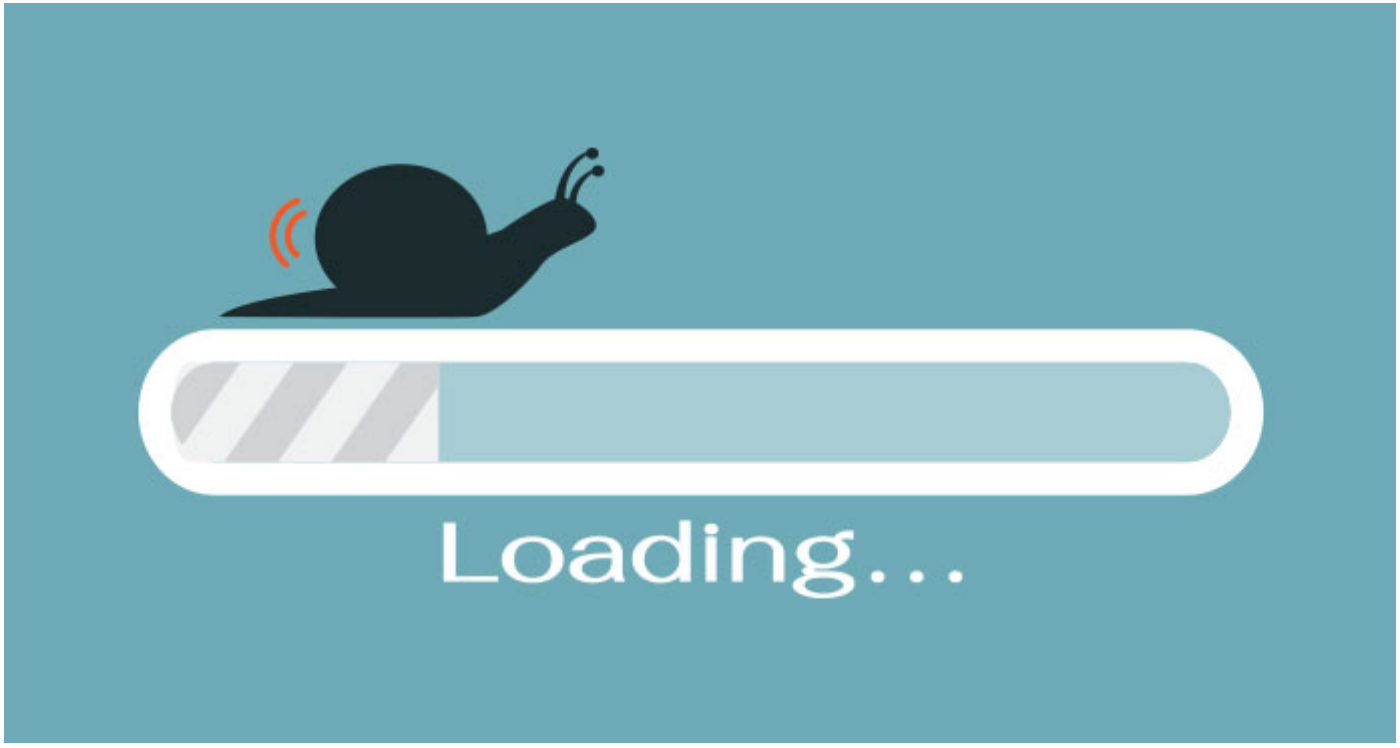


Selenium WebDriver – How To Handle Page Synchronization Using Awaitility

11 Comments / Articles, Best Practices, Selenium / By vIns / September 25, 2017

Overview:

One of the most common challenges with test automation is dealing with test flakiness! People who have done some automation or developers who have worked on multi-threaded application development can only understand the pain! It is painful because there are 2 different processes (test scripts in your favorite language & the browser driver) trying to talk to each other! So one might have to wait for other to be in desired state!

Usual Wait Techniques:

Page Synchronization is not very difficult as it might sound! There are many ways to handle the situations.

- **Thread.sleep()** – This is something like Alcohol! We know that Consuming Alcohol is injurious to health. But we still have that! Similarly we all use 'Thread.sleep' in our script once in a while even if we know that it is brittle and but we never accept to others we use it! It might get the job done sometimes, but affects the performance of the scripts very badly & does not produce consistent test results! [That is what flakiness is, right?]
- **Implicit Wait** – WebDriver is configured with specific duration for timeout to find an element. The timeout is used throughout the test script! Again, it is not a good idea & it affects the performance of the test scripts very badly!

