプロイ (CD) パイプラインとの統合を Silk Test がサポートできるようになりました。

この Silk Test の新しい機能を使うと、アジャイル CI および CD プロセスに機能テストを組み込むことができるようになります。つまり、Jenkins、Bamboo、Team City などの CI サーバーから機能テストを自動的にトリガーすることができます。Docker 化した Silk Test 環境でテストを実行できるため、新しいテスト環境の追加とその準備が、容易かつ迅速に行えるようになりました。

記録のカスタマイズ

テスト対象アプリケーションのコントロールに対して分かりやすい識別子を作成することは、非常に重要です。Silk4J では高度な方法で、可能な限り分かりやすいロケーターやプロジェクト マップ エントリの名前を記録時に生成しますが、ユーザーの要件を 100% 満たすことは困難です。たとえば、生成したロケーターに期待する属性が使用されていなかったり、記録したオブジェクトに対して自動的に生成されたオブジェクト マップ エントリが分かりにくかったり、特殊な文字を含んでいることがあります。Silk4J 19.0 を使用すると、Web コントロールに対するロケーターやオブジェクト マップ エントリを、記録時に UI から直接編集できます。

この新しい機能は、Internet Explorer 以外のすべてのサポートするデスクトップおよびモバイル ブラウザーで利用できます。

ユーザビリティーの改善

このセクションでは、Silk Test 19.0 に対して行われたユーザビリティーの改善点の一覧を提供します。

並列テストのテスト実行

並列テストのテスト実行が、すべての Silk Test クライアントにおいてデフォルトで有効になりました。

Web ページ全体のスクリーンショットのキャプチャ

テスト中に Web ページの表示されている部分のスクリーンショットをキャプチャすることは今までても可能でした。 このバージョンでは、BrowserWindow クラスの captureFullPageBitmap メソッドが Silk Test に追加され、Web ページのすべてのコンテンツのスクリーンショットをキャプチャできるようになりました。

コマンド ラインからの Open Agent の停止

-shutDown オプションを指定して、コマンド ラインから Open Agent を停止できるようになりました。

ブラウザー テストの iframe および frame サポートの無効化

ブラウザー テストの再生パフォーマンスを改善するため、Web アプリケーションの iframe および frame のコンテンツがテスト対象ではない場合に、iframe および frame サポートを無効化できるようになりました。たとえば、多くの広告が表示される Web ページやモバイル ブラウザーでテストする際に iframe サポートを無効にすると、再生パフォーマンスが大幅に向上する可能性があります。Internet Explorer を除くすべてのブラウザーに対して iframe と frame サポートを無効化できます。

複数の UI セッションを使ったテストの改善

MicroFocus.SilkTest.MultiSessionLauncher.exe を使用する代わりに、次のコマンドを実行して、UI セッションから直接 Open Agent を起動できるようになりました。

```
openAgent.exe  -infoServicePort=<port>
```

API の改善

このセクションでは、Silk Test 19.0 に対して行われた API の改善点の一覧を提供します。

XPath の追加の軸のサポート

XPath の軸 (ancestor、preceding-sibling、following-sibling) のサポートが Silk Test に追加されました。 コントロールの階層的な位置の特定に役立ちます。

TrueLog セクションの追加による構造化

Silk4J または Silk4NET の使用時に、TrueLog にセクションを追加できるようになったため、複雑なスクリプトを、名前の付けられた小さなパーツに論理的に分割して構造化できます。

DOM オブジェクトのフォーカスの確認

指定した DOM オブジェクトにフォーカスが設定されているかどうかを、isFocused プロパティを使って確認できるようになりました。

モバイル テストの改善

Silk4J は、Android 上のネイティブ モバイル アプリケーションに対するトーストのテストをサポートするようになりました。

使用技術の更新

このセクションでは、Silk Test 19.0 に対して行われた重要な使用技術の更新をリストします。

Mozilla Firefox の新しいバージョン

Silk Test の前のバージョンでテストされた Mozilla Firefox のバージョンに加えて、Silk Test は Mozilla Firefox の次の新しいバージョンでテストされました。

- Mozilla Firefox 57
- Mozilla Firefox 58
- Mozilla Firefox 59
- Mozilla Firefox 60

注: このリストには、Silk Test 19.0 のリリースまでに Silk Test 19.0 でテストされた Mozilla Firefox の新しいバージョンが含まれます。Mozilla Firefox の新しいバージョンが、Silk Test 19.0 のリリースより後にリリースされた場合でも、ブラウザー側で互換性が保たれていれば、Silk Test 19.0 は新しいバージョンを使用してテストできます。

Google Chrome の新しいバージョン

Silk Test の前のバージョンでテストされた Google Chrome のバージョンに加えて、Silk Test は Google Chrome の次のバージョンでテストされました。

- Google Chrome 63
- Google Chrome 64
- Google Chrome 65
- Google Chrome 66
- Google Chrome 67

注: このリストには、Silk Test 19.0 のリリースまでに Silk Test 19.0 でテストされた Google Chrome のバージョンが含まれます。Google Chrome の新しいバージョンが Silk Test 19.0 のリリースより後にリリースされた場合でも、ブラウザー側で互換性が保たれていれば、Silk Test 19.0 は新しいバージョンを使用してテストできます。

Microsoft Edge の新しいバージョン

Silk Test の前のバージョンでテストされた Microsoft Edge のバージョンに加えて、Silk Test は、Microsoft Edge 42.17134（Windows 10 April 2018 Update の Microsoft Edge）を使用して、記録と再生がテストされました。

Eclipse の新しいバージョン

Silk4J は、Eclipse Oxygen 3a (4.7.3a) をサポートするようになりました。

Java 9 と Java 10 のサポート

Silk4J は、Java 9 や Java 10 ベースのアプリケーションのテストをサポートするようになりました。

MFC サポートの改善

Silk4J が MFC (Microsoft Foundation Class) コントロールに対して安定したロケーターを生成できるようになりました。

Silk4J クイック スタート チュートリアル

このチュートリアルでは、Silk4J を使用し、動的オブジェクト解決を用いた Web アプリケーションのテストが行えるよう、導入手順をステップ by ステップで提供します。動的オブジェクト解決により、オブジェクトを検索し識別する XPath クエリを使用した、テスト ケースの記述が可能になります。

重要: このチュートリアルでの作業をスムーズに完了させるには、Java および JUnit の基礎知識が必要となります。

説明をより簡潔にするため、本ガイドでは Silk4J がすでにインストールされており、*http://demo.borland.com/InsuranceWebExtJS/* から入手可能なサンプルの Insurance Company (保険会社) Web アプリケーションを使用することを前提にしています。

注: Silk4J を実行するには、ローカルの管理者権限を持っている必要があります。

Silk4J プロジェクトの作成

新規 Silk4J プロジェクト ウィザードを使用して Silk4J プロジェクトを作成する際、このウィザードには、**新規 Java プロジェクト** ウィザードを使用して Java プロジェクトを作成する際に利用できるオプションと同じものが含まれています。さらに、この Silk4J ウィザードでは、Java プロジェクトを自動的に Silk4J プロジェクトにします。

1. Eclipse ワークスペースで、次のステップのいずれかを行います：
   - Silk Test ツールバー アイコン の隣にある、ドロップダウン矢印をクリックし、**新規 Silk4J プロジェクト** を選択します。
   - **パッケージ エクスプローラ** で右クリックし、**新規** > **その他...** を選択します。Silk4J フォルダを展開し、**Silk4J プロジェクト** をダブルクリックします。
   - 既存の Eclipse の場所へ Silk4J をインストールまたは更新した場合には、**ファイル** > **新規** > **その他...** を選択します。Silk4J フォルダを展開し、**Silk4J プロジェクト** をダブルクリックします。

   新規 Silk4J プロジェクト ウィザードが開きます。
2. **プロジェクト名** テキスト ボックスに、プロジェクトの名前を入力します。
   たとえば、*Tutorial* と入力します。
3. キーワード駆動テストまたは Silk Central を使用した構成テストを実行したい場合で、有効な Silk Central ライセンスを持っているのであれば、**Silk Central に接続** チェック ボックスをオンにして、キーワード駆動テスト用に Silk Central への接続を設定します。
   Silk Central サーバーは、この新しいプロジェクトだけではなく、すべてのプロジェクトに対して設定されます。
4. **次へ** をクリックします。 **アプリケーションの選択** ページが開きます。
5. 現在のプロジェクトに対してアプリケーション構成が設定されていない場合、テストするアプリケーションの種類に対応するタブを選択します。
   - ブラウザで実行しない標準アプリケーションをテストする場合は、**Windows** タブを選択します。
   - Web アプリケーションまたはモバイル Web アプリケーションをテストする場合は、**Web** タブを選択します。
   - ネイティブ モバイル アプリケーションをテストする場合は、**モバイル** タブを選択します。
6. 標準アプリケーションをテストするには、現在のプロジェクトに対してアプリケーション構成が設定されていない場合は、リストからアプリケーションを選択します。

7. Web アプリケーションまたはモバイル Web アプリケーションをテストするには、現在のプロジェクトに対してアプリケーション構成が設定されていない場合は、リストからインストール済みのブラウザまたはモバイル ブラウザのうちの 1 つを選択します。
   a) **移動する URL の入力** テキスト ボックスに、開く Web ページを指定します。選択したブラウザのインスタンスが既に実行されている場合、**実行中のブラウザーの URL を使用する** をクリックして、実行中のブラウザ インスタンスに現在表示されている URL の記録を行うことができます。チュートリアルの場合、**Internet Explorer** を選択し、**移動する URL の入力** テキスト ボックスに *http://demo.borland.com/InsuranceWebExtJS/* を指定します。
   b) 省略可能：あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。
   たとえば、Apple Safari 上の Web アプリケーションを Apple iPhone 7 の画面と同じ大きさのブラウザー ウィンドウでテストするには、リストから **Apple iPhone 7** を選択します。
   c) 省略可能：ブラウザー ウィンドウの **向き** を選択します。
   d) 省略可能：**ブラウザー サイズの編集** をクリックすると、新しいブラウザー サイズを指定したり、**ブラウザー サイズ** リストに表示するブラウザー サイズを選択することができます。
8. 現在のプロジェクトに対してアプリケーション構成が設定されていない場合に、ネイティブ モバイル アプリケーション (アプリ) をテストするには：
   a) アプリをテストするモバイル デバイスをリストから選択します。
   b) **参照** をクリックしてアプリ ファイルを選択するか、アプリ ファイルへの完全パスを **モバイル アプリ ファイル** テキスト フィールドに入力します。
   このパスでは、Silk4J は HTTP および UNC 形式をサポートします。
   Silk4J は、モバイル デバイスまたはエミュレータ上に指定したアプリをインストールします。
9. **終了** をクリックします。 JRE システム ライブラリと必要な .jar ファイル（silktest-jtf-nodeps.jar と junit.jar）を含んだ、新しい Silk4J プロジェクトが作成されます。
10. 記録するテストのタイプを選択します。
   - 記録した操作をキーワードにまとめる場合は、**Silk4J キーワード駆動テスト** を選択します。
   - キーワードを作成せずにテストを記録する場合は、**Silk Test JUnit テスト** を選択します。

   チュートリアルでは、**Silk Test JUnit テスト** を選択します。
11. **はい** をクリックすると新しい Silk4J テストの記録が開始され、**いいえ** をクリックすると Eclipse ワークスペースに戻ります。
   チュートリアルでは、**いいえ** をクリックします。

Insurance Company Web アプリケーションのテストを記録する

Silk4J テストを作成する前に、Silk4J プロジェクトを作成する必要があります。

Insurance Company Web アプリケーション（*http://demo.borland.com/InsuranceWebExtJS/*）で **Agent Lookup** ページまで移動する新しいテストを記録します。テクノロジの種類ごとにテストを記録する方法やテスト アプリケーションを設定する方法の詳細な説明については、『Silk4J ユーザー ガイド』の「テストの作成」セクションを参照してください。

1. ツールバーで、**操作の記録** をクリックします。
2. 使用するブラウザーを選択します。
3. **記録** をクリックします。 テスト対象アプリケーションと **Silk Recorder** ウィンドウが開きます。Silk4J は基本状態を作成し、記録を開始します。
4. Insurance Company Web サイトでは、次のステップのいずれかを行います：
   a) **Select a Service or login** リスト ボックスから **Auto Quote** を選択します。**Automobile Instant Quote** ページが開きます。

b) 郵便番号と電子メール アドレスを適切なテキスト ボックスに入力し、自動車タイプをクリックして、**Next** をクリックします。
たとえば、郵便番号に 92121、電子メール アドレスに jsmith@gmail.com をそれぞれ入力し、自動車タイプとして Car を指定します。

c) 年齢を指定し、性別と運転履歴タイプをクリックして、**Next** をクリックします。
たとえば、年齢に 42 を入力し、性別と運転履歴タイプに Male および Good をそれぞれ指定します。

d) 製造年、車種、モデルを指定し、財務情報タイプをクリックして、**Next** をクリックします。
たとえば、製造年に 2010 と入力し、車種とモデルに Lexus および RX400 をそれぞれ指定し、財務情報タイプとして Lease を指定します。
指定した情報の概要が現れます。

e) 指定した **Zip Code** をポイントし、Ctrl+Alt を押して、スクリプトに検証を追加します。
表示されたどの情報に対しても、検証を追加することができます。
検証タイプの選択 ダイアログ ボックスが開きます。

f) プロパティの検証を作成するか、イメージ検証を作成するかを選択します。
チュートリアルの場合、**TestObject のプロパティの検証** を選択します。
プロパティの検証 ダイアログ ボックスが開きます。

g) **TextContents** チェック ボックスをオンにし、**OK** をクリックします。 検証操作が、郵便番号テキストに対するスクリプトに追加されます。

h) **Home** をクリックします。

各ステップに相当する操作が記録されました。

5. **停止** をクリックします。 **記録完了** ダイアログ ボックスが開きます。
6. **ソース フォルダ** フィールドは、選択したプロジェクトのソース ファイルの場所で、自動的に埋められています。別のソース フォルダを使用するには、**選択** をクリックし、使用するフォルダまで辿っていきます。
7. 省略可能：**パッケージ** テキスト ボックスに、パッケージ名を指定します。
たとえば、次のように入力します：com.example。
既存のパッケージを使用するには、**選択** をクリックし、使用するパッケージを選択します。
8. **テスト クラス** テキスト ボックスに、テスト クラスの名前を指定します。
たとえば、次のように入力します：AutoQuoteInput。
既存のクラスを使用するには、**選択** をクリックし、使用するクラスを選択します。
9. **テスト メソッド** テキスト ボックスに、テスト メソッドの名前を指定します。
たとえば、次のように入力します：autoQuote。

10**OK** をクリックします。

テストが期待通りの動作をするか確認するためにテストを再生します。必要な場合には変更をするために、テストを編集することも可能です。

Insurance Company Web アプリケーションのテストを再生する

1. パッケージ エクスプローラーで **Tutorial** プロジェクトを展開します。
2. **AutoQuoteInput** クラスを右クリックし、**実行** > **Silk4J テスト** を選択します。再生をサポートしている複数のブラウザーがマシンにインストールされている場合、**ブラウザーの選択** ダイアログ ボックスが開きます。
3. ブラウザーを選択して、 **実行** をクリックします。テストの実行が完了すると、**再生完了** ダイアログ ボックスが開きます。
4. **結果の検討** をクリックして、完了したテストの TrueLog を確認します。 この例では、テスト アプリケーションの **Zip Code** フィールドがクリーンでないため、検証は失敗します。

Silk4J プロジェクトの操作

このセクションでは、Silk4J プロジェクトの使用方法について説明します。

Silk4J プロジェクトには、Silk4J を使用してアプリケーションの機能をテストするために必要なリソースがすべて含まれています。

注: Silk4J が提供する機能を最適に使用するには、同じテストで複数のアプリケーションをテストする場合を除き、テストするアプリケーションごとに個別のプロジェクトを作成します。

Silk4J プロジェクトの作成

新規 Silk4J プロジェクト ウィザードを使用して Silk4J プロジェクトを作成する際、このウィザードには、**新規 Java プロジェクト** ウィザードを使用して Java プロジェクトを作成する際に利用できるオプションと同じものが含まれています。さらに、この Silk4J ウィザードでは、Java プロジェクトを自動的に Silk4J プロジェクトにします。

1. Eclipse ワークスペースで、次のステップのいずれかを行います：
   - Silk Test ツールバー アイコン の隣にある、ドロップダウン矢印をクリックし、**新規 Silk4J プロジェクト** を選択します。
   - **パッケージ エクスプローラ** で右クリックし、**新規** > **その他...** を選択します。Silk4J フォルダを展開し、**Silk4J プロジェクト** をダブルクリックします。
   - 既存の Eclipse の場所へ Silk4J をインストールまたは更新した場合には、**ファイル** > **新規** > **その他...** を選択します。Silk4J フォルダを展開し、**Silk4J プロジェクト** をダブルクリックします。

   新規 Silk4J プロジェクト ウィザードが開きます。
2. **プロジェクト名** テキスト ボックスに、プロジェクトの名前を入力します。
   たとえば、*Tutorial* と入力します。
3. キーワード駆動テストまたは Silk Central を使用した構成テストを実行したい場合で、有効な Silk Central ライセンスを持っているのであれば、**Silk Central に接続** チェック ボックスをオンにして、キーワード駆動テスト用に Silk Central への接続を設定します。

   Silk Central サーバーは、この新しいプロジェクトだけではなく、すべてのプロジェクトに対して設定されます。
4. **次へ** をクリックします。 **アプリケーションの選択** ページが開きます。
5. 現在のプロジェクトに対してアプリケーション構成が設定されていない場合、テストするアプリケーションの種類に対応するタブを選択します。
   - ブラウザで実行しない標準アプリケーションをテストする場合は、**Windows** タブを選択します。
   - Web アプリケーションまたはモバイル Web アプリケーションをテストする場合は、**Web** タブを選択します。
   - ネイティブ モバイル アプリケーションをテストする場合は、**モバイル** タブを選択します。
6. 標準アプリケーションをテストするには、現在のプロジェクトに対してアプリケーション構成が設定されていない場合は、リストからアプリケーションを選択します。
7. Web アプリケーションまたはモバイル Web アプリケーションをテストするには、現在のプロジェクトに対してアプリケーション構成が設定されていない場合は、リストからインストール済みのブラウザまたはモバイル ブラウザのうちの 1 つを選択します。

   a) **移動する URL の入力** テキスト ボックスに、開く Web ページを指定します。選択したブラウザのインスタンスが既に実行されている場合、**実行中のブラウザーの URL を使用する** をクリックして、実行中のブラウザ インスタンスに現在表示されている URL の記録を行うことができます。チュート

リアルの場合、**Internet Explorer** を選択し、**移動する URL の入力** テキスト ボックスに *http://demo.borland.com/InsuranceWebExtJS/* を指定します。

b) 省略可能：あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。
たとえば、Apple Safari 上の Web アプリケーションを Apple iPhone 7 の画面と同じ大きさのブラウザー ウィンドウでテストするには、リストから **Apple iPhone 7** を選択します。

c) 省略可能：ブラウザー ウィンドウの **向き** を選択します。

d) 省略可能：**ブラウザー サイズの編集** をクリックすると、新しいブラウザー サイズを指定したり、**ブラウザー サイズ** リストに表示するブラウザー サイズを選択することができます。

8. 現在のプロジェクトに対してアプリケーション構成が設定されていない場合に、ネイティブ モバイル アプリケーション (アプリ) をテストするには：
   a) アプリをテストするモバイル デバイスをリストから選択します。
   b) **参照** をクリックしてアプリ ファイルを選択するか、アプリ ファイルへの完全パスを **モバイル アプリ ファイル** テキスト フィールドに入力します。
   このパスでは、Silk4J は HTTP および UNC 形式をサポートします。
   Silk4J は、モバイル デバイスまたはエミュレータ上に指定したアプリをインストールします。
9. **終了** をクリックします。 JRE システム ライブラリと必要な .jar ファイル（silktest-jtf-nodeps.jar と junit.jar）を含んだ、新しい Silk4J プロジェクトが作成されます。
10. 記録するテストのタイプを選択します。
    - 記録した操作をキーワードにまとめる場合は、**Silk4J キーワード駆動テスト** を選択します。
    - キーワードを作成せずにテストを記録する場合は、**Silk Test JUnit テスト** を選択します。

    チュートリアルでは、**Silk Test JUnit テスト** を選択します。
11. **はい** をクリックすると新しい Silk4J テストの記録が開始され、**いいえ** をクリックすると Eclipse ワークスペースに戻ります。

    チュートリアルでは、**いいえ** をクリックします。

Silk4J プロジェクトのインポート

中央リポジトリや他のマシンにある Silk4J プロジェクトにアクセスする必要がある場合、そのプロジェクトを Eclipse ワークスペースにインポートすることができます。

1. Eclipse で、ワークスペースを新規作成します。詳細については、Eclipse のドキュメントを参照してください。
2. Eclipse メニューで、**ファイル** > **インポート** をクリックします。**インポート** ダイアログ ボックスが開きます。
3. ツリーで **General** ノードを展開します。
4. **既存プロジェクトをワークスペースへ** を選択します。
5. **次へ** をクリックします。**プロジェクトのインポート** ダイアログ ボックスが開きます。
6. **ルート・ディレクトリーの選択** をクリックします。
7. **参照** をクリックして、プロジェクトの場所を選択します。
8. **フォルダーの参照** ダイアログ ボックスで、**OK** をクリックします。
9. **プロジェクト** リスト ボックスで、インポートするプロジェクトをチェックします。
10. **プロジェクトのインポート** ダイアログ ボックスで、**終了** をクリックします。

選択したプロジェクトが、Eclipse ワークスペースにインポートされます。

テストの作成

Silk4J を使用して、オブジェクトを検索し識別する XPath クエリを使用したテストを作成します。通常、テストを作成するために、**新規 Silk4J テスト** ウィザードを使用します。最初のテスト メソッドを作成した後に、既存のテスト クラスにテスト メソッドを追加することもできます。

テスト クラスを作成するときに、Silk4J はアプリケーションの基本状態を自動的に作成します。アプリケーションの基本状態とは、各テストの実行開始前にアプリケーションに想定される既知の安定した状態です。アプリケーションは、各テストの実行が終了したあとに基本状態に戻る場合もあります。詳細については、「基本状態」を参照してください。

Web アプリケーションのテストの作成

Silk4J テストを作成する前に、Silk4J プロジェクトを作成する必要があります。

Web アプリケーションのテストを記録するには：

1. **パッケージ エクスプローラー** で、新しいテストを追加するプロジェクトを選択します。
2. ツールバーで、**操作の記録** をクリックします。
3. 使用するブラウザーを選択します。
4. 省略可能：あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。
5. 省略可能：ブラウザー ウィンドウの **向き** を選択します。
6. **記録** をクリックします。 テスト対象アプリケーションと **Silk Recorder** ウィンドウが開きます。 Silk4J は基本状態を作成し、記録を開始します。
7. 省略可能：Silk4J ロケーターの代わりに WebDriver ロケーターを記録するには、**Silk Recorder** で **WebDriver** をクリックします。
   この機能は、次のブラウザーに対する記録時に利用できます。
   - Microsoft Edge
   - Mozilla Firefox
   - Google Chrome
   - Apple Safari

   詳細については、「*Selenium WebDriver* の使用」を参照してください。
8. テスト対象アプリケーションで、テストする操作を実行します。
   記録中に利用可能な操作についての詳細は、「記録中に利用可能な操作」を参照してください。
9. **停止** をクリックします。 **記録完了** ダイアログ ボックスが開きます。
10. **ソース フォルダ** フィールドは、選択したプロジェクトのソース ファイルの場所で、自動的に埋められています。別のソース フォルダを使用するには、**選択** をクリックし、使用するフォルダまで辿っていきます。
11. 省略可能：**パッケージ** テキスト ボックスに、パッケージ名を指定します。
    たとえば、次のように入力します：com.example。
    既存のパッケージを使用するには、**選択** をクリックし、使用するパッケージを選択します。
12. **テスト クラス** テキスト ボックスに、テスト クラスの名前を指定します。
    たとえば、次のように入力します：AutoQuoteInput。
    既存のクラスを使用するには、**選択** をクリックし、使用するクラスを選択します。
13. **テスト メソッド** テキスト ボックスに、テスト メソッドの名前を指定します。
    たとえば、次のように入力します：autoQuote。

14ΩK をクリックします。

テストが期待通りの動作をするか確認するためにテストを再生します。必要な場合には変更をするために、テストを編集することも可能です。

標準アプリケーションのテストの作成

Silk4J テストを作成する前に、Silk4J プロジェクトを作成する必要があります。

標準アプリケーションのテストを記録するには：

1. **パッケージ エクスプローラー** で、新しいテストを追加するプロジェクトを選択します。
2. ツールバーで、**操作の記録** をクリックします。
3. テスト対象アプリケーションで、テストする操作を実行します。
   たとえば、アプリケーションのメニュー コマンドをテストする場合は、**ファイル** > **新規** のようなメニュー コマンドを選択します。記録中に利用可能な操作についての詳細は、「記録中に利用可能な操作」を参照してください。
4. **停止** をクリックします。 **記録完了** ダイアログ ボックスが開きます。
5. **ソース フォルダ** フィールドは、選択したプロジェクトのソース ファイルの場所で、自動的に埋められています。別のソース フォルダを使用するには、**選択** をクリックし、使用するフォルダまで辿っていきます。
6. 省略可能：**パッケージ** テキスト ボックスに、パッケージ名を指定します。
   たとえば、次のように入力します：com.example。
   既存のパッケージを使用するには、**選択** をクリックし、使用するパッケージを選択します。
7. **テスト クラス** テキスト ボックスに、テスト クラスの名前を指定します。
8. **テスト メソッド** テキスト ボックスに、テスト メソッドの名前を指定します。
9. **OK** をクリックします。

テストが期待通りの動作をするか確認するためにテストを再生します。必要な場合には変更をするために、テストを編集することも可能です。

モバイル Web アプリケーションのテストを作成する

Silk4J テストを作成する前に、Silk4J プロジェクトを作成する必要があります。

モバイル デバイス上のモバイル Web アプリケーションに対する新しいテストを記録するには：

1. **パッケージ エクスプローラー** で、新しいテストを追加するプロジェクトを選択します。
2. ツールバーで、**操作の記録** をクリックします。
3. **ブラウザーの選択** ダイアログ ボックスで、モバイル デバイス上のブラウザーを選択します。
4. **記録** をクリックします。
5. **記録** ウィンドウが開き、モバイル デバイスの画面が表示されます。画面上で、記録したい操作を実行します。
   a) 操作したいオブジェクトをクリックします。Silk4J は、オブジェクトのデフォルトの操作を実行します。デフォルトの操作がない場合や、テキストを挿入する場合は、**操作の選択** ダイアログ ボックスが開きます。
   b) 省略可能：デフォルトの操作以外のオブジェクトの操作を選択するには、オブジェクトを右クリックします。**操作の選択** ダイアログ ボックスが開きます。
   c) 省略可能：スワイプやジェスチャーを記録するには、マウス カーソルをクリックしてドラッグします。
   d) 省略可能：操作にパラメータある場合は、**操作の選択** ダイアログ ボックスのパラメータ フィールドにパラメータを入力します。

Silk4J は自動的にパラメータを検証します。

e) **OK** をクリックして、**操作の選択** ダイアログ ボックスを閉じます。Silk4J は、記録した操作にその操作を追加し、モバイル デバイスまたはエミュレータ上でそれを再生します。

詳細については、「モバイル デバイスの操作」を参照してください。

6. **停止** をクリックします。 **記録完了** ダイアログ ボックスが開きます。
7. **ソース フォルダ** フィールドは、選択したプロジェクトのソース ファイルの場所で、自動的に埋められています。別のソース フォルダを使用するには、**選択** をクリックし、使用するフォルダまで辿っていきます。
8. 省略可能：**パッケージ** テキスト ボックスに、パッケージ名を指定します。
   たとえば、次のように入力します：com.example。

   既存のパッケージを使用するには、**選択** をクリックし、使用するパッケージを選択します。
9. **テスト クラス** テキスト ボックスに、テスト クラスの名前を指定します。

   既存のクラスを使用するには、**選択** をクリックし、使用するクラスを選択します。

10テスト メソッド テキスト ボックスに、テスト メソッドの名前を指定します。

11OK をクリックします。

テストが期待通りの動作をするか確認するためにテストを再生します。必要な場合には変更をするために、テストを編集することも可能です。

モバイル ネイティブ アプリケーションのテストを作成する

Silk4J テストを作成する前に、Silk4J プロジェクトを作成する必要があります。

モバイル デバイス上のモバイル ネイティブ アプリケーション (アプリ) に対する新しいテストを記録するには：

1. **パッケージ エクスプローラー** で、新しいテストを追加するプロジェクトを選択します。
2. ツールバーで、**操作の記録** をクリックします。
3. **モバイル デバイスの選択** ダイアログ ボックスで、次のアクションを実行します。
   a) アプリをテストするモバイル デバイスをリストから選択します。
   b) **参照** をクリックしてアプリ ファイルを選択するか、アプリ ファイルへの完全パスを **モバイル アプリ ファイル** テキスト フィールドに入力します。

      このパスでは、Silk4J は HTTP および UNC 形式をサポートします。

      Silk4J は、モバイル デバイスまたはエミュレータ上に指定したアプリをインストールします。
4. **記録** をクリックします。
5. **記録** ウィンドウが開き、モバイル デバイスの画面が表示されます。画面上で、記録したい操作を実行します。
   a) 操作したいオブジェクトをクリックします。Silk4J は、オブジェクトのデフォルトの操作を実行します。デフォルトの操作がない場合や、テキストを挿入する場合は、**操作の選択** ダイアログ ボックスが開きます。
   b) 省略可能：デフォルトの操作以外のオブジェクトの操作を選択するには、オブジェクトを右クリックします。**操作の選択** ダイアログ ボックスが開きます。
   c) 省略可能：スワイプやジェスチャーを記録するには、マウス カーソルをクリックしてドラッグします。
   d) 省略可能：操作にパラメータある場合は、**操作の選択** ダイアログ ボックスのパラメータ フィールドにパラメータを入力します。

      Silk4J は自動的にパラメータを検証します。
   e) **OK** をクリックして、**操作の選択** ダイアログ ボックスを閉じます。Silk4J は、記録した操作にその操作を追加し、モバイル デバイスまたはエミュレータ上でそれを再生します。

   詳細については、「モバイル デバイスの操作」を参照してください。

6. **停止** をクリックします。 **記録完了** ダイアログ ボックスが開きます。
7. **ソース フォルダ** フィールドは、選択したプロジェクトのソース ファイルの場所で、自動的に埋められています。別のソース フォルダを使用するには、**選択** をクリックし、使用するフォルダまで辿っていきます。
8. 省略可能：**パッケージ** テキスト ボックスに、パッケージ名を指定します。
   たとえば、次のように入力します：com.example。
   既存のパッケージを使用するには、**選択** をクリックし、使用するパッケージを選択します。
9. **テスト クラス** テキスト ボックスに、テスト クラスの名前を指定します。
   既存のクラスを使用するには、**選択** をクリックし、使用するクラスを選択します。
10. **テスト メソッド** テキスト ボックスに、テスト メソッドの名前を指定します。
11. **OK** をクリックします。

テストが期待通りの動作をするか確認するためにテストを再生します。必要な場合には変更をするために、テストを編集することも可能です。

Microsoft Edge 上でのテストの記録

Silk4J テストを記録する前に、Silk4J プロジェクトを作成する必要があります。

Microsoft Edge 上の Web アプリケーションとのやり取りを開始するとき、Silk4J は Microsoft Edge のすべての開いているインスタンスを閉じ、新しくブラウザーを開始します。新しいブラウザーは、アドオン無しのキャッシュを空にした状態の一時プロファイルを使用します。この Microsoft Edge のインスタンスは、Open Agent のシャットダウン時または、Microsoft Edge 外のほかのアプリケーションのテストを開始するときに閉じられます。

注: 現時点では、Microsoft Edge でキーワード駆動テストを記録することはできません。

Microsoft Edge 上の Web アプリケーションに対して新しいテストを記録するには：

1. 新しいテストを追加するプロジェクトを選択します。
2. ツールバーで、**操作の記録** をクリックします。
3. **ブラウザーの選択** ダイアログ ボックスで、使用するブラウザーを選択します。
4. 省略可能：あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。
5. 省略可能：ブラウザー ウィンドウの **向き** を選択します。
6. 省略可能：Silk4J ロケーターの代わりに WebDriver ロケーターを記録するには、**Silk Recorder** で **WebDriver** をクリックします。
   この機能は、次のブラウザーに対する記録時に利用できます。
   - Microsoft Edge
   - Mozilla Firefox
   - Google Chrome
   - Apple Safari

   詳細については、「*Selenium WebDriver* の使用」を参照してください。
7. **記録** をクリックします。
8. **記録** ウィンドウが開き、Web アプリケーションを表示します。記録したい操作を実行します。
   a) 操作したいオブジェクトをクリックします。Silk4J は、オブジェクトのデフォルトの操作を実行します。デフォルトの操作がない場合や、テキストを挿入したりパラメータを指定する場合は、**操作の選択** ダイアログ ボックスが開きます。
   b) 省略可能：デフォルトの操作以外のオブジェクトの操作を選択するには、オブジェクトを右クリックします。**操作の選択** ダイアログ ボックスが開きます。

c) 省略可能：操作にパラメータある場合は、パラメータ フィールドにパラメータを入力します。
Silk4J は自動的にパラメータを検証します。
d) **OK** をクリックして、**操作の選択** ダイアログ ボックスを閉じます。Silk4J は、記録した操作にその操作を追加し、モバイル デバイスまたはエミュレータ上でそれを再生します。

記録中、Silk4J は記録ウィンドウの隣にマウスの位置を表示します。その表示をクリックすると、デバイス画面に絶対的な位置とアクティブ オブジェクトに相対的な位置を切り替えることができます。記録中に利用可能なアクションについての詳細は、「記録中に利用可能なアクション」を参照してください。

9. **停止** をクリックします。 **記録完了** ダイアログ ボックスが開きます。
10. **ソース フォルダ** フィールドは、選択したプロジェクトのソース ファイルの場所で、自動的に埋められています。別のソース フォルダを使用するには、**選択** をクリックし、使用するフォルダまで辿っていきます。
11. 省略可能：**パッケージ** テキスト ボックスに、パッケージ名を指定します。
たとえば、次のように入力します：com.example。
既存のパッケージを使用するには、**選択** をクリックし、使用するパッケージを選択します。
12. **テスト クラス** テキスト ボックスに、テスト クラスの名前を指定します。
既存のクラスを使用するには、**選択** をクリックし、使用するクラスを選択します。
13. **テスト メソッド** テキスト ボックスに、テスト メソッドの名前を指定します。
14. **OK** をクリックします。

テストが期待通りの動作をするか確認するためにテストを再生します。必要な場合には変更をするために、テストを編集することも可能です。

Mozilla Firefox 上でのテストの記録

Silk4J テストを記録する前に、Silk4J プロジェクトを作成する必要があります。

Mozilla Firefox 上の Web アプリケーションに対して新しいテストを記録するには：

1. 新しいテストを追加するプロジェクトを選択します。
2. ツールバーで、**操作の記録** をクリックします。
3. **ブラウザーの選択** ダイアログ ボックスで、使用するブラウザーを選択します。
4. 省略可能：あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。
5. 省略可能：ブラウザー ウィンドウの **向き** を選択します。
6. 省略可能：Silk4J ロケーターの代わりに WebDriver ロケーターを記録するには、**Silk Recorder** で **WebDriver** をクリックします。
この機能は、次のブラウザーに対する記録時に利用できます。
   - Microsoft Edge
   - Mozilla Firefox
   - Google Chrome
   - Apple Safari

   詳細については、「*Selenium WebDriver* の使用」を参照してください。
7. **記録** をクリックします。
8. **記録** ウィンドウが開き、Web アプリケーションを表示します。記録したい操作を実行します。
   a) 操作したいオブジェクトをクリックします。Silk4J は、オブジェクトのデフォルトの操作を実行します。デフォルトの操作がない場合や、テキストを挿入したりパラメータを指定する場合は、**操作の選択** ダイアログ ボックスが開きます。

b) 省略可能 : デフォルトの操作以外のオブジェクトの操作を選択するには、オブジェクトを右クリックします。**操作の選択** ダイアログ ボックスが開きます。
c) 省略可能：操作にパラメータある場合は、パラメータ フィールドにパラメータを入力します。
Silk4J は自動的にパラメータを検証します。
d) **OK** をクリックして、**操作の選択** ダイアログ ボックスを閉じます。Silk4J は、記録した操作にその操作を追加し、モバイル デバイスまたはエミュレータ上でそれを再生します。

記録中、Silk4J は記録ウィンドウの隣にマウスの位置を表示します。その表示をクリックすると、デバイス画面に絶対的な位置とアクティブ オブジェクトに相対的な位置を切り替えることができます。記録中に利用可能なアクションについての詳細は、「記録中に利用可能なアクション」を参照してください。

9. **停止** をクリックします。 **記録完了** ダイアログ ボックスが開きます。
10. **ソース フォルダ** フィールドは、選択したプロジェクトのソース ファイルの場所で、自動的に埋められています。別のソース フォルダを使用するには、**選択** をクリックし、使用するフォルダまで辿っていきます。
11. 省略可能：**パッケージ** テキスト ボックスに、パッケージ名を指定します。
たとえば、次のように入力します：com.example。
既存のパッケージを使用するには、**選択** をクリックし、使用するパッケージを選択します。
12. **テスト クラス** テキスト ボックスに、テスト クラスの名前を指定します。
既存のクラスを使用するには、**選択** をクリックし、使用するクラスを選択します。
13. **テスト メソッド** テキスト ボックスに、テスト メソッドの名前を指定します。
14. **OK** をクリックします。

テストが期待通りの動作をするか確認するためにテストを再生します。必要な場合には変更をするために、テストを編集することも可能です。

Google Chrome 上でのテストの記録

Silk4J は、Google Chrome 50 以降でのテストの記録をサポートします。Google Chrome のそれ以前のバージョンでは、Silk4J はテストの再生とロケーターの記録のみをサポートします。

Silk4J テストを記録する前に、Silk4J プロジェクトを作成する必要があります。

注: バージョン 50 より前のバージョンの Google Chrome でテストを記録することはできません。

Google Chrome 50 以降の Web アプリケーションに対して新しいテストを記録するには :

1. 新しいテストを追加するプロジェクトを選択します。
2. ツールバーで、**操作の記録** をクリックします。
3. **ブラウザーの選択** ダイアログ ボックスで、使用するブラウザーを選択します。
4. 省略可能 : あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。
5. 省略可能：ブラウザー ウィンドウの **向き** を選択します。
6. 省略可能：Silk4J ロケーターの代わりに WebDriver ロケーターを記録するには、**Silk Recorder** で **WebDriver** をクリックします。
この機能は、次のブラウザーに対する記録時に利用できます。

   - Microsoft Edge
   - Mozilla Firefox
   - Google Chrome
   - Apple Safari

   詳細については、「*Selenium WebDriver* の使用」を参照してください。

7. **記録** をクリックします。
8. **記録** ウィンドウが開き、Web アプリケーションを表示します。記録したい操作を実行します。
   a) 操作したいオブジェクトをクリックします。Silk4J は、オブジェクトのデフォルトの操作を実行します。デフォルトの操作がない場合や、テキストを挿入したりパラメータを指定する場合は、**操作の選択** ダイアログ ボックスが開きます。
   b) 省略可能 : デフォルトの操作以外のオブジェクトの操作を選択するには、オブジェクトを右クリックします。**操作の選択** ダイアログ ボックスが開きます。
   c) 省略可能：操作にパラメータある場合は、パラメータ フィールドにパラメータを入力します。
   Silk4J は自動的にパラメータを検証します。
   d) **OK** をクリックして、**操作の選択** ダイアログ ボックスを閉じます。Silk4J は、記録した操作にその操作を追加し、モバイル デバイスまたはエミュレータ上でそれを再生します。

   記録中、Silk4J は記録ウィンドウの隣にマウスの位置を表示します。その表示をクリックすると、デバイス画面に絶対的な位置とアクティブ オブジェクトに相対的な位置を切り替えることができます。記録中に利用可能なアクションについての詳細は、「記録中に利用可能なアクション」を参照してください。
9. **停止** をクリックします。 **記録完了** ダイアログ ボックスが開きます。
10. **ソース フォルダ** フィールドは、選択したプロジェクトのソース ファイルの場所で、自動的に埋められています。別のソース フォルダを使用するには、**選択** をクリックし、使用するフォルダまで辿っていきます。
11. 省略可能：**パッケージ** テキスト ボックスに、パッケージ名を指定します。
    たとえば、次のように入力します：com.example。
    既存のパッケージを使用するには、**選択** をクリックし、使用するパッケージを選択します。
12. **テスト クラス** テキスト ボックスに、テスト クラスの名前を指定します。
    既存のクラスを使用するには、**選択** をクリックし、使用するクラスを選択します。
13. **テスト メソッド** テキスト ボックスに、テスト メソッドの名前を指定します。
14. **OK** をクリックします。

テストが期待通りの動作をするか確認するためにテストを再生します。必要な場合には変更をするために、テストを編集することも可能です。

テスト ケースを手動で作成する

通常は、**基本状態** ウィザードを使用して、Silk4J のテスト ケースを作成します。テスト ケースを手動で作成したい場合は、この手順を使用します。

1. **ファイル** > **新規** > **JUnit テスト ケース** を選択します。**新規 JUnit テスト ケース** ダイアログ ボックスが開きます。
2. **新規 JUnit 4 テスト** オプションが選択されていることを確認します。このオプションがデフォルトで選択されています。
3. **パッケージ** テキスト ボックスに、パッケージ名を指定します。
   デフォルトでは、このテキスト ボックスに最も最近使用されたパッケージが表示されています。デフォルトのパッケージを使用したくない場合には、次のステップのいずれかを選択します：
   - パッケージをまだ作成していない場合は、パッケージ名をテキスト ボックスに入力します。
   - すでにパッケージを作成済みの場合は、**参照** をクリックし、そのパッケージの場所に移動して、パッケージを選択します。
4. **名前** テキスト ボックスには、テスト ケースの名前を指定します。
5. **終了** をクリックします。以下のようなコードの新しいクラス ファイルが開きます。

```
package  com.borland.demo;

public  class  DynamicObjectRecognitionDemo  {
```

```
}
```

ここで、com.borland.demo は、指定したパッケージで、DynamicObjectRecognitionDemo は、指定したクラスです。

基本状態を作成するか、または attach メソッドを使用して、テスト アプリケーションに接続します。

テスト スクリプト作成のベスト プラクティス

テスト ケースの書き方によっては、テスト セットのパフォーマンスと安定性に非常に大きな影響を与える場合があります。記録時に、可能な限り速く安定したスクリプトを Silk4J は作成します。しかし、テスト スクリプトを手動で作成したり編集する必要が生じる場合もあります。このトピックでは、いくつかの一般的なガイドラインを提示します。このガイドラインに従うことによって、保守性、再利用性の高いテスト スクリプトの作成、およびテストの安定性が高まるでしょう。

- テストの名前の一貫性を保ち、内容のわかりやすい名前をつけてください。テスト対象アプリケーションとテストの機能を表す名前にすると良いでしょう。たとえば、テストの名前を *MyApp_SuccessfulLogin* や *MyApp_FailingLogin* とすると、*Untitled_42* や *Untitled_43* とするよりも、他のユーザーにとって非常にわかりやすいものになります。
- できるだけ細かくテスト ケースの説明をコメントに記述してください。テスト ケースについての説明が普通の言葉で記述されていないと、実装されたコードを変更する必要がある人が、そのテストが正確に何を行っているのかを理解することが困難になります。
- テスト ケースを開始する時に、テスト対象アプリケーションが確実に適切な状態になるようにしてください。テスト ケースの操作を実行する前に、テスト対象アプリケーションを正しい状態に戻します。
- テスト ケースを完了する時に、テスト対象アプリケーションが確実に適切な状態になるようにしてください。追加のテストがそのテストの結果に依存する場合は、それらが開始できる状態にします。テスト ケースの操作が実行される時に、テスト対象アプリケーションを正しい状態に戻します。
- 可能な限り､テスト ケースは他のテスト ケースの結果に依存しないようにしてください。これが不可能な場合、テスト ケースが正しい順番で実行されるようにしてください。
- 機能的な流れだけでなく、テスト対象アプリケーションの正確さをテストするために、テストに検証を追加してください。
- キーワード駆動テストを使用して、操作セットの再利用性を高めてください。共通に使用する操作をキーワードにまとめ、頻繁に同じ順番で実行されるキーワードをキーワード シーケンスに結合し、キーワードやキーワード シーケンスの組み合わせをキーワード駆動テストとして実行します。
- テストの保守性と再利用性を保つためには、複雑なテスト ケースを記述するよりも、複数の単純なテスト ケースを記述して組み合わせる方が好まれます。
- テスト セットの冗長性を避けるためには、新しいテスト ケースを追加するよりも、既存のテスト ケースを更新することが好まれます。

記録中に利用可能なアクション

記録中に次のアクションを **記録中** ウィンドウで実行できます。

アクション	ステップ
記録の一時停止	**一時停止** をクリックして、操作を記録せずに AUT を特定の状態にしてから、**記録** をクリックして記録を再開できます。
記録した操作の順番の変更	**記録中** ウィンドウで記録した操作の順番を変更するには、移動したい操作を選択して新しい場所にそれらをドラッグします。

アクション	ステップ
複数の操作の選択	複数の操作を選択するには、Ctrl を押しながら操作をクリックするか、もしくは選択する最初の操作をクリックしてから、**Shift** を押しながら最後のアクションをクリックします。
記録した操作の再生	**記録中** ウィンドウから記録した操作を再生するには、操作を選択してから **再生** をクリックします。すべての記録した操作を再生するには、**記録した操作** をクリックしてから **再生** をクリックします。
記録した操作の削除	誤って記録した操作を **記録中** ウィンドウから削除するには、マウス カーソルをその操作上にポイントすると表示される **削除** をクリックします。
イメージまたはコントロールのプロパティの検証	検証するオブジェクトの上にマウス カーソルを移動して、**Ctrl+Alt** を押します。
オブジェクト マップ エントリの変更	Web アプリケーション、またはモバイル Web アプリを記録しているときに、記録したオブジェクトに対して自動的に生成されたオブジェクト マップ エントリが分かりにくい場合や、特殊な文字が含まれてしまっている場合、分かりやすいオブジェクト マップ エントリに変更したいことがあります。この様な場合は、記録中にオブジェクトを右クリックして、**操作の選択** ダイアログの **オブジェクトの識別** 領域を展開します。そして、**オブジェクト マップ ID** フィールドでオブジェクト マップ エントリを編集できます。たとえば、デモ アプリケーションの画像に対して自動的に生成されるオブジェクト マップ エントリは、「*http demo borland*」になります。このエントリが何を参照しているのかを理解するのは、オブジェクト マップを見ただけでは困難です。このオブジェクト マップ エントリを「*InsuranceWebHomePageBanner*」のように変更すれば、より分かりやすくなります。この機能は、Internet Explorer 以外のすべてのサポートするデスクトップおよびモバイル ブラウザで利用できます。
別のロケーターの選択	Web アプリケーション、またはモバイル Web アプリに対して記録しているときに、記録したオブジェクトに対して自動的に生成されたロケーターが要求を満たしていない場合、**ロケーター** フィールドにある矢印をクリックして、Silk4J に他のロケーターを提示させることができます。提示されるロケーターはすべて、オブジェクトを一意に識別できるものです。この機能は、Internet Explorer 以外のすべてのサポートするデスクトップおよびモバイル ブラウザで利用できます。

記録中のスクリプトへの検証の追加

以下の操作を行って、スクリプトの記録中に検証を追加します。

1. 記録を開始します。
2. 検証するオブジェクトの上にマウス カーソルを移動して、**Ctrl+Alt** を押します。

   モバイル Web アプリケーションを記録する場合は、オブジェクトをクリックして **検証の追加** をクリックすることもできます。

   このオプションを実行すると、記録が一時的に停止され、**検証タイプの選択** ダイアログ ボックスが表示されます。
3. **TestObject のプロパティの検証** を選択します。

   イメージ検証をスクリプトに追加する方法については、記録中にイメージ検証を追加する を参照してください。
4. **OK** をクリックします。 **プロパティの検証** ダイアログ ボックスが開きます。
5. 検証したいプロパティを選択するには、対応するチェック ボックスをオンにします。
6. **OK** をクリックします。 Silk4J は記録したスクリプトに検証を追加し、記録は続行されます。

Locator Spy を使用したロケーターまたはオブジェクト マップ項目のテスト メソッドへの追加

Locator Spy を使用して、ロケーターまたはオブジェクト マップ項目を手動でキャプチャし、ロケーターまたはオブジェクト マップ項目をテスト メソッドにコピーします。たとえば、**Locator Spy** を使って、GUI オブジェクトのキャプションや XPath ロケーター文字列を識別できます。そして、関係するロケーター文字列や属性をスクリプト内のテストメソッドにコピーします。

1. 変更したいテスト クラスを開きます。
2. Silk4J ツール バーで、**Locator Spy** をクリックします。**Locator Spy** とテスト対象アプリケーションが開きます。モバイル アプリケーションをテストしている場合、モバイルデバイスの画面を表示する [モバイルの記録] ウィンドウが開きます。この記録ウィンドウで操作を実行することはできませんが、モバイル デバイスやエミュレータ上で操作を実行してから、記録ウィンドウの表示を更新することができます。
3. 省略可能：ロケーターを記録する前に、テスト対象アプリケーションを特定の状態にするには、**ロケーターの記録を停止** をクリックします。これ以降、テスト対象アプリケーションで実行する操作は、記録されなくなります。ロケーターの記録を再開する場合は、**ロケーターの記録を開始** をクリックします。
4. Microsoft Edge、Mozilla Firefox、Google Chrome、Apple Safari 上で Web アプリケーションをテストしている場合は、**記録モード** を選択できます。
   - Silk4J モードでロケーターを記録する場合は、**Silk Test ロケーターの記録** を選択します。
   - WebDriver モードでロケーターを記録する場合は、**WebDriver ロケーターの記録** を選択します。

   注: WebDriver ロケーターを記録する場合、オブジェクト ツリー上の「>」は、ある IFrame から他に切り替わることを意味します。
5. 省略可能：オブジェクト マップ項目の代わりにロケーターを **ロケーター** 列に表示するには、**オブジェクト マップ識別子の表示** チェック ボックスをオフにします。

   この設定は、WebDriver ロケーターの記録時には利用できません。オブジェクト マップ項目名は、コントロールまたはウィンドウに対して、コントロールやウィンドウのロケーターではなく論理名 (エイリアス) を関連付けます。デフォルトでは、オブジェクト マップ項目名が表示されます。

   注: このチェック ボックスをオンまたはオフにすると、変更が自動的にロケーターの詳細に反映されます。**ロケーターの詳細** テーブルのエントリを更新するには、エントリをクリックします。
6. 記録するオブジェクトの上にマウスを移動します。関連するロケーター文字列またはオブジェクト マップ項目は、**選択済みロケーター** テキスト ボックスに表示されます。

   注: ブラウザーでテストしている場合、**選択済みロケーター** フィールドに実際にキャプチャするとロケーターのみが表示される。
7. **Ctrl+Alt** を押してオブジェクトをキャプチャします。

   注: スクリプト オプション ダイアログ ボックスの **全般記録オプション** ページで別の記録停止キー操作を指定した場合は、**Ctrl+Shift** を押してオブジェクトをキャプチャします。
8. 省略可能：**追加のロケーター属性の表示** をクリックすると、関係する属性のすべてが **ロケーター属性** テーブルに表示されます。
9. 省略可能：記録したロケーター属性は、**ロケーター属性** テーブルの別のロケーター属性で置き換えることができます。

   たとえば、記録したロケーターは以下のように表示されます。

   ```
   /BrowserApplication//BrowserWindow//input[@id='loginButton']
   ```

   ロケーター属性 テーブルに textContents Login がリストされている場合、以下のようにしてロケーターを手動で変更できます。

   ```
   /BrowserApplication//BrowserWindow//input[@textContents='Login']
   ```

新しいロケーターは、**選択済みロケーター** テキスト ボックスに表示されます。

10. WebDriver ロケーターを記録している場合は、**選択したロケーター** でロケーターの種類を選択します。
    - XPath ロケーター。コントロール クラスの名前と優先付けられた属性のコレクションの組み合わせによって、コントロールをユニークなロケーターに識別します。組み合わせたロケーターでコントロールをユニークに識別できない場合には、Silk4J は、ロケーターにインデックスを追加したり、「//」を使用して親の UI コントロールを前に付け足します。
    - ID による識別。 id 属性によってコントロールを識別します。
    - 名前による識別。name 属性によってコントロールを識別します。
    - リンク テキストによる識別。ハイパーリンクの場合にのみ選択できます。
11. ロケーターをコピーするには、**ロケーターをクリップボードにコピー** をクリックします。

    選択済みロケーター テキスト ボックスで、コピーするロケーター文字列の位置をマークし、マークしたテキストを右クリックして **コピー** をクリックすることもできます。
12. スクリプト内で、記録したロケーターを貼り付ける位置にカーソルを置きます。

    たとえば、スクリプト内の Find メソッドの該当するパラメータにカーソルを置きます。

    ロケーターを貼り付けるテスト メソッドでは、ロケーターをパラメータとして受け取れるメソッドを使用する必要があります。**Locator Spy** を使用することで、クエリ文字列が正しいことが保障されます。
13. ロケーターまたはオブジェクト マップ項目をテスト ケースまたはクリップボードにコピーします。
14. **閉じる** をクリックします。

テストにカスタム属性を含める

テストにカスタム属性を含めると、テストをより安定させることができます。 たとえば、Java SWT では、GUI を実装する開発者が silkTestAutomationId のような属性をウィジェットに対して定義することによって、アプリケーション内でそのウィジェットを一意に識別することができます。 これにより、Silk4J を使用するテスト担当者は、その属性（この場合は silkTestAutomationId）をカスタム属性のリストに追加すると、その一意の ID によってコントロールを識別できるようになります。

一意の ID を使用すると、caption や index のような他の属性よりも高い信頼性を得ることができます。これは、caption はアプリケーションを他の言語に翻訳した場合に変更され、index は定義済みのウィジェットより前に他のウィジェットが追加されると変更されるためです。

注: Flex または Windows API ベースのクライアント／サーバー (Win32) アプリケーションには、カスタム属性を設定できません。

カスタム属性をテストに含めるには、作成したテストに直接カスタム属性を含めます。
たとえば、アプリケーション内で、一意の ID 'loginName' が入力されている最初のテキスト ボックスを検索するには、以下のクエリを使用します。

```
myWindow.find(".//TextField[@silkTestAutomationId='loginName']")
```

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード？および * をサポートしています。

たとえば、Web アプリケーションで「bcauid」という属性を追加するには、以下のように入力します。

```
 <input  type='button'  bcauid='abc'
value='click  me'  />
```

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード？および * をサポートしています。

記録中および再生中に除外される文字

記録および再生中に Silk Test が無視する文字を以下に示します。

文字	コントロール
...	MenuItem
タブ	MenuItem
&	すべてのコントロール。 アンパサンド (&) はアクセラレータとして使用されるため、記録されません。

テストの再生

Eclipse 内から、あるいはコマンド ラインを使用してテスト メソッドを実行します。

Eclipse からのテストの再生

1. 再生するテスト メソッドまたはキーワード駆動テストに移動します。
2. 次のいずれか 1 つのステップを行います：
   - パッケージ内のすべてのテスト メソッドまたはキーワード駆動テストを再生するには、**パッケージ エクスプローラー** で、パッケージ名を右クリックします。
   - クラス内のすべてのテスト メソッドを再生するには、**パッケージ エクスプローラー**で、クラス名を右クリックします。または、ソース エディタでそのクラスを開き、ソース エディタで右クリックします。
   - キーワード駆動テストを再生するには、**パッケージ エクスプローラー** で、キーワード駆動テストの名前を右クリックします。
   - 特定のメソッドのみのテストを再生するには、**パッケージ エクスプローラー** で、メソッド名を右クリックします。または、ソース エディタでそのクラスを開き、テスト メソッドの名前をクリックして選択してから右クリックします。
3. **実行** > **Silk4J テスト** を選択します。

   注: 実行 > **JUnit テスト** を選択すると、多くの Silk4J 機能が無効になるため、利用できません。
4. Web アプリケーションをテストする場合は、**ブラウザーの選択** ダイアログ ボックスが開きます。ブラウザーを選択して、 **実行** をクリックします。

   注: 複数のアプリケーションが現在のプロジェクトに対して設定されている場合、**ブラウザーの選択** ダイアログ ボックスは表示されません。
5. 省略可能：必要に応じて、両方の **Shift** キーを同時に押して、テストの実行を停止できます。
6. テストの実行が完了すると、**再生完了** ダイアログ ボックスが開きます。**結果の検討** をクリックして、完了したテストの TrueLog を確認します。

コマンド ラインからのテストの再生

このタスクを実行する前に、JDK の場所を参照できるように PATH 変数を更新する必要があります。詳細については、『*JDK Installation for Microsoft Windows*』を参照してください。

1. CLASSPATH に以下を含めます。
   - junit.jar
   - org.hamcrest.core JAR ファイル
   - silktest-jtf-nodeps.jar
   - テストを含んだフォルダの JAR

   ```
   set CLASSPATH=<eclipse_install_directory>¥plugins
   ¥org.junit4_4.3.1¥junit.jar;<eclipse_install_directory>¥plugins
   ¥org.hamcrest.core_1.3.0.v201303031735.jar;%OPEN_AGENT_HOME%¥JTF¥silktest-jtf-
   nodeps.jar;C:¥myTests.jar
   ```
2. 次のように入力して JUnit テスト メソッドを実行します：

   ```
   java org.junit.runner.JUnitCore <test class name>
   ```

注: トラブル シューティングの情報については、次の JUnit のドキュメントを参照してください : *http://junit.sourceforge.net/doc/faq/faq.htm#running_1* を設定する必要がなくなりました。

3. Silk4J を使用していくつかのテスト クラスを実行して TrueLog を作成するには、SilkTestSuite クラスを使用して Silk4J テストを実行します。
   たとえば、2 つのクラス *MyTestClass1* と *MyTestClass2* を TrueLog を有効にして実行するには、次のコードをスクリプトに入力します。

```
package demo;
import org.junit.runner.RunWith;
import org.junit.runners.Suite.SuiteClasses;
import com.borland.silktest.jtf.SilkTestSuite;

@RunWith(SilkTestSuite.class)
@SuiteClasses({ MyTestClass1.class, MyTestClass2.class })
public class MyTestSuite {}
```

   これらのテスト クラスをコマンド ラインから実行するには、次のように入力します。

```
java org.junit.runner.JUnitCore demo.MyTestSuite
```

Apache Ant を使用したテストの再生

このトピックで述べる手順を実行するには、コンピュータに Apache Ant がインストールされている必要があります。

Apache Ant を使用してテストを再生し、たとえば、テスト実行の HTML レポートを生成するには、SilkTestSuite クラスを使用します。Apache Ant を使用してキーワード駆動テストを再生するには、KeywordTestSuite クラスを使用します。Apache Ant を使用したキーワード駆動テストの再生についての詳細は、「*Apache Ant* を使用したキーワード駆動テストの再生」を参照してください。

1. Apache Ant を使用してテストを実行するには、@SuiteClasses アノテーションを使用して JUnit テスト スイートを作成します。 たとえば、同じ Silk4J プロジェクトにある *MyTestClass1* クラスと *MyTestClass2* クラスにあるテストを実行する場合は、次のよな JUnit テスト スイート MyTestSuite を作成します。

```
@RunWith(SilkTestSuite.class)
@SuiteClasses({ MyTestClass1.class, MyTestClass2.class})
public class MyTestSuite {

}
```

2. テストを含んだ Silk4J プロジェクトの build.xml ファイルを開きます。
3. テストを実行するには、次のターゲットを build.xml ファイルに追加します。

注: 次のサンプル コードは、Silk Test 15.5 以降で作成された Silk4J プロジェクトでのみ動作します。

```
<target name="runTests" depends="compile">
  <mkdir dir="./reports"/>
  <junit printsummary="true" showoutput="true" fork="true">
    <sysproperty key="agentRmiHost" value="${agentRmiHost}" />
    <sysproperty key="silktest.configurationName" value="$
{silktest.configurationName}" />
    <classpath>
      <fileset dir="${output}">
        <include name="**/*.jar" />
      </fileset>
      <fileset dir="${buildlib}">
        <include name="**/*.jar" />
      </fileset>
```

```
    </classpath>

    <test name="MyTestSuite" todir="./reports"/>
  </junit>
</target>
```

JUnit タスクの詳細については、『*https://ant.apache.org/manual/Tasks/junit.html*』を参照してください。

4. 省略可能：すべてのテストの XML レポートを作成するには、ターゲットに次のコードを追加します。

```
<formatter type="xml" />
```

5. 省略可能：XML レポートから HTML レポートを作成するには、ターゲットに次のコードを追加します。

```
<junitreport todir="./reports">
  <fileset dir="./reports">
    <include name="TEST-*.xml" />
  </fileset>
  <report format="noframes" todir="./report/html" />
</junitreport>
```

JUnitReport タスクの詳細については、『*https://ant.apache.org/manual/Tasks/junitreport.html*』を参照してください。

完全なターゲットは、次のようになります。

```
<target name="runTests" depends="compile">

  <mkdir dir="./reports"/>
  <junit printsummary="true" showoutput="true" fork="true">
    <sysproperty key="agentRmiHost" value="${agentRmiHost}" />
    <sysproperty key="silktest.configurationName" value="$
{silktest.configurationName}" />
    <classpath>
      <fileset dir="${output}">
        <include name="**/*.jar" />
      </fileset>
      <fileset dir="${buildlib}">
        <include name="**/*.jar" />
      </fileset>
    </classpath>

    <formatter type="xml" />

    <test name="MyTestSuite" todir="./reports"/>
  </junit>
  <junitreport todir="./reports">
    <fileset dir="./reports">
      <include name="TEST-*.xml" />
    </fileset>
    <report format="noframes" todir="./report/html" />
  </junitreport>
</target>
```

6. Eclipse からテストを実行するには、以下の手順を実行します。
   a) **パッケージ・エクスプローラー** で、build.xml ファイルを右クリックします。
   b) **実行** > **Ant ビルド...** を選択します。
   c) **構成の編集** ダイアログ ボックスの **ターゲット** タブで、**runTests** をチェックします。
   d) **実行** をクリックします。

コマンド ラインや CI サーバーからテストを実行することもできます。詳細については、『*https://ant.apache.org/manual/running.html*』および *Silk4J* ヘルプ の「*CI (*継続的インテグレーション*)* サーバーからのテストの再生」を参照してください。

Ant を使用したテストの再生時のトラブルシューティング

Apache Ant を使用して Silk4J テストを実行する場合、JUnit タスクに fork="yes" を指定するとテストがハングします。これは、Apache Ant の既知の問題です (*https://issues.apache.org/bugzilla/show_bug.cgi?id=27614*)。以下のいずれかの回避策を使用できます。

- fork="yes" を使用しない。
- fork="yes" を使用する場合は、テストが実行される前に必ず Open Agent を起動しておく。Open Agent は、手動あるいは以下の Ant ターゲットを使って起動できます。

```
<property environment="env" />
<target name="launchOpenAgent">
<echo message="OpenAgent launch as spawned process" />
<exec spawn="true" executable="${env.OPEN_AGENT_HOME}/agent/openAgent.exe" />
<!-- give the agent time to start -->
<sleep seconds="30" />
</target>
```

CI (継続的インテグレーション) サーバーからのテストの再生

CI (継続的インテグレーション) サーバーから Silk4J のテストを実行するには、CI サーバーを設定する必要があります。このトピックでは、Jenkins を例として使用します。

1. Silk4J のテストをコンパイルする新しいジョブを CI サーバーに追加します。
   詳細については、CI サーバーのドキュメントを参照してください。
2. Silk4J のテストを実行する新しいジョブを CI サーバーに追加します。
3. Apache Ant ファイルを使用して CI サーバーからテストを再生します。Ant ファイルを使用してテストを実行すると、コマンド ラインからテストを実行した場合のように JUnit の結果が作成されます。

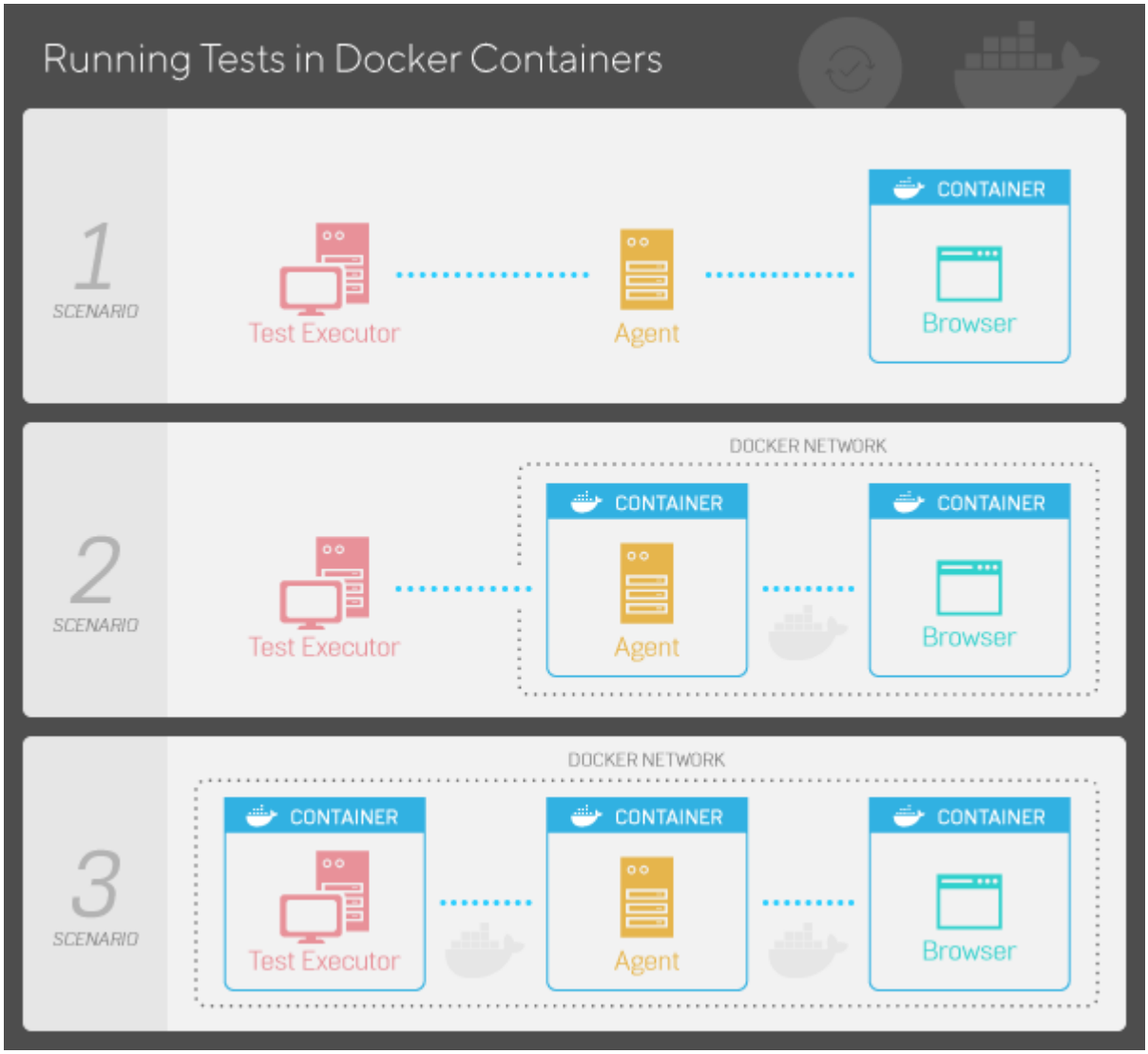
CI ジョブが実行されると、指定した Silk4J のテストの実行も動作します。Jenkins では、JUnit プラグインに Ant 出力が表示され、TrueLog ファイルが保存されます。

Docker コンテナ上でのテストの実行

Docker とは、コンテナ型のプラットフォーム プロバイダであり、アプリケーションを依存関係を含んだ仮想コンテナにパッケージ化できます。

Silk4J では、以下のシナリオにおける Docker コンテナ上でのテスト実行をサポートします。

- Docker コンテナ上のブラウザーに対するテストの実行：Open Agent とテストの実行プログラムは、ローカル Windows マシン上で実行します。
- 他の Docker コンテナ上で実行している Open Agent を使った Docker コンテナ上のブラウザーに対するテストの実行：ブラウザーと Open Agent は、Docker ネットワークを介して通信します。テストの実行プログラムは、ローカル Windows マシン上で実行します。
- Docker コンテナ上でのテスト全体の実行：Silk4J、テストの実行プログラム、ブラウザーを、それぞれの Docker コンテナで実行し、Docker ネットワークを介して通信します。



Silk4J が Docker コンテナ上での Web アプリケーションのテストをサポートするデスクトップ ブラウザーは以下の通りです。

- Microsoft Edge
- Mozilla Firefox
- Google Chrome
- Apple Safari

Docker コンテナ上で Silk4J テストを実行すると、既存の CI ワークフローに Silk4J テストを簡単に統合でき、次のメリットを享受できます。

- Windows、Linux、macOS 上でテストを実行できる。
- テストを実行するマシン上に 、Silk4J や Java などの実行に必要なソフトウェアをインストールする必要がない。
- Docker が提供するイメージを使用するだけで、ブラウザーのバージョンを更新する必要がない。
- テストをバックグラウンドで実行し、他の作業を行うことができる。
- Docker イメージとテストの実行にコマンド ラインを使用できる。

Silk4J の提供する Docker イメージは、*functionaltesting/silktest* という名前の基本的なイメージで、そこには Open Agent と Silk4J テストを実行に必要な環境が含まれています。このイメージは、*Docker Hub* からダウンロードできます。

注: このセクションのトピックは、Docker に関する知識があることを前提としています。Docker に関する知識がない場合は、読み進める前に *Docker* Web サイトを参照してください。

Silk Test イメージの環境変数

Silk Test Docker コンテナの実行を制御するのに使用できる環境変数を、以下の表に示します。

環境変数	説明
SILK_LICENSE_SERVER	Silk Meter ライセンス サーバーのホスト名または IP アドレス。
SILK_SELENIUM_SERVER_PORT	省略可能 : 組み込み Selenium サーバーのリッスン ポート。この環境変数を設定しないと、Selenium サポートはオフになります。 例 : `SILK_SELENIUM_SERVER_PORT=4444`
SILK_RMI_SERVER_PORT	省略可能 : JTF 用 RMI サーバーのリッスン ポート。この環境変数を設定しないと、動的にポートが割り当てられます。Docker コンテナ外の JTF を使って Open Agent と通信する必要がある場合や、ポート フォワーディング用に固定ポートが必要な場合にだけ、設定してください。 例 : `SILK_RMI_SERVER_PORT=30000`
SILK_LOG_FILE_PATH	省略可能 : Open Agent ログ ファイルを出力する Docker コンテナ内のパス。 例 : `SILK_LOG_FILE_PATH=/output/log.txt`
SILK_CONSOLE_LOG	省略可能 : Open Agent ログのほかに、コンソールにログを出力するかどうかを指定します。 例 : `SILK_CONSOLE_LOG=true`
SILK_LOG_LEVEL	省略可能 : ログ レベル。 例 : `SILK_LOG_LEVEL=DEBUG`

各変数は -e オプションを使って次のように指定します。

```
docker run -e SILK_LOG_FILE_PATH=/logs/log.txt
  -v c:/temp/ChromeExample/logs:/logs
  -e SILK_SELENIUM_SERVER_PORT=4444
  -e SILK_RMI_SERVER_PORT=30000
  -e SILK_CONSOLE_LOG=true
  -e SILK_LOG_LEVEL=DEBUG
  -e SILK_LICENSE_SERVER=<license-server address>
  --name agent
  functionaltesting/silktest:latest
```

例 : Google Chrome 上でのテストの実行

このトピックでは、Linux マシン上の Google Chrome で Silk4J テスト セットを Apache Ant を使って実行するために、Docker コンテナを使用する方法を例をあげて説明します。テスト セットを実行する前に、次の手順を実行してください。

- マシン上に Docker をインストールします。
- Ant を使ってテストを実行する Silk4J プロジェクトを準備します。詳細については、「*Apache Ant を使用したテストの再生*」を参照してください。
- プロジェクトを Linux マシンにコピーします（/home/<ユーザー名>/projects/InsuranceWeb など）。

Silk4J テスト セットを Google Chrome 上で実行するには：

1. 関連するイメージの最新のバージョンをレジストリにプルします。
   たとえば、この例では次の 3 つのイメージが必要です。
   - 最新の Silk Test イメージ
     ```
     docker pull lnzdevdocker01.microfocus.com:5000/nanoagent:latest
     ```
   - 最新の Google Chrome イメージ
     ```
     docker pull selenium/standalone-chrome:latest
     ```
   - テストを実行する Ant コンテナ
     ```
     docker pull webratio/ant:latest
     ```
2. Docker の仮想ネットワークを作成し、Docker コンテナがお互いに通信できるようにします。
   この例では、*my-network* という名前のネットワークを作成します。
   ```
   docker network create my-network
   ```
3. Google Chrome Docker コンテナを開始します。
   ```
   docker run --network my-network --name chrome selenium/standalone-chrome:latest
   ```
4. Open Agent Docker コンテナを開始します。
   ```
   docker run -e SILK_LICENSE_SERVER=<license-server address>
     -e SILK_LOG_FILE_PATH=/logs/log.txt
     -v /home/<user name>/projects/logs:/logs
     --network my-network
     --name agent
     lnzdevdocker01.microfocus.com:5000/nanoagent:latest
   ```
   利用可能な環境変数についての詳細は、「*Silk Test イメージの環境変数*」を参照してください。
5. Ant Docker コンテナを開始して、テストを実行します。
   ```
   docker -run -v /home/<user name>/projects/InsuranceWeb:/tmp/project
     --network my-network
     --name=test-runner
     -it webratio/ant:1.10.1 ant
        -DagentRmiHost=agent:22902
        -Dsilktest.configurationName="host=http://chrome:4444/wd/hub - Chrome"
        -buildfile /tmp/project/build.xml runTests
   ```
6. /home/<ユーザー名>/projects/logs の下にあるテスト結果を表示します。
7. 省略可能：テスト環境をクリーンアップするには、次のコマンドを実行します。
   - Open Agent Docker コンテナを停止します。
     ```
     docker stop agent
     ```
   - Google Chrome Docker コンテナを停止します。
     ```
     docker stop chrome
     ```
   - Open Agent Docker コンテナを削除します。
     ```
     docker rm agent
     ```
   - Google Chrome Docker コンテナを削除します。
     ```
     docker rm chrome
     ```
   - Ant Docker コンテナを削除します。
     ```
     docker rm test-runner
     ```

- 仮想ネットワークを削除します。

```
docker network rm my-network
```

例：docker-compose の利用

このトピックでは、Linux マシン上の Google Chrome で Silk4J テスト セットを Apache Ant を使って実行するために、Docker コンテナを使用する方法を例をあげて説明します。テスト セットを実行する前に、次の手順を実行してください。

- マシン上に Docker をインストールします。
- Ant を使ってテストを実行する Silk4J プロジェクトを準備します。詳細については、「*Apache Ant を使用したテストの再生*」を参照してください。
- プロジェクトを Linux マシンにコピーします（/home/<ユーザー名>/projects/InsuranceWeb など）。

Silk4J プロジェクトのテストを Google Chrome 上で実行するには：

1. docker-compose .yml ファイルを作成します。

```
version: '3'
services:
  chrome:
    image: selenium/standalone-chrome:latest
    environment:
      - JAVA_OPTS=-Dselenium.LOGGER.level=WARNING
  agent:
    image: lnzdevdocker01.microfocus.com:5000/nanoagent:latest
    environment:
      - SILK_LICENSE_SERVER=lnz-lic1.microfocus.com
      - SILK_LOG_FILE_PATH=/logs/log.txt
    depends_on:
      - chrome
    links:
      - chrome
    volumes:
      - /home/<user name>/proajects/logs:/logs
  tests-runner:
    image: webratio/ant:1.10.1
    volumes:
      - /home/ralph/InsuranceWeb:/tmp/project
    command: ["ant", "-DagentRmiHost=agent:22902", "-
Dsilktest.configurationName=host=http://chrome:4444/wd/hub - GoogleChrome", "-
buildfile", "/tmp/project/build.xml", "runTests"]
    depends_on:
      - agent
    links:
      - agent
```

2. 関連するイメージの最新のバージョンをレジストリにプルします。

```
docker-compose pull
```

3. テストを実行し、最初のコンテナ（この場合は Ant コンテナ）が停止したらすべてのコンテナを停止します。

```
docker-compose up --abort-on-container-exit
```

4. /home/<ユーザー名>/projects/logs の下にあるテスト結果を表示します。
5. 省略可能：テスト環境をクリーンアップします。

```
docker-compose down
```

最新のバージョンのプル、テストの実行、テスト環境のクリーンナップを、コマンドにまとめて指定することもできます：

```
docker-compose pull && docker-compose up --abort-on-container-exit && docker-compose down
```

Docker コンテナ上でのテスト実行時の制限事項

Docker コンテナ上でテストを再生した場合の制限事項を以下に示します。

- イメージ解決（imageClick、imageRect、imageExists を含む）とイメージ検証はサポートされません。
- Desktop クラスのマウス メソッドとキーボード入力メソッド（mouseMove、typeKeys、click など）はサポートされません。
- BrowserWindow クラスの getViewPortName メソッドと getViewPortName メソッドはサポートされません。
- 次のシステム関数はサポートされません。
    - closeFile
    - closeIniFile
    - createRegistryKey
    - createRegistryValue
    - deleteRegistryKey
    - deleteRegistryValue
    - existsRegistryKey
    - fileWriteLine
    - getAgentVersion
    - getClipboardText
    - getCurrentDrive
    - getCursorPosition
    - getCursorType
    - getFileInfo
    - getFreeDiskSpace
    - getIniFileValue
    - getLocale
    - getRegistryKeyNames
    - getRegistryValue
    - getRegistryValueNames
    - openFile
    - openIniFile
    - readLine
    - setClipboardText
    - setCurrentDrive
    - setEnvironmentVariable
    - setFilePointer
    - setIniFileValue
    - setRegistryValue

Docker コンテナ上でのテスト実行時のトラブルシューティング

Docker コンテナ上で実行中のテストが行っていることを確認する方法

Docker コンテナ上の Google Chrome または Mozilla Firefox に対して実行しているテストをデバッグする場合は、Google Chrome または Mozilla Firefox のデバッグ イメージを *Docker Hub* から利用できます。これらを使うと、VNC 接続を使ってテストの実行を確認できます。

Silk Central からの Silk4J テストの再生

Silk Central から Silk4J テストにアクセスするには、Silk Central がソース管理プロファイルを介してアクセスできるリポジトリに Silk4J テストを含んだ JAR ファイルを格納する必要があります。

Silk Central から Silk4J の機能テスト (キーワード駆動テストなど) を再生するには：

1. Silk Central で、Silk4J テストを実行するプロジェクトを作成します。
2. **テスト** > **詳細ビュー** を開き、新しいプロジェクト用に新しいテスト コンテナを作成します。

   Silk Central に関する追加の情報については、『*Silk Central ヘルプ*』を参照してください。

   テスト コンテナには、Silk4J テストを格納するソース管理プロファイルを指定する必要があります。

   a) **テスト** ツリーで、その下に新しいテスト コンテナを追加するノードをクリックします。
   b) **テスト コンテナの新規作成** をクリックします。**テスト コンテナの新規作成** ダイアログ ボックスが開きます。
   c) **名前** フィールドに、新しいテスト コンテナの名前を入力します。
   たとえば、キーワード駆動テスト を入力します。
   d) **ソース管理プロファイル** フィールドに、Silk4J テストを含んだ JAR ファイルを格納するソース管理プロファイルを選択します。
   e) **OK** をクリックします。
3. 新しいテスト コンテナに新しい JUnit テストを作成します。

   Silk Central に関する追加の情報については、『*Silk Central ヘルプ*』を参照してください。

   a) **JUnit テスト プロパティ** ダイアログ ボックスの **テスト クラス** フィールドに、テスト クラスの名前を入力します。

   テスト スイート クラスの完全修飾名を指定します。詳細については、「コマンド ラインからのキーワード駆動テストの再生」を参照してください。

   b) **クラスパス** フィールドに、テストを含む JAR ファイルの名前を指定します。
   c) キーワード駆動テストの場合、セミコロンで区切って次のファイルへのパスも指定します。

   - com.borland.silk.keyworddriven.engine.jar
   - com.borland.silk.keyworddriven.jar
   - silktest-jtf-nodeps.jar

   これらのファイルは、Silk Test インストール ディレクトリにあります。たとえば、キーワード駆動テストが tests.jar JAR ファイルに含まれている場合、**クラスパス** フィールドは、次のようになります。

```
tests.jar;C:¥Program  Files
(x86)¥Silk¥SilkTest¥ng¥KeywordDrivenTesting
¥com.borland.silk.keyworddriven.engine.jar;C:¥Program  Files
(x86)¥Silk¥SilkTest¥ng¥KeywordDrivenTesting¥com.borland.silk.keyworddriven.jar;C:
¥Program  Files
(x86)¥Silk¥SilkTest¥ng¥JTF¥silktest-jtf-nodeps.jar
```

4. **終了** をクリックします。
5. テストを実行します。
   Silk Central でのテストの実行に関する追加の情報については、『*Silk Central ヘルプ*』を参照してください。

CI (継続的インテグレーション) サーバーから Silk Central でのテストの実行

CI サーバーから Silk4J のテストを実行するには、次のインフラが必要です。

- 適切な実行定義を持つ Silk Central サーバー

  注: このトピックでは、Silk Central との統合を中心に説明しますが、他のテスト スケジュール ツールを使用することもできます。
- Hudson や Jenkins のような CI (継続的インテグレーション) サーバーこのトピックでは、Jenkins を例として使用します。

CI サーバーから機能テストを再生するには：

1. Silk Central で、CI サーバーから実行する実行計画のプロジェクト ID と実行計画 ID を取得します。
   a) **実行計画** > **詳細ビュー** を選択します。
   b) **実行計画** ツリーで、実行を含むプロジェクトを選択します。**プロジェクト ID** がプロジェクトの **プロパティ** ペインに表示されます。
   c) **実行計画** ツリーで、実行計画を選択します。**実行計画 ID** が実行計画の **プロパティ** ペインに表示されます。
2. CI サーバーに **SCTMExecutor** プラグインをインストールします。 このプラグインは、CI サーバーと Silk Central サーバーを接続します。
3. **SCTMExecutor** プラグインを設定します。
   a) Jenkins 上で、Jenkins のグローバル設定ページにある **Silk Central Test Manager Configuration** 設定に移動します。
   b) **Service URL** フィールドに、 Silk Central サービスのアドレスを入力します。
   たとえば、サーバーの名前が *sctm-server* の場合は、http://sctm-server:19120/services を入力します。
4. CI ビルド ジョブを拡張します。
   a) Jenkins 上で、**ビルド手順の追加** リストから **Silk Central Test Manager Execution** を選択します。
   b) **Execution Plan ID** フィールドに、実行計画 ID を入力します。

      ID をカンマ区切りで入力して、複数の実行計画を実行できます。
   c) **SCTM Project ID** フィールドに、Silk Central プロジェクトのプロジェクト ID を入力します。

CI ビルド ジョブが実行されると、指定した Silk Central 実行計画の実行も動作します。

特定の順番でのテストの再生

Java 1.6 以前を使って JUnit テストを実行すると、ソース ファイルで宣言された順番で実行されます。

注: しかし、Java 1.7 以降では、JUnit テストを実行する順番を指定することはできません。これは、テスト実行における JUnit の制約です。

JUnit テストの実行順序は JUnit のバージョンによって異なります。JUnit 4.11 より前のバージョンを使用すると、テストは特定の順番は無く、実行されます。テスト実行のたびに異なることもあります。JUnit 4.11 以降では、各テスト実行で同じ順番でテストが実行されますが、順番は予測できません。

テスト環境によっては、この制約を回避できる場合があります。

回避策の例

テスト セットにモジュールやスイートが含まれない場合、ソース ファイルの最初に次の行を追加します。

```
import  org.junit.FixMethodOrder;
import  org.junit.runners.MethodSorters;
@FixMethodOrder(MethodSorters.JVM)
```

FixMethodOrder には、3 種類の値を指定できます。

MethodSorters.JVM	JVM によって返されるメソッドの順番。テストの実行ごとに変化する可能性があります。テスト セットが正しく実行されない場合があります。
MethodSorters.DEFAULT	メソッド名の hashCode に基づいた一意に決定される順序。適切な hashCode になるようにメソッド名を定義しなければならないため、順番を変更することは困難です。
MethodSorters.NAME_ASCENDING	テストの名前の辞書式順序に基づいた順序。テスト名のアルファベット順がテストを実行する順番と一致するように、テストの名前を変更する必要があります。

1.7 より前のバージョンの Java を使用することもできます。

テストの並列実行

複数の JUnit プロセスを使用して、複数のブラウザーやモバイル デバイスに対してテストを並列に実行することができます。たとえば、CI サーバーや Silk Central からテストを実行するときに、この機能を使用できます。

デフォルトでは、Silk Test は、次のブラウザーやプラットフォームでの並列テストをサポートします。

- Google Chrome
- Mozilla Firefox
- 以下のプラットフォーム上での Web、ネイティブ、ハイブリッド アプリ
  - 物理 Android デバイス
  - Android エミュレータ
  - 物理 iOS デバイス

既存の Silk4J スクリプトに WebDriver 機能を追加して混合スクリプトにする場合は、getWebDriver メソッドを使って新しいドライバーを取得します。詳細については、「既存の *Silk4J* スクリプトと *Selenium* スクリプトの使用」を参照してください。

注: 新しい WebDriver オブジェクトの取得に new RemoteWebDriver を使っても、混合スクリプトの並列テストは動作しません。

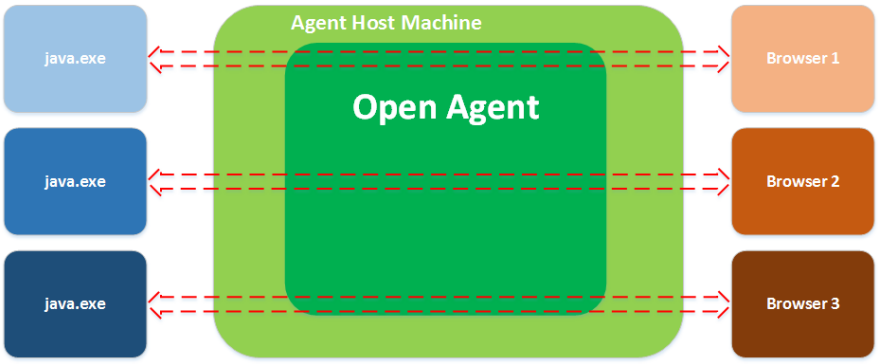
テストの並列再生を無効にするには、環境変数 *SILKTEST_ENABLE_PARALLEL_TESTING* を false に設定します。

注: 並列テストを有効にすると、Open Agent はテストの実行プロセスのそれぞれを別々に処理しようとします。ある Silk Test クライアントでテストされたアプリケーションは、最初のクライアントが実行している間、他のクライアントでテストすることはできません。たとえば、Silk4J と Silk4NET で交互に同じアプリケーションをテストすることはできません。

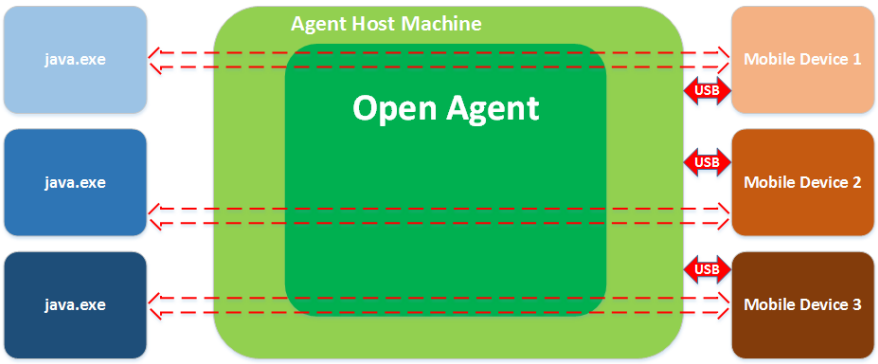
注: 同じモバイル デバイス上で同時に複数のテストを実行することはできません。テストを並列実行する前に、十分なデバイスとエミュレータが利用できることを確認してください。モバイル デバイスやエミュレータが割り当てられないと、テストの実行は失敗します。

並列テストの実行それぞれが、1 個のブラウザーまたはモバイル デバイスに対して個々の java.exe を開始します。ブラウザーやモバイル デバイスを特定の java.exe と関連付けたい場合には、接続文字列を使用できます。詳細については、「モバイル デバイスの接続文字列」または「リモート デスクトップ ブラウザーの接続文字列」を参照してください。

次の図は、複数のブラウザーの並列テストのイメージを表します。



次の図は、複数のデバイスの並列テストのイメージを表します。



複数のプロセスが同時に開始されると、Silk4J を実行しているマシン上で Open Agent を何度も開始しようとする可能性があります。同じマシンで複数回 Open Agent を実行することはできないため、Silk4J は例外をスローすることになります。これを避けるため、テストを並列実行する前に、Open Agent が実行されていることを確認してください。

テスト結果は、複数の TrueLog ファイル（各テスト実行毎に 1 つ）に保存されます。TrueLog ファイルが上書きされないようにするため、TrueLog ファイル名にプレースホルダーを使用することができます。詳細については、「*TrueLog* オプションの設定」を参照してください。

注: テスト中のメモリ消費量が激しいと感じた場合は、テスト結果が XLG 形式ではなく、圧縮ファイルである TLZ 形式で保存されていることを確認してください。並列テストでは、Silk4J は TrueLog API をサポートしません。

Silk4J がテストを同期する方法

ほとんどの予期しないテストの失敗は、同期問題に起因します。テストの再生時の同期が弱いと、予期しない結果が生成されるため、実際のアプリケーションの問題を見つけることが困難になります。同期エラーはタイミングの問題で、テストの環境に強く依存しており、このようなエラーを再現し解決することは困難です。たとえば、特定のテスト環境で発生する同期エラーが、開発環境では再現できないことがあります。同期が弱いと、自動化プロジェクトにおいて、自動化テスト セットの肥大化に伴い管理不能に陥る、最も一般的な理由のひとつになります。

Silk4J では、自動的なテスト同期をすべてのサポートするテクノロジに対して提供しているため、ロバストで管理可能なテスト セットを構築できます。テストの再生中に、Silk4J は、AUT が次の操作を実行する準備ができるまで、常にテストが待機するようにします。テストの検証ステップでは、Silk4J は、テストのそれより前のテストが検証を実行する前に完了しているようにします。

AUT の特定の動作にテストを適用する場合は、次の同期タイムアウトの値を変更できます。

同期タイムアウト (OPT_SYNC_TIMEOUT)	再生中に AUT が準備完了状態になるまで Silk4J が待機する最大時間をミリ秒で指定します。デフォルト値は、300000 ミリ秒です。
オブジェクト解決タイムアウト (OBJ_WAIT_RESOLVE_OBJDEF)	find メソッドがオブジェクトを検索する最大時間をミリ秒で指定します。デフォルト値は、5000 ミリ秒です。 **注:** 低速な接続でアクセスした仮想マシンなどの低速なシステムや高負荷な環境でテストを正しく実行できるようにするために、Silk4J は内部タイムアウト値を増やすことがあります（AUT の開始時やダイアログやウィンドウの表示時など）。AUT、ダイアログ、ウィンドウが完全に表示されると、Silk4J は再びタイムアウト値を OPT_WAIT_RESOLVE_OBJDEF で指定された値に戻します。
オブジェクト解決再試行間隔 (OPT_WAIT_RESOLVE_OBJDEF_RETRY)	Silk4J がオブジェクトを直ちに見つけることができなかった場合、Silk4J はオブジェクト解決タイムアウトが経過するまでオブジェクトの検索を再試行します。オブジェクト解決再試行間隔は、Silk4J がオブジェクトの検索を再試行するまで待機する時間をミリ秒で指定します。デフォルト値は、1000 ミリ秒です。
オブジェクト有効化タイムアウト (OPT_OBJECT_ENABLED_TIMEOUT)	再生中にオブジェクトが有効になるまで Silk4J が待機する最大時間をミリ秒で指定します。デフォルト値は、1000 ミリ秒です。

注: タイムアウトは重なりません。

Silk4J が提供する自動的な同期のうち、特に Web アプリケーションに関する詳細な情報については、「*xBrowser のページ同期*」を参照してください。Silk4J が提供する同期のうち、特に AJAX アプリケーションに関する詳細な情報については、「*How to Synchronize Test Automation Scripts for Ajax Applications*」を参照してください。

自動的な同期に加えて、Silk4J では、スクリプトに待機関数を手動で追加することもできます。Silk4J は、手動同期用に次の待機関数を提供します。

waitForObject	指定したロケータに一致するオブジェクトを待機します。XPath ロケーターやオブジェクト マップ識別子に対して動作します。
waitForProperty	propertyName パラメータで指定したプロパティが expectedValue パラメータで指定した値を返すまで、またはタイムアウトに到達するまで待機します。
waitForDisappearance	オブジェクトが存在しなくなるか、タイムアウト値に到達するまで待機します。
waitForChildDisappearance	locator パラメータで指定された子オブジェクトが存在しなくなるか、タイムアウト値に到達するまで待機します。

テストが ObjectNotFoundException で不規則に失敗する場合、オブジェクト解決タイムアウト を増やします (30 秒にするなど)。進行状況ダイアログが表示され、長時間計算が行われた後に表示されるオブジェクトをクリックする場合など、特定の操作に特別に時間のかかる場合は、waitForObject メソッドをテスト スクリプトに手動で追加してオブジェクトが見つかるまで待機するようにしたり、waitForDissapearance メソッドをテスト スクリプトに追加して進行状況ダイアログが表示されなくなるまで待機するようにします。

自動的な同期の例

Silk4J がキャプション **Ok** を持つボタンをクリック する、次のサンプル コードを考えます。

```
PushButton button = _desktop.find("//PushButton[@caption='ok'");
button.Click();
```

このサンプル コードの操作を再生するために、Silk4J は次の同期操作を実行します。

1. Silk4J はボタンを検索します。そして、オブジェクト解決タイムアウト が経過した場合、Silk4J は再生を停止して例外をスローします。
2. Silk4J はテスト対象アプリケーション (AUT) が準備完了状態になるまで待機します。AUT が準備完了状態になる前に、同期タイムアウト が経過した場合、Silk4J は再生を停止して例外をスローします。
3. Silk4J はボタンが有効になるまで待機します。オブジェクト有効化タイムアウト がボタンが有効になる前に経過した場合、Silk4J は再生を停止して例外をスローします。
4. Silk4J はボタンをクリックします。
5. Silk4J はテスト対象アプリケーション (AUT) が再び準備完了状態になるまで待機します。

再生ステータス ダイアログ ボックスの有効化

再生ステータス ダイアログ ボックスを有効にすると、テストの再生中に実行される操作を表示できます。このダイアログ ボックスは、リモート Mac やモバイル デバイスなど、他のマシン上でテストを再生する場合に非常に役立ちます。

再生ステータス ダイアログ ボックスを有効化するには :

1. **Silk4J** > **オプションの編集** をクリックします。
2. **再生** タブ をクリックします。
3. **再生ステータス** ダイアログ ボックスを有効化するには、**OPT_SHOW_PLAYBACK_STATUS_DIALOG** チェック ボックスをオンにします。
4. **再生ステータス** ダイアログ ボックスにテスト対象アプリケーションのビデオまたはスクリーンショットを表示するには、**OPT_PLAYBACK_STATUS_DIALOG_SCREENSHOTS** チェック ボックスをオンにします。

