Software Testing

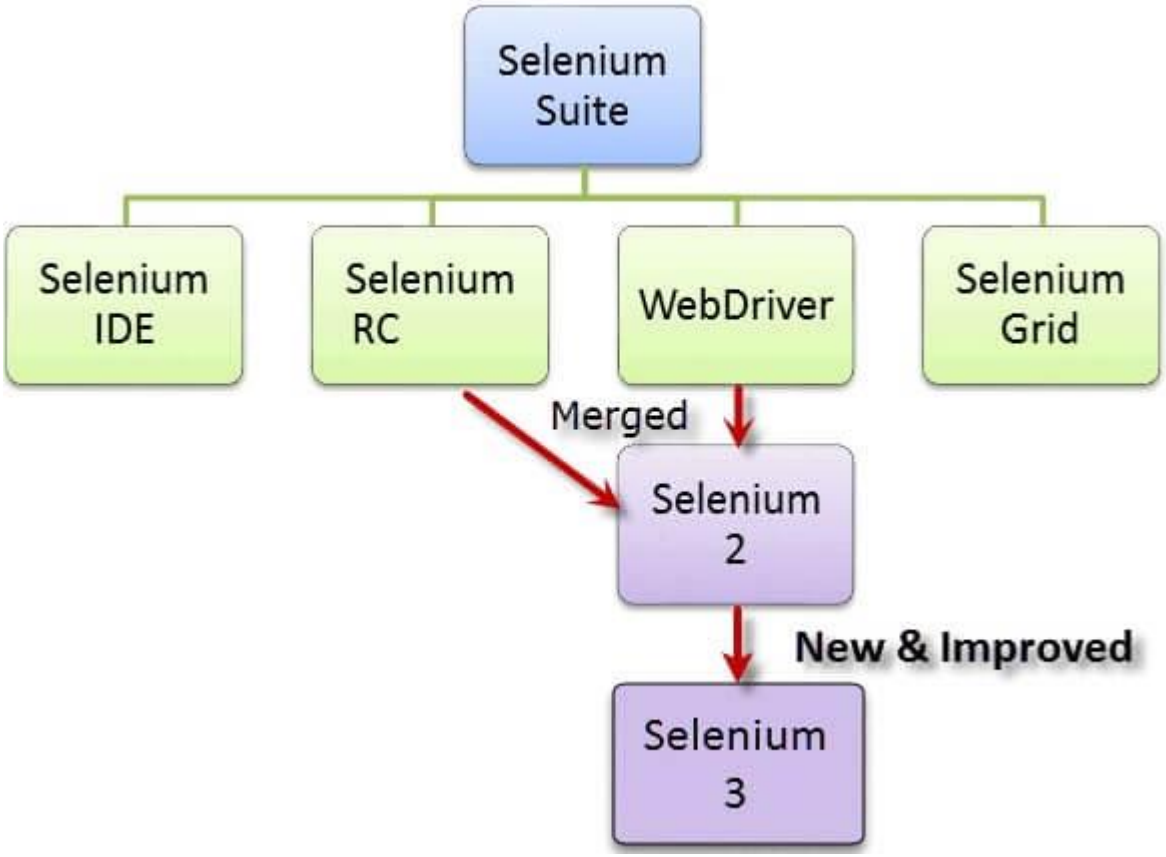
#06 - Automation Test (Selenium)

Herman Kabetta

What is Selenium?

Selenium is a free (open source) automated testing suite for web applications across different browsers and platforms

What is Selenium?



What is Selenium?

What is Selenium?

Advantages :

- Free and Open Source
- Supports a wide variety of programming languages
- Support for cross browser testing

