

Way2Automation - Tutorial 7 – Hooks in Cucumber

What you will Learn :

- o About cucumber Hooks
- o Annotate tags with hooks
- o Practical demonstration
- o Multiple @Before with orders
- o Multiple @After with orders
- o Pass scenario object to the hooks' method
- o @BeforeStep
- o @AfterStep
- o Annotate tags with hooks demo
- o References

About cucumber Hooks

We will discuss various hooks available in cucumber. We can define setup methods (example: initializing driver, launch browser, read cookies etc) and teardown methods (example: close the browser, close the database connection etc) in the forms of hooks. Unlike 'Background' keyword that we studied in previous tutorial, 'Hooks' are not part of feature files.

Hooks can be written in :

- a) step definition file or
- b) a separate configuration class

@Before hook defines setup methods and @After defines teardown methods.

@Before will be executed before each scenario

@After will be executed after every scenario

Similarly:

@BeforeStep will be executed before each step of the scenario

@AfterStep will be executed after every step of the scenario

We can also have multiple @Before annotations. We can define the orders. Let us say we have 2 @Before annotations. The annotation that has 'order = 1' will be executed first before every scenario and the annotation that has 'order = 2' will be executed next before every scenario:

@Before

launching the browser method (**order = 1**)

@Before

initializing database method (**order = 2**)

Similarly we can have:

@After

closing the browser method (**order = 1**)

@After

disconnecting the database method (**order = 2**)

Annotate tags with hooks

Let us say we have 2 'search' scenarios tagged with @Smoke

@Smoke

Scenario: Search

@Smoke

Scenario: Advance Search

Let us also have another scenario tagged with @Regression

@Regression

Scenario: Mouse hover

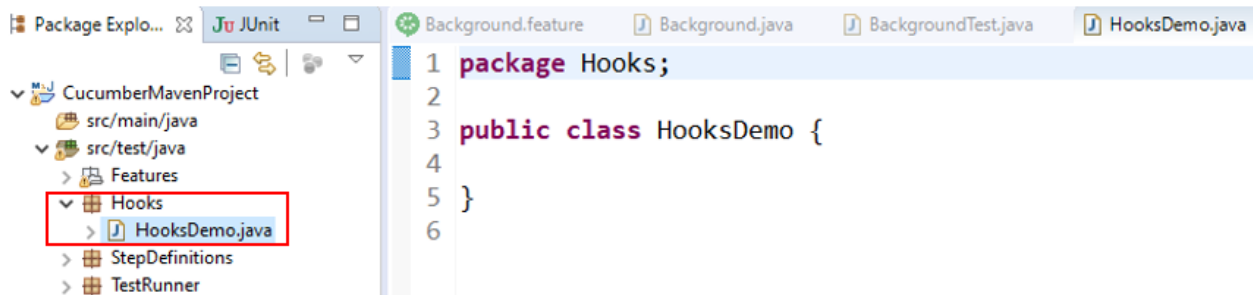
Now, we can create @Before annotation and you can tag them with hooks. So we can say:

@Before("@Smoke")

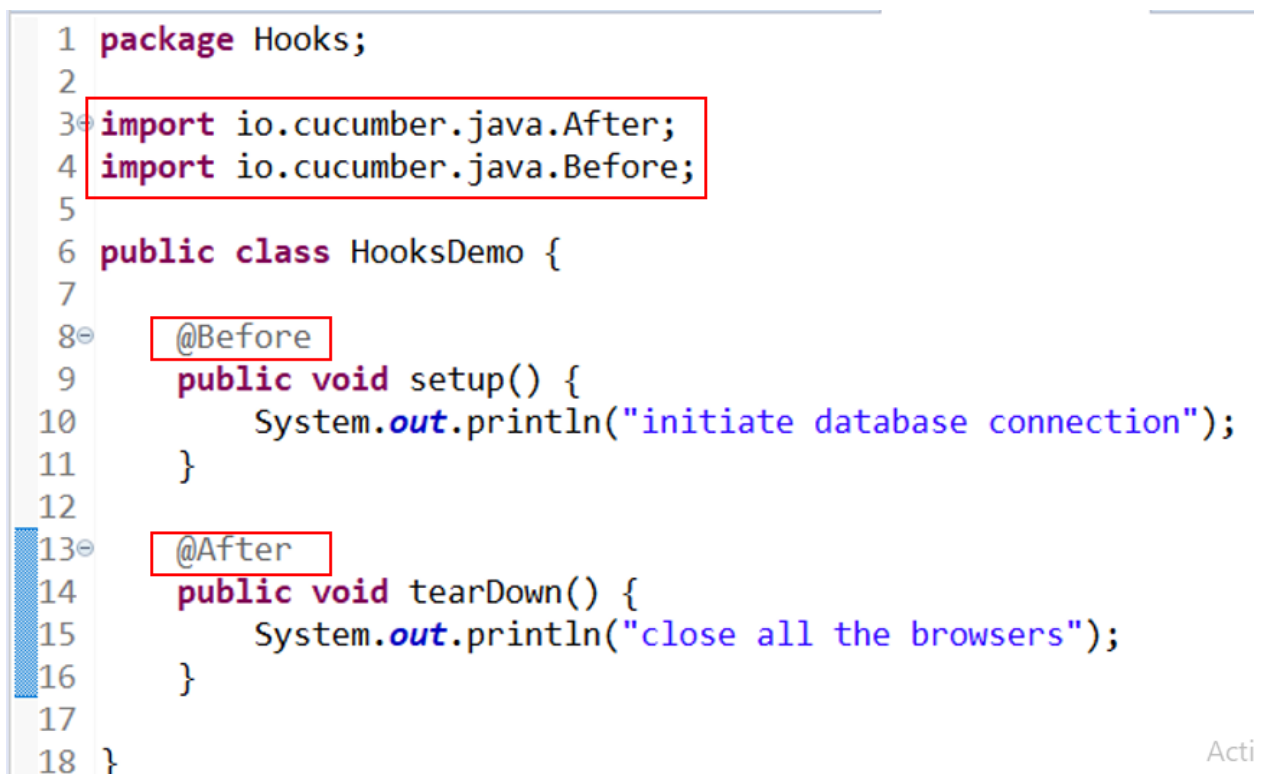
So it will execute only those scenarios which are tagged with @Smoke

Practical demonstration

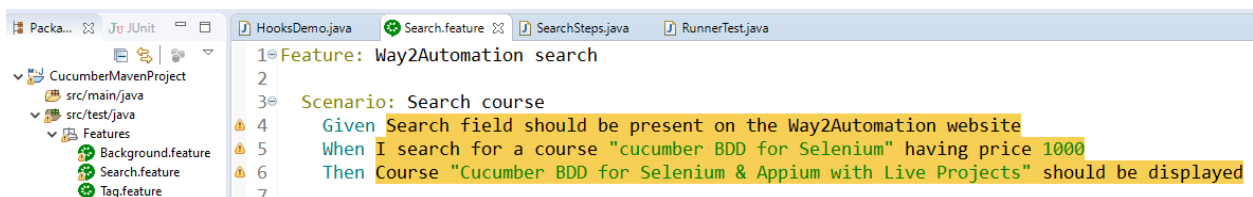
Create a package 'Hooks' under src/test/java. Inside the package, create a class 'HooksDemo'



