Jameleonによる機能テスト／回帰テストの自動化について

2007年1月30日

加藤 大受

Agenda

- 自動化の目的
- 自動化に向けての準備
- テストの自動化
- Jameleonの概要
- Jameleonの使い方
- Jameleonのメリット・デメリット

一般的に自動化の目的は・・・・

a)テスト工数が足りない
b)テスターが足りない
c)テストを効率化したい
d)テストのミスを減らしたい
e)テストの単価を減らしたい
f)決まり切ったテストを何度も行っているので効率を上げたい
g)テストパターンを蓄積していきたい
h)毎回のビルドの品質を確認したい
i)デグレードを確認したい

これらの目的が達成されるには、自動化だけでなくテストプロセスの変更が必要。ではテストの自動化で変わることは・・・・

自動化の目的は

- 筆者の経験をベースにすると、テストの自動化により
  - 決まり切ったテストを自動化することにより、空いた工数で他のテストを実施
  - ビルド毎に自動化したテストを実行することで品質を数値化
  - 自動化テストをスモークテスト（テスト側受け入れテスト）として利用
  - 発見された不具合を自動化し、デグレードがないかを確認

テストの自動化により、すぐに実現できることは

テストの自動化は品質の数値化

自動化する内容の決定

- プロジェクトの概要、テストスケジュール、利用するツールの種類などを考慮する。たとえば
  - 機能テスト
  - スモークテスト（テスト側受け入れテスト）
  - 回帰テスト
  - 単体テスト
  - ・・・・・・・
- マスターテスト計画の中で各テストプロセスを計画するときに、自動化するかどうかを検討してみる
  - 概略設計時に作成するマスターテスト計画で各テストプロセス内で自動化をした場合のメリット・デメリットをまとめてみる

テストの自動化について

- どこを自動化するか
  - 優先順位が高いところであるのは当然
  - 自動化したものがどの程度利用できるかを検討する
    - ビルドチェックに使う（スモークテスト扱い）
    - 保守テストに使う
    - 以後のエンハンスの開発プロジェクトでも利用する

自動化の成功のポイント

- 欲をかかないこと。
- 自動化することでかかる工数を常に考える。
- 自動化するのに必要なスキルを身につけること。

テストの自動化について

- 自動化することでマイルストーンごとに行っていた回帰テストをビルドごとに行うことが可能となる
- 未解決の不具合の自動化
  - 不具合が依然として同じ障害を発生させることを確認する
  - 長期のプロジェクトで非常に有効
- 解決済みの不具合の自動化
  - 不具合がきちんと修正されていることを確認する
  - 影響範囲も含めたテストケースを作成する

どの不具合を自動化するか

- ビジネスリスクの高いものを自動化する
  - 単に不具合の深刻度の高いものではなく、不具合の重要度をベースに行う
- 未解決と解決済みはどちらが重要か
  - 基本的に解決済みの不具合が重要
  - 簡単に自動化できる手段があれば未解決も着手する
    - 次期マイルストーンで修正する不具合は自動化しておくとよい。

回帰テストの自動化を行う場合はマスターテスト計画でそれなりに回帰テストの自動化にかかる工数を含めておく必要がある。

Jameleonについて

- Jameleon開発チームが開発しているオープンソースのテストツール
  - http://jameleon.sourceforge.net/
- 最新バージョン
  - 2006年12月12日にリリースされた3.3-M4
- ライセンス
  - GNU Lesser General Public License (GNU LGPL)

Jameleonについて

- **サポートしているテスト**
  - JUnitプラグイン
    - 使ったホワイトボックスレベルのテスト。すべてのJUnitの機能タグは他のプラグインの機能と一緒に利用することも可能
  - Jiffieプラグイン
    - IEを使ったテスト。このテストはWindows環境でのみ実行可能
  - HtmlUnitプラグイン
    - 一部のJavaScriptsとXPathについてもサポートしている。
  - HTTPUnitプラグイン
    - HtmlUnitプラグインではコンテンツベースでテストするのに対し、HTTPUnitではHTTPの通信ベース、つまりプロトコルベースのテストとなる。
  - 3270 (Jagacy)プラグイン
    - 3270ベースのIBMメインフレームで稼働するアプリケーションのテスト

