

How to use a Retry class to repeat actions before failing the test

Using Selenium, most of the time we do integration tests at the minimum. Many of our UI tests are either system-level tests or end-to-end tests. This poses a challenge to our testing. Our system may not always be as stable as we would want it to be. It's easier with unit tests - just mock out whatever integration you do not require for your test and your set to go.

Things happen at the integration and system level - services may be down, and external servers may be slow, a refresh is needed from time to time. While it is OK from the manual test point of view, it is a big problem from the test automation point of view.

There is no magic answer to solve this issue. We can minimize it as much as possible. Having a retry mechanism helps. In this section, we will demonstrate a Retry mechanism for failed service or a page refresh error.

Let's start by describing the full test and then we will walk through it step by step:

```
1  @Test
2  @DisplayName("Test to retry operation")
3  public void retryTest(VerisoftDriver driver){
4      driver.get("https://www.wikipedia.org/");
5      driver.findElement(By.id("searchInput")).sendKeys("Test Automation");
6      new Actions(driver).sendKeys(Keys.ENTER).build().perform();
7
8      Retry retry = new Retry(driver, 3, 1500, TimeUnit.MILLISECONDS);
9      retry.attempt(new Attemptable() {
10         @Override
11         public void attempt() throws Throwable {
12             //replace string with "Test automations" to see the retry in action
13             if (driver.getTitle().contains("Test automation"))
14                 return;
15             else
16                 throw new NotFoundException("Could not find element - retrying");
17         }
18
19         @Override
20         public void onAttemptFail() {
21             driver.navigate().refresh();
22         }
23     });
24 }
```

The mechanism is essentially a class named `Retry` that receives an [anonymous class](#) as a parameter (using an interface).

The purpose of this test is to search for a term and then expect to see the term at the title of the search result page. We will focus on the `Retry` part:

```
1  Retry retry = new Retry(driver, 3, 1500, TimeUnit.MILLISECONDS);
```

▼ Click here to expand...

First, we will create an object from the `Retry` class. Its constructor receives a `WebDriver` object, the total number of attempted we will want to perform if the operation fails (in this case - 3), and how many seconds to wait between attempts (in this case 1500 milliseconds).

```
1  retry.attempt(new Attemptable() {
```

▼ Click here to expand...

We will use the method `attempt`, which receives an interface named `Attemptable`. The `Attemptable` interface **does not and will not** have a class that implements it, since the situation we wish to retry changes from one situation to another. So we will implement its methods as an anonymous class.

The `Attemptable` interface has 2 methods we will need to implement:

1. `attempt`
2. `onAttemptFail`

```
1 @Override
2 public void attempt() throws Throwable {
3     if (driver.getTitle().contains("Test automation"))
4         return;
5     else
6         throw new NotFoundException("Could not find element - retrying");
7 }
```

▼ Click here to expand...

After hitting enter in the search text box, we expect a new page will open with the title of the term we were looking for, in our case - "Test automation". If the title contains the phrase "Test automation", the `attempt` method is successful and the test continues. If the phrase was not found, a `NotFoundException` will be thrown, taking us automatically to the `onAttemptFail` method.

```
1 @Override
2 public void onAttemptFail() {
3     driver.navigate().refresh();
4 }
```

▼ Click here to expand...

If the title was not found, let's refresh the page. It will then wait for the amount of time we defined earlier (1500 milliseconds) and try again. If after 3 attempts the result, we expected, the `NotFoundException` that was thrown in the `attempt`.