Write 2 methods inside the class, see below. Make sure that you import the classes from cucumber library

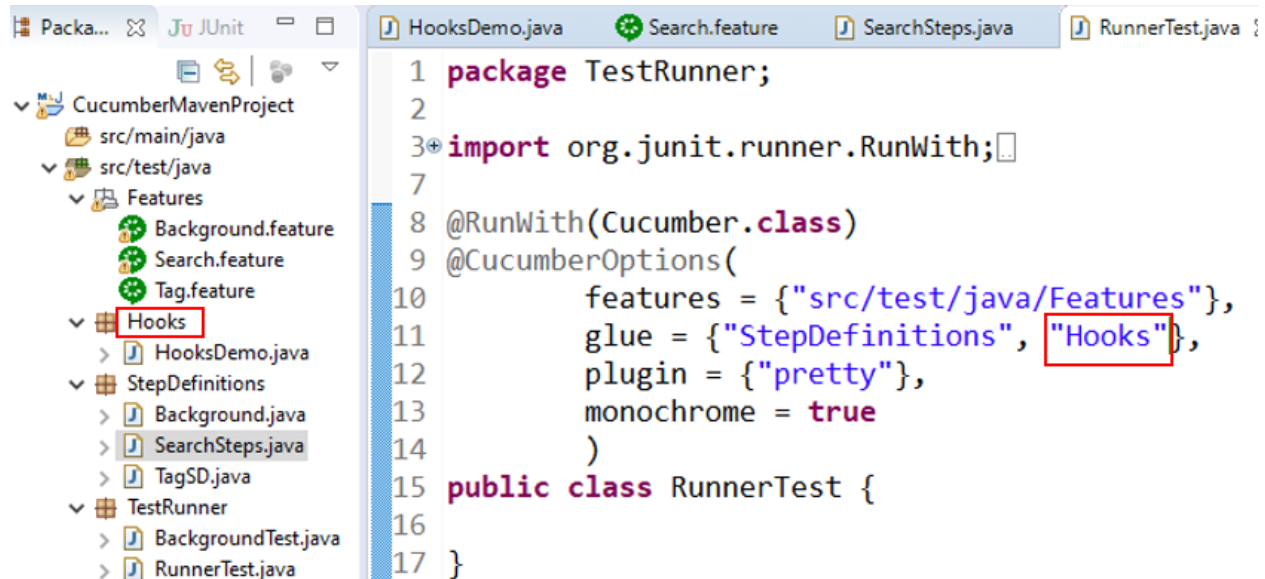


Let us look at the 'Search' feature that we had written in one of our previous tutorials. This feature has only 1 scenario



Now, before this particular scenario, @Before hook will be executed. After the scenario completion, @After hook will be executed.

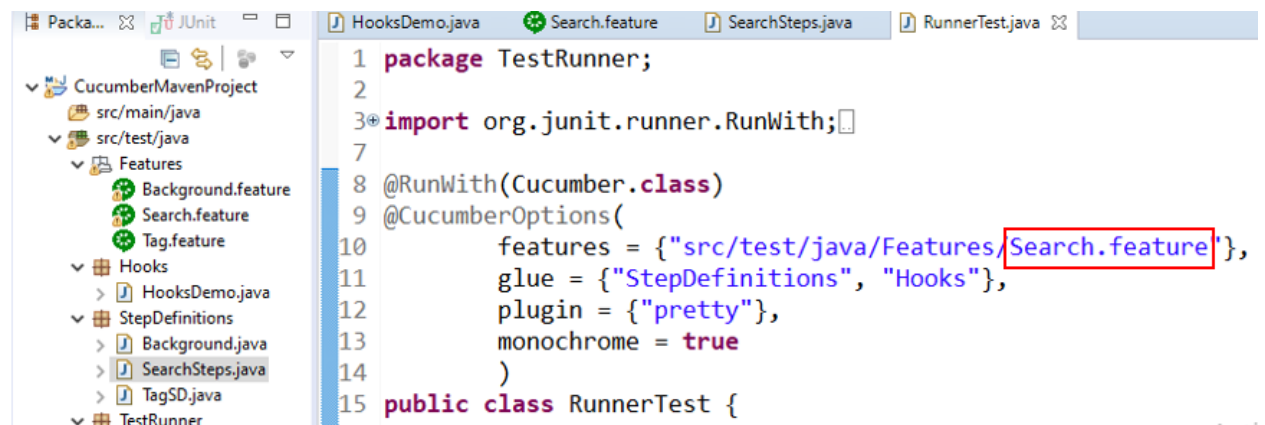
What we have to do is that, go to our 'Runner' file (see below) and mention the 'Hooks' package name in the 'glue' options



The screenshot shows an IDE with a project named 'CucumberMavenProject'. The left sidebar shows a tree view of the project structure. The 'src/test/java' directory is expanded, showing 'Features' and 'Hooks' subdirectories. The 'Hooks' directory is highlighted with a red box. The main editor shows the 'RunnerTest.java' file. The code is as follows:

```
1 package TestRunner;
2
3 import org.junit.runner.RunWith;
4
5 @RunWith(Cucumber.class)
6 @CucumberOptions(
7     features = {"src/test/java/Features"},
8     glue = {"StepDefinitions", "Hooks"},
9     plugin = {"pretty"},
10    monochrome = true
11 )
12 public class RunnerTest {
13
14 }
```

