

Seleniumを使用して、テストの自動化に挑む！JUnit編

[Eclipse](#) [java](#) [JUnit](#) [UIテスト](#) [WEB制作](#) [テスト](#) [ホームページ制作](#) [自動化](#) こんにちは！

糖質制限ダイエットの成果が

徐々に出てきて調子にのってる遠藤です。

今回はSelenium&JUnitを使ってFirefox、Chrome、Safariのブラウザテストを実施します。

JUnitテストコード作成

操作ログの取得

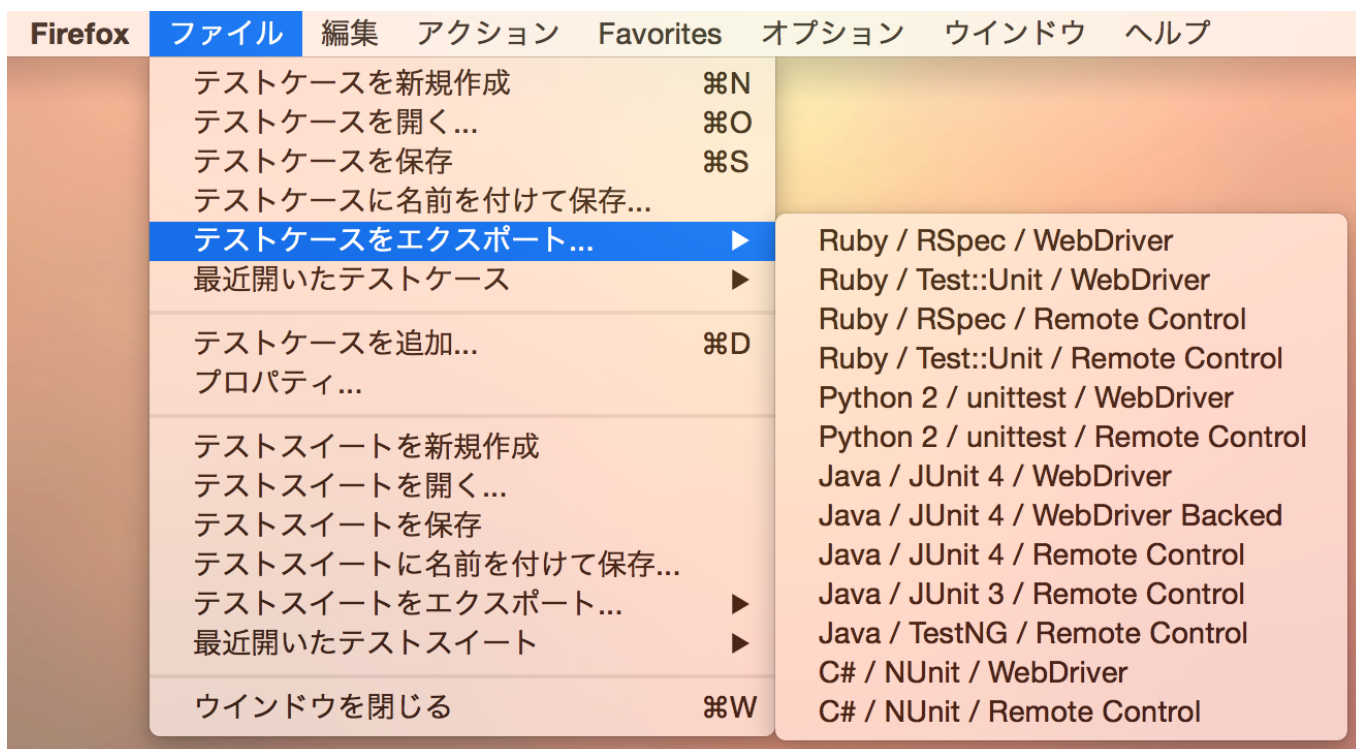
基本編で使ったSelenium IDEを使って操作ログを記録します。

基本編はこちら→<https://www.seekcloud.co.jp/blog/memorandum/158>

操作ログのエクスポート

操作ログをJUnit向けにエクスポートします。

1. ファイル → テストケースをエクスポート → Java / JUnit4 / WebDriver を選択



Eclipse テストプロジェクト作成

Eclipseにはいつからか標準でJUnitを実行する環境が整っています。

ってわけで、早速ガツガツテスト環境を構築します。！

Eclipse プロジェクト作成

1. ファイル→新規→Java プロジェクトをクリック
2. プロジェクト名に「SeleniumSample」と入力後、完了ボタンをクリック

新規 Java プロジェクト

Java プロジェクトの作成

Java プロジェクトをワークスペースまたは外部ロケーションに作成します。

プロジェクト名:

☒ デフォルト・ロケーションを使用

ロケーション: [参照...](#)

JRE

☒ 実行環境 JRE の使用:

☐ プロジェクト固有の JRE を使用:

☐ デフォルト JRE の使用 (現在は 'Java SE 7 [1.7.0_45]') [JRE を構成...](#)

プロジェクト・レイアウト

☐ プロジェクト・フォルダーをソースおよびクラス・ファイルのルートとして使用

☒ ソースおよびクラス・ファイルのフォルダーを個別に作成 [デフォルトを構成...](#)

ワーキング・セット

☐ ワーキング・セットにプロジェクトを追加

ワーキング・セット: [選択...](#)

[?<戻る](#) [次へ>](#) [キャンセル](#) [完了](#)

3. ファイル名に「任意のファイル名.java」と入力し、保存をクリック
- ここでは「TestCase001.java」と入力します。

Save test case Untitled as...

名前: ▼

タグ:

場所: 📁 デスクトップ ⬆

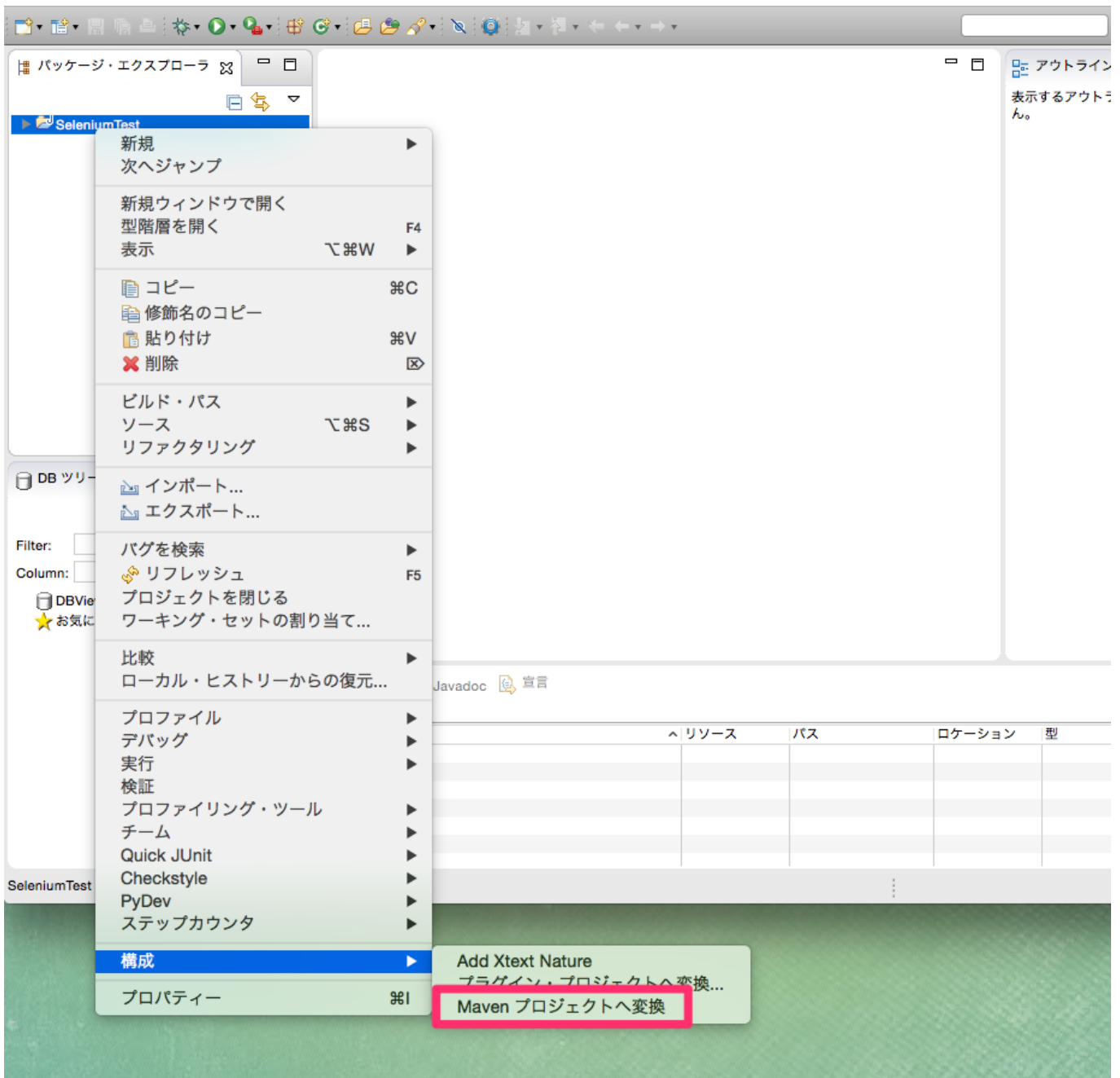
ファイル形式: すべてのファイル ⬆

キャンセル 保存

Mavenプロジェクトに変更

Mavenプロジェクトに変更して、Mavenでライブラリの管理を行います。

1. プロジェクトを右クリック → 構成 → Mavenプロジェクトへ変換



2. 各項目はそのままで、「完了」ボタンをクリック。

レッドブルを飲みながらしばらく待ちます。

新規 POM の作成

Maven POM

このウィザードは Maven の新規 POM (pom.xml) 記述子を作成します。

プロジェクト: /SeleniumSample

成果物

グループ Id: SeleniumSample

アーティファクト Id: SeleniumSample

バージョン: 0.0.1-SNAPSHOT

パッケージング: jar

名前:

説明:

?

