Rational®

IBM

機能テストの作成

目次

機能テストの作成 1

概要: 機能テストの作成 1
演習 1: Rational Functional Tester のセットアップ . . 2
ロギングのオプションの設定 2
Functional Tester プロジェクトの作成 2
演習 2: スクリプトの記録 3
記録の開始 3
アプリケーションの開始 4
アクションの記録. 4
演習 3: 検査ポイントの作成 4
データ検査ポイントの作成. 4
イメージ検査ポイントの作成 5
プロパティー検査ポイントの作成 5
パスワード・フィールドのテスト 6
演習 4: スクリプトの再生 7
演習 5: 検査ポイントおよびオブジェクト・マップの表示 8
検査ポイントの表示 8
オブジェクト・マップの表示 9
演習 6: レグレッション・テストの実行 10
演習 7: コンパレーターを使用した、検査ポイントの更新 11
演習 8: オブジェクト・マップの更新. 12
オブジェクト・マップ内のオブジェクト認識プロパティーの表示 13
新しいオブジェクトのマップへの追加 14
オブジェクトの統合 14
再びスクリプトを再生する 15
演習 9: 認識設定の変更 16
演習 10: 正規表現の使用 16
オブジェクト・マップを開いてオブジェクトを統合する 17
プロパティー値の正規表現への変換 17
要約: 機能テストの作成 18

機能テストの作成

この Rational® Functional Tester チュートリアルは、機能テストの作成および再生の主なユースケースを案内します。この包括的なチュートリアルは、製品と共にインストールされるサンプル Java™ アプリケーションを使用します。

学習目標

このチュートリアルを完了すると、以下を実行できるようになります。

- 機能テスト・プロジェクトの作成およびスクリプトの記録
- 検査ポイント、オブジェクト・マップ、および正規表現を使用する作業
- コンパレーターを使用した、検査ポイントの更新
- スクリプトの再生
- レグレッション・テストの実行

必要な時間

45 分

関連情報

 PDF 版の表示

チュートリアル: データ駆動型の機能テストの作成

チュートリアル: キーワードに基づくマニュアル・テストの自動化

サンプル: 機能テスト・プロジェクト

概要: 機能テストの作成

このチュートリアルでは、Functional Tester の使用に関する入門を学習します。また、テストおよび基本操作の実行に関する主なユースケースを体験します。このチュートリアルは、Functional Tester に付属のサンプル・アプリケーションを使用して、すべてのタスクを実行します。

Functional Tester のチュートリアルは、10 の演習に分かれています。チュートリアルを正しく行うためには、これらの演習を順番に完了する必要があります。

学習目標

このチュートリアルを完了後、以下が可能になります。

- 機能テスト・プロジェクトの作成およびスクリプトの記録
- 検査ポイント、オブジェクト・マップ、および正規表現を使用する作業
- コンパレーターを使用した、検査ポイントの更新
- スクリプトの再生
- レグレッション・テストの実行

注: 開始する前にチュートリアルを印刷して、演習を行う際にその印刷物を使用することを検討してください。チュートリアルの PDF 版を印刷するか、または各トピックの内部を右クリックして「**印刷**」をクリックすることにより個々の演習を印刷することができます。

必要な時間

このチュートリアルを完了するには約 45 分かかります。このチュートリアルに関連した他の概念も調べる場合には、完了するまでにさらに多くの時間がかかります。

前提条件

これは入門のチュートリアルです。 Rational Functional Tester を使用した経験がほとんど、またはまったくなくても、これらのタスクを実行できます。

演習 1: Rational Functional Tester のセットアップ

IBM® は、インストール済みで Java アプリケーションのテストに使用できる、 Java ランタイム環境 (JRE) を提供します。この JRE をチュートリアルに使用してください。独自の Java または HTML アプリケーションをテストする場合には、イネーブラーを実行して環境およびアプリケーションを構成する必要があります。これらのセットアップ・タスクについて詳しくは、製品の「ようこそ」の『ファースト・ステップ』セクションにある『Functional Tester 入門ウィザード』を参照してください。ここでは、事前構成された JRE を使用して続行するために何かを行う必要はありません。

Rational Functional Tester を開始してから、最初のテスト・スクリプトを記録する前に、以下のタスクを実行してください。

ロギングのオプションの設定

Rational Functional Tester には、いくつかのロギングのオプションが備わっています。ここでは、HTML ログを使用します。

1. 「**ウィンドウ**」 → 「**パースペクティブを開く**」 → 「**その他**」をクリックして、機能テスト・パースペクティブを開きます。「パースペクティブを開く」ダイアログ・ボックスで、「**Functional Test**」オプションを選択します。
2. HTML ロギングが設定されていることを確認するために、「**ウィンドウ**」 → 「**設定**」をクリックします。
3. 「設定」ウィンドウの左ペインで、「**Functional Test**」を展開してから「**再生**」を展開し、「**ロギング**」をクリックします。
4. 「**デフォルトの使用**」チェック・ボックスが選択済みであること、および「**ログ・タイプ**」フィールドに「**html**」が表示されていることを確認します。
5. 「**OK**」をクリックします。

この設定により、スクリプトの再生後に HTML ログが自動的に開きます。

Functional Tester プロジェクトの作成

記録を開始するには、その前に Functional Tester プロジェクトを作成しておく必要があります。

1. 「Functional Tester」メニューで、「**ファイル**」 → 「**新規**」 → 「**Functional Test プロジェクト**」をクリックします。
2. 「**プロジェクト名**」に、`FTtutorial` とスペースなしで入力します。

3. 「**プロジェクト・パス**」に、C:¥FTproject と入力します。 Functional Tester によってこのディレクトリーが作成されます。
4. ソース管理オプションが使用可能な場合、「**ソース管理へのプロジェクトの追加**」を選択しないでください。
5. プロジェクトの関連付けオプションが使用可能な場合、「**現在の Rational プロジェクトへのプロジェクトの関連付け**」を選択しないでください。
6. 「**終了**」をクリックします。

これで、FTtutorial プロジェクトが、「Functional Test」パースペクティブの左ペインである「Functional Test プロジェクト」ビューに表示されます。

演習 2: スクリプトの記録

この演習では、Functional Tester の記録モニターを使用してスクリプトを記録します。

記録の開始

記録を開始する準備は完了しています。

1. 記録を開始するには、「Functional Test」ツールバーで、「**Functional Test スクリプトの記録**」ボタン ( ) をクリックします。
2. 先程作成した FTtutorial プロジェクトを選択します。
3. 「**スクリプト名**」フィールドに、 Classics (使用するアプリケーションの名前) と入力します。
4. 「**ソース管理へのスクリプトの追加**」オプションが使用可能な場合、それを選択しないでください。
5. 「**終了**」をクリックします。

