CUCUMBER

1.Gün:

Daha önce projemize kütüphane yüklemenin iki farklı yolunu görmüş ve bir üçüncü yol olduğundan bahsetmiştik. Artık cucumber için oluşturacağımız projemizde bu üçüncü yöntem ile kütüphane yönetimimizi sağlayacağız. Projemizi oluştururken kodlama dili olarak Java ve Build System olarak da Maven seçiyoruz. JDK mızı 11 olarak ayarlayalım.

A screenshot of a computer

Description automatically generated

Projeyi açtığımızda bizi pom.xml dosyası karşılayacak.

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>Cucumber6</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>

Proje yapımızda bir src klasörümüz mevcut. Bu klasörün altında yapı proje ilk açıldığında şu şekilde olacak;

A screenshot of a computer

Description automatically generated

Biz test dosyasının altında klasörlerimizi oluşturacağız. Page Object Model yapısında projemizi tanımlayacağız. Bu yapı; FeatureFiles package, Pages package, Runners package, StepDefinitions package ve Utilities package olarak açılacak.

FeatureFiles 🡪 Test senaryolarımızın ve test case’lerimizin olduğu klasör

StepDefinitions 🡪 Senaryolarımızın kodlarının yazıldığı class’lar bu klasörde olacak

Pages 🡪 Web uygulamamızın sayfaları burada olacak

Utilities 🡪 Driver ayarlarımız ve ortak kullanılacak olan metodların tanımlandığı class’lar bu klasörde yer alacak

Runners 🡪 Testlerin çalıştırılacağı class’lar burada yer alacak

pom.xml dosyamızda ilk olarak Maven repository’sinden çağıracağımız kütüphaneler şu şekilde olacak;

<dependencies>  
 <dependency>  
 <groupId>org.seleniumhq.selenium</groupId>  
 <artifactId>selenium-java</artifactId>  
 <version>4.11.0</version>  
 </dependency>  
 <dependency>  
 <groupId>io.cucumber</groupId>  
 <artifactId>cucumber-java</artifactId>  
 <version>7.9.0</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.testng</groupId>  
 <artifactId>testng</artifactId>  
 <version>6.14.3</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>io.cucumber</groupId>  
 <artifactId>cucumber-testng</artifactId>  
 <version>7.9.0</version>  
 <scope>test</scope>  
 </dependency>

Dependency’leri ekledikten hemen sonra sağ tık Maven başlığı altında Reload Project dediğimizde eklediğimiz kütüphaneleri Maven reposu içerisinden indirip yükleyecektir.

A screenshot of a computer

Description automatically generated

Intellij içerisinde plugins olarak Cucumber for Java ve Gherkin eklememiz gerekiyor.

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Artık ide’mizi restart ettikten sonra hazırız.

FeatureFiles içerisine \_01\_Login.feature dosyasını oluşturuyoruz. İçerisinde test scenario ve test cases yazıyoruz.

Given, When, And ve Then tanımları ile test case’leri yazıyoruz. Feature hangi functionality’yi test edeceğimizi yazdığımız kısım. Scenario ise test senaryomuzun adını yazdığımız kısım.

Test adımlarını bağlaçlar ile bağlamak ve sıralamak gerekiyor. Biz de bu sebeple Given, When, And, Then kullandık.

Feature: Login Functionality  
  
 Scenario: Login with valid username and password  
  
 Given Navigate to Campus  
 When Enter username and password and click login button  
 Then User should login succesfully

steps class içerisinde bu adımlar tanımlı olmadan senaryo çalıştırıldığında hata verecektir. Terminalde bulunan hata metninde case’leri bizim için hazırladığını göreceğiz. Burada yazılı olanları kopyalayarak \_01\_LoginSteps adıyla açtığımız case içerisine yapıştıralım. Import class dediğimizde ve cucumber class’ını import ettiğimizde hata kalkacaktır.

