ADP Selenium Interview Questions with Answers (2026) | Saravana Tech

Elaborate answers, examples and practical code for 10 common interview questions

Use URLs and locators from your test application.

1. How do you handle multiple windows in Selenium?

Concept and methods

A browser can open another tab or window after a click. Selenium identifies windows with handles. getWindowHandle() returns the current window handle; getWindowHandles() returns all open handles; switchTo().window(handle) changes the active window; close() closes the current window; quit() closes all windows and ends the WebDriver session.

```
WebDriver driver = new ChromeDriver();
driver.get("https://your-test-app.example");
String parent = driver.getWindowHandle();

driver.findElement(By.id("openWindow")).click();

WebDriverWait wait =
    new WebDriverWait(driver, Duration.ofSeconds(10));
wait.until(ExpectedConditions.numberOfWindowsToBe(2));

for (String handle : driver.getWindowHandles()) {
    if (!handle.equals(parent)) {
        driver.switchTo().window(handle);
        break;
    }
}
System.out.println(driver.getTitle());

driver.close();                 // close child window
driver.switchTo().window(parent); // return to parent
driver.quit();
```

> Interview tip: Do not assume that the second handle is always the child. Compare handles or identify the target window by title, URL or a unique element.

2. How do you wait for a particular element?

Types of waits

Implicit wait sets a general timeout for locating elements. Explicit wait waits for a specific condition and is usually preferred for particular elements. Fluent wait lets you configure timeout, polling frequency and exceptions to ignore. Avoid mixing implicit and explicit waits because combined timing can be unpredictable.

```
WebDriverWait wait =
    new WebDriverWait(driver, Duration.ofSeconds(10));

WebElement button = wait.until(
    ExpectedConditions.elementToBeClickable(By.id("login"))
);
button.click();

// Wait for a loader to disappear
wait.until(ExpectedConditions.invisibilityOfElementLocated(
    By.cssSelector(".loading-spinner")
));
```

Useful conditions: visibilityOfElementLocated, presenceOfElementLocated, elementToBeClickable, invisibilityOfElementLocated, textToBePresentInElementLocated and numberOfWindowsToBe. Prefer condition-based waits over Thread.sleep().

3. How do you handle different alert methods?

Alert types and methods

A simple alert has a message and OK button; a confirmation alert has OK and Cancel; a prompt alert accepts text. Selenium Alert methods are accept(), dismiss(), getText() and sendKeys(). These are for JavaScript alerts, not HTML modal dialogs.

```
WebDriverWait wait =
    new WebDriverWait(driver, Duration.ofSeconds(10));

Alert alert = wait.until(ExpectedConditions.alertIsPresent());
System.out.println(alert.getText());
alert.accept(); // click OK

// Confirmation alert: use alert.dismiss() to click Cancel
// Prompt alert:
alert.sendKeys("Sample text");
alert.accept();
```

If a JavaScript alert is open, normal webpage interactions may fail. Handle the alert first, then continue.

4. Write Selenium code to take a screenshot

Capture the current browser view

TakesScreenshot captures a screenshot. Create the destination folder if needed and use a unique filename for each test to avoid overwriting files.

```
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;

public static void captureScreenshot(WebDriver driver,
                                     String fileName)
                                     throws IOException {
    File source = ((TakesScreenshot) driver)
        .getScreenshotAs(OutputType.FILE);
    Path target = Paths.get("screenshots", fileName + ".png");
    Files.createDirectories(target.getParent());
    Files.copy(source.toPath(), target);
```

```
}

// Usage:
captureScreenshot(driver, "LoginPage");
```

Screenshot on TestNG failure

```
public class ScreenshotListener implements ITestListener {
    @Override
    public void onTestFailure(ITestResult result) {
        WebDriver driver = DriverManager.getDriver(); // your framework method
        if (driver != null) {
            try {
                captureScreenshot(driver, result.getName());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Register the listener in testng.xml. DriverManager.getDriver() is a placeholder for the WebDriver management method in your framework. In parallel runs, use a driver per test thread and unique filenames.

5. How do you fetch values from a web table?

Read all rows and cells

Use findElements() to retrieve rows and cells. The example assumes a table with id employeeTable and uses th/td so it works with header and data cells.