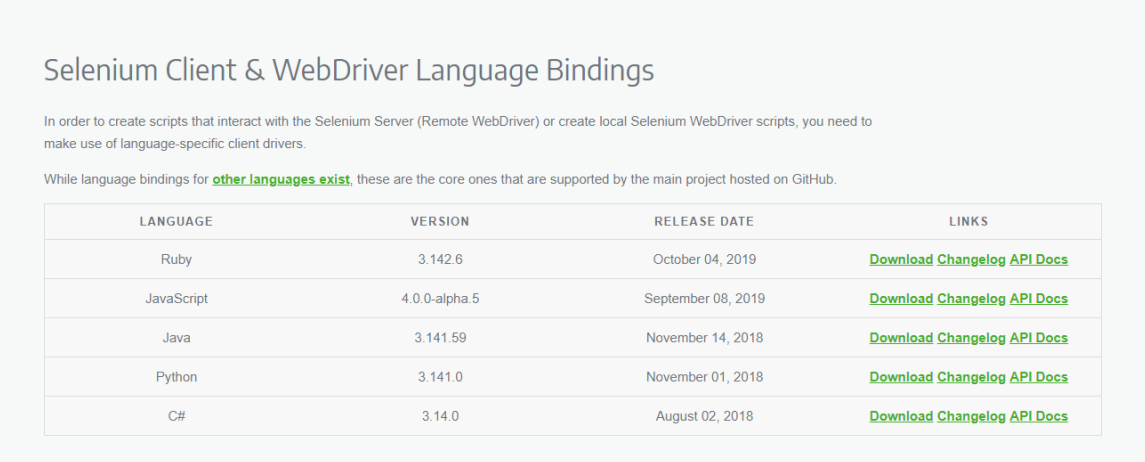
Disadvantages :

- Only for Web Based App
- Lack of professional customer support

Selenium Requirements

- Java v8
- Java IDE (Netbeans, Eclipse, InteliJ, etc.)
- Selenium Client
- WebDriver (ChromeDriver, GeckoDriver, etc.)

https://selenium.dev/downloads/



Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers.

While language bindings for other languages exist, these are the core ones that are supported by the main project hosted on GitHub.

LANGUAGE	VERSION	RELEASE DATE	LINKS
Ruby	3.142.6	October 04, 2019	Download Changelog API Docs
JavaScript	4.0.0-alpha.5	September 08, 2019	Download Changelog API Docs
Java	3.141.59	November 14, 2018	Download Changelog API Docs
Python	3.141.0	November 01, 2018	Download Changelog API Docs
C#	3.14.0	August 02, 2018	Download Changelog API Docs

Using Maven

- Create Maven Project
- Search “selenium” and “webdrivermanager” on mvnrepository.com
- Copy all dependencies from mvnrepository.com to pom.xml
- Clean and Build

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.141.59</version>
    </dependency>

    <!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
    <dependency>
        <groupId>io.github.bonigarcia</groupId>
        <artifactId>webdrivermanager</artifactId>
        <version>3.7.1</version>
    </dependency>
</dependencies>
```

Selenium First Script

Example 1

```
public static void main(String[] args) {
    System.setProperty("webdriver.chrome.driver", "path to chromedriver.exe");
    WebDriver driver = new ChromeDriver();
    String baseUrl = "http://google.com";
    String expectedTitle = "Google";
    String actualTitle = "";
    driver.get(baseUrl);
    actualTitle = driver.getTitle();
    if (actualTitle.contentEquals(expectedTitle)){
        System.out.println("Test Passed!");
    } else {
        System.out.println("Test Failed");
    }
    driver.close();
}
```

FindElement and FindElements

```
WebElement elementName = driver.findElement(By.LocatorStrategy("LocatorValue"));

WebElement loginLink = driver.findElement(By.linkText("Login"));
```

- **findElement()** – finds a single web element and returns as a WebElement object.
- **findElements()** – returns a list of WebElement objects matching the locator criteria.

Locator Strategy can by any of the following values.

- ID
- Name
- Class Name
- Tag Name
- Link Text
- Partial Link Text
- XPATH

FindElement and FindElements

```
List<WebElement> elementName = driver.findElements(By.LocatorStrategy("LocatorValue"));

List<WebElement> listOfElements = driver.findElements(By.xpath("//div"));
```

Locator Strategy can by any of the following values.

- ID
- Name
- Class Name
- Tag Name
- Link Text
- Partial Link Text
- XPATH

FindElement Example

1. Download AUT https://github.com/hermanka/aut
2. Extract to localhost
3. Type code below :

Example 2

```
System.setProperty("webdriver.chrome.driver", "parth\\to\\chromedriver.exe");
WebDriver driver = new ChromeDriver();

driver.get("http://localhost/ajax.html");

// Find the radio button for "No" using its ID and click on it
driver.findElement(By.id("no")).click();

//Click on Check Button
driver.findElement(By.id("buttoncheck")).click();
```

FindElements Example

Example 3

Type code below :

