SYNTAX TECHNOLOGIES

Cucumber

Class 2

Agenda

Cucumber Tags

What are the Hooks in Cucumber

Background in Cucumber

Cucumber Test Runner Class

We can write n-number of scenarios and test steps and steps definitions using cucumber, but cucumber does not not implicitly run your test scenarios. We have to provide what we want to execute in cucumber.

Test Runner class helps us to run the cucumber scenarios. This is the starting point of the Execution of Cucumber.

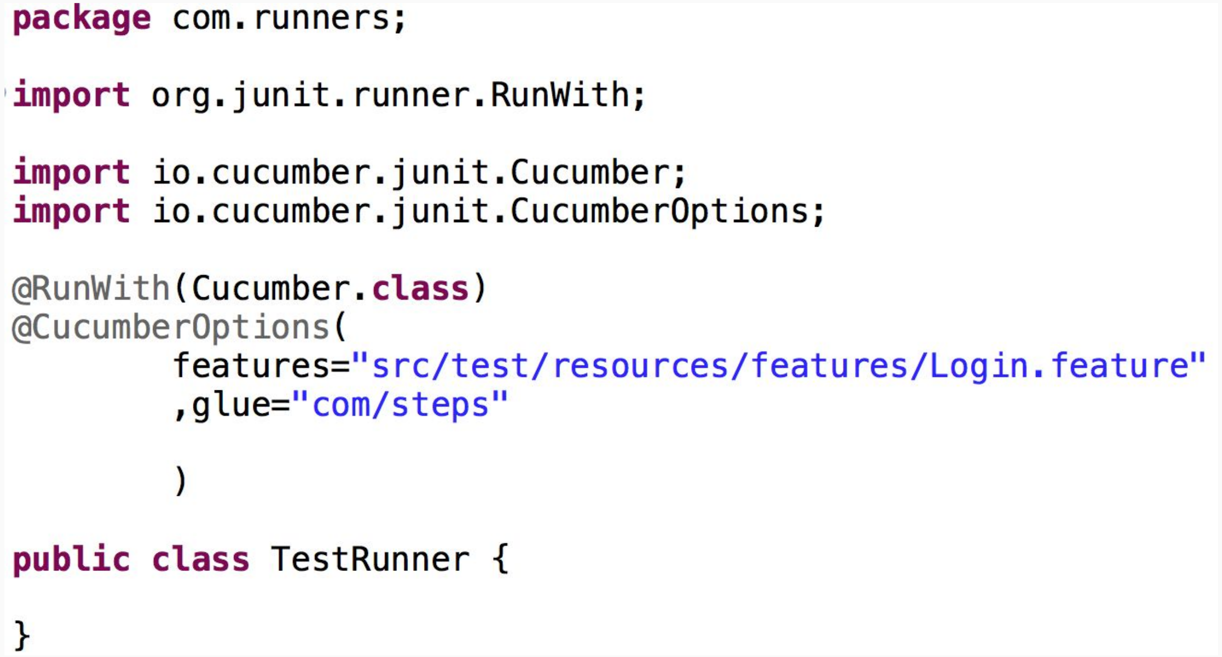
The execution point of the cucumber test scenarios depends on the framework that we are using for executing the tests it could be either JUnit or TestNG. We will be using the **JUnit runner.**

With a test runner class, you have the option to run either a single feature file, or multiple feature files as well.

Cucumber Test Runner Class

We have to import the Runner class from the JUnit and also we have to provide where the feature files are present.

We have to pass Feature file location and the steps related to those Feature file with CucumberOptions annotation or decorator



```
package com.runners;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)
@CucumberOptions(
        features="src/test/resources/features/Login.feature"
        ,glue="com/steps"

        )

public class TestRunner {

}
```

What is Cucumber Options ?

@CucumberOptions are settings for our test.

There are several options we can use to specify: path to the feature files and step definitions, reporting format etc.

Options Type	Purpose	Default Value
features	set: Path to the Feature Files	{ }
glue	set: Path to the Step Definitions	{ }
monochrome	true: Display console Output is more readable format	false
dryRun	true: checks if all the Gherkin Steps have implemented steps	false
tags	instruct: What tags in feature files should be executed	{ }
plugin	set: What all report formats to use	false

Cucumber Hooks

Cucumber Hooks are blocks of code that run before or after each scenario.

We can define them anywhere in our project or step definition layers, using the methods @Before and @After.

Cucumber Hooks allows us to better manage the code workflow and helps us to reduce the code redundancy.

