**********CUCUMBER**********

TechPro Education

cucumber

- **Cucumber bizim son framework'umuz olacak.**

- **Cucumber BDD (behaviour driven development) (Davranış tabanli geliştirme) yaklaşımı için kullanılmakta olan açık kaynak kodlu bir kütüphanedir.**

- **Cucumber bir iş ararken önemli bir rol alacaktir.**

```
Feature: US1001 Ck Hotels Login
  @wip
  Scenario: TC01 kullanici gecerli bilgilerle giris yapar
    Given kullanici Ck Hotels ana sayfasinda
    Then Log in yazisina tiklar
    And gecerli username girer
    And gecerli password girer
    And Login butonuna basar
    Then sayfaya giris yaptigini kontrol eder
    And kulllanici sayfayi kapatir
```

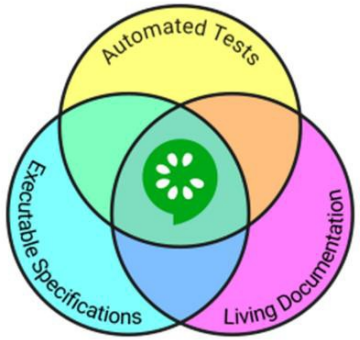
- **TestNg hakkında her şeyi bilmemek sorun değil ama Cucumber hakkında her şeyi bilmeniz GEREKİR.**

- **Agile methodolojisinde, insanlar uygulamanın işlevselliğini geliştirmek için birlikte çalışmak zorundadır. İnsanlar development sırasında birlikte hareket etmeli.**

cucumber

- BDD(behaviour driven development) (Davranış güdümlü geliştirme)- ilk olarak behavior(davranis) veya functionalitileri yaziyorsunuz (Epic=Feature, Story, AC, etc), daha sonra development and testing basliyor.

- BDD'de behaviour'lar başarısız olduğunda kod başarısız olur..



cucumber ♡ BDD

- Anlasilabilir Gherkin Language nedeniyle BDD development icin Cucumber harika bir uygulamadir.

- **Gherkin:** Projede her bir davranış için .feature uzantılı bir Gherkin dosyası oluşturulur. Bu feature dosyasına ilgili özelliğin farklı durumlardaki davranışları tanımlanır.

  Given anahtar kelimesi ile ön koşul yani başlangıç durumu tanımlanır,

  When, And anahtar kelimeleri ile olayı

  Then anahtar kelimesi ile de sonuç tanımlanır.

  Given-When-And-Then adımları sadece okuyanlar icin farkeder Java icin hepsi birdir.

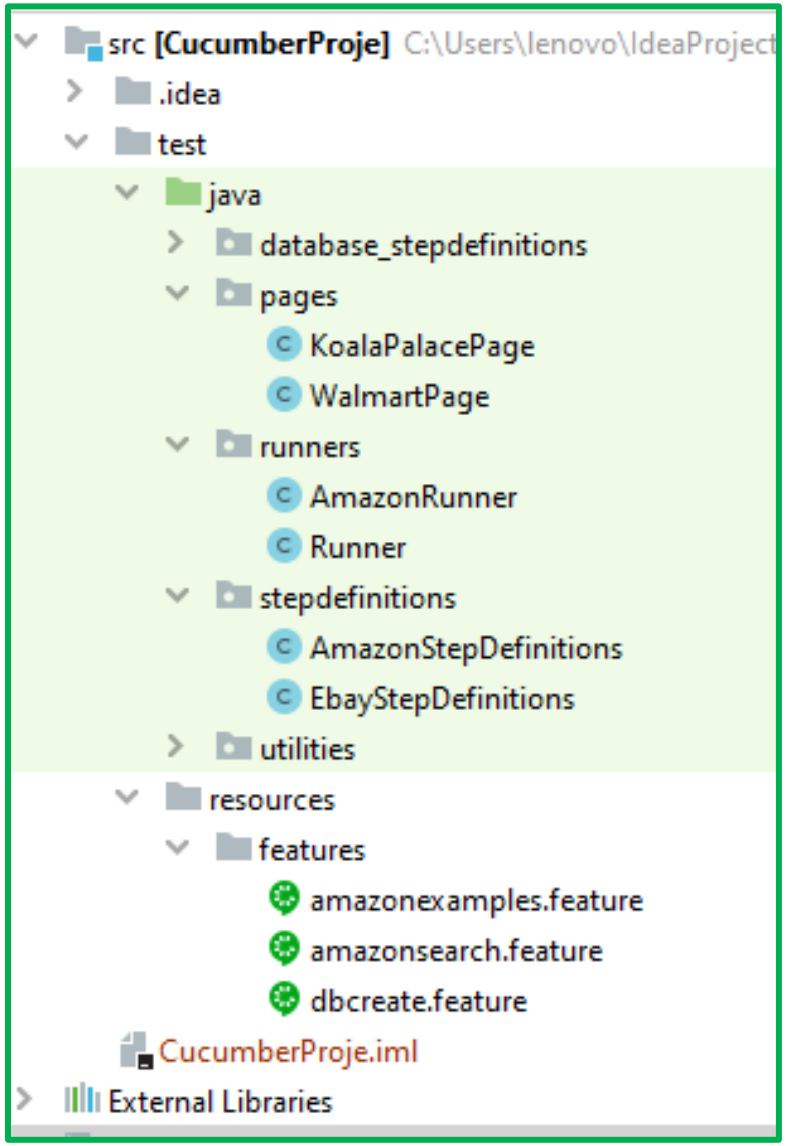
cucumber

- Cucumber (TDD)(Test Driven Development) test odaklı geliştirmeye izin verir, çünkü Cucumber ile Junit veya TestNG kullanabiliriz.

- Cucumber iş için önemlidir, çünkü anlaşılabilir ve harika raporlara sahiptir.

- Cucumber, teknik olmayan (none-technical) kişileri ve teknik kişileri birbirine bağlar.

- Developer veya team lead gibi teknik elemanlar da bazen testerlarin yaptığını anlayamayabilir. Gherkin onlarin da testlerimizi anlamalarini kolaylastirir



cucumber

Proje Olusturma

1. **Create Project:** File -> New -> Project-> Select maven -> click next
2. Name: batch129Cucumber->finish
3. Add Dependencies =>Selenium-java, webdrivermanager, cucumber java, cucumber-junit
4. Click Maven => click “Enable auto-reload after any changes” (BU MESAJ GORUNMEYEDEBILIR)
5. Javanin sürümüyle alakali sorunlari halletmek icin compiler dependency yuklenebilir

```
<properties>

        <maven.compiler.source>1.8</maven.compiler.source>

        <maven.compiler.target>1.8</maven.compiler.target>

</properties>
```

cucumber

Proje Olusturma

6. Java'ya sag click yapip asagidaki paketleri olusturalim
    a. utilities
    b. pages
    c. runners (test case'leri calistirmak ve control etmek icin kullanacagiz)
    d. stepdefinitions (kodlarimizi burada olusturacagiz)
7. Projeye sag click yapip configuration.properties dosyasi olusturalim
8. Utilities paketi altinda Driver ve ConfigReader Class'larini olusturalim
9. test paketi altinda yeni bir klasor olusturalim : resources
10. resources klasoru altinda yeni bir klasor olusturalim : features (Java kodu icermeyen dosyalari buraya koyacagiz)
11. features'a sag clik yapip dosya olusturalim ilkfeaturefile.feature
12. cucumber plugin'i intelliJ'e ekleyelim (settings/Plugins)
    1. (Mac)IntelliJ Idea->Preference->Plugins->Marketplace->Type Cucumber for Java->Install->Restart
    2. (Windows)File->Setting->->Plugins->Marketplace->Type Cucumber for Java->Install->Restart

BATCH : 129 NT

LESSON : **SELENIUM**

DATE : 08.05.2023

SUBJECT : Background/Parameter

ZOOM GİRİŞLERİNİZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ

Cucumber da Automation Stepleri

- **Feature File**
  - Gherken language ile test senaryolarini olustur
- **Runner package:(DO THIS ONLY ONCE)**
  - Runner ile test caseleri calistir
    - **@RunWith(Cucumber.class)**
    - @CucumberOptions(
    - features= “features folder path”
    - glue=“stepdefinitions folder path”
    - tags= “@tagname”)
- **Stepdefinitions package:**
  - JAVA kodlarının oluşturulduğu yer
- **Runner class run et**
- **Step definitions lari step definitions paketine at**
- **Java kodlarini tamamla**

- Feature File : **googlesearch.feature**

- **Scenario 1**
- When user search for iPhone on google
- Then verify the result has iPhone related results

- **Scenario 2**
- When user search for Tea Pot on google
- Then verify the result has Tea Pot related results

- **Scenario 3**
- When user search for flower on google
- Then verify the result has flower related results

- **Scenario 4**
- When user search for bmw on google
- Then verify the result has bmw related results

Feature file Gherkin Keywords

Feature file da test caseler GHERKIN language ile olusturulur

The primary keywords are:

- `Feature`
- `Rule` (as of Gherkin 6)
- `Example` (or `Scenario`)
- `Given`, `When`, `Then`, `And`, `But` for steps (or `*`)
- `Background`
- `Scenario Outline` (or `Scenario Template`)
- `Examples` (or `Scenarios`)

There are a few secondary keywords as well:

- `"""` (Doc Strings)
- `|` (Data Tables)
- `@` (Tags)
- `#` (Comments)

Each FF must begin with a **Feature:** keyword
We cannot have more than 1 Feature keyword in FF

```
# Comment
@tag
Feature: Eating too many cucumbers may not be good for you

  Eating too much of anything may not be good for you.

  Scenario: Eating a few is no problem
    Given Alice is hungry
    When she eats 3 cucumbers
    Then she will be full
```

Given,When,Then,.. annotations are used to create Test Steps

Scenario: is used to create Test Case(Test Scenario).
We CAN have more than 1 **Scenario:** in each Feature file

Runner

- Runner feature file I run etmek icin kullanilir

```
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "./src/test/resources/features",
        glue = "stepdefinitions",
        tags = "@google_search",
        dryRun = false
)
public class Runner {
}
```

@RunWith *is used to run the class. Without this, Runner class will not be runnable*

@CucumberOptions *is used to add feature path, step definition path,tags, dryRun, report plug ins*

features *: path of the features folder. this can point path of specific feature file*

glue *: path of step definitions folder. this can pint path of specific step definition class*

tags *: this marks which feature file to run. we add this tag in the feature files*

dryRun *: -dryRun is used to check if there is any MISSING JAVA CODE(STEP DEFINITIONS)*
** -dryRun takes 2 values, true or false*
** -dryRun = false :Runs tests on the browser while checking if there is any JAVA code missing*
** -dryRun = true :Runs tests without browser while checking if there is any JAVA code missing*

Unimplemented(tanimlanmamis) Step Definitions

1. Feature File

```
@google_search
Feature: Google Search Functionality

  Scenario: TC01_iPhone search

#    Describe the behaviour
    Given user in on the google page
    And user search for iPhone
    Then verify the result has iPhone
    Then close the application
```