```
System.setProperty("webdriver.chrome.driver", "path\\to\\chromedriver.exe");
WebDriver driver = new ChromeDriver();

driver.get("http://localhost/ajax.html");
List<WebElement> elements = driver.findElements(By.name("name"));
System.out.println("Number of elements:" +elements.size());

for (int i=0; i<elements.size();i++){
  System.out.println("Radio button text:" + elements.get(i).getAttribute("value"));
}

driver.close();
```

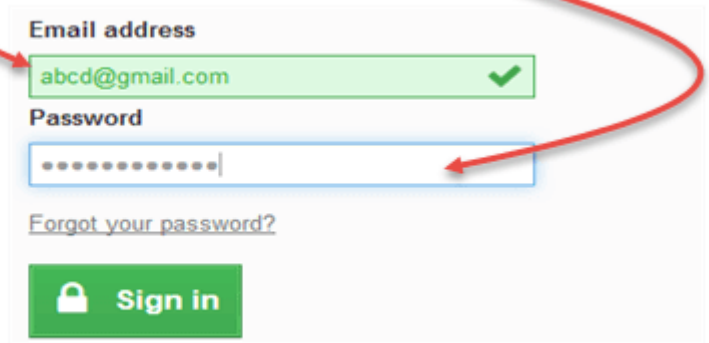
Try It!

- Try to get all text on top menu lists!

Form WebElement

Entering Values in Input Boxes

```
// Get the WebElement corresponding to the Email Address(TextField)
WebElement email = driver.findElement(By.id("email")); 1

// Retrieve the WebElement corresponding to the Password Field
WebElement password = driver.findElement(By.name("passwd")); 2

email.sendKeys("abcd@gmail.com"); 3

password.sendKeys("abcdefghlkjl"); 4
```



1) Find the "Email Address" Text Field using id locator

2) Find the "Password" Field using name locator

3) Enter text into the "Email Address"

4) Enter password into the "Password" using sendKeys()

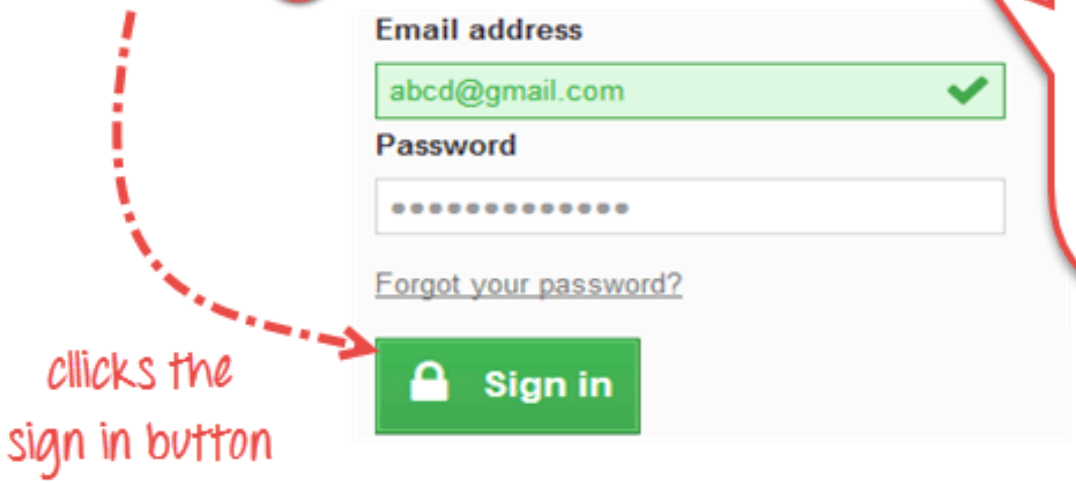
Form WebElement

Button Click