Spring WebClient with Feign
Spring Data R2DBC Query By Example

Selenium WebDriver - How To Test REST API

Introducing PDFUtil - Compare two PDF files textually or Visually

JMeter - How To Run Multiple

- **Explicit Wait** – We use ExpectedConditions api along with WebDriverWait to make the driver wait for specific conditions like element becomes visible, expected text to appear etc! Compared to the above approaches, this is definitely better! But sometimes we might end up writing many lines of code using this lib.
- Fluent Wait

I personally use the **Arquillian Graphene** library which has much better synchronization methods in fluent API style – which makes the code looks neat and clean. You can check here.

If you do not want to use Arquillian for some reason, then you can take a look at this Awaitility library. Lets see how we could handle the page synchronization using this library in this article.

Using Awaitility:

- Include below dependencies in your pom file.

```
<dependency>
    <groupId>org.awaitility</groupId>
    <artifactId>awaitility</artifactId>
    <version>3.0.0</version>
</dependency>
<dependency>
    <groupId>org.awaitility</groupId>
    <artifactId>awaitility-proxy</artifactId>
    <version>3.0.0</version>
</dependency>
```

Thread Groups in Multiple Test Environments

Selenium WebDriver - Design Patterns in Test Automation - Factory Pattern

Kafka Stream With Spring Boot

JMeter - Real Time Results - InfluxDB &

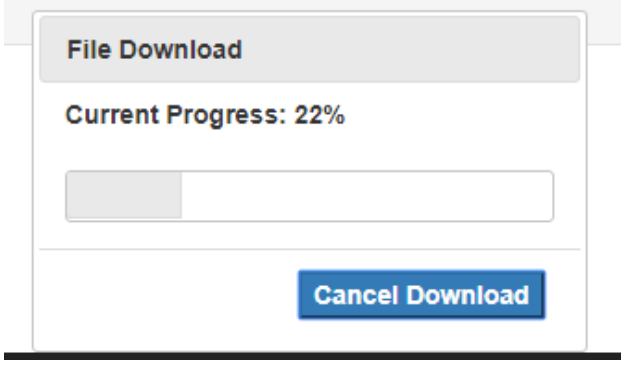
- Add the below static imports to use the Awaitility effectively.

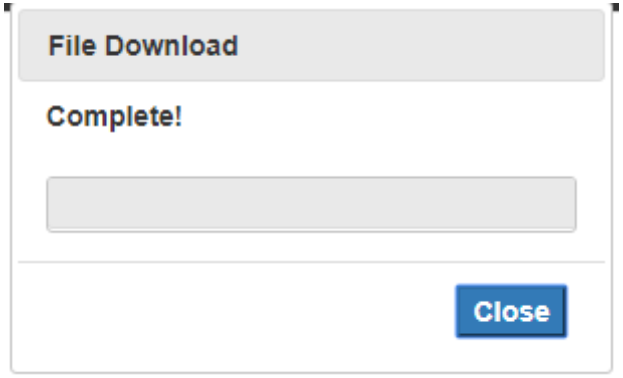
```
import static org.awaitility.Awaitility.*;
import static org.awaitility.Duration.*;
import static java.util.concurrent.TimeUnit.*;
import static org.hamcrest.Matchers.*;
import static org.junit.Assert.*;
```

Awaitility with WebDriver:

Waiting For Element To Appear:

In the below example – As soon as you click on a 'Download' button – a File Download progress dialog appears. We are waiting for the File download progress bar to show the **Complete!** message.





Grafana - Part 1 - Basic Setup

JMeter - Distributed Load Testing using Docker

JMeter - How To Test REST API / MicroServices

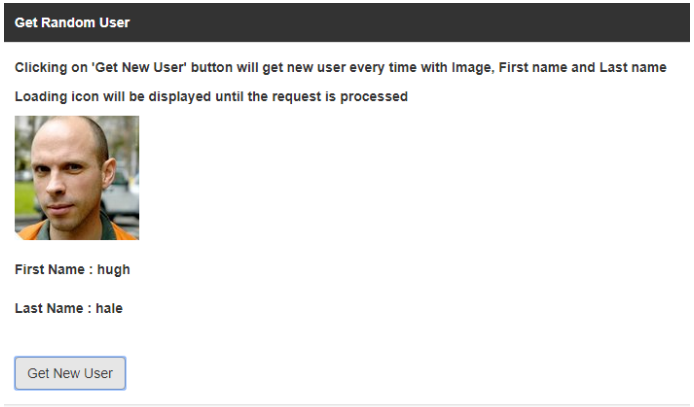
JMeter - Property File Reader - A custom config element