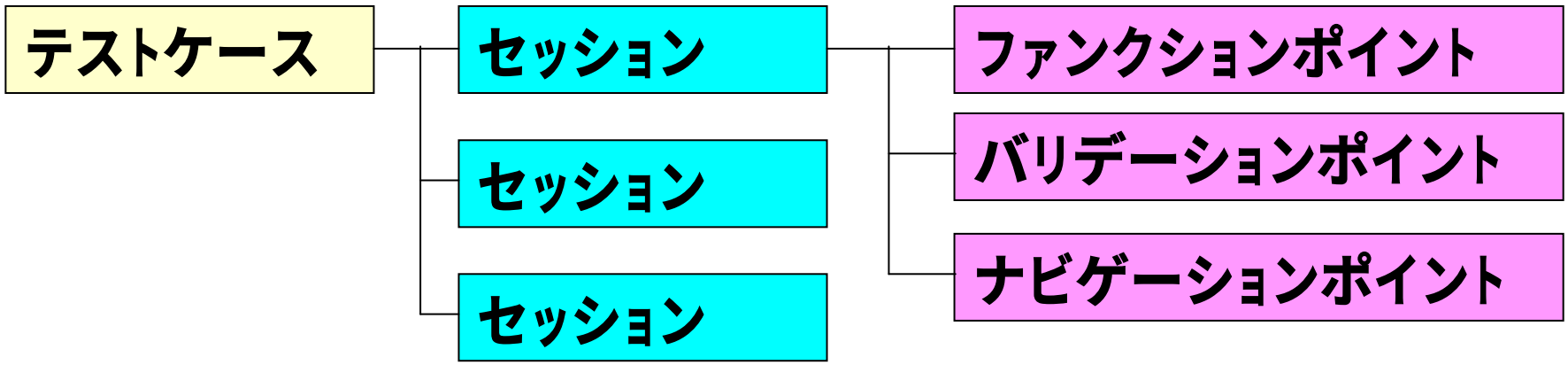
Jameleonのインストール

- ダウンロード
  - JameleonのWebサイトより、Jameleon 3.3-M4のJameleon Test Suiteをダウンロード
- インストール準備
  - J2SE SDKが必要
- インストール
  - ダウンロードしたZIPファイルを解凍するだけ
- インストールすると以下のディレクトリができる
  - scripts - XMLスクリプトを格納
  - src/java - カスタムXMLタグを格納
  - data - データセットを使ったテスト（データドリブンテスト）のためにCSVファイルを格納
  - res - プロパティファイルを格納
  - lib - Jameleonのライブラリファイル

Jameleonのテストスクリプト

- スクリプト言語
  - Jellyスクリプトエンジンを利用
  - JellyはApache Jakartaプロジェクトで開発しているJava・XMLベースのスクリプトエンジン。XMLベースのスクリプトファイルを実行可能とする
  - タグを利用してテストスクリプトを記述していく