@Given("Navigate to Campus")  
 public void navigateToCampus() {  
  
 }  
  
 @When("Enter username and password and click login button")  
 public void enterUsernameAndPasswordAndClickLoginButton() {  
  
 }  
  
 @Then("User should login succesfully")  
 public void userShouldLoginSuccesfully() {  
  
 }

Şimdi senaryoyu çalıştırdığımızda artık hata vermeden senaryonun çalıştığını terminalde göreceğiz. İçlerine sout ile birer mesaj yazdıralım ve terminalde sonucu görelim.

A screenshot of a computer

Description automatically generated

Senaryo case’lerini tanımlamak için ikinci seçeneğimiz ise feature file içerisinde case’lerin üzerine gelip More Action ve sonrasında ise Create all step definitions seçeneklerini tıklıyoruz. Bize nerede tanımlamamız gerektiğini soracak. Bu alanda yeni class oluşturmasını isteyeceğiz ve yerini tanımlayacağız.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

Pages sayfasının altında DialogContent class tanımlıyoruz ve burada WebElement tanımlamaları ile locator’larını veriyoruz. GWD\_old kısmında ilk oluşturduğumuz driver tanımlama vb. işlemlerimiz mevcut.A screenshot of a computer

Description automatically generated

POM yapısında biz testlerimizin adımlarını yazdığımız bir stepdefinition file tanımlamış olduk, locator vererek WebElementlerimizi page ve dialogcontent class’larımızda tanımladık. Burada locatorları iki veya daha fazla class yapısına bölmemizin mantığı bir uygulama içerisinde ortak olan elementleri belirli class’lar altında toplamak ve ortak olanları her seferinde tekrar tekrar yazmamak. Cucumber dosyalarımızı ise featurefiles klasörü altında topladık ve böylece yönetilebilir bir repository oluşturmuş olduk. BaseDriver olarak daha önce Selenium ve TestNG derslerimizde oluşturduğumuz yapıya bu sefer GWD adını verdik.

İçerisinde bir WebDriver döndüren getDriver() adında bir metot tanımladık. Bu metot bize bir driver oluşturarak döndürecek.

public static WebDriver getDriver() {  
  
 // extend report türkçe bilg çalışmaması sebebiyle eklendi  
 Locale.*setDefault*(new Locale("EN"));  
 System.*setProperty*("user.language", "EN");  
  
 if (*driver*==null) { // ilk kez 1 defa çalışssın  
 *driver* = new ChromeDriver();  
 *driver*.manage().window().maximize();  
 *driver*.manage().timeouts().pageLoadTimeout(Duration.*ofSeconds*(20)); // 20 sn mühlet: sayfayı yükleme mühlet  
 }  
  
 return *driver*;  
}

DialogContent içerisinde bir constructor yapmamız gerekiyor. Driver hazır olana kadar locatorlarda hata vermemesi için bu ayarlamayı yapmamız gerekiyor.

public DialogContent() {  
 PageFactory.*initElements*(GWD.*getDriver*(), this);  
 //PageFactory'ye DialogContent içerisinde tanımladığımız locator'ları kullanmak istediğimizde hazır et  
 //Kullanmadıklarımız için hata verme demiş olduk. Böylece driver.findElement dememize gerek kalmadı  
}

Şimdi elementler hazır olduğunda (visible, clickable vb.) onlara işlem yaptırmamız için wait tanımlamamız gerekiyor.

A screenshot of a computer

Description automatically generated

quitDriver() metodu ayarlayarak testlerimiz sonrasında bu metodun driver’ı kapatmasını sağlıyoruz.

public static void quitDriver() {  
 //test sonucu ekranı bir miktar beklesin diye  
 try {  
 Thread.*sleep*(5000);  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
  
 //driver kapat  
 if (*driver*!=null) { //driver var ise  
 *driver*.quit();  
 *driver*=null;  
 }  
}

2.Gün

Genellikle hooks adı verilen bir klasör içerisinde AfterClass ve BeforeClass olarak tanımlanan işlemler yer alır. Biz de bir hooks klasörü oluşturarak içerisine public void after() metodu ekliyoruz. @After annotation ekleyerek bu metodun senaryodan sonra otomatik olarak çalışmasını sağlamış oluyoruz.