   「Functional Tester」ウィンドウが自動的に最小化し、記録モニターが表示されます。

   記録モニターについてさらに学習したい方に: Functional Tester の記録モニターは、記録を開始するたびに表示されます。モニターは、画面上に表示したくない場合に最小化することも、およびサイズ変更することもできます。さらに、「**ツールバーのみを表示**」ボタン ( ) をクリックして、記録モニターを非表示にしてツールバーだけを表示することもできます。「**モニターの表示**」ボタン ( ) をクリックすると、モニターが再び表示されます。このチュートリアルの間は、モニターを表示させたままにしてください。モニターには、記録の開始と一時停止、アプリケーションまたはブラウザーの開始、アプリケーション内のクリック、検査ポイントの挿入、その他のアイテムのスクリプト内への挿入など、記録セッション中に実行された各アクションに関するメッセージが表示されます。
6. 「**モニター・メッセージ設定**」ツールバー・ボタン ( ) をクリックします。これらのオプションをいつでも使用して、モニター内のテキストの外観を制御できます。
7. 「**キャンセル**」をクリックします。
8. 「**スクリプト・サポート・コマンドの挿入**」ツールバー・ボタン ( ) をクリックします。

   これにより「スクリプト・サポート機能」ウィンドウが開いて、別のスクリプトの呼び出し、ログ項目の挿入、タイマーの挿入、スリープ・コマンド (遅延) の挿入、またはスクリプト内へのコメントの挿入が可能になります。
9. 「**閉じる**」をクリックします。

アプリケーションの開始

1. テスト・アプリケーションを開始するには、「**アプリケーションの開始**」ツールバー・ボタン ( ) をクリックします。
2. 「アプリケーションの開始」ウィンドウで、「**ClassicsJavaA**」を選択してから、「**OK**」をクリックします。

   Functional Tester チュートリアルのサンプル・アプリケーション ClassicsCD が開きます。記録モニターがアプリケーションの手前にある場合、それを画面の右下の角にドラッグできます。

アクションの記録

これから、このアプリケーションで注文を出す操作を記録します。

1. 「**Haydn**」の横にある「**+**」をクリックして、「**Composers**」ツリー内のフォルダーを展開します。
2. リスト内で、「**Symphonies Nos. 94 & 98**」をクリックします。
3. 「**Place Order**」ボタンをクリックします。
4. 「Member Logon」ウィンドウで、既存の設定値の「**Existing Customer**」および「**Trent Culpito**」をそのままにします。この時点では、どちらのパスワード・フィールドもクリックしないでください。
5. 「**OK**」をクリックします。
6. 「**card number**」フィールドに、クレジット・カード番号を入力します。ここでは、有効な形式として、4 桁の数字を 4 セット使用する必要があります (`7777 7777 7777 7777` など)。
7. 「**expiration date**」フィールドに、有効な形式の有効期限である `07/07` を入力します。
8. 「**Place Order**」をクリックします。
9. 注文を確認するメッセージ・ウィンドウで、「**OK**」をクリックします。

演習 3: 検査ポイントの作成

この演習では、オブジェクトをテストするための検査ポイントを記録します。検査ポイントは、特定のアクションが実行されたことを確認したりあるオブジェクトの状態を調べたりするためのものです。

プロパティー検査ポイント、イメージ検査ポイント、または 6 種類のデータ検査ポイントを作成できます。検査ポイントを作成すると、アプリケーション内のオブジェクトに関する情報を収集し、再生時の比較操作で使用するベースライン情報を確立することができます。

データ検査ポイントの作成

データ検査ポイントを記録して、作曲者のツリーをキャプチャーします。

1. 記録モニターで、「**検査ポイント・コマンドまたはアクション・コマンドの挿入**」ボタン ( ) をクリックします。
2. 検査ポイントおよびアクション・ウィザードの「オブジェクトの選択」ページで、「**オブジェクトを選択した後、次のページに進む**」オプションが選択されていればそれをクリアします。
3. オブジェクト・ファインダー ( ) を使用して、アプリケーション内の「Composers」ツリーを選択します。「**オブジェクト・ファインダー**」をクリックして、それをツリーの上にドラッグします。マウス・ボタンを押したままにすると、ツリー全体が赤色の枠で縁取られて、オブジェクト名 (`javax.swing.JTree`) が赤色の枠の横の画面ヒントに表示されます。マウス・ボタンを解放して選択を実行すると、オブジェクトに対する認識プロパティーが「オブジェクトの選択」ページの下部にあるグリッド内にリストされることに注目してください。

4. 「**次へ**」をクリックします。
5. 「アクションの選択」ページで、「**データ検査ポイントの実行**」が選択されていることを確認してから「**次へ**」をクリックします。
6. 「検査ポイント・データ・コマンドの挿入」ページの「**データ値**」フィールドで、「**ツリー階層**」テストを選択します。 このテストは、ツリー階層全体についての情報をキャプチャーします。
7. 「検査ポイント名」フィールドで、 `Classics_tree` と入力して「**次へ**」をクリックします。
8. 「検査ポイント・データ」ページは、キャプチャーしたデータを右側ペインのグリッドに表示します。項目の横にあるボックスにチェック・マークが表示されている場合、その項目がテスト対象となります。デフォルトでは、すべての項目が選択されています。それらをチェックされたままにしてください。それらが選択されていない場合、「**すべてチェック**」ボタンをクリックします。
9. 「**終了**」をクリックします。

イメージ検査ポイントの作成

イメージ検査ポイントを挿入して、正しいアルバムが選択した CD 用に表示されていることを確認できます。

1. 記録モニターで、「**検査ポイント・コマンドまたはアクション・コマンドの挿入**」ボタン () をクリックします。
2. 検査ポイントおよびアクション・ウィザードの「オブジェクトの選択」ページで、「**オブジェクトを選択した後、次のページに進む**」オプションが選択されていればそれをクリアします。
3. オブジェクト・ファインダー () を使用して、アプリケーション内のアルバム・イメージを選択します。「**オブジェクト・ファインダー**」をクリックして、それをアルバム・イメージの上にドラッグします。マウス・ボタンを押したままにすると、アルバム・イメージが赤色の枠で縁取られて、オブジェクト名 (`javax.swing.JLabel`) が赤色の枠の横の画面ヒントに表示されます。マウス・ボタンを解放して選択を実行すると、オブジェクトに対する認識プロパティーが「オブジェクトの選択」ページの下部にあるグリッド内にリストされることに注目してください。
4. 「**次へ**」をクリックします。
5. 「アクションの選択」ページで、「**イメージ検査ポイントの実行**」を選択し、「**次へ**」をクリックします。
6. 「イメージ検査ポイント・コマンドの挿入」ページで、「**検査ポイント名**」として `Album_image` と入力します。
7. オプション「**フル・イメージの選択**」が選択済みであることを確認して、「**次へ**」をクリックします。
8. 「検査ポイント・データ」ページは、キャプチャーしたイメージを右側ペインに表示します。「**終了**」をクリックします。