5. **OK** をクリックします。

注: リモート Mac やモバイル デバイス上でテストする場合は、Mac またはデバイスの画面がテスト中にオフにならないようにしてください。オフになると、**再生ステータス** ダイアログ ボックスには何も表示されなくなります。

TrueLog を使用したビジュアル実行ログ

TrueLog は、ビジュアルな検証を通じてテスト ケースの失敗の根本的な原因の分析を単純化するための強力なテクノロジです。 テストの結果は、TrueLog Explorer で検証できます。 テストの実行中にエラーが発生すると、TrueLog はそのエラーが発生したスクリプトの行を簡単に特定し、問題を解決できるようにします。

注: TrueLog は、スクリプトに対して単一のローカル エージェントまたはリモート エージェントのみをサポートしています。 たとえば、1 つのマシンでアプリケーションをテストし、そのアプリケーションが別のマシンのデータベースにデータを書き込む場合のように、複数のエージェントを使用する場合は、スクリプトで使用された最初のエージェントに対してのみ TrueLog が書き出されます。 リモート エージェントを使用する場合は、リモート マシンにも TrueLog ファイルが書き出されます。

TrueLog Explorer の詳細については、(Microsoft Windows 7) **スタート** > **すべてのプログラム** > **Silk** > **Silk Test** > **ドキュメント**、または (Microsoft Windows 10) **スタート** > **Silk** にある *Silk TrueLog Explorer* ユーザー ガイド を参照してください。

Silk4J で TrueLog を有効にして、Silk4J テストの実行中にビジュアル実行ログを作成できます。 TrueLog ファイルは、Silk4J テストが実行されたプロセスの作業ディレクトリに作成されます。

注: Silk4J テストの実行中に TrueLog を作成するには、JUnit バージョン 4.6 以降が使用されている必要があります。 JUnit バージョンが 4.6 よりも古い場合に TrueLog を作成しようとすると、TrueLog を書き出すことができないことを示すエラー メッセージを、Silk4J はコンソールに出力します。

デフォルトの設定では、スクリプトでエラーが発生した場合にのみスクリーンショットが作成され、エラーの発生したテスト ケースのログのみが作成されます。

TrueLog の有効化

TrueLog を有効にするには、以下を実行します。

1. **Silk4J** > **オプションの編集** をクリックします。 **スクリプト オプション** ダイアログ ボックスが表示されます。
2. **TrueLog** タブをクリックします。
3. **基本設定** 領域で **TrueLog の有効化** チェック ボックスをオンにします。
   - 正常なものとエラーになったものを両方とも含めて、すべてのテスト ケースのアクティビティを記録するには、**すべてのテストケース** をクリックします。これはデフォルトの設定です。
   - エラーが発生したテスト ケースのみのアクティビティを記録するには、**エラーのあるテストケース** をクリックします。

デフォルトでは、TrueLog ファイルは、Silk4J テストが実行されたプロセスの作業ディレクトリに作成されます。TrueLog の別の場所を指定するには、**Silk4J** > **オプションの編集** をクリックして **スクリプト オプション**ダイアログ ボックスを開き、**TrueLog ファイル** フィールドの右側にある **参照** をクリックします。

Silk4J テストの実行が完了したら、**再生の完了** ダイアログ ボックスが開き、完了したテストの TrueLog を選択して確認できます。

TrueLog の場所の変更

デフォルトでは、Visual Studio ソリューション ファイルおよび Visual Studio Unit Testing Framework の結果が格納されているディレクトリの TestResults サブディレクトリに TrueLog が作成されます。

TrueLog の場所を指定するには：

1. メニューで、**Silk4J** > **オプションの編集** をクリックします。**スクリプト オプション** ダイアログ ボックスが開きます。
2. **TrueLog** タブを選択します。
3. **TrueLog ファイル** フィールドの右側にある **参照** をクリックします。

現在のプロジェクトでテストを実行すると、TrueLog は指定した場所に保存されます。

TrueLog セクション

TrueLog にセクションを追加して、複雑なスクリプトを、名前の付けられた小さなパーツに論理的に分割して構造化できます。

TrueLog セクションは入れ子状にすることができます。 また、名前は一意である必要は無く、複数のセクションで同じ名前を使用できます。

TrueLog ファイルに新しいセクションを作成するには、次のコードを使用します。

```
openTrueLogSection("section_name")
```

最後に開いた TrueLog セクションを閉じるには、次のコードを使用します。

```
closeCurrentTrueLogSection()
```

次の場合には、セクションは自動的に閉じられます。

- テスト ケースの終了
- テスト クラスの終了
- テスト実行の終了
- キーワードの終了
- 例外の発生

以下に、TrueLog セクション の使用例を示します。

```
@Test
public void subSections() {
   Desktop desktop = new Desktop();
   desktop.openTrueLogSection("Section a");
   desktop.logInfo("In section a");
   desktop.openTrueLogSection("Section b");
   desktop.logInfo("In section b");
   desktop.closeCurrentTrueLogSection();
   desktop.logInfo("In section a again");
}
```

サポートするメソッドの詳細については、Silk Test クライアントの API ドキュメントを参照してください。

Web ページ コンテンツのキャプチャ

ブラウザー ウィンドウに現在表示されている Web ページの一部をスクリーンショットにキャプチャする場合は、captureBitmap メソッドを使用できます。画像ファイルの場所と名前は、絶対パスまたは相対パスでパラメータで指定します。例：

```
browserWindow.captureBitmap("C:¥Temp¥MyPage.png");
```

Web ページ コンテンツ全体のスクリーンショットを 1 つの画像としてキャプチャする場合は、captureFullPageBitmap メソッドを使用できます。画像ファイルの場所と名前は、絶対パスまたは相対パスでパラメータで指定します。例：

```
browserWindow.captureFullPageBitmap("C:¥Temp¥MyPage.png");
```

TrueLog で非 ASCII 文字が正しく表示されない理由

TrueLog Explorer は MBCS ベースのアプリケーションであるため、正しく表示するには、すべての文字列が MBCS 形式でエンコードされている必要があります。 TrueLog Explorer でデータを表示およびカスタマイズすると、データが表示される前に、多数の文字列変換処理が発生することがあります。

UTF-8 でエンコードされた Web サイトをテストする場合は、文字列を含むデータをアクティブな Windows システム コード ページに変換できないことがあります。 このような場合、TrueLog Explorer は変換できない非 ASCII 文字列を、構成可能な置換文字 (通常は「?」) で置き換えます。

TrueLog Explorer で非 ASCII 文字列を正確に表示するには、システム コード ページに適切な言語 (日本語など) を設定します。

Silk Test Open Agent

Silk Test Open Agent は、スクリプトのコマンドを GUI 固有のコマンドに翻訳するソフトウェア プロセスです。 つまり、Open Agent がテストするアプリケーションを動かし、監視しています。

ホストマシン上で 1 つのエージェントをローカルに実行できます。 ネットワーク環境では、任意の数のエージェントがリモート マシン上でテストを再生できます。 ただし、記録はローカル マシン上でのみ実行できます。

Silk Test Open Agent の起動

テストの作成またはサンプル スクリプトの実行前に、Silk Test Open Agent が実行されている必要があります。 通常は、製品を起動したときにエージェントが実行されます。 Open Agent を手動で開始しなければならない場合には、次のステップを実行してください。

(Microsoft Windows 7) **スタート** > **すべてのプログラム** > **Silk** > **Silk Test** > **ツール** > **Silk Test Open Agent**、または (Microsoft Windows 10) **スタート** > **Silk** > **Silk Test Open Agent** をクリックします。 Silk Test Open Agent アイコン が、システム トレイに表示されます。

テスト実行後に Open Agent を停止する

コマンド ライン オプション -shutDown を使うか、スクリプトから Open Agent を停止して、テスト実行が終わってもエージェントが実行し続けることがないようにすることができます。次のコマンドをコマンド ラインから実行して、Open Agent を停止することができます：

```
openAgent.exe  -shutDown
```

スクリプトからエージェントを停止するには：

1. テスト実行が完了したときに実行するスクリプトを開く、または作成します。
   たとえば、テスト実行後にクリーンアップのために使用する既存のスクリプトを開きます。
2. スクリプトに shutdown メソッドを追加します。

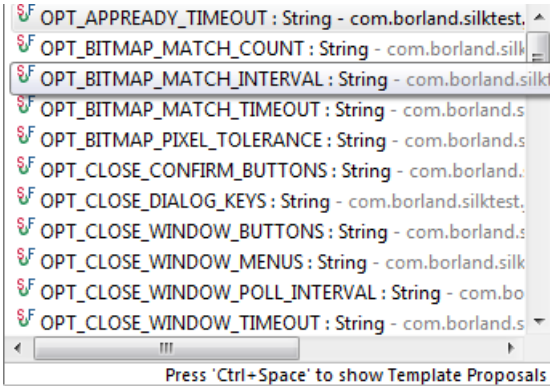
注: 他のスクリプトが Open Agent を必要になった時点で、エージェントは再起動されます。

エージェント オプション

このセクションでは、getOption および setOption メソッドで操作できるオプションについて説明します。

注: デフォルトでは、メニューから **Silk4J** > **オプションの編集** を選択してグローバル オプションを設定できます。

スクリプトでエージェント オプションを設定または取得するには、getOption または setOption メソッドを使用します。たとえば、Desktop.setOption(Commonoptions. と入力すると、オートコンプリートと構文ヒント テクノロジを使用して、エージェント オプションがアクティブなエディター ウィンドウに自動的に表示されます。たとえば、以下のオプションがアクティブなエディター ウィンドウに表示されます。



スクリプトでオプションを挿入する場合は、次のように記述します。

```
desktop.setOption(Commonoptions.ApplicationReadyTimeout, 100000);
```

あるいは、スクリプトでエージェント オプション名を使用することもできます。たとえば、以下のように入力できます。

```
desktop.setOption("OPT_APPREADY_TIMEOUT", 100000);
```

.NET オプション名	エージェント オプション名	定数の型	説明
ApplicationReadyTimeout	OPT_APPREADY_TIMEOUT	数値	新しく起動したアプリケーションが準備完了状態になるまで待機する時間をミリ秒で指定します。指定したタイムアウト時間内にアプリケーションの準備が整わない場合は、Silk4J によって例外がスローされます。
BitmapMatchCount	OPT_BITMAP_MATCH_COUNT	整数	ビットマップが安定していると判断するために、連続して同じビットマップでなければならないスナップショットの数を指定します。スナップショットは、OPT_BITMAP_MATCH_TIMEOUT で指定された秒数まで取得され、各スナップショット間で OPT_BITMAP_MATCH_INTERVAL で指定された時間間隔で一時停止します。 デフォルトで、これは 0 です。
BitmapMatchInterval	OPT_BITMAP_MATCH_INTERVAL	実数	ビットマップ画像が安定していることを保証するために使用するスナップショット間の時間間隔を指定します。スナップショットは、OPT_BITMAP_MATCH_TIMEOUT で指定された時間まで取得されます。 デフォルトで、これは 0.1 です。
BitmapMatchTimeout	OPT_BITMAP_MATCH_TIMEOUT	実数	ビットマップ画像が安定するまでの合計許容時間を指定します。 この期間に Silk4J は画像の複数のスナップショットを取得し、スナップショット間で OPT_BITMAP_MATCH_TIMEOUT で

.NET オプション名	エージェント オプション名	定数の型	説明
			指定された秒数だけ待機します。OPT_BITMAP_MATCH_COUNT で指定されたビットマップ数と一致する前に、OPT_BITMAP_MATCH_TIMEOUT から返される値に到達すると、Silk4J でスナップショットの取得が停止し、例外 E_BITMAP_NOT_STABLE が発生します。 デフォルトで、これは 5 です。
BitmapPixelTolerance	OPT_BITMAP_PIXEL_TOLERANCE	整数	2 つのビットマップが一致しているとみなされる許容差異 (ピクセル単位) を指定します。差異ピクセル数がこのオプションで指定された値より小さい場合、ビットマップは同一であるとみなされます。最大許容値は 32767 ピクセルです。 デフォルトで、これは 0 です。
ButttonsToCloseWindows	OPT_CLOSE_WINDOW_BUTTONS	文字列のリスト	CloseSynchron メソッドでウィンドウを閉じるために使用するボタンを指定します。
ButttonsToConfirmDialogs	OPT_CLOSE_CONFIRM_BUTTONS	文字列のリスト	CloseSynchron メソッドでウィンドウを閉じる際に表示された確認ダイアログ ボックスを閉じるために使用するボタンを指定します。
CloseUnresponsiveApplications	OPT_KILL_HANGING_APPS	ブール値	無応答のアプリケーションを閉じるかどうかを指定します。 タイムアウトなどの理由で、エージェントとアプリケーション間の通信に失敗した場合に、アプリケーションが無応答になります。 複数のインスタンスを実行できないアプリケーションをテストする場合は、このオプションを TRUE に設定します。デフォルトで、これは FALSE です。
CloseWindowTimeout	OPT_CLOSE_WINDOW_TIMEOUT	数値	ウィンドウ閉じるための次の方式を試行する前に待機する時間をミリ秒で指定します。 最終的に失敗と判断する前に、4 種類の方式が Agent によって実行されます。つまり、閉じるのに失敗するまでにかかる合計時間は、指定した値の 4 倍の時間になります。
Compatibility	OPT_COMPATIBILITY	文字列	Silk4J の最新バージョンで特定の機能の動作が変更されている場合は、これらの機能について、指定した Silk4J バージョンの動作を使用することができます。 文字列の例： • 12

.NET オプション名	エージェント オプション名	定数の型	説明
			• 11.1 • 13.0.1 デフォルトでは、このオプションは設定されていません。
EnsureObjectIsActive	OPT_ENSURE_ACTIVE_OBJDEF	ブール値	ターゲット オブジェクトがアクティブであることを保証します。 デフォルトで、これは FALSE です。
EnableAccessibility	OPT_ENABLE_ACCESSIBILITY	ブール値	Win32 アプリケーションをテストしているときに、Silk4J がオブジェクトを認識できない場合、ユーザー補助を有効にするには TRUE に設定します。ユーザー補助は、オブジェクトの認識機能をクラス レベルで強化するためのものです。 ユーザー補助を無効にするには FALSE に設定します。 デフォルトで、これは FALSE です。 **注:** Mozilla Firefox と Google Chrome の場合、ユーザー補助は常に有効で、無効にすることはできません。
EnableMobileWebviewFallbackSupport	OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT	ブール値	デフォルトのブラウザー サポートではテストできないハイブリッド モバイル アプリケーションに対して、モバイル ネイティブ フォールバックのサポートを有効化します。 デフォルトで、これは FALSE です。
HangAppTimeOut	OPT_HANG_APP_TIME_OUT	数値	応答のない再生操作をキャンセルするまでのタイムアウト値を、**無応答のアプリケーションのタイムアウト** で指定します。 デフォルト値は、5000 ミリ秒です。
HighlightObjectDuringPlayback	OPT_REPLAY_HIGHLIGHT	ブール値	現在のオブジェクトが再生中にハイライトされるかどうかを指定します。 デフォルトで、これは FALSE です。つまり、デフォルトではオブジェクトはハイライトされません。
KeyboardEventDelay	OPT_KEYBOARD_INPUT_DELAY	数値	再生時のキー入力間の遅延をミリ秒で指定します。 テストするアプリケーションに応じて、選択する最適な値が異なることに注意してください。 たとえば、Web アプリケーションをテストする場合、1 ミリ秒に設定すると、ブラウザが極端に遅くなります。 ただし、この値を 0

.NET オプション名	エージェント オプション名	定数の型	説明
			(ゼロ) に設定すると、基本的なアプリケーション テストに失敗する可能性があります。
KeysToCloseDialogs	OPT_CLOSE_DIALOG_KEYS	文字列のリスト	CloseSynchron メソッドでウィンドウを閉じる際に表示されたダイアログ ボックスを閉じるキー シーケンスを指定します。 例：<ESC>、<Alt+F4>。
LocatorAttributesCaseSensitive	OPT_LOCATOR_ATTRIBUTES_CASE_SENSITIVE	ブール値	はい に設定すると、ロケーター属性名の大文字と小文字が区別されるようになり、いいえ に設定すると、ロケーター名の大文字と小文字が区別されなくなります。モバイル Web アプリケーションのロケーター属性の名前は、常に大文字と小文字の区別はされません。つまり、モバイル Web アプリケーションの記録や再生時に、このオプションは無視されます。
MenuItemsToCloseWindows	OPT_CLOSE_WINDOW_MENUS	文字列のリスト	CloseSynchron メソッドでウィンドウを閉じるために使用するメニュー項目を指定します。 例：「ファイル/*終了」、「ファイル/*中止」。
MouseEventDelay	OPT_MOUSE_INPUT_DELAY	数値	各マウス イベントの前に使用される遅延をミリ秒で指定します。
ObjectResolveRetryInterval	OPT_WAIT_RESOLVE_OBJDEF_RETRY	数値	再生中にオブジェクトが解決できなかった場合に、再試行する前に待機する時間をミリ秒で指定します。 ObjectResolveTimeout の値を超過した場合は、再試行は行われません。
ObjectResolveTimeout	OPT_WAIT_RESOLVE_OBJDEF	数値	再生中にオブジェクトの解決を待機する時間をミリ秒で指定します。 オブジェクトが解決されるとすぐに、Silk4J はこのオブジェクトを認識することができます。
PlaybackMode	OPT_REPLAY_MODE	数値	コントロールの再生方法を定義します。 「低レベル」を使用すると、マウスとキーボードを使用して各コントロールが再生されます。 「高レベル」を使用すると、API を使用して各コントロールが再生されます。 コントロールごとにデフォルトの再生モードが割り当てられています。 デフォルトの再生モードが選択されると、各コントロールごとのデフォルトの再生モードが使用されます。 デフォルト モードを使用すると、最も信頼できる結果が得られます。 「低レベル」または「高レベル」の再生を選択すると、すべてのコントロールの再生モードが選択した再生モードで上書きされます。

.NET オプション名	エージェント オプション名	定数の型	説明
			有効な値は、0、1、2 です。0 はデフォルト、1 は高レベル、2 は低レベルです。デフォルトで、これは 0 です。
PostReplayDelay	OPT_POST_REPLAY_DELAY	数値	関数の呼び出しまたはプロパティの設定後に待機する時間をミリ秒で指定します。
RemoveFocusOnCaptureText	OPT_REMOVE_FOCUS_ON_CAPTURE_TEXT	ブール値	はい に設定すると、テキストのキャプチャ中にテスト対象アプリケーションからフォーカスが外れます。デフォルトでは いいえ に設定されており、テスト対象アプリケーションにフォーカスが残ります。テキストのキャプチャは、次のメソッドによる記録および再生中に実行されます。 • TextClick • TextCapture • TextExists • TextRect
SyncTimeout	OPT_SYNC_TIMEOUT	数値	オブジェクトが準備完了状態になる最大時間をミリ秒で指定します。 **注:** Silk Test 13.0 より前のバージョンの Silk Test からアップグレードする場合、OPT_XBROWSER_SYNC_TIMEOUT オプションが設定されていると、**オプション** ダイアログ ボックスに OPT_SYNC_TIMEOUT のデフォルト値が表示されますが、タイムアウトは定義した値に設定されています。
TransparentClasses	OPT_TRANSPARENT_CLASSES	文字列のリスト	オブジェクト階層を単純化し、テストスクリプトや関数のコードの行の長さを短くするために、次のテクノロジの確実に不要なクラスに対するコントロールを抑制できます。 • Win32 • Java AWT/Swing • Java SWT/Eclipse • Windows Presentation Foundation (WPF) 記録や再生中に無視したいクラスの名前を指定します。
	OPT_WPF_CHECK_DISPATCHER_FOR_IDLE	ブール値	WPF アプリケーションによっては、コントロールの実装方法が影響して、Silk Test 同期が機能しない場合があります。これにより、WPF アプリケーションがアイドルであることを Silk4J が

.NET オプション名	エージェント オプション名	定数の型	説明
			認識できなくなります。このオプションを FALSE に設定すると、WPF 同期が無効になり、WPF アプリケーションを制御するスレッドである WPF ディスパッチャーを Silk4J が確認しないようになります。このオプションを FALSE に設定すると、一部の WPF アプリケーションで発生する同期問題を解決できます。デフォルトで、これは TRUE です。
WPFCustomClasses	OPT_WPF_CUSTOM_CLASSES	文字列のリスト	記録や再生の対象にしたい WPF クラスの名前を指定します。たとえば、*MyGrid* というカスタム クラスが WPF Grid クラスから継承された場合、*MyGrid* カスタム クラスのオブジェクトは記録や再生に使用できません。Grid クラスはレイアウト目的のためにのみ存在し、機能テストとは無関係であるため、Grid オブジェクトは記録や再生に使用できません。この結果、Grid オブジェクトはデフォルトでは公開されません。機能テストに無関係なクラスに基づいたカスタム クラスを使用するには、カスタム クラス (この場合は *MyGrid*) を **OPT_WPF_CUSTOM_CLASSES** オプションに追加します。これによって、記録、再生、検索、プロパティの検証など、すべてのサポートされる操作を指定したクラスに対して実行できるようになります。
WPFPrefillItems	OPT_WPF_PREFILL_ITEMS	ブール値	記録および再生中に 、WPFComboBox や WPFListBox のような WPFItemsControl 内の項目を事前に入力するかどうかを定義します。WPF 自体が特定のコントロールの項目を遅延読み込みするため、項目がビューにスクロールされない場合、それらの項目は Silk4J では使用できません。ビューにスクロールされないとアクセスできない項目にアクセスするには、事前入力をオンにします。これはデフォルトの設定です。ただし、一部のアプリケーションでは Silk4J によってバックグランドで項目が事前入力されると問題が発生し、そのためアプリケーションがクラッシュすることがあります。この場合、事前入力をオフにします。
XbrowserEnableIframeSupport	OPT_XBROWSER_ENABLE_IFRAME_SUPPORT	ブール値	ブラウザーの iframe および frame のサポートを有効にするかどうかを指定

.NET オプション名	エージェント オプション名	定数の型	説明
			します。Web アプリケーションの iframe の内容を無視して良い場合、iframe のサポートを無効にすると再生速度が速くなる可能性があります。たとえば、多くの広告が表示される Web ページやモバイル ブラウザーでテストする際に iframe サポートを無効にすると、再生パフォーマンスが大幅に向上する可能性があります。このオプションは、Internet Explorer では無視されます。このオプションは、デフォルトで有効になっています。
XBrowserExcludeIFrames	OPT_XBROWSER_EXCLUDE_IFRAMES	文字列	リストの各項目で、属性名と対応する値を定義します。項目のどれにも一致しない iframe と frame は、テスト対象に含まれます。ワイルドカードを使用できます。たとえば、"src:*advertising*" という項目を定義すると、<IFRAME src=http://my.domain/advertising-banner.html> がテスト対象から除外されます。このオプションは、Internet Explorer では無視されます。リストが空の場合は、すべての iframe と frame がテストの対象となります。複数のエントリをカンマで区切って指定します。
XBrowserFindHiddenInputFields	OPT_XBROWSER_FIND_HIDDEN_INPUT_FIELDS	ブール値	非表示入力フィールドを表示するかどうかを指定します（タグに type="hidden" を指定した HTML フィールドです）。デフォルト値は、TRUE です。
XBrowserIncludeIFrames	OPT_XBROWSER_INCLUDE_IFRAMES	文字列	リストの各項目で、属性名と対応する値を定義します。項目のどれにも一致しない iframe と frame は、テスト対象から除外されます。ワイルドカードを使用できます。たとえば、"name:*form" という項目を定義すると、<IFRAME name="user-form" src=...> がテスト対象に含まれます。このオプションは、Internet Explorer では無視されます。リストが空の場合は、すべての iframe と frame がテストの対象となります。複数のエントリをカンマで区切って指定します。
XBrowserSynchronizationMode	OPT_XBROWSER_SYNC_MODE	文字列	サポートする同期モード (HTML または AJAX) を設定します。 HTML モードを使用すると、すべての HTML ドキュメントが対話的な状態になることが保証されます。 このモードでは、単純な Web ページをテストすることがで

.NET オプション名	エージェント オプション名	定数の型	説明
			きます。 Java Script が含まれるより複雑なシナリオが使用される場合は、WaitForObject、WaitForProperty、WaitForDisappearance、または WaitForChildDisappearance などの同期関数を使用して手動でスクリプトを記述することが必要になる可能性があります。 AJAX モードを使用すると、同期関数を手動で記述する必要はなくなります。 デフォルト値は、**AJAX** に設定されています。
XBrowserSynchronizationTimeout	OPT_XBROWSER_SYNC_TIMEOUT	数値	オブジェクトが準備完了状態になる最大時間をミリ秒で指定します。 **注:** 推奨されません。代わりに OPT_SYNC_TIMEOUT オプションを使用してください。
XBrowserSynchronizationURLExcludes	OPT_XBROWSER_SYNC_EXCLUDE_URLS	文字列	ページ同期中に除外するサービスまたは Web ページの URL を指定します。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 **同期除外リスト** 設定で指定します。 *http://test.com/timeService* のように URL 全体を入力したり、*timeService* のように URL の一部を入力します。 次のようにカンマで項目を区切ります。 `desktop.setOption(Commonoptions.XBrowserSynchronizationURLExcludes, { "fpdownload.macromedia.com", "fpdownload.adobe.com", "download.microsoft.com" });`

Open Agent のポート番号

Open Agent が起動すると、Silk4J およびテストするアプリケーションに対して、使用可能なポートがランダムに割り当てられます。ポート番号は Information Service に登録されます。Silk4J は、Open Agent に接続するために使用するポートを決定するために Information Service に接続します。

Information Service は適切なポートと通信し、Silk4J はそのポートに接続します。通信は、エージェントと Silk4J との間で直接行われます。

デフォルトでは、ポート 22901 を使用して Open Agent は Information Service と通信します。デフォルト ポートが利用可能でない場合に機能する代替ポートとして、Information Service の追加のポートを構成できます。デフォルトでは、Information Service は、代替ポートとして 2966、11998、および 11999 を使用します。

大抵の場合、手動でポート番号を設定する必要はありません。しかし、ポート番号が競合したり、ファイアウォールとの問題があったりした場合には、そのマシンや Information Service に対してポート番号を設定する必要があります。各マシンごとに異なるポート番号を使用することも、すべてのマシンに対して同じ番号を使用することも可能です。

Information Service に接続するポートの構成

このタスクを開始する前に、Silk Test Open Agent を停止します。

大抵の場合、手動でポート番号を設定する必要はありません。Information Service はポート構成を自動的に処理します。エージェントとの接続には、Information Service のデフォルトのポートを使用します。これにより、Information Service によって、エージェントが使用するポートに通信が転送されます。

Information Service のデフォルトのポートは 22901 です。デフォルトのポートが使用可能であれば、ポート番号を指定せずに単純に hostname だけを入力できます。ポート番号を指定する場合には、Information Service のデフォルトのポートまたは追加したポートの 1 つと一致していることを確認ください。間違ったポートが指定されていると、通信に失敗します。

必要に応じて、Information Service に接続するためにすべてのクライアントが使用するポート番号を変更できます。

1. infoservice.properties.sample ファイルに移動し、開きます。
   - Microsoft Windows システムでは、このファイルは C:¥ProgramData¥Silk¥Silk Test¥conf にあります。ここで、「C:¥ProgramData」は、Windows システムでデフォルトで設定されている環境変数 ALLUSERSPROFILE の値です。
   - macOS では、このファイルは /Users/<ユーザー>/.silk/silktest/conf にあります。

   このファイルには、コメントとサンプルの代替ポート設定が含まれています。
2. 代替ポートの値を変更します。

   大抵の場合、ファイアウォールとの問題を避けるために、特定のポートに通信を強制するように Information Service ポートの設定を構成します。

   ポート番号は、1 から 65535 の間の任意の数値を指定できます。
   - infoservice.default.port : Information Service が実行されているデフォルト ポートです。デフォルトでは、このポートは 22901 に設定されています。
3. ファイルを infoservice.properties という名前で保存します。
4. Open Agent、Silk Test クライアント、およびテストするアプリケーションを再起動します。

Open Agent に接続するポートの構成

このタスクを開始する前に、Silk Test Open Agent を停止します。

大抵の場合、手動でポート番号を設定する必要はありません。Information Service はポート構成を自動的に処理します。エージェントとの接続には、Information Service のデフォルトのポートを使用します。これにより、Information Service によって、エージェントが使用するポートに通信が転送されます。

必要に応じて、Silk Test クライアントまたはテストするアプリケーションが Open Agent に接続するために使用するポート番号を変更します。

1. agent.properties.sample ファイルに移動し、開きます。
   デフォルトでは、このファイルは次の場所にあります：%APPDATA%¥Silk¥SilkTest¥conf。 大抵の場合、C:¥Users¥<ユーザー名>¥AppData¥Silk¥SilkTest¥conf になります。ここで、<ユーザー名> は、現在のユーザー名に一致します。
2. 代替ポートの値を変更します。
   大抵の場合、ポートの競合を解決するためにポートの設定を構成します。

   注: 各ポート番号は一意でなければなりません。エージェントのポート番号が Information Service のポート設定とは異なることを確認してください。

   ポート番号は、1 から 65535 の間の任意の数値を指定できます。

   ポートの設定には次のものがあります：

   - agent.vtadapter.port： テストの実行時に、Silk Test Workbench と Open Agent 間の通信を制御します。
   - agent.xpmodule.port： テストの実行時に、Silk Test Classic とエージェント間の通信を制御します。
   - agent.autcommunication.port： Open Agent とテストするアプリケーション間の通信を制御します。
   - agent.rmi.port： Open Agent と Silk4J 間の通信を制御します。
   - agent.ntfadapter.port： Open Agent と Silk4NET 間の通信を制御します。

   注: Apache Flex のテスト時に使用されるポートは、この構成ファイルでは制御できません。Flex アプリケーションのテストで割り当てられるポート番号は、6000 から始まり、各 Flex アプリケーションがテストされる度に 1 ずつ増加していきます。Flex テスト用に開始ポートを構成することはできません。
3. ファイルを agent.properties という名前で保存します。
4. Open Agent、Silk Test クライアント、およびテストするアプリケーションを再起動します。

Silk Test Recorder に接続するポートの構成

このタスクを開始する前に、Silk Test Open Agent を停止します。

大抵の場合､手動でポート番号を設定する必要はありません。Information Service はポート構成を自動的に処理します。エージェントとの接続には､Information Service のデフォルトのポートを使用します。これにより、Information Service によって、エージェントが使用するポートに通信が転送されます。

必要に応じて、Silk Test Classic、Silk4J、または Silk4NET が Silk Test Recorder に接続するのに使用するポート番号を変更します。

1. recorder.properties.sample ファイルに移動し、開きます。
   デフォルトでは、このファイルは次の場所にあります：%APPDATA%¥Silk¥Silk Test¥conf。大抵の場合、C:¥Documents and Settings¥<ユーザー名>¥AppData¥Silk¥SilkTest¥conf になります。ここで、<ユーザー名> は、現在のユーザー名に一致します。
2. recorder.api.rmi.port を、使用するポートに変更します。
   ポート番号は、1 から 65535 の間の任意の数値を指定できます。

   注: 各ポート番号は一意でなければなりません。エージェントのポート番号が Recorder や Information Service のポート設定とは異なることを確認してください。
3. ファイルを recorder.properties という名前で保存します。
4. Open Agent、Silk Test クライアント、およびテストするアプリケーションを再起動します。

Open Agent を使用したリモート テスト

リモート マシン上に Silk4J をインストールして、ローカル マシンにインストールされている Silk4J からリモート ロケーション上のアプリケーションをテストすることができます。

注: リモート マシンに接続されているモバイル デバイスや、リモート マシン上のエミュレータまたはシミュレータ上のモバイル アプリケーションをテストする場合や、Apple Safari またはリモート Microsoft Edge 上の Web アプリケーションをテストする場合には、リモート Open Agent ではなく、リモート Silk Test Information Service を使用する必要があります。

リモート Open Agent を使用したテスト

Silk4J を使用してリモート マシン上のアプリケーションに対してテストを再生するには、次のタスクを実行します。

1. ローカル マシン上のアプリケーションに対してテストを作成します。
2. リモート マシン上に Open Agent をインストールします。
   詳細については、『*Silk Test* インストール ガイド』を参照してください。
3. リモート マシン上で Open Agent を開始します。
4. テスト スクリプトで、リモート マシンを新しいデスクトップとして指定します。
   たとえば、次の行をスクリプトに追加します。

```
private Desktop desktop = new Desktop("<IP address or network name of remote machine>");
```

基本状態

アプリケーションの基本状態とは、各テスト ケースの実行開始前にアプリケーションに想定される既知の安定した状態です。アプリケーションは、各テスト ケースの実行が終了したあとに基本状態に戻る場合もあります。大抵の場合この状態は、アプリケーションを最初に起動したときの状態になります。

アプリケーションに対してクラスを作成するとき、Silk4J は自動的に基本状態を作成します。

基本状態はテストの整合性を保障するための重要な一因です。各テスト ケースが安定した基本状態から開始することができることを保障することによって、あるテスト ケースのエラーによって、後続のテスト ケースが失敗しないことを保障することができます。

Silk4J は、次の段階の間に、アプリケーションがその基本状態にあることを自動的に保障します。

- テストの実行前
- テストの実行中
- テストが成功裏に完了した後

注: 定義済みの基本状態を使って Web アプリケーションをテストするときに、ブラウザー アプリケーション構成を複数追加しないでください。

基本状態は、次の方法で編集できます。

- UI： 記録時と再生時の両方で AUT の開始方法を指定したい場合など。
- スクリプト： スクリプトでテストを再生するときに、指定した方法で AUT を開始したい場合など。

ユーザー インターフェイスからの基本状態の変更

Silk4J が記録や再生時にテスト対象アプリケーションを開始する方法を指定する基本状態を、ユーザー インターフェイスから編集できます。基本状態では、AUT の実行可能ファイルの場所、作業ディレクトリや、Web アプリケーションの URL や接続文字列などを指定できます。たとえば、ステージング Web サイトで既に実行したテストを、プロダクション Web サイトで実行する場合には、基本状態の URL を変更すれば、新しい Web サイトに対してテストを実行することができます。

注: 再生時に特定のテストに対してのみ Silk4J がテスト対象アプリケーション (AUT) を開始するように指定する場合は、テストを含むスクリプトで基本状態を編集します。詳細については、「スクリプトでの基本状態の変更」を参照してください。

ユーザー インターフェイスから基本状態を編集するには：

1. Silk Test ツールバー アイコン の隣にあるドロップダウン矢印 をクリックして、**アプリケーション構成の編集** を選択します。 **アプリケーション構成の編集** ダイアログ ボックスが開き、既存のアプリケーション構成がリストされます。
2. 変更するアプリケーション構成の右側にある **編集** をクリックします。
3. Web アプリケーションまたはモバイル Web アプリケーションをテストするには、現在のプロジェクトに対してアプリケーション構成が設定されていない場合は、リストからインストール済みのブラウザまたはモバイル ブラウザのうちの 1 つを選択します。

   変更 をクリックして **アプリケーションの選択** ダイアログ ボックスを開き、使用するブラウザーを選択することもできます。
4. 実行可能ファイルを指定するには、**実行可能ファイル** フィールドに実行可能ファイルへの完全パスを入力します。

   注: Web アプリケーションをテストする場合に、ブラウザーの実行可能ファイルを指定する場合は、ブラウザーの種類からカスタムを選択します。

たとえば、Mozilla Firefox を起動するには、「C:¥Program Files (x86)¥Mozilla Firefox ¥firefox.exe」を入力します。

5. コマンド ライン引数を指定するには、**コマンド ライン引数** フィールドに引数を入力します。

   注: Web アプリケーションをテストする場合に、コマンド ライン引数を指定してブラウザーを開始する場合は、ブラウザーの種類からカスタムを選択します。

   たとえば、Mozilla Firefox を *myProfile* プロファイルを指定して起動するには、「-p myProfile」を入力します。

6. テストするアプリケーションがディレクトリに依存する場合は、**作業ディレクトリ** フィールドにディレクトリへのパスを指定します。

   たとえば、Java アプリケーションを起動するバッチ ファイルを使用する場合には、バッチ ファイルは JAR ファイルを相対パスで参照することができます。この場合、正しく相対パスが処理されるように作業ディレクトリを指定します。

7. デスクトップ アプリケーションをテストする場合は、**ロケーター** フィールドにアプリケーションのメイン ウィンドウを指定します。
   たとえば、ロケーターは、/Shell[@caption='Swt Test Application'] のようになります。

8. 実行可能ファイル パターンを指定して、デスクトップ アプリケーションをテストする場合は、**実行可能ファイル パターン** テキスト ボックスに、テストするデスクトップ アプリケーションの実行可能ファイルの名前とファイルへのパスを入力します。
   たとえば、電卓を指定する場合には、*¥calc.exe と入力します。

9. デスクトップ アプリケーションをテストし、実行可能ファイルと一緒にコマンド ライン パターンを使用したい場合には、コマンド ライン パターンを **コマンド ライン パターン** テキスト ボックスに入力します。

   コマンド ラインの使用は、Java アプリケーションに対して特に有益です。これは、ほとんどの Java プログラムが javaw.exe を使用して実行されるためです。つまり、典型的な Java アプリケーションに対してアプリケーション構成を作成する場合、実行可能パターンには *¥javaw.exe が使用され、このパターンはすべての Java プロセスに一致します。このような場合、コマンド ライン パターンを使用して、該当するアプリケーションのみがテストに対して有効化されるようにします。たとえば、アプリケーションのコマンド ラインが **com.example.MyMainClass** で終わる場合には、コマンド ライン パターンに ***com.example.MyMainClass** を使用します。

10. Web アプリケーションをテストする場合は、**移動する URL** テキスト ボックスに、Web アプリケーションのアドレスを入力します。

11. 省略可能 : あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。
    たとえば、Apple Safari 上の Web アプリケーションを Apple iPhone 7 の画面と同じ大きさのブラウザー ウィンドウでテストするには、リストから **Apple iPhone 7** を選択します。

12. 省略可能 : ブラウザー ウィンドウの **向き** を選択します。

13. 省略可能 : **ブラウザー サイズの編集** をクリックすると、新しいブラウザー サイズを指定したり、**ブラウザー サイズ** リストに表示するブラウザー サイズを選択することができます。

14. リモート ロケーション上の Web アプリケーションまたはモバイル ネイティブ アプリケーション (Mac に接続されているモバイル デバイスなど) をテストする場合に、リモート ロケーションを変更する場合は、**変更** をクリックして **アプリケーションの選択** ダイアログ ボックスを開き、**リモート ロケーションの編集** をクリックします。

15. モバイル アプリケーションや Apple Safari 上の Web アプリケーションをテストする場合は、**接続文字列** テキスト ボックスに接続文字列を入力します。

    詳細については、「接続文字列」を参照してください。

16. WebDriver ベースのブラウザーのケイパビリティを編集する場合は、**接続文字列** テキスト ボックスを使用できます。
    たとえば、Google Chrome のブラウザー ウィンドウを最大化して開始する場合には、**接続文字列** テキスト ボックスに次のように入力します。

    ```
    chromeOptions={"args":["--start-maximized"]}
    ```

詳細については、「*WebDriver ベースのブラウザーのケイパビリティの設定*」を参照してください。

17. モバイル ネイティブ アプリケーションのテストを行う場合は、**モバイル アプリ** テキスト ボックスに、テストするモバイル アプリケーションの名前を入力します。
18. **OK** をクリックします。
19. テスト対象アプリケーションの起動に時間がかかる場合は、再生オプションのアプリケーション準備完了タイムアウトの値を増やしてください。

アプリケーションが実行していない場合は、基本状態を実行することで開始されます。アプリケーションがすでに実行中の場合には、Silk4J によってアプリケーションの別のインスタンスは開始されません。

複数のアプリケーション構成がテストに含まれており、基本状態に関連付けられているオブジェクト以外のアプリケーションまたは Web ページを変更する場合は、基本状態をオフにすることができます。このことは、変更を記録したり再生したりするときに基本状態が使用されないこと意味します。このため、アプリケーションまたは Web ページを起動するステップをテスト内に記録する必要があります。たとえば、Web ページをテストする場合は、テスト内で Internet Explorer を起動します。

注: 定義済みの基本状態を使って Web アプリケーションをテストするときに、ブラウザー アプリケーション構成を複数追加しないでください。

スクリプトでの基本状態の変更

Silk4J が再生時にテスト対象アプリケーションを開始する方法を指定する基本状態を、スクリプトで編集できます。基本状態では、AUT の実行可能ファイルの場所、作業ディレクトリ、Web アプリケーションの URL や接続文字列などを指定できます。たとえば、ステージング Web サイトで既に実行したテストを、プロダクション Web サイトで実行する場合には、基本状態の URL を変更すれば、新しい Web サイトに対してテストを実行することができます。

注: Silk4J が記録や再生時にテスト対象アプリケーション (AUT) を開始する方法を指定するには、ユーザー インターフェイスから基本状態を編集します。詳細については、「基本状態の変更」を参照してください。

スクリプトで基本状態を編集するには：

1. スクリプトを開きます。
2. baseState メソッドを変更します。

   基本状態の作成と実行との間にコードを追加できます。

   ```
   // Web ページ 'demo.borland.com/InsuranceWebExtJS' へ移動します。
   BrowserBaseState baseState = new BrowserBaseState();
   // <-- ここに変更を挿入します。
   baseState.execute();
   ```
3. テストするアプリケーションの実行可能ファイルの名前とファイルへのパスを指定する場合は、次のコードを使用します。

   ```
   baseState.setExecutable(executable);
   ```

   たとえば、電卓を指定する場合は、次のように入力します。

   ```
   baseState.setExecutable("C:¥¥Windows¥¥SysWOW64¥¥calc.exe");
   ```

   Mozilla Firefox を指定する場合は、次のように入力します。

   ```
   baseState.setExecutable("C:¥¥Program Files (x86)¥¥Mozilla Firefox¥¥firefox.exe");
   ```
4. コマンド ライン引数を指定する場合は、次のコードを使用します。

   ```
   baseState.setCommandLineArguments(commandLineArguments);
   ```

   たとえば、Mozilla Firefox を *myProfile* プロファイルを指定して起動する場合は、次のように入力します。

   ```
   baseState.setCommandLineArguments("-p myProfile");
   ```

5. 他の作業ディレクトリを指定する場合は、次のコードを使用します。

```
baseState.setWorkingDirectory(workingDirectory);
```

6. 実行可能ファイル パターンを使用する場合には、次のコードを使用します。

```
baseState.setExecutablePattern(executablePattern);
```

たとえば、電卓に対して実行可能ファイル パターンを指定する場合は、次のように入力します。

```
baseState.setExecutablePattern("*¥¥calc.exe");
```

7. 実行可能ファイルと一緒にコマンド ライン パターンを使用する場合は、次のコードを使用します。

```
baseState.setCommandLinePattern(commandLinePattern);
```

コマンド ラインの使用は、Java アプリケーションに対して特に有益です。これは、ほとんどの Java プログラムが javaw.exe を使用して実行されるためです。つまり、典型的な Java アプリケーションに対してアプリケーション構成を作成する場合、実行可能パターンには *¥javaw.exe が使用され、このパターンはすべての Java プロセスに一致します。このような場合、コマンド ライン パターンを使用して、該当するアプリケーションのみがテストに対して有効化されるようにします。

たとえば、アプリケーションのコマンド ラインが **com.example.MyMainClass** で終わる場合には、コマンド ライン パターンに ***com.example.MyMainClass** を使用します。

```
baseState.setCommandLinePattern("*com.example.MyMainClass");
```

8. Web アプリケーションまたはモバイル Web アプリケーションをテストする場合、現在のプロジェクトに対してアプリケーション構成が設定されていない場合は、インストール済みのブラウザまたはモバイル ブラウザから 1 つを指定します。
たとえば、Google Chrome を指定する場合は、次のように入力します。

```
baseState.setBrowserType(BrowserType.GoogleChrome);
```

9. Web アプリケーションまたはモバイル Web アプリケーションをテストする場合、現在のプロジェクトに対してアプリケーション構成が設定されていない場合は、テストする Web アプリケーションのアドレスを指定します。

```
baseState.setUrl(url);
```

たとえば、次のように入力します。

```
baseState.setUrl("demo.borland.com/InsuranceWebExtJS/");
```

10. デスクトップ ブラウザー上の Web アプリケーションをテストする場合は、ブラウザー ウィンドウの幅と高さを指定できます。

```
baseState.setViewportHeight(viewportHeight);
baseState.setViewportWidth(viewportWidth);
```

11. リモート ロケーション上の Web アプリケーション、またはモバイル ネイティブ アプリケーションをテストする場合は、接続文字列を指定します。

```
new  MobileBaseState(connectionString);
```

接続文字列についての詳細は、「リモート デスクトップ ブラウザーの接続文字列」または「モバイル デバイスの接続文字列」を参照してください。

12. Mozilla Firefox または Google Chrome のケイパビリティを編集する場合にも、接続文字列を使用できます。
たとえば、Mozilla Firefox のダウンロード フォルダーを設定する場合は、次のように入力します。

```
baseState.setConnectionString(
   "moz:firefoxOptions="
   +  "{"
   +  "  ¥"prefs¥":  {"
   +  "            ¥"browser.download.dir¥":¥"C:¥¥¥¥Download¥¥¥¥¥""
   +  "        }"
   +  "};");
```

詳細については、「*WebDriver* ベースのブラウザーのケイパビリティの設定」を参照してください。

基本状態の実行

アプリケーションに対してテストの記録を開始する前に基本状態を実行することで、記録するすべてのアプリケーションを起動して、記録に適した状態にすることができます。

基本状態を実行するには：

Silk4J > **基本状態の実行** をクリックします。

アプリケーションの種類に応じて、Silk4J は次のアクションを実行します。

- 現在のプロジェクトで定義されているアプリケーション構成に対応するすべてのアプリケーションのアプリケーション構成を実行します。
- Web アプリケーションの場合、Silk4J は指定したブラウザで指定した URL の Web アプリケーションを開きます。
- モバイル Web アプリケーションの場合、Silk4J は指定したモバイル デバイスまたはエミュレータ上の指定したブラウザで指定した URL を開きます。
- モバイル ネイティブ アプリケーションの場合、Silk4J は指定したモバイル デバイスまたはエミュレータ上で指定したアプリを開きます。指定したアプリが指定したモバイル デバイスまたはエミュレータ上にインストールされていない場合、Silk4J はアプリをインストールしてから開きます。

アプリケーション構成

アプリケーション構成は、テストするアプリケーションに Silk4J が接続する方法を定義します。Silk4J は、基本状態を作成するときに、アプリケーション構成を自動的に作成します。しかし、アプリケーション構成を追加したり、変更や削除をすることが必要になる場合があります。たとえば、データベースを変更するアプリケーションをテストしているときに、データベースの内容を確認するためにデータベースのビューアー ツールを使用する場合には、そのデータベースのビューアー ツール用のアプリケーション構成を追加する必要があります。

- Windows アプリケーションの場合、アプリケーション構成には以下が含まれます。
  - 実行可能ファイル パターン

    このパターンに一致するすべてのプロセスは、テストに対して有効化されます。たとえば、Internet Explorer の実行可能パターンは *¥IEXPLORE.EXE です。実行可能ファイルの名前が IEXPLORE.EXE で、任意のディレクトリに置かれているプロセスはすべて有効化されます。
  - コマンド ライン パターン

    コマンドライン パターンは、テストを行うために有効化されるプロセスの制約に使用される補足パターンで、コマンドライン引数の一部 (実行可能ファイル名の後ろ部分) をマッピングすることにより行います。コマンド ライン パターンを含むアプリケーション構成では、実行可能パターンとコマンド ライン パターンの両方に一致するプロセスのみが、テストに対して有効化されます。コマンド ライン パターンが定義されていない場合は、指定された実行可能ファイル パターンを持つすべてのプロセスが有効化されます。コマンド ラインの使用は、Java アプリケーションに対して特に有益です。これは、ほとんどの Java プログラムが javaw.exe を使用して実行されるためです。つまり、典型的な Java アプリケーションに対してアプリケーション構成を作成する場合、実行可能パターンには *¥javaw.exe が使用され、このパターンはすべての Java プロセスに一致します。このような場合、コマンド ライン パターンを使用して、該当するアプリケーションのみがテストに対して有効化されるようにします。たとえば、アプリケーションのコマンド ラインが **com.example.MyMainClass** で終わる場合には、コマンド ライン パターンに ***com.example.MyMainClass** を使用します。
- ローカル マシン上のデスクトップ ブラウザの Web アプリケーションの場合、アプリケーション構成にはブラウザの種類だけが含まれます。

  注: コマンド ライン引数を指定してブラウザを開始したり、ブラウザの作業ディレクトリや実行可能ファイルを指定するには、ブラウザの種類でカスタムを選択します。詳細については、「基本状態の変更」を参照してください。
- リモート マシン上の Apple Safari や Microsoft Edge の Web アプリケーションの場合、アプリケーション構成には以下が含まれます。
  - ブラウザの種類
  - 接続文字列
- モバイル ブラウザの Web アプリケーションの場合、アプリケーション構成には以下が含まれます。
  - ブラウザの種類
  - 接続文字列
- ネイティブ モバイル アプリケーションの場合、アプリケーション構成には以下が含まれます。
  - 接続文字列
  - シンプルなアプリケーション名。モバイル デバイス上で複数のアプリケーションが同じ名前を持つ場合は、アプリケーションの完全修飾名が使用されます。

注: 定義済みの基本状態を使って Web アプリケーションをテストするときに、ブラウザー アプリケーション構成を複数追加しないでください。

アプリケーション構成の変更

アプリケーション構成は、テストするアプリケーションに Silk4J が接続する方法を定義します。Silk4J は、基本状態を作成するときに、アプリケーション構成を自動的に作成します。しかし、アプリケーション構成を追加したり、変更や削除をすることが必要になる場合があります。たとえば、データベースを変更するアプリケーションをテストしているときに、データベースの内容を確認するためにデータベースのビューアー ツールを使用する場合には、そのデータベースのビューアー ツール用のアプリケーション構成を追加する必要があります。

1. Silk Test ツールバー アイコン  の隣にあるドロップダウン矢印 をクリックして、**アプリケーション構成の編集** を選択します。 **アプリケーション構成の編集** ダイアログ ボックスが開き、既存のアプリケーション構成がリストされます。
2. アプリケーション構成をさらに追加するには、**アプリケーション構成の追加** をクリックします。

   注: 定義済みの基本状態を使って Web アプリケーションをテストするときに、ブラウザー アプリケーション構成を複数追加しないでください。

   アプリケーションの選択 ダイアログ ボックスが開きます。タブを選択してからテストするアプリケーションを選択して **OK** をクリックします。
3. アプリケーション構成を削除するには、該当するアプリケーション構成の隣にある **削除** をクリックします。
4. アプリケーション構成を編集するには、**編集** をクリックします。
5. デスクトップ アプリケーションのテストを行う場合は、**実行可能ファイル パターン** テキスト ボックスに、テストするデスクトップ アプリケーションの実行可能ファイルの名前とファイルへのパスを入力します。

   たとえば、電卓を指定する場合には、*¥calc.exe と入力します。
6. デスクトップ アプリケーションをテストし、実行可能ファイルと一緒にコマンド ライン パターンを使用したい場合には、コマンド ライン パターンを **コマンド ライン パターン** テキスト ボックスに入力します。
7. Web アプリケーションまたはモバイル Web アプリケーションをテストするには、現在のプロジェクトに対してアプリケーション構成が設定されていない場合は、リストからインストール済みのブラウザまたはモバイル ブラウザのうちの 1 つを選択します。

   変更 をクリックして **アプリケーションの選択** ダイアログ ボックスを開き、使用するブラウザーを選択することもできます。
8. Web アプリケーションをテストする場合は、**移動する URL** テキスト ボックスに、テストの開始時に表示する Web ページの Web アドレスを入力します。
9. Web アプリケーションのビジュアル ブレークポイントを検出する場合は、**分析** をクリックします。

   詳細については、「ビジュアル ブレークポイントの検出」を参照してください。
10. 省略可能 : あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。たとえば、Apple Safari 上の Web アプリケーションを Apple iPhone 7 の画面と同じ大きさのブラウザー ウィンドウでテストするには、リストから **Apple iPhone 7** を選択します。
11. 省略可能 : ブラウザー ウィンドウの **向き** を選択します。
12. モバイル アプリケーションや Apple Safari 上の Web アプリケーションをテストする場合は、**接続文字列** テキスト ボックスに接続文字列を入力します。

    詳細については、「モバイル デバイスの接続文字列」を参照してください。
13. **OK** をクリックします。

[アプリケーションの選択] ダイアログ ボックス

アプリケーションの選択 ダイアログ ボックスを使用して、テストしたアプリケーションを選択し、アプリケーションとオブジェクト マップを関連付けたり、アプリケーション構成をテストに追加したりします。アプリケーションの種類は、ダイアログ ボックスのタブとしてリストされます。使用したいアプリケーションの種類に対応したタブを選択します。

Windows システムで実行中のすべての Microsoft Windows アプリケーションの一覧が表示されます。リストから項目を選択して、**OK** をクリックします。

キャプションを持たないプロセスを表示しない チェック ボックスを使用して、キャプションを持たないアプリケーションを一覧から除去します。

Web 利用可能なすべてのブラウザーの一覧が表示されます (任意の接続済みモバイル デバイス上のモバイル ブラウザーや Mac 上の Apple Safari を含む)。**移動する URL の入力** テキスト ボックスに、開く Web ページを指定します。選択したブラウザのインスタンスが既に実行されている場合、**実行中のブラウザーの URL を使用する** をクリックして、実行中のブラウザ インスタンスに現在表示されている URL の記録を行うことができます。あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。ブラウザー サイズを選択する場合は、選択したブラウザーの **向き** も選択できます。

注: 定義済みの基本状態を使って Web アプリケーションをテストするときに、ブラウザー アプリケーション構成を複数追加しないでください。

モバイル 利用可能なすべてのモバイル デバイスとすべての実行中の Android エミュレーターの一覧が表示されます (リモート ロケーションに接続されたデバイスを含む)。モバイル ネイティブ アプリケーションをテストする場合は、このタブを選択します。選択したモバイル デバイス上で現在実行中のモバイル アプリケーション (アプリ) をテストするために選択したり、テストするアプリの名前やファイルを手動で参照したり指定することができます。

リモート ロケーションの編集

リモート ロケーション ダイアログ ボックスを使用すると、リモート ロケーション上のブラウザやモバイル デバイスを、テストするアプリケーションのセットに追加できます。

1. **Silk4J** > **リモート ロケーションの編集** をクリックします。**リモート ロケーション** ダイアログ ボックスが表示されます。
2. リモート ロケーションを追加するには、次の操作を実行します。
   a) **ロケーションの追加** の右側にある矢印をクリックして、Silk Test Information Service を使用してリモート ロケーションを追加するのか、Silk Central を追加するのかを指定します。

      注: Silk Central は 1 つだけリモート ロケーションとして設定できます。Silk Central との統合が既に設定されている場合は、リモート ロケーション リストに Silk Central が表示されます。
   b) **ロケーションの追加** をクリックします。**ロケーションの追加** ダイアログ ボックスが表示されます。
   c) Silk4J がリモート マシン上の Information Service に接続するためのリモート ロケーションの URL とポートを、**ホスト** フィールドに入力します。

      デフォルトのポートは 22901 です。
   d) 省略可能：**名前** フィールドにリモート ロケーションの名前を入力します。
3. 既存のリモート ロケーションを編集するには、**編集** をクリックします。
4. リモート ロケーションを削除するには、**削除** をクリックします。

5. 省略可能：**アプリケーションの選択** ダイアログに表示されるブラウザーやデバイスの数を減らすには、**このロケーション上のデバイスとブラウザーを非表示** をクリックします。そのリモート ロケーションにインストールされたブラウザーや接続されたデバイスが、**アプリケーションの選択** ダイアログに表示されなくなります。デフォルトでは、すべてのリモート ロケーションにインストールされたブラウザーや接続されたデバイスのすべてが、**アプリケーションの選択** ダイアログに表示されます。
6. デフォルトでは、リモート ロケーションは %APPDATA%¥Silk¥SilkTest¥conf¥remoteLocations.xml に保存されます。他のチーム メンバーと設定を共有したい場合など、リモート ロケーションを保存するファイルを変更するには、次の操作を実行します。
   a) **共有オプション** をクリックします。**共有オプション** ダイアログ ボックスが表示されます。
   b) **共有ファイルに保存** をクリックします。
   c) **参照** をクリックして、ファイルを選択します。

   リポジトリにあるファイルを使用したい場合は、remoteLocationsFileLocation.properties という名前のファイルにリモート ロケーションへの完全パスを設定できます。たとえば、C:¥mySources に保存した remoteLocations.xml で指定したリモート ロケーションを使用する場合は、%APPDATA%¥Silk¥SilkTest¥conf¥ フォルダーに remoteLocationsFileLocation.properties という名前のファイルを作成し、*fileLocation=C¥:¥¥mySources¥¥remoteLocations.xml* を入力します。
7. 他のチーム メンバーによって指定されたリモート ロケーションを使用する場合など、共有ファイルから設定済みのリモート ロケーションのセットを読み込むには、次の操作を実行します。
   a) **共有オプション** をクリックします。**共有オプション** ダイアログ ボックスが表示されます。
   b) **共有ファイルから読み込む** をクリックします。
   c) **参照** をクリックして、ファイルを選択します。
8. **OK** をクリックします。

リモート ロケーションを追加すると、リモート ロケーション上にインストールされているブラウザ (Mac 上の Apple Safari など) が、**アプリケーションの選択** ダイアログ ボックスの **Web** タブで利用可能になり、リモート ロケーションに接続されているモバイル デバイスが、**アプリケーションの選択** ダイアログ ボックスの **モバイル** タブで利用可能になります。

アプリケーション構成エラー

アプリケーションにアタッチできない場合、以下のエラー メッセージが表示されます。
アプリケーション <アプリケーション名> にアタッチするのに失敗しました。詳細については、ヘルプを参照してください。

この場合、以下の表に示されている 1 つ以上の問題が原因である可能性があります。

問題	原因	解決策
タイムアウト	• システムが遅すぎます。 • システムのメモリ サイズが小さすぎます。	速いシステムを使用するか、現在使用しているシステムのメモリ使用量を減らします。
ユーザー アカウント制御 (UAC) の失敗	システムの管理者権限がありません。	管理者権限を持つユーザー アカウントでログインします。
コマンド ライン パターン	コマンド ライン パターンが固有すぎます。この問題は特に Java の場合に発生します。再生が意図したとおりに機能しないことがあります。	パターンから不明瞭なコマンドを削除します。
• **ブラウザーの選択** ダイアログ ボックスが、Web アプリケーションに対してテストを実行しているときに表示されない	基本状態と複数のブラウザー アプリケーション構成がテスト ケースに対して定義されています。	1 つを除くすべてのブラウザー アプリケーション構成をテスト ケースから削除します。

問題	原因	解決策
• Web アプリケーションに対してテストを実行しているときに、複数のブラウザー インスタンスが開始される • 開いているブラウザー インスタンスを使用して Web アプリケーションに対してテストを実行しているときに、Silk4J が動作を停止することがある		
テスト実行時の再生エラー	テストに対してアプリケーション構成が定義されていません。 次の例外が表示される可能性があります： アプリケーション構成が存在しません。	• キーワード駆動テストを実行するときに、基本状態を実行するキーワードがテストに含まれていることを確認します。 • アプリケーション構成が現在のプロジェクトに対して設定されていることを確認します。

アプリケーション構成のトラブルシューティング

アプリケーションの選択 ダイアログ ボックスにアプリケーションが表示されない理由

- **キャプションを持たないプロセスを表示しない** チェック ボックスをオフにします。このチェック ボックスは、デフォルトではオンになっており、キャプションを持たないアプリケーションはダイアログ ボックスに表示されません。
- Silk4J をシステム特権で実行します。

  1. Silk4J を閉じます。
  2. Open Agent を停止します。
  3. Silk4J を管理者として実行します。
- **タスク マネージャー** を使用して、アプリケーションが他のユーザー アカウントで実行されていないか確認します。
- アプリケーションが runas コマンド、または同等のコマンドで起動されていないか確認します。

Silk4J を構成して、Java Network Launching Protocol (JNLP) を使用するアプリケーションを起動する

Java Network Launching Protocol (JNLP) を使用して起動するアプリケーションでは、Silk4J に追加の構成が必要です。これらのアプリケーションは Web から起動されるため、実際のアプリケーションと「Web Start」を起動するように、アプリケーション構成を手動で構成する必要があります。このようにしない場合、アプリケーションがすでに実行されていないかぎり、テストを再生すると、失敗します。

1. Silk4J がアプリケーションを起動できずにテストが失敗する場合は、アプリケーション構成を編集します。

2. Silk Test ツールバー アイコン  の隣にあるドロップダウン矢印 をクリックして、**アプリケーション構成の編集** を選択します。 **アプリケーション構成の編集** ダイアログ ボックスが開き、既存のアプリケーション構成がリストされます。

3. 基本状態を編集して、再生中に Web Start が起動されるようにします。

a) **編集** をクリックします。
b) **実行可能ファイル パターン** テキスト ボックスに、javaws.exe への絶対パスを入力します。
たとえば、以下のように入力します。

```
%ProgramFiles%¥Java¥jre6¥bin¥javaws.exe
```

c) **コマンド ライン パターン** テキスト ボックスに、Web Start への URL を含むコマンド ライン パターンを入力します。

```
"<url-to-jnlp-file>"
```

たとえば、SwingSet3 アプリケーションの場合、以下のように入力します。

```
"http://download.java.net/javadesktop/swingset3/SwingSet3.jnlp"
```

d) **OK** をクリックします。

4. **OK** をクリックします。テストは、基本状態を使用し、Web 起動アプリケーションとアプリケーション構成の実行可能パターンを開始して javaw.exe に接続し、テストを実行します。

テストを実行すると、アプリケーション構成の EXE ファイルが基本状態の EXE ファイルと一致しないという警告が表示されます。テストは予想したとおりに実行されているため、このメッセージは無視できます。

複数のアプリケーションをテストするテストの作成

単一のテスト スクリプトで複数のアプリケーションをテストできます。このようなテスト スクリプトを作成するには、テストするアプリケーションそれぞれに対するアプリケーション構成を、スクリプトが存在するプロジェクトに追加する必要があります。

1. テストする主要なアプリケーション用に、テストを記録するか手動でスクリプトを作成します。
2. Silk Test ツールバー アイコン  の隣にあるドロップダウン矢印 をクリックして、**アプリケーション構成の編集** を選択します。 **アプリケーション構成の編集** ダイアログ ボックスが開き、既存のアプリケーション構成がリストされます。
3. アプリケーション構成をさらに追加するには、**アプリケーション構成の追加** をクリックします。

   注: 定義済みの基本状態を使って Web アプリケーションをテストするときに、ブラウザー アプリケーション構成を複数追加しないでください。

   アプリケーションの選択 ダイアログ ボックスが開きます。タブを選択してからテストするアプリケーションを選択して **OK** をクリックします。
4. **OK** をクリックします。
5. 新しいアプリケーション構成を使用して、スクリプトに追加の操作を記録するか手動でスクリプトを作成します。

注: 定義済みの基本状態を使って Web アプリケーションをテストするときに、ブラウザー アプリケーション構成を複数追加しないでください。

スクリプト オプションの設定

記録、ブラウザ、カスタム属性、無視するクラス、同期、および再生モードに関するスクリプト オプションを指定します。

TrueLog オプションの設定

ビットマップをキャプチャして Silk4J の情報を記録するように TrueLog を有効化します。

ビットマップとコントロールを TrueLog に記録すると、Silk4J のパフォーマンスに悪影響が出ることがあります。ビットマップをキャプチャして情報を記録すると TrueLog ファイルが大きくなることがあるので、エラーを含むテスト ケースのみを記録するように、詳細情報が必要なテスト ケース用に TrueLog オプションを調整できます。

テストの結果は、TrueLog Explorer で検証できます。詳細については、『*Silk TrueLog Explorer Silk Test のヘルプ*』を参照してください。

注: Silk Test 17.5 以降では、TrueLog ファイルのサイズを小さくするために、TrueLog ファイルのファイル形式が .xlg 形式から .tlz 形式に変更されました。.xlg 拡張子のファイルには、自動的に .tlz 拡張子が追加されます。.tlz 形式の結果データを解析するときには、.tlz ファイルを展開し、展開された .xlg ファイルを使ってデータを解析することもできます。

TrueLog を有効にして、TrueLog が Silk4J 用に収集する情報をカスタマイズするには、次の手順に従います。

1. **Silk4J** > **オプションの編集** をクリックします。 **スクリプト オプション** ダイアログ ボックスが表示されます。
2. **TrueLog** タブをクリックします。
3. **基本設定** 領域で **TrueLog の有効化** チェック ボックスをオンにします。
   - 正常なものとエラーになったものを両方とも含めて、すべてのテスト ケースのアクティビティを記録するには、**すべてのテストケース** をクリックします。これはデフォルトの設定です。
   - エラーが発生したテスト ケースのみのアクティビティを記録するには、**エラーのあるテストケース** をクリックします。
4. **TrueLog の場所** フィールドで TrueLog ファイルの名前とパス（省略可能）を入力するか、**参照** をクリックしてファイルを選択します。

   パスは、エージェントを実行しているマシンの相対パスです。デフォルト パスは Silk4J プロジェクト フォルダのパスであり、デフォルトの名前はスイート クラスの名前に拡張子 .tlz を付けたものになります。テストを並列実行する場合などに、TrueLog ファイルが上書きされないようにするため、TrueLog ファイル名にプレースホルダーを使用することができます。これらのプレースホルダーは、実行時に適切なデータに置換されます。

   注: パスは、実行時に検証されます。Silk Central によって実行されるテストは、スクリーンショットを結果ビューに表示できるようにするために、この値を Silk Central 結果ディレクトリに設定します。
5. **スクリーンショット モード** を選択します。

   デフォルトは、**エラー時** です。
6. 省略可能：**遅延** を設定します。

   この遅延により、ビットマップが撮影される前にオペレーティング システムがアプリケーション ウィンドウを描画する時間を確保できます。キャプチャされたビットマップでアプリケーションが適切に描画されない場合は、遅延時間を増やしてください。

7. **OK** をクリックします。

記録オプションの設定

記録を一時停止するためのショートカット キーの組み合わせを設定したり、絶対値による指定やマウスの移動操作が記録されるかどうかを指定したりします。

注: 以下の設定はすべて任意です。テスト メソッドの品質が向上する場合に、これらの設定を変更してください。

1. **Silk4J** > **オプションの編集** をクリックします。
2. **記録** タブ をクリックします。
3. 記録の一時停止に使用するショートカット キーの組み合わせとして Ctrl+Shift を設定するには、**OPT_ALTERNATE_RECORD_BREAK** チェック ボックスをオンにします。

   デフォルトのショートカット キーの組み合わせは、Ctrl+Alt です。

   注: SAP アプリケーションでは、ショートカット キーの組み合わせとして Ctrl+Shift を設定する必要があります。
4. スクロール イベントの絶対値を記録するには、**OPT_RECORD_SCROLLBAR_ABSOLUT** チェック ボックスをオンにします。
5. Web アプリケーション、Win32 アプリケーション、および Windows Forms アプリケーションのマウス移動操作を記録するには、**OPT_RECORD_MOUSEMOVES** チェック ボックスをオンにします。たとえば、Apache Flex や Swing など、xBrowser テクノロジー ドメインの子テクノロジ ドメインのマウス移動操作を記録することはできません。
6. マウスの移動操作を記録する場合は、**OPT_RECORD_MOUSEMOVE_DELAY** テキスト ボックスで、MouseMove 操作を記録する前に必要なマウスの静止時間をミリ秒で指定します。

   デフォルト値は、200 に設定されています。
7. 概して、TextClick 操作のほうが Click 操作よりも望ましいオブジェクトで、Click 操作ではなくテキストのクリック操作を記録するには、**OPT_RECORD_TEXT_CLICK** チェック ボックスをオンにします。
8. 概して、ImageClick 操作のほうが Click 操作よりも望ましいオブジェクトで、Click 操作ではなくイメージのクリック操作を記録するには、**OPT_RECORD_IMAGE_CLICK** チェック ボックスをオンにします。
9. オブジェクト マップ エントリを記録するか、XPath ロケーターを記録するかを定義するには、**OPT_RECORD_OBJECTMAPS_MODE** リストから適切な記録モードを選択します。
   - **オブジェクト マップ エントリ (新しいオブジェクトと既存のオブジェクト)**。これはデフォルトのモードです。
   - **XPath ロケーター (新しいオブジェクトと既存のオブジェクト)**。
   - **XPath ロケーター (新しいオブジェクトのみ)**。オブジェクト マップに既に存在するオブジェクトに対しては、オブジェクト マップ エントリが再利用されます。この設定を選択すると、AUT のメイン コントロールに対するオブジェクト マップを作成し、AUT に対して追加のテストを作成する間、これらのオブジェクト マップを保持することができます。
10. 記録セッションの開始時にテスト対象アプリケーション (AUT) のサイズを変更する場合は、**OPT_RESIZE_APPLICATION_BEFORE_RECORDING** チェック ボックスをオンにします。

    このチェック ボックスはデフォルトでオンになっており、AUT の隣に **Silk Recorder** が表示されます。このチェック ボックスをオフにすると、AUT と **Silk Recorder** が重なって表示される場合があります。
11. **OK** をクリックします。

ブラウザの記録オプションの設定

記録中に無視するブラウザの属性や DOM 関数の代わりに、ユーザーの入力そのものを記録するかどうかを指定します。

注: 以下の設定はすべて任意です。テスト メソッドの品質が向上する場合に、これらの設定を変更してください。

1. **Silk4J** > **オプションの編集** をクリックします。 **スクリプト オプション** ダイアログ ボックスが表示されます。
2. **ブラウザー** タブをクリックします。
3. **ロケーター属性名除外リスト** グリッドで、記録中に無視する属性名を入力します。
   たとえば、height という名前の属性を記録しない場合には、height 属性名をグリッドに追加します。
   複数の属性名を指定する場合にはカンマで区切ります。
4. **ロケーター属性値除外リスト** グリッドで、記録中に無視する属性値を入力します。
   たとえば、x-auto という値を持つ属性を記録しない場合には、x-auto をグリッドに追加します。
   複数の属性値を指定する場合にはカンマで区切ります。
5. DOM 関数の代わりにユーザーの入力そのものを記録するには、**OPT_XBROWSER_RECORD_LOWLEVEL** チェック ボックスをオンにします。
   たとえば、DomClick の代わりに Click、SetText の代わりに TypeKeys を記録するには、このチェック ボックスをオンにします。
   アプリケーションでプラグインまたは AJAX を使用している場合は、ユーザーの入力そのものを使用します。アプリケーションでプラグインまたは AJAX を使用していない場合は、再生中にブラウザにフォーカスを設定したりブラウザをアクティブにしたりする必要がない高レベル DOM 関数を使用することをお勧めします。テストで DOM 関数を使用すると、より高速になり、信頼性も高まります。
6. ロケーター属性値の最大長を設定するには、**属性値の最大の長さ** セクションのフィールドに長さを入力します。
   実際の長さがこの制限を超えると、値は切り捨てられ、ワイルド カード (*) が付加されます。デフォルト値は、20 文字に設定されています。
7. 指定したターゲット要素上の遮るもののないクリック スポットを自動的に検索するには、**OPT_XBROWSER_ENABLE_SMART_CLICK_POSITION** チェック ボックスをオンにします。
8. Mozilla Firefox で外部リンクを新しいウィンドウではなく新しいタブで開く場合は、**OPT_FIREFOX_SINGLE_WINDOW_MODE** をオンにします。

   注: このオプションは、Mozilla Firefox 52 以降を使用している場合にのみ動作します。
9. ブラウザーの iframe および frame サポートを無効にするには、**OPT_XBROWSER_ENABLE_IFRAME_SUPPORT** チェック ボックスをオフにします。
   Web アプリケーションの iframe の内容を無視して良い場合、iframe のサポートを無効にすると再生速度が速くなる可能性があります。たとえば、多くの広告が表示される Web ページやモバイル ブラウザーでテストする際に iframe サポートを無効にすると、再生パフォーマンスが大幅に向上する可能性があります。このオプションは、Internet Explorer では無視されます。このオプションは、デフォルトで有効になっています。
10. **iframe サポートのホワイトリスト** に、テストの対象となる iframe と frame の属性リストを指定します。
    リストの各項目で、属性名と対応する値を定義します。項目のどれにも一致しない iframe と frame は、テスト対象から除外されます。ワイルドカードを使用できます。たとえば、"name:*form" という項目を定義すると、<IFRAME name="user-form" src=...> がテスト対象に含まれます。このオプションは、Internet Explorer では無視されます。リストが空の場合は、すべての iframe と frame がテストの対象となります。複数のエントリをカンマで区切って指定します。

11 **iframe サポートのブラックリスト** に、テストから除外する iframe と frame の属性リストを指定します。

リストの各項目で、属性名と対応する値を定義します。項目のどれにも一致しない iframe と frame は、テスト対象に含まれます。ワイルドカードを使用できます。たとえば、"src:*advertising*" という項目を定義すると、<IFRAME src=http://my.domain/advertising-banner.html> がテスト対象から除外されます。このオプションは、Internet Explorer では無視されます。リストが空の場合は、すべての iframe と frame がテストの対象となります。複数のエントリをカンマで区切って指定します。

12 **OK** をクリックします。

カスタム属性の設定

Silk4J には、ロケーターが記録時に一意となり、メンテナンスが容易になるようにする、高度なロケーター生成メカニズムが備えられています。使用するアプリケーションやフレームワークに応じて、最適な結果を得るためにデフォルト設定を変更できます。それぞれのテクノロジで使用できる任意のプロパティ (整数や倍精度の数値、文字列、項目識別子、列挙値) を、カスタム属性として使用できます。

頻繁には変更されない属性を利用して、適切に定義されたロケーターでは、メンテナンス作業が少なく抑えられます。カスタム属性を使用すると、caption や index などの他の属性を使用するよりも高い信頼性を得ることができます。これは、caption はアプリケーションを他の言語に翻訳した場合に変更され、index は他のオブジェクトが追加されると変更される可能性があるためです。

カスタム属性 タブのリスト ボックスに一覧表示されているテクノロジ ドメインの場合、任意のプロパティー (*myCustomProperty* を定義する WPFButton など) を取得し、それらのプロパティーをカスタム属性として使用することもできます。最適な結果を得るために、テストで利用する要素にカスタム オートメーション ID を追加します。Web アプリケーションでは、操作する要素に などの属性を追加できます。また、Java SWT では、GUI を実装する開発者が属性 (testAutomationId など) をウィジェットに対して定義することによって、アプリケーション内でそのウィジェットを一意に識別できます。テスト担当者は、その属性をカスタム属性 (この場合は testAutomationId) のリストに追加し、その一意の ID によってコントロールを識別できます。この手法によって、ロケーターの変更に伴うメンテナンス作業を回避することができます。

caption のように、複数のオブジェクトで同じ属性値が共有されている場合、Silk4J は、複数の利用可能な属性を "and" 操作で結合してロケーターを一意にするよう試み、一致したオブジェクトのリストを単一のオブジェクトになるまで絞り込んでいきます。それができなくなった場合には、索引を付加します。つまり、ロケーターは caption が *xyz* である *n* 番め のオブジェクトを探すことを意味します。

複数のオブジェクトに同じカスタム属性の値が割り当てられた場合は、そのカスタム属性を呼び出したときにその値を持つすべてのオブジェクトが返されます。たとえば、一意の ID として loginName を 2 つの異なるテキスト フィールドに割り当てた場合は、loginName 属性を呼び出したときに、両方のフィールドが返されます。

1. **Silk4J** > **オプションの編集** をクリックします。 **スクリプト オプション** ダイアログ ボックスが表示されます。
2. **カスタム属性** タブを選択します。
3. **テクノロジ ドメインを選択します** リスト ボックスから、テストするアプリケーションのテクノロジ ドメインを選択します。

   注: Flex または Windows API ベースのクライアント／サーバー (Win32) アプリケーションには、カスタム属性を設定できません。
4. 使用する属性をリストに追加します。

   カスタム属性が利用可能な場合は、ロケーター生成プログラムは、他の属性の前にそれらの属性を使用します。リストの順番は、ロケーター生成プログラムが使用する属性の優先順位を表しています。指定した属性が選択したオブジェクトに対して利用可能ではなかった場合には、Silk4J はテストしているアプリケーションのデフォルトの属性を使用します。

   複数の属性名を指定する場合にはカンマで区切ります。

注: Web アプリケーションにカスタム属性を含めるためには、HTML タグとして追加します。たとえば、bcauid という属性を追加するには、<input type='button' bcauid='abc' value='click me' /> と入力します。

注: Java SWT コントロールにカスタム属性を含めるには、org.swt.widgets.Widget.setData(String *key*, Object *value*) メソッドを使用します。

注: Swing コントロールにカスタム属性を含めるには、putClientProperty("propertyName", "propertyValue") メソッドを使用します。

5. **OK** をクリックします。

無視するクラスの設定

オブジェクト階層を単純化し、テスト スクリプトや関数のコードの行の長さを短くするために、次のテクノロジの確実に不要なクラスに対するコントロールを抑制できます。

- Win32
- Java AWT/Swing
- Java SWT/Eclipse
- Windows Presentation Foundation (WPF)

1. **Silk4J** > **オプションの編集** をクリックします。 **スクリプト オプション** ダイアログ ボックスが表示されます。
2. **無視するクラス** タブをクリックします。
3. **無視するクラス** グリッドで、記録や再生中に無視するクラスの名前を入力します。
   複数のクラス名を指定する場合にはカンマで区切ります。
4. **OK** をクリックします。

記録/再生の対象とする WPF クラスの設定

記録や再生の対象にしたい WPF クラスの名前を指定します。たとえば、*MyGrid* というカスタム クラスが WPF Grid クラスから継承された場合、*MyGrid* カスタム クラスのオブジェクトは記録や再生に使用できません。Grid クラスはレイアウト目的のためにのみ存在し、機能テストとは無関係であるため、Grid オブジェクトは記録や再生に使用できません。この結果、Grid オブジェクトはデフォルトでは公開されません。機能テストに無関係なクラスに基づいたカスタム クラスを使用するには、カスタム クラス (この場合は *MyGrid*) を **OPT_WPF_CUSTOM_CLASSES** オプションに追加します。これによって、記録、再生、検索、プロパティの検証など、すべてのサポートされる操作を指定したクラスに対して実行できるようになります。

1. **Silk4J** > **オプションの編集** をクリックします。 **スクリプト オプション** ダイアログ ボックスが表示されます。
2. **WPF** タブをクリックします。
3. **カスタム WPF クラス名** グリッドで、記録や再生中に公開するクラスの名前を入力します。
   複数のクラス名を指定する場合にはカンマで区切ります。
4. **OK** をクリックします。

同期オプションの設定

Web アプリケーションの同期およびタイムアウトの値を指定します。

注: 以下の設定はすべて任意です。テスト メソッドの品質が向上する場合に、これらの設定を変更してください。

1. **Silk4J** > **オプションの編集** をクリックします。 **スクリプト オプション** ダイアログ ボックスが表示されます。
2. **同期** タブを選択します。
3. Web アプリケーションを準備完了状態にするための同期アルゴリズムを指定するには、**OPT_XBROWSER_SYNC_MODE** リスト ボックスからオプションを選択します。

   同期アルゴリズムは、呼び出しが可能になる状態までの待機時間を設定します。デフォルト値は、**AJAX** に設定されています。
4. **同期除外リスト** 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 **同期除外リスト** 設定で指定します。

   たとえば、クライアントからデータをポーリングすることによってサーバー時間を表示するウィジェットを Web アプリケーションで使用する場合は、このウィジェットのトラフィックが永続的にサーバーに送信されます。このサービスを同期から除外するには、サービス URL を判別し、除外リストに入力します。

   たとえば、以下のように入力します。

   - http://example.com/syncsample/timeService
   - timeService
   - UICallBackServiceHandler

   複数のエントリをカンマで区切って指定します。

   注: アプリケーションで 1 つのサービスのみが使用されている場合、そのサービスでテストを無効にするには、サービス URL を除外リストに追加するのではなく、HTML 同期モードを使用する必要があります。

   ヒント: 除外リストには、URL 全体ではなく URL の部分文字列を追加することを、Micro Focus ではお勧めしています。たとえば、http://example.com/syncsample/timeService ではなく、/syncsample を除外リストに追加します。ブラウザは相対 URL のみを Silk4J に返す場合があるため、URL 全体を指定すると機能しない場合があります。たとえば、ブラウザは /syncsample/timeService だけを返すのに対して、除外リストに http://example.com/syncsample/timeService を追加していると、Silk4J は返された URL を除外しません。
5. オブジェクトが準備完了状態になるまでの最大待機時間を指定するには、**OPT_SYNC_TIMEOUT** テキスト ボックスにミリ秒で値を指定します。

   デフォルト値は、**300000** に設定されています。
6. 再生中にオブジェクトが解決されるまでの最大待機時間を指定するには、**OPT_WAIT_RESOLVE_OBJDEF** テキスト ボックスにミリ秒で値を入力します。

   デフォルト値は、**5000** に設定されています。
7. エージェントがオブジェクトの解決を再試行するまでの最大待機時間を指定するには、**OPT_WAIT_RESOLVE_OBJDEF_RETRY** テキスト ボックスにミリ秒で値を入力します。

   デフォルト値は、**500** に設定されています。
8. **OK** をクリックします。

再生オプションの設定

テストするオブジェクトがアクティブであることを確実にしたいかどうかや、デフォルトの再生モードを上書きしたいかどうかを指定します。再生モードは、コントロールがマウスやキーボードによって再生さ

れるか、API で再生されるかを定義します。デフォルト モードを使用すると、最も信頼できる結果が得られます。他のモードを選択した場合は、すべてのコントロールが選択したモードを使用します。

1. **Silk4J** > **オプションの編集** をクリックします。 **スクリプト オプション** ダイアログ ボックスが表示されます。
2. **再生** タブを選択します。 **再生オプション** ページが表示されます。
3. テスト対象アプリケーションの起動に時間がかかる場合は、**OPT_APPREADY_TIMEOUT** テキスト ボックスの値を増やして、アプリケーションを待機する時間を増やしてください。
4. **OPT_REPLAY_MODE** リスト ボックスから、以下のいずれかのオプションを選択します。
   - **デフォルト**： このモードを使用すると、最も信頼できる結果が得られます。デフォルトでは、各コントロールそれぞれが、マウスやキーボード (低レベル)、あるいは API (高レベル) モードのどちらかを使用します。デフォルト モードを使用すると、各コントロールがコントロールの種類に応じて適切なモードが使用されます。
   - **高レベル**： このモードを使用すると、対象のテクノロジの API を使用して各コントロールが再生されます。たとえば、Rumba コントロールの場合、Rumba RDE API がコントロールの再生に使用されます。
   - **低レベル**： このモードを使用すると、マウスやキーボードを使用して各コントロールが再生されます。
5. テストするオブジェクトがアクティブであることを確実にするには、**OPT_ENSURE_ACTIVE_OBJDEF** チェック ボックスをオンにします。
6. 再生中にオブジェクトが有効になるまでの待機時間を変更するには、**オブジェクト有効化タイムアウト** セクションのフィールドに新しい時間を入力します。

   この時間は、ミリ秒単位で指定されます。デフォルト値は、1000 です。
7. **再生ステータス** ダイアログ ボックスを有効化するには、**OPT_SHOW_PLAYBACK_STATUS_DIALOG** チェック ボックスをオンにします。

   リモート　デバイス上のモバイル アプリケーションに対するテストを実行している場合など、リモート ロケーション上のテストの再生時に実行される操作を表示するために、**再生ステータス** ダイアログ ボックスを使用できます。
8. **再生ステータス** ダイアログ ボックスにテスト対象アプリケーションのビデオまたはスクリーンショットを表示するには、**OPT_PLAYBACK_STATUS_DIALOG_SCREENSHOTS** チェック ボックスをオンにします。
9. 資産が現在のプロジェクトに配置されることを指定するプレフィックスを編集するには、**OPT_ASSET_NAMESPACE** テキスト ボックスの **資産の名前空間** オプションのテキストを編集します。

10OK をクリックします。

詳細オプションの設定

詳細オプションでは、ハイブリッド アプリのフォールバック サポートを有効化したり、ロケーター属性名の大文字小文字を区別するかどうか、などを設定できます。

1. **Silk4J** > **オプションの編集** をクリックします。 **スクリプト オプション** ダイアログ ボックスが表示されます。
2. **詳細設定** タブをクリックします。 **詳細オプション** ページが表示されます。
3. 埋め込み Chrome アプリケーションをテストするには、**埋め込み Chrome サポートの有効化** フィールドに実行可能ファイルとポートを値ペア形式で指定します。
   たとえば、myApp.exe=9222 のように指定します。

   複数の埋め込み Chrome アプリケーションを指定する場合は、複数の値ペアをカンマで区切って指定します。

4. デフォルトのブラウザー サポートではテストできないハイブリッド モバイル アプリケーションに対するフォールバック サポートを使用するには、**OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT** を有効化します。
5. 標準の Win32 コントロールの解決に加えて、Microsoft ユーザー補助を有効にするには、**OPT_ENABLE_ACCESSIBILITY** を有効化にします。
6. テキストをキャプチャする前にウィンドウからフォーカスを外すには、**OPT_REMOVE_FOCUS_ON_CAPTURE_TEXT** を有効化します。
   テキストのキャプチャは、次のメソッドによる記録および再生中に実行されます。
   - TextClick
   - TextCapture
   - TextExists
   - TextRect
7. ロケーター属性名の大文字と小文字が区別されるように設定するには、**OPT_LOCATOR_ATTRIBUTES_CASE_SENSITIVE** を有効化します。モバイル Web アプリケーションのロケーター属性の名前は、常に大文字と小文字の区別はされません。つまり、モバイル Web アプリケーションの記録や再生時に、このオプションは無視されます。
8. **OPT_IMAGE_ASSET_DEFAULT_ACCURACY** リスト ボックスから、1 (低精度) から 10 (高精度) までの値を選択し、新しいイメージ資産のデフォルト精度レベルを設定します。
9. **OPT_IMAGE_VERIFICATION_DEFAULT_ACCURACY** リスト ボックスから、1 (低精度) から 10 (高精度) までの値を選択し、新しいイメージ検証資産のデフォルト精度レベルを設定します。
10. **OK** をクリックします。

Silk4J の設定を変更する

Silk4J は、Java Runtime Environment（JRE）のバージョン 1.6 以降を必要とします。

デフォルトでは、Silk4J は、Silk4J が起動するときに毎回 JRE のバージョンを確認し、JRE のバージョンが Silk4J と互換性のない場合にエラー メッセージを表示します。

1. エラー メッセージを表示されないようにするには、 **ウィンドウ** > **設定** > **Silk4J** を選択します。
2. **Silk4J** ノードを選択し、**JRE のバージョンに互換性がない場合にエラー メッセージを表示** チェック ボックスのチェックをオフにします。
3. **OK** をクリックします。

Silk4J プロジェクトを変換する

Silk4J プロジェクトには、標準 Java プロジェクトと比較して、以下のような特徴があります。

- Silk4J ライブラリおよび JUnit ライブラリへの依存。

Java プロジェクトを Silk4J プロジェクトに変換する

Silk4J で既存の Java プロジェクトを使用する場合は、以下の手順に従います。

1. **パッケージ エクスプローラー**で、Silk4J プロジェクトに変換する Java プロジェクトを右クリックします。 プロジェクトのコンテキスト メニューが表示されます。
2. Silk4J **ツール** > **プロジェクトの Silk4J 作成** を選択します。

   Silk4J のライブラリがプロジェクトに追加されます。 また、プロジェクトに JUnit ライブラリが含まれていない場合には、プロジェクトに追加されます。

Silk4J プロジェクトを Java プロジェクトに変換する

1. **パッケージ エクスプローラー** で、Java プロジェクトに変換する Silk4J プロジェクトを右クリックします。 プロジェクトのコンテキスト メニューが表示されます。
2. Silk4J **ツール** > **機能の Silk4J 除去** を選択します。

   Silk4J のライブラリがプロジェクトから除去されます。

   注: このプロジェクトは、JUnit を使用し続ける可能性が高いため、JUnit ライブラリへの依存はそのまま保持されます。

特定の環境のテスト

Silk4J では、複数の種類の環境でのテストがサポートされています。

ActiveX/Visual Basic アプリケーション

Silk4J は、ActiveX/Visual Basic アプリケーションのテストをサポートしています。

サポートするバージョン、既知の問題、および回避策についての最新の情報は、リリース ノートを確認してください。

ActiveX/Visual Basic メソッドの動的な呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティーの取得、またはプロパティーの設定を直接実行できます。また、このコントロールの Silk4J API で使用できないメソッドおよびプロパティーも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4J API を通して公開されていない場合に特に便利です。

オブジェクトの動的メソッドは invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

オブジェクトの複数の動的メソッドは invokeMethods メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

動的プロパティの取得には getProperty メソッドを、動的プロパティの設定には setProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、getPropertyList メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを String 型の入力パラメータとして設定する必要がある SetTitle というメソッドを呼び出すには、次のように入力します：

```
control.invoke("SetTitle","my  new  title");
```

注: 通常、ほとんどのプロパティは読み取り専用で、設定できません。

注: ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティーを取得する場合、Reflection を使用します。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4J がサポートするコントロールのメソッドとプロパティー。
- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティー。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- すべての組み込み Silk4J 型

Silk4J 型には、プリミティブ型（boolean、int、string など）、リスト、およびその他の型（Point など）が含まれます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4J 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、null が返されます。

Apache Flex のサポート

Silk4J は、Internet Explorer やスタンドアロンの Flash Player を使用した Apache Flex アプリケーション、および Apache Flex 4 以降でビルドした Adobe AIR アプリケーションのテストを組み込みでサポートしています。

Silk4J では、Apache Flex 3.x および 4.x アプリケーションにおいて複数のアプリケーション ドメインもサポートされているため、サブアプリケーションをテストできます。Silk4J では、ロケーター階層ツリーの各サブアプリケーションが、関連するアプリケーション ドメイン コンテキストを持つアプリケーション ツリーとして認識されます。Apache Flex 4.x サブアプリケーションでは、ロケーター属性テーブルのルート レベルで SparkApplication クラスが使用されます。Apache Flex 3.x サブアプリケーションでは、FlexApplication クラスが使用されます。

サポートするコントロール

Apache Flex のテストで記録および再生できるコントロールの完全なリストについては、「*API* リファレンス」の com.borland.silktest.jtf.flex パッケージ内でサポートされる Flex クラスの一覧を参照してください。

注: Silk Test Flex オートメーション SDK は、Apache Flex のオートメーション API に基づいています。Silk Test オートメーション SDK は、Apache Flex のオートメーション API でサポートされているものと同じコンポーネントが同様にサポートされます。たとえば、Flex オートメーション API の typekey ステートメントでは、すべてのキーはサポートされません。テキスト入力ステートメントを使用してこの問題を解決できます。Flex オートメーション API の詳細については、『*Apache Flex* リリース ノート』を参照してください。

Adobe Flash Player で実行するための Flex アプリケーションの構成

Apache Flex アプリケーションを Flash Player で実行するには、以下のいずれか、または両方の条件が満たされている必要があります。

- Flex アプリケーションを作成する開発者は、アプリケーションを EXE ファイルとしてコンパイルする必要があります。アプリケーションは、ユーザーが起動すると、Flash Player で開きます。Windows Flash Player は、*http://www.adobe.com/support/flashplayer/downloads.html* からインストールします。
- ユーザーが、Windows Flash Player Projector をインストールしている必要があります。ユーザーは、Flex の .SWF ファイルを開いた場合に Flash Player で開くように構成できます。Apache Flex 開発者スイートをインストールしないと、Flash Player をインストールしても Windows Flash Projector はインストールされません。Windows Flash Projector は、*http://www.adobe.com/support/flashplayer/downloads.html* からインストールします。

1. Microsoft Windows 7 および Microsoft Windows Server 2008 R2 では、管理者として実行されるように Flash Player を構成します。以下の手順を実行します。

a) Adobe Flash Player プログラム ショートカットまたは FlashPlayer.exe ファイルを右クリックして、**プロパティ** をクリックします。
b) **プロパティ** ダイアログ ボックスで、**互換性** タブをクリックします。
c) **管理者としてこのプログラムを実行する** チェック ボックスをオンにして、**OK** をクリックします。

2. コマンド プロンプト (cmd.exe) で以下のコマンドを入力して、Flash Player で .SWF ファイルを起動します。

"*<Application_Install_Directory>*¥*ApplicationName*.swf"

デフォルトで、*<SilkTest_Install_Directory>* は Program Files¥Silk¥Silk Test にあります。

Component Explorer の起動

Silk Test には、Component Explorer というサンプルの Apache Flex アプリケーションが含まれています。 Component Explorer は、Adobe オートメーション SDK および Silk Test 固有のオートメーション実装を使用してコンパイルされており、テスト用に事前に構成されています。

Internet Explorer で、*http://demo.borland.com/flex/SilkTest19.0/index.html* を開きます。 デフォルト ブラウザでアプリケーションが起動します。

Apache Flex アプリケーションのテスト

Silk Test は、Apache Flex アプリケーションのテストを組み込みでサポートしています。Silk Test では、いくつかのサンプル Apache Flex アプリケーションを提供しています。サンプル アプリケーションには、*http://demo.borland.com/flex/SilkTest19.0/index.html* からアクセスできます。

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

独自の Apache Flex アプリケーションをテストする前に、Apache Flex 開発者は以下のステップを実行する必要があります。

- Apache Flex アプリケーションのテストの有効化
- テスト可能な Apache Flex アプリケーションの作成
- Apache Flex コンテナのコーディング
- カスタム コントロールのオートメーション サポートの実装

独自の Apache Flex アプリケーションをテストするには、以下のステップを実行します。

- ローカルの Flash Player のセキュリティ設定の構成
- テストの記録
- テストの再生
- Apache Flex スクリプトのカスタマイズ
- カスタム Apache Flex コントロールのテスト

注: Apache Flex アプリケーションを読み込み、Flex オートメーション フレームワークを初期化するとき、テストを実行するマシンおよび Apache Flex アプリケーションの複雑度に応じて、多少の時間がかかる場合があります。アプリケーションが完全に読み込まれるように、ウィンドウのタイムアウト値を高い値に設定します。

Apache Flex カスタム コントロールのテスト

Silk4J では、Apache Flex カスタム コントロールのテストがサポートされています。ただし、デフォルトでは、Silk4J は、カスタム コントロールの個別のサブコントロールを記録および再生することはできません。

カスタム コントロールをテストする場合、以下のオプションが存在します。

- 基本サポート

  基本サポートでは、動的呼び出しを使用して、再生中にカスタム コントロールと対話します。作業量が少なく済むこのアプローチは、テスト アプリケーションにおいて、Silk4J が公開しないカスタム コントロールのプロパティおよびメソッドにアクセスする場合に使用します。カスタム コントロールの開発者は、コントロールのテストを容易にすることのみを目的としたメソッドおよびプロパティをカスタム コントロールに追加することもできます。ユーザーは、動的呼び出し機能を使用してこれらのメソッドやプロパティを呼び出すことができます。

  基本サポートには以下のような利点があります。

  - 動的呼び出しでは、テスト アプリケーションのコードを変更する必要がありません。
  - 動的呼び出しを使用することによって、ほとんどのテストのニーズを満たすことができます。

  基本サポートには以下のような短所があります。

  - ロケーターには、具体的なクラス名が組み込まれません (たとえば、Silk4J では「//FlexSpinner」ではなく「//FlexBox」と記録されます)。
  - 記録のサポートが限定されます。
  - Silk4J では、イベントを再生できません。

  例を含む動的呼び出しの詳細については、「*Apache Flex* メソッドの動的呼び出し」を参照してください。
- 高度なサポート

  高度なサポートでは、カスタム コントロールに対して、特定のオートメーション サポートを作成できます。この追加のオートメーション サポートによって、記録のサポートおよびより強力な再生のサポートが提供されます。高度なサポートには以下のような利点があります。

  - イベントの記録と再生を含む、高レベルの記録および再生のサポートが提供されます。
  - Silk4J では、カスタム コントロールが他のすべての組み込み Apache Flex コントロールと同様に処理されます。
  - Silk4J API とシームレスに統合できます。
  - Silk4J では、ロケーターで具体的なクラス名が使用されます (たとえば、Silk4J では「//FlexSpinner」と記録されます)。