@After// her senaryodan sonra çalışacak bölüm  
public void after(Scenario senaryo) {  
 System.*out*.println("Senaryo Bitti");  
 GWD.*quitDriver*();  
}

İkinci bir düzenleme olarak şimdi wait işlemini yeniden yapılandıracağız. Bunun için yeni bir class oluşturarak içerisine wait işlemlerinin de tanımlı olduğu ve click, sendKeys, assertion işlemlerimizi kendimize göre tekrardan yazdığımız metotlarımızı oluşturacağız. Böylece her stepdefinition class içerisinde tekrar tekrar wait tanımlamamıza gerek kalmayacak.

Öncelikle scroll function oluşturuyoruz sonrasında ise sırasıyla click function ve sendKeys function yazıyoruz.

A screenshot of a computer program

Description automatically generated

verifyContainsTextFunction içerisinde tanımladığımız wait condation textToBePresentInElement bizden hem webelement istiyor ve onun kullanılabilir olduğunu kontrol ediyor hem de içerisinde geçen text istediğimiz koşulu sağlıyor mu diye kontrol ediyor.

Artık oluşturduğumuz yeni metotlarımızı loginsteps class içerisinde kullanabiliriz.

@When("Enter username and password and click login button")  
 public void enterUsernameAndPasswordAndClickLoginButton() {  
 // System.out.println("merhaba 2");  
  
// WebDriverWait wait=new WebDriverWait(GWD.getDriver(), Duration.ofSeconds(30));  
// wait.until(ExpectedConditions.visibilityOf(dc.username));  
// dc.username.sendKeys("turkeyts");  
// dc.password.sendKeys("TechnoStudy123!");  
// dc.loginButton.click();  
  
 dc.sendKeysFunction(dc.username, "turkeyts");  
 dc.sendKeysFunction(dc.password,"TechnoStudy123");  
 dc.clickFunction(dc.loginButton);  
 }  
  
 @Then("User should login succesfully")  
 public void userShouldLoginSuccesfully() {  
  
 dc.verifyContainsTextFunction(dc.txtTechnoStudy,"Techno Study");  
 }

Şimdi ikinci bir senaryo yazarak dersimize devam edelim. Artık login işlemini tekrardan yazmamıza gerek kalmadı, daha önce yazdığımız login case’lerini çağırarak kullanabiliriz.

Feature: Country Functionality  
  
 @SmokeTest  
 Scenario: Create a Country  
 Given Navigate to Campus  
 When Enter username and password and click login button  
 Then User should login succesfully  
 And Navigate to country page  
 When create a country  
 Then Success message should be displayed

Login adımlarından farklı olarak yeni senaryomuzun adımlarını yazıyoruz. Artık bizden bu yeni oluşturduklarımızın kodlarını üretmemizi isteyecek. Yeni bir stepdefinition class oluşturacağız ve onun içerisinde kodları yazarak senaryomuzu gerçekleştireceğiz.

POM yapımızda pages tanımlamıştık. Şimdi bu pages klasörü içerisine çalıştığımız web uygulamasının menülerini tanımlamak için bir header class tanımlayacağız. Eğer sol veya sağ tarafta bir menümüz olsaydı ona da bir class açarak locator’larını burada tanımlayacaktık. Eğer bir footer bölümümüz var ise onun locator’larını bir footer class açarak içerisine locator’ları tanımlayacağız.

3. Gün

Öncelikle slf4j uyarılarımızı kaldırmak için pom.xml dosyamız içerisinde dependency olarak şu iki dependency’yi ekleyeceğiz.

<dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.8.0-beta2</version>  
</dependency>  
  
<dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-simple</artifactId>  
 <version>1.8.0-beta2</version>  
 <scope>test</scope>  
</dependency>

Sonrasında ise Cucumber Report için bir bilgilendirme kutusunu terminalden kaldırmak için bir ekleme daha yapacağız;

Test -> Resource dosyasını bulacağız eğer yoksa oluşturacağız. src/test/resources/cucumber.properties dosyamızın adı cucumber.properties. Bu dosyanın içerisinde cucumber.publish.quiet=true komutunu yazıp kaydettiğimizde artık terminalde bize cucumber report için bir bilgilendirme vermeyecek.

A close up of a website

Description automatically generated A close up of a sign

Description automatically generated

Şimdi birden fazla senaryo yazmayı göreceğiz.

Feature: Country Multi Scenario  
  
 Background: *# before senaryo (yorum satırı diyez işareti ile başlar)* Given Navigate to Campus  
 When Enter username and password and click login button  
 Then User should login successfully  
 And Navigate to country  
  
 @Regression  
 Scenario: Create country  
 When Create a country  
 Then Success message should be displayed  
  
 @Regression  
 Scenario: Create country 2  
 When Create a country name as **"gokhan1"** code as **"12345gk"** Then Success message should be displayed

İki senaryoda da ortak çalışması istenilen kısımlar varsa o case’leri Background: etiketinin altında tanımlayarak her senaryonun öncesinde bu adımların çalışmasını sağlayabiliyoruz.

İkinci senaryoda name ve code alanlarına direkt birer parametreyi test içerisine yollayabiliyoruz. Çift tırnak içerisinde parametreyi tanımlamış oluyoruz ve create step definition dediğimizde bu parametreli olan kısmı otomatik oluşturacak.

@When("Create a country name as {string} code as {string}")  
public void createACountryNameAsCodeAs(String name, String code) {  
 dc.myClick(dc.addButton);  
 dc.mySendKeys(dc.nameInput,name);  
 dc.mySendKeys(dc.codeInput,code);  
 dc.myClick(dc.saveButton);  
}

Yukarıda gördüğümüz gibi içerisine iki adet String parametre alan bir test metodu oluşturdu. Bir senaryoda sürekli veri gönderip senaryodan verdiğimiz parametreler ile senaryonun çalışmasını Scenario Outline ile sağlıyoruz.

Feature: Citizenship Functionality  
  
 Background:  
 Given Navigate to Campus  
 When Enter username and password and click login button  
 Then User should login successfully  
 And Navigate to Citizenship  
  
 Scenario Outline: Create a Citizenship  
 When Create a Citizenship name as **"<name>"** short name as **"<shortname>"** Then Success message should be displayed  
  
 When Create a Citizenship name as **"<name>"** short name as **"<shortname>"** Then Already exist message should be displayed  
  
 Examples:  
 | **name** | **shortname** |  
 | **gek23cs1** | **gk56er1** |  
 | **gek23cs2** | **gk56er2** |  
 | **gek23cs3** | **gk56er3** |  
 | **gek23cs4** | **gk56er4** |  
 | **gek23cs5** | **gk56er5** |

Examles içerisinde verdiğimiz parametreleri nerede kullanacağını anlaması için senaryonun case’i içerisinde <> arasına kolon adını veriyoruz. Sonrasında examples içerisinde bir tablo oluşturarak ilk satırda kolon isimlerini, sonraki satırlarda ise kullanılacak parametreleri veriyoruz. Kaç tane değişkenimiz varsa o kadar kolon ve satır açıyoruz. Açtığımız satır kadar senaryoyu çalıştırıyor. TestNG de DataProvider olarak tanımlı olan Cucumber’da karşımıza Scenario Outline olarak çıkıyor.

@When("Create a Citizenship name as {string} short name as {string}")  
public void createACitizenshipNameAsShortNameAs(String name, String shortname) {  
 dc.myClick(dc.addButton);  
 dc.mySendKeys(dc.nameInput, name);  
 dc.mySendKeys(dc.ShortName, shortname);  
 dc.myClick(dc.saveButton);  
}

Burada da yine create step definition dediğimiz zaman bizim için parametreli bir metot oluşturdu.

