



29 ARALIK 2020  
DERS 28

**Cucumber**  
**@tags, Background**  
**dryRun, Html Reports**  
**Parametre ile kullanma**

Mehmet BULUTLUOZ  
Elektronik Muh.

TECH PRO ED

---



## Önceki Dersten Aklımızda Kalanlar

- Cucumber: BDD behaviour DrivenDevelopment (Davranış tabanlı geliştirme) kodlarımızı kullanıcının davranışlarına göre yazdığımız için bu şekilde adlandırılmıştır
- Testlerimizi yazmaya feature dosyasında sözel olarak kullanıcı, adımlarını yazmakla başlıyoruz. Feature dosyasındaki adımları teknik olmayan insanlar da anlayabildiği için takımın birlikte hareket etmesini kolaylaştırıyor.
- Feature dosyasında adımları yazarken Gherkin language kullanıyoruz. Bu dil temel İngilizce ile adımları tanımlıyor. Kullanabileceğimiz kelimeler Given, When, And, Then. Kelimelerin kod açısından hiç bir farkı yok ancak okuyan kişinin anlayabilmesi için günlük konuşma diline uygun akış ile kelimeleri kullanıyoruz.
- Cucumber framework çalıştırmak için pom.xml'e selenium java, webdrivermanager, cucumber java ve cucumber junit (@RunWith) dependency'lerini yüklüyoruz
- Test oluşturma adımları 1- feature dosyasına sözel olarak adımları yazıyoruz 2- runner classını çalıştırıp gerekli methodları oluşturuyoruz 3- runner'da oluşan kodları stepdefinition classına yapılandırıp orada methodların içine kodları yazıyoruz

---

# cucumber

## dryRun=true

- **dryRun=true** test yürütülmeden çalışır.
- Temel amaç, adımları uygulamadan eksik adımların(missing step definitions) oluşturulmasıdır. Bunu, özellik dosyamızda uygulanmayan herhangi bir adım olup olmadığını kontrol etmek için kullanabiliriz.
- Yürütmeyi önlemek için `dryRun = true` ayarlıyoruz ve yalnızca uygulanmayan feature için method üretiyoruz.
- By default(default olarak), **dryRun=false**.

```
@CucumberOptions(  
  Features = "src/test/resources/features",  
  glue = "stepdefinitions"  
  dryRun = false  
)
```

TECHPROED

---



# cucumber

## Background

Farklı senaryoların başında ortak adımlarımız varsa:

1. Feature file in basına Background oluşturun.
2. Bu ortak adımları Background altına yazın.
3. Background, aynı Feature file'daki her Scenario'dan önce çalışır
4. Duplication olmadigindan emin olun.  
Background un altındaki adımı yazdıktan sonra senaryolardan silin.

```
Feature: First Feature

Background: User search for amazon on hte google page
  Given user is on the google page
  And user search for amazon

@googlesearch
Scenario: TC02_Google search test
  Then user should should see amazon link on the search result
  Then clear the search box
  #Scenario 2 Search for flower and check if the the page related images
  Given user search for flowers on google page
  Then verify the page has flower

@googlesearch
Scenario: TC04_Google search test 2
  When user click on the first link
  Then user verify the amazon page displays
```

This two step will run before each scenario on the same feature file. In this example there are two Scenario: keyword So Background will run twice

**Class Work** :US1001 feature file'daki tekrar eden aramalar yerine parameter kullanarak arama yapabilecegimiz sekilde US1002 feature file veTC02 parameter ile arama Scenerio'su olusturalim



- Tag leri yalnızca belirli senaryoları (scenario) çalıştırmak için kullanırız.
- Tag leri senaryolarımızı gruplandırmak için kullanabiliriz (smoke test, regression test, vs.)

```
@CucumberOptions(  
  features = "src/test/resources/features",  
  glue = "stepdefinitions",  
  dryRun = false,  
  tags="@amazon",  
)
```

```
Feature: US1001 amazon page search
```

```
@amazon @search @apple
```

```
Scenario: TC01 amazon arama testi
```

```
Given kullanıcı amazon sayfasına gider
```

```
And "apple" için arama yapar
```

```
Then sonuçların "apple" içerdiğini test eder
```

```
Then sayfayı kapatır
```