```
WebElement login = driver.findElement(By.id("SubmitLogin"));
login.click();
```

1

2

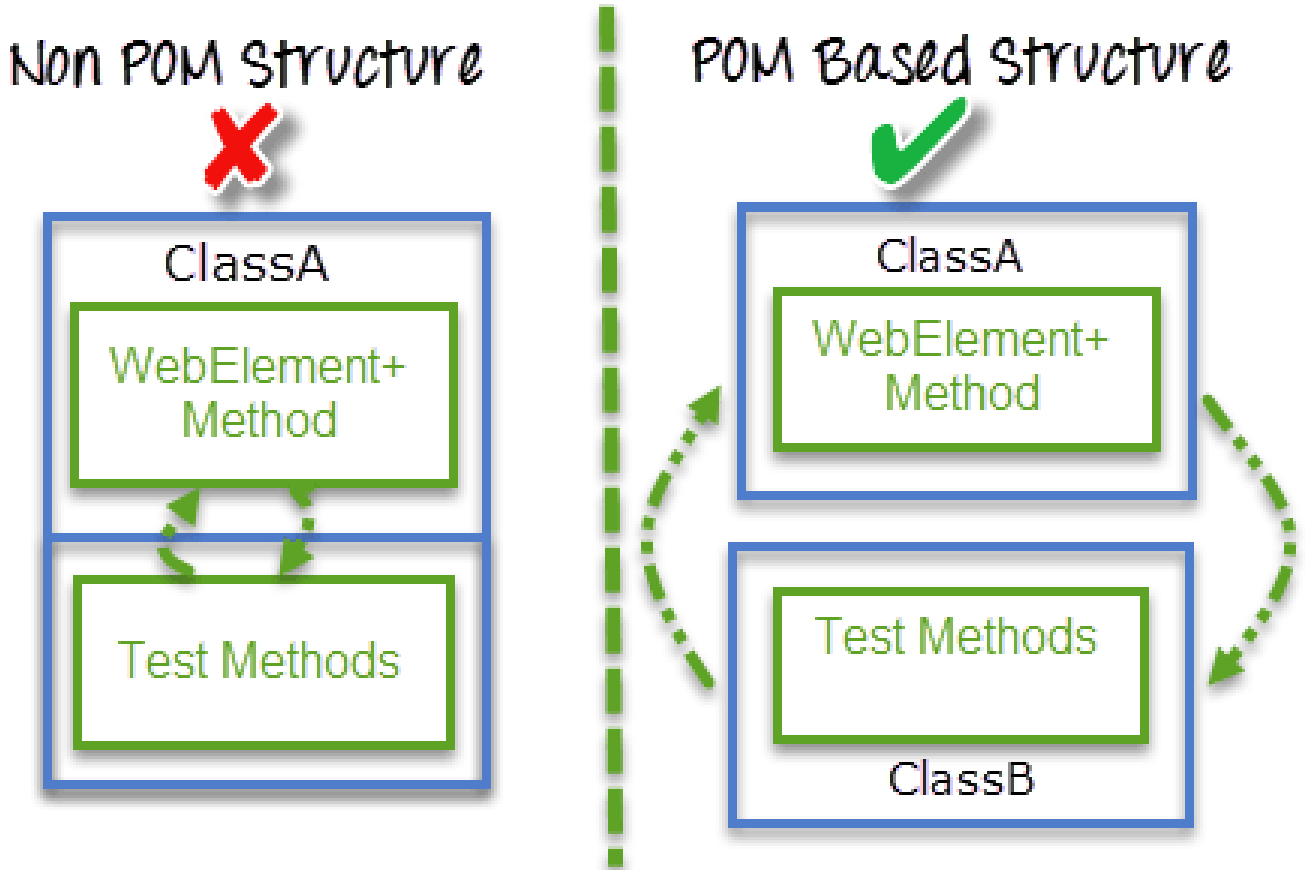


Form WebElement Example

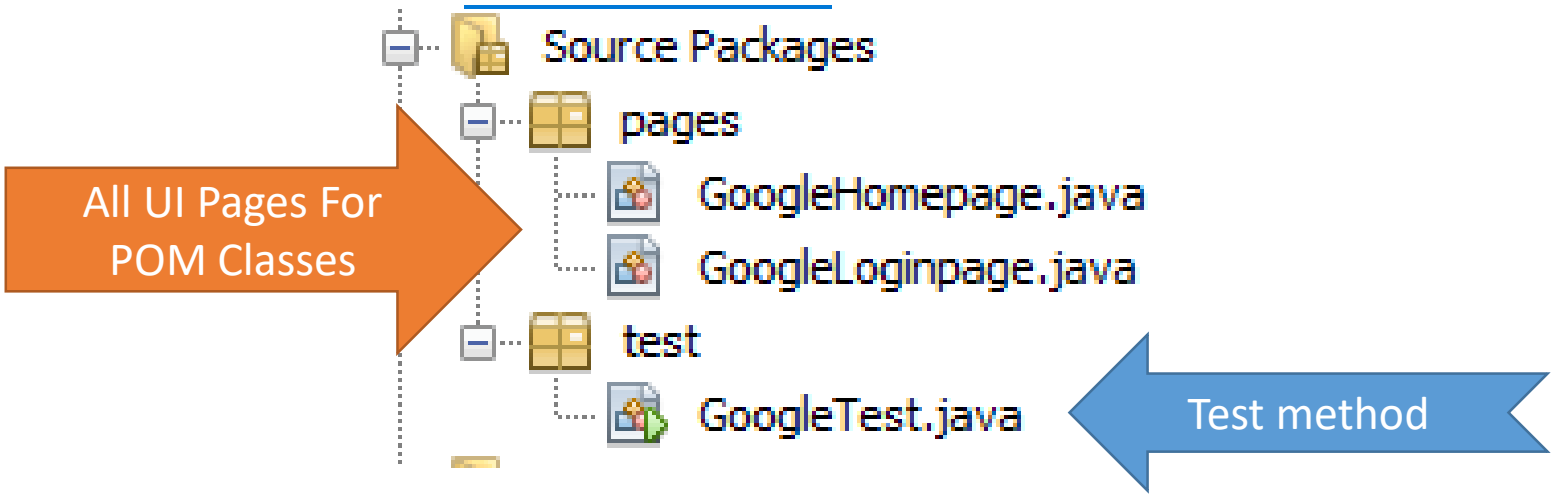
```
System.setProperty("webdriver.chrome.driver", "path\\to\\chromedriver.exe");
WebDriver driver = new ChromeDriver();

