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

Elaborate answers with practical examples — 33 topics

Use this guide to prepare clear interview responses. Code examples are illustrative; adapt browser versions, dependencies, locators, and project details to your environment. Some topics refer to legacy Internet Explorer, which is retired and unsupported by modern Selenium releases.

1. Eclipse IDE and Selenium interaction — architecture

Eclipse is the IDE in which you write, compile, debug, and run Java test code. Selenium is the browser-automation library. When a test calls WebDriver methods, the Java binding sends commands through Selenium's driver communication layer to a browser-specific driver or browser automation endpoint. The browser executes the command and returns a response, which WebDriver exposes to the Java code. In Selenium 4, W3C WebDriver is the standard protocol. Selenium Manager can often resolve and manage compatible drivers automatically.

Example

```
Java test in Eclipse
    -> Selenium Java WebDriver API
    -> WebDriver protocol / browser driver
    -> Browser (Chrome / Firefox / Edge)
    <- response: title, element state, errors, etc.
```

Interview tip: Eclipse does not control the browser by itself; it runs the Java program that uses Selenium.

2. System.setProperty("webdriver.ie.driver", "path of ie driver.exe")

System.setProperty sets a Java system property. In older Selenium setups, the property webdriver.ie.driver specified the filesystem path to IEDriverServer.exe, the executable Selenium used to communicate with Internet Explorer. The path must point to the actual driver executable, not to a webpage or application file. The exact property and driver setup depend on the Selenium version. Selenium 4.6+ includes Selenium Manager for supported modern browsers, but Internet Explorer is retired.

Example

```
System.setProperty("webdriver.ie.driver",
        "C:\\WebDrivers\\IEDriverServer.exe");
WebDriver driver = new InternetExplorerDriver();
```

Interview tip: The executable is a browser-driver process, not a signal that Selenium is automating arbitrary desktop software.

3. Does using an .exe mean Selenium interacts with desktop applications?

No. The .exe in the example is IEDriverServer.exe, a helper process that exposes WebDriver commands for the Internet Explorer browser. Selenium WebDriver primarily automates web browsers and web content. It is not a general-purpose desktop UI automation tool for native Windows applications. For native desktop automation, tools such as Windows Application Driver (where applicable), Appium Windows driver, or other UI automation frameworks may be considered; check their current maintenance and compatibility before choosing one.

4. Why do we need browser drivers?

A driver bridges WebDriver commands from the test code to a specific browser. It translates standard commands such as navigate, click, send keys, and find element into browser-specific operations, then returns results and errors. Historically, a compatible driver executable had to be downloaded and configured. Modern Selenium Manager can automatically manage drivers for supported browsers, reducing manual setup. Browser and driver compatibility still matters.

5. Java code to read a file

Java offers several ways to read files. For small text files, Files.readString is concise (Java 11+). For line-by-line processing, use Files.readAllLines or a BufferedReader. Use try-with-resources for resources that must be closed. Choose the encoding explicitly when required.

Example

```
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;

public class ReadFileExample {
    public static void main(String[] args) throws Exception {
        Path path = Path.of("testdata", "input.txt");
        String content = Files.readString(path, StandardCharsets.UTF_8);
        System.out.println(content);
    }
}
```

> **Interview tip:** In an interview, mention file path, character encoding, exception handling, and whether the file is small or large.

6. Implicit wait and explicit wait — syntax and details

Implicit wait is a global WebDriver setting for element location. If an element is not immediately found, WebDriver keeps trying until the timeout, then throws NoSuchElementException. Explicit wait targets a particular condition, such as visibility, clickability, or a URL change. It is generally preferred for dynamic pages because it states exactly what must happen. Avoid mixing large implicit waits with explicit waits because their combined timing can be confusing.

Example

```
// Implicit wait
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));

// Explicit wait
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
WebElement saveButton = wait.until(
    ExpectedConditions.elementToBeClickable(By.id("save"))
);
saveButton.click();
```

> **Interview tip:** Selenium 4 uses Duration-based timeout syntax. Import java.time.Duration and the relevant support.ui classes.

7. How to upload a file in Selenium

For a standard HTML file input, send the absolute file path to the input element using sendKeys. Usually, do not click the OS file chooser because WebDriver cannot reliably control native dialogs through ordinary DOM locators. If the input is hidden or the page uses a custom upload widget, inspect the DOM and interact with the underlying input where possible.

Example

```
Path file = Path.of("C:", "TestData", "sample.xlsx").toAbsolutePath();
WebElement upload = driver.findElement(By.cssSelector("input[type='file']"));
upload.sendKeys(file.toString());
driver.findElement(By.id("uploadButton")).click();
```

> **Interview tip:** Use a real file path available on the machine or CI agent where the test runs.