TECHPROED



## Html Rapor Ekleme

- Cucumber raporları, şirketlerin Cucumber kullanmasının ana nedenlerinden biridir.
- Html rapor almak için runner classına eklenti(plugin) eklememiz yeterlidir.

```
@CucumberOptions(  
    plugin={"html:target\\cucumber-reports.html"},  
    features = "src/test/resources/features",  
    glue = "stepdefinitions",  
    dryRun = false,  
    tags="@amazon"  
)
```

```
@RunWith(Cucumber.class)  
@CucumberOptions(  
    plugin={"html:target/cucumber_rapor.html"},  
    features = "src/test/resources/features",  
    glue="stepdefinitions",  
    tags="@amazon",  
  
    dryRun = false  
)
```

TECHPROED





## Feature File'i Parametre ile Kullanma

- Parametrelendirmek ve dinamik hale getirmek için feature file da çift tırnak " " kullanırız.
- Şimdi sadece " " içindeki değeri değiştirerek test datalarını feature file dan kontrol edebiliriz.
- Bu framework'u daha dinamik hale getirir, yani kodumuz artık hard coded değildir diyebiliriz.

**Class Work :** US1001 de kullandığımız feature dosyası altında yeni bir Scenario oluşturalım TC03 ve orada yaptığımız aramayı parametre kullanarak yapalım

```
Feature: US1000 Amazon search test

Scenario: TC01 iphone aramasi testi
  Given kullanıcı amazon sayfasına gider
  And iphone için arama yapar
  Then sonuçların iphone içerdiğini test eder

Scenario: TC02 tea pot aramasi testi
  Given kullanıcı amazon sayfasına gider
  And tea pot için arama yapar
  Then sonuçların tea pot içerdiğini test eder

Scenario: TC03 flower aramasi testi
  Given kullanıcı amazon sayfasına gider
  And flower için arama yapar
  Then sonuçların flower içerdiğini test eder
  Then sayfayı kapatır
```

```
Feature: US1000 Amazon search test

Scenario: TC01 iphone aramasi testi
  Given kullanıcı amazon sayfasına gider
  And "iphone" için arama yapar
  Then sonuçların "iphone" içerdiğini test eder
  Then sayfayı kapatır
```



## Feature File'i Parametre ile Kullanma

- **Scenario Outline**: ayni testte birden fazla datayi kullanmamizi saglar
- Bir liste kullanmak istedigimiz degeri “<value>” seklinde yazariz
- Daha sonra testin sonuna **Examples**: yazip ilk satir olarak |value| yazariz ve altina kullanmak istedigimiz degerleri ekleriz. (|elma|,|armut|... gibi)

```
@amazon @search @apple
Scenario Outline: TC01 amazon arama testi

    Given kullanıcı amazon sayfasına gider
    And "<kelime>" için arama yapar
    Then sonuçların "<kelime>" içerdiğini test eder
    Then sayfayı kapatır

Examples:
|kelime|
|teapot|
|flower|
|avsfdnfhvbjl|
```

**Class Work** : US1001 de kullandigimiz feature dosyasi altinda yeni bir Scenario olusturalim TC04 ve orada yaptigimiz aramayi **Scenario Outline** kullanarak farkli urunler icin yapalim





## Class Work

Yeni bir feature file olusturalim: **US1002**\_ConfigReaderdan\_parameter\_alma.feature

Yeni bir Scenario olusturalim: TC05 Configreader ile farkli parametreler kullanma

Configuration.properties dosyasinda olusturacagim farkli web sitesi url'lerini okuyup o sayfalara gidecek ve sayfa url'lerini test edecek stepDefinitions'i olusturup runner ile kodumuzu calistiralim

Scenario Outline: TC01 amazon arama testi

Given kullanıcı "<sayfa\_url>" sayfasına gider  
And url'in "<sayfa\_url>" olduğunu test eder  
Then sayfayı kapatır

Examples:

|sayfa\_url|  
|amazon\_url|  
|bestbuy\_url|  
|ebay\_url|