driver.get("http://localhost/admin");
WebElement user = driver.findElement(By.name("user"));
WebElement pass = driver.findElement(By.name("pass"));

user.sendKeys("admin");
pass.sendKeys("admin123");
System.out.println("Text Field Set");

WebElement login = driver.findElement(By.name("login"));
login.click();
System.out.println("Login Done with Click");

WebElement welcometext = driver.findElement(By.id("welcome"));
System.out.println(welcometext.getText());
driver.close();
```

Example 4

Page Object Model



Page Object Model



Page Object Model

Example 5

```
public class GoogleHomepage {

    private static WebElement element = null;

    public static WebElement searchText(WebDriver driver){
        element = driver.findElement(By.name("q"));
        return element;
    }

    public static WebElement searchButton(WebDriver driver){
        element = driver.findElement(By.name("btnK"));
        return element;
    }
}
```

Page Object Model

Example 5

```
public class GoogleTest {

    public static void main(String[] args){
        WebDriverManager.firefoxdriver().setup();
        WebDriver driver = new FirefoxDriver();
        driver.get("http://google.com");

        GoogleHomepage.searchText(driver).sendKeys("Selenium tutorial");
        GoogleHomepage.searchButton(driver).click();

    }
}
```

Page Object Model

Example 5 v2

```
public class GoogleHomepage {

    private WebElement element = null;
    private WebDriver driver;

    private By inputSearch = By.name("q");
    private By buttonSearch = By.name("btnK");

    public GoogleHomepage(WebDriver driver){
        this.driver = driver;
    }

    public void searchText(String text){
        this.driver.findElement(inputSearch).sendKeys(text);
    }

    public void searchButton(){
        this.driver.findElement(buttonSearch).sendKeys(Keys.RETURN);
    }
}
```

Advantages of POM

- Page Object Pattern says operations and flows in the UI should be separated from verification. This concept makes our code cleaner and easy to understand.
- The Second benefit is the object repository is independent of test cases, so we can use the same object repository for a different purpose with different tools.
- Code becomes less and optimized because of the reusable page methods in the POM classes.
- Methods get more realistic names which can be easily mapped with the operation happening in UI. i.e. if after clicking on the button we land on the home page, the method name will be like 'gotoHomePage()'.

Try it!

- Try to POM-ify example 4!

TestNG

- TestNG is an automation testing framework
- TestNG is inspired from JUnit which uses the annotations (@)
- Default Selenium tests do not generate a proper format for the test results

TestNG Features

- Generate the report in a proper format including a number of test cases runs, the number of test cases passed, the number of test cases failed, and the number of test cases skipped.
- Multiple test cases can be grouped more easily by converting them into testng.xml file. In which you can make priorities which test case should be executed first.
- The same test case can be executed multiple times without loops just by using keyword called 'invocation count.'
- Using testng, you can execute multiple test cases on multiple browsers, i.e., cross browser testing.
- The testing framework can be easily integrated with tools like Maven, Jenkins, etc.
- Annotations used in the testing are very easy to understand
  ex: @BeforeMethod, @AfterMethod, @BeforeTest, @AfterTest

TestNG Features

- TestNG simplifies the way the tests are coded.

Usual structure
(somewhat difficult to read)

