Cucumber - hooks

In **Cucumber**, **hooks** are blocks of code that run **before or after** each **scenario**, **step**, or **feature**. They are used to **set up preconditions** or **clean up after test execution**. Hooks help you avoid repetitive code in your step definitions and keep the test logic separate from setup and teardown.

🔧 Types of Hooks in Cucumber (Java)

Hook Type	Method Annotation	Purpose
Before	`@Before`	Runs **before each scenario**
After	`@After`	Runs **after each scenario**
BeforeStep	`@BeforeStep`	Runs **before each step** in a scenario
AfterStep	`@AfterStep`	Runs **after each step** in a scenario

✅ Example: Basic `@Before` and `@After` Hook

```
java


import io.cucumber.java.Before;
import io.cucumber.java.After;

public class Hooks {

    @Before
    public void setUp() {
        System.out.println("🚀 Launching browser before each scenario");
        // e.g., open browser, initialize driver
    }

    @After
    public void tearDown() {
        System.out.println("🧹 Closing browser after each scenario");
        // e.g., close browser, clear cookies
    }
}
```

✅ Example: Conditional Hooks with Tags

You can run hooks **only for specific scenarios** using tags:

```
java


@Before("@Smoke")
public void setUpForSmokeTest() {
    System.out.println("Setup for Smoke Test Only");
}
```

Then use the tag in your `.feature` file:

```
gherkin


@Smoke
Scenario: Verify login functionality
  Given user is on login page
  When user enters valid credentials
  Then user is redirected to dashboard
```

✅ Example: `@BeforeStep` and `@AfterStep`

```
java


@BeforeStep
public void beforeEachStep() {
    System.out.println("➡ Before step");
}

@AfterStep
public void afterEachStep() {
    System.out.println("⬅ After step");
}
```

📦 Placement

Hooks should be placed in a separate **`Hooks.java`** file inside your **`stepDefinitions`** or **`hooks`** package.

💡 Use Cases

- Starting/stopping WebDriver
- Creating/deleting test data
- Database connection setup
- Logging test start/end
- Taking screenshots on failure

Would you like a mini-project that demonstrates hooks in action using Selenium + Cucumber?

In **Cucumber (Java)**, when you have **multiple hooks of the same type** (e.g., multiple `@Before` or multiple `@After` hooks), you can control the **order of execution** using the `order` attribute.

🔢 Hook Execution Order Rules

Hook Type	Execution Order
`@Before`	Executed in **ascending** order (lower number runs first)
`@After`	Executed in **descending** order (higher number runs first)

✅ Example: `@Before` Hooks Order

```
java

@Before(order = 1)
public void beforeHook1() {
    System.out.println("Before Hook 1");
}

@Before(order = 2)
public void beforeHook2() {
    System.out.println("Before Hook 2");
}
```

Output:

```
mathematica

Before Hook 1
Before Hook 2
```

✅ Example: `@After` Hooks Order

```
java

@After(order = 2)
public void afterHook2() {
    System.out.println("After Hook 2");
}

@After(order = 1)
public void afterHook1() {
    System.out.println("After Hook 1");
}
```

Output:

```
mathematica

After Hook 2
After Hook 1
```

⚠️ Default Order

If you don't specify `order`, the default is `order = 0`. All hooks with the same order value will be executed in arbitrary order.

📌 Practical Tip

Use the order feature to:

- Ensure **setup** steps like database connection happen **before** browser launch.
- Ensure **screenshots** or logging in `@After` happen **before** browser close.

Let me know if you want a complete demo with multiple ordered hooks using Selenium + Cucumber.

Using **hooks** in **Cucumber** is the best way to handle **test setup and cleanup** automatically before and after each scenario. This ensures your tests are **isolated**, **repeatable**, and **easy to manage**.

✅ Common Setup and Cleanup Tasks with Hooks

Purpose	Hook Type	Typical Code
Launch browser	`@Before`	WebDriver initialization
Load config	`@Before`	Read from properties file or env
Login as user	`@Before`	Pre-authenticate user
Capture screenshot	`@After`	On failure
Close browser	`@After`	Quit WebDriver
Clean test data	`@After`	Delete dummy records from DB

🔪 Example: Selenium Setup and Teardown using Hooks

- **`Hooks.java`**

```
java

import io.cucumber.java.Before;
import io.cucumber.java.After;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class Hooks {

    public static WebDriver driver;

    @Before
    public void setUp() {
        System.out.println("🔧 Setup: Launching Chrome browser");
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");
        driver = new ChromeDriver();
        driver.manage().window().maximize();
    }

    @After
    public void tearDown() {
        System.out.println("🧹 Teardown: Closing browser");
        if (driver != null) {
            driver.quit();
        }
    }
}
```

🔁 This **`driver`** can be used across your step definition classes by making it **`public static`**.

🖼️ Example: Capture Screenshot on Failure

To capture screenshots when a scenario fails:

- **Enhanced `@After` Hook**

```
java
```

```java
import io.cucumber.java.After;
import io.cucumber.java.Scenario;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;

@After
public void tearDown(Scenario scenario) {
    if (scenario.isFailed()) {
        byte[] screenshot = ((TakesScreenshot) Hooks.driver)
                            .getScreenshotAs(OutputType.BYTES);
        scenario.attach(screenshot, "image/png", "Failure Screenshot");
    }

    if (Hooks.driver != null) {
        Hooks.driver.quit();
    }
}
```

📌 Best Practices

- **Separate concerns**: Keep hooks in their own class.
- **Avoid hardcoded paths**: Use a config file or environment variables.
- **Minimize global state**: Share WebDriver only if necessary.
- **Use tags** if you want setup/cleanup only for specific scenarios.

Would you like a small Selenium + Cucumber project with setup/cleanup hooks implemented for practice?