8. JavaScriptExecutor

JavascriptExecutor is a Selenium interface that executes JavaScript in the context of the current page. It can scroll, read DOM properties, or perform actions when standard WebDriver interactions are unsuitable. Prefer normal WebDriver methods first: JavaScript clicks can bypass normal user-interaction checks and may hide real UI problems. It cannot directly automate arbitrary desktop applications.

Example

```
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("window.scrollTo(0, document.body.scrollHeight);");

// Example: read page title through JavaScript
String title = (String) js.executeScript("return document.title;");
```

9. What is POM?

POM means Page Object Model, a design pattern that separates page locators and page actions from test assertions and scenarios. Each page or component is represented by a class. This improves maintainability: if a locator changes, it can often be updated in one place. Keep page methods meaningful, avoid putting all test logic into page classes, and avoid unnecessary duplication.

Example

```
public class LoginPage {
    private final WebDriver driver;
    private final By username = By.id("username");
    private final By password = By.id("password");
    private final By loginButton = By.id("login");

    public LoginPage(WebDriver driver) { this.driver = driver; }

    public void login(String user, String pass) {
        driver.findElement(username).sendKeys(user);
        driver.findElement(password).sendKeys(pass);
        driver.findElement(loginButton).click();
    }
}
```

10. Have you worked on Cucumber?

A truthful interview answer should reflect your actual experience. If you have used Cucumber, explain that it supports Behavior-Driven Development through Gherkin scenarios and connects steps to automation code. A typical framework contains feature files, step definitions, hooks, runner/configuration, page objects, test data, and reporting. Explain one scenario you automated and how you manage setup, teardown, and reusable steps.

Example

```
Feature: Login
  Scenario: Valid user can sign in
    Given the user opens the login page
    When the user signs in with valid credentials
    Then the dashboard should be displayed
```

> **Interview tip:** Use a real project example; do not claim experience you have not had.

11. Click an element without trying every locator

First inspect the DOM and choose the most stable, unique locator. Prefer a unique id or a meaningful data-testid when available, then accessible role/name or a stable CSS selector. Use XPath when relationships or text-based DOM traversal are genuinely needed. Verify uniqueness and wait for the element to be actionable. Do not cycle through all locator types blindly.

Example

```
By saveButton = By.cssSelector("[data-testid='save-button']");
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
wait.until(ExpectedConditions.elementToBeClickable(saveButton)).click();
```

> **Interview tip:** Explain how you choose a locator: stability, uniqueness, readability, and resistance to UI changes.

12. Can main() return type be int?

For a standard Java application entry point, main is declared with a void return type, for example public static void main(String[] args). A method named main can be overloaded or declared with another return type, but it will not serve as the conventional Java launcher entry point. The launcher does not use an int return value from main as the process exit code; use System.exit(status) when appropriate.

Example

```
public static void main(String[] args) {
    System.out.println("Hello");
    // System.exit(0); // optional explicit process status
}
```

13. Frame handling code in Selenium

An iframe has its own browsing context. Selenium must switch into that frame before locating elements inside it, and switch back to the main document or parent frame afterward. Frames can be selected by index, name/id, or WebElement; a stable frame locator is generally preferable to an index.

Example

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt(
    By.cssSelector("iframe.payment-frame")
));
driver.findElement(By.id("cardNumber")).sendKeys("4111111111111111");
driver.switchTo().defaultContent();
```

> **Interview tip:** If frames are nested, use parentFrame() or switch through each frame in order.

14. Launch Internet Explorer browser

Legacy Selenium code used InternetExplorerDriver with IEDriverServer.exe and Internet Explorer-specific configuration. However, Internet Explorer 11 desktop application has been retired and is not supported by current mainstream Selenium/browser setups. For new automation, use Edge or Chrome. If a legacy application requires IE behavior, investigate Microsoft Edge IE mode and your organization's supported test strategy.

Example

```
// Legacy illustration only; requires a compatible legacy setup
System.setProperty("webdriver.ie.driver", "C:\\WebDrivers\\IEDriverServer.exe");
WebDriver driver = new InternetExplorerDriver();
driver.get("https://example.com");
```

> **Interview tip:** Clearly state this is legacy-only; do not present it as a recommended modern setup.

15. SQL: pull employee IDs from two tables where IDs match

Use an INNER JOIN when you want only employee IDs present in both tables. If duplicate IDs exist, the result can contain duplicates; use DISTINCT if the requirement is a unique list.

Example

```
SELECT a.emp_id
FROM employees_a AS a
INNER JOIN employees_b AS b
    ON a.emp_id = b.emp_id;
```

16. What is a JOIN?

A JOIN combines rows from two or more tables using a related column or join condition. INNER JOIN returns matching rows; LEFT JOIN returns all rows from the left table and matching rows from the right; RIGHT JOIN does the reverse; FULL OUTER JOIN returns matched and unmatched rows from both sides where supported. CROSS JOIN returns combinations of every row from each table.

Example

