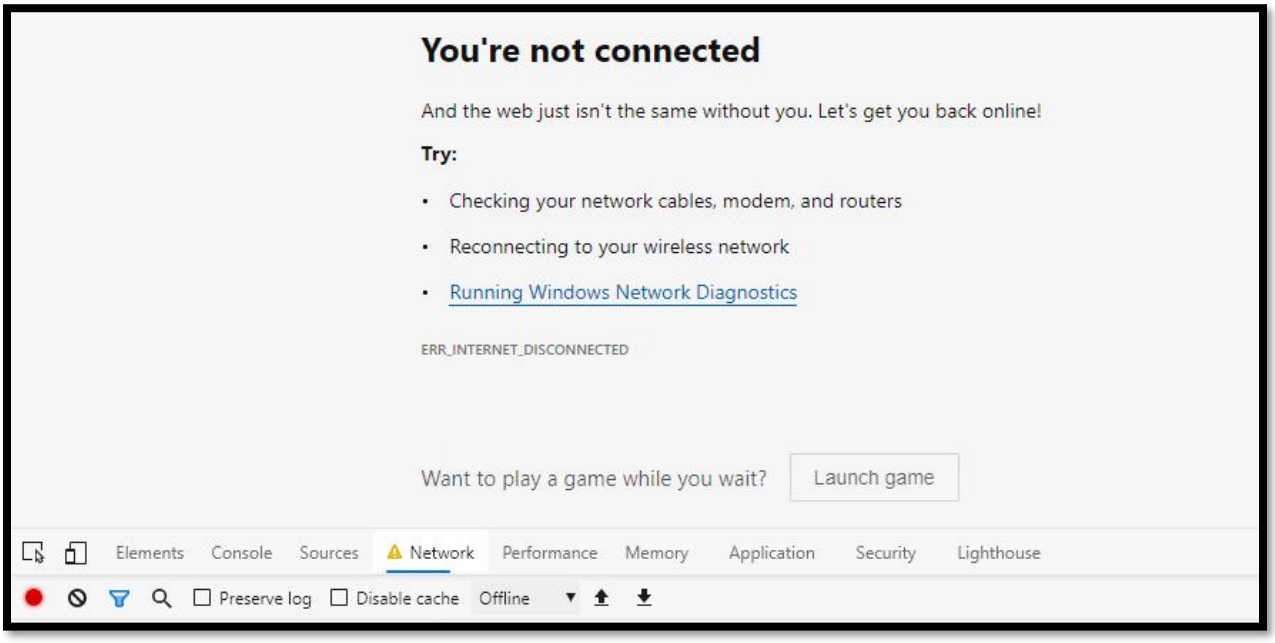
Rex Jones II

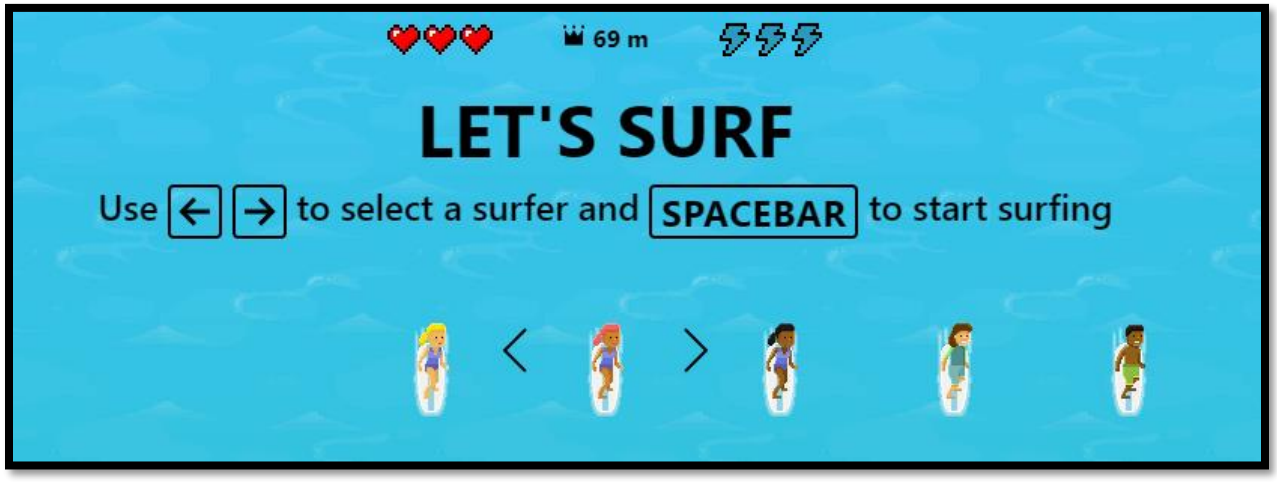
Selenium 4 How To Automate Offline Network