2.Unimplemented Step Definitions Generated by Cucumber in the console

```
io.cucumber.junit.UndefinedStepException: The step 'user in on the google
You can implement these steps using the snippet(s) below:

@Given("user in on the google page")
public void user_in_on_the_google_page() {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}
@Given("user search for iPhone")
public void user_search_for_i_phone() {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}
@Then("verify the result has iPhone")
public void verify_the_result_has_i_phone() {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}
@Then("close the application")
public void close_the_application() {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}
```

3. Write JAVA Code in the step definitions

```
public class Day17_GoogleSearchStepDefinitions {
    /*This step definition class will have JAVA CODE
    * */
    GooglePage googlePage = new GooglePage();
    @Given("user in on the google page")
    public void user_in_on_the_google_page() {
        Driver.getDriver().get(ConfigReader.getProperty("google_url"));
    }
    @Given("user search for iPhone")
    public void user_search_for_i_phone() {
        googlePage.googleSearchBox.sendKeys( ...keysToSend: "iPhone"+ Keys.EI
    }
    @Then("verify the result has iPhone")
    public void verify_the_result_has_i_phone() {
        String title=Driver.getDriver().getTitle();
        Assert.assertTrue(title.contains("iPhone"));
    }
    @Then("close the application")
    public void close_the_application() {
        Driver.closeDriver();
    }
}
```

Unimplemented(tanimlanmamis) Step Definitions

NOTE: io.cucumber.junit.UndefinedStepException:
Bu hata mesaji tanimlanmamis eksik java kodları oldugunda consolda cikar

SOLUTION: Step definitionslari consoldan al step definitions dosyasina koy

NOTE: Bu sekilde yeni step definitionslar beklendiginde, zamandan tasarruf saglamak icin, runnerda **dryRun=true**, kullanilir

```
..
Then close the browser
```

```
io.cucumber.junit.UndefinedStepException: The step 'close the browser' is
You can implement this step using the snippet(s) below:

@Then("close the browser")
public void close_the_browser() {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}
```

Tags

Tags can be placed above the following Gherkin elements:

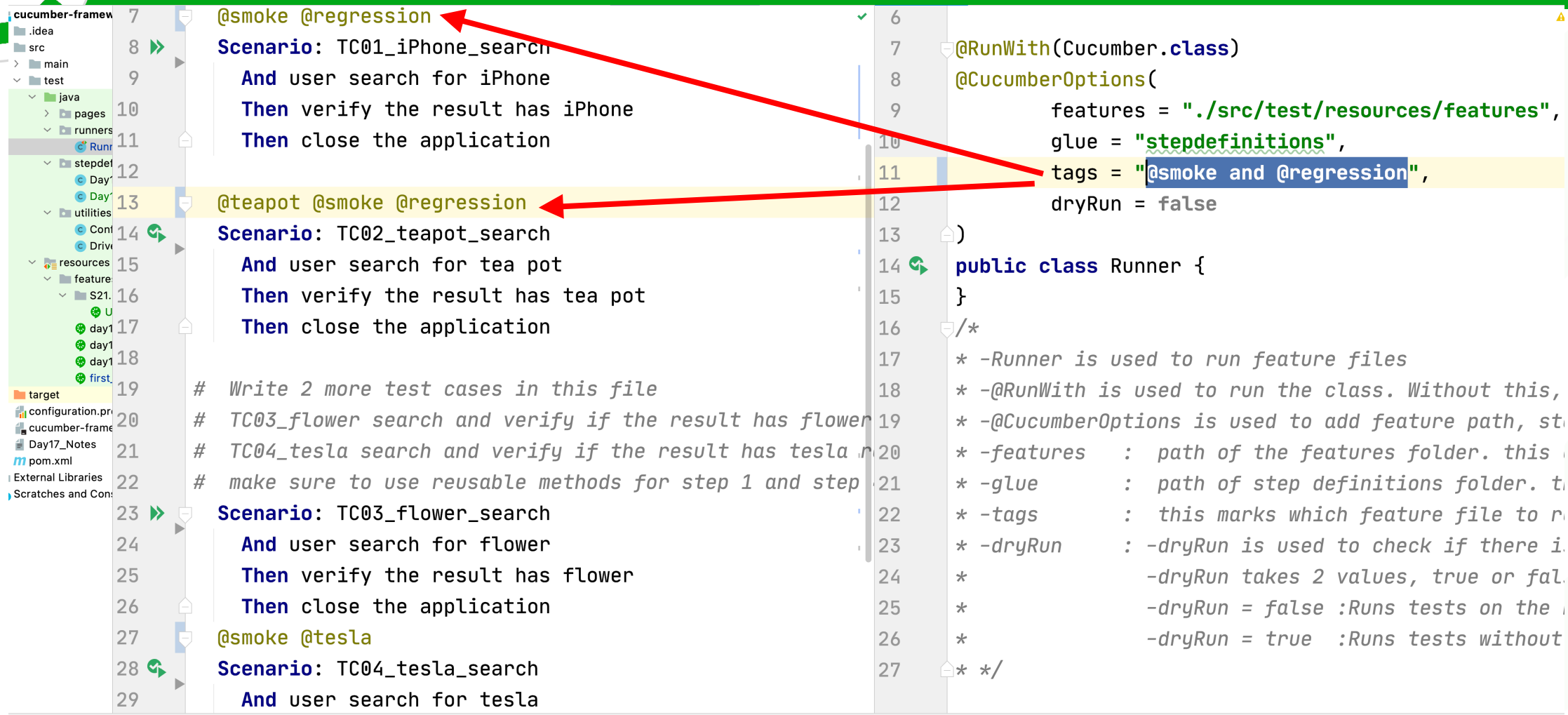
- Feature
- Scenario
- Scenario Outline
- Examples

```
@google_search
Feature: Google_Search_Functionality

  @teapot
  Scenario: TC02_teapot_search
    And user search for tea pot
#   YES
  @admin_login_test
  Scenario Outline: login_with_admin_credentials

    @admin_login_test
    Examples: admin_data
      |admin id     |admin p
```

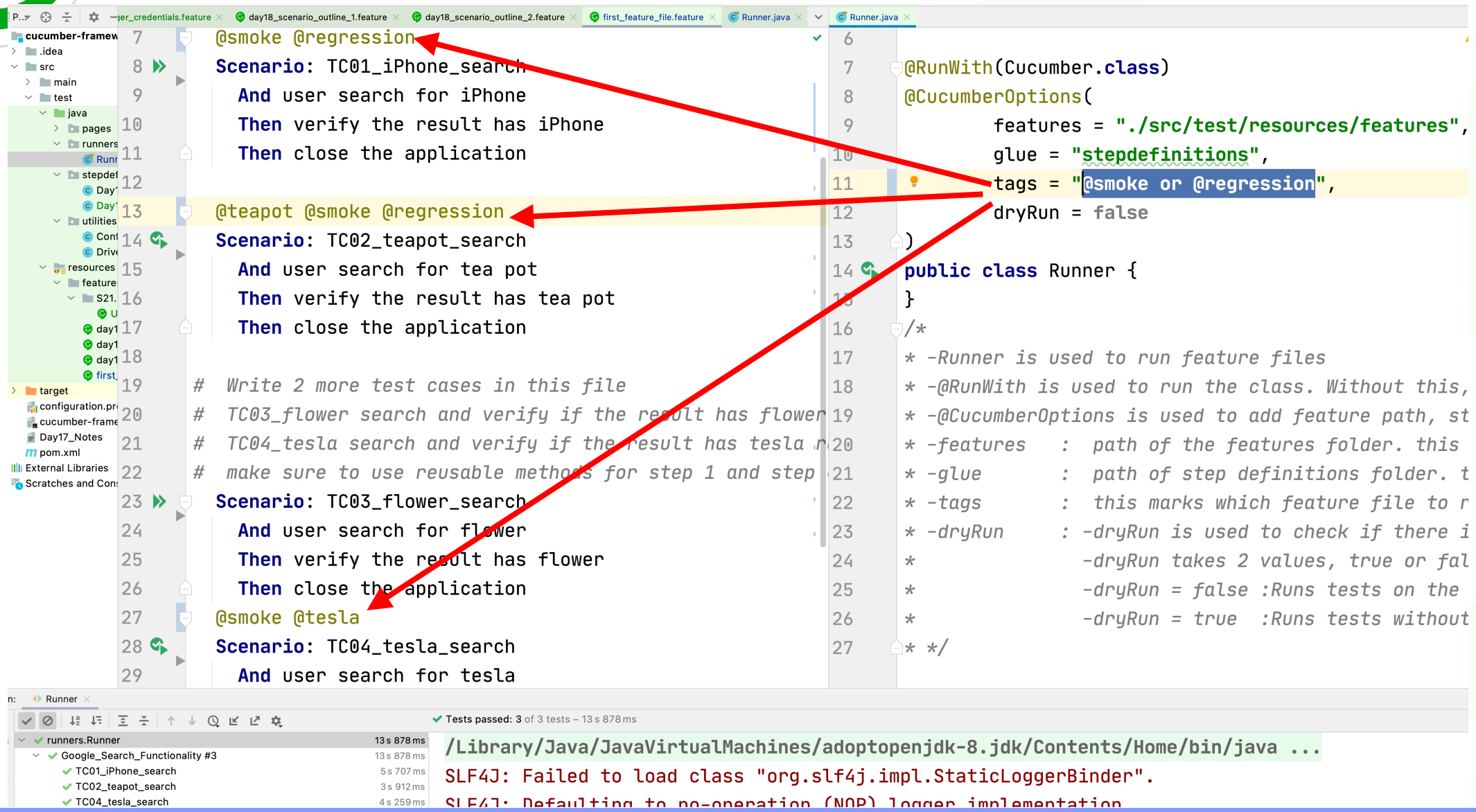
Tags

Expression	Description
`@fast`	Scenarios tagged with `@fast`
`@wip and not @slow`	Scenarios tagged with `@wip` that aren't also tagged with `@slow`
`@smoke and @fast`	Scenarios tagged with both `@smoke` and `@fast`
`@gui or @database`	Scenarios tagged with either `@gui` or `@database`