4.Gün -> StaleElement hatası eğer bir sayfada elementler çeşitli işlemlere göre güncelleme yapar ancak elementi güncelleme öncesinde bulup güncelleme sonrasında işlem yapmaya kalkarsa oluşur. Element bayatladı hatası demektir ve bu gibi problemlerde çözüm yolumuz element sayısı 0 olana kadar bekle kullanıp ve elementi de progress bar locator’ını vererek progress bar işini yapıp kapandığında diğer elementleri bulmasını sağlayabiliriz.

Biz Scenario Outline ile Example dosyasında verdiğimiz parametreler için testlerin çalışmasını sağladık. Peki biz çok sayıda parametre yollayacaksak nasıl yapacağız. Yan yana sürekli kolon açıp içerisine örnek yazmak daha zor bir kullanım yöntemi. Bu gibi durumlar için kolay bir kullanımımız var. DataTable adını verdiğimiz yöntem ile test case içerisinde string verdiğimiz bir değişkeni direkt test içerisinde kullanabiliyoruz.

Pages içerisinde tanımladığımız elementleri private olarak tanımladığımızda elementlere başka class içerisinde ulaşabilmek için getter ve setter lardan yararlanmamız gerekir. Daha önce TestNG derslerimiz içerisinde Data Provider konusunda biz string değer göndererek test verilerimizi göndermiş ve başarılı bir şekilde kullanmıştık. Şimdi yapacağımız düzenleme sayesinde senaryo yazımı içerisinde string olarak gönderdiğimiz bir webelement’i testlerde kullanabileceğiz.

Test Case yazarken kullanılacak birden fazla parametre için parametre as DataTable dediğimizde alt kısmında aynı Example da olduğu gibi istediğimiz kadar parametre verebiliyoruz. Step Definition class içerisinde ise ilgili metot içerisine değişken olarak DataTable veriyoruz. Sonrasında ise metot içerisinde type casting yaparak list formatına dönüştürmek gerekiyor.

<https://material.angular.io/components/progress-bar/overview> sitesinde örnek proggres bar kodları mevcut. Bu progress bar yapısı StaleElement hatasına yol açıyor. Bu sebeple kodları ona göre düzenlemek gerekebiliyor. Bunun için .numberOfElementsToBe -> verilen locatora sahip elemanların sayısı bu olana kadar bekle wait seçeneğini kullanıp elementin olması gereken sayıyı vererek o sayıya gelene kadar bekleme oluşturabiliriz.

Scenario: Users List  
 When write username **"ismet"**

Eğer biz senaryo içerisinde bir parametre verirsek bu şekilde. iki tırank arasında tanımlamamız gerekiyor. Step definition class içerisinde ise;

@When("write username {string}")  
public void writeUsername(String userName) {  
 System.*out*.println("userName = " + userName);  
}

Bu şekilde nerede bu parametre kullanılacak ve nasıl kullanılacak onu tanımlamış oluyoruz. Eğer iki ayrı parametre yollayacaksak;

And Write username and Password **"ismet"** and **"1234"**

senaryo içerisinde bu şekilde yazıyoruz ve step definition class içerisinde tanımlamasını

@And("Write username and Password {string} and {string}")  
public void writeUsernameAndPasswordAnd(String userName, String password) {  
 System.*out*.println("userName = " + userName);  
 System.*out*.println("password = " + password);  
}

şeklinde yapıyoruz. Eğer biz bir tablo hazırlayıp tek seferde aynı değişkene birden fazla parametre göndererek her parametre için ayrı ayrı testimizi çalıştırmasını istiyorsak;

\_05\_DataTableExample.feature

Feature: Search Item with Data Table  
  
 Background:  
 Given Navigate to Opencart  
 When Enter username and password and click login button  
 Then User should login succesfully  
  
 Scenario: Add new item with parameter  
 When Enter item name as DataTable  
 | **ipod** |  
 | **samsung** |  
 | **macbook** |  
  
