Magento Test Automation Framework Installation Guide

Table of Contents

Introduction

This user's guide provides the information about the Magento test automation framework which can be used for automating the major part of routine processes, like registering or creating customers in the frontend and in the backend, adding addresses to customer accounts, creating websites, web stores, and store views, etc. It provides instructions on installing and configuring all components required for Magento TAF to work properly as well as the instructions on how to use the test automation framework. Also, it describes how users can run a single test, a test suite, or all tests at a time.

Introducing Magento Test Automation Framework

Magento Test Automation Framework (Magento TAF) is a system of software tools used for running repeatable tests against the Magento application being tested.

The automation framework can be used for both performing the actual testing and writing test automation scripts. Test automation scripts created in the framework can cover testing most of the Magento functionality which does not have a relation to an external system.
With MagetoAFW, tests can be run repeatedly and at a high speed.

Magento TAF Organization

Magento TAF Environment

At the base level, Magento Test Automation Framework will require the following software:

- PHP: 5.2.0 or later
- PHPUnit: 3.5.13 or later
- Java SE Development Kit
- Web browsers:
    - Selenium RC 2.0.3
    - Mozilla Firefox 2.x or 3.x
    - Google Chrome
    - Internet Explorer
    - Apple Safari 2.x

Also, this environment includes system requirements of Magento Community edition (see here Magento System Requirements)

Preparing Environment for Using Magento TAF

Current version limitations

- Remote tests executing
- UNIX running
- Usage Firefox profile through configuration file

Version's recommendations
For better speed performance run of a tests:

- Use a local installation of Magento
- Use Firefox 3.6.x or Google Chrome

Useful Information
You must perform the following steps to write and run Automated Test-Scripts (aka Test-Cases):

Recommended File Structure for Windows OS

For more convenient work and clear subject vision, we recommend to use the following file structure:

```
C:\
|__Programs\ - Folder where environment is located
   |-------PHP\ - Folder for PHP installation
   |-------Apache\ - Folder for Apache installation
   |-------MySQL\ - Folder for MySQL installation
   |-------WWW\ - Folder for HTML\PHP content
   |-------magento-afw-x.x.x\ - Directory with unarchived Magento TAF
   |-------Firefox\
```

Installing MySQL Server

Download and install MySQL Server in the recommended location using the link Download MySQL Server 5.5.x

Download Recommendations

- MySQL Community Server
- MSI Installer (32-bit OR 64-bit)

Installing Apache

Download and install Apache WEB Server in the recommended location using the link: Download Apache 2.2.x

Download Recommendations

- MSI Installer (*.msi) (Win Binary)
- Pack with OpenSSL 0.9.8o

Install Recommendations

- Network domain: localhost.com
- Server name: www.localhost.com
- Admin email: admin@localhost.com

Configuration Recommendations

- Change 'DocumentRoot' in the ***httpd.conf*** to "**C:/Programs/www**"
- Add "127.0.0.1 localhost localhost.com www.localhost.com" to ***hosts*** file which located in C:\Windows\System32\drivers\etc\
- RESTART Apache server after previous modifications

Installing PHP

Download and install PHP in the recommended location using the link: Download PHP 5.3.x

Download Recommendations

- MSI Installer (*.msi) (Win Binary)
- Pack with Thread Safe

Install Recommendations

- Web server setup: Apache 2.2.x Module
- Apache configuration directory: C:\Programs\Apache\conf\

Upgrading PEAR

First of all, it is better to set up Windows security level to lowest and grant full permissions to **C:\Programs** (Directory where all necessary software(PHP) will be installed.)

> **Useful Information**
> Change or add the **Windows** PATH environment variables:
>
> **Win 7**: Control Panel \ System and Security \ System \ Environment Variable
> **Win XP**: Control Panel \ System \ Advanced \ Environment Variable
> Environment Variable

You may need to add the following environment variables to the environment.