Jameleonのスクリプトの構造

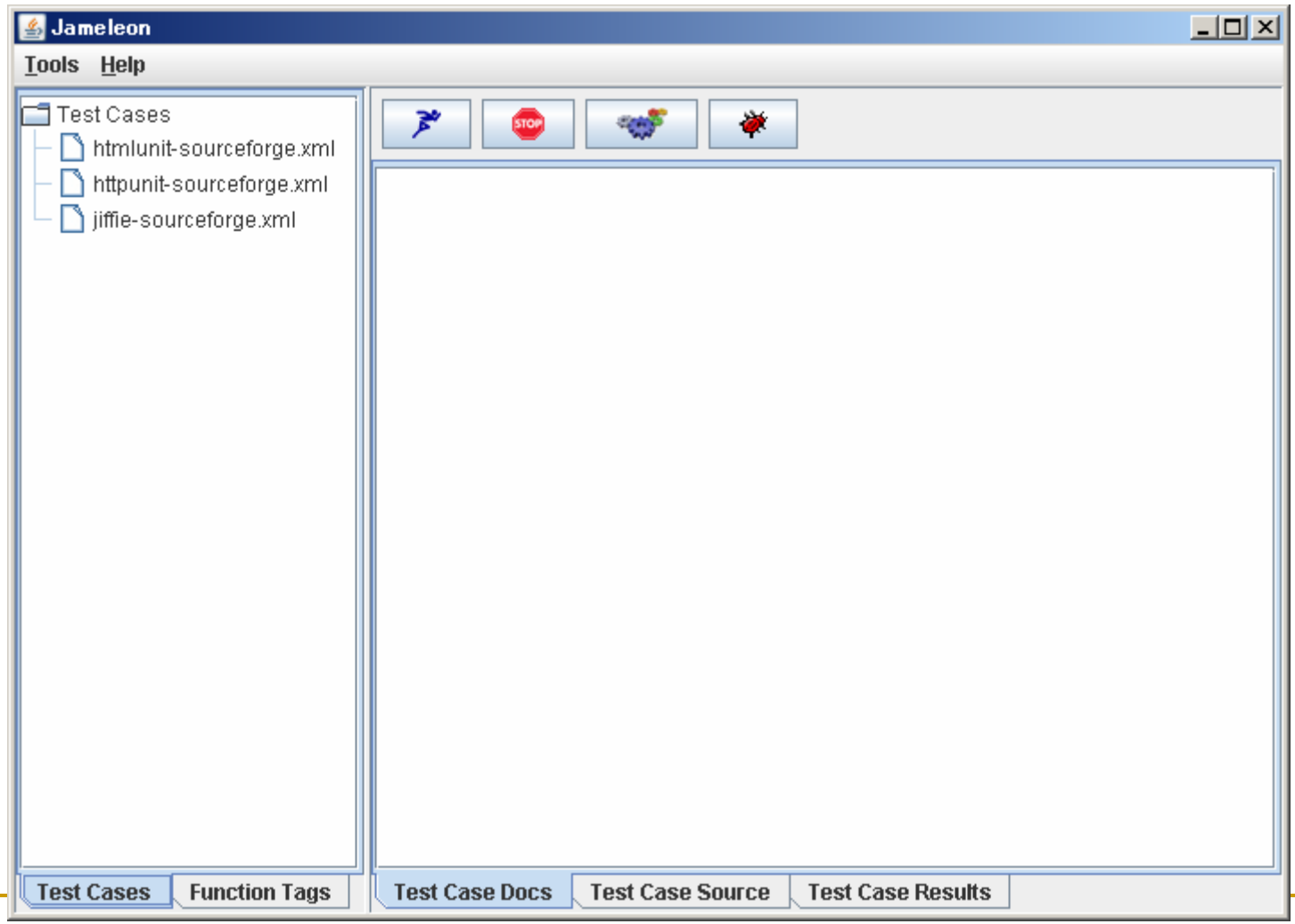


- ファンクションポイント
  - データ入力画面など。
- バリデーションポイント
  - 結果表示画面など。異常ケースもあり。
- ナビゲーションポイント
  - 画面遷移。リンクやナビゲーションツールバーなど