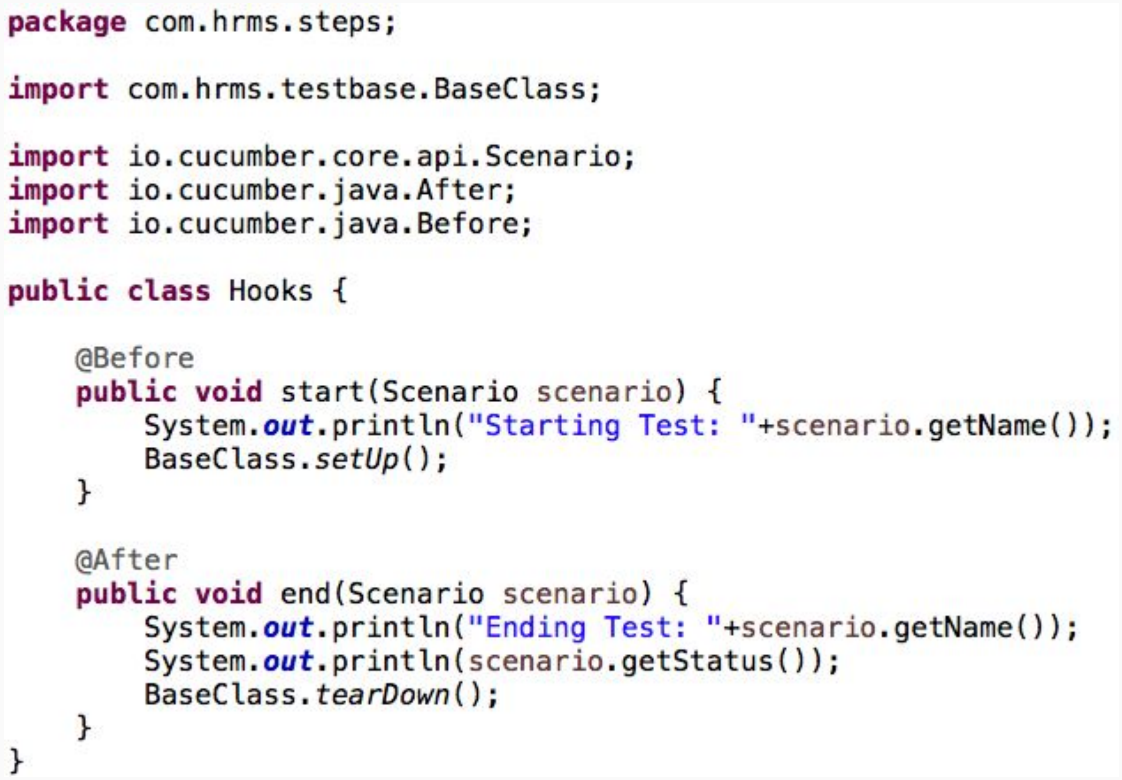
@Before hooks will be run before the first step of each scenario. They will run in the same order of which they are registered.

@After hooks will be run after the last step of each scenario, even when there are failing, undefined, pending or skipped steps. They will run in the opposite order of which they are registered.

Cucumber supports only two hooks (Before & After) which works at the start and the end of the test scenario.

We just need to define hooks, no need to associate the hooks, cucumber takes care of associating.

Cucumber Hooks



```
package com.hrms.steps;

import com.hrms.testbase.BaseClass;

import io.cucumber.core.api.Scenario;
import io.cucumber.java.After;
import io.cucumber.java.Before;

public class Hooks {

    @Before
    public void start(Scenario scenario) {
        System.out.println("Starting Test: "+scenario.getName());
        BaseClass.setUp();
    }

    @After
    public void end(Scenario scenario) {
        System.out.println("Ending Test: "+scenario.getName());
        System.out.println(scenario.getStatus());
        BaseClass.tearDown();
    }
}
```

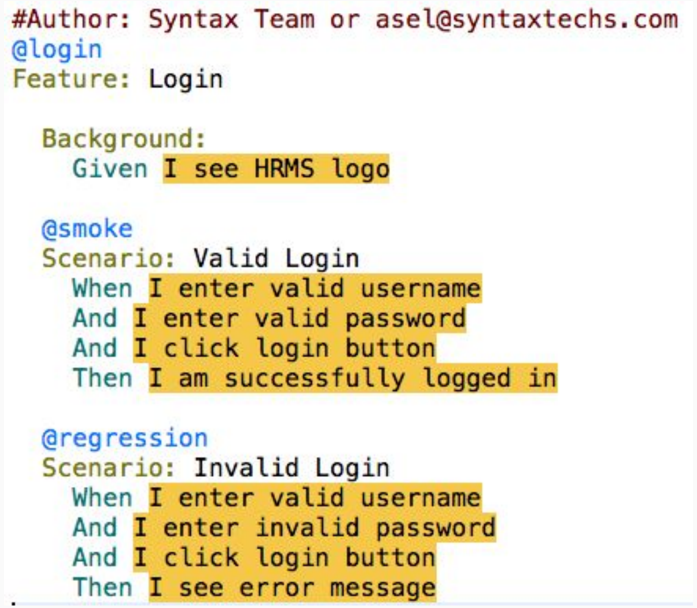
Note: After hook will get executed even when test case fails

Background Keyword in Cucumber

Background in Cucumber is used to define a step or series of steps which are common to all the tests in the feature file.