  高度なサポートには以下のような短所があります。

  - 実装作業が必要です。テスト アプリケーションを変更し、Open Agent を拡張する必要があります。

Flex メソッドの動的呼び出し

動的呼び出し機能を使用して Silk4J が対象としないコントロールのメソッドを呼び出したり、プロパティを取得/設定することができます。この機能は、カスタム コントロールを使用したり、カスタマイズせずに Silk4J がサポートするコントロールを使用する場合に有効です。

オブジェクトの動的メソッドは invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

動的プロパティの取得には getProperty メソッドを、動的プロパティの設定には setProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、getPropertyList メソッドを使用します。

注: 通常、ほとんどのプロパティは読み取り専用で、設定できません。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4J がサポートするコントロールのメソッドとプロパティー。
- Flex API で定義されているすべてのパブリック メソッド

- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティー。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- すべての組み込み Silk4J 型

  Silk4J 型には、プリミティブ型 (boolean、int、string など)、リスト、およびその他の型 (Point など) が含まれます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4J 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、null が返されます。

テスト アプリケーションでのカスタム コントロールの定義

通常、テスト アプリケーションには、アプリケーションの開発中に追加されたカスタム コントロールがすでに含まれています。テスト アプリケーションにすでにカスタム コントロールが含まれている場合は、「動的呼び出しを使用して *Flex* カスタム コントロールをテストする」または「オートメーション サポートを使用してカスタム コントロールをテストする」に進んでください。

この手順では、Flex アプリケーション開発者が Flex で Spinner カスタム コントロールを作成する方法を示します。このトピックで作成する Spinner カスタム コントロールは、カスタム コントロールの実装およびテストのプロセスを説明するために、いくつかのトピックで使用されています。

Spinner カスタム コントロールは、以下のグラフィックに示すように、2 つのボタンと 1 つのテキスト フィールドを含んでいます。

ユーザーは、**Down** をクリックしてテキスト フィールドに表示されている値を 1 減分させ、**Up** をクリックしてテキスト フィールドの値を 1 増分させることができます。

カスタム コントロールには、設定および取得が可能なパブリックの Value プロパティが用意されています。

1. テスト アプリケーションで、コントロールのレイアウトを定義します。
   たとえば、Spinner コントロール タイプでは、以下のように記述します。

```
<?xml version="1.0" encoding="utf-8"?>
<customcontrols:SpinnerClass xmlns:mx="http://www.adobe.com/2006/mxml"
xmlns:controls="mx.controls.*" xmlns:customcontrols="customcontrols.*">
      <controls:Button id="downButton" label="Down" />
      <controls:TextInput id="text" enabled="false" />
      <controls:Button id="upButton" label="Up"/>
</customcontrols:SpinnerClass>
```

2. カスタム コントロールの実装を定義します。
   たとえば、Spinner コントロール タイプでは、以下のように記述します。

```
package customcontrols
{
      import flash.events.MouseEvent;

      import mx.containers.HBox;
```

```
    import mx.controls.Button;
    import mx.controls.TextInput;
    import mx.core.UIComponent;
    import mx.events.FlexEvent;

    [Event(name="increment",        type="customcontrols.SpinnerEvent")]
    [Event(name="decrement",        type="customcontrols.SpinnerEvent")]

    public class SpinnerClass extends HBox
    {
        public var downButton : Button;
        public var upButton : Button;
        public var text : TextInput;
        public var ssss: SpinnerAutomationDelegate;
        private var _lowerBound : int = 0;
        private var _upperBound : int = 5;

        private var _value : int = 0;
        private var _stepSize : int = 1;


        public function SpinnerClass() {
            addEventListener(FlexEvent.CREATION_COMPLETE,
creationCompleteHandler);
        }

        private function creationCompleteHandler(event:FlexEvent) : void {
            downButton.addEventListener(MouseEvent.CLICK, downButtonClickHandler);
            upButton.addEventListener(MouseEvent.CLICK, upButtonClickHandler);
            updateText();
        }

        private function downButtonClickHandler(event : MouseEvent) : void {
            if(Value - stepSize >= lowerBound) {
                Value = Value - stepSize;
            }
            else {
                Value = upperBound - stepSize + Value - lowerBound + 1;
            }

            var spinnerEvent : SpinnerEvent = new
SpinnerEvent(SpinnerEvent.DECREMENT);
            spinnerEvent.steps = _stepSize;
            dispatchEvent(spinnerEvent);
        }

        private function upButtonClickHandler(event : MouseEvent) : void {
            if(cValue <= upperBound - stepSize) {
                Value = Value + stepSize;
            }
            else {
                Value = lowerBound + Value + stepSize - upperBound - 1;
            }

            var spinnerEvent : SpinnerEvent = new
SpinnerEvent(SpinnerEvent.INCREMENT);
            spinnerEvent.steps = _stepSize;
            dispatchEvent(spinnerEvent);
        }
```

```
        private function updateText() : void {
            if(text != null) {
                text.text = _value.toString();
            }
        }

        public function get Value() : int {
            return _value;
        }

        public function set Value(v : int) : void {
            _value = v;
            if(v < lowerBound) {
                _value = lowerBound;
            }
            else if(v > upperBound) {
                _value = upperBound;
            }
            updateText();

        }

        public function get stepSize() : int {
            return _stepSize;
        }

        public function set stepSize(v : int) : void {
            _stepSize = v;
        }

        public function get lowerBound() : int {
            return _lowerBound;
        }

        public function set lowerBound(v : int) : void {
            _lowerBound = v;
            if(Value < lowerBound) {
                Value = lowerBound;
            }
        }

        public function get upperBound() : int {
            return _upperBound;
        }

        public function set upperBound(v : int) : void {
            _upperBound = v;
            if(Value > upperBound) {
                Value = upperBound;
            }
        }
    }
}
```

3. コントロールが使用するイベントを定義します。
たとえば、Spinner コントロール タイプでは、以下のように記述します。

```
package customcontrols
{
```

```
    import flash.events.Event;

    public class SpinnerEvent extends Event
    {
        public static const INCREMENT : String = "increment";
        public static const DECREMENT : String = "decrement";

        private var _steps : int;

        public function SpinnerEvent(eventName : String) {
            super(eventName);
        }

        public function set steps(value:int) : void {
            _steps = value;
        }

        public function get steps() : int {
            return _steps;
        }

    }
}
```

次のステップでは、テスト アプリケーションのオートメーション サポートを実装します。

動的呼び出しを使用した Flex カスタム コントロールのテスト

Silk4J では、動的呼び出しを使用したカスタム コントロールの記録と再生のサポートが提供されており、これにより再生中にカスタム コントロールを操作できます。作業量が少なく済むこのアプローチは、テスト アプリケーションにおいて、Silk4J が公開しないカスタム コントロールのプロパティおよびメソッドにアクセスする場合に使用します。カスタム コントロールの開発者は、コントロールのテストを容易にすることのみを目的としたメソッドおよびプロパティをカスタム コントロールに追加することもできます。

1. コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。
2. オブジェクトの動的メソッドは invoke メソッドを使用して呼び出します。
3. コントロールでサポートされている動的プロパティのリストを取得するには、getPropertyList メソッドを使用します。
4. 動的プロパティの取得には getProperty メソッドを、動的プロパティの設定には setProperty メソッドを使用します。

例

この例では、以下の図に示すように、2 つのボタンと 1 つのテキスト フィールドを含む Spinner カスタム コントロールをテストします。

ユーザーは、**Down** をクリックしてテキスト フィールドに表示されている値を 1 減分させ、**Up** をクリックしてテキスト フィールドの値を 1 増分させることができます。

カスタム コントロールには、設定および取得が可能なパブリックの Value プロパティが用意されています。

Spinner の値を 4 に設定するには、以下のように入力します。

```
FlexBox spinner = _desktop.<FlexBox>find("//
FlexBox[@className=customcontrols.Spinner]");
spinner.setProperty("Value", 4);
```

オートメーション サポートを使用したカスタム コントロールのテスト

カスタム コントロールに対して、特定のオートメーション サポートを作成できます。この追加のオートメーション サポートによって、記録のサポートおよびより強力な再生のサポートが提供されます。オートメーション サポートを作成するには、テスト アプリケーションを変更し、Open Agent を拡張する必要があります。

Silk4J でカスタム コントロールをテストする前に、以下のステップを実行します。

- テスト アプリケーションでのカスタム コントロールの定義
- オートメーション サポートの実装

テスト アプリケーションを変更してオートメーション サポートを組み込んだあと、以下のステップを実行します。

1. テスト内のカスタム コントロールをテストするために、カスタム コントロールの Java クラスを作成します。

   たとえば、Spinner コントロール クラスは、以下の内容を含んでいる必要があります。

```
package customcontrols;

import com.borland.silktest.jtf.Desktop;
import com.borland.silktest.jtf.common.JtfObjectHandle;
import com.borland.silktest.jtf.flex.FlexBox;

/**
 * Implementation of the FlexSpinner Custom Control.
 */
public class FlexSpinner extends FlexBox {

  protected FlexSpinner(JtfObjectHandle handle, Desktop desktop) {
    super(handle, desktop);
  }

  @Override
  protected String getCustomTypeName() {
    return "FlexSpinner";
  }

  public Integer getLowerBound() {
    return (Integer) getProperty("lowerBound");
  }

  public Integer getUpperBound() {
    return (Integer) getProperty("upperBound");
  }

  public Integer getValue() {
    return (Integer) getProperty("Value");
  }

  public void setValue(Integer Value) {
    setProperty("Value", Value);
  }
```

```
  public Integer getStepSize() {
    return (Integer) getProperty("stepSize");
  }

  public void increment(Integer steps) {
    invoke("Increment", steps);
  }

  public void decrement(Integer steps) {
    invoke("Decrement", steps);
  }

}
```

2. この Java クラスをテストが含まれている Silk4J テスト プロジェクトに追加します。

   ヒント: 同じカスタム コントロールを複数の Silk4J プロジェクトで使用する場合は、カスタム コントロールを含む別個のプロジェクトを作成し、Silk4J テスト プロジェクトからそれを参照することをお勧めします。

3. <Silk Test のインストール ディレクトリ>¥ng¥agent¥plugins ¥com.borland.silktest.jtf.agent.customcontrols_<バージョン>¥config¥classMapping.properties ファイルに以下の行を追加します。

   ```
   FlexSpinner=customcontrols.FlexSpinner
   ```

   等号記号の左側のコードは、XML ファイルに定義されているカスタム コントロール名である必要があります。等号記号の右側のコードは、カスタム コントロールの Java クラスの完全修飾名である必要があります。

   これで、Silk4J でカスタム コントロールを使用する場合に、記録および再生が完全にサポートされるようになりました。

   例

   以下の例は、オートメーションの委譲に実装されている「Increment」メソッドを使用して、Spinner の値を 3 つ増分する方法を示しています。

   ```
   desktop.<FlexSpinner>find("//FlexSpinner[@caption='index:
   1']").increment(3);
   ```

   以下の例は、Spinner の値を 3 に設定する方法を示しています。

   ```
   desktop.<FlexSpinner>find("//FlexSpinner[@caption='index:1']").setValue(3);
   ```

カスタム コントロールのオートメーション サポートの実装

カスタム コントロールをテストする前に、カスタム コントロールの ActionScript でオートメーション サポート（オートメーションの委譲）を実装し、テスト アプリケーションにコンパイルします。

以下の手順では、Flex のカスタム Spinner コントロールを使用して、カスタム コントロールのオートメーション サポートの実装方法を示します。Spinner カスタム コントロールは、以下のグラフィックに示すように、2 つのボタンと 1 つのテキスト フィールドを含んでいます。

ユーザーは、**Down** をクリックしてテキスト フィールドに表示されている値を 1 減分させ、**Up** をクリックしてテキスト フィールドの値を 1 増分させることができます。

カスタム コントロールには、設定および取得が可能なパブリックの Value プロパティが用意されています。

1. カスタム コントロールの ActionScript でオートメーション サポート（オートメーションの委譲）を実装します。

   オートメーションの委譲の実装の詳細については、Adobe Live ドキュメント（*http://livedocs.adobe.com/flex/3/html/help.html?content=functest_components2_14.html*）を参照してください。

   この例では、オートメーションの委譲によって、「increment」および「decrement」メソッドに対してサポートが追加されます。オートメーションの委譲のコード例は以下のとおりです。

```
package customcontrols
{
    import flash.display.DisplayObject;
    import mx.automation.Automation;
    import customcontrols.SpinnerEvent;
    import mx.automation.delegates.containers.BoxAutomationImpl;
    import flash.events.Event;
    import mx.automation.IAutomationObjectHelper;
    import mx.events.FlexEvent;
    import flash.events.IEventDispatcher;
    import mx.preloaders.DownloadProgressBar;
    import flash.events.MouseEvent;
    import mx.core.EventPriority;

    [Mixin]
    public class SpinnerAutomationDelegate extends BoxAutomationImpl
    {

        public static function init(root:DisplayObject) : void {
            // register delegate for the automation
            Automation.registerDelegateClass(Spinner, SpinnerAutomationDelegate);
        }

        public function SpinnerAutomationDelegate(obj:Spinner) {
            super(obj);
            // listen to the events of interest (for recording)
            obj.addEventListener(SpinnerEvent.DECREMENT, decrementHandler);
            obj.addEventListener(SpinnerEvent.INCREMENT, incrementHandler);
        }

        protected function decrementHandler(event : SpinnerEvent) : void {
            recordAutomatableEvent(event);
        }

        protected function incrementHandler(event : SpinnerEvent) : void {
            recordAutomatableEvent(event);
        }

        protected function get spinner() : Spinner {
            return uiComponent as Spinner;
        }

        //--------------------------------
        //   override functions
        //--------------------------------

        override public function get automationValue():Array {
            return [ spinner.Value.toString() ];
        }

        private function replayClicks(button : IEventDispatcher, steps : int) : Boolean
```

```
{
            var helper : IAutomationObjectHelper =
Automation.automationObjectHelper;
            var result : Boolean;
            for(var i:int; i < steps; i++) {
                helper.replayClick(button);
            }
            return result;
        }

        override public function replayAutomatableEvent(event:Event):Boolean {

            if(event is SpinnerEvent) {
                var spinnerEvent : SpinnerEvent = event as SpinnerEvent;
                if(event.type == SpinnerEvent.INCREMENT) {
                    return replayClicks(spinner.upButton, spinnerEvent.steps);
                }
                else if(event.type == SpinnerEvent.DECREMENT) {
                    return replayClicks(spinner.downButton, spinnerEvent.steps);
                }
                else {
                    return false;
                }

            }
            else {
                return super.replayAutomatableEvent(event);
            }
        }

        // do not expose the child controls (i.e the buttons and the textfield) as
individual controls
        override public function get numAutomationChildren():int {
            return 0;
        }

    }
}
```

2. Open Agent にオートメーションの委譲を導入するために、カスタム コントロールを記述する XML ファイルを作成します。

クラス定義ファイルには、インストルメント化されたすべての Flex コンポーネントについての情報が含まれています。このファイルでは、記録中にイベントを送信でき、再生中にイベントを受け取ることができるコンポーネントについての情報が提供されます。クラス定義ファイルには、サポートされているプロパティの定義も含まれています。

Spinner カスタム コントロールの XML ファイルは以下のようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<TypeInformation>
    <ClassInfo Name="FlexSpinner" Extends="FlexBox">
        <Implementation
            Class="customcontrols.Spinner" />
        <Events>
            <Event Name="Decrement">
                <Implementation
                    Class="customcontrols.SpinnerEvent"
                    Type="decrement" />
                <Property Name="steps">
                    <PropertyType Type="integer" />
                </Property>
```

```
            </Event>
            <Event Name="Increment">
                <Implementation
                    Class="customcontrols.SpinnerEvent"
                    Type="increment" />
                <Property Name="steps">
                    <PropertyType Type="integer" />
                </Property>
            </Event>
        </Events>
        <Properties>
            <Property Name="lowerBound" accessType="read">
                <PropertyType Type="integer" />
            </Property>
            <Property Name="upperBound" accessType="read">
                <PropertyType Type="integer" />
            </Property>
            <!-- expose read and write access for the Value property -->
            <Property Name="Value" accessType="both">
                <PropertyType Type="integer" />
            </Property>
            <Property Name="stepSize" accessType="read">
                <PropertyType Type="integer" />
            </Property>
        </Properties>
    </ClassInfo>
</TypeInformation>
```

3. サポートされている Flex コントロールのすべてのクラス、およびそのメソッドとプロパティを記述するすべての XML ファイルが格納されるフォルダに、カスタム コントロールの XML ファイルを配置します。

   Silk Test には、サポートされている Flex コントロールのすべてのクラス、およびそのメソッドとプロパティを記述するいくつかの XML ファイルが含まれています。これらの XML ファイルは、<Silk Test_install_directory>¥ng¥agent¥plugins¥com.borland.fastxd.techdomain.flex.agent_<バージョン>¥config¥automationEnvironment フォルダにあります。

   独自の XML ファイルを提供する場合は、XML ファイルをこのフォルダにコピーする必要があります。Open Agent が起動して、Apache Flex のサポートを初期化する場合、このディレクトリの内容が読み込まれます。

   Flex の Spinner サンプル コントロールをテストするには、CustomControls.xml ファイルをこのフォルダにコピーする必要があります。Open Agent が現在実行されている場合は、ファイルをフォルダにコピーしたあと、Open Agent を再起動します。

Flex クラス定義ファイル

クラス定義ファイルには、インストルメント化されたすべての Flex コンポーネントについての情報が含まれています。このファイルでは、記録中にイベントを送信でき、再生中にイベントを受け取ることができるコンポーネントについての情報が提供されます。クラス定義ファイルには、サポートされているプロパティの定義も含まれています。

Silk Test には、Flex の共通コントロールおよび特殊化されたコントロールのすべてのクラス、イベント、およびプロパティを記述するいくつかの XML ファイルが含まれています。これらの XML ファイルは、<Silk Test_install_directory>¥ng¥agent¥plugins¥com.borland.fastxd.techdomain.flex.agent_<バージョン>¥config¥automationEnvironment フォルダにあります。

独自の XML ファイルを提供する場合は、XML ファイルをこのフォルダにコピーする必要があります。Silk Test のエージェントが起動して Apache Flex のサポートを初期化するとき、このディレクトリの内容が読み込まれます。

XML ファイルの基本的な構造は以下のとおりです。

```
<TypeInformation>
<ClassInfo>
<Implementation />
<Events>
<Event />
...
</Events>
<Properties>
<Property />
...
</Properties>
</ClassInfo>
</TypeInformation>
```

Apache Flex スクリプトのカスタマイズ

手動で Flex スクリプトをカスタマイズできます。Flex オブジェクトのプロパティに対して Verify 関数を使用して、手動で検証を挿入できます。各 Flex オブジェクトには、検証可能な一連のプロパティがあります。検証に使用できるプロパティのリストについては、「*API* リファレンス」の com.borland.silktest.jtf.flex パッケージ内でサポートされる Flex クラスのリストを参照してください。

1. Flex アプリケーションのテストを記録します。
2. カスタマイズするスクリプト ファイルを開きます。
3. 追加するコードを手動で入力します。

同一 Web ページ上の複数の Flex アプリケーションのテスト

同じ Web ページに複数の Flex アプリケーションが存在する場合、Silk4J は、Flex アプリケーションの ID またはアプリケーションの size プロパティを使用して、テスト対象アプリケーションを特定します。同じページに複数のアプリケーションが存在し、それらのサイズが異なる場合、Silk4J は、size プロパティを使用して操作実行対象のアプリケーションを特定します。追加の操作は必要ありません。

以下の場合、Silk4J は、JavaScript を使用して Flex アプリケーションの ID を検索し、操作実行対象のアプリケーションを特定します。

- 単一の Web ページ上に複数の Flex アプリケーションが存在する場合。
- これらのアプリケーションのサイズが同じである場合。

注: この場合、ブラウザ マシンで JavaScript が有効になっていないと、スクリプト実行時にエラーが発生します。

1. JavaScript を有効にします。

2. Internet Explorer で、以下の手順を実行します。
   a) **ツール** > **インターネット オプション** を選択します。
   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。
   d) **スクリプト作成** セクションの **アクティブ スクリプト** で、**有効にする** をクリックして **OK** をクリックします。
3. 「*Apache Flex* アプリケーションのテスト」の手順に従います。

注: Web ページにフレームが存在し、アプリケーションが同じサイズである場合、この方法は動作しません。

Adobe AIR のサポート

Silk4J がサポートする Adobe AIR でのテストは、Flex 4 コンパイラを使用してコンパイルされたアプリケーションのみです。 サポートされているバージョンの詳細については、リリース ノートで最新の情報を確認してください。

Silk Test には、サンプルの Adobe AIR アプリケーションが含まれています。 *http://demo.borland.com/flex/SilkTest19.0/index.html* にあるサンプル アプリケーションにアクセスして、使用する Adobe AIR アプリケーションをクリックしてください。 オートメーションあり、またはオートメーションなしのアプリケーションを選択できます。 AIR アプリケーションを実行するには、Adobe AIR ランタイムをインストールする必要があります。

名前またはインデックスを使用する Flex の Select メソッドの概要

Flex の Select メソッドは、選択するコントロールの Name または Index を使用して記録できます。デフォルトで、Silk4J では、コントロールの名前を使用して Select メソッドが記録されます。ただし、コントロールのインデックスを使用して Select イベントを記録するように環境を変更したり、名前を使用した記録とインデックスを使用した記録を切り替えたりすることができます。

以下のコントロールでは、インデックスを使用して Select イベントを記録できます。

- FlexList
- FlexTree
- FlexDataGrid
- FlexAdvancedDataGrid
- FlexOLAPDataGrid
- FlexComboBox

デフォルト設定は、コントロールの名前を使用する ItemBasedSelection（Select イベント）です。インデックスを使用するには、IndexBasedSelection（SelectIndex イベント）を使用するように AutomationEnvironment を変更する必要があります。これらのクラスのいずれかの動作を変更するには、以下のコードを使用して FlexCommonControls.xml、AdvancedDataGrid.xml、または OLAPDataGrid.xml ファイルを変更する必要があります。これらの XML ファイルは、<Silk Test_install_directory>¥ng¥agent¥plugins¥com.borland.fastxd.techdomain.flex.agent_<バージョン>¥config¥automationEnvironment フォルダにあります。対応する xml ファイルで、以下の変更を行います。

```
<ClassInfo  Extends="FlexList"  Name="FlexControlName"
EnableIndexBasedSelection=”true”  >

…

</ClassInfo>
```

この変更では、FlexList::SelectIndex イベントの記録に IndexBasedSelection が使用されています。コードの EnableIndexBasedSelection= を false に設定するか、またはこのブール値を削除すると、記録で名前が使用される設定に戻ります（FlexList::Select イベント）。

注: これらの変更内容を有効にするには、アプリケーションを再起動する必要があります。アプリケーションを再起動すると、Silk Test Agent も自動的に再起動されます。

FlexDataGrid コントロールでの項目の選択

FlexDataGrid コントロールの項目は、インデックス値または内容値を使用して選択します。

1. インデックス値を使用して FlexDataGrid コントロールの項目を選択するには、SelectIndex メソッドを使用します。
   たとえば、FlexDataGrid.SelectIndex(1) のように入力します。
2. 内容値を使用して FlexDataGrid コントロールの項目を選択するには、Select メソッドを使用します。
   必要な形式の文字列を使用して、選択する行を識別します。 項目と項目の間は、縦線文字（|）で区切る必要があります。 少なくとも 1 つの項目を 2 つのアスタリスク（*）で囲む必要があります。 これにより、クリックが実行される項目が識別されます。
   構文は FlexDataGrid.Select("*Item1* | Item2 | Item3") です。

Flex アプリケーションのテストの有効化

Flex アプリケーションをテストに対して有効化するには、Apache Flex 開発者は Flex アプリケーションに以下のコンポーネントを組み込む必要があります。

- Apache Flex オートメーション パッケージ
- Silk Test オートメーション パッケージ

Apache Flex オートメーション パッケージ

開発者は、Flex オートメーション パッケージを使用して、オートメーション API を使用する Flex アプリケーションを作成できます。Flex オートメーション パッケージは、Adobe の Web サイト（*http://www.adobe.com*）からダウンロードできます。パッケージには、以下の内容が含まれています。

- オートメーション ライブラリ：automation.swc ライブラリおよび automation_agent.swc ライブラリは、Flex フレームワーク コンポーネントの委譲の実装です。automation_agent.swc ファイルおよび関連するリソース バンドルは、汎用的なエージェント メカニズムです。Silk Test Agent などのエージェントは、これらのライブラリの上に構築されます。
- サンプル

注: Silk Test Flex オートメーション SDK は、Flex のオートメーション API に基づいています。Silk Test オートメーション SDK は、Flex のオートメーション API でサポートされているものと同じコンポーネントが同様にサポートされます。たとえば、Flex オートメーション API の typekey ステートメントでは、すべてのキーはサポートされません。テキスト入力ステートメントを使用してこの問題を解決できます。Flex オートメーション API の詳細については、『*Apache Flex* リリース ノート』を参照してください。

Silk Test オートメーション パッケージ

Silk Test の Open Agent は、Apache Flex オートメーション エージェント ライブラリを使用しています。FlexTechDomain.swc ファイルに、Silk Test 固有の実装が含まれています。

以下のいずれかの方法を使用して、アプリケーションをテストに対して有効化できます。

- Flex アプリケーションへのオートメーション パッケージのリンク
- 実行時の読み込み

Flex アプリケーションへのオートメーション パッケージのリンク

テストする予定の Flex アプリケーションを事前にコンパイルする必要があります。機能テスト クラスは、コンパイル時にアプリケーションに埋め込まれ、アプリケーションは実行時に自動テストに関する外部依存関係を持ちません。

コンパイル時にアプリケーションの SWF ファイルに機能テスト クラスを埋め込むと、SWF ファイルのサイズが大きくなります。SWF ファイルのサイズが重要でない場合は、機能テストと展開に同じ SWF ファイルを使用します。SWF ファイルのサイズが重要である場合は、2 つの SWF ファイルを生成します。1 つは機能テスト クラスが埋め込まれたファイル、もう 1 つは機能テスト クラスが埋め込まれていないファイルです。展開には、テスト クラスが埋め込まれていない SWF ファイルを使用します。

include-libraries コンパイラ オプションを指定してテストのために Flex アプリケーションを事前にコンパイルする場合は、以下のファイルを参照します。

- automation.swc
- automation_agent.swc
- FlexTechDomain.swc
- automation_charts.swc（アプリケーションでグラフおよび Flex 2.0 を使用する場合のみインクルード）
- automation_dmv.swc（アプリケーションでグラフおよび Flex 3.x 以降を使用する場合にインクルード）
- automation_flasflexkit.swc（アプリケーションで埋め込みの Flash コンテンツを使用する場合にインクルード）
- automation_spark.swc（アプリケーションで新しい Flex 4.x コントロールを使用する場合にインクルード）
- automation_air.swc（アプリケーションが AIR アプリケーションである場合にインクルード）
- automation_airspark.swc（アプリケーションが AIR アプリケーションであり、新しい Flex 4.x コントロールを使用する場合にインクルード）

Flex アプリケーションの最終リリース バージョンを作成する場合は、これらの SWC ファイルへの参照なしでアプリケーションを再コンパイルします。オートメーション SWC ファイルの使用の詳細については、『*Apache Flex* リリース ノート』を参照してください。

アプリケーションをサーバーに展開しないで、ファイル プロトコルを使用して要求したり、Apache Flex Builder 内で実行したりする場合は、各 SWF ファイルをローカルの信頼済みサンドボックスに組み込む必要があります。このためには、追加の構成情報が必要です。コンパイラの構成ファイルを変更するか、またはコマンド ライン オプションを使用して、構成情報を追加します。

注: Silk Test Flex オートメーション SDK は、Flex のオートメーション API に基づいています。Silk Test オートメーション SDK は、Flex のオートメーション API でサポートされているものと同じコンポーネントが同様にサポートされます。たとえば、オートメーション コードを使用してアプリケーションがコンパイルされ、連続的に SWF ファイルが読み込まれる場合、メモリ リークが発生して、最終的にアプリケーションでメモリが不足します。Flex Control Explorer サンプル アプリケーションは、この問題の影響を受けます。回避策として、Explorer が読み込むアプリケーションの SWF ファイルをオートメーション ライブラリを使用してコンパイルしない方法があります。たとえば、Explorer のメイン アプリケーションのみをオートメーション ライブラリを使用してコンパイルします。SWFLoader の代わりにモジュール ローダーを使用する方法もあります。Flex オートメーション API の詳細については、『*Apache Flex* リリース ノート』を参照してください。

テストのための Flex アプリケーションの事前コンパイル

アプリケーションをテスト用に事前コンパイルするか、または実行時の読み込みを使用することによって、アプリケーションをテストに対して有効化できます。

1. 以下のコードを構成ファイルに追加することによって、コンパイラの構成ファイルに automation.swc、automation_agent.swc、および FlexTechDomain.swc ライブラリをインクルードします。

```
<include-libraries>

...

<library>/libs/automation.swc</library>

<library>/libs/automation_agent.swc</library>

<library>pathinfo/FlexTechDomain.swc</library>

</include-libraries>
```

   注: アプリケーションでグラフを使用する場合は、automation_charts.swc ファイルも追加する必要があります。

2. コマンド ライン コンパイラで include-libraries コンパイラ オプションを使用して、automation.swc、automation_agent.swc、および FlexTechDomain.swc ライブラリの場所を指定します。

   構成ファイルは以下の場所にあります。

   Apache Flex 2 SDK – <flex_installation_directory>/frameworks/flex-config.xml

   Apache Flex データ サービス – <flex_installation_directory>/flex/WEB-INF/flex/flex-config.xml

   以下の例では、automation.swc ファイルと automation_agent.swc ファイルがアプリケーションに追加されています。

```
mxmlc  -include-libraries+=../frameworks/libs/automation.swc;../frameworks/libs/
automation_agent.swc;pathinfo/FlexTechDomain.swc  MyApp.mxml
```

   注: コマンド ラインで include-libraries オプションを明示的に設定すると、既存のライブラリに対して追加されるのではなく、既存のライブラリが上書きされます。コマンド ラインで include-libraries オプションを使用して automation.swc ファイルと automation_agent.swc ファイルを追加する場合は、+= 演算子を使用します。これにより、インクルードされる既存のライブラリが上書きされるのではなく、インクルードされる既存のライブラリに対して追加されます。

   注: Silk Test Flex オートメーション SDK は、Flex のオートメーション API に基づいています。Silk Test オートメーション SDK は、Flex のオートメーション API でサポートされているものと同じコンポーネントが同様にサポートされます。たとえば、オートメーション コードを使用してアプリケーションがコンパイルされ、連続的に SWF ファイルが読み込まれる場合、メモリ リークが発生して、最終的にアプリケーションでメモリが不足します。Flex Control Explorer サンプル アプリケーションは、この問題の影響を受けます。回避策として、Explorer が読み込むアプリケーションの SWF ファイルをオートメーション ライブラリを使用してコンパイルしない方法があります。たとえば、Explorer のメイン アプリケーションのみをオートメーション ライブラリを使用してコンパイルします。SWFLoader の代わりにモジュール ローダーを使用する方法もあります。Flex オートメーション API の詳細については、『*Apache Flex* リリース ノート』を参照してください。

実行時の読み込み

Silk Test Flex オートメーション ランチャを使用して、実行時に Flex オートメーション サポートを読み込むことができます。このアプリケーションは、オートメーション ライブラリを使用してコンパイルされており、SWFLoader クラスを使用してユーザーのアプリケーションを読み込みます。これにより、SWF ファイルにオートメーション ライブラリをコンパイルしなくても、アプリケーションが自動的にテストに対して有効化されます。Silk Test Flex オートメーション ランチャは、HTML および SWF のファイル形式で利用できます。

制限事項

- Flex オートメーション ランチャ アプリケーションは、自動的にルート アプリケーションとなります。ユーザーのアプリケーションをルート アプリケーションにする必要がある場合は、Silk Test Flex オートメーション ランチャを使用してオートメーション サポートを読み込むことができません。
- 外部ライブラリを読み込むアプリケーション（他の SWF ファイル ライブラリを読み込むアプリケーション）をテストするには、自動テストに特別な設定が必要です。実行時に読み込まれるライブラリ（ランタイム共有ライブラリ（RSL）を含む）は、読み込むアプリケーションの ApplicationDomain に読み込まれる必要があります。アプリケーションで使用される SWF ファイルが異なるアプリケーション ドメインに読み込まれた場合、自動テストの記録と再生が正しく動作しません。以下に、同じ ApplicationDomain に読み込まれるライブラリの例を示します。

```
import flash.display.*;

import flash.net.URLRequest;

import flash.system.ApplicationDomain;

import flash.system.LoaderContext;



var ldr:Loader = new Loader();



var urlReq:URLRequest = new URLRequest("RuntimeClasses.swf");

var context:LoaderContext = new LoaderContext();

context.applicationDomain = ApplicationDomain.currentDomain;

loader.load(request, context);
```

実行時読み込み

1. Silk¥Silk Test¥ng¥AutomationSDK¥Flex¥<バージョン>¥FlexAutomationLauncher ディレクトリの内容を、テストする Flex アプリケーションのディレクトリにコピーします。
2. Windows Explorer で FlexAutomationLauncher.html を開き、ファイル パスへの接尾辞として以下のパラメータを追加します。

   ```
   ?automationurl=YourApplication.swf
   ```

   YourApplication.swf は Flex アプリケーションの SWF ファイルの名前です。
3. ファイル パスへの接頭辞として file:/// を追加します。
   たとえば、ファイルの URL に ?automationurl=explorer.swf などのパラメータが含まれている場合は、以下のように入力します。を設定して、インストールするコンポーネントを指定できます。

   ```
   file:///C:/Program%20Files/Silk/Silk Test/ng/sampleapplications/Flex/3.2/
   FlexControlExplorer32/FlexAutomationLauncher.html?automationurl=explorer.swf
   ```

コマンド ラインを使用した構成情報の追加

コマンド ライン コンパイラを使用して automation.swc、automation_agent.swc、および FlexTechDomain.swc ライブラリの場所を指定するには、include-libraries コンパイラ オプションを使用します。

次の例では、automation.swc ファイルと automation_agent.swc ファイルがアプリケーションに追加されます。

```
mxmlc  -include-libraries+=../frameworks/libs/automation.swc;../frameworks/libs/
automation_agent.swc;pathinfo/FlexTechDomain.swc  MyApp.mxml
```

注: アプリケーションでグラフを使用する場合は、include-libraries コンパイラ オプションに automation_charts.swc ファイルも追加する必要があります。

コマンド ラインで include-libraries オプションを明示的に設定すると、既存のライブラリに対して追加されるのではなく、既存のライブラリが上書きされます。 コマンド ラインで include-libraries オプションを使用して automation.swc ファイルと automation_agent.swc ファイルを追加する場合は、+= 演算子を使用します。 これにより、インクルードされる既存のライブラリが上書きされるのではなく、インクルードされる既存のライブラリに対して追加されます。

Flex Builder プロジェクトに自動テストサポートを追加するには、include-libraries コンパイラ オプションに automation.swc および automation_agent.swc ライブラリも追加する必要があります。

Flex アプリケーションにパラメータを渡す

以下の手順に従って、Flex アプリケーションにパラメータを渡すことができます。

実行する前に Flex アプリケーションにパラメータを渡す

オートメーション ライブラリを使用して、実行する前に Flex アプリケーションにパラメータを渡すことができます。

1. 適切なオートメーション ライブラリを使用して、アプリケーションをコンパイルします。
2. パラメータの指定には、通常どおり標準的な Flex のメカニズムを使用します。

Flex オートメーション ランチャを使用して、実行時に Flex アプリケーションにパラメータを渡す

このタスクを開始する前に、実行時の読み込みに対応するようにアプリケーションを準備します。

1. FlexAutomationLauncher.html ファイルを開くか、または例として FlexAutomationLauncher.html を使用してファイルを作成します。
2. 以下のセクションに移動します。

```
<script  language="JavaScript"  type="text/javascript">

                        AC_FL_RunContent(eef

                            "src",  "FlexAutomationLauncher",

                            "width",  "100%",

                            "height",  "100%",

                            "align",  "middle",

                            "id",  "FlexAutomationLauncher",

                            "quality",  "high",

                            "bgcolor",  "white",

                            "name",  "FlexAutomationLauncher",

                            "allowScriptAccess","sameDomain",

                            "type",  "application/x-shockwave-flash",
```

```
                        "pluginspage",  "http://www.adobe.com/go/getflashplayer",

                        "flashvars",  "yourParameter=yourParameterValue"+
"&automationurl=YourApplication.swf"



                    );

            </script>
```

注: 「src」、「id」、および「name」の「FlexAutomationLauncher」の値は変更しないでください。

3. 「*yourParameter=yourParameterValue*」に、独自のパラメータを追加します。
4. 「*& automationurl=YourApplication.swf*」の値として、テストする Flex アプリケーションの名前を渡します。
5. ファイルを保存します。

テスト可能な Flex アプリケーションの作成

Flex 開発者は、Flex アプリケーションを可能なかぎりテストしやすくするためのテクニックを利用できます。 以下のテクニックがあります。

- オブジェクトに対するわかりやすい ID の指定
- オブジェクトの重複の回避

オブジェクトに対するわかりやすい ID の指定

テストしやすいアプリケーションを作成するには、 スクリプト内でオブジェクトを識別しやすくする必要があります。 テストするすべてのコントロールに対して、わかりやすい文字列を使用した ID プロパティの値を設定できます。

オブジェクトに対してわかりやすい ID を指定するには：

- テスト可能なすべての MXML コンポーネントに対して ID を指定して、その Flex コントロールの参照時にテスト スクリプトで一意の ID が使用できるようにします。
- これらの ID は、ユーザーがテスト スクリプト内でそのオブジェクトを容易に識別できるように、可能なかぎり人間が理解しやすい文字列にします。 たとえば、TabNavigator 内の Panel コンテナの id プロパティは、panel1 や p1 ではなく submit_panel とします。

Silk4J を使用する場合、id や childIndex などの特定のタグに基づいて、オブジェクトに対して自動的に名前が設定されます。 id プロパティに値がない場合、Silk4J では、childIndex プロパティなどの他のプロパティが使用されます。 id プロパティに値を割り当てると、テスト スクリプトを読みやすくすることができます。

オブジェクトの重複の回避

自動エージェントの処理は、実行中にオブジェクト インスタンスの一部のプロパティが変更されないことを前提としています。 実行時に Silk4J によってオブジェクト名として使用されている Flex コンポーネント プロパティを変更すると、予期しない結果が発生する可能性があります。 たとえば、automationName プロパティのない Button コントロールを作成し、最初は label プロパティに値を設定しないで、その後、label プロパティに値を設定した場合、問題が発生することがあります。 この場合、Silk4J では、automationName プロパティが設定されていない場合は Button コントロールを識別するためにコントロールの label プロパティの値が使用されます。 あとから label プロパティの値を設定したり、既存の label の値を変更すると、Silk4J ではこのオブジェクトを新しいオブジェクトとして識別し、既存のオブジェクトを参照しなくなります。

重複オブジェクトを回避するには：

- エージェントにおいて、オブジェクトの識別にどのプロパティが使用されているかを理解し、実行時にそれらのプロパティを変更しないようにします。
- 記録されたスクリプトに含まれているすべてのオブジェクトに対して、人間が理解しやすい一意の id プロパティまたは automationName プロパティを設定します。

Apache Flex アプリケーションのカスタム属性

Apache Flex アプリケーションは、あらかじめ定義されたプロパティ automationName を使用して、次のように Apache Flex コントロールに対して安定した識別子を指定します。

```
<?xml version="1.0" encoding="utf-8"?>
  <s:Group xmlns:fx="http://ns.adobe.com/mxml/2009"
       xmlns:s="library://ns.adobe.com/flex/spark"
       xmlns:mx="library://ns.adobe.com/flex/mx" width="400" height="300">
    <fx:Script>
    ...
    </fx:Script>
    <s:Button x="247" y="81" label="Button" id="button1" enabled="true"
click="button1_clickHandler(event)"
         automationName="AID_buttonRepeat"/>
    <s:Label x="128" y="123" width="315" height="18" id="label1" verticalAlign="middle"
         text="awaiting your click" textAlign="center"/>
  </s:Group>
```

Apache Flex アプリケーションのロケーターは次のようになります。

```
...//SparkApplication//SparkButton[@caption='AID_buttonRepeat'
```

注目: Apache Flex アプリケーションの場合、Silk4J では *automationName* はロケーター属性 caption に常にマップされます。*automationName* 属性が指定されていない場合、Silk4J は属性 ID をロケーター属性 caption にマップします。

Flex の AutomationName プロパティと AutomationIndex プロパティ

Flex オートメーション API には、automationName プロパティと automationIndex プロパティが用意されています。automationName を指定すると、Silk4J では、記録されたウィンドウ宣言の名前としてこの値が使用されます。わかりやすい名前を指定すると、そのオブジェクトを Silk4J で識別しやすくなります。ベスト プラクティスとして、アプリケーションのテストに含まれているすべてのオブジェクトの automationName プロパティに値を設定することをお勧めします。

automationIndex プロパティを使用して、オブジェクトに対して一意のインデックス値を割り当てます。たとえば、2 つのオブジェクトが同じ名前を共有している場合は、インデックス値を割り当てて、2 つのオブジェクトを識別します。

注: Silk Test Flex オートメーション SDK は、Flex のオートメーション API に基づいています。Silk Test オートメーション SDK は、Flex のオートメーション API でサポートされているものと同じコンポーネントが同様にサポートされます。たとえば、オートメーション コードを使用してアプリケーションがコンパイルされ、連続的に SWF ファイルが読み込まれる場合、メモリ リークが発生して、最終的にアプリケーションでメモリが不足します。Flex Control Explorer サンプル アプリケーションは、この問題の影響を受けます。回避策として、Explorer が読み込むアプリケーションの SWF ファイルをオートメーション ライブラリを使用してコンパイルしない方法があります。たとえば、Explorer のメイン アプリケーションのみをオートメーション ライブラリを使用してコンパイルします。SWFLoader の代わりにモジュール ローダーを使用する方法もあります。Flex オートメーション API の詳細については、『*Apache Flex* リリース ノート』を参照してください。

Flex クラス定義ファイル

クラス定義ファイルには、インストルメント化されたすべての Flex コンポーネントについての情報が含まれています。このファイルでは、記録中にイベントを送信でき、再生中にイベントを受け取ることができるコンポーネントについての情報が提供されます。クラス定義ファイルには、サポートされているプロパティの定義も含まれています。

Silk Test には、Flex の共通コントロールおよび特殊化されたコントロールのすべてのクラス、イベント、およびプロパティを記述するいくつかの XML ファイルが含まれています。これらの XML ファイルは、<Silk Test_install_directory>¥ng¥agent¥plugins¥com.borland.fastxd.techdomain.flex.agent_<バージョン>¥config¥automationEnvironment フォルダにあります。

独自の XML ファイルを提供する場合は、XML ファイルをこのフォルダにコピーする必要があります。Silk Test のエージェントが起動して Apache Flex のサポートを初期化するとき、このディレクトリの内容が読み込まれます。

XML ファイルの基本的な構造は以下のとおりです。

```
<TypeInformation>

<ClassInfo>

<Implementation />

<Events>

<Event />

...

</Events>

<Properties>

<Property />

...

</Properties>

</ClassInfo>

</TypeInformation>
```

Flex の automationName プロパティの設定

automationName プロパティは、テストに表示されるコンポーネント名を定義します。このプロパティのデフォルト値は、コンポーネントの種類に応じて異なります。たとえば、Button コントロールの automationName は、Button コントロールのラベルです。automationName がコントロールの id プロパティと同じ場合もありますが、常に同じであるわけではありません。

一部のコンポーネントでは、automationName プロパティの値は、Flex によってそのコンポーネントを認識しやすい属性に設定されています。これにより、テスト担当者は、テストでコンポーネントを認識しやすくなります。通常、テスト担当者は、アプリケーションの基になるソース コードにアクセスできないため、コントロールの表示されるプロパティによってそのコントロールを認識できるようにすることは有用です。たとえば、「Process Form Now」というラベルが設定された Button は、テストで FlexButton("Process Form Now") と表示されます。

新しいコンポーネントを実装する場合や、既存のコンポーネントから派生する場合は、automationName プロパティのデフォルト値をオーバーライドできます。たとえば、UIComponent では、automationName の値は、デフォルトでコンポーネントの id プロパティに設定されます。ただし、一部のコンポーネントでは、独自の方法を使用して値が設定されます。たとえば、Flex Store サンプル アプリケーションでは、コンテナを使用して製品のサムネイルが作成されています。コンテナのデフォルトの automationName はコンテナの id プロパティと同じ値となるため、あまり役立ちません。そのため、Flex Store では、製品のサムネイルを生成するカスタム コンポーネントで明示的に automationName を製品名に設定して、アプリケーションをテストしやすくしています。

例

以下の CatalogPanel.mxml カスタム コンポーネントの例では、automationName プロパティの値をカタログに表示される項目名に設定しています。これにより、デフォルトのオートメーション名を使用するよりもサムネイルを認識しやすくなります。

```
thumbs[i].automationName = catalog[i].name;
```

例

以下の例では、ComboBox コントロールの automationName プロパティを「Credit Card List」に設定しています。このように設定すると、通常、テスト ツールでは、スクリプトにおいて id プロパティではなく「Credit Card List」を使用して ComboBox が識別されます。

```
<?xml version="1.0"?>
<!-- at/SimpleComboBox.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>
    <![CDATA[
      [Bindable]
      public var cards: Array = [
        {label:"Visa", data:1},
        {label:"MasterCard", data:2},
        {label:"American Express", data:3}
      ];

      [Bindable]
      public var selectedItem:Object;
      ]
   ]>
  </mx:Script>
  <mx:Panel title="ComboBox Control Example">
    <mx:ComboBox id="cb1" dataProvider="{cards}"
      width="150"
      close="selectedItem=ComboBox(event.target).selectedItem"
      automationName="Credit Card List"
    />
    <mx:VBox width="250">
      <mx:Text width="200" color="blue" text="Select a type of credit
card." />
      <mx:Label text="You selected: {selectedItem.label}"/>
      <mx:Label text="Data: {selectedItem.data}"/>
    </mx:VBox>
  </mx:Panel>
</mx:Application>
```

automationName プロパティの値を設定すると、オブジェクト名が実行時に変更されないことが保証されます。このことは、予期しない結果の回避に役立ちます。

automationName プロパティの値を設定すると、テストでは、デフォルト値ではなく、その値が使用されます。たとえば、Silk4J では、デフォルトで、スクリプトにおいて Button コントロールの label プロパティがボタンの名前として使用されます。この場合、ラベルが変更されると、スクリプトが動作しなくなります。automationName プロパティの値を明示的に設定することによって、このような事態を回避できます。

ラベルがなく、アイコンがあるボタンは、インデックス番号によって記録されます。この場合は、automationName プロパティをわかりやすい文字列に設定して、テスト担当者がスクリプトでボタンを認識できるようにします。automationName プロパティ

の値を設定したあとは、コンポーネントのライフ サイクル全体を通して値を変更しないでください。項目レンダラでは、automationName プロパティではなく automationValue プロパティを使用します。automationValue プロパティを使用するには、createAutomationIDPart() メソッドをオーバーライドして、automationName プロパティに割り当てる新しい値を返します。以下に例を示します。

```
<mx:List xmlns:mx="http://www.adobe.com/2006/mxml">
  <mx:Script>

      import mx.automation.IAutomationObject;
      override public function
      createAutomationIDPart(item:IAutomationObject):Object {
        var id:Object = super.createAutomationIDPart(item);
        id["automationName"] = id["automationIndex"];
        return id;
      }

  </mx:Script>
</mx:List>
```

このテクニックを使用して、任意のコンテナまたはリスト形式コントロールの子にインデックス値を追加します。子が自分自身のインデックスを指定する方法はありません。

名前またはインデックスを使用するように Flex の Select メソッドを設定

Flex の Select メソッドは、選択するコントロールの Name または Index を使用して記録できます。デフォルトで、Silk Test では、コントロールの名前を使用して Select メソッドが記録されます。ただし、コントロールのインデックスを使用して Select イベントを記録するように環境を変更したり、名前を使用した記録とインデックスを使用した記録を切り替えたりすることができます。

1. インデックスを使用するように変更するクラスを特定します。
   以下のコントロールでは、インデックスを使用して Select イベントを記録できます。
   - FlexList
   - FlexTree
   - FlexDataGrid
   - FlexOLAPDataGrid
   - FlexComboBox
   - FlexAdvancedDataGrid
2. 変更するクラスに関連する XML ファイルを特定します。
   上記のコントロールに関連する XML ファイルには、FlexCommonControls.xml、AdvancedDataGrid.xml、または OLAPDataGrid.xml があります。
3. 変更するクラスに関連する XML ファイルに移動します。
   XML ファイルは、<Silk Test_install_directory>¥ng¥agent¥plugins ¥com.borland.fastxd.techdomain.flex.agent_<バージョン>¥config¥automationEnvironment フォルダにあります。
4. 対応する XML ファイルで、以下の変更を行います。

   ```
   <ClassInfo Extends="FlexList" Name="FlexControlName"
   EnableIndexBasedSelection=”true” >

   …

   </ClassInfo>
   ```

   たとえば、「FlexControlName」として「FlexList」を使用し、FlexCommonControls.xml ファイルを変更できます。

この変更では、FlexList::SelectIndex イベントの記録に IndexBasedSelection が使用されています。

注: コードの EnableIndexBasedSelection= を false に設定するか、またはこのブール値を削除すると、記録で名前が使用される設定に戻ります（FlexList::Select イベント）。

5. これらの変更内容を有効にするには、Flex アプリケーションおよび Open Agent を再起動します。

Flex コンテナのコーディング

コンテナは、ユーザー対話（ユーザーが Accordion コンテナの次のページに移動したなど）の記録、およびテスト スクリプト内でのコントロールに対する一意の場所の提供の両方の目的で使用されるため、他の種類のコントロールとは異なります。

オートメーション階層におけるコンテナの追加と削除

通常、自動テスト機能のスクリプトでは、ネストされたコンテナについての詳細情報は少なく抑えられます。テストの結果やコントロールの識別に影響がないコンテナは、スクリプトから削除されます。削除対象となるコンテナは、HBox、VBox、Canvas などの、レイアウトの目的でのみ使用されるコンテナです。ただし、ViewStack、TabNavigator、Accordion などの複数ビュー ナビゲータ コンテナで使用されている場合は削除されません。このような場合、コンテナはオートメーション階層に追加されて、ナビゲーションに使用されます。

多くの複合コンポーネントでは、Canvas や VBox などのコンテナを使用して、子が整理されます。これらのコンテナは、アプリケーション上では視覚的な効果を持ちません。この結果、これらのコンテナでは、ユーザー操作は実行されず、操作を視覚的に記録する必要もないため、通常、これらのコンテナはテストから除外されます。テストからコンテナを除外することによって、関連するテスト スクリプトが簡潔になり、読みやすくなります。

コンテナを記録から除外するには、コンテナの showInAutomationHierarchy プロパティを false に設定します（子は除外されません）。このプロパティは、UIComponent クラスによって定義されているため、UIComponent のサブクラスであるすべてのコンテナにこのプロパティが存在します。階層で表示されないコンテナの子は、階層内でそのコンテナの次に上位の親の子として表示されます。

showInAutomationHierarchy プロパティのデフォルト値は、コンテナの種類に応じて異なります。Panel、Accordion、Application、DividedBox、Form などのコンテナではデフォルト値は true であり、Canvas、HBox、VBox、FormItem などのコンテナではデフォルト値は false です。

以下の例では、VBox コンテナがテスト スクリプトの階層に組み込まれています。

```
<?xml version="1.0"?>
<!-- at/NestedButton.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
<mx:Panel title="ComboBox Control Example">
<mx:HBox id="hb">
<mx:VBox id="vb1" showInAutomationHierarchy="true">
<mx:Canvas id="c1">
<mx:Button id="b1" automationName="Nested Button 1" label="Click Me" />
</mx:Canvas>
</mx:VBox>
<mx:VBox id="vb2" showInAutomationHierarchy="true">
<mx:Canvas id="c2">
<mx:Button id="b2" automationName="Nested Button 2" label="Click Me 2" />
</mx:Canvas>
</mx:VBox>
</mx:HBox>
</mx:Panel>
</mx:Application>
```

複数ビュー コンテナ

TabNavigator や Accordion コンテナなどの複数ビュー コンテナ内の複数のタブに同じラベルを使用しないでください。 同じラベルを使用することもできますが、このことは、通常、推奨 UI 設計プラクティス

として推奨されていません。このようにすると、テスト環境においてコントロールの識別に問題が発生することがあります。

Flex 自動テスト ワークフロー

Flex アプリケーションのテストの Silk4J ワークフローは、以下のとおりです。

- 自動テストの初期化
- 自動テストの記録
- 自動テストの再生

Flex 自動テストの初期化

ユーザーが Flex アプリケーションを起動すると、以下の初期化イベントが発生します。

1. オートメーション初期化コードによって、コンポーネントの委譲クラスがコンポーネントのクラスに関連付けられます。
2. コンポーネントの委譲クラスは、IAutomationObject インターフェイスを実装します。
3. AutomationManager のインスタンスがミックスインの init() メソッドで作成されます。（AutomationManager はミックスインです。）
4. SystemManager によってアプリケーションが初期化されます。コンポーネント インスタンスおよび対応する委譲インスタンスが作成されます。委譲インスタンスによって、目的のイベントに対するイベント リスナーが追加されます。
5. Silk4J FlexTechDomain はミックスインです。FlexTechDomain の init() メソッドで、FlexTechDomain が SystemManager.APPLICATION_COMPLETE イベントに登録されます。イベントを受信すると、FlexTechDomain インスタンスが作成されます。
6. FlexTechDomain インスタンスが、同じマシン上の記録および再生機能に登録する Silk Test Agent に TCP/IP ソケット経由で接続します。
7. FlexTechDomain は、自動環境についての情報を要求します。この情報は XML ファイルに格納され、Silk Test Agent から FlexTechDomain に転送されます。

Flex 自動テストの記録

ユーザーが Silk4J で Flex アプリケーションの新しいテストを記録すると、以下のイベントが発生します。

1. Silk4J によって Silk Test Agent が呼び出されて、記録が開始されます。Agent は、このコマンドを FlexTechDomain インスタンスに転送します。
2. FlexTechDomain は、beginRecording() を呼び出すことによって、AutomationManager に対して記録の開始を通知します。AutomationManager は、SystemManager からの AutomationRecordEvent.RECORD イベントに対するリスナーを追加します。
3. ユーザーがアプリケーションを操作します。たとえば、ユーザーが Button コントロールをクリックしたとします。
4. ButtonDelegate.clickEventHandler() メソッドによって、プロパティとしてクリック イベントと Button のインスタンスが指定された AutomationRecordEvent イベントがディスパッチされます。
5. AutomationManager は、XML 環境情報に基づいて、クリック イベントのどのプロパティを格納するかを決定します。値が適切な型または書式に変換されます。記録イベントがディスパッチされます。
6. FlexTechDomain イベント ハンドラがイベントを受信します。AutomationManager.createID() メソッドが呼び出されて、ボタンの AutomationID オブジェクトが作成されます。このオブジェクトは、オブジェクト識別用の構造体を提供します。AutomationID 構造体は、AutomationIDParts の配列になっています。AutomationIDParts は、IAutomationObject を使用して作成されます。（Button コントロールの UIComponent.id、automationName、automationValue、childIndex、および label プロパティが読み込まれて、オブジェクトに格納されます。XML 情報に、label プロパティを Button の識別に使用できることが指定されているため、label プロパティが使用されます。）
7. FlexTechDomain は、AutomationManager.getParent() メソッドを使用して、Button の論理的な親を取得します。アプリケーション レベルまでの各レベルで、親コントロールの AutomationIDParts オブジェクトが収集されます。

8. すべての AutomationIDParts が AutomationID オブジェクトの一部として組み込まれます。
9. FlexTechDomain は、Silk4J への呼び出しでこの情報を送信します。
10. ユーザーが記録を停止すると、FlexTechDomain.endRecording() メソッドが呼び出されます。

Flex 自動テストの再生

ユーザーが Silk4J で **再生** ボタンをクリックすると、以下のイベントが発生します。

1. 各スクリプト呼び出しにおいて、Silk4J は Silk Test Agent に接続し、実行されるスクリプト呼び出しの情報を送信します。 この情報には、完全なウィンドウ宣言、イベント名、およびパラメータが含まれています。
2. Silk Test Agent は、その情報を FlexTechDomain に転送します。
3. FlexTechDomain は、ウィンドウ宣言情報と共に AutomaionManager.resolveIDToSingleObject を使用します。 AutomationManager は、説明情報（automationName、automationIndex、id など）に基づいて、解決したオブジェクトを返します。
4. Flex コントロールが解決されると、FlexTechDomain は AutomationManager.replayAutomatableEvent() を呼び出して、イベントを再生します。
5. AutomationManager.replayAutomatableEvent() メソッドによって、委譲クラスの IAutomationObject.replayAutomatableEvent() メソッドが呼び出されます。 委譲では、IAutomationObjectHelper.replayMouseEvent() メソッド（または replayKeyboardEvent() などの他のいずれかの再生メソッド）を使用してイベントが再生されます。
6. スクリプトに検証がある場合、FlexTechDomain は AutomationManager.getProperties() を呼び出して、検証する必要がある値にアクセスします。

Apache Flex アプリケーションのスタイル

Apache Flex 3.x で開発されたアプリケーションについて、Silk4J ではスタイルとプロパティを区別しません。この結果、スタイルはプロパティとして公開されます。ただし、Apache Flex 4.x の Spark という接頭辞が付いているすべての新しい Flex コントロール（SparkButton など）では、スタイルがプロパティとして公開されません。この結果、Flex 4.x コントロールの GetProperty() メソッドおよび GetPropertyList() メソッドでは color や fontSize などのスタイルが返されず、text や name などのプロパティのみが返されます。

GetStyle(string styleName) メソッドは、スタイルの値を文字列として返します。どのようなスタイルが存在するかを確認するには、次の Adobe ヘルプを参照してください： *http://help.adobe.com/ja_JP/FlashPlatform/reference/actionscript/3/package-detail.html*.

スタイルが設定されていない場合は、再生中に StyleNotSetException が発生します。

FlexTree などの Flex 3.x コントロールでは、GetProperty() を使用してスタイルを取得できます。GetStyle() を使用することもできます。Flex 3.x コントロールでは、GetProperty() メソッドと GetStyle() メソッドの両方が動作します。

色スタイルの計算

Flex では、色は数値として表されます。色は、以下の式を使用して計算できます。

```
red*65536  +  green*256  +  blue
```

例

以下のスクリプト例では、フォント サイズが 12 であるかどうかを検証しています。16711680 という数値は、255*65536 + 0*256 + 0 という式から算出されます。この値は赤色を表し、スクリプトは背景色に対してこの色を検証します。

```
Assert.That(control.GetStyle("fontSize"),  [Is].EqualTo("12"))
Assert.That(label.GetStyle("backgroundColor"),  [Is].EqualTo("16711680"))
```

Adobe Flash Player のセキュリティ制約に対応するための Flex アプリケーションの構成

Adobe Flash Player 10 では、セキュリティ モデルが以前のバージョンから変更されています。Flash Player を使用するテストを記録する場合、記録は想定どおりに動作します。ただし、テストを再生する場合は、特定の状況で高レベルのクリックが行われると、予期しない結果が発生します。たとえば、**ファイル参照** ダイアログ ボックスをプログラムから開くことができません。このシナリオの再生を試みると、セキュリティ制約が原因でテストに失敗します。

このセキュリティ制約を回避するには、ダイアログ ボックスを開くボタンに対して低レベルのクリックを実行します。低レベルのクリックを作成するには、click メソッドにパラメータを追加します。

たとえば、SparkButton.click() の代わりに SparkButton.click(MouseButton.LEFT) を使用します。パラメータを指定しない click() は高レベルのクリックとして再生され、パラメータを指定したクリック（ボタンなど）は低レベルのクリックとして再生されます。

1. Flash Player を使用するテストを記録します。
2. click メソッドに移動して、パラメータを追加します。
   たとえば、**ファイルを開く** ダイアログ ボックスを開くには、以下のように指定します： 。

   ```
   SparkButton("@caption='Open  File  Dialog...'").click(MouseButton.LEFT)
   ```

   テストを再生すると、想定どおりに動作します。

Apache Flex アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Flex アプリケーションがサポートする属性は次のとおりです。

- automationName
- caption（automationName と同様）
- automationClassName（FlexButton など）
- className（実装クラスの完全修飾名。 mx.controls.Button など）
- automationIndex（FlexAutomation のビューでのコントロールのインデックス。 index:1 など）
- index（automationIndex と同様。ただし、接頭辞はなし。 1 など）
- id (コントロールの ID)
- windowId (id と同様)
- label（コントロールのラベル）
- すべての動的ロケーター属性

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

動的ロケーター属性の詳細については、「動的ロケーター属性」を参照してください。

Silk4J が Apache Flex コントロールを認識できない理由

Web サーバーを通してアクセスしている Apache Flex アプリケーションのコントロールを Silk4J が認識できない場合、以下のことを行ってみてください。

- Adobe オートメーション ライブラリと Apache Flex バージョン用に適した FlexTechDomain.swc を使用して、Apache Flex アプリケーションをコンパイルします。

- 実行時ローディングを使用します。
- Apache Flex アプリケーションを空の id 属性を使用して埋め込んでいると、Apache Flex コントロールは認識されません。

Java AWT/Swing のサポート

Silk4J は、Java AWT/Swing コントロールを使用するアプリケーションまたはアプレットのテストを組み込みでサポートしています。Java AWT/Swing を使用するアプリケーションまたはアプレットを設定すると、Silk4J は標準の AWT/Swing コントロールのテストのサポートを組み込みで提供します。

注: Java AWT/Swing アプリケーションまたはアプレットに埋め込まれた Java SWT コントロールや、Java SWT アプリケーションに埋め込まれた Java AWT/Swing コントロールもテストできます。

注: イメージ クリックの記録は、Java AWT/Swing コントロールを使用するアプリケーションまたはアプレットではサポートされません。

サンプル アプリケーション

Silk Test は、サンプルの Swing テスト アプリケーションを提供しています。サンプル アプリケーションを *http://supportline.microfocus.com/websync/SilkTest.aspx* からダウンロードしてインストールします。 サンプル アプリケーションをインストールしたあと、(Microsoft Windows 7) **スタート** > **すべてのプログラム** > **Silk** > **Silk Test** > **Sample Applications** > **Java Swing** > **Swing Test Application**、または (Microsoft Windows 10) **スタート** > **Silk** をクリックします。

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

サポートするコントロール

Java AWT/Swing のテストで利用可能なコントロールの完全な一覧については、「API リファレンス」の以下のパッケージ内でサポートされる Swing クラスの一覧を参照してください。

- com.borland.silktest.jtf.swing：Java Swing 固有のクラスを含みます。
- com.borland.silktest.jtf.common.types：データ型を含みます。

Java AWT/Swing アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Java AWT/Swing でサポートされる属性には以下のものがあります。

- caption
- priorlabel : 隣接するラベル フィールドのテキストによってテキスト入力フィールドを識別します。 通常、フォームのすべての入力フィールドに、入力の目的を説明するラベルがあります。 caption のないコントロールの場合、自動的に属性 **priorlabel** がロケーターに使用されます。 コントロールの **priorlabel** 値 (テキスト入力フィールドなど) には、コントロールの左側または上にある最も近いラベルの caption が使用されます。
- name
- accessibleName
- *Swing のみ*：すべてのカスタム オブジェクトの定義属性は、ウィジェットに putClientProperty("propertyName", "propertyValue") で設定されます。

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別され

ますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード？および * をサポートしています。

Java メソッドの動的な呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティーの取得、またはプロパティーの設定を直接実行できます。また、このコントロールの Silk4J API で使用できないメソッドおよびプロパティーも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4J API を通して公開されていない場合に特に便利です。

オブジェクトの動的メソッドは invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

オブジェクトの複数の動的メソッドは invokeMethods メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

動的プロパティの取得には getProperty メソッドを、動的プロパティの設定には setProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、getPropertyList メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを String 型の入力パラメータとして設定する必要がある SetTitle というメソッドを呼び出すには、次のように入力します：

```
control.invoke("SetTitle","my  new  title");
```

注: 通常、ほとんどのプロパティは読み取り専用で、設定できません。

注: ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティーを取得する場合、Reflection を使用します。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4J がサポートするコントロールのメソッドとプロパティー。
- SWT、AWT、または Swing ウィジェットのすべてのパブリック メソッド
- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティー。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- プリミティブ型 (boolean、integer、long、double、string)

  プリミティブ型 (int など) とオブジェクト タイプ (java.lang.Integer など) の両方がサポートされます。プリミティブ型は必要に応じて拡大変換されます。たとえば、long が必要な場所で int を渡すことができます。
- 列挙型

  列挙パラメータは文字列として渡す必要があります。文字列は、列挙値の名前と一致しなければなりません。たとえば、メソッドが列挙型 java.sql.ClientInfoStatus のパラメータを必要とする場合、次の文字列値を使用できます： REASON_UNKNOWN、REASON_UNKNOWN_PROPERTY、REASON_VALUE_INVALID、REASON_VALUE_TRUNCATED
- リスト

  リスト、配列、または可変長引数のパラメータを持つメソッドを呼び出すことができます。リストの要素がターゲットの配列型に代入可能の場合、配列型への変換は自動的に行われます。

- その他のコントロール

  コントロール パラメーターは、TestObject として渡したり、返したりできます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4J 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、null が返されます。

Silk4J を構成して、Java Network Launching Protocol (JNLP) を使用するアプリケーションを起動する

Java Network Launching Protocol (JNLP) を使用して起動するアプリケーションでは、Silk4J に追加の構成が必要です。これらのアプリケーションは Web から起動されるため、実際のアプリケーションと「Web Start」を起動するように、アプリケーション構成を手動で構成する必要があります。このようにしない場合、アプリケーションがすでに実行されていないかぎり、テストを再生すると、失敗します。

1. Silk4J がアプリケーションを起動できずにテストが失敗する場合は、アプリケーション構成を編集します。
2. Silk Test ツールバー アイコン の隣にあるドロップダウン矢印 をクリックして、**アプリケーション構成の編集** を選択します。 **アプリケーション構成の編集** ダイアログ ボックスが開き、既存のアプリケーション構成がリストされます。
3. 基本状態を編集して、再生中に Web Start が起動されるようにします。
   a) **編集** をクリックします。
   b) **実行可能ファイル パターン** テキスト ボックスに、javaws.exe への絶対パスを入力します。
   たとえば、以下のように入力します。
   ```
   %ProgramFiles%¥Java¥jre6¥bin¥javaws.exe
   ```
   c) **コマンド ライン パターン** テキスト ボックスに、Web Start への URL を含むコマンド ライン パターンを入力します。
   ```
   "<url-to-jnlp-file>"
   ```
   たとえば、SwingSet3 アプリケーションの場合、以下のように入力します。
   ```
   "http://download.java.net/javadesktop/swingset3/SwingSet3.jnlp"
   ```
   d) **OK** をクリックします。
4. **OK** をクリックします。テストは、基本状態を使用し、Web 起動アプリケーションとアプリケーション構成の実行可能パターンを開始して javaw.exe に接続し、テストを実行します。

テストを実行すると、アプリケーション構成の EXE ファイルが基本状態の EXE ファイルと一致しないという警告が表示されます。テストは予想したとおりに実行されているため、このメッセージは無視できます。

Java AWT/Swing テクノロジ ドメインでの priorLabel の判別

Java AWT/Swing テクノロジ ドメインで priorLabel を判別するには、ターゲット コントロールと同じウィンドウ内のすべてのラベルおよびグループを考慮する必要があります。 判別の基準は、次のとおりです。

- priorLabel の候補とみなされるのは、コントロールの上または左にあるラベル、およびコントロールが属しているグループのみです。
- コントロールの親が JViewPort または ScrollPane の場合、アルゴリズムはこのコントロールを含むウィンドウが親であるかのように機能し、外側の要素はどれも関連しないとみなされます。

- 最も単純なケースでは、コントロールに最も近いラベルが priorLabel として使用されます。
- 2 つのラベルがコントロールから等距離にあり、1 つがコントロールの左、もう 1 つが上にある場合は、左側のラベルが優先します。
- 適したラベルがない場合は、最も近いグループのキャプションが使用されます。

Oracle Forms のサポート

この機能がサポートされるのは、Open Agent を使用している場合のみです。

Silk4J は、Oracle Forms ベースのアプリケーションのテストを組み込みでサポートします。

注: コントロールによっては、Silk4J は低レベルの記録のみをサポートします。

サポート対象バージョン、既知の問題、回避策の詳細については、「リリース ノート」を参照してください。 Oracle Forms のテストで利用可能なコントロールの完全な一覧については、「API リファレンス」の以下のパッケージ内でサポートされる Oracle Forms クラスの一覧を参照してください。

Oracle Forms テストの前提条件

Oracle Forms を使ってビルドしたアプリケーションをテストするには、次の前提条件を満たす必要があります。

- 次世代 Java プラグインを有効化する必要があります。この設定は、デフォルトで有効になっています。**Java コントロール パネル** で設定を変更できます。次世代 Java プラグインの詳細については、Java のドキュメントを参照してください。
- Java セキュリティ ダイアログをテストの実行中に表示されないようにするには、アプレットにサインする必要があります。
- Micro Focus は、Names プロパティを有効にすることを推奨します。このプロパティを有効にすると、Oracle Forms ランタイムは内部の名前 を公開します。この名前は、コントロールの開発者が、コントロールの Name プロパティとしてコントロールに対して指定したものです。そうでない場合、Name プロパティには、コントロールのクラス名とインデックスから構成された値が設定されます。これを使用すると、Silk4J はコントロールに対して安定したロケーターを生成できます。

Oracle Forms アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Oracle Forms がサポートする属性は次のとおりです。

- priorlabel : 隣接するラベル フィールドのテキストによってテキスト入力フィールドを識別します。 通常、フォームのすべての入力フィールドに、入力の目的を説明するラベルがあります。 caption のないコントロールの場合、自動的に属性 **priorlabel** がロケーターに使用されます。 コントロールの **priorlabel** 値 (テキスト入力フィールドなど) には、コントロールの左側または上にある最も近いラベルの caption が使用されます。
- name
- accessibleName

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Java SWT と Eclipse RCP のサポート

Silk Test は、SWT (Standard Widget Toolkit) コントロールのウィジェットを使用したアプリケーションのテストの組み込みサポートを提供します。Java SWT/RCP アプリケーションを設定すると、Silk Test は標準の Java SWT/RCP コントロールのテストのサポートを組み込みで提供します。

Silk Test では、以下をサポートしています。

- Java AWT/Swing アプリケーションに埋め込まれた Java SWT コントロール、および Java SWT アプリケーションに埋め込まれた Java AWT/Swing コントロールのテスト。
- Java SWT アプリケーションのテスト。
- レンダリングに SWT ウィジェットを使用した Eclipse ベースのアプリケーション。 Silk Test は、Eclipse IDE ベース、および RCP ベースの両方のアプリケーションをサポートします。

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

サポートするコントロール

SWT テストで使用できるウィジェットの完全なリストについては、「*Java SWT* クラス リファレンス」を参照してください。

Java SWT カスタム属性

カスタム属性をテスト アプリケーションに追加して、テストをより安定させることができます。 たとえば、Java SWT では、GUI を実装する開発者が属性（'silkTestAutomationId' など）をウィジェットに対して定義することによって、アプリケーション内でそのウィジェットを一意に識別することができます。これにより、Silk4J を使用するテスト担当者は、その属性（この場合は 'silkTestAutomationId'）をカスタム属性のリストに追加すると、その一意の ID によってコントロールを識別できるようになります。 カスタム属性を使用すると、caption や index のような他の属性よりも高い信頼性を得ることができます。これは、caption はアプリケーションを他の言語に翻訳した場合に変更され、index は定義済みのウィジェットより前に他のウィジェットが追加されると変更されるためです。

複数のオブジェクトに同じカスタム属性の値が割り当てられた場合は、そのカスタム属性を呼び出したときにその値を持つすべてのオブジェクトが返されます。 たとえば、一意の ID として 'loginName' を 2 つの異なるテキスト フィールドに割り当てた場合は、'loginName' 属性を呼び出したときに、両方のフィールドが返されます。

Java SWT の例

以下のコードを使用して、テストするアプリケーションにボタンを作成する場合：

```
Button myButton = Button(parent, SWT.NONE);

myButton.setData("SilkTestAutomationId", "myButtonId");
```

テストの XPath クエリ文字列に属性を追加するには、以下のクエリを使用します。

```
Dim button =
desktop.PushButton("@SilkTestAutomationId='myButton'")
```

Java SWT アプリケーションをカスタム属性のテストに対して有効化にするには、開発者はカスタム属性をアプリケーションに含める必要があります。 属性を含めるには org.swt.widgets.Widget.setData(String key, Object value) メソッドを使用します。

Java SWT アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Java SWT がサポートする属性は次のとおりです。

- caption
- すべてのカスタム オブジェクト定義属性

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Java メソッドの動的な呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティーの取得、またはプロパティーの設定を直接実行できます。また、このコントロールの Silk4J API で使用できないメソッドおよびプロパティーも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4J API を通して公開されていない場合に特に便利です。

オブジェクトの動的メソッドは invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

オブジェクトの複数の動的メソッドは invokeMethods メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

動的プロパティの取得には getProperty メソッドを、動的プロパティの設定には setProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、getPropertyList メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを String 型の入力パラメータとして設定する必要がある SetTitle というメソッドを呼び出すには、次のように入力します：

```
control.invoke("SetTitle","my  new  title");
```

注: 通常、ほとんどのプロパティは読み取り専用で、設定できません。

注: ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティーを取得する場合、Reflection を使用します。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4J がサポートするコントロールのメソッドとプロパティー。
- SWT、AWT、または Swing ウィジェットのすべてのパブリック メソッド
- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティー。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- プリミティブ型 (boolean、integer、long、double、string)

  プリミティブ型 (int など) とオブジェクト タイプ (java.lang.Integer など) の両方がサポートされます。プリミティブ型は必要に応じて拡大変換されます。たとえば、long が必要な場所で int を渡すことができます。
- 列挙型

  列挙パラメータは文字列として渡す必要があります。文字列は、列挙値の名前と一致しなければなりません。たとえば、メソッドが列挙型 java.sql.ClientInfoStatus のパラメータを必要とする場合、次の文字列値を使用できます： REASON_UNKNOWN、REASON_UNKNOWN_PROPERTY、REASON_VALUE_INVALID、REASON_VALUE_TRUNCATED
- リスト

  リスト、配列、または可変長引数のパラメータを持つメソッドを呼び出すことができます。リストの要素がターゲットの配列型に代入可能の場合、配列型への変換は自動的に行われます。
- その他のコントロール

  コントロール パラメーターは、TestObject として渡したり、返したりできます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4J 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、null が返されます。

Java SWT と Eclipse アプリケーションのトラブルシューティング

一部の SWTTree メソッドが低レベル再生時に再生されない

低レベル再生を使用したとき、expand や collapse などの一部の SWTTree メソッドが再生されません。

この問題を解決するには、再生モードを **デフォルト** に設定します。詳細については、「再生オプションの設定」を参照してください。

SWTTree の非表示ノードの選択

低レベル再生を使用したとき、Silk4J は SWTTree の非表示ノードを操作できません。

この問題を解決するには、再生モードを **デフォルト** に設定します。詳細については、「再生オプションの設定」を参照してください。

モバイル アプリケーションのテスト

Silk4J では、ネイティブ モバイル アプリケーション（アプリ）およびモバイル Web アプリケーションを自動的にテストすることができます。Silk4J を使用してモバイル アプリケーションを自動的にテストすることには、次のメリットがあります。

- モバイル アプリケーションのテスト時間を大幅に減少させることができます。
- テストを一旦作成すれば、数多くの異なるデバイスやプラットフォーム上でモバイル アプリケーションをテストできます。
- エンタープライズ モバイル アプリケーションに要求される信頼性とパフォーマンスを確保できます。
- QA チームのメンバーおよびモバイル アプリケーションの開発者の効率を向上できます。

- モバイル アプリケーションは、多くのモバイル デバイスとプラットフォームで動作することを要求されるため、アジャイルにフォーカスした開発環境にとって手動テストは十分効率的とは言えない場合があります。

注: Silk4J を使用してネイティブ モバイル アプリケーションやハイブリッド アプリケーションをテストするには、ネイティブ モバイル ライセンスが必要です。詳細については、「ライセンス情報」を参照してください。

注: Silk4J は、Android および iOS デバイスの両方でのモバイル アプリのテストをサポートします。

モバイル アプリケーションのテストをサポートするオペレーティング システムとサポートするブラウザーについての情報は、『リリース ノート』を参照してください。

Android

Silk4J では、Android デバイスまたは Android エミュレータ上のモバイル アプリケーションをテストすることができます。

Android 上のモバイル アプリケーションのテストにおける前提条件

Android デバイスや Android エミュレータ上のモバイル アプリケーションをテストする前に、次の前提条件を満たしていることを確認してください。

- Silk4J を実行するマシンに Java インストールされており、Java へのパスが *Path* 環境変数に追加されていることを確認してください。 Silk4J は、JRE または JDK のいずれかをモバイル テストで必要とします。Java がインストールされていない場合は、C:¥Program Files (x86)¥Silk¥SilkTest¥ng¥jre¥bin を *Path* に追加してください。
- ネイティブ モバイル アプリに Web ビューを追加したハイブリッド アプリを作成した場合は、アプリを Silk4J でテスト可能にするために、次のコードをアプリに追加してください。

```
WebView.setWebContentsDebuggingEnabled(true);
webView.getSettings().setJavaScriptEnabled(true);
```

- Silk4J では、Android 4.4 のデバイス画面のライブ ビュー表示をサポートしていません。 このため、Android 5 以降を使用することを Micro Focus は推奨しています。

Android 上のモバイル アプリケーションのテスト

物理 Android デバイスや Android エミュレータ上のモバイル アプリケーションをテストするには、次のタスクを実行します。

1. Android 上のモバイル アプリケーションのテストにおける前提条件を満たすことを確認します。
   詳細については、「*Android 上のモバイル アプリケーションのテストにおける前提条件*」を参照してください。
2. Android エミュレータ上のモバイル アプリケーションをテストするには、Silk4J 用にエミュレータを設定します。
   詳細については、「 *Silk Test 用に Android エミュレータを設定する*」を参照してください。
3. Silk4J をインストールしたマシンで Android エミュレータを開始するか、デバイスを接続します。
4. 物理 Android デバイスを初めて使用するマシンでテストする場合、適切な Android USB ドライバをマシンにインストールします。
   詳細については、「*USB ドライバのインストール*」を参照してください。
5. 物理 Android デバイス上でモバイル アプリケーションをテストする場合は、Android デバイスの USB デバッグを有効化します。
   詳細については、「*USB デバッグの有効化*」を参照してください。
6. モバイル アプリケーション用の Silk4J プロジェクトを作成します。
7. モバイル アプリケーション用のテストを作成します。

8. テストで実行する操作を記録します。**記録** ウィンドウを開始すると、**アプリケーションの選択** ダイアログ ボックスが開きます。
9. モバイル Web アプリケーションをテストするには：
   a) **Web** タブを選択します。
   b) 使用するモバイル ブラウザーを選択します。
   c) **移動する URL の入力** テキスト ボックスに、開く Web ページを指定します。
10. ネイティブ モバイル アプリケーションまたはハイブリッド アプリケーションをテストするには：

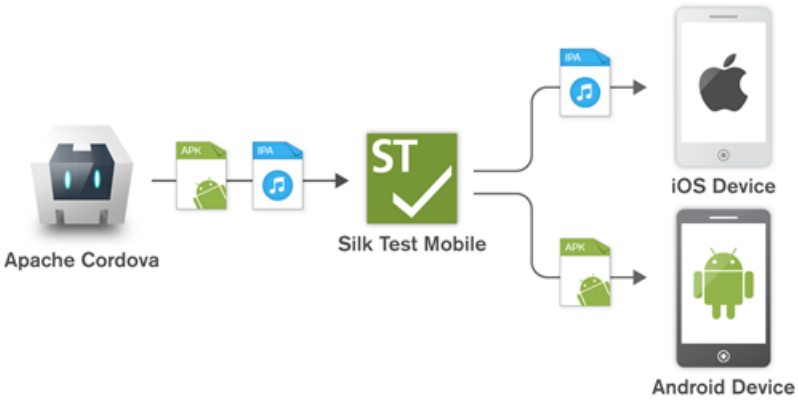
    注: Silk4J を使用してネイティブ モバイル アプリケーションやハイブリッド アプリケーションをテストするには、ネイティブ モバイル ライセンスが必要です。詳細については、「ライセンス情報」を参照してください。

   a) **モバイル** タブを選択します。
   b) アプリをテストするモバイル デバイスをリストから選択します。
   c) **参照** をクリックしてアプリ ファイルを選択するか、アプリ ファイルへの完全パスを **モバイル アプリ ファイル** テキスト フィールドに入力します。
      このパスでは、Silk4J は HTTP および UNC 形式をサポートします。
      Silk4J は、モバイル デバイスまたはエミュレータ上に指定したアプリをインストールします。
11. **終了** をクリックします。
    Android デバイスまたはエミュレータの画面が、テスト中にロックされないようにしてください。マシンに接続中にデバイスがロックされないようにするには、**開発者向けオプション** を開きます。**スリープモードにしない** または **充電中に画面をスリープにしない** をチェックします。
12. **記録** ウィンドウを使用して、モバイル アプリケーションに対するテストを記録します。
    詳細については、「モバイル アプリケーションの記録」を参照してください。
13. すべての操作の記録を終えたら、記録を停止します。
14. テストを再生します。
15. テスト結果を分析します。

Android 上のハイブリッド アプリケーションのテスト

ハイブリッド アプリケーション（アプリ）は、デバイス上で実行されるネイティブ アプリケーションのようなアプリですが、HTML5、CSS、JavaScript などの Web テクノロジを使用して記述されたアプリです。

Silk4J は、ネイティブ コンテナに埋め込まれた単一の Web ビューで構成されたデバッグ ハイブリッド アプリのテストにする完全なブラウザー サポートを提供します。このようなハイブリッド アプリの一般的な例は、Apache Cordova アプリケーションです。



非デバッグ ハイブリッド アプリをテスト可能にするには、次のコードをアプリに追加してアプリをリモート デバッグできるようにします。

```
WebView.setWebContentsDebuggingEnabled(true);
webView.getSettings().setJavaScriptEnabled(true);
```

リモート デバッグを有効化しない非デバッグ ハイブリッド アプリや、複数の Web ビューを含んだハイブリッド アプリをテストするには、Silk4J フォールバック サポートを有効化するために、オプション OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT を TRUE に設定します。詳細については、「詳細オプションの設定」を参照してください。フォールバック サポートを有効化すると、Silk4J は Web ビューのコントロールをブラウザー コントロールではなく、ネイティブ モバイル コントロールとして解決して処理します。たとえば、以下のコードは、ブラウザー サポートを使用したときのリンクのクリックです。

```
desktop.setOption(CommonOptions.OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT,
false);
desktop.<DomLink>  find("//INPUT[@id='email']").click();
```

フォールバック サポートを有効化すると、同じリンクをクリックするコードは次のようになります。