Jameleonの実行

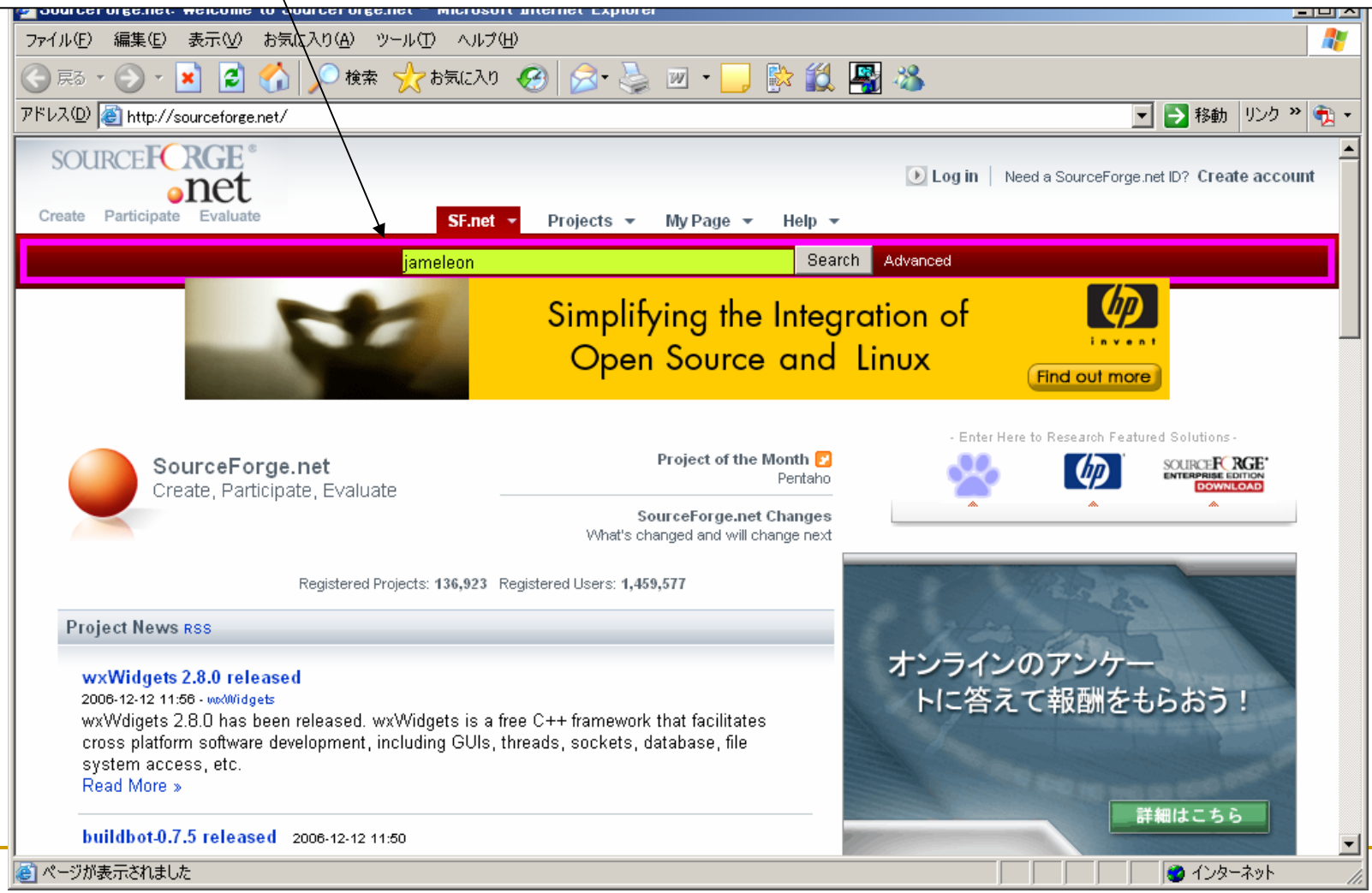
- スクリプトファイルの置き場
  - scriptsフォルダに配置
- スクリプトの実行
  - Jameleon.sh/jameleon.batの起動
  - Jameleon GUIツールで指定されたスクリプトを選択
  - 実行ボタンを押す