And tag


cucumber-framew
.idea
src
main
test
java
pages
runners
stepdef
utilities
resources
target
Day17_Notes
pom.xml
External Libraries
@smoke @regression
Scenario: TC01_iPhone_search
And user search for iPhone
Then verify the result has iPhone
Then close the application
@teapot @smoke @regression
Scenario: TC02_teapot_search
And user search for tea pot
Then verify the result has tea pot
Then close the application
Write 2 more test cases in this file
TC03_flower search and verify if the result has flower
TC04_tesla search and verify if the result has tesla
make sure to use reusable methods for step 1 and step
Scenario: TC03_flower_search
And user search for flower
Then verify the result has flower
Then close the application
@smoke @tesla
Scenario: TC04_tesla_search
And user search for tesla
@RunWith(Cucumber.class)
@CucumberOptions(
features = "./src/test/resources/features",
glue = "stepdefinitions",
tags = "@smoke and @regression",
dryRun = false
)
public class Runner {
}
/*
* -Runner is used to run feature files
* -@RunWith is used to run the class. Without this,
* -@CucumberOptions is used to add feature path, st
* -features : path of the features folder. this
* -glue : path of step definitions folder. t
* -tags : this marks which feature file to r
* -dryRun : -dryRun is used to check if there i
* -dryRun takes 2 values, true or fal
* -dryRun = false :Runs tests on the
* -dryRun = true :Runs tests without
* */
Runner
Tests passed: 2 of 2 tests – 9 s 718 ms
runners.Runner 9 s 718 ms
Google_Search_Functionality #3 9 s 718 ms
TC01_iPhone_search 4 s 674 ms
TC02_teapot_search 5 s 44 ms
/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

Or tag



```
@smoke @regression
Scenario: TC01_iPhone_search
  And user search for iPhone
  Then verify the result has iPhone
  Then close the application

@teapot @smoke @regression
Scenario: TC02_teapot_search
  And user search for tea pot
  Then verify the result has tea pot
  Then close the application

#  Write 2 more test cases in this file
#  TC03_flower search and verify if the result has flower
#  TC04_tesla search and verify if the result has tesla
#  make sure to use reusable methods for step 1 and step
Scenario: TC03_flower_search
  And user search for flower
  Then verify the result has flower
  Then close the application
@smoke @tesla
Scenario: TC04_tesla_search
  And user search for tesla
```

```
@RunWith(Cucumber.class)
@CucumberOptions(
        features = "./src/test/resources/features",
        glue = "stepdefinitions",
        tags = "@smoke or @regression",
        dryRun = false
)
public class Runner {
}
/*
* -Runner is used to run feature files
* -@RunWith is used to run the class. Without this,
* -@CucumberOptions is used to add feature path, st
* -features   :  path of the features folder. this
* -glue       :  path of step definitions folder. t
* -tags       :  this marks which feature file to r
* -dryRun     : -dryRun is used to check if there i
*               -dryRun takes 2 values, true or fal
*               -dryRun = false :Runs tests on the
*               -dryRun = true  :Runs tests without
* */
```

```
Tests passed: 3 of 3 tests – 13 s 878 ms
runners.Runner 13 s 878 ms
  Google_Search_Functionality #3 13 s 878 ms
    TC01_iPhone_search 5 s 707 ms
    TC02_teapot_search 3 s 912 ms
    TC04_tesla_search 4 s 259 ms
/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
```

Background

- **@Background** prerequisites(Tekrar eden on koşullarda kullanilir)
- Gereksiz tekrarlardan kurtulmak icin kullanilir
- Her bir **SCENARIO** dan once kullanilir
- Ornegin, 5 scenario kullanilmissa background 5 kez calisir.
- @BeforeMethod a benzer

```
Feature: Google_Search_Functionality

  Background: Navigating_to_the_page
    Given user is on the google page

  Scenario: TC01_iPhone_search
    And user search for iPhone
```

Parametrizing the Feature File using “”

CUCUMBER REVIEW

- What is cucumber?
  - Cucumber is a testing framework. It is a BDD(Behaviour Driven Development) framework.
  - BDD means we describe the behavior(functionality) in the feature file first, then write the code
- What is feature file?
  - It is used to write test cases in Gherkin language. It has keywords like given, when , then,…This is plane English.
- What is Background?
  - It is used for repeated PRE CONDITIONS. This runs before each Scenario keyword.
- What is Scenario:
  - It is used to create test cases in feature files. I can use more than 1 scenario keyword in a future file.
- What is Feature:
  - This is used to create a feature file. Each feature file must begin with a Feature keyword. They cannot be more than 1 Feature keyword in a feature file
- How do you run **specific** test cases in cucumber?
  - We use runner class to run the test cases. We can run specific test scenarios or feature files using tags.
- What is dryRun in cucumber?
  - To generate missing/unimplemented step definitions, we use dryRun=true. When I want to run test cases, I do dryRun=false.
- What is step definition in cucumber?
  - Technical codes goes to step definition classes. We write the java code in these classes.
- How do you parametrize the feature files?How do you get the data from the feature file? How do you make the test cases more reusable in cucumber.What is your strategy?
  - **String “”** makes the data dynamic. We can use **Scenario Outline**, **data tables** as well

PARAMETRIZING FF USING “”

Without Parameter

- Kodlarimizi parametreli ve dinamik hale getirmek için feature file da degisken olarak kullanacagimiz kelimeyi çift tırnak " " icine aliriz.

```
@rapor2
Scenario: TC07 istenen kelimenin oldugunu test etme
  Given kullanici "amazonUrl" sayfasina gider
  And "iphone" icin arama yapar
  Then sonucun "iphone" icerdigini test eder
  And sayfayi kapatir
```

- Kodlarimizi parametreli olarak yazdiktan sonra sadece " " içindeki değeri değiştirerek test datalarini feature file dan kontrol edebiliriz.

- Kodlarimizi parametreli olarak yazmak framework’u daha dinamik hale getirir(kodumuz artik hard coded degildir diyebiliriz).

PARAMETRIZING FF USING “”

PARAMETRESIZ

```
Scenario 1:
And user search for iPhone
Then verify the result has iPhone

Scenario 2:
And user search for tea pot
Then verify the result has tea pot

Scenario 3:
And user search for flower
Then verify the result has flower

Scenario 4:
And user search for tesla
Then verify the result has tesla
```

8 step definitions a ihtiyac var

PARAMETRELI

```
Scenario 1:
And user search for “iPhone”
Then verify the result has “iPhone”

Scenario 2:
And user search for “tea pot”
Then verify the result has "tea pot”

Scenario 3:
And user search for “flower”
Then verify the result has “flower”

Scenario 4:
And user search for “tesla”
Then verify the result has “tesla”
```

2 step definitions a ihtiyac var

```
@Given("user search for {string}")
public void user_search_for(String string) {
    googlePage.googleSearchBox.sendKeys(
            ...keysToSend: string+ Keys.ENTER);
}

@Then("verify the result has {string}")
public void verify_the_result_has(String string) {
    String title =Driver.getDriver().getTitle();
    Assert.assertTrue(title.contains(string));
}
```

PARAMETERIZING- WORK FLOW

STEP DEFINITIONS

```
@When("user search for {string}")
public void user_search_for(String string) {
    googlePage.searchBox.sendKeys( ...keysToSend: string+Keys.ENTER);
}

@Then("verify the result has {string}")
public void verify_the_result_has(String string) {
    String title=Driver.getDriver().getTitle().toLowerCase();
    Assert.assertTrue(title.contains(string));
}
```

FEATURE FILE

```
Given user is on the google page

Scenario: TC_01_iphone search
  When user search for "iPhone"
  Then verify the result has "iphone"
```

BATCH : 129 NT

LESSON : **SELENIUM**

DATE : 09.05.2023

SUBJECT : Scenario Outline/
HTML Report

ZOOM GİRİŞLERİNİZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ

Scenario Outline

Scenario Outline

- `Scenario Outline` ayni `Scenario` i farklı degerlerle tekrar tekrar run etmek icin kullanilir
- **“<column_name>”** ifadesi verileri parametre seklinde göndermek icin kullanilir
- **Examples,** Scenario Outline dan sonra verileri girmek icin kullanilmalidir.
- Scenario Outline, **Data Driven Test** icin kullanilir
- Documentation: https://cucumber.io/docs/gherkin/reference/#scenario-outline

Scenario Outline

```
Scenario: eat 5 out of 12
  Given there are 12 cucumbers
  When I eat 5 cucumbers
  Then I should have 7 cucumbers

Scenario: eat 5 out of 20
  Given there are 20 cucumbers
  When I eat 5 cucumbers
  Then I should have 15 cucumbers
```

```
Scenario Outline: eating
  Given there are <start> cucumbers
  When I eat <eat> cucumbers
  Then I should have <left> cucumbers

  Examples:
    | start | eat | left |
    |    12 |   5 |    7 |
    |    20 |   5 |   15 |
```

SCENARIO OUTLINE NE ZAMAN KULLANILIR?

- Reusable test stepleri oluşturmak icin
- Birden fazla veri göndermek icin
- Feature file kisa hale getirmek icin. Birden fazla Scenario yerine tek bir Scenario Outline kullanilabilir

- Scenario Outline AYNI SCENARIO yu FARKLI VERILERLE test etmek icin kullanilir
  - Ornek: birden fazla kullanıcı bilgileriyle sayfaya girmek
    - Kullanici Bilgileri={name=Jim,id=1}, {name=John,id=10},{name=Sam,id=80},…
  - Birden fazla kullanıcı hesabi oluşturmak
  - Birden fazla urun aratmak
    - Veriler = Tesla, Honda, Nissan,…

Scenario Outline

- When user search for the following capitals: London, Paris, Vienna, Berlin, Madrid
- Then verify the the result includes the the searched values
- FF: day18_scenario_outline_2.feature

Scenario Outline

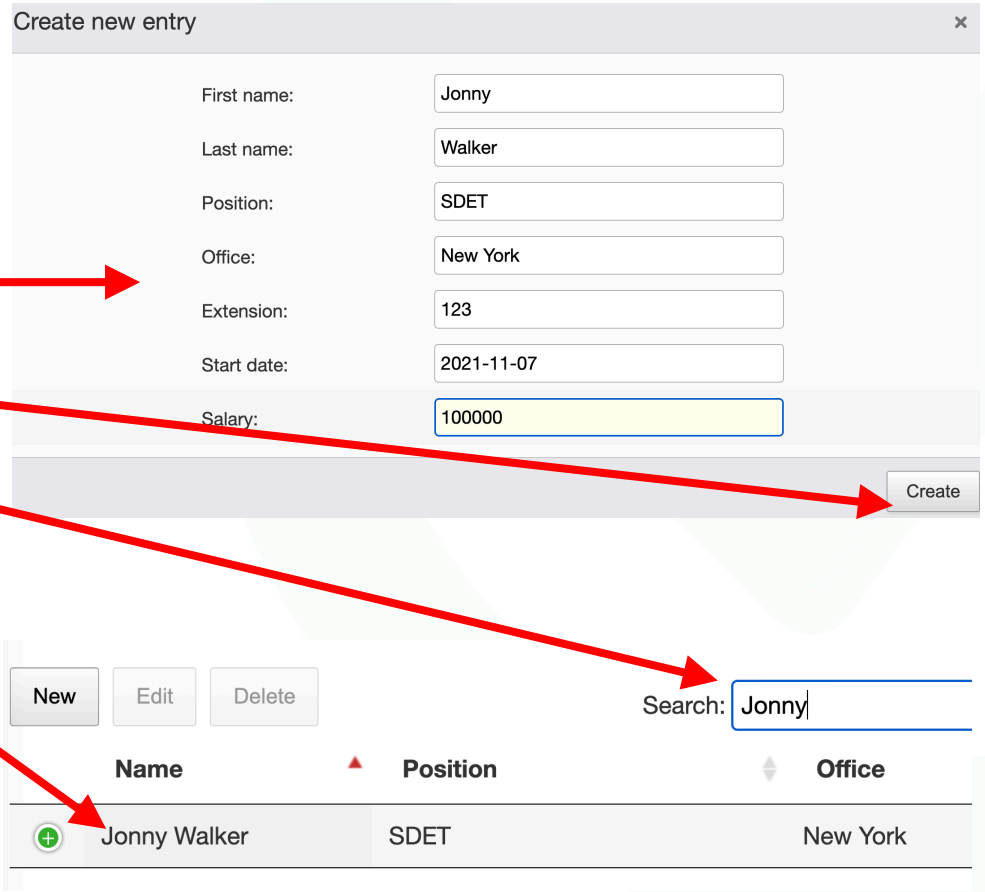
```
@scenario_outline_google_search
Feature: Google_Search_Functionality
  Scenario Outline: google_search
#shift + < puts data in <>
#control + alt + L = aligns the pipelines

    Given user is on the google page
    And user search for "<item>"
    Then verify the result has "<item>"
    Then close the application

    Examples: Test Data
    |item    |
    |iPhone  |
    |tea pot|
    |flower  |
    |tesla   |
    |bmw     |
    |buick   |
    |nissan  |
```

