## Introduction

This test automation implementation is created by Peter Dobrosi for BetVictor Automation Tester Task on 2015.08.13.

**System Under Test:** Cashier service <http://mock.cashier.bvtpay.com/>

The goal of the test automation task is to cover the functional validation of the following use cases:

* Successful Transaction On Deposit With Non 3DS Payment
* Successful Transaction On Deposit With 3DS Payment
* Failed Transaction On Deposit With 3DS Payment
* Form Validation with incorrect card number

The solution is available on GitHub: <https://github.com/Jumper303/bvt-cashier-automation-testing.git>

The automated tests are implemented in Java, with TestNG and Selenium WebDriver.

**Prerequisites:**

* + Java SDK 1.7 or higher
  + Maven <https://maven.apache.org/>
  + Git
  + Firefox / Chrome (Tested versions: Firefox 39.0.3 / Chrome 44.0.2403.155)

Other dependencies are managed with Maven.

## Setup

**Clone repository:** *git clone* [*https://github.com/Jumper303/bvt-cashier-automation-testing.git*](https://github.com/Jumper303/bvt-cashier-automation-testing.git)

**Download dependencies:** *mvn dependency:resolve*

## Configuration

The following parameters are configured in the ***testng.xml***test suite:

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Type** |
| siteUrl | Base URL of the target site. Can be configured to run on different environments. | Mandatory |
| chromeDriverLocation | Location of the Chrome specific driver. | Mandatory |
| ieDriverLocation | Location of the IE specific driver. | Mandatory |
| seleniumGridHubUrl | URL of the Selenium Grid Hub in case of Grid execution. | Optional |

The framework is prepared for cross browser testing, also with Selenium Grid.

The actual browser type is specified as a system property (One of the followings):

-Dbrowser=Chrome

-Dbrowser=Firefox

-Dbrowser=InternetExplorer

## Test data

The tests are data driven. For simplicity, the parameters are defined in *testng.xml* per test case as xml attributes. TestNG passes these parameters to the test cases in runtime based on the test class and method name. Note: Possible future improvement could be to use a custom data provider.

The optional parameters are marked in the test classes.

The suite contains 5 test cases. (The form validation test is defined twice with two different parameter sets)

To add a new test case, copy-paste a test block and set the class name, method names and parameters per method. The test are executed in the order of definition in the suite, by default.

## Execution

### Command line

There are 2 predefined execution files for the specific browser.

run-with-chrome.bat

run-with-firefox.bat

Or you can execute directly from command line: *mvn clean test -Dbrowser=Firefox*

### Eclipse

Create a TestNG run configuration and select testng.xml for the suite

and add “-Dbrowser=firefox” as a VM argument.

## Report

Maven surefire plugin is used for report generation.

The reports are generated under: **bvt-cashier-automation-testing\target\surefire-reports\**

The **index.html** is automatically opened after the tests are finished.

The reports contains information about each test case with execution parameters, results, execution time and exception stack trace in case of test failure. In case of test failure, a screenshot is automatically created and saved in the report directory with the name of the test case.

## Logging

**Application is logging to the file:** *bvt-cashier-automation-testing\log4j-application.log*

**Configuration file for logging:** *bvt-cashier-automation-testing\src\main\resources\log4j.properties*

## Implementation

The tests are implemented in package: *com.bvt.cashier.test.acceptance.tests*

A PageObject is created for each page to provide interaction with the controls. WebElements are initialized by a factory in the constructor of the **PageBase** super class (by their defined locators in annotation) which also provide some helper methods to handle explicit timeouts. Pages also provides service methods for the specific use cases like **performDeposit**(), **autenticatWith**(). The intermediate steps are defined inside the service methods to keep the validation logic simple in tests. A new browser is created per test class and instantiated for the specified browser in the **TestFixture** base class. An additional helper class is created to wrap around the **DatePicker** component and set card expiry date. The structure of the application contains a static menu and the specific pages are opened in an embedded iframe.

## Results

At the time of execution 2015.08.13 22:00, the tests provided the following results.

|  |  |  |
| --- | --- | --- |
| **Scenario** | **Expected result** | **Actual result** |
| Successful Transaction On Deposit With Non 3DS Payment | Success confirmation page comes up | Pressing “Deposit” with amount “1” redirects to 3D secure authentication page. |
| Successful Transaction On Deposit With 3DS Payment | Success confirmation page comes up | Success confirmation page comes up |
| Failed Transaction On Deposit With 3DS Payment | Unsuccessful Transaction | Unsuccessful Transaction |
| Form Validation with incorrect card number | “I*nvalid card number. Please review the number of digits*” message appears | “I*nvalid card number. Please review the number of digits*” message appears |