プロパティー検査ポイントの作成

ここで、異なる検査ポイントを挿入して注文が正しい顧客に対するものであることを確認できます。プロパティー検査ポイントは、確認画面のテキストをキャプチャーします。

1. ClassicsCD アプリケーションで、「**Order**」 → 「**View Existing Order Status**」をクリックします。この時点では、どちらのパスワード・フィールドもクリックしないでください。
2. 「**OK**」をクリックします。 「View Existing Orders」ウィンドウで、ラベル「Order for Trent Culpito」をテストします。
3. 記録モニターで、「**検査ポイント・コマンドまたはアクション・コマンドの挿入**」ボタン () をクリックします。

4. 「オブジェクトの選択」ページで、「**オブジェクトを選択した後、次のページに進む**」オプションを選択します。
5. 「**オブジェクト・ファインダー**」をラベル「Order for Trent Culpito」の上にドラッグして、それを選択します。マウス・ボタンを押したままにすると、ラベルが赤色の枠で縁取られて、オブジェクト名 (`javax.swing.JLabel`) が表示されることに注目してください。 「**次のページに進む**」オプションを選択したので、オブジェクトを選択した後は「アクションの選択」ページが開きます。
6. 上から 2 番目のアクションである「**プロパティー検査ポイントの実行**」を選択して、「**次へ**」をクリックします。
7. 「プロパティー検査ポイント・コマンドの挿入」ページで、「**子を含める**」フィールドが「**なし**」に設定されていることを確認します。
8. 「**検査ポイント名**」の下で、推奨されたデフォルトを受け入れます。
9. 「**標準プロパティー (すべてのプラットフォームで使用可能なプロパティー) を使用する**」を選択されたままにして、「**次へ**」をクリックします。 「検査ポイント・データ」ページに、テスト・オブジェクト・プロパティーおよびその値がグリッド形式で表示されます。「プロパティー」列でテストするプロパティーを選択して、「値」列でプロパティー値を編集できます。

   オブジェクト・プロパティーの選択についてさらに学習したい方に: デフォルトでは、どのプロパティーも選択されていません。オブジェクト・プロパティーをテストするには、各プロパティーを選択することによってテストするプロパティーを選択します。選択したプロパティーは、スクリプトを再生するたびにこの検査ポイントでテストされます。グリッドの上にある「**すべてチェック**」ツールバー・ボタンをクリックして、リスト内のすべてのプロパティーを選択することもできます。すべてのプロパティーをクリアするには、「**すべてチェック解除**」ボタンを使用します。プロパティー検査ポイントを使用して最適な結果を得るには、関心のあるプロパティーだけをテストしてください。この例では、注文が正しい顧客のものかどうかを判別するために、「**text**」プロパティーだけが関心の対象となります。
10. 「プロパティー」列で、「**text**」、「**opaque**」、および「**visible**」プロパティーだけを選択して、再生の際にテストするようにします。選択を持続させるために、チェック・ボックスを 2 度クリックしなければならないことがあります。
11. 「**終了**」をクリックします。
12. ClassicsCD の「View Existing Orders」ウィンドウで、「**Close**」をクリックします。

パスワード・フィールドのテスト

ここで別の簡単な注文を出して、以前にテストしていないパスワード・フィールドをテストします。

1. 「composers」ツリーで、「**Haydn**」フォルダーを展開します。
2. 「**Symphonies Nos. 94 & 98**」をクリックします。
3. 「**Place Order**」ボタンをクリックします。
4. 「Member Logon」ウィンドウで、既存の設定値の「**Existing Customer**」および「**Trent Culpito**」をそのままにします。
5. 今回は、「**Password**」フィールドに `xxxx` と入力します。
6. 「**Remember Password**」オプションを選択します。
7. 「**OK**」をクリックします。
8. 有効な「**card number**」番号および「**expiration date**」を入力します。例えば、 `7777 7777 7777 7777`、有効期限を `07/07` のようにします。
9. 「**Place Order**」をクリックします。
10. 注文を確認するメッセージ・ボックスで、「**OK**」をクリックします。

11. 「**x**」ボタンをクリックして、ClassicsCD アプリケーションを閉じます。
12. 「記録中」ツールバーの「**記録の停止**」ボタン ( ) をクリックします。

記録を停止すると、Rational Functional Tester は記録モニターを閉じて、スクリプトとオブジェクト・マップをプロジェクト・ディレクトリーに書き込みます。 Rational の「Functional Test」ウィンドウが復元されて、メイン・ウィンドウにスクリプトが表示されます。

演習 4: スクリプトの再生

この演習では、スクリプトを再生して、 Rational Functional Tester インターフェースのいくつかの部分に注目します。直前に記録したスクリプトはアクティブ・スクリプトなので、再生ボタンをクリックするとそのスクリプトが再生されます。

1. スクリプトを再生するには、Functional Test ツールバーで、「**Functional Test スクリプトの実行**」ボタン ( ) をクリックします。
2. 「ログの選択」ウィンドウで、ログ名をデフォルトの「**Classics**」のままにして、「**終了**」をクリックします。

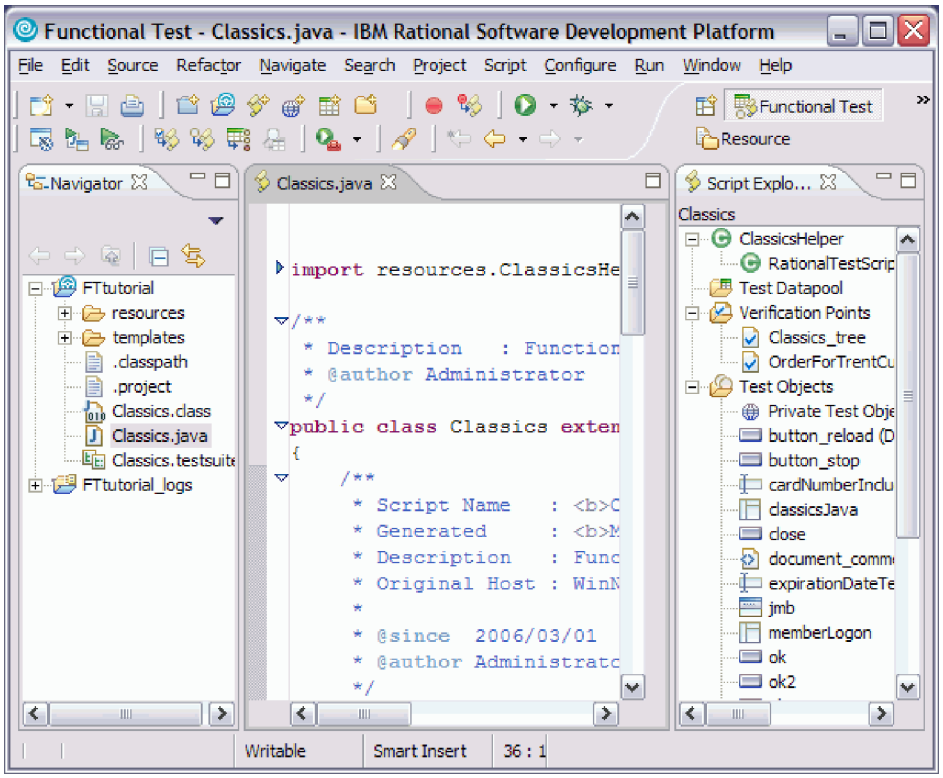
   Rational Functional Tester が最小化され、画面の右上で再生モニターが始動します。スクリプトの再生中に、再生モニターにメッセージが表示されます。 Rational Functional Tester は、アプリケーションの開始、アプリケーション上で実行したアクション、検査ポイントなどの、記録されたアクションをすべて再生します。