 And Add to Cart and Check  
 Then Should be added new item in Shop Cart

public class \_05\_DataTableSteps extends Parent {  
 DialogContent dc = new DialogContent();  
 Header h = new Header();  
  
 @When("Enter item name as DataTable")  
 public void enterItemNameAsDataTable(DataTable dataTable) {  
 List<String> itemList = dataTable.asList(String.class);  
  
 for (String s: itemList  
 ) {  
 System.*out*.println("s = " + s);  
 sendKeysFunction(h.searchInput,s);  
 clickFunction(h.searchButton);  
 clickFunction(dc.item1);  
 clickFunction(h.addCart);  
 clickFunction(h.viewCart);  
 verifyContainsTextFunction(dc.assertMessage, "Gift");  
  
 }  
  
 }  
}

senaryo içerisinde bu şekilde tanımlıyoruz. Step Definition class içerisinde ise;

@And("Write username as DataTable")  
public void writeUsernameAsDataTable(DataTable userlar) {  
 List<String> listUserlar= userlar.asList(String.class); // liste  
  
 for (String user : listUserlar)  
 System.*out*.println("user = " + user);  
}

Şeklinde bir tip dönüşümü yaparak tabloyu String alan bir listeye çeviriyoruz. Eğer bunda da iki ayrı parametre tipi yollamak istiyorsak;

\_06\_DataTableMulty.feature

Feature: Multy Data Table  
  
 Scenario: DataTable multy items  
 And Send this items with Data Table  
 | **iphone** | **15 pro** |  
 | **macbook** | **13" pro** |  
 | **ipad** | **12.9" pro** |

senaryo içerisinde bu şekilde bir tablo ile yan yana kolon açarak veriyoruz. Step definition class içerisinde tanımlarken bu sefer asLists olarak tanımlayacağız ve iki boyutlu bir liste oluşturacağız.

@And("Write username and Password as DataTable")  
public void writeUsernameAndPasswordAsDataTable(DataTable userlarVeSifreler) {  
 List< List<String> > userSifrList= userlarVeSifreler.asLists(String.class);  
  
 for (int i = 0; i < userSifrList.size(); i++) {  
 System.*out*.println(userSifrList.get(i).get(0)+" "+userSifrList.get(i).get(1));  
 }  
  
}

ikiden fazla parametre için liste yapısı aynı şekilde devam ediyor olacak.

Eğer biz butonları da senaryo içerisinde yazar ve DataTable olarak buton isimlerini gönderirsek;

Scenario: Fee Functionality  
 And Click on the element in LeftNav  
 | **setup** |  
 | **parameters** |  
 | **fees**

Step Definition class içerisinde de Data Table tip dönüşümünü yaparsak;

@And("Click on the element in LeftNav")  
public void clickOnTheElementInLeftNav(DataTable linkler) {  
 List<String> strlinkList = linkler.asList(String.class);  
  
 for (int i = 0; i < strlinkList.size(); i++) {  
 WebElement linkWebElement = ln.getWebElement(strlinkList.get(i));  
 ln.myClick(linkWebElement);  
 }  
}

Pages class’ları içerisinde bu butonlara birer switch-case yapısı ile tıklama işlemlerini düzenleyerek, senaryoda verdiğimiz butonların otomatik olarak tıklanmasını sağlayabiliriz.

Böylece otomasyon senaryolarımızda tıklama işlemlerimizi de yazdığımız senaryo içerisinde yollayabiliriz.

public WebElement getWebElement(String strElement){  
 switch (strElement){  
 case "setup": return this.setup;  
 case "parameters": return this.parameters;  
 case "countries": return this.countries;  
 case "citizenShip": return this.citizenShip;  
 case "nationalities": return this.nationalities;  
 case "fees": return this.fees;  
 case "entranceExamsOne": return this.entranceExamsOne;  
 case "setupTwo": return this.setupTwo;  
 case "entranceExamsTwo": return this.entranceExamsTwo;  
 }  
  
 return null;  
}

Şimdi iki parametreli bir yapı oluşturalım. Birinci parametrede WebElement’i göndereceğiz, ikinci parametede ise içerisine gönderilecek text’i vereceğiz. Bunun için yine DataTable yapısından faydalanıyoruz.