Also, mention the name of the feature file

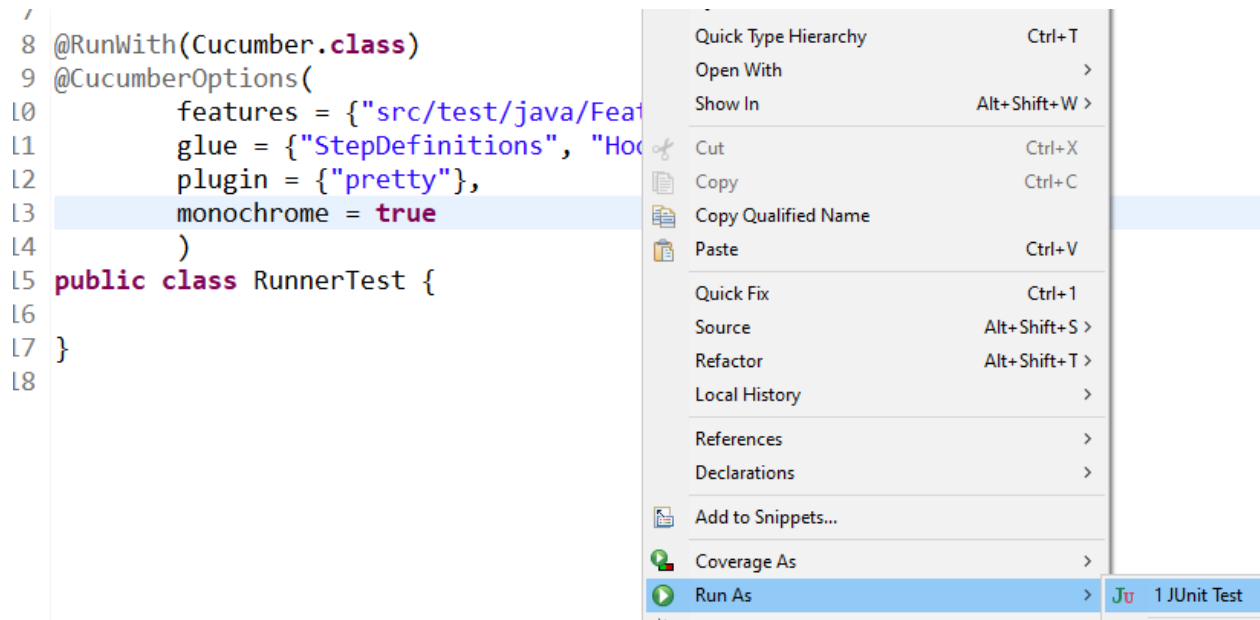


The screenshot shows the same IDE as before, but now the 'Search.feature' file is highlighted in the 'Features' directory in the left sidebar with a red box. The main editor shows the 'RunnerTest.java' file with the following code:

```
1 package TestRunner;
2
3 import org.junit.runner.RunWith;
4
5 @RunWith(Cucumber.class)
6 @CucumberOptions(
7     features = {"src/test/java/Features/Search.feature"},
8     glue = {"StepDefinitions", "Hooks"},
9     plugin = {"pretty"},
10    monochrome = true
11 )
12 public class RunnerTest {
13
14 }
```

Save the file

Let us run the RunnerTest.java



Notice the console o/p. The @Before hook is executed before all the scenario steps and @After hook is executed after all the scenario steps are over

```
Console
<terminated> RunnerTest [JUnit] C:\Program Files\Java\jdk1.8.0_191\bin\javaw.exe (27-Feb-2021, 2:26:51 PM)

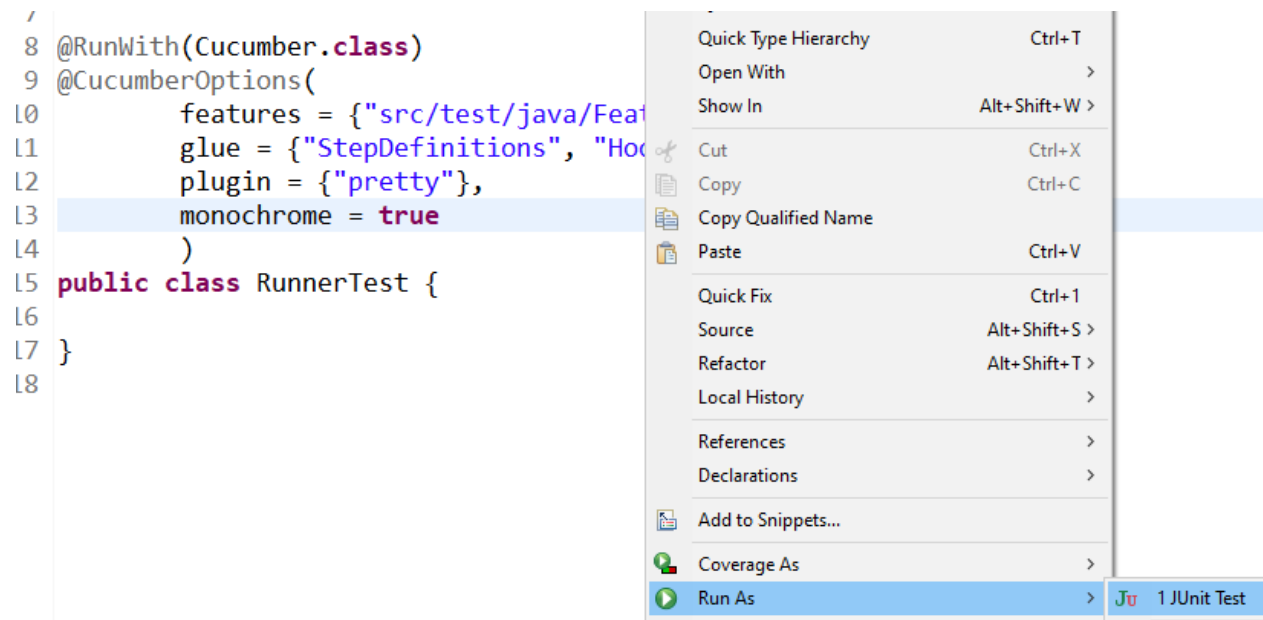
Scenario: Search course
initiate database connection
Step 1: I landed on search page
    Given Search field should be present on the Way2Automation website
Step 2: Search the course with name: cucumber BDD for Selenium price: 1000
    When I search for a course "cucumber BDD for Selenium" having price 1000
Step 3: course Cucumber BDD for Selenium & Appium with Live Projects is displayed
    Then Course "Cucumber BDD for Selenium & Appium with Live Projects" should be displayed
close all the browsers
```

Let us now add one more scenario to our feature file

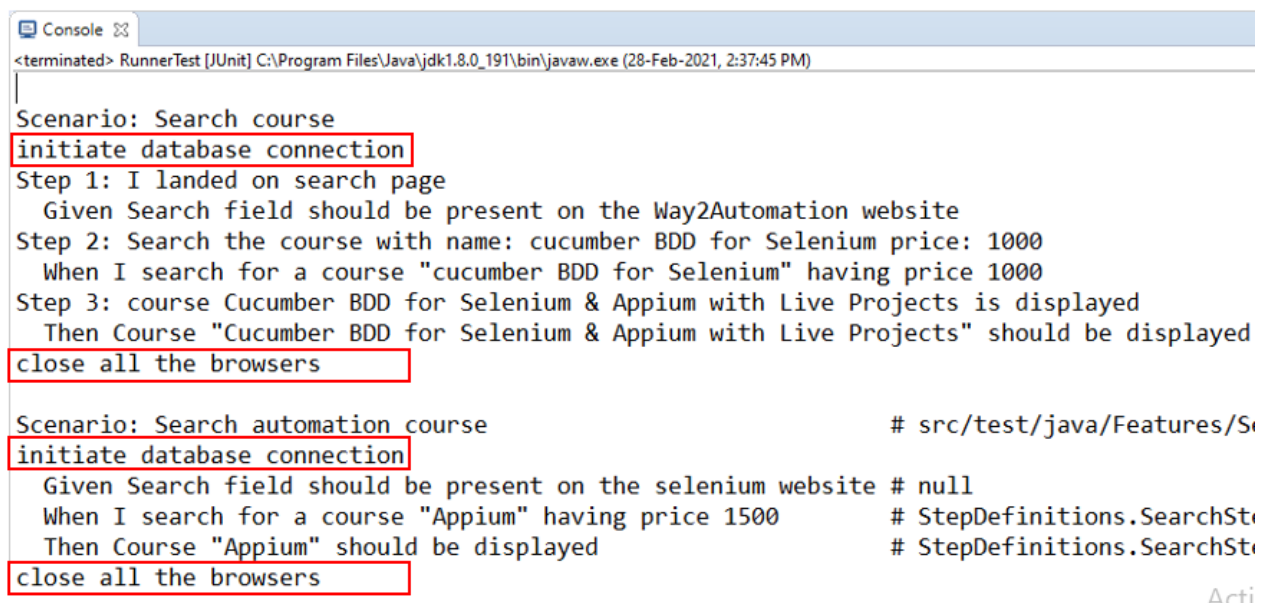
```
HooksDemo.java Search.feature SearchSteps.java RunnerTest.java
1 Feature: Way2Automation search
2
3 Scenario: Search course
4     Given Search field should be present on the Way2Automation website
5     When I search for a course "cucumber BDD for Selenium" having price 1000
6     Then Course "Cucumber BDD for Selenium & Appium with Live Projects" should be displayed
7
8 Scenario: Search automation course
9     Given Search field should be present on the selenium website
10    When I search for a course "Appium" having price 1500
11    Then Course "Appium" should be displayed
12
```