Jameleon GUIツール



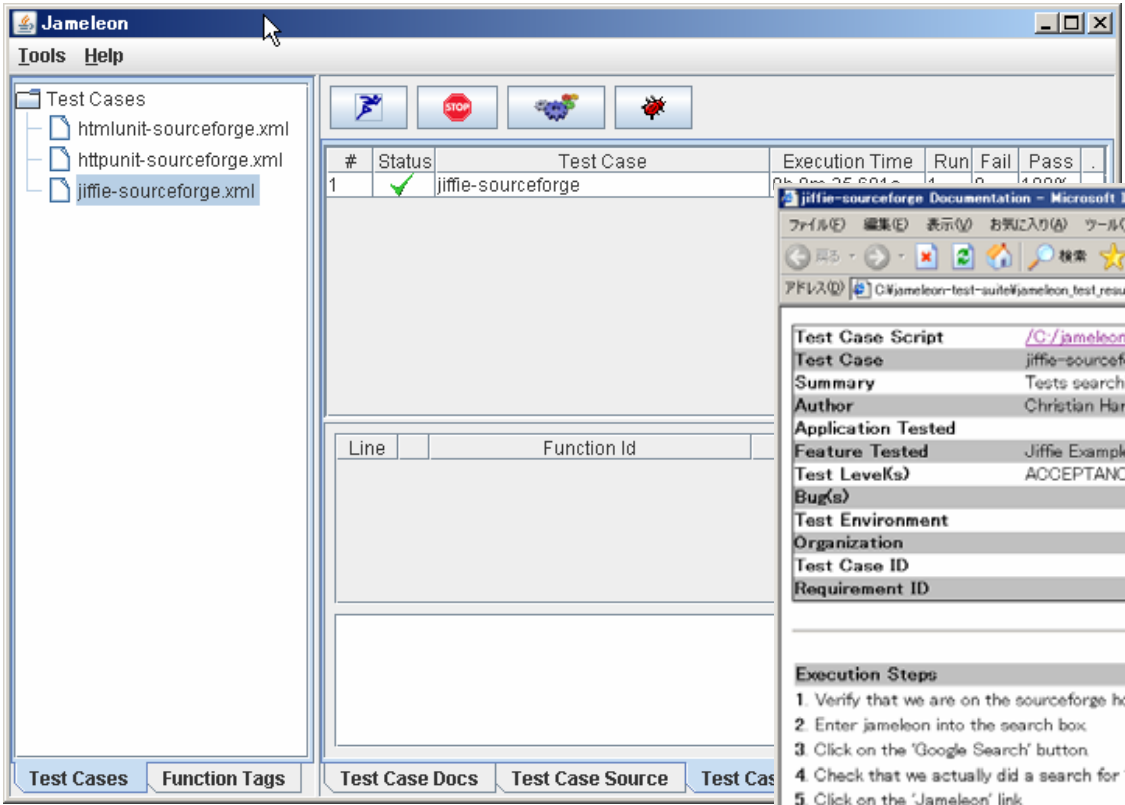
Jameleonの実行

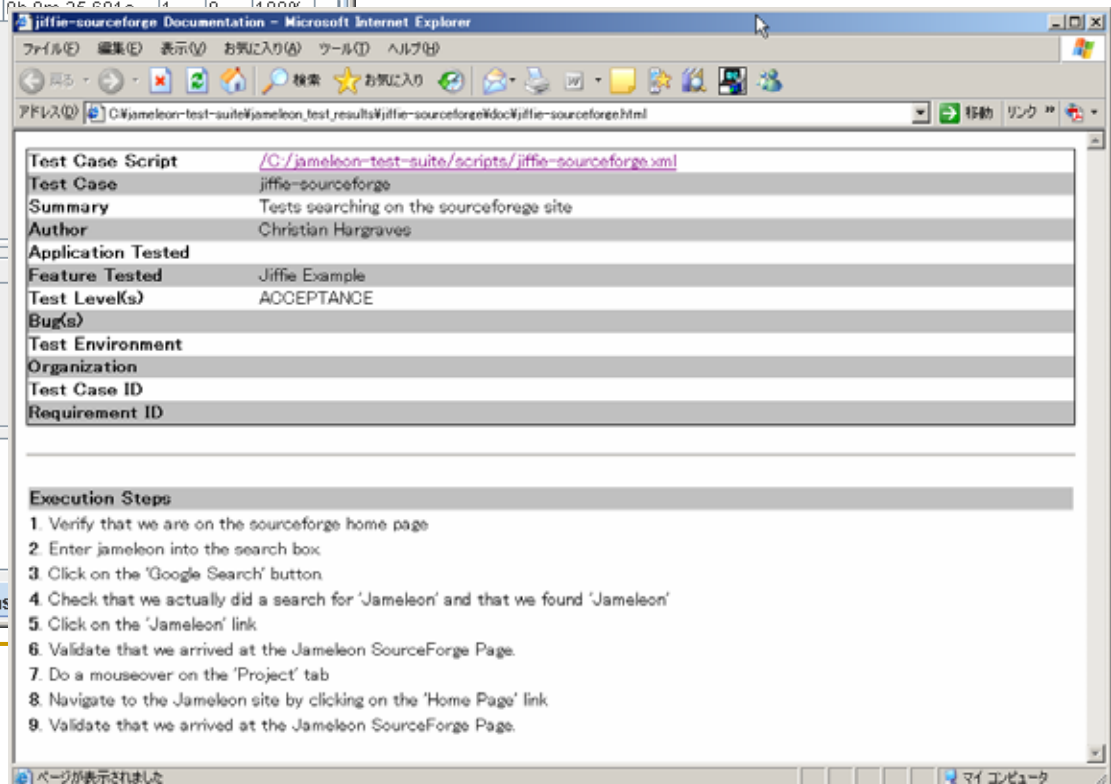
Jameleonのファンクションポイント、バリデーションポイントがピンク色で囲まれる



テスト結果

- Jameleon GUIに表示されるとともに、テスト結果ドキュメントがjameleon_test_results/jiffie-sourceforge/doc以下にHTMLファイルで作成される。





まずは動作をみてみよう

Jameleonのスクリプトファイル

- jiffie-sourceforge.xmlを見てみよう

Jellyスクリプトであることを宣言

```
<testcase xmlns="jelly:jameleon">
```

テストスクリプトの概要

```
  <test-case-summary>Tests searching on the sourceforege
    site</test-case-summary>
  <test-case-author>Christian Hargraves</test-case-author>
 <test-case-level>ACCEPTANCE</test-case-level>
 <functional-point-tested>Jiffie Example</functional-point-tested>
```