```
desktop.setOption(CommonOptions.OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT,
true);
desktop.<DomLink>  find("//MobileTextField[@resource-id='email']").click();
```

Silk4J は、Chrome リモート デバッグをサポートする Web ビューを検出できます。 Silk4J は、*com.android.webview* パッケージまたは *com.google.android.webview* パッケージ（多くの Android デバイス上のデフォルト パッケージ）のいずれかを使用した Web ビューを検出できます。

注: Silk4J は、Android 4.4 以降でのハイブリッド アプリのテストをサポートします。Android でハイブリッド アプリをテストするには、Android システムの WebView バージョン 51 以降が必要です。

Android 上のハイブリッド アプリをテストする手順は、モバイル ネイティブ アプリケーションをテストする手順と同じです。詳細については、「*Android 上のモバイル アプリケーションのテスト*」を参照してください。

USB ドライバのインストール

モバイル アプリケーションをテストするために、ローカル マシンに最初に Android デバイスに接続するには、適切な USB ドライバをインストールする必要があります。

デバイスの製造元は、そのデバイスに必要なすべてのドライバをもった実行可能ファイルを提供している可能性があります。この場合、ローカル マシンにその実行可能ファイルをインストールするだけです。製造元がこのような実行可能ファイルを提供していない場合、マシン上にデバイスに対する単一の USB ドライバをインストールできます。

Android USB ドライバをインストールするには：

1. デバイス用の適切なドライバをダウンロードします。
   たとえば、Google Nexus デバイス用の USB ドライバを検索し、インストールする場合は、*http://developer.android.com/tools/extras/oem-usb.html* を参照します。
2. Android デバイスをローカル マシンの USB ポートに接続します。
3. デスクトップ、または **Windows Explorer** から、**コンピュータ** を右クリックし、**管理** を選択します。
4. 左側のペインで、**デバイス マネージャ** を選択します。
5. 右側のペインで、**その他のデバイス** を探して展開します。
6. デバイス名 (*Nexus 5x* など) を右クリックして、**ドライバ ソフトウェアの更新** を選択します。**ハードウェアの更新ウィザード** が開きます。
7. **コンピュータを参照してドライバ ソフトウェアを検索します** を選択して、**次へ** をクリックします。
8. **参照** をクリックし、USB ドライバをダウンロードしたフォルダに移動します。
9. USB ドライバを選択します。
10. **次へ** をクリックしてドライバをインストールします。

USB デバッグの有効化

Android Debug Bridge (adb) 上で Android デバイスと通信するために USB デバッグを有効化します。

1. Android デバイスで設定を開きます。
2. **開発者向けオプション** (Dev Settings) をタップします。
   開発者向けオプションは、デフォルトでは表示されません。開発者向けオプションがデバイスの設定メニューに含まれていない場合：
   a) 画面を下にスクロールさせて、デバイスが携帯電話の場合は **端末情報** を、タブレットの場合は **タブレット情報** をタップします。
   b) 再度画面を下にスクロールさせて **ビルド番号** を 7 回タップします。
3. **開発者向けオプション** ウィンドウで、**USB デバッグ** をオンにします。
4. デバイスの USB モードをデフォルトの設定である **メディア デバイス (MTP)** に設定します。
   詳細については、デバイスのドキュメントを参照してください。

Android デバイスの推奨設定

Silk4J を使用したテストを最適化するために、テストしたい Android デバイスで次の設定を行ってください。

- USB デバッグを Android デバイスで有効化します。詳細については、「*USB デバッグの有効化*」を参照してください。
- Android デバイスは、Open Agent を実行しているマシンに、メディア デバイスとして接続されている必要があります。Android デバイスの USB モードは、**メディア デバイス (MTP)** を設定します。
- Android デバイスまたはエミュレータの画面が、テスト中にロックされないようにしてください。マシンに接続中にデバイスがロックされないようにするには、**開発者向けオプション** を開きます。**スリープモードにしない** または **充電中に画面をスリープにしない** をチェックします。

Silk4J 用に Android エミュレータを設定する

注: Android エミュレータを使用する場合、Silk4J が使用している adb サーバー以外の adb サーバーが実行されてます。実行中の adb サーバーのバージョンが異なる場合、Open Agent とデバイスとの接続が不安定になったり、接続できない場合があります。このようなバージョンの不一致によるエラーを避けるには、環境変数 *SILK_ANDROID_HOME* に Android SDK ディレクトリへのパス（C:¥Users¥<ユーザー>¥AppData¥Local¥Android¥android-sdk など）を指定してください。Information Service が実行中の場合、Windows のサービス マネージャーを使用して、Silk Test Information Service を再起動して更新した環境変数を適用する必要があります。環境変数が設定されていない場合は、Silk4J は Silk4J に同梱されたバージョンの adb を使用します。

Silk4J を使用して Android エミュレータ上でモバイル アプリケーションをテストする場合、テスト用にエミュレータを設定する必要があります。

1. Android SDK の最新版をインストールします。
   Android SDK のインストールと設定についての詳細は、「*Get the Android SDK*」を参照してください。
2. Android Studio 2 をインストールします。

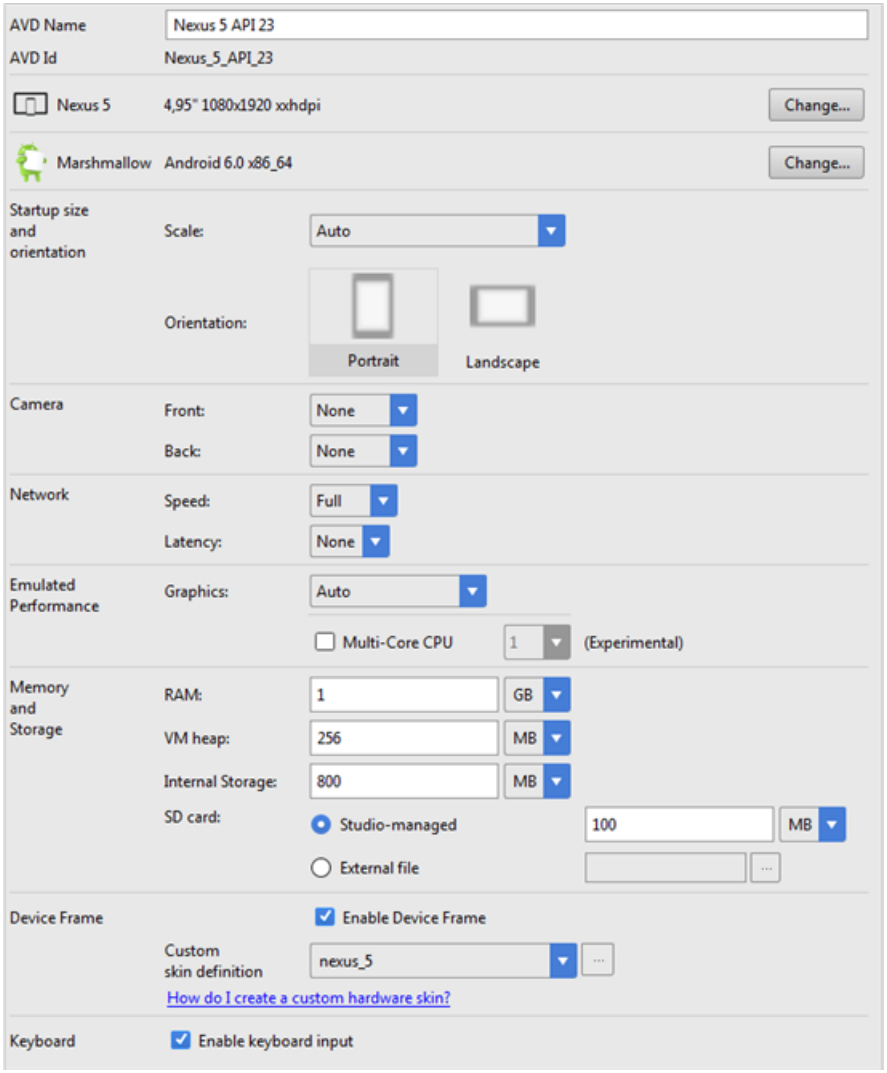
   ヒント: Android Studio 2 のインストールをスキップして、Android SDK で提供されるエミュレータを使用することもできます。ただし、エミュレータのパフォーマンスを向上させるため、Android Studio 2 をインストールすることを、Micro Focus では推奨しています。このトピックの以下の手順では、Android Studio 2 がインストールされていることを前提としています。
3. Android Studio 2 から **AVD Manager** を開きます。
4. **Create Virtual Device** をクリックします。
5. 仮想デバイスを選択します。
6. **Next** をクリックします。
7. Google API を含む Android のシステム イメージをダウンロードして選択します。
8. **Next** をクリックします。
9. 要件に従って仮想デバイスを設定します。

10 **Show Advanced Settings** をクリックします。

11 エミュレータが使用する RAM サイズとヒープ領域を対象のマシンで管理可能な量に調整します。

ヒント: 最低 1 GB の RAM と 256 MB のヒープ領域を使用することを、Micro Focus では推奨しています。

12 **Emulated Performance** 領域のリストで **Auto** を選択します。

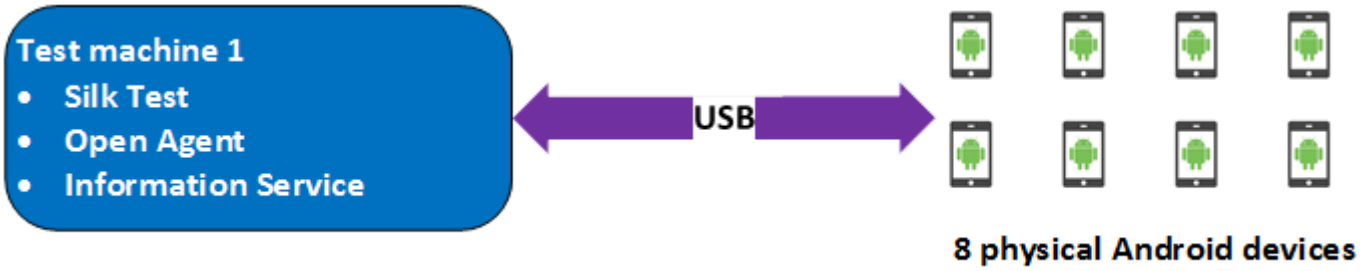


13 **Finish** をクリックします。

並列テスト実行のテスト済みの構成

Silk4J を使用して、複数の Android デバイス上で並列に自動テストを実行することができます。並列に実行できる Android デバイスの数は利用可能なハードウェアに依存します。 Micro Focus では、次のハードウェア構成でテストの実行を確認しています。

単一テスト マシンを使用した構成

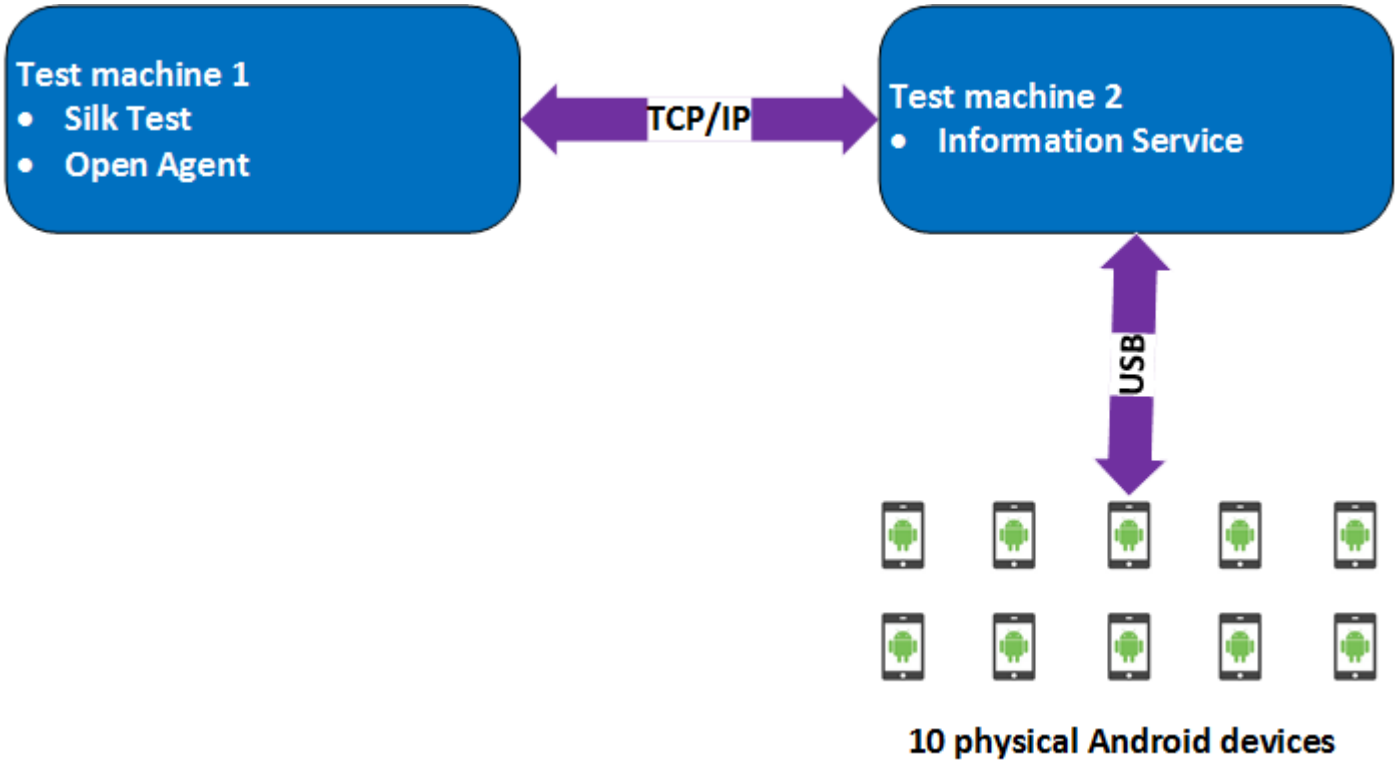


USB により Android デバイスを直接接続した単一のテスト マシンを使用して、最大 8 台の物理 Android デバイスまでの並列テストを行いました。

テスト マシンは、次のハードウェア仕様の Lenovo ThinkPad T450 です。

- Intel® Core™ i7 - 5600U CPU @ 2.60 GHz
- 2 コア (4 スレッド)
- 8 GB RAM

2 台のテスト マシンを使用した構成



また、2 台のテスト マシンを使用してテストを行いました。 1 台に Silk4J をインストールし、もう 1 台には Silk Test Information Service をインストールして最初のマシンのリモート ロケーションとして設定しました。このような構成を使用して、最大 10 台の物理 Android デバイスまでの並列テストを行いました。

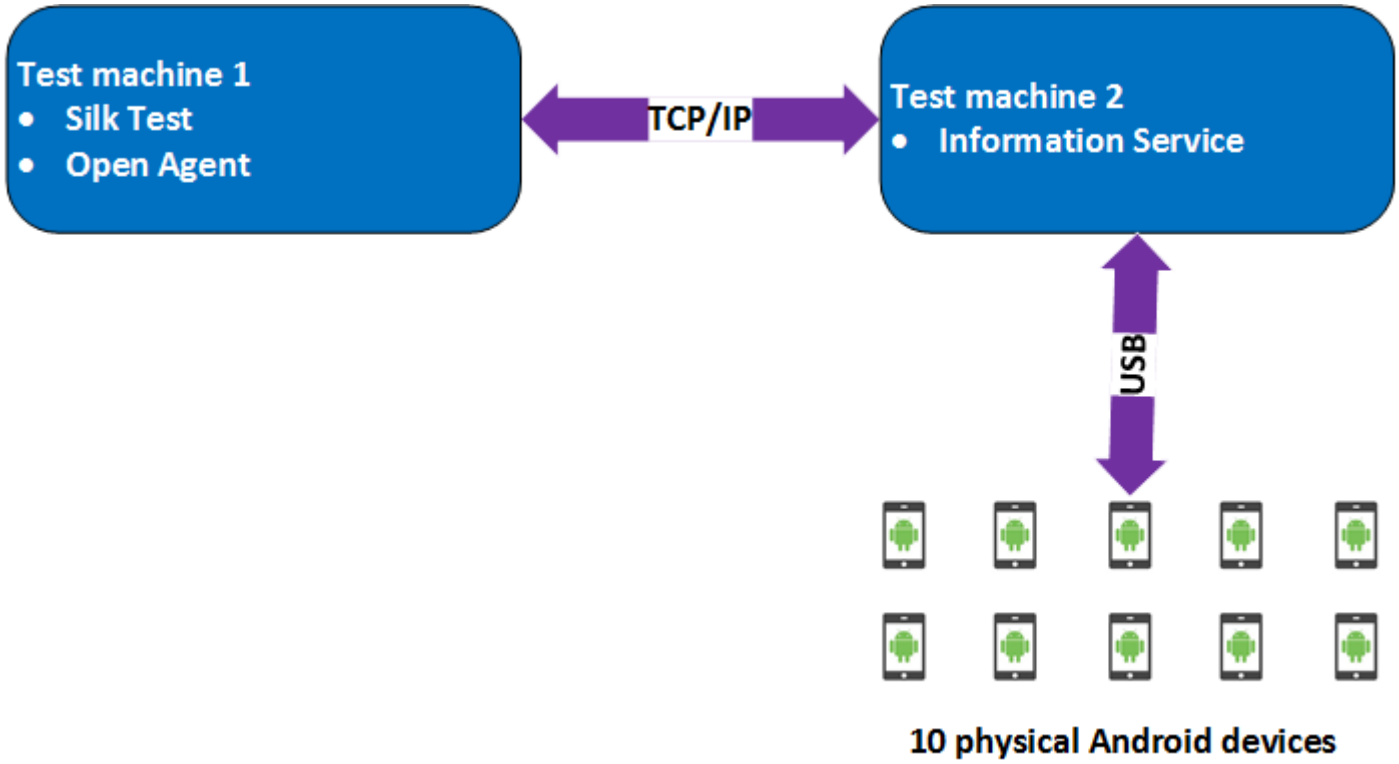
テスト マシン 1 は、次のハードウェア仕様の Lenovo ThinkPad T450 です。

- Intel® Core™ i7 - 5600U CPU @ 2.60 GHz
- 2 コア (4 スレッド)
- 8 GB RAM

テスト マシン 2 は、次のハードウェア仕様の Dell Precision T1700 です。

- Intel® Core™ i7 - 4770 CPU @ 3.40 GHz
- 4 コア (8 スレッド)
- 16 GB RAM

Windows マシンと Mac を使用した構成



さらに、2 台のテスト マシンを使用したテストで、Windows マシンに Silk4J をインストールし、Mac には Silk Test Information Service をインストールして最初のマシンのリモート ロケーションとして設定しました。このような構成を使用して、最大 10 台の物理 Android デバイスまでの並列テストを行いました。

テスト マシン 1 は、次のハードウェア仕様の Lenovo ThinkPad T450 です。

- Intel® Core™ i7 - 5600U CPU @ 2.60 GHz
- 2 コア (4 スレッド)
- 8 GB RAM

テスト マシン 2 は、次のハードウェア仕様の Apple Mac Mini です。

- Intel® Core™ i5 - 4782U CPU @ 2.60 GHz
- 2 コア (4 スレッド)
- 16 GB RAM

iOS

Silk4J では、iOS デバイスまたは iOS シミュレータ上のモバイル アプリケーションをテストすることができます。

iOS の以前のバージョンと比べて、iOS 9.3 に対して大幅な変更が Apple によってなされたため、Silk Test は iOS 9.3 以降のモバイル アプリケーションのテストをサポートします。サポートされている iOS のバージョンのリストについては、『リリース ノート』を参照してください。

注: iOS 11 上のモバイル アプリケーションのテストには Xcode 9 が必要です。Mac 上で Xcode 9 を使用している場合は、この Mac に接続 (または実行) されている iOS 11 より前のバージョンの iOS をインストールした物理デバイスおよびシミュレータ上でのテストはサポートされません。iOS 9.3 または iOS 10 の物理デバイスまたはシミュレータでテストする場合は、Xcode 8.3 を使用してください。

ヒント: iOS 9.3 より前のバージョンの iOS をテストする場合は、Silk Test 17.5 を使用してください。Silk Test 19.0 を使用した iOS 上でのテストについて、Silk Test 17.5 でのテストとの主な違いを以下の表に示します。

Silk Test19.0	Silk Test 17.5
iOS 9.3、iOS 10、iOS 11 をサポートします。	iOS 8.1、8.2、8.3、8.4、9.0、9.1、9.2、9.3 をサポートします。
Xcode 8.3、Xcode 9 をサポートします。	Xcode 6、Xcode 7 をサポートします。
同じユーザー セッションでの複数の物理 iOS デバイスのテストをサポートします。 **ヒント:** Silk Test 17.5 を使用してテストするために、複数のユーザー セッションを作成している場合、1 つ以外のすべてのユーザー セッションを削除することを Micro Focus は推奨します。	複数の物理 iOS デバイスをテストする場合は、Mac 上に複数のユーザー セッションが必要です。

iOS 上のモバイル アプリケーションのテストにおける前提条件

iOS デバイスや iOS シミュレータ上のモバイル アプリケーションをテストする前に、次の前提条件を満たしていることを確認してください。

- 現在のバージョンの Silk Test Information Service が Mac 上にインストールされている。詳細については、「*Silk Test Information Service を Mac にインストールする*」を参照してください。
- 物理 iOS デバイス上でアプリケーションをテストする場合は、以下の項目を確認してください。
  - デバイスが Mac に接続されている。
  - デバイスが iOS のサポートされているバージョンである。サポートされている iOS のバージョンのリストについては、『リリース ノート』を参照してください。
- iOS シミュレータ上でアプリケーションをテストする場合は、以下の項目を確認してください。
  - iOS シミュレータのイメージが Mac 上にインストールされている。
  - iOS シミュレータのイメージが iOS のサポートされているバージョンである。サポートされている iOS のバージョンのリストについては、『リリース ノート』を参照してください。
- 物理 iOS デバイス上でアプリケーションをテストする場合は、デバイスと Mac の両方ともタイム ゾーンが同じである。
- サポートするバージョンの Xcode が Mac 上にインストールされている。
- Windows マシン上に Silk4J がインストールされている。
- Mac が Windows マシンと同じネットワークにあり、Windows マシンにリモート ロケーションとして追加されている。
- iOS デバイス上でネイティブ モバイル アプリをテストする場合は、Developer Account でサインされたアプリの .ipa ファイルが作成されている。詳細については、「*iOS アプリのテストの準備*」を参照してください。
- iOS シミュレータ上でネイティブ モバイル アプリをテストする場合は、ZIP 形式に圧縮したアプリが作成されている。詳細については、「*iOS シミュレータ上のネイティブ モバイル アプリケーションのテスト*」を参照してください。
- iOS デバイスと iOS シミュレータの両方でネイティブ モバイル アプリをテストする場合は、サインした .ipa ファイルと ZIP した .app ディレクトリの両方が同じフォルダに作成されている。
- ネイティブ モバイル アプリをテストする場合は、iOS デバイスの ID がアプリをサインするのに使用した Developer Profile に関連付けられている。

- iOS デバイスのテスト中に、スリープ モードに移行してはいけません。画面ロックとパスワードをオフにするには、**設定** > **一般** > **パスコードロック** を選択します。
- テスト中に Mac の画面がオフにならないようにしてください。オフになると、**再生ステータス** ダイアログ ボックスに何も表示されなくなります。
- iOS シミュレータ上でモバイル アプリケーションをテストする場合は、テスト中に Mac のディスプレイ スリープが無効化されている。
- 物理 iOS デバイス上のネイティブ モバイル アプリをテストする場合は、デバイスで UI オートメーションが有効化されている。詳細については、「*iOS デバイスのテストの準備*」を参照してください。
- 物理 iOS デバイス上で Apple Safari を使用してのモバイル Web アプリケーションをテストする場合は、**Web インスペクタ** が有効化されている。詳細については、「*iOS デバイスのテストの準備*」を参照してください。
- Lightning コネクタを持つ iOS デバイスを使用することを Micro Focus は推奨しています。これは、Silk4J では、Lightning ケーブルで Mac に接続していない iOS デバイスのデバイス画面のライブ ビュー表示をサポートしていないためです。

物理 iOS デバイス上のネイティブ モバイル アプリケーションのテスト

注: Silk4J を使用してネイティブ モバイル アプリケーションやハイブリッド アプリケーションをテストするには、ネイティブ モバイル ライセンスが必要です。詳細については、「ライセンス情報」を参照してください。

iOS 上のモバイル アプリケーションのテストにおける前提条件については、「*iOS 上のモバイル アプリケーションのテストにおける前提条件*」を参照してください。ネイティブ モバイル アプリケーションをテストする際の既知の制限については、「モバイル ネイティブ アプリケーションのテストにおける制限事項」を参照してください。

物理 iOS デバイス上のネイティブ モバイル アプリケーション (アプリ) やハイブリッド アプリケーションをテストするには、次のタスクを実行します。

1. テストする iOS デバイスを準備します。
   詳細については、「*iOS デバイスのテストの準備*」を参照してください。
2. テストするアプリを準備します。
   詳細については、「*iOS アプリのテストの準備*」を参照してください。
3. テストする Mac を準備します。詳細については、「*iOS 上でのモバイル アプリケーションのテストのための Mac の準備*」を参照してください。
4. iOS デバイスが接続されている Mac を、Silk Test がインストールされている Windows マシンに、リモート ロケーションとして追加します。
   詳細については、「リモート ロケーションの編集」を参照してください。

   注: 任意の時点で、Mac に接続されている複数の物理 iOS デバイス上でテストできますが、iOS シミュレータは Mac 上で実行している 1 つに対してのみテストできます。Silk Test 17.5 Hotfix 1 以降を使用した場合、iOS 上のモバイル アプリケーションをテストするために、Mac 上で複数のユーザー セッションを使用する必要はありません。
5. モバイル アプリケーション用の Silk4J プロジェクトを作成します。
6. モバイル アプリケーション用のテストを作成します。
7. テストで実行する操作を記録します。**記録** ウィンドウを開始すると、**アプリケーションの選択** ダイアログ ボックスが開きます。
8. **モバイル** タブを選択します。
9. アプリをテストするモバイル デバイスをリストから選択します。
10. **参照** をクリックしてアプリ ファイルを選択するか、アプリ ファイルへの完全パスを **モバイル アプリ ファイル** テキスト フィールドに入力します。
    このパスでは、Silk4J は HTTP および UNC 形式をサポートします。
    Silk4J は、モバイル デバイス上に指定したアプリをインストールします。

11. **終了** をクリックします。

    iOS デバイスやシミュレータのテスト中に、スリープ モードに移行してはいけません。画面ロックとパスワードをオフにするには、**設定** > **一般** > **パスコードロック** を選択します。

12. すべての操作の記録を終えたら、記録を停止します。
13. テストを再生します。
14. テスト結果を分析します。

注: iOS デバイスと iOS シミュレータの両方でネイティブ モバイル アプリをテストする場合は、サインした .ipa ファイルと ZIP した .app ディレクトリの両方が同じフォルダに作成されている。

iOS シミュレータ上のネイティブ モバイル アプリケーションのテスト

注: Silk4J を使用してネイティブ モバイル アプリケーションやハイブリッド アプリケーションをテストするには、ネイティブ モバイル ライセンスが必要です。詳細については、「ライセンス情報」を参照してください。

iOS 上のモバイル アプリケーションのテストにおける前提条件については、「*iOS 上のモバイル アプリケーションのテストにおける前提条件*」を参照してください。ネイティブ モバイル アプリケーションをテストする際の既知の制限については、「モバイル ネイティブ アプリケーションのテストにおける制限事項」を参照してください。

iOS シミュレータ上のネイティブ モバイル アプリケーション (アプリ) やハイブリッド アプリケーションをテストするには、次のタスクを実行します。

1. テストする Mac を準備します。詳細については、「*iOS 上でのモバイル アプリケーションのテストのための Mac の準備*」を参照してください。
2. アプリの Xcode プロジェクトで、iOS シミュレータ用にアプリをコンパイルします。

   Xcode UI からでも、コマンドラインからでもアプリをコンパイルできます。たとえば、iOS 10.0 で iOS シミュレータ用のアプリをコマンドラインでコンパイルするには、次のコマンドを実行します。

   ```
   xcodebuild  -sdk  iphonesimulator10.0
   ```

3. アプリの .app ディレクトリを .zip ファイルに Zip します。

   デフォルトでは、.app ディレクトリは、~/Library/Developer/Xcode/DerivedData ディレクトリにあります。Xcode で **File** > **Project Settings** をクリックすれば、ディレクトリがある場所を確認できます。
4. iOS シミュレータがインストールされている Mac を、Silk4J がインストールされている Windows マシンに、リモート ロケーションとして追加します。

   詳細については、「リモート ロケーションの編集」を参照してください。

   注: Mac にインストールされている 1 つの iOS シミュレータでのみテストを実行できます。複数の Silk4J ユーザーが、同じ Mac にインストールされている複数の iOS シミュレータ上で同時にテストを実行することはできません。
5. モバイル アプリケーション用の Silk4J プロジェクトを作成します。
6. モバイル アプリケーション用のテストを作成します。
7. テストで実行する操作を記録します。**記録** ウィンドウを開始すると、**アプリケーションの選択** ダイアログ ボックスが開きます。
8. **モバイル** タブを選択します。
9. リストから iOS シミュレータを選択します。
10. **参照** をクリックして Zip したアプリ ファイルを選択するか、Zip したアプリ ファイルへの完全パスを **モバイル アプリ ファイル** テキスト フィールドに入力します。

    このパスでは、Silk4J は HTTP および UNC 形式をサポートします。

    Silk4J は、iOS シミュレータ上に指定したアプリをインストールします。
11. **終了** をクリックします。

iOS デバイスやシミュレータのテスト中に、スリープ モードに移行してはいけません。画面ロックとパスワードをオフにするには、**設定** > **一般** > **パスコードロック** を選択します。

12. すべての操作の記録を終えたら、記録を停止します。
13. テストを再生します。
14. テスト結果を分析します。

注: iOS デバイスと iOS シミュレータの両方でネイティブ モバイル アプリをテストする場合は、サインした .ipa ファイルと ZIP した .app ディレクトリの両方が同じフォルダに作成されている。

物理 iOS デバイス上のモバイル Web アプリケーションのテスト

iOS 上のモバイル アプリケーションのテストにおける前提条件については、「*iOS 上のモバイル アプリケーションのテストにおける前提条件*」を参照してください。モバイル Web アプリケーションをテストする際の既知の制限については、「*モバイル Web アプリケーションのテストにおける制限事項*」を参照してください。

物理 iOS デバイス上のモバイル Web アプリケーションをテストするには、次のタスクを実行します。

1. テストする iOS デバイスを準備します。
   詳細については、「*iOS デバイスのテストの準備*」を参照してください。
2. テストする Mac を準備します。詳細については、「*iOS 上でのモバイル アプリケーションのテストのための Mac の準備*」を参照してください。
3. iOS デバイスが接続されている Mac を、Silk Test がインストールされている Windows マシンに、リモート ロケーションとして追加します。
   詳細については、「リモート ロケーションの編集」を参照してください。

   注: 任意の時点で、Mac に接続されている複数の物理 iOS デバイス上でテストできますが、iOS シミュレータは Mac 上で実行している 1 つに対してのみテストできます。Silk Test 17.5 Hotfix 1 以降を使用した場合、iOS 上のモバイル アプリケーションをテストするために、Mac 上で複数のユーザー セッションを使用する必要はありません。
4. モバイル アプリケーション用の Silk4J プロジェクトを作成します。
5. モバイル アプリケーション用のテストを作成します。
6. テストで実行する操作を記録します。**記録** ウィンドウを開始すると、**アプリケーションの選択** ダイアログ ボックスが開きます。
7. モバイル Web アプリケーションをテストするには：
   a) **Web** タブを選択します。
   b) 使用するモバイル ブラウザーを選択します。
   c) **移動する URL の入力** テキスト ボックスに、開く Web ページを指定します。
8. **終了** をクリックします。
   iOS デバイスやシミュレータのテスト中に、スリープ モードに移行してはいけません。画面ロックとパスワードをオフにするには、**設定** > **一般** > **パスコードロック** を選択します。
9. すべての操作の記録を終えたら、記録を停止します。
10. テストを再生します。
11. テスト結果を分析します。

iOS シミュレータ上のモバイル Web アプリケーションのテスト

モバイル Web アプリケーションをテストする際の既知の制限については、「*モバイル Web アプリケーションのテストにおける制限事項*」を参照してください。

iOS シミュレータ上のモバイル Web アプリケーションをテストするには、次のタスクを実行します。

1. テストする Mac を準備します。詳細については、「*iOS 上でのモバイル アプリケーションのテストのための Mac の準備*」を参照してください。

2. iOS シミュレータがインストールされている Mac を、Silk Test がインストールされている Windows マシンに、リモート ロケーションとして追加します。

   詳細については、「リモート ロケーションの編集」を参照してください。

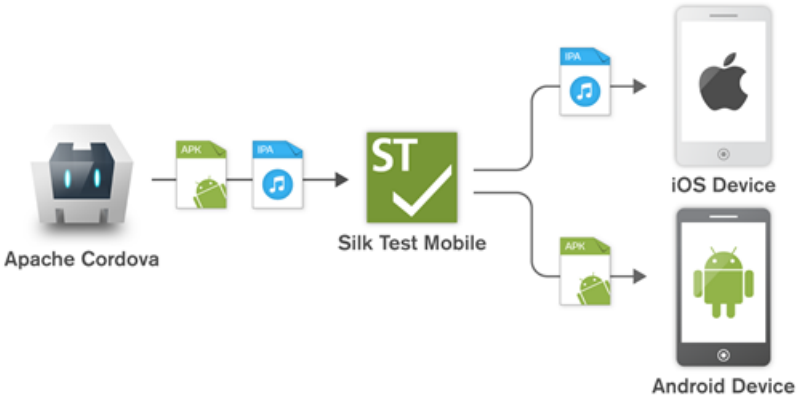
   注: 任意の時点で、Mac に接続されている複数の物理 iOS デバイス上でテストできますが、iOS シミュレータは Mac 上で実行している 1 つに対してのみテストできます。Silk Test 17.5 Hotfix 1 以降を使用した場合、iOS 上のモバイル アプリケーションをテストするために、Mac 上で複数のユーザー セッションを使用する必要はありません。
3. モバイル アプリケーション用の Silk4J プロジェクトを作成します。
4. モバイル アプリケーション用のテストを作成します。
5. テストで実行する操作を記録します。**記録** ウィンドウを開始すると、**アプリケーションの選択** ダイアログ ボックスが開きます。
6. モバイル Web アプリケーションをテストするには：
   a) **Web** タブを選択します。
   b) 使用するモバイル ブラウザーを選択します。
   c) **移動する URL の入力** テキスト ボックスに、開く Web ページを指定します。
7. **終了** をクリックします。

   iOS デバイスやシミュレータのテスト中に、スリープ モードに移行してはいけません。画面ロックとパスワードをオフにするには、**設定** > **一般** > **パスコードロック** を選択します。
8. すべての操作の記録を終えたら、記録を停止します。
9. テストを再生します。
10. テスト結果を分析します。

iOS 上のハイブリッド アプリケーションのテスト

ハイブリッド アプリケーション（アプリ）は、デバイス上で実行されるネイティブ アプリケーションのようなアプリですが、HTML5、CSS、JavaScript などの Web テクノロジを使用して記述されたアプリです。

Silk4J は、ネイティブ コンテナに埋め込まれた単一の Web ビューで構成されたデバッグ ハイブリッド アプリのテストにする完全なブラウザー サポートを提供します。このようなハイブリッド アプリの一般的な例は、Apache Cordova アプリケーションです。



リモート デバッグを有効化しない非デバッグ ハイブリッド アプリや、複数の Web ビューを含んだハイブリッド アプリをテストするには、Silk4J フォールバック サポートを有効化するために、オプション OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT を TRUE に設定します。詳細については、「詳細オプションの設定」を参照してください。フォールバック サポートを有効化すると、Silk4J は Web ビューのコントロールをブラウザー コントロールではなく、ネイティブ モバイル コントロールとして解決

して処理します。たとえば、以下のコードは、ブラウザー サポートを使用したときのリンクのクリックです。

```
desktop.setOption(CommonOptions.OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT,
false);
desktop.<DomLink>  find("//INPUT[@id='email']").click();
```

フォールバック サポートを有効化すると、同じリンクをクリックするコードは次のようになります。

```
desktop.setOption(CommonOptions.OPT_ENABLE_MOBILE_WEBVIEW_FALLBACK_SUPPORT,
true);
desktop.<DomLink>  find("//MobileTextField[@resource-id='email']").click();
```

iOS 上のハイブリッド アプリをテストする手順は、モバイル ネイティブ アプリケーションをテストする手順と同じです。詳細については、「物理 *iOS* デバイス上のネイティブ モバイル アプリケーションのテスト」または「*iOS* シミュレータ上のネイティブ モバイル アプリケーションのテスト」を参照してください。

iOS デバイス上のハイブリッド アプリをテストする前に、デバイスで **Web インスペクタ** が有効化されていることを確認してください。

iOS デバイスのテストの準備

注: Silk4J を使用してネイティブ モバイル アプリケーションやハイブリッド アプリケーションをテストするには、ネイティブ モバイル ライセンスが必要です。詳細については、「ライセンス情報」を参照してください。

モバイル アプリケーションをテストするために iOS デバイスを準備するには :

1. Mac 上で Xcode を起動します。
2. iOS デバイスを Mac に接続します。
3. iOS デバイスで、**設定** > **デベロッパ** をクリックします。

   ヒント: デベロッパ メニューが iOS デバイスに表示されていない場合は、デバイスと Mac を再起動します。
4. **Enable UI Automation** をオンにします。
5. Apple Safari 上でモバイル Web アプリケーションをテストするには、**設定** > **Safari** > **詳細** をクリックします。
6. **Web インスペクタ** をオンにします。
7. iOS シミュレータ上でテストを行う場合は、**Rotate Device Automatically** をオンにします。

   この設定は、**Silk Test Configuration Assistant** を使うか、手動でオンにできます。Mac 上で **Configuration Assistant** を開くには、ステータス メニューの Silk Test アイコンをクリックして、**Configuration Assistant** を選択します。手動でオンにするには、以下の操作を実行します。

   a) Mac 上で iOS シミュレータを開始します。
   b) Xcode 9 以降を使用している場合は、**Hardware** メニューを開きます。
      以前のバージョンの Xcode を使用している場合は、**Debug** メニューを開きます。
   c) **Rotate Device Automatically** をオンにします。
8. Xcode 9 以降を使用して iOS シミュレータ上でテストを行う場合は、**Show Device Bezels** をオフにします。

   この設定は、**Silk Test Configuration Assistant** を使うか、手動でオフできます。Mac 上で **Configuration Assistant** を開くには、ステータス メニューの Silk Test アイコンをクリックして、**Configuration Assistant** を選択します。手動でオフにするには、以下の操作を実行します。

   a) Mac 上で iOS シミュレータを開始します。
   b) **Window** メニューを開きます。
   c) **Show Device Bezels** をオフにします。

iOS アプリのテストの準備

Silk4J を使用して特定の デバイス上で特定の iOS アプリをテストできるようにするには、次の項目を考慮する必要があります。

- 特定の デバイスに手動でインストールできる iOS アプリに対してのみテストを自動化できます。iOS アプリにサインできるようにするために、*Apple Developer Program* のメンバーシップに登録する必要があります。詳細については、「メンバーシップの選択」を参照してください。*Apple Developer Program* のメンバーシップに登録せずにテストを行う場合は、「パーソナル チーム プロファイルを使用した物理 *iOS* デバイス上でのテスト 」を参照してください。

  注: App Store で配布するように作成された アプリや、任意の iOS デバイスに手動でインストールできるアプリを自動的にテストできません。
- 特定の デバイスで アプリをインストールして実行する前に、iOS デバイスを Apple Developer アカウントを使用して登録する必要があります。
- デバイスに iOS アプリをインストールするには、Xcode を使用してアプリの IPA ファイルを作成する必要があります。特定の iOS デバイスでのテスト用に IPA ファイルを作成するには、Apple Developer Program のメンバーとして Xcode の *Archive* 機能を使用します。この機能には 2 つのオプションがあります。

  - *Apple Developer Enterprise Program* のメンバーの場合、**Save for Ad Hoc Deployment** オプションを使用できます。
  - Apple Developer Enterprise Program のメンバー以外の Apple Developer Program のメンバーの場合、**Save for Development Deployment** オプションを使用できます。

  詳細については、「*Exporting Your App for Testing (iOS, tvOS, watchOS)*」を参照してください。

シミュレータ上の特定の アプリを Silk4J を使用してテストできるようにするには、Xcode を使用して アプリの Zip ファイルを作成してから iOS シミュレータ上にインストールします。詳細については、Xcode のドキュメントを参照してください。

Silk Test Information Service を Mac にインストールする

注: Information Service を Mac にインストールするには、Mac の管理者権限が必要です。

Mac 上の Apple Safari や、Mac に接続されている iOS や Android デバイス上のモバイル アプリケーションに対するテストを作成して実行するには、Mac に Silk Test Information Service (Information Service) をインストールしてから、**リモート ロケーション** ダイアログ ボックスを使用して、Silk4J をインストールした Windows マシンと Mac を接続する必要があります。

Information Service を Mac にインストールするには：

1. Java JDK が Mac 上にインストールされていることを確認します。
2. iOS デバイス上でモバイル アプリケーションをテストする場合は、Xcode が Mac 上にインストールされていることを確認します。
3. Information Service セットアップ ファイル（SilkTestInformationService<バージョン>-<ビルド番号>.pkg にアクセスします。

   - Silk Test のインストール時に Information Service セットアップ ファイルをダウンロードした場合は、Silk Test インストール ディレクトリ（C:¥Program Files (x86)¥Silk¥SilkTest など）の macOS フォルダを開きます。
   - Silk Test のインストール時に Information Service セットアップ ファイルをダウンロードしなかった場合は、*Micro Focus SupportLine* からセットアップ ファイルをダウンロードできます。
4. SilkTestInformationService<バージョン>-<ビルド番号>.pkg ファイルを Mac にコピーします。
5. SilkTestInformationService<バージョン>-<ビルド番号>.pkg を実行して、Information Service をインストールします。

6. インストール ウィザードの指示に従います。
7. パスワードを尋ねられた場合、現在サインインしている Mac ユーザーのパスワードを入力します。
8. Apple Safari が開き、SafariDriver を信頼するかどうかを尋ねるメッセージ ボックスが表示されたら、**信頼** をクリックします。

注: リモート接続ではなく、直接 Mac にログインしている場合には、SafariDriver だけをインストールできます。

インストールを完了するために、現在の Mac ユーザーをログアウトします。Information Service が正しくインストールされていることを確認するには、Mac にログインし、画面の右上隅にある Silk Test アイコンをクリックして、利用可能なデバイスとブラウザーを表示させます。

ヒント: Silk Test アイコンが表示されない場合は、Mac を再起動してください。

iOS 上でのモバイル アプリケーションのテストのための Mac の準備

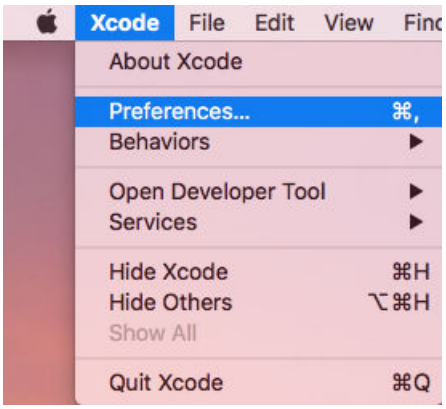
注: Silk4J を使用してネイティブ モバイル アプリケーションやハイブリッド アプリケーションをテストするには、ネイティブ モバイル ライセンスが必要です。詳細については、「ライセンス情報」を参照してください。

iOS 上でモバイル アプリケーションをテストするためには、iOS デバイスを接続した、または iOS シミュレータを実行している Mac が必要です。この Mac には Xcode がインストールされている必要があります。iOS 上のモバイル アプリケーションのテストにおける前提条件についての詳細は、「*iOS 上のモバイル アプリケーションのテストにおける前提条件*」を参照してください。

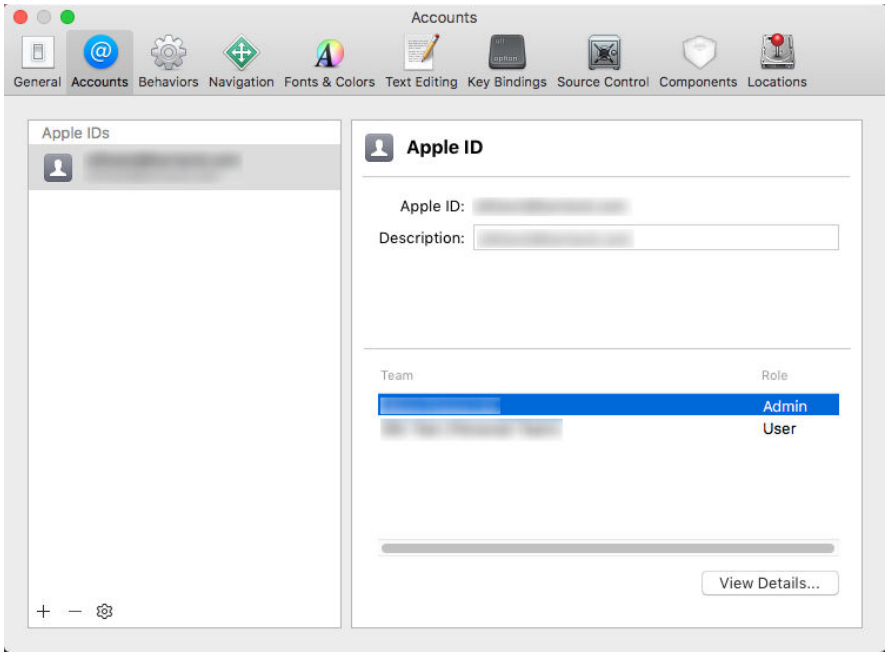
iOS テストを物理 iOS デバイス上で実行するには、**Silk Test Configuration Assistant** の指示に従って WebDriverAgentRunner Xcode プロジェクトを設定します。**Configuration Assistant** を開くには、ステータス メニューの Silk Test アイコンをクリックして、**Configuration Assistant** を選択します。

WebDriverAgentRunner Xcode プロジェクトを手動でビルドしたい場合には、次の手順を実行します。

1. Mac 上で Xcode を起動します。
2. **Xcode** > **Preferences** を選択します。



3. **Preferences** ウィンドウで、アカウントを選択します。
   a) **Accounts** タブを選択します。
   b) **Apple ID** を選択します。
   c) **Team** を選択します。
   d) **View Details** をクリックします。



4. **Apple Member Center** にアクセスし、開発チーム ID を取得します。
5. ターミナルで、~/.silk/silktest/conf/ に移動します。
6. xcconfig ファイル テンプレート silktest.xcconfig.sample を silktest.xcconfig という名前に変更します。
7. 開発チーム ID を silktest.xcconfig ファイルに追加します。

   ```
   DEVELOPMENT_TEAM = <your development team>
   ```
8. Mac 上のターミナルで次のコマンドを実行すると、WebDriverAgentRunner プロジェクトが正しく準備できたかどうかを確認できます。
   a) 物理 iOS デバイスの UDID（Unique Device ID）を見つけます。

   ```
   idevice_id -l
   ```
   b) WebDriverAgentRunner プロジェクトに移動します。

   ```
   cd /Application/Silk/Mobile/common/Appium/node_modules/appium-xcuitest-driver/WebDriverAgent
   ```
   c) WebDriverAgent がビルドできるかどうかをテストします。

   ```
   xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner –xcconfig ~/.silk/silktest/conf/silktest.xcconfig -destination 'id=<udid>' test
   ```

   <udid> を先ほど見つけた UDID で置き換えます。

   ヒント: xcodebuild コマンドが失敗する場合には、エラー メッセージの手順に従ってください。さらに、WebDriverAgentRunner プロジェクトで Preferences ウィンドウを開き、**General** タブの **Automatically manage signing** チェック ボックスがオフになっていることを確認します。
9. 省略可能：infoservice.properties ファイルには、Silk Test Information Service のポートや、Mac 上でのすべてのテストの実行で使用されるケイパビリティを指定することができます。
   詳細については、「*Silk Test Information Service プロパティの編集*」を参照してください。

パーソナル チーム プロファイルを使用した物理 iOS デバイス上でのテスト

Apple Developer Program のメンバーシップに登録していない場合は、パーソナル チーム プロファイルを使用して、物理 iOS デバイス上のアプリケーションをテストできます。

1. Mac 上で、/Application/Silk/Mobile/common/Appium/node_modules/appium-xcuitestdriver/WebDriverAgent に移動します。
2. WebDriverAgent.xcodeproj プロジェクトを Xcode で開きます。
3. **TARGETS** リストから、WebDriverAgentLib ターゲットを選択します。
   a) **General** タブをクリックします。
   b) **Automatically manage signing** を選択します。
   c) 開発チームを選択します。

   Signing Certificate は自動的に選択されます。
4. **TARGETS** リストから、WebDriverAgentRunner ターゲットを選択します。
   a) **General** タブをクリックします。
   b) **Automatically manage signing** を選択します。
   c) 開発チームを選択します。

   Signing Certificate は自動的に選択されます。
5. WebDriverAgentRunner ターゲットに対するプロビジョニング プロファイルの作成に Xcode が失敗する場合は、ターゲットのバンドル ID を手動で変更します。
   a) **Build Settings** タブをクリックします。
   b) **Product Bundle Identifier** の値を Xcode が受け付ける適切な値に変更します。

      たとえば、**Product Bundle Identifier** が *com.facebook.WebDriverAgentRunner* の場合、*io.appium.WebDriverAgentRunner* や *io.borland.WebDriverAgentRunner* などに変更します。
   c) **General** タブをクリックします。

   これでターゲットにプロビジョニング プロファイルが設定されました。
6. WebDriverAgent.xcodeproj プロジェクトを保存します。
7. すべてが期待通り動作することを確認するために、プロジェクトをビルドします。

   ```
   xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner -destination 'id=<udid>' test IPHONEOS_DEPLOYMENT_TARGET=10.3
   ```
8. WebDriverAgent アプリの再インストール時に問題が発生する場合は、同じプロビジョニング プロファイルを使ってデバイスに追加のアプリを永続的にインストールします。たとえば、WebDriverAgent Xcode プロジェクトの IntegrationApp をインストールします。
   a) **TARGETS** リストから、IntegrationApp ターゲットを選択します。
   b) **General** タブをクリックします。
   c) **Automatically manage signing** を選択します。
   d) 開発チームを選択します。
9. IntegrationApp ターゲットに対するプロビジョニング プロファイルの作成に Xcode が失敗する場合は、上記の WebDriverAgentRunner ターゲットに対する対処と同じ手順で、ターゲットのバンドル ID を手動で変更します。
10. IntegrationApp ターゲットの設定に成功したら、IntegrationApp を物理 iOS デバイスにインストールして実行します。
    a) ターゲットと iOS デバイスを選択します。
    b) **Play** をクリックします。

アプリのデバイスへのインストールが成功しても、次のようなエラー メッセージがコンソールや Appium ログ ファイルに出力される場合があります。
2017-01-24 09:02:18.358 xcodebuild[30385:339674] Error Domain=com.apple.platform.iphoneos Code=-12 "Unable to launch com.apple.test.WebDriverAgentRunner-Runner" UserInfo={NSLocalizedDescription=Unable to launch com.apple.test.WebDriverAgentRunner-Runner, NSUnderlyingError=0x7fa839cadc60 {Error Domain=DTXMessage Code=1 "(null)" UserInfo={DTXExceptionKey=The operation couldn't be completed. Unable to launch com.apple.test.WebDriverAgentRunner-Runner because it has an invalid code signature, inadequate entitlements or its profile has not been explicitly trusted by the user. : Failed to launch process with bundle identifier

'com.apple.test.WebDriverAgentRunner-Runner'}}} 2017-01-24 09:02:18.358 xcodebuild[30385:339674] Error Domain=IDETestOperationsObserverErrorDomain Code=5 "Early unexpected exit, operation never finished bootstrapping - no restart will be attempted" UserInfo={NSLocalizedDescription=Early unexpected exit, operation never finished bootstrapping - no restart will be attempted} Testing failed: Test target WebDriverAgentRunner encountered an error (Early unexpected exit, operation never finished bootstrapping - no restart will be attempted)
この問題は、開発者がそのデバイスで信頼されていないため発生します。そのデバイス上でアプリを手動で実行しようとすると、**信頼されていない開発元** メッセージが表示される場合があります。

デバイス上でこの問題を解決するには、**設定** > **一般** > **プロファイル** または **設定** > **一般** > **デバイス管理**（デバイスの種類と iOS のバージョンによって異なります）に移動します。そして、開発元を信頼し、アプリの実行を許可します。

Silk Test Information Service プロパティの編集

infoservice.properties ファイルを使用して、Silk Test Information Service のポートやさまざまなケイパビリティ（Capabilities）を指定できます。これらのケイパビリティは、Silk Test Information Service を実行しているマシン上で Silk Test がテストを実行するたびに毎回適用されます。

1. infoservice.properties.sample ファイルがあるディレクトリに移動します。
   たとえば、Mac の場合は ~/.silk/silktest/conf/ に移動します。Windows マシンの場合は C:¥ProgramData¥Silk¥SilkTest¥conf に移動します。
2. infoservice.properties.sample ファイルの名前を infoservice.properties に変更します。
3. 使用していないポートを指定します。
   Silk Test Information Service のデフォルトのポートは 22901 です。
4. ケイパビリティを指定するには、次の行を infoservice.properties ファイルに追加します。

```
customCapabilities=<custom  capability_1>;<custom_capability_2>;...
```

> **例：指定した言語での iOS シミュレータの実行**
>
> Mac 上で iOS シミュレータを常に同じ言語で実行するようにするには、カスタム ケイパビリティ *language* および *locale* を指定します。たとえば日本語の場合、次の行を infoservice.properties ファイルに追加します。
>
> ```
> customCapabilities=language=ja;locale=ja_JP
> ```

Silk Test Information Service を Mac からアンインストールする

Mac 上の Apple Safari に対するテストを実行する必要がなくなった場合など、次の手順で Silk Test Information Service を Mac からアンインストールすることができます。

1. uninstallInfoService.sh のような新しいシェル ファイルを作成します。
2. 新しいファイルに以下のコードを入力します。

```
#!/bin/sh

if launchctl list | grep com.borland.infoservice ; then
   launchctl unload /Library/LaunchAgents/com.borland.infoservice.plist
   echo "unloading Launch Daemon"
fi

if [ -d "/Applications/Silk" ]
then
   sudo rm -rf /Applications/Silk
fi

if [ -f "/Library/LaunchAgents/com.borland.infoservice.plist" ]
```

```
then
   sudo  rm  /Library/LaunchAgents/com.borland.infoservice.plist
fi

if  [  -f  "/usr/local/bin/ideviceinstaller"  ]
then
   sudo  rm  /usr/local/bin/ideviceinstaller
fi

exit  0
```

3. コマンドラインで chmod +x uninstallInfoService.sh を実行し、シェル ファイルの実行可能にします。
4. コマンドラインからシェル ファイルを実行します。

iOS デバイスの推奨設定

Silk4J を使用したテストを最適化するために、テストしたい iOS デバイスで次の設定を行ってください。

- 実際にユーザーが行った操作をテストに反映させるために、Apple Safari の自動入力とパスワードの保存を無効化します。**設定** > **Safari** > **パスワードと自動入力** をタップし、**ユーザ名とパスワード** 設定をオフにします。
- iOS デバイスのテスト中に、スリープ モードに移行してはいけません。画面ロックとパスワードをオフにするには、**設定** > **一般** > **パスコードロック** を選択します。

XCUITest を使用した iOS 上の既存のスクリプトの実行

注目: 以前のバージョンの Silk4J では、*Instruments* を使用して iOS デバイスを自動化していました。iOS 9.3 から、Instruments のサポートは *XCUITest* フレームワークのサポートに Apple によって置き換えられたため、Silk4J でも *Instruments* をサポートできなくなりました。この変更により、既存の iOS テスト スクリプトを Silk4J の現在のバージョンで実行できなくなる場合があります。

- XCUITest における classname 属性の動作は Instruments での動作と異なります。ほとんどの場合、Silk4J はこの変更に自動的に対応します。しかし、このような classname 属性が原因で既存のテスト スクリプトが動作しなくなった場合には、対応するオブジェクトの新しいロケーターを記録する必要があります。
- オブジェクトの階層が変更されました。

インストール済みアプリのテスト

デバイス、エミュレーター、またはシミュレータ上に既にインストールされているネイティブ モバイル アプリをテストするには、接続文字列でアプリを指定します。

1. ネイティブ モバイル アプリをテストする既存のプロジェクトを開きます。
2. **アプリケーション構成の編集** ダイアログ ボックスを開きます。
3. 次のいずれかの方法で、既存のアプリの指定部分を接続文字列で置き換えます。
   - iOS アプリをテストする場合は、*bundleId* を使ってアプリを指定します。たとえば、app=MyApp.ipa を bundleId=silktest.InsuranceMobile で置き換えます。
   - Android アプリをテストする場合は、*appActivity* と *appPackage* を使用してアプリを指定します。たとえば、app=MyApp.apk を appActivity=.LoginActivity;appPackage=silktest.insurancemobile で置き換えます。

詳細については、「接続文字列」を参照してください。

モバイル アプリケーションの記録

Silk4J とモバイル デバイスまたはエミュレータとの間の接続が一旦確立すると、デバイス上で実行する操作を記録できます。モバイル アプリケーションを記録するために、Silk4J では次の機能を持つ **記録** ウィンドウを使用します。

- テストするモバイル デバイスまたは Android エミュレータの画面を表示します。
- **記録** ウィンドウで操作を実行すると、モバイル デバイス上でも同じ操作が実行されます。
- 画面上のコントロールを操作すると、**記録** ウィンドウはデフォルトの操作を事前に選択します。
  - デフォルトの操作が Click の場合、コントロール上で左クリックすると、その操作が実行されます。右クリックすると、コントロールに対して利用可能な操作のリストが表示されます。この場合は、実行する操作を選択し、**OK** をクリックします。
  - デフォルトの操作が Click ではない場合、コントロールに対して有効なすべての操作がリストで表示されるので、実行したい操作を選択するか、単に **OK** をクリックして事前に選択された操作を受け入れます。

  リストから操作を選択する場合、選択した操作のパラメータの値をパラメータ フィールドに入力することができます。Silk4J は自動的にパラメータを検証します。
- 記録中、Silk4J は記録ウィンドウの隣にマウスの位置を表示します。その表示をクリックすると、デバイス画面に絶対的な位置とアクティブ オブジェクトに相対的な位置を切り替えることができます。
- 記録を一時停止すると、画面上での操作は記録されないため、デバイスを記録を続けたい状態に変更することができます。
- 記録を停止すると、記録した操作でスクリプトが生成されるため、続いてテストの再生を行うことができます。

テストを再生するモバイル デバイスの選択

テストを再生するために使用するモバイル デバイスを定義できます。

- Silk4J の UI からテストを実行する場合、**モバイル デバイスの選択** ダイアログ ボックスが表示され、このダイアログ ボックスで選択したモバイル デバイス、Android エミュレータ、iOS シミュレータが使用され、テスト スクリプトで設定されているモバイル デバイス Silk4J は無視します。
- **モバイル デバイスの選択** ダイアログ ボックスが無効の場合 (**再び表示しない** チェック ボックスをオンにした場合)、個々のテスト スクリプトのアプリケーション構成によって、テストを実行するために使用するモバイル デバイスが決定されます。

  注: モバイル デバイスの選択 ダイアログ ボックスを再び有効にするには、**Silk4J** > **アプリケーション構成の編集** をクリックして、**記録および再生前に 'モバイル デバイスの選択' ダイアログを表示する** チェックボックスをオンにします。
- スクリプトをコマンド ラインや CI サーバーから実行する場合は、スクリプトのアプリケーション構成で接続文字列を指定します。

  アプリケーション構成で指定したモバイル デバイスを上書きするには、*silktest.configurationName* 環境変数を使用します。
- Silk Central からテストを実行する場合、接続文字列を指定する代わりに、Silk Central の実行定義の **配置** タブにある **モバイル デバイスの選択** 領域でモバイル デバイスを指定します。詳細については、『*Silk Central ヘルプ*』を参照してください。

デバイス プールなどがある場合に、特定のモバイル デバイスを指定したり、利用可能なデバイス群のサブセットをフィルタするために接続文字列を使用できます。最初に一致したデバイスが再生に使用されます。特に指定がない場合には、次のルールに従って一致したモバイル デバイスが使用されます（高い優先度順）。

- リモート ロケーションに接続されたモバイル デバイスよりも、ローカル マシンに接続されたモバイル デバイスが優先されます。

- ブラウザーの種類が接続文字列で指定されている場合、古いバージョンのブラウザーよりも、新しいバージョンのブラウザーが優先されます。
- 古いプラットフォームよりも、新しいプラットフォームが優先されます。
- 物理デバイスがエミュレータやシミュレータよりも優先されます。
- アルファベット順で後者のデバイス名のデバイスが優先されます。たとえば、"iphone 5"という名前のデバイスよりも、"iphone 6"という名前のデバイスが優先されます。

例：リモート マシンに接続されている Android デバイス上のアプリに対する接続文字列

リモート マシンに接続されている Android デバイス上で MyApp.apk アプリをテストするには、接続文字列は次のようになります。

```
"platformName=Android;deviceName=MotoG3;host=http://
10.0.0.1;app=MyApp.apk"
```

例：Mac の iOS シミュレータ上のアプリに対する接続文字列

```
"platformName=iOS;platformVersion=10.0;deviceName=iPhone6;host=10.0.0.
1;app=MyApp.ipa;isSimulator=true"
```

Mobile Center デバイスの使用

Mobile Center は、モバイル デバイスのテストを管理するためのモバイル用ゲートウェイです。

Mobile Center によって管理されたデバイスに Silk4J からアクセスするには、次の手順を実行します。

1. Silk4J と Silk Central を統合します。
   詳細については、「*Silk4J と Silk Central の統合*」を参照してください。
2. Mobile Center を使用するように Silk Central を設定します。

   注: Mobile Center のインストール時に、適切なバージョンの Android SDK が使用されていることを確認してください。Silk4J で同じバージョンを使用するように設定するには、環境変数 SILK_ANDROID_HOME に C:¥Users¥<ユーザー>¥AppData¥Local¥Android¥android-sdk などを指定します。詳細については、『*Silk Central* ヘルプ』を参照してください。
3. iOS 上でテストする場合は、次の IPA ファイルがサインされていることを確認します。
   - HP4M-Agent.ipa
   - HPMC-AgentLauncher.ipa
   - WebDriverAgentRunner-Runner.ipa

   注: Silk4J は、Mobile Center を介した iOS シミュレータのテストをサポートしません。

以上により、**アプリケーションの選択** ダイアログに Mobile Center デバイスが表示されます。テストするデバイスを選択してください。

注: Silk4J と Mobile Center の両方で同時に同じモバイル デバイスをテストできません。Silk4J を使用してテストしたモバイル デバイスを、Mobile Center から引き続きテストする場合は、そのデバイスを再起動します。

注: Mobile Center 上で管理されているデバイスをテストする場合は、Silk4J は typeKeys メソッドや setText メソッドを使用して **Enter** などのキー コードの入力をサポートしません。さらに、Silk4J は iOS デバイス上の **ホーム** ボタンの押下をサポートしません。

注: Android エミュレータ上でテストする場合は、GPU HW アクセラレーションを無効にしてください。

Sauce Labs デバイスの使用

Sauce Labs は、自動テスト プラットフォームを提供しており、さまざまなモバイル デバイスやモバイル プラットフォームのバージョン上でテストを行うことができます。 自分自身のインフラストラクチャとしてデバイスを購入、保守する必要はありません。

Sauce Labs デバイスには Silk Central を介してアクセスすることができます。 次の手順に従って設定してください。

1. Silk4J が Silk Central と統合されていることを確認してください。
   詳細については、「*Silk4J と Silk Central の統合*」を参照してください。
2. Silk Central が Sauce Labs を使用するように設定されていることを確認してください。
   詳細については、*Silk Central ヘルプ* を参照してください。

以上により、**アプリケーションの選択** ダイアログに Sauce Labs デバイスが表示されます。テストするデバイスを選択してください。

モバイル デバイスの接続文字列

接続文字列 は、テストに使用するモバイル デバイスを指定します。モバイル テストを実行する場合、Silk4J は接続文字列を使用してモバイル デバイスに接続します。接続文字列は、アプリケーション構成の主要な一部です。テスト対象アプリケーションを構成するときに、接続文字列は設定されます。接続文字列を変更するには、**アプリケーション構成の編集** ダイアログ ボックスを使用します。

注: Silk Central からテストを実行する場合、接続文字列を指定する代わりに、Silk Central の実行定義の **配置** タブにある **モバイル デバイスの選択** 領域でモバイル デバイスを指定します。詳細については、『*Silk Central ヘルプ*』を参照してください。

デバイス プールなどがある場合に、特定のモバイル デバイスを指定したり、利用可能なデバイス群のサブセットをフィルタするために接続文字列を使用できます。最初に一致したデバイスが再生に使用されます。特に指定がない場合には、次のルールに従って一致したモバイル デバイスが使用されます（高い優先度順）。

- リモート ロケーションに接続されたモバイル デバイスよりも、ローカル マシンに接続されたモバイル デバイスが優先されます。
- ブラウザーの種類が接続文字列で指定されている場合、古いバージョンのブラウザーよりも、新しいバージョンのブラウザーが優先されます。
- 古いプラットフォームよりも、新しいプラットフォームが優先されます。
- 物理デバイスがエミュレータやシミュレータよりも優先されます。
- アルファベット順で後者のデバイス名のデバイスが優先されます。たとえば、"iphone 5"という名前のデバイスよりも、"iphone 6"という名前のデバイスが優先されます。

次のコンポーネントが接続文字列で使用できます。

コンポーネント	説明
deviceName	モバイル デバイスの名前。物理モバイル デバイス上でテストをする場合、デバイス ID を代わりに使用します。ワイルドカードをサポートします。大文字と小文字は区別されません。
platformName	Android または iOS。必須。
deviceId	省略可能：モバイル デバイスの ID。物理モバイル デバイスでテストをする場合に、デバイス名の代わりに使用します。ワイルドカードをサポートします。大文字と小文字は区別されません。
platformVersion	省略可能：Android または iOS のバージョン。特定の Android または iOS のバージョンのモバイル デバイス上でのみテストする場合に、バージョンを指定します。ワイルドカードをサポートします。大文字と小文字は区別されません。

コンポーネント	説明
browserVersion	省略可能：特定のブラウザーのバージョンでのみテストする場合に、ブラウザーの種類と共に使用します。ワイルドカードをサポートします。大文字と小文字は区別されません。
host	省略可能：設定しない場合は、任意のリモート ロケーションがホストとして使用されます。ワイルドカードをサポートします。大文字と小文字は区別されません。
• app • appActivity • appPackage	Android 上のネイティブ モバイル アプリケーションのテストでは必須です。アプリへのフルパス、または *appActivity* と *appPackage* の組み合わせで指定します。たとえば、app=MyApp.apk、appActivity=.LoginActivity;appPackage=silktest.insurancemobile などです。
• app • bundleId	iOS 上のネイティブ モバイル アプリケーションのテストでは必須です。アプリへのフルパス、または *bundleId* で指定します。たとえば、app=MyApp.ipa、bundleId=silktest.InsuranceMobile などです。
noReset	省略可能：ネイティブ モバイル アプリケーションをテストする場合に設定できます。*app* が指定されている場合にのみ有効です。テストの前にアプリを再インストールしない場合は True。テストの前にアプリを再インストールする場合は False を指定します。デフォルト値は、False です。
isSimulator	省略可能：iOS シミュレータ上でのみテストを実行する場合に指定します。デバイス名を代わりに使用できます。
isPhysicalDevice	省略可能：物理デバイス上でのみテストを実行する場合に指定します。デバイス名を代わりに使用できます。

デバイス プールを使用して、テストで実際に使用されるデバイスを確認するには、MobileDevice クラスの generateConnectionString メソッドの戻り値を使用できます。

モバイル デバイスまたは Android エミュレータ上のモバイル Web アプリケーションのテスト

モバイル デバイスまたは Android エミュレータ上でモバイル Web アプリケーションをテストする場合、接続文字列は次の要素から構成されます。

1. モバイル デバイス名（MotoG3 など）、またはデバイス ID（11111111 など）。

   注: デバイス ID は可読性に欠けるため、デバイス名がユニークであれば、デバイス名を接続文字列に使用することを Micro Focus では推奨します。
2. プラットフォーム名。
3. ブラウザーのバージョン。ブラウザーの種類の設定との組み合わせでのみ使用されます。
4. 特定のリモート マシンの IP アドレスまたはホスト名（10.0.0.1 など）。**リモート ロケーションの編集** ダイアログ ボックスで指定したリモート ロケーションの名前をホスト名として使用することもできます（*MyRemoteLocation* など）。リモート ロケーション名を使用する場合に、ワイルドカードを使用することもできます。ローカル マシンに接続されている Android デバイスをテストする場合は、ローカル マシンの IP アドレスまたはホスト名を指定します。

例：利用可能な任意の Android デバイスの接続文字列

```
"platformName=Android"
```

例：ローカル マシンに接続されている Android デバイス上のブラウザーに対する接続文字列

ローカル マシンに接続されている Android デバイス上でモバイル ブラウザーをテストするには、接続文字列は次のようになります。

```
"deviceName=MotoG3;platformName=Android;host=localhost"
```

または

```
"platformName=Android;deviceId=11111111;host=localhost"
```

例：リモート マシンに接続されている Android デバイス上のブラウザーに対する接続文字列

リモート Android デバイス上のモバイル ブラウザーをテストするには、接続文字列は次のようになります。

```
"deviceName=MotoG3;platformName=Android;host=10.0.0.1"
```

```
"deviceName=MotoG3;platformName=Android;host=MyRemoteLocation*"
```

例：Mac に接続されている iOS デバイス上のブラウザーに対する接続文字列

リモート iOS デバイス上のモバイル ブラウザーをテストするには、接続文字列は次のようになります。

```
"deviceName=myiPhone6;platformName=iOS;host=10.0.0.1"
```

モバイル デバイスまたは Android エミュレータ上のネイティブ モバイル アプリケーションのテスト

モバイル デバイスまたは Android エミュレータ上でネイティブ モバイル アプリケーションをテストする場合、接続文字列は次の要素から構成されます。

1. モバイル デバイス名（MotoG3 など）、またはデバイス ID（11111111 など）。

   注: デバイス ID は可読性に欠けるため、デバイス名がユニークであれば、デバイス名を接続文字列に使用することを Micro Focus では推奨します。
2. プラットフォーム名。
3. 特定のリモート マシンの IP アドレスまたはホスト名（10.0.0.1 など）。**リモート ロケーションの編集** ダイアログ ボックスで指定したリモート ロケーションの名前をホスト名として使用することもできます（*MyRemoteLocation* など）。リモート ロケーション名を使用する場合に、ワイルドカードを使用することもできます。ローカル マシンに接続されている Android デバイスをテストする場合は、ローカル マシンの IP アドレスまたはホスト名を指定します。
4. テストするアプリのファイルの名前または、ファイルが Web サーバー上にある場合には、ファイルの URL。たとえば、C:/MyApp.apk や MyApp.ipa など。

   - Android アプリは、常に .apk ファイルを指定します。
   - 物理デバイス上の iOS アプリは、常に .ipa ファイルを指定します。
   - シミュレータ上の iOS アプリは、ZIP ファイルまたは、*app* という名前のディレクトリを指定します。

例：リモート マシンに接続されている Android デバイス上のアプリに対する接続文字列

リモート マシンに接続されている Android デバイス上で MyApp.apk アプリをテストするには、接続文字列は次のようになります。

```
"platformName=Android;deviceName=MotoG3;host=http://
10.0.0.1;app=MyApp.apk"
```

例：Mac に接続されている iOS デバイス上のアプリに対する接続文字列

リモート マシンに接続されている iOS デバイス上で MyApp.ipa アプリをテストするには、接続文字列は次のようになります。

```
"platformName=iOS;deviceName=MyiPhone;host=http://
10.0.0.1;app=MyApp.ipa"
```

iOS シミュレータ上のモバイル Web アプリケーションのテスト

iOS シミュレータ上でモバイル Web アプリケーションをテストする場合、接続文字列は次の要素から構成されます。

1. プラットフォーム名（iOS）。
2. プラットフォームのバージョン（10.0 など）。
3. モバイル デバイス名（iPhone6 など）。
4. iOS シミュレータを実行している Mac の IP アドレスまたはホスト名。

例：Mac の iOS シミュレータ上のブラウザーに対する接続文字列

```
"platformName=iOS;platformVersion=10.0;deviceName=iPhone6;host=10.0.0.1;isSimulator=true"
```

iOS シミュレータ上のネイティブ モバイル アプリケーションのテスト

Mac の iOS シミュレータ上でネイティブ モバイル アプリケーションをテストする場合、接続文字列は次の要素から構成されます。

1. プラットフォーム名（iOS）。
2. プラットフォームのバージョン（10.0 など）。
3. モバイル デバイス名（iPhone6 など）。
4. リモート マシンの IP アドレスまたはホスト名（10.0.0.1 など）。
5. テストするアプリの名前（MyApp.ipa など）。

例：Mac の iOS シミュレータ上のアプリに対する接続文字列

```
"platformName=iOS;platformVersion=10.0;deviceName=iPhone6;host=10.0.0.1;app=MyApp.ipa;isSimulator=true"
```

モバイル デバイスの操作

モバイル デバイスを操作したり、テスト対象アプリケーションでスワイプのような操作を実行するには、次の手順を実行します。

1. **記録** ウィンドウで、**モバイル デバイス操作の表示** をクリックします。モバイル デバイスに対して実行できるすべての操作がリストされます。
2. リストからリストから実行したい操作を選択します。
3. Android デバイスまたはエミュレータで、スワイプを記録するには、マウスの左ボタンを押したままマウスを動かします。
4. テストの記録を続行します。

モバイル デバイスの開放

モバイル デバイスに対するテストを記録または再生する場合、Open Agent インスタンスはそのデバイスの所有権を確保します。これによって、Open Agent は、他の Silk Test ユーザーがそのデバイスを使用することを防止します。デバイス上でのテストの記録や再生が終了した後に、他の Silk Test ユーザーがデバイスを使用できるようにするために、Silk Test クライアントが閉じたとき、無人のテスト プロセスが完了したとき、または Open Agent が閉じたときに、Silk Test は自動的にデバイスを開放します。また、デバイスを手動で開放することもできます。

注: モバイル デバイスを開放すると、モバイル デバイス上のテスト対象アプリケーション (AUT) は閉じられます。

記録後のモバイル デバイスの開放

他の Silk Test ユーザーがモバイル デバイスでテストできるようにするために、記録後にデバイスを開放します。

記録が完了した後にモバイル デバイスを開放するには、次のいずれかを実行します。

- システム トレイから Open Agent を停止する。
- Silk4J を閉じる。並列テストが有効な場合には、この操作でのみデバイスが開放されます。

注: モバイル デバイスを開放すると、モバイル デバイス上のテスト対象アプリケーション (AUT) は閉じられます。

再生後のモバイル デバイスの開放

他の Silk Test ユーザーがモバイル デバイスでテストできるようにするために、再生後にデバイスを開放します。

再生が完了した後に手動でモバイル デバイスを開放するために、次のいずれかを実行することもできます。

- モバイル Web アプリケーションのテストを再生した場合、BrowserApplication クラスの close メソッド、または closeSynchron メソッド を使用します。これらのメソッド の詳細については、 API ドキュメントを参照してください。

  ```
  webBrowser.close();
  ```
- モバイル ネイティブ アプリケーションのテストを再生した場合、MobileDevice クラスの closeApp メソッドを使用します。

  たとえば、次のように入力します。

  ```
  MobileDevice mobileDevice = desktop.find("//MobileDevice");
  mobileDevice.closeApp();
  ```
- *desktop*.detachAll() ステートメントをテスト スクリプトに追加します。

次の条件を満たしている場合は、モバイル デバイスは自動的に開放されます。

- Open Agent を閉じる。
- 無人テスト中にテスト プロセスが停止する。並列テストが有効な場合には、この操作でのみデバイスが開放されます。
- Silk4J を閉じる。並列テストが有効な場合には、この操作でのみデバイスが開放されます。

注: モバイル デバイスを開放すると、モバイル デバイス上のテスト対象アプリケーション (AUT) は閉じられます。

モバイル アプリケーションのテスト時のトラブルシューティング

[アプリケーションの選択] ダイアログにモバイル デバイスが表示されない理由

Silk4J がモバイル デバイスやエミュレータを認識できないと、**アプリケーションの選択** ダイアログの **モバイル** タブにはデバイスやエミュレータが表示されません。さらに、**アプリケーションの選択** ダイアログの **Web** タブにも、そのデバイスやエミュレータ上にインストールされているモバイル ブラウザーが表示されません。

Silk4J が、次の何れかが原因でモバイル デバイスまたはエミュレータを認識していない可能性があります。

原因	解決策
エミュレータが実行されていない。	エミュレータを開始します。

原因	解決策
Android Debug Bridge (adb) がモバイル デバイスを認識しない。	モバイル デバイスが adb によって認識されているかどうか確認するには： 1. Android SDK をインストールしたフォルダで、Android Debug Bridge (adb) がある場所に移動します。Android SDK がインストールされていない場合、C:¥Program Files (x86)¥Silk¥SilkTest¥ng¥Mobile¥windows¥AndroidTools¥platform-tools に移動して、Silk4J がインストールした adb を使用します。 2. **Shift** を押しながら、**ファイル エクスプローラ** ウィンドウで右クリックします。 3. **コマンド ウィンドウをここで開く** を選択します。 4. コマンド ウィンドウで、adb devices を入力して、アタッチしたすべてのデバイスのリストを得ます。 5. デバイスがリストされない場合、USB デバッグがデバイスで有効化されていること、および適切な USB ドライバがインストールされていることを確認します。 6. 「adb server is out of date」のようなエラーが発生する場合は、C:¥Program Files (x86)¥Silk¥SilkTest¥ng¥Mobile¥windows¥AndroidTools¥platform-tools の adb のバージョンがローカルの Android SDK の adb のバージョンと一致していることを確認してください。詳細については、「*Open Agent* とデバイスとの接続が不安定な場合の対処方法」を参照してください。
デバイスのオペレーティング システムのバージョンを Silk4J がサポートしていない。	サポートするモバイル オペレーティング システムのバージョンについては、リリース ノート を参照してください。
デバイスの USB ドライバがローカル マシンにインストールされていない。	デバイスの USB ドライバをローカル マシンにインストールしてください。詳細については、「*USB* ドライバをインストールする」を参照してください。
USB デバッグがデバイスで有効化されていない。	USB デバッグをデバイスで有効化してください。詳細については、「*USB* デバッグの有効化」を参照してください。

注: どの解決策を適用しても解決できない場合は、デバイスを再起動してみてください。

URL に移動せずに Silk4J が Chrome for Android で URL を検索する理由

アドレス バーに入力された URL を、Chrome for Android が検索として解釈する場合があります。回避策として、URL に移動するコマンドをスクリプトに手動で追加できます。

adb サーバーが正しく起動しない場合にすべきこと

Android Debug Bridge (adb) サーバーが開始するとき、ローカル TCP ポート 5037 にバインドし、adb クライアントから送信されてくるコマンドをリッスンします。すべての adb クライアントは、ポート 5037 を使用して、adb サーバーと通信します。adb サーバーは、5555 から 5585 の範囲 (エミュレータやデバイスで使用される範囲) で奇数のポートをスキャンしてエミュレータやデバイス インスタンスを探します。adb はこれらのポートの変更を許しません。adb 開始中に問題が発生した場合、これらの範囲のポートの 1 つが、他のプログラムによって既に使用されているかどうか確認します。

詳細については、*http://developer.android.com/tools/help/adb.html* を参照してください。

Open Agent とデバイスとの接続が不安定な場合の対処方法

Android SDK、または Android Debug Bridge (adb) を使用するその他のツールをインストールしている場合、Silk4J が使用する　adb サーバー以外のサーバーが実行中の可能性があります。バージョンの異なる adb サーバーが実行中の場合、Open Agent とデバイスとの接続が不安定になったり、接続できない場合があります。

このようなバージョンの不一致によるエラーを避けるには、環境変数 *SILK_ANDROID_HOME* に Android SDK ディレクトリへのパス（C:¥Users¥<ユーザー>¥AppData¥Local¥Android¥android-sdk など）を指定してください。Information Service が実行中の場合、Windows のサービス マネージャーを使用して、Silk Test Information Service を再起動して更新した環境変数を適用する必要があります。環境変数が設定されていない場合は、Silk4J は Silk4J に同梱されたバージョンの adb を使用します。

エラー「メモリの割り当てに失敗しました：8」が発生する理由

エミュレータを開始しているときに、システムが十分なメモリを割り当てることができない場合に、このエラーが表示されます。以下を行ってみてください。

1. エミュレータのメモリ オプションの RAM サイズを下げる
2. Intel HAXM の RAM サイズを下げる RAM サイズを下げるには、IntelHaxm.exe を再度実行して、**Change** を選択します。
3. **タスク マネージャ** を開き、十分なフリー メモリが利用可能かどうかを確認します。不足している場合、プログラムを閉じてメモリを開放してください。

iOS デバイスのテスト時に「Silk Test は指定したアプリを開始できません」というエラーが発生する理由

このエラーが発生する原因として、以下の理由が考えられます。

原因	解決策
iOS デバイスがデベロッパ モードになっていない。	次の 2 種類の方法のいずれかで、デベロッパ モードを有効化できます。 • Xcode がインストールされている Mac にデバイスを接続し、テストするアプリをデバイスで開始します。 • プロビジョニング プロファイルをデバイスに追加します。 1. Xcode を開きます。 2. **Window** > **Devices** を選択します。 3. iOS デバイスを右クリックします。 4. **Show Provisioning Profiles** を選択します。 5. プロビジョニング プロファイルを追加します。
デバイスの iOS のバージョンを最近更新した。	1. Xcode を開きます。 2. **Window** > **Devices** を選択します。 3. Xcode がシンボル ファイルを処理するまで待機します。
UI オートメーションが iOS デバイスで有効化されていない。	1. **設定** > **デベロッパ** を選択します。 2. **Enable UI Automation** をオンにします。
Web インスペクタ が iOS デバイスで有効化されていない（モバイル Web アプリケーションのテストの場合）。	1. **設定** > **Safari** > **詳細** をクリックします。 2. **Web インスペクタ** をオンにします。

原因	解決策
テストするアプリがテストしようとしている iOS デバイスの iOS バージョン用にビルドされていない。	Xcode を使用してデバイスの iOS バージョン用にアプリをビルドします。
ソフトウェア・アップデート ダイアログ ボックスが iOS デバイス上で開いている。	ダイアログ ボックスを閉じ、ソフトウェアの自動アップデートを無効化します。 1. **設定** > **iTunes & App Store** > **自動ダウンロード** を選択します。 2. **アップデート** をオフにします。

Android デバイスの動的ハードウェア コントロールに戻るボタンだけが表示される理由

テストの開始時に Android デバイスや Android エミュレータの画面がロックされると、デバイスやエミュレータが動的ハードウェア コントロールに **戻る** ボタンだけを表示する場合があります。

この問題を解決するには、Open Agent を停止し、デバイスを再起動してから、デバイスの設定を画面のロックをしないように設定してください。

Android デバイスまたはエミュレータにキーボードが表示されなくなる理由

Unicode 文字列をサポートするために、Silk4J は標準キーボードをカスタム キーボードに置き換えます。そして、テストの完了時に元のキーボードに戻します。テスト中にエラーが発生すると、カスタム キーボードが設定されたまま、元に戻らない場合があります。

この問題を解決するには、**設定** > **言語と入力** > **現在のキーボード** を開き、手動で元のキーボードに戻してください。

テスト中にデバイスが応答しなくなる理由

テストの開始時に、デバイス、エミュレータ、シミュレータの画面がロックされると、Silk4J は画面のロックを解除できず、デバイス、エミュレータ、シミュレータが操作に応答しなくなる場合があります。

この問題を解決するには、Open Agent を停止し、デバイスの設定を画面のロックをしないように設定してください。

Information Service を Mac にインストールできない理由

システム環境設定の **セキュリティとプライバシー** で、**一般** タブの **ダウンロードしたアプリケーションの実行許可** 設定が **Mac App Store と確認済みの開発元からのアプリケーションを許可**（デフォルト値）に設定されている場合、Information Service セットアップを開いているときに次のエラー メッセージが表示されます。
"SilkTestInformationService<バージョン>.pkg" は、開発元が未確認のため開けません。

この問題を解決するには、次のいずれかを行います。

- セットアップ ファイルを右クリックして、**開く** を選択します。警告メッセージが表示されても、ファイルを開くことができます。
- **ダウンロードしたアプリケーションの実行許可** 設定を **すべてのアプリケーションを許可** に設定します。
- ファイルを開いた後、システム環境設定の **セキュリティとプライバシー** の **一般** タブを開き、**このまま開く** をクリックします。

Android アプリの記録時に記録ウィンドウが真っ暗になる理由

金融取引を処理するアプリなど、高レベルなセキュリティを必要とする Android アプリでは、Silk4J がアプリのキャプチャをできないようにするために、*FLAG_SECURE* フラグが設定されている可能性があります。Silk4J は、記録時に Android デバイスのスクリーンショットやビデオを利用しますが、テストする Android アプリにこのフラグが設定されていると、**記録中** ウィンドウにはデバイスの真っ黒な画面が表示

されます。Silk Test でこのようなアプリをテストするには、テスト中に FLAG_SECURE フラグを設定しないよう、アプリの開発チームに依頼してください。

Android エミュレータでのテスト時に Silk4J がビデオを表示しない理由

エミュレータがレンダリングにコンピュータのグラフィック カードを使用している場合、Silk4J のビデオキャプチャが機能しない場合があります。この問題を解決するには、ソフトウェアでグラフィックでエミュレートします。

1. **Android Virtual Device Manager** を開きます。
2. エミュレータの **Actions** 列の **Edit** をクリックします。
3. **Virtual Device Configuration** ダイアログ **Emulated Performance** 領域で、リストから **Software** を選択します。

クラウド環境でのテスト時に Silk4J がビデオを表示しない理由

クラウド環境でテストする場合、必要なポートがオープンになっていないことなどが原因で、テストの記録や再生時にビデオの表示が機能しない場合があります。

この問題を解決するには、infoservice.properties ファイルに WebDriver ホストの URL リストを指定します。このプロパティ ファイルについての詳細は、「*Silk Test Information Service* プロパティの編集」を参照してください。infoservice.disableScreencastHosts オプションをファイルに追加し、次のように入力します。

```
infoservice.disableScreencastHosts=<URL_1>,<URL_2>, ...
```

例：

```
infoservice.disableScreencastHosts=http://my-webdriver-server-url.com:80/wd/hub
```

ワイルドカードとしてアスタリスク（*）を使用して、**my-webdriver-server-url.com* のような URL パターンを指定することができます。

この設定により、指定したホストでの記録と再生時に、Silk4J はビデオの代わりに連続したスクリーンショットを表示するようになります。

Xcode のインストール バージョンを変更する方法

Xcode の最新版にアップグレードしてしまった場合など、使用している Xcode のバージョンを Silk4J がサポートしていない場合、iOS でテストする際にエラー メッセージが表示される場合があります。

インストールした Xcode のバージョンをサポートするバージョンで置換するには、サポートするバージョンの Xcode を *https://developer.apple.com/download/more/* からダウンロードし、サポートされていないバージョンをダウンロードしたバージョンで置換します。サポートする Xcode のバージョンについての情報は、『リリース ノート』を参照してください。

Mac のディスクの空き容量がなくなった場合の対処

Silk4J は iOS デバイスの自動化に Instruments を使用します。このツールは、/Library/Caches/com.apple.dt.instruments ディレクトリに大きなログ ファイルを生成するため、Mac のディスクの空き容量を圧迫している場合があります。この問題を解決するために、手動であるいは cron ジョブを使用して、これらのログ ファイルを定期的に削除することを Micro Focus は推奨します。たとえば、毎日同じ時間にファイルを削除するには、次の手順を実行します。

1. ターミナルで「sudo crontab -e」を入力します。crontab を root として編集できるエディタが開きます。
2. 次の行を crontab に追加します。

   ```
   0 2 1 * * find /Library/Caches/com.apple.dt.instruments -mtime +10 -delete
   ```
3. crontab を保存します。

この例では、10 日より前のすべてのログ ファイルが、毎日午前 2 時にディレクトリから削除されます。

「Unable to sign WebDriver Agent for testing」というエラー メッセージによりテストが失敗する理由

物理 iOS デバイス上でテストを行う場合、通常このエラーは WebDriverAgent アプリのビルド プロセス中に、プロビジョニング プロファイルでサインされていないか問題があったことを意味します。

デバイスが接続されている Mac マシンで次のコマンドを実行して、実際の問題を確認できます。

```
cd /Applications/Silk/Mobile/common/Appium/node_modules/appium-xcuitest-driver
xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner -destination 'id=<udid>' test
```

/Applications/Silk/Mobile/common/Appium/node_modules/appium-xcuitest-driver フォルダーにある Resources フォルダーが存在し、そのフォルダーに WebDriverAgent.bundle ファイルがあることを確認します。存在しない場合は、このフォルダーを作成し、空の WebDriverAgent.bundle ファイルを作成します。たとえば、次のコマンドを実行します。

```
mkdir -p Resources/WebDriverAgent.bundle
```

Developer Tools Access による他のプロセスの制御の要求を抑える方法

iOS 上でテストの実行を開始すると、つぎのメッセージようなメッセージ ボックスが表示される場合があります。
Developer Tools Access は、デバッグを続けるためにほかのプロセスを制御する必要があります。 これを許可するには、パスワードを入力してください。パスワードを入力して許可します。

このメッセージを抑制するには、ターミナルで次のコマンドを実行します。

```
sudo /usr/sbin/DevToolsSecurity --enable
```

iPad 上のモバイル Web アプリケーションのテスト時に矩形領域がずれる理由

iPad 上のモバイル Web アプリケーションのテスト時にコントロールを囲む矩形領域がずれている場合、複数のブラウザー タブが開いており、タブ バーが表示されている場合があります。この問題を回避するには、1 つを残して、ほかのすべてのタブを閉じてください。

Silk4J のアップデート後にデバイス上でテストの記録と再生が動作しない理由

Silk4J の新しいバージョンにアップデートすると、Silk4J の以前のバージョンでモバイル テスト用に使用してきた物理モバイル デバイス上の Appium アプリも自動的にアップデートされます。何らかの理由でこれらのアプリが自動的にアップデートされないと、そのデバイス上でテストの記録と再生が正常に動作しない場合があります。

Silk4J のアップデート後に、Android デバイスでこのような問題が発生した場合は、次のアプリをデバイスから手動でアンインストールしてください。

- Appium Android Input Manager
- Appium Settings
- io.appium.uiautomator2.server
- io.appium.uiautomator2.server.test
- Unlock

Silk4J のアップデート後に、iOS デバイスでこのような問題が発生した場合は、WebDriverAgentRunner をデバイスから手動でアンインストールしてください。

モバイルアプリケーションを記録できないのはなぜですか

Silk4J は Appium を使用してモバイルアプリケーションをテストします。 Appium で設定されている一部のネットワークプロキシ設定が、Silk4J の記録を妨げる可能性があります。モバイルデバイスまたはエミュレータでネットワークプロキシの設定を無効にすることができます。

テストの再生に Chrome for Android を使用する方法

デフォルトでは、**ブラウザーの選択** ダイアログ ボックスを使用して、再生に使用するブラウザーを選択できます。

スクリプトをコマンド ラインや CI サーバーから実行する場合は、スクリプトのアプリケーション構成で接続文字列を指定できます。アプリケーション構成で指定したブラウザーを上書きするには、*silktest.configurationName* 環境変数を使用します。

BrowserApplication クラスの browsertype プロパティを使用して、再生に使用するブラウザーの種類を設定することもできます。ただし、browsertype は Chrome for Android の明示的な値を含みません。

テストを再生するブラウザーとして Chrome for Android を使用するように指定するには、browsertype に *GoogleChrome* を設定し、Android をプラットフォームとして指定します。Android が指定されると、デスクトップ マシン上の Google Chrome の代わりに Chrome for Android を使用して、Silk4J はテストを実行します。

使用例

次のサンプル コードは、*silktest.configurationName* を使用して Nexus 7 上の Chrome for Android を使用するテストの基本状態を設定する方法を示しています。

```
SET
silktest.configurationName="platformName=Android;deviceName=Nexus
7;host=10.0.0.1  -  Chrome"
```

次の のサンプル コードは、browsertype を使用して Chrome for Android を使用するテストの基本状態を設定する方法を示しています。

```
BrowserBaseState  baseState  =  new
BrowserBaseState(BrowserType.GoogleChrome,  "demo.borland.com/
InsuranceWebExtJS/");
baseState.setConnectionString("platformName=Android");
baseState.execute(desktop);
```

モバイル Web アプリケーションのテストにおける制限事項

モバイル ブラウザ上でのテストの再生とロケーターの記録のサポートは、サポートされている他のブラウザほど完全なものではありません。モバイル Web アプリケーションに対するテストの再生とロケーターの記録の既知の制限事項を以下に示します。

- 次のクラス、インターフェイス、メソッド、プロパティは、モバイル Web アプリケーションでは現時点ではサポートされません。
  - BrowserApplication クラス。
    - closeOtherTabs メソッド
    - closeTab メソッド
    - existsTab メソッド
    - getActiveTab メソッド
    - getSelectedTab メソッド
    - getSelectedTabIndex メソッド
    - getSelectedTabName メソッド
    - getTabCount メソッド
    - imageClick メソッド
    - openTab メソッド

    - selectTab メソッド
  - DomElement クラス。
    - domDoubleClick メソッド
    - domMouseMove メソッド
    - getDomAttributeList メソッド
  - IKeyable インターフェイス。
    - pressKeys メソッド
    - releaseKeys メソッド
- Silk4J は、iOS 上の Apple Safari を使用した HTML フレームおよび iframe のテストをサポートしません。
- 横固定モードでの記録はシステム バーに仮想ボタンを含むエミュレータに対してサポートされません。このようなエミュレータは、回転を正しく検出せずに、横固定モードのシステム バーを画面の下部ではなく画面の右側に配置します。ただし、このようなエミュレータは縦固定モードで記録することができます。
- モバイル アプリケーションに対する XPath 式では、HTML DOM の HTML 属性だけがサポートされます。 Silk4J は、XPath 式のプロパティをサポートしません。
- Android 上でのモバイル Web アプリケーションのテストでは、Silk4J は、拡大縮小をサポートしません。
- BrowserWindow クラスの以下の JavaScript 警告処理メソッドが、Original Android Stock (AOSP) ブラウザー上でのテストでは機能しません。
  - acceptAlert メソッド
  - dismissAlert メソッド
  - getAlertText メソッド
  - isAlertPresent メソッド
- 任意の時点で、Mac に接続されている複数の物理 iOS デバイス上でテストできますが、iOS シミュレータは Mac 上で実行している 1 つに対してのみテストできます。
- モバイル Web アプリケーションのテストを開始する前に、ブラウザーのタブが開いていないことを確認してください。

  ヒント: iPad 上で Apple Safari のタブを無効に出来ます。**設定** > **Safari** を選択して、**タブバーを表示** をオフにすると無効になります。
- モバイル Web アプリケーションのテスト中に、ブラウザーのタブは 1 つだけ開くことができます。
- Silk4J は、ネイティブ モバイル アプリケーションによって開かれたモバイル Web アプリケーションのテストをサポートしません。

ネイティブ モバイル アプリケーションのテストにおける制限事項

ネイティブ モバイル アプリケーションに対するテストの再生とロケーターの記録の既知の制限事項は次の通りです。

- 次のクラス、インターフェイス、メソッド、プロパティは、ネイティブ モバイル アプリケーションでは現時点ではサポートされません。
  - IKeyable インターフェイス。
    - pressKeys メソッド
    - releaseKeys メソッド
  - MobileDevice クラス。
    - iOS 上でのネイティブ モバイル アプリケーションのテスト時に、setLocation メソッドはサポートされません。

    - Android 6.0 より前のバージョンの Android 上でのネイティブ モバイル アプリケーションのテスト時に、setLocation メソッドを使用する場合は、**擬似ロケーションを許可** を有効にする必要があります。設定は、Android デバイスまたはエミュレータの設定を開き、**開発者向けオプション** をタップします。
    - Android 6.0 以降上でのネイティブ モバイル アプリケーションのテスト時に、setLocation メソッドを使用する場合は、**Appium Settings** をアプリとして設定する必要があります。設定は、Android デバイスまたはエミュレータの設定を開き、**開発者向けオプション** > **仮の現在地情報アプリを選択** をタップします。そして、**Appium Settings** を選択します。

      注: Appium Settings という項目は、Android デバイスまたはエミュレータ上の Appium でテストを既に実行した場合にのみ表示されます。
    - iOS 上でのテスト時に、find メソッドが次の状況では機能しません。
        - path 属性が設定されている場合。
        - 属性値が空の場合。
- iOS 上でのテスト時に、XCUIElementTypeSwitch クラスの getValue メソッドがチェック状態に応じて、文字列 0、1 ではなく、文字列 false、true を返します。
- 横固定モードでの記録はシステム バーに仮想ボタンを持つ Android エミュレータではサポートされません。このようなエミュレータは、回転を正しく検出せずに、横固定モードのシステム バーを画面の下部ではなく画面の右側に配置します。ただし、このようなエミュレータは縦固定モードで記録することができます。
- モバイル アプリケーションに対する XPath 式では、HTML DOM の HTML 属性だけがサポートされます。Silk4J は、XPath 式のプロパティをサポートしません。
- 任意の時点で、Mac に接続されている複数の物理 iOS デバイス上でテストできますが、iOS シミュレータは Mac 上で実行している 1 つに対してのみテストできます。Silk Test 17.5 Hotfix 1 以降を使用した場合、iOS 上のモバイル アプリケーションをテストするために、Mac 上で複数のユーザー セッションを使用する必要はありません。
- Silk4J は、Android と iOS の両方とも、ネイティブ モバイル アプリケーションのテスト時にテキスト解決をサポートしません。

  テキスト解決は次のメソッドを含みます。

    - textCapture
    - textClick
    - textExists
    - textRectangle
- Silk4J は、複数の Web ビューを持つネイティブ モバイル アプリケーションのテストをサポートしません。
- iOS 上でのテスト時に、isVisible プロパティの状態は、要素が非表示であったとしても常に true になります。
- iOS 上でのテスト時に、複数のステップを持つスワイプ操作は、あるポイントへスワイプし、マウス ポインタをリリースした後、次のポイントへスワイプします。iOS の以前のバージョンでは、この操作はスワイプ間でマウス ポインタをリリースしません。
- iOS 上でのテスト時に、Silk4J はピンチ以外のマルチタッチ操作をサポートしません。
- iOS 上でのテスト時に、Silk4J は pinchIn メソッドをサポートしません。
- iOS 上でのテスト時に、警告ダイアログ ボックスに対して承認と解除のみを行えます。**キャンセル** ボタンは利用できないため、Silk4J はダイアログを解除できません。デフォルトの操作はダイアログの承認です。
- Android 上でのテスト時に、Silk4J は *Animation* クラスのコントロールに対する自動同期を行いません。
- Android 上でのトーストのテスト時には、次の制限事項があります。
    - 記録時に、トースト用の矩形領域は、トーストの実際の位置とは関係なく、Silk4J の **記録** ウィンドウの下部に表示されます。
    - 記録時と再生時に、トーストがすぐに表示された場合でも、Silk4J によるトーストの検出に、常に 5 秒間かかります。

- iOS 上でのテスト時に、Silk4J は UIView.animate 関数または UIView.animateWithDuration 関数を呼び出すコントロールに対する自動同期を行いません。

  アプリケーション デリゲートでアニメーションの速度を速くすることで、問題が回避できる可能性があります。

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
  //...
  if NSProcessInfo.processInfo().environment["automationName"] == "Silk Test" {
    // Speed animations up (recommended)
    window!.layer.speed = 100;
  }
}
```

  このようなアニメーションを完全に無効にすることは、アプリケーションの挙動が変わってしまう可能性があるため、Micro Focus では推奨していません。しかし、アニメーションの速度を速めても同期問題が解決できない場合は、次のようにしてアプリケーション デリゲートでアニメーションを完全に無効化することもできます。

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
  //...
  if NSProcessInfo.processInfo().environment["automationName"] == "Silk Test" {
    UIView.setAnimationsEnabled(false)
  }
}
```

- iOS 上でのテスト時には、さらに次の制限事項があります。
  - テストの記録および再生時にパフォーマンスの低下を感じる場合があります。
  - iOS の内部的な変更の影響で一部のコントロールのロケーターが変ったため、既存のテストが動作しない場合があります。
  - フォーカスを持たないテキスト フィールドが、テキスト フィールドとして解決されない場合があります。テキスト フィールドを正しく解決するために、操作する前にテキスト フィールドのクリックなどを実行して、テキスト フィールドにフォーカスを移してください。

モバイル Web サイトでのオブジェクトのクリック

自動テストの記録と再生中にオブジェクトをクリックするとき、モバイル Web サイトではデスクトップ Web サイトと比較して、次のような困難があります。

- 拡大/縮小率やデバイス ピクセル比が異なる
- さまざまなモバイル デバイスによって画面サイズが異なる
- モバイル デバイス間でのフォントとグラフィックサイズが異なる (通常、デスクトップ ブラウザの Web サイトよりも小さい)。
- さまざまなモバイルデバイスによってピクセル サイズと解像度が異なる

Silk4J は、このような困難をものともせずに、モバイル Web サイトの適切なオブジェクトをクリックできます。

モバイル デバイスでテストを記録するときに、Silk4J は Click の記録時に座標を記録しません。ただし、クロス ブラウザ テストの場合、再生中に座標が許されています。また、Click に座標を手動で追加することもできます。Silk4J は、これらの座標をオブジェクトの HTML 座標として解釈します。モバイル デバイスのテストの再生時に BrowserWindow の内側の適切なオブジェクトをクリックするために、Silk4J はオブジェクトの HTML 座標に現在の拡大/縮小率を適用します。デバイスのピクセル座標は、オブジェクトの HTML 座標に現在の拡大/縮小率をかけた座標です。

モバイル Web サイトの現在表示されている領域にオブジェクトが表示されていない場合、Silk4J は Web サイトの適切な位置にスクロールします。

例

HTML ページで 100 x 20 ピクセルの固定サイズの DomButton をテストするコードを以下に示します。

```
DomButton domButton = desktop.find("locator for the button");
domButton.click(MouseButton.LEFT, new Point(50, 10));
```

異なるモバイル デバイスまたは異なる拡大/縮小率で再生すると、たとえば DomButton は、デバイス画面上では実際は 10 ピクセルの幅かもしれません。しかし、現在の拡大/縮小率の影響は受けず、上記のコードを使用したときに Silk4J は要素の中央をクリックします。これは、Silk4J が座標を HTML 座標として解釈し、現在の拡大/縮小率を適用するためです。

既存のモバイル Web テストの使用方法

Silk Test 17.0 以降では、モバイル Web テストの扱いが、前のバージョンの Silk Test とは異なります。この変更により、以前のモバイル Web テストが Silk Test 17.0 以降では動作しなくなる可能性があります。このトピックでは、Silk Test 17.0 で行われた変更について説明し、既存のモバイル Web テストを Silk Test 17.0 以降で使用できるように変更する方法を説明します。

Silk Test 17.0 でモバイル Web テストに対して行われた変更は、以下の通りです。

- Silk Test の以前のバージョンでは、Windows マシンに USB で接続された iOS デバイスをテストすることができました。Silk Test 17.0 以降では、OS X マシン（Mac）に接続された iOS デバイスに対してのみテストすることができます。
- 以前のバージョンの Silk Test を使用して Android デバイス上のモバイル Web アプリケーションをテストしていた場合、Silk Test 17.0 以降で Web アプリケーションをテストするには、Android デバイスのプロキシ設定を手動で削除する必要があります。Silk Test 17.0 以降では、プロキシは使用しません。プロキシが設定されていると、「プロキシ サーバーに接続できません」というメッセージがデバイスに表示されます。

.NET のサポート

Silk Test は、以下の .NET アプリケーションのテストを組み込みでサポートしています。

- Windows Forms（Win Forms）アプリケーション
- Windows Presentation Foundation (WPF) アプリケーション
- Microsoft Silverlight アプリケーション

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

Windows Forms のサポート

Silk4J は、.NET スタンドアロン アプリケーションとノータッチ Windows Forms (Win Forms) アプリケーションのテストを組み込みでサポートしています。ただし、スタンドアロン アプリケーションでは、side-by-side 実行はサポートされていません。

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

オブジェクト解決

アプリケーション中の要素に指定された name が使用可能な場合、ロケーターの automationId 属性として使用されます。この結果、多くのオブジェクトは、この属性のみを使用して一意に識別できます。

サポートするコントロール

Win Forms テストで使用可能な記録／再生コントロールの完全な一覧については、「*Windows Forms クラス リファレンス*」を参照してください。

Windows Forms アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Windows Forms アプリケーションがサポートする属性は次のとおりです。

- automationid
- caption
- windowid
- priorlabel (caption のないコントロールの場合、自動的に priorlabel が caption として使用されます。caption のあるコントロールの場合、caption を使う方が簡単な場合があります。)

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Windows Forms アプリケーションのカスタム属性

Windows Forms アプリケーションは、あらかじめ定義された自動化用プロパティ automationId を使用して、Windows Forms コントロールに対して安定した識別子を指定します。

Silk4J は、ロケーターを識別するために、自動的にこのプロパティを使用します。Windows Forms アプリケーションのロケーターは次のようになります。

```
/FormsWindow//PushButton[@automationId='btnBasicControls']
```

Windows Forms メソッドの動的な呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティーの取得、またはプロパティーの設定を直接実行できます。また、このコントロールの Silk4J API で使用できないメソッドおよびプロパティーも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4J API を通して公開されていない場合に特に便利です。

オブジェクトの動的メソッドは invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

オブジェクトの複数の動的メソッドは invokeMethods メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

動的プロパティの取得には getProperty メソッドを、動的プロパティの設定には setProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、getPropertyList メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを String 型の入力パラメータとして設定する必要がある SetTitle というメソッドを呼び出すには、次のように入力します：

```
control.invoke("SetTitle","my  new  title");
```

注: 通常、ほとんどのプロパティは読み取り専用で、設定できません。

注: ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティーを取得する場合、Reflection を使用します。

invoke メソッド

Windows Forms または WPF コントロールでは、invoke メソッドを使用して、以下のメソッドを呼び出すことができます。

- MSDN が定義するコントロールのパブリック メソッド。
- MSDN が定義する静的パブリック メソッド。
- ユーザーが定義する任意の型の静的パブリック メソッド。

invoke メソッドの最初の例

Silk4J の DataGrid 型のオブジェクトでは、MSDN が System.Windows.Forms.DataGrid 型に定義しているすべてのメソッドを呼び出すことができます。

System.Windows.Forms.DataGrid クラスのメソッド IsExpanded を呼び出すには、次のコードを使用します。

```
//Java code
boolean isExpanded = (Boolean) dataGrid.invoke("IsExpanded", 3);
```

invoke メソッドの 2 番目の例

AUT 内の静的メソッド String.compare(String s1, String s2) を呼び出すには、次のコードを使用します。

```
//Java code
int result = (Integer) mainWindow.invoke("System.String.Compare", "a",
"b");
```

invoke メソッドの 3 番目の例

この例では、ユーザーが生成したメソッド GetContents を動的に呼び出す方法を示します。

テスト対象アプリケーション (AUT) のコントロールの操作に使用するコードを作成できます (この例では UltraGrid)。UltraGrid の内容を取得するために、複雑な動的呼び出しを作成するのではなく、新しいメソッド GetContents を生成し、この新しいメソッドを動的に呼び出すことができます。

Visual Studio で、AUT 内の次のコードによって GetContents メソッドを UltraGridUtil クラスのメソッドとして定義します。

```
//C# code, because this is code in the AUT
namespace UltraGridExtensions {
   public class UltraGridUtil {
      /// <summary>
      /// Retrieves the contents of an UltraGrid as nested list
      /// </summary>
      /// <param name="grid"></param>
      /// <returns></returns>
      public static List<List<string>>
GetContents(Infragistics.Win.UltraWinGrid.UltraGrid grid) {
         var result = new List<List<string>>();
         foreach (var row in grid.Rows) {
            var rowContent = new List<string>();
            foreach (var cell in row.Cells) {
```