Save the file

Let us re-run the RunnerTest.java



Notice the console o/p. The `@Before` hook is executed before the starting of both the scenarios. Similarly, `@After` hook is executed after the completion of both the scenarios



Multiple `@Before` with orders

Let us have one more `@Before` hook with a different method, see below. Let us add the order number 1 and 2

```

HooksDemo.java  Search.feature  SearchSteps.java  RunnerTest.java
1 package Hooks;
2
3 import io.cucumber.java.After;
4
5
6 public class HooksDemo {
7
8     @Before(order = 1)
9     public void setup_db() {
10         System.out.println("initiate database connection");
11     }
12
13     @Before(order = 2)
14     public void setup_browser() {
15         System.out.println("launch browser");
16     }
17
18     @After
19     public void tearDown() {
20         System.out.println("close all the browsers");
21     }
22
23 }

```

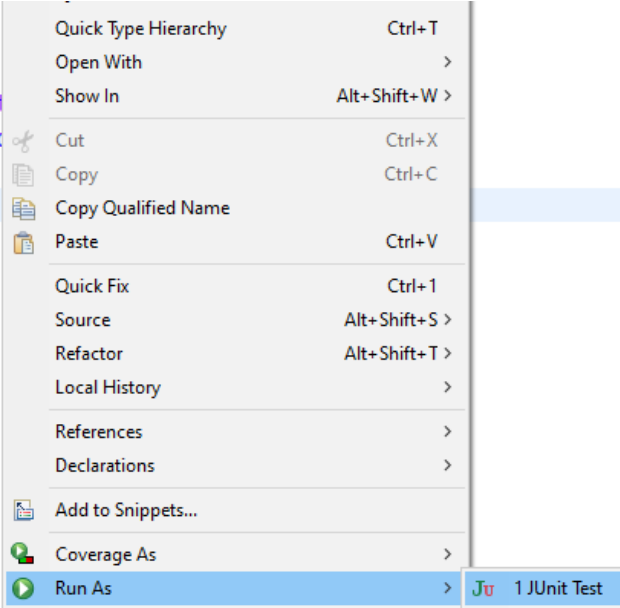
Save the file

Let us re-run the RunnerTest.java

```

/
8 @RunWith(Cucumber.class)
9 @CucumberOptions(
10     features = {"src/test/java/Feat
11     glue = {"StepDefinitions", "Hoo
12     plugin = {"pretty"},
13     monochrome = true
14 )
15 public class RunnerTest {
16
17 }
18

```



Notice the console o/p. The 2 @Before hook methods are executed as per the order given

```

Console
<terminated> RunnerTest [JUnit] C:\Program Files\Java\jdk1.8.0_191\bin\javaw.exe (28-Feb-2021, 2:53:32 PM)

Scenario: Search course
initiate database connection
launch browser
Step 1: I landed on search page
    Given Search field should be present on the Way2Automation website
Step 2: Search the course with name: cucumber BDD for Selenium price: 1000
    When I search for a course "cucumber BDD for Selenium" having price 1000
Step 3: course Cucumber BDD for Selenium & Appium with Live Projects is displayed
    Then Course "Cucumber BDD for Selenium & Appium with Live Projects" should be displayed
close all the browsers

Scenario: Search automation course # src/test/java/Features/Se
initiate database connection
launch browser
    Given Search field should be present on the selenium website # null
    When I search for a course "Appium" having price 1500 # StepDefinitions.SearchSte
    Then Course "Appium" should be displayed # StepDefinitions.SearchSte
close all the browsers

```

Multiple @After with orders

Let us now have 3 @After methods with different orders. In the case of @After hook, the orders work in revers order. So order=3 will be executed first, than order=2, finally order=1