テストの手順

```
<ie-session baseUrl="http://sourceforge.net" beginSession="true">
    <ie-validate
        functionId="Verify that we are on the sourceforge home page"
        title="SourceForge.net: Welcome to SourceForge.net"
        textPresent="Open Source Technology Group"/>
```

Jameleonのテストスクリプト

- <ie-session baseUrl="http://sourceforge.net" beginSession="true">
  - Jiffleプラグインでのテストの開始
  - baseUrlパラメータ：アクセス先のURL
  - beginSessionパラメータ：セッションを開始するかどうか
- <ie-validate
  functionId="Verify that we are on the sourceforge home page"
  title="SourceForge.net: Welcome to SourceForge.net"
  textPresent="Open Source Technology Group"/>
  - functionIdパラメータ： バリデーションポイントの説明
  - titileパラメータ：ページタイトルの期待結果（部分一致）
  - textPresent：ページ内に含まれる文字列（含まれていないときはエラー）

Jameleonのテストスクリプト

- <ie-set-text-field
  functionId="Enter jameleon into the search box."
  name="words"
  value="jameleon"
  form="searchform"/>
  - functionIdパラメータ：このファンクションポイントの説明
  - nameパラメータ：テキストボックスの名前

このファンクションポイントでは、searchformというフォームオブジェクト内のwordsという名称のテキストボックスに“jameleon”という文字列を入力している。

Jameleonのテストスクリプト

- <ie-click-button
  functionId="Click on the 'Google Search' button."
  form="searchform"
  value="Search"
  eventToFire="onclick"
  functionDelay="300"/>
  - functionIdパラメータ：ナビゲーションポイントの説明
  - formパラメータ：フォームオブジェクトの名称
  - valueパラメータ：ボタンのテキスト文字列
  - eventToFileパラメータ：JavaScriptsのイベント名称
  - functionDelayパラメータ：イベント実行後に待つ時間 (ms)

このナビゲーションポイントでは、searchformというフォームオブジェクト内にある”Search”というテキスト文字のボタンをクリックし、クリック後、300ms 待機している。

Jameleonのテストスクリプト

- <ie-validate
  functionId="Check that we actually did a search for 'Jameleon' and that we found 'Jameleon'"
  title="SourceForge.net: Software Search"
  linkPresent="Jameleon"/>
  - functionIdパラメータ：バリデーションポイントの説明
  - titleパラメータ：ページタイトルの期待値
  - linkPresentパラメータ：指定された文字列のリンクがあるかどうか

このバリデーションポイントでは、遷移されたページが、“SourceForge.net: Software Search”というページタイトルであり、linkPresentパラメータで“Jameleon”という文字列のリンクがあるかどうかをチェックしている。

Jameleonのテストスクリプト

- <ie-click-link
  functionId="Click on the 'Jameleon' link"
  link="Jameleon"/>
  - functionId：**ナビゲーションポイントの説明**
  - link**パラメータ：”クリックするリンクの文字列**

ナビゲーションポイントであり、“Jameleon”というリンクをクリックしている。

Jameleonのテストスクリプト

- <ie-validate

  functionId="Validate that we arrived at the Jameleon SourceForge Page."

  title="SourceForge.net: Jameleon"

  linkPresent="engrean"/>
  - functionId：**ナビゲーションポイントの説明**
  - title**パラメータ：ページタイトルの期待値**
  - linkPresent**パラメータ：指定された文字列のリンクの確認**

このバリデーションポイントは、直前の“Jameleon”**というリンクをクリックするナビデーションポイントによる動作の結果、期待するページが表示されているかどうかを確認するためのもので、**“SourceForge.net: Jameleon”**というページタイトルで、ページ内に**“engrean”**というリンクが存在するかどうかを確認している。**

Jameleonのテストスクリプト

- <ie-fire-event
  functionId="Do a mouseover on the 'Project' tab"
  htmlElement="li" eventToFire="onmouseover">
  <ie-attribute>
  <ie-attribute-name>class</ie-attribute-name>
  <ie-attribute-value>begin selected</ie-attribute-value>
  </ie-attribute>
  </ie-fire-event>
  - functionId**パラメータ：ファンクションポイントの説明**
  - htmlElement**パラメータ：Htmlタグの指定**
  - eventToFilre**パラメータ：JavaScriptsのイベント名称**
  - <ie-attribute-name>**タグ：属性の名前**
  - <ie-attibute-value>**タグ：属性の値**