キャンセル

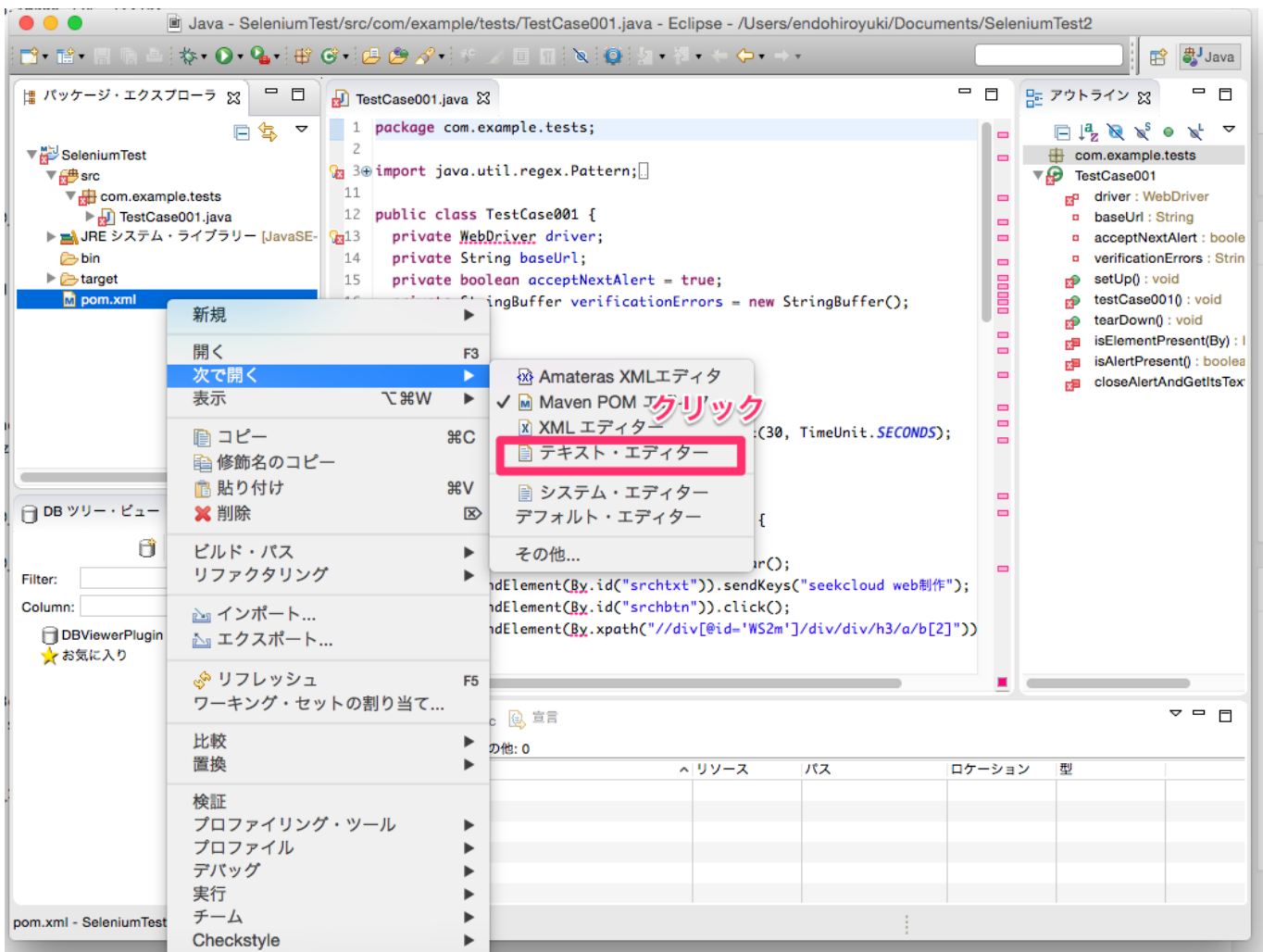
完了

Mavenに依存ライブラリを追加

<https://www.seekcloud.co.jp/blog/memorandum/197>

5/16

1. 「pom.xml」を右クリック → 次で開く → テキスト・エディター をクリック



2. 下記のように編集します。

```

1  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www
2  <modelVersion>4.0.0</modelVersion>
3  <groupId>SeleniumSample</groupId>
4  <artifactId>SeleniumSample</artifactId>
5  <version>0.0.1-SNAPSHOT</version>
6  <build>
7      <sourceDirectory>src</sourceDirectory>
8      <plugins>
9          <plugin>
10             <artifactId>maven-compiler-plugin</artifactId>
11             <version>3.1</version>
12             <configuration>
13                 <source>1.8</source>
14                 <target>1.8</target>
15             </configuration>
16         </plugin>
17     </plugins>
18 </build>
19 <dependencies>
20     <dependency>
21         <groupId>org.seleniumhq.selenium</groupId>
22         <artifactId>selenium-java</artifactId>
23         <version>2.45.0</version>
24     </dependency>
25     <dependency>

```

```

26     <groupId>junit</groupId>
27     <artifactId>junit</artifactId>
28     <version>4.12</version>
29 </dependency>
30 </dependencies>
31 </project>

```

Mavenの最新の設定値に関しては、[ここ](#)を参照してください。

Selenium Maven

<http://www.seleniumhq.org/download/maven.jsp>

JUnit

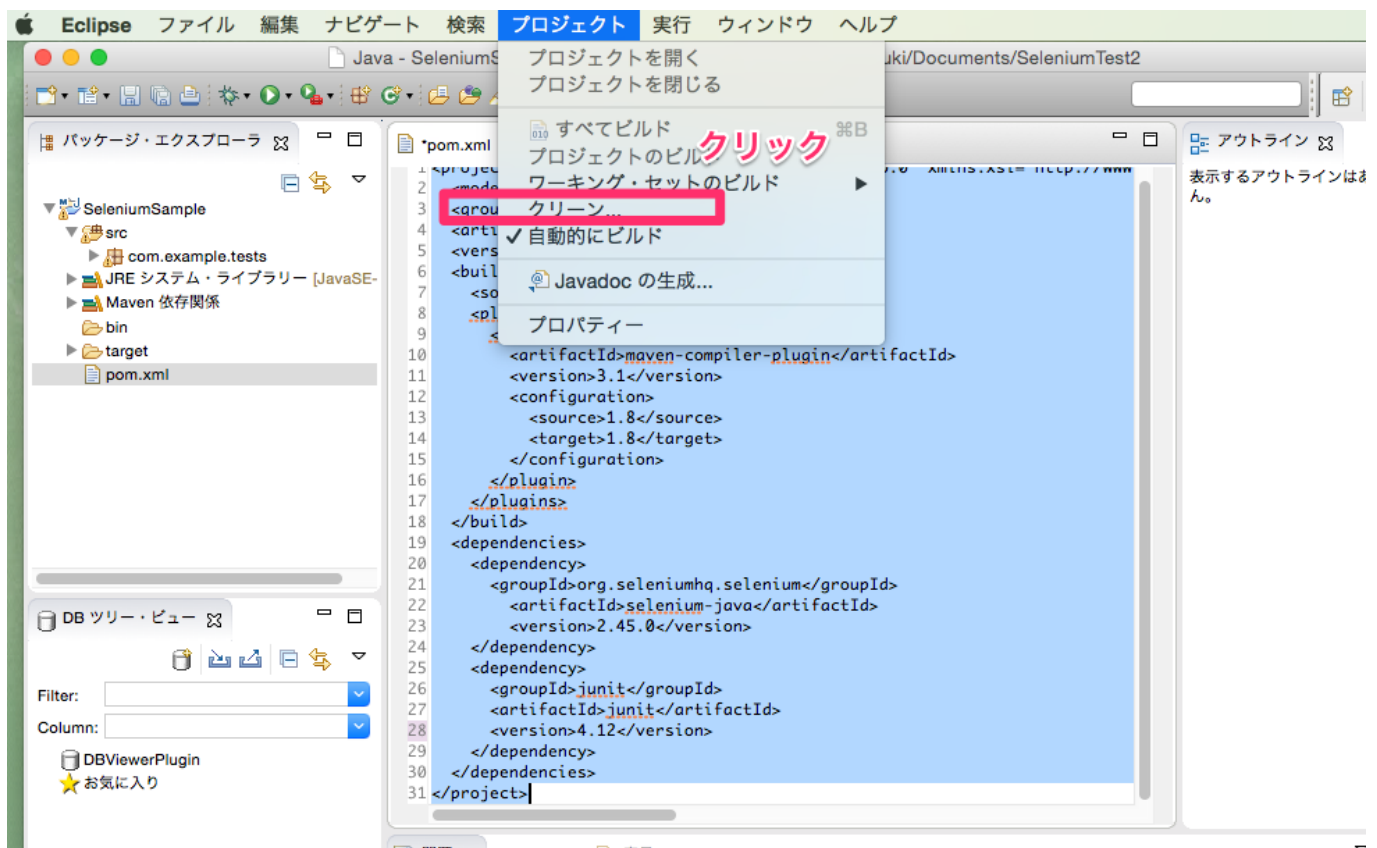
<http://mvnrepository.com/artifact/junit/junit>