And User sending the keys in Dialog  
 | **nameInput** | **ige1Fee** |  
 | **codeInput** | **i2323** |  
 | **integrationCode** | **121** |  
 | **priorityCode** | **2323** |

Senaryomuz içerisinde bu şekilde tanımladık ve step definitions içerisinde ise;

@And("User sending the keys in Dialog")  
public void userSendingTheKeysInDialog(DataTable dt) {  
 List< List<String> > items= dt.asLists(String.class);  
  
 for (int i = 0; i < items.size(); i++) {  
 WebElement e=dc.getWebElement(items.get(i).get(0)); // 0. webelement  
 String yazi = items.get(i).get(1);//1. yazısı  
  
 dc.mySendKeys(e, yazi);  
 }  
}

şeklinde metodumuzu oluşturuyoruz. Burada daha önce oluşturduğumuz getWebElement metodunu çağırarak senaryo içerisinde verdiğimiz items.get(i).get(0) elementini bu metot içerisine yollamış oluyoruz. Senaryoda ilgili yerden elementin adını alıyor ve dialog content içerisinde tanımladığımız elementi switch-case den çağırarak kullanıyor. Sonrasında ise daha önce Parent class içerisinde tanımladığımız sendKeysFunction içerisine bu sefer senaryoda verdiğimiz ikinci parametreyi gönderiyoruz. Bu işi kolaylaştırmak için bir String oluşturduk ve o String içerisine değeri senaryo içerisinde gönderdiğimiz parametreyi verdik. Sonra bu String ifadeyi sendKeysFunction içerisine yolladık ve artık sadece senaryoda parametre yazarak senaryolarımızı çalıştırır hale geldik.

5 Gün

Scenario Outline olarak yeni bir senaryo tanımladığımızda bizden Examples içerisinde parametre vermemizi istiyor. Examples içerisinde önce ilk satırda başlıkları veriyoruz. Sonrasındaki satırlarda ise parametrelerimizi dolduruyoruz. Bu parametreleri senaryo içerisinde nerede kullanacaksak o kısımlarda da aynı kolon isimlerini kullanarak yerlerini işaretlememiz gerekiyor.

Scenario Outline: Fee Functionality

And User sending the keys in Dialog Content  
 | **nameInput** | **<name>** |  
 | **codeInput** | **<code>** |  
 | **integrationCode** | **<intCode>** |  
 | **priorityCode** | **<priority>** |

Examples:  
 | **name** | **code** | **intCode** | **priority** |  
 | **gekFee1** | **2222** | **payPal** | **12234** |  
 | **gekFee2** | **2221** | **cash** | **12235** |  
 | **gekFee3** | **2223** | **cheque** | **12236** |  
 | **gekFee4** | **2224** | **applePay** | **12237** |  
 | **gekFee5** | **2226** | **crypto** | **12238** |

Scenario Outline bize şu kolaylığı sağlıyor, biz senaryoda ne işlem yapılacaksa test case’leri yazıyoruz ve examples tablomuzda verdiğimiz bütün parametreler için senaryo en baştan sona koşuluyor ve her parametre için sonuç raporu üretiliyor. Biz böylelikle farklı parametrelerde aynı senaryonun tekrar tekrar koşulmasını sağlamış oluyoruz. 29.03.23 part1 den devam.