   再生が完了すると、別のウィンドウでテスト実行の結果が HTML ログに表示されます。ログにリストされた各イベントには、イベント見出しに緑色で「合格」が含まれているはずです。記録した 2 つの検査ポイントがリストされていることに注目してください。
3. ログを閉じます。 これでスクリプトの記録および再生が正常に実行できました。ここで、 Functional Test パースペクティブをより詳細に観察してみましょう。
4. 「Functional Test」ウィンドウが最小化されている場合には、それを復元してください。 複数のスクリプトがあるとき、Functional Tester はプロジェクト内のすべての開いているスクリプトを Java エディター (スクリプト・ウィンドウ) に表示します。

   Java エディターについてさらに学習したい方に: スクリプト全体で、スクリプトについての情報が上部に明るい青色で先頭にアスタリスクが付いて表示されていることに注目してください。この情報は、変更可能なスクリプト・テンプレートからのものです。スクリプト・テンプレートの変更について詳しくは、Functional Tester のヘルプを参照してください。

   Functional Tester は、続く行が参照するオブジェクトを識別するための短いコメントを緑色の文字でスクリプトに追加することに注目してください。この情報により、スクリプトのナビゲートが容易になります。ユーザー入力など、記録中に引数としてメソッドに渡されるストリングは、明るい青色で表示されます。

   カーソルをスクリプトの特定の箇所の上に移動すると、 Functional Tester はポップアップ・テキスト・ボックス内に有用な情報を表示します。例えば、ヘルパー・メソッドに関しては、オブジェクト・マップ内の記述プロパティー・セット、およびその後にオブジェクトの認識プロパティーが表示されます。吹き出し機能は、「設定」によって制御されます。この機能をオフにするか、表示内容を変更するには、「**ウィンドウ**」 → 「**設定**」をクリックしてから、「**Java**」 → 「**エディター**」を選択して、「**吹き出し**」タブをクリックします。吹き出し機能は、デフォルトではオンになっています。



Java エディター (スクリプト・ウィンドウ) の左には「Functional Test プロジェクト」ビューがあり、ここには現在接続している Functional Tester プロジェクトがリストされています。各プロジェクト内のすべてのスクリプトは、プロジェクト名の下にリストされています。この「プロジェクト」ビューは、異なるスクリプトにナビゲートするための別の方法を提供します。「プロジェクト」ビュー内のスクリプトをダブルクリックすると、そのスクリプトがスクリプト・ウィンドウ内で開いてアクティブ・スクリプトになります。

Java エディターの右側には、アクティブ・スクリプトの検査ポイントおよびオブジェクト・マップをリストする、スクリプト・エクスプローラーがあります。スクリプト・エクスプローラーから、検査ポイント・エディターを開始して検査ポイントを表示および編集すること、およびオブジェクト・マップ・エディターを開始してオブジェクト・マップを表示および編集することができます。スクリプト・エクスプローラー、または「タスク」ビューや「コンソール」ビューなどの Functional Test パースペクティブの他の部分について詳しくは、 Functional Tester のヘルプを参照してください。

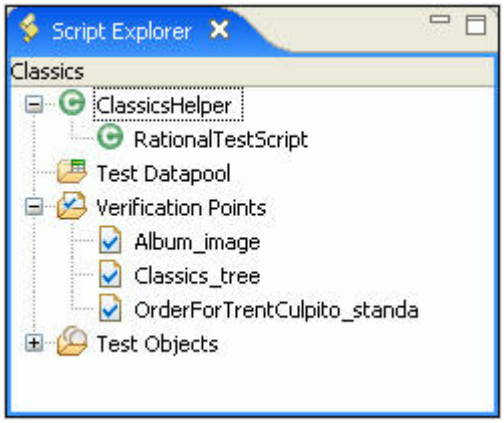
演習 5: 検査ポイントおよびオブジェクト・マップの表示

この演習では、検査ポイントおよびオブジェクト・マップのプロパティーを表示および変更する方法について学習します。

検査ポイントの表示

検査ポイント内のデータを検討して変更することができます。

1. Rational Functional Tester で、スクリプト Classics.java が Java エディター内で引き続きアクティブ・スクリプトであることを確認します。
2. 記録した 3 つの検査ポイントが、スクリプトの右側にあるスクリプト・エクスプローラー内にリストされているはずです。必要であれば、検査ポイントの横にあるプラス記号 (+) をクリックしてリストを展開してください。



3. 「**Classics_tree**」をダブルクリックします。

   これは作曲家のリストについて、記録した最初の検査ポイントです。検査ポイント・エディターが始動します。後で再生するために、検査ポイント・データを更新することができます。

   検査ポイントの更新: データ検査ポイントには、可能な 6 つの表示タイプがあります。これはデータ (ツリー) 検査ポイントです。この例では、オブジェクト型はツリーで、javax.swing.JTree となります。このツリー内のデータを編集するには、ツリー内のいずれかの副項目をダブルクリックして小さな編集ボックスを開き、そこで変更を行うことができます。各項目の横にあるチェック・ボックスを使用して、その項目を後の再生時にテストするかどうかを指定します。検査ポイント・エディターの使用方法について詳しくは、 Functional Tester のヘルプを参照してください。
4. 検査ポイント・エディターを閉じます。

オブジェクト・マップの表示

オブジェクト・マップ内のデータを検討して変更することができます。

1. スクリプト・エクスプローラーで、「**テスト・オブジェクト**」フォルダーを展開します。

   最初の項目の「専用テスト・オブジェクト・マップ」が、このスクリプトのオブジェクト・マップです。「専用テスト・オブジェクト・マップ」の下にリストされた個別のオブジェクトは、記録の際に処理対象となったオブジェクトへの参照です。
2. 「**専用テスト・オブジェクト・マップ**」( ) をダブルクリックして開きます。

   オブジェクト・マップ・タイプ: スクリプトを記録すると、テスト対象アプリケーションのオブジェクト・マップが Functional Tester により作成されます。各スクリプトごとに、1 つのオブジェクト・マップ・ファイルが関連付けられています。マップ・ファイルは、専用にする (1 つのスクリプトに排他的に関連付ける) ことも、多数のスクリプトで共用することもできます。スクリプトを記録したとき、Rational Functional Tester はデフォルトの設定 (専用マップ) を使用しました。オブジェクト・マップには各オブジェクトのプロパティーが含まれているため、 1 つの場所での集中管理により情報の更新が容易になります。その後、そのオブジェクトを参照するスクリプトがある場合、どのスクリプトも更新後の情報を共用できます。
3. トップレベルのオブジェクト `Java: Frame: logFrame1: javax.swing.JFrame` を展開します。