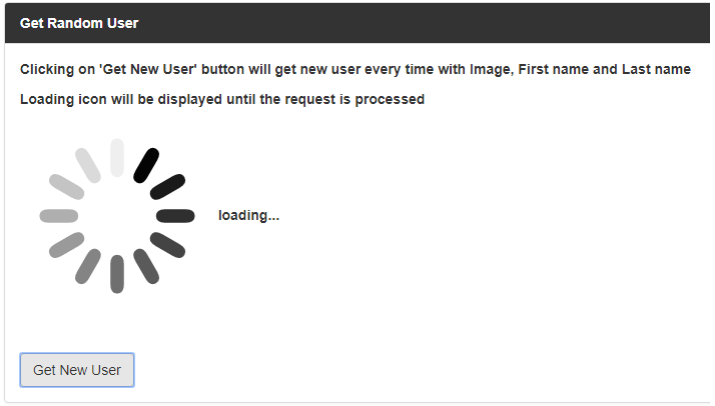
Selenium WebDriver - How To Run Automated

```
driver.get("http://www.seleniumeasy.com/test/jquery-download-progress-bar-de
driver.findElement(By.id("downloadButton")).click();
WebElement progress= driver.findElement(By.cssSelector("div.progress-label")

// Wait for the download to complete - max 20 seconds
await("Download did not complete within 20 seconds").atMost(20, TimeUnit.SEC
                                                      .until(progress::getText

//Did you notice? The robust wait statement is just an one-liner! It looks n
System.out.println("DONE!!");
```

Waiting For Element To Disappear:

In the below example, whenever you click on the 'Get New User' button, an AJAX request is made to get a new user information. A loading message is shown as shown below till the request is complete. So if we need to automate this flow – you need to wait for the loading message to disappear.





Tests Inside A Docker Container - Part 1

Categories

Architecture (60)
Arquillian (9)
Articles (201)
AWS / Cloud (17)
AWS (4)
Best Practices (75)
CI / CD / DevOps (51)
Data Stream / Event Stream (27)

```
driver.get("http://www.seleniumeasy.com/test/dynamic-data-loading-demo.html"
WebElement newUserBtn = driver.findElement(By.id("save"));
WebElement loadingElement = driver.findElement(By.id("loading"));

// Get a new User
newUserBtn.click();

//Wait for the loading to disappear
await("Wait for new user to load").atMost(5, SECONDS)
                                  .until(loadingElement::getText, not("loadi
```

- I clicked on the 'Get New User' 20 times in a loop – It worked like charm.

```
for(int i=0; i<20; i++){
    newUserBtn.click();
    await("Wait for new user to load").atMost(5, SECONDS)
                                      .until(loadingElement::getText, not("l
}
```


Database (9)

Design Pattern (40)

Architectural Design Pattern (25)

Factory Pattern (1)

Kubernetes Design Pattern (18)

Strategy Pattern (1)

Distributed Load Test (9)

Docker (24)

ElasticSearch (2)

EMail Validation (1)

Framework (104)

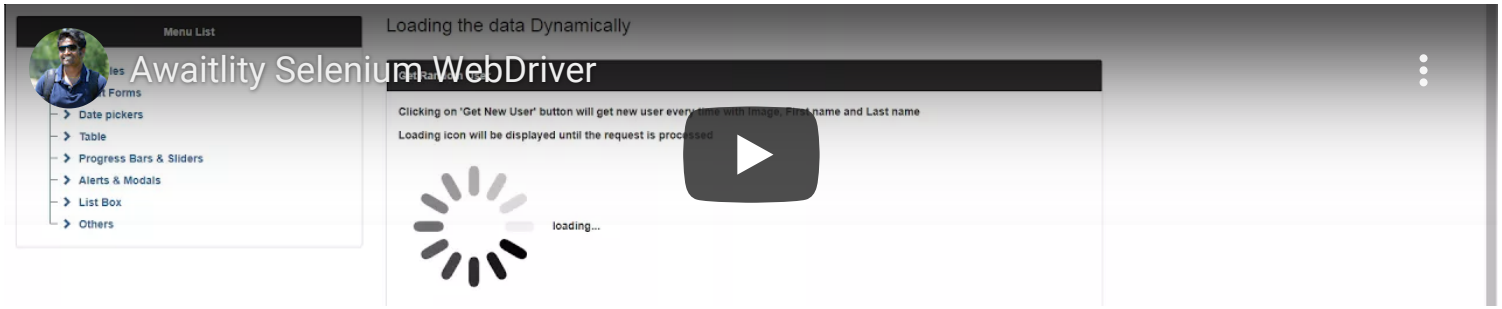
Functional Test Automation (83)

Puppeteer (1)

QTP (10)

Selenium (76)

Extend WebDriver (11)



Ignoring Exceptions:

In the below example, we wait for the alert to appear – when we try to switch to alert when it is not present, it throws an exception – we handle that using 'ignoreExceptions()'

```
driver.get("http://www.seleniumframework.com/Practiceform/");
WebElement alertButton = driver.findElement(By.id("timingAlert"));

//To check if alert is present - By the way, predicates are cool!
Predicate<WebDriver> isAlertPresent = (d) -> {
    d.switchTo().alert();
    return true;
};

//click - alert appears after 3 seconds
alertButton.click();

//wait for 5 seconds - ignore alert not present exception
```

