



*****CUCUMBER*****

TechPro Education

Copyright Material

For only TechPro Education Students



cucumber

- Cucumber bizim son framework'umuz olacak.
- Cucumber **BDD** (behaviour driven development) (Davranış tabanlı geliştirme) yaklaşımı için kullanılmakta olan açık kaynak kodlu bir kütüphanedir.
- Cucumber bir iş ararken önemli bir rol alacaktır.
- TestNg hakkında her şeyi bilmemek sorun değil ama Cucumber hakkında her şeyi bilmeniz GEREKİR.
- Agile metodolojisinde, insanlar uygulamanın işlevsellliğini geliştirmek için birlikte çalışmak zorundadır. İnsanlar development sırasında birlikte hareket etmeli.

```
Feature: US1001 Ck Hotels Login
@wip
Scenario: TC01 kullanıcı gecerli bilgilerle giriş yapar
    Given kullanıcı Ck Hotels ana sayfasında
    Then Log in yazısına tıklar
    And gecerli username girer
    And gecerli password girer
    And Login butonuna basar
    Then sayfaya giriş yaptığını kontrol eder
    And kullanıcı sayfayı kapatır
```



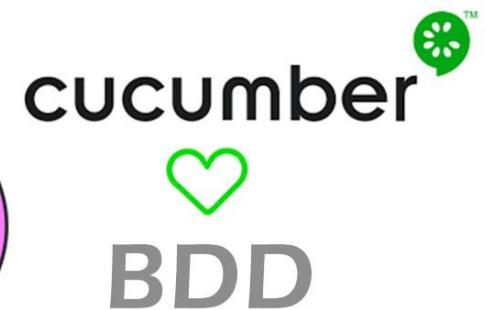
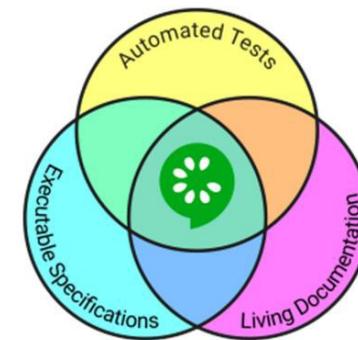
- BDD(behaviour driven development) (Davranış güdümlü geliştirme)- ilk olarak behavior(davranis) veya functionalitileri yaziyorsunuz (Epic=Feature, Story, AC, etc), daha sonra development and testing baslıyor.
- BDD'de behaviour'lar başarısız olduğunda kod başarısız olur..
- Anlasılabilir Gherkin Language nedeniyle BDD development icin Cucumber harika bir uygulamadır.
- **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