```
public class myclass {

    public static String baseUrl = "http://newtours.demoaut.com/";
    public static WebDriver driver = new FirefoxDriver();

    public static void main(String[] args) {
        driver.get(baseUrl);
        verifyHomepageTitle();
        driver.quit();
    }

    public static void verifyHomepageTitle() {
        String expectedTitle = "Welcome: Mercury Tours";
        String actualTitle = driver.getTitle();
        try {
            Assert.assertEquals(actualTitle, expectedTitle);
            System.out.println("Test Passed");
        } catch (Throwable e) {
            System.out.println("Test Failed");
        }
    }
}
```

TestNG structure
(easier to understand)

```
public class SampleTestNGTest {
    public String baseUrl = "http://newtours.demoaut.com/";
    public WebDriver driver;

    @BeforeTest
    public void setBaseURL() {
        driver = new FirefoxDriver();
        driver.get(baseUrl);
    }

    @Test
    public void verifyHomepageTitle() {
        String expectedTitle = "Welcome: Mercury Tours";
        String actualTitle = driver.getTitle();
        Assert.assertEquals(actualTitle, expectedTitle);
    }


    @AfterTest
    public void endSession() {
        driver.quit();
    }
}
```

TestNG

Installation

- Right click on the “libraries” folder
- Choose “Add Library”, then add TestNG Library
- If using maven,
  - Search “TestNG” on mvnrepository.com
  - Copy the xml dependency to your pom.xml
  - Clean and Build

Maven | Gradle | SBT | Ivy | Grape | Leiningen | Buildr

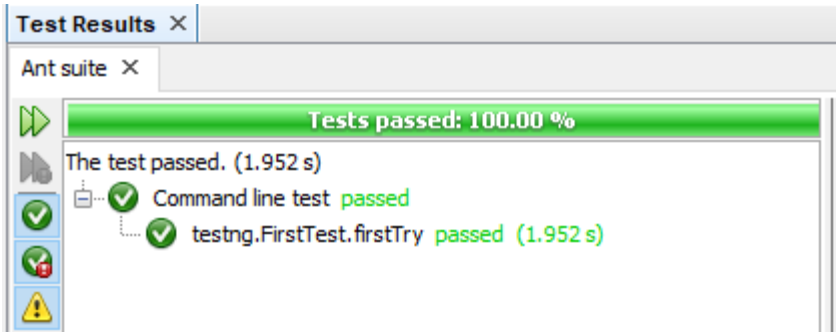
```
<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.1.0</version>
    <scope>test</scope>
</dependency>
```

TestNG

RIght click > New > Other > TestNG Test Case

Example 6

```
WebDriver driver = null;

@BeforeTest
public void setUp() {
    System.setProperty("webdriver.chrome.driver", "path\\to\\chromedriver.exe");
    driver = new ChromeDriver();
}

@Test
public void firstTry(){
    driver.get("http://localhost:8080/ajax.html");
    String expectedTitle = "Ajax Test";
    String actualTitle = driver.getTitle();
    Assert.assertEquals(actualTitle, expectedTitle);

}

@AfterTest
public void tearDown(){
    driver.close();
    driver.quit();
}
```



Try it!

- Try to add secondTry() to example 6!

```
@Test
public void secondTry(){
    driver.get("http://localhost:8080/ajax.html");
    String expectedTitle = "Ajax";
    String actualTitle = driver.getTitle();
    Assert.assertEquals(actualTitle, expectedTitle);

}
```

TestNG - Suite Test

A test suite is a collection of test cases intended to test a behavior or a set of behaviors of software program.

RIght click > New > Other > TestNG Test Suite