プロジェクトのリビルド

Mavenの依存ライブラリを追加したので、プロジェクトをリビルドします。

1. メニューバーの「プロジェクト」→「クリーン」をクリック

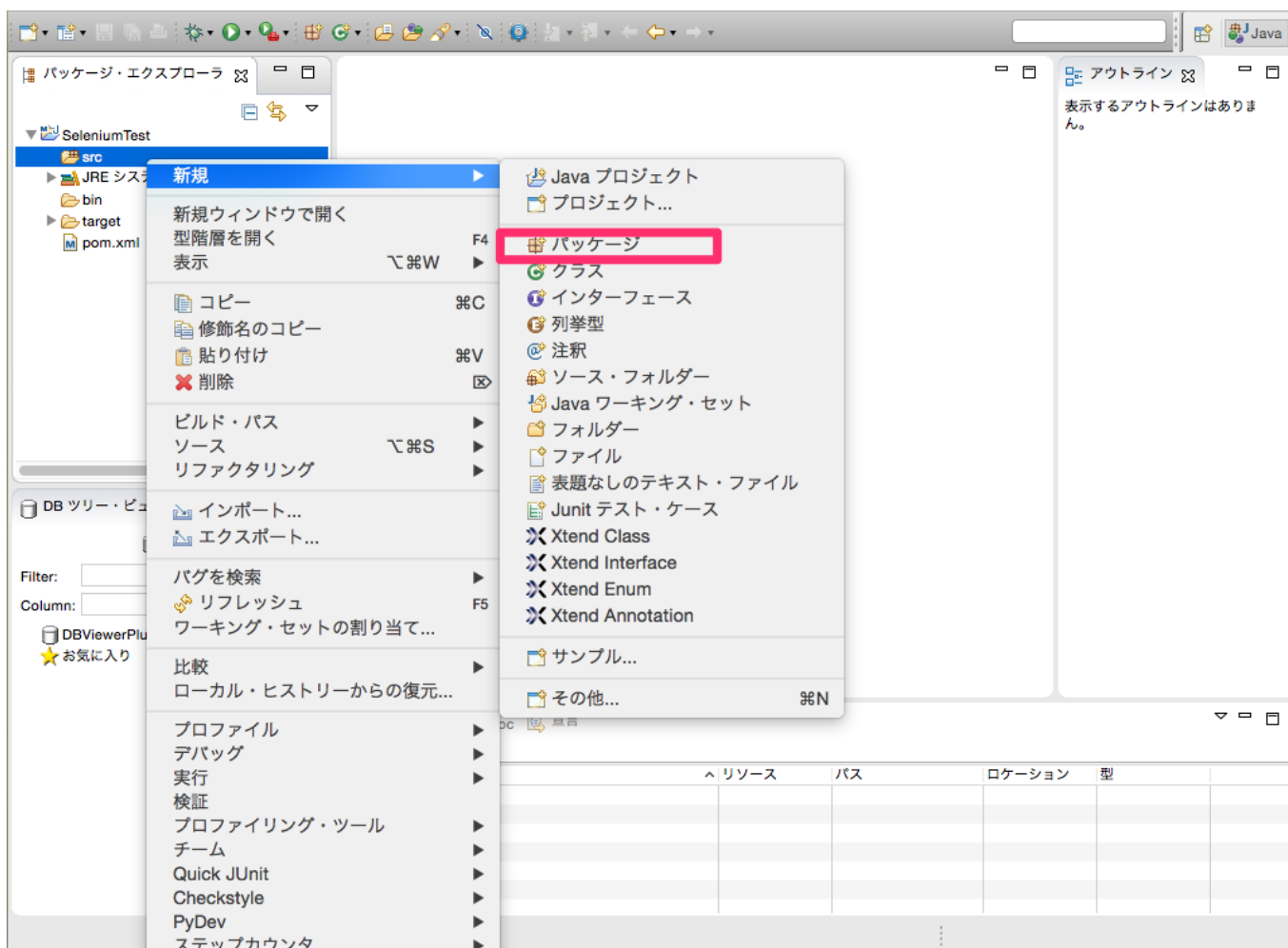
モンスターエナジーを飲みながらしばらく待ちます。



インポート先パッケージ作成

インポート先のパッケージ作成を作成します。

1. 「src」 右クリック → 新規 → パッケージ をクリック



2. 名前に「com.example.tests」と入力して、「完了」ボタンをクリック

新規 Java パッケージ

Java パッケージ

新規 Java パッケージを作成します。

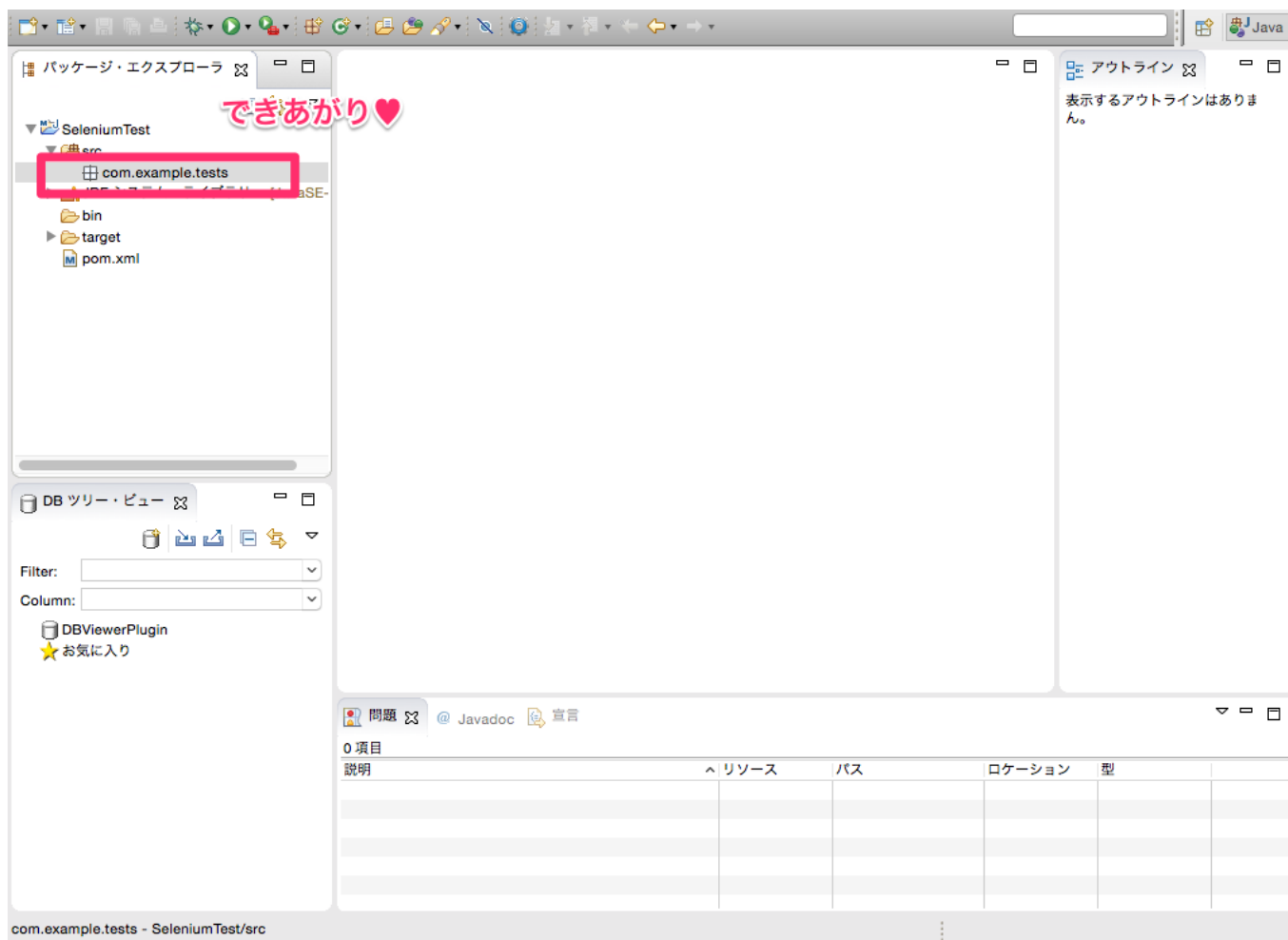
パッケージに対応するフォルダーを作成します。

ソース・フォルダー: SeleniumTest/src 参照...