In this session, we are going to enable the network to be offline. It's very good when an application has functionality to continue working offline. There are many reasons for an application to be offline such as bad weather, a power outage, or even driving into a dead zone.

Here's our Application Under Test. Most of the times, I use Chrome but not this time. For some reason Chrome does not include the same offline functionality as Microsoft Edge. I'm going to right click, Inspect, and go to the Network tab. The Network Throttling drop down has different options. We want Offline. Refresh and notice the message says "You're not connected" and it also shows INTERNET_DISCONNECTED. A lot of applications stop right here and do not let us continue. However, we can still perform an action while offline. Do you see Want to play a game while you wait and this Launch game button?



Inspect this launch game button and we see the id value is game-button. Now, at this point, we are going to do the same steps in our automation. In our test, we are going to set the test to be offline then click this Launch game button. It's a surf game.

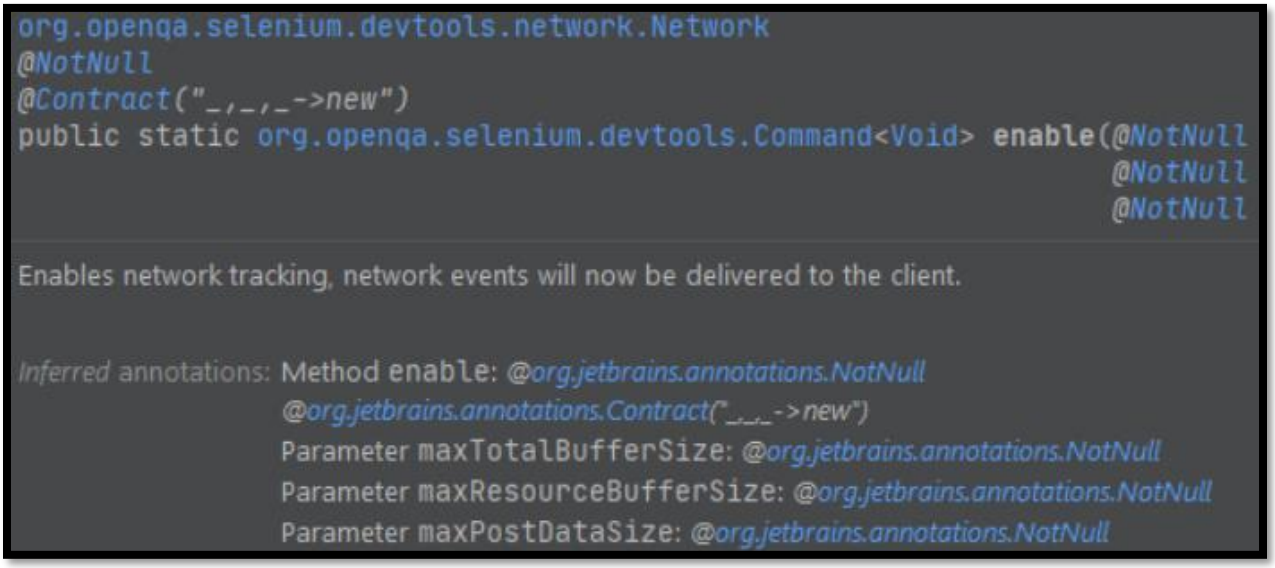


For our Test Script, I already have EdgeDriver, DevTools, and the setup method for EdgeDriver. It's from the last session when we slowed down the network connection using 3G and accessed LinkedIn.