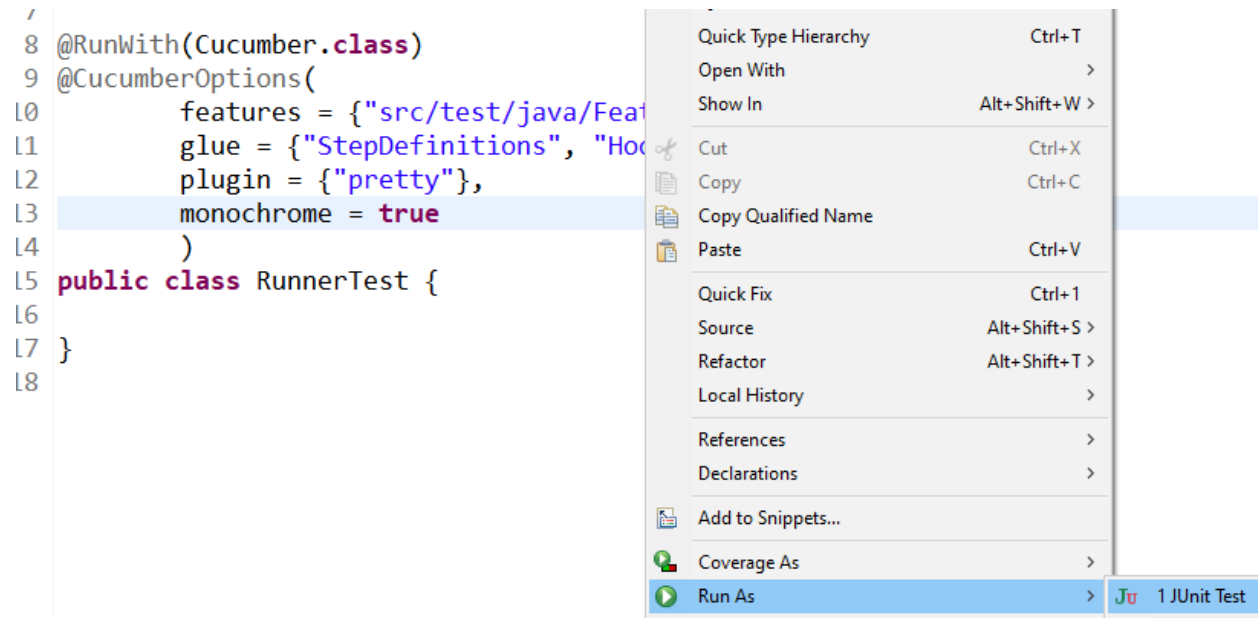
```

HooksDemo.java Search.feature SearchSteps.java RunnerTest.java
3* import io.cucumber.java.After;
5
6 public class HooksDemo {
7
8     @Before(order = 1)
9     public void setup_db() {
10         System.out.println("initiate database connection");
11     }
12
13     @Before(order = 2)
14     public void setup_browser() {
15         System.out.println("launch browser");
16     }
17
18     @After(order = 2)
19     public void tearDown_browser() {
20         System.out.println("close all the browsers");
21     }
22     @After(order = 1)
23     public void tearDown_database() {
24         System.out.println("disconnect from db");
25     }
26     @After(order = 3)
27     public void tearDown() {
28         System.out.println("exiting from system");
29     }

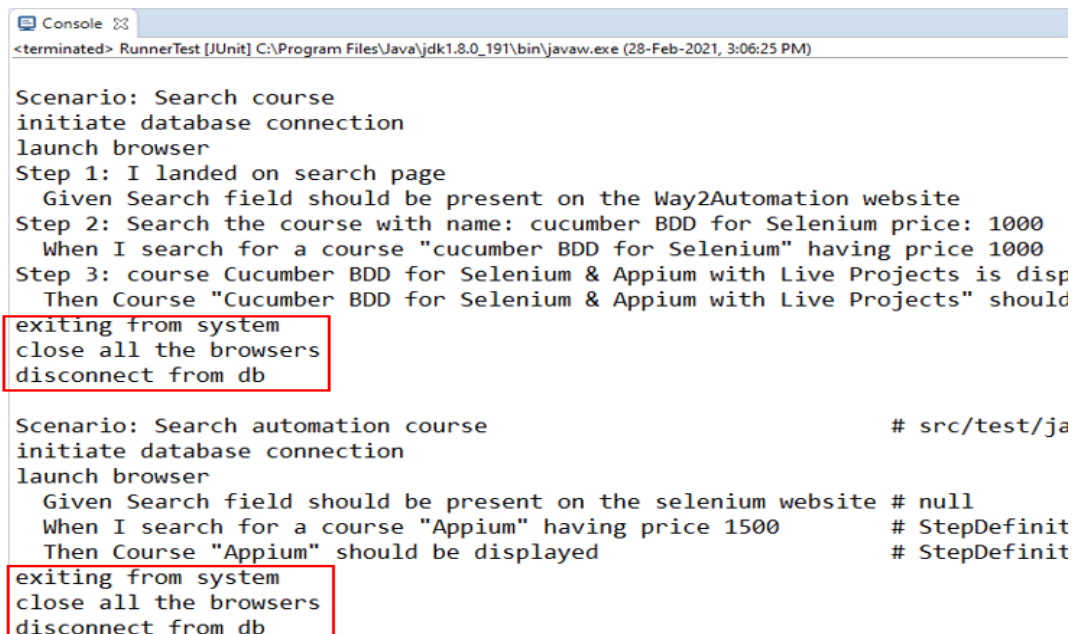
```


Save the file

Let us re-run the RunnerTest.java

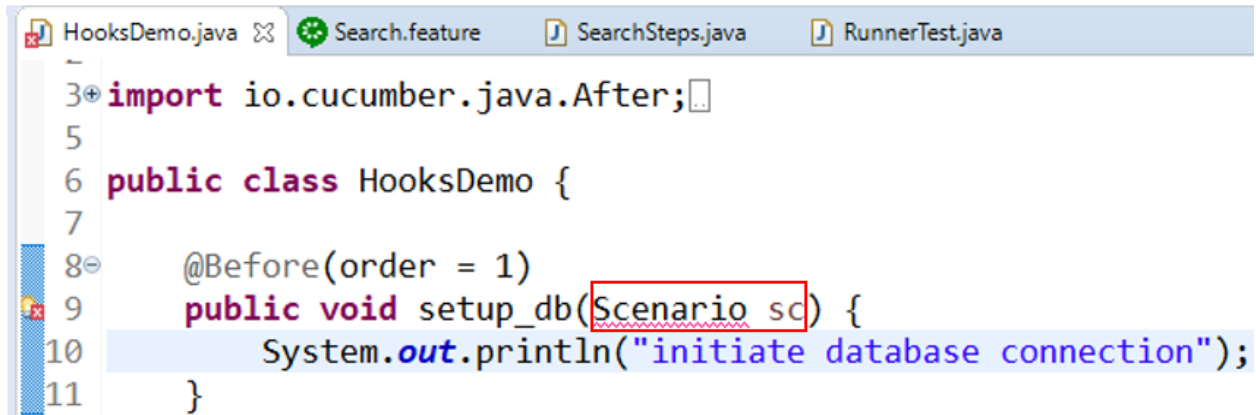


Notice the console o/p. The 3 @After hook methods are executed. Order=3 is executed first and then order=2 and finally order=1



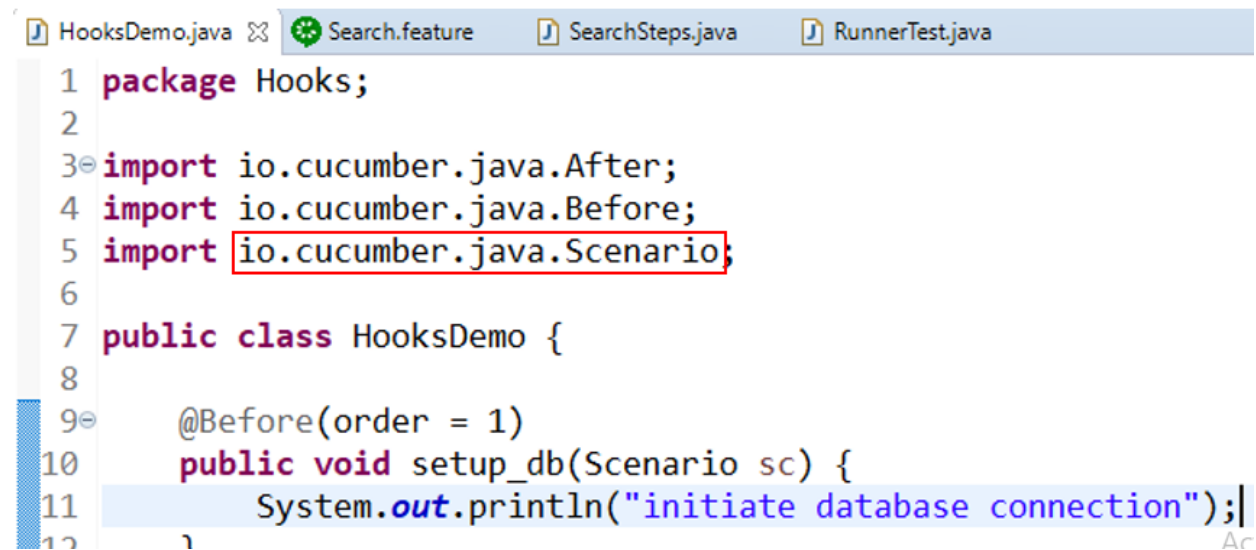
Pass scenario object to the hooks' method