7 kez bu scenario calisacakdir
Burda aslında 7 Scenario vardir
Ve tekbir Scenario Outline vardir

```
66      @Then("verify the result has tesla")
67      public void verify_the_result_has_tesla() {
68          String title = Driver.getDriver().getTitle();
69          Assert.assertTrue(title.contains("tesla"));
70      }
71      @Given("user search for {string}")
72      public void user_search_for(String string) {
73          googlePage.googleSearchBox.sendKeys(
74                  ...keysToSend: string+ Keys.ENTER);
75      }
76      @Then("verify the result has {string}")
77      public void verify_the_result_has(String string)
78          String title =Driver.getDriver().getTitle();
79          Assert.assertTrue(title.contains(string));
80      }
81
82  }
83
```

Scenario Outline

- **STORY NAME:** Yeni kullanıcı oluşturma
- **AC:** Given user is on the application
- And enter all fields
- Then user should be able to find the data with first name
- **LINK:** https://editor.datatables.net/
  - When user go to
  - https://editor.datatables.net/
  - Click on the new button
  - When user enters all fields
  - When clicks create button
  - And search for the first name
  - Then verify the name fields contains first name
  - NOTE: Try getting the test data in different forms:
  - 1. Scenario outline
  - 2. Data tables
  - 3. Excel



HOMEWORK

- **USER STORY :** US167854_manager_login_test
- **NAME:**kullanici tum manager login bilgileriyle giriş yapabilmeli
- **AC:**
- Given kullanici ana sayfada
- When kullanici login sayfasina gider
- And kullanıcı adini girer->>>>>parametre
- And sifreyi girer->>>>>parametre
- And login buttoning basar
- Then login islemi gerçekleşir
- **ARTIFACT:**
- url : https://www.bluerentalcars.com/
- Credentials:

{"sam.walker@bluerentalcars.com","c!fas_art"},

{"kate.brown@bluerentalcars.com","tad1$Fas"},

{"raj.khan@bluerentalcars.com","v7Hg_va^"},

{“pam.raymond@bluerentalcars.com”,"Nga^g6!"}

NOT: scenario outline kullanalim

OZET - TEST STEPS / TEST FLOWS IN CUCUMBER

- 1. Feature file olustur
- 2. Scenarios (Test Cases, Test Scripts) olustur
  - Scenario:
  - Scenario Outline:
- 3. Missing step definitions lari step definitions a aktar
- 4. JAVA kodunu page object model kullanarak yap
- NOTE: Reusable feature stepsleri kullandigimizdan emin olalim. Yeni java kodlarini sadece ihtiyac oldukça oluştururuz

Cucumber HTML reports

Cucumber HMTL reports

- Cucumber ile ilk raporumuzu cucumber plugins leri @CucumberOptions a ekleyerek alacağız
- html, json, yada junit raporlari oluşturabiliriz
- En onemlisi ve kullanislisi HTML raporudur

```
Add below to the runner class:
```

Report Tipi

Target Folder

Report ismi

Repor uzantisi

```
plugin = {
        "html:target/default-cucumber-reports.html",
        "json:target/json-reports/cucumber.json",
        "junit:target/xml-report/cucumber.xml"
}

NOTE: Ilerleyen derslerimizde maven-cucumber-reporting plug ins raporlar
alacaz
Ayrica Spark HTML and PDF reporlari da ekleyecez(EXTENT REPORTS A BENZER)
```

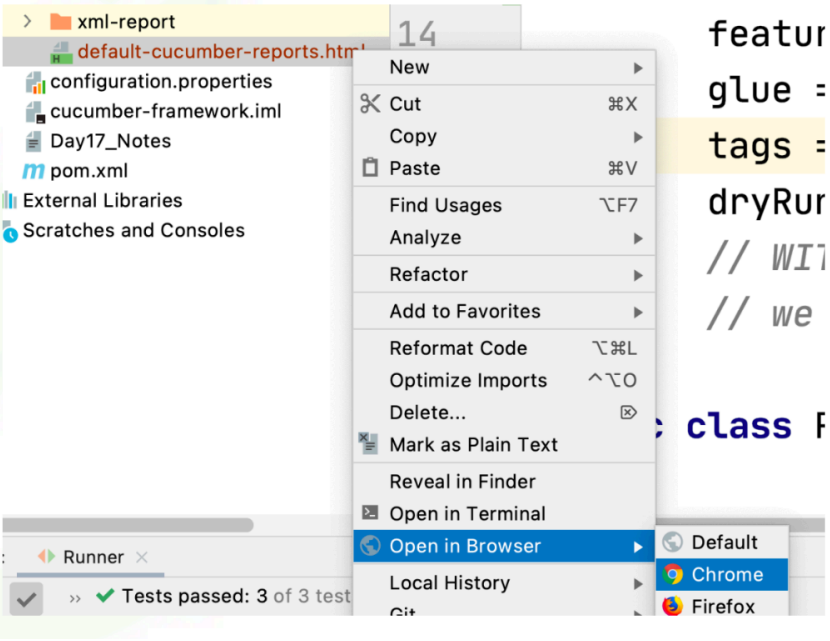
Cucumber HMTL reports

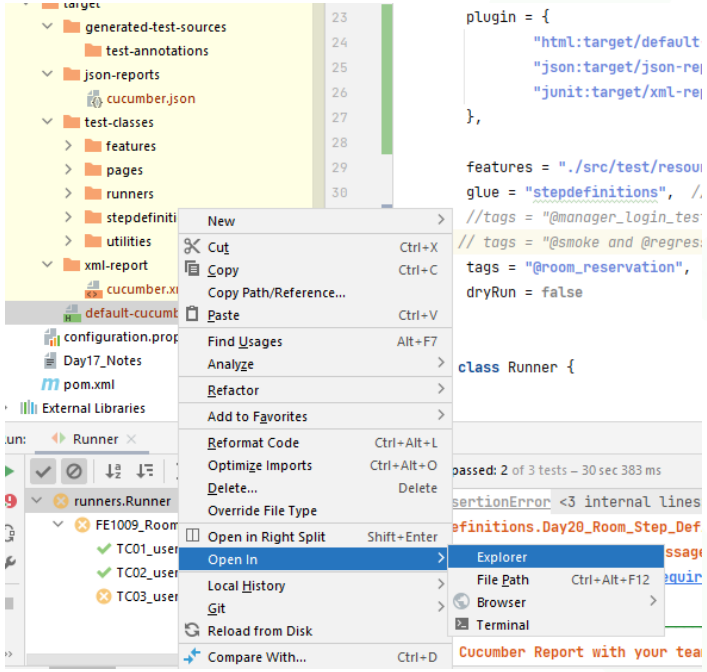
```
plugin = {
        "pretty",
        "html:target/default-cucumber-reports.html",
        "json:target/json-reports/cucumber.json",
        "junit:target/xml-report/cucumber.xml"
},
monochrome=true,
```

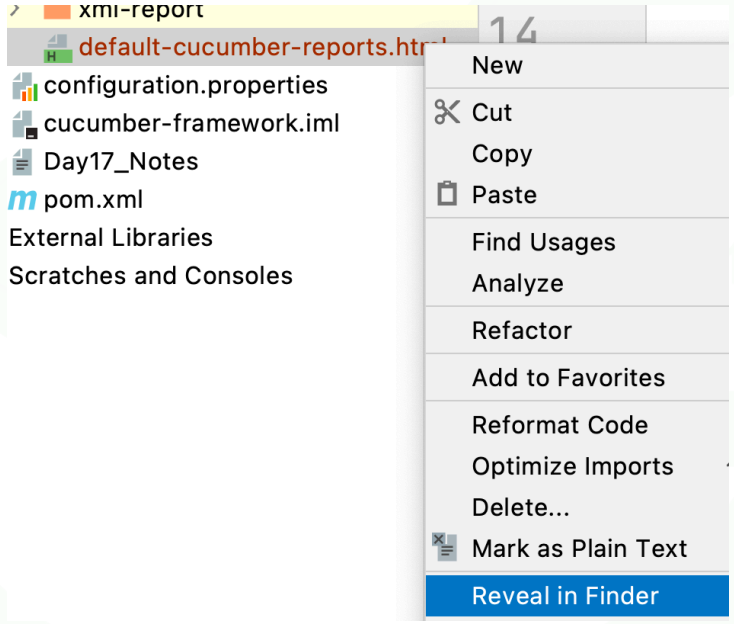
“pretty" : console bilgilerin okunaklı olması