Additional System Variables

PHP_PEAR_BIN_DIR — C:\Programs\PHP
PHP_PEAR_DATA_DIR — C:\Programs\PHP\PEAR\data
PHP_PEAR_DOC_DIR — C:\Programs\PHP\PEAR\docs
PHP_PEAR_INSTALL_DIR — C:\Programs\PHP\pear
PHP_PEAR_PHP_BIN — C:\Programs\PHP\php.exe
PHP_PEAR_SYSCONF_DIR — C:\Programs\PHP
PHP_PEAR_TEST_DIR — C:\Programs\PHP\PEAR\tests
PHP_PEAR_INCLUDE_PATH — C:\Programs\PHP\PEAR

To update your PEAR installation, request http://pear.php.net/go-pear.phar in your browser and save the output to a local file *go-pear.phar*. Replace *go-pear.phar* under C:\Programs\PHP\PEAR with the new *go-pear.phar* !
After changing the current directory to PHP executable directory (*C:\Programs\PHP in this case*), run command ***php -d phar.require_hash=0 PEAR/go-pear.phar*** on the command line.

```
C:\Programs\PHP>php -d phar.require_hash=0 PEAR/go-pear.phar
```

System PATH Variables

Check that the PATH in the System Variables contains the direct path to PHP C:\Programs\PHP\.

After successfully completing previous command run **pear upgrade** command in the same directory.

```
C:\Programs\PHP>pear upgrade
```

Installing PHPUnit

Open the command line window (or use the previous one under the PHP directory) and then run the following:

```
C:\Programs\PHP>pear channel-discover pear.phpunit.de
C:\Programs\PHP>pear channel-discover pear.symfony-project.com
C:\Programs\PHP>pear channel-discover components.ez.no
C:\Programs\PHP>pear install pear.symfony-project.com/YAML-1.0.2
C:\Programs\PHP>pear install phpunit/PHPUnit_Selenium-1.0.1
C:\Programs\PHP>pear install phpunit/Text_Template-1.0.0
C:\Programs\PHP>pear install phpunit/PHPUnit_MockObject-1.0.3
C:\Programs\PHP>pear install phpunit/PHP_Timer-1.0.0
C:\Programs\PHP>pear install phpunit/File_Iterator-1.2.3
C:\Programs\PHP>pear install phpunit/PHP_CodeCoverage-1.0.2
C:\Programs\PHP>pear install phpunit/DbUnit-1.0.0
C:\Programs\PHP>pear install phpunit/PHPUnit-3.5.15
C:\Programs\PHP>pear install pear.symfony-project.com/YAML-1.0.6
```

Linux note
Ubuntu: To get the PHPUnit's executable you may want to simply install it via APT instead of doing `phpize` (this does not render unnecessary the above step, though):

```
sudo apt-get install phpunit
```

You also may fall into a condition when necessary Symfony component is not installed. Please make sure you have `SymfonyComponents` directory inside your PEAR folder (usually `/usr/share/php`). If you don't have it there, install the component using this command:

```
pear install symfony/YAML
```

`sfYaml` may conflict with default PHP YAML component. In this case you'll need to install the component into separate location and then manually copy over to PEAR folder the `SymfonyComponents` directory. PEAR command to install the component to another location:

```
pear install -R /usr/share/php/sfYAML symfony/YAML
```

Copy necessary folder back to location where it could be found via `include_path`:

```
cp -R /usr/share/php/sfYaml/usr/share/php/SymfonyComponents /usr/share/php
```

Installing Java SE Development Kit

Download and install Java SE Development Kit in the desired location using the link: Download Java SE Development Kit

Installing NetBeans IDE 6.9.1 with the PHP pack

Download and install NetBeans IDE 6.9.1 in the desired location using the link: Download NetBeans IDE 6.9.x

Installing Selenium RC

Download and install Selenium RC 1.0.3 in the recommended location using the link:Download Selenium RC (Now available NEW 2.0b)

> ⚠ The latest available version is 2.x, Download Selenium RC (Now available NEW 2.0b), but automated test cases will probably not work with it.