```
WebElement table = driver.findElement(By.id("employeeTable"));
List<WebElement> rows =
    table.findElements(By.cssSelector("tr"));

for (WebElement row : rows) {
    List<WebElement> cells =
        row.findElements(By.cssSelector("th, td"));
    for (WebElement cell : cells) {
        System.out.print(cell.getText() + " | ");
    }
    System.out.println();
}
```

Find a record by employee ID

```
List<WebElement> rows = driver.findElements(
    By.cssSelector("#employeeTable tr"));

for (WebElement row : rows) {
    List<WebElement> cells =
        row.findElements(By.cssSelector("th, td"));

    if (cells.size() >= 3 &&
        cells.get(0).getText().trim().equals("102")) {
        System.out.println("Name: " + cells.get(1).getText());
        System.out.println("Department: " + cells.get(2).getText());
        break;
    }
}
```

For paginated tables, read the current page, click Next, wait for the table to update, and repeat. Virtualised tables may render only visible rows.

6. What is a DataProvider in TestNG?

Concept

A DataProvider supplies multiple input data sets to one TestNG test method. TestNG invokes the method once per data row, enabling data-driven testing without duplicating test methods.

```
@DataProvider(name = "loginData")
public Object[][] loginData() {
    return new Object[][] {
        {"validUser", "validPassword", true},
        {"invalidUser", "wrongPassword", false},
        {"", "validPassword", false}
    };
}

@Test(dataProvider = "loginData")
public void loginTest(String username, String password,
                      boolean expectedSuccess) {
    // Enter values, submit, and assert the actual outcome.
}
```

DataProvider is best for multiple test data sets. @Parameters usually supplies named configuration values from testng.xml, such as browser or base URL. If a DataProvider runs in parallel, tests and test data must be thread-safe.

7. How do you read Excel values?

Using Apache POI

Apache POI can read .xls and .xlsx workbooks. For .xlsx, add the poi-ooxml dependency to Maven, using a current compatible version approved for your project. Save the workbook, for example, as src/test/resources/TestData.xlsx, with a sheet called LoginData.

```
// Maven dependency (replace with an approved real version):
<dependency>
  <groupId>org.apache.poi</groupId>
  <artifactId>poi-ooxml</artifactId>
  <version>YOUR_APPROVED_VERSION</version>
</dependency>
```

```
public class ExcelReader {
    public static void main(String[] args) throws IOException {
        String path = "src/test/resources/TestData.xlsx";

        try (FileInputStream fis = new FileInputStream(path);
             Workbook workbook = new XSSFWorkbook(fis)) {
            Sheet sheet = workbook.getSheet("LoginData");
            DataFormatter formatter = new DataFormatter();

            for (Row row : sheet) {
                for (Cell cell : row) {
                    System.out.print(
                        formatter.formatCellValue(cell) + "\t");
                }
                System.out.println();
            }
        }
    }
}
```

Imports include java.io.FileInputStream, java.io.IOException, org.apache.poi.ss.usermodel.* and org.apache.poi.xssf.usermodel.XSSFWorkbook. DataFormatter returns values in display-friendly text form. Use try-with-resources to close the workbook and input stream.

Use Excel data in TestNG

```
@DataProvider(name = "excelLoginData")
public Object[][] readLoginData() throws IOException {
    try (FileInputStream fis = new FileInputStream(
              "src/test/resources/TestData.xlsx");
         Workbook workbook = new XSSFWorkbook(fis)) {

        Sheet sheet = workbook.getSheet("LoginData");
        int lastRow = sheet.getLastRowNum();
        Object[][] data = new Object[lastRow][3];
        DataFormatter fmt = new DataFormatter();

        for (int r = 1; r <= lastRow; r++) { // skip header
            Row row = sheet.getRow(r);
            data[r - 1][0] = fmt.formatCellValue(row.getCell(0));
            data[r - 1][1] = fmt.formatCellValue(row.getCell(1));
            data[r - 1][2] = fmt.formatCellValue(row.getCell(2));
        }
        return data;
    }
}
```

This example assumes every data row contains three populated cells. Add null checks for blank rows or cells, and avoid storing real passwords in plain-text workbooks.

8. BVA and test coverage: write test cases for a scenario

Boundary Value Analysis (BVA)

BVA focuses on values at and around boundaries because defects often occur at the minimum and maximum. Example requirement: age must be an integer from 18 to 60 inclusive.

Test case	Input age	Expected result
TC01	17	Rejected
TC02	18	Accepted
TC03	19	Accepted
TC04	59	Accepted
TC05	60	Accepted
TC06	61	Rejected

Additional tests: empty field, alphabetic input, decimals if only integers are allowed, negative values, very large values, whitespace, error messages, successful submission and keyboard accessibility where required.

Types of coverage

Requirements coverage measures the proportion of in-scope requirements mapped to tests. Execution coverage measures the proportion of planned tests executed. Pass rate measures passed tests among executed tests. Code coverage measures source code exercised by tests; it does not prove test quality.

