Selenium Screenshot in Java

Taking screenshots in Selenium using Java is a common requirement for debugging, reporting test failures, and documentation. Selenium provides an easy interface to capture screenshots of the entire page or specific elements.

✅ Concept Overview

Selenium WebDriver provides the **`TakesScreenshot`** interface, which includes a method **`getScreenshotAs()`** to capture screenshots.

✅ Full Working Code: Capture Full Page Screenshot

```
java

import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;

import java.io.File;
import java.io.IOException;
import org.apache.commons.io.FileUtils;

public class ScreenshotExample {
    public static void main(String[] args) throws IOException {
        // Set the path to your ChromeDriver
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

        // Launch Chrome browser
        WebDriver driver = new ChromeDriver();

        // Open a website
        driver.get("https://www.google.com");

        // Take screenshot and store it as a file format
        File src = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);

        // Specify the location to save the screenshot
        File dest = new File("screenshot.png");

        // Copy file to destination
        FileUtils.copyFile(src, dest);

        System.out.println("Screenshot captured!");

        // Close browser
        driver.quit();
    }
}
```

✅ Explanation of Key Parts

Line	Code	Explanation
`System.setProperty`	Sets the path to the ChromeDriver executable	

Line	Code	Explanation
`WebDriver driver = new ChromeDriver();`	Starts a new Chrome browser session	
`driver.get("https://...")`	Opens the desired URL	
`(TakesScreenshot) driver`	Casts **`WebDriver`** to **`TakesScreenshot`** interface	
`getScreenshotAs(OutputType.FILE)`	Captures screenshot as a file	
`FileUtils.copyFile()`	Saves screenshot to the specified path	

✅ Capture Screenshot of a Specific WebElement

```
java


import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;

import java.io.File;
import java.io.IOException;
import org.apache.commons.io.FileUtils;

public class ElementScreenshot {
    public static void main(String[] args) throws IOException {
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

        WebDriver driver = new ChromeDriver();
        driver.get("https://www.google.com");

        // Locate the element
        WebElement logo = driver.findElement(By.id("hplogo")); // may change based on Google's
layout

        // Take screenshot of the element
        File src = logo.getScreenshotAs(OutputType.FILE);
        File dest = new File("element_screenshot.png");

        FileUtils.copyFile(src, dest);
        System.out.println("Element screenshot saved.");

        driver.quit();
    }
}
```

- Note: Not all drivers support element-level screenshots. Chrome and Firefox usually do.

✅ Required Maven Dependency (for FileUtils)

If you're using Maven, add the following for **`FileUtils`**:

```
xml


<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.11.0</version>
</dependency>
```

✅ Best Practices

- Capture screenshots on failure (using try-catch in test frameworks like JUnit/TestNG).
- Use timestamped filenames to avoid overwriting.
- Save in a structured folder like **`screenshots/<testname>/<timestamp>.png`**.

✅ Example: Screenshot on Exception

```
java


try {
    driver.findElement(By.id("non_existing_id")).click();
} catch (Exception e) {
    File src = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
    File dest = new File("error_screenshot.png");
    FileUtils.copyFile(src, dest);
    System.out.println("Error occurred! Screenshot captured.");
}
```

Would you like me to help you integrate this into a **TestNG** or **JUnit** framework with reusable screenshot methods and reporting?

Great! Let's now integrate **screenshot capturing into a TestNG framework** in Java. This is useful for automatically taking screenshots when a test fails.

✅ TestNG with Screenshot on Failure

We'll do the following:

1. Create a **base test** to handle browser setup/teardown.
2. Use **ITestListener** to hook into test lifecycle events.
3. Automatically **take a screenshot** on failure.

✅ 1. Maven Dependencies (`pom.xml`)

Ensure you have the following:

```
xml


<dependencies>
    <!-- Selenium -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.19.0</version>
    </dependency>

    <!-- TestNG -->
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>7.10.1</version>
        <scope>test</scope>
    </dependency>

    <!-- Apache Commons IO for FileUtils -->
    <dependency>
        <groupId>commons-io</groupId>
        <artifactId>commons-io</artifactId>
        <version>2.11.0</version>
    </dependency>
</dependencies>
```

✅ 2. BaseTest Class

This will launch and close the browser.

```
java


import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.BeforeMethod;

public class BaseTest {
    public static WebDriver driver;

    @BeforeMethod
    public void setup() {
```

```
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
        driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.google.com");
    }

    @AfterMethod
    public void tearDown() {
        if (driver != null) {
            driver.quit();
        }
    }
}
```

✅ 3. Screenshot Utility Class

```
java


import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import java.io.File;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import org.apache.commons.io.FileUtils;

public class ScreenshotUtil {
    public static void takeScreenshot(WebDriver driver, String testName) {
        File src = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
        String timestamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
        String fileName = "screenshots/" + testName + "_" + timestamp + ".png";
        try {
            FileUtils.copyFile(src, new File(fileName));
            System.out.println("Screenshot saved: " + fileName);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

✅ 4. TestNG Listener to Capture Screenshots on Failure

```
java


import org.testng.ITestListener;
import org.testng.ITestResult;

public class ScreenshotListener implements ITestListener {

    @Override
    public void onTestFailure(ITestResult result) {
        Object testClass = result.getInstance();
        WebDriver driver = BaseTest.driver;

        if (driver != null) {
            ScreenshotUtil.takeScreenshot(driver, result.getName());
        }
    }
}
```

✅ 5. Sample Test Class

```
java

import org.testng.Assert;
import org.testng.annotations.Listeners;
import org.testng.annotations.Test;

@Listeners(ScreenshotListener.class)
public class GoogleSearchTest extends BaseTest {

    @Test
    public void testThatPasses() {
        String title = driver.getTitle();
        System.out.println("Page Title: " + title);
        Assert.assertTrue(title.contains("Google"));
    }

    @Test
    public void testThatFails() {
        // This will fail intentionally
        Assert.assertTrue(driver.getTitle().contains("Bing"));
    }
}
```

✅ Folder Structure Suggestion

```
bash

project-root/
├── screenshots/
├── src/
│   └── test/
│       └── java/
│           ├── BaseTest.java
│           ├── ScreenshotUtil.java
│           ├── ScreenshotListener.java
│           └── GoogleSearchTest.java
```

✅ Run Tests

You can run using Maven:

```
bash

mvn test
```

Or use your IDE (like IntelliJ or Eclipse) to run **`GoogleSearchTest`**.

✅ Result

- If **`testThatFails`** fails, a screenshot is saved under **`screenshots/`** with a timestamp.
- **`testThatPasses`** will not save any screenshot.

Would you like me to help you **generate an HTML test report** or **email the screenshot** after test failure next?