Installation & Running

1. Download.
2. Unpack.
3. Copy the *selenium-server.jar* file to a location which is convenient for running it through the command line (Example: C:).
4. Run command line.
    a. Run command: C:\>java -jar selenium-server.jar.
    b. Expected behavior will look like the following:

```
c:\>java -jar selenium-server.jar
23:46:04.852 INFO - Java: Sun Microsystems Inc. 19.0-b09
23:46:04.865 INFO - OS: Windows 7 6.1 x86
23:46:04.934 INFO - v2.0 [a2], with Core v2.0 [a2]
23:46:05.347 INFO - RemoteWebDriver instances should connect to:
http://127.0.0.1:4444/wd/hub
23:46:05.348 INFO - Version Jetty/5.1.x
23:46:05.349 INFO - Started HttpContext[/selenium-server/driver,/selenium-server/driver]
23:46:05.351 INFO - Started HttpContext[/selenium-server,/selenium-server]
23:46:05.352 INFO - Started HttpContext[/,/]
23:46:05.502 INFO - Started org.openqa.jetty.jetty.servlet.ServletHandler@19134f4
23:46:05.502 INFO - Started HttpContext[/wd,/wd]
23:46:05.517 INFO - Started SocketListener on 0.0.0.0:4444
23:46:05.517 INFO - Started org.openqa.jetty.jetty.Server@b6ece5
```

    It means that the server successfully started and is ready to work.

    c. Clicking **Ctrl+C** will SHUTDOWN Selenium Server.

Installing TortoiseSVN for Updating Your Magento TAF Copy from the Repository

Download and install GIT

Download and install TortoiseGIT in the desired location using the link: Download TortoiseGIT

Downloading and Installing Magento TAF Project

Choose one of the following ways of downloading Magento TAF project:

- As a source tree from GIT: git clone http://git.magento.com/taf DIR_for_CLONE

Install

- Unpack the downloaded archive according to Recommended File Structure for Windows OS\

Downloading, Installing and Configuring Magento Community Edition

Choose one of the following ways of downloading Magento Community Edition:

- As archived pack: **Download the last build of Magento Community Edition**.
- As a source tree from SVN: **Create SVN checkout by this instruction**

Install and configure

Attention
When Magento Community Edition has been installed you SHOULD disable using of 'Secret Key in URLs':

- Login to Back end
- Open System > Configuration
- Open Advanced > Admin > Security
- Change value of 'Add Secret Key to URLs' to "No"
- Click on 'Save Config' button

- Unpack the downloaded archive
- Install and configure Magento Community Edition according to **Magento Knowledge Base > Installation & Configuration**

Configuring Mozilla FireFox

Download and install Mozilla FireFox 3.6.x in the recommended location using the link: Download Mozilla Firefox 3.6.x

You need to install **Mozilla Firefox 3.x**.

Selenium RC is not compatible with Firefox 4 currently.

Every time that Selenium starts running your test script, a new browsing session will be created.

It means that a new Firefox process will be started with a new "fresh" profile.

Thus, the issues with the previously accepted cookies, stored passwords, etc. are avoided, and every test gets with the same browser environment.

Making Firefox Available to Selenium RC

For successful browser running, Selenium RC server must be able to locate the *firefox.exe* file.

Please add the full path to the system Path variable.

In some operating systems (Win7, WinXP), Selenium RC will not be able to locate *firefox.exe* when the full path contains spaces ("C:\Program Files\Mozilla\Firefox").

Two possible solutions can be recommended:

- Reinstall Firefox to another directory (C:\Programs\Firefox for example).

or

- Use mklink Windows7 built-in utility as follows:

```
mklink /D  C:\Firefox "C:\Program Files (x86)\Mozilla Firefox\"
```

System PATH Variables

Add the direct path to Firefox (or to its link) C:\Programs\Firefox\ to the PATH in System Variables.

Creating Selenium Profile

It is strongly recommended to create a preconfigured Firefox profile for running Selenium tests.
First of all, make sure that all of your Firefox instances are closed.