名前: com.example.tests

☐ package-info.java を作成する

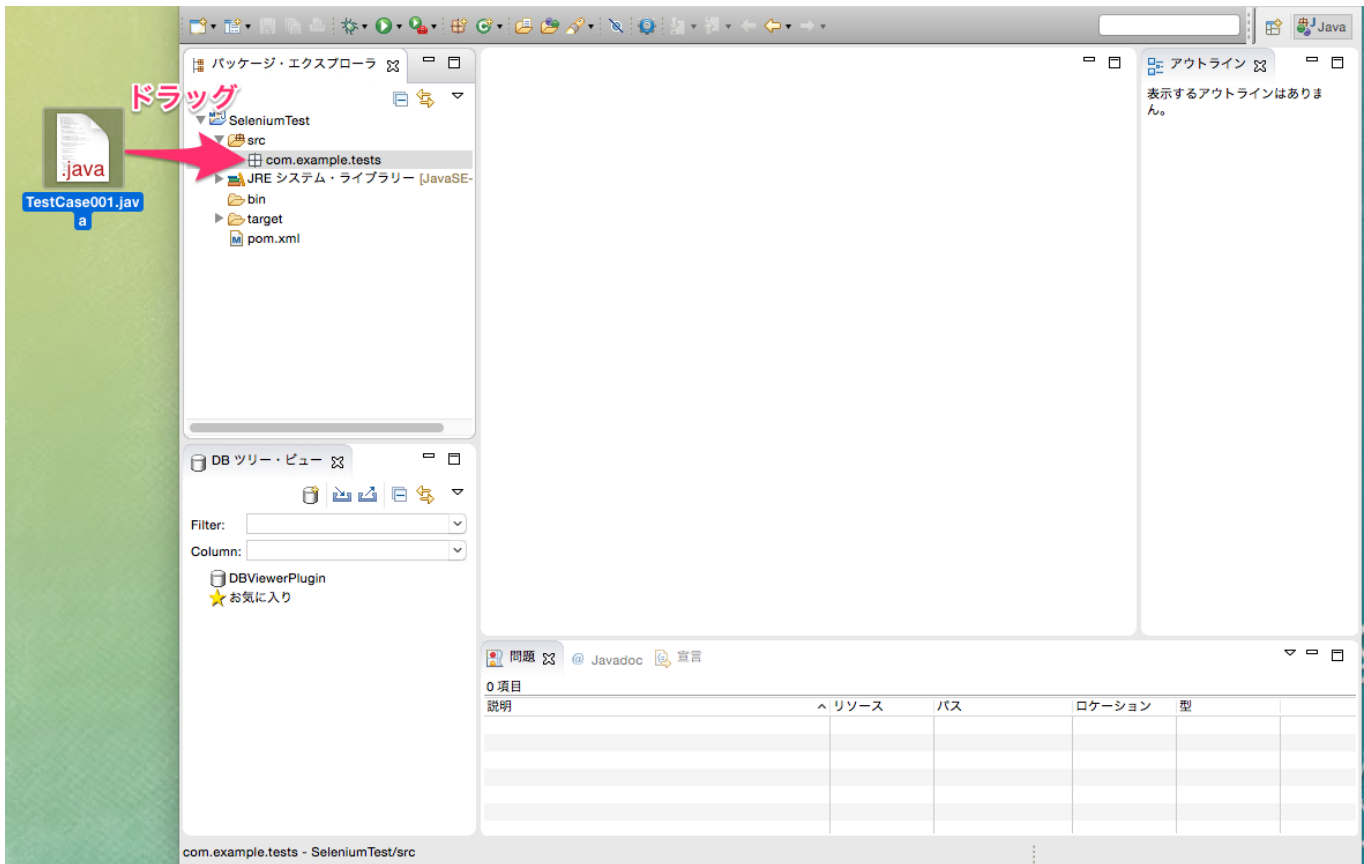
? キャンセル 完了



ファイルをインポート

Eclipseのプロジェクトにテストコードをインポートします。

1. 先ほど作成したプロジェクトにJUnitテストコードをドラック



2. ファイル操作についてウィンドウが立ち上がったら、そのままOK

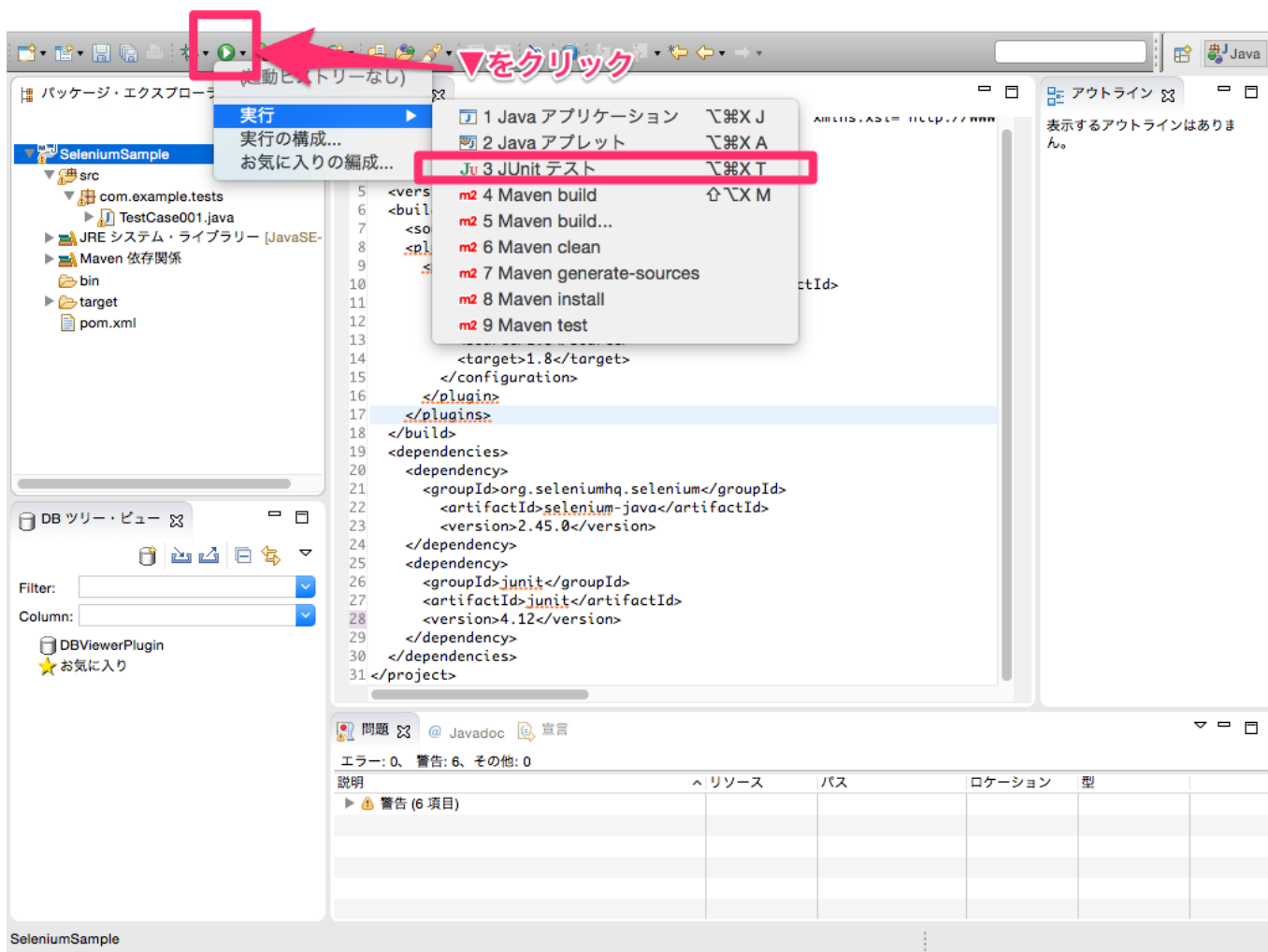


テスト実行

いよいよテストを実行します！！！！

1. 実行ボタンの右のちっさい「▼」をクリック → 実行 → JUnitテストをクリック

自動でFireFoxが立ち上がり、弊社サイトが表示後、FireFoxが閉じれば成功です！



