SYNTAX TECHNOLOGIES

Cucumber

Class 2

Agenda

Cucumber Tags

What are the Hooks in Cucumber

Background in Cucumber

Cucumber Tags

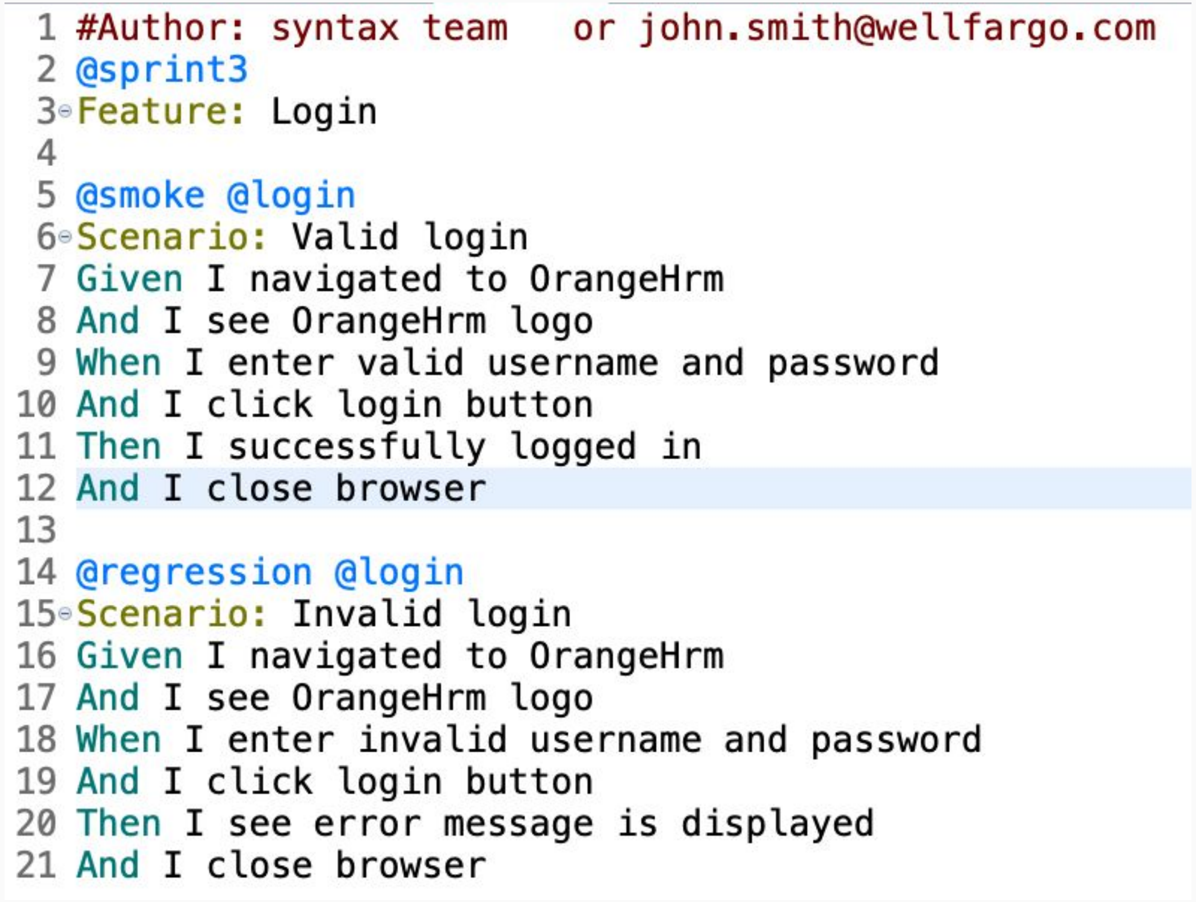
In cucumber projects, there will be many scenarios in a single features file. We should be creating a feature files based on the application feature or based on the functionality.

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



```
#Author: syntax team   or john.smith@wellfargo.com
@sprint3
Feature: Login

@smoke @login
Scenario: Valid login
Given I navigated to OrangeHrm
And I see OrangeHrm logo
When I enter valid username and password
And I click login button
Then I successfully logged in
And I close browser

@regression @login
Scenario: Invalid login
Given I navigated to OrangeHrm
And I see OrangeHrm logo
When I enter invalid username and password
And I click login button
Then I see error message is displayed
And I close browser
```

Cucumber Tags

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

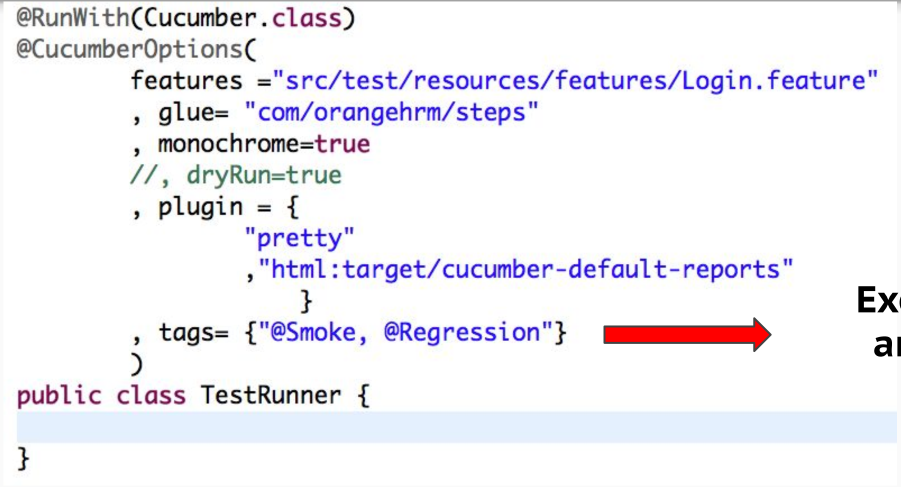
Sometimes we might need to run more than one tags at a time, in such cases we can use AND & OR to combine the cucumber tags to run the feature files.

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 **{"Sanity, smoke, regression"}**

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

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.

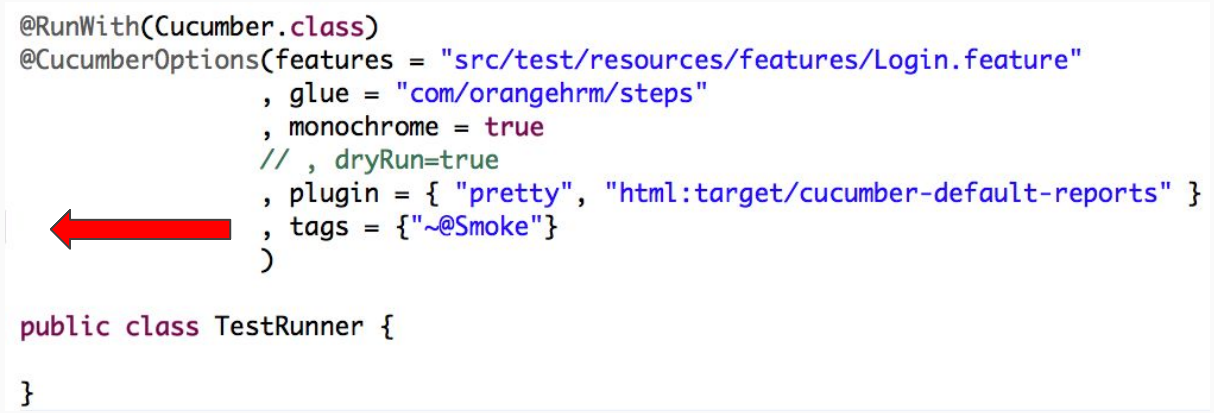
Cucumber Tags



```
@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, @Regression"}
        )
public class TestRunner {

}
```

Executing Smoke and Regression scenarios



```
@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 {

}
```

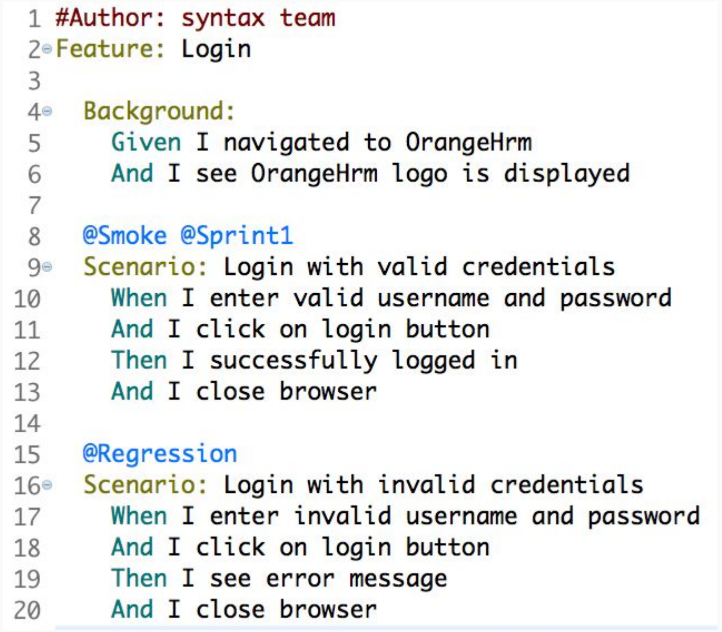
Excluding Smoke scenarios

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
Feature: Login

  Background:
    Given I navigated to OrangeHrm
    And I see OrangeHrm logo is displayed

  @Smoke @Sprint1
  Scenario: Login with valid credentials
    When I enter valid username and password
    And I click on login button
    Then I successfully logged in
    And I close browser

  @Regression
  Scenario: Login with invalid credentials
    When I enter invalid username and password
    And I click on login button
    Then I see error message
    And I close browser
```

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.

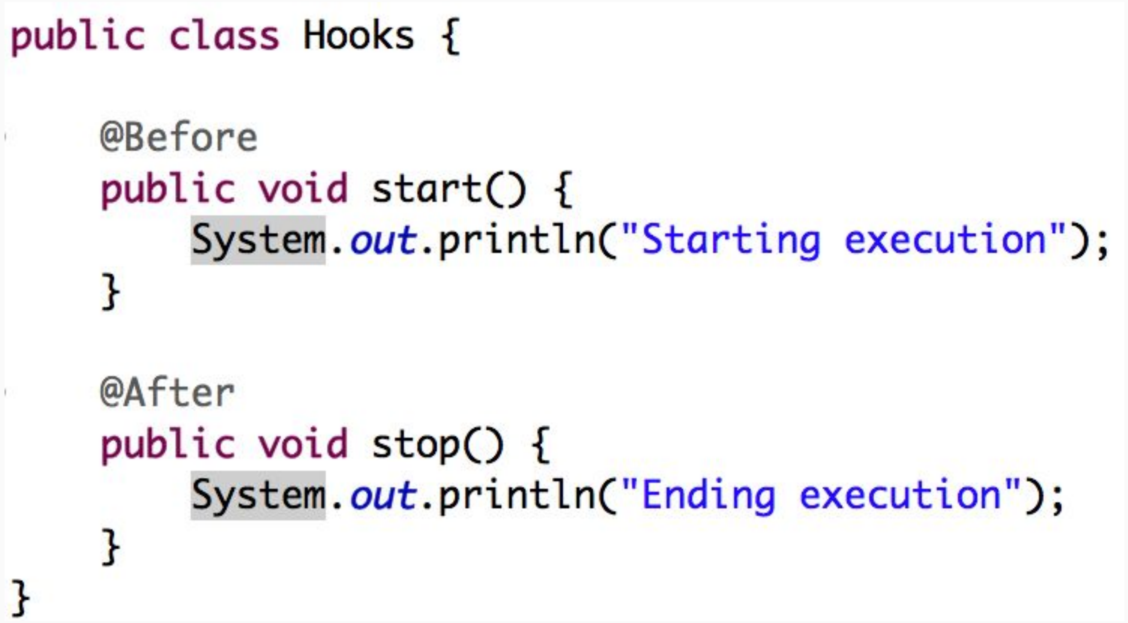
@Before hooks will be run before the first step of each scenario. They will run in the same order of which they are registered.

@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



```
public class Hooks {

	@Before
	public void start() {
		System.out.println("Starting execution");
	}

	@After
	public void stop() {
		System.out.println("Ending execution");
	}
}
```

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