monochrome=true : console bilgilerin okunaklı olması

How to Open Reports





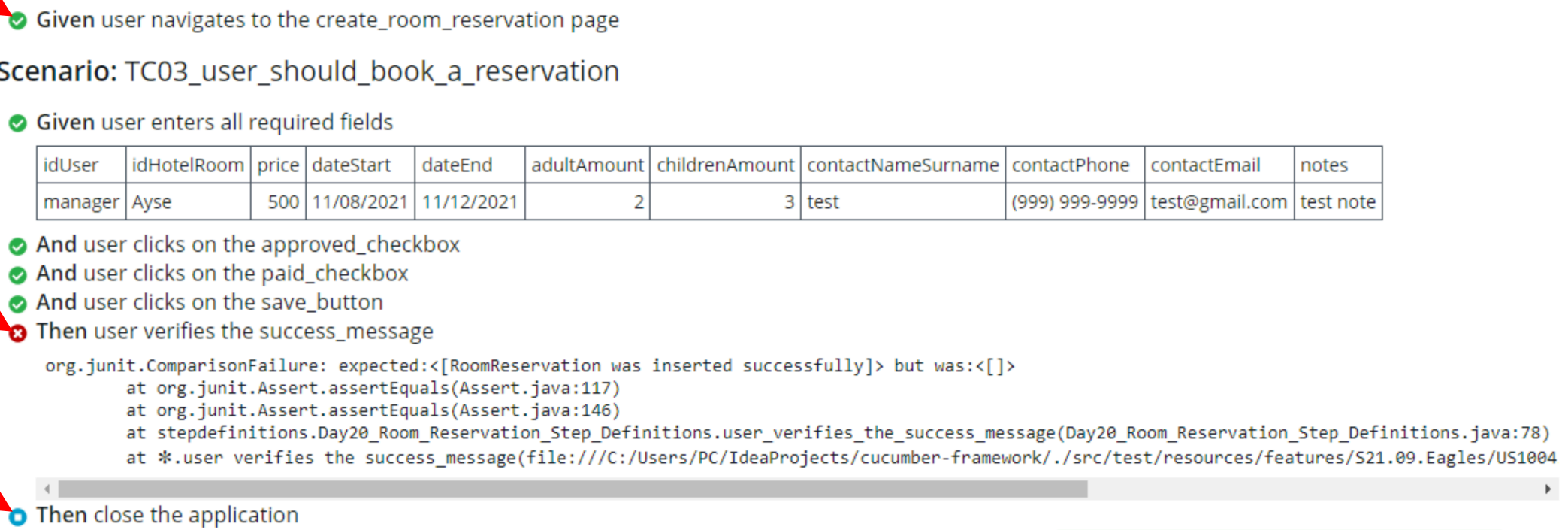


Cucumber HMTL reports

Pass

Fails

Skip



Cucumber raporlarını cloud a ekleme

- 1. resources dosyasinin altında cucumber properties dosyası olustur : **cucumber.properties**
- 2. Su kodu ekle: cucumber.publish.enabled=true
- 3. Testi calistir. Bu sekilde cucumber html raporu cloud yuklenecektir. Fakat GitHub ile girls yapilmassa 24 saat icinde silinecektir. Bu yüzden GitHub ile giriş yapalım.
- 4. Login with GitHub a tikla > Authorize SmartBear
- 5. Create Collection
- 6. Tokeni cucumber.properties e ekle
- 7. Testi tekrar calistir. Collections klasorunde raporlar olusucakdir.

DataTables

DIFFERENT OPTIONS FOR USING DATA

Regular steps

```
And user enters IDUser    -> I need new step defs
And user enters IDHotel   -> I need another step defs
And user enters price     -> I need another step defs
And user enters dateStart -> I need another step defs
And user enters dateEnd   -> I need another step defs
```

No reusable step definitions.
This is not a good way to create FF.
because no parameterization.
Since we are passing multiple data,
We can parameterize this FF

Parametrizing with “”

```
#   When I parameterize, I can reuse them for different data
And user enters "IDUser"   -> STEP1: I need a new step defs
And user enters "IDHotel"  -> I CAN USE STEP DEFINITION FROM STEP 1
And user enters "price"    -> I CAN USE STEP DEFINITION FROM STEP 1
And user enters "dateStart"-> I CAN USE STEP DEFINITION FROM STEP 1
And user enters "dateEnd"  -> I CAN USE STEP DEFINITION FROM STEP 1
```

There is reusable step refs.
This is good for passing multiple data.
But we have alternatives.

Parametrizing with Scenario Outline

```
Scenario Outline: TC03_user_should_book_a_reservation
#   When I use scenario outline, I can reuse them for different data
  And user enters "<IDUser>"
  And user enters "<IDHotel>"
  And user enters "<price>"
  And user enters "<dateStart>"
  And user enters "<dateEnd>"
Examples: Test Data
|IDUser|IDHotel|price|dateStart|dateEnd|
|userid|1234   |900  |10/30/2021|11/20/2021|
|user4 |1233   |200  |11/10/2021|11/30/2021|
```

Scenario Outline is #1 choice when we want to run SAME scenario with DIFFERENT data.
With Scenario Outline, there is reusable step defs.
This is good for passing multiple data.
But we have alternatives.

Parametrizing with Data Tables

```
#    Sending data using DataTable in cucumber
Scenario: TC03_user_should_book_a_reservation
  Given user enters all required fields
  |idUser |idHotelRoom|price|dateStart |dateEnd   |
  |manager|Ayse       |500  |11/08/2021|11/12/2021|
```

WithDatTables, there is reusable step defs.
This is good for passing multiple data. Note that, scenario outlet is used more than DataTable, cause it is easy to use.

DataTables

- 1. Datatables using a List<String>:

```
@Given("user enters manager_id and manager_password")
public void user_enters_manager_id_and_manager_password(DataTable credentials) {
    List<String> managerCredentials = credentials.row(0);//Test data is on the fi
    System.out.println(managerCredentials);//[manager, Manager1!]
    System.out.println(managerCredentials.get(0));//manager
    System.out.println(managerCredentials.get(1));//Manager1!
    loginPage.username.sendKeys(managerCredentials.get(0));
    loginPage.password.sendKeys(managerCredentials.get(1));
}
```

- 2. Datatables as List<List<String>>

```
//        Getting the data as List<List<String>>
        List<List<String>> managerCredentials = credentials.asLists();
//        System.out.println(managerCredentials);//[[username, passwor
        for (List<String> eachCredentials:managerCredentials){
            System.out.println(eachCredentials);//[username, password]
            System.out.println(eachCredentials.get(0));
            System.out.println(eachCredentials.get(1));
            if (!eachCredentials.get(0).equals("username")) {
            loginPage.username.sendKeys(eachCredentials.get(0));
            loginPage.password.sendKeys(eachCredentials.get(1));
            }
        }
```

- 3. Datatables as List<Map<String,String>>

```
// 3. List<Map<String,String>>
List<Map<String,String>> managerCredentials= credentials.asMaps(String.class,String.class);
System.out.println(managerCredentials);//[{username=manager, password=Manager1!}]
for (Map<String,String> eachCredentials:managerCredentials){
    System.out.println(eachCredentials);
    loginPage.username.sendKeys(eachCredentials.get("username"));
    loginPage.password.sendKeys(eachCredentials.get("password"));
```

Scenario Outline VS Data Tables

- Scenari Outline ile DDT yapılabilir. Scenario Outline iyi bilinmelidir.
- Test adimlarini(STEPS) data reusable yapmak icin kullanilir
- En çok kullanılan cucumber ozelliklerinden biridir
- Examples kelisi veri girisleri icin kullanilmalidir
- Step definitionslardaki data tipi STRINGdir
- Examples kelimesi en sonda kullanilir

- Data Tables ile DDT yapılabilir
- Test adimlarini(STEPS) data reusable yapmak icin kullanilir
- Scenari Outline kadar yaygın degildir
- Examples kelimesi kullanılmaz
- Step definitionslardaki data tipi DATATABLE
- DataTablelar ara adimlardada(STEPS) kullanilabilir

- NOT: Datatables ve Scenario Outline birlikle kullanilabilir

How do you use collections in your framework?

- What is collections in JAVA? -> list, set, queue
- How did you use collections in your framework? -> I use collections when especially I need to store **multiple test data .** I use Collections when I try to get data form Excel, or DataTables. For example, I store the multiple login credentials in my excel sheet, and I use ExcelUtil class to get the data as a List. I actually have ExcelUtil class that has Java Codes to get the data from an excel sheet. And in that ExcelUtil we use collection, I can get the list of each column name, or I can get the list of data in different form.

- I also use collections when I expect multiple elements. For example **findElements()** method returns a List of WebElement- **List<WebElement>**

- **getWindowHandles** returns set of string- **Set<String>**

- When I use Data Table in my cucumber framework, I use collections
  - I can store the data that comes from DataTables as **List<Map<String,String>>**
  - I can also store the data that comes from DataTables as **List<List<String>>** or Just **List<String>**

- I have Utils classes. In the util classes I used collection a lot. For example, I have **Excel Util** class
  - When I get the column names of an excel sheet, I store column names in **List<String>**
  - When I the data list from excel, I put data list in **List<Map<String, String>>**

- Collectionslar nelerdir? -> list, set, queue
- Collectionslari frameworkunde nasıl kullandın? -> Collectionslari ozellikle coklu data ihtiyacimiz oldugunda kullanırız.
- Ornegin, **findElements** metotu kullanildiginda birden fazla element beklediğim icin, List<WebElement> kullanırız.
- Yada, çoklu pencere geçişlerinde **getWindowHandles** ile Set kullanırız.
- Dropdowndaki tum elementleri **getOptions** metot ile aldigimda List<WebElement> e koyariz.
- Frameforkumuzde bazı utility classlar kullanıyoruz. Bunlarin icince çok sayıda collections kullanıyoruz. Ornegin **ExcelUtils** datalarını list e koyan metotlarımız vardır.
- Cucumberda **DataTables** kullandigimda, gelen verileri List yada Map de depolayabiliriz.

Hooks

HOOKS AND TAKING SCREENSHOTS

- Hooks:
  - Hooks nedir?
    - Hooks her bir **Scenario** yada **Scenario Outline** dan **ONCE** yada **SONRA** calistirilan bir classdir.
  - Neden hooks kullanilir?
    - Raporlamalarda (After metottaki kod ile)
    - After metotumuzda bulunan kod ile otomatik olarak ekran goruntusu alma ve rapora ekleme
  - Hooks da ne tur kodlar var?
    - Before ve After metot lari. After da ekran goruntusu almak icin kullandigim kodlar vardir

Checking if a scenario failed
isFailed() method will detect if scenario failed

The screenshot will be attached
If only test case fails
Screenshot will be attached
In the cucumber html file

```
@After
public void tearDown(Scenario scenario){
    final byte[] screenshot=((TakesScreenshot) Driver.getDriver()).getScreenshotAs(OutputType.BYTES);
    if (scenario.isFailed()) {
        scenario.attach(screenshot, "image/png","screenshots");
        Driver.closeDriver();
    }

}
```

HOOKS AND TAKING SCREENSHOTS

- Create a class in step defs folder
- Name : Hooks

@admin_login_test

Scenario Outline: login_with_admin_credentials

- **Given** user is on the application page
- **And** user enters manager id "<admin_id>"
- **And** user enters manager password "<admin_password>"
- **When** user clicks on the login button
- **Then** verify the "<default_page_ID>" is displayed
- **Then** close the application

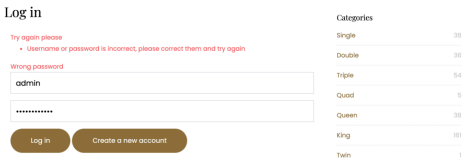
▶ Attached Image

Click on attached image

Examples: admin_data

Screenshot is attached
For failed scenario
Log in failed
And screenshot is captured
By Hooks

▼ Attached Image



Conditional hooks

Conditional hooks

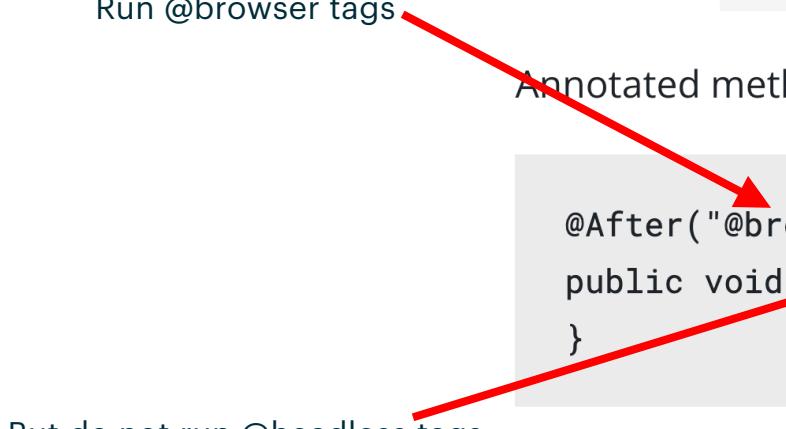
Hooks can be conditionally selected for execution based on the tags of the scenario. To run a particular hook only for certain scenarios, you can associate a `Before` or `After` hook with a tag expression.