```
-- All employees, including those without a matching department
SELECT e.emp_id, e.emp_name, d.department_name
FROM employee e
LEFT JOIN department d
    ON e.department_id = d.department_id;
```

17. Read a CSV file (flat-file reading)

A CSV is a text file with fields separated by commas, but values may contain quoted commas, escaped quotes, or newlines. For a simple controlled file, Java can read lines and split them; for real CSV data, prefer a CSV library such as Apache Commons CSV or OpenCSV because split(",") does not correctly handle all CSV rules.

Example

```
// Simple example only: assumes no quoted commas and no embedded newlines
List<String> lines = Files.readAllLines(
    Path.of("testdata", "employees.csv"), StandardCharsets.UTF_8);
for (String line : lines) {
    String[] columns = line.split(",", -1);
    System.out.println(Arrays.toString(columns));
}
```

> **Interview tip:** Mention the limitation of split() and use a CSV parser for production-quality data.

18. What is XPath?

XPath is a language for selecting nodes in an XML/HTML document. Selenium uses XPath expressions to locate web elements. Absolute XPath starts from the document root and is usually brittle; relative XPath is more maintainable. XPath supports attributes, text, axes, and functions.

Example

```
// By id-like attribute
//input[@id='email']

// By visible text (exact normalized text)
 //button[normalize-space()='Submit']
```

19. XPath sibling and following axes

The sibling axes select nodes with the same parent: following-sibling selects siblings after the current node, while preceding-sibling selects siblings before it. The following axis selects nodes that appear later in document order, excluding descendants; it is broader than following-sibling.

Example

```
//label[normalize-space()='Email']/following-sibling::input
//td[normalize-space()='Employee ID']/following-sibling::td[1]
```

> **Interview tip:** Use axes when the DOM relationship is stable and there is no better direct locator.

20. final keyword on variables and methods

A final variable can be assigned only once. A final reference cannot be reassigned, although the referenced object may still be mutable. A final method cannot be overridden in a subclass. A final class cannot be extended. final is different from finally, which is part of exception handling, and finalize(), which is a legacy lifecycle method.

Example

```
final int maxRetries = 3;
// maxRetries = 4; // compile-time error

final class Utility { }
// class Child extends Utility {} // compile-time error
```

21. static keyword

static members belong to the class rather than to each object instance. A static method can be called using the class name and cannot directly access instance members without an object reference. Static fields are shared across instances. Use static for class-level utilities or constants, but avoid mutable global state where possible.

Example

```
class Counter {
    static int total = 0;
    Counter() { total++; }
}
System.out.println(Counter.total);
```

22. What does String[] args mean?

String[] args is the parameter of the Java main method. It is an array of strings passed from the command line. Each token supplied after the class name can be read from args; the array may be empty. The name args is conventional and can be changed.

Example

```
public static void main(String[] args) {
    for (String arg : args) {
        System.out.println(arg);
    }
}
// Run: java Demo smoke chrome
```

23. Method overriding

Overriding occurs when a subclass provides a new implementation of an inherited instance method with the same signature and a compatible return type. It supports runtime polymorphism. Static methods are hidden rather than overridden; private methods are not inherited for overriding; final methods cannot be overridden.

Example

```
class Animal {
    void sound() { System.out.println("Animal sound"); }
}
class Dog extends Animal {
    @Override
    void sound() { System.out.println("Bark"); }
}
```

24. Scroll down the page

Prefer scrolling through normal user interaction when possible. JavaScriptExecutor can scroll the window or a specific container. If an element is the target, scroll it into view and then wait for it to be actionable.

Example

```
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("window.scrollTo(0, document.body.scrollHeight);");

// Scroll a particular element into view
WebElement target = driver.findElement(By.id("footerLink"));
js.executeScript("arguments[0].scrollIntoView({block:'center'});", target);
```

25. BDD and Cucumber framework

Behavior-Driven Development encourages collaboration among business, QA, and development by describing expected behavior in a shared language. Cucumber executes Gherkin feature files by matching Given/When/Then steps to step definitions. BDD is a practice; Cucumber is one tool that can support it. Avoid writing scenarios as mere implementation checklists; describe business behavior and observable outcomes.

Example

```
Feature: Search employees
  Scenario: Find an existing employee
    Given an employee record exists
    When I search by the employee ID
    Then the employee details are displayed
```

26. What is Bitbucket?

Bitbucket is a Git-based source-code hosting and collaboration platform from Atlassian. Teams use repositories, branches, pull requests, code review, access controls, and integrations with tools such as Jira and CI/CD systems. It serves a similar source-control hosting purpose to GitHub, although product features and integrations differ.

Example