ここではHtmlタグの<LI>タグで、“begin selected”というスタイルシートで定義されたものに対して、“mouseover”のイベントを実行している。つまり、Projectタブにマウスポインタを置き、Pojectタブのメニューを表示している。

Jameleonのテストスクリプト

- <ie-click-link
  functionId="Navigate to the Jameleon site by clicking on the 'Home Page' link"
  link="Web Site"/>
  - functionIdパラメータ：ナビゲーションポイントの説明
  - linkパラメータ：指定された文字列のリンクをクリック

このナビゲーションポイントでは、以前のファンクションポイントで表示した“Project”タグのメニュー内の“Web Site”というリンクをクリックする。

Jameleonのテストスクリプト

- <ie-validate
  functionId="Validate that we arrived at the Jameleon SourceForge Page."
  title="Jameleon - An Automated Testing Tool - Overview"/>
  - functionIdパラメータ：バリデーションポイントの説明
  - Titleパラメータ：ページタイトルの期待値

このバリデーションポイントは、以前のナビゲーションポイントによって遷移したページが“Jameleon - An Automated Testing Tool - Overview”というページタイトルであることを確認している。

Jameleonのテストスクリプト

- スクリプトファイルの流れ

テスト対象のURL		http://sourceforge.net
順序	種類	内容
1	バリデーションポイント	SourceForge.netのトップページの確認
2	ファンクションポイント	wordsテキストボックスに"Jameleon"をセット
3	ナビゲーションポイント	Searchボタンを実行
4	バリデーションポイント	"SourceForge.net: Software Search"の表示。"Jameleon"のリンクがあるか
5	ナビゲーションポイント	"Jameleon"のリンクをクリック
6	バリデーションポイント	SourceForge.net: Jameleon"というページタイトルのページかどうか。"engrean"のリンクがあるか
7	ファンクションポイント	"Project"タブが選択されているか
8	ナビゲーションポイント	"Web Site"のリンクをクリック
9	バリデーションポイント	"Jameleon - An Automated Testing Tool - Overview"というページタイトルのページかどうか

テストスクリプトの作成の流れ

- テストケースの設計
  - テストする機能をテストケースとして設計する
- 設計したテストケースを分解
  - バリデーションポイント、ナビゲーションポイント、ファンクションポイントに分割する
- スクリプトの作成
  - 実際にアプリケーションを動かしながらポイントをタグで記述していく
- 実行して確認
  - 確認するときはjameleon.confのfunctionDelayを利用する
- 自動実行スクリプトとして活用

Jameleonのメリット

- Javaの知識がほとんどいらない
  - Version 2.x時代はJavaの知識を非常に必要とした。
  - 現在のバージョンではJellyのスクリプトを書くだけ
- 手軽に利用できる
  - オープンソースの製品なので手軽に利用が可能
- テスト対象のページが開発中でもスクリプト作成が可能
  - オブジェクトの名称が決まっていればレイアウトに依存しない
- 開発が活発である
  - 3.3-M4ではSelenium用のプラグインも含まれた（βレベル）

Jameleonのデメリット

- 日本語で書かれた情報がほとんどない
  - 英語ベースならMLやWebサイトに情報は多数ある
- HTMLの知識が必要
  - HTMLが苦手だと利用は厳しい
- 判定するポイントの見極めが難しい
  - スクリーンショット機能がないのでテストが失敗したケースの確認が難しい（現在、開発は始まっている）
  - それなりにノウハウの蓄積が必要
- 待っているだけじゃ何も情報は来ない
  - ベンダー製品ではないので、自分から積極的に情報の取得が必要。

参考資料

- JameleonのWebサイト内
  - Jameleon Manual (PDF形式：英文)
    - Jameleonの基本的な使い方をまとめたもの
- Jameleonに関する記事
  - ITmedia エンタープライズ
    - オープンソースの自動化テストツール「Jameleon」の概要
      - http://www.itmedia.co.jp/enterprise/articles/0601/12/news001.html
    - Jameleonを使ったテスト
      - http://www.itmedia.co.jp/enterprise/articles/0612/26/news018_5.html

ご静聴ありがとうございました