# **Taking Screenshots in Selenium with Java**

## **📸 Screenshot Capture Methods**

### **1. Basic Full-Page Screenshot**

java

import org.openqa.selenium.OutputType;

import org.openqa.selenium.TakesScreenshot;

import org.apache.commons.io.FileUtils;

// Capture screenshot

File srcFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

// Save to file

FileUtils.copyFile(srcFile, new File("screenshots/fullpage.png"));

### **2. Element-Specific Screenshot**

java

WebElement element = driver.findElement(By.id("header"));

File srcFile = element.getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(srcFile, new File("screenshots/header.png"));

### **3. Screenshot with Timestamp**

java

String timestamp = new SimpleDateFormat("yyyyMMdd\_HHmmss").format(new Date());

FileUtils.copyFile(srcFile, new File("screenshots/screenshot\_" + timestamp + ".png"));

## **🛠 Production-Grade Implementation**

### **1. Screenshot Utility Class**

java

import org.apache.commons.io.FileUtils;

import org.openqa.selenium.\*;

import java.io.File;

import java.text.SimpleDateFormat;

import java.util.Date;

public class ScreenshotUtil {

private final WebDriver driver;

public ScreenshotUtil(WebDriver driver) {

this.driver = driver;

}

public String captureFullPageScreenshot(String testName) {

try {

File srcFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

String filePath = "screenshots/" + testName + "\_" + getTimestamp() + ".png";

FileUtils.copyFile(srcFile, new File(filePath));

return filePath;

} catch (Exception e) {

System.err.println("Failed to capture screenshot: " + e.getMessage());

return null;

}

}

public String captureElementScreenshot(WebElement element, String elementName) {

try {

File srcFile = element.getScreenshotAs(OutputType.FILE);

String filePath = "screenshots/element\_" + elementName + "\_" + getTimestamp() + ".png";

FileUtils.copyFile(srcFile, new File(filePath));

return filePath;

} catch (Exception e) {

System.err.println("Failed to capture element screenshot: " + e.getMessage());

return null;

}

}

private String getTimestamp() {

return new SimpleDateFormat("yyyyMMdd\_HHmmssSSS").format(new Date());

}

}

### **2. Screenshot on Test Failure**

java

import org.testng.ITestListener;

import org.testng.ITestResult;

public class ScreenshotListener implements ITestListener {

@Override

public void onTestFailure(ITestResult result) {

WebDriver driver = (WebDriver)result.getTestContext().getAttribute("driver");

String screenshotPath = new ScreenshotUtil(driver)

.captureFullPageScreenshot(result.getName());

System.out.println("Screenshot saved: " + screenshotPath);

}

}

## **⚡ Advanced Techniques**

### **1. Screenshot with Highlighted Element**

java

public String captureHighlightedScreenshot(WebElement element, String testName) {

// Highlight element

JavascriptExecutor js = (JavascriptExecutor)driver;

String originalStyle = element.getAttribute("style");

js.executeScript("arguments[0].style.border='3px solid red'", element);

// Take screenshot

String path = captureFullPageScreenshot(testName);

// Reset style

js.executeScript("arguments[0].style.border='" + originalStyle + "'", element);

return path;

}

### **2. Full Page Screenshot (Chrome Only)**

java

ChromeOptions options = new ChromeOptions();

options.addArguments("--start-maximized");

options.addArguments("--headless"); // For headless screenshots

options.addArguments("--hide-scrollbars");

WebDriver driver = new ChromeDriver(options);

// Full page screenshot (Chrome specific)

driver.get("https://example.com");

Screenshot screenshot = new AShot()

.shootingStrategy(ShootingStrategies.viewportPasting(1000))

.takeScreenshot(driver);

ImageIO.write(screenshot.getImage(), "PNG", new File("fullpage.png"));

## **📁 File Management Best Practices**

### **1. Organized Directory Structure**

text

test-output/

├── screenshots/

│ ├── login\_tests/

│ │ ├── successful\_login\_20230801\_143022.png

│ │ └── failed\_login\_20230801\_143045.png

│ └── checkout\_tests/

│ ├── add\_to\_cart\_20230801\_143107.png

│ └── payment\_failure\_20230801\_143129.png

### **2. Automatic Directory Creation**

java

File screenshotDir = new File("test-output/screenshots");

if (!screenshotDir.exists()) {

screenshotDir.mkdirs();

}

## **🔗 Dependencies**

Add to pom.xml:

xml

<!-- For basic screenshots -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.10.0</version>

</dependency>

<dependency>

<groupId>commons-io</groupId>

<artifactId>commons-io</artifactId>

<version>2.11.0</version>

</dependency>

<!-- For advanced screenshots -->

<dependency>

<groupId>ru.yandex.qatools.ashot</groupId>

<artifactId>ashot</artifactId>

<version>1.5.4</version>

</dependency>

## **🚀 Integration with Test Frameworks**

### **TestNG Example**

java

public class LoginTests {

WebDriver driver;

ScreenshotUtil screenshotUtil;

@BeforeClass

public void setup() {

driver = new ChromeDriver();

screenshotUtil = new ScreenshotUtil(driver);

}

@Test

public void testSuccessfulLogin() {

try {

// Test steps

screenshotUtil.captureFullPageScreenshot("before\_login");

// More test steps

} catch (Exception e) {

screenshotUtil.captureFullPageScreenshot("login\_error");

throw e;

}

}

@AfterClass

public void teardown() {

driver.quit();

}

}

### **JUnit 5 Example**

java

@ExtendWith(ScreenshotExtension.class)

class ProductTests {

@Test

void testAddToCart() {

// Test will automatically save screenshot on failure

}

}

class ScreenshotExtension implements TestWatcher {

@Override

public void testFailed(ExtensionContext context, Throwable cause) {

WebDriver driver = // get driver from context

new ScreenshotUtil(driver).captureFullPageScreenshot(context.getDisplayName());

}

}

## **💡 Pro Tips**

1. For CI/CD Pipelines: Compress screenshots after test runs
2. Reduce Storage: Delete screenshots from successful tests
3. Parallel Tests: Include thread ID in filename for parallel execution
4. Visual Testing: Integrate with Applitools/Percy for advanced comparison

This comprehensive guide covers all aspects of screenshot capture in Selenium WebDriver with Java, from basic implementations to production-ready solutions with proper file management and test framework integration.