We can pass scenario object to a method, see below



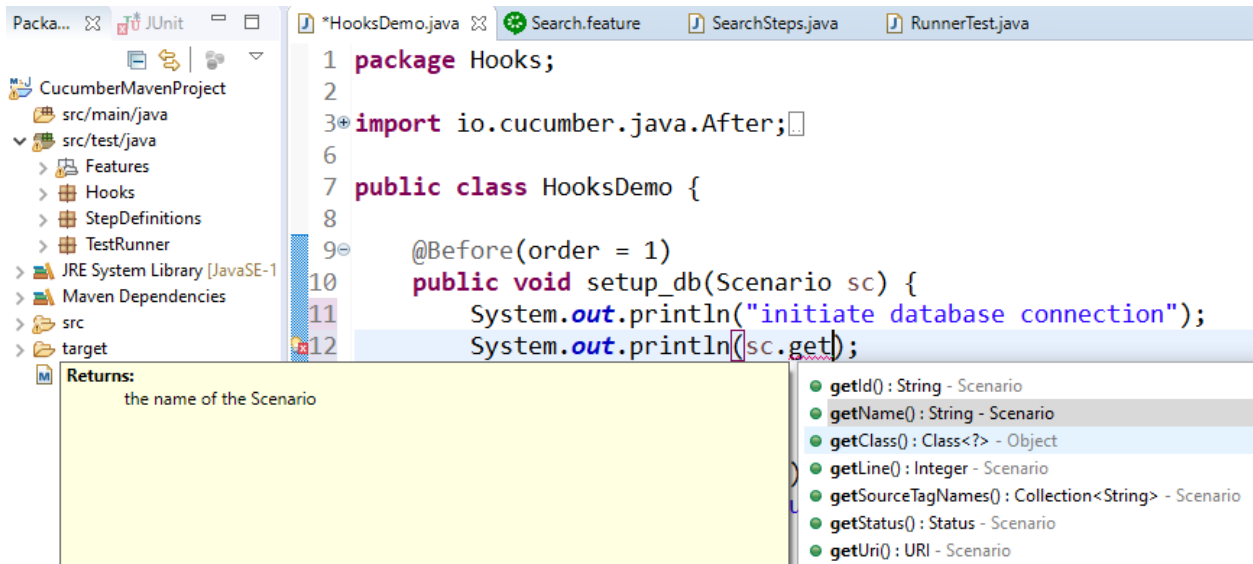
```
HooksDemo.java Search.feature SearchSteps.java RunnerTest.java
3+ import io.cucumber.java.After;
5
6 public class HooksDemo {
7
8     @Before(order = 1)
9     public void setup_db(Scenario sc) {
10         System.out.println("initiate database connection");
11     }
```

Import 'Scenario' from io.cucumber.java

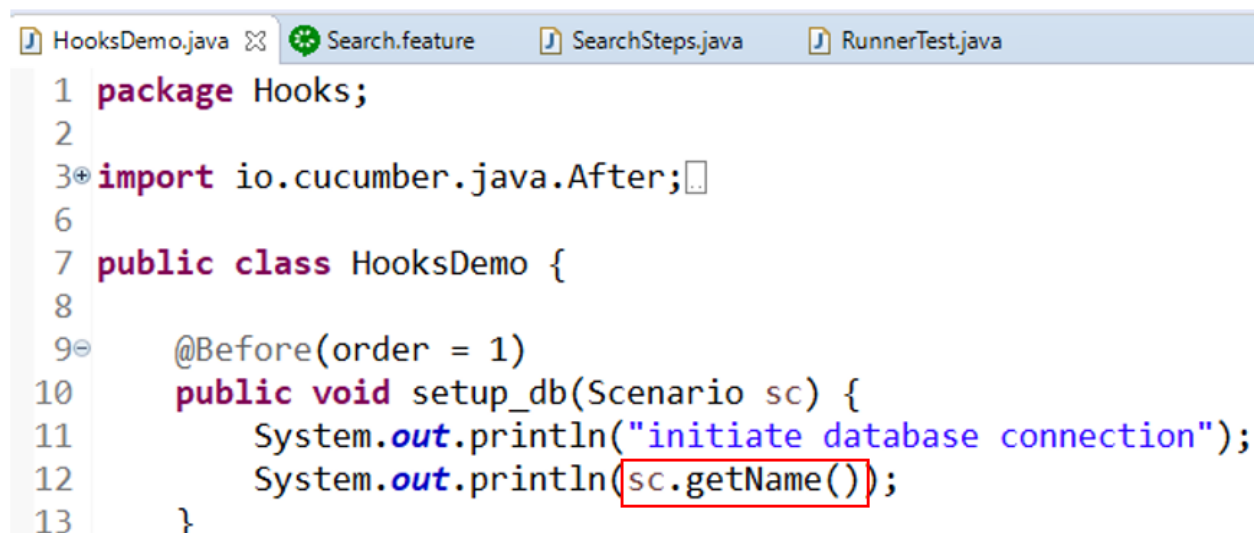


```
HooksDemo.java Search.feature SearchSteps.java RunnerTest.java
1 package Hooks;
2
3 import io.cucumber.java.After;
4 import io.cucumber.java.Before;
5 import io.cucumber.java.Scenario;
6
7 public class HooksDemo {
8
9     @Before(order = 1)
10     public void setup_db(Scenario sc) {
11         System.out.println("initiate database connection");
12     }
```

At runtime, we can now use this object to return us the scenario id, scenario name etc