Background steps will be executed for all the scenarios present in the Gherkin feature file.

A Background is much like a scenario containing a number of steps. But it runs before each and every scenario where for a feature in which it is define.



```
#Author: Syntax Team or asel@syntaxtechs.com
@login
Feature: Login

  Background:
    Given I see HRMS logo

  @smoke
  Scenario: Valid Login
    When I enter valid username
    And I enter valid password
    And I click login button
    Then I am successfully logged in

  @regression
  Scenario: Invalid Login
    When I enter valid username
    And I enter invalid password
    And I click login button
    Then I see error message
```

Cucumber Tags

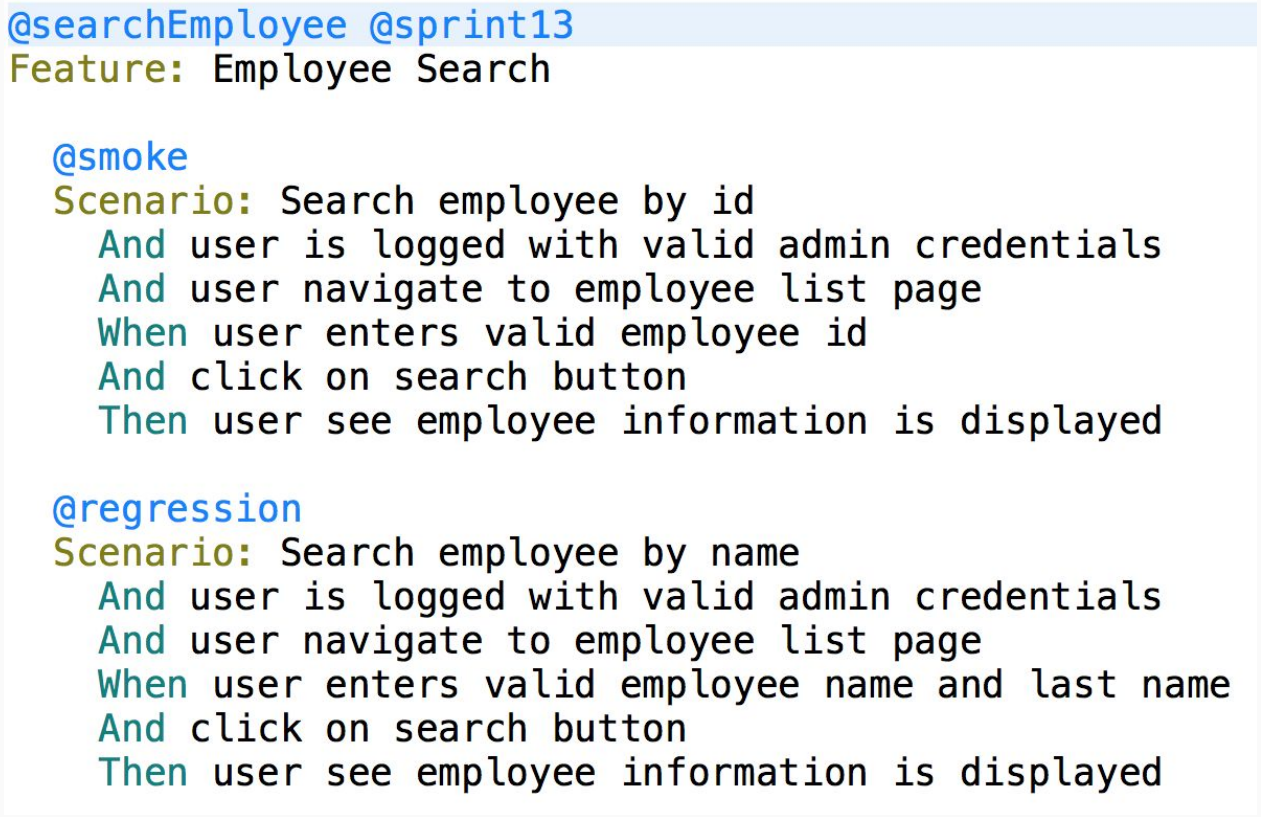
In cucumber projects, there will be many scenarios in a single features file. We should be creating a feature files based on the application feature or based on the functionality.