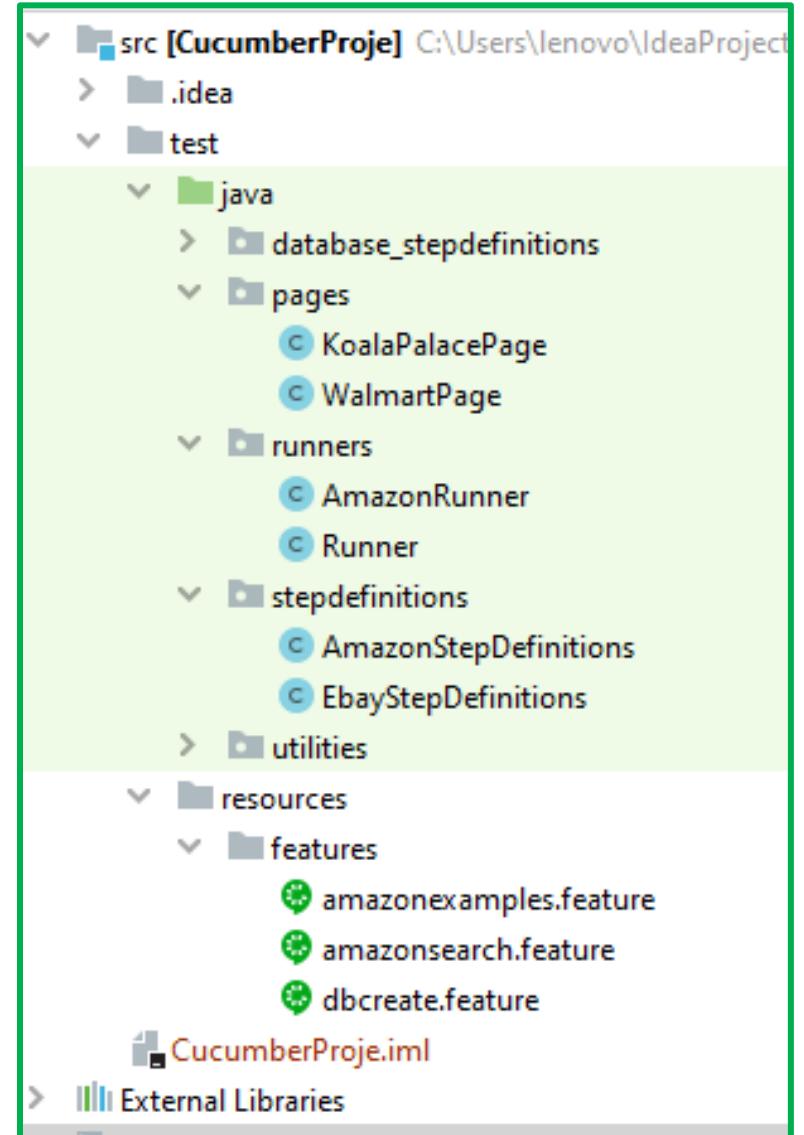
Then anahtar kelimesi ile de sonuç tanımlanır.

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





- 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 testerların yaptığını anlayamayabilir. Gherkin onların da testlerimizi anlamalarını kolaylaştırır





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 sorunları halletmek icin compiler dependency yuklenebilir

<properties>

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

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

</properties>



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**



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(tanımlanmamış) 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.ENTER);
    }

    @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

```
feature x day18_scenario_outline_1.feature x day18_scenario_outline_2.feature x tirs
  @google_search
  Feature: Google_Search_Functionality
```

- Scenario

```
    ] @teapot
    Scenario: TC02_teapot_search
      And user search for tea pot
```

- Scenario Outline

```
    # YES
    @admin_login_test
    Scenario Outline: login_with_admin_credentials
```

- Examples

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



Tags

Expression

Description

@fast

Scenarios tagged with @fast

@wip and not

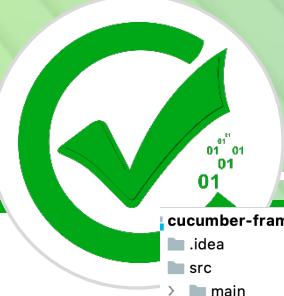
Scenarios tagged with @wip that aren't also tagged with
@slow

@smoke and @fast

Scenarios tagged with both @smoke and @fast

@gui or @database

Scenarios tagged with either @gui or @database



And tag

```
cucumber-frame 7 @smoke @regression
 8 Scenario: TC01_iPhone_search
 9   And user search for iPhone
10  Then verify the result has iPhone
11  Then close the application
12
13 @teapot @smoke @regression
14 Scenario: TC02_teapot_search
15   And user search for tea pot
16  Then verify the result has tea pot
17  Then close the application
18
19 # Write 2 more test cases in this file
20 # TC03_flower search and verify if the result has flower
21 # TC04_tesla search and verify if the result has tesla
22 # make sure to use reusable methods for step 1 and step 2
23 Scenario: TC03_flower_search
24   And user search for flower
25   Then verify the result has flower
26   Then close the application
27 @smoke @tesla
28 Scenario: TC04_tesla_search
29   And user search for tesla
6
7 @RunWith(Cucumber.class)
8 @CucumberOptions(
9   features = "./src/test/resources/features",
10  glue = "stepdefinitions",
11  tags = "@smoke and @regression",
12  dryRun = false
13 )
14 public class Runner {
15 }
16 /**
17  * -Runner is used to run feature files
18  * -@RunWith is used to run the class. Without this,
19  * -@CucumberOptions is used to add feature path, step
20  * -features : path of the features folder. this
21  * -glue : path of step definitions folder. the
22  * -tags : this marks which feature file to run
23  * -dryRun : -dryRun is used to check if there is
24  * -dryRun takes 2 values, true or false
25  * -dryRun = false :Runs tests on the command
26  * -dryRun = true :Runs tests without
27 */
```