   フレーム・オブジェクトには、ログオン・ダイアログ・ボックスが含まれます。ラジオ・ボタン、パスワード・フィールド、およびアクション・ボタンは、フレーム・オブジェクトの下にリストされています。
4. オブジェクトの 1 つをクリックします。

認識プロパティーが、オブジェクト・ツリーの下のグリッドに表示されていることに注目してください。さらに、オブジェクト・マップは、簡単な操作でスクリプトにオブジェクト参照を追加する手段となります。「オブジェクト・マップ」メニューで、**「テスト・オブジェクト」 → 「オブジェクトの挿入」**をクリックして、オブジェクトを挿入できます。また、認識プロパティーの重みの変更や認識プロパティーと値の編集など、他の操作をオブジェクト・マップから行うこともできます。チュートリアルでは後に、オブジェクト・マップを使用するいくつかの高度な手順を実行します。

5. 「オブジェクト・マップ」メニューで、**「設定」 → 「クローズ時に状態をクリア」**をクリックします。

   「クローズ時に状態をクリア」コマンドはトグル・メニュー項目であり、デフォルトではオンになっているので、それをクリアします。これをオンのままにした場合、マップを閉じるときにすべてのオブジェクトが受け入れられることになります。この操作は、後のステップでオブジェクト・マップに戻って変更を加えるときに行います。

6. オブジェクト・マップを閉じます。加えた変更は保管しないでください。

演習 6: レグレッション・テストの実行

この演習では、スクリプトを異なるビルド上で実行します。アプリケーションの新しいビルドがあるとき、スクリプトを新規のビルド上で再生することにより、記録した自動化されたテストを実行できます。スクリプトを新規のビルド上で実行するには、スクリプト内のアプリケーション名を変更する必要があります。(これは開発プロジェクトでは行う必要がありません。これをここで行うのは、アプリケーションの新規ビルドの取得をシミュレートするためです。)

1. Java エディター (スクリプト・ウィンドウ) で、スクリプト (Classics.java) がアクティブ・スクリプトであることを確認します。

   スクリプトの上部の、テンプレート情報の下にある、次のアプリケーション開始コマンドに注目してください。

   ```
   startApp("ClassicsJavaA");
   ```

2. 「A」を「B」に変更します。

   Java コードは大/小文字を区別するので、必ず大文字の B を使用してください。変更を有効にするために、スクリプトを保管またはコンパイルする必要はありません。それはスクリプトの実行時に自動的に行われます。

3. **「Functional Test スクリプトの実行」**ツールバー・ボタン ( ) をクリックして、スクリプトを再生します。

4. 「ログの選択」ウィンドウで、**「Classics」**を選択してから**「終了」**をクリックします。 ログの上書きを確認するプロンプトが表示されます。

5. **「はい」**をクリックします。

   スクリプトは高速で再生を開始しますが、終わり近くの「Member Logon」ウィンドウでは低速になります。これは、アプリケーションのビルド B ではチェック・ボックスの横のフィールドに異なるテキストがあるためです。 Functional Tester は、ビルド A で記録された認識プロパティーと一致するオブジェクトを検索しています。この問題を解決する方法は、チュートリアルの後の段階で示します。

6. 再生後にログが開いたら、メッセージを確認します。ログ内に 2 つの失敗と 1 つの警告が含まれているはずです。 (演習 7 のための準備として、ログは開いたままにしておきます。)

   アプリケーション内の変更のために、プロパティー検査ポイント (OrderForTrentCulpito_standa) およびイメージ検査ポイント (Album_image) が失敗しています。次に、これを修正するために検査ポイント・ベースラインを更新する方法を学習します。パスワード・チェック・ボックス・フィールドに対してオ

ブジェクト認識警告が生成されました。チュートリアルの後のセクションでは、正規表現を使用してオブジェクト・マップ内でこの問題を修正する方法についても示します。

ClassicsB の主画面が ClassicsA とは異なることに注目してください。ただし、この相異によってスクリプトが失敗することはありません。同じオブジェクトが存在していますが、それらは 2 つのアプリケーション上の異なるロケーションにあります。 Functional Tester は強力な認識方法を使用してオブジェクトを見つけるので、これが原因で失敗することはありません。例えば、オブジェクトを検索するときに画面座標などの表面的なプロパティーに依存してはいません。その代わりに、内部認識プロパティーを使用します。この方法により、スクリプトの変更や再記録を必要としない、柔軟なユーザー・インターフェース設計が可能になります。

演習 7: コンパレーターを使用した、検査ポイントの更新

検査ポイント・コンパレーターを使用して、スクリプトの再生後に検査ポイント・データを比較できます。検査ポイントは、プロパティーのベースラインまたはオブジェクトのデータを提供します。アプリケーションの後続のビルド上で検査ポイントが失敗する場合、アプリケーションに欠陥または意図的な変更があることが判明します。変更が意図的なものである場合、将来のビルドでもテストが有効になるように検査ポイントの情報を更新できます。

演習 6 の終了時に、ログを開いたままにしておきました。ログを閉じた場合には、「プロジェクト」ビューでログ名をダブルクリックして、それを再び開いてください。

1. ログ内で、失敗したイメージ検査ポイント項目の末尾にある「**結果の表示**」リンクをクリックします。イベント見出しは、「Verification Point (Album_image)」です。

   Functional Tester の検査ポイント・コンパレーターは、検査ポイント・データを表示します。コンパレーター・バナーには、検査ポイントの名前が含まれることに注目してください。

   コンパレーターに問題が生じた場合: コンパレーターが開かないか、またはエラー・メッセージが出される場合、ブラウザーの Java プラグインを使用可能にする必要があります。その方法については、Functional Tester のヘルプにある『記録する前に』セクションの、『ブラウザーの Java プラグインの使用可能化』というトピックを参照してください。

   検査ポイントが失敗すると、コンパレーターは差異の分析に役立つように予期される値および実際の値を表示します。その後、ベースライン・ファイルをロードして編集すること、または実際のファイルの値を使用してそれを更新することができます。失敗は赤色で表示されます。

   ClassicsA 上に検査ポイントを作成したとき、キャプチャーされたアルバム・イメージはオブジェクト `javax.swing.JLabel` に基づいています。 ClassicsB でスクリプトを再生したときには、オブジェクト `javax.swing.JLabel` の高さと幅が異なるため、イメージ検査ポイントは失敗しました。そのため、ベースライン・ファイルを更新して ClassicsB と一致するようにオブジェクトを更新する必要があります。
2. 「コンパレーター」ツールバー上の「**編集するベースラインのロード**」ボタン ( ) をクリックします。
3. 「コンパレーター」ツールバー上の「**ベースラインから実際の値への置き換え**」ボタン ( ) をクリックします。 実際のイメージがベースライン・イメージとしてロードされます。
4. コンパレーターを閉じます。
5. ログ内で、失敗したプロパティー検査ポイント項目の末尾にある「**結果の表示**」リンクをクリックします。イベント見出しは、「Verification Point (OrderforTrentCulpito_standard)」です。
6. 「**text**」プロパティーまでスクロールします。