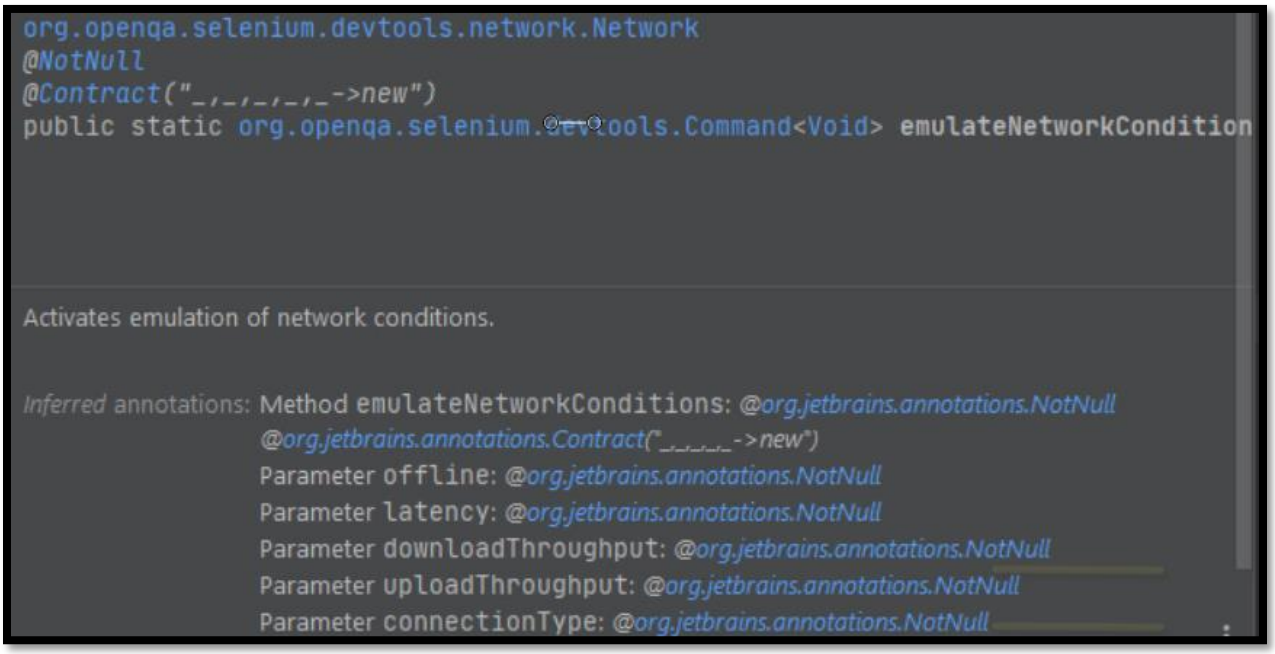
Now, let's write @Test public void enableOfflineNetwork () { }. We always begin by creating the session. devTools.createSession(). This will start a session in the Edge browser. Next, is to send a command so we can enable the network. devTools.send(Network.enable()) The enable method has 3 parameters and all 3 of them are optional. Therefore, we write Optional.empty(), Optional.empty(), Optional.empty().

```
@Test
public void enableOfflineNetwork () {
  devTools.createSession();
  devTools.send(Network.enable(
          Optional.empty(),
          Optional.empty(),
          Optional.empty()));
```

If I hover enable, the description shows "Enables network tracking, network events will now be delivered to the client". Those 3 empty parameters are maxTotalBufferSize, maxResourceBufferSize, and maxPostDataSize.