Annotated method style:

```
@After("@browser and not @headless")
public void doSomethingAfter(Scenario scenario){
}
```

Lambda style:

```
After("@browser and not @headless", (Scenario scenario) -> {
});
```



Excel Automation in Cucumber

Excel

- Excel automation test verilerini almak icin kullanilir
- Cucumberda degisik sekillerle test dataları alınabilir:
  - Scenario Outline(cucumber)
  - DataTables(cucumber)
  - Excel(cucumber,testng,junit) Apache poi dependency si gerekli

Excel

1.Apache Poi dependencies

```
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>4.1.2</version>
</dependency>


<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>4.1.2</version>
</dependency>
```

Excel

- 2. Feature File e Step definitions lari olustur
  - Feature File : day_excel_automation.feature
  - Step Definitions: ExcelStepDefinitions
  -

```
Feature: FE1010_login_feature
  @excel_admin
  Scenario: admin_login
    Given user log in the application using excel admin
```

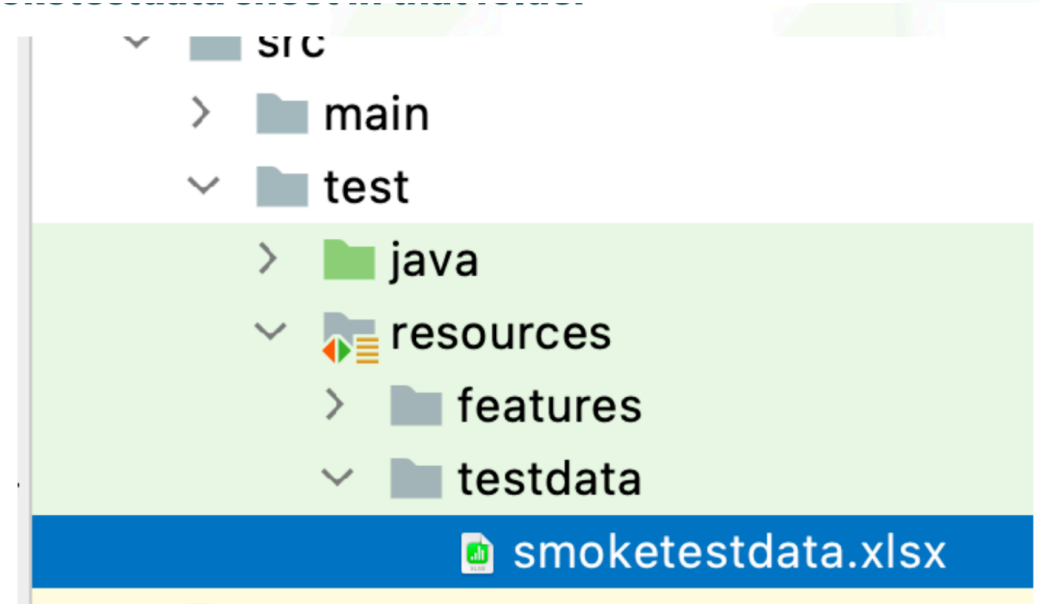
```
@Given("user log in the application using excel admin")
public void user_log_in_the_application_using_excel_admin() {

}
```

Excel

- 3. ExcelUtil i utilities klasorune ekle

Add Excel data in the resources folder

- 4. Resources da yeni bir klasör olustur: testdata
- smoketestdata excel dosyasini bu klasöre ekle



Add Excel data in the resources folder

- 5.step definitions kodlari tamamla

Excel - TestNG vs Cucumber

TEST NG TEST CLASS

```
@Test
public void adminLoginTest() throws InterruptedException {
    String path ="./src/test/java/resources/smoketestdata.xlsx";
    String sheetName="admin_login_info";
    excelUtil= new ExcelUtil(path,sheetName);

    testData=excelUtil.getDataList();
    System.out.println(testData);//[{password=Techproed123!, username=ad

    for(Map<String,String> eachData : testData ){//eachData represent ea
        setUp();//login in each loop
        loginPage.username.sendKeys(eachData.get("username"));//admin
        loginPage.password.sendKeys(eachData.get("password"));//Techproe
        loginPage.loginButton.click();
    }
}
```

CUCUMBER STEP DEFINITION

```
}

@Given("user log in the application using excel admin")
public void user_log_in_the_application_using_excel_admin() throws
        String path ="./src/test/java/resources/smoketestdata.xlsx"
        String sheetName="admin_login_info";
        excelUtil= new ExcelUtil(path,sheetName);

        testData=excelUtil.getDataList();
        System.out.println(testData);//[{password=Techproed123!, us

        for(Map<String,String> eachData : testData ){//eachData rep
            setUp();//login in each loop
            loginPage.username.sendKeys(eachData.get("username"));/
            loginPage.password.sendKeys(eachData.get("password"));/
            loginPage.loginButton.click();
    }
}
```

Failed Scenario lari Tekrar calistirma

Rerun Failed Scenarios

- TestNG de Listeners lari Failed Scenariolari tekrar calistirmak icin kullaniriz
- Cucumber da, sadece **rerun plugin** i Runnera eklememiz yeterlidir

Rerun Failed Scenarios

- **Question** : 500 test cases iniz var diyelim. 6 saatte biter. 10 tanesi kaldi 490 tanesi gecti. Bu durumda ne yapılır?
- **Answer** : Sadece failed olan 10 scenario yu calistiririz.
- **Question** : Nasil?
- **Answer** : Kalan test olursa, sadece bu test caselerin olustugu bir rapor olusturulur. Bu raporda hangi test caselerin fail ettigi gorulur. Sadece bu rapordaki test caseler calistirilir.

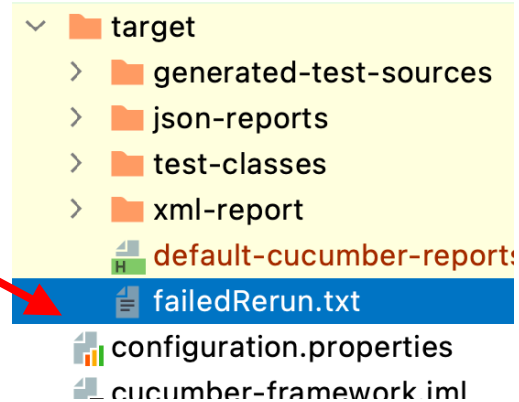
- **STEP 1: Rerun plugins i Runner a ekle**
- **"rerun:target/failedRerun.txt"**
- Bu plugins Sadece Failed test caselerin bulunduğu bir .txt file oluşturur.

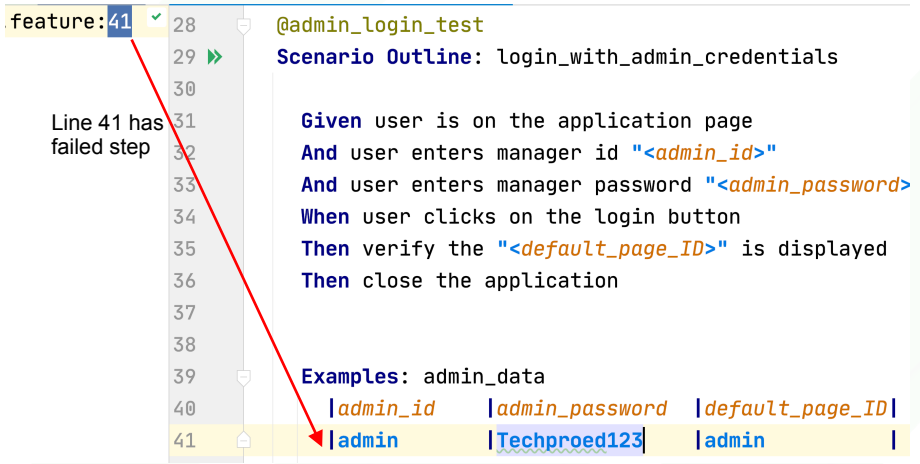
- **STEP 2: Bu .txt file'i calistirmak icin yeni bir Runner class olusturur**

```
package runners;
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;
import org.junit.runner.RunWith;

@RunWith(Cucumber.class)
@CucumberOptions(
        plugin = {
                "html:target/default-cucumber-reports",
                "json:target/json-report/cucumber.json",
                "junit:target/xml-report/cucumber.xml",
                "rerun:target/failedRerun.txt"
        },
        features = "@target/failedRerun.txt",
        glue = "stepdefinitions",
        tags = "@google_search",
        dryRun = false
)
public class FailedScenarioRunner {
}
```

WHAT HAPPENS IF A STEP FAILS?

- Runner .txt fdosyasi oluşturur. Bu dosya sadece failed scenario larin olduğu bir dosya oluşturur





- **Failed Scenario nin absolute path**

```
file:src/test/resources/features/S21.08/Eagles/US100450_user_should_login_using_manager_credentials.feature:41
```

Feature file name

Line number

- Bir test case Fail ettiginde FailedScenarioRunner i calistiririz.