When we have many different feature files which cover all the different functionalities of the application and we want to execute just a Smoke or Regression Tests we can use Cucumber Tags.

Tag starts with @, followed by tag name like regression test or smoke test or anything we wish, our tag will look like @SanityTests just above the scenario keyword.

One scenario can have more than one tag separated by space.

Cucumber Tags



```
@searchEmployee @sprint13
Feature: Employee Search

  @smoke
  Scenario: Search employee by id
    And user is logged with valid admin credentials
    And user navigate to employee list page
    When user enters valid employee id
    And click on search button
    Then user see employee information is displayed

  @regression
  Scenario: Search employee by name
    And user is logged with valid admin credentials
    And user navigate to employee list page
    When user enters valid employee name and last name
    And click on search button
    Then user see employee information is displayed
```

Cucumber Tags

We have to specify the tag name which want to run in the cucumber runner using tags = {"@Smoke"} in CucumberOptions.

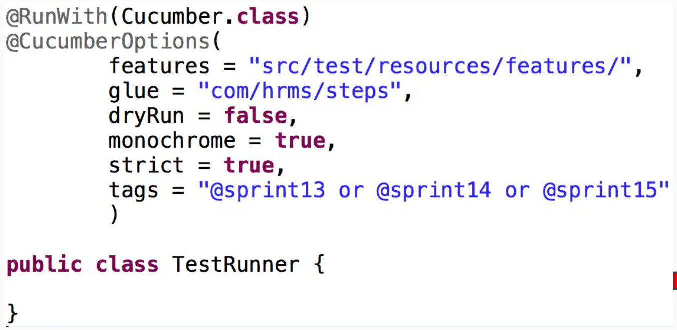
Sometimes we might need to run more than one tags at a time, in such cases we can use AND & OR to combine the cucumber tags to run the feature files.

OR: Runs the scenario if it has at least one give tag, there are separated with comma, all the tags will be include in one double quotes like **{"@sprint11 or @sprint12 or @sprint13"}**

AND: Runs the scenario if it has all the given tags, all the tags are separated with double quotes **{"@smoke and @sprint13"}**

Sometimes we might need to skip tags in cucumber BDD, we can use a special Character **~** to skip the tags. This also work both for Scenarios and Features, this can also works along with AND or OR.

Cucumber Tags

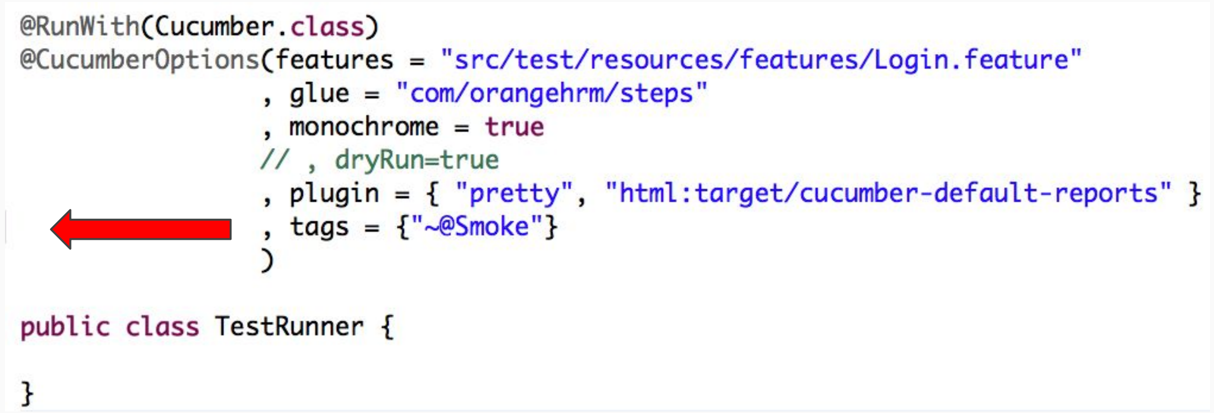


```java
@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/test/resources/features/",
        glue = "com/hrms/steps",
        dryRun = false,
        monochrome = true,
        strict = true,
        tags = "@sprint13 or @sprint14 or @sprint15"
        )

public class TestRunner {

}
```

Executing scenarios from sprint 13, 14 and sprint 15



```java
@RunWith(Cucumber.class)
@CucumberOptions(features = "src/test/resources/features/Login.feature"
                , glue = "com/orangehrm/steps"
                , monochrome = true
                // , dryRun=true
                , plugin = { "pretty", "html:target/cucumber-default-reports" }
                , tags = {"~@Smoke"}
                )

public class TestRunner {

}
```

Excluding Smoke scenarios