```
                rowContent.Add(cell.Text);
            }
            result.Add(rowContent);
        }
        return  result;
    }
  }
}
```

UltraGridUtil クラスのコードを AUT に追加する必要があります。これは、次のようにして行います。

- アプリケーション開発者は、クラスのコードを AUT にコンパイルできます。アセンブリがすでにロードされている必要があります。
- テストの実行時に AUT にロードされる新しいアセンブリを作成できます。

アセンブリをロードするには、次のコードを使用します。

```
FormsWindow.LoadAssembly(String  assemblyFileName)
```

次のようにして、フルパスで指定してアセンブリをロードします。

```
mainWindow.LoadAssembly("C:/temp/ultraGridExtensions.dll")
```

UltraGridUtil クラスのコードが AUT 内にある場合は、次のコードをテスト スクリプトに追加して、GetContents メソッドを呼び出すことができます。

```
List<List<String>>  contents  =
mainWindow.invoke("UltraGridExtensions.UltraGridUtil.GetContents",
ultraGrid);
```

invoke メソッドを呼び出す mainWindow オブジェクトは、AUT を特定しているだけなので、同じ AUT の他のオブジェクトに置き換えてもかまいません。

invokeMethods メソッド

Windows Forms または WPF コントロールでは、invokeMethods メソッドを使用して、ネストされたメソッドのシーケンスを呼び出すことができます。以下のメソッドを呼び出すことができます。

- MSDN が定義するコントロールのパブリック メソッド。
- MSDN が定義する静的パブリック メソッド。
- ユーザーが定義する任意の型の静的パブリック メソッド。

例：カスタム データ グリッドのセルの内容のテキストでの取得

Infragistics ライブラリのカスタム データ グリッドのセルの内容をテキストで取得するには、AUT で次の C# コードを使用できます。

```
string  cellText  =  dataGrid.Rows[rowIndex].Cells[columnIndex].Text;
```

次の C# コードのサンプルは、最初の行の 3 番目のセルの内容をテキストで取得します。

```
string  cellText  =  dataGrid.Rows[0].Cells[2];
```

invokeMethods メソッドを使用して同じ例をスクリプト化すると、比較的複雑なスクリプトになります。これは、対応するパラメータを持つ 5 つのメソッドを invokeMethods メソッドに渡さなければならないためです。

```
WPFControl  dataGrid  =  mainWindow.find("//
WPFControl[@automationId='Custom  Data  Grid']");

// Get  text  contents  of  third  cell  in  first  row.
int  rowIndex  =  0;
```

```
int columnIndex = 2;

List<String> methodNames = Arrays.asList("Rows", "get_Item", "Cells",
"get_Item", "Text");
List<List<Object>> parameters = Arrays.asList(new ArrayList<Object>(),
Arrays.<Object>asList(rowIndex), new ArrayList<Object>(),
Arrays.<Object>asList(rowIndex), new ArrayList<Object>());

String cellText = (String) dataGrid.invokeMethods(methodNames,
parameters);
```

このような場合に、より簡単にするアプローチは、テスト対象アプリケーションにコードを追加して、invokeMethods メソッドを使用することです。たとえば、getCellText メソッドを AUT に追加します。

```
// C# code, if the AUT is implemented in C#.
public static string GetCellText(Infragistics.Win.UltraWinGrid.UltraGrid
dataGrid, int rowIndex, int columnIndex) {
   return dataGrid.Rows[rowIndex].Cells[columnIndex].Text;
```

```
' VB code, if the AUT is implemented in VB.
public static string GetCellText(Infragistics.Win.UltraWinGrid.UltraGrid
dataGrid, int rowIndex, int columnIndex) {
   return dataGrid.Rows[rowIndex].Cells[columnIndex].Text;
```

テスト スクリプトから GetCellText メソッドを動的に呼び出して、セルの内容をテキストで取得します。

```
String cellText = (String) mainWindow.invoke("GetCellText", dataGrid,
rowIndex, columnIndex);
```

詳細については、「テスト対象アプリケーションにコードを追加してカスタム コントロールをテストする」を参照してください。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4J がサポートするコントロールのメソッドとプロパティー。
- MSDN が定義するコントロールのパブリック メソッドとプロパティー。
- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティー。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- すべての組み込み Silk4J 型

  Silk4J 型には、プリミティブ型 (boolean、int、string など)、リスト、およびその他の型 (Point や Rect など) が含まれます。
- 列挙型

  列挙パラメータは文字列として渡す必要があります。文字列は、列挙値の名前と一致しなければなりません。たとえば、メソッドが .NET 列挙型 System.Windows.Visiblity のパラメータを必要とする場合、次の文字列値を使用できます： Visible、Hidden、Collapsed。
- .NET 構造体とオブジェクト

  .NET 構造体とオブジェクト パラメータはリストとして渡す必要があります。リスト内の要素は、テスト アプリケーションの .NET オブジェクトで定義されているコンストラクタの 1 つと一致しなければなりません。たとえば、メソッドが .NET 型 System.Windows.Vector のパラメータを必要とする場

合、2 つの整数値を持つリストを渡すことができます。これが機能するのは、System.Windows.Vector 型が 2 つの整数値を引数に取るコンストラクタを持つためです。
- その他のコントロール

  コントロール パラメーターは、TestObject として渡したり、返したりできます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4J 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、null が返されます。

Windows Presentation Foundation（WPF）のサポート

Silk4J は、Windows Presentation Foundation (WPF) アプリケーションのテストを組み込みでサポートしています。Silk4J は、スタンドアロン WPF アプリケーションをサポートしており、.NET バージョン 3.5 以降に組み込まれているコントロールを記録し、再生できます。

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

サポートするコントロール

WPF テストで使用可能なコントロールの完全な一覧については、「*WPF* クラス リファレンス」を参照してください。

Silk4J WPF がサポートするすべての WPF クラスは、WPFWindow や WPFListBox のように接頭辞 *WPF* で始まります。

WPF コントロールでサポートされるメソッドとプロパティは、実際の実装とランタイム状態によって異なります。メソッドとプロパティは、対応するクラスに対して定義されたリストと異なる場合があります。特定の状況でサポートされるメソッドとプロパティを判別するには、以下のコードを使用します。

- GetPropertyList()
- GetDynamicMethodList()

WPF の詳細については、*MSDN* を参照してください。

Windows Presentation Foundation (WPF) アプリケーションの属性

WPF アプリケーションがサポートする属性は次のとおりです。

- *automationId*
- *caption*
- *className*
- *name*
- すべての動的ロケーター属性。

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

動的ロケーター属性の詳細については、「動的ロケーター属性」を参照してください。

オブジェクト解決

WPF スクリプト内のコンポーネントを識別するために、*automationId*、*caption*、*className*、あるいは *name* を指定できます。 アプリケーション中の要素に指定された name が利用可能な場合、ロケーターの *automationId* 属性として使用されます。 この結果、多くのオブジェクトは、この属性のみを使用して一意に識別できます。 たとえば、*automationId* を持つロケーターは、以下のようになります：// WPFButton[@automationId='okButton']"

automationId や他の属性を定義した場合、再生中に *automationId* だけが使用されます。 *automationId* が定義されていない場合には、コンポーネントを解決するのに *name* が使用されます。 *name* も *automationId* もどちらも定義されていない場合には、*caption* 値が使用されます。 caption が定義されていない場合は、*className* が使用されます。 *automationId* は非常に役立つプロパティであるため、使用することを推奨します。

属性の種類	説明	例
automationId	テスト アプリケーションの開発者によって提供された ID	//WPFButton[@automationId='okButton']"
name	コントロールの名前。 Visual Studio デザイナは、デザイナ上で作成されたすべてのコントロールに自動的に名前を割り当てます。 アプリケーション開発者は、アプリケーションのコード上でコントロールを識別するために、この名前を使用します。	//WPFButton[@name='okButton']"
caption	コントロールが表示するテキスト。 複数の言語にローカライズされたアプリケーションをテストする場合、caption の代わりに automationId や name 属性を使用することを推奨します。	//WPFButton[@automationId='Ok']"
className	WPF の .NET 単純クラス名 (名前空間なし)。 クラス名属性を使用すると、Silk4J が解決する標準 WPF コントロールから派生したカスタム コントロールを識別するのに役立ちます。	//WPFButton[@className='MyCustomButton']"

Silk4J は、*automationId*、*name*、*caption*、*className* 属性をこの表に示した順番に使用して WPF コントロールのロケーターを記録時に作成します。 たとえば、コントロールが *automationId* と *name* を持つ場合、Silk4J がロケーターを作成する際には *automationId* が使用されます。

以下の例では、アプリケーション開発者がアプリケーションの WPF ボタンに対して *name* と *automationId* を XAML コードに定義する方法を示します。

```
<Button  Name="okButton"  AutomationProperties.AutomationId="okButton"
Click="okButton_Click">Ok</Button>
```

WPF アプリケーションのカスタム属性

WPF アプリケーションは、あらかじめ定義された自動化用プロパティ AutomationProperties.AutomationId を使用して、次のように WPF コントロールに対して安定した識別子を指定します。

```
<Window  x:Class="Test.MainWindow"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      Title="MainWindow"  Height="350"  Width="525">
   <Grid>
      <Button  AutomationProperties.AutomationId="AID_buttonA">The
Button</Button>
   </Grid>
</Window>
```

Silk4J は、ロケーターを識別するために、自動的にこのプロパティを使用します。WPF アプリケーションのロケーターは次のようになります。

```
/WPFWindow[@caption='MainWindow']//WPFButton[@automationId='AID_buttonA']
```

WPFItemsControl クラスから派生したクラス

Silk4J は、2 つの方法を使用して WPFItemsControl から派生したクラス (WPFListBox、WPFTreeView、WPFMenu など) を操作することができます。

- コントロールでの作業

  ほとんどのコントロールには、標準的なユースケースのためのメソッドやプロパティがあります。 項目は、テキストや索引によって識別されます。
- WPFListBoxItem、WPFTreeViewItem、WPFMenuItem などの個々の項目での作業

  高度なユースケースの場合、個々の項目を使用します。 たとえば、リスト ボックスの特定の項目のコンテキスト メニューを開いたり、項目に相対的な場所をクリックしたりする場合に個々の項目を使用します。

カスタム WPF コントロール

一般的に、Silk4J では、すべての標準 WPF コントロールの記録と再生がサポートされています。

Silk4J は、カスタム コントロールが実装された方法を基にしてカスタム コントロールを処理します。 次の方法を使用してカスタム コントロールを実装することができます。

- UserControl から派生したクラスを定義する

  複合コントロールを作成する典型的な方法です。 Silk4J は、これらのユーザー コントロールを WPFUserControl として認識し、含まれるコントロールを完全にサポートしています。
- ListBox などの標準 WPF コントロールから派生したクラスを定義する

  Silk4J は、これらのコントロールを派生元の標準 WPF コントロールのインスタンスとして扱います。ユーザー コントロールの振る舞いがその基底クラスの実装と大きく異なる場合には、子の記録、再生、解決は機能しない可能性があります。
- テンプレートを使用して視覚デザインを変更した標準コントロールを使用する

  低レベルの再生が機能しない可能性があります。 その場合には、「高レベル」再生モードに切り替えます。 再生モードを変更するには、**スクリプト オプション** ダイアログ ボックスを使用して、**OPT_REPLAY_MODE** オプションを変更します。

Silk4J は、一般的に機能テストに無関係なコントロールは除外します。 たとえば、レイアウトを目的として使用されるコントロールは含まれません。 しかし、カスタム コントロールが除外されたクラスから派生している場合、除外されたコントロールを記録/再生の対象とするためには、関連する WPF クラスの名前を指定します。

WPF メソッドの動的な呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティーの取得、またはプロパティーの設定を直接実行できます。また、このコントロールの Silk4J API で使用できないメソッドおよびプロパティーも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4J API を通して公開されていない場合に特に便利です。

オブジェクトの動的メソッドは invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

オブジェクトの複数の動的メソッドは invokeMethods メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

動的プロパティの取得には getProperty メソッドを、動的プロパティの設定には setProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、getPropertyList メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを String 型の入力パラメータとして設定する必要がある SetTitle というメソッドを呼び出すには、次のように入力します：

```
control.invoke("SetTitle","my  new  title");
```

注: 通常、ほとんどのプロパティは読み取り専用で、設定できません。

注: ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティーを取得する場合、Reflection を使用します。

invoke メソッド

Windows Forms または WPF コントロールでは、invoke メソッドを使用して、以下のメソッドを呼び出すことができます。

- MSDN が定義するコントロールのパブリック メソッド。
- MSDN が定義する静的パブリック メソッド。
- ユーザーが定義する任意の型の静的パブリック メソッド。

invoke メソッドの最初の例

Silk4J の DataGrid 型のオブジェクトでは、MSDN が System.Windows.Forms.DataGrid 型に定義しているすべてのメソッドを呼び出すことができます。

System.Windows.Forms.DataGrid クラスのメソッド IsExpanded を呼び出すには、次のコードを使用します。

```
//Java  code
boolean  isExpanded  =  (Boolean)  dataGrid.invoke("IsExpanded",  3);
```

invoke メソッドの 2 番目の例

AUT 内の静的メソッド String.compare(String s1, String s2) を呼び出すには、次のコードを使用します。

```
//Java  code
int  result  =  (Integer)  mainWindow.invoke("System.String.Compare",  "a",
"b");
```

invoke メソッドの 3 番目の例

この例では、ユーザーが生成したメソッド GetContents を動的に呼び出す方法を示します。

テスト対象アプリケーション (AUT) のコントロールの操作に使用するコードを作成できます (この例では UltraGrid)。UltraGrid の内容を取得するために、複雑な動的呼び出しを作成するのではなく、新しいメソッド GetContents を生成し、この新しいメソッドを動的に呼び出すことができます。

Visual Studio で、AUT 内の次のコードによって GetContents メソッドを UltraGridUtil クラスのメソッドとして定義します。

```
//C# code, because this is code in the AUT
namespace UltraGridExtensions {
   public class UltraGridUtil {
      /// <summary>
      /// Retrieves the contents of an UltraGrid as nested list
      /// </summary>
      /// <param name="grid"></param>
      /// <returns></returns>
      public static List<List<string>>
GetContents(Infragistics.Win.UltraWinGrid.UltraGrid grid) {
         var result = new List<List<string>>();
         foreach (var row in grid.Rows) {
            var rowContent = new List<string>();
            foreach (var cell in row.Cells) {
               rowContent.Add(cell.Text);
            }
            result.Add(rowContent);
         }
         return result;
      }
   }
}
```

UltraGridUtil クラスのコードを AUT に追加する必要があります。これは、次のようにして行います。

- アプリケーション開発者は、クラスのコードを AUT にコンパイルできます。アセンブリがすでにロードされている必要があります。
- テストの実行時に AUT にロードされる新しいアセンブリを作成できます。

アセンブリをロードするには、次のコードを使用します。

```
FormsWindow.LoadAssembly(String  assemblyFileName)
```

次のようにして、フルパスで指定してアセンブリをロードします。

```
mainWindow.LoadAssembly("C:/temp/ultraGridExtensions.dll")
```

UltraGridUtil クラスのコードが AUT 内にある場合は、次のコードをテスト スクリプトに追加して、GetContents メソッドを呼び出すことができます。

```
List<List<String>> contents =
mainWindow.invoke("UltraGridExtensions.UltraGridUtil.GetContents",
ultraGrid);
```

invoke メソッドを呼び出す mainWindow オブジェクトは、AUT を特定しているだけなので、同じ AUT の他のオブジェクトに置き換えてもかまいません。

invokeMethods メソッド

Windows Forms または WPF コントロールでは、invokeMethods メソッドを使用して、ネストされたメソッドのシーケンスを呼び出すことができます。以下のメソッドを呼び出すことができます。

- MSDN が定義するコントロールのパブリック メソッド。
- MSDN が定義する静的パブリック メソッド。
- ユーザーが定義する任意の型の静的パブリック メソッド。

例：カスタム データ グリッドのセルの内容のテキストでの取得

Infragistics ライブラリのカスタム データ グリッドのセルの内容をテキストで取得するには、AUT で次の C# コードを使用できます。

```
string cellText = dataGrid.Rows[rowIndex].Cells[columnIndex].Text;
```

次の C# コードのサンプルは、最初の行の 3 番目のセルの内容をテキストで取得します。

```
string cellText = dataGrid.Rows[0].Cells[2];
```

invokeMethods メソッドを使用して同じ例をスクリプト化すると、比較的複雑なスクリプトになります。これは、対応するパラメータを持つ 5 つのメソッドを invokeMethods メソッドに渡さなければならないためです。

```
WPFControl dataGrid = mainWindow.find("//
WPFControl[@automationId='Custom Data Grid']");

// Get text contents of third cell in first row.
int rowIndex = 0;
int columnIndex = 2;

List<String> methodNames = Arrays.asList("Rows", "get_Item", "Cells",
"get_Item", "Text");
List<List<Object>> parameters = Arrays.asList(new ArrayList<Object>(),
Arrays.<Object>asList(rowIndex), new ArrayList<Object>(),
Arrays.<Object>asList(rowIndex), new ArrayList<Object>());

String cellText = (String) dataGrid.invokeMethods(methodNames,
parameters);
```

このような場合に、より簡単にするアプローチは、テスト対象アプリケーションにコードを追加して、invokeMethods メソッドを使用することです。たとえば、getCellText メソッドを AUT に追加します。

```
// C# code, if the AUT is implemented in C#.
public static string GetCellText(Infragistics.Win.UltraWinGrid.UltraGrid
dataGrid, int rowIndex, int columnIndex) {
  return dataGrid.Rows[rowIndex].Cells[columnIndex].Text;
```

```
' VB code, if the AUT is implemented in VB.
public static string GetCellText(Infragistics.Win.UltraWinGrid.UltraGrid
dataGrid, int rowIndex, int columnIndex) {
  return dataGrid.Rows[rowIndex].Cells[columnIndex].Text;
```

テスト スクリプトから GetCellText メソッドを動的に呼び出して、セルの内容をテキストで取得します。

```
String cellText = (String) mainWindow.invoke("GetCellText", dataGrid,
rowIndex, columnIndex);
```

詳細については、「テスト対象アプリケーションにコードを追加してカスタム コントロールをテストする」を参照してください。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4J がサポートするコントロールのメソッドとプロパティー。
- MSDN が定義するコントロールのパブリック メソッドとプロパティー。
- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティー。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- すべての組み込み Silk4J 型

  Silk4J 型には、プリミティブ型 (boolean、int、string など)、リスト、およびその他の型 (Point や Rect など) が含まれます。
- 列挙型

  列挙パラメータは文字列として渡す必要があります。文字列は、列挙値の名前と一致しなければなりません。たとえば、メソッドが .NET 列挙型 System.Windows.Visiblity のパラメータを必要とする場合、次の文字列値を使用できます： Visible、Hidden、Collapsed。
- .NET 構造体とオブジェクト

  .NET 構造体とオブジェクト パラメータはリストとして渡す必要があります。リスト内の要素は、テスト アプリケーションの .NET オブジェクトで定義されているコンストラクタの 1 つと一致しなければなりません。たとえば、メソッドが .NET 型 System.Windows.Vector のパラメータを必要とする場合、2 つの整数値を持つリストを渡すことができます。これが機能するのは、System.Windows.Vector 型が 2 つの整数値を引数に取るコンストラクタを持つためです。
- WPF コントロール

  WPF コントロール パラメータは TestObject として渡すことができます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4J 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、null が返されます。
- すべてのその他の型の場合は文字列

  返された .NET オブジェクトに対して ToString を呼び出せば、文字列表現を取得できます。

例

たとえば、アプリケーション開発者が次のメソッドとプロパティを持つ Calculator カスタム コントロールを作成したとします。

```
public void Reset()
public int Add(int number1, int number2)
public System.Windows.Vector StrechVector(System.Windows.Vector vector,
double
factor)
public String Description { get;}
```

テスト担当者は、テスト内からメソッドを直接呼び出すことができます。例：

```
customControl.invoke("Reset");
int sum = customControl.invoke("Add", 1, 2);
// the vector can be passed as list of integer
List<Integer> vector = new ArrayList<Integer>();
vector.add(3);
```

```
vector.add(4);
// returns "6;8" because this is the string representation of the .NET
object
String strechedVector = customControl.invoke("StrechVector", vector, 2.0);
String description = customControl.getProperty("Description");
```

記録/再生の対象とする WPF クラスの設定

Silk4J は、一般的に機能テストに無関係なコントロールは除外します。たとえば、レイアウトを目的として使用されるコントロールは含まれません。しかし、カスタム コントロールが除外されたクラスから派生している場合、除外されたコントロールを記録/再生の対象とするためには、関連する WPF クラスの名前を指定します。

記録や再生の対象にしたい WPF クラスの名前を指定します。たとえば、*MyGrid* というカスタム クラスが WPF Grid クラスから継承された場合、*MyGrid* カスタム クラスのオブジェクトは記録や再生に使用できません。Grid クラスはレイアウト目的のためにのみ存在し、機能テストとは無関係であるため、Grid オブジェクトは記録や再生に使用できません。この結果、Grid オブジェクトはデフォルトでは公開されません。機能テストに無関係なクラスに基づいたカスタム クラスを使用するには、カスタム クラス (この場合は *MyGrid*) を **OPT_WPF_CUSTOM_CLASSES** オプションに追加します。これによって、記録、再生、検索、プロパティの検証など、すべてのサポートされる操作を指定したクラスに対して実行できるようになります。

1. **Silk4J** > **オプションの編集** をクリックします。
2. **WPF** タブ をクリックします。
3. **カスタム WPF クラス名** グリッドで、記録や再生中に公開するクラスの名前を入力します。
   複数のクラス名を指定する場合にはカンマで区切ります。
4. **OK** をクリックします。

記録/再生時の事前読み込みの設定

記録および再生中に 、WPFComboBox や WPFListBox のような WPFItemsControl 内の項目を事前に入力するかどうかを定義します。WPF 自体が特定のコントロールの項目を遅延読み込みするため、項目がビューにスクロールされない場合、それらの項目は Silk4J では使用できません。ビューにスクロールされないとアクセスできない項目にアクセスするには、事前入力をオンにします。これはデフォルトの設定です。ただし、一部のアプリケーションでは Silk4J によってバックグランドで項目が事前入力されると問題が発生し、そのためアプリケーションがクラッシュすることがあります。この場合、事前入力をオフにします。

1. **Silk4J** > **オプションの編集** をクリックします。
2. **WPF** タブ をクリックします。
3. **項目の事前読み込み** 領域で、**OPT_WPF_PREFILL_ITEMS** チェック ボックスをオンにします。
4. **OK** をクリックします。

Silverlight アプリケーションのサポート

Microsoft Silverlight (Silverlight) は、リッチ インターネット アプリケーションを記述し、実行するためのアプリケーション フレームワークで、Adobe Flash と同様の機能と目的を備えています。Silverlight の実行時環境は、大部分の Web ブラウザでプラグインとして使用できます。

Silk4J は、Silverlight アプリケーションのテストを組み込みでサポートしています。Silk4J は、ブラウザ内部と同様ブラウザ外部でも実行される Silverlight アプリケーションをサポートしており、.NET バージョン 3.5 以降でコントロールを記録し、再生できます。

Silverlight をベースとする以下のアプリケーションがサポートされます。

- Internet Explorer で実行される Silverlight アプリケーション

- ブラウザー外実行 Silverlight アプリケーション

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

サポートするコントロール

Silk4J は、Silverlight コントロールの記録と再生をサポートしています。

Silverlight テストで使用可能なコントロールの完全な一覧については、「*Silverlight* クラス リファレンス」を参照してください。

注: Silk Test 14.0 以降では、Silk4J は、画面上で操作可能でかつ表示されている Silverlight コントロールのみを認識します。この変更は、Silk Test 14.0 より前のバージョンの Silk Test を使用して記録されたテストの動作に影響を与える可能性があります。Silk Test 14.0 以降を使用してこのようなテストを実行するには、不可視な、または利用可能でないすべての Silverlight コントロールをテストから削除してください。

前提条件

Microsoft Windows XP で Silverlight アプリケーションをテストする場合、サービス パック 3 をインストールし、Windows 7 で提供される Microsoft User Interface Automation の Windows XP 用の更新プログラムを適用する必要があります。更新プログラムは、*http://www.microsoft.com/download/en/details.aspx?id=13821* からダウンロードできます。

注: Microsoft User Interface Automation は、Silverlight サポート用にインストールする必要があります。Windows オペレーティング システムを使用していて、Silverlight サポートが機能しない場合は、使用しているオペレーティング システム用の Microsoft User Interface Automation の更新プログラムを *http://support.microsoft.com/kb/971513* からダウンロードしてインストールしてください。

Silverlight コントロールを識別するためのロケーター属性

Silverlight コントロールでサポートされているロケーター属性は次のとおりです。

- *automationId*
- *caption*
- *className*
- *name*
- すべての動的ロケーター属性

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

動的ロケーター属性の詳細については、「動的ロケーター属性」を参照してください。

Silverlight スクリプト内のコンポーネントを識別するために、*automationId*、*caption*、*className*、*name*、または任意の動的ロケーター属性を指定できます。 *automationId* はアプリケーション開発者が設定します。 たとえば、*automationId* を持つロケーターは、以下のようになります：// SLButton[@automationId="okButton"]

automationId は一般に非常に有用で安定した属性であるため、使用することを推奨します。

属性の種類	説明	例
automationId	テスト対象アプリケーションの開発者によって設定される識別子。 Visual Studio デザイナは、デザイナ上で作成されたすべてのコントロールに自動的に *automationId* を割り当てます。 アプリケーション開	// SLButton[@automationId="okButton"]

属性の種類	説明	例
	発者は、アプリケーションのコード上でコントロールを識別するために、この ID を使用します。	
caption	コントロールが表示するテキスト。 複数の言語にローカライズされたアプリケーションをテストする場合、*caption* の代わりに *automationId* や *name* 属性を使用することを推奨します。	//SLButton[@caption="Ok"]
className	Silverlight コントロールの .NET 単純クラス名 (名前空間なし)。 *className* 属性を使用すると、Silk4J が解決する標準 Silverlight コントロールから派生したカスタム コントロールを識別するのに役立ちます。	//SLButton[@className='MyCustomButton']
name	コントロールの名前。 テスト対象アプリケーションの開発者によって設定されます。	//SLButton[@name="okButton"]

注目: XAML コードの *name* 属性は、ロケーター属性 *name* ではなく、ロケーター属性 *automationId* にマップされます。

Silk4J は、*automationId*、*name*、*caption*、*className* 属性をこの表に示した順番に使用して Silverlight コントロールのロケーターを記録時に作成します。 たとえば、コントロールが *automationId* と *name* を持つ場合、*automationId* が固有の場合は Silk4J がロケーターを作成する際に使用されます。

以下の表は、アプリケーション開発者がテキスト「Ok」を持つ Silverlight ボタンをアプリケーションの XAML コードに定義する方法を示しています。

オブジェクトの XAML コード	Silk Test からオブジェクトを検索するためのロケーター
<Button>Ok</Button>	//SLButton[@caption="Ok"]
<Button Name="okButton">Ok</Button>	//SLButton[@automationId="okButton"]
<Button AutomationProperties.AutomationId="okButton">Ok</Button>	//SLButton[@automationId="okButton"]
<Button AutomationProperties.Name="okButton">Ok</Button>	//SLButton[@name="okButton"]

Silverlight メソッドの動的呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティーの取得、またはプロパティーの設定を直接実行できます。また、このコントロールの Silk4J API で使用できないメソッドおよびプロパティーも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4J API を通して公開されていない場合に特に便利です。

オブジェクトの動的メソッドは invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

オブジェクトの複数の動的メソッドは invokeMethods メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

動的プロパティの取得には getProperty メソッドを、動的プロパティの設定には setProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、getPropertyList メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを String 型の入力パラメータとして設定する必要がある SetTitle というメソッドを呼び出すには、次のように入力します：

```
control.invoke("SetTitle","my  new  title");
```

注: 通常、ほとんどのプロパティは読み取り専用で、設定できません。

注: ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティーを取得する場合、Reflection を使用します。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- すべての組み込み Silk4J 型。

  Silk4J 型には、プリミティブ型 (boolean、int、string など)、リスト、およびその他の型 (Point、Rect など) が含まれます。
- 列挙型。

  列挙パラメータは文字列として渡す必要があります。文字列は、列挙値の名前と一致しなければなりません。たとえば、メソッドが .NET 列挙型 System.Windows.Visiblity のパラメータを必要とする場合には、Visible、Hidden、Collapsed の文字列値を使用できます。
- .NET 構造体とオブジェクト。

  .NET 構造体とオブジェクト パラメータはリストとして渡します。リスト内の要素は、テスト アプリケーションの .NET オブジェクトで定義されているコンストラクタの 1 つと一致しなければなりません。たとえば、メソッドが .NET 型 System.Windows.Vector のパラメータを必要とする場合、2 つの整数値を持つリストを渡すことができます。これが機能するのは、System.Windows.Vector 型が 2 つの整数値を引数に取るコンストラクタを持つためです。
- その他のコントロール。

  コントロール パラメータは TestObject として渡すことができます。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- MSDN が定義する AutomationElement クラスのすべてのパブリック メソッドとプロパティ。詳細については、*http://msdn.microsoft.com/en-us/library/system.windows.automation.automationelement.aspx* を参照してください。
- MSUIA が公開するすべてのメソッドとプロパティ。利用可能なメソッドとプロパティは「パターン」で分類されます。パターンとは、MSUIA 固有の用語です。すべてのコントロールは、いくつかのパターンを実装します。一般的なパターンについての概要およびすべての利用可能なパターンについては、*http://msdn.microsoft.com/en-us/library/ms752362.aspx* を参照してください。カスタム コントロールの開発者は、MSUIA パターン セットを実装することによって、カスタム コントロールのテスト サポートを提供できます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4J 型の場合は正しい値。
- 戻り値を持たないすべてのメソッドの場合、null が返されます。
- すべてのその他の型の場合は文字列。

  この文字列表現を取得するには、テスト対象アプリケーションの返された .NET オブジェクトに対して ToString メソッドを呼び出します。

例

Silverlight の TabItem。 これは TabControl の項目です。

```
tabItem.invoke("SelectionItemPattern.Select");
mySilverligtObject.getProperty("IsPassword");
```

Silverlight でのスクロール

Silk4J では、Silverlight コントロールに応じて、2 種類のスクロール方法とプロパティを提供します。

- 1 つめの種類のコントロールには、それ自体でスクロール可能なコントロールが含まれ、スクロールバーは子として明示的に表示されません。 たとえば、コンボ ボックス、ペイン、リスト ボックス、ツリー コントロール、データ グリッド、オート コンプリート ボックスなどがあります。
- 2 つめの種類のコントロールには、それ自体ではスクロール不可能なコントロールが含まれ、スクロール用にスクロールバーが子として表示されます。 たとえば、テキスト フィールドがあります。

Silk4J にこのような違いがあるのは、Silk4J のコントロールがこの 2 通りの方法でスクロールを実装するためです。

スクロールをサポートするコントロール

この場合、スクロール方法とプロパティは、スクロールバーを含むコントロールで使用できます。 したがって、 Silk4J ではスクロールバー オブジェクトは表示されません。

使用例

以下のコマンドでは、リスト ボックスが一番下までスクロールされます。

```
listBox.SetVerticalScrollPercent(100)
```

以下のコマンドでは、リスト ボックスが 1 ユニットずつ下方へスクロールされます。

```
listBox.ScrollVertical(ScrollAmount.SmallIncrement)
```

スクロールをサポートしないコントロール

この場合、スクロールバーが表示されます。 コントロール自体で可能なスクロール方法とプロパティはありません。 水平スクロールバーと垂直スクロールバーの各オブジェクトを使用すると、対応する API 関数でパラメータとして増分または減分、または最終位置を指定することでコントロール内をスクロールできます。 増分または減分として ScrollAmount 列挙の値を使用できます。 詳細については、Silverlight の製品マニュアルを参照してください。 最終位置は、オブジェクトの位置に関連し、アプリケーション設計者によって定義されます。

使用例

以下のコマンドでは、テキスト ボックス内の垂直スクロールバーが 15 の位置までスクロールされます。

```
textBox.SLVerticalScrollBar().ScrollToPosition(15)
```

以下のコマンドでは、テキスト ボックス内の垂直スクロールバーが一番下までスクロールされます。

```
textBox.SLVerticalScrollBar().ScrollToMaximum()
```

Silverlight アプリケーションのテスト時のトラブルシューティング

Silk4J で Silverlight アプリケーションの内部を確認できず、記録時に緑色の矩形領域が描画されない

次の理由により、Silk4J は Silverlight アプリケーションの内部を確認できなくなっています。

原因	解決策
使用している Silverlight のバージョンが 3 以前である	Silverlight 3 (Silverlight Runtime 4) または Silverlight 4 (Silverlight Runtime 4) を使用します。
使用している Silverlight アプリケーションがウィンドウレス モードで実行されている	Silk4J は、ウィンドウレス モードで実行される Silverlight アプリケーションをサポートしません。このようなアプリケーションをテストするには、Silverlight アプリケーションが実行されている Web サイトを変更する必要があります。つまり、Silverlight アプリケーションがホストされている HTML または ASPX ファイルのオブジェクト タグの windowless パラメータを false に設定する必要があります。 以下のコードは、windowless パラメータを false に設定する例を示します。 `<object ...> <param name="windowless" value="false"/> ... </object>`

Visual COBOL のサポート

Silk4J は、Visual COBOL アプリケーションに対するテストの記録と再生をサポートします。.NET COBOL から Silk4J API を使用して、次の Visual COBOL アプリケーションに対して自動テストのスクリプトを記述することもできます。

- ダイアログ システム アプリケーション
- CGI アプリケーション
- COBOL Win32 アプリケーション
- .NET COBOL - WPF および Windows Forms アプリケーション
- COBOL JVM - Swing アプリケーション
- バックエンドにある COBOL を呼び出す非 COBOL のフロントエンド アプリケーション

注: コントロールによっては、Silk4J は低レベルの記録のみをサポートします。

サポートする Visual COBOL のバージョンについての情報は、『リリース ノート』を参照してください。

Rumba のサポート

Rumba は、世界トップクラスの Windows デスクトップ端末エミュレーション ソリューションです。Silk Test は、Rumba の記録および再生を組み込みでサポートしています。

Rumba でのテスト時には、以下の点を考慮してください。

- Rumba のバージョンは、Silk Test のバージョンと互換性がある必要があります。バージョン 8.1 以前の Rumba はサポートされていません。

- Rumba のグリーン スクリーンの周囲にあるコントロールはすべて WPF の基本機能 (または Win32) を使用しています。
- サポートされている Rumba デスクトップ タイプは、以下のとおりです。
  - メインフレーム ディスプレイ
  - AS400 ディスプレイ
  - Unix ディスプレイ

Rumba テストで使用できる記録および再生のコントロールの完全な一覧については、「*Rumba* クラス リファレンス」を参照してください。

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

Rumba の有効化と無効化

Rumba は、世界トップクラスの Windows デスクトップ端末エミュレーション ソリューションです。 Rumba は、メインフレーム、ミッドレンジ、UNIX、Linux、および HP サーバーとの接続ソリューションを提供します。

サポートの有効化

Rumba スクリプトを記録および再生する前に、サポートを有効にする必要があります。

1. Rumba デスクトップ クライアント ソフトウェア バージョン 8.1 以降をインストールします。
2. (Microsoft Windows 7) **スタート** > **すべてのプログラム** > **Silk** > **Silk Test** > **管理** > **Rumba プラグイン** > **Silk Test Rumba プラグインの有効化**、または (Microsoft Windows 10) **スタート** > **Silk** > **Silk Test Rumba プラグインの有効化** をクリックします。

サポートの無効化

(Microsoft Windows 7) **スタート** > **すべてのプログラム** > **Silk** > **Silk Test** > **管理** > **Rumba プラグイン** > **Silk Test Rumba プラグインの無効化**、または (Microsoft Windows 10) **スタート** > **Silk** > **Silk Test Rumba プラグインの無効化** をクリックします。

Rumba コントロールを識別するためのロケーター属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。 サポートされている属性は次のとおりです。

caption	コントロールが表示するテキスト。
priorlabel	フォームの入力フィールドには通常入力の目的を説明するラベルがあるため、**priorlabel** の目的は隣接するラベル フィールド **RumbaLabel** のテキストによってテキスト入力フィールド **RumbaTextField** を識別することです。 テキスト フィールドの同じ行の直前にラベルがない場合、または右側のラベルが左側のラベルよりテキスト フィールドに近い場合、テキスト フィールドの右側にあるラベルが使用されます。
StartRow	この属性は記録されていませんが、手動でロケーターに追加することができます。 **StartRow** を使用して、この行で始まるテキスト入力フィールド、**RumbaTextField** を識別します。
StartColumn	この属性は記録されていませんが、手動でロケーターに追加することができます。 **StartColumn** を使用して、この列で始まるテキスト入力フィールド、**RumbaTextField** を識別します。

すべての動的ロケーター属性。 動的ロケーター属性の詳細については、「動的ロケーター属性」を参照してください。

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Unix ディスプレイのテスト

Rumba の Unix ディスプレイは完全にテキスト ベースで、メインの **RUMBA 画面** コントロール以外に UI コントロールはありません。Unix ディスプレイ上のテストを再生するには、sendKeys メソッドを使用して、Unix ディスプレイにキーを送信します。Silk4J は、Unix ディスプレイ上での記録をサポートしません。

SAP のサポート

Silk4J は、Windows ベースの GUI モジュールを基にした SAP クライアント／サーバー アプリケーションのテストを組み込みでサポートしています。

注: Silk4J のプレミアム ライセンスを所有している場合にのみ、Silk4J で SAP アプリケーションをテストできます。ライセンス モードについての詳細は、「ライセンス情報」を参照してください。

注: Internet Explorer や Firefox 上から SAP NetWeaver を使用する場合、Silk4J は、xBrowser テクノロジ ドメインを使用してアプリケーションをテストします。

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

サポートするコントロール

SAP のテストで利用可能な記録および再生コントロールの完全な一覧については、「*SAP* クラス リファレンス」を参照してください。

サポートされている属性の一覧については、「*SAP* アプリケーションの属性」を参照してください。

SAP アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

SAP がサポートする属性は次のとおりです。

- automationId
- caption

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

SAP メソッドの動的な呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティーの取得、またはプロパティーの設定を直接実行できます。また、このコントロールの Silk4J API で使用できないメソッドおよびプロパティーも呼び出すことができます。

動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4J API を通して公開されていない場合に特に便利です。

オブジェクトの動的メソッドは invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

オブジェクトの複数の動的メソッドは invokeMethods メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

動的プロパティの取得には getProperty メソッドを、動的プロパティの設定には setProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、getPropertyList メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを String 型の入力パラメータとして設定する必要がある SetTitle というメソッドを呼び出すには、次のように入力します：

```
control.invoke("SetTitle","my  new  title");
```

注: 通常、ほとんどのプロパティは読み取り専用で、設定できません。

注: ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティーを取得する場合、Reflection を使用します。

サポートされているメソッドおよびプロパティ

次のメソッドとプロパティを呼び出すことができます。

- Silk4J がサポートするコントロールのメソッドとプロパティー。
- SAP オートメーション インターフェイスによって定義されているすべての public メソッド
- コントロールが標準コントロールから派生したカスタム コントロールの場合、標準コントロールが呼び出すことのできるすべてのメソッドとプロパティー。

サポートされているパラメータ型

次のパラメータ型がサポートされます。

- すべての組み込み Silk4J 型

  Silk4J 型には、プリミティブ型 (boolean、int、string など)、リスト、およびその他の型 (Point や Rect など) が含まれます。
- UI コントロール

  UI コントロールは、TestObject として渡したり、返したりできます。

戻り値

プロパティや戻り値を持つメソッドの場合は、次の値が返されます。

- すべての組み込み Silk4J 型の場合は正しい値。これらの型は、「サポートされているパラメータ型」のセクションに記載されています。
- 戻り値を持たないすべてのメソッドの場合、null が返されます。

SAP コントロールの動的呼び出し

Silk4J で SAP コントロールに対する操作を記録できない場合、SAP で利用できるレコーダーで操作を記録してから、記録されたメソッドを Silk4J スクリプトで動的に呼び出すことができます。これによって、記録できない SAP コントロールに対する操作を再生できます。

1. コントロールに対して実行する操作を記録するには、SAP で利用できる **SAP GUI スクリプト作成** ツールを使用できます。

SAP GUI スクリプト作成 ツールの詳細については、SAP のドキュメントを参照してください。

2. 記録された操作を **SAP GUI スクリプト作成** ツールによって保存された場所から開き、記録されたメソッドを確認します。
3. Silk4J で、記録されたメソッドをスクリプトから動的に呼び出します。

使用例

たとえば、SAP UI で *Test* というラベルが付いた、ボタンとリスト ボックスの組み合わせである特別なコントロールを押し、コントロールのサブメニュー *subsub2* を選択する操作を再生したい場合、SAP で利用できるレコーダーでこの操作を記録できます。結果のコードは、以下のようになります。

```
session.findById("wnd[0]/usr/cntlCONTAINER/shellcont/
shell").pressContextButton  "TEST"
session.findById("wnd[0]/usr/cntlCONTAINER/shellcont/
shell").selectContextMenuItem  "subsub2"
```

ここで、メソッド pressContextButton と selectContextMenuItem を Silk4J のスクリプトから動的に呼び出すには、次のコードを使用できます。

```
.SapToolbarControl("shell  ToolbarControl").invoke("pressContextButton",
"TEST")
.SapToolbarControl("shell  ToolbarControl").invoke("selectContextMenuItem",
"subsub2")
```

このコードを再生すると、SAP UI のコントロールが押下され、サブメニューが選択されます。

SAP の自動セキュリティ設定の構成

SAP アプリケーションを起動する前に、セキュリティ警告設定を構成する必要があります。 このようにしないと、テストで SAP アプリケーションが再生されるたびにセキュリティ警告「スクリプトから GUI に接続しようとしています」が表示されます。

1. Windows の **コントロール パネル** で **SAP システム設定** を選択します。 **SAP システム設定** ダイアログ ボックスが開きます。
2. **デザイン選択** タブで、**スクリプトが実行中 SAP GUI に追加されるとき通知** をオフにします。

Windows API ベースのアプリケーションのサポート

Silk4J は、Microsoft Windows API ベースのアプリケーションのテストを組み込みでサポートしています。アクセシビリティを有効にすると Microsoft のアプリケーションのいくつかのオブジェクトが Silk4J によってより詳細に認識されます。たとえば、アクセシビリティを有効にしないと、Silk4J は Microsoft Word のメニューバーおよび バージョン 7.0 より後の Internet Explorer に表示されるタブについて基本的な情報のみを記録します。ただし、アクセシビリティを有効にすると、Silk4J によってそれらのオブジェクトがすべて認識されます。必要な場合、新しいウィンドウを定義すると、Silk4J によるオブジェクトの認識を向上させることもできます。

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

サポートするコントロール

Windows ベースのテストで利用可能な記録および再生コントロールの完全な一覧については、「*Win32* クラス リファレンス」を参照してください。

Windows API ベースのクライアント/サーバー アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Windows API ベースのクライアント/サーバー アプリケーションがサポートする属性は次のとおりです。

- caption
- windowid
- priorlabel : 隣接するラベル フィールドのテキストによってテキスト入力フィールドを識別します。 通常、フォームのすべての入力フィールドに、入力の目的を説明するラベルがあります。 caption のないコントロールの場合、自動的に属性 **priorlabel** がロケーターに使用されます。 コントロールの **priorlabel** 値 (テキスト ボックスなど) には、コントロールの左側または上にある最も近いラベルの caption が使用されます。

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Win32 テクノロジ ドメインにおける priorLabel の決定方法

Win32 テクノロジ ドメインにおいて priorLabel を決定する場合、同じウィンドウ内のすべてのラベルとグループが対象のコントロールとみなされます。 以下の条件に従って、コントロールが決定されます。

- コントロールの上または左側にあるラベル、およびコントロールを囲むグループが priorLabel の候補とみなされます。
- 最も単純なケースでは、コントロールに最も近いラベルが priorLabel として使用されます。
- コントロールからの距離が等しい 2 つのラベルが存在する場合、次の条件に基づいて priorLabel が決定されます。
  - 一方のラベルがコントロールの左側にあり、他方が上にある場合、左側のものが優先されます。
  - 両方のラベルがコントロールの左側にある場合、上にあるものが優先されます。
  - 両方のラベルがコントロールの上にある場合、左側のものが優先されます。
- 最も近いコントロールがグループ コントロールである場合、まずグループ内のすべてのラベルが上記の規則に従って決定されます。 グループ内に適切なラベルが見つからない場合は、グループのキャプションが priorLabel として使用されます。

埋め込み Chrome アプリケーションのテスト

埋め込み Chrome アプリケーションは、Chromium コアをベースとした埋め込み Web ブラウザー エンジンを使用したデスクトップ アプリケーションです。このようなアプリケーションでは、デスクトップ アプリケーションに Web ブラウザーの機能を追加できます。Chromium Embedded Framework (CEF) や Electron フレームワークを使用して、このようなアプリケーションを作成できます。

Silk4J は、--remote-debugging-port コマンド ライン引数を指定してリモート デバッグを有効にした埋め込み Chrome アプリケーションのテストをサポートします。Silk4J では、Java（Java AWT や Swing アプリケーションなど）をベースにした埋め込み Chrome アプリケーションのテストをサポートしません。

Silk4J を使用して埋め込み Chrome アプリケーションをテストするには、アプリケーションの実行可能ファイルにデバッグ ポートを設定する必要があります。コマンド ラインからアプリケーションを開始して、リモート デバッグ ポートを設定します。

- Silk4J は、-remote-debugging-port 引数が埋め込み Chrome アプリケーションのコマンド ライン引数に設定されているかどうかを確認します。引数が設定されている場合、Silk4J は **埋め込み Chrome サポートの有効化** フィールドに適切な実行可能ファイルとデバッグ用ポートを自動的に設定します。
- -remote-debugging-port 引数が埋め込み Chrome アプリケーションのコマンド ライン引数に設定されていない場合は、実行可能ファイルとポートを **埋め込み Chrome サポートの有効化** フィールドに次の手順で指定する必要があります。

  1. Silk4J UI で、**オプションの編集** を選択します。
  2. **オプション** ダイアログで、**詳細設定** タブを選択します。
  3. **埋め込み Chrome サポートの有効化** オプションで、実行可能ファイルとポートの値ペアを、カンマ区切りで次のように指定します。

     ```
     <application name>.exe=<port number>
     ```

注: Silk4J によるリモート デバッグを許可しない埋め込み Chrome アプリケーションをテストすることはできません。

注: Silk4J では、Electron アプリの非ブラウザー メニューのテストをサポートしません。

例

たとえば、コマンド ラインから *myApp* アプリケーションを次のように実行します。

```
myApp.exe --remote-debugging-port=9222
```

この場合、**埋め込み Chrome サポートの有効化** オプションに実行可能ファイルとポートの値ペアを次のように指定します。

```
myApp.exe=9222
```

MFC（Microsoft Foundation Class）のサポート

MFC（Microsoft Foundation Class）のクラス ID は、時間がたつと変更されることがあるため、安定したロケーターを生成するために使用することはできません。不安定なロケーターの生成を避けるため、Silk4J では次の属性をロケーターとして使用します。

- MFC クラス名（MFC コントロールの Windows クラス名が Afx: で始まる場合）
- Windows クラス名（MFC コントロールの Windows クラス名が Afx: で始まらない場合）

Silk4J は MFC バージョン 140 のみをサポートします。 また、サポートする組み合わせは次の通りです。

- Release、x86、MBCS
- Release、x86、Unicode
- Debug、x86、MBCS
- Debug、x86、Unicode
- Release、x64、MBCS
- Release、x64、Unicode
- Debug、x64、MBCS
- Debug、x64、Unicode

注: Silk4J 18.5 以前で生成した MFC コントロールのロケーターを持つ既存のテストを実行する場合は、影響するテスト スクリプトで OPT_COMPATIBILITY オプションをバージョン 18.5.0 以前に設定してください。

```
'VB .NET code
Agent.SetOption("OPT_COMPATIBILITY", "18.5.0")
```

クロス ブラウザ テスト

Silk4J を使用すると、単一のポータブルなテスト スクリプトでさまざまなブラウザに対して非常に高度な Web アプリケーションの機能を簡単に検証できます。 Silk4J は、最新の Web テクノロジを使用した、効率的でメンテナンスしやすいクロス ブラウザ テストのトップレベルのサポートを提供します。

テスト自動化における主要な困難のひとつが、テストの作成と保守にかかるコストの効率化です。ブラウザによって動作が異なるため、Web アプリケーションの検証を生産的に実行することは困難です。Silk4J は、クロス ブラウザ テストにおける次の 3 つの領域をうまく扱うため、ユーザーはテストの記述に集中することができます。

組み込みの同期処理 これにより、サポートするすべてのブラウザ上で実行するスクリプトを作成することができます。 AJAX や HTML5 などの高度な動的 Web アプリケーションに特有の非同期イベントを手動で同期する必要はありません。Silk4J は、HTML や AJAX だけでなく、Apache Flex、Microsoft Silverlight、HTML5/AJAX など、すべての主要な Web 環境に対する同期モードをサポートします。詳細については、「*xBrowser のページ同期*」を参照してください。

統合オブジェクト モデル Silk4J では、さまざまなブラウザ上で幅広く実行するテストを作成、保守できます。すべてのブラウザに対応する統合オブジェクト モデルにより、ユーザーはテストの作成と保守を単一のブラウザを中心に行うことができます。Silk4J は、すべての他のブラウザ上のオブジェクトに同じ方法でアクセスできるようにすることで、さまざまなブラウザに対する回避策を探すことなく、時間を節約してテストの作成に集中できます。

クロス ブラウザ スクリプトの記録 スクリプトを記録すると、修正することなくすべての他のブラウザで再生できます。これにより、テスト スクリプトの作成と保守にかかる時間と労力は劇的に減少します。シミュレーションではなく、テストは実際のブラウザ上で実行されます。つまり、テストは、エンド ユーザーの動作とまったく同じように動作します。

Silk4J では、次のブラウザを使用して Web アプリケーションのテストを再生できます。

- Internet Explorer
- Mozilla Firefox（Microsoft Windows 上または macOS 上）
- Google Chrome（Microsoft Windows 上または macOS 上）
- Microsoft Edge
- Chrome for Android（Android デバイス上）
- Apple Safari（macOS 上または iOS デバイス上）
- 埋め込みブラウザ コントロール

注: 次のブラウザのいずれかを使用して Web アプリケーションのテストを記録できます。

- Internet Explorer
- Microsoft Edge
- Mozilla Firefox（Microsoft Windows 上または macOS 上）
- Google Chrome 50 以降（Microsoft Windows 上または macOS 上）
- モバイル ブラウザ（モバイル デバイス上）

クロス ブラウザー テスト用にスクリプトを記録する場合、Google Chrome、Mozilla Firefox、または Microsoft Edge を使用することを Micro Focus では推奨しています。 Internet Explorer を使って Silk4J で記録したスクリプトは、ほかのブラウザーで記録したスクリプトと若干異なる場合があります。

注: Web アプリケーションを記録または再生する前に、システムにインストールされているすべてのブラウザ アドオンを無効にします。Internet Explorer でアドオンを無効にするには、**ツール** > **インターネット オプション** をクリックし、**プログラム** タブをクリックし、**アドオンの管理** をクリックし、アドオンを選択してから **無効にする** をクリックします。

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

サンプル アプリケーション

Silk Test のサンプル Web アプリケーションには、以下の URL からアクセスします。

- *http://demo.borland.com/InsuranceWebExtJS/*
- *http://demo.borland.com/gmopost*
- *http://demo.borland.com/gmoajax*

テストを再生するブラウザーの選択

テストを再生するために使用するブラウザーを定義できます。

- Silk4J の UI からテストを実行する場合、**ブラウザーの選択** ダイアログ ボックスが表示され、このダイアログ ボックスで選択したブラウザーが使用され、テスト スクリプトで設定されているブラウザーを Silk4J は無視します。
- **ブラウザーの選択** ダイアログ ボックスが無効の場合 (**再び表示しない** をチェックした場合)、個々のテスト スクリプトのアプリケーション構成によってテストを実行するために使用するブラウザーが決定されます。

  注: ブラウザーの選択 ダイアログ ボックスを再び有効にするには、**Silk4J** > **アプリケーション構成の編集** をクリックして、**記録および再生前に 'ブラウザーの選択' ダイアログを表示する** チェックボックスをオンにします。
- スクリプトをコマンド ラインや CI サーバーから実行する場合は、スクリプトのアプリケーション構成で接続文字列を指定します。

  アプリケーション構成で指定したブラウザーは、*silktest.configurationName* 環境変数 を使用して上書き指定できます。
- Silk Central からテストを実行する場合は、テストするブラウザーそれぞれの構成を持つ構成スイートを作成します。そして、適切な構成名を指定します。詳細については、『*Silk Central ヘルプ*』を参照してください。

silktest.configurationName 環境変数を使用したブラウザーの設定の例

- ブラウザーとして Internet Explorer を使用する場合は、次のように入力します。

  ```
  SET  silktest.configurationName=InternetExplorer
  ```
- ブラウザーとして Microsoft Edge を使用する場合は、次のように入力します。

  ```
  SET  silktest.configurationName=Edge
  ```
- ブラウザーとして Mozilla Firefox を使用する場合は、次のように入力します。

  ```
  SET  silktest.configurationName=Firefox
  ```
- ブラウザーとして Google Chrome を使用する場合は、次のように入力します。

  ```
  SET  silktest.configurationName=GoogleChrome
  ```
- ブラウザーとして Mac 上の Apple Safari を使用する場合は、次のように入力します。

  ```
  SET  silktest.configurationName=host=10.0.0.1    -  Safari
  ```

  この例では、*host* は Apple Safari をテストする Mac です。ホストは、リモート ロケーションとして Silk4J がインストールされているマシンに接続されている必要があります。詳細については、「リモート ロケーションの編集」を参照してください。

- ブラウザーとして Android デバイス上の Google Chrome を使用する場合は、次のような接続文字列を使用します。たとえば、デバイス ID が 11111111 で、デバイスが IP アドレス 10.0.0.1 のリモート マシンに接続されている場合は、次のように入力します。

  ```
  SET
  silktest.configurationName="platformName=Android;deviceName=MotoG3;
  deviceId=11111111;host=10.0.0.1 - Chrome"
  ```

- ブラウザーとして iOS デバイス上の Apple Safari を使用する場合は、次のような接続文字列を使用します。たとえば、デバイス ID が 11111111 で、デバイスが IP アドレス 10.0.0.1 のリモート マシンに接続されている場合は、次のように入力します。

  ```
  SET silktest.configurationName="platformName=iOS;deviceName=iPad
  mini;deviceId=11111111;host=10.0.0.1"
  ```

  さらに、アプリケーション構成でブラウザーを指定する必要があります。

ヒント: すべての例で、環境変数 *silktest.configurationName* を設定する代わりに、Java システム プロパティ *-Dsilktest.configurationName* を設定して、ブラウザーを設定することもできます。たとえば、ブラウザーとして Mac 上の Apple Safari を使用する場合は、次のように入力することもできます。

```
-Dsilktest.configurationName=host=10.0.0.1 - Safari
```

コマンド ラインからテストを実行するには、次のように入力します。

```
java -cp "....¥junit.jar;....
¥org.hamcrest.core_1.3.0.v201303031735.jar;C:
¥Program Files (x86)¥Silk¥SilkTest¥ng¥JTF¥silktest-jtf-
nodeps.jar;....¥mytests¥bin" -
Dsilktest.configurationName="host=10.0.0.1 - Safari"
org.junit.runner.JUnitCore Tests
```

ヒント: Silk4J の UI から再生または記録を開始すると、**ブラウザーの選択** ダイアログ ボックスが開き、システムで現在利用可能なブラウザーのリストが表示されます。

xBrowser でのテスト オブジェクト

Silk4J では、以下のクラスを使用して Web アプリケーションがモデル化されます。

クラス	説明
BrowserApplication	Web ブラウザのメイン ウィンドウを公開し、タブ化するための方法を提供します。
BrowserObject	BrowserApplication に含まれるすべてのオブジェクトの基底クラスを表します。
BrowserWindow	タブおよび埋め込みブラウザ コントロールへのアクセスを提供し、異なるページに移動するための方法を提供します。
DomElement	Web アプリケーションの DOM ツリー (フレームを含む) を提供し、すべての DOM 属性へのアクセスを提供します。一部の DOM 要素では、特殊なクラスを使用できます。

クラス	説明
DomButton	Web ページの最上位のコンテナを表します。DomElement を介して DOM ツリーを公開します。
DomCheckBox	<input type='checkbox'> タグを使用して指定された、すべての DOM 要素を表します。
DomForm	<form> タグを使用して指定された、すべての DOM 要素を表します。
DomLink	<a> タグを使用して指定された、すべての DOM 要素を表します。
DomListBox	<select> タグを使用して指定された、すべての DOM 要素を表します。
DomRadioButton	<input type='radio'> タグを使用して指定された、すべての DOM 要素を表します。
DomTable	<table> タグを使用して指定された、すべての DOM 要素を表します。
DomTableRow	<tr> タグを使用して指定された、すべての DOM 要素を表します。
DomTextField	次のタグのいずれかを使用して指定された、すべての DOM 要素を表します。 • <input type='text'> • <input type='password'> • <input type='file'> • <textarea>

xBrowser オブジェクトに対するオブジェクト解決

xBrowser テクノロジ ドメインは動的オブジェクト解決をサポートします。

テストケースは、オブジェクトを検索し、識別するためにロケーター文字列を使用します。一般的なロケーターには、"//LocatorName[@locatorAttribute='value']" のようにロケーター名と少なくとも 1 つのロケーター属性が含まれます。

ロケーター名 Java SWT などの他の種類のテクノロジでは、テスト オブジェクトのクラス名を使用してロケーター名が作成されます。xBrowser では、DOM 要素のタグ名もロケーター名として使用できます。以下のロケーターは、同じ要素を示しています。

1. タグ名を使用した場合："//a[@href='http://www.microfocus.com']"
2. クラス名を使用した場合："//DomLink[@href='http://www.microfocus.com']"

再生速度を最適化するには、クラス名ではなくタグ名を使用します。

ロケーター属性 すべての DOM 属性は、ロケーター文字列属性として使用できます。たとえば、要素 <button automationid='123'>Click Me</button> はロケーター "//button[@automationid='123']" を使用して識別できます。

ロケーターの記録 Silk4J では、テストを記録したり、**オブジェクトの識別** ダイアログ ボックス を使用したりするときに、組み込みロケーター生成プログラムが使用されます。特定のアプリケーションの結果を向上するように、ロケーター生成プログラムを構成することができます。

xBrowser のページ同期

同期は、すべてのメソッド呼び出しの前後に自動的に実行されます。メソッド呼び出しは、同期条件が満たされるまで開始せず、終了もしません。

注: プロパティのアクセスは同期されません。

同期モード

Silk4J には、HTML および AJAX 用の同期モードがあります。

HTML モードを使用すると、すべての HTML ドキュメントが対話的な状態になることが保証されます。このモードでは、単純な Web ページをテストすることができます。Java Script が含まれるより複雑なシナリオが使用される場合は、以下の同期関数を使用して、手動でスクリプトを記述することが必要になることがあります。

- WaitForObject
- WaitForProperty
- WaitForDisappearance
- WaitForChildDisappearance

AJAX モードでは、ブラウザがアイドル状態に類似した状態になるまで待機します。このことは、AJAX アプリケーションまたは AJAX コンポーネントを含むページに対して特に効果的です。AJAX モードを使用すると、同期関数を手動で記述する必要がなくなるため、スクリプト（オブジェクトの表示または非表示を待機したり、特定のプロパティ値を待機するなど）の作成処理が大幅に簡略化されます。また、この自動同期は、スクリプトを手動で適用しないで記録と再生を正常に行うための基礎となります。

トラブルシューティング

AJAX の非同期の特性のため、ブラウザが完全にアイドル状態になることはありません。このため、Silk4J でメソッド呼び出しの終了が認識されず、特定のタイムアウト時間が経過したあとで、タイムアウト エラーが発生することがまれにあります。この場合は、少なくとも、問題が発生する呼び出しに対して、同期モードを HTML に設定する必要があります。

注: 使用するページ同期メソッドにかかわらず、Flash オブジェクトがサーバーからデータを取得し、計算を実行してデータをレンダリングするテストでは、手動でテストに同期メソッドを追加する必要があります。メソッドを追加しないと、Silk4J は、Flash オブジェクトが計算を完了するまで待機しません。たとえば、Thread.sleep(millisecs) を使用します。

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 **同期除外リスト** 設定で指定します。

監視ツールを使用して、同期の問題により再生エラーが発生するかどうかを判断します。たとえば、FindBugs（*http://findbugs.sourceforge.net/*）を使用して、AJAX 呼び出しが再生に影響を及ぼしているかどうかを判断できます。次に、問題が発生するサービスを **同期除外リスト** に追加します。

注: URL を除外すると、指定した URL を対象とする各呼び出しに対して同期が無効になります。その URL に対して必要な同期は、手動で呼び出す必要があります。たとえば、WaitForObject をテストに手動で追加する必要がある場合があります。手動で数多くの呼び出しを追加することを避けるために、可能なかぎり、最上位の URL ではなく、具体的に対象を絞って URL を除外します。

ページ同期設定の構成

スクリプト オプション ダイアログ ボックスでは、各テストのページ同期設定を個別に構成したり、すべてのテストに適用するグローバル オプションを設定したりできます。

URL を除外フィルタに追加するには、**スクリプト オプション** ダイアログ ボックスの **同期除外リスト** で URL を指定します。

テストの個別の設定を構成するには、テストを記録し、次にグローバル再生値を上書きするコードを挿入します。たとえば、タイム サービスを除外するには、以下のように入力します。

```
desktop.setOption(CommonOptions.OPT_XBROWSER_SYNC_EXCLUDE_URLS,
      Arrays.asList("timeService"));
```

- OPT_XBROWSER_SYNC_MODE
- OPT_XBROWSER_SYNC_EXCLUDE_URLS
- OPT_SYNC_TIMEOUT

xBrowser における API 再生とネイティブ再生の比較

Silk4J では、Web アプリケーション用に API 再生とネイティブ再生がサポートされています。アプリケーションでプラグインまたは AJAX を使用している場合は、ユーザーの入力そのものを使用します。アプリケーションでプラグインまたは AJAX を使用していない場合は、API 再生を使用することをお勧めします。

ネイティブ再生には以下のような利点があります。

- ネイティブ再生では、マウス ポインタを要素上に移動し、対応する要素を押すことによって、エージェントはユーザー入力をエミュレートします。この結果、再生はほとんどのアプリケーションで変更なしで動作します。
- ネイティブ再生では、Flash や Java アプレットなどのプラグイン、および AJAX を使用するアプリケーションをサポートしていますが、高レベルの API 記録はサポートしていません。

API 再生には以下のような利点があります。

- API 再生では、Web ページが onmouseover や onclick などの DOM イベントによって直接実行されます。
- API 再生を使用するスクリプトでは、ブラウザをフォアグラウンドで実行する必要はありません。
- API 再生を使用するスクリプトでは、要素をクリックする前に、要素が表示されるようにスクロールする必要はありません。
- 一般的に、高レベルのユーザー入力は再生中にポップアップ ウィンドウやユーザー対話の影響を受けないため、API スクリプトの信頼性は高くなります。
- API 再生は、ネイティブ再生よりも高速です。

スクリプト オプション ダイアログ ボックスを使用して、記録する関数の種類を構成したり、ユーザーの入力そのものを使用するかどうかを指定したりできます。

API 再生とネイティブ再生の関数の違い

DomElement クラスには、API 再生とネイティブ再生に対して異なる関数が備えられています。

以下の表に、API 再生とネイティブ再生で使用する関数を示します。

	API 再生	ネイティブ再生
マウス操作	DomClick	Click
	DomDoubleClick	DoubleClick
	DomMouseMove	MoveMouse
		PressMouse
		ReleaseMouse
キーボード操作	使用不可	TypeKeys

	API 再生	ネイティブ再生
特殊な関数	Select SetText など	使用不可

マウス移動の詳細設定

マウス移動イベントを使用する Web アプリケーション、Win32 アプリケーション、および Windows Forms アプリケーションでマウス移動操作を記録するかどうかを指定します。たとえば、Apache Flex や Swing など、xBrowser テクノロジ ドメインの子ドメインのマウス移動イベントを記録することはできません。

1. **Silk4J** > **オプションの編集** をクリックします。
2. **オプション** メニュー ツリーの **記録** の隣にあるプラス記号（+）をクリックします。 **記録** オプションが右側のパネルに表示されます。
3. **記録** をクリックします。
4. マウス移動操作を記録するには、OPT_RECORD_MOUSEMOVES オプションをオンにします。
   Silk4J では、スクリプトを短くするために、マウスが置かれた要素またはその親が変化するマウスの移動イベントのみが記録されます。
5. マウスの移動操作を記録する場合、MoveMouse 操作が記録される前に、どのくらいの間マウスが不動状態になければならないかを、**マウスの移動記録遅延** テキスト ボックスにミリ秒単位で指定します。
   デフォルト値は、200 に設定されています。
   マウスの移動操作は、この時間、マウスが静止している場合にのみ記録されます。遅延を短くすると、予期しないマウスの移動操作が増加します。遅延を長くすると、操作を記録するためにマウスを静止しておく必要があります。
6. **OK** をクリックします。

xBrowser のブラウザ構成の設定

いくつかのブラウザ設定は、テストを継続的に安定して実行するのに役立ちます。設定を変更しなくても Silk4J は動作しますが、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	読み込みに時間を要する Web ページではなく、about:blank をホーム ページとして使用する
ブラウザの予期しない動作を回避する	• ポップアップ ウィンドウや警告ダイアログ 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
ブラウザの誤動作を防止する	不要なサードパーティ製プラグインを無効にする

以下のセクションでは、対応するブラウザにおけるこれらの設定場所について説明します。

Internet Explorer

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。以下の表に、調整できるオプションの一覧を示します。

タブ	オプション	設定	コメント
全般	ホーム ページ	about:blank に設定します。	新しいタブの起動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します。 • 新しいタブを作成したとき、新しいタブに切り替えます。	• 予期しないダイアログ ボックスが表示されないようにします。 • 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場合があります。
プライバシー	ポップアップ ブロック	ポップアップ ブロックを無効にします。	Web サイトで新しいウィンドウを開くことができることを確認します。
コンテンツ	オートコンプリート	完全にオフにします。	• 予期しないダイアログ ボックスが表示されないようにします。 • キー入力するときに予期しない動作を回避します。
プログラム	アドオンの管理	最低限必要なアドオンのみを有効にします。	• サードパーティ製アドオンにはバグが含まれていることがあります。 • Silk4J と互換性がない可能性があります。
詳細設定	設定	• **Internet Explorer の更新について自動的に確認する** を無効にします。 • **スクリプトのデバッグを使用しない (Internet Explorer)** を有効にします。 • **スクリプトのデバッグを使用しない (その他)** を有効にします。 • **自動クラッシュ回復機能を有効にする** を無効にします。 • **スクリプト エラーごとに通知を表示する** を無効にします。 • すべての **...警告する** 設定を無効にします。	予期しないダイアログ ボックスが表示されないようにします。

注: 100% 以外の拡大レベルを使用して Internet Explorer で Web アプリケーションを記録すると、期待通り機能しない可能性があります。Internet Explorer で Web アプリケーションに対する操作を記録する前に、拡大レベルを 100% に設定してください。

Mozilla Firefox

Mozilla Firefox のブラウザ設定を変更する必要はありません。 Silk4J により、適切なコマンドライン パラメータが指定され、自動的に Mozilla Firefox が起動します。

注: Web アプリケーションのテスト時に予期しない動作を避けるため、Mozilla Firefox の自動更新を無効にします。詳細については、『*Stop automatic updates*』を参照してください。

Google Chrome

Google Chrome のブラウザ設定を変更する必要はありません。Silk4J により、適切なコマンドライン パラメータが指定され、自動的に Google Chrome が起動します。

注: Web アプリケーションのテスト時に予期しない動作を避けるため、Google Chrome の自動更新を無効にします。詳細については、『*Turning Off Auto Updates in Google Chrome*』を参照してください。

ロケーター生成プログラムを xBrowser 用に構成する

Open Agent には、ロケーターが記録時に一意となり、メンテナンスが容易になるようにする、高度なロケーター生成メカニズムが備えられています。 使用するアプリケーションやフレームワークに応じて、最適な結果を得るためにデフォルト設定を変更できます。

頻繁には変更されない属性を利用して、適切に定義されたロケーターでは、メンテナンス作業が少なく抑えられます。 カスタム属性を使用すると、caption や index などの他の属性を使用するよりも高い信頼性を得ることができます。これは、caption はアプリケーションを他の言語に翻訳した場合に変更され、index は他のオブジェクトが追加されると変更される可能性があるためです。

最適な結果を得るために、テストで利用する要素にカスタム オートメーション ID を追加することもできます。 Web アプリケーションの場合は、利用した要素に <div myAutomationId=”my unique element name” /> のような属性を追加できます。 この手法によって、ロケーターの変更に伴うメンテナンス作業を回避することができます。

1. **Silk4J** > **オプションの編集** をクリックしてから、 **カスタム属性** タブをクリックします。
2. カスタム オートメーション ID を使用する場合、 **テクノロジー・ドメインを選択します** リスト ボックスから、 **xBrowser** を選択してから、ID をリストに追加します。

   カスタム属性リストには、ロケーターに適した属性が含まれます。 カスタム属性が利用可能な場合は、ロケーター生成プログラムは、他の属性の前にそれらの属性を使用します。 リストの順番は、ロケーター生成プログラムが使用する属性の優先順位を表しています。 指定した属性が選択したオブジェクトに対して利用できない場合は、 Silk4J は xBrowser のデフォルトの属性を使用します。
3. **ブラウザー** タブをクリックします。
4. **ロケーター属性名除外リスト** グリッドで、記録中に無視する属性名を入力します。

   たとえば、このリストを使用して、size、width、height、style などの頻繁に変更される属性を指定します。 ロケーター属性名除外リストでは、ワイルドカード‘*’および‘?’を使用できます。

   複数の属性名を指定する場合にはコンマで区切ります。
5. **ロケーター属性値除外リスト** グリッドで、記録中に無視する属性値を入力します。

   一部の AJAX フレームワークでは、ページが再読み込みされるたびに変わる属性値が生成されます。 このリストを使用して、そのような値を無視します。 このリストでワイルドカードを使用することもできます。

   複数の属性値を指定する場合にはコンマで区切ります。
6. **OK** をクリックします。

   以上で、テスト ケースを記録したり、手動で作成する準備ができました。

リモート デスクトップ ブラウザーの接続文字列

接続文字列 は、テストに使用するリモート デスクトップ ブラウザーを指定します。リモート ブラウザーで Web アプリケーションをテストする場合、Silk4J はリモート ロケーションに接続するために接続文字列を使用します。接続文字列は、アプリケーション構成の主要な一部です。テストする Web アプリケーションを構成するときに、接続文字列は設定されます。接続文字列を変更するには、**アプリケーション構成の編集** ダイアログ ボックスを使用します。

リモート ブラウザーで Web アプリケーションをテストする場合、接続文字列にはリモート マシンの IP アドレスまたはホスト名を意味する host だけが含まれます (10.0.0.1 など)。正しいブラウザーを選択するために、Silk4J はブラウザーの種類と共に接続文字列を使用します。ブラウザーの種類は、**アプリケーション構成の編集** ダイアログ ボックスでも指定できます。

ホスト名は、大文字と小文字は区別されません。

注: リモート デスクトップ ブラウザーのテストは、リモート Microsoft Windows マシン上の Microsoft Edge およびリモート Mac 上の Apple Safari に対してのみサポートされます。

接続文字列の例

```
"host=10.0.0.1"
```

リモート Windows マシン上でのブラウザーのテスト

Microsoft Edge、Google Chrome、Mozilla Firefox をリモート Windows マシン上でテストするには、リモート マシン上に Silk4J をインストールする必要があります。Microsoft Edge は、Microsoft Windows 10 マシン上でのみ実行できます。Silk4J は、リモート Windows マシン上の Mozilla Firefox 55 以前のテストをサポートします。

1. ブラウザーのテストを実行するローカル マシン上で、リモート Windows マシンをリモート ロケーションとして追加します。
   詳細については、「リモート ロケーションの編集」を参照してください。
2. Silk Test Information Service がすでにリモート マシン上で管理者権限で実行されている場合は、Silk Test Information Service を無効化します。
   a) リモート マシンに管理者としてログインします。
   b) **コントロール パネル** (アイコン表示) を開きます。
   c) **管理ツール** をクリックします。
   d) **サービス** をダブルクリックします。
   e) Silk Test Information Service をダブルクリックします。
   f) サービスが実行中の場合、**停止** をクリックして、サービスの状態が 停止 になるまで待機します。
   g) **スタートアップの種類** を **無効** に変更します。
   h) **OK** をクリックします。
3. Silk Test Information Service を標準の整合性レベル（管理者権限無し）で開始します。
   a) ファイル エクスプローラーを開き、%OPEN_AGENT_HOME%/InfoService に移動します。
      たとえば、C:¥Program Files (x86)¥Silk¥SilkTest¥ng¥InfoService になります。
   b) infoservice_start.bat をダブルクリックします。
4. 以上で、リモート マシン上のブラウザーを **アプリケーションの選択** ダイアログ ボックスで選択できるようになります。

注: Silk Test Information Service が、サービス セッションではなくユーザー セッションで実行されていることを確認します。Silk Test Information Service は、UI 操作が有効でなければなりません。セッション 0 であるサービス セッションは、UI 操作が有効ではありません。

Mac 上の Google Chrome または Mozilla Firefox のテスト

リモート macOS マシン上の Google Chrome または Mozilla Firefox をテストするには、リモート マシン上に Silk Test Information Service をインストールする必要があります。詳細については、「*Silk Test Information Service を Mac にインストールする*」を参照してください。

1. Google Chrome または Mozilla Firefox をテストするローカル マシン上で、リモート macOS マシンをリモート ロケーションとして追加します。
   詳細については、「リモート ロケーションの編集」を参照してください。
2. 以上で、リモート macOS マシン上の Google Chrome または Mozilla Firefox を **アプリケーションの選択** ダイアログ ボックスで選択できるようになります。

注: Silk Test Information Service が、サービス セッションではなくユーザー セッションで実行されていることを確認します。Silk Test Information Service は、UI 操作が有効でなければなりません。セッション 0 であるサービス セッションは、UI 操作が有効ではありません。

Google Chrome 上でテストを実行する際の前提条件と制限事項についての情報は、「*Google Chrome を使用したテスト*」セクションにあるトピックを参照してください。Mozilla Firefox 上でテストを実行する際の前提条件と制限事項についての情報は、「*Mozilla Firefox を使用したテスト*」セクションにあるトピックを参照してください。

WebDriver ベースのブラウザーのケイパビリティの設定

WebDriver ベースのブラウザー上で Web アプリケーションをテストする場合は、ケイパビリティを設定してブラウザー セッションの設定をカスタマイズできます。

Silk4J では、次のブラウザーの種類に対する接続文字列で WebDriver ケイパビリティを指定できます。

- Google Chrome
- Mozilla Firefox

Mozilla Firefox 48 以降で利用可能なオプションとケイパビリティについての詳細は、『*https://github.com/mozilla/geckodriver*』を参照してください。Google Chrome で利用可能なオプションとケイパビリティについての詳細は、『*Capabilities & ChromeOptions*』を参照してください。

Silk4J でケイパビリティを設定するには：

1. ケイパビリティを変更する Web アプリケーションのプロジェクトを選択します。
2. プロジェクトの基本状態の接続文字列を編集します。
   接続文字列は、次の方法で編集できます。
   - **アプリケーション構成の編集** ダイアログ（カスタマイズしたブラウザーに対する操作を記録する場合など）
   - スクリプト（カスタマイズしたブラウザーに対してテスト スクリプトを実行するだけの場合）

   詳細については、「基本状態」を参照してください。
3. スクリプトを実行して、指定したオプションとケイパビリティでブラウザーを開始します。

使用例

Mozilla Firefox から実行可能ファイルを自動的にダウンロードするには、スクリプトの基本状態に次のコードを追加します。

```
baseState.setConnectionString(
   "moz:firefoxOptions="
   + "{"
   + " ¥"prefs¥": {"
   + "          ¥"browser.download.folderList¥": 2,"
   + "          ¥"browser.helperApps.neverAsk.saveToDisk¥": ¥"application/
octet-stream¥"
   + "       }"
   + "};");
```

Mozilla Firefox のダウンロード フォルダーを指定するには、スクリプトの基本状態に次のコードを追加します。

```
baseState.setConnectionString("moz:firefoxOptions={¥"prefs¥":
{ ¥"browser.download.dir¥" : ¥"C:/Download¥"} };");
```

Mozilla Firefox のコマンド ライン引数を設定するには、スクリプトの基本状態に次のコードを追加します。

```
baseState.setConnectionString("moz:firefoxOptions={¥"args¥": [¥"--devtools
¥"]};");
```

Google Chrome から実行可能ファイルを指定したフォルダーに自動的にダウンロードするには、スクリプトの基本状態に次のコードを追加します。

```
baseState.setConnectionString(
   "chromeOptions="
   + "{"
   + " ¥"prefs¥": {"
   + "
¥"profile.default_content_setting_values.automatic_downloads¥":1,"
   + "                    ¥"download.default_directory¥":¥"c:¥¥¥Download¥","
   + "                    ¥"download.prompt_for_download¥":false"
   + "                }"
   + "};");
```

Google Chrome のパスワード マネージャーにメッセージを表示させないようにするには、スクリプトの基本状態に次のコードを追加します。

```
baseState.setConnectionString(
   "chromeOptions="
   + "{"
   + " ¥"args¥":[¥"--disable-save-password-bubble¥"],"
   + " ¥"prefs¥":  {"
   + "                    ¥"profile.password_manager_enabled¥": false,"
   + "                    ¥"credentials_enable_service¥": false"
   + "                }"
   + "};");
```

Mac 上の Apple Safari を使用したテスト

このセクションでは、Silk4J がインストールされている Windows マシンと接続し、Mac マシン上の Apple Safari をテストすることによって、クロス ブラウザー テスト セットを拡張する方法について説明します。

Mac 上の Apple Safari を使用したテストにおける前提条件

Mac 上の Apple Safari を使用してテストを行う前に、次の前提条件を満たしていることを確認してください。

- Mac がリモート ロケーションとして Silk4J がインストールされている Windows マシンに接続されている。詳細については、「リモート ロケーションの編集」を参照してください。
- Apple Safari 9 でテストする場合、 Apple Safari 用の WebDriver 拡張である SafariDriver が Mac 上にインストールされている（SafariDriver は従来のクライアント/サーバーの関係を逆転し、WebSocket を使用して WebDriver クライアントと通信します）。Apple Safari 10.1 を使用する場合、Safari はドライバーの実装を組み込みで備えています。
- Java JDK が Mac 上にインストールされている。
- Information Service が Mac 上にインストールされている。Information Service に必要なファイルを取得するには、Silk Test インストーラを使用します。詳細については、「*Silk Test Information Service を Mac にインストールする*」を参照してください。
- Apple Safari 上でテストを実行するには、Information Service をインストールしたユーザーが Mac にログインしている必要があります。

  ヒント: Micro Focus では、起動時に正しいユーザーで Mac に自動的にログインするように設定することを推奨します。詳細については、「起動時に自動的にログインするように *Mac* を設定する」を参照してください。
- Mac 上の Apple Safari に対して無人テストを実行する場合は、**システム環境設定** の **省エネルギー** ペインで、次の設定を行います。
  - **ディスプレイをオフにするまでの時間** を **しない** に設定します。

- **ディスプレイがオフのときにコンピュータを自動でスリープさせない** チェック ボックスをオンにします。

  注: これらの設定は、**Silk Test Configuration Assistant** を使用すると簡単に行えます。Mac 上で **Configuration Assistant** を開くには、ステータス メニューの Silk Test アイコンをクリックして、**Configuration Assistant** を選択します。
- Mac 上の Apple Safari に対して無人テストを実行するには、スクリーン セーバーを無効にします。

  1. **システム環境設定** > **デスクトップとスクリーンセーバ** を開きます。
  2. **スクリーンセーバ** タブをクリックします。
  3. **開始までの時間** を **開始しない** に設定します。

  注: これらの設定は、**Silk Test Configuration Assistant** を使用すると簡単に行えます。Mac 上で **Configuration Assistant** を開くには、ステータス メニューの Silk Test アイコンをクリックして、**Configuration Assistant** を選択します。
- Apple Safari 10.1 を使用してテストする場合は、Safari 開発メニューを有効にします。**Safari** > **環境設定** を選択して、**詳細** をクリックし、**メニューバーに '開発' メニューを表示** をオンにします。
- Apple Safari 10.1 を使用してテストする場合は、リモート オートメーションを有効にします。Safari 開発メニューから、**リモート オートメーションを許可** をオンにします。
- 初めて Apple Safari 10.1 でテストを実行する際に、パスワード入力が必要になる場合があります。

Apple Safari のテストの準備

Apple Safari 10.1 以降で Web アプリケーションをテストする場合、**Silk Test Configuration Assistant** を使うと、簡単に Apple Safari を設定できます。Mac 上で **Configuration Assistant** を開くには、ステータス メニューの Silk Test アイコンをクリックして、**Configuration Assistant** を選択します。または、「*Mac 上の Apple Safari を使用したテストにおける前提条件*」に書かれている要件を満たすよう、次の手順を実行します。

1. Apple Safari のリモート オートメーションを有効化するために、**開発** メニューを開き、**リモート オートメーションを許可** をオンにします。

   開発 メニューは、デフォルトでは表示されません。このメニューを開くには：

   a) **Safari** メニューから **環境設定** を選択します。
   b) **環境設定** ウィンドウで、**詳細** タブを選択します。
   c) **メニュー バーに "開発" メニューを表示** チェック ボックスをオンにします。
   d) **環境設定** ウィンドウを閉じます。
2. Apple Safari 上で最初にテストを実行すると、ブラウザー ウィンドウが自動テストによってリモート制御されていることを示すダイアログ ボックスが表示されます。**セッションを続ける** をクリックします。

Apple Safari と Selenium WebDriver についての詳細は、『*https://webkit.org/blog/6900/webdriver-support-in-safari-10/*』を参照してください。

Silk Test Information Service を Mac にインストールする

注: Information Service を Mac にインストールするには、Mac の管理者権限が必要です。

Mac 上の Apple Safari や、Mac に接続されている iOS や Android デバイス上のモバイル アプリケーションに対するテストを作成して実行するには、Mac に Silk Test Information Service (Information Service) をインストールしてから、**リモート ロケーション** ダイアログ ボックスを使用して、Silk4J をインストールした Windows マシンと Mac を接続する必要があります。

Information Service を Mac にインストールするには：

1. Java JDK が Mac 上にインストールされていることを確認します。
2. iOS デバイス上でモバイル アプリケーションをテストする場合は、Xcode が Mac 上にインストールされていることを確認します。

3. Information Service セットアップ ファイル（SilkTestInformationService<バージョン>-<ビルド番号>.pkg にアクセスします。
   - Silk Test のインストール時に Information Service セットアップ ファイルをダウンロードした場合は、Silk Test インストール ディレクトリ（C:¥Program Files (x86)¥Silk¥SilkTest など）の macOS フォルダを開きます。
   - Silk Test のインストール時に Information Service セットアップ ファイルをダウンロードしなかった場合は、*Micro Focus SupportLine* からセットアップ ファイルをダウンロードできます。
4. SilkTestInformationService<バージョン>-<ビルド番号>.pkg ファイルを Mac にコピーします。
5. SilkTestInformationService<バージョン>-<ビルド番号>.pkg を実行して、Information Service をインストールします。
6. インストール ウィザードの指示に従います。
7. パスワードを尋ねられた場合、現在サインインしている Mac ユーザーのパスワードを入力します。
8. Apple Safari が開き、SafariDriver を信頼するかどうかを尋ねるメッセージ ボックスが表示されたら、**信頼** をクリックします。

   注: リモート接続ではなく、直接 Mac にログインしている場合には、SafariDriver だけをインストールできます。

インストールを完了するために、現在の Mac ユーザーをログアウトします。Information Service が正しくインストールされていることを確認するには、Mac にログインし、画面の右上隅にある Silk Test アイコンをクリックして、利用可能なデバイスとブラウザーを表示させます。

ヒント: Silk Test アイコンが表示されない場合は、Mac を再起動してください。

Apple Safari を使用したテストの制限事項

以下に、Mac 上の Apple Safari を使用してテストする際の既知の制限事項を一覧します。

- 次のクラス、インターフェイス、メソッド、プロパティは、Mac 上の Apple Safari を使用した Web アプリケーションのテストでは現時点ではサポートされません：
  - BrowserApplication クラス。
    - clearCache メソッド
    - closeOtherTabs メソッド
    - closeTab メソッド
    - existsTab メソッド
    - getHorizontalScrollbar メソッド
    - getNextCloseWindow メソッド
    - getSelectedTab メソッド
    - getSelectedTabIndex メソッド
    - getSelectedTabName メソッド
    - getTabCount メソッド
    - getVerticalScrollbar メソッド
    - isActive メソッド
    - minimize メソッド
    - openContextMenu メソッド
    - openTab メソッド
    - restore メソッド
    - selectTab メソッド
    - setActive メソッド
    - windowState プロパティ
  - BrowserWindow クラス。

    - acceptAlert メソッド
    - dismissAlert メソッド
    - getAlertText メソッド
    - isAlertPresent メソッド
    - mouseMove メソッド
    - pressKeys メソッド
    - pressMouse メソッド
    - releaseKeys メソッド
    - releaseMouse メソッド
  - IMoveable クラス。
    - getFocus メソッド。
- Silk4J は、typeKeys メソッドで **CMD** キーをサポートしません。
- Silk4J は、Apache Flex のテストをサポートしません。
- Silk4J は、Apple Safari 上の JavaScript ソースを使った iframe のテストをサポートしません。
- Apple Safari 上で HTTPS を使用したセキュアな Web アプリケーションをテストするには、必要なすべてのサーバー証明書が信頼済みであることを確認ください。
- Silk4J は、Apple Safari のネイティブ サポートは提供しません。内部 Apple Safari 機能をテストすることはできません。たとえば、テストで、ナビゲーション バーにテキストを追加して、現在表示されている Web ページを変更することはできません。回避策として、API コールを使用して Web ページ間を移動できます。
- Silk4J は、Apple Safari に対する JavaScript ダイアログ API 関数をサポートしません。回避策として、このような関数が無視されるようにパッチを当てることができます。詳細については、「*https://groups.google.com/forum/#!topic/selenium-developer-activity/qsovJw93g9c*」を参照してください。
- Silk4J は、Apple Safari でのテキスト解決をサポートしません。

  テキスト解決は次のメソッドを含みます。

  - textCapture
  - textClick
  - textExists
  - textRectangle
- Silk4J は、Apple Safari のタブをサポートしません。
- 複数ウィンドウのアプリケーションをテストするには、Apple Safari のポップアップ ブロックを解除してください。解除する場合は、Apple Safari を起動し、**Safari 環境設定** > **セキュリティ** > **ポップアップウィンドウを開かない** のチェックを外します。
- Silk4J は、パスワードを保存するダイアログ ボックスのテストをサポートしません。このダイアログ ボックスを表示しないようにする場合は、Apple Safari を起動し、**Safari 環境設定** > **自動入力** に移動して **ユーザー名とパスワード** チェック ボックスのチェックを外します。
- Silk4J は、Apple Safari では XPath 式のプロパティをサポートしません。XPath 式では、属性のみがサポートされます。
- Silk4J は、Content-Security-Policy HTTP ヘッダーを含んだ Web アプリケーションのテストをサポートしません。
- Apple Safari 10.1 を使用する場合、Silk4J はブラウザーの前に戻る操作をサポートしません。
- Apple Safari 10.1 を使用する場合、Silk4J は typeKeys メソッドでの Ctrl キーの使用をサポートしません。
- Apple Safari 10.1 を使用する場合、Silk4J は Frame および IFrame 内では DOM 操作のみをサポートします。
- Apple Safari 10.1 を使用する場合、Silk4J は Frame および IFrame での移動操作をサポートしません。
- Apple Safari 10.1 を使用する場合、Silk4J は記録時の直接スクロールをサポートしません。回避策として、executeJavaScript メソッドを使用できます。

複数の Apple Safari テストの同時実行

Apple Safari 上でテストを実行するには、Silk Test がインストールされた Windows マシンに接続された Mac が必要です。複数の Apple Safari テストを Apple Safari 上で実行する場合、これらのテストは同じ Mac 上で同時に実行されることになります。

注: Mac 上の Apple Safari に対して実行されるテストそれぞれが、Apple Safari のインスタンスを個々に開きます。数多くの Apple Safari のインスタンスが同時に実行すると、Mac のパフォーマンスが低下する可能性があります。

Silk Test Information Service を Mac からアンインストールする

Mac 上の Apple Safari に対するテストを実行する必要がなくなった場合など、次の手順で Silk Test Information Service を Mac からアンインストールすることができます。

1. uninstallInfoService.sh のような新しいシェル ファイルを作成します。
2. 新しいファイルに以下のコードを入力します。

```
#!/bin/sh

if launchctl list | grep com.borland.infoservice ; then
   launchctl unload /Library/LaunchAgents/com.borland.infoservice.plist
   echo "unloading Launch Daemon"
fi

if [ -d "/Applications/Silk" ]
then
   sudo rm -rf /Applications/Silk
fi

if [ -f "/Library/LaunchAgents/com.borland.infoservice.plist" ]
then
   sudo rm /Library/LaunchAgents/com.borland.infoservice.plist
fi

if [ -f "/usr/local/bin/ideviceinstaller" ]
then
   sudo rm /usr/local/bin/ideviceinstaller
fi

exit 0
```

3. コマンドラインで chmod +x uninstallInfoService.sh を実行し、シェル ファイルの実行可能にします。
4. コマンドラインからシェル ファイルを実行します。

Google Chrome を使用したテスト

このセクションでは、Google Chrome を使用してテストすることによって、クロス ブラウザー テスト セットを拡張する方法について説明します。

Silk4J は、Google Chrome 50 以降で操作を記録したり、テストを再生することをサポートします。また、Google Chrome 50 より前のバージョンでテストを再生したり、ロケーターを記録することをサポートします。

- Google Chrome 50 以降でテストを開始するときに、Google Chrome のインスタンスが実行していない場合、Silk4J は Google Chrome の新しいインスタンスを開始します。新しいブラウザーは、アドオン無しのキャッシュを空にした状態の一時プロファイルを使用します。
- Google Chrome 50 以降のインスタンス上でのテストを開始するときに、すでに実行中であれば、Silk4J はそのインスタンスが最初に開始されたときに使用されたコマンド ライン引数と同じ引数で

Google Chrome を再起動します。この再起動は、Silk4J オートメーション サポートを有効化するために必要です。

- Google Chrome 50 以降でテストするとき、Google Chrome インスタンスは、Open Agent のシャットダウン時または、Google Chrome 外のほかのアプリケーションのテストを開始するときに閉じられます。

ヒント: Google Chrome 50 以降を使用して既存のテスト スクリプトを実行する場合は、基本状態を使用して、URL へ移動するコマンドをテスト スクリプトに追加することを、Micro Focus はお勧めします。

例 1

Google Chrome 50 以降の実行中のインスタンスが、コマンド「C:/Program Files (x86)/Google/Chrome/Application/chrome.exe www.borland.com」で最初に起動されていた場合、Google Chrome は再起動後に *www.borland.com* を開きます。

例 2

Google Chrome 50 以降の実行中のインスタンスが、コマンド「C:/Program Files (x86)/Google/Chrome/Application/chrome.exe」で最初に起動されていた場合、Google Chrome は再起動後に *about:blank* を開きます。

Google Chrome を使用したテスト再生の前提条件

コマンド ライン パラメータ

Google Chrome を使用してテストの再生やロケーターの記録を行うと、次のコマンドで Google Chrome が起動されます。

```
%LOCALAPPDATA%¥Google¥Chrome¥Application¥chrome.exe
  --enable-logging
  --log-level=1
  --disable-web-security
  --disable-hang-monitor
  --disable-prompt-on-repost
  --dom-automation
  --full-memory-crash-report
  --no-default-browser-check
  --no-first-run
  --homepage=about:blank
  --disable-web-resources
  --disable-preconnect
  --enable-logging
  --log-level=1
  --safebrowsing-disable-auto-update
  --test-type=ui
  --noerrdialogs
  --metrics-recording-only
  --allow-file-access-from-files
  --disable-tab-closeable-state-watcher
  --allow-file-access
  --disable-sync
  --testing-channel=NamedTestingInterface:st_42
```

ウィザードを使用してアプリケーションに追加する場合は、これらのコマンド ライン パラメータは、基本状態に自動的に追加されます。テストを開始したときに、適切なコマンド ライン パラメータなしで Google Chrome のインスタンスがすでに実行されている場合、Silk4J は Google Chrome を終了して、コ

マンド ライン パラメータを使用してブラウザを再起動しようとします。ブラウザを再起動できない場合は、エラー メッセージが表示されます。

注: クロスドメインのドキュメントを記録または再生する場合は、コマンド ライン パラメータ disable-web-security が必要です。

注: ローカル ファイル システムに保存された Web アプリケーションをテストするには、Google Chrome で chrome://extensions に移動し、**Silk Test Chrome Extension** の **ファイルの URL へのアクセスを許可する** チェックボックスをオンにします。

Google Chrome 拡張のテスト

Google Chrome 拡張 (アドオン) を Silk4J を使ってテストするには、以下の 2 つの手段のいずれかを使用できます。

Google Chrome の起動時に .crx ファイルとして拡張をインストールする

.crx ファイルとしてインストールされた Google Chrome 拡張をテストするには、基本状態に次のコマンド ラインを追加します。

```
chrome.exe  --load-extension=C:/myExtension/myExtension.crx
```

注: Google Chrome には単一の拡張のみを .crx ファイルとしてインストールできます。Google Chrome に複数の拡張をインストールするには、複数の .crx ファイルをカンマ区切りで指定します。例：

```
chrome.exe  --load-extension=C:/myExtension/myExtension.crx,C:/myExtension2/myExtension2.crx
```

ブラウザーのコマンド ライン引数指定方法の詳細については、「基本状態の変更」を参照してください。

プロファイルに拡張を追加する

Google Chrome ユーザー データ ディレクトリに拡張を追加し、そのプロファイルをテストに使用します。詳細については、「ユーザー データ ディレクトリを使用した *Google Chrome* のテスト」を参照してください。

ユーザー データ ディレクトリを使用した Google Chrome のテスト

ホーム ページ、使用するツールバーの設定や保存したパスワード、ブックマークなど、Google Chrome で行ったすべての変更は、ユーザー データ ディレクトリと呼ばれる特別なフォルダに格納されます。

Silk4J を使用して、テスト対象アプリケーションの基本状態にユーザー データ ディレクトリへのパスを指定することによって、Google Chrome ユーザー データ ディレクトリをテストできます。次のコマンド ラインには、プロファイルへのパスが含まれています。

```
chrome.exe  "--user-data-dir=C:/Users/MyUser/AppData/Local/Google/Chrome/User  Data"
```

サンプル Web アプリケーションのプロファイル ディレクトリを設定するには、次のコードを使用できます。