Runner

Tests passed: 2 of 2 tests – 9 s 718ms

runners.Runner

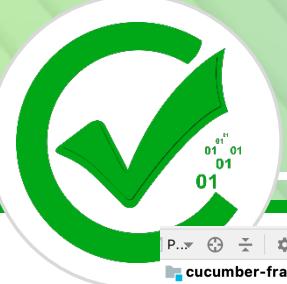
Google_Search_Functionality #3

TC01_iPhone_search

TC02_teapot_search

/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java ...

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".



Or tag

The screenshot shows the IntelliJ IDEA interface with several files open:

- cucumber-framework**: Project structure.
- Runner.java**: Java code defining a runner class with Cucumber options.
- day18_scenario_outline_1.feature**, **day18_scenario_outline_2.feature**, **first_feature_file.feature**: Feature files containing scenarios.
- Runner**: Runner configuration file.

Annotations in the code and feature files highlight specific tags:

- @smoke @regression**: Annotates the first scenario in each feature file.
- @teapot @smoke @regression**: Annotates the second scenario in each feature file.
- @tesla**: Annotates the third scenario in each feature file.

The **Runner.java** code includes the following relevant configuration:

```
@RunWith(Cucumber.class)
@CucumberOptions(
    features = "./src/test/resources/features",
    glue = "stepdefinitions",
    tags = "@smoke or @regression",
    dryRun = false)
```

The **Runner** configuration file contains:

```
public class Runner {}
```

The bottom panel shows the test results:

- Tests passed: 3 of 3 tests - 13s 878ms
- Test cases: runners.Runner, Google_Search_Functionality #3, TC01_iPhone_search, TC02_teapot_search, TC04_tesla_search
- Log output:
/Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin/java ...
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation.



Background

- @Background prerequisites(Tekrar eden on koşullarda kullanılır)
- Gereksiz tekrarlardan kurtulmak için kullanılır
- Her bir SCENARIO dan önce kullanılır
- Ornegin, 5 scenario kullanılmışsa background 5 kez çalışır.
- @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

- Kodlarımızı parametreli ve dinamik hale getirmek için feature file da degisen olarak kullanacagimiz kelimeyi çift tırnak " " icine alırız.

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

- Kodlarımızı parametreli olarak yazdıktan sonra sadece " " içindeki değeri değiştirerek test datalarını feature file dan kontrol edebiliriz.
- Kodlarımızı parametreli olarak yazmak framework'u daha dinamik hale getirir(kodumuz artık hard coded degildir diyebiliriz).



PARAMETRIZING FF USING “”

PARAMETRESİZ

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

PARAMETRELİ

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"



Scenario Outline



Scenario Outline

- **Scenario Outline** aynı **Scenario** i farklı değerlerle tekrar tekrar run etmek için kullanılır
- “<column_name>” ifadesi verileri parametre şeklinde göndermek için kullanılır
- **Examples**, Scenario Outline dan sonra verileri girmek için kullanılmalıdır.
- Scenario Outline, **Data Driven Test** için kullanılır
- 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 için
- Birden fazla veri göndermek için
- Feature file kısa hale getirmek için. Birden fazla Scenario yerine tek bir Scenario Outline kullanılabilir