ClassicsA 上に検査ポイントを作成したとき、バナー・タイトルは「Order for Trent Culpito」でした。ClassicsB 上でスクリプトを再生したとき、バナー・タイトルは「Orders for Trent Culpito」でした。顧客は「Orders」ウィンドウで複数の注文を行う場合があるので、正しいのは「Orders」のほうです。そのため、ベースライン・ファイルを更新して ClassicsB と一致するようにテキストを更新する必要があります。

ベースライン・ファイルだけを編集できます。

7. 「コンパレーター」ツールバー上の「**編集するベースラインのロード**」ボタン ( ) をクリックします。 左の「**値**」列には、現在「**ベースライン値**」が表示されていることに注目してください。
8. 「**text**」プロパティーにスクロールする代わりに、「プロパティー」列の上にある「**最初の相違にジャンプします**」ボタン ( ) をクリックできます。 4 つのナビゲーション・ボタンは、ベースライン・ファイルと実際のファイルとの相異を見つけるために役立ちます。

   ベースライン・ファイルは、2 つの方法で更新できます。グリッドのそのセルを編集して文字「s」をワード「Order」に追加するか、またはベースラインの置き換えコマンドを使用できます。ベースラインを置換すると、ベースライン・ファイルのすべての値が実際のファイルの値に置き換えられます。一般に、1 つまたは少数の値だけを編集するときには、個別の値を編集してください。
9. このテストでは更新する差異が 1 つだけなので、「コンパレーター」ツールバー上の「**ベースラインから実際の値への置き換え**」ボタン ( ) をクリックします。 「**text**」プロパティー内の両方の値はこれで一致して、プロパティーは赤色で表示されなくなります。コンパレーターの使用方法について詳しくは、Functional Tester のヘルプを参照してください。
10. コンパレーターを閉じます。

    ここでスクリプトを再び再生して、失敗に対する更新済みのベースライン値を指定した場合に、検査ポイントが合格することを確認します。
11. ログを閉じます。
12. 「Functional Tester」ツールバーにある「**Functional Test スクリプトの実行**」ボタンをクリックします。
13. 「**Classics**」ログを選択してから、「**終了**」をクリックします。
14. ログの上書きを確認するプロンプトが表示された場合、「**はい**」をクリックします。

    この認識に関する問題をまだ修正していないので、Functional Tester は「Member Logon」ウィンドウで一時停止します。再生の最後に、Functional Tester はログを表示します。これで検査ポイントは合格になります。コンパレーターを使用して、テスト対象のアプリケーションの変更に対応するためにオブジェクト・データおよびプロパティーを更新することは非常に簡単です。
15. ログは開いたままにしてください。

演習 8: オブジェクト・マップの更新

この演習では、オブジェクト・マップを使用してオブジェクト認識に関する警告を修正します。また、より柔軟なオブジェクト認識のために正規表現を使用します。

認識の失敗または警告が生じたときは、ログ・メッセージを参照してください。演習 7 の終了時に、ログを開いたままにしておきました。開いていない場合は、「プロジェクト」ビューでログをダブルクリックして開きます。 1 つの警告がログ内に残っています。イベント見出しは、「**オブジェクト認識が弱いです (警告しきい値を超えています)**」です。

1. ログの末尾に近い警告セクションで、「**ObjectLookedFor**」および「**objectFound**」フィールドを参照します。

   ClassicsA で、パスワード・フィールドの名前は「**Remember Password**」です。 ClassicsB では、「**Remember The Password**」です。 ClassicsB 上でスクリプトを再生したとき、この相異のためにオブジェクト認識は完全に一致しませんでした。
2. ログ内の「**行番号**」フィールドを参照します。その番号を記録して、ログを閉じ、Functional Tester に戻ります。
3. スクリプト・ウィンドウ内の任意の箇所をクリックしてから、「**ナビゲート**」 → 「**指定行へジャンプ**」をクリックします。
4. ログ失敗メッセージの行番号を入力してから、「**OK**」をクリックします。

   カーソルが、その行番号の左マージンに移動します。

   注: 「Functional Tester」ウィンドウの下部にあるインディケーターを参照して、行番号を知ることもできます。例えば、「43:9」は行 43 の位置 9 を示します。
   スクリプト内の行は、次のようになっています。

   ```
   RememberPassword().clickToState(SELECTED);
   ```

   この行は、パスワード・チェック・ボックス上のクリックを表します。スクリプト内のこの行は、どのオブジェクトが失敗しているかを示します。これで、オブジェクト・マップ内でそのオブジェクトを探すことができます。
5. オブジェクトを検索するには、スクリプト・エクスプローラー (右ペイン) 内でテスト・オブジェクトのリストに戻ります。 「**テスト・オブジェクト**」フォルダーの下に「rememberPassword」がリストされていることが分かります。

オブジェクト・マップ内のオブジェクト認識プロパティーの表示

1. 「**rememberPassword**」オブジェクトをダブルクリックして、オブジェクト・マップ内でそれを開きます。
2. オブジェクト・マップ・メニューで、「**テスト・オブジェクト**」 → 「**すべてを受け入れる**」をクリックします。コマンドがグレー表示になっている場合、何も行わないでください。

   すべてのオブジェクトが黒のテキストに変わることに注目してください。テキストは、マップ内のオブジェクトを受け入れるまで青色 (新規のオブジェクトを示すため) です。新規に作成されたオブジェクト・マップを初めて見るときに、オブジェクトを受け入れてください。
3. パスワード・チェック・ボックス・オブジェクトがマップ内で選択されていない場合、それを選択します。 (それは、**Java: checkBox: checkRemember: javax.swing.JCheckBox** というオブジェクトです。)
4. オブジェクト・マップの下部にある、「**認識**」タブにリストされた認識プロパティーを参照してください。

   「**text**」プロパティーに 「Remember Password」とあるので、これは ClassicsA のオブジェクトであることが分かります。これは「古い」オブジェクトです。ただし、ClassicsB でスクリプトを再生したときに、オブジェクトのテキストが変更されたので、 Functional Tester はそれを「新しい」オブジェクトとして認識します。この例では新しいオブジェクトのプロパティーを使用するので、それをマップに追加する必要があります。

新しいオブジェクトのマップへの追加

新しいオブジェクトをマップに追加するには、 ClassicsB を開いてから、「Member Logon」ウィンドウを開きます。

1. オブジェクト・マップ・メニューで、「**アプリケーション**」 → 「**実行**」をクリックします。
2. 「**ClassicsJavaB**」 を選択します。(必ず B を選ぶようにします)
3. 「**OK**」をクリックします。
4. ClassicsCD で、任意の CD を選択してから「**Place Order**」をクリックします。

   「Member Logon」ウィンドウが開きます。
5. 必要であれば、全体が見えるように、オブジェクト・マップを画面の低い位置に移動します。「オブジェクト・マップ」メニューで、「**テスト・オブジェクト**」 → 「**オブジェクトの挿入**」をクリックします。