```
BrowserBaseState baseState = new BrowserBaseState(BrowserType.GoogleChrome,
"demo.borland.com¥InsuranceWebExtJS");
String myProfileDir = "--user-data-dir=C:¥¥temp¥¥SilkTest ¥"--profile-directory=my user¥"";
baseState.setCommandLineArguments(myProfileDir);
desktop.executeBaseState(baseState);
```

注: Google Chrome が Silk4J によって起動されるときには、空のユーザー データ ディレクトリが使用されます。これにより、クリーンな状態でテストが開始されることになります。

Google Chrome を使用したテストの制限事項

以下のリストに、ローカル Windows マシン上で Google Chrome を使用した場合の、テストの再生とロケーターの記録の既知の制限事項をリストします。

- Silk Test は、Google Chrome を使用した xBrowser ドメインの子テクノロジ ドメインのテストをサポートしていません。たとえば、Apache Flex または Microsoft Silverlight は Google Chrome ではサポートされていません。
- Silk4J は、HTTP 基本認証ダイアログのテストの記録をサポートしません。
- Silk Test は、Google Chrome のネイティブ サポートは提供しません。内部 Google Chrome 機能をテストすることはできません。たとえば、テストで、Win32 でナビゲーション バーにテキストを追加して現在表示されている Web ページを変更することはできません。回避策として、API コールを使用して Web ページ間を移動できます。Silk Test は、Alert API を使用した警告および類似のダイアログ ボックスの処理をサポートします。
- Silk4J は、Google Chrome でのテキスト解決をサポートしません。

  テキスト解決は次のメソッドを含みます。

  - textCapture
  - textClick
  - textExists
  - textRectangle
- Silk4J は、IMoveable クラスの getFocus メソッドをサポートしません。
- Silk Test は、Google Chrome メニューを使用して Google Chrome の **印刷** ダイアログ ボックスが開かれたことは認識しません。Google Chrome でダイアログ ボックスを開く動作を追加してテストするには、TypeKeys メソッドを使用して **Ctrl+Shift+P** を送信する必要があります。Internet Explorer はこのショートカットを認識しません。したがって、最初に Internet Explorer でテストを記録してから、手動で **Ctrl+Shift+P** を押す操作をテストに追加する必要があります。
- Google Chrome の複数ウィンドウの同時テストは、最初の Google Chrome ウィンドウから AUT 自身が追加のウィンドウを開いた場合にのみサポートされます。追加の Google Chrome ウィンドウを手動で開いた場合、Silk4J はこれらの Google Chrome ウィンドウ上の要素を解決できません。たとえば、記録中に AUT のリンクやボタンをクリックして開かれた Google Chrome ウィンドウ上の要素は Silk4J は解決できますが、記録中に **CTRL+N** を押して開かれた Google Chrome ウィンドウ上の要素は Silk4J は解決できません。
- Google Chrome 49 以前を使用する場合、Internet Explorer を使用してテストを再生する場合、executeJavaScript をテストするために次のコードを使用できます。

  ```
  // Java code
  desktop.<BrowserWindow> find("//BrowserWindow")
    .executeJavaScript("function foo() { alert('Silk Test'); }");
  desktop.<BrowserWindow> find("//BrowserWindow")
    .executeJavaScript("foo();");
  ```

  Google Chrome 上でテストを再生する場合、スクリプトはクロージャで実行され、グローバル コンテキスト (window) では実行されません。すべては関数内で実行されます。上記のサンプル コードの最初の ExecuteJavaScript 呼び出しは、Google Chrome では機能しません。これは、関数 foo が ExecuteJavaScript 呼び出しが存続する間だけ有効であるためです。

  Google Chrome 上で同じテストを再生する場合は、次のような関数式を使用できます。

  ```
  // Java code
  desktop.<BrowserWindow> find("//BrowserWindow")
    .executeJavaScript("window.foo = function() { alert('Silk Test'); }");
  desktop.<BrowserWindow> find("//BrowserWindow")
    .executeJavaScript("window.foo();");
  ```

  前のサンプル コードは Silk4J では機能します。他の Silk Test クライアントに対するコードも同様です。詳細については、Silk Test クライアントのヘルプにある ExecuteJavaScript メソッドのドキュメントを参照してください。

macOS 上の Google Chrome を使用したテストの制限事項

以下のリストに、macOS 上で Google Chrome を使用した場合の、テストの再生とロケーターの記録の既知の制限事項をリストします。

- Silk4J は、typeKeys メソッドで **CMD** キーをサポートしません。
- Silk Test は、Google Chrome を使用した xBrowser ドメインの子テクノロジ ドメインのテストをサポートしていません。たとえば、Apache Flex または Microsoft Silverlight は Google Chrome ではサポートされていません。
- Silk4J は、HTTP 基本認証ダイアログのテストの記録をサポートしません。
- Silk Test は、Google Chrome のネイティブ サポートは提供しません。内部 Google Chrome 機能をテストすることはできません。たとえば、テストで、Win32 でナビゲーション バーにテキストを追加して現在表示されている Web ページを変更することはできません。回避策として、API コールを使用して Web ページ間を移動できます。Silk Test は、Alert API を使用した警告および類似のダイアログ ボックスの処理をサポートします。
- Silk4J は、Google Chrome でのテキスト解決をサポートしません。

  テキスト解決は次のメソッドを含みます。

  - textCapture
  - textClick
  - textExists
  - textRectangle
- Silk4J は、IMoveable クラスの getFocus メソッドをサポートしません。
- 複数の Google Chrome ウィンドウ上での同時テストは、macOS 上ではサポートされません。
- 既に開いている Google Chrome ウィンドウへのアタッチは、macOS 上ではサポートされません。
- Google Chrome 49 以前を使用する場合、Internet Explorer を使用してテストを再生する場合、executeJavaScript をテストするために次のコードを使用できます。

```
// Java code
desktop.<BrowserWindow> find("//BrowserWindow")
   .executeJavaScript("function foo() { alert('Silk Test'); }");
desktop.<BrowserWindow> find("//BrowserWindow")
   .executeJavaScript("foo();");
```

  Google Chrome 上でテストを再生する場合、スクリプトはクロージャで実行され、グローバル コンテキスト (window) では実行されません。すべては関数内で実行されます。上記のサンプル コードの最初の ExecuteJavaScript 呼び出しは、Google Chrome では機能しません。これは、関数 foo が ExecuteJavaScript 呼び出しが存続する間だけ有効であるためです。

  Google Chrome 上で同じテストを再生する場合は、次のような関数式を使用できます。

```
// Java code
desktop.<BrowserWindow> find("//BrowserWindow")
   .executeJavaScript("window.foo = function() { alert('Silk Test'); }");
desktop.<BrowserWindow> find("//BrowserWindow")
   .executeJavaScript("window.foo();");
```

  前のサンプル コードは Silk4J では機能します。他の Silk Test クライアントに対するコードも同様です。詳細については、Silk Test クライアントのヘルプにある ExecuteJavaScript メソッドのドキュメントを参照してください。

Mozilla Firefox を使用したテスト

このセクションでは、Mozilla Firefox を使用してテストすることによって、クロス ブラウザー テスト セットを拡張する方法について説明します。

Silk4J は、Mozilla Firefox 上での操作の記録とテストの再生をサポートします。

- Mozilla Firefox でテストを開始するときに、Mozilla Firefox のインスタンスが実行していない場合、Silk4J は Mozilla Firefox の新しいインスタンスを開始します。新しいブラウザーは、アドオン無しのキャッシュを空にした状態の一時プロファイルを使用します。

- Mozilla Firefox のインスタンス上でのテストを開始するときに、すでに実行中であれば、Silk4J はそのインスタンスが最初に開始されたときに使用されたコマンド ライン引数と同じ引数で Mozilla Firefox を再起動します。この再起動は、Silk4J オートメーション サポートを有効化するために必要です。
- この Mozilla Firefox のインスタンスは、Open Agent のシャットダウン時または、Mozilla Firefox 外のほかのアプリケーションのテストを開始するときに閉じられます。

ヒント: Mozilla Firefox を使用して既存のテスト スクリプトを実行する場合は、基本状態を使用して、URL へ移動するコマンドをテスト スクリプトに追加することを、Micro Focus はお勧めします。

Mozilla Firefox 52 以降で Silk4J を使用して記録する場合、Mozilla Firefox は外部リンクを新しいウィンドウではなく新しいタブで開きます。外部リンクを新しいウインドウで開く場合は、**ブラウザー** オプションの **OPT_FIREFOX_SINGLE_WINDOW_MODE** オプションを無効にします。

例 1

Mozilla Firefox の実行中のインスタンスが、コマンド「C:/program files/Mozilla/firefox.exe www.borland.com」で最初に起動されていた場合、Mozilla Firefox は再起動後に *www.borland.com* を開きます。

例 2

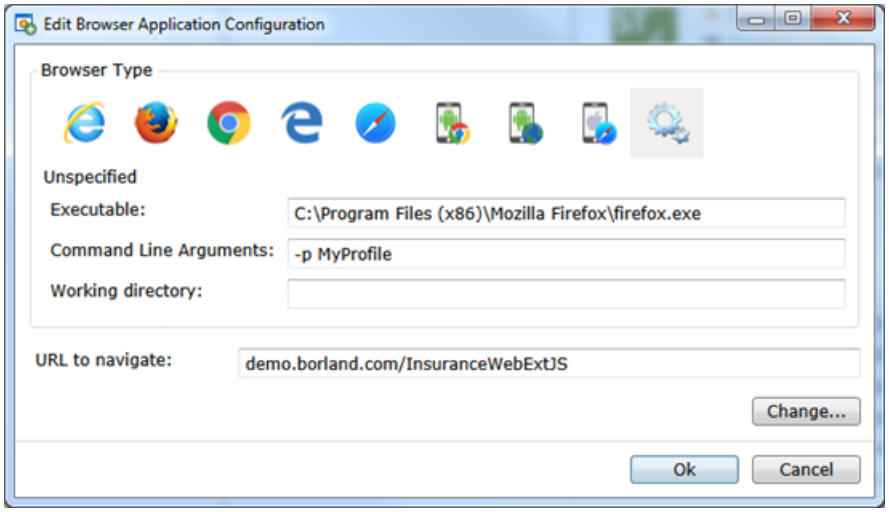
Mozilla Firefox の実行中のインスタンスが、コマンド「C:/program files/Mozilla/firefox.exe」で最初に起動されていた場合、Mozilla Firefox は再起動後に *about:blank* を開きます。

プロファイルを使用した Mozilla Firefox のテスト

ホーム ページ、使用するツールバーの設定や保存したパスワード、ブックマークなど、Mozilla Firefox で行ったすべての変更は、プロファイルと呼ばれる特別なフォルダに格納されます。

Mozilla Firefox プロファイルをテストするには、プロファイルの名前、またはプロファイルへのパスのどちらかをカスタム ブラウザー コマンド ライン引数として **アプリケーション構成の編集** ダイアログ ボックス で指定します。

1. Mozilla Firefox プロファイルをテストするプロジェクトを選択します。
2. メニューから **アプリケーション構成の編集** をクリックします。**アプリケーション構成の編集** ダイアログ ボックスが表示されます。
3. **記録および再生前に 'ブラウザーの選択' ダイアログを表示する** チェック ボックスをオフにします。
4. **編集** をクリックして、既存の **ブラウザー : Firefox** アプリケーション構成を編集します。
5. ブラウザーの種類から **カスタム** を選択します。
6. **実行可能ファイル** フィールドに、Mozilla Firefox 実行可能ファイルへのパスを入力します。たとえば、C:¥Program Files (x86)¥Mozilla Firefox¥firefox.exe など。
7. **コマンド ライン引数** フィールドに、Firefox プロファイル名または Firefox プロファイルへのパスを入力します。たとえば、-p myProfile または -profile C:/Temp など。



プロファイルの名前の使用

次のコマンド ライン引数では、プロファイル名を指定しています。

```
-p  myProfile
```

プロファイル マネージャ で名前をつけてプロファイルを設定できます。**プロファイル マネージャ** を起動するには、コマンド ウィンドウで firefox.exe -P を入力します。

プロファイルへのパスの使用

次のコマンド ライン引数では、プロファイルへのパスを指定しています。

```
-profile  C:/<path  to  profile  folder>
```

注: Silk4J が Mozilla Firefox を起動する場合、空のプロファイルが使用されます。これにより、クリーンな状態でテストが開始されることになります。ユーザーが手動で Mozilla Firefox を起動した場合は、デフォルトのプロファイルが使用されます。

注: プロファイルはテスト マシンにデプロイする必要があり、メモリ消費量も大きいため、プロファイルのテスト時に問題が発生する可能性があります。いくつかのブラウザーの設定を変更するだけであれば、プロファイルの代わりにケイパビリティを使用できます。詳細については、「*WebDriver ベースのブラウザーのケイパビリティの設定*」を参照してください。

Mozilla Firefox 拡張機能のテスト

Silk4J を使って Mozilla Firefox 拡張機能（アドオン）をテストするには、Mozilla Firefox プロファイルに拡張機能を追加し、そのプロファイルをテストに使用します。詳細については、「*プロファイルを使用した Mozilla Firefox のテスト*」を参照してください。

Mozilla Firefox を使用したテストの制限事項

Silk4J を使用して Mozilla Firefox 上の Web アプリケーションをテストする際の既知の制限事項を以下に示します。

- Silk4J は Mozilla Firefox 47、48、49、50、51 をサポートしません。ただし、Silk4J は Mozilla Firefox 47.0.1 および 47.0.2 をサポートします。
- Mozilla Firefox の複数ウィンドウの同時テストは、最初の Mozilla Firefox ウィンドウから AUT 自身が追加のウィンドウを開いた場合にのみサポートされます。追加の Mozilla Firefox ウィンドウを手動で開いた場合、Silk4J はこれらの Mozilla Firefox ウィンドウ上の要素を解決できません。たとえば、記録中に AUT のリンクやボタンをクリックして開かれた Mozilla Firefox ウィンドウ上の要素は Silk4J は解決できますが、記録中に **CTRL+N** を押して開かれた Mozilla Firefox ウィンドウ上の要素は Silk4J は解決できません。
- Silk4J は window.showmodaldialog コマンドを使って表示されるウィンドウであるモーダル ブラウザー ウィンドウのテストをサポートしません。これらのモーダル ブラウザー ウィンドウは、公式に廃止されており、Google Chrome 37 以降では無効化されています。さらに、Mozilla Firefox の今後のバージョンではサポートされる計画はありません。低レベルな操作を使ってこの問題を回避できます。

たとえば、オブジェクトの座標を使用したネイティブなクリックやテキスト フィールドを typeKeys を使って入力する方法が利用できます。

- Silk4J は、Mozilla Firefox を使用した Silverlight のテストをサポートしません。
- Silk4J は Mozilla Firefox の **バージョン情報** ダイアログなどの一部のブラウザー ダイアログのテストをサポートしません。
- Silk4J は、Mozilla Firefox を使用した about:* ページのテストをサポートしません。
- Silk4J は、Mozilla Firefox の **印刷** ボタンのクリックの記録をサポートしません。再生時にこのボタンをクリックする場合は、座標によるデスクトップのクリックをテスト スクリプトに手動で追加することにより行えます。例：

  ```
  desktop.click(MouseButton.LEFT,  printButton.getRect(true).getCenter());
  ```
- Silk4J は、Mozilla Firefox でのテキスト解決をサポートしません。

  テキスト解決は次のメソッドを含みます。

  - textCapture
  - textClick
  - textExists
  - textRectangle
- Silk Test は、Mozilla Firefox のネイティブ サポートは提供しません。内部 Mozilla Firefox 機能をテストすることはできません。たとえば、テストで、Win32 でナビゲーション バーにテキストを追加して現在表示されている Web ページを変更することはできません。回避策として、API コールを使用して Web ページ間を移動できます。Silk Test は、Alert API を使用した警告および類似のダイアログ ボックスの処理をサポートします。
- Silk4J は Mozilla Firefox の JavaScript 警告ボックスのロケーターの記録をサポートしません。Mozilla Firefox バージョン 58 以前を使用する場合、Javascript 警告ダイアログを処理するために、次のメソッドを使用できます。

  - acceptAlert
  - dismissAlert
  - getAlertText
  - isAlertPresent

  注: Mozilla Firefox 59 以降を使用する場合、これらのメソッドは使用できません。
- Silk4J は、Mozilla Firefox 52 以降では Java アプレットをサポートしません。Silk4J は、Mozilla Firefox 47.0.1 以前では Java アプレットをサポートします。ただし、次のような制限事項があります。

  - Silk4J は ロケーター //AppletContainer をサポートしません。
  - アプレットがモーダル ダイアログを開くとき、//BrowserApplication//BrowserWindow/JDialog[@caption='Information']//JButton[@caption='OK'] のようなロケーターが機能しない場合があります。代わりに、//JDialog[@caption='Information']//JButton[@caption='OK'] のようなロケーターは使用できます。
- Silk4J は、Mozilla Firefox では XPath 式のプロパティをサポートしません。XPath 式では、属性のみがサポートされます。
- Silk4J は、IMoveable クラスの getFocus メソッドをサポートしません。
- Silk Test は、Mozilla Firefox を使用した xBrowser ドメインの子テクノロジ ドメインのテストをサポートしていません。たとえば、Apache Flex または Microsoft Silverlight は Mozilla Firefox ではサポートされていません。
- Mozilla Firefox 52 以降を使用する場合、次のメソッドはサポートされません。

  - pressKeys
  - releaseKeys
- Mozilla Firefox 52 以前を使用する場合、BrowserWindow クラスの setViewportSize メソッドはサポートされません。
- Mozilla Firefox 52 以降を使用する場合、次のネイティブ再生はサポートされません。

    - ダブルクリック。
    - マウス ボタンの右と中央のクリック。
- Mozilla Firefox 52 以降を使用する場合、domClick メソッドは、警告を開くコントロールではサポートされません。
- Mozilla Firefox 55 を使用する場合、ファイルのアップロードは機能しません。詳細については、『*File upload no longer works with geckodriver 0.18.0 and Firefox 55*』を参照してください。

macOS 上の Mozilla Firefox を使用したテストの制限事項

Mozilla Firefox を使用して macOS 上の Web アプリケーションをテストする際の既知の制限事項を以下に示します。

- Silk4J は、macOS 上の Mozilla Firefox 54 以降を使用してテストされました。
- Mozilla Firefox の複数ウィンドウの同時テストは、最初の Mozilla Firefox ウィンドウから AUT 自身が追加のウィンドウを開いた場合にのみサポートされます。追加の Mozilla Firefox ウィンドウを手動で開いた場合、Silk4J はこれらの Mozilla Firefox ウィンドウ上の要素を解決できません。たとえば、記録中に AUT のリンクやボタンをクリックして開かれた Mozilla Firefox ウィンドウ上の要素は Silk4J は解決できますが、記録中に **CTRL+N** を押して開かれた Mozilla Firefox ウィンドウ上の要素は Silk4J は解決できません。
- Silk4J は window.showmodaldialog コマンドを使って表示されるウィンドウであるモーダル ブラウザー ウィンドウのテストをサポートしません。これらのモーダル ブラウザー ウィンドウは、公式に廃止されており、Google Chrome 37 以降では無効化されています。さらに、Mozilla Firefox の今後のバージョンではサポートされる計画はありません。低レベルな操作を使ってこの問題を回避できます。たとえば、オブジェクトの座標を使用したネイティブなクリックやテキスト フィールドを typeKeys を使って入力する方法が利用できます。
- Silk4J は、Mozilla Firefox を使用した Silverlight のテストをサポートしません。
- Silk4J は Mozilla Firefox の **バージョン情報** ダイアログなどの一部のブラウザー ダイアログのテストをサポートしません。
- Silk4J は、Mozilla Firefox を使用した about:* ページのテストをサポートしません。
- Silk4J は、Mozilla Firefox の **印刷** ボタンのクリックの記録をサポートしません。再生時にこのボタンをクリックする場合は、座標によるデスクトップのクリックをテスト スクリプトに手動で追加することにより行えます。例：

  ```
  desktop.click(MouseButton.LEFT,  printButton.getRect(true).getCenter());
  ```
- Silk4J は、Mozilla Firefox でのテキスト解決をサポートしません。

  テキスト解決は次のメソッドを含みます。

    - textCapture
    - textClick
    - textExists
    - textRectangle
- Silk Test は、Mozilla Firefox のネイティブ サポートは提供しません。内部 Mozilla Firefox 機能をテストすることはできません。たとえば、テストで、Win32 でナビゲーション バーにテキストを追加して現在表示されている Web ページを変更することはできません。回避策として、API コールを使用して Web ページ間を移動できます。Silk Test は、Alert API を使用した警告および類似のダイアログ ボックスの処理をサポートします。
- Silk4J は Mozilla Firefox の JavaScript 警告ボックスのロケーターの記録をサポートしません。Mozilla Firefox バージョン 58 以前を使用する場合、Javascript 警告ダイアログを処理するために、次のメソッドを使用できます。

    - acceptAlert
    - dismissAlert
    - getAlertText
    - isAlertPresent

注: Mozilla Firefox 59 以降を使用する場合、これらのメソッドは使用できません。

- Silk4J は、macOS 上の Mozilla Firefox では Java アプレットをサポートしません。
- Silk4J は、Mozilla Firefox では XPath 式のプロパティをサポートしません。XPath 式では、属性のみがサポートされます。
- Silk4J は、IMoveable クラスの getFocus メソッドをサポートしません。
- Silk Test は、Mozilla Firefox を使用した xBrowser ドメインの子テクノロジ ドメインのテストをサポートしていません。たとえば、Apache Flex または Microsoft Silverlight は Mozilla Firefox ではサポートされていません。
- 次のメソッドはサポートされません。
  - pressKeys
  - releaseKeys
- 次のネイティブ再生はサポートされません。
  - ダブルクリック。
  - マウス ボタンの右と中央のクリック。
- domClick メソッドは、警告を開くコントロールではサポートされません。
- Mozilla Firefox 55 を使用する場合、ファイルのアップロードは機能しません。詳細については、『*File upload no longer works with geckodriver 0.18.0 and Firefox 55*』を参照してください。

Microsoft Edge を使用したテスト

このセクションでは、Microsoft Edge を使用してテストすることによって、クロス ブラウザー テスト セットを拡張する方法について説明します。

Microsoft Edge を使用したテストの制限事項

以下に、Microsoft Edge を使用してテストする際の既知の制限事項を一覧します。

- 次のクラス、インターフェイス、メソッド、プロパティは、Microsoft Edge 上の Web アプリケーションのテストでは現時点ではサポートされません：
  - BrowserApplication クラス。
    - clearCache メソッド
    - closeOtherTabs メソッド
    - closeTab メソッド
    - existsTab メソッド
    - getHorizontalScrollbar メソッド
    - getNextCloseWindow メソッド
    - getSelectedTab メソッド
    - getSelectedTabIndex メソッド
    - getSelectedTabName メソッド
    - getTabCount メソッド
    - getVerticalScrollbar メソッド
    - isActive メソッド
    - minimize メソッド
    - openContextMenu メソッド
    - openTab メソッド
    - restore メソッド
    - selectTab メソッド
    - setActive メソッド
    - windowState メソッド

    - BrowserWindow クラスの次のメソッドは、Microsoft Edge のビルド 38.14393（Microsoft Windows 10 Anniversary Update のバージョンの Microsoft Edge）より前のバージョンではサポートされません。
        - pressKeys メソッド
        - releaseKeys メソッド
- Silk4J は、Microsoft Edge に対する操作を記録するとき、ブラウザを自動的に最前面に表示しません。
- Microsoft Edge を使用してテストすると、BrowserApplication の矩形領域は絶対値ではありません。
- Silk4J は、Apache Flex のテストをサポートしません。
- Silk4J は、Microsoft Edge のネイティブ サポートは提供しません。内部 Microsoft Edge 機能をテストすることはできません。たとえば、テストで、ナビゲーション バーにテキストを追加して、現在表示されている Web ページを変更することはできません。回避策として、API コールを使用して Web ページ間を移動できます。
- Silk4J は、Microsoft Edge の警告ダイアログなどのダイアログ ボックスをサポートしません。
- イメージ クリックは、Microsoft Edge Threshold 2 (ビルド 25.10586) 以降に対してのみサポートされます。Microsoft Edge の以前のバージョン上で Web アプリケーションをテストする場合は、イメージ検証のみ使用できます。
- Silk4J は、Microsoft Edge でのテキスト解決をサポートしません。

  テキスト解決は次のメソッドを含みます。

    - textCapture
    - textClick
    - textExists
    - textRectangle
- Silk4J は、Microsoft Edge のタブをサポートしません。タブはウィンドウとして解決されます。
- Microsoft Edge 上の Web アプリケーションをテストするとき、Silk4J は http-equiv 属性などのメタタグを検索できません。たとえば、Silk4J は次のメタタグを検索できません：

  ```
  <meta  http-equiv="content-type"  content="text/html;  charset=utf-8">
  ```
- Microsoft Edge を使用する場合、Silk4J は DOM 要素の currentStyle 属性の直接読み取りをサポートしません。DomElement クラスの getCssStyle メソッドを使用して、指定したスタイル名の算出 CSS スタイルを取得できます。
- Microsoft Edge 上の Web アプリケーションとのやり取りを開始するとき、Silk4J は Microsoft Edge のすべての開いているインスタンスを閉じ、新しくブラウザーを開始します。新しいブラウザーは、アドオン無しのキャッシュを空にした状態の一時プロファイルを使用します。この Microsoft Edge のインスタンスは、Open Agent のシャットダウン時または、Microsoft Edge 外のほかのアプリケーションのテストを開始するときに閉じられます。
- Microsoft Edge を使用する場合、Silk4J は操作やロケーターの記録中に textContents 属性を認識しません。しかし、オブジェクト マップで textContents 属性を使用して、Microsoft Edge 上でテストを再生するときに使用できます。
- Silk4J は、Microsoft Edge では XPath 式のプロパティをサポートしません。XPath 式では、属性のみがサポートされます。
- Silk4J は、IMoveable クラスの getFocus メソッドをサポートしません。
- Silk4J は、Content-Security-Policy HTTP ヘッダーを含んだ Web アプリケーションのテストをサポートしません。

レスポンシブ Web デザインのテスト

レスポンシブ Web デザインに基づいて構築されたデスクトップ Web アプリケーションは、アプリケーションを表示する画面や Web ブラウザーのサイズに応じて、その外観が変化します。このようなテストでは、再生するウィンドウのサイズを適切に選択できないと、その安定性に多大な影響を与える可能性があります。

Silk4J では、次のタイミングでブラウザー ウィンドウの正確なサイズを指定することができます。

- Web アプリケーションに対して新しいプロジェクトを作成する時点
- 新しいテストを記録する時点
- 既存のテストに操作を記録する時点
- Web アプリケーションに対するテストを再生する時点

次の設定を使用して、ブラウザー ウィンドウのサイズを指定することができます。

- **ブラウザー サイズ** リストには、あらかじめ定義されたブラウザー ウィンドウのサイズとカスタマイズしたサイズの両方が表示され、ここからテストで使用するサイズを選択できます。
- **向き** リストでは、ブラウザー ウィンドウの向きが横向きか縦向きかを選択できます。
- **ブラウザー サイズの編集** をクリックすると、**ブラウザー サイズ** リストにカスタム ブラウザー サイズを追加したり、リストに表示するブラウザー サイズを選択することができます。
  - 新しいカスタム ブラウザー サイズをリストに追加するには、**ブラウザー サイズの追加** をクリックします。
  - リストにサイズを表示しないようにするには、対応するチェック ボックスをオフにします。
  - 特定の Web アプリケーションのビジュアル ブレークポイントを **ブラウザー サイズ** リストに追加するには、**ビジュアル ブレークポイントの検出** をクリックします。

コマンド ラインや Silk Central から Web アプリケーションに対するテストを再生する場合は、*silktest.browserViewportSize* 環境変数を設定してブラウザー ビューポートのサイズを指定できます。**ブラウザー サイズ** リストで定義したブラウザーの名前や特定のサイズを指定することができます。

例：自動再生するブラウザー サイズに名前を使用して指定する場合

ブラウザー サイズ リストの SVGA エントリを使用して、ブラウザー サイズを SVGA (800, 600) に設定するサンプル コードを以下に示します。

```
SET  silktest.browserViewportSize=name=SVGA;orientation=landscape
```

例：自動再生するブラウザー サイズに幅と高さを使用して指定する場合

width と height パラメータをを使用して、ブラウザー サイズを SVGA (800, 600) に設定するサンプル コードを以下に示します。

```
SET
silktest.browserViewportSize=width=800;height=600;orientation=landscape
```

ビジュアル ブレークポイントの検出

レスポンシブ Web アプリケーションのビジュアル ブレークポイントを検出する前に、Mozilla Firefox 56 以降または Google Chrome 60 以降を Silk4J を実行しているマシン上にインストールしてください。

レスポンシブ Web デザイン技術を使用して実装する多くの Web アプリケーションは、表示されるブラウザーやデバイスのサイズに応じて、そのレイアウトを変更します。レイアウトの変更が発生する特定の解像度を、ビジュアル ブレークポイント と言います。

Silk4J では、ビジュアル ブレークポイントを検出し、それを使用して特定のサイズに記録ウィンドウのサイズを変更することで、このようなアプリケーションのテストをサポートします。

Silk4J では、次のタイミングでブラウザー ウィンドウの正確なサイズを指定することができます。

- Web アプリケーションに対して新しいプロジェクトを作成する時点
- 新しいテストを記録する時点
- 既存のテストに操作を記録する時点
- Web アプリケーションに対するテストを再生する時点

Web アプリケーションのビジュアル ブレークポイントを検出し、対応する解像度を **ブラウザー サイズ** リストに表示させるには、次の手順を実行します。

1. **ブラウザー サイズの編集** をクリックします。 **ブラウザー サイズの編集** ダイアログ ボックスが表示されます。
2. **ビジュアル ブレークポイントの検出** をクリックします。 Web アプリケーションの URL がアプリケーション構成や基本状態で指定されていない場合は、**ビジュアル ブレークポイント検出 URL** ダイアログが表示されます。
3. Web アプリケーションの URL がアプリケーション構成や基本状態で指定されていない場合は、**ビジュアル ブレークポイント検出 URL** ダイアログが表示されます。URL をテキスト フィールドに入力し、**OK** をクリックします。 Silk4J は Web アプリケーションのすべてのビジュアル ブレークポイントを検出し、**ブラウザー サイズ** リストに追加します。
4. **OK** をクリックして **ブラウザー サイズの編集** ダイアログを閉じます。

以上により、テストするブラウザー ウィンドウまたはモバイル デバイスのサイズに合わせて、ビジュアル ブレークポイントを選択できるようになります。

追加のブラウザーのバージョンでのテスト

このトピックでは、Silk4J のバージョンに自動的に含まれない WebDriver ベースのブラウザーの追加のバージョンでテストを行う方法について説明します。

注: Silk4J は、ブラウザーの最新のバージョンを自動的にサポートします。ただし、ブラウザーのベンダーが Silk4J のアップデートを必要とする変更を行った場合、このトピックに記述した手順を行っても、新しいブラウザーのバージョンでテストを行うことができない場合もあります。

このトピックで説明されている機能は、 次のブラウザーに対してサポートされます。

- Google Chrome
- Microsoft Edge
- Mozilla Firefox

1. テストするブラウザーのバージョンに対して適切なドライバーをダウンロードします。
   - Google Chrome の場合、追加の ChromeDriver のバージョンは *Downloads - ChromeDriver* からダウンロードできます。
   - Mozilla Firefox の場合、追加の geckodriver のバージョンは *Releases - mozilla/geckodriver* からダウンロードできます。
   - Microsoft Edge の場合、追加の Microsoft WebDriver のバージョンは *WebDriver - Microsoft Edge* からダウンロードできます。
2. Silk4J インストール フォルダーの、¥ng¥WebDrivers¥ フォルダーに移動します。
3. 使用する新しいブラウザーのバージョンのオペレーティング システムに対応するフォルダーを開きます。
   - Microsoft Windows の場合、Windows フォルダーを開きます。
   - macOS の場合、osx64 フォルダーを開きます。
4. ブラウザーに対して適切なフォルダーを開きます。
   - Google Chrome の場合、Chrome を開きます。
   - Mozilla Firefox の場合、Gecko を開きます。
   - Microsoft Edge の場合、Edge を開きます。
5. 新しいドライバーのバージョン用に新しいフォルダを作成します。
   たとえば、ドライバーが ChromeDriver 2.26 であれば、2.26 という新しいフォルダーを作成します。
6. ダウンロードしたドライバーを新しいフォルダーで展開します。
7. Silk4J インストール フォルダーで、¥ng¥WebDrivers¥Common¥Config¥ フォルダーに移動します。
8. ブラウザーに対して適切なフォルダーを開きます。
   - Google Chrome の場合、Chrome を開きます。

- Mozilla Firefox の場合、Gecko を開きます。
- Microsoft Edge の場合、Edge を開きます。

9. プロパティ ファイルをテキスト エディターで開きます。
たとえば、Google Chrome の場合、Chrome.properties を開きます。

10. 新しいブラウザーのバージョンと新しいドライバーのバージョンを次のように追加します。

```
<browser  version>=<driver  version>
```

たとえば、Google Chrome 53 でテストする場合、ChromeDriver 2.26 が必要であるため、Chrome.properties ファイルに次の行を追加する必要があります。

```
53=2.26
```

11. プロパティ ファイルを保存します。

クロス ブラウザ テスト:よくある質問

このセクションでは、さまざまなブラウザ上で Web アプリケーションをテストするときに発生することがある質問を示します。

再生中にダイアログが認識されない

スクリプトを記録するときに、 Silk4J はいくつかのウィンドウを Dialog として認識します。 スクリプトをクロス ブラウザ スクリプトとして使用する場合は、ブラウザによっては Dialog が認識されないため、Dialog を Window に置き換える必要があります。

> たとえば、スクリプトに以下の行があるとします。
> `/BrowserApplication//Dialog//PushButton[@caption='OK']`
> クロス ブラウザ テストを可能にするには、次のように行を書き換えます。
> `/BrowserApplication//Window//PushButton[@caption='OK']`

DomClick(x, y) が Click(x, y) のように動作しない

アプリケーションで onclick イベントを使用しており、座標を必要とする場合、DomClick メソッドは動作しません。 代わりに、Click を使用します。

FileInputField.DomClick() でダイアログが開かない

代わりに、Click を使用します。

テストの開始時にブラウザー ウィンドウを最大化する方法

テストの開始時など、テスト スクリプトからブラウザーを最大化するために、BrowserApplication クラスの maximize メソッドを使用できます。

> ブラウザーを最大化するには、テスト スクリプトに次のコードを追加します。
> `desktop.<BrowserApplication>  find("//BrowserApplication").maximize();`

ブラウザをスクロールさせる方法

Silk4J では、次のようにして再生中にブラウザの表示領域にコントロールをスクロールさせることができます。

executeJavaScript メソッド (DomElement)	scrollIntoView メソッドを使用して、特定の DOM 要素をブラウザ ウィンドウの表示領域にスクロールできます。

executeJavaScript メソッド (BrowserWindow)	executeJavaScript メソッドを使用して、範囲を指定してページ全体を上下にスクロールさせることができます。

使用例

次のコマンドは、 1 ページ下にスクロールさせます。

```
browserWindow.executeJavaScript("window.scrollBy(0, window.innerHeight)");
```

次のコマンドは、 1 00 ピクセル下にスクロールさせます。

```
browserWindow.executeJavaScript("window.scrollBy(0,  100)");
```

次のコマンドは、 1 00 ピクセル上にスクロールさせます。

```
browserWindow.executeJavaScript("window.scrollBy(0,  -100)");
```

現在使用しているブラウザの確認方法

BrowserApplication クラスには、ブラウザの種類を返すプロパティ "browsertype" があります。このプロパティをロケーターに追加することで、どのブラウザに一致させるかを定義できます。

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

使用例

ブラウザの種類を取得するには、次のコードをロケーターに入力します。

```
browserApplication.GetProperty("browsertype")
```

また、BrowserWindow には、現在のウィンドウのユーザー エージェント文字列を返すメソッド GetUserAgent があります。

要素のテキストに使用されるフォント タイプの確認方法

属性名を「:」で区切ると、DOM 要素の currentStyle 属性のすべての属性にアクセスできます。

Internet Explorer 8 以前	wDomElement.GetProperty("currentStyle:fontName")
Internet Explorer 9 またはそれ以降および Mozilla Firefox などの他のすべてのブラウザ	wDomElement.GetProperty("currentStyle:font-name")

innerText をカスタム クラス属性として構成したが、ロケーターで使用されない

ロケーター文字列に使用する属性には最大長があります。 InnerText は長くなりすぎる傾向があり、ロケーターで使用できない場合があります。 可能な場合は、textContents を代わりに使用してください。

xBrowser API で公開されていない機能が必要な場合の対処方法

ExecuteJavaScript() を使用して、JavaScript コードを Web アプリケーションから直接実行できます。この方法は、ほとんどすべての問題の回避策となります。

Link.Select で、Internet Explorer で新しく開いたウィンドウにフォーカスが設定されない

この制限は、ブラウザの構成設定を変更することで修正できます。 新しく開いたウィンドウが常にアクティブ化されるようにオプションを設定します。

アプリケーションのログ出力に正しくないタイムスタンプが含まれる

この方法によって、同期に関して予期しない結果が発生する場合があります。この問題を回避するには、HTML 同期モードを指定します。

新しいページに移動したあと、スクリプトがハングする

この問題は、AJAX アプリケーションによりブラウザがビジー（サーバー プッシュ／ActiveX コンポーネントの接続が開いている）のままになっている場合に、発生することがあります。 HTML 同期モードを設定してください。 他のトラブルシューティングのヒントについては、「*xBrowser* のページ同期」のトピックを参照してください。

正しくないロケーターが記録されている

マウスを要素上に移動したときに、要素の属性が変更することがあります。 Silk4J によってこのシナリオの追跡が試行されますが、失敗することがあります。 影響を受ける属性を特定し、それが Silk4J で無視されるように構成してください。

Internet Explorer で要素を囲む四角形の位置が正しくない

- 拡大率が 100% に設定されていることを確認します。 このようにしないと、四角形が正しく配置されません。
- ブラウザ ウィンドウの上に通知バーが表示されていないことを確認します。 Silk4J では、通知バーを処理できません。

マウス移動設定がオンになっているにもかかわらず、すべての操作が記録されない理由

多くの無用な MoveMouse 操作がスクリプトに影響を及ぼさないように、Silk4J では以下の操作が行われます。

- マウスが一定時間静止している場合にのみ、MoveMouse 操作が記録されます。
- マウスを要素上に移動したあとで操作が行われていることが確認された場合にのみ、MoveMouse 操作が記録されます。 場合によっては、スクリプトに手動操作を追加することが必要となることがあります。
- Silk4J は、Web アプリケーション、Win32 アプリケーション、および Windows Forms アプリケーションに対してのみ、マウス移動の記録をサポートします。 Silk4J は、Apache Flex や Swing など、xBrowser テクノロジ ドメインの子テクノロジ ドメインのマウス移動を記録することはできません。

textContents、innerText、および innerHtml の違い

- textContents は、書式設定のみを目的とする要素およびその子要素に含まれるすべてのテキストです。
- innerText は、要素およびその子要素に含まれるすべてのテキストを返します。
- innerHtml は、要素に含まれるすべてのテキスト（html タグも含む）を返します。

以下の html コードについて検討します。

```
<div id="mylinks">
  This is my <b>link collection</b>:
  <ul>
    <li><a href="www.borland.com">Bye bye <b>Borland</b> </a></li>
```

```
    <li><a href="www.microfocus.com">Welcome to <b>Micro Focus</b></a></li>
  </ul>
</div>
```

以下の表に、返されるプロパティの詳細を示します。

コード	返される値
browser.DomElement("//div[@id='mylinks']").GetProperty("textContents")	This is my link collection:
browser.DomElement("//div[@id='mylinks']").GetProperty("innerText")	This is my link collection:Bye bye Borland Welcome to Micro Focus
browser.DomElement("//div[@id='mylinks']").GetProperty("innerHtml")	This is my link collection: Bye bye Borland Welcome to Micro Focus

注: Silk Test 13.5 以降で、要素の textContents プロパティを介して取得されるテキストの空白はサポートするブラウザすべてで一貫してトリムされます。ブラウザのバージョンによっては、この空白の処理方法が Silk Test 13.5 以前の Silk Test バージョンで異なる場合があります。以前の動作を有効にしたい場合、OPT_COMPATIBILITY オプションを 13.5.0 より低いバージョンに設定してください。たとえば、オプションを Silk Test 13.0 に設定するには、スクリプトで次のように入力します：

```
desktop.setOption("OPT_COMPATIBILITY", "13.0.0");
```

クロス ブラウザー スクリプトの作成時に必要な処置

クロス ブラウザー スクリプトを作成する場合は、以下の 1 つまたは複数の問題に遭遇する場合があります。

- クロス ブラウザー テスト用にスクリプトを記録する場合、Google Chrome、Mozilla Firefox、または Microsoft Edge を使用することを Micro Focus では推奨しています。 Internet Explorer を使って Silk4J で記録したスクリプトは、ほかのブラウザーで記録したスクリプトと若干異なる場合があります。
- 属性値が異なる。たとえば、Internet Explorer の色が "# FF0000" として、Mozilla Firefox の色が "rgb(255,0,0)" として返されます。
- 属性名が異なる。たとえば、Internet Explorer 8 以前のバージョンのフォント サイズ属性は "fontSize" でしたが、Internet Explorer 9 以降および Mozilla Firefox などの他のすべてのブラウザでは "font-size" となります。
- 一部のフレームワークで 異なる DOM ツリーがレンダリングされることがある

安定したクロス ブラウザ テストを実現するために最適なロケーター

組み込みロケーター生成プログラムでは、安定したロケーターの作成が試みられます。ただし、情報を使用できない場合、高品質のロケーターを生成することは困難です。この場合、ロケーター生成プログラムでは、階層形式の情報およびインデックスが使用されます。その結果、直接的な記録／再生には適していても、安定した日常的な実行には適さない脆弱なロケーターが生成されます。さらに、クロス ブラウザ テストでは、いくつかの AJAX フレームワークで異なるブラウザに対して異なる DOM 階層がレンダリングされることがあります。

この問題を回避するには、アプリケーションの UI 要素にカスタム ID を使用します。

ロケーターでクラスとスタイルの属性が使用されない理由

これらの属性は AJAX アプリケーションで頻繁に変更され、ロケーターの安定性が損なわれることがありため、無視リストに含まれています。 ただし、多くの場合、これらの属性を使用してオブジェクトを識別できるため、アプリケーションで使用することに意味がある場合があります。

Internet Explorer 10 で Click の記録が異なる理由

Internet Explorer 10 の DomElement で Click を記録し、DomElement が Click の後で破棄された場合、記録動作が予期したとおりにならないことがあります。 別の DomElement が最初の DomElement の下にある場合、Silk Test では、1 つの Click が記録されるのではなく、Click、MouseMove、および ReleaseMouse が記録されます。

予期しない記録動作を回避する方法は、テスト対象のアプリケーションによって異なります。 通常は、記録されたスクリプトから不必要な MouseMove イベントと ReleaseMouse イベントを削除すれば十分です。

ハンドル無効エラーが表示される理由

このトピックでは、Silk4J に「このオブジェクトのハンドルは無効になりました。」というエラー メッセージが表示された場合の対処法について説明します。

このメッセージは、たとえば WaitForProperty などのメソッドを呼び出したオブジェクトが何らかの理由で消失していることを示しています。たとえば、Web アプリケーションでメソッドを呼び出しているときに、何らかの理由でブラウザが 新しいページに移動した場合、以前のページのすべてのオブジェクトは自動的に無効になります。

Web アプリケーションのテストでは、組み込みの同期がこの問題の原因の場合があります。たとえば、テスト対象のアプリケーションにショッピング カートが含まれていて、このショッピング カートに品物を追加したとします。ユーザーは次のページが読み込まれ、ショッピング カートのステータスが 品物がある 状態に変わるまで待機しています。品物を追加するという操作からの戻り時間が短すぎた場合、最初のページのショッピング カートはステータスが変わるまで待機しますが、その間も新しいページは読み込まれています。したがって、最初のページのショッピング カートは無効になります。この動作によって、ハンドル無効エラーが発生します。

この問題を回避するには、2 番目のページでのみ有効なオブジェクトが表示されるまで待機してから、ショッピング カートのステータスを確認するようにしてください。このオブジェクトが有効になるとすぐに、ショッピング カートのステータスを確認できるようになり、2 番目のページで正しく検証されるようになります。

すべてのアプリケーションに対するベスト プラクティスとして、テスト内でよく利用するコントロールを検索するためのメソッドを分離することを Micro Focus はお勧めします。例：

```
public Dialog getSaveAsDialog(Desktop desktop) {
 return desktop.find("//Dialog[@caption = 'Save As']");
}
```

Find および FindAll メソッドはそれぞれ一致したオブジェクトのハンドルを返し、そのハンドルは、アプリケーション内でオブジェクトが存在する間だけ有効です。たとえば、ダイアログへのハンドルは、ダイアログが一旦閉じられると無効になります。ダイアログを閉じたあとに、このハンドルに対してメソッドを実行すると、InvalidObjectHandleException がスローされます。同様に、Web ページ上の DOM オブジェクトのハンドルも、Web ページが再読み込みされると無効になります。テスト メソッド間の実行や、その順番の独立性を保ってデザインすることは共通のプラクティスであるため、それぞれのテスト メソッドでオブジェクトの新しいハンドルを取得するようにします。XPath クエリの重複を避けるため、getSaveAsDialog のようなヘルパ メソッドを作成します。例：

```
@Test
public void testSaveAsDialog() {
 // ... some code to open the 'Save As' dialog (e.g by clicking a menu item) ...
 Dialog saveAsDialog = getSaveAsDialog(desktop);
```

```
 saveAsDialog.close();
 // ... some code to open the 'Save As' dialog again
 getSaveAsDialog(desktop).click(); // works as expected
 saveAsDialog.click(); //  fails because an InvalidObjectHandleException is thrown
}
```

このコードの最後の行は、存在しないオブジェクトのハンドルを使用しているため、失敗します。

スクリプトからのブラウザの起動

テストの開始時に再生するブラウザを選択するのではなく、再生時にテスト スクリプトから特定のブラウザを起動することが必要になる場合があります。

BrowserBaseState クラスの使用

テスト スクリプトからブラウザを起動するために、BrowserBaseState クラスを使用すると、Executable プロパティで指定したブラウザが実行され、テストの準備ができている状態になります。さらに、基本状態として Url プロパティで指定した URL に移動され、ブラウザが最前面に表示されます。

次のサンプル コードは、BrowserBaseState を使用して Internet Explorer を起動します。

ブラウザの複数インスタンスの使用

複数のブラウザ ウィンドウまたはタブが開かれている場合、Silk4J は各ブラウザ ウィンドウまたはタブを、ユニークなロケーターを持つ個別のオブジェクトとして処理します。ロケーターは、*WebBrowser*、*WebBrowser[1]*、*WebBrowser[2]* のように、インデックス付きで表されます。

非表示入力フィールドの検索

非表示入力フィールドは、タグに type="hidden" を指定した HTML フィールドです。find で非表示入力フィールドを検索できるようにするために、OPT_XBROWSER_FIND_HIDDEN_INPUT_FIELDS オプションを使用できます。このオプションのデフォルト値は、TRUE です。

```
desktop.setOption(CommonOptions.OPT_XBROWSER_FIND_HIDDEN_INPUT_FIELDS, true);
```

Web アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Web アプリケーションがサポートする属性は次のとおりです。

- caption (次のワイルドカードをサポート：? および *)
- すべての DOM 属性 (次のワイルドカードをサポート：? および *)

注: 各ブラウザによって、空のスペースの処理に違いがあります。 この結果、「textContent」および「innerText」属性は正規化されています。 空のスペースのあとに別の空のスペースが続く場合、空のスペースはスキップされるか、または 1 文字の空白で置き換えられます。 空のスペースとは、検出されたスペース、キャリッジ リターン、改行、タブのことです。 また、このような値に一致するものも正規化されます。 例：

```
<a>abc
abc</a>
```

以下のロケーターを使用します。

```
//A[@innerText='abc  abc']
```

Web アプリケーションのカスタム属性

HTML は、安定した識別子を表すことができる一般的な属性 ID を定義します。定義により、ID は文書内の要素を一意に識別します。特定の ID を持つ要素は文書内で 1 つだけ存在します。

ただし、多くの場合 (特に AJAX アプリケーションでは)、ID は HTML 要素に関連付けられたサーバー ハンドラを動的に識別するために使用されます。つまり、Web 文書の作成のたびに ID は変わることになります。このような場合、ID は安定した識別子ではなく、Web アプリケーションの UI コントロールを識別するのに適しません。

Web アプリケーションの場合、より確実にするには、Silk4J に UI コントロールの情報を公開するためだけに使用されるカスタム HTML 属性を新たに導入することです。

カスタム HTML 属性はブラウザーは無視するため、AUT の動作は変わりません。ブラウザーの DOM を通してアクセスすることができます。Silk4J では、このような属性を (属性がコントロール クラスのカスタム 属性であっても) 識別時のデフォルト属性として使用するように設定することができます。特定のテクノロジ ドメインのデフォルト識別属性としてカスタム属性を設定するには、**Silk4J** > **オプションの編集** > **カスタム属性** をクリックして、テクノロジ ドメインを選択します。

アプリケーション開発者は、Web 要素にさらに HTML 属性を追加することが必要です。

元の HTML コード：

```
<A  HREF="http://abc.com/control=4543772788784322..."  <IMG
src="http://abc.com/xxx.gif"  width=16  height=16>  </A>
```

新しいカスタム HTML 属性 *AUTOMATION_ID* を持つ HTML コード：

```
<A  HREF="http://abc.com/control=4543772788784322..."
AUTOMATION_ID  =  "AID_Login"  <IMG  src="http://abc.com/xxx.gif"
width=16  height=16>  </A>
```

カスタム属性を設定すると、Silk4J は、できる限りカスタム属性を使用して、一意のロケーターを構成しようとします。Web ロケーターは次のようになります。

```
…//DomLink[@AUTOMATION_ID='AID_Login'
```

例：変化する ID

変化する ID の 1 例は、Google Widget Toolkit (GWT) で、ID は Web 文書の作成のたびに変化する動的な値を保持します：

```
ID  =  'gwt-uid-<nnn>'
```

この場合、<nnn> が頻繁に変化します。

Microsoft Windows 10 上のテストの制限事項

以下に、Microsoft Windows 10 (Windows 10) 上でテストする際の既知の制限事項を一覧します。

- Silk Test は、Windows 10 上のユニバーサル Windows プラットフォーム (UWP) アプリのテストをサポートしません。

64 ビット アプリケーションのサポート

Silk4J では、以下のテクノロジについて、64 ビット アプリケーションのテストがサポートされています。

- Windows Forms

- Windows Presentation Foundation (WPF)
- Microsoft Windows API ベース
- Java AWT/Swing
- Java SWT

サポートするバージョン、既知の問題、および回避策についての最新の情報は、リリース ノートを確認してください。

サポートする属性の種類

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。 必要に応じて、以下のいずれかの方法を使用して属性の種類を変更できます。

- 他の属性の種類と値を手動で入力する。
- **推奨属性リスト** の値を変更して、デフォルトの属性の種類に対して別の設定を指定する。

Apache Flex アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Flex アプリケーションがサポートする属性は次のとおりです。

- automationName
- caption（automationName と同様）
- automationClassName（FlexButton など）
- className（実装クラスの完全修飾名。 mx.controls.Button など）
- automationIndex（FlexAutomation のビューでのコントロールのインデックス。 index:1 など）
- index（automationIndex と同様。ただし、接頭辞はなし。 1 など）
- id (コントロールの ID)
- windowId (id と同様)
- label（コントロールのラベル）
- すべての動的ロケーター属性

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

動的ロケーター属性の詳細については、「動的ロケーター属性」を参照してください。

Java AWT/Swing アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Java AWT/Swing でサポートされる属性には以下のものがあります。

- caption
- priorlabel : 隣接するラベル フィールドのテキストによってテキスト入力フィールドを識別します。 通常、フォームのすべての入力フィールドに、入力の目的を説明するラベルがあります。 caption のないコントロールの場合、自動的に属性 **priorlabel** がロケーターに使用されます。 コントロールの

priorlabel 値 (テキスト入力フィールドなど) には、コントロールの左側または上にある最も近いラベルの caption が使用されます。
- name
- accessibleName
- *Swing のみ*：すべてのカスタム オブジェクトの定義属性は、ウィジェットに putClientProperty("propertyName", "propertyValue") で設定されます。

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Java SWT アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Java SWT がサポートする属性は次のとおりです。

- caption
- すべてのカスタム オブジェクト定義属性

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

SAP アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

SAP がサポートする属性は次のとおりです。

- automationId
- caption

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Silverlight コントロールを識別するためのロケーター属性

Silverlight コントロールでサポートされているロケーター属性は次のとおりです。

- *automationId*
- *caption*
- *className*
- *name*
- すべての動的ロケーター属性

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別され

ますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード？および * をサポートしています。

動的ロケーター属性の詳細については、「動的ロケーター属性」を参照してください。

Silverlight スクリプト内のコンポーネントを識別するために、*automationId*、*caption*、*className*、*name*、または任意の動的ロケーター属性を指定できます。 *automationId* はアプリケーション開発者が設定します。 たとえば、*automationId* を持つロケーターは、以下のようになります：//SLButton[@automationId="okButton"]

automationId は一般に非常に有用で安定した属性であるため、使用することを推奨します。

属性の種類	説明	例
automationId	テスト対象アプリケーションの開発者によって設定される識別子。 Visual Studio デザイナは、デザイナ上で作成されたすべてのコントロールに自動的に *automationId* を割り当てます。 アプリケーション開発者は、アプリケーションのコード上でコントロールを識別するために、この ID を使用します。	//SLButton[@automationId="okButton"]
caption	コントロールが表示するテキスト。 複数の言語にローカライズされたアプリケーションをテストする場合、*caption* の代わりに *automationId* や *name* 属性を使用することを推奨します。	//SLButton[@caption="Ok"]
className	Silverlight コントロールの .NET 単純クラス名 (名前空間なし)。 *className* 属性を使用すると、Silk4J が解決する標準 Silverlight コントロールから派生したカスタム コントロールを識別するのに役立ちます。	//SLButton[@className='MyCustomButton']
name	コントロールの名前。 テスト対象アプリケーションの開発者によって設定されます。	//SLButton[@name="okButton"]

注目: XAML コードの *name* 属性は、ロケーター属性 *name* ではなく、ロケーター属性 *automationId* にマップされます。

Silk4J は、*automationId*、*name*、*caption*、*className* 属性をこの表に示した順番に使用して Silverlight コントロールのロケーターを記録時に作成します。 たとえば、コントロールが *automationId* と *name* を持つ場合、*automationId* が固有の場合は Silk4J がロケーターを作成する際に使用されます。

以下の表は、アプリケーション開発者がテキスト「Ok」を持つ Silverlight ボタンをアプリケーションの XAML コードに定義する方法を示しています。

オブジェクトの XAML コード	Silk Test からオブジェクトを検索するためのロケーター
<Button>Ok</Button>	//SLButton[@caption="Ok"]
<Button Name="okButton">Ok</Button>	//SLButton[@automationId="okButton"]
<Button AutomationProperties.AutomationId="okButton">Ok</Button>	//SLButton[@automationId="okButton"]
<Button AutomationProperties.Name="okButton">Ok</Button>	//SLButton[@name="okButton"]

Rumba コントロールを識別するためのロケーター属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。 サポートされている属性は次のとおりです。

caption	コントロールが表示するテキスト。
priorlabel	フォームの入力フィールドには通常入力の目的を説明するラベルがあるため、**priorlabel** の目的は隣接するラベル フィールド **RumbaLabel** のテキストによってテキスト入力フィールド **RumbaTextField** を識別することです。 テキスト フィールドの同じ行の直前にラベルがない場合、または右側のラベルが左側のラベルよりテキスト フィールドに近い場合、テキスト フィールドの右側にあるラベルが使用されます。
StartRow	この属性は記録されていませんが、手動でロケーターに追加することができます。**StartRow** を使用して、この行で始まるテキスト入力フィールド、**RumbaTextField** を識別します。
StartColumn	この属性は記録されていませんが、手動でロケーターに追加することができます。**StartColumn** を使用して、この列で始まるテキスト入力フィールド、**RumbaTextField** を識別します。
すべての動的ロケーター属性。	動的ロケーター属性の詳細については、「動的ロケーター属性」を参照してください。

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Web アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Web アプリケーションがサポートする属性は次のとおりです。

- caption (次のワイルドカードをサポート：? および *)
- すべての DOM 属性 (次のワイルドカードをサポート：? および *)

注: 各ブラウザによって、空のスペースの処理に違いがあります。 この結果、「textContent」および「innerText」属性は正規化されています。 空のスペースのあとに別の空のスペースが続く場合、空のスペースはスキップされるか、または 1 文字の空白で置き換えられます。 空のスペースとは、検出されたスペース、キャリッジ リターン、改行、タブのことです。 また、このような値に一致するものも正規化されます。 例：

```
<a>abc
abc</a>
```

以下のロケーターを使用します。

```
//A[@innerText='abc  abc']
```

Windows Forms アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Windows Forms アプリケーションがサポートする属性は次のとおりです。

- automationid
- caption
- windowid
- priorlabel (caption のないコントロールの場合、自動的に priorlabel が caption として使用されます。caption のあるコントロールの場合、caption を使う方が簡単な場合があります。)

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

Windows Presentation Foundation (WPF) アプリケーションの属性

WPF アプリケーションがサポートする属性は次のとおりです。

- *automationId*
- *caption*
- *className*
- *name*
- すべての動的ロケーター属性。

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

動的ロケーター属性の詳細については、「動的ロケーター属性」を参照してください。

オブジェクト解決

WPF スクリプト内のコンポーネントを識別するために、*automationId*、*caption*、*className*、あるいは *name* を指定できます。 アプリケーション中の要素に指定された name が利用可能な場合、ロケーターの *automationId* 属性として使用されます。 この結果、多くのオブジェクトは、この属性のみを使用して一意に識別できます。 たとえば、*automationId* を持つロケーターは、以下のようになります : // WPFButton[@automationId='okButton']"

automationId や他の属性を定義した場合、再生中に *automationId* だけが使用されます。 *automationId* が定義されていない場合には、コンポーネントを解決するのに *name* が使用されます。 *name* も *automationId* もどちらも定義されていない場合には、*caption* 値が使用されます。 caption が定義されていない場合は、*className* が使用されます。 *automationId* は非常に役立つプロパティであるため、使用することを推奨します。

属性の種類	説明	例
automationId	テスト アプリケーションの開発者によって提供された ID	//WPFButton[@automationId='okButton']"
name	コントロールの名前。 Visual Studio デザイナは、デザイナ上で作成されたすべてのコントロールに自動的に名前を割り当てます。 アプリケーション開発者は、アプリケーションのコード上でコントロールを識別するために、この名前を使用します。	//WPFButton[@name='okButton']"
caption	コントロールが表示するテキスト。 複数の言語にローカライズされたアプリケー	//WPFButton[@automationId='Ok']"

属性の種類	説明	例
	ションをテストする場合、caption の代わりに automationId や name 属性を使用することを推奨します。	
className	WPF の .NET 単純クラス名 (名前空間なし)。 クラス名属性を使用すると、Silk4J が解決する標準 WPF コントロールから派生したカスタム コントロールを識別するのに役立ちます。	//WPFButton[@className='MyCustomButton']"

Silk4J は、*automationId*、*name*、*caption*、*className* 属性をこの表に示した順番に使用して WPF コントロールのロケーターを記録時に作成します。 たとえば、コントロールが *automationId* と *name* を持つ場合、Silk4J がロケーターを作成する際には *automationId* が使用されます。

以下の例では、アプリケーション開発者がアプリケーションの WPF ボタンに対して *name* と *automationId* を XAML コードに定義する方法を示します。

```
<Button  Name="okButton"  AutomationProperties.AutomationId="okButton"
Click="okButton_Click">Ok</Button>
```

Windows API ベースのクライアント/サーバー アプリケーションの属性

ロケーターが作成されるとき、属性の種類はアプリケーションが使用するテクノロジ ドメインに基づいて自動的に割り当てられます。 属性の種類と値によって、ロケーターがテスト内のオブジェクトを識別する方法が決定されます。

Windows API ベースのクライアント/サーバー アプリケーションがサポートする属性は次のとおりです。

- caption
- windowid
- priorlabel : 隣接するラベル フィールドのテキストによってテキスト入力フィールドを識別します。 通常、フォームのすべての入力フィールドに、入力の目的を説明するラベルがあります。 caption のないコントロールの場合、自動的に属性 **priorlabel** がロケーターに使用されます。 コントロールの **priorlabel** 値 (テキスト ボックスなど) には、コントロールの左側または上にある最も近いラベルの caption が使用されます。

注: 属性名は、大文字小文字が区別されます (モバイル アプリケーションを除く。モバイル アプリケーションでは、大文字小文字は無視されます)。デフォルトで、属性値では大文字と小文字が区別されますが、他のオプションと同様にこのデフォルト設定は変更できます。ロケーター属性は、ワイルドカード ? および * をサポートしています。

動的ロケーター属性

再生中にコントロールを識別するために、事前に定義されたロケーター属性のセット (*caption* や *automationId* など。テクノロジ ドメインに依存します) をロケーターに使用できます。 しかし、動的プロパティを含む、コントロールのすべての属性をロケーター属性として使用することもできます。 特定のコントロールで使用可能なプロパティのリストを取得するには、GetPropertyList メソッドを使用します。返されたプロパティはすべて、ロケーターを使用してコントロールを識別するのに使用できます。

注: 特定のプロパティの実際の値を取得するには、GetProperty メソッドを使用します。 この値はロケーターで使用できます。

例

Silverlight アプリケーションのダイアログ ボックスにあるボタンを識別する場合、以下のように入力します。

```
browser.Find("//SLButton[@IsKeyboardFocused=true]")
```

または

```
Dim button = dialog.SLButton("@IsKeyboardFocused=true")
```

これが機能するのは、Silk4J により Silverlight ボタン コントロールの IsDefault というプロパティが公開されるためです。

例

Silverlight アプリケーションのフォント サイズ 12 のボタンを識別する場合、以下のように入力します。

```
Dim button = browser.Find("//SLButton[@FontSize=12]")
```

または

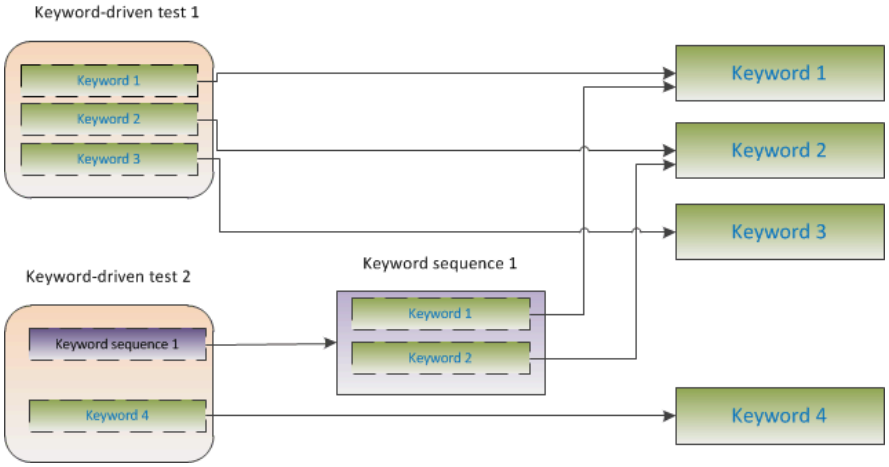
```
Dim button = browser.SLButton("@FontSize=12")
```

これが機能するのは、テスト対象アプリケーションの基になるコントロール (この場合、Silverlight ボタン) が FontSize というプロパティを持つためです。

キーワード駆動テスト

キーワード駆動テスト は、テスト開発からテスト設計を分離するソフトウェア テスト手法です。 このため、テスト自動化プロセスにビジネス アナリストなどの専門家を含めることができます。Silk Central と Silk Test はキーワード駆動テストをサポートしており、Silk Test のキーワードの形式での共有資産として構成されるメンテナンス可能な自動化フレームワークを自動化エンジニアが開発することによって、自動化エンジニアとビジネス アナリスト間で密接な共同作業を行うことができます。その後、これらのキーワードは、Silk Test で新しいキーワード駆動テストを作成したり、Silk Central で既存の手動テスト資産を自動テストに変換したり、新しいキーワード駆動テストを作成するために、ビジネス アナリストが使用することができます。

- キーワード駆動テスト は、実行可能なキーワードのコレクションです。キーワード駆動テストは、他のテストと同様に再生することができます。
- キーワード シーケンスは、他のキーワードを組み合わせたものです。キーワード シーケンスは、頻繁に使用するキーワードの組み合わせを 1 つのキーワードにまとめることにより、メンテナンスの労力を低減し、テストを理解しやすくすることができます。
- キーワード は、テスト オブジェクトに対する複数の操作の組み合わせを定義したものです。キーワードの実装は、さまざまなツールとプログラム言語 (Java や .NET など) を使用して行えます。



キーワード駆動テストの作成には、次の 2 つのフェーズがあります。

1. テストの設計
2. キーワードの実装

キーワード駆動テストで利用可能な記録/再生コントロールの完全な一覧については、「*API* リファレンス」の com.borland.silk.keyworddriven.annotations パッケージを参照してください。

キーワード駆動テストの利点

キーワード駆動テスト手法を使用する利点を次に示します。

- キーワード駆動テストを使用すると、テスト自動化とテスト ケースのデザインが分離され、うまく分業できるようになり、キーワードを実装するテスト エンジニアとテスト ケースをデザインする専門家が共同作業できます。
- テスト対象アプリケーションにアクセスすることなく、初期の段階からテストを開発でき、後からキーワードを実装できます。
- プログラムの知識がなくてもテストを開発できます。

- キーワード駆動テストは、長期的に見るとメンテナンス コストを低減できます。キーワードのメンテナンスが必要で、これらのキーワードを使用するすべてのキーワード駆動テストは自動的に更新されます。
- テストケースが簡潔です。
- 技術者でなくてもテスト ケースが読みやすく、理解しやすくなります。
- テスト ケースの変更が簡単です。
- 既存のキーワードを再利用して新しいテストを再利用できます。これにより、より広範囲なテスト カバレッジを実現しやすくなります。
- キーワード実装の内部的な複雑性を、キーワード駆動テストを作成または実行するユーザーに対して隠蔽できます。

キーワード

キーワード は、テスト オブジェクトに対する複数の操作の組み合わせを定義したものです。キーワードの実装は、さまざまなツールとプログラム言語 (Java や .NET など) を使用して行えます。 Silk4J でのキーワードは、アノテーション付きのテスト メソッド (@Keyword)です。キーワードは、キーワード資産として保存されます。

キーワード駆動テストの作成中にキーワードやキーワード シーケンスを定義し、後でそれらをテスト メソッドとして実装できます。既存のテスト メソッドに @Keyword アノテーションを付けて、キーワードとしてマークすることもできます。 Java では、キーワードは次のアノテーションで定義されます：

```
@Keyword("keyword_name")
```

キーワード シーケンスは、他のキーワードを組み合わせたものです。キーワード シーケンスは、頻繁に使用するキーワードの組み合わせを 1 つのキーワードにまとめることにより、メンテナンスの労力を低減し、テストを理解しやすくすることができます。

キーワードまたはキーワード・シーケンスは、合計 20 の入力および出力パラメーターを持つことができます。キーワードを実装するテスト メソッドのパラメータは、キーワードのパラメータです。 キーワードのパラメータに違う名前を指定するために、次を使用できます：

```
// Java code
@Argument("parameter_name")
```

デフォルトでは、パラメータは Silk4J の入力パラメータです。出力パラメータを定義するには、OutParameter クラスを使用します。

注: キーワード駆動テスト エディタ でキーワードの出力パラメータを指定するには、次のように記述します。

```
${parameter_name}
```

キーワード駆動テスト エディタ で、キーワードの出力パラメータを他のキーワードの入力パラメータとして使用する場合も、同じように記述します。

例

キーワードとしてマークされたテスト メソッドは、次のようになります。

```
// Java code
@Keyword("Login")
public void login(){
      ... // method implementation
}
```

または

```
// Java code
@Keyword(value="Login", description="Logs in with the given name and password.")
public void login(@Argument("UserName") String userName,
```

```
  @Argument("Password") String password,
  @Argument("Success") OutParameter success) {
    ... // method implementation
}
```

このキーワードは、指定したユーザー名とパスワードを使ってテスト対象アプリケーションにログインし、ログインが成功したかどうかを返します。 出力パラメータを他のキーワードの入力パラメータとして使用するには、キーワード内で出力パラメータに値を設定します。

注: ヘルプ トピックを PDF で参照している場合、このサンプル コードは、実際のスクリプトでは許されない場所で改行されてしまっている場合があります。スクリプトでこのサンプル コードを使用する場合は、これらの改行を削除してください。

- Keyword アノテーション のキーワード名パラメータは、省略可能です。メソッドの名前とは異なる名前を指定する場合に、キーワード名パラメータを使用できます。パラメータが指定されていない場合、メソッドの名前がキーワード名として使用されます。
- Argument アノテーションも省略可能です。メソッドをキーワードとしてマークすると、自動的にすべての引数がキーワードの引数として使用されます。例のように、*userName* を *UserName* にしたい場合など、キーワードの引数とは異なる名前を指定する場合に、Argument アノテーションを使用できます。

Silk4J でキーワード駆動テストを作成する

キーワード駆動テスト エディター を使って、新しいキーワードと既存のキーワードを新しいキーワード駆動テストに結合できます。新しいキーワードは、後のステップで自動テストのメソッドとして実装する必要があります。

1. **Silk4J** > **新規キーワード駆動テスト** をクリックします。 **新規キーワード駆動テスト** ダイアログ ボックスが開きます。
2. 新しいテストの名前を **名前** フィールドに入力します。
3. 新しいテストを追加したいプロジェクトを選択します。

   デフォルトでは、プロジェクトがアクティブであれば、そのアクティブなプロジェクトに新しいテストが作成されます。

   注: Silk4J が提供する機能を最適に使用するには、同じテストで複数のアプリケーションをテストする場合を除き、テストするアプリケーションごとに個別のプロジェクトを作成します。
4. **終了** をクリックして、キーワード駆動テストを保存します。
5. **いいえ** をクリックして、空のキーワード駆動テストを作成します。 **キーワード駆動テスト エディター** が開きます。
6. 次のアクションのいずれかを実行します。
   - 新しいキーワードを追加する場合は、**新しいキーワード** フィールドにキーワードの名前を入力します。
   - 既存のキーワードを追加する場合は、リストを展開して追加するキーワードを選択します。
7. Enter を押します。
8. 実行するすべてのキーワードを追加するまで、上記の 2 つの手順を繰り返します。
9. **ファイル** > **保存** をクリックします。

続いて、キーワードを実装します。すべてのキーワードが実装されている場合は、テストを実行します。

Silk4J でのキーワード駆動テストの記録

Silk4J でキーワード駆動テストを作成する前に、プロジェクトを選択する必要があります。

単一のキーワードを記録する場合は、「キーワードの記録」を参照してください。

キーワード駆動テストを記録するには：

1. **Silk4J** > **新規キーワード駆動テスト** をクリックします。 **新規キーワード駆動テスト** ダイアログ ボックスが開きます。
2. 新しいテストの名前を **名前** フィールドに入力します。
3. 新しいテストを追加したいプロジェクトを選択します。
   デフォルトでは、プロジェクトがアクティブであれば、そのアクティブなプロジェクトに新しいテストが作成されます。

   注: Silk4J が提供する機能を最適に使用するには、同じテストで複数のアプリケーションをテストする場合を除き、テストするアプリケーションごとに個別のプロジェクトを作成します。
4. **終了** をクリックして、キーワード駆動テストを保存します。
5. **はい** をクリックして、キーワード駆動テストの記録を開始します。**キーワード駆動テストの記録** ダイアログ ボックスが開きます。
6. 現在のプロジェクトに対してアプリケーション構成が設定されていない場合、テストするアプリケーションの種類に対応するタブを選択します。
   - ブラウザで実行しない標準アプリケーションをテストする場合は、**Windows** タブを選択します。
   - Web アプリケーションまたはモバイル Web アプリケーションをテストする場合は、**Web** タブを選択します。
   - ネイティブ モバイル アプリケーションをテストする場合は、**モバイル** タブを選択します。
7. 標準アプリケーションをテストするには、リストからアプリケーションを選択します。
8. Web アプリケーションまたはモバイル Web アプリケーションをテストするには、現在のプロジェクトに対してアプリケーション構成が設定されていない場合は、リストからインストール済みのブラウザまたはモバイル ブラウザのうちの 1 つを選択します。
   a) **移動する URL の入力** テキスト ボックスに、開く Web ページを指定します。選択したブラウザのインスタンスが既に実行されている場合、**実行中のブラウザーの URL を使用する** をクリックして、実行中のブラウザ インスタンスに現在表示されている URL の記録を行うことができます。チュートリアルの場合、**Internet Explorer** を選択し、**移動する URL の入力** テキスト ボックスに *http://demo.borland.com/InsuranceWebExtJS/* を指定します。
   b) 省略可能：あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。
   たとえば、Apple Safari 上の Web アプリケーションを Apple iPhone 7 の画面と同じ大きさのブラウザー ウィンドウでテストするには、リストから **Apple iPhone 7** を選択します。
   c) 省略可能：ブラウザー ウィンドウの **向き** を選択します。
   d) 省略可能：**ブラウザー サイズの編集** をクリックすると、新しいブラウザー サイズを指定したり、**ブラウザー サイズ** リストに表示するブラウザー サイズを選択することができます。
9. 現在のプロジェクトに対してアプリケーション構成が設定されていない場合に、ネイティブ モバイル アプリケーション (アプリ) をテストするには：
   a) アプリをテストするモバイル デバイスをリストから選択します。
   b) **参照** をクリックしてアプリ ファイルを選択するか、アプリ ファイルへの完全パスを **モバイル アプリ ファイル** テキスト フィールドに入力します。
   このパスでは、Silk4J は HTTP および UNC 形式をサポートします。
   Silk4J は、モバイル デバイスまたはエミュレータ上に指定したアプリをインストールします。

10. 現在のプロジェクトに対してアプリケーション構成が設定されており、Web アプリケーションをテストする場合、**ブラウザーの選択** ダイアログ ボックスが開きます。
    a) ブラウザーを選択します。
    b) 省略可能：あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。
    たとえば、Apple Safari 上の Web アプリケーションを Apple iPhone 7 の画面と同じ大きさのブラウザー ウィンドウでテストするには、リストから **Apple iPhone 7** を選択します。
    c) 省略可能：ブラウザー ウィンドウの **向き** を選択します。
    d) 省略可能：**ブラウザー サイズの編集** をクリックすると、新しいブラウザー サイズを指定したり、**ブラウザー サイズ** リストに表示するブラウザー サイズを選択することができます。
11. 開いているダイアログに応じて、次のいずれかを実行します。
    - **アプリケーションの選択** ダイアログ ボックスで、**OK** をクリックします。
    - **ブラウザーの選択** ダイアログ ボックスで、**記録** をクリックします。
12. テスト対象アプリケーションで、最初のキーワードに含める操作を実行します。
    記録中に利用可能な操作についての詳細は、「記録中に利用可能な操作」を参照してください。
13. キーワードの名前を指定するには、**記録中** ウィンドウでキーワードの名前の上にマウス カーソルを動かして、**編集** をクリックします。

    注: Silk4J は、キーワード駆動テストの開始に アプリケーションの開始 キーワードを自動的に追加します。このキーワードで、アプリケーションの基本状態が実行され、テストを正しく再生できるようになります。基本状態についての詳細は、「基本状態」を参照してください。
14. キーワードの名前を **キーワードの名前** フィールドに入力します。
15. **OK** をクリックします。
16. 次のキーワードの操作を記録するには、**新しいキーワードの名前** フィールドに新しいキーワードの名前を入力し、**追加** をクリックします。Silk4J は、新しいキーワードに新しい操作を記録します。
17. キーワード駆動テスト全体を記録すまで、新しいキーワードを作成し、キーワードに対する操作を記録します。
18. **停止** をクリックします。 **記録完了** ダイアログ ボックスが開きます。
19. 省略可能：**パッケージ** テキスト ボックスに、パッケージ名を指定します。
    たとえば、次のように入力します：com.example。
    既存のパッケージを使用するには、**選択** をクリックし、使用するパッケージを選択します。
20. **テスト クラス** テキスト ボックスに、テスト クラスの名前を指定します。
    たとえば、次のように入力します：AutoQuoteInput。
    既存のクラスを使用するには、**選択** をクリックし、使用するクラスを選択します。
21. **OK** をクリックします。

Silk4J は、すべての記録したキーワードを含む新しいキーワード駆動テストを作成します。

Silk4J でのキーワード駆動テストの基本状態の設定

Silk4J でキーワード駆動テストを実行すると、キーワード駆動テストは基本状態のキーワードを呼び出すことにより、Silk4J は AUT を基本状態から開始します。

キーワード駆動テストの記録時に、Silk4J は基本状態のキーワード (isBaseState プロパティが *true* に設定されているキーワード) を現在のプロジェクトから検索します。

- 基本状態のキーワードが現在のプロジェクトに存在した場合、Silk4J は、キーワード駆動テストの最初のキーワードとして、このキーワードを挿入します。
- 基本状態のキーワードがプロジェクトに存在しない場合、Silk4J は、アプリケーションの開始 という名前の新しい基本状態のキーワードを作成し、キーワード駆動テストの最初のキーワードとして挿入します。

キーワードを手動で基本状態のキーワードとしてマークするには、isBaseState プロパティを Keyword アノテーションに追加し、プロパティの値を *true* に設定します。

```
@Keyword(value = "Start application", isBaseState = true)
public void start_application() {
  // Base state implementation
}
```

Silk4J でのキーワードの実装

キーワードを実装する前に、キーワード駆動テストの一部としてキーワードを定義します。

キーワード駆動テストで再利用するためにキーワードを実装するには：

1. 実装するキーワードが含まれているキーワード駆動テストを開きます。
2. **キーワード駆動テスト エディター** で、実装するキーワードの左側に表示される **キーワードの実装** をクリックします。 **キーワードの場所の選択** ダイアログ ボックスが開きます。
3. **選択** をクリックして、キーワード実装に追加するパッケージやクラスを選択します。
4. 省略可能：**パッケージ** フィールドに、新しいキーワード実装のパッケージ名を入力します。
5. **クラス** フィールドに、新しいキーワード実装のクラス名を入力します。
6. **OK** をクリックします。
7. 次のアクションのいずれかを実行します。
   - キーワードを記録するには、**はい** をクリックします。
   - 空のキーワード メソッドを作成するには、**いいえ** をクリックします。
8. 現在のプロジェクトに対してアプリケーション構成が設定されており、Web アプリケーションをテストする場合、**ブラウザーの選択** ダイアログ ボックスが開きます。
   a) ブラウザーを選択します。
   b) 省略可能：あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。
   たとえば、Apple Safari 上の Web アプリケーションを Apple iPhone 7 の画面と同じ大きさのブラウザー ウィンドウでテストするには、リストから **Apple iPhone 7** を選択します。
   c) 省略可能：ブラウザー ウィンドウの **向き** を選択します。
   d) 省略可能：**ブラウザー サイズの編集** をクリックすると、新しいブラウザー サイズを指定したり、**ブラウザー サイズ** リストに表示するブラウザー サイズを選択することができます。
9. **記録** をクリックします。
   記録の詳細については、「キーワードの記録」を参照してください。

実装したキーワードが **キーワード** ウィンドウで未実装として表示されている場合は、Eclipse メニューで **プロジェクト** > **自動的にビルド** をチェックします。

Silk4J でのキーワードの記録

完全に新しいキーワードに対してではなく、キーワード駆動テストに既に存在するキーワードに対しては、操作の記録のみを行えます。新しいキーワード駆動テストを記録する場合は、「キーワード駆動テストの記録」を参照してください。

新しいキーワードの操作を記録するには：

1. 記録するキーワードが含まれているキーワード駆動テストを開きます。
2. **キーワード駆動テスト エディター** で、実装するキーワードの左側に表示される **キーワードの実装** をクリックします。 **キーワードの場所の選択** ダイアログ ボックスが開きます。

3. **選択** をクリックして、キーワード実装に追加するパッケージやクラスを選択します。
4. 省略可能：**パッケージ** フィールドに、新しいキーワード実装のパッケージ名を入力します。
5. **クラス** フィールドに、新しいキーワード実装のクラス名を入力します。
6. **OK** をクリックします。
7. 現在のプロジェクトに対してアプリケーション構成が設定されており、Web アプリケーションをテストする場合、**ブラウザーの選択** ダイアログ ボックスが開きます。
   a) ブラウザーを選択します。
   b) 省略可能：あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。
   たとえば、Apple Safari 上の Web アプリケーションを Apple iPhone 7 の画面と同じ大きさのブラウザー ウィンドウでテストするには、リストから **Apple iPhone 7** を選択します。
   c) 省略可能：ブラウザー ウィンドウの **向き** を選択します。
   d) 省略可能：**ブラウザー サイズの編集** をクリックすると、新しいブラウザー サイズを指定したり、**ブラウザー サイズ** リストに表示するブラウザー サイズを選択することができます。
8. **記録** をクリックします。 **記録中** ウィンドウが開き、Silk4J はキーワードの操作の記録を開始します。
9. テスト対象アプリケーションで、テストする操作を実行します。
   記録中に利用可能な操作についての詳細は、「記録中に利用可能な操作」を参照してください。
10. **停止** をクリックします。 **記録完了** ダイアログ ボックスが開きます。

記録した操作は、定義したクラスのコンテキストに表示されます。

スクリプトのテスト メソッドをキーワードとして指定

スクリプトの既存のテスト メソッドをキーワードとして指定して、キーワード駆動テストのメソッドを再利用します。

1. キーワードとして指定するテスト メソッドを含む スクリプトを開きます。
2. @keyword() をテスト メソッドの直前に追加します。
   デフォルトでは、キーワード名はテスト メソッドの名前です。
3. 省略可能：@keyword("keywordName") をテスト メソッドの直前に追加すると、キーワードに他の名前を設定できます。

これで、テスト メソッドをキーワード駆動テストでキーワードとして使用できるようになります。

使用例

テスト メソッド login を *Login* という名前で新しいキーワードとして指定するには、テスト メソッドの直前に以下を入力します。

```
@Keyword("Login")
```

2 つの入力パラメータ UserName と PassWord を持つテスト メソッド login を *Login* という名前で新しいキーワードとして指定するには、以下を入力します。

```
@Keyword(value="Login", description="Logs in with the given name and password.")
public void login(@Argument("UserName") String userName,
@Argument("PassWord") String password) {
    ...     // method implementation
}
```

注: ヘルプ トピックを PDF で参照している場合、このサンプル コードは、実際のスクリプトでは許されない場所で改行されて

しまっている場合があります。スクリプトでこのサンプル コードを使用する場合は、これらの改行を削除してください。

キーワード駆動テストの編集

注: Silk4J では、Silk4J にあるキーワード駆動テストの編集と実行を行うことができ、また、Silk Central に格納されているキーワード駆動テストを実行することができます。Silk Central に格納されているキーワード駆動テストを編集するには、**キーワード駆動テスト エディター** でキーワード駆動テストを開き、**編集** をクリックします。

キーワード駆動テストを編集するには：

1. **キーワード駆動テスト エディター** でキーワードを開きます。
   a) **パッケージ エクスプローラー** で、キーワード駆動テストが存在するプロジェクトを展開します。
   b) **Keyword Driven Tests** フォルダを展開します。
   c) 編集するキーワード駆動テストをダブルクリックします。
2. 新しいキーワードをキーワード駆動テストに追加するには：
   a) **新しいキーワード** フィールドをクリックします。
   b) 新しいキーワードの名前を入力します。
   c) Enter を押します。
3. 既存のキーワードを編集するには、キーワードの左側にある **キーワードを開く** をクリックします。

   注: Silk Central は、Silk Central で作成したすべてのキーワードの所有権を持ちます。このことは、このようなキーワードに対して行う変更は Silk4J でではなく、Silk Central で保存されることを意味します。
4. キーワードをキーワード駆動テストにコピーするには：
   a) キーワードを選択します。

      ヒント: 行番号の列で **Ctrl + クリック** または **Shift + クリック** を使用すると、複数のキーワードを選択できます。
   b) **Ctrl + C** を押します。
   c) キーワードを挿入する上の行を選択します。
   d) **Ctrl + V** を押します。
5. キーワード駆動テストのほかの場所にキーワードを移動するには、キーワードをクリックして他の場所にドラッグするか、または次の手順を実行します。
   a) キーワードを選択します。

      ヒント: 行番号の列で **Ctrl + クリック** または **Shift + クリック** を使用すると、複数のキーワードを選択できます。
   b) **Ctrl + X** を押します。
   c) キーワードを挿入する上の行を選択します。
   d) **Ctrl + V** を押します。
6. キーワード駆動テストからキーワードを削除するには、キーワードの左側にある **キーワードの削除** をクリックします。

   キーワードは、**キーワード** ウィンドウでまだ利用可能なので、いつでもキーワード駆動テストに再度追加することができます。
7. 変更を保存するには、**ファイル** > **保存** をクリックします。

Silk Central でテストのキーワードを管理する

キーワード ページでは、選択したキーワード駆動テストのキーワードを管理できます。次のアクションを実行できます。

タスク	ステップ
テストまたはキーワード シーケンスを Silk Test で開く	**Silk Test で開く** をクリックして、選択したテストまたはキーワード シーケンスを Silk Test で開きます。
キーワードの追加	1. キーワード リストの一番下にある **新しいキーワード** をクリックするか、キーワードを右クリックして、コンテキスト メニューから **キーワードを上に挿入** を選択します。  **注:** キーワードの使用状況に基づいて、Silk Test にキーワードを推薦させることができます。推薦させるかどうかを切り替えるには、コンテキスト メニューの **レコメンドの有効化** または **レコメンドの無効化** を使用します。詳細については、「*Silk4J* が推薦するキーワード」を参照してください。 2. 利用可能なキーワードのリストからキーワードを選択するか、新しいキーワードを作成します。 3. **保存** をクリックします。 または、右側にある **すべてのキーワード** ペインから、既存のキーワードをダブル クリックするか、ドラッグ&ドロップします。  **ヒント: Ctrl + クリック** を使用すると、複数のキーワードを選択できます。ドロップするとき、キーワードは選択した順番に並び替えられます。
キーワードの削除	削除するキーワードの **アクション** 列で  をクリックします。**保存** をクリックします。
キーワードの順序の変更	キーワードを移動したい位置にドラッグ&ドロップします。**保存** をクリックします。
キーワード シーケンス (他のキーワードから構成されるキーワード) の作成	1. キーワード リストから結合したいキーワードを選択します。行番号の列で **Ctrl + クリック** または **Shift + クリック** を使用すると、複数のキーワードを選択できます。 2. 選択範囲を右クリックして、**結合** をクリックします。 3. 新しいキーワード シーケンスの **名前** と **説明** を指定します。
キーワード シーケンスからキーワードの抽出	キーワード シーケンスを右クリックし、**キーワードの抽出** を選択します。これによって、元のキーワード シーケンスがそれに含まれるキーワードによって置換されますが、ライブラリからは削除されません。**保存** をクリックします。
テストまたはキーワード シーケンスへのキーワードのコピーと貼り付け	1. キーワード リストからコピーしたいキーワードを選択します。行番号の列で **Ctrl + クリック** または **Shift + クリック** を使用すると、複数のキーワードを選択できます。 2. **Ctrl + C** を押すと選択項目がコピーされます。キーワードを移動する場合には、**Ctrl + X** を押します。 3. キーワードをコピーするテストまたはキーワード シーケンスを開き、キーワードを挿入する上の行を選択します。 4. **Ctrl + V** を押します。

タスク	ステップ
	ヒント: 選択したキーワードを Excel に貼り付けて編集し、それをコピーしてテストまたはキーワード シーケンスに貼り付けることもできます。
キーワード シーケンスのパラメータの定義	1. キーワード リスト上の **パラメータ** をクリックします。**パラメータ** ダイアログ ボックスが表示されます。 2. **パラメータの追加** をクリックします。 3. 新しいパラメータに対して **名前** を指定します。パラメータが出力パラメータ (入力値を必要とするのではなく、値を戻します) の場合、**出力** チェック ボックスをチェックします。 4. **OK** をクリックします。 5. **保存** をクリックします。 **注:** キーワードまたはキーワード シーケンスは、入力パラメータと出力パラメータを合わせて 20 個まで持つことができます。
ドラフト キーワードの編集	1. 編集するドラフト キーワードの**アクション** 列で、をクリックします。 2. キーワードの **グループ** を選択するか、新しいグループを指定します。 3. **説明** にキーワードの説明を入力します。この情報は、キーワードを実装するエンジニアにとって有益な情報です。 4. **OK** をクリックします。 5. 省略可能：パラメータ フィールドをクリックして、キーワードのパラメータを追加します。キーワードが Silk Test で実装されると、これらのパラメータは、生成されたコード スタブに現れます。 6. **保存** をクリックします。
キーワードの検索	**キーワード** ビュー の検索フィールドを使用して、特定のキーワードを検索します。文字を入力するとリストが更新され、その文字に一致する既存のキーワードが動的に表示されます。検索のヒント： • 検索では大文字小文字は区別されません。doAction を入力すると、doaction や DOAction が表示されます。 • 大文字のみを入力すると、いわゆる キャメルケース検索が実行されます。ECD を入力すると、Enter Car Details、Enter Contact Details、EnterContactDetails が表示されます。 • キーワード名とグループ名が検索対象になります。test を入力すると、test を含むすべてのキーワードと、グループ名に test を含むグループのすべてのキーワードが表示されます。 • ? は、0 または 1 文字と一致します。user?test を入力すると、userTest や usersTest が表示されます。 • * は、0 または n 文字と一致します。my*keyword を入力すると、myKeyword や myNewKeyword や my_other_keyword が表示されます。 • <文字列>. はグループ名のみを検索します。group. を入力すると、グループ名に group を含むグループのすべてのキーワードが表示されます。 • .<文字列> はキーワード名のみを検索します。.keyword を入力すると、keyword を含むすべてのキーワードが表示されます。 • <string>.<string> は特定のグループのキーワードを検索します。group.word を入力すると、myGroup グループの myKeyword が表示されます。

タスク	ステップ
	• 引用符を使用して完全一致のみを検索します。'Keyword' を入力すると、Keyword や MyKeyword は表示されますが、keyword は表示されません。

Silk4J のキーワード レコメンド機能

キーワード駆動テスト エディター でキーワード駆動テストにキーワードを追加するとき、Silk4J は、そのテストの次のキーワードとして使用する可能性のある既存のキーワードを推薦します。推薦するキーワードはキーワード リストの上位にリストされ、Silk4J がどの程度そのキーワードを推薦しているかが、棒グラフの塗りつぶした長さによって示されます。

Silk4J は、次のアルゴリズムに基づいてキーワードを推薦します。

- キーワード駆動テストまたはキーワード シーケンスに最初のキーワードを追加する場合、Silk4J は他のキーワード駆動テストまたはキーワード シーケンスの最初のキーワードとして使用されているのと同様なキーワードを検索します。最も頻繁に使用されるキーワードがより上位に推薦されます。
- 既に他のキーワードを含んでいるキーワード駆動テストまたはキーワード シーケンスに、さらにキーワードを追加する場合、Silk4J は次のようにしてキーワードを推薦します。
  - 新しいキーワードを追加するキーワード駆動テストまたはキーワード シーケンスの位置の前にキーワードがある場合、Silk4J は前のキーワード群と、すべてのほかのキーワード駆動テストとキーワード シーケンスのキーワードの組み合わせとを比較し、最も頻繁に使用されているキーワードの組み合わせに続いて現れるキーワードを推薦します。
  - キーワード駆動テストまたはキーワード シーケンスの位置の前にキーワードがないが、現在の位置の後にキーワードがある場合、Silk4J は後のキーワード群と、すべてのほかのキーワード駆動テストとキーワード シーケンスのキーワードの組み合わせとを比較し、最も頻繁に使用されているキーワードの組み合わせの前に現れるキーワードを推薦します。
- さらに、Silk4J は見つかったキーワードの類似性も考慮します。たとえば、2 つのキーワードの名前とグループの両方が一致する場合、Silk4J は名前だけが一致するキーワードよりもこれらをより上位に推薦します。
- Silk Central との接続が確立すると、現在のプロジェクトに対応するキーワード ライブラリに属したキーワード駆動テストに含まれるキーワードも考慮されます。

キーワードでのパラメータの使用

キーワードまたはキーワード シーケンスは、入力パラメータと出力パラメータを合わせて 20 個まで持つことができます。このトピックでは、Silk4J を使用してこれらのパラメータを処理する方法について説明します。

キーワード駆動テスト エディター で、キーワードまたはキーワード シーケンスに対して定義されたパラメータを表示し、パラメータの値を編集できます。

キーワード ウィンドウで、キーワードまたはキーワード シーケンス上にマウス カーソルを動かすと、キーワードまたはキーワード シーケンスに割り当てられているパラメータを確認できます。

単純なキーワードの入力パラメータ

他のテスト メソッドに対するものと同じ方法で、 キーワードの入力パラメータを定義して使用することができます。

次のサンプル コードでは、2 つの入力パラメータ userName と password を持つキーワード setUserDetails を定義する方法を説明します。

```
@Keyword
public void setUserDetails(String userName, String password) {
```

```
  ...
}
```

単純なキーワードの出力パラメータ

キーワードの戻り値または 1 つ以上の出力パラメータを定義することができます。また、戻り値と 1 つ以上の出力パラメータを組み合わせて使用することもできます。

次のサンプル コードでは、文字列を返すキーワード getText を定義する方法を説明します。

```
@Keyword
public String getText() {
  return "text";
}
```

次のサンプル コードでは、2 つの出力パラメータ userName と password を持つキーワード getUserDetails を定義する方法を説明します。

```
@Keyword
public void getUserDetails(OutParameter userName, OutParameter password) {
  userName.setValue("name");
  password.setValue("password");
}
```

キーワード シーケンスのパラメータ

キーワード シーケンスのパラメータは、**パラメータ** ダイアログ ボックスで定義し、編集できます。このダイアログ ボックスは、**キーワード シーケンス エディター** の **パラメータ** をクリックして開くことができます。

例：パラメータを取るキーワード

このトピックでは、パラメータを取るキーワードを使用する方法の例を紹介します。キーワードまたはキーワード シーケンスは、入力パラメータと出力パラメータを合わせて 20 個まで持つことができます。

最初の手順として、使用するキーワードを含んだキーワード駆動テストを作成します。これは、キーワード駆動テスト全体を記録するか、新しいキーワード駆動テストを作成してからキーワード駆動テスト エディターでキーワードを追加することによって行います。

この例では、キーワード駆動テストは次のキーワードを含んでいます。

アプリケーションの開始	これは、AUT を開始して基本状態を設定する標準キーワードです。
Login	これは、ユーザー名とパスワードで識別される特定のユーザーで AUT にログインするキーワードです。
GetCurrentUser	このキーワードは、AUT に現在ログインしているユーザーの名前を返します。
AssertEquals	このキーワードは、2 つの値を比較します。
Logout	これは、AUT からユーザーをログアウトするキーワードです。

次のステップでは、パラメータをキーワードに追加します。これを行うには、キーワードのテスト スクリプトを開き、メソッドにパラメータを追加します。

入力パラメータ UserName と Password をキーワード Login に追加するには、

```
@Keyword("Login")
public void login() {
  ...
}
```

を、次のように変更します。

```
@Keyword("Login")
public void login(String UserName, String Password) {
```

```
  …
}
```

出力パラメータ UserName をキーワード GetCurrentUser に追加するには、

```
@Keyword("GetCurrentUser")
public void getCurrentUser() {
  …
}
```

を、次のように変更します。

