Best Practices and tips for selenium using BDD

1. Use explicit or fluent wait instead of sleep in code for faster run of execution

🗙

Thread.sleep(1000);

✔

Fluent wait example

Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)

.withTimeout(20, TimeUnit.SECONDS)

.pollingEvery(5, TimeUnit.SECONDS)

.ignoring(NoSuchElementException.class);

WebElement aboutMe= wait.until(new Function<WebDriver, WebElement>() {

public WebElement apply(WebDriver driver) {

return driver.findElement(By.id("about\_me"));

}

});

Explicit Wait example

WebDriverWait wait = new WebDriverWait(WebDriverRefrence,20);

WebElement aboutMe;

aboutMe= wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("about\_me")));

1. Do not write XPATH/CSS directly in your code. Always create a separate file to store all XPATH/CSS together.

🗙

driver.findElement(By.xpath(//\*[@id=])).click();

✔

driver.findElement(By.xpath(ElementPath.getProperty("Editorial\_ "))).click();

1. Don’t rely on specific Driver implementation, always create a framework which works with different browsers Firefox, IE and Chrome.

🗙

System.setProperty("webdriver.ie.driver", "IEDriverServer.exe");

driver = new InternetExplorerDriver();

✔

if (device.equalsIgnoreCase("CHROME"))

{

driver = new ChromeDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

}

else if (device.equalsIgnoreCase("IE"))

{

System.setProperty("webdriver.ie.driver", "IEDriverServer.exe");

driver = new InternetExplorerDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

}

else if (device.equalsIgnoreCase("FIREFOX"))

{

driver = new FirefoxDriver();

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

}

1. Always create multiple branches of master in git. Do not directly push your code in Master. It should be reviewed and merged by SME of the project.
2. Always do tagging in git after successful build.
3. Do not write direct values in your code, always create a separate file to store all the values, which can further be modified by layman user directly without help of coder.

🗙

driver.findElement(By.xpath(property.getproperty("ElementXpath")).sendkeys(“This is Hard coded name”);

✔

driver.findElement(By.xpath(property.getproperty("ElementXpath")).sendkeys(excel.getproperty“Name”);

1. To ensure error free code, test/Execute your code before commiting it to your central repository.
2. Always capture screenshots for failure at any step of code.

public void embedScreenshot(Scenario s) throws IOException {

if(s.isFailed()) {

byte[] screenshot = ((TakesScreenshot)driver).getScreenshotAs(OutputType.BYTES);

s.embed(screenshot, "image/png");

}

1. Run your test on every commit through integration tool like Jenkins to avoid any failure in future.
2. If Click is not working – use JavaScript.

public void JavascriptClick(WebElement elemen) {

((JavascriptExecutor)driver).executeScript( "arguments[0].click();", element) );

}

1. Use Tags

Tags are a great way to organize your features and scenarios in nonfunctional ways. You could use @small, @medium and @large, or maybe @hare, @tortoise, and @sloth. Using tags let you keep a feature’s scenarios together structurally, but run them separately. It also makes it easy to move features/scenarios between groups, and to have a given feature’s scenarios split between groups.

The advantage of separating them this way is that you can selectively run scenarios at different times and/or frequencies, i.e. run faster scenarios more often, or run really big/slow scenarios overnight on a schedule.

Tagging has uses beyond separating scenarios into groups based on how fast they are:

* When they should be run: on @checkin, @hourly, @daily
* What external dependencies they have: @local, @database, @network
* Level: @functional, @system, @smoke

✔

Example

@PROD @DEV @IPHONE

Scenario: User is able to download image of particular story

Given User has access to Application

When I login with my credentials Username and Password

Then I should see All Page

And able to verify the notes and able to download the Image

1. Make BDD Scenarios Independent and Deterministic
2. Reuse Step Definitions- Reuse step definitions that are not feature specific
3. Use backgrounds wisely

Background steps are run before each scenario. If you use the same steps at the beginning of all scenarios of a feature, put them into the feature’s Background. But take care that you don’t put too many steps in there as your scenarios may become hard to understand.

1. Do not create multiple scenarios for different input values.

@QA @TEST @DEV @IPHONE

Scenario: order should be consistent

Given I entered all required data to create

And I navigate to the all page

And I navigate to unknown tab

When Add details roles, name and units

|vvt|Sablok, Neharika|abc|

|Photographer|Manchanda, Mohit|def|

|xrs|Manchanda, Mohit|mnp|

|JPM|Manchanda, Mohit|def|