   これは検査ポイント・ウィザードの「オブジェクトの選択」ページにある「オブジェクト・ファインダー」ツールと同じです。
6. 「**オブジェクトを選択した後、次のページに進む**」チェック・ボックスが選択されている場合、それをクリアします。
7. 「オブジェクト・ファインダー」ツールを使用して、「Member Logon」ウィンドウで「**Remember the Password**」チェック・ボックスを選択します。

   チェック・ボックスを選択した後に、「**text**」 プロパティーが `Remember The Password` になっていることが分かります。必要であれば、オブジェクト・マップの枠を拡大してプロパティーを表示してください。
8. 「オブジェクトの選択」ページで、「**次へ**」をクリックします。
9. 「オブジェクトの選択オプション」ページ上では何も変更しないで、「**終了**」をクリックします。

   これで、新しいチェック・ボックス・オブジェクトがオブジェクト・マップに表示されるようになりました。
10. 別のオブジェクトをクリックして、新しい項目が青色でリストされていること、および「新規」という語が行の先頭に表示されていることに注目してください。

    これで、古いオブジェクトと新しいオブジェクトの両方がマップにリストされるようになりました。次に、2 つのオブジェクトを統合して、新規オブジェクトのためのプロパティーをそれぞれから取得できるようにします。

オブジェクトの統合

1. オブジェクトを統合するには、古いオブジェクト (**CheckBox: checkRemember** というラベルの元のチェック・ボックス) をクリックしてから、それをリスト内の新しいオブジェクトの上にドラッグします。マウス・ボタンを離す前に、カーソル矢印の先端を新しいオブジェクトに置きます。その後、マウス・ボタンを離してください。

   「テスト・オブジェクトの統合」ウィザードが開きます。
2. 必要であれば「統合」ウィザードを拡大して、下の方のセクションにある他の情報を表示します。

左下のセクションに、元のオブジェクトのプロパティーが表示されています。そのラベルは、「Source: RememberPassword」です。これが ClassicsA でのチェック・ボックスのテキストでした。右下のセクションでは、「Target: RememberThePassword」というラベルが付いています。これが ClassicsB でのチェック・ボックスのテキストです。

古いオブジェクトを新しいオブジェクトにドラッグしたので、新しいオブジェクトの認識プロパティーがウィザードの最上部に入ります。一般に、新規プロパティーが優先されるプロパティーであれば、Functional Tester はそれを最上部に配置します。ただし、いくつかの古い管理プロパティーが優先される場合もあります。例えば、Functional Tester は古いプロパティー・セットの正規表現を保持します。古いオブジェクトからのプロパティーを使用するには、古いオブジェクトのグリッド内でそのプロパティーをダブルクリックすれば、それは統合されるオブジェクト内にコピーされます。この例では、すでに入っている、新しいオブジェクトのすべてのプロパティーを使用します。

3. 「**次へ**」をクリックします。

   オブジェクト・マップ内のこの変更により影響を受けるすべてのスクリプトがリストされます。 1 つのスクリプト、「Classics」だけが影響を受けます。

4. 「**終了**」をクリックします。
5. オブジェクト・マップで、オブジェクト・マップ・ツールバー上の「**ファイル**」 → 「**保管**」メニューをクリックして、加えた変更を保管してからオブジェクト・マップを閉じます。

再びスクリプトを再生する

ここで ClassicsB に対してスクリプトを再び再生して、それが合格することを確認します。

1. ClassicsCD の両方のダイアログ・ボックスを閉じます。
2. Functional Tester で、ツールバー上の「**Functional Test スクリプトの実行**」をクリックします。
3. 「**Classics**」ログを選択してから、「**終了**」をクリックします。

   これでスクリプトは警告を出さずに成功します。今回は認識プロパティーが一致するために、パスワード・チェック・ボックス・オブジェクトで再生が一時停止しなくなったことに注目してください。

   このオブジェクト統合機能は、オブジェクトの認識プロパティーが意図的に変更されたときに、スクリプトを更新するための簡単な方法となります。この機能の主な利点の 1 つは、オブジェクト・マップが多数のスクリプトによって使用されている場合に、ウィザードで変更を行うことによってそれらのスクリプトすべてを更新できることです。複数のスクリプトを手作業で編集する代わりに、マップ内で一度だけ変更を行うことにより、変更内容はそのマップを使用するすべてのスクリプトに自動的に伝わります。この機能により、時間を節約できます。

   認識プロパティーを更新する別の方法: テスト・オブジェクトが変更された場合に、その認識プロパティーを更新するより簡単な方法もあります。この課題で説明された「統合」ウィザードを使用する代わりに、オブジェクト・マップから、更新する認識プロパティーを持つテスト・オブジェクトを選択できます。オブジェクト・マップ・ツリーに表示されたテスト・オブジェクトを右クリックして、ポップアップ・メニューから「**認識プロパティーの更新**」を選択します。このアクションを行う際には、Functional Tester が更新された認識プロパティーを取得できるように、テスト・アプリケーションを実行している必要があります。この更新方法を使用するのは、古いオブジェクトのどのプロパティーも使用しない場合だけです。

4. ログを閉じます。

演習 9: 認識設定の変更

直前の演習では、オブジェクトが変更されたときにその認識プロパティーを更新する方法について学習しました。変更できる別の要素は、Functional Tester が再生の際に使用する認識の加重です。これは、ScriptAssure™ 認識設定を使用して設定します。 2 番目の検査ポイントでテストしたラベル・オブジェクトで、この仕組みを例示できます。

1. 「Functional Tester」メニューで、「**ウィンドウ**」 → 「**設定**」をクリックします。
2. 「**Functional Test**」 → 「**再生**」 → 「**ScriptAssure**」をクリックします。
3. 「**拡張**」ボタンをクリックします。

   デフォルト設定の 1 つが、「**許容スコアがこの数値を超える場合、警告を出す: 10000**」であることに注目してください。スコアの 10000 は、重要なプロパティーの 1 つが誤っている可能性があることを示しています。スコアを 5000 に下げて、その結果を見てみましょう。
4. このフィールドの横にある、「**デフォルトの使用**」チェック・ボックスを選択します。
5. その後、フィールドに `5000` と入力して、「**OK**」をクリックします。
6. ClassicsB に対してスクリプトを再び再生します。

   今回は、ログにはラベル・オブジェクトに関する警告が含まれています。「**objectFound**」フィールドで示されている理由は、認識スコアが 10000 であるというものです。この矛盾は、ラベルで「Order」という語を「Orders」に変更したために生じました。
7. ログを閉じます。
8. 以下のようにして、認識スコアのデフォルト値を復元します。
   a. 「**ウィンドウ**」 → 「**設定**」をクリックします。
   b. 「**Functional Test**」 → 「**再生**」 → 「**ScriptAssure**」をクリックします。
   c. 「**拡張**」ボタンをクリックします。
   d. 「**許容スコアがこの数値を超える場合、警告を出す**」フィールドの横にある、「**デフォルトの使用**」チェック・ボックスを選択します。