- Scenario Outline AYNI SCENARIO yu FARKLI VERILERLE test etmek için kullanılır
 - Ornek: birden fazla kullanıcı bilgileriyle sayfaya girmek
 - Kullanici Bilgileri={name=Jim,id=1}, {name=John,id=10},{name=Sam,id=80},...
 - Birden fazla kullanıcı hesabı oluşturmak
 - Birden fazla ürün 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	7 kez bu scenario calisacakdir
flower	Burda aslinda 7 Scenario vardir
tesla	Ve tekbir Scenario Outline vardir
bmw	
buick	
nissan	

```
Talking: Techpro Education Z4
@Then("verify the result has {string}")
public void verify_the_result_has_tesla() {
    String title = Driver.getDriver().getTitle();
    Assert.assertTrue(title.contains("tesla"));
}

@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));
```



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

Create new entry x

First name:	Jonny
Last name:	Walker
Position:	SDET
Office:	New York
Extension:	123
Start date:	2021-11-07
Salary:	100000

Create

Name	Position	Office
New	Edit	Delete
Jonny Walker	SDET	New York

Search: Jonny



HOMEWORK

➤ **USER STORY :** US167854_manager_login_test

➤ **NAME:** kullanıcı tüm manager login bilgileriyle giriş yapabilmeli

➤ **AC:**

➤ Given kullanıcı ana sayfada

➤ When kullanıcı login sayfasına gider

➤ And kullanıcı adını girer->>>>parametre

➤ And şifreyi girer->>>>parametre

➤ And login buttonuna basar

➤ Then login işlemi 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 kullanalım



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 raporları oluşturabiliriz
- En onemlisi ve kullanislısı 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: İlerleyen derslerimizde maven-cucumber-reporting plug ins raporlar alacağ

Ayrıca Spark HTML and PDF reporları da ekleyeceğ(EXTENT REPORTS A BENZER)



Cucumber HMTL reports

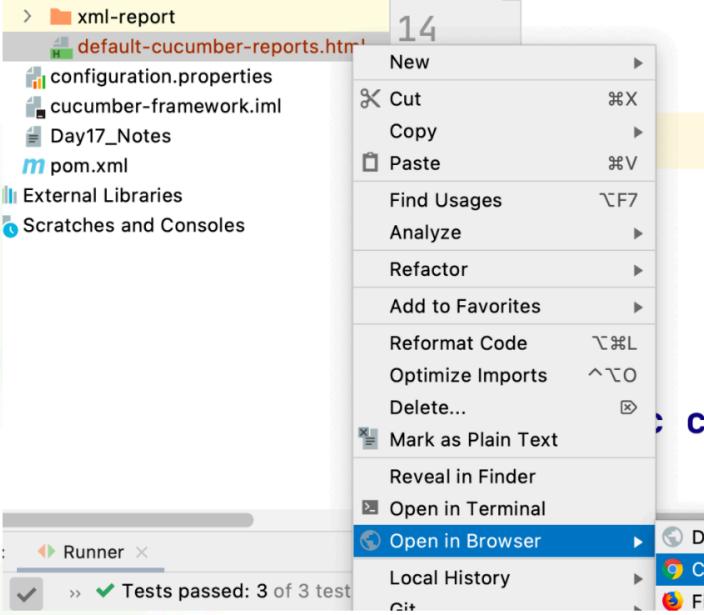
```
plugin = {
    "pretty",
    "html:target/default-cucumber-reports.html",
    "json:target/json-reports/cucumber.json",
    "junit:target/xml-report/cucumber.xml"
},
monochrome=true,  
  

"pretty" : console bilgilerin okunaklı olması
monochrome=true : console bilgilerin okunaklı olması
```