```
<test name="All tests for AutoTester">
    <classes>
        <class name="testng.FirstTest" />
    </classes>
</test>
```

Try it!

- Try to add SecondTest Class in TestNG Suite

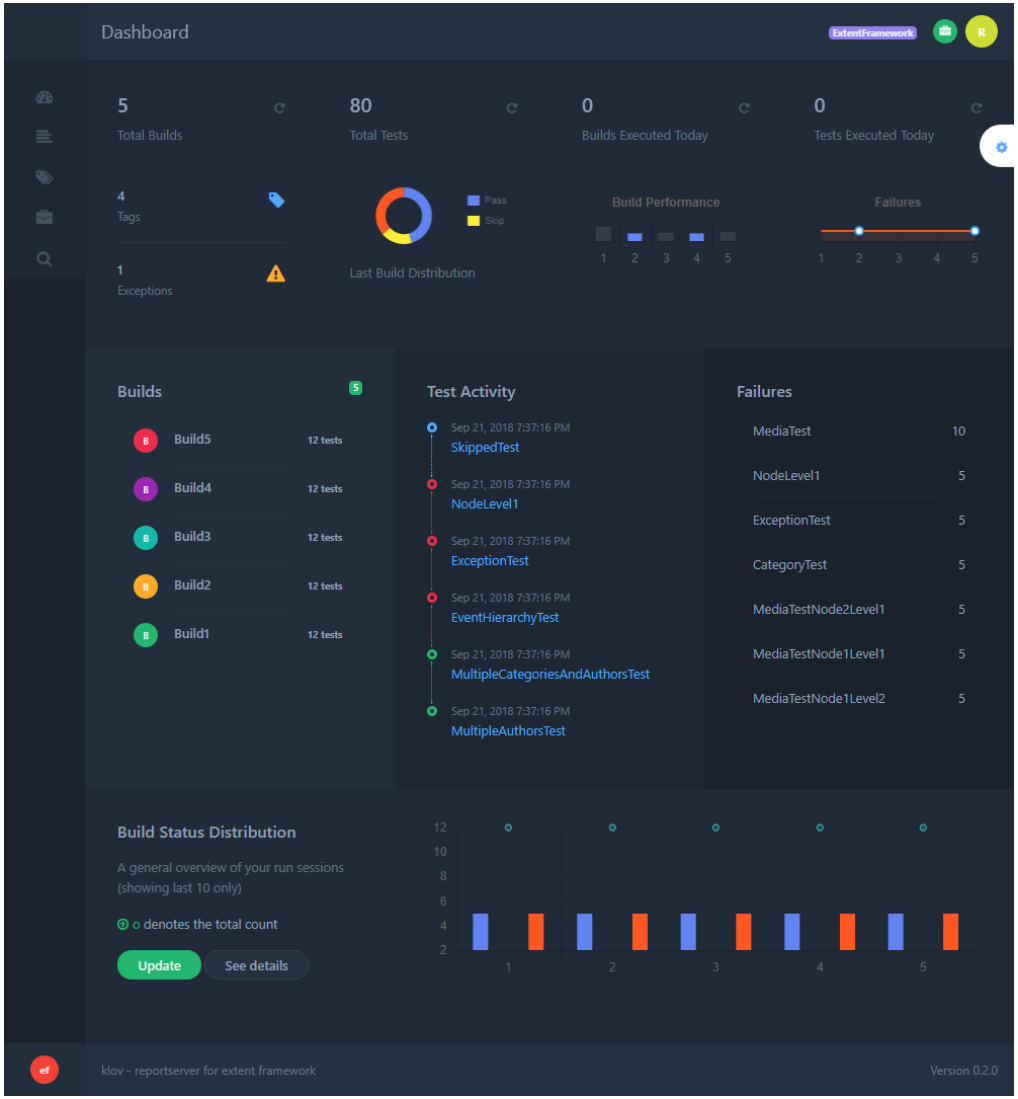
```
<test name="All tests for AutoTester">
    <classes>
        <class name="testng.FirstTest" />
        <class name="testng.SecondTest" />
    </classes>
</test>
```

More about:
http://testng.org/doc/documentation-main.html#testng-xml

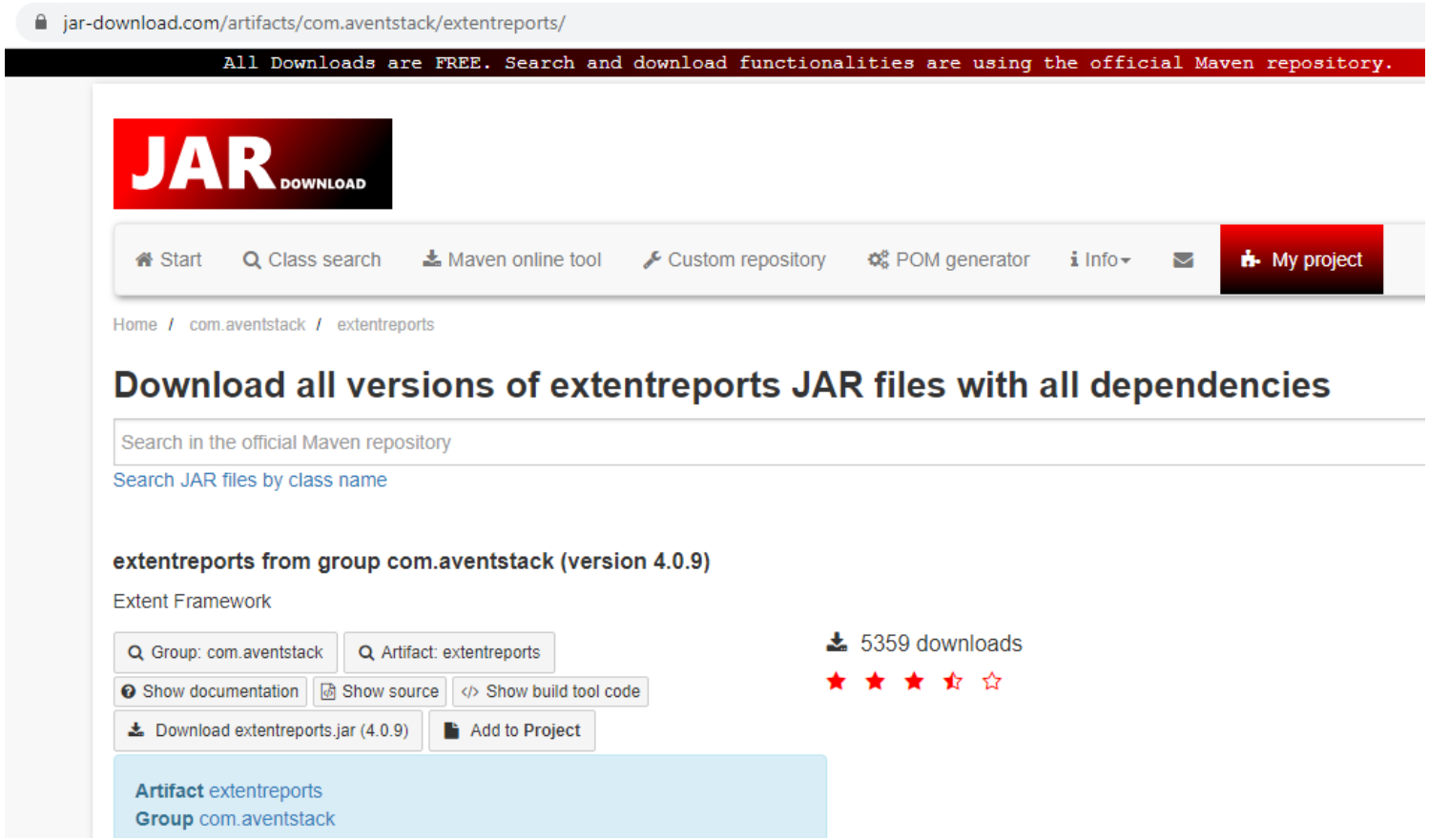
Try it Again!

Try to add more cases to TestNG Suite based on Example 4!

Extent Reports



Extent Reports



Download and import to project

Extent Reports

```
public static void main(String[] args) {
    ExtentHtmlReporter htmlReporter = new ExtentHtmlReporter("D:\\reports.html");
    ExtentReports extent = new ExtentReports();
    extent.attachReporter(htmlReporter);
    ExtentTest test1 = extent.createTest("Test case 1","Test to validate google search");
    System.setProperty("webdriver.chrome.driver", "path\\to\\chromedriver.exe");
    ChromeDriver driver = new ChromeDriver();
    test1.log(Status.INFO, "Starting test case 1");
    driver.get("https://google.com");
    test1.pass("Navigate to google.com");
    WebElement q = driver.findElement(By.name("q"));
    q.sendKeys("Selenium tutorials");
    test1.pass("Entered text in search box");
    q.sendKeys(Keys.RETURN);
    test1.pass("Pressed keyboard enter key");
    driver.close();
    driver.quit();
    test1.pass("Browser closed");
    test1.info("Test completed");
    extent.flush();
}
```