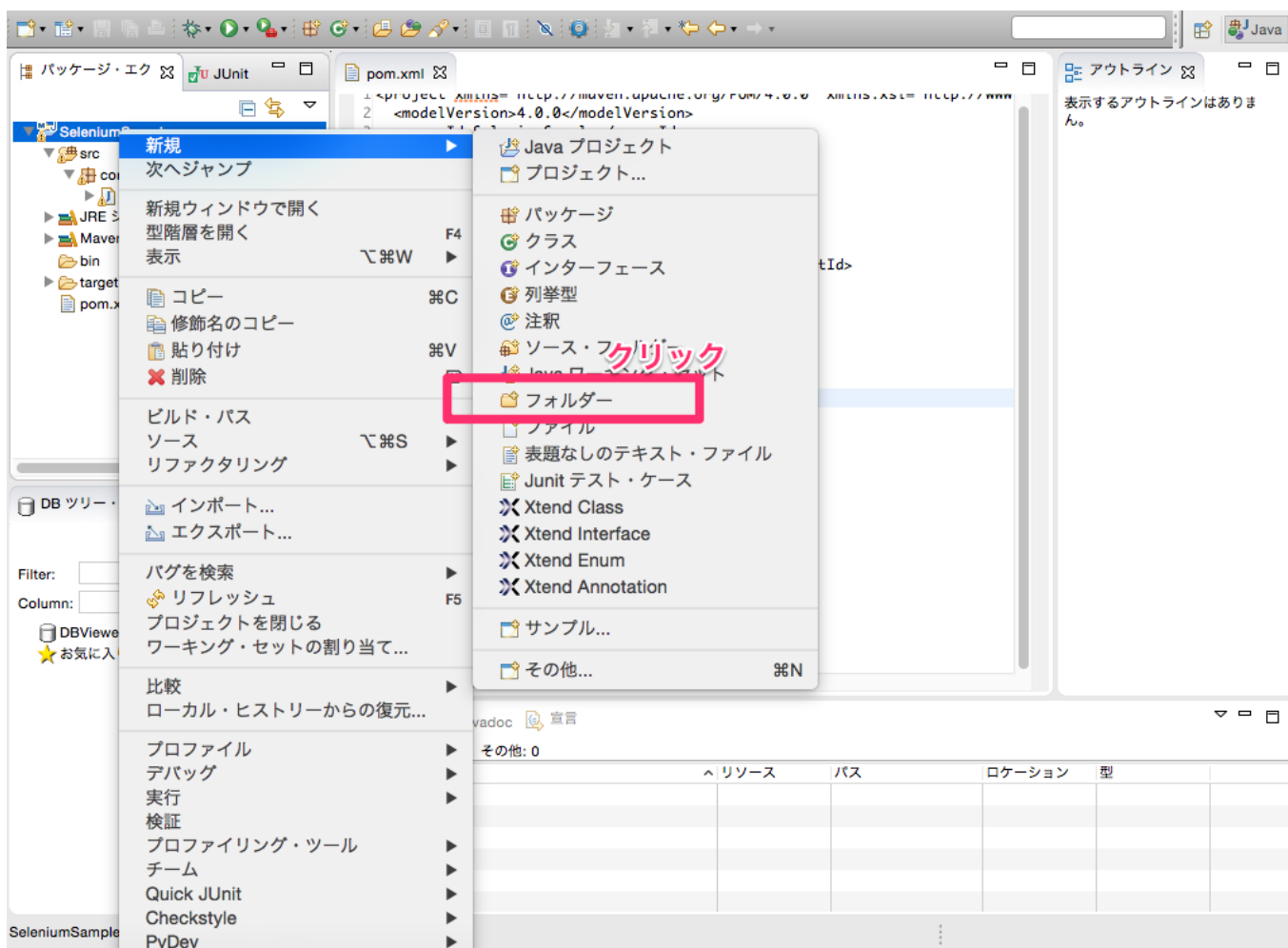
Chromeでテストを実行する

Chromeで実行するようにします。

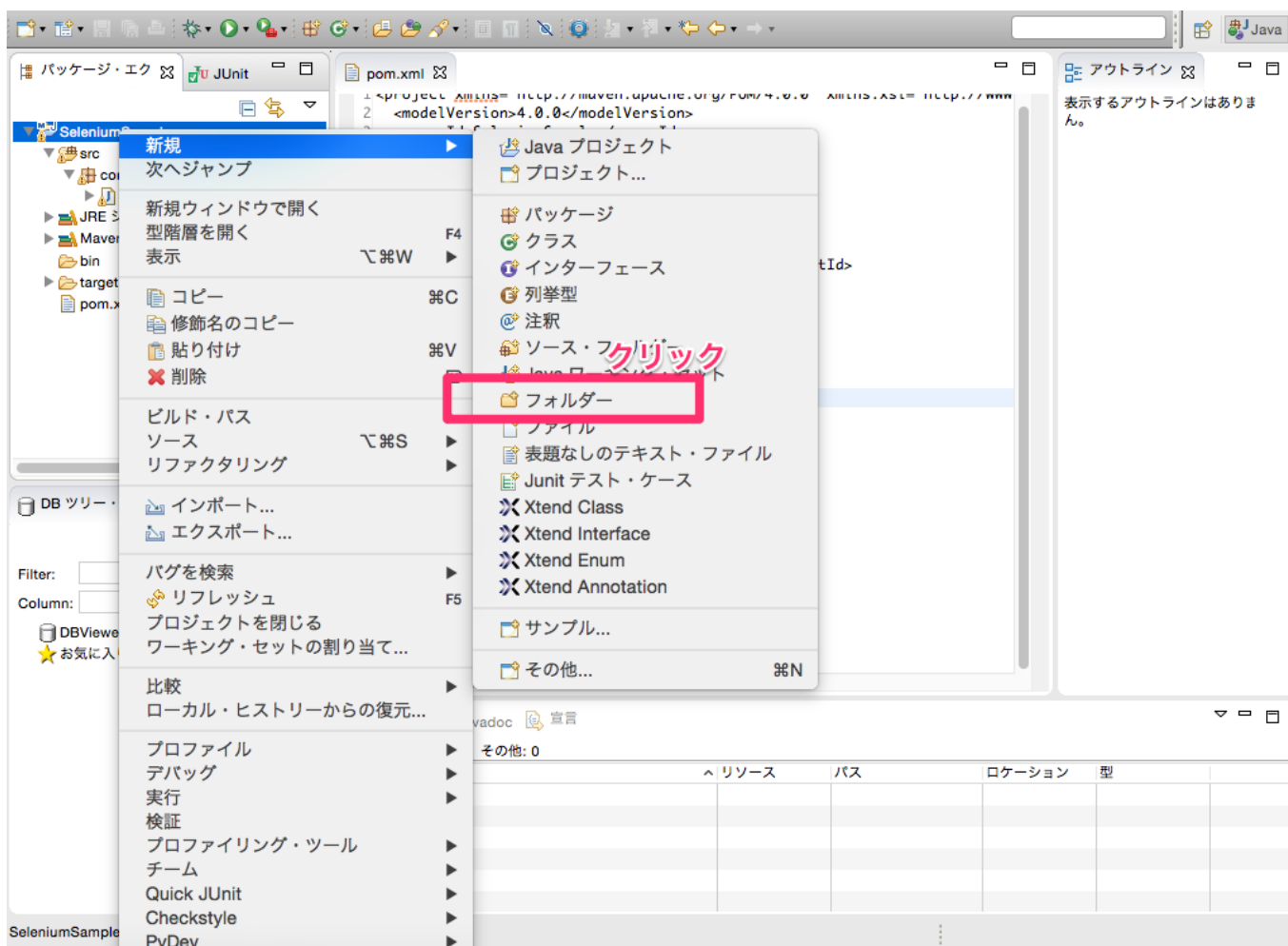
WebDriver設置フォルダ作成

プロジェクトフォルダにDriverフォルダを作成します

1. プロジェクト名を右クリック → 新規 → フォルダー



2. フォルダ名に「Driver」と入力後、「完了」ボタンをクリック



WebDriverダウンロード

1. 下記のサイトから最新版のファイルをダウンロードし解凍後、作成したフォルダに設置

<http://chromedriver.storage.googleapis.com/index.html>

2. ターミナルから実行権限を付与する

```
1 | chmod +x chromedriver
```

3. TestCase001.javaを書き換える

書き換え後、importの編成の更新をお忘れなく！

```
1 | public void setUp() throws Exception {
2 |     //driver = new FirefoxDriver(); // コメントアウト
3 |     System.setProperty("webdriver.chrome.driver", "Driver/chromedriver");
4 |     driver = new ChromeDriver(); // 追記
5 |     baseUrl = "http://www.yahoo.co.jp/";
6 |     driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
7 | }
```