1. Click 'Windows button' > 'Run' (WinXP) or 'Win+R' (all Windows versions0).
2. Type “firefox.exe -ProfileManager -no-remote”.
3. In the 'Choose User profile' window click “Create Profile”.
4. Click “Next”.
5. Enter a new profile name (e.g. selenium).
6. Select a directory folder to store your new profile.
7. Click “Finish”.
8. Make sure that “Don’t ask at startup” check box is selected.

9. Click “Start Firefox” for the profile being created and configure the settings based on the following suggestions: Suggested Settings for Selenium Profile\ (after starting Firefox with this profile it becomes the default one - to make the “default” profile a default one again, start Firefox with the “default” profile (if needed).

Suggested Settings for Selenium Profile

The following actions are performed in the Firefox toolbar menu.

- Click “View" > "Toolbars” > and then clear the “Bookmarks Toolbar” check box.
- Right-click the toolbar and then click “Customize”.
    - Remove “Google search” by dragging it to the 'Customize Toolbar' window.
    - In the 'Customize Toolbar' window, select the “Use Small Icons” check box and then click “Done”.
- Click “Tools" > "Options” and then set the following in the 'Options' window:
    - “General” tab
        - Set 'Home Page' to “about:blank”.
        - Clear the “Show the Downloads...” option.
    - “Tabs” tab
        - Select "Open new windows in a new tab instead" (Firefox 3.6) OR "a new window" for new pages (oldest FF versions).
        - Clear all warning options.
    - “Content” tab
        - Clear the “Block pop-up windows" option.
    - “Privacy” tab
        - In the "Firefox will" drop-down list select "Use custom settings for history".
            - Clear all “History” options.
    - “Security” tab
        - Clear all “Security” options.
        - Click “Settings” and then clear all warning options.
    - “Advanced” tab
        - “General” subtab
            - Clear the “Use autoscrolling” option under the 'Browsing' group.
        - “Update” subtab
            - Clear all options under "Automatically check for updates to" group.
            - Clear the “Warn me ...” option.
        - "Encryption" subtab
            - "When a server requests my personal certificate" group
                - Select the "Select one automatically" option button
                - Click the "View Certificates" button
                    - "Certificate Manager" window
                        - "Servers" tab
                            - Click the "Add Exception..." button
                            - In the "Location" field, enter your main URL (e.g. https://www.mydomain.com/)
                            - Click the "Get Certificate" button
                            - Make sure that the "Permanently store this exception" check box is selected
                            - Click the "Confirm Security Exception" button
                            - In the "Certificate Manager" window, click "OK"
- Click "OK" in the "Options" window
- Verify that the Cookies are enabled:
    - On the "Web Developer" toolbar select 'Cookies' -> 'Disable Cookies'
        - Make sure that 'All Cookies' is not selected

Also it can be reasonable to perform the following:

- Install useful add-ons (FireBug, Selenium IDE, ScreenGrab, Web Developer, Remember Certificate Exception)
- Accept SSL-certificates for working websites before using

Using Custom Firefox Profile

To use a custom profile for Selenium test, you should use *-firefoxProfileTemplate* parameter.

You must start/restart Selenium Server using the custom profile:

```
c:\>java -jar selenium-server.jar -firefoxProfileTemplate “<Selenium Profile Directory>”
```

Installing Add-ons for Firefox:

- Firebug: allows you to edit, debug, and monitor CSS, HTML, and JavaScript in your application being tested
- Selenium IDE: allows you to record, edit, and debug Selenium tests
- ScreenGrab: saves entire webpages as images
- FirePath: is a Firebug extension that adds a development tool for editing, inspecting, and generating XPath 1.0 expressions, CSS 3 selectors and JQuery selectors
- Web Developer: is an extension which adds various web developer tools to the browser
- Remember Certificate Exception: allows you to automatically add Certificate Exception

You can install add-ons on the "Tools" -> "Add-ons" menu.

Configuring Magento TAF

The **Magento TAF** contains the following two configuration files:

- **browsers.yml** - contains the settings of the Selenium Server.

  

  If you use SVN
  Settings from **browsers.yml** can be overridden by **local.yml** (similar file, which is *IGNORED* by SVN).

  Located in:

  ```
  ..\magento-afw-x.x.x\config\browsers.yml
  ```

  Default content:

```
// List of browsers which can be used
browsers:

  // Default browser settings
  default:
    // Browser name
    browser : '*chrome'

    // Host where it is installed and will run
    host    : '127.0.0.1'

    // Port on the Host where this browser will be available
    port    : 5555

    // Parameter which sets up the frequency of browser running:
    // false - Browser starts every time when new tests are run and shuts down in the end of
each test (recommended).
    // true - Browser starts ONLY before the first test and shuts down after the last one.
    doNotKillBrowsers : false

    // Parameter sets up the count of tests which must be run
    // sequentially before the browser will forcibly shut down.
    browserTimeoutPeriod : 10000

// List of applications which can be tested
applications:

  // Settings of Magento application
  magento: &magento

    // General link to the Frontend
    frontendUrl : 'http://www.localhost.com/magento/'

    // General link to Backend
    adminUrl    : 'http://www.localhost.com/magento/admin/'

    // Login to Backend
    adminLogin  : admin

    // Password to Backend
    adminPassword : admin

  // Pointer to the default application setting.
  // Used when more than one application is described
  // and one of which is required to be used by default
  default: *magento
```

- **phpunit.xml** - Contains the list of test-scripts (test cases) to be run.

```
..\magento-afw-x.x.x\phpunit.xml
```

Default content:

```
<phpunit bootstrap="bootstrap.php"
    colors="false"
    convertErrorsToExceptions="true"
    convertNoticesToExceptions="true"
    convertWarningsToExceptions="true"
    stopOnFailure="false"
    syntaxCheck="false"
    verbose="false"
    strict="true">
    <testsuites>
        <testsuite name="All Tests">
            <directory suffix="Test.php">tests</directory>
        </testsuite>
    </testsuites>
</phpunit>
```

Starting Magento TAF

1. Run Selenium Server.
2. Execute *runtests.bat* (in Windows) through the command prompt for tests running.
3. Executing *runtests.sh* for Linux/*NIX OS is not implemented yet

Running Tests

Adding Magento TAF to NetBeans IDE

- Run NetBeans IDE.
- Click "File" > "New Project".
- Step 1. ‘Choose Project’:
    - Select Categories > ‘PHP’ and projects > ‘PHP Application with Existing Sources’.
    - Click ‘Next’.
- Step 2. ‘Name and Location’:
    - In the ‘Sources Folder’ field, set the path to the project.
    - In the ‘PHP version’ field, select PHP 5.3.
    - Click “Next”.
- Step 3. ‘Run Configuration’:
    - In the ‘Run As’ field, select the ‘Script (run in command line)’ value.
    - There must be the path to the php.exe file in the ‘PHP Interpreter’ field.
    - The ‘Index File’ field is left blank.
    - Click ‘Finish’.
- In the Projects window, right-click the node for your project and then select "Properties".
    - On the ‘Sources’ tab: In the ‘Test Folder’ field, set the path to the folder with tests.
    - On the ‘PHPUnit’ tab: select ‘Use XML Configuration’ and in ‘XML Configuration’ field select *phpunit.xml* file from the project’s folder.
    - Click ‘Ok’.

Adding Magento TAF to PhpStorm IDE

It is supposed that you already have the project with Magento and it is opened.

- Click "**Run**" > "**Edit Configurations**"
- Click "Add New Configuration" (the yellow "**+**" at the top left corner)
- Choose "**PHPUnit**" (don't mess up with "PHPUnit on server")
- Enter "*Name:*" (e.g. "Selenium")
- On "*Configuration*" subtab choose "**XML File**"
- Select path to `phpunit.xml` in *"Use XML configuration file:"*
- Click "**OK**"

Now you can run PHPUnit tests by choosing "Selenium" (or whatever name you've entered) from the dropdown menu near the "Run" and "Debug" buttons and clicking the "Run" button.

Running All Tests

- Using BAT File with phpunit.xml Configuration
  - Edit the phpunit.xml file using the following example.

```
<testsuite name="All Tests">
      <directory suffix=".php">tests</directory>
</testsuite>
```

  - Then run the BAT file from the command prompt.
- Using NetBeans IDE
  - Add **Magento TAF** to NetBeans IDE (see 'Adding **Magento TAF** to the NetBeans IDE').
  - Select the project node in the file tree (Projects window).
  - Right-click this node and then select ‘Test’ or Press 'Alt+F6'.

Running Test Suites

- Using BAT File with phpunit.xml Configuration
  - Edit the phpunit.xml file using the following example.

```
<testsuite name="Test Suite">
      <directory suffix="Test.php">tests/Customer</directory>
      <directory suffix="Test.php">tests/Store</directory>
</testsuite>
```

  - Then run the BAT file from the command prompt.

Running Single Tests

- Using BAT File with phpunit.xml Configuration
  - Edit the phpunit.xml file using the following example.

```
<testsuite name="Single Test">
     <directory suffix="test_file _full_name">file_path_from_test_category</directory>
</testsuite>
```

  - Then run the BAT file from the command prompt.
- Using NetBeans IDE
  - Add **Magento TAF** to NetBeans IDE (see 'Adding **Magento TAF** to the NetBeans IDE').
  - Select the test file (the file node), which you want to run.
  - Right-click this file (in the file tree aka 'Projects window') and then click ‘Run’.
    OR
  - Press 'Shift+F6' to run the test opened in the edit window.

Results of tests running

Command prompt

After executing ***runtests.bet*** in the command prompt you can see current progress:

```
.............F.............F.........................  63 / 228 ( 27%)
...S.......................................................... 126 / 228 ( 55%)
.........I.....E..........I.................................. 196 / 228 ( 75%)
.....................................E...................... 228 / 228 (100%)
Tests: 228, Assertions: 396, Failures: 2, Errors: 2, Incomplete: 2, Skipped: 1.
```

Legend:

- "." - test PASSsed
- "S" - test SKIPped
- "I" - test INCOMPLETE
- "F" - test FAILed
- "E" - error appears during test running

When all tests has been executed, you'll see description of all Failures, Errors, Incompletes, Skips.

Logs

After executing ***runtests.bet*** in the same directory creates log file for this execution.
It contains full logging information in view:

```
Time
     Request to RC server : request
     Response from RC server: response
```

Instructions for Using Magento TAF

After unpacking the downloaded project, you will see the following structure of the **Magento TAF**:

```
__/magento-afw-x.x.x
 |
 |__/config             // Contains files with test configuration
 | |
 | |--browsers.yml      // Contains the list of browsers for testing
 | |--*.yml
 |
 |__/data                // Contains test data for test cases
 | |
 | |--DataSetFile.yml    // Contains files with test data according to specific functionality
 | |--*.yml
 |
 |__/lib                 // Contains libraries which provide the programming support that allows
running tests
 | |__/Mage
 |   |__/Selenium
 |     |--LibraryFile.php
 |     |--*.php
 |
 |__/tests               // Contains the files with test scripts
 | |
 | |__/FunctionalityGroupName1  // example: /Customer/
 | | |--FGN1TestScript.php      // example: Register.php
 | |
 | |__/FunctionalityGroupName*
 |
 |__/uimaps              // Contains page elements UIMaps for all pages that need to be tested
 | |__/admin
 | | |--FunctionalityRelatesUIMAPFile.yml     // for example:  dashboard.yml
 | | |--*.yml
 | |__/front
 |   |--FunctionalityRelatesUIMAPFile.yml
 |   |--*.yml
 |
 |__phpunit.xml         // Contains list of test cases that will be run
```

How to Create Custom TestScript

Creating a custom test script is easy:

- **First**, you need to know what you want to test
- **Second**, create trivial TestCase
- **Third**, create TestScript, based on TestCase and example of code below

Custom TestScript Example

- Write a new TestScript by creating **CreateTest/test.php** with the following structure:

```
class Something_Create_Test extends Mage_Selenium_TestCase
{
 protected function assertPreConditions()     // Preconditions
    {
        $this->loginAdminUser();             // Log-in
        $this->assertTrue($this->checkCurrentPage('dashboard'),
                'Wrong page is opened');
        $this->navigate('manage_stores');    // Navigate to System -> Manage Stores
        $this->assertTrue($this->checkCurrentPage('manage_stores'),
                'Wrong page is opened');
    }

 public function test_Navigation()          // Test case, which verify presence of a controls
on the page
    {
        $this->assertTrue($this->clickButton('create_store_view'),
                'There is no "Create Store View" button on the page');   // action: clickButton
        $this->assertTrue($this->checkCurrentPage('new_store_view'),
                'Wrong page is opened');
        $this->assertTrue($this->controlIsPresent('button', 'back'),
                'There is no "Back" button on the page');
        $this->assertTrue($this->controlIsPresent('button', 'save_store_view'),
                'There is no "Save" button on the page');
        $this->assertTrue($this->controlIsPresent('button', 'reset'),
                'There is no "Reset" button on the page');
    }

 /**
  * Create Store. Fill in only required fields.
  * Steps:
  * 1. Click 'Create Store' button.
  * 2. Fill in required fields.
  * 3. Click 'Save Store' button.
  * Expected result:
  * Store is created.
  * Success Message is displayed
  */
 public function test_WithRequiredFieldsOnly()
    {
        //Data
        $storeData = $this->loadData('generic_store', Null, 'store_name');
        //Steps
        $this->clickButton('create_store');
        $this->fillForm($storeData);
        $this->saveForm('save_store');
        //Verifying
        $this->assertTrue($this->successMessage('success_saved_store'), $this->messages);
        $this->assertTrue($this->checkCurrentPage('manage_stores'),
                'After successful creation store should be redirected to Manage Stores page');
    }
}
```

How to Create Custom DataSet

Custom DataSet is a file which contains sets for form filling.

- Each set corresponds for each variant of form filling.
- Each set contains following pairs: *forms_field_name* : *value*

Custom DataSet Example

Create a new *.yml file under /magento-afw-x.x.x/data, for example with the following structure:

```
generic_store_view:                     // list of fields with test data value
  store_name: Main Website Store
  store_view_name: Test Store View Name
  store_view_code: test_store_view_code
  store_view_status: Enabled

all_fields_store_view:
  store_name: Main Website Store
  store_view_name: Test Store View Name 2
  store_view_code: test_store_view_code_2
  store_view_status: Enabled
  store_view_sort_order: 1
```

How to Create Custom UIMap

Custom UIMap Template

- In general, the uimap template looks like the following:

```
_page_name
  |__mca: URL without BaseURL
  |
  |__title: page title
  |
  |__uimap:
  |  |
  |  |__form: &formLink
  |  |  |
  |  |  |__tabs:
  |  |  |  |
  |  |  |  |__-
  |  |  |  |  |
  |  |  |  |  |__tab_name_1:
  |  |  |  |  |  |
  |  |  |  |  |  |__xpath: tabXpath
  |  |  |  |  |  |
  |  |  |  |  |  |__fieldsets:
  |  |  |  |  |  |  |
  |  |  |  |  |  |  |__-
  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |__fieldset_name_1:
  |  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |  |__xpath: fieldsetXpath
  |  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |  |__buttons:
  |  |  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |  |  |__fieldset_button_name_1: buttonXpath
  |  |  |  |  |  |  |  |  |  |__fieldset_button_name_2: buttonXpath
  |  |  |  |  |  |  |  |  |  |__fieldset_button_name_3: buttonXpath
  |  |  |  |  |  |  |  |  |  |__ ...
  |  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |  |__checkboxes:
  |  |  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |  |  |__checkbox_name_1: checkboxXpath
  |  |  |  |  |  |  |  |  |  |__checkbox_name_2: checkboxXpath
  |  |  |  |  |  |  |  |  |  |__ ...
  |  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |  |__dropdowns:
  |  |  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |  |  |__dropdown_name_1: dropdownXpath
  |  |  |  |  |  |  |  |  |  |__dropdown_name_2: dropdownXpath
  |  |  |  |  |  |  |  |  |  |__ ...
  |  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |  |__links:
  |  |  |  |  |  |  |  |  |  |
```

```
|  |  |  |  |  |  |  |  |  |__link_name_1: linkXpath
|  |  |  |  |  |  |  |  |  |__link_name_2: linkXpath
|  |  |  |  |  |  |  |  |  |__ ...
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |__multiselects:
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |__multiselect_field_name_1: Xpath
|  |  |  |  |  |  |  |  |  |__multiselect_field_name_2: Xpath
|  |  |  |  |  |  |  |  |  |__ ...
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |__fields:
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |__field_name_1: fieldXpath
|  |  |  |  |  |  |  |  |  |__field_name_2: fieldXpath
|  |  |  |  |  |  |  |  |  |__ ...
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |__radiobuttons:
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |__radiobutton_name_1: radiobuttonXpath
|  |  |  |  |  |  |  |  |  |__radiobutton_name_2: radiobuttonXpath
|  |  |  |  |  |  |  |  |  |__ ...
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |__required: [required_field_name1, required_field_name2, ....]
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |__-
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |__fieldset_name_2:
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |__xpath: fieldsetXpath
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |__ ...
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |__ ...
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |__-
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |__fieldset_name_3:
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |__xpath: fieldsetXpath
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |__ ...
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |__ ...
|  |  |  |  |
|  |  |  |__-
|  |  |  |  |
|  |  |  |  |__tab_name_2:
|  |  |  |  |  |
|  |  |  |  |  |__xpath: tabXpath
|  |  |  |  |  |
|  |  |  |  |  |__fieldsets:
|  |  |  |  |  |  |
|  |  |  |  |  |  |__-
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |__fieldset_name_4:
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |__xpath: fieldsetXpath
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |__ ...
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |__ ...
|  |  |
|  |  |__buttons:
|  |  |  |
|  |  |  |__button_name_1: buttonXpath
|  |  |  |__button_name_2: buttonXpath
|  |  |
|  |  |__messages:
|  |  |  |
|  |  |  |__success_saved_message: Xpath
```

```
    |  |  |__success_deleted_message: Xpath
    |  |  |__error_message: Xpath
    |  |  |__ ...
```

Custom UIMap Example

- Create a new *.yml file under /magento-afw-x.x.x/uimaps, using either the UIMap template or creating a custom one using the example:

```
# 'Manage Stores' page
manage_stores:
  mca: system_store/
  title: Stores / System / Magento Admin
  uimap:
    form:
      fieldsets:
        -
          manage_stores:
            xpath: div[@id='storeGrid']
            buttons:
              reset_filter: button[span='Reset Filter']
              search: button[span='Search']
            links:
              select_store_view: td[normalize-space(@class)='a-left last']/a[text()='%NAME%']
            fields:
              website_name: input[@id='filter_website_title']
              store_name: input[@id='filter_group_title']
              store_view_name: input[@id='filter_store_title']
    buttons:
      create_website: button[span='Create Website']
      create_store: button[span='Create Store']
      create_store_view: button[span='Create Store View']
    messages:
      success_saved_store: li[normalize-space(@class)='success-msg']//span[text()='The store has been
saved.']
      success_saved_store_view: li[normalize-space(@class)='success-msg']//span[text()='The store view
has been saved']
      success_saved_website: li[normalize-space(@class)='success-msg']//span[text()='The website has
been saved.']
      success_deleted_store: li[normalize-space(@class)='success-msg']//span[text()='The store has
been deleted.']
      success_deleted_store_view: li[normalize-space(@class)='success-msg']//span[text()='The store
view has been deleted.']
      success_deleted_website: li[normalize-space(@class)='success-msg']//span[text()='The website has
been deleted.']
```

Appendix A. References and Resources

Appendix B. Glossary