      これにより、5000 が 10000 に戻ります。
   e. 「**OK**」をクリックします。
   f. スクリプトを再び再生します。

      今回は警告が出されないで、すべてが成功します。
   g. ログを閉じます。

この演習では、認識スコアを微調整することによってオブジェクト認識の望ましい感度を実現する方法を示しました。 ScriptAssure の使用方法について詳しくは、 Functional Tester のヘルプを参照してください。

演習 10: 正規表現の使用

オブジェクト・マップを使用して行う最後の事柄として、プロパティー値を正規表現に変換します。この例では、正規表現を使用するとオブジェクト認識をより柔軟に行うことができます。

直前の演習では、ClassicsB でスクリプトが完全に成功する様子を学習しました。 ClassicsB でアプリケーションに対して加えられた変更は正しいものなので、それが目標でした。そのため、現在のスクリプトはさらに続行するのに適切な状態にあります。ここで、スクリプトを ClassicsA に対して再生すると、以前に加えられた変更のために再生は失敗します。成功させるために、オブジェクトの複数のバリアントを許可する

ことができます。ダイナミック・オブジェクトや、少しずつ異なるバージョンのオブジェクトを持つ複数のバージョンのアプリケーションを使用することがあり、その場合はそれらは両方とも正しいものです。正規表現を使用して、テキストなどのプロパティー値の複数のバージョンを許可して、このシナリオに適応させることができます。

オブジェクト・マップを開いてオブジェクトを統合する

1. ClassicsA に対して再生するには、スクリプトの先頭にある startApp コマンドを編集して、 B を A に変更します。
2. 「Functional Test」ツールバーにある「**Functional Test スクリプトの実行**」をクリックします。 再生の際に、Functional Tester はパスワード・チェック・ボックス・オブジェクトで短く一時停止しますが、やがてそれは終了します。ここでスクリプトは警告を出します。ログ内で、それが同じオブジェクトの **rememberPassword** テスト・オブジェクトであることに注意してください。
3. ログを閉じて、パスワード・チェック・ボックス・オブジェクトからオブジェクト・マップを開きます。これは演習 8 で行ったように、スクリプト・エクスプローラーで「**rememberPassword**」をダブルクリックすることによって行います。
4. オブジェクト・マップで、「**アプリケーション**」 → 「**実行**」をクリックしてアプリケーションを開きます。「**ClassicsJavaA**」を選択してから、「**OK**」をクリックします。
5. 任意の CD を選択し、ClassicsCD で「**Place Order**」をクリックして、「Member Logon」ウィンドウを開きます。
6. 「**テスト・オブジェクト**」 → 「**オブジェクトの挿入**」をクリックして、新規オブジェクトをマップに追加します。
7. オブジェクト・ファインダーを使用して、アプリケーション内の「Member Logon」ウィンドウでパスワード・チェック・ボックスを選択します。
8. 「**次へ**」をクリックしてから、「**終了**」をクリックします。
9. オブジェクト・マップの上部ペインで、古いチェック・ボックス・オブジェクトを新規のチェック・ボックス・オブジェクトにドラッグすることにより、それらのオブジェクトを統合します。
10. 必要であれば、「テスト・オブジェクトの統合」ウィザードのいずれかの側を外側にドラッグして拡大し、フィールドを長くしてください。

    ここでは 2 つの異なる正規表現を使用します。 1 つは **name** プロパティーに対するもので、 1 つは **accessibleName** プロパティーに対するものです。

    統合されたオブジェクトは、「**統合されたテスト・オブジェクトのプロパティー**」グリッド (上部ペイン) に表示されます。 **name** プロパティーの値は `checkRemember` です。

プロパティー値の正規表現への変換

1. 上部ペインで、「`checkRemember`」値を右クリックしてから、「**正規表現への値の変換**」をクリックします。

   Functional Tester は値テキストの前に「xy」アイコンを表示して、値を正規表現として指定します。
2. **name** 値を再度ダブルクリックして、フィールドを編集可能にします。
3. `check` という語を削除してから、残りの部分を編集して `[rR]emember` とします。
4. セルの外側をクリックします。

このパターンにより、「remember」の語が大文字の「R」でも小文字の「r」でも許可されて、成功するようになります。比較は大/小文字を区別するので、完全一致だけが成功するため、これは重要です。 **accessibleContext.accessibleName** プロパティーの値は、「Remember Password」です。

5. 「Remember Password」値を右クリックしてから、「**正規表現への値の変換**」を選択して変換します。
6. 値をダブルクリックして、`Remember.*Password` となるように編集します。スペースを除去し、ピリオド (.) およびアスタリスク (*) 文字を追加することになります。
7. 別のセルをクリックします。

   「**.**」は、その位置に任意の文字が出現することを許可します。アプリケーションの 1 つのバージョンではこのプロパティー内の 2 語の間にスペースがあり、別のバージョンではスペースがありません。このパターンを使用すると、両方のケースに対応します。
8. 「**次へ**」をクリックしてから、「**終了**」をクリックします。
9. オブジェクト・マップで「**ファイル**」 → 「**保管**」をクリックして、変更を保管してから、オブジェクト・マップを閉じます。
10. ClassicsCD を閉じます。
11. ClassicsA に対してスクリプトを再び再生します。 この場合、テキスト `Orders for Trent Culpito` は正規表現に変更されていないので、検査ポイントは失敗することが予期されます。 ClassicsA についてのオブジェクト認識に関する警告は、ログからなくなっています。
12. ログを閉じます。
13. ClassicsB を再生するように startApp コマンドを変更してから、スクリプトを実行します。

    オブジェクト認識が、ClassicsB に対しても成功します。正規表現を使用すると、アプリケーションの異なるバージョンで異なるプロパティーを持つオブジェクトをより柔軟に認識できるようになり、どちらも再生中に認識されます。正規表現について詳しくは、Functional Tester のヘルプを参照してください。

要約: 機能テストの作成

この Functional Tester チュートリアルでは、スクリプトのテスト、記録、および再生、検査ポイントの作成、およびオブジェクトのプロパティーまたはデータを更新するための検査ポイント・コンパレーターの使用のために Functional Tester をセットアップする方法、そしてオブジェクト・マップを有効に活用するためのいくつかの方法について示しました。

学習した演習

このチュートリアルを完了すると、以下を行う方法を学習したことになります。

- Functional Tester プロジェクトの作成
- テスト・アプリケーション上のアクションに対するスクリプトの記録
- 記録中でのテスト・アプリケーションの適切な始動
- 検査ポイントの作成
- スクリプトの再生
- Functional Tester ログの使用
- コンパレーターを使用した、検査ポイントの更新
- オブジェクト・マップの更新
- オブジェクトの認識設定の変更

- オブジェクト認識をより柔軟にするための正規表現の使用

追加リソース

このチュートリアルで扱われたトピックについてもっと詳しく知りたい場合、以下のリソースを参照してください。

- Functional Tester のヘルプ
- Functional Tester API Reference
- Functional Tester の「ようこそ」ページ

関連情報

ibm.com

eclipse.org