So let us get the scenario name

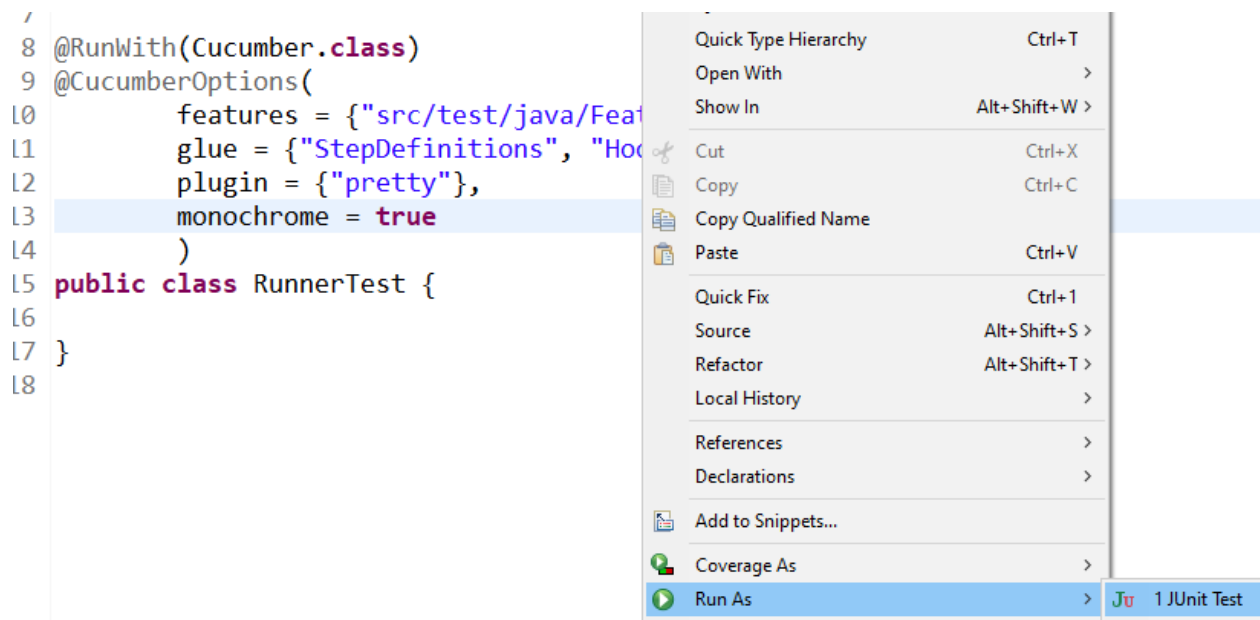


Similarly let us do the same thing in for @After method

```
@After(order = 2)
public void tearDown_browser(Scenario sc) {
    System.out.println("close all the browsers");
    System.out.println(sc.getName());
}
```

Save the file

Let us re-run the RunnerTest.java



Notice the console o/p. The respective scenario names are getting printed for the 2 scenarios

```

Console
<terminated> RunnerTest [JUnit] C:\Program Files\Java\jdk1.8.0_191\bin\javaw.exe (28-Feb-2021, 3:27:31 PM)
Scenario: Search course
initiate database connection
Search course
launch browser
Step 1: I landed on search page
    Given Search field should be present on the Way2Automation website
Step 2: Search the course with name: cucumber BDD for Selenium price: 1000
    When I search for a course "cucumber BDD for Selenium" having price 1000
Step 3: course Cucumber BDD for Selenium & Appium with Live Projects is displayed
    Then Course "Cucumber BDD for Selenium & Appium with Live Projects" should be displayed
exiting from system
close all the browsers
Search course
disconnect from db

Scenario: Search automation course # src/test/java/Features/S
initiate database connection
Search automation course
launch browser
    Given Search field should be present on the selenium website # null
    When I search for a course "Appium" having price 1500 # StepDefinitions.SearchSt
    Then Course "Appium" should be displayed # StepDefinitions.SearchSt
exiting from system
close all the browsers
Search automation course
disconnect from db

```

@BeforeStep

Let us write @BeforeStep

```
HooksDemo.java Search.feature SearchSteps.java RunnerTest.java
/
8 public class HooksDemo {
9
10 @Before(order = 1)
11 public void setup_db(Scenario sc) {
12     System.out.println("initiate database
13     System.out.println(sc.getName());
14 }
15
16 @Before(order = 2)
17 public void setup_browser() {
18     System.out.println("launch browser");
19 }
20
21 @BeforeStep
22 public void take_screenshot() {
23     System.out.println("took screenshot");
24 }
25 }
```

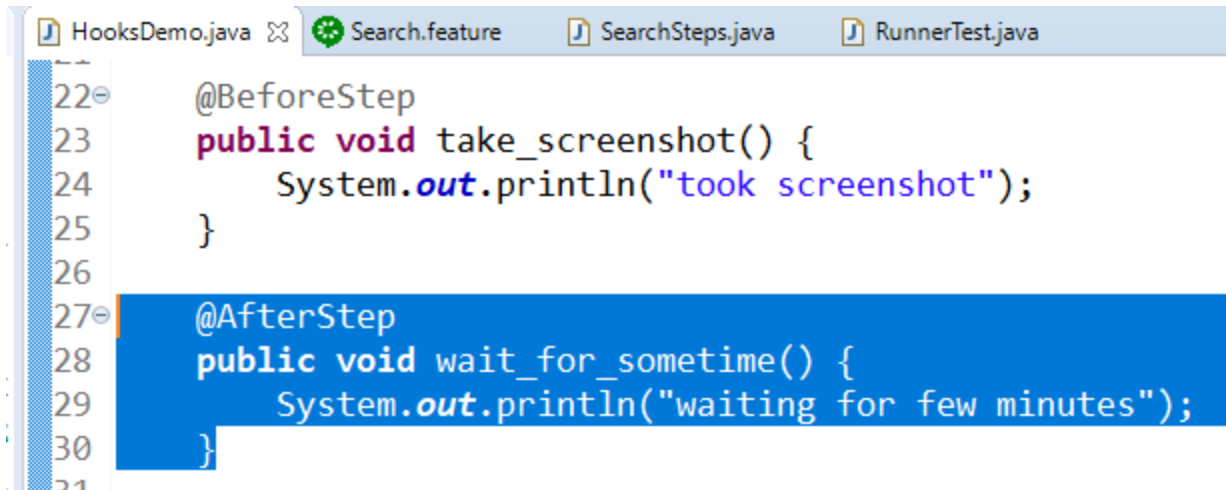
Save the file and run RunnerTest.java

Notice below that @BeforeStep is executed before each step

```
Console
<terminated> RunnerTest [JUnit] C:\Program Files\Java\jdk1.8.0_191\bin\javaw.exe (28-
Scenario: Search course
initiate database connection
Search course
launch browser
took screenshot
Step 1: I landed on search page
    Given Search field should be present on the
took screenshot
Step 2: Search the course with name: cucumber
    When I search for a course "cucumber BDD for
took screenshot
Step 3: course Cucumber BDD for Selenium & App
    Then Course "Cucumber BDD for Selenium & App
```