Ocular (2)
Page Object Design (17)
Report (8)
Selenium Grid (10)
TestNG (7)
gRPC (14)
Java (78)
Guice (2)
Reactor (39)
Jenkins (17)
Kafka (9)
Kubernetes (8)
Linkerd (2)
Maven (7)
messaging (11)
MicroService (74)
Mongo (4)
Monitoring (13)
FileBeat (1)
Grafana (5)

```
await("Wait for alert").atMost(5, SECONDS)
                       .ignoreExceptions()
                       .until(() -> isAlertPresent.test(driver));

driver.switchTo().alert().accept();
```

At Least Check:

Lets say we have a requirement to show the alert only after 2 seconds – It should not appear before that. Awaitility can verify that too!

```
//alert should not appear within 2 seconds - max 5 seconds - ignore alert no
await("Wait for alert").atLeast(2, SECONDS)
                       .and()
                       .atMost(5, SECONDS)
                       .ignoreExceptions()
                       .until(() -> isAlertPresent.test(driver));
```

Periodic Check:

Lets assume that – a button click triggers a **Welcome EMail** which you need to validate as part of your automated test. As you know, It might take sometime to receive an email – we need to check for an email at regular intervals until the given timeout period.

```
await().atMost(2, MINUTES)          // max wait
      .pollDelay(5, SECONDS)        // do not check immediately - wait for 5
```


InfluxDB (7)
Kibana (2)
Multi Factor Authentication (2)
nats (4)
Performance Testing (44)
Extend JMeter (5)
JMeter (43)
Workload Model (2)
Little's Law (1)
Web Scraping (1)
Protocol Buffers (14)
r2dbc (4)
Reactive Programming (38)
Redis (8)
rsocket (7)
Slack (3)
SMS (1)
Spring (72)

```
        .pollInterval(10, SECONDS)    // check every 10 seconds
        .until(() -> {                // until email is received
           return EMailUtil.hasEmailReceived("Welcome Email");
        });
```

Downloading File:

Sometimes, you click on a link to download a file. Once the download starts, then you can not control via selenium API. You might need to check if the download is complete.

```
Path filePath = Paths.get(".", "filename");
await().atMost(1, MINUTES)
        .ignoreExceptions()
        .until(() -> filePath.toFile().exists());
```

Summary:

Awaitility is a cool library with a lot of features. This was a quick tutorial. I might not be able to cover all the possible combinations in this. But hopefully you got the idea! I would suggest you to read more on this wiki. You could also combine this lib with ExpectedConditions API to write a better readable code.

All the best for you to have a robust test suite with Awaitility!

Happy Testing & Subscribe 🙂

Spring Boot (62)
Spring Data (11)
Spring WebFlux (61)
Udemy Courses (5)
Utility (20)
WebSocket (2)

Share This:

« Selenium WebDriver – How To Query HTML Web Table

Selenium WebDriver – How To Automate Responsive Design Testing »

11 thoughts on "Selenium WebDriver – How To Handle Page Synchronization Using Awaitility"

asim

October 6, 2017 at 6:04 AM

Hi

Can such a nice and comprehensive automation related essay be available using combination of selenium+python.

I mean your articles are too good and they are about actual problems instead of some typical "selenium tutorial, but these are for JAVA ppl only :(.

Reply

Dam Dao

October 6, 2017 at 6:38 AM

Thanks so much. Useful for me.

Reply

Umbrella

January 26, 2018 at 10:31 AM

Thanks for this wonderful discussion.

Reply

Sandeep

March 3, 2018 at 5:55 AM

Thank's for sharing your knowledge. It will help us to solving problems related to automation testing using selenium web Driver.

Reply

vlns

March 3, 2018 at 4:23 PM

Glad that you find it useful.

Reply

Deepu

March 20, 2018 at 2:01 AM

Great article. Thanks for sharing awaitility tools with us

Reply

nevil

July 7, 2018 at 3:51 PM

Nice post..useful stuff ..thanks to introducing to awaititylity library .

Reply

Vinod

August 6, 2018 at 11:26 PM

Good stuff

Reply

santhosh

December 9, 2018 at 3:18 PM

really very useful content ,thanks a lot

Reply

romi

March 14, 2019 at 7:29 PM

Very useful for me. Thanks a lot!

Reply

Rajaselvan

August 21, 2019 at 4:35 AM

Very useful information. Thanks a lot

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

☐ Save my name, email, and website in this browser for the next time I comment.

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

Post Comment