Rex Jones II

Selenium 4
Take WebElement Screenshot

WebElement Screenshot

In this session, we are going to take a screenshot of a WebElement. Until now, Selenium already had a feature to take a screenshot of a page but not to take a screenshot of a WebElement. Before Selenium 4, we had to use an API called Ashot to take a screenshot of a WebElement. However, in this session I am going to demo how to take a screenshot of 1 WebElement and page section that includes more than 1 WebElement.

If you are interested in more videos, you can subscribe to my YouTube channel then click the bell icon. Follow me on Twitter and connect with me on LinkedIn and Facebook. Also, I'm going to place the transcript and code on GitHub.

We are going to use this Orange HRM site. I will take a screenshot of this logo for 1 WebElement. Do you see how this part of the page has more than 1 WebElement? We see Username, Password, Login Button, and the Forgot Password link plus this Orange image? I'm going to take screenshot of this entire section using Selenium 4. Let's start with the logo and Inspect then find the value by writing #divLogo. This is the id value for the parent > img is the tag name for the logo. Let's start by taking a screenshot of the WebElement logo.

One WebElement Screenshot

We have our test setup to load Chrome and the AUT.

```
@BeforeClass
public void setUp () {
 WebDriverManager.chromedriver().setup();
 driver = new ChromeDriver();
 driver.manage().window().maximize();
 driver.get("https://opensource-demo.orangehrmlive.com/");
}

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

Now, let's take a screenshot of the logo by writing @Test / public void takeWebElementScreenshot () { } Let's find the WebElement by writing WebElement logoOrangeHRM = driver.findElement(By.cssSelector("#divLogo > img")); Import the WebElement and @Test annotation from TestNG. Now that we have the logo for OrangeHRM logoOrangeHRM., we get the screenshotAs(OutputType.FILE). The getScreenshotAs method Capture the screenshot and store it in the specified location. Notice the return type is File. Let's hover the OutputType and it defines the output type for a screenshot. FILE is used to obtain the screenshot into a temporary file. Therefore, we assign the screenshot to a File and name it source =.

Now we need a File for our destination = new File called ("Orange HRM Logo.png"); We have our source file and destination file. A class called FileUtils. has a method to copyFile. The description shows it copies a file to a new location. So, for the source file we are going to have source and the destination file we are going to have our destination. Depending on how your Selenium 4 is setup. You may have to download the Apache Commons IO jar from Maven's Repository to get this FileUtils class then import the jar. It's also a dependency hierarchy for WebDriverManager. Last step is to add the throws declaration for IOException.

```
@Test
Run | Debug
public void takeWebElementScreenshot () throws IOException {
  WebElement logoOrangeHRM = driver.findElement(By.cssSelector("#divLogo > img"));
  File source = logoOrangeHRM.getScreenshotAs(OutputType.FILE);
  File destination = new File ("Orange HRM Logo.png");
  FileUtils.copyFile(source, destination);
}
```

That's it. Let's Run. We see the screenshot in our project. I setup Eclipse to auto refresh so I did not have to manually right click the project and select F5. Video 72 will show you how to Auto Refresh your project automatically. Right click, select Open, and here's the logo WebElement screenshot.

Next is to take a screenshot of a page section.

Page Section Screenshot

In a section, there are multiple WebElements so we need to find the parent tag of those WebElements. Inspect this section and we see the parent tag has an id value of divLoginImage. On this page, another

section is the Social Media icons that can also be used to take a screenshot of multiple WebElements. All we need to find is the parent tag and Selenium will take a screenshot of all 4 Social Media icons.

In this example, social-icons is the parent tag. Go back to our IDE and write our test for the page section. So, I'm going to write @Test / public void takePageSectionScreenshot () { } The page section is a WebElement so we write WebElement pageSectionOrangeHRM = driver.findElement(By.id("divLoginImage")); Next is to store the screenshot in a file. Therefore, we write File source = What is the source? The source pageSectionOrangeHRM.getScreenshotAs(OutputType.FILE);

This statement is an abstract representation of the File class and will save the screenshot in memory. We need a physical file since this is an abstract representation. Therefore, we copy the source file to the destination. I'm going to use these 2 statements and combine them into 1 line. So, I'm going to write FileUtils.copyFile(source,). The destination file will contain the name of the new File("Orange HRM Page Section.png"). You can also use .jpg file but I like to use png. This statement will convert the abstract source file into a physical .png file so we can view the screenshot. Add the throws declaration and Run. We see the Page Section screenshot. Let's open and all of the WebElements are located in this screenshot.

That's it and Thanks for watching.

Contact

- ✔ YouTube https://www.youtube.com/c/RexJonesII/videos
- ✔ Facebook http://facebook.com/JonesRexII

- ✔ Twitter https://twitter.com/RexJonesII
- ✔ GitHub https://github.com/RexJonesII/Free-Videos
- ✔ LinkedIn https://www.linkedin.com/in/rexjones34/