```
Requirements coverage =
    (Requirements with mapped tests / Total in-scope requirements) * 100

Execution coverage =
    (Executed test cases / Planned test cases) * 100

Pass rate =
    (Passed test cases / Executed test cases) * 100
```

Example: 18 of 20 requirements have tests, so requirements coverage is 90%. If 80 of 100 test cases are executed, execution coverage is 80%. If 70 of those 80 pass, the pass rate is 87.5%. Report blocked and skipped cases separately where appropriate.

Interview answer

“I analyse the requirement and identify positive, negative, boundary and exception scenarios. I map test cases to requirements in an RTM, review missing coverage, prioritise high-risk workflows, execute or automate appropriate tests, and track results. I report requirements coverage, execution coverage and pass rate separately because each metric measures a different thing.”

9. Scenario: open Chrome, enter values and search

Sample workflow

Open the application, enter username and password, click Login, wait for the dashboard, enter search text,

Use URLs and locators from your test environment.

```
WebDriver driver = new ChromeDriver();
WebDriverWait wait =
    new WebDriverWait(driver, Duration.ofSeconds(15));

try {
    driver.manage().window().maximize();
    driver.get("https://your-test-app.example/login");

    wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("username")))
        .sendKeys(System.getenv("TEST_USERNAME"));

    driver.findElement(By.id("password"))
        .sendKeys(System.getenv("TEST_PASSWORD"));

    wait.until(ExpectedConditions.elementToBeClickable(
        By.id("loginButton"))).click();

    wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("dashboard")));

    wait.until(ExpectedConditions.visibilityOfElementLocated(
        By.id("searchBox"))).sendKeys("Selenium");

    driver.findElement(By.id("searchButton")).click();

    WebElement result = wait.until(
        ExpectedConditions.visibilityOfElementLocated(
            By.id("searchResult")));

    Assert.assertTrue(result.getText().contains("Selenium"));
} finally {
    driver.quit();
}
```

Useful follow-ups: use clear() before sendKeys() if replacing existing text; use Select for native HTML <select> elements; custom Angular/React dropdowns usually need click-and-select interactions. Store credentials in environment variables or a secrets manager, not source code.

Imports commonly needed: java.time.Duration; org.openqa.selenium.*; org.openqa.selenium.support.ui.*; org.testng.Assert.

10. Test case coverage: additional interview examples

Scenario A: Login page

Example test cases: valid credentials; invalid username; invalid password; both fields empty; only username empty; only password empty; leading/trailing spaces; overlong username; repeated failed logins and rate limiting; Enter-key submission if supported; browser Back after login; direct access to a protected URL. Confirm expected behaviour from requirements.

Test ID	Scenario	Expected result
TC01	Valid credentials	Login succeeds
TC02	Invalid username	Appropriate error
TC03	Invalid password	Appropriate error
TC04	Both fields empty	Required-field validation
TC05	Password empty	Password validation
TC06	Direct access to protected URL	Redirect or deny access

Scenario B: Requirements Traceability Matrix (RTM)

An RTM maps requirements to test cases and exposes gaps.

Requirement	Mapped test cases	Coverage status
REQ-001: Login	TC01, TC02, TC03	Covered
REQ-002: Search	TC04, TC05	Covered
REQ-003: Booking date	TC06, TC07	Partially covered
REQ-004: Cancellation	None	Not covered

How do you improve test coverage?

Review requirements, identify high-risk and business-critical workflows, create positive/negative/boundary/exception scenarios, map tests to requirements, discuss gaps with stakeholders, automate stable repetitive regression tests, and review execution results and defect trends. Use production defects and lessons learned to identify missing scenarios. High coverage does not guarantee zero defects.