One feature file its ease to run, you just press run like usual .java class, that's it. The point of the lecture is that you can not run selected scenario from the IDE, and you can not run for example more than 1 feature fail. Right now I checked in Intellege that it is updated and actually you can do it from the editor. In any way I am gonna finish the lecture.

So there is 3 runners:

1)-cucumber cli (command line interface)

2)-JUnit: Runner class. We can use annotation like before class, after class, so on. We cannot execute scenarios in parallel

3)-TestNG: Runner class. Support annotated methods. We can use annotation like before class, after class, so on. We can execute scenarios in parallel.

Junit and TestNG - are unit testing frameworks. The only disadvantage of using these annotated methods is that automation will be less portable (не переносимые с одного framework на другой). If for example you use TestNG annotations and then you want to migrate to JUnit - it's not possible. And some classes have very similar names, so it's very easy to make a mistake when you import them. Make sure that you import the correct one according to the framework you use.

**TestNG and cucumber integration (Class runner)**

<https://cucumber.io/docs/cucumber/api/?lang=java#options> ← The source

To be able to run the multiple scenarios using testNG you need to have class runner that extends

AbstractTestNGCucumberTests class

**Cucumber option**

There should be the annotation before class @CucumberOptions it comes from framework cucumber and it has predefined attributes and values for them, it can mentioned one by one through the como(запятую):

features = "src/test/java/resources" <- - - tells the jvm to look for features files.

plugin = "message:target/cucumber-report.ndjson" <- - enables the plugin to make a report and place it in target folder.

plugin = {"pretty", "html:target/cucumber"}<- - The same but report with extension of HTML which can be opened in browser. pretty - (pretty format) reflects the scenario and steps of definition in the console in a pretty colored, readable way. If you remove pretty nothing will show up in the console, but you still be able to see it in browser, since it HTML. After the ‘:’ indicates the path in project structure where the document will be generated.

plugin = {"pretty", "summary"}, snippets = CAMELCASE <- - if you want to tell Cucumber to print code snippets for missing step definitions use the summary plugin. The default option for snippets is UNDERSCORE. This settings can be used to specify the way code snippets will be created by Cucumber. когда к примеру какой то шаг в сценарии не имеет реализации в java, можно используя эту настройку в консоли получить пример как будет выглядеть метод, при этом название метода будет не через нижнее подчёркивание а кэмэл кейсом. Если не указывать то по умолчанию пример реализации шага будет приведён в консоли в качестве метода название которого будет с нижним подчеркиванием. По мне дак ерунда какаято.

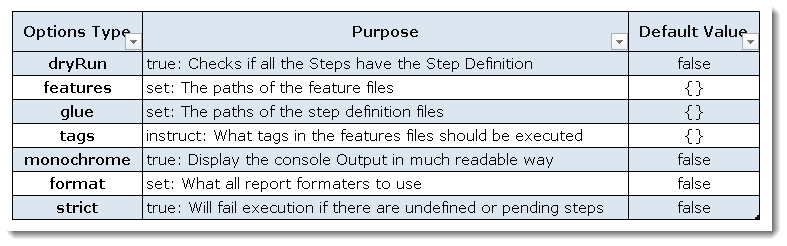
dryRun=true <- - if you want to check whether all scenarios has step definitions you can use this command, it will check if there is step definitions without running them, might be helpful if there are many feature fails. In that case cucumber will display all scenarios in console and if the steps are defined they will be in a green color. Otherwise you will get the usual suggestion / example of how to define the missing step.The default option for dryRun is false

tags =”nazvanie\_annotacii” < - - выполниться только указанное scenario

glue = “path from package java to the package of step definition” <- - if cucumber does not see step definition because of to many packages in the project, this is how to show him where it is located

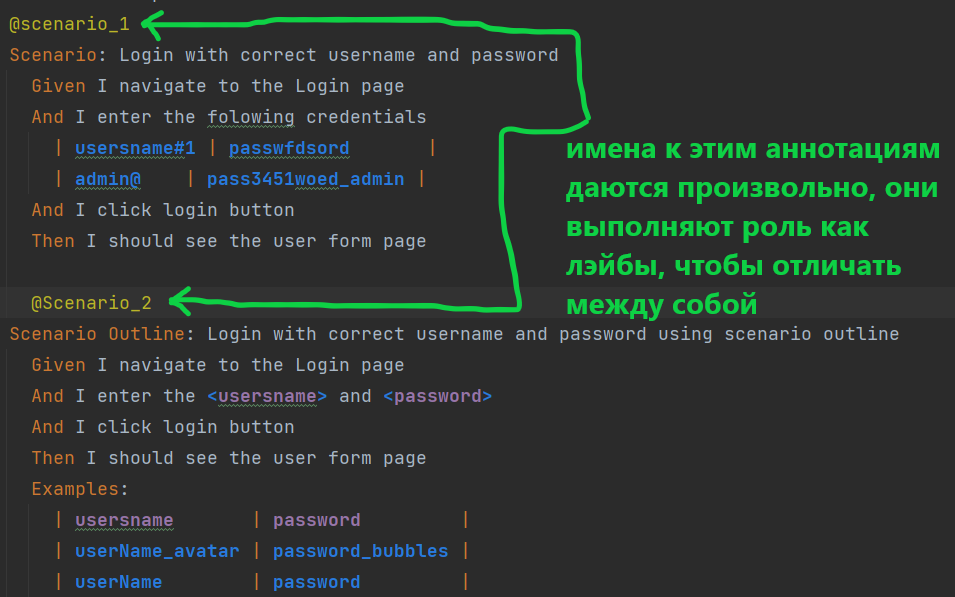
If an attribute has more than 1 value then it is allowed to put values one by one after a comma.

Inside the class runner we can use annotations of testNG that are common for all the scenarios, because since the class runner is common for all the scenarios, probably there cannot be specific settings for particular scenarios. And actually we have Hooks of cucumber, so if you decide to use annotations like @beforeclass or @afterclass from TestNG there might be conflicts because cucumber has the same annotations (hooks), better use them.



Tagging the scenarios

We can give tags to the scenarios or to features. For example we can give any tag name to the scenario using @someName.



If there no need to run all the scenarios we can choose and use attribute tags in the @CucumberOptions

**CUCUMBER CLI (MAVEN)**

Чтобы запустить maven надо

1)-быть в директории своего проекта. Например:

**C:\Users\user\IdeaProjects\Cucumber2>**

2)-Иметь class Runner, в моём случае я использую TestNG.

3)-**mvn clean test** команда для запуска mavena в терминале. Она прогонит тест с теми настройками которые есть в раннере.

4) - C:\Users\user\IdeaProjects\Cucumber2>**mvn clean test -Dbrowser=chrome** <- - запуск тестов на хроме, точно также можно поменять на firefox или чёнить другое

5)- **mvn clean test -Denv=PROD**