4. テスト実行！

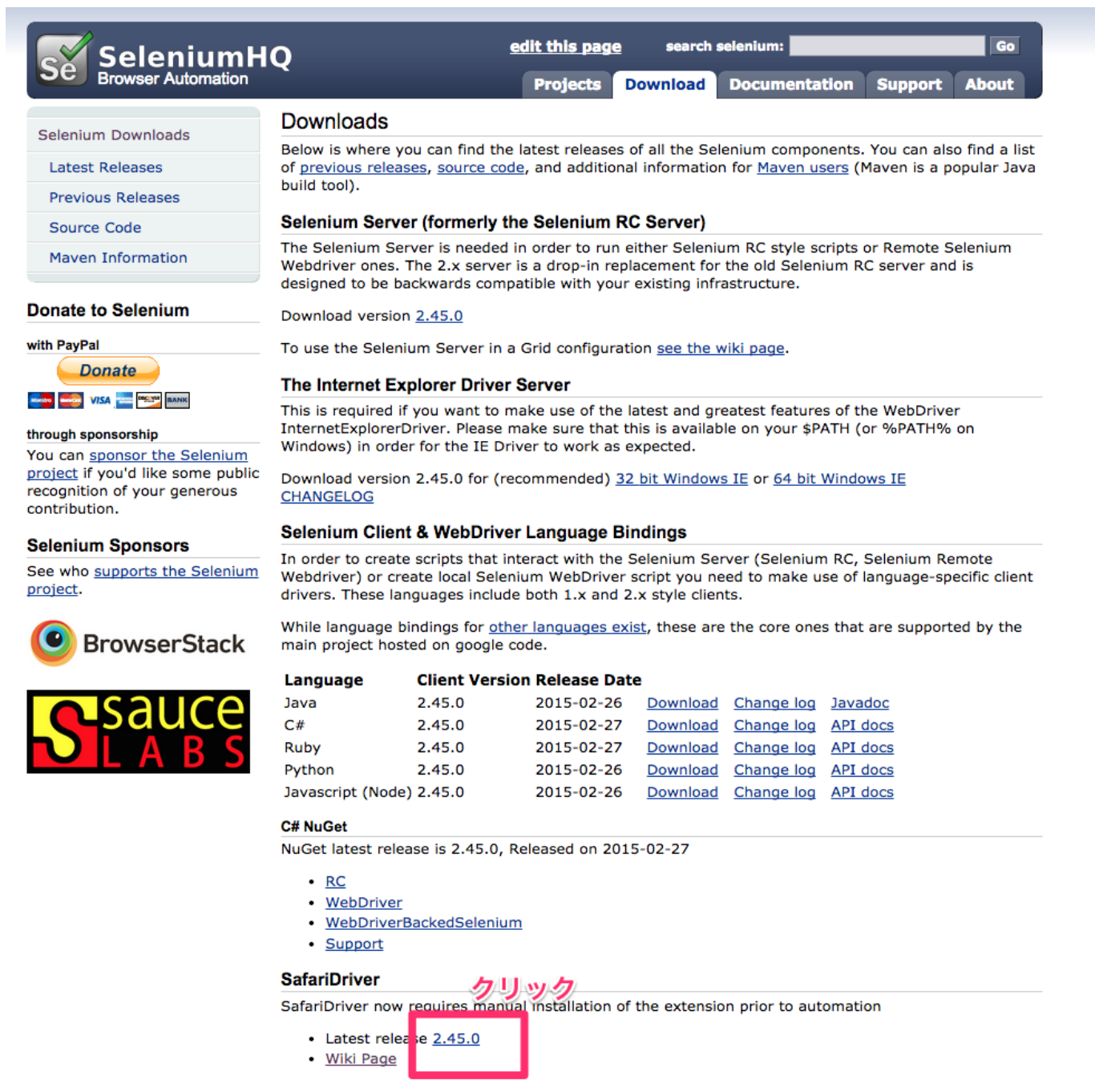
Chromeで弊社のサイトが表示されれば成功！！

Safariでテストを実行する

Safariで実行するようにします。

Safariの拡張機能ダウンロード

1. Seleniumの公式サイトからSafariの拡張機能をダウンロード



SeleniumHQ Browser Automation

[edit this page](#) search selenium: [Go](#)

[Projects](#) [Download](#) [Documentation](#) [Support](#) [About](#)

Selenium Downloads

- [Latest Releases](#)
- [Previous Releases](#)
- [Source Code](#)
- [Maven Information](#)

Donate to Selenium

with PayPal

[Donate](#)

[through sponsorship](#)

You can [sponsor the Selenium project](#) if you'd like some public recognition of your generous contribution.

Selenium Sponsors

See who [supports the Selenium project](#).

BrowserStack

Sauce Labs

Downloads

Below is where you can find the latest releases of all the Selenium components. You can also find a list of [previous releases](#), [source code](#), and additional information for [Maven users](#) (Maven is a popular Java build tool).

Selenium Server (formerly the Selenium RC Server)

The Selenium Server is needed in order to run either Selenium RC style scripts or Remote Selenium Webdriver ones. The 2.x server is a drop-in replacement for the old Selenium RC server and is designed to be backwards compatible with your existing infrastructure.

Download version [2.45.0](#)

To use the Selenium Server in a Grid configuration [see the wiki page](#).

The Internet Explorer Driver Server

This is required if you want to make use of the latest and greatest features of the WebDriver InternetExplorerDriver. Please make sure that this is available on your \$PATH (or %PATH% on Windows) in order for the IE Driver to work as expected.

Download version 2.45.0 for (recommended) [32 bit Windows IE](#) or [64 bit Windows IE](#)

[CHANGELOG](#)

Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote Webdriver) or create local Selenium WebDriver script you need to make use of language-specific client drivers. These languages include both 1.x and 2.x style clients.

While language bindings for [other languages exist](#), these are the core ones that are supported by the main project hosted on google code.

Language	Client Version	Release Date	Download	Change log	Javadoc
Java	2.45.0	2015-02-26	Download	Change log	Javadoc
C#	2.45.0	2015-02-27	Download	Change log	API docs
Ruby	2.45.0	2015-02-27	Download	Change log	API docs
Python	2.45.0	2015-02-26	Download	Change log	API docs
Javascript (Node)	2.45.0	2015-02-26	Download	Change log	API docs

C# NuGet

NuGet latest release is 2.45.0, Released on 2015-02-27

- [RC](#)
- [WebDriver](#)
- [WebDriverBackedSelenium](#)
- [Support](#)

SafariDriver

SafariDriver now requires manual installation of the extension prior to automation

- Latest release [2.45.0](#)
- [Wiki Page](#)

2. ダウンロードしたファイル「SafariDriver.safariextz」をダブルクリック

Safariに拡張機能がインストールされます。



3. TestCase001.javaを書き換える

書き換え後、importの編成の更新をお忘れなく！

```
1 public void setUp() throws Exception {
2     //driver = new FirefoxDriver(); // コメントアウト
3     // System.setProperty("webdriver.chrome.driver", "Driver/chromedriver
4     // driver = new ChromeDriver(); // 追記
5     driver = new SafariDriver(); // 追記
6     baseUrl = "http://www.yahoo.co.jp/";
7     driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
8 }
```

4. テスト実行！

Safariで弊社のサイトが表示されれば成功！！