After enabling the network, we are going to send a command to emulate the network. devTools.send(Network.emulateNetworkConditions()). Hover the method to see it activates an emulation of the network conditions using 5 parameters: offline, latency, downloadThroughput, uploadThroughput and connectionType. I don't see it here but Connection Type is optional.



Now, let's go ahead and continue with our test script by writing the value for offline and that value will be true. This means yes, the network will be offline. 10 is the value for latency. Latency is sometimes called lag. It's a delay in communication across the network. 100 for the downloadThroughput and 50 for the uploadThroughput. Throughput refers to the amount of data that transfers from the source to

the destination. For connection type, we write Optional.of(ConnectionType.G) and we see the different G connections but let's select WIFI.

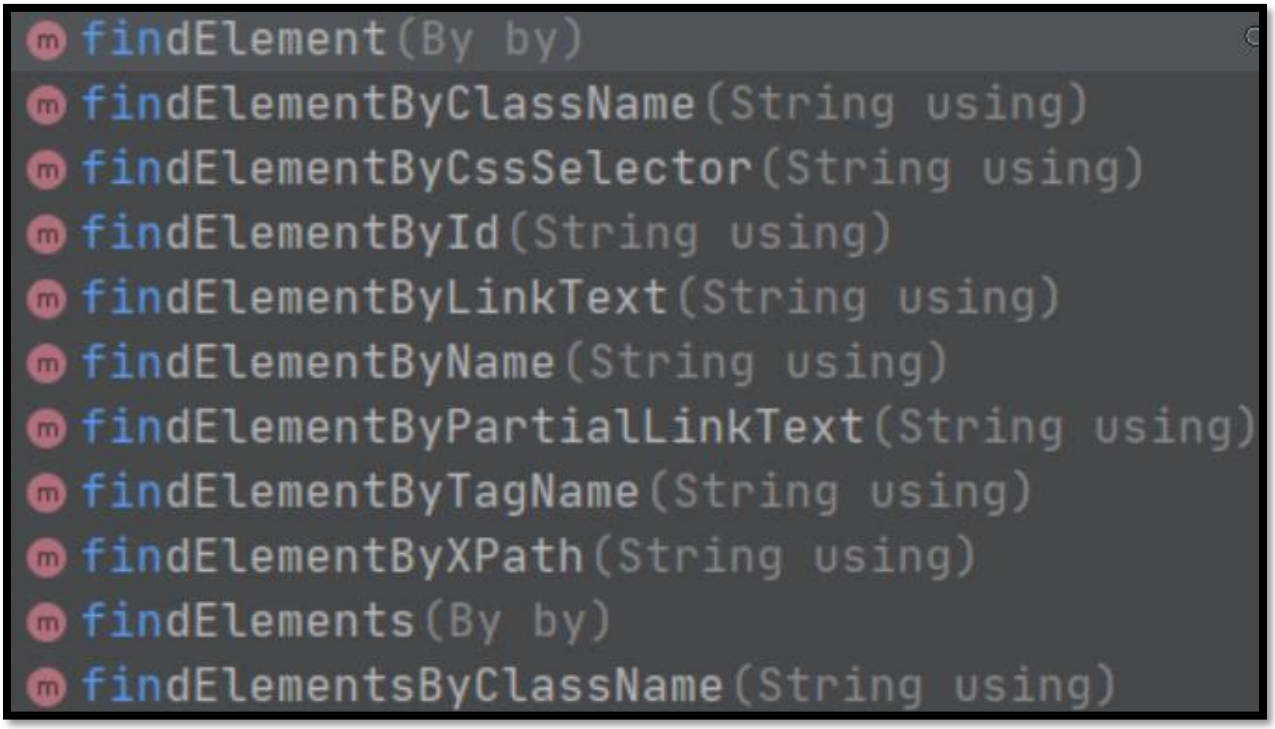
```
devTools.send(Network.emulateNetworkConditions(
        offline: true,
        latency: 10,
        downloadThroughput: 100,
        uploadThroughput: 50,
       Optional.of(ConnectionType.WIFI)));
```