Feature: Fees multiple values functionality  
  
 Background:  
 Given Navigate to Campus  
 When Enter username and password and click login button  
 Then User should login succesfully  
  
 Scenario Outline: Fee Functionality  
 And Click on the element in LeftNav  
 | **setup** |  
 | **parameters** |  
 | **fees** |  
  
 And Click on the element in Dialog  
 | **addButton** |  
  
 And User sending the keys in Dialog Content  
 | **nameInput** | **<name>** |  
 | **codeInput** | **<code>** |  
 | **integrationCode** | **<intCode>** |  
 | **priorityCode** | **<priority>** |  
  
 And Click on the element in Dialog  
 | **toggleBar** |  
 | **saveButton** |  
  
 Then Success message should be displayed  
  
 And User delete item from Dialog Content  
 | **<name>** |  
  
 Then Success message should be displayed  
  
 Examples:  
 | **name** | **code** | **intCode** | **priority** |  
 | **ismFee1** | **2222** | **payPal** | **12234** |  
 | **ismFee2** | **2221** | **cash** | **12235** |  
 | **ismFee3** | **2223** | **cheque** | **12236** |  
 | **ismFee4** | **2224** | **applePay** | **12237** |  
 | **ismFee5** | **2226** | **crypto** | **12238** |

package StepDefinitions;  
  
import Pages.DialogContent;  
import Pages.LeftNav;  
import io.cucumber.datatable.DataTable;  
import io.cucumber.java.en.And;  
import org.openqa.selenium.WebElement;  
  
import java.util.List;  
  
public class \_06\_DataTableSteps {  
 LeftNav ln=new LeftNav();  
 DialogContent dc=new DialogContent();  
 @And("Click on the element in LeftNav")  
 public void clickOnTheElementInLeftNav(DataTable items) {  
 List<String> strButtons=items.asList(String.class);  
  
 for (String strButton: strButtons) {  
 WebElement element=ln.getWebElement(strButton); // webelemnti bul  
 ln.clickFunction(element); // webelemente tıklat  
 }  
 }  
 @And("Click on the element in Dialog")  
 public void clickOnTheElementInDialog(DataTable dt) {  
 List<String> strButtons=dt.asList(String.class);  
  
 for (String strButton: strButtons) {  
 WebElement element=dc.getWebElement(strButton); // webelemnti bul  
 dc.clickFunction(element); // webelemente tıklat  
 }  
 }  
 @And("User sending the keys in Dialog Content")  
 public void userSendingTheKeysInDialogContent(DataTable dt) {  
 List< List<String> > items= dt.asLists(String.class);  
  
 for (int i = 0; i < items.size(); i++) {  
 WebElement element= dc.getWebElement(items.get(i).get(0));  
 dc.sendKeysFunction(element, items.get(i).get(1));  
 }  
 }  
 @And("User delete item from Dialog Content")  
 public void userDeleteItemFromDialogContent(DataTable dt) {  
 List<String> strButtons=dt.asList(String.class);  
  
 for (String strDeleteText: strButtons) {  
 dc.deleteItem(strDeleteText);  
 }  
 }  
}

*#Senaryo :  
# Bir önceki Exam Giriş Senaryosunu  
# Aşağıdaki değerler için çalıştırınız.  
#  
#| ExamName | AcademicPeriodOption | GradeLevelOption |  
#| Math exam is1 | academicPeriod1 | gradeLevel2 |  
#| IT exam is1 | academicPeriod1 | gradeLevel3 |  
#| Oracle exam is1 | academicPeriod1 | gradeLevel4 |  
#| Math exam is2 | academicPeriod1 | gradeLevel5 |*Feature: ExamEnter functionality  
  
 Background:  
 Given Navigate to Campus  
 When Enter username and password and click login button  
 Then User should login succesfully  
  
 Scenario Outline: Exam Create and Delete  
 And Click on the element in LeftNav  
 | **entranceExamsOne** |  
 | **setupTwo** |  
 | **entranceExamsTwo** |  
  
 And Click on the element in Dialog  
 | **addButton** |  
  
 And User sending the keys in Dialog Content  
 | **<ExamName>** | **<AcademicPeriodOption>** |  
  
 And Click on the element in Dialog  
 | **academicPeriod** |  
 | **<AcademicPeriodOption>** |  
 | **gradeLevel** |  
 | **<GradeLevelOption>** |  
  
 And Click on the element in Dialog  
 | **saveButton** |  
  
 Then Success message should be displayed  
  
 And Click on the element in LeