Headway

Setup and how-to-use guide

Release 1

Jan, 2023

Content

Chapter 1

Prerequisites

1. Install Java

   https://www.java.com/en/download/help/download_options.html

2. Install Eclipse IDE

   https://www.eclipse.org/downloads/packages/installer

3. Import TestNG Plugin

   https://testng.org/doc/download.html

Chapter 2

Setup Instructions

Step 1. Import as Maven Project in Eclipse IDE

1. Download / Clone Headway in your local machine.
2. Import this project as an existing **Maven** project in Eclipse through
   a. File › Import › Maven › Existing Maven Projects › Next › Browse to selenium framework directory (For ex. *headway*)
   b. Ensure pom.xml file is found
   c. Finish

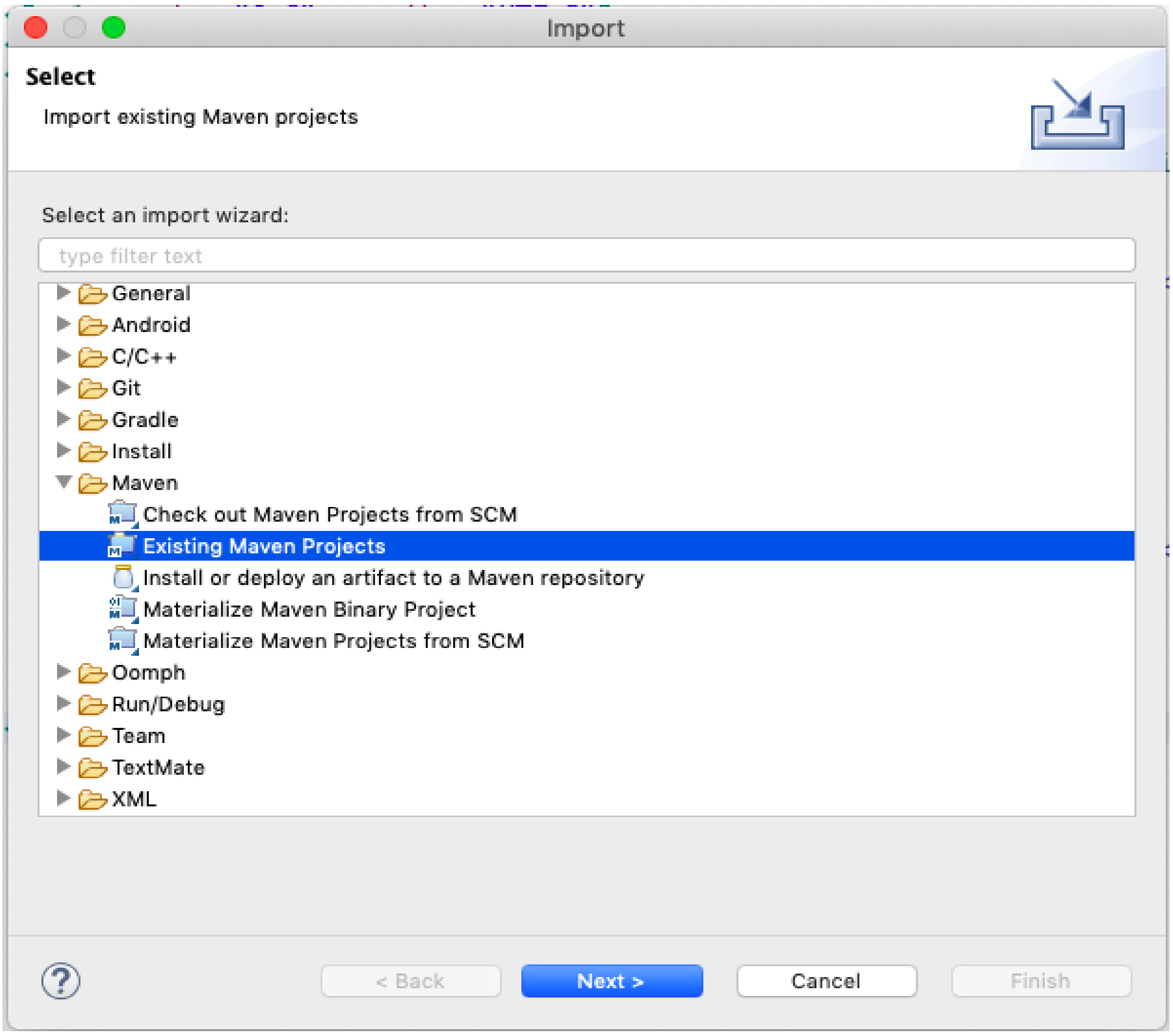
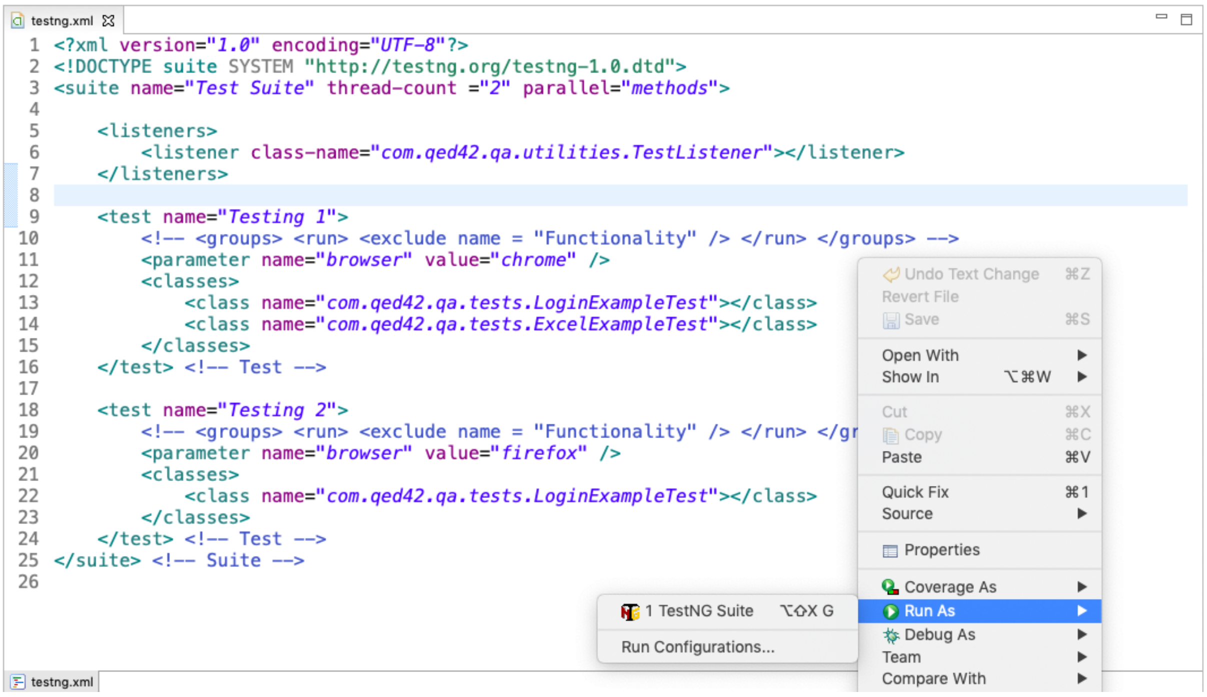

Import
Select
Import existing Maven projects
Select an import wizard:
type filter text
General
Android
C/C++
Git
Gradle
Install
Maven
Check out Maven Projects from SCM
Existing Maven Projects
Install or deploy an artifact to a Maven repository
Materialize Maven Binary Project
Materialize Maven Projects from SCM
Oomph
Run/Debug
Team
TextMate
XML
< Back
Next >
Cancel
Finish

Figure No 2.1

Step 2. Run the testng.xml

Headway provides example tests. Run these tests to confirm that all setup is successful and completed as expected.

- To run a complete test suite: Run the test by right clicking on the TestNG xml file and select **Run As → TestNG Suite**.
- To run individual classes containing tests: Run the test by right clicking on the test class file and select **Run As → TestNG Test.**



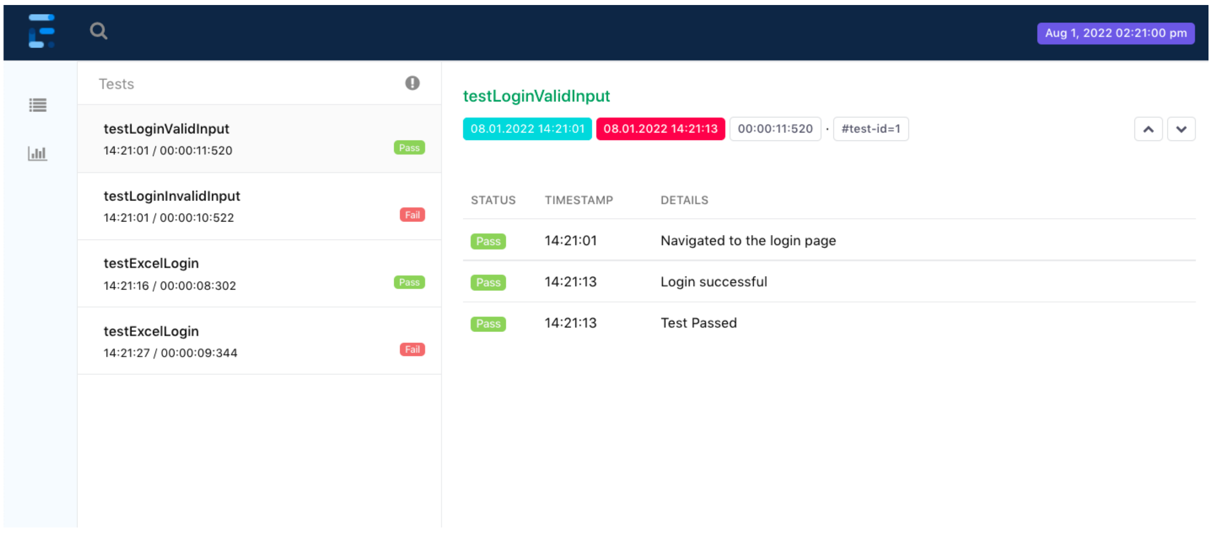
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Test Suite" thread-count ="2" parallel="methods">

    <listeners>
        <listener class-name="com.qed42.qa.utilities.TestListener"></listener>
    </listeners>

    <test name="Testing 1">
        <!-- <groups> <run> <exclude name = "Functionality" /> </run> </groups> -->
        <parameter name="browser" value="chrome" />
        <classes>
            <class name="com.qed42.qa.tests.LoginExampleTest"></class>
            <class name="com.qed42.qa.tests.ExcelExampleTest"></class>
        </classes>
    </test> <!-- Test -->

    <test name="Testing 2">
        <!-- <groups> <run> <exclude name = "Functionality" /> </run> </gr
        <parameter name="browser" value="firefox" />
        <classes>
            <class name="com.qed42.qa.tests.LoginExampleTest"></class>
        </classes>
    </test> <!-- Test -->
</suite> <!-- Suite -->
```

Figure No 2.2

Step 3. Verify the result

1. Goto "/reports" in the project root directory and verify the "Report.html" and "logfile.log" files are generated.
2. Verify the "screenshots" directory is created under the project root directory, when the test fails for the first time. And the screenshot files are captured under this "screenshots" directory for failed tests.

Note: In case files/folder are not generated, refresh the project folder in Eclipse



Aug 1, 2022 02:21:00 pm

Tests

testLoginValidInput 14:21:01 / 00:00:11:520	Pass
testLoginInvalidInput 14:21:01 / 00:00:10:522	Fail
testExcelLogin 14:21:16 / 00:00:08:302	Pass
testExcelLogin 14:21:27 / 00:00:09:344	Fail

testLoginValidInput

08.01.2022 14:21:01 08.01.2022 14:21:13 00:00:11:520 · #test-id=1

STATUS	TIMESTAMP	DETAILS
Pass	14:21:01	Navigated to the login page
Pass	14:21:13	Login successful
Pass	14:21:13	Test Passed

Figure No 2.3

Chapter 3

Page Object Model

3.1. Creating Page Object class

Page Object class is used to store the **WebElements** for a web page and contains **Page methods** which perform operations on those WebElements. A good practice is to have a separate class for every single WebPage.

You need to create page object classes under below directory:

- **Location:** "/src/test/java/com/qed42/qa/pageobjects"

An example page object class 'LoginExamplePage' is provided in the headway framework.

Steps to create new page object class in selenium:

1. Right click on "com.qed42.qa.pageobjects" folder
2. Select New › Class
3. Enter a meaningful class name that is consistent with naming conventions in Java
4. Click on "Finish" button
5. Add below lines of codes to this new page class
    a. This class should extend base class "**BaseDriver**"
    b. Create a instance of **WebDriver**
    c. Add **parameterised** constructor for this class (WebDriver as parameter).
    d. **Initialize** the WebDriver instance inside this constructor

```
public class PageClassOne extends BaseDriver {
        WebDriver driver;

        public PageClassOne(WebDriver driver) {
                this.driver = driver;
        }
}
```

6. Now create objects using `By` interface. For example,

```
public By txtUserName = By.name("userName");
public By txtPassword = By.name("password");
public By btnSubmit = By.name("submit");
```

7. Add methods for actions to be performed on above objects. Make sure to give more realistic names which can be easily mapped with the operation happening in UI. For example, `public void PageMethodOne(String param1, String param2)`
8. Add java documentation for each method by typing /** and then pressing **Enter key**. Similar to below:

```
/**
 * Login with valid credentials
 *
 * @param param1
 * @param param2
 * @throws Exception
 */
public void PageMethodOne(String param1, String param2) throws Exception     {}
```

9. Save the changes using Ctrl+S (Window) or Cmd+S (MAC)

3.2. Creating Page Test class

Page objects themselves should never make verifications or assertions. This is part of your test and should always be within the test's code, never in a page object. Code related to what is being tested should be within the test class.

Steps to create new page test class in selenium:

1. Right click on "com.qed42.qa.tests" folder
2. Select New › Class
3. Enter a meaningful class name that is consistent with naming conventions in Java and represent test cases functionality
4. Click on "Finish" button

5. Add below lines of codes to this new test class

   a. This class should extend base class "**BaseDriver**"

   b. Add **@Listener** annotation before the test class

   c. Add test methods with **@Test** annotation

   d. Create instance of **Page** class inside the test method using below syntax:

   e. Use this instance to call required Page class methods and pass parameter values

   f. Use Report log method to track progress in the report

```
Report.log(Status.PASS, "Test passed");
Report.log(Status.FAIL, "Test failed");
Report.log(Status.INFO, "Test info");
```

```
@Listeners(com.qed42.qa.utilities.TestListener.class)
public class TestClassOne extends BaseDriver {

        @Test
        public void testMethodOne() throws Exception {
                PageClassOne obj = new PageClassOne(getDriver());
                obj1.PageMethodOne("param1_value", "param2_value");
                Report.log(Status.PASS, "Test successful");
        }
```

3.3 Test Data Repository

This repository is used for storing the test resources (that is to be used in tests) such as excel file, json, pdf, images, etc. There is a sample excel file 'testdata.xlsx' under this directory.

Location: "/src/test/resources"

Excel Manager

For reading and writing data with excel files (xlsx and xls), create an instance of 'ExcelManager' class and access methods to read/write data to excel

```
ExcelManager fillo = new ExcelManager();
```

The ExcelManager provides below methods to read/write data to excel file :

1. getAllData() : Performs "SELECT * From ". Returns all data from an excel sheet.
2. getDataWithWhere() : Performs "SELECT * From Where ". Returns data that meet the condition(s) given in where clause, from an excel sheet.
3. insertRowData() : Inserts the data in an excel sheet.
4. updateDataWithWhere() : Updates value of specified field for the records that meet the condition(s) given in where clause.

JSON File Reader

The JSONFileReader class reads a json file using below methods :

1. JSONObject readJson(String filename)
2. JSONArray readJson(String filename, String JSONkey)

Properties File Reader

The PropertiesFileReader class reads properties files and returns an instance of the "Properties" class. It provides one method "read()" and it can be used as below in "Configuration" interface:

```
PropertiesFileReader.read(PROJECT_DIR + "/config.properties");
```

3.4. Updating the existing testng.xml

The **testng.xml** is already provided in the headway's root directory, with sample format supporting parallel and cross browser testing.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Test Suite" thread-count ="2" parallel="methods">

        <listeners>
                <listener class-name="com.qed42.qa.utilities.TestListener"></listener>
        </listeners>

        <test name="Testing 1">
                <!- <groups> <run> <exclude name = "Functionality" /> </run> </groups> ->
                <parameter name="browser" value="chrome" />
                <classes>
                <class name="com.qed42.qa.tests.LoginExampleTest"></class>
                <class name="com.qed42.qa.tests.ExcelExampleTest"></class>
            </classes>
        </test> <!-- Test -->

        <test name="Testing 2">
            <!-- <groups> <run> <exclude name = "Functionality" /> </run> </groups> -->
            <parameter name="browser" value="firefox" />
            <classes>
                <class name="com.qed42.qa.tests.LoginExampleTest"></class>
            </classes>
        </test> <!-- Test -->
    </suite> <!-- Suite -->
```

Update testng.xml as per your project requirement:

1. **Suite name:** Update suite name as per your testing requirements. For example, if you need to run the regression tests, you can update the suite name as "Regression Suite".
2. **thread-count:** This attribute is used to pass the number of maximum threads to be created.
3. **parallel:** The *parallel* attribute on the <suite> tag can take one of following values:
   a. **parallel="methods":** TestNG will run all your test methods in separate threads. Dependent methods will also run in separate threads but they will respect the order that you specified.

      ```
      <suite name="My suite" parallel="methods" thread-count="3">
      ```

   b. **parallel="tests":** TestNG will run all the methods in the same <test> tag in the same thread, but each <test> tag will be in a separate thread. This allows you to group all your classes that are not thread safe in the same <test> and guarantee they will all run in the same thread while taking advantage of TestNG using as many threads as possible to run your tests.

      ```
      <suite name="My suite" parallel="tests" thread-count="3">
      ```

   c. **parallel="classes":** TestNG will run all the methods in the same class in the same thread, but each class will be run in a separate thread.

      ```
      <suite name="My suite" parallel="classes" thread-count="3">
      ```

d. **parallel="instances":** TestNG will run all the methods in the same instance in the same thread, but two methods on two different instances will be running in different threads.

```
<suite name="My suite" parallel="instances" thread-count="3">
```

e. **parallel="none":** TestNG will run all the tests in the non-parallel mode.

```
<suite name="My suite" parallel="none" thread-count="0">
```

4. **listeners:** The TestNG listens to the “TestListener” class that implements ITestListener interface. It is already added in the **testng.xml** file provided in the Headway. You can add more listeners that you implement in your respective project.
5. **test:** A test is represented by <test> and can contain one or more TestNG classes.
6. **parameter:** For any tests, you can pass a browser parameter. In case, nothing passed, then tests will run on chrome browser.

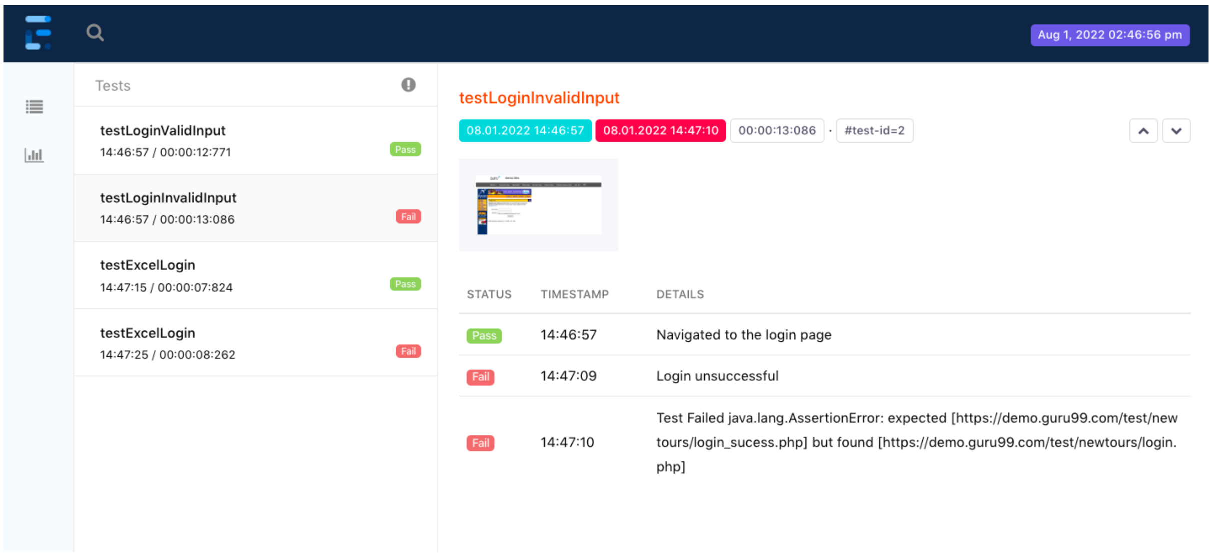
```
<parameter name="browser" value="chrome" />
```

7. **class:** A TestNG class is a Java class that contains at least one TestNG annotation. It is represented by the <class> tag and can contain one or more test methods. You can add classes of which tests are to be run.

3.5. Run the testng.xml

Once the testng.xml file is updated, run tests using testng.xml.

- To run a complete test suite: Run the test by right clicking on the TestNG xml file and select **Run As → TestNG Suite**.
- To run individual class containing tests: Run the test by right clicking on the test class file and select **Run As → TestNG**
- Open “/reports/Report.html” report file and check the test results.
- Check the screenshot files are captured under this “screenshots” directory for failed tests.


Aug 1, 2022 02:46:56 pm

Tests

testLoginValidInput
14:46:57 / 00:00:12:771 Pass

testLoginInvalidInput
14:46:57 / 00:00:13:086 Fail

testExcelLogin
14:47:15 / 00:00:07:824 Pass

testExcelLogin
14:47:25 / 00:00:08:262 Fail

testLoginInvalidInput

08.01.2022 14:46:57 08.01.2022 14:47:10 00:00:13:086 · #test-id=2

STATUS	TIMESTAMP	DETAILS
Pass	14:46:57	Navigated to the login page
Fail	14:47:09	Login unsuccessful
Fail	14:47:10	Test Failed java.lang.AssertionError: expected [https://demo.guru99.com/test/new tours/login_sucess.php] but found [https://demo.guru99.com/test/newtours/login.php]

Figure No 3.5.1