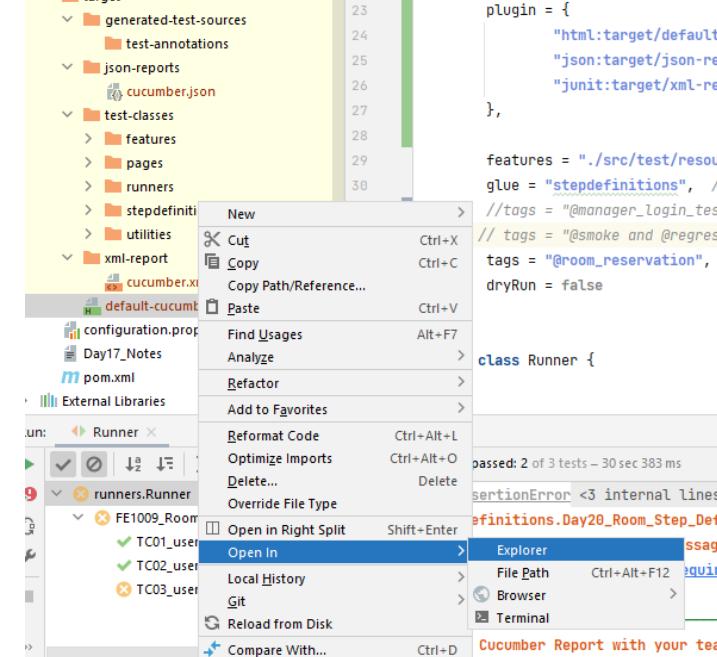
How to Open Reports

14

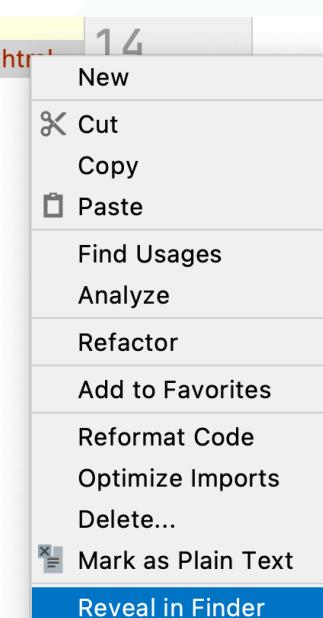


feature
glue =
tags =
dryRun
// WI
// we

class F



xmi-report
default-cucumber-reports.html
configuration.properties
cucumber-framework.iml
Day17_Notes
pom.xml
External Libraries
Scratches and Consoles



Cucumber Report with your team



Cucumber HMTL reports

Pass

- ✓ Given user navigates to the create_room_reservation page

Scenario: TC03_user_should_book_a_reservation

- ✓ Given user enters all required fields

idUser	idHotelRoom	price	dateStart	dateEnd	adultAmount	childrenAmount	contactNameSurname	contactPhone	contactEmail	notes
manager	Ayse	500	11/08/2021	11/12/2021	2	3	test	(999) 999-9999	test@gmail.com	test note

Fails

- ✓ And user clicks on the approved_checkbox
- ✓ And user clicks on the paid_checkbox
- ✓ And user clicks on the save_button
- ✗ Then user verifies the success_message

```
org.junit.ComparisonFailure: expected:<[RoomReservation was inserted successfully]> but was:<[]>
```

```
at org.junit.Assert.assertEquals(Assert.java:117)
```

```
at org.junit.Assert.assertEquals(Assert.java:146)
```

```
at stepdefinitions.Day20_Room_Reservation_Step_Definitions.user_verifies_the_success_message(Day20_Room_Reservation_Step_Definitions.java:78)
```

```
at *.user verifies the success_message(file:///C:/Users/PC/IdeaProjects/cucumber-framework/.src/test/resources/features/S21.09.Eagles/US1004)
```