```
@Keyword("GetCurrentUser")
public void getCurrentUser(OutParameter CurrentUser) {
  …
}
```

キーワード駆動テスト エディター 上のキーワード駆動テストは、次のようになります。

		Keyword	Parameters	
1		Start application		
2		Login	*UserName*	*Password*
3		GetCurrentUser	*CurrentUser ←*	
4		AssertEquals	*Expected*	*Actual*
5		Logout		

これで、**キーワード駆動テスト エディター** 上で入力パラメータの実際の値を指定できます。キーワード GetCurrentUser の出力パラメータ UserName の値を取得するには、*${current user}* のように、変数を指定します。その後のキーワードで、変数を指定して格納された値を渡すことができます。

		Keyword	Parameters	
1		Start application		
2		Login	*UserName*	*Password*
3		GetCurrentUser	${current user}	
4		AssertEquals	John Smith	${current user}
5		Logout		

キーワードのキーワード シーケンスへの結合

キーワード駆動テスト エディター を使って、複数のキーワード駆動テストで順番に実行したいキーワードを結合してキーワード シーケンスを作成できます。

1. 結合するキーワードが含まれているキーワード駆動テストを開きます。
2. **キーワード駆動テスト エディター** で、Ctrl キーを押しながら結合したいキーワードをクリックします。
3. 選択範囲を右クリックして、**結合** をクリックします。**キーワードの結合** ダイアログ ボックスが開きます。
4. **名前** フィールドに、新しいキーワード シーケンスの名前を入力します。

5. 省略可能：**説明** フィールドに、新しいキーワード シーケンスの説明を入力します。
6. **結合** をクリックします。

新しいキーワード シーケンスが開き、**キーワード** ウィンドウにも表示されます。キーワード駆動テストでキーワード シーケンスを使用できます。

注: 他のキーワードと同じように、キーワード シーケンス自身を実行することはできませんが、キーワード駆動テストの一部として実行することができます。

Eclipse からのキーワード駆動テストの再生

1. **プロジェクト エクスプローラー**で、再生するキーワード駆動テスト資産に移動します。
2. 資産名を右クリックします。
3. **実行** > **キーワード駆動テスト** を選択します。
4. 省略可能： **構成の実行** ダイアログ ボックスを開くには、**実行** > **実行の構成** を選択します。
5. 省略可能： **実行構成** ダイアログ ボックスで、他のテストやプロジェクトを選択できます。
6. 省略可能：**実行の構成** ダイアログ ボックスの **グローバル変数** グリッドで、キーワード駆動テストの実行に使用される任意の変数の値を設定できます。これらの値は、キーワード駆動テスト資産を実行するときに常に使用されます。
   a) 変数の **変数名** と **値** を対応するフィールドに入力します。
   b) **Enter** を入力すると、グリッドに新しい行が追加されます。
   c) 前の 2 つの手順を繰り返して、使用するすべてのグローバル変数の値を設定します。
   Silk Central などのテスト管理ツールで管理されている自動化フレームワークの一部としてキーワード駆動テストを実行する場合、新しい *.properties* ファイルをプロジェクトに追加して、プロジェクト全体のグローバル変数の値を設定できます。詳細については、「変数を指定したキーワード駆動テストの再生」を参照してください。
7. 省略可能： **実行の構成** ダイアログ ボックスを閉じて、キーワード駆動テスト資産の実行を開始するには、**実行** をクリックします。
8. Web アプリケーションをテストする場合は、**ブラウザーの選択** ダイアログ ボックスが開きます。ブラウザーを選択して、 **実行** をクリックします。

   注: 複数のアプリケーションが現在のプロジェクトに対して設定されている場合、**ブラウザーの選択** ダイアログ ボックスは表示されません。
9. **実行** をクリックします。
10. 省略可能：必要に応じて、両方の **Shift** キーを同時に押して、テストの実行を停止できます。
11. テストの実行が完了すると、**再生完了** ダイアログ ボックスが開きます。**結果の検討** をクリックして、完了したテストの TrueLog を確認します。

Silk Central に保存されたキーワード駆動テストの再生

Silk Central に保存されたキーワード駆動テストを再生して、テストした機能が期待通り動作しているか検証します。

1. Silk4J メニューで、**Silk4J** > **キーワード ビューの表示** をクリックします。
2. **キーワード** ビューで、キーワード駆動テストをダブル クリックします。
   キーワード ビューで Silk Central による変更を更新するには、**更新** をクリックします。
3. ツールバーで、**実行** をクリックします。
4. Web アプリケーションをテストする場合は、**ブラウザーの選択** ダイアログ ボックスが開きます。ブラウザーを選択して、 **実行** をクリックします。

注: 複数のアプリケーションが現在のプロジェクトに対して設定されている場合、**ブラウザーの選択** ダイアログ ボックスは表示されません。

5. **実行** をクリックします。
6. 省略可能：必要に応じて、両方の **Shift** キーを同時に押して、テストの実行を停止できます。
7. テストの実行が完了すると、**再生完了** ダイアログ ボックスが開きます。**結果の検討** をクリックして、完了したテストの TrueLog を確認します。

コマンド ラインからのキーワード駆動テストの再生

このタスクを実行する前に、JDK の場所を参照できるように PATH 変数を更新する必要があります。詳細については、『*JDK Installation for Microsoft Windows*』を参照してください。

CI サーバーからテストを再生する場合など、コマンド ラインからキーワード駆動テストを再生するには、KeywordTestSuite クラスを使用します。

1. コマンド ラインからキーワード駆動テストを実行するには、@KeywordTests アノテーションを使用して JUnit テスト スイートを作成します。たとえば、キーワード駆動テスト *My Keyword-Driven Test* を実行する場合は、次のよな JUnit テスト スイート MyTestSuite を作成します。

```
@RunWith(KeywordTestSuite.class)
@KeywordTests({ "My Keyword-Driven Test" })
public class MyTestSuite {

}
```

2. CLASSPATH に以下を含めます。
   - junit.jar
   - org.hamcrest.core JAR ファイル
   - silktest-jtf-nodeps.jar
   - com.borland.silk.keyworddriven.engine.jar
   - キーワード駆動テストを含んだフォルダの JAR

```
set CLASSPATH=<eclipse_install_directory>¥plugins
¥org.junit_4.11.0.v201303080030¥junit.jar;<eclipse_install_directory>¥plugins
¥org.hamcrest.core_1.3.0.v201303031735.jar;%OPEN_AGENT_HOME%¥JTF¥silktest-jtf-
nodeps.jar;%OPEN_AGENT_HOME%¥KeywordDrivenTesting
¥com.borland.silk.keyworddriven.engine.jar;C:¥myTests.jar
```

3. 省略可能：新しい *.properties* ファイルをプロジェクトに追加して、キーワード駆動テストの実行に使用する任意の変数の値を設定します。
   詳細については、「変数を指定したキーワード駆動テストの再生」を参照してください。
4. java org.junit.runner.JUnitCore <Name> を入力して JUnit テスト メソッドを実行します。ここで、*Name* は、最初の手順で作成した JUnit テスト スイートの名前です。

注: トラブル シューティングの情報については、次の JUnit のドキュメントを参照してください：*http://junit.sourceforge.net/doc/faq/faq.htm#running_1*.

例

たとえば、*My Keyword Driven Test 1* と *My Keyword Driven Test 2* の 2 つのキーワード駆動テストを実行するには、次のクラスを作成します。

```
package demo;

import org.junit.runner.RunWith;

import com.borland.silktest.jtf.keyworddriven.KeywordTestSuite;
import com.borland.silktest.jtf.keyworddriven.KeywordTests;
```

```
@RunWith(KeywordTestSuite.class)
@KeywordTests({ "My Keyword Driven Test 1", "My Keyword Driven Test 2" })
public class MyTestSuite {

}
```

コマンド ラインからクラスを実行するには、次のように入力します。

```
java org.junit.runner.JUnitCore demo.KeywordTestSuite
```

ファイル *c:¥temp¥globalvariables.properties* に格納されたグローバル変数を使用してコマンド ラインからクラスを実行するには、次のように入力します。

```
java -Dsilk.keyworddriven.engine.globalVariablesFile=c:¥temp¥globalvariables.properties org.junit.runner.JUnitCore demo.KeywordTestSuite
```

詳細については、「変数を指定したキーワード駆動テストの再生」を参照してください。

Apache Ant を使用したキーワード駆動テストの再生

このトピックで述べる手順を実行するには、コンピュータに Apache Ant がインストールされている必要があります。

Apache Ant を使用してキーワード駆動テストを再生し、たとえば、テスト実行の HTML レポートを生成するには、KeywordTestSuite クラスを使用します。

1. Apache Ant を使用してキーワード駆動テストを実行するには、@KeywordTests アノテーションを使用して JUnit テスト スイートを作成します。たとえば、キーワード駆動テスト *My Keyword-Driven Test* を実行する場合は、次のよな JUnit テスト スイート MyTestSuite を作成します。

   ```
   @RunWith(KeywordTestSuite.class)
   @KeywordTests({ "My Keyword-Driven Test" })
   public class MyTestSuite {

   }
   ```
2. キーワード駆動テストを含んだ Silk4J プロジェクトの build.xml ファイルを開きます。
3. キーワード駆動テストを実行するには、次のターゲットを build.xml ファイルに追加します。

   ```
   <target name="runTests" depends="compile">
     <mkdir dir="./reports"/>
     <junit printsummary="true" showoutput="true" fork="true">
       <classpath>
         <fileset dir="${output}">
           <include name="**/*.jar" />
         </fileset>
         <fileset dir="${buildlib}">
           <include name="**/*.jar" />
         </fileset>
         <fileset dir="C:/Program Files (x86)/Silk/SilkTest/ng/KeywordDrivenTesting">
           <include name="**/*.jar" />
         </fileset>
       </classpath>

       <test name="MyTestSuite" todir="./reports"/>
     </junit>
   </target>
   ```

   JUnit タスクの詳細については、『*https://ant.apache.org/manual/Tasks/junit.html*』を参照してください。

4. 省略可能：すべてのテストの XML レポートを作成するには、ターゲットに次のコードを追加します。

```
<formatter type="xml" />
```

5. 省略可能：XML レポートから HTML レポートを作成するには、ターゲットに次のコードを追加します。

```
<junitreport todir="./reports">
  <fileset dir="./reports">
    <include name="TEST-*.xml" />
  </fileset>
  <report format="noframes" todir="./report/html" />
</junitreport>
```

JUnitReport タスクの詳細については、『*https://ant.apache.org/manual/Tasks/junitreport.html*』を参照してください。
完全なターゲットは、次のようになります。

```
<target name="runTests" depends="compile">
  <mkdir dir="./reports"/>
  <junit printsummary="true" showoutput="true" fork="true">
    <classpath>
      <fileset dir="${output}">
        <include name="**/*.jar" />
      </fileset>
      <fileset dir="${buildlib}">
        <include name="**/*.jar" />
      </fileset>
      <fileset dir="C:/Program Files (x86)/Silk/SilkTest/ng/KeywordDrivenTesting">
        <include name="**/*.jar" />
      </fileset>
    </classpath>

    <formatter type="xml" />

    <test name="MyTestSuite" todir="./reports"/>
  </junit>
  <junitreport todir="./reports">
    <fileset dir="./reports">
      <include name="TEST-*.xml" />
    </fileset>
    <report format="noframes" todir="./report/html" />
  </junitreport>
</target>
```

6. Eclipse からテストを実行するには、以下の手順を実行します。
   a) **パッケージ・エクスプローラー** で、build.xml ファイルを右クリックします。
   b) **実行** > **Ant ビルド...** を選択します。
   c) **構成の編集** ダイアログ ボックスの **ターゲット** タブで、**runTests** をチェックします。
   d) **実行** をクリックします。

コマンド ラインや CI サーバーからテストを実行することもできます。詳細については、『*https://ant.apache.org/manual/running.html*』および *Silk4J* ヘルプ の「*CI (*継続的インテグレーション*)* サーバーからのテストの再生」を参照してください。

変数を指定したキーワード駆動テストの再生

キーワード駆動テスト用の変数の値を設定する前に、プロジェクトを作成する必要があります。

ユーザーが実行するキーワード駆動テスト資産のすべての実行に対するグローバル変数の値を設定するには、**実行の構成** ダイアログ ボックスの **グローバル変数** グリッドを使用します。詳細については、「*Eclipse* からのキーワード駆動テストの再生」を参照してください。

Silk Central などのテスト管理ツールで管理されている自動化フレームワークの一部としてキーワード駆動テストを実行するときに、Silk4J でのキーワード駆動テストの実行に使用する変数の値を設定できます。プロジェクト全体に対するグローバル変数の値 (つまり、このプロジェクトのキーワード駆動テスト資産を Silk4J ユーザーが実行するときに常にこれらの値が使用されます) を設定するには、次の手順を実行します。

1. **パッケージ・エクスプローラー** で、変数を指定して実行するキーワード駆動テストが存在するプロジェクトを展開します。
2. プロジェクトの **src** フォルダを右クリックして、**新規** > **ファイル** を選択します。 **新規ファイル** ダイアログ ボックスが開きます。
3. **ファイル名** フィールドに、globalvariables.properties を入力します。
4. **終了** をクリックします。 新しいプロパティ ファイルが開きます。
5. ファイルの新しい行を追加して変数を指定します。

   新しい変数のフォーマットは次のようになります。

   ```
   name=value
   ```

   たとえば、*user* と *password* という 2 つの変数を指定する場合は、次のように入力します。

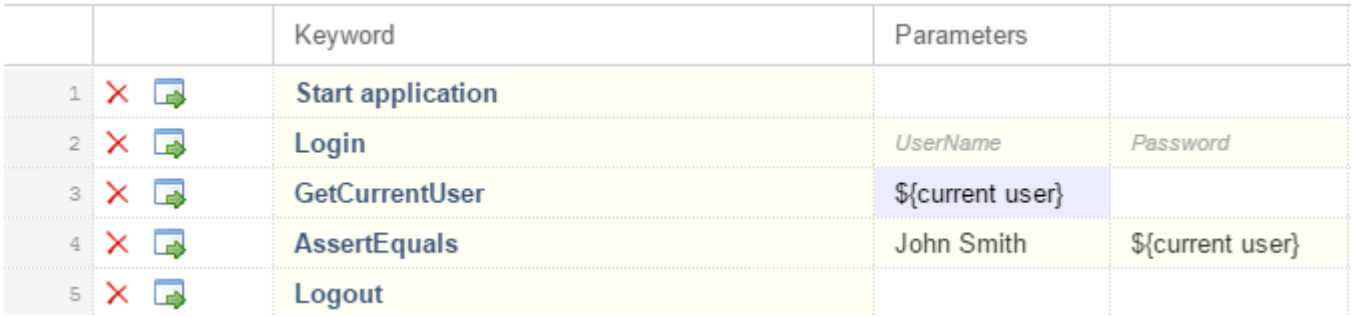
   ```
   user=John
   password=john5673
   ```

   プロパティ ファイルのフォーマットや、空白類などの Unicode 文字の入力方法についての情報は、『*Properties File Format*』を参照してください。
6. globalvariables.properties ファイルを保存します。
7. 実行するキーワード駆動テストを開きます。
8. **キーワード駆動テスト エディター** で、新しい変数を使用するパラメーターを編集します。

   次のように記述します。

   ```
   ${variable  name}
   ```

   たとえば、次のキーワード駆動テストでは、${current user} パラメーターはグローバル変数を使用します。



プロジェクトのキーワード駆動テスト が Silk4J から実行されるときはいつでも、変数が使用されます。

Silk4J と Silk Central の統合

Silk4J と Silk Central を統合することによって、技術者と非技術者のユーザー間で共同作業を行えます。

Silk4J と Silk Central が統合され、Silk Central に存在するライブラリとアクティブな Silk4J プロジェクトが同じ名前であれば、**キーワード** ビュー (**Silk4J** > **キーワード ビューの表示** から開く) には、アクティブな Silk4J プロジェクトで定義されたキーワードに加えて、Silk Central ライブラリのすべてのキーワードが表示されます。

注: Silk Central の接続情報は、すべての Silk4J ユーザーに別々に保存されるため、Silk Central のキーワードおよびキーワード シーケンスで作業するすべての Silk4J ユーザーは、Silk4J を Silk Central と統合する必要があります。

Silk4J と Silk Central を統合すると、次のメリットがあります。

- テスト管理と実行を Silk Central で処理できる
- キーワードが Silk Central データベースに格納され (ライブラリのアップロード)、Silk Central のすべてのプロジェクトで利用できる
- 手動テストを Silk Central で直接自動化し、作成したキーワード駆動テストを Silk Central から Silk4J で実行できる

注: Silk4J では、Silk4J にあるキーワード駆動テストの編集と実行を行うことができ、また、Silk Central に格納されているキーワード駆動テストを実行することができます。Silk Central に格納されているキーワード駆動テストを編集するには、**キーワード駆動テスト エディター** でキーワード駆動テストを開き、**編集** をクリックします。

1. Eclipse メニューから、**Silk4J** > **Silk Central の設定** を選択します。 **設定** ダイアログ ボックスが開きます。
2. **URL** フィールドに、 Silk Central サーバーの URL を入力します。
   たとえば、Silk Central のサーバー名が *sctm-server* で、Silk Central のポート番号が 13450 の場合は、http://sctm-server:13450 と入力します。
3. 認証用 Web サービス トークンを指定します。

   Web サービス トークンは、Silk Central の **ユーザー設定** ページから生成できます。 このページは、Silk Central のメニューに表示されているユーザー名をクリックするとアクセスできます。

   注: Silk Central ユーザー名とパスワードを使って認証を行う場合は、**認証** リストから **ユーザー名とパスワード** を選択します。ただし、セキュリティ上の理由から、ユーザー名とパスワードをネットワーク越しに送信するのではなく、Web サービス トークンを使用することを Micro Focus では推奨しています。
4. 有効な ユーザー名 と パスワード を、それぞれのフィールドに入力します。
5. **検証** をクリックして、Silk4J が指定したユーザーで Silk Central サーバーにアクセスできるかどうか確認します。
6. **OK** をクリックします。

Silk4J での Silk Central キーワードの実装

Silk Central キーワードを実装する前に、Silk Central でキーワード駆動テストの一部としてキーワードを定義します。

Silk4J で Silk Central キーワードを実装するには：

1. Silk4J のプロジェクトをキーワード駆動テストを含んだ Silk Central のキーワード ライブラリと同じ名前で作成します。
2. Silk Central のキーワード ライブラリにタイプが割り当てられていない場合、**Silk4J** > **キーワード ライブラリのアップロード** をクリックして、ライブラリ タイプを設定します。
3. 省略可能：Silk Central の特定のキーワードを Silk4J で実装するには、ライブラリの **キーワード** タブを Silk Central で開き、キーワードの **アクション** 列から **Silk Test で実装** をクリックします。
4. Silk4J メニューで、**Silk4J** > **キーワード ビューの表示** をクリックします。
5. **キーワード** ビューで、キーワード駆動テストをダブル クリックします。
   キーワード ビューで Silk Central による変更を更新するには、**更新** をクリックします。
6. ツールバーで、**操作の記録** をクリックします。
7. 現在のプロジェクトに対してアプリケーション構成が設定されており、Web アプリケーションをテストする場合、**ブラウザーの選択** ダイアログ ボックスが開きます。

a) ブラウザーを選択します。
b) 省略可能：あらかじめ定義されたブラウザー サイズを使用してデスクトップ ブラウザー上の Web アプリケーションをテストする場合は、**ブラウザー サイズ** リストからブラウザー サイズを選択します。
たとえば、Apple Safari 上の Web アプリケーションを Apple iPhone 7 の画面と同じ大きさのブラウザー ウィンドウでテストするには、リストから **Apple iPhone 7** を選択します。
c) 省略可能：ブラウザー ウィンドウの **向き** を選択します。
d) 省略可能：**ブラウザー サイズの編集** をクリックすると、新しいブラウザー サイズを指定したり、**ブラウザー サイズ** リストに表示するブラウザー サイズを選択することができます。

8. **記録** をクリックします。
   記録の詳細については、「キーワードの記録」を参照してください。
9. 最初の未実装のキーワードの操作を記録します。
10. 現在のキーワードに対するすべての操作を記録し終わったら、**次のキーワード** をクリックします。
11. **記録中** ウィンドウでキーワードを切り替えるには、**前のキーワード** と **次のキーワード** をクリックします。
12. **停止** をクリックします。 **記録完了** ダイアログ ボックスが開きます。

注: Silk Central のキーワード駆動テストは、Silk4J では読み取り専用であるため、キーワードを削除したり、キーワードの順番を変更することはできません。

実装したキーワードが **キーワード** ウィンドウで未実装として表示されている場合は、Eclipse メニューで **プロジェクト** > **自動的にビルド** をチェックします。

Silk Central へのキーワード ライブラリのアップロード

Silk Central で作業するためには、有効な Silk Central の場所が設定されている必要があります。詳細については、「*Silk4J と Silk Central の統合*」を参照してください。

Silk Central で自動テストを自動化するために、Silk4J プロジェクトで実装したキーワードをキーワード ライブラリとして Silk Central にアップロードできます。そして、キーワードを使用して手動テストを自動化できます。

1. Silk4J で、キーワード駆動テストが存在するプロジェクトを選択します。
2. 同じ名前のライブラリが Silk Central (**テスト** > **ライブラリ**) に存在していることを確認します。
3. ツールバーで、**キーワード ライブラリのアップロード** をクリックします。
4. 省略可能：キーワード ライブラリに対する変更の説明を指定します。
5. 省略可能：**設定** をクリックして、Silk Central への接続を設定します。
6. 省略可能：接続した Silk Central インスタンスで利用可能なライブラリを確認するには、リンクをクリックします。
7. **アップロード** をクリックします。

注意: Silk Central のキーワード ライブラリが既に他の自動化ツール、つまり他の Silk Test クライアントに割り当てられている場合、キーワード ライブラリのタイプを変更するかどうか確認されます。タイプの変更を行う場合にのみ、ライブラリはアップロードされます。

Silk4J は、プロジェクトで実装されたすべてのキーワードからキーワード ライブラリを作成します。その後、Silk4J は library.zip という名前で、キーワード ライブラリをプロジェクトの出力フォルダに保存します。ライブラリの整合性が検証され、Silk Central の既存のテストが影響を受ける変更が **キーワード ライブラリを Silk Central にアップロード** ダイアログ ボックスに一覧されます。最後に、Silk4J はライブラリを Silk Central にアップロードします。これで、Silk Central でキーワードを使用できるようになります。キーワード ライブラリに含まれるキーワードを使用する Silk Central のキーワード駆動テストは、現在のキーワードの実装を自動的に使用します。

Silk Test 15.5 で作成したプロジェクトからのキーワード ライブラリのアップロード

Silk Test 15.5 で作成した Silk4J プロジェクトからキーワード ライブラリをアップロードする場合は、プロジェクトの build.xml ファイルを編集する必要があります。

1. **パッケージ・エクスプローラー** で、キーワード ライブラリをアップロードするプロジェクトのフォルダを展開します。
2. build.xml ファイルを開きます。
3. プロジェクトの Keyword Assets ディレクトリを *compile* ターゲットの JAR ビルド ステップに追加します。

```
<fileset dir="Keyword Assets" includes="**/*.kwd"
erroronmissingdir="false" />
```

4. キーワード ライブラリ用の次のターゲットを追加します。

```
<target name="build.keyword.library" depends="compile">
  <java classname="com.borland.silk.kwd.library.docbuilder.DocBuilder"
fork="true">
    <classpath refid="project.classpath" />

    <arg value="AutoQuote Silk4J Library" />
    <arg value="${output}" />
    <arg value="${output}/library.zip" />
  </java>
</target>
```

新しい build.xml ファイルは、以下のようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="AutoQuote" default="compile">

  <property name="src" value="src" />
  <property name="bin" value="build" />
  <property name="output" value="output" />
  <property name="lib" value="lib" />
  <property name="buildlib" value="buildlib" />

  <path id="project.classpath">
    <fileset dir="${lib}" includes="*.jar" excludes="*source*" />
    <fileset dir="${buildlib}" includes="*.jar" excludes="*source*" />
  </path>

  <target name="clean">
    <delete dir="${output}" />
  </target>

  <target name="compile" depends="clean">
    <mkdir dir="${output}" />

    <delete dir="${bin}" />
    <mkdir dir="${bin}" />

    <componentdef name="ecj"
classname="org.eclipse.jdt.core.JDTCompilerAdapter"
classpathref="project.classpath" />
    <javac srcdir="${src}" destdir="${bin}" debug="true" source="1.7"
target="1.7" encoding="utf-8" includeantruntime="false">
      <classpath refid="project.classpath" />
      <ecj />
    </javac>

    <jar destfile="${output}/tests.jar" >
```

```
        <fileset dir="${bin}" includes="**/*.class" />
        <fileset dir="${src}" includes="**/*" excludes="**/*.java" />
        <fileset dir="Object Maps" includes="**/*.objectmap"
erroronmissingdir="false" />
        <fileset dir="Image Assets" includes="**/*.imageasset"
erroronmissingdir="false" />
        <fileset dir="Verifications" includes="**/*.verification"
erroronmissingdir="false" />
        <fileset dir="Keyword Assets" includes="**/*.kwd"
erroronmissingdir="false" />
      </jar>

      <copy todir="${output}" overwrite="true">
        <fileset dir="${lib}" includes="*.jar" excludes="*source*" />
      </copy>
      <delete dir="${bin}" />
    </target>

    <target name="build.keyword.library" depends="compile">
      <java
classname="com.borland.silk.kwd.library.docbuilder.DocBuilder"
fork="true">
        <classpath refid="project.classpath" />

        <arg value="AutoQuote Silk4J Library" />
        <arg value="${output}" />
        <arg value="${output}/library.zip" />
       </java>
    </target>
</project>
```

コマンド ラインから Silk Central へのキーワード ライブラリの更新

Java ベースのコマンド ラインから外部キーワード ライブラリを Silk Central にアップロードすることで、Silk Central とキーワード駆動テストを Jenkins などの継続的インテグレーション ビルド システムに統合することができます。

Java ベースのコマンド ラインからキーワード ライブラリを Silk Central にアップロードするには :

1. Silk Central で、**ヘルプ** > **ツール** を選択し、**Java キーワード ライブラリ ツール** をダウンロードします。
2. ダウンロードした jar ファイルに含まれるコマンド ライン ツールを次の引数で呼び出します。
   - java
   - -jar com.borland.silk.keyworddriven.jar
   - -upload
   - 更新または存在しない場合は作成される Silk Central のライブラリの ライブラリ名。
   - アップロードするライブラリ パッケージ (zip アーカイブ) の パッケージ名。
   - Silk Central フロントエンド サーバーのホスト名:ポート。
   - Silk Central ユーザーの Web サービス トークン。認証に必要です。Web サービス トークンは、Silk Central の **ユーザー設定** ページから生成できます。 このページは、 Silk Central のメニューに表示されているユーザー名をクリックするとアクセスできます。

注: セキュリティ上の理由から、ユーザー名とパスワードをネットワーク越しに送信するのではなく、Web サービス トークンを使用することを Micro Focus では推奨しています。

- Silk Central ユーザーの ユーザー名。Web サービス トークンを認証に使用する場合は必要ありません。
- Silk Central ユーザーの パスワード。Web サービス トークンを認証に使用する場合は必要ありません。
- ライブラリに適用された変更を説明する 更新情報 (引用符で囲む)。
- テストまたはキーワード シーケンスで使用されているキーワードの削除を許可するための [-allowUsedKeywordDeletion] フラグ (省略可能)。デフォルトでは、使用中のキーワードを削除しようとするとエラーが発生します。

Java 9 以降を使ってライブラリを Silk Central にアップロードするコマンド ラインは次のようになります。

```
java --add-modules=java.activation,java.xml.ws -jar com.borland.silk.keyworddriven.jar -upload
"My library" "./output/library.zip" silkcentral:19120 scLogin
scPassword "Build xy: Implemented missing keywords"
```

使用例

認証に Web サービス トークンを使い、Java 8 以前を使ってライブラリを Silk Central にアップロードするコマンド ラインは次のようになります。

```
java -jar com.borland.silk.keyworddriven.jar -upload
"My library" "./output/library.zip" silkcentral:19120 scToken "Build xy:
Implemented missing keywords"
```

認証にユーザー名とパスワードを使い、Java 8 以前を使って上記と同じライブラリをアップロードするコマンド ラインは次のようになります。

```
java -jar com.borland.silk.keyworddriven.jar -upload
"My library" "./output/library.zip" silkcentral:19120 scLogin
scPassword "Build xy: Implemented missing keywords"
```

Java 9 以降を使った場合は、それぞれ次のようになります。

```
java --add-modules=java.activation,java.xml.ws -jar
com.borland.silk.keyworddriven.jar -upload
"My library" "./output/library.zip" silkcentral:19120 scToken "Build xy:
Implemented missing keywords"
```

```
java --add-modules=java.activation,java.xml.ws -jar
com.borland.silk.keyworddriven.jar -upload
"My library" "./output/library.zip" silkcentral:19120 scLogin
scPassword "Build xy: Implemented missing keywords"
```

注: Java 9 以降を使ってキーワード駆動ライブラリをアップロードする場合は、実行サーバー上で JAVA_HOME が定義され、対応するバージョンの JDK を指していることを確認してください。

キーワードの検索

キーワード ビュー の検索フィールドを使用して、特定のキーワードを検索します。文字を入力するとリストが更新され、その文字に一致する既存のキーワードが動的に表示されます。検索のヒント：

- 検索では大文字小文字は区別されません。doAction を入力すると、doaction や DOAction が表示されます。

- 大文字のみを入力すると、いわゆる キャメルケース検索が実行されます。ECD を入力すると、Enter Car Details、Enter Contact Details、EnterContactDetails が表示されます。
- キーワード名とグループ名が検索対象になります。test を入力すると、test を含むすべてのキーワードと、グループ名に test を含むグループのすべてのキーワードが表示されます。
- ? は、0 または 1 文字と一致します。user?test を入力すると、userTest や usersTest が表示されます。
- * は、0 または n 文字と一致します。my*keyword を入力すると、myKeyword や myNewKeyword や my_other_keyword が表示されます。
- <文字列>. はグループ名のみを検索します。group. を入力すると、グループ名に group を含むグループのすべてのキーワードが表示されます。
- .<文字列> はキーワード名のみを検索します。.keyword を入力すると、keyword を含むすべてのキーワードが表示されます。
- <string>.<string> は特定のグループのキーワードを検索します。group.word を入力すると、myGroup グループの myKeyword が表示されます。
- 引用符を使用して完全一致のみを検索します。'Keyword' を入力すると、Keyword や MyKeyword は表示されますが、keyword は表示されません。

キーワードのフィルタリング

現在のプロジェクトの特定のキーワードを検索すために、**キーワード** ウィンドウに表示されているキーワードをフィルタすることができます。Silk Central との統合が設定されている場合、結果には Silk Central に関連付けられたキーワードも含まれます。

1. メニューで、**Silk4J** > **キーワード ビューの表示** をクリックして、**キーワード** ウィンドウを開きます。
2. **キーワード** ウィンドウで、検索フィールドに検索するキーワードの名前を入力します。**キーワード** ウィンドウに、現在のプロジェクト内の指定した名前を持つすべてのキーワードが表示されます。
3. 省略可能 : どのキーワード駆動テストまたはキーワード シーケンスでキーワードが使用されているかを確認するには、**キーワード** ウィンドウでキーワード上にマウス カーソルを移動して、**キーワードの利用状況の検索** をクリックします。

   Silk Central との統合が設定されている場合、結果には Silk Central に関連付けられたキーワードも含まれます。
4. 省略可能 : キーワードを編集するには、**キーワード** ウィンドウでキーワード上にマウス カーソルを移動して、**実装へ移動** をクリックします。

キーワードのすべての参照の検索

キーワードが参照されているすべてのキーワード駆動テストおよび Java ファイルを検索するには :

1. **キーワード駆動テスト エディター** で、**キーワードを開く** をクリックします。キーワードが実装されている Java ファイルが開きます。
2. キーワードを実装するメソッドの名前を右クリックします。
3. **参照** をクリックします。
4. キーワードのすべての参照をワークスペースで検索する場合は、**ワークスペース** をクリックします。

キーワードが参照されているすべてのキーワード駆動テストおよび Java ファイルが、**検索** ウィンドウに表示されます。

キーワードのグループ化

キーワードをライブラリで構造化するために、グループ化することができます。

このトピックでは、キーワードを特定のグループに追加する方法について説明します。 このグループ名は Silk Central でも使用され、キーワードはグループ名に従って分類されます。

キーワードを特定のグループに追加するには：

1. キーワードの実装を開きます。
   a) キーワードを実装しているプロジェクトを開きます。
   b) **キーワード** ウィンドウを開きます。
   c) **キーワード** ウィンドウで、キーワードを選択します。
   d) **実装へ移動** をクリックします。
2. クラスのすべてのメソッドをキーワード グループに追加するには、クラス定義の前にキーワード グループを追加します。

   たとえば、キーワードを Calculator グループ追加するには、次のように入力します。

   ```
   @KeywordGroup("Calculator")
   ```

キーワード ウィンドウで表示されるキーワード名にグループが含まれるようになります。たとえば、*Addition* キーワードが *Calculator* グループに属している場合は、Calculator.Addition として表示されます。

キーワード駆動テストのトラブルシューティング

キーワード ウィンドウでキーワードが未実装として表示される

実装したキーワードが **キーワード** ウィンドウで未実装として表示されている場合は、Eclipse メニューで **プロジェクト** > **自動的にビルド** をチェックします。

キーワード駆動テストの再生時に「アプリケーション構成が存在しません」エラーが発生する

このエラーが発生した場合、キーワード駆動テストの最初のキーワードとして アプリケーションの開始 キーワードが含まれていません。 Silk4J では、キーワード駆動テストにプロジェクトのアプリケーション構成を適用するには、アプリケーションの開始 キーワードが必要です。新しいキーワード駆動テストを記録する場合には、Silk4J は、アプリケーションの開始キーワードを最初のキーワードとして自動的にキーワード駆動テストに追加します。この問題を回避するには、テスト対象アプリケーションに対して新しいキーワード駆動テストを記録します。その後、実行中にエラーをスローするキーワード駆動テストを開き、記録した アプリケーションの開始 キーワードを最初のキーワードとしてテストに追加します。

オブジェクト解決

Silk4J は、コントロール クラスの名前と優先付けられた属性のコレクションの組み合わせによって、テスト対象アプリケーション（AUT）のコントロールをユニークな XPath ロケーターに識別します。組み合わせた XPath ロケーターでコントロールをユニークに識別できない場合には、Silk4J は、ロケーターにインデックスを追加します。Silk4J で記録中に、**操作の選択** ダイアログの **ロケーター** フィールドのリストから、コントロールの他のロケーターを選択できます。

Silk Test ロケーターではなく WebDriver ロケーターを記録している場合には、コントロールの識別において、Silk4J は XPath ロケーターの代わりに次のロケーターを提供します。

- ID による識別。 id 属性によってコントロールを識別します。
- 名前による識別。name 属性によってコントロールを識別します。
- リンク テキストによる識別。ハイパーリンクの場合にのみ選択できます。

Silk4J では、識別されたオブジェクトのリテラル参照はロケーターと呼ばれます。Silk4J は、ロケーターを使用して、テスト対象アプリケーション (AUT) のオブジェクトを検索して識別します。ロケーターは、W3C (World Wide Web Consortium) によって定義された 共通の XML ベース言語である XPath クエリー言語のサブセットです。

ロケーターの基本概念

Silk4J は、XPath クエリ言語のサブセットをサポートしています。XPath の詳細については、*http://www.w3.org/TR/xpath20/* を参照してください。

XPath 式は現在のコンテキスト、つまり、Find メソッドを呼び出したオブジェクトの階層上における位置に依存します。ファイル システムと同じように、すべての XPath 式は、この位置に依存します。例：

- "//Shell" は、現在のコンテキストから始まるすべての階層にあるすべての Shell を見つけます。
- "Shell"は、現在のコンテキストの直下の子であるすべての Shell を見つけます。

さらに、ある XPath 式は、コンテキストの影響を受けます。たとえば、myWindow.find(xPath) は、myWindow が現在のコンテキストとなります。

動的オブジェクト解決は、テスト ケース内でオブジェクトを識別するために、Find または FindAll メソッドを使用します。

オブジェクト タイプと検索範囲

典型的なロケーターには、検索するオブジェクトのタイプと検索範囲が含まれます。 検索範囲は以下のいずれかです。

- //
- /

ロケーターは、ロケーターを指定する対象となるオブジェクトである、現在のオブジェクトに依存します。現在のオブジェクトは、アプリケーション UI のオブジェクト階層における 位置を特定します。 ファイル システムと同じように、すべてのロケーターは、この階層における現在のオブジェクトの位置に依存します。

XPath 式は、 現在のコンテキスト、つまり、 Find メソッドを呼び出したオブジェクトの階層上における位置に依存します。 ファイル システムと同じように、すべての XPath 式は、この位置に依存します。

注:

HTML 要素に対するロケーターにおけるオブジェクト タイプは、HTML タグ名または、このオブジェクトに対して Silk4J が使用するクラス名のいずれかになります。 たとえば、ロケーター //a と // DomLink（ここで、 DomLink は Silk4J でのハイパーリンクに対する名前です）は同じです。 HTML ベースでないテクノロジの場合は、 Silk4J クラス名だけが使用されます。

例

- //a は､現在のオブジェクトに相対的なすべての階層にあるハイパーリンク オブジェクトを識別します。
- /a は、現在のオブジェクトの直下の子であるハイパーリンク オブジェクトを識別します。

注: <*a*> は、Web ページのハイパーリンクを表す HTML タグです。

例

以下のコード例は、ブラウザ内の最初のハイパーリンクを識別します。 この例では、実行中のブラウザ インスタンスを参照するスクリプトに *browserWindow* という名前の変数が存在することを仮定しています。 ここで、タイプは "a" で、現在のオブジェクトは *browserWindow* です。

```
DomLink link = browserWindow.<DomLink>find("//a");
```

属性を使用したオブジェクトの識別

プロパティに基づいてオブジェクトを識別するために、ロケーター属性を使用できます。ロケーター属性は、オブジェクトの型の後にかぎ括弧で指定します。

例

次の例では、テキスト *Home* を持つハイパーリンクを識別するために、textContents 属性を使用します。同じテキストを持つハイパーリンクが複数存在する場合は、最初のハイパーリンクがロケーターによって識別されます。

```
DomLink link = browserWindow.<DomLink>find(//
a[@textContents='Home']");
```

ロケーターの構文

Silk4J では、UI コントロールを特定するために、XPath クエリ言語のサブセットをサポートしています。

以下の表に、Silk4J がサポートする構成子を一覧します。

注: <*a*> は、Web ページのハイパーリンクを表す HTML タグです。

以下の表には、Silk4J がサポートしないロケーター構成子を一覧します。

ロケーターの使用

Silk4J では、識別されたオブジェクトのリテラル参照はロケーターと呼ばれます。必要に応じて、ロケーター文字列の短縮形をスクリプトで使用できます。スクリプトを再生すると、Silk4J によって自動的に構

文が展開されて完全なロケーター文字列が使用されます。スクリプトを手動でコーディングする場合は、次の順番で次の部分を省略できます。

- 検索スコープ「//」。
- オブジェクトの型名。Silk4J のデフォルトはクラス名です。
- 属性を囲む角かっこ「[ ]」。

スクリプトを手動で記述する場合は、使用可能な最も短い形式を使用することをお勧めします。

注: オブジェクトを識別する場合は、完全なロケーター文字列がデフォルトでキャプチャされます。

以下のロケーターは同じです。

- 最初の例では、完全なロケーター文字列が使用されています。

  ```
  _desktop.<DomLink>find("//BrowserApplication//BrowserWindow//
  a[@textContents='Home']").select();
  ```

  完全なロケーター文字列を確認するには、**Locator Spy** ダイアログ ボックスを使用します。
- 2 番目の例は、ブラウザー ウィンドウが既に存在する場合に機能します。

  ```
  browserWindow.<DomLink>find("//a[@textContents='Home']").select();
  ```

  または、短縮形を使用することができます。

  ```
  browserWindow.<DomLink>find("@textContents='Home'").select();
  ```

識別のための実際の属性がないオブジェクトを検索するには、インデックスを使用します。たとえば、Web ページの 2 つめのハイパーリンクを選択するには、以下のように入力します。

```
browserWindow.<DomLink>find("//DomLink[2]").select();
```

さらに、その種類の最初のオブジェクトを検索する（このことは、オブジェクトに実際の属性がない場合に便利です）には、以下のように入力します。

```
browserWindow.<DomLink>find("//DomLink").select();
```

ロケーターを使用したオブジェクトの存在確認

Exists メソッドを使用して、オブジェクトがテスト対象アプリケーションに存在するかどうかを確認できます。

次のコードは、「*Log out*」というテキストのハイパーリンクが Web ページに存在するか確認します。

```
if (browserWindow.exists( "//a[@textContents='Log out']" )) {
  // do something
}
```

Find メソッドの使用

Find メソッドや FindOptions メソッドを使用して、後で使用したいオブジェクトが存在するか確認できます。

次のコードは、ウィンドウを検索し、ウィンドウが見つかった場合にウィンドウを閉じます。

```
Window mainWindow = _desktop.<Window>find("//Window[@caption='My Window']", New
FindOptions(False));
if (mainWindow){
  mainWindow.closeSynchron();
}
```

1 つのロケーターで複数のオブジェクトを識別する

FindAll メソッドを使用して、ロケーターに一致する最初のオブジェクトのみを識別するだけでなく、ロケーターに一致するすべてのオブジェクトを識別できます。

例

次のコードの例は、FindAll メソッドを使用して、Web ページのすべてのハイパーリンクを取得します。

```
List<DomLink> links = browserWindow.<DomLink>findAll("//a");
```

ロケーターのカスタマイズ

このセクションでは、テスト対象アプリケーション (AUT) のコントロールを Silk4J が確実に解決できるようにするために、安定したロケーターを作成する方法について説明します。

Silk4J は、AUT がその UI コントロールに対して公開する識別子を利用して、非常に柔軟で強力な UI コントロールの識別方法を提供します。Silk4J は、任意の UI コントロールに対して宣言された任意のプロパティを使用して、UI コントロールの階層を使ってロケーターを作成できます。Silk4J は、それぞれの UI コントロールを識別するのに最も適した項目とプロパティを階層から選択します。

Silk4J は、UI コントロールの階層から多くのコントロールを動的に除外するため、AUT の変更に対して非常に影響を受けにくいオブジェクト解決方法を Silk4J は提供します。Web ページの書式要素のよな UI コントロール ツリーの階層を変更する中間のグループ化されたコントロールは、オブジェクト解決から除外することができます。

UI コントロールによっては、それを固有に識別できるようにする有意義なプロパティを公開しません。このようなコントロールを含んだアプリケーションは低いテスト容易性を持つアプリケーションとみなされます。階層、とくに動的な階層は、このようなアプリケーションに対する固有のロケーターを作成するのに、重要な意味を持ちます。高いテスト容易性を持つアプリケーションは、固有の UI コントロールを識別するための単純な仕組みを常に提供します。

AUT のテストを容易にする最も単純で最も効果的な慣例のひとつが、コントロールに対する安定した識別子を導入し、アプリケーションの既存のインターフェイスを通して、これらの安定した識別子を公開することです。

安定した識別子

UI コントロールの 安定した識別子 とは、コントロールの呼び出しごとに、または UI コントロールが存在するアプリケーションのバージョンが変わっても変更されない識別子を言います。安定した識別子は、その使用されるコンテキストにおいて一意である必要があります。つまり、同じ識別子を持つコントロールが同時にアクセス可能でないことが求められます。つまり、グローバルなコンテキストで一意である GUID 形式の識別子を使用する必要はありません。コントロールの識別子は、可読性が高く、有意な名前であるべきです。これらの識別子の命名規則によって、実際のコントロールに識別子を関連付けるのがより容易になります。

例：キャプションはコントロールの良い識別子と言えるか

ほとんどのテスト ツールでは、UI コントロールのデフォルト識別子として キャプション を使用します。キャプションは、コントロールに関連付けられた UI のテキストです。しかし、UI コントロールを識別するためにキャプションを使用することには、次のような欠点があります。

- キャプションは安定していません。キャプションは開発プロセスの間に頻繁に変更されます。たとえば、AUT の UI が開発プロセスの終わりにレビューされる場合があります。UI が安定していないため、開発プロセスの初期の段階で UI テストを導入することが困難になります。
- キャプションは一意ではありません。たとえば、アプリケーションには **OK** というキャプションを持つボタンが複数存在する可能性があります。
- 多くのコントロールはキャプションを表示しないため、識別するために、ほかのプロパティを使用する必要があります。
- ローカライズしたアプリケーションのテストにキャプションを使用する場合、各言語ごとにコントロールのキャプションを保守する必要があるため、非常に扱いにくく、さらに言語ごとに適切なキャプションを動的に割り当てることができるように、複雑なスクリプト ロジックを保守する必要があります。

安定したロケーターを作成する

Silk4J の主要なメリットのひとつが、柔軟で強力なオブジェクト解決の仕組みです。UI コントロールを特定するために XPath 記法を使用することによって、UI コントロールが適切な属性を持っていない場合でも、適切な属性を持つ対象要素のそばにある限り、Silk4J は確実に識別できます。Silk4J の XPath ロケーターは、UI コントロールを識別するために、UI コントロールの階層全体を使用することもできれば、その一部を使用することもできます。特に最近の AJAX ツールキットは、とても複雑なドキュメント オブジェクト モデル (DOM) を動的に生成するので、UI コントロールを特定するために使用できる適切なコントロール属性を提供しません。

このような場合、インテリジェントなオブジェクト解決の仕組みを提供しないテスト ツールでは、UI コントロールを識別するために、ほとんどの場合インデックス ベースの解決方法を使用することが必要になります。たとえば、展開 アイコンの n 番目のコントロールを識別します。このようなテスト スクリプトは保守することが容易でなく、アプリケーションにほんのわずかな変更を加えるだけでテスト スクリプトが動作しなくなってしまうことが良くあります。

有用な属性を提供しない UI コントロールに対して安定したロケーターを作成する良い方法は、階層の中で安定したロケーターを持つアンカー要素を見つけることです。そして、そのアンカー要素からロケーターを作成したい要素まで辿っていくことができます。

Silk4J は、この方法を使用してロケーターを作成しますが、ときにはコントロールからの安定したロケーターを手動で作成することが必要となる場合もあります。

例：コントロールの同列要素の指定

このトピックでは、ロケーターに使用することのできる安定した属性を提供しないコントロールを、そのコントロールの同列要素に対する安定したロケーターが利用できる場合に、それを使用して指定する方法を説明します。

次の安定したロケーターを持つコントロール **Item 0.0** が既に識別されていると想定します。

```
/BrowserApplication//BrowserWindow//DIV[@textContent='Item  0.0']
```

Item 0.0 が、タイプ *a* の後ろに同列要素を持っていることがわかっている場合、次のコードを使用して、同列要素の安定したロケーターを構築できます。

```
/BrowserApplication//BrowserWindow//DIV[@textContent='Item  0.0']/following-sibling::a
```

テキスト フィールドを識別するときにも、この同列要素による方法を使用できます。テキスト フィールドは、たいていの場合、ロケーターで使用できる有用な属性を提供しません。テキスト フィールドのラベルを使用すると、テキスト フィールドの有用なロケーターを作成できます。同列要素による方法を使って、テキスト フィールドのロケーターの一部として、ラベルを使用することは簡単です。たとえば、テキスト フィールドがテキスト **User Name** を持つラベルの前に同列要素である場合、次のロケーターを使用できます。

```
/BrowserApplication//BrowserWindow//DIV[@textContent='User  Name']/preceding-
sibling::input[@type='text']
```

例：動的 GWT ツリーの展開アイコンの検索

Google Widget Toolkit (GWT) は、とても人気のある強力なツールキットですが、テストしにくいです。動的ツリー コントロールは、とても一般的に使用されている GWT の UI コントロールです。ツリーを展開するには、**展開** アイコン要素を識別する必要があります。

動的 GWT ツリーのサンプルは、*http://samples.gwtproject.org/samples/Showcase/Showcase.html#!CwTree* にあります。

Silk4J が生成するデフォルトのロケーターは次のようになります。

```
/BrowserApplication//BrowserWindow//DIV[@id='gwt-debug-cwTree-dynamicTree-root-child0']/DIV/DIV[1]//IMG[@border='0']
```

次の理由で、このデフォルトのロケーターは、**Item 0.0** の **展開** アイコンを識別するロケーターとしては信頼できるものではありません。

- ロケーターは複雑で、複数の階層から構成されています。AJAX で動的に DOM 構造が少し変わるとロケーターは使えなくなります。
- ロケーターには、階層のいくつかのコントロールにインデックスが含まれています。インデックス ベースのロケーターは、その出現番号によってコントロールを検索するため、一般にもろく、たとえば、ツリーの 6 番目の展開アイコンを見つけるなど、うまく特定のコントロールを定義できません。このルールの例外は、たとえばグリッドの 6 番目のデータ行など、識別するさまざまなデータ セットを表すためにインデックスが使用される場合です。

多くの場合、より良いロケーターを見つける良い方法は、検索する要素の同列要素を探し出すことです。より良いロケーターの同列要素を見つけると、XPath は、これらの同列要素を識別してロケーターを構成することができます。この場合、ツリー項目 **Item 0.0** は、**展開** アイコンよりも良いロケーターです。ツリー項目 **Item 0.0** のロケーターは、コントロールの @textContents プロパティを使用するため、安定した単純なロケーターです。

デフォルトでは、Silk4J は @id プロパティを使用しますが、GWT では @id には ='gwt-uid-<nnn>' のような値 (ここで、<nnn> は同じ要素でも呼出し毎に頻繁に変わります) が含まれるため、たいてい安定したプロパティではありません。

@textContents プロパティを @id の代わりに使用してロケーターを手動で変更できます。

元のロケーター：

```
/BrowserApplication//BrowserWindow//DIV[@id='gwt-uid-109']
```

別のロケーター：

```
/BrowserApplication//BrowserWindow//DIV[@textContents='Item  0.0']
```

もしくは、@id='gwt-uid-<nnn>' を使用しないように Silk4J を設定できます。この場合、Silk4J は自動的に安定したロケーターを記録します。たとえば、@id プロパティで使用されるテキスト パターンをロケーター属性値除外リストに追加ます。この場合、gwt-uid* を除外リストに追加します。

要素の階層を調べると、**Item 0.0** コントロールと **展開** アイコン コントロールは、共通のルートノードとして DomTableRow コントロールを持つことが分かります。

展開 アイコンの安定したロケーターを作成するには、次のロケーターを使って **Item 0.0** をまず検索する必要があります。

```
/BrowserApplication//BrowserWindow//DIV[@textContent='Item  0.0']
```

そして、要素の階層を 2 レベル上がって DomTableRow 要素まで移動します。これは、XPath では、ロケーターに /../.. を追加して表現します。最後に、DomTableRow から **展開** アイコンを検索します。**展開** アイコンは、サブツリー内では唯一の IMG コントロールであるので、容易に検索できます。これは、XPath では、ロケーターに //IMG を追加して表現します。**展開** アイコンの最終的な安定したロケーターは次のようになります。

```
/BrowserApplication//BrowserWindow//DIV[@textContent='Item  0.0']/../..//IMG
```

XPath の ancestor 軸を使うと、より簡単に **展開** アイコンを指定できます。

```
/BrowserApplication//BrowserWindow//DIV[@textContent='Item  0.0']/ancestor::tr//IMG
```

カスタム属性

多くの UI テクノロジーは、UI コントロールのあらかじめ定義された属性のセットをカスタム属性で拡張する方法を提供します。アプリケーション開発者は、コントロールを一意に識別する安定した識別子を導入するためにカスタム属性を使用できます。Silk4J は、UI コントロールのカスタム属性にアクセスでき、UI コントロールを識別するために、これらのカスタム属性を使用することもできます。

UI コントロールを識別するために特別に自動化用属性を使用すると、caption のような定義済み属性を使用する場合と比較して、いくつかのメリットを享受できます。アプリケーション コードで安定した識別子を指定でき、カスタム属性や定義済みの自動化用プロパティの何れかを通して識別子を公開することで、テスト自動化スクリプトが理解しやすくなり、保守性も高まり、開発プロセスの初期の段階からテストの自動化を開始することができるようになります。

Silk4J はロケーター生成の柔軟性が高く、識別に使用する属性を設定することができます。

Apache Flex アプリケーションのカスタム属性

Apache Flex アプリケーションは、あらかじめ定義されたプロパティ automationName を使用して、次のように Apache Flex コントロールに対して安定した識別子を指定します。

```
<?xml  version="1.0"  encoding="utf-8"?>
   <s:Group  xmlns:fx="http://ns.adobe.com/mxml/2009"
         xmlns:s="library://ns.adobe.com/flex/spark"
         xmlns:mx="library://ns.adobe.com/flex/mx"  width="400"  height="300">
      <fx:Script>
      ...
      </fx:Script>
      <s:Button  x="247"  y="81"  label="Button"  id="button1"  enabled="true"
click="button1_clickHandler(event)"
            automationName="AID_buttonRepeat"/>
      <s:Label  x="128"  y="123"  width="315"  height="18"  id="label1"  verticalAlign="middle"
            text="awaiting  your  click"  textAlign="center"/>
   </s:Group>
```

Apache Flex アプリケーションのロケーターは次のようになります。

```
...//SparkApplication//SparkButton[@caption='AID_buttonRepeat'
```

注目: Apache Flex アプリケーションの場合、Silk4J では *automationName* はロケーター属性 caption に常にマップされます。*automationName* 属性が指定されていない場合、Silk4J は属性 ID をロケーター属性 caption にマップします。

Java SWT カスタム属性

カスタム属性をテスト アプリケーションに追加して、テストをより安定させることができます。 たとえば、Java SWT では、GUI を実装する開発者が属性（'silkTestAutomationId' など）をウィジェットに対して定義することによって、アプリケーション内でそのウィジェットを一意に識別することができます。これにより、Silk4J を使用するテスト担当者は、その属性（この場合は 'silkTestAutomationId'）をカスタム属性のリストに追加すると、その一意の ID によってコントロールを識別できるようになります。 カスタム属性を使用すると、caption や index のような他の属性よりも高い信頼性を得ることができます。これは、caption はアプリケーションを他の言語に翻訳した場合に変更され、index は定義済みのウィジェットより前に他のウィジェットが追加されると変更されるためです。

複数のオブジェクトに同じカスタム属性の値が割り当てられた場合は、そのカスタム属性を呼び出したときにその値を持つすべてのオブジェクトが返されます。 たとえば、一意の ID として 'loginName' を 2 つの異なるテキスト フィールドに割り当てた場合は、'loginName' 属性を呼び出したときに、両方のフィールドが返されます。

Java SWT の例

以下のコードを使用して、テストするアプリケーションにボタンを作成する場合：

```
Button myButton = Button(parent, SWT.NONE);

myButton.setData("SilkTestAutomationId", "myButtonId");
```

テストの XPath クエリ文字列に属性を追加するには、以下のクエリを使用します。

```
Dim button =
desktop.PushButton("@SilkTestAutomationId='myButton'")
```

Java SWT アプリケーションをカスタム属性のテストに対して有効化にするには、開発者はカスタム属性をアプリケーションに含める必要があります。 属性を含めるには org.swt.widgets.Widget.setData(String key, Object value) メソッドを使用します。

Web アプリケーションのカスタム属性

HTML は、安定した識別子を表すことができる一般的な属性 ID を定義します。定義により、ID は文書内の要素を一意に識別します。特定の ID を持つ要素は文書内で 1 つだけ存在します。

ただし、多くの場合 (特に AJAX アプリケーションでは)、ID は HTML 要素に関連付けられたサーバー ハンドラを動的に識別するために使用されます。つまり、Web 文書の作成のたびに ID は変わることになります。このような場合、ID は安定した識別子ではなく、Web アプリケーションの UI コントロールを識別するのに適しません。

Web アプリケーションの場合、より確実にするには、Silk4J に UI コントロールの情報を公開するためだけに使用されるカスタム HTML 属性を新たに導入することです。

カスタム HTML 属性はブラウザーは無視するため、AUT の動作は変わりません。ブラウザーの DOM を通してアクセスすることができます。Silk4J では、このような属性を (属性がコントロール クラスのカスタム 属性であっても) 識別時のデフォルト属性として使用するように設定することができます。特定のテクノロジ ドメインのデフォルト識別属性としてカスタム属性を設定するには、**Silk4J** > **オプションの編集** > **カスタム属性** をクリックして、テクノロジ ドメインを選択します。

アプリケーション開発者は、Web 要素にさらに HTML 属性を追加することが必要です。

元の HTML コード：

```
<A HREF="http://abc.com/control=4543772788784322..." <IMG
src="http://abc.com/xxx.gif" width=16 height=16> </A>
```

新しいカスタム HTML 属性 *AUTOMATION_ID* を持つ HTML コード：

```
<A HREF="http://abc.com/control=4543772788784322..."
AUTOMATION_ID = "AID_Login" <IMG src="http://abc.com/xxx.gif"
width=16 height=16> </A>
```

カスタム属性を設定すると、Silk4J は、できる限りカスタム属性を使用して、一意のロケーターを構成しようとします。Web ロケーターは次のようになります。

```
…//DomLink[@AUTOMATION_ID='AID_Login'
```

例：変化する ID

変化する ID の 1 例は、Google Widget Toolkit (GWT) で、ID は Web 文書の作成のたびに変化する動的な値を保持します：

```
ID = 'gwt-uid-<nnn>'
```

この場合、<nnn> が頻繁に変化します。

Windows Forms アプリケーションのカスタム属性

Windows Forms アプリケーションは、あらかじめ定義された自動化用プロパティ automationId を使用して、Windows Forms コントロールに対して安定した識別子を指定します。

Silk4J は、ロケーターを識別するために、自動的にこのプロパティを使用します。Windows Forms アプリケーションのロケーターは次のようになります。

```
/FormsWindow//PushButton[@automationId='btnBasicControls']
```

WPF アプリケーションのカスタム属性

WPF アプリケーションは、あらかじめ定義された自動化用プロパティ AutomationProperties.AutomationId を使用して、次のように WPF コントロールに対して安定した識別子を指定します。

```
<Window  x:Class="Test.MainWindow"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      Title="MainWindow"  Height="350"  Width="525">
   <Grid>
      <Button  AutomationProperties.AutomationId="AID_buttonA">The
Button</Button>
   </Grid>
</Window>
```

Silk4J は、ロケーターを識別するために、自動的にこのプロパティを使用します。WPF アプリケーションのロケーターは次のようになります。

```
/WPFWindow[@caption='MainWindow']//WPFButton[@automationId='AID_buttonA']
```

XPath のパフォーマンス問題のトラブルシューティング

複雑な Web アプリケーションの場合など、オブジェクトの構造が複雑なアプリケーションをテストする場合、パフォーマンスの問題や、スクリプトの信頼性に関連する問題が発生することがあります。このトピックでは、Silk4J が記録中に自動的に生成したロケーターとは異なるロケーターを使用して、スクリプトのパフォーマンスを向上させる方法について説明します。

注: 一般に、複雑なロケーターを使用することは推奨しません。複雑なロケーターを使用すると、テストの信頼性を損なう恐れがあります。複雑なロケーターは、テスト アプリケーションの構造をほんの少し変更しただけで機能しなくなってしまう可能性があります。それにもかかわらず、スクリプトのパフォーマンスが要求を満たしていない場合には、より固有のロケーターを使用することによってテストのパフォーマンスを向上できる可能性があります。

例として、MyApplication アプリケーションの要素ツリーを以下に示します。

```
Root
  Node  id=1
     Leaf  id=2
     Leaf  id=3
     Leaf  id=4
     Leaf  id=5
  Node  id=6
     Node  id=7
        Leaf  id=8
        Leaf  id=9
     Node  id=9
        Leaf  id=10
```

以下の最適化手法のいくつかを使用して、スクリプトのパフォーマンスを改善させることができます。

- 複雑なオブジェクト構造内の要素を特定したい場合は、オブジェクト構造全体ではなく、その特定の部分だけを検索するようにします。たとえば、サンプル ツリーの識別子 4 を持つ要素を検索する場合に Root.Find("//Leaf[@id='4']") というクエリーを使用している場合、Root.Find("/Node[@id='1']/Leaf[@id='4']") というクエリーで置き換えます。最初のクエリーでは、識別子 4 を持つリーフが、アプリケーションの要素ツリー全体から検索されます。そして、最初のリーフが見つかった時点で返されます。2 番目のクエリーでは、識別子 1 を持つノードと識別子 6 を持つノードがある最初のレベルのノードがまず検索された後、識別子 4 を持つすべてのリーフが識別子 1 を持つノードのサブツリー内から検索されます。
- 同じ階層内の複数の項目を特定したい場合は、まずは階層を特定してからループ内で項目を特定します。Root.FindAll("/Node[@id='1']/Leaf") というクエリーを使用している場合、次のようなループで置き換えます。

```
public void test() {
   TestObject node;
   int i;

   node = desktop.find("//Node[@id='1']");
   for (i=1; i<=4; i++)
   node.find("/Leaf[@id='"+i+"']");
}
```

Locator Spy

Locator Spy を使用して、テスト対象アプリケーション (AUT) の任意のコントロールに対する一意の Silk Test ロケーター、または WebDriver ロケーターを記録することができます。コントロールのロケーターやコントロールの属性を、**Locator Spy** からスクリプトのメソッドにコピーすることができます。コントロールのロケーターの属性を編集して、変更を検証するために、**Locator Spy** を使用することもできます。**Locator Spy** を使用することで、コントロールのロケーターがが正しいことが保証されます。

Locator Spy のオブジェクト ツリーには、AUT で利用可能なすべてのコントロールがリストされます。AUT の利用可能なコントロールとコントロールの階層を調べるためにオブジェクト ツリーを使用できます。WebDriver ロケーターを記録する場合、オブジェクト ツリー上の「>」は、ある IFrame から他に切り替わることを意味します。

注: Locator Spy のロケーター属性テーブルには、ロケーターで使用できるすべての属性が表示されます。Web アプリケーションの場合は、記録中に無視するように定義したすべての属性もテーブルに含まれます。

オブジェクト マップ

オブジェクト マップはテスト資産の一種であり、コントロールまたはウィンドウのロケーターではなく、コントロールまたはウィンドウに論理名 (エイリアス) を関連付ける項目が含まれています。コントロールがオブジェクト マップ資産に登録されると、スクリプトでのそのコントロールに対する参照はすべて、実際のロケーター名ではなく、そのエイリアスによって行われます。

複数のスクリプトで頻繁に使用するオブジェクトを格納するために、オブジェクト マップを使用できます。複数のテストで 1 つのオブジェクト マップ項目の定義を参照できるため、ユーザーがそのオブジェクト マップ定義を 1 回更新すると、オブジェクト マップ定義を参照するすべてのテストでそのオブジェクト マップ定義が Silk4J によって更新されます。

スクリプトで、オブジェクト マップ識別子とロケーターを混在させることができます。この機能により、オブジェクト マップを比較的小さいまま保ち、管理しやすくすることが可能です。共通で使用するオブジェクトをオブジェクト マップに格納し、まれにしかしようしないオブジェクトを参照するにはロケーターを使用します。

ヒント: オブジェクト マップが提供する機能を最適に使用するには、テストしたいアプリケーションごとに個々のプロジェクトを Silk4J に作成します。

オブジェクト マップの例

以下の構成では、ロケーターが使用されている BrowserWindow の定義が示されています。

```
_desktop.BrowserApplication("cnn_com").BrowserWindow("//
BrowserWindow[1]")
```

オブジェクト マップ資産の名前は *cnn_com* です。オブジェクト マップのエイリアスによって置き換えることができるロケーターは、以下のとおりです。

```
"//BrowserWindow[1]"
```

BrowserWindow のオブジェクト マップ エントリは *BrowserWindow* です。

結果的に、スクリプト内の BrowserWindow の定義は以下のようになります。

```
_desktop.BrowserApplication("cnn_com").BrowserWindow("BrowserWindow")
```

ロケーターのインデックスが変更された場合、テスト スクリプトのロケーターのすべての外観を変更する必要はなく、オブジェクト マップのエイリアスを変更するだけで済みます。Silk4J によって、オブジェクト マップ定義を参照するすべてのテストが更新されます。

オブジェクト マップ識別子とロケーターを混在させる例

つぎのサンプル コードは、オブジェクト マップ識別子と、オブジェクト マップに格納されたオブジェクトのまれに使用される子オブジェクトを指定するロケーターを混在させる方法を示します：

```
Window window = _desktop.find("MyApplication"); // object map id - the
application window is used often
MenuItem aboutMenuItem = _desktop.find("//
MenuItem[@caption='About']"); // locator - the About dialog is only
used once
aboutMenuItem.select();
```

オブジェクト マップを使用する利点

オブジェクト マップには、以下の利点があります。

- オブジェクト マップ項目のロケーターに加えられた変更を、対応するオブジェクト マップ項目を含むすべてのテストに適用することによって、テストのメンテナンスが簡単になる。
- 大規模な機能テスト環境において、ロケーターの扱いが容易になる。
- 個々のスクリプトから独立して管理することができるようになる。
- 複雑なロケーター名がわかりやすい名前で置き換えられるため、スクリプトが読みやすくなる。
- テスト アプリケーションが変更された場合に変わる可能性のあるロケーターに依存しなくなる。

オブジェクト マップのオン／オフの切り替え

記録時に Silk4J でロケーター名またはオブジェクト マップのエイリアスのいずれを使用するかを設定できます。

記録中にオブジェクト マップからエイリアスを使用するには、以下を実行します。

1. **Silk4J** > **オプションの編集** をクリックします。
2. **記録** をクリックします。
3. オブジェクト マップ エントリを記録するか、XPath ロケーターを記録するかを定義するには、**OPT_RECORD_OBJECTMAPS_MODE** リストから適切な記録モードを選択します。
   - **オブジェクト マップ エントリ (新しいオブジェクトと既存のオブジェクト)**。これはデフォルトのモードです。
   - **XPath ロケーター (新しいオブジェクトと既存のオブジェクト)**。
   - **XPath ロケーター (新しいオブジェクトのみ)**。オブジェクト マップに既に存在するオブジェクトに対しては、オブジェクト マップ エントリが再利用されます。この設定を選択すると、AUT のメイン コントロールに対するオブジェクト マップを作成し、AUT に対して追加のテストを作成する間、これらのオブジェクト マップを保持することができます。

注: XPath 属性のほかに、ロケーターの記録中にオブジェクト マップをマージする際に、Silk4J は要素の追加の属性を使用します。ただし、記録したスクリプトでオブジェクト マップ ID の用法を曖昧にする可能性のある属性は既存のオブジェクト マップ エントリにロケーターをマップするために使用されません。

注: オブジェクト マップを記録する 設定を有効にすると、Silk4J 全体にわたって、ロケーター名の代わりにオブジェクト マップの項目名が表示されます。たとえば、**プロパティ** ペインで **アプリケーション構成** カテゴリを表示する場合、ロケーター名ではなくオブジェクト マップ項目名が **ロケーター** ボックスに表示されます。

複数のプロジェクトでの資産の使用

Silk4J では、イメージ資産、イメージ検証、およびオブジェクト マップが資産と呼ばれます。資産が配置されているプロジェクトのスコープ外でそれらの資産を使用する場合、資産を使用するプロジェクトから、資産を配置するプロジェクトに、プロジェクトの直接的な依存関係を追加する必要があります。Eclipse からテストを再生する場合、すべての依存プロジェクトがテストを実行するクラスパスに追加されます。このため、Silk4J は依存プロジェクトの資産も見つけることができます。

再生中に資産が使用されると、Silk4J は、最初に現在のプロジェクト内でその資産を検索します。現在のプロジェクトは、現在実行されるテスト コードを含んだ JAR ファイルです。Silk4J で現在のプロジェクト内に資産が検出されなかった場合、Silk4J は現在のプロジェクトがプロジェクト クラスパス内のすべての

他のプロジェクトを持つプロジェクトを追加検索します。それでも資産が見つからない場合、Silk4J はエラーをスローします。

複数のプロジェクトに同じ名前の資産が存在する場合に、現在のプロジェクトに含まれている資産を使用しないときは、資産を使用するメソッドで使用する特定の資産を定義できます。使用する資産を定義するには、メソッドを呼び出すときに、資産の名前空間を接頭辞として資産名に追加します。資産の名前空間は、デフォルトでプロジェクト名に設定されます。

注: Silk4J での作業を開始すると、資産の名前空間オプションが、前のバージョンの Silk4J で作成されたワークスペースにある各 Silk4J の silk4j.settings ファイルに追加されます。

例：プロジェクトの依存関係の追加

プロジェクト *ProjectA* にコード

```
window.imageClick("imageAsset");
```

を呼び出すテストが含まれており、イメージ資産 *imageAsset* がプロジェクト *ProjectB* に置かれている場合、プロジェクトの直接的な依存関係を *ProjectA* から *ProjectB* に追加する必要があります。

Eclipse にプロジェクト依存関係を追加するには、プロジェクトを右クリックし、**プロパティ**を選択します。**Java のビルド・パス** を選択い、**プロジェクト** タブをクリックして、ここにプロジェクトを追加します。

注: プロジェクト参照 を **Java のビルド・パス** の代わりに設定しても機能しません。

例：特定の資産の呼び出し

ProjectA と *ProjectB* の両方に *anotherImageAsset* という名前のイメージ資産が含まれている場合に、*ProjectB* からイメージ資産を明示的にクリックする場合、次のコードを使用します：

```
window.imageClick("ProjectB:anotherImageAsset")
```

操作の記録中でのオブジェクト マップのマージ

Silk4J を使用して操作を記録するときに、Silk4J は、既存のオブジェクト マップ エントリが再利用できるかどうか確認します。Silk4J は、新しいロケーターが生成されるときに、記録中に直接確認します。Silk4J は、テスト対象アプリケーションで現在記録されているオブジェクトが既存のオブジェクト マップ エントリと完全に一致するかどうか確認し、一致する場合に Silk4J はオブジェクト マップからそのオブジェクト マップ識別子を再利用します。

この動作には以下のような利点があります。

- オブジェクト マップのロケーターが変更された場合でも、Silk4J は記録中にオブジェクト マップ識別子を正しく再利用します。
- 記録したスクリプトに間違ったオブジェクト マップ識別子を含むはずがないため、間違ったオブジェクト マップ識別子によって再生に失敗することは決してありません。
- 階層のレベルをさらに追加した場合など、オブジェクト マップを再構成した場合でも、オブジェクト マップ識別子を再利用することができます。

例

Micro Focus Web サイト (*http://www.borland.com*) の **Products** リンクをクリックしたとき、Silk4J は、次のスクリプトを記録します。

```
With _desktop.BrowserApplication( "borland_com" )
  With .BrowserWindow( "BrowserWindow" )
```

```
      .DomLink( "Products" ).Click( MouseButton .Left, New Point (47, 18))
   End With
End With
```

記録したオブジェクト マップは次のようになります。

```
borland_com                                    //BrowserApplication
   BrowserWindow                                    //BrowserWindow
      Products                                       //
A[@textContents='Products']
```

ここで、Micro Focus Web サイトのヘッダー部分を含むようにオブジェクト マップを手動で再構成した場合を考えます。

```
borland_com                                    //BrowserApplication
   BrowserWindow                                    //BrowserWindow
      header                                         //
HEADER[@role='banner']
         Products                                          //
A[@textContents='Products']
```

Products リンクのクリックを記録すると、オブジェクト マップが正しく再利用され、次のスクリプトが記録されます。

```
With _desktop.BrowserApplication( "borland_com" )
   With .BrowserWindow( "BrowserWindow" )
      .DomElement("header").DomLink( "Products" ).Click( MouseButton .Left
, New Point (47, 18))
   End With
End With
```

注: About リンクなどの Micro Focus Web サイトのヘッダー部分にあるほかのオブジェクトを記録すると、Silk4J は **header** ではなく、**BrowserWindow** の子として **About** オブジェクト マップ エントリを追加します。

Web アプリケーションでのオブジェクト マップの使用

デフォルトで、Web アプリケーションに対する操作を記録すると、Silk4J は、ネイティブ ブラウザのコントロール用に *WebBrowser* という名前のオブジェクト マップを作成し、各 Web ドメイン用にオブジェクト マップ資産を作成します。

印刷または設定用のメイン ウィンドウやダイアログ ボックスなど、Web ドメインに特有ではない共通のブラウザ コントロールの場合、*WebBrowser* という名前を使用して、現在のプロジェクトに追加のオブジェクト マップが生成されます。

オブジェクト マップで、オブジェクト マップのエントリのグループ化に使用される URL パターンを編集できます。パターンを編集すると、Silk4J はそのパターンの構文検証を行います。パターンには、ワイルドカード * および ? を使用できます。

例

http://www.borland.com および *http://www.microfocus.co.jp* で何らかの操作を記録した後、プリンタ ダイアログを開くと、次の 3 つの新しいオブジェクト マップ資産が**アセット ブラウザ**に追加されます。

- WebBrowser
- borland_com
- microfocus_com

注: Silk4J では、オブジェクト マップのないプロジェクトに対してのみ、新しいオブジェクト マップ資産が生成されます。バージョン 14.0 よりも前のバージョンの Silk4J を使用して生成されたオブジェクト マップをすでに含む Silk4J の Web アプリケーションに対する操作を記録すると、追加で記録されたエントリは既存のオブジェクト マップに保存されます。Web ドメインに対して追加のオブジェクト マップ資産が生成されることはありません。

オブジェクト マップ項目名の変更

オブジェクト マップでは、項目とロケーターの名前を手動で変更できます。

警告: オブジェクト マップ項目の名前を変更すると、その項目を使用するすべてのスクリプトが影響を受けます。たとえば、**キャンセル** ボタンのオブジェクト マップ項目の名前を **CancelMe** から **Cancel** に変更すると、**CancelMe** を使用するすべてのスクリプトを、**Cancel** を使用するように手動で変更する必要があります。

オブジェクト マップ項目は一意である必要があります。重複するオブジェクト マップ項目を追加しようとすると、オブジェクト マップ項目は一意である必要があることが Silk4J から通知されます。

無効な文字またはロケーターを使用すると、項目名またはロケーター テキストが赤で表示され、ツール ヒントにエラーの説明が表示されます。オブジェクト マップ項目として無効な文字は、¥、/、<、>、"、:、*、?、|、=、.、@、[、] です。無効なロケーター パスは、空または不完全なロケーター パスです。

1. **パッケージ エクスプローラー** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。
2. 次のいずれか 1 つを選んでください :
   - 名前を変更するオブジェクト マップ項目を含むオブジェクト マップをダブルクリックします。
   - 名前を変更するオブジェクト マップ項目を含むオブジェクト マップを右クリックし、**開く** を選択します。

   オブジェクト マップ項目および各項目に関連付けられたロケーターの階層が、オブジェクト マップに表示されます。
3. 名前を変更するオブジェクト マップ項目に移動します。
   たとえば、名前を変更する項目を検索するには、ノードの展開が必要な場合があります。
4. 名前を変更するオブジェクトをクリックしてから、オブジェクトを再度クリックします。
5. 使用する項目名を入力し、Enter を押します。

   無効な文字を使用すると、項目名が赤で表示されます。

   新しい名前が **項目名** リストに表示されます。
6. **Ctrl+S** を押して、変更を保存します。

注: オブジェクト マップ ツリーに含まれるすべてのノードのすべての子ノードは、オブジェクト マップを保存するときにアルファベット順にソートされます。

変更した項目名を既存のスクリプトで使用する場合は、新しい項目名を使用するようにスクリプトを手動で変更する必要があります。

注: Web アプリケーションまたはモバイル Web アプリの記録中に、**操作の選択** ダイアログでオブジェクト マップ エントリの名前を直接変更できます。オブジェクトを右クリックして、**操作の選択** ダイアログの **オブジェクトの識別** 領域を展開します。そして、**オブジェクト マップ ID** フィールドでオブジェクト マップ エントリを編集できます。この機能は、次のブラウザーに対するテストで利用できます。

- Microsoft Edge
- Apple Safari
- Mozilla Firefox 41 以降
- Google Chrome 50 以降

- モバイル ブラウザー

オブジェクト マップの変更

既存のオブジェクト マップは、オブジェクト マップに構造化要素をさらに追加したとしても、記録中に既存のオブジェクト マップ識別子を再利用することができます。

例：既存のオブジェクト マップへの DIV の追加

次の単純なオブジェクト マップの email フィールドと login フィールドをまとめる DIV 要素を追加することを考えます。

```
demo_borland_com                                    //BrowserApplication
   BrowserWindow                                          //
BrowserWindow
      login-form  email                                  //
INPUT[@id='login-form:email']
      login-form  login                                  //
INPUT[@id='login-form:login']
```

新たに DIV *loginArea* を追加することにより、オブジェクト マップの構造を変更できますが、オブジェクト マップは、記録中にオブジェクト マップ識別子は正しく再利用することができます。

```
demo_borland_com                                    //BrowserApplication
   BrowserWindow                                          //
BrowserWindow

loginArea                                                 //'DIV[@id='logi
n']
         login-form  email                                //
INPUT[@id='login-form:email']
         login-form  login                                //
INPUT[@id='login-form:login']
```

オブジェクト マップのロケーターの変更

スクリプトを記録するときに、ロケーターは自動的にオブジェクト マップ項目に関連付けられます。ただし、より汎用的にするために、ロケーター パスを変更できます。たとえば、テスト アプリケーションで特定のコントロールに自動的に日付または時刻が割り当てられる場合、ワイルドカードを使用するようにそのコントロールのロケーターを変更できます。ワイルドカードを使用すると、それぞれのテストで異なる日付または時刻が挿入される場合でも、各テストで同じロケーターを使用できます。

1. **パッケージ エクスプローラー** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。
2. 次のいずれか 1 つを選んでください：
   - 変更するロケーターを含むオブジェクト マップをダブルクリックします。
   - 変更するロケーターを含むオブジェクト マップを右クリックし、**開く** を選択します。

   オブジェクト マップ項目および各項目に関連付けられたロケーターの階層が、オブジェクト マップに表示されます。
3. 変更するロケーターに移動します。
   たとえば、変更するロケーターを検索するには、ノードの展開が必要な場合があります。
4. 変更するロケーター パスをクリックしてから、ロケーター パスを再度クリックします。

5. 有効なロケーター パスがある場合は、使用する項目名とロケーター パスを入力して Enter を押すことができます。有効なロケーター パスを判別するには、以下のステップで説明するように、**Locator Spy** ダイアログ ボックスを使用します。
   a) Silk4J ツール バーで、**Locator Spy** をクリックします。
   b) 記録したいオブジェクトの上にマウスを移動して **Ctrl+Alt** を押します。Silk4J の **ロケーター** テキスト フィールドにロケーター文字列が表示されます。
   c) **ロケーターの詳細** テーブルで、使用するロケーターを選択します。
   d) ロケーターをコピーしてオブジェクト マップに貼り付けます。
6. 必要に応じて、ニーズに合わせて項目名またはロケーター テキストを変更します。

   無効な文字またはロケーターを使用すると、項目名またはロケーター テキストが赤で表示され、ツール ヒントにエラーの説明が表示されます。

   オブジェクト マップ項目として無効な文字は、¥、/、<、>、"、:、*、?、|、=、.、@、[、] です。

   無効なロケーター パスは、空または不完全なロケーター パスです。
7. **Ctrl+S** を押して、変更を保存します。

変更したロケーター パスが既存のスクリプトによって使用されている場合は、新しいロケーター パスを使用するように、そのビジュアル テストまたはスクリプトを手動で変更する必要があります。

テスト アプリケーションからのオブジェクト マップの更新

テスト アプリケーションの項目が変化した場合は、**オブジェクト マップ** UI を使用してそれらの項目のロケーターを更新できます。

1. **パッケージ エクスプローラー** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。
2. 次のいずれか 1 つを選んでください：

   - 使用するオブジェクト マップをダブルクリックします。
   - 使用するオブジェクト マップを右クリックし、**開く** をクリックします。

   オブジェクト マップ項目および各項目に関連付けられたロケーターの階層が、オブジェクト マップに表示されます。
3. **ロケータの更新** をクリックします。**Locator Spy** が表示され、Silk4J によってテスト アプリケーションが開かれます。
4. 記録するオブジェクトの上にカーソルを合わせて、**Ctrl+Alt** を押します。Silk4J の **ロケーター** テキスト フィールドにロケーター文字列が表示されます。
5. **ロケーターの詳細** テーブルで、使用するロケーターを選択します。
6. **ロケーター** テキスト フィールドに表示されているロケーターから、使用しない属性を削除します。
7. **ロケーターの検証** をクリックして、ロケーターが機能することを検証します。
8. **ロケーターをエディターに貼り付け** をクリックして、オブジェクト マップのロケーターを更新します。
9. 変更されたオブジェクト マップを保存します。

AUT からオブジェクト マップ項目を更新するときに、オブジェクト マップ ツリーのリーフ ノードの XPath 表現のみを変更できます。親ノードの XPath 表現を変更することはできません。オブジェクト マップ ツリー内のより高いレベルのノードにある XPath 表現が更新後に整合しなくなると、エラー メッセージが表示されます。

例

たとえば、次の 3 つの階層レベルを持つオブジェクト マップ ID を含むオブジェクト マップ項目があるとします：

```
WebBrowser.Dialog.Cancel
```

これらの階層レベルに対応する XPath 表現は次のようになります：

```
/BrowserApplication//Dialog//PushButton[@caption='Cancel']
```

- 最初の階層レベル： /BrowserApplication
- 2 番目の階層レベル：//Dialog
- 3 番目の階層レベル：//PushButton[@caption='Cancel']

次のロケーターを使用して、オブジェクト マップ項目を更新できます：

```
/BrowserApplication//Dialog//PushButton[@id='123']
```

- 最初の階層レベル： /BrowserApplication
- 2 番目の階層レベル：//Dialog
- 3 番目の階層レベル：//PushButton[@id='123']

2 番目のレベルの階層が一致しないため、次のロケーターを使用してオブジェクト マップ項目を更新することはできません：

```
/BrowserApplication//BrowserWindow//PushButton[@id='9999999']
```

- 最初の階層レベル： /BrowserApplication
- 2 番目の階層レベル：//BrowserWindow
- 3 番目の階層レベル：//PushButton[@id='9999999']

オブジェクト マップ項目のコピー

オブジェクト マップ内、またはオブジェクト マップ間で、オブジェクト マップ エントリをコピーおよび貼り付けできます。 たとえば、2 つの異なるテスト アプリケーションに同じ機能が存在する場合は、一方のオブジェクト マップの一部分をコピーして、他方のオブジェクト マップに貼り付けることができます。

1. **パッケージ エクスプローラー** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。
2. 次のいずれか 1 つを選んでください：
   - コピーするオブジェクト マップ項目を含むオブジェクト マップをダブルクリックします。
   - コピーするオブジェクト マップ項目を含むオブジェクト マップを右クリックし、**開く** を選択します。

   オブジェクト マップ項目および各項目に関連付けられたロケーターの階層が、オブジェクト マップに表示されます。
3. コピーするオブジェクト マップ項目に移動します。
   たとえば、コピーするオブジェクト マップ項目を検索するには、ノードの展開が必要な場合があります。
4. 次のいずれか 1 つを選んでください：
   - コピーするオブジェクト マップ項目を右クリックし、**ツリーのコピー** を選択します。
   - コピーするオブジェクト マップ項目をクリックし、Ctrl+C を押します。
5. オブジェクト マップ階層で、コピーした項目を貼り付ける位置に移動します。
   たとえば、階層の第 1 レベルに項目を組み込むには、項目リストの最初の項目の名前をクリックします。 特定の項目の 1 レベル下にコピーする項目の位置を設定するには、コピーする項目の上にある項目をクリックします。
   オブジェクト マップ間でコピーして貼り付けるには、オブジェクト マップ項目をコピーしたマップを終了し、オブジェクト マップ項目を貼り付けるオブジェクト マップを開いて編集する必要があります。
6. 次のいずれか 1 つを選んでください：

- コピーしたオブジェクト マップ項目を貼り付けるオブジェクト マップ内の位置を右クリックし、**貼り付け** を選択します。
- コピーしたオブジェクト マップ項目を貼り付けるオブジェクト マップ内の位置をクリックし、Ctrl+V を押します。

オブジェクト マップ項目が、階層内の新しい位置に表示されます。

7. **Ctrl+S** を押して、変更を保存します。

移動したオブジェクト マップ項目を既存のスクリプトで使用する場合は、階層内の新しい位置を使用するようにスクリプトを手動で変更する必要があります。

オブジェクト マップ項目の追加

スクリプトを記録すると、オブジェクト マップ項目が自動的に作成されます。場合によっては、手動でオブジェクト マップ項目を追加することもできます。

1. **パッケージ エクスプローラー** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。
2. 新しい項目を追加したい場所で、オブジェクト マップをダブルクリックします。オブジェクト マップ項目および各項目に関連付けられたロケーターの階層が、オブジェクト マップに表示されます。
3. オブジェクト マップ階層で、新しいオブジェクト マップ項目を追加したい位置の下の項目を右クリックします。

   たとえば、階層の第 1 レベルに項目を組み込むには、項目リストの最初の項目の名前を右クリックします。特定の項目の 1 レベル下に新しい項目の位置を設定するには、新しい項目を配置したい位置の下の項目をクリックします。
4. **新規挿入** をクリックします。新しい項目が、現在のノードの最初の子として階層に追加されます。
5. 有効なロケーター パスがある場合は、使用する項目名とロケーター パスを入力して Enter を押すことができます。有効なロケーター パスを判別するには、以下のステップで説明するように、**Locator Spy** ダイアログ ボックスを使用します。
   a) Silk4J ツール バーで、**Locator Spy** をクリックします。
   b) 記録したいオブジェクトの上にマウスを移動して **Ctrl+Alt** を押します。Silk4J の **ロケーター** テキスト フィールドにロケーター文字列が表示されます。
   c) **ロケーターの詳細** テーブルで、使用するロケーターを選択します。
   d) ロケーターをコピーしてオブジェクト マップに貼り付けます。
6. 必要に応じて、ニーズに合わせて項目名またはロケーター テキストを変更します。

   無効な文字またはロケーターを使用すると、項目名またはロケーター テキストが赤で表示され、ツール ヒントにエラーの説明が表示されます。

   オブジェクト マップ項目として無効な文字は、¥、/、<、>、"、:、*、?、|、=、.、@、[、] です。

   無効なロケーター パスは、空または不完全なロケーター パスです。
7. **Ctrl+S** を押して、変更を保存します。

注: オブジェクト マップ ツリーに含まれるすべてのノードのすべての子ノードは、オブジェクト マップを保存するときにアルファベット順にソートされます。

スクリプトからオブジェクト マップを開く

スクリプトを編集している際に、スクリプトのオブジェクト マップ エントリを右クリックし、 **Silk4J 資産を開く** を選択してオブジェクト マップを開くことができます。オブジェクト マップは GUI 上で開かれます。

Ctrl+Click を使用し、オブジェクト マップ エントリをクリックすると、オブジェクト マップ エントリはハイパーリンクに変わります。クリックして開きます。

例

```
@Test
public void test() {
          Window mainWindow = desktop.<Window>find("Untitled - Notepad");
       mainWindow.<TextField>find("TextField").typeKeys("hello");
}
```

上記のコード例で、 Untitled - Notepad エントリをオブジェクト マップで開く場合は、 Untitled - Notepad を右クリックします。 オブジェクト マップで Untitled - Notepad.TextField エントリをオブジェクト マップで開く場合は、TextField を右クリックします。

テスト アプリケーションでのオブジェクト マップ項目のハイライト

オブジェクト マップ項目を追加または記録したあと、**ハイライト** をクリックして、テスト アプリケーションで項目をハイライトできます。 変更する項目であることをオブジェクト マップ内で確認する場合などに項目をハイライトできます。

1. **パッケージ エクスプローラー** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。
2. 次のいずれか 1 つを選んでください：
   - 使用するオブジェクト マップをダブルクリックします。
   - 使用するオブジェクト マップを右クリックし、**開く** をクリックします。

   オブジェクト マップ項目および各項目に関連付けられたロケーターの階層が、オブジェクト マップに表示されます。
3. オブジェクト マップ階層で、テスト アプリケーションでハイライトするオブジェクト マップ項目を選択します。

   注: テスト アプリケーションの 1 つのインスタンスのみが実行中であることを確認します。 テスト アプリケーションの複数のインスタンスを実行すると、複数のオブジェクトがロケーターと一致するためエラーになります。
4. **ハイライト** をクリックします。

   テスト アプリケーションがオブジェクト マップに関連付けられていないと、**アプリケーションの選択** ダイアログ ボックスが表示されることがあります。 この場合は、テストするアプリケーションを選択し、**OK** をクリックします。

   Silk4J によってテスト アプリケーションが開かれ、オブジェクト マップ項目を示すコントロールの周囲に緑のボックスが表示されます。

オブジェクト マップのエラーの検出

無効な文字またはロケーターを使用すると、項目名またはロケーター テキストが赤で表示され、ツール ヒントにエラーの説明が表示されます。 **オブジェクト マップ** ウィンドウのツール バーを使用して、エラーに移動します。

1. **パッケージ エクスプローラー** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。

2. 次のいずれか 1 つを選んでください：
   - トラブルシュートするオブジェクト マップをダブルクリックします。
   - トラブルシュートするオブジェクト マップを右クリックし、**開く** を選択します。

   オブジェクト マップ項目および各項目に関連付けられたロケーターの階層が、オブジェクト マップに表示されます。
3. 赤色で表示された項目名またはロケーター テキストを探します。
4. 必要に応じて、ニーズに合わせて項目名またはロケーター テキストを変更します。
   無効な文字またはロケーターを使用すると、項目名またはロケーター テキストが赤で表示され、ツールヒントにエラーの説明が表示されます。

   オブジェクト マップ項目として無効な文字は、¥、/、<、>、"、:、*、?、|、=、.、@、[、] です。

   無効なロケーター パスは、空または不完全なロケーター パスです。
5. **Ctrl+S** を押して、変更を保存します。

オブジェクト マップ項目の削除

テスト アプリケーションに存在しなくなったなどの理由により、オブジェクト マップから項目を削除できます。

1. **パッケージ エクスプローラー** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。
2. 削除するオブジェクト マップ項目を含むオブジェクト マップをダブルクリックします。オブジェクト マップ項目および各項目に関連付けられたロケーターの階層が、オブジェクト マップに表示されます。
3. 削除するオブジェクト マップ項目に移動します。
   たとえば、削除するオブジェクト マップ項目を検索するには、ノードの展開が必要な場合があります。
4. 次のいずれか 1 つを選んでください：
   - 削除するオブジェクト マップ項目を右クリックし、**削除** を選択するか、そのオブジェクト マップ項目のすべての子項目も削除する場合は **ツリーの削除** を選択します。
   - 削除するオブジェクト マップ項目をクリックし、**Del** を押すか、そのオブジェクト マップ項目のすべての子項目も削除する場合は **Ctrl+Del** を押します。

   オブジェクト マップ項目を削除した後、フォーカスはオブジェクト マップの次の項目に移動します。
5. **Ctrl+S** を押して、変更を保存します。

削除したオブジェクト マップ項目または子オブジェクトを既存のスクリプトで使用する場合は、そのオブジェクト マップ項目への参照をスクリプトで、手動で変更する必要があります。

オブジェクト マップを最初に書き出す

ベスト プラクティスとして、テストを記録する前に、すべてのオブジェクト マップ項目を書き出し。確認することをお勧めします。

AUT のすべての利用可能な項目をもつオブジェクト マップを最初に書き出すためには、テスト対象アプリケーションのすべてのオブジェクトをクリックし、すべてのウィンドウとダイアログ ボックスを開くテストを作成する必要があります。その後、各オブジェクトに対するオブジェクト マップ項目を確認し、機能テストを記録する前に、必要な変更を加えることができます。オブジェクト マップ項目を確認し、修正した後に、オブジェクト マップを書き出すために作成したテストを削除できます。

ヒント: オブジェクト マップ中の項目間を移動するには、矢印キーを使用できます。

オブジェクト マップの要素のグループ化

オブジェクト マップの項目が一貫した親オブジェクトを持たない場合、XPath の現在の要素のロケーターを意味するロケーター "." を持つ新しいツリー項目を追加して、これらの要素をグループ化できます。

警告: オブジェクト マップ項目をグループ化すると、それらの項目を使用するすべてのスクリプトが影響を受けます。これらの項目を使用するすべてのスクリプトは、新しいロケーターを使用するように手動で変更する必要があります。

1. **パッケージ エクスプローラー** で、変更するオブジェクト マップがあるプロジェクトの **オブジェクト マップ** フォルダをクリックします。
2. 次のいずれか 1 つを選んでください：
   - 編集するオブジェクト マップをダブルクリックします。
   - 編集するオブジェクト マップを右クリックし、**開く** をクリックします。

   オブジェクト マップ項目および各項目に関連付けられたロケーターの階層が、オブジェクト マップに表示されます。
3. 新しく追加する構成項目のすぐ上のツリー項目を右クリックして、**新規挿入** を選択します。
4. 新しいオブジェクト マップ項目の **項目名** フィールドをダブルクリックします。
5. 使用する項目名を入力し、Enter を押します。

   無効な文字を使用すると、項目名が赤で表示されます。

   新しい名前が **項目名** リストに表示されます。
6. 新しいオブジェクト マップ項目の **ロケーター パス** フィールドをクリックして、フィールドに「.」を入力します。
7. Enter を押します。
8. 新しい項目の下に新たに配置し直したいすべてのオブジェクト マップ項目に対して、次の操作を行います。

   a) 配置し直したい項目を右クリックして、**ツリーの切り取り** を選択します。

   b) 新しい構成項目を右クリックして、**貼り付け** を選択します。
9. **Ctrl+S** を押して、変更を保存します。

オブジェクト マップ：よくある質問

このセクションでは、Silk4J でオブジェクト マップを使用する時に直面する可能性のある質問を一覧します。

複数のオブジェクト マップを単一のオブジェクト マップのマージする方法

既存のオブジェクト マップをマージせずに、同じオブジェクト マップに記録することを Micro Focus は推奨しますが、テキスト エディタを使用して複数のオブジェクト マップを単一のマップにマージすることができます。

テスト スクリプトを削除したときにオブジェクト マップで起こること

オブジェクト マップ エントリを含んだテスト スクリプトを削除しても、関連するオブジェクト マップは変更されません。すべてのオブジェクト マップ エントリはそのまま残ります。

テスト対象アプリケーションのオブジェクト マップを手動で作成する方法

Micro Focus では、テストの記録時にオブジェクト マップを作成することを推奨します。しかし、空のオブジェクト マップを作成し、このマップにオブジェクト マップの項目を手動で追加することもできます。

新しいオブジェクト マップを作成するには、**Silk4J** > **新規オブジェクト マップ** をメニューから選択します。

イメージ解決のサポート

イメージ解決は、次の場合に使用できます。

- オブジェクト解決で識別できない、高度にカスタマイズされたコントロールを含むテスト アプリケーションを簡便に操作する場合。座標ベースのクリックの代わりにイメージ クリックを使用し、指定されたイメージをクリックできます。
- テスト対象アプリケーションのグラフィカル オブジェクト (グラフなど) をテストする場合。
- テスト対象アプリケーションの視覚的な UI のチェックを実行する場合。

認識されないコントロールをクリックするには、imageClick メソッドとイメージ資産を使用します。テスト対象アプリケーションに、認識されないコントロールがあるかどうかを確認するには、verifyAsset メソッドとイメージ検証を使用します。

イメージ解決メソッドは、Silk4J でサポートされるすべてのテクノロジ ドメインでサポートされます。

注: イメージ解決メソッドは、画面に表示されていないコントロールに対しては機能しません。たとえば、スクロールして画面外にあるイメージに対しては、イメージ解決を使用できません。

イメージ クリックの記録

イメージ クリックの記録を行うと、大量のイメージが生成されてわかりにくくなるため、デフォルトではイメージ クリックの記録が無効となり、座標ベースのクリックの記録が優先されます。

イメージ クリックの記録を有効にするには、**Silk4J** > **オプションの編集** をクリックして、**記録** タブを選択し、 **'ImageClick' を記録する** セクションのチェック ボックスをチェックします。

注: モバイル ブラウザを記録する場合、イメージ クリックの記録を有効にする必要はありません。

イメージ クリックの記録を有効にすると、Silk4J は、オブジェクト解決またはテキスト解決ができない場合に、ImageClick メソッドを記録します。イメージ クリックが記録されない場合でも、任意のコントロールに対してイメージ クリックをスクリプトに挿入できます。

ImageClick 操作を記録しない場合は、イメージ クリックの記録をオフに切り替えて通常のクリックまたはテキスト クリックを記録できます。

注: 記録したイメージは再利用されません。 Silk4J は、記録するイメージ クリックごとに新しいイメージ資産を作成します。

注: イメージ クリックの記録は、Java AWT/Swing コントロールを使用するアプリケーションまたはアプレットではサポートされません。

イメージ解決メソッド

Silk4J では、イメージ解決用に次のメソッドが用意されています。

メソッド	説明
imageClick	資産で指定されたイメージの中央をクリックします。イメージが見つかるか、オブジェクト解決タイムアウト (同期オプションで定義可能) が経過するまで待機します。
imageExists	資産で指定されたイメージが存在するかどうかを返します。