@AfterStep

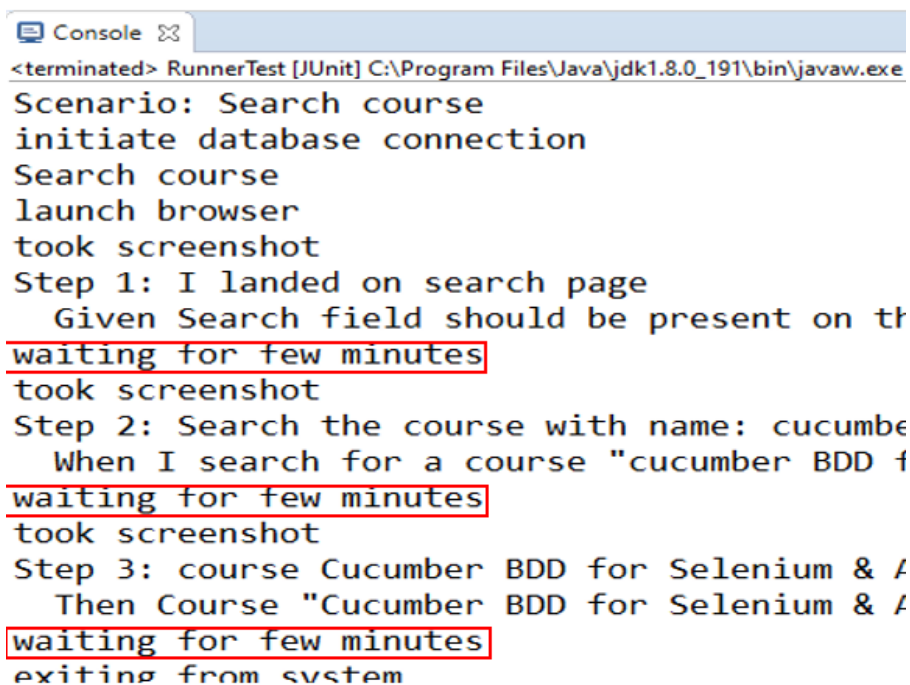
Let us now write @AfterStep



```
HooksDemo.java Search.feature SearchSteps.java RunnerTest.java
22 @BeforeStep
23 public void take_screenshot() {
24     System.out.println("took screenshot");
25 }
26
27 @AfterStep
28 public void wait_for_sometime() {
29     System.out.println("waiting for few minutes");
30 }
31
```

Save the file and run RunnerTest.java

Notice below that @AfterStep is executed after each step



```
Console
<terminated> RunnerTest [JUnit] C:\Program Files\Java\jdk1.8.0_191\bin\javaw.exe
Scenario: Search course
initiate database connection
Search course
launch browser
took screenshot
Step 1: I landed on search page
    Given Search field should be present on the page
    waiting for few minutes
    took screenshot
Step 2: Search the course with name: cucumber BDD for Selenium & /
    When I search for a course "cucumber BDD for Selenium & /"
    waiting for few minutes
    took screenshot
Step 3: course Cucumber BDD for Selenium & /
    Then Course "Cucumber BDD for Selenium & /" should be present
    waiting for few minutes
    exiting from system
```

Annotate tags with hooks demo

In our feature file, let us tag the 2 scenarios by tagnames @Smoke and @Production

```
HooksDemo.java Search.feature SearchSteps.java RunnerTest.java
1 Feature: Way2Automation search
2
3 @Smoke
4 Scenario: Search course
5   Given Search field should be present on the Way2Automation website
6   When I search for a course "cucumber BDD for Selenium" having price 1000
7   Then Course "Cucumber BDD for Selenium & Appium with Live Projects" should be displayed
8
9 @Production
10 Scenario: Search automation course
11   Given Search field should be present on the selenium website
12   When I search for a course "Appium" having price 1500
13   Then Course "Appium" should be displayed
```

Save the file

In our hook file, comment all the methods and just retain @Before and @After hooks, see below. Now pass the tagname @Smoke to both of these hooks

```
HooksDemo.java Search.feature SearchSteps.java RunnerTest.java
1 package Hooks;
2
3 import io.cucumber.java.After;
4
5
6
7
8
9 public class HooksDemo {
10
11     @Before("@Smoke")
12     public void setup_db() {
13         System.out.println("initiate database connection");
14         //System.out.println(sc.getName());
15     }
16
17     @After("@Smoke")
18     public void tearDown_database() {
19         System.out.println("disconnect from db");
20     }
21 }
```

Save the file

Run the RunnerTest.java.

Notice below that @Before and @After hooks are executed only for the scenario tagged with @Smoke


```

Console
<terminated> RunnerTest [JUnit] C:\Program Files\Java\jdk1.8.0_191\bin\javaw.exe (28-Feb-2021, 3:57:31 PM)
@Smoke
Scenario: Search course
initiate database connection
Step 1: I landed on search page
    Given Search field should be present on the Way2Automation website
Step 2: Search the course with name: cucumber BDD for Selenium price: 1000
    When I search for a course "cucumber BDD for Selenium" having price 1000
Step 3: course Cucumber BDD for Selenium & Appium with Live Projects is displayed
    Then Course "Cucumber BDD for Selenium & Appium with Live Projects" should be di
disconnect from db

@Production
Scenario: Search automation course                                # src/test/java/Fea
    Given Search field should be present on the selenium website # null
    When I search for a course "Appium" having price 1500         # StepDefinitions.S
    Then Course "Appium" should be displayed                       # StepDefinitions.S

```

Let us now change the tagname to @Production

```

HooksDemo.java Search.feature SearchSteps.java RunnerTest.java
1 package Hooks;
2
3 import io.cucumber.java.After;
4
5
6
7
8
9 public class HooksDemo {
10
11     @Before("@Production")
12     public void setup_db() {
13         System.out.println("initiate database connection");
14         //System.out.println(sc.getName());
15     }
16
17     @After("@Production")
18     public void tearDown_database() {
19         System.out.println("disconnect from db");
20     }

```

Save the file

Run the RunnerTest.java.

Notice below that @Before and @After hooks are executed only for the scenario tagged with @Production


```

Console
<terminated> RunnerTest [JUnit] C:\Program Files\Java\jdk1.8.0_191\bin\javaw.exe (28-Feb-2021, 4:05:20 PM)
@Smoke
Scenario: Search course
Step 1: I landed on search page
    Given Search field should be present on the Way2Automation website
Step 2: Search the course with name: cucumber BDD for Selenium price: 1000
    When I search for a course "cucumber BDD for Selenium" having price 1000
Step 3: course Cucumber BDD for Selenium & Appium with Live Projects is displayed
    Then Course "Cucumber BDD for Selenium & Appium with Live Projects" should be displayed

@Production
Scenario: Search automation course                                     # src/test/java/Features/Se
initiate database connection                                         # null
    Given Search field should be present on the selenium website # null
    When I search for a course "Appium" having price 1500          # StepDefinitions.SearchSte
    Then Course "Appium" should be displayed                        # StepDefinitions.SearchSte
disconnect from db

```

You will get the same results even if you define the 'tags' option in RunnerTest.java, see below

```

HooksDemo.java  Search.feature  SearchSteps.java  RunnerTest.java
1 package TestRunner;
2
3 import org.junit.runner.RunWith;
7
8 @RunWith(Cucumber.class)
9 @CucumberOptions(
10     features = {"src/test/java/Features/Search.feature"},
11     glue = {"StepDefinitions", "Hooks"},
12     plugin = {"pretty"},
13     tags = "@Smoke or @Production",
14     monochrome = true
15 )
16
17 public class RunnerTest {
18
19 }

```

References

You can refer www.baeldung.com/java-cucumber-hooks for related documentation



1. Introduction

Cucumber hooks can come in handy when we want to perform specific actions for every scenario or step, but without having these actions explicitly in the Gherkin code.

In this tutorial, we'll look at the `@Before`, `@BeforeStep`, `@AfterStep`, and `@After` Cucumber hooks.

2. Overview of Hooks in Cucumber

2.1. When Should Hooks Be Used?

Hooks can be used to perform background tasks that are not part of business functionality. Such tasks could be:

- Starting up a browser
- Setting or clearing cookies
- Connecting to a database

Thank you for reading!