Now, let's add a listener and what we expect. devTools.addListener() At this point, I'm going to write some data inside the addListener some parameters and that starts with (loadingFailed(),) and we are going to select the example with lambda expression because there are 2 options. Now, we write assertEquals() and inside assertEquals we are going to write loadingFailed and select the one that do not have parenthesis dot getErrorText(). Remember when we saw Internet Disconnected after setting the network to offline then refreshing the page. That's what we expect "net::ERR_INTERNET_DISCONNECTED"));

```
devTools.addListener(loadingFailed(),
       loadingFailed -> assertEquals(loadingFailed.getErrorText(),
               expected: "net::ERR_INTERNET_DISCONNECTED"));
```

The last step is to load the page. Since it's offline, let's try { } to load the page by writing driver.get("https://www.google.com").

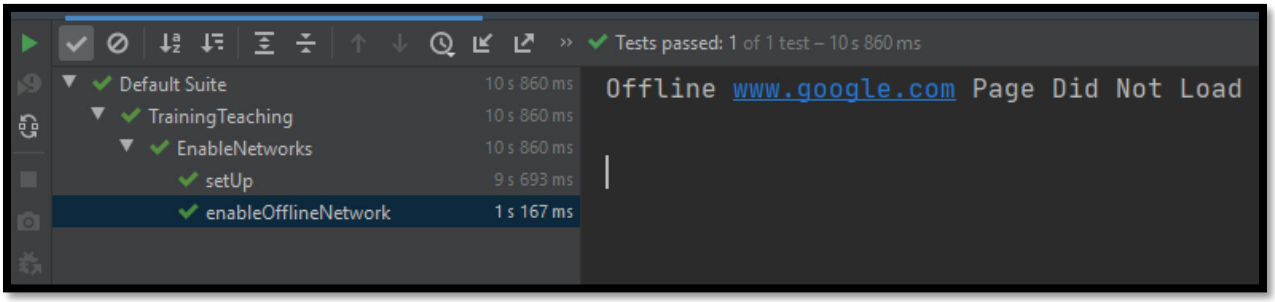
Next, is to catch () the possible exception (WebDriverException exc) { } and click the button by writing driver.find. Do you see the extra findElementBy options? We have these options because our type is EdgeDriver. It's been available to us for a long time. The same with ChromeDriver and FirefoxDriver. However, they supposed to be deprecated for Selenium 4.



I spoke about this in the Introduction to Selenium 4. The first video. I'm going to select driver.findElement(By.id("game-button")).click(). Also, print the page title: sout("Offline " + driver.getTitle() + "Page Did Not Load");

```
try {
  driver.get("https://www.google.com");
} catch (WebDriverException exc) {
  driver.findElement(By.id("game-button")).click();
  System.out.println("Offline " + driver.getTitle() + " Page Did Not Load");
}
```

That's it and let's run. It passed and we see the game. Now, for our Console, let's see what it shows enableOfflineNetwork. We see it shows Offline www.google.com and the Page Did Not Load.



Thank You for watching and I'll see you in the next session. I create videos every week and if you are interested in videos that I create, feel free to subscribe to my YouTube channel and click the bell icon. Also follow me on Twitter, connect with me on LinkedIn and Facebook. I will make sure to place the transcript and code on GitHub.

Contact

- ✔ YouTube https://www.youtube.com/c/RexJonesII/videos
- ✔ Facebook https://facebook.com/JonesRexII
- ✔ Twitter https://twitter.com/RexJonesII
- ✔ GitHub https://github.com/RexJonesII/Free-Videos
- ✔ LinkedIn https://www.linkedin.com/in/rexjones34/