```
git clone https://bitbucket.org/workspace/repository.git
git checkout -b feature/login-test
git add .
git commit -m "Add login test"
git push -u origin feature/login-test
```

27. CSS selector vs XPath

CSS selectors are concise and widely used for IDs, classes, attributes, and descendant relationships. XPath can navigate in both directions through the DOM, select by text, and use axes such as following-sibling. Neither is always faster in every browser; choose based on clarity, stability, and the DOM. Avoid brittle absolute paths and selectors tied to generated class names.

Example

```
// CSS
By.cssSelector("input[name='email']");

// XPath
By.xpath("//label[normalize-space()='Email']/following-sibling::input");
```

28. Extract only numbers from a string in Java

Use a regular expression to remove all characters that are not digits. If negative signs, decimals, or a particular locale matter, define the requirement first because the simple expression below retains digits only.

Example

```
String input = "Order#AB-1209, amount $45.75";
String digits = input.replaceAll("\\D", "");
System.out.println(digits); // 12094575
```

> **Interview tip:** This combines all digits into one string; it does not preserve separate numeric values or decimal points.

29. Difference between Selenium 3 and Selenium 2 / older versions

The question wording is ambiguous, so clarify which older version the interviewer means. Common Selenium 3 changes included moving toward the W3C WebDriver standard while retaining legacy JSON Wire Protocol support during the transition, and requiring browser drivers for many browsers. Selenium 4 completed the W3C WebDriver transition and introduced features such as relative locators, a redesigned Grid, and Selenium Manager in newer 4.x releases. Avoid saying Selenium 3 and Selenium 2 are identical.

Example

```
Illustrative comparison:
Selenium 2: WebDriver + Selenium RC-era integration; JSON Wire Protocol era
Selenium 3: WebDriver-focused; browser driver executables commonly configured
Selenium 4: W3C WebDriver; improved Grid; relative locators; Selenium Manager in 4.6+
```

> **Interview tip:** If asked specifically about Selenium 3 vs Selenium 2, answer that exact comparison rather than switching to Selenium 3 vs 4.

30. TestNG annotations

TestNG annotations define lifecycle methods and test methods. Common annotations include @BeforeSuite, @BeforeTest, @BeforeClass, @BeforeMethod, @Test, @AfterMethod, @AfterClass, @AfterTest, and @AfterSuite. Setup annotations run at different scopes. @DataProvider supplies data-driven test inputs; @Parameters can receive values from test configuration. The exact order depends on suite configuration and inheritance.

Example

```
@BeforeMethod
public void setUp() {
    // initialize browser
}

@Test
public void validLoginTest() {
    // test steps
}

@AfterMethod(alwaysRun = true)
public void tearDown() {
    // quit browser safely
}
```

31. Continuous Integration (CI) tools

CI automatically builds and tests changes when developers commit or merge code. Common tools include Jenkins, GitHub Actions, GitLab CI/CD, Azure Pipelines, and Bitbucket Pipelines. A Selenium pipeline typically checks out code, configures Java/Maven and browsers, runs tests, publishes reports, and archives logs/screenshots. Tests should be reliable, isolated, and configured for headless or remote browser execution where needed.

Example

```
# Example Maven command in a CI pipeline
mvn clean test
```

> **Interview tip:** Name the tools you have actually used and explain the pipeline you personally worked with.

32. When is a defect deferred?

A defect is deferred when the team decides not to fix it in the current release or sprint and plans to reconsider it later. Reasons may include low severity or priority, limited business impact, a workaround, release constraints, dependency on another change, or insufficient time. The decision should be approved by the appropriate product/business stakeholders, documented with rationale, and assigned a future target or review point. Deferral is not the same as rejecting a valid defect.

> **Interview tip:** Explain how you record severity, priority, impact, workaround, owner, and target release in your defect-tracking tool.

33. What is GitHub?

GitHub is a cloud platform for hosting Git repositories and collaborating on code. Git is the version-control system; GitHub adds remote repositories, pull requests, reviews, issues, permissions, and automation through GitHub Actions. A typical workflow is clone, branch, edit, commit, push, open a pull request, review, and merge.

Example

```
git clone https://github.com/org/project.git
git checkout -b feature/add-login-test
git add .
git commit -m "Add login automation test"
git push -u origin feature/add-login-test
```

Quick revision checklist

- Explain WebDriver architecture clearly and distinguish browser automation from native desktop automation.
- Know implicit wait vs explicit wait and why explicit waits are often preferred for dynamic elements.
- Practice file upload, frames, scrolling, JavaScriptExecutor, POM, and stable locator selection.
- Revise Java basics: main method, String[] args, final, static, overriding, regex.
- Practice SQL INNER JOIN and LEFT JOIN with sample tables.
- Know BDD vs Cucumber, TestNG lifecycle annotations, Git vs GitHub/Bitbucket, and CI concepts.
- Be honest about tools and projects you have personally used; adapt the examples to your actual experience.