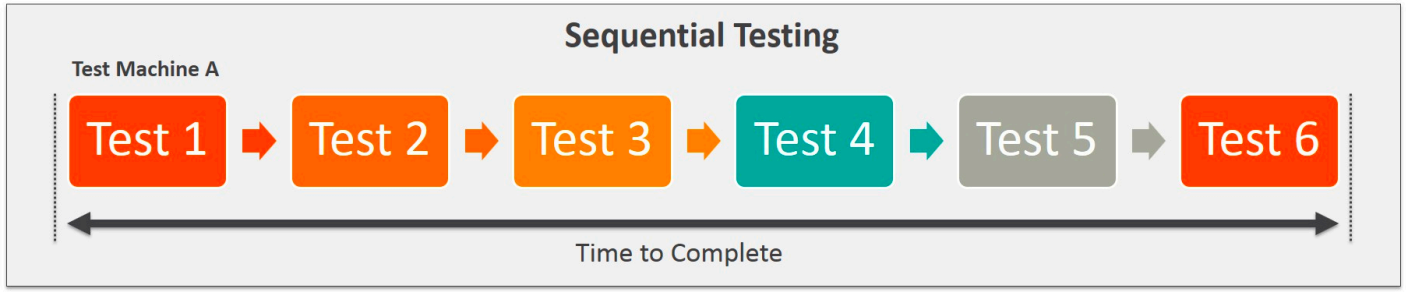
Failed Scenarios

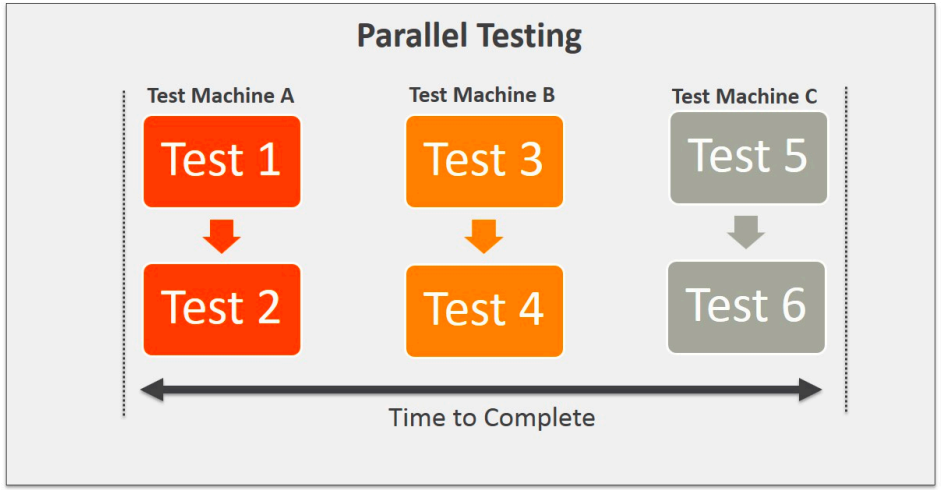
- **1. Failed Scenariolar tekrar nasıl calistirilir?**
  - Rerun plugin yardimiyla olusturulan rapor dosyasini tekrar calistirilir.
- **2. Tum test caselerin gecmesi durumunda .txt dosyasında scenario bulunur mu?**
  - Hayir Failed test case olmamasi durumunda .txt de scerio bulunmaz

Parallel testing

Parallel Testing

- **Parallel Testing nedir?**
  - Ayni anda birden fazla browser da test caseleri calistirma islemine parallel test denir
  - **Thread:** Kod. Bu ornekde 3 thread vardır
  - **Thread-Safe:** Thread-safe, paralel testi mumkun kilan bir ozellikdir. Thread-safe ile ayni anda birden fazla kodu paralel sekilde calistircbiliriz.
- **Parallel Testing yapilma yöntemleri nelerdir?**
  - Cucumber
  - TestNG
  - Selenium Grids
  - Jenkins
- **Alternatif ücretli şirketler de vardır:**
  - Sauce Labs
  - Browser Stack
  - ...





Parallel Testing

- Oncelikle pom.xml de cucumber ile paralel testi mümkün kılan plug ins leri koyabiliriz.
  - Surefire plugin
  - Failsafe plugin
- Sonra birden fazla Runners olusturup, bu runner lari ayni anda maven commend ile calistirilirilarak paralel test yapilir.

- Terminalden maven projesini aclistirmak icin : **mvn clean verify**

Dependency vs Plugin

- **Dependency :**
  - Dependency hair classlari ve metotlari saglar. Ornegin, selenium dependency Actions, Select, JavascriptExecutor, findElements gibi classlari ve metotlari saglar. TestNG dependency, @Test, Assert,… gibi ozellikleri saglar.

- **Plugins:**
  - Plugins daha sinirli ektra ozellikleri saglar. Bu extra özellik bir kodun calismasi(execution) esnasında gorulur. Ornegin, surefire-plugin parallel execution yapmamizi saglar. Cucumber reporter plugins html, json, xml raporlarinin olusmasini saglar.

Parallel Testing-Plug Ins

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-failsafe-plugin</artifactId>
    <version>3.0.0-M1</version>
    <configuration>
        <testFailureIgnore>true</testFailureIgnore>
        <skipTests>false</skipTests>
        <includes>
            <include>**/runners/*TestRunner*.java</include>
        </includes>
    </configuration>
</plugin>
```

1. Failsafe plugin

2. maven-surefire-plugin

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>3.0.0-M1</version>
    <configuration>
        <parallel>classes</parallel>
        <forkMode>perthread</forkMode>
        <threadCount>2</threadCount>
        <reuseForks>false</reuseForks>
        <argLine>-Duser.language=en</argLine>
        <argLine>-Xmx1024m</argLine>
        <argLine>-XX:MaxPermSize=256m</argLine>
        <argLine>-Dfile.encoding=UTF-8</argLine>
        <useFile>false</useFile>
        <includes>
            <include>**/runners/*TestRunner*.java</include>
        </includes>
        <testFailureIgnore>true</testFailureIgnore>
    </configuration>
</plugin>
```

Parallel Testing-Plug Ins

3. maven-cucumber-reporting:

```
<plugin>
    <groupId>net.masterthought</groupId>
    <artifactId>maven-cucumber-reporting</artifactId>
    <version>5.0.0</version>
    <executions>
        <execution>
            <id>execution</id>
            <phase>verify</phase>
            <goals>
                <goal>generate</goal>
            </goals>
            <configuration>
                <projectName>cucumber-framework-21</projectName>
                <outputDirectory>${project.build.directory}</outputDirectory>
                <inputDirectory>${project.build.directory}</inputDirectory>
                <jsonFiles>
                    <param>**/json-report/*.json</param>
                </jsonFiles><classificationFiles>->
                <param>sample.properties</param>
                <param>other.properties</param>
            </classificationFiles>
            </configuration>
        </execution>
    </executions>
```

Parallel Testing-Plug Ins

Failsafe helps test fail safely
It helps other tests run safely even if a test case fails

Plugin name

-Path of the Runner Class
-Class Name : TestRunner
-Folder Name: runners

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-failsafe-plugin</artifactId>
    <version>3.0.0-M1</version>
    <configuration>
        <testFailureIgnore>true</testFailureIgnore>
        <skipTests>false</skipTests>
        <includes>
            <include>**/runners/*TestRunner*.java</include>
        </includes>
    </configuration>
</plugin>
```

```
*TestRunner*
* means ANY
* As long as Runner name
* Contains TestRunner
* Then those classes will be run
* SmokeTestRunner
* RegressionTestRunner
* MyTestRunnerClass
```

Parallel Testing-Plug Ins

Surefire plugin is used to runs tests in parallel

-Number of thread(number of browser)

-Path of the Runner Class
-Class Name : TestRunner
-Folder Name: runners

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>3.0.0-M1</version>
    <configuration>
        <parallel>classes</parallel>
        <forkMode>perthread</forkMode>
        <threadCount>2</threadCount>
        <reuseForks>false</reuseForks>
        <argLine>-Duser.language=en</argLine>
        <argLine>-Xmx1024m</argLine>
        <argLine>-XX:MaxPermSize=256m</argLine>
        <argLine>-Dfile.encoding=UTF-8</argLine>
        <useFile>false</useFile>
        <includes>
            <include>**/runners/*TestRunner*.java</include>
        </includes>
        <testFailureIgnore>true</testFailureIgnore>
    </configuration>
</plugin>
```

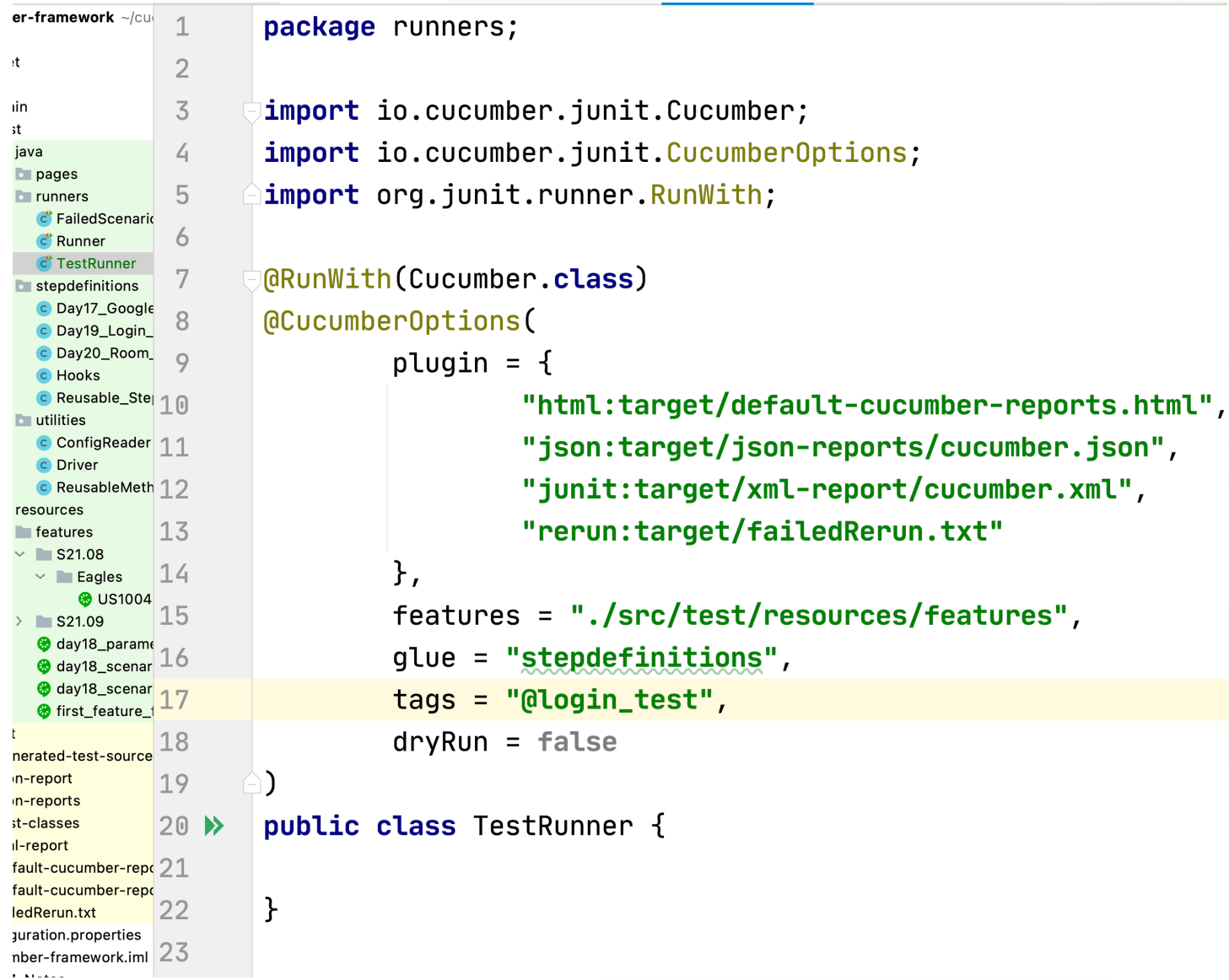
Parallel Testing-Plug Ins

The json reports in Runner class Will be used to generate Reports

```
<plugin>
    <groupId>net.masterthought</groupId>
    <artifactId>maven-cucumber-reporting</artifactId>
    <version>5.0.0</version>
    <executions>
        <execution>
            <id>execution</id>
            <phase>verify</phase>
            <goals>
                <goal>generate</goal>
            </goals>
            <configuration>
                <projectName>cucumber-framework</projectName>
                <outputDirectory>${project.build.directory}</outputDirectory>
                <inputDirectory>${project.build.directory}</inputDirectory>
                <jsonFiles>
                    <param>**/json-reports/*.json</param>
                </jsonFiles><classificationFiles>->
                <param>sample.properties</param>
                <param>other.properties</param>
            </classificationFiles>
            </configuration>
        </execution>
    </executions>
</plugin>
```

Birden Fazla Test Runners

- Add TestRunner classes:
  - SmokeRunner
  - RegressionRunner



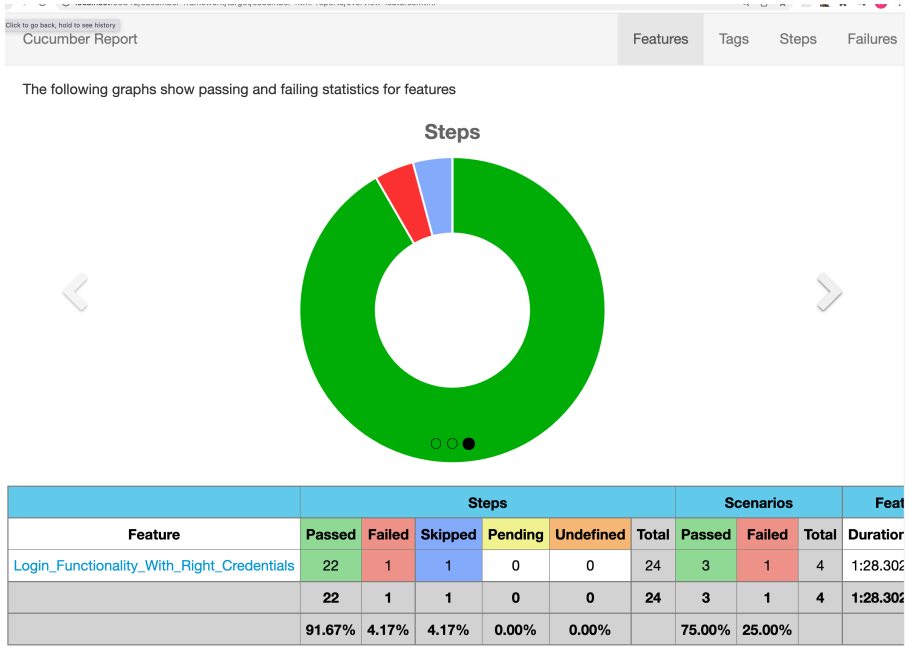
Test Caselerini mvn commend ile Calistir

- Terminali Ac
- Burdaki command lardan birini calistir
  - mvn clean verify
  - mvn clean install

zsh: command not found: mvn. => maven yuklu degil
mvn -version calistirildiginda maven versiyonu gormuyorsaniz maven yuk degildir

Spark Report

Spark Report



	Steps						Scenarios			Feat
Feature	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration
Login_Functionality_With_Right_Credentials	22	1	1	0	0	24	3	1	4	1:28.302
	22	1	1	0	0	24	3	1	4	1:28.302
	91.67%	4.17%	4.17%	0.00%	0.00%		75.00%	25.00%		

Failures Overview

The following summary displays scenarios that failed.

Feature: Login_Functionality_With_Right_Credentials

Tags: @login_test @admin_login_test

Scenario Outline login_with_admin_credentials

Hooks

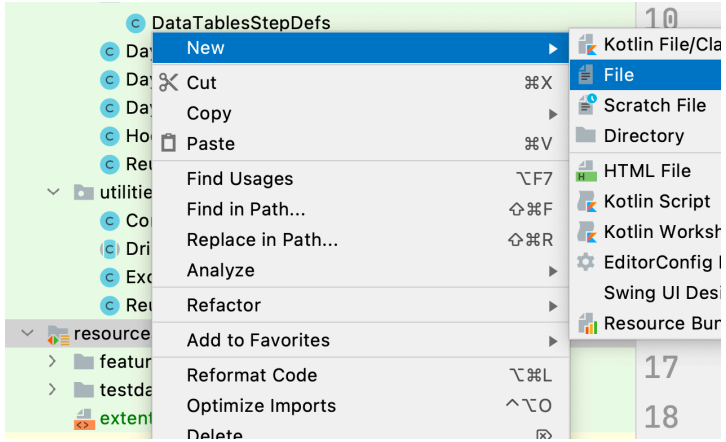
Steps

Hooks

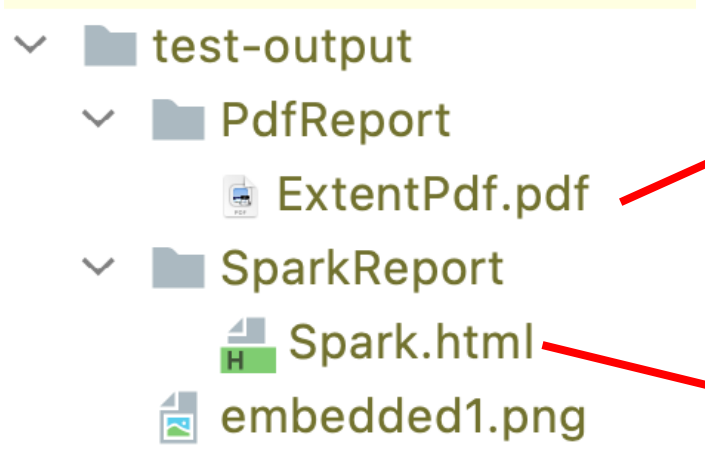
Spark Report

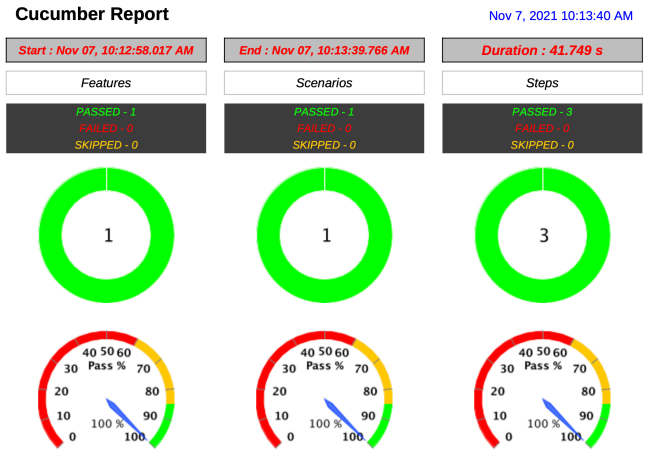
- 1.resources klasorunde **extent-config.xml**
- 2.resources klasorunde **extent.properties**
- 3. Spark report dependency pom a ekle: https://mvnrepository.com/artifact/tech.grasshopper/extentreports-cucumber7-adapter/1.6.0
- 4. Spark report plugin i Runner a ekle

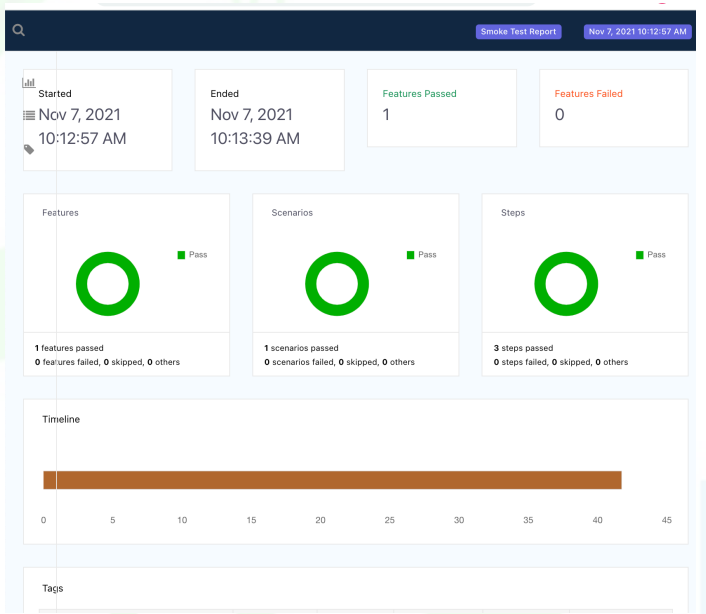
```
"com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:"
```



REPORT FOLDERS







REPORTS IN CUCUMBER

- Cucumberda raporlama nasıl olur?
  - Runners da **cucumber plugin**
    - Html,json,xml
  - **Extent Spark reports:**
    - Dependencies
    - Configurasyon dosyaları
  - **maven-cucumber-reporting** plugin
    - mvn clean verify ile HTML raporlari oluşur
- Ekran goruntuleri raporlara nasıl eklenir?
  - Hooks class daki @After metot da bulunan kod ile ekran goruntusu alınıp fail eden test caselere eklenir
  - Ayrica olusturulan bir step ile (Then ekran goruntusu al) istenildiğinde ekran goruntusu alınabilir
- Test Case Fail ettiginde izlenilen adımlar nelerdir?
  - Test case I tekrar calistirilir
  - Tekrar fail olmuşsa manual test yapılır
  - Bug bulunmuşsa ekran goruntuleri ile rapor oluşturulup Jira da bug ticker oluşturulur.
- Iletisim araçları nelerdir?
  - Genelde Outlook, Skype business, Slack gibi guvelilir iletişim araçları.

Framework Tipleri

- **Data driven, keyword driven**, ve **Hybrid** framework arasındaki farklar nelerdir?
  - DATA DRIVEN:
    - Cok sayida data ile test etme islemidir. Bu yontemde, cok sayida data ile test ler yapilir ve hepsinin gecmesi beklenir. Bir test case fail ederse, developer duzeltir, tester tekrar test eder. Tum test caseler gecene kadar bu islem devam eder. Excel data, database data, xml data, scenario outline data,…
  - KEYWORD DRIVEN:
    - Keyword driven de excel isin merkezindedir.
    - Test Step:
    - Test Object: Web Page object/element, Username & Password….
    - Action: Such as click, open browser, input etc.
    - Test Data:
  - HYBRID FRAMEWORK
    - Behavior Driven Testing + Data Driven Testing = Hybrid Framework
  - BDD FRAMEWORK
    - Once behavior (fonksiyon, özellik, davranis) tanımlanır, sonrasında kod yazılır.