メソッド	説明
imageRectangle	資産で指定されたイメージのオブジェクト相対矩形を返します。
imageClickFile	ファイルで指定されたイメージをクリックします。
imageExistsFile	ファイルで指定されたイメージが存在するかどうかを返します。
imageRectangleFile	ファイルで指定されたイメージのオブジェクト相対矩形を返します。
verifyAsset	検証資産を実行します。検証に合格しなかった場合、VerificationFailedException がスローされます。
tryVerifyAsset	検証資産を実行し、検証に合格したかどうかを返します。

注: イメージ解決メソッドは、画面に表示されていないコントロールに対しては機能しません。たとえば、スクロールして画面外にあるイメージに対しては、イメージ解決を使用できません。

イメージ資産

イメージ資産は、次の場合に使用できます。

- オブジェクト解決で識別できない、高度にカスタマイズされたコントロールを含むテスト アプリケーションを簡便に操作する場合。 座標ベースのクリックの代わりにイメージ クリックを使用し、指定されたイメージをクリックできます。
- テスト対象アプリケーションのグラフィカル オブジェクト (グラフなど) をテストする場合。

イメージ資産は、イメージと、Silk4J で資産を操作するために必要な追加情報で構成されます。

Silk4J では、イメージ資産用に次のメソッドが用意されています。

メソッド	説明
imageClick	指定されたイメージ資産の中央をクリックします。 イメージが見つかるか、オブジェクト解決タイムアウト (同期オプションで定義可能) が経過するまで待機します。
imageExists	指定されたイメージ資産があるかどうかを返します。
imageRectangle	指定されたイメージ資産のオブジェクト相対矩形を返します。

イメージ資産は、プロジェクトの Image Assets フォルダに置く必要があります。 .imageasset ファイルは、埋め込みリソースにする必要があります。

イメージ資産の作成

イメージ資産は、次のいずれかの方法で作成できます。

- 新しいイメージ資産を既存のスクリプトに挿入。
- 記録時。
- メニューから。

新しいイメージ資産を で作成するには、以下のステップを実行します。

1. メニューで、**Silk4J** > **新規イメージ資産** をクリックします。
2. 新しいイメージ資産を追加するプロジェクトを選択し、資産のわかりやすい名前を **名前** フィールドに入力します。
3. **終了** をクリックします。イメージ資産の UI が開きます。
4. 資産にイメージを追加する方法を選択します。

- 既存のイメージを使用する場合は、**参照** をクリックし、イメージ ファイルを選択します。
- テスト対象アプリケーションの UI から新しいイメージをキャプチャする場合は、**キャプチャ** をクリックします。Web アプリケーションをテストする場合、**ブラウザーの選択** ウィンドウからイメージをキャプチャするブラウザーを選択できます。

5. 新しいイメージをキャプチャする場合は、キャプチャする画面領域を選択し、**選択範囲をキャプチャします** をクリックします。
6. 省略可能：**検証** をクリックして、Silk4J が AUT の UI でイメージ資産を見つけることができるかどうかを確認します。

   Web アプリケーションをテストする場合、**ブラウザーの選択** ウィンドウからイメージをキャプチャするブラウザーを選択できます。
7. 省略可能：**クリック位置** チェック ボックスをオンにすると、イメージ資産上でクリックを実行する位置を選択できます。

   デフォルトの位置は、イメージの中央です。**x** フィールドと **y** フィールドに位置を入力するか、イメージ上の位置を選択します。
8. **精度レベル** を指定します。

   精度レベルは、検証されるイメージがテスト対象アプリケーションのイメージと異なっていてもよい度合いを定義し、これを超えて異なっている場合、Silk4J はイメージが異なっていると判断します。これは、画面解像度が異なる複数のシステムまたはブラウザをテストする場合に役立ちます。誤検出を防ぐため、できるだけ精度レベルを高くすることを推奨します。デフォルトの精度レベル値は、オプションで変更できます。

   注: 精度レベル を 5 未満に設定した場合、イメージの実際の色が比較で考慮されなくなります。イメージのグレースケール表現だけが比較されます。
9. イメージ資産を保存します。

新しいイメージ資産が、**パッケージ エクスプローラー** で現在のプロジェクトの下に表示され、これを使用してイメージ クリックを実行できます。

同じイメージ資産に複数のイメージを追加できます。

注: モバイル ブラウザーに対する記録中にイメージ クリックを追加するには、**記録** ウィンドウで右クリックして、操作のリストから **ImageClick** を選択します。

同じイメージ資産に複数のイメージを追加する

テスト時に、異なるテスト構成を使用して、複数の環境の機能をテストする必要が生じることがよくあります。環境が異なると、イメージ資産にキャプチャしたイメージと実際のイメージが若干異なることがあり、イメージが存在するにもかかわらずイメージ クリックが失敗することがあります。このような場合、同じイメージ資産に複数のイメージを追加できます。

イメージ資産に別のイメージを追加するには、以下を実行します。

1. イメージ資産に追加したいイメージをダブルクリックします。イメージ資産の UI が開きます。
2. UI の下部に表示されるプラス記号をクリックして、新しいイメージをイメージ資産に追加します。
3. イメージ資産を保存します。

新しいイメージが資産に追加されます。イメージ クリックが呼び出されるたびに、一致するものに到達するまで、Silk4J は資産のイメージとテスト対象アプリケーションの UI のイメージを比較します。デフォルトで、Silk4J は資産に追加された順にイメージを比較します。

注: Silk4J で比較するイメージの順番を変更するには、イメージ資産の UI の下部でイメージをクリックし、目的の場所にドラッグします。左から右の順に比較されます。最初に比較されるのは、最も左にあるイメージです。

スクリプトから資産を開く

スクリプトを編集しているときに、資産を右クリックして **Silk4J 資産を開く** を選択し、資産を開くことができます。 これにより、GUI で資産が開きます。

資産がシステム上のファイルへの参照である場合 (ImageClickFile によって参照される場合など)、ファイルはシステムのデフォルト エディターで開かれます。

Ctrl+クリックを使用して資産をクリックすると、その資産はハイパーリンクに変わります。 それをクリックすると開きます。

イメージ検証

イメージ検証を使用して、テスト対象アプリケーション (AUT) の UI にイメージがあるかどうかをチェックできます。

イメージ検証は、イメージと、Silk4J で資産を操作するために必要な追加情報で構成されます。

イメージ検証を実行するには、verifyAsset メソッドを使用します。

イメージ検証資産は、プロジェクトの Verifications フォルダに置く必要があります。 .verification ファイルは、埋め込みリソースにする必要があります。

Silk4J が AUT のイメージを見つけることができなかった場合、イメージ検証は失敗します。 この場合、スクリプトの実行は中断され、VerificationFailedException がスローされます。 この動作を防止するには、tryVerifyAsset メソッドを使用します。

AUT 内でイメージ検証のロケーターが見つからなかった場合、Silk4J は ObjectNotFoundException をスローします。

TrueLog Explorer で成功したイメージ検証を開くには、検証ステップの **情報** タブで **検証を開く** をクリックします。 TrueLog Explorer で失敗したイメージ検証を開くには、検証ステップの **情報** タブで **相違点の表示** をクリックします。 失敗したイメージ検証が、精度レベルを低くすれば成功すると判断された場合は、成功する精度レベルが提示されます。

イメージ検証の作成

イメージ検証は、次のいずれかの方法で作成できます。

- メニュー を使用。
- 記録時。

新しいイメージ検証を メニュー で作成するには、以下のステップを実行します。

1. **Silk4J** > **新規イメージ検証** をクリックします。
2. 新しいイメージ検証を追加するプロジェクトを選択し、検証のわかりやすい名前を **名前** フィールドに入力します。
3. **終了** をクリックします。イメージ検証の UI が開きます。
4. **識別** をクリックして、テスト対象アプリケーションの、検証するイメージを識別します。
5. 省略可能：最初にキャプチャしたイメージから変更されたために、テスト対象アプリケーションから同じイメージを再キャプチャする必要がある場合は、**再キャプチャ** をクリックします。

   Web アプリケーションをテストする場合、**ブラウザーの選択** ウィンドウからイメージをキャプチャするブラウザーを選択できます。
6. 省略可能：**検証** をクリックすると、イメージ検証が機能するかどうかをテストできます。Silk4J は、AUT の UI でイメージを、上から下へ、左から右へと検索し、最初に一致したイメージをハイライトします。

7. 省略可能 : Silk4J がイメージ検証とテスト対象アプリケーション (AUT) の UI を比較するときに考慮しない除外領域をイメージ検証に追加できます。
8. 省略可能 : オプション **クライアント領域のみ** を設定して、Silk4J がイメージ検証と AUT の UI を比較するときに、実際に AUT の一部であるイメージの部分だけを考慮するように定義できます。
9. **精度レベル** を指定します。

   精度レベルは、検証されるイメージがテスト対象アプリケーションのイメージと異なっていてもよい度合いを定義し、これを超えて異なっている場合、Silk4J はイメージが異なっていると判断します。これは、画面解像度が異なる複数のシステムまたはブラウザをテストする場合に役立ちます。誤検出を防ぐため、できるだけ精度レベルを高くすることを推奨します。デフォルトの精度レベル値は、オプションで変更できます。

   注: 精度レベル を 5 未満に設定した場合、イメージの実際の色が比較で考慮されなくなります。イメージのグレースケール表現だけが比較されます。

10. イメージ検証を保存します。

新しいイメージ検証が**パッケージ エクスプローラー** に表示され、これを使用して、テスト対象アプリケーションの UI にイメージが存在するかどうかをチェックできます。

記録中にイメージ検証を追加する

イメージ検証をスクリプトに追加して、テスト対象アプリケーションの UI に認識されないコントロールがあるかどうかをチェックできます。 スクリプトの記録中にイメージ検証を追加するには、以下のステップを実行します。

1. 記録を開始します。
2. 検証するイメージの上にマウス カーソルを移動して、**Ctrl+Alt** を押しながらクリックします。 Silk4J から、プロパティーまたはイメージを検証するかどうかを尋ねられます。
3. **イメージ検証の作成または挿入** を選択します。
4. 次のいずれか 1 つのステップを行います :
   - イメージ検証の UI で新しいイメージ検証を作成するには、リスト ボックスから **新規** を選択します。
   - 既存のイメージ検証資産を挿入するには、リスト ボックスからイメージ検証資産を選択します。
5. **OK** をクリックします。
   - 新しいイメージ検証の作成を選択した場合は、イメージ検証の UI が表示されます。
   - 既存のイメージ検証の使用を選択した場合は、イメージ検証がスクリプトに追加されます。 この場合、このトピックの残りのステップはスキップできます。
6. 新しいイメージ検証を作成するには、イメージ検証の UI で **検証** をクリックします。
7. AUT のイメージの上にマウス カーソルを移動して、**Ctrl+Alt** を押しながらクリックします。 イメージ検証の UI に、新しいイメージ検証が表示されます。
8. **OK** をクリックします。 新しいイメージ検証が現在のプロジェクトに追加されます。
9. 記録を続けます。

複数のプロジェクトでの資産の使用

Silk4J では、イメージ資産、イメージ検証、およびオブジェクト マップが資産と呼ばれます。資産が配置されているプロジェクトのスコープ外でそれらの資産を使用する場合、資産を使用するプロジェクトから、資産を配置するプロジェクトに、プロジェクトの直接的な依存関係を追加する必要があります。Eclipse からテストを再生する場合、すべての依存プロジェクトがテストを実行するクラスパスに追加されます。このため、Silk4J は依存プロジェクトの資産も見つけることができます。

再生中に資産が使用されると、Silk4J は、最初に現在のプロジェクト内でその資産を検索します。現在のプロジェクトは、現在実行されるテスト コードを含んだ JAR ファイルです。Silk4J で現在のプロジェクト内に資産が検出されなかった場合、Silk4J は現在のプロジェクトがプロジェクト クラスパス内のすべての

他のプロジェクトを持つプロジェクトを追加検索します。それでも資産が見つからない場合、Silk4J はエラーをスローします。

複数のプロジェクトに同じ名前の資産が存在する場合に、現在のプロジェクトに含まれている資産を使用しないときは、資産を使用するメソッドで使用する特定の資産を定義できます。使用する資産を定義するには、メソッドを呼び出すときに、資産の名前空間を接頭辞として資産名に追加します。資産の名前空間は、デフォルトでプロジェクト名に設定されます。

注: Silk4J での作業を開始すると、資産の名前空間オプションが、前のバージョンの Silk4J で作成されたワークスペースにある各 Silk4J の silk4j.settings ファイルに追加されます。

例：プロジェクトの依存関係の追加

プロジェクト *ProjectA* にコード

```
window.imageClick("imageAsset");
```

を呼び出すテストが含まれており、イメージ資産 *imageAsset* がプロジェクト *ProjectB* に置かれている場合、プロジェクトの直接的な依存関係を *ProjectA* から *ProjectB* に追加する必要があります。

Eclipse にプロジェクト依存関係を追加するには、プロジェクトを右クリックし、**プロパティ**を選択します。**Java のビルド・パス** を選択い、**プロジェクト** タブをクリックして、ここにプロジェクトを追加します。

注: プロジェクト参照 を **Java のビルド・パス** の代わりに設定しても機能しません。

例：特定の資産の呼び出し

ProjectA と *ProjectB* の両方に *anotherImageAsset* という名前のイメージ資産が含まれている場合に、*ProjectB* からイメージ資産を明示的にクリックする場合、次のコードを使用します：

```
window.imageClick("ProjectB:anotherImageAsset")
```

テストの拡張

このセクションでは、テストの拡張方法について説明します。

既存のテストへの追加操作の記録

テストを作成したあと、テストを開き、テストの任意の場所から追加操作を記録できます。これにより、既存のテストを追加操作で更新できます。

1. 既存のテスト スクリプトを開きます。
2. 追加操作を記録するテスト スクリプトの場所を選択します。

   注: 記録した操作は、選択した場所の後に挿入されます。テスト対象アプリケーション (AUT) は基本状態に戻りません。代わりに、テスト スクリプトの直前の操作が記録された範囲で AUT を開いておきます。
3. **操作の記録** をクリックします。

   Silk4J が最小化され、**記録中** ダイアログ ボックスが開きます。
4. AUT に対して実行したい追加操作を記録します。

   記録中に利用可能な操作についての詳細は、「記録中に利用可能な操作」を参照してください。
5. 記録を停止するには、**記録中** ウィンドウの **停止** をクリックします。

Windows DLL の呼び出し

このセクションでは、DLL を呼び出す方法について説明します。 DLL は Open Agent のプロセス内から、または AUT (テスト対象アプリケーション) から呼び出すことができます。 これにより、テスト スクリプト内の既存のネイティブ DLL を再利用できます。

Open Agent 内の DLL 呼び出しは通常、AUT 内の UI コントロールと対話しないグローバル関数を呼び出す場合に使用されます。

AUT 内の DLL 呼び出しは通常、アプリケーションの UI コントロールと対話する関数を呼び出す場合に使用されます。 これにより、Silk4J は再生中に DLL 呼び出しを自動的に同期できます。

注: 32 ビット アプリケーションでは 32 ビット DLL を、64 ビット アプリケーションでは 64 ビット DLL を呼び出すことができます。 Open Agent は 32 ビットと 64 ビットの両方の DLL を実行できます。

注: DLL を呼び出すには、C インターフェイスを使用する必要があります。 ファイル拡張子が .dll である .NET アセンブリの呼び出しは、サポートされていません。

スクリプトからの Windows DLL の呼び出し

DLL 呼び出しに関連するすべてのクラスおよび注釈は、パッケージ com.borland.silktest.jtf.dll に含まれています。

DLL の宣言を開始するには、DLL 属性を持つインターフェイスを使用します。宣言の構文は次のとおりです。

```
@Dll("dllname.dll")
public interface DllInterfaceName {
   FunctionDeclaration
```

```
  [FunctionDeclaration]...
}
```

dllname	Java スクリプトから呼び出す関数が含まれた DLL ファイルの完全パスの名前。DLL パスの環境変数は自動的に解決されます。パスでは二重のバックスラッシュ (¥¥) を使用する必要はありません。単一のバックスラッシュ (¥) で十分です。
DllInterfaceName	スクリプト内で DLL と対話するために使用される識別子。
FunctionDeclaration	呼び出そうとしている DLL 関数の関数宣言。

DLL 関数の宣言構文

DLL 関数の宣言は、一般に以下の形式を取ります。

```
return-type  function-name(  [arg-list]  )
```

戻り値のない関数の場合、宣言の形式は以下のとおりです、

```
void  function-name(  [arg-list]  )
```

return-type	戻り値のデータ型。
function-name	関数の名前。
arg-list	関数に渡される引数のリスト。 リストは以下のように指定します。 `data-type  identifier` **data-type** 引数のデータ型。 • 関数によって変更可能な引数、または関数から出力可能な引数を指定するには、InOutArgument および OutArgument クラスを使用します。 • DLL 関数を引数の値に設定する場合は、OutArgument クラスを使用します。 • 値を関数に渡し、関数によって値を変更して、新しい値を出力する場合は、InOutArgument クラスを使用します。 **identifier** 引数の名前。

DLL 呼び出しの例

この例では、user32.dll の SendMessage DLL 関数を呼び出して、フィールドに「*hello world!*」というテキストを書き出します。

DLL の宣言：

```
@Dll("user32.dll")
public  interface  IUserDll32Functions  {
   int  SendMessageW(TestObject  obj,  int  message,  int  wParam,  Object  lParam);
}
```

以下のコードは、AUT で宣言された DLL 関数を呼び出す方法を示します。

```
IUserDll32Functions  user32Function  =
DllCall.createInProcessDllCall(IUserDll32Functions.class,  desktop);
```

```
TextField textField = desktop.find("//TextField");
user32Function.SendMessageW(textField, WindowsMessages.WM_SETTEXT, 0, "my text");
```

注: DLL 関数の最初のパラメーターに C データ型の HWND が指定されている場合は、AUT 内で DLL 関数の呼び出しのみを実行できます。

次のコードは、Open Agent のプロセスで宣言された DLL 関数を呼び出す方法を示します。

```
IUserDll32Functions user32Function = DllCall.createAgentDllCall(IUserDll32Functions.class,
desktop);
TextField textField = desktop.find("//TextField");
user32Function.SendMessageW(textField, WindowsMessages.WM_SETTEXT, 0, "my text");
```

注: コード例では、DLL 関数で使用するのに便利な Windows メッセージングに関連する定数を定義した WindowsMessages クラスを使用しています。

DLL 関数への引数の受け渡し

DLL 関数は C で記述されているため、これらの関数に渡す引数には適切な C データ型を指定する必要があります。 次のデータ型がサポートされます。

int	次のデータ型を持つ引数または戻り値には、このデータ型を使用します。 • int • INT • long • LONG • DWORD • BOOL • WPARAM • HWND Java の int 型は、4 バイト値を持つすべての DLL 引数に対して有効です。
long	C データ型 long および int64 を持つ引数または戻り値には、このデータ型を使用します。 Java の long 型は、8 バイト値を持つすべての DLL 引数に対して有効です。
short	C データ型 short および WORD を持つ引数または戻り値には、このデータ型を使用します。 Java の short 型は、2 バイト値を持つすべての DLL 引数に対して有効です。
boolean	C データ型 bool を持つ引数または戻り値には、このデータ型を使用します。
String	C で String となる引数または戻り値には、このデータ型を使用します。
double	C データ型 double を持つ引数または戻り値には、このデータ型を使用します。
com.borland.silktest.jtf.Rect	C データ型 RECT を持つ引数には、このデータ型を使用します。 Rect は戻り値として使用できません。
com.borland.silktest.jtf.Point	C データ型 POINT を持つ引数には、このデータ型を使用します。 POINT は戻り値として使用できません。
com.borland.silktest.jtf.TestObject	C データ型 HWND を持つ引数には、このデータ型を使用します。 TestObject は戻り値として使用できませんが、戻り値型

として Integer を持つ HWND を戻す DLL 関数を宣言できます。

注: 渡された TestObject は com.borland.silktest.jtf.INativeWindow インターフェイスを実装して、DLL 関数に渡される TestObject のウィンドウ ハンドルを Silk4J が 判別できるようにする必要があります。 そうしないと、この DLL 関数を呼び出すときに、例外がスローされます。

List

ユーザー定義の C 構造体の配列には、このデータ型を使用します。 List は戻り値として使用できません。

注: List を入出力パラメーターとして使用する場合は、渡されるリストに、戻される内容を保持できるだけのサイズを確保する必要があります。

注: C 構造体は List で表すことができます。この場合、すべてのリスト要素は構造体のメンバに対応しています。 最初の構造体メンバは、リスト内の最初の要素で表されます。2 番目の構造体メンバは、リスト内の 2 番目の要素で表されます (以下同様)。

注: DLL 関数に渡す引数の前には、いずれかの Java データ型を配置する必要があります。

DLL 関数で変更できる引数の受け渡し

値が DLL 関数によって変更される引数は、InOutArgument (値を変更する場合)、または OutArgument を使用して渡す必要があります。

例

この例では、現在のカーソル位置を取得するために、user32.dll の GetCursorPos 関数を使用しています。

DLL の宣言：

```
@Dll( "user32.dll" )
public interface IUserDll32Functions {
   int GetCursorPos( OutArgument<Point> point);
}
```

使用法：

```
IUserDll32Functions user32Function =
DllCall.createAgentDllCall(IUserDll32Functions.class, desktop);

OutArgument<Point> point = new OutArgument<Point>(Point.class);
user32Function.GetCursorPos(point);

System.out.println("cursor position = " + point.getValue());
```

DLL 関数への文字列引数の受け渡し

DLL 関数に渡している文字列、または DLL 関数から戻される文字列は、デフォルトでは Unicode Strings として処理されます。 DLL 関数に ANSI String 引数が必要な場合は、DllFunctionOptions 属性の CharacterSet プロパティーを使用します。

例

```
@Dll( "user32.dll" )
public interface IUserDll32Functions {
   @FunctionOptions(characterSet=DllCharacterSet.Ansi)
   int SendMessageA(TestObject obj, int message, int wParam, Object
lParam);
}
```

DLL 呼び出しから String を OutArgument として戻した場合、String のサイズが 256 文字以下であれば、デフォルトの動作に従います。 戻される String が 256 文字を超えている場合は、作成された String を保持できるだけの長さを持つ、String を使用して InOurArgument を渡します。

例

1024 個の空白文字を含む String を作成するには、以下のコードを使用します。

```
char[] charArray = new char[1024];
Arrays.fill(charArray,' ');
String longEmptyString = new String(charArray);
```

この InOutArgument を引数として DLL 関数に渡します。すると、この DLL 関数は最大 1024 文字の String を戻します。

関数の戻り値として DLL から String が戻される場合、DLL は DLL 関数 FreeDllMemory を実装し、DLL 関数から戻される C String ポインターを受け入れて、以前に割り当てられたメモリーを解放する必要があります。 このような関数が存在しない場合、メモリーはリークされます。

DLL 名のエイリアス設定

DLL 関数に、Java の予約語と同じ名前が付いている場合、または DLL 関数に名前でなく序数が付いている場合は、宣言内でこの関数の名前を変更し、エイリアス ステートメントを使用して、宣言した名前と実際の名前をマッピングする必要があります。

例

たとえば、goto ステートメントは Java コンパイラーで予約されています。 したがって、関数 goto を呼び出すには、次のようにその関数を別の名前で宣言し、エイリアス ステートメントを追加する必要があります。

```
@Dll("mydll.dll")
public interface IMyDllFunctions {
   @FunctionOptions(alias="break")
   void MyBreak();
}
```

DLL 関数呼び出しの表記規則

DLL 関数を呼び出す場合は、次に示す呼び出し規則がサポートされています。

- __stdcall
- __cdecl

DLL 関数を呼び出す場合は、*__stdcall* 呼び出し規則がデフォルトで使用されます。 この呼び出し規則は、すべての Windows API DLL 関数で使用されます。

DLL 関数の呼び出し規則を変更するには、DllFunctionOptions 注釈の CallingConvention プロパティーを使用します。

例

次のコード例では、*__decl* 呼び出し規則を使用して DLL 関数を宣言します。

```
@Dll("msvcrt.dll")
public interface IMsVisualCRuntime {
   @FunctionOptions(callingConvention=CallingConvention.Cdecl)
   double cos(double inputInRadians);
}
```

カスタム コントロール

Silk4J では、カスタム コントロールを扱うときに、以下の機能がサポートされます。

- 動的呼び出しを使用すると、テスト対象アプリケーション (AUT) 内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティの取得、またはプロパティの設定を Silk4J で直接実行できます。
- クラス マッピング 機能によって、カスタム コントロール クラスの名前を標準 Silk Test クラスの名前にマップできます。このようにすると、標準 Silk Test クラスでサポートされる機能をテストで使用できます。

  Silk4J は、次のテクノロジ ドメインに対する UI のカスタム コントロールの管理をサポートします。

  - Win32
  - Windows Presentation Foundation (WPF)
  - Windows Forms
  - Java AWT/Swing
  - Java SWT
- AUT にコードを追加して、カスタム コントロールをテストできます。
- **カスタム コントロールの管理** ダイアログ ボックスを使用して、ロケーターで使用できるカスタム コントロールの名前を指定したり、カスタム コントロールを操作する再利用可能なコードを作成することができます。

注: カスタム コントロールでは、click、textClick、typeKeys などのメソッドだけが、Silk4J で記録できます。Apache Flex アプリケーションをテストする場合を除き、カスタム コントロールのカスタム メソッドは記録できません。

動的呼び出し

動的呼び出しを使用すると、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、メソッドの呼び出し、プロパティーの取得、またはプロパティーの設定を直接実行できます。また、このコントロールの Silk4J API で使用できないメソッドおよびプロパティーも呼び出すことができます。動的呼び出しは、作業しているカスタム コントロールを操作するために必要な機能が、Silk4J API を通して公開されていない場合に特に便利です。

オブジェクトの動的メソッドは invoke メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

オブジェクトの複数の動的メソッドは invokeMethods メソッドを使用して呼び出します。コントロールでサポートされている動的メソッドのリストを取得するには、getDynamicMethodList メソッドを使用します。

動的プロパティの取得には getProperty メソッドを、動的プロパティの設定には setProperty メソッドを使用します。コントロールでサポートされている動的プロパティのリストを取得するには、getPropertyList メソッドを使用します。

たとえば、テスト対象アプリケーション内のコントロールの実際のインスタンスに関して、タイトルを String 型の入力パラメータとして設定する必要がある SetTitle というメソッドを呼び出すには、次のように入力します：

```
control.invoke("SetTitle","my  new  title");
```

注: 通常、ほとんどのプロパティは読み取り専用で、設定できません。

注: ほとんどのテクノロジー ドメインでは、メソッドを呼び出してプロパティーを取得する場合、Reflection を使用します。

注: DOM 要素のメソッドを動的に呼び出すことはできません。

動的呼び出しに関するよくある質問

このセクションでは、カスタム コントロールをテストするために動的にメソッドを呼び出すときの質問を示します。

invoke メソッドを使用して呼び出せるメソッド

特定のテスト オブジェクトに対して、invoke メソッドを使用して呼び出せるすべてのメソッドのリストを取得するには、getDynamicMethodList を使用します。リストを表示するには、コンソールに出力したり、デバッガーで表示することなどができます。

呼び出しで複雑なオブジェクトが返されることが期待されるときに単純な文字列が返される理由

invoke メソッドは単純なデータ型のみを返すことができます。 複雑な型は文字列として返されます。 Silk4J は ToString メソッドを使用して、戻り値の文字列表現を取得します。 個々のメソッドを呼び出し、最初のメソッドの呼び出しで返される複雑なオブジェクトのプロパティーを読み取るには、invoke ではなく、invokeMethods を使用します。

複数の invokeMethods 呼び出しを使用するときにスクリプトを単純化する方法

スクリプトで大量の invokeMethods を使用すると、すべてのメソッド名を文字列として渡し、すべてのパラメータをリストとして渡す必要があるため、複雑になります。 このような複雑なスクリプトを単純化するには、invokeMethods を通じてコントロールを操作するのではなく、AUT の実際のコントロールを操作する静的メソッドを作成します。 詳細については、「テスト対象アプリケーションにコードを追加してカスタム コントロールをテストする」を参照してください。

テスト対象アプリケーションにコードを追加してカスタム コントロールをテストする

Windows Forms アプリケーションまたは WPF アプリケーションをテストし、複雑なカスタム コントロールまたは invoke および invokeMethods メソッドを使用するだけではテストできないカスタム コントロールをテストする場合は、テスト対象アプリケーション (AUT) の実際のコントロールを操作する静的メソッドを作成し、このコードを AUT に追加できます。

AUT にコードを追加することのメリットは、AUT のコードで、動的呼び出しメソッドによるメソッド呼び出しのリフレクション形式ではなく、通常のメソッド呼び出しを使用してコントロールを操作できるとい

う点です。そのため、コードを作成する時に、コード補完と IntelliSense を使用できます。その後、AUT のコードを単純な呼び出しで呼び出し、該当するコントロールをパラメータとして渡すことができます。

AUT にコードを追加するには、次の方法があります。

- AUT でコードをコンパイルします。実装は簡単ですが、意図しない AUT の変更を行うことになります。
- テスト スクリプトの LoadAssembly メソッドを使用して、実行時にコードを AUT に挿入します。AUT でコードをコンパイルする場合よりも作業は多くなりますが、挿入されたコードはテスト コードの近くに配置されます。LoadAssembly は、WPFWindow クラスおよび FormsWindow クラスで使用できます。

例：UltraGrid Infragistics コントロールのテスト

この例では、UltraGrid コントロールの内容を取得する方法を示します。UltraGrid コントロールは、Infragistics が提供する NETAdvantage for Windows Forms ライブラリに含まれています。ライブラリの試用版を *http://www.infragistics.com/products/windows-forms/downloads* からダウンロードできます。

UltraGridUtil クラスを作成するには、以下の操作を実行します。

1. C# または VB .NET で、新しいクラス ライブラリを Microsoft Visual Studio を開いて作成します。新しいプロジェクト AUTExtensions を呼び出します。

   注: クラス ライブラリは、AUT と同じバージョンの .NET バージョンを使用する必要があります。

2. 必要な依存関係への参照をプロジェクトに追加します。たとえば、Infragistics バージョン 12.2 の場合、次のアセンブリへの参照が必要です。

   - Infragistics4.Shared.v12.2
   - Infragistics4.Win.UltraWinGrid.v12.2
   - Infragistics4.Win.v12.2

   AUT で使用している Infragistics のバージョンが不明な場合は、Microsoft の **Process Explorer** ツールを使用して、AUT にロードされているアセンブリを確認できます。

   a. AUTExtensions プロジェクトで、次の内容を持つ新しいクラス UltraGridUtil を作成します：

```
' VB code
Public Class UltraGridUtil

    Public Shared Function GetContents(ultraGrid As
Infragistics.Win.UltraWinGrid.UltraGrid) As List(Of List(Of String))
        Dim contents = New List(Of List(Of String))
        For Each row In ultraGrid.Rows
            Dim rowContents = New List(Of String)
            For Each cell In row.Cells
                rowContents.Add(cell.Text)
            Next
            contents.Add(rowContents)
        Next
        Return contents
    End Function

End Class
```

```
// C# code
using System.Collections.Generic;

namespace AUTExtensions {
```

```
  public class UltraGridUtil {

     public static List<List<string>>
GetContents(Infragistics.Win.UltraWinGrid.UltraGrid grid) {
        var result = new List<List<string>>();
        foreach (var row in grid.Rows) {
           var rowContent = new List<string>();
           foreach (var cell in row.Cells) {
              rowContent.Add(cell.Text);
           }
           result.Add(rowContent);
        }
        return result;
     }

  }

}
```

注: Shared 修飾子によって、GetContents メソッドが静的メソッドになります。

3. AUTExtensions プロジェクトを構築します。
4. 再生中に、AUT にアセンブリをロードします。
   - 既存のテスト スクリプトを開くか、新しいテスト スクリプトを作成します。
   - ここで構築したアセンブリをファイル システムからロードするコードをテスト スクリプトに追加します。例：

     ```
     mainWindow.loadAssembly("C:/buildoutput/AUTExtensions.dll");
     ```
5. 挿入したコードの静的メソッドを呼び出して、UltraGrid の内容を取得します：

   ```
   // Java code
   Control ultraGrid = mainWindow.find("//Control[@automationId='my
   grid']");
   List<List<String>> contents = (List<List<String>>)
   mainWindow.invoke("AUTExtensions.UltraGridUtil.GetContents", ultraGrid);
   ```

AUT へのコードの追加に関するよくある質問

このセクションでは、カスタム コントロールをテストするために AUT にコードを追加するときの質問を示します。

LoadAssembly メソッドを使用して AUT に挿入したコードが AUT で更新されない理由

AUT 内のコードが、LoadAssembly メソッドを使用して AUT に挿入したコードによって置き換えられない場合、アセンブリがすでに AUT にロードされている可能性があります。 アセンブリをアンロードすることはできないため、AUT を閉じてから、再開する必要があります。

メソッドを呼び出すと入力引数の型が一致しない理由

何らかのメソッドを呼び出したときに、入力引数の型が一致しないことを示すエラーが表示される場合は、呼び出すメソッドは見つかりましたが、引数が正しくありません。 スクリプトで正しいデータ型を使用していることを確認します。

スクリプトで LoadAssembly メソッドを使用してアセンブリを AUT にロードする場合にこのエラーが発生するもう 1 つの理由として、AUT が使用するバージョンとは異なるサード パーティ ライブラリのバージョンに対してアセンブリが作成されている可能性があります。 この問題を修正するには、プロジェクトで参照されているアセンブリを変更します。 AUT で使用されているサード パーティ ライブラリのバージョンが不明な場合は、Microsoft の **Process Explorer** ツールを使用できます。

アセンブリをコピーできないときにコンパイル エラーを修正する方法

LoadAssembly メソッドで AUT にコードを追加しようとしたときに、次のコンパイル エラーが発生することがあります。
Could not copy '<assembly_name>.dll' to '<assembly_name>.dll'. The process cannot access the file.
このコンパイル エラーは、アセンブリがすでに AUT にロードされていて、上書きできないために発生します。

このコンパイル エラーを修正するには、AUT を閉じて、再度スクリプトをコンパイルします。

Apache Flex カスタム コントロールのテスト

Silk4J では、Apache Flex カスタム コントロールのテストがサポートされています。ただし、デフォルトでは、Silk4J は、カスタム コントロールの個別のサブコントロールを記録および再生することはできません。

カスタム コントロールをテストする場合、以下のオプションが存在します。

- 基本サポート

  基本サポートでは、動的呼び出しを使用して、再生中にカスタム コントロールと対話します。作業量が少なく済むこのアプローチは、テスト アプリケーションにおいて、Silk4J が公開しないカスタム コントロールのプロパティおよびメソッドにアクセスする場合に使用します。カスタム コントロールの開発者は、コントロールのテストを容易にすることのみを目的としたメソッドおよびプロパティをカスタム コントロールに追加することもできます。ユーザーは、動的呼び出し機能を使用してこれらのメソッドやプロパティを呼び出すことができます。

  基本サポートには以下のような利点があります。

  - 動的呼び出しでは、テスト アプリケーションのコードを変更する必要がありません。
  - 動的呼び出しを使用することによって、ほとんどのテストのニーズを満たすことができます。

  基本サポートには以下のような短所があります。

  - ロケーターには、具体的なクラス名が組み込まれません (たとえば、Silk4J では「//FlexSpinner」ではなく「//FlexBox」と記録されます)。
  - 記録のサポートが限定されます。
  - Silk4J では、イベントを再生できません。

  例を含む動的呼び出しの詳細については、「*Apache Flex* メソッドの動的呼び出し」を参照してください。
- 高度なサポート

  高度なサポートでは、カスタム コントロールに対して、特定のオートメーション サポートを作成できます。この追加のオートメーション サポートによって、記録のサポートおよびより強力な再生のサポートが提供されます。高度なサポートには以下のような利点があります。

  - イベントの記録と再生を含む、高レベルの記録および再生のサポートが提供されます。
  - Silk4J では、カスタム コントロールが他のすべての組み込み Apache Flex コントロールと同様に処理されます。
  - Silk4J API とシームレスに統合できます。
  - Silk4J では、ロケーターで具体的なクラス名が使用されます (たとえば、Silk4J では「//FlexSpinner」と記録されます)。

  高度なサポートには以下のような短所があります。

  - 実装作業が必要です。テスト アプリケーションを変更し、Open Agent を拡張する必要があります。

カスタム コントロールの管理

Silk4J が専用サポートを提供していないカスタム コントロールに対応するカスタム クラスを作成できます。 カスタム クラスを作成すると、以下の利点があります。

- スクリプトのロケーターが効率化されます。
- カスタム コントロールと対話するための再利用可能コードを簡単に記述できます。

例 : UltraGrid Infragistics コントロールのテスト

カスタム グリッド コントロールが Silk4J で汎用クラス Control として認識されるとします。 Silk4J のカスタム コントロール サポートを使用すると、以下の利点があります。

カスタム コントロール クラス名をロケーターで使用できるため、オブジェクトの認識率が高まります。

複数のオブジェクトが Control として認識されることがあります。 ローケーターには、特定のオブジェクトを識別するためのインデックスが必要です。 たとえば、オブジェクトはロケーター // Control[13] を使用して識別できます。 このコントロールのカスタム クラス (クラス UltraGrid など) を作成する場合は、ロケーター //UltraGrid を使用できます。 カスタム クラスを作成することによって、テスト対象アプリケーションが変更された場合にオブジェクト識別子が変わりやすい、大きな数字のインデックスを使用する必要がなくなります。

スクリプト内のコントロールに、再利用可能な再生操作を実行できます。

カスタム クラスを使用している場合、ユーザー インターフェイスにカスタム コントロールを指定すると生成されるクラスであるカスタム クラスに以下のコードを追加することで、グリッドのコンテンツをメソッド内に取り込む動作をカプセル化できます。

通常は、以下のいずれかの方法で、メソッドをカスタム コントロール クラスに実装できます。

- click、typeKeys、textClick、および textCapture などのメソッドを使用できます。
- AUT のオブジェクトで動的にメソッドを呼び出せます。
- AUT に追加したメソッドを動的に呼び出せます。 これは、この例で説明されている手法です。

以下のコードを使用して、「テスト対象アプリケーションにコードを追加してカスタム コントロールをテストする」の例で定義されている静的メソッドを呼び出すことができます。 メソッド GetContents が、生成されたクラス UltraGrid に追加されます。

```
// Java code
import com.borland.silktest.jtf.Desktop;
import
com.borland.silktest.jtf.common.JtfObjectHandle;
```

```
public class UltraGrid extends
com.borland.silktest.jtf.Control {

   protected UltraGrid(JtfObjectHandle
handle, Desktop desktop) {
      super(handle, desktop);
   }

   public List<List<String>> getContents()
{
      return (List<List<String>>)
invoke("AUTExtensions.UltraGridUtil.GetCont
ents", this);
   }
}
```

クラスをカスタム コントロールとして定義すると、Dialog クラスのように、すべての組み込みクラスの場合と同じ方法でそのクラスを使用できます。

```
// Java code
UltraGrid ultraGrid = mainWindow.find("//
UltraGrid[@automationId='my grid']");
List<List<String>> contents =
ultraGrid.getContents();
```

カスタム コントロールのサポート

Silk4J は、次のテクノロジ ドメインに対する UI のカスタム コントロールの管理をサポートします。

- Win32
- Windows Presentation Foundation (WPF)
- Windows Forms
- Java AWT/Swing
- Java SWT

Silk4J が専用サポートを提供していないカスタム コントロールに対応するカスタム クラスを作成するには、以下を実行します。

1. **Silk4J** > **カスタム コントロールの管理** をクリックします。**カスタム コントロールの管理** ダイアログ ボックスが開きます。
2. **Silk4J カスタム コントロールのパッケージ** フィールドで、任意の名前を入力するか、**参照** をクリックして、カスタム コントロールを含めるパッケージ を選択します。
3. 新しいカスタム クラスを作成するテクノロジ ドメインのタブをクリックします。
4. **追加** をクリックします。
5. 次のいずれかをクリックします。
   - **新しいカスタム コントロールの識別** をクリックし、**オブジェクトの識別** ダイアログ ボックスを使ってアプリケーション内のカスタム コントロールを直接選択します。
   - **新しいカスタム コントロールの追加** をクリックし、カスタム コントロールを手動でリストに追加します。

   新しい行がカスタム コントロールのリストに追加されます。
6. カスタム コントロールを手動でリストに追加するように選択した場合は、以下を実行します。
   a) **Silk Test 基本クラス** 列で、クラスの取得元となる既存の基本クラスを選択します。

このクラスは、ご使用のカスタム コントロールのタイプに最も一致率が高くなければなりません。

b) **Silk Test クラス** 列で、クラスの参照に使用する名前を入力します。

この名前は、ロケーターに表示されます。たとえば、//Control[13] でなく //UltraGrid を入力します。

注: 有効なクラスを追加すると、そのクラスは **Silk Test 基本クラス** リストで使用できるようになります。追加したクラスは、基本クラスとして再使用できます。

c) **カスタム コントロール クラス名** 列に、マップしているクラスの完全修飾クラス名を入力します。

たとえば、Infragistics.Win.UltraWinGrid.UltraGrid です。Win32 アプリケーションの場合、クラス名にワイルドカード ? および * を使用できます。

7. *Win32* アプリケーションの場合のみ：**クラスの宣言を使用する** 列で、値を **False** に設定して、カスタム コントロール クラスの名前を標準 Silk Test クラスの名前に単純にマップします。

   カスタム コントロール クラスを標準 Silk Test クラスにマップすると、テストの際に標準 Silk Test クラスでサポートされている機能を使用できます。カスタム コントロール クラスのクラス宣言を追加して使用する場合は、この値を **True** にします。

8. **OK** をクリックします。
9. スクリプトの場合のみ：

   a) カスタム コントロール用のクラスにカスタム メソッドおよびプロパティーを追加します。

   b) スクリプト内で新しいクラスのカスタム メソッドおよびプロパティーを使用します。

   注: カスタム メソッドおよびプロパティは記録されません。

注: スクリプト ファイル内のカスタム クラスまたは基本クラスの名前を変更しないでください。スクリプト内に生成されたクラスを変更した場合、予期しない動作を起こすことがあります。カスタム クラスにプロパティおよびメソッドを追加する場合にのみスクリプトを使用してください。それ以外の変更をカスタム クラスに加える場合は **カスタム コントロールの管理** ダイアログ ボックスを使用してください。

カスタム コントロール オプション

Silk4J > **カスタム コントロールの管理**。

Silk4J は、次のテクノロジ ドメインに対する UI のカスタム コントロールの管理をサポートします。

- Win32
- Windows Presentation Foundation (WPF)
- Windows Forms
- Java AWT/Swing
- Java SWT

Silk4J カスタム コントロールのパッケージ で、新しいカスタム クラスをその中に生成するパッケージを定義します。

カスタム コントロール クラスを標準 Silk Test クラスにマップすると、テストの際に標準 Silk Test クラスでサポートされている機能を使用できます。次の **カスタム コントロール** オプションが使用できます。

オプション	説明
Silk Test 基本クラス	自分のクラスの派生元として使用する既存の基本クラスを選択します。このクラスは、ご使用のカスタム コントロールのタイプに最も一致率が高くなければなりません。
Silk Test クラス	クラスの参照に使用する名前を入力します。この名前は、ロケーターに表示されます。
カスタム コントロールのクラス名	マッピングされているクラスの完全修飾クラス名を入力します。クラス名には、ワイルドカード ? および * を使用できます。

オプション	説明
クラスの宣言を使用する	このオプションは Win32 アプリケーションの場合のみ使用できます。デフォルト値は False で、カスタム コントロール クラスの名前が標準 Silk Test クラスの名前にマップされることを意味します。カスタム コントロール クラスのクラス宣言を追加して使用する場合は、この設定を True にします。

注: 有効なクラスを追加すると、そのクラスは **Silk Test 基本クラス** リストで使用できるようになります。追加したクラスは、基本クラスとして再使用できます。

例：UltraGrid Infragistics コントロールのオプションの設定

UltraGrid Infragistics コントロールをサポートするには、次の値を使用します。

オプション	値
Silk Test 基本クラス	Control
Silk Test クラス	UltraGrid
カスタム コントロールのクラス名	Infragistics.Win.UltraWinGrid. UltraGrid

Microsoft ユーザー補助を使用したオブジェクト解決の向上

Microsoft ユーザー補助を、クラス レベルでオブジェクトを簡単に認識するために使用することができます。 Internet Explorer や Microsoft アプリケーションのいくつかのオブジェクトには、ユーザー補助を有効にすることで Silk4J によってより良く認識されるようになるものがあります。 たとえば、ユーザー補助を有効にしないと、Silk4J は Microsoft Word のメニュー バーや表示されるタブについて基本的な情報のみを記録します。 しかし、ユーザー補助を有効にすると、 Silk4J はそれらのオブジェクトを完全に認識するようになります。

例

ユーザー補助を使用しないと、 Silk4J は DirectUIHwnd コントロールを完全に認識できません。これは、このコントロールのパブリックな情報が存在しないためです。 Internet Explorer は、2 つの DirectUIHwnd コントロールを使用しています。1 つはブラウザ ウィンドウの下部に表示されるポップアップです。 このポップアップには、通常、次の情報が表示されます。

- Internet Explorer を既定のブラウザにしたいかどうかを尋ねるダイアログ ボックス。
- ダウンロード オプション（ **開く**、 **保存**、 **キャンセル**）。

Silk4J でプロジェクトを開始して、 DirectUIHwnd ポップアップに対してロケーターを記録すると、ユーザー補助を無効にしている場合、単一のコントロールのみが表示されます。 ユーザー補助を有効にした場合には、 DirectUIHwnd コントロールを完全に認識した情報が得られます。

ユーザー補助の使用

Win32 では、ジェネリック コントロールとして認識されるコントロールにユーザー補助 サポートが使用されます。 Win32 は、コントロールを特定すると、ユーザー補助オブジェクトをコントロールのすべてのユーザー補助の子とともに取得しようとします。

ユーザー補助によって返されるオブジェクトは、AccessibleControl、Button、CheckBox のいずれかのクラスになります。 Button および Checkbox は、そのクラス用に定義されたメソッドとプロパティの通常

セットをサポートするので個別に扱われます。 ユーザー補助によって返されるすべてのジェネリック オブジェクトの場合、クラスは AccessibleControl です。

例

ユーザー補助が有効になる前、アプリケーションのコントロール階層が次のようになっていたとします。

- コントロール
  - コントロール
- ボタン

ユーザー補助を有効にすると、階層は次のように変わります。

- コントロール
  - コントロール
    - ユーザー補助コントロール
    - ユーザー補助コントロール
      - ボタン
- ボタン

ユーザー補助の有効化

Win32 アプリケーションをテストしているときに、Silk4J がオブジェクトを認識できない場合は、最初にユーザー補助を有効にする必要があります。ユーザー補助は、オブジェクトの認識機能をクラス レベルで強化するためのものです。

ユーザー補助を有効にするには、以下の手順を実行します。

1. **Silk4J** > **オプションの編集** をクリックします。**スクリプト オプション** ダイアログ ボックスが表示されます。
2. **詳細設定** をクリックします。
3. **Microsoft ユーザー補助を使用する** オプションを選択します。ユーザー補助が有効になります。

Silk4J での Unicode コンテンツのサポートの概要

Open Agent は、Unicode 対応済みです。つまり、Open Agent は、2 バイト (ワイド) 言語を認識できます。

Silk4J を使用して、中国語、韓国語、日本語 (漢字) などの 2 バイト言語や、それらを組み合わせたコンテンツを含んだアプリケーションをテストできます。

Open Agent は、以下をサポートします。

- Windows のローカライズ版。
- 国際化キーボードとネイティブ言語の入力方式エディター (IME)。
- テストケース、メソッドなどにパラメータとして国際化文字列を渡す、および文字列の比較。
- 複数の形式でのテキスト ファイルの読み書き：ANSI、Unicode、UTF-8。

新しい機能、サポートするプラットフォーム、テスト済みのバージョンについての情報は、『リリース ノート』を参照してください。

Silk4J を使用した 2 バイト文字のテストを行う前に

国際化されたアプリケーション、特に 2 バイト文字を含んだアプリケーションをテストするのは、英語 (1 バイト文字) のみを含んだアプリケーションをテストするよりも複雑です。国際化アプリケーションのテ

ストにおいては、オペレーティング システムのサポートから、言語パック、フォント、IME の動作、さらに複合した言語など、さまざまな問題を理解する必要があります。

Silk4J を使用してアプリケーションのテストを始める前に、以下を確認する必要があります。

- 必要なローカライズ OS、地域の設定、必要な言語パックがテスト対象アプリケーション (AUT) の要求を満たしているか。
- AUT を表示するのに必要なフォントがインストールされているか。
- データ入力に IME が必要なアプリケーションをテストする場合、適切な IME がインストールされているか。

テキスト解決のサポート

テキスト解決メソッドを使用して、オブジェクト解決で識別できない、高度にカスタマイズされたコントロールを含むテスト アプリケーションを便利に操作できます。座標ベースのクリックの代わりにテキスト クリックを使用し、コントロール内に指定されたテキスト文字列をクリックできます。

たとえば、次の表の 2 行目の最初のセルを選択することをシミュレートできます。

CustomerName	FirstOrder	ID	IsActive	CreditCard
Bob Villa	01.01.2008	0	☑	MasterCard
Brian Miller	02.01.2008	1	☐	Visa
Caral Rudd	03.01.2008	2	☑	American Ex...
Dan Rundgren	04.01.2008	3	☐	MasterCard
Devie Yingstein	05.01.2008	4	☑	Visa

セルのテキストを指定すると、次のコードが生成されます：

```
table.textClick("Brian  Miller");
```

テキスト解決メソッドは、次のテクノロジ ドメインでサポートされます。

- Win32
- WPF
- Windows Forms
- Java SWT と Eclipse
- Java AWT/Swing

  注: Java アプレット、および Java バージョンが 1.6.10 より前の Swing アプリケーションの場合、テキスト解決は追加設定なしでサポートされます。Direct3D をサポートしない Java バージョン 1.6.10 以降の Swing アプリケーションの場合は、アプリケーションの起動時に次のコマンドライン要素を追加する必要があります。

  ```
  -Dsun.java2d.d3d=false
  ```

  例：

  ```
  javaw.exe  -Dsun.java2d.d3d=false  -jar  mySwingApplication.jar
  ```

  Direct3D をサポートする Java アプレットや Swing アプリケーションでは、テキスト解決はサポートされません。
- Internet Explorer。

注: テキスト解決は、画面に表示されていないコントロールに対しては機能しません。たとえば、スクロールして画面外にあるテキストに対しては、テキスト解決を使用できません。

注: 対象のテキストで使用されたフォントがテストが実行されるマシン上にインストールされていない場合、テキスト解決が機能しない場合があります。

テキスト解決メソッド

次のメソッドにより、コントロールのテキストを操作できます。

TextCapture	コントロール内のテキストを返します。子コントロールのテキストも返します。
TextClick	コントロール内の指定テキストをクリックします。テキストが検出されるか、同期オプションで定義できるオブジェクト解決タイムアウトに達するまで待機します。
TextRectangle	コントロール内の特定テキストの矩形、またはコントロールの領域を返します。
TextExists	コントロール内またはコントロールの領域内に特定テキストが存在するかどうかを判断します。

テキスト クリックの記録

テキスト クリックの記録を有効にすると、Silk4J は、相対座標でクリックを記録するのではなく、TextClick メソッドを記録します。通常の座標ベースのクリックよりも TextClick 記録の方が結果が良いコントロールには、この方法を使用します。テキスト クリックが記録されない場合でも、任意のコントロールに対してテキスト クリックをスクリプトに挿入できます。

TextClick 操作を記録しない場合は、テキスト クリックの記録をオフに切り替えて通常のクリックを記録できます。

テキスト解決メソッドでは、部分的に一致する単語よりも完全に一致する単語が優先されます。Silk4J では、完全に一致する単語の前に部分的に一致する単語が画面に表示されていても、部分的に一致する単語よりも完全に一致した単語の出現が先に解決されます。完全に一致する単語がない場合は、部分的に一致する単語が画面に表示される順序で使用されます。

例

ユーザー インターフェイスには、テキスト「*the hostname is the name of the host*」が表示されているとします。次のコードでは、画面上では *host* の前に *hostname* が表示されていますが、*hostname* ではなく *host* がクリックされます。

```
control.textClick("host");
```

次のコードでは、2 番目の出現箇所であることを指定することで、単語 *hostname* 中の部分文字列 *host* がクリックされます。

```
control.textClick("host",  2);
```

Silk4J テストのグループ化

SilkTestCategories クラスを使用して、アノテーションを使用した Silk4J テストの実行、TrueLog の書き込み、およびテストのフィルタやグループ化を行うことができます。テスト クラスのカテゴリを定義して、Silk4J テストをこれらのカテゴリにグループ化することで、指定したカテゴリ、またはカテゴリのサブタイプに属しているテストのみを実行できます。詳細については、「*Grouping tests using JUnit categorie*」(JUnit カテゴリを使用したテストのグループ化) を参照してください。

Silk4J テストをカテゴリに含めるには、@IncludeCategory アノテーションを使用します。

カテゴリ SilkTestCategories クラスを使用して、カテゴリに含まれる Silk4J テストに対して TrueLog を書き込むことができます。また、SilkTestSuite クラスを使用して TruelLog を書き込みこともできます。詳細については、「コマンド ラインからテスト メソッドを再生する」を参照してください。

例

次の例では、カテゴリに含まれる Silk4J テストを実行する方法を紹介します。

テスト スクリプトの始めに次のような行を追加して、Category クラスをインポートしてください。

```
import  org.junit.experimental.categories.Category;
```

カテゴリは、クラス、またはインターフェイスとして実装することができます。たとえば、次のようになります。

```
public interface FastTests {}
  public interface SlowTests {}
```

カテゴリを使って、クラス全体にフラグを付けることができます。次のサンプル コードでは、クラスのすべてのメソッドが SlowTests カテゴリでフラグ付けされています。

```
@Category( { SlowTests.class})
public class A {
  @Test
  public void a() {
    ...
  }

  @Test
  public void b() {
    ...
  }
}
```

また、カテゴリを使って、クラスの個々のメソッドにフラグを付けることもできます。次のサンプル コードでは、メソッド d だけが FastTests カテゴリでフラグ付けされています。

```
public class B {
  @Test
  public void c() {
    ...
  }

  @Category(FastTests.class)
  @Test
  public void d() {
    ...
  }
}
```

複数のカテゴリを使って、クラスまたはメソッドにフラグを付けることができます。

```
@Category( { SlowTests.class, FastTests.class })
public static class C {
  @Test
  public void e() {
    ...
  }
}
```

特定のカテゴリのテストを実行するには、テスト スイートを用意する必要があります。

```
@RunWith(SilkTestCategories.class)
@IncludeCategory(SlowTests.class)
@SuiteClasses( { A.class, C.class })
// Note: SilkTestCategories is a kind of Suite
public static class SlowTestSuite {}
```

エラー「Category を型に解決できません」が発生する理由

Silk4J テストをグループ化するためにカテゴリを使用する場合に、「Category を型に解決できません」というエラーが表示される場合があります。これは、Category クラスをインポートしていない可能性があります。

テスト スクリプトの始めに次のような行を追加して、Category クラスをインポートしてください。

```
import org.junit.experimental.categories.Category;
```

スクリプトへの結果コメントの挿入

テストに関する補足情報を提供するために、結果コメントをテスト スクリプトに追加することができます。テストの実行中、結果コメントはテストの TrueLog ファイルに追加されます。

情報、警告、エラーの種類のコメントを追加できます。各コメントの種類のサンプルとして、以下にコード例を示します。

```
desktop.logInfo("This is a comment!");
desktop.logWarning("This is a warning!");
desktop.logError("This is an error!");
```

Silk Central からのパラメータを使用する

To enable Silk Central で テストに対して設定されたパラメータを Silk4J で使用することができるようにするには、 メソッド System.getProperty("myparam") を使用します。

Silk Central Connect を使用した構成テスト

Silk Central で作業するためには、有効な Silk Central の場所が設定されている必要があります。詳細については、「*Silk4J と Silk Central の統合*」を参照してください。

さまざまな構成 (オペレーティング システムと Web ブラウザーの組み合わせ) で自動テストを実行するために、Silk Central Connect を使用できます。Silk Central Connect は、使いやすいインターフェイスを提供し、テスト実行管理と構成テストの特徴を兼ね備えたツールで、以下のようなメリットがあります。

- すべての自動ユニット テストをさまざまな構成で簡単に実行できます。
- 先行投資無くさまざまな種類の構成に簡単にアクセスできるという、Amazon Web Services のメリットを利用できます。
- Silk Central Connect と Silk4J が強固に統合されているため、テストの作成、メンテナンス、実行が簡単に行えます。
- 結果を並べて分析することにより、さまざまな構成間でテスト全体がどのようになっているかを把握できます。

Silk Central Connect に関する追加の情報については、『*Silk Central Connect ヘルプ*』を参照してください。

Silk Central Connect のインストール、デプロイメント、ライセンス管理に関する情報については、『*Silk Central インストール ヘルプ*』を参照してください。

テスト環境の設定についての情報は、「実行サーバーを設定する」を参照してください。

実行時間の計測

Timer クラスが提供するメソッドやプロパティを使用して、テストの実行にかかる時間を計測できます。詳細については、Javadoc の「*Timer* クラス」を参照してください。

特に、 これらのメソッドとプロパティは、Silk Performer から呼び出されるテスト実行の計測に使用できるという利点があります。Silk4J と Silk Performer の統合についての詳細は、『*Silk Performer ヘルプ*』を参照してください。

テスト実行の遅延

テスト対象アプリケーションによっては、UI でのアプリケーション データの読み込みに多くの時間を必要とするため、テストの再生に必要なオブジェクトのロードが時間内に終わらない場合があります。このような AUT でテストの再生を正しく行うには、操作を実行する前にオブジェクトの存在を確認したり、操作の実行前にスリープを挿入する必要があります。

注: Micro Focus では、テストにスリープを追加することは基本的には推奨していません。 たいていの場合、オブジェクトが利用可能かどうかを Silk4J が自動的に検出するので、スリープはテストのパフォーマンスを落とす結果になるためです。

1. オブジェクトが AUT で利用可能かどうかを確認するには、exists メソッドを使用します。
   たとえば、INPUT ボタンが利用可能になるまで 6 秒間待機する場合は、テスト スクリプトに次の行を追加します。
   ```
   browserWindow.exists("//INPUT", 6000);
   ```
2. コントロールの操作を実行する前にスリープを追加するには、sleep メソッドを使用します。
   たとえば、6 秒間スリープする場合は、テスト スクリプトに次の行を追加します。
   ```
   Utils.sleep(6000);
   ```

これらのメソッドの詳細については、Javadoc を参照してください。

単一マシンでの複数 UI セッションのアプリケーションのテスト

単一マシンで複数の UI セッションを持つアプリケーションや、単一マシンで複数のエージェントをテストするには、そのマシンで複数の Open Agent インスタンスに接続します。すべてのエージェントがそれ自身の UI セッションで実行します。UI セッションは、リモート デスクトップ プロトコル (RDP) や Citrix ベースの接続です。

1. UI セッションを作成します。
2. コマンドライン ウィンドウを開きます。
3. Silk Test インストール ディレクトリの /ng/agent フォルダに移動します。
   たとえば、デフォルトでは、フォルダのパスは次のようになります：C:¥Program Files (x86)¥Silk¥SilkTest¥ng¥agent。
4. 各 UI セッションで、次のコマンドを実行します。openAgent.exe -infoServicePort=<port>

   注: このポート番号は、Silk4J スクリプトで Open Agent とエージェントが実行している UI セッションを識別するために使用されるため、一意のポート番号を使用してください。
5. Silk4J スクリプトを変更して、Open Agent インスタンスに接続します。
   Open Agent インスタンスに接続するために、スクリプトに次の行を追加します。
   ```
   Desktop desktopSession = new Desktop("hostname:port");
   ```

ここで、*hostname* はエージェントが実行しているマシンの名前で、*port* は指定した一意のポート番号です。

結果のオブジェクトは互いに独立しており、単一スレッド、複数スレッドのどちらででも使用することができます。

注: 複数の UI セッションでアプリケーションを起動する場合には、それぞれの UI セッションに対して基本状態を実行する必要があります。

注: リモート マシン上の複数の UI セッションでアプリケーションをテストするときに TrueLog を使用するには、生成された TrueLog ファイルをリモート マシンからローカル マシンに手動でコピーする必要があります。

例

複数の UI セッションをホストしているサーバー マシンの名前を *ui-srv* とします。ポート番号 22903、22904、22905 を使用して 3 つの UI セッションを作成します。

最初のセッションのために、コマンドライン ウィンドウを開き、agent ディレクトリに移動して、次を入力します。

```
openAgent.exe  -infoServicePort=22903
```

他の 2 つのセッション対して、ポート番号 22904 と 22905 をそれぞれ使用して同じことを行います。

Open Agent インスタンスに接続するために、スクリプトに次の行を追加します。

```
Desktop desktopSession1 = new Desktop("ui-srv:22903");
Desktop desktopSession2 = new Desktop("ui-srv:22904");
Desktop desktopSession3 = new Desktop("ui-srv:22905");
```

次のサンプル スクリプトでは、3 つの UI セッションそれぞれに対して単純なテキストを出力します。

```
public class TestMultiSession {
   Desktop d1 = new Desktop("ui-srv:22903");
   Desktop d2 = new Desktop("ui-srv:22904");
   Desktop d3 = new Desktop("ui-srv:22905");

   @Test
   public void test() {
      BaseState basestate = new BaseState();
      basestate.execute(d1);
      basestate.execute(d2);
      basestate.execute(d3);

      d1.<Window>find("//Window").typeKeys("Hello to session 1!");
      d2.<Window>find("//Window").typeKeys("Hello to session 2!");
      d3.<Window>find("//Window").typeKeys("Hello to session 3!");
   }
}
```

Selenium WebDriver の使用

Selenium WebDriver は強力なオープンソースの自動化ツールで、WebDriver に準拠したドライバを使用して任意のブラウザー上で Web アプリケーションの自動テストを行うことができます。Silk4J を WebDriver と使用すると、次のメリットがあります。

- コードを書く代わりに、新しい WebDriver スクリプトを簡単に記録できます。
- 既存の WebDriver スクリプトを Silk4J を使用して再生でき、さらに Silk Central を使用して実行をスケジュールすることもできます。
- WebDriver スクリプトに対して TrueLog ファイルを生成できます。Silk4J の TrueLog を使用すると、エラーを発生させたスクリプトの行を簡単に見つけることができます。

既存の Silk4J スクリプトと Selenium スクリプトの使用

注: このトピックに記述した機能には、Java 8 以降が必要です。

WebDriver 機能を既存の Silk4J スクリプトに追加して、混合スクリプトを作成することができます。このような混合スクリプトを作成するには、Open Agent を Selenium サーバーとして使用します。 Open Agent は、Silk4J の基本状態を実行するときに自動的に起動します。

1. *Silk Test Selenium* クライアント ライブラリ を Silk4J スクリプトを含むプロジェクトに追加します。
   a) **パッケージ エクスプローラー** で、プロジェクト ノードを右クリックします。
   b) **Silk4J ツール** > **WebDriver 機能の追加** を選択します。
2. Silk4J スクリプトを開きます。
3. 参照するブラウザー インスタンスの WebDriver ID を取得します。

```
BrowserApplication browserApplication = desktop.find("//BrowserApplication");
WebDriver driver = browserApplication.getWebDriver();
```

注: ハイブリッド アプリケーションをテストする場合には、次のコードを使用します。

```
BrowserWindow browserWindow = desktop.find("//BrowserWindow");
WebDriver driver = browserWindow.getWebDriver();
```

スタンドアロンの Selenium を使用しているときと同じ方法で、RemoteWebDriver オブジェクトを使用できるようになります。例：

```
driver.get("http://demo.borland.com/InsuranceWebExtJS/");
driver.findElementById("login-form:login");
```

スクリプトを実行すると、すべての Selenium の操作と Silk4J の操作が、スクリーンショットやパラメータと共に TrueLog に記録されます。

Selenium スクリプトの実行

Silk4J を使用して Selenium WebDriver スクリプトを実行すると、テスト実行中に同期を使用したり、TrueLog を作成することができます。Open Agent を実行しているマシン上でブラウザーの種類をケイパビリティ（Capabilities）で指定し、RemoteWebDriver を作成して Open Agent 上で実行している Selenium サーバーに接続する場合は、Silk4J は自動的に対応するブラウザーを起動します。ブラウザーの種類が指定されていない場合、Open Agent は、あらかじめ Silk4J の基本状態の実行によって起動されたブラウザーなど、既存のブラウザー インスタンスを再利用しようとします。

Selenium スクリプトの実行中に、実行をビジュアル化する TrueLog ファイルを生成するには、「*Silk Test TrueLog API for Selenium WebDriver*」を使用できます。この API は REST インターフェイスとして実

装されており、この API によって使用されるエンドポイント ホストとポートは、Selenium サーバーによって使用されるものと同じです。たとえば、*http://localhost:4444/silktest/truelog* のようになります。

REST API ファイルは、Silk Test インストール フォルダの ¥ng¥TrueLogAPI¥ にあります。

- SilkTestTrueLogService-doc.html ファイルは、REST API のドキュメントです。
- SilkTestTrueLogService.yaml ファイルは、REST API の宣言です。
- trueLogApiClient.jar ファイルは、REST API の Java クライアントです。

REST API 宣言ファイルを *Swagger Editor* にインポートして、エディタの **Generate Client** 機能を使用すると、宣言ファイルを変更をせずに、Python、Ruby、JavaScript、C# などのさまざまな言語のクライアント API ライブラリを生成できます。そして、選択したプログラム言語の TrueLog API をダウンロードできます。

Silk4J を使用して、次のブラウザーに対して Selenium スクリプトを実行できます。

- Google Chrome
- Microsoft Edge
- Mozilla Firefox
- Apple Safari

1. Silk4J をインストールしたマシンで Open Agent を開始します。
2. アプリケーションをテストするブラウザーを開始します。
   - Google Chrome、Microsoft Edge、Mozilla Firefox の場合、WebDriver のデフォルトのケイパビリティを使用して、ブラウザーを開始できます。たとえば、次のコードで示すように、Java バインディングを使用して Google Chrome を開始できます。

     ```
     RemoteWebDriver driver = new RemoteWebDriver(new URL("http://localhost:
     4444/wd/hub"), DesiredCapabilities.chrome());
     ```
   - Apple Safari の場合、次のコードで示すように、カスタム ブラウザー名をケイパビリティとして指定する必要があります。

     ```
     DesiredCapabilities safari = new DesiredCapabilities();
      safari.setCapability("browserName", "SilkSafari");
     ```
3. 省略可能：追加のオプションをケイパビリティとして渡します。

   Silk4J のカスタム オプションは、*silkTestOptions* というケイパビリティ名を持つマップとして渡します。

   たとえば、次のサンプル コードは、**syncEnabled** オプションを設定し、Selenium での自動同期を有効化する方法を示しています。

   ```
   Map<String, Object> options = new HashMap<>();
   options.put( "syncEnabled" , true);
   capabilities.setCapability( "silkTestOptions" , options);
   ```

   以下のオプションを利用できます。

オプション	型	説明
commandLineArguments	String	起動するブラウザーに渡すコマンドライン引数を指定します。
connectionString	String	リモート マシン上で実行するブラウザーの接続文字列を指定します。
startUrl	String	ブラウザーの起動時に移動する URL を指定します。
syncEnabled	boolean	Silk Test AJAX 同期の有効/無効を指定します。デフォルト値は、false です。
trueLogEnabled	boolean	このオプションを使用して、TrueLog の有効/無効を指定します。デフォルト値は、true です。

オプション	型	説明
trueLogId	String	TrueLog API の StartTrueLog メソッド呼び出し時に返される TrueLog セッションの識別子です。Open Agent に対して WebDriver 操作を含める TrueLog を指定する場合に、この識別子が必要になります。
trueLogPath	String	カスタム TrueLog ファイルのパスを指定します。デフォルトでは、Silk Test のログ ディレクトリである %LOCALAPPDATA%¥Silk¥SilkTest¥logs の下に TrueLog は出力されます。
trueLogScreenshotMode	String	スクリーンショットを TrueLog に追加するモードを指定します。 **OnError** エラーが発生したときに、スクリーンショットが TrueLog に追加されます。 **always** 各操作に対して、スクリーンショットが TrueLog に追加されます。

4. 省略可能：Selenium サーバーが使用するポートを指定するには、次の設定を行います。
   a) %APPDATA%¥Silk¥SilkTest¥conf に移動します。
   b) Selenium.properties という名前の新しいプロパティ ファイルを作成します。
   c) 新しいファイルに、selenium.server.port=<port name> を入力します。

次のサンプル コードは、Selenium Java バインディングを使用して、同期をオンにして Google Chrome を開始します。

```
DesiredCapabilities capabilities = DesiredCapabilities.chrome();
Map<String, Object> options = new HashMap<>();
options.put("syncEnabled", true);
capabilities.setCapability("silkTestOptions", options);
RemoteWebDriver driver = new RemoteWebDriver(new URL("http://localhost:4444/wd/
hub"), capabilities);
```

次のサンプル コードは、Selenium Java バインディングを使用して、Open Agent に対して TrueLog ファイルを指定して Google Chrome を開始します。

```
TrueLogAPI truelogAPI = new TrueLogAPI();
truelogAPI.startTrueLog("C:/temp/myTrueLogFile.tlz");
String truelogId = truelogAPI.getTrueLogId();

DesiredCapabilities capabilities = DesiredCapabilities.chrome();
Map<String, Object> options = new HashMap<>();
options.put("trueLogId", trueLogId);
capabilities.setCapability("silkTestOptions", options);
RemoteWebDriver driver = new RemoteWebDriver(new URL("http://localhost:4444/wd/
hub"), capabilities);
```

スタンドアロンの Selenium を使用しているときと同じ方法で、RemoteWebDriver オブジェクトを使用できるようになります。例：

```
driver.get("http://demo.borland.com/InsuranceWebExtJS/");
driver.findElementById("login-form:login");
```

スクリプトを実行すると、すべての Selenium の操作と Silk4J の操作が、スクリーンショットやパラメータと共に TrueLog に記録されます。

テキスト フィールドへの特殊キーの入力

Web アプリケーションをテストする場合、Silk4J では次の記録モードを選択できます。

- **Silk Test**： Silk4J ロケーターを記録します。
- **WebDriver**： WebDriver ロケーターを記録します。この記録モードは、Internet Explorer に対して記録を行う場合には、サポートされません。

このトピックでは、**WebDriver** 記録モードを使って記録するスクリプトに、特殊キーを入力するコードを追加する方法について説明します。**Silk Test** 記録モードを使った特殊キーの処理についての情報は、API ドキュメントの typeKeys メソッドを参照してください。

WebDriver 記録モードでは、特殊キーをかぎ括弧で囲んで指定します。たとえば、<back_space> や <enter> のように指定します。

クラス *org.openqa.selenium.Keys.java* のすべての特殊キーを Silk4J では指定できます。キーの値は、大文字小文字を区別しません。

注: Mozilla Firefox 以外のすべてのサポート ブラウザーで、sendKeys メソッドを使って複数の特殊キーやキー コードを送信できます。ただし、Mozilla Firefox で動作するクロスブラウザー スクリプトを作成する場合は、sendKeys の呼び出しにつき、1 つの文字列、特殊キー、キー コードを送信することを Micro Focus では推奨しています。

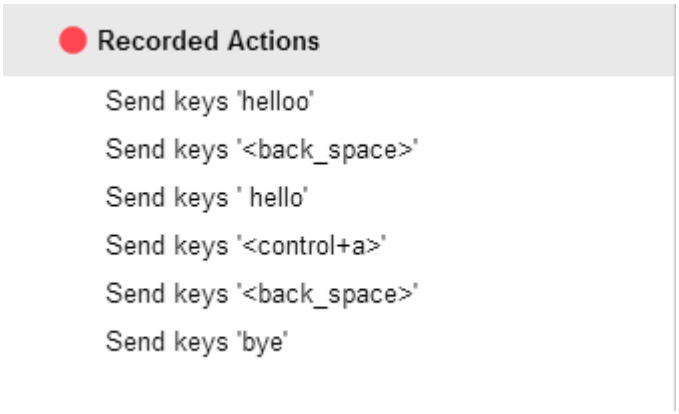
例

記録時にテキスト フィールドに対して sendKeys メソッドを使用するとき、Keys パラメーターに対して、次のようなパラメーター値を指定できます。

Keys パラメーター	Keys パラメーター タイプ	生成される Java コード
hello	単純な文字列	driver.findElement(By.id("login-form:email")).sendKeys("hello");
<back_space>	特殊文字	driver.findElement(By.id("login-form:email")).sendKeys(Keys.BACK_SPACE);

Keysパラメーター	Keysパラメーターのタイプ	生成される Java コード
<control+a>	コード	driver.findElement(By.id("login-form:email")).sendKeys(Keys.chord(Keys.CONTROL, "a"));

たとえば、次のような操作を記録したとします。



1 つの特殊文字またはキー コードだけを sendKeys メソッドの呼び出しで指定しているため、Mozilla Firefox を含むすべてのサポート ブラウザーでこれらの操作を再生できます。

対応する生成コード（Java）は、次のようになります。

```
driver.findElement(By.id("login-form:email")).sendKeys("helloo");
driver.findElement(By.id("login-form:email")).sendKeys(Keys.BACK_SPACE);
driver.findElement(By.id("login-form:email")).sendKeys("  hello");
driver.findElement(By.id("login-form:email")).sendKeys(Keys.chord(Keys.CONTROL, "a"));
driver.findElement(By.id("login-form:email")).sendKeys(Keys.BACK_SPACE);
driver.findElement(By.id("login-form:email")).sendKeys("bye");
```

キーワード駆動テストのパフォーマンス テストとしての使用

Silk4J を使用した機能テストや回帰テストの実行に加えて、キーワード駆動テストを Silk Performer にエクスポートして、パフォーマンス テストや負荷テストに使用できます。

キーワード駆動テストを Silk Performer にエクスポートする前に、Silk Performer 18.0 以降がインストールされていることを確認してください。キーワード駆動テストを含んだ Silk4J プロジェクトを選択し、Silk4J メニューの **パフォーマンス テストとしてエクスポート** をクリックします。

Silk4J プロジェクトを Silk Performer にエクスポートした後で、Silk Performer の対応するプロジェクトを更新する場合は、次の手順を実行します。

1. Silk Performer を起動します。
2. 更新するプロジェクトを開きます。
3. Silk4J プロジェクトを開きます。
4. **パフォーマンス テストとしてエクスポート** をクリックします。
5. 現在の Silk Performer プロジェクトを更新することを確認します。

Silk Performer プロジェクトの **データ ファイル** ノードで、ライブラリへの参照が更新されます。

その他の情報については、Silk Performer のドキュメントを参照してください。

既知の問題

このセクションでは、Silk4J の既知の問題とその解決策を示します。

全般的な問題

オブジェクト マップを開くのに時間がかかる

.NET 4 を使用している場合、大規模なオブジェクト マップ資産があると読み込みに時間がかかります。.NET 4.5 をインストールすると、この問題を解決できます。

リモート デスクトップまたはリモート デスクトップ接続 (RDC) が最小化されると、Silk Test が機能しない

リモート デスクトップ プロトコル (RDP) 経由でデスクトップに接続している場合、マウスとキーボードを使ってデスクトップに接続することにより、デスクトップの所有権を獲得します。デスクトップの所有権を解放せずにデスクトップを最小化すると、マウス クリックまたはキーストロークの再生がすべて未定義になります。

回避策として、VNC ベースのリモート操作ツールを使用することができます。この場合、クライアントが最小化された場合でも再生を継続できます。

Check Point ファイアウォールがインストールされている場合に Open Agent が起動しない

システムに Check Point ファイヤウォールまたは Check Point ZoneAlarm ファイヤウォールをインストールしている場合は、ファイヤウォールが Agent と infoservice 間の通信を中断するために Open Agent を起動できません。

Open Agent を起動するには、システムから Check Point ファイヤウォールをアンインストールする必要があります。

domDoubleClick メソッドの modifiers パラメータが無視される

domDoubleClick メソッドのオーバーロードで修飾キーを指定できません。パラメータが指定されている場合でも、修飾キーはダブルクリックされません。修飾キーを指定できる domDoubleClick メソッドのオーバーロードは、非推奨です。修飾キーを指定するには、doubleClick メソッド (modifiers パラメータを取るオーバーロード メソッドをサポートするクライアントを使用している場合)、または PressKeys および ReleaseKeys メソッドを使用します。

Silk Test が Metro スタイル アプリのテストをサポートしない

Silk Test は、Microsoft Windows 8、Microsoft Windows 8.1、Microsoft Windows 10 上の Metro スタイル アプリのテストをサポートしません。Metro スタイル アプリは、Windows 8 スタイル、Modern UI スタイル、Windows ストア スタイル、ユニバーサル Windows プラットフォーム (UWP) アプリとも呼ばれます。

Microsoft Windows 8 の組み込みスペル チェックがテストの再生に干渉する

Microsoft Windows 8 の組み込みスペル チェックは、Internet Explorer 10 などのアプリケーションで有効にできます。

記録中に単語のスペルを間違え、この単語の入力を再生すると、スペル チェッカーはこの単語をマークするか、間違われやすい単語の場合は自動的に修正します。これは、実際のユーザーに対して行われる動作と同じです。スペル チェック機能が含まれていないオペレーティング システムでテストを作成した場合、

Microsoft Windows 8 でテストを再生すると、予期せぬ結果が生じることがあります。スペル チェックを無効にするには、次の手順を実行します。

1. **Windows キー + C** を押します。
2. チャーム バーで **設定** をクリックします。
3. **PC 設定の変更** を選択します。
4. **全般** を選択すると、スペル チェック セクションが表示されます（使用中の言語によってスペル チェック機能の有無は異なります）。

   注: これはシステム全体の設定で、Internet Explorer 固有の設定ではありません。

5. **スペル ミスの語句を自動修正する (Autocorrect misspelled words)** をオフに設定します。
6. **スペル ミスの語句を強調表示する (Highlight misspelled words)** をオフに設定します。

.NET アプリケーションを DevPartner Studio (DPS) から起動すると、Silk Test で認識されないことがある

この問題を解決するには、以下のステップを実行します。

1. Silk Test のインストール フォルダに移動します (デフォルトでは C:¥Program Files¥Silk¥SilkTest)。
2. Windows Forms アプリケーションの場合は、ng¥agent¥plugins ¥com.borland.fastxd.techdomain.windowsforms.agent_<バージョン番号> へ移動します。
3. Windows Presentation Foundation (WPF) の場合は、ng¥agent¥plugins ¥com.microfocus.silktest.techdomain.wpf.agent_<バージョン番号> へ移動します。
4. メモ帳で、plugin.xml ファイルを開いて、<loadparameters> セクションに以下の行を追加します。

   ```
   <param  name="frameworkAssembly">mscoree.dll</param>
   ```

5. コンピュータからログアウトして、再びログインします。 DevPartner Studio によって起動されたアプリケーションに対して、Silk Test が期待どおりの動作をします。

イメージ領域に対するクリックの記録時に矩形領域によってハイライトされる位置がずれる

エリア マップのような複雑なイメージの一部に対するクリックを記録する場合、イメージの適切な領域が緑色の矩形領域によってハイライトされません。ただし、再生時にクリックは正しく実行されます。

Silk Test のインストール中に Windows Defender が有効になっていると Open Agent が起動しない

Silk Test のインストール中にシステムの Windows Defender が有効になっていると、インストールが完了した後に Open Agent を起動できなくなる場合があります。Windows Defender は、ホットフィックス セットアップに必要な操作を妨げる場合があります。回避策として、Silk Test のインストール中は Windows Defender を無効にしてください。

IME エディタが特定の場所から開かれたときにポップアップする

IME エディタを開くと、テキスト フィールド内に開くのではなく、現在の画面の左上隅にポップアップして開きます。

この動作は、次の場所から IME エディタを開いたときに発生します。

- **Silk Recorder**
- **キーワード駆動テスト エディター**
- **キーワード** ビュー

Num Lock キーがオンの状態で Shift + Insert を使用できない

Left Shift + Insert や **Right Shift + Insert** を使用して typeKeys メソッドでクリップボードの内容を貼り付ける操作は、Num Lock キーがオンの場合に機能しません。

typeKeys メソッドを呼び出す前にテスト スクリプトで Num Lock キーをオフにすることで、この問題を回避できます。

モバイル Web アプリケーション

Apple Safari のフレームを Silk4J がサポートしない

Silk4J は、iOS 上の Apple Safari を使用した HTML フレームおよび iframe のテストをサポートしません。

Chrome for Android 43 以降： 拡大縮小とスクロールを同時に Silk4J がサポートしない

Chrome for Android 43 以降でモバイル Web アプリケーションを記録しているとき、モバイル Web アプリケーションが拡大されて左上隅が画面から見えなくなると、期待通り機能しない場合があります。モバイル Web アプリケーションのコントロールに対して緑の矩形領域が記録時に正しく表示されない場合、モバイル Web アプリケーションを完全に縮小し、モバイル Web アプリケーションの左上隅にスクロールして、**記録** ウィンドウを更新してください。

Web アプリケーション

100% 以外の拡大レベルを使用して記録すると期待通り機能しない可能性がある

100% 以外の拡大レベルを使用して Web アプリケーションを記録すると、期待通り機能しない可能性があります。Web アプリケーションに対する操作を記録する前に、ブラウザの拡大レベルを 100% に設定してください。

Google Chrome

Windows で記録中のロケーターが Google Chrome で失敗する

Google Chrome で Web アプリケーションをテストするときに、アプリケーションを実行している Google Chrome インスタンスのアプリケーション構成中に複数のウィンドウが開いていると、Windows で記録中のロケーターが失敗します。アプリケーション構成中に他の Google Chrome ウィンドウを閉じると、エラーは発生しなくなります。

Google Chrome のバックグラウンド アプリケーションでオートメーション サポートを読み込むことができない

Google Chrome を使用して Web アプリケーションをテストしている場合に、**Google Chrome を閉じた際にバックグラウンド アプリケーションの処理を続行する** チェックボックスがチェックされていると、Silk Test は Google Chrome を再起動してオートメーション サポートを読み込むことができません。

Windows Aero が無効なときにモーダル ダイアログのロケーターを Silk Test が記録できない

Windows Aero が無効化されている場合、モーダル ダイアログが認識されないため、このようなダイアログのロケーターを選択できません。回避策として、モーダル ダイアログが表示されているときには、**Locator Spy** または **オブジェクトの識別** ダイアログ ボックスを使用して、ロケーターを手動で作成および検証してください。

Silk Test が埋め込み PDF を表示しない

Google Chrome 42 以降を使用すると、Google Chrome は、埋め込み PDF を表示するために使用する NPAPI プラグインをデフォルトでブロックします。このため、Silk Test は埋め込み PDF を Google Chrome 42 以降に表示する代わりに、埋め込み PDF をダウンロードします。

- Google Chrome 44 以前を使用している場合、アドレス バーに以下を入力することにより、Google Chrome での NSAPI プラグインのブロックを解除できます。

  ```
  chrome://flags/#enable-npapi
  ```
- Google Chrome 45 以降を使用している場合、NPAPI プラグインは Google Chrome から完全に削除されており、再有効化するオプションも無いため、すべての PDF はダウンロードされます。

テストの実行時に Google Chrome (49 以前) との接続がタイムアウトする

低速なマシンで Google Chrome (49 以前) に対するテストを実行すると、接続タイムアウトが発生し、テストが失敗する場合があります。以下のエラー メッセージが表示されます。
'*' の実行中にエラーが発生しました。ブラウザー オートメーションとの通信がタイムアウトしました。

このような接続タイムアウトを回避するために、十分な処理速度を持つテスト マシンを使用してください。たとえば、低速の仮想マシン (VM) 上でテストしている場合には、VM に CPU コアを追加することで処理速度を向上させることができます。

Internet Explorer

Google ツールバーを使用すると、Web アプリケーションの記録に支障をきたす

Internet Explorer 8 で Google ツールバーを使用すると、Web アプリケーションのロケーターの記録に支障をきたします。

Google ツールバーをオフにしてから、Web アプリケーションを記録してください。

Microsoft Silverlight アプリケーション

一部の Microsoft Silverlight アプリケーションで、Silk Test との通信の際に Internet Explorer がハングします。32 ビット プラットフォームでは、問題の防止に役立つ MS KB 2564958 (Active Accessibility の更新プログラム) を参照してください。

Silk Test 13.5 より前のバージョンの Silk Test を使用して記録したロケーターが Internet Explorer で動作しない

Silk Test 13.5 で、Internet Explorer の textContents 属性における空白文字の標準化を改良しました。この変更は、Silk Test のクロスブラウザ機能を改善するための措置で、textContents 属性を利用しているロケーターに影響を与える可能性があります。この属性は、Silk Test 13.5 以前のリリースを使用して記録されたスクリプトで使用されています。

UAC が Microsoft Windows 8 以降および Internet Explorer 11 で有効化されているとき、Open Agent に対して高い昇格を有効化できない

UAC が有効化され、Internet Explorer と Open Agent の両方を高い昇格で実行するとき、Microsoft Windows 8 以降で Internet Explorer 11 上の Web アプリケーションをテストできません。

入力方式エディター (IME) における既知の問題

- Silk Test は、Internet Explorer 11 での日本語入力において、半角スペースを記録しません。
- Silk Test は、Internet Explorer 11 の互換モードでの IME 入力を記録しません。
- 日本語 IME モードで、**スペース** を押すと、Silk Test は現在の IME の候補を記録することがあります。この問題を避けるには、**変換** を使用してください。

Java 1.7 update 71 よりも新しいバージョンを使ったアプレットのテスト時に Internet Explorer が動作を停止する場合がある

Java 1.7 update 71 (7u71) よりも新しいバージョンを使ってアプレットをテストすると、Internet Explorer が動作を停止する場合があります。

テストの実行中に Internet Explorer が応答しなくなる

Internet Explorer 10 以降でテストを実行していると、Internet Explorer のスレッドが中断され、デッドロックが発生したことが原因で、応答しなくなる場合があります。この問題は、Internet Explorer の新しいセキュリティ機能が原因です。

この問題を解決するには、セキュリティ機能を無効化します。

1. **レジストリ エディタ** が開きます。
2. **OverrideMemoryProtectionSetting** エントリが **HKEY_CURRENT_USER¥SOFTWARE¥Microsoft¥Internet Explorer¥Main** キーに存在しない場合は、エントリを作成します。
3. レジストリ キー **HKEY_CURRENT_USER¥SOFTWARE¥Microsoft¥Internet Explorer¥Main::OverrideMemoryProtectionSetting** の値を 2 に設定します。

JavaScript 警告処理 API メソッドが埋め込み Internet Explorer で機能しない

BrowserWindow クラスの以下の JavaScript 警告処理メソッドが、埋め込み Internet Explorer のテスト時に機能しません。

- acceptAlert メソッド
- dismissAlert メソッド
- getAlertText メソッド
- isAlertPresent メソッド

Microsoft Edge

Microsoft Edge から開いたウィンドウがサポートされない

タブではなく、Microsoft Edge の新しいウィンドウとして開いたウィンドウはサポートされません。Silk4J は、このようなウィンドウを正しく閉じることができず、このようなウィンドウを閉じると、エージェントの状態が不正になります。

iframe または frame で開いた JavaScript 警告を閉じることができない

JavaScript 警告 alert()、prompt()、confirm() が、iframe または frame で開かれた場合、Microsoft Edge は閉じることができません。

ネイティブ ブラウザーのコンテキスト メニューを開けない

Microsoft Edge を右クリックして、ネイティブ ブラウザーのコンテキスト メニューを開く操作がテストに含まれていると、そのテストはハングします。Microsoft Edge によって開かれた HTML メニューでは、この問題は発生しません。

UAC を無効にした状態で Microsoft Edge に対するテストを実行できない

UAC を無効にした状態で Microsoft Edge に対してテストを実行すると、次のエラー メッセージが表示されます。
Failed to start application 'Edge'. Unable to parse remote response: Unknown error

この問題を解決するには、UAC を有効にしてください。

Mozilla Firefox

Adobe FlashPlayer を使用したアプリケーションの呼び出しが Mozilla Firefox を使用した場合に正しく同期されない

最近のバージョンの Adobe FlashPlayer を Mozilla Firefox で使用している場合、呼び出しが正しく同期されないことがあります。次の問題が発生する可能性があります。

- Mozilla Firefox がスクリプトの実行がハングしたものと誤って認識し、スクリプトが正しく実行されているにも関わらず、スクリプトの実行を続けるかどうかを確認するダイアログ ボックスが表示される場合があります。
- SetFocus が正しく機能しないため、文字の入力が機能しない場合があります。
- UI には新しい値が表示されているにも関わらず、Adobe オートメーションが古い値を返す場合があります。

Adobe FlashPlayer を使用するアプリケーションでこのような問題に遭遇した場合は、Adobe FlashPlayer の ProtectedMode をオフにしてください。詳細については、*http://forums.adobe.com/thread/1018071* を参照し、*Last Resort* に記述された情報をお読みください。

SAP アプリケーション

SAPTree クラスの HierarchyHeaderWidth および ColumnOrder プロパティが書き込み専用になっている

自動化に関するドキュメントに記載されている場合を除いて、SAPTree の HierarchyHeaderWidth および ColumnOrder プロパティは書き込み専用で、読み込むことはできません。

これらのプロパティを使用する場合、スクリプトで、読み込みではなく、書き込みが使用されていることを確認します。

SAPTree クラスの GetColumnIndexFromName() が「特定できないエラー」により失敗する

SapTree クラスの GetColumnIndexFromName() は「特定できないエラー」により失敗することがあります。これは SAP オートメーションの既知の問題です。

SAP Web サイトでこの問題が解決されているかどうか確認してください。

コンテキスト メニュー項目の SAPTree クラスの Select() メソッドの呼び出しに失敗する

コンテキスト メニュー項目の SAPTree クラスの Select() メソッドの呼び出しに失敗することがあります。

代わりに親コントロールの SelectContextMenuItem を呼び出します。これは SAP オートメーションの既知の問題です。

水平スクロールバーの Position プロパティが常に 1 を返す

水平スクロールバーの Position プロパティは常に 1 を返します。これは SAP オートメーションの既知の問題です。

SAP Web サイトでこの問題が解決されているかどうか確認してください。

SAPNetPlan クラスがサポートされていない

この問題は今後のリリースで解決される予定です。

SAP スクリプトを実行すると、再生エラーが発生する

ある状況下で、SAP テストを記録して再生すると、次のエラーが発生する場合があります：この操作を完了するのに必要なデータは、まだ利用できません。このエラーは、Silk4J が記録した操作の実行が速すぎたことを意味しています。

この問題を解決するには、テスト スクリプトにスリープを追加したり、関数の呼び出しやプロパティの設定の後に Silk4J が待機する時間を増やすために、再生後の遅延を増やしてください。このオプションについての詳細は、「エージェント オプション」を参照してください。また、xBrowser テクノロジ ドメインではなく、SAP オートメーションを使用して問題が発生する操作を再生するように、スクリプトを変更します。たとえば、その操作を DomLink.Select から SapHTMLViewer.SapEvent に変更します。

SAPGUI クライアント 7.30 を使用するとメソッド getCurrentRow が誤った値を返す

SAPGUI クライアント 7.30 を使用しているときに、メソッド getCurrentRow を呼び出すと、メソッドは行番号ではなく誤った値 -1 を返す場合があります。

メソッド resizeWorkingPane が SAPGUI クライアント 7.30 で正しく機能しない

SAPGUI クライアント 7.30 を使用しているときに、メソッド resizeWorkingPaneEx を呼び出すと、メソッドは workingPane のサイズを変更せずに、getSapWindow().getWidth() の呼び出し結果は、ウィンドウの幅と異なる値を返します。

Oracle Forms

Java 1.7 update 60 よりも新しいバージョンを使った Oracle Forms のテストを Silk4J がサポートしない

Silk4J は、Java 1.7 update 60 よりも新しいバージョンを使った Oracle Forms のテストをサポートしません。

Silk4J

Silk4J メニューが Eclipse 4.1 で正しく開かない

Eclipse 4.1 で Silk4J メニュー ボタンをクリックしても、メニュー項目が表示されません。これは、Eclipse 4.2 で修正されている Eclipse の問題です。詳細については、*https://bugs.eclipse.org/bugs/show_bug.cgi?id=367159* を参照してください。

オブジェクト マップを使用していると、先頭がスラッシュになっていない既存のロケーターが機能しなくなる

クラス名のみを含み、スラッシュで開始していないロケーター (PushButton など) は、オブジェクト マップが存在する場合、機能しなくなります。この問題により、Silk Test 14.0 よりも前のバージョンの Silk Test で作成された既存のスクリプトが壊れる可能性があります。前の例では、スクリプトは次のエラーで失敗します：
Identifier 'PushButton' was not found in the Object Map.

クラス名以外のものも含む、より複雑なロケーター (PushButton[@caption=OK] など) の場合は、オブジェクト マップが存在する場合でも機能し続けます。

この問題を修正するには、そのようなロケーターの先頭に // を追加します。たとえば、次のコード内のロケーター PushButton が機能しなくなったとします：

```
PushButton button = mainWindow.find("PushButton");
```

その場合、コードを次のように変更してください：

```
PushButton button = mainWindow.find("//PushButton");
```

使用状況データの収集の有効化/無効化

Micro Focus では、テスト全体にわたるユーザー エクスペリエンスの向上を図るため、Micro Focus ソフトウェアとサービスの使用方法に関する情報を収集し、Micro Focus に提供していただくことを望んでいます。Silk4J のインストール時に使用許諾契約の条項に同意することにより、Silk4J の使用方法に関する情報と、Silk4J をインストールしたコンピューターに関する情報の収集を Micro Focus に許可したことになります。Micro Focus は、名前や住所などの個人の特定を可能にする情報の収集や、スクリプトやパスワードなどのデータ ファイルの収集は行いません。この情報の収集を Micro Focus に許可することにより、Micro Focus では、その傾向と使用方法のパターンの識別に役立てることができます。

Micro Focus による使用状況データの収集を有効化または無効化するには：

1. メニューから **Silk4J** > **Silk4J のバージョン情報** をクリックします。
2. バージョン情報ダイアログ ボックスで、**カスタマ フィードバック オプション** をクリックします。
3. 以下のオプションのいずれかを選択します。
   - 使用状況データの収集を有効化するには、**はい、参加します** をクリックします。
   - 使用状況データの収集を無効化するには、**いいえ、参加しません** をクリックします。
4. **OK** をクリックします。

Micro Focus へのお問い合わせ

Micro Focus は、世界的規模のテクニカル サポートおよびコンサルティング サービスを提供します。すべての顧客のビジネスを成功に導くために、信頼できるサービスをタイムリーに提供するように、Micro Focus はワールドワイドのサポート体制を整えています。

保守およびサポート契約を結んだすべてのお客様、および製品を評価中のお客様は、カスタマー サポートを受けることができます。高度なトレーニングを積んだスタッフが、お客様の質問にできる限り迅速かつ専門的にお答えします。

http://supportline.microfocus.com/assistedservices.asp にアクセスするか、またはメールを supportline@microfocus.com に送信して、Micro Focus SupportLine と直接連絡できます。

また、*http://supportline.microfocus.com* の Micro Focus SupportLine では、 最新のサポートに関するニュースや、さまざまなサポート情報を得ることができます。 このサイトに初めてアクセスした場合は、ユーザー登録が必要な場合があります。

Micro Focus SupportLine が必要とする情報

Micro Focus SupportLine をご利用の場合は、可能な限り次の情報を提供ください。情報が多ければ多いほど、Micro Focus SupportLine はお客様に適切なサービスを提供できます。

- 問題の原因と思われるすべての製品の名前およびバージョン番号
- 使用しているコンピュータの製造元およびモデル
- システム情報 (オペレーティング システムの名前やバージョン、プロセッサやメモリの詳細など)
- 問題の詳細な説明 (問題の再現手順など)
- 発生したエラー メッセージ
- お客様のシリアル番号

これらの番号は、Micro Focus から受け取った Electronic Product Delivery Notice 電子メールの件名および本文に記述されています。

索引

記号

数字

A

C

D

E

F

G

I

J

L

M

O

V

W

X

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

つ

て

と

に

ね

は

ひ

ふ

へ

ほ

ま

む

め

も

ゆ

よ

ら

り

れ

ろ