Skip

- Then close the application



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 yüklenecektir. Fakat GitHub ile girls yapilmassa 24 saat icinde silinecektir. Bu yuzden GitHub ile giriș yapalim.
- 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 |  
| userid1 | 1234 | 900 | 10/30/2021 | 11/20/2021 |  
| user4 | 1233 | 1200 | 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 |
```

With DataTables, 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 first row
    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, password], [manager, Manager1!]]
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 kullanılır
- En çok kullanılan cucumber ozelliklerinden biridir
- Examples kelisi veri girişleri icin kullanılmalıdır
- Step definitionslardaki data tipi STRINGdir
- Examples kelimesi en sonda kullanılır

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

- NOT: Datatables ve Scenario Outline birlikle kullanılabilir



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 from 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 özellikle çoklu data ihtiyacımız olduğunda kullanırız.
- Orneğin, **findElements** metodu kullanıldığında birden fazla element beklediğim için, **List<WebElement>** kullanırız.
- Yada, çoklu pencere geçişlerinde **getWindowHandles** ile Set kullanırız.
- Dropdowndaki tüm elementleri **getOptions** metod ile aldığımızda **List<WebElement>** e koyarız.
- Frameforkumuzde bazı utility classlar kullanıyoruz. Bunların içince çok sayıda collections kullanıyoruz. Orneğin **ExcelUtils** datalarını list e koyan metodlarımız vardır.
- Cucumberda **DataTables** kullandığımızda, 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

www.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



Screenshot is attached
For failed scenario

Log in failed

And screenshot is captured
By Hooks

▼ Attached Image



Log in

Try again please
• Username or password is incorrect; please correct them and try again

Wrong password

admin

.....

Log in Create a new account

Categories

Single	30
Double	38
Triple	54
Quad	6
Queen	30
King	101
Twin	1
Double-double	10
Studio	203

Recent Blog	
	Even the all-powerful Pointing has no control about the blind texts 6 on 21.09.2019
	Even the all-powerful Pointing has no control about the blind texts 5 on 21.09.2019



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**.

Run @browser tags

Annotated method style:

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

But do not run @headless tags

Lambda style:

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



Excel Automation in Cucumber



Excel

- Excel automation test verilerini almak için kullanılır
- Cucumberda değişik sekillerle testdataları alınabilir:
 - Scenario Outline(cucumber)
 - DataTables(cucumber)
 - Excel(cucumber,testng,junit) Apache poi dependency si gereklidir



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() {
    |
}
```



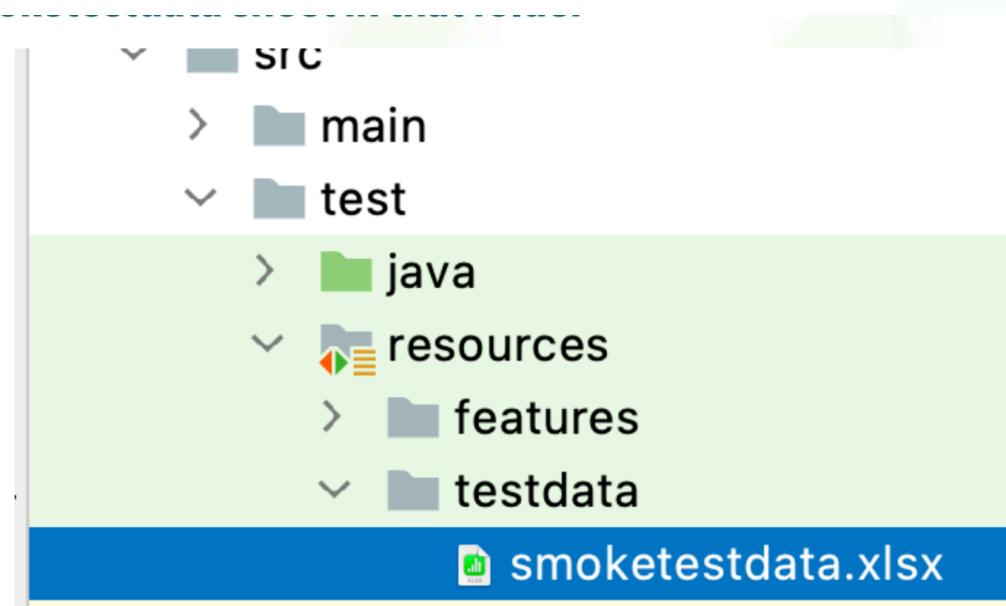
Excel

- 3. ExcelUtil i utilities klasorune ekle



Add Excel data in the resources folder

- 4. Resources da yeni bir klasör oluştur: testdata
- smoketestdata excel dosyasını bu klasöre ekle





Add Excel data in the resources folder

➤ 5.step definitions kodlari tamamla



Excel - TestNG vs Cucumber

TESTING 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")); //Techpro
            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 iari Failed Scenario'ları tekrar calistirmak icin kullaniriz
- Cucumber da, sadece **rerun plugin** i Runner'a eklememiz yeterlidir



Rerun Failed Scenarios

- **Question** : 500 test casesiniz var diyelim. 6 saatte biter. 10 tanesi kaldı 490 tanesi geçti. Bu durumda ne yapılır?
- **Answer** : Sadece failed olan 10 scenario'yu çalıştırırız.
- **Question** : Nasıl?
- **Answer** : Kalan test olursa, sadece bu test caselerin oluşturduğu bir rapor oluşturulur. Bu raporda hangi test caselerin fail ettiği görürlür. Sadece bu rapordaki test caseleri çalıştırılır.

➤ 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 çalıştırmak için yeni bir Runner class oluşturur

```
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 dosyası oluşturur. Bu dosya sadece failed scenario ların olduğu bir dosya oluşturur

A screenshot of a file explorer window. The 'target' folder contains several sub-folders: 'generated-test-sources', 'json-reports', 'test-classes', and 'xml-report'. Below these is a file named 'default-cucumber-report.html'. A red arrow points from this file to another red arrow pointing at a screenshot of a code editor.

The code editor shows a Cucumber feature file with the following content:

```
feature:41
  @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
  Examples: admin_data
    |admin_id|admin_password|default_page_ID|
    |admin|Techproed123|admin|
```

Line 41 is highlighted in yellow, indicating a failed step. The line number '41' is also highlighted in yellow. A red arrow points from the 'failedRerun.txt' file in the file explorer to this line in the code editor.

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 ettiğinde FailedScenarioRunner i çalıştırırız.



Failed Scenarios

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



Parallel testing



Parallel Testing

➤ Parallel Testing nedir?

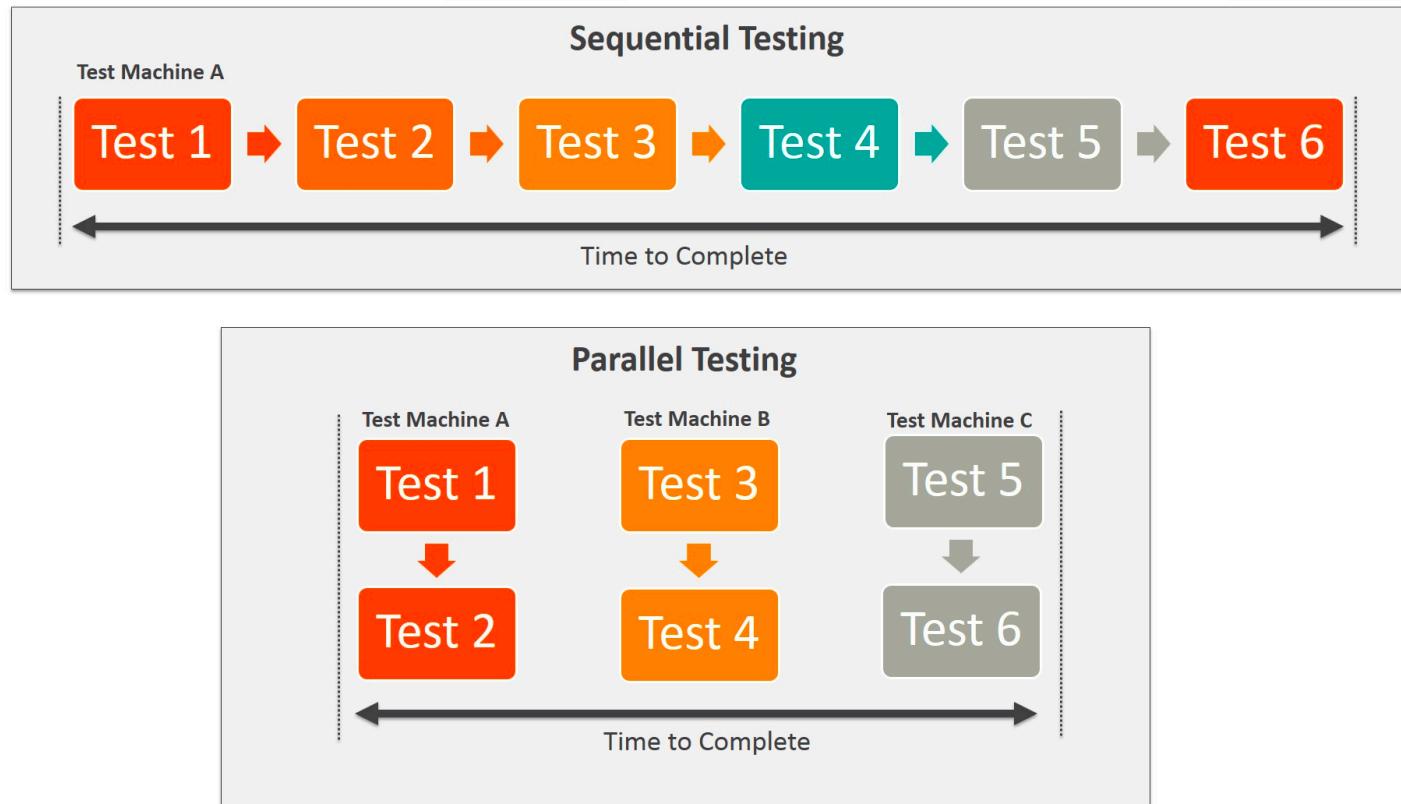
- Aynı anda birden fazla browser da test caseleri calistirma islemine parallel test denir
- **Thread:** Kod. Bu ornekde 3 thread vardır
- **Thread-Safe:** Thread-safe, paralel testi mumkun kilan bir ozellikdir. Thread-safe ile aynı anda birden fazla kodu paralel sekilde calistircibiliz.

➤ Parallel Testing yapılma 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 aynı 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 extra ozellikleri saglar. Bu extra ozellik bir kodun calismasi(execution) esnasinda 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

TestRunner

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

```
<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>
```



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

The screenshot shows a Java code editor with a file named `TestRunner.java` open. The code defines a `TestRunner` class that extends `Cucumber` and implements `RunWith`. It uses `CucumberOptions` to specify report generation and feature loading. The code editor highlights several parts of the code in green and yellow, and the `TestRunner` class name is underlined in red, indicating it's a reference to another part of the code.

```
1 package runners;
2
3 import io.cucumber.junit.Cucumber;
4 import io.cucumber.junit.CucumberOptions;
5 import org.junit.runner.RunWith;
6
7 @RunWith(Cucumber.class)
8 @CucumberOptions(
9     plugin = {
10         "html:target/default-cucumber-reports.html",
11         "json:target/json-reports/cucumber.json",
12         "junit:target/xml-report/cucumber.xml",
13         "rerun:target/failedRerun.txt"
14     },
15     features = "./src/test/resources/features",
16     glue = "stepdefinitions",
17     tags = "@login_test",
18     dryRun = false
19 )
20 public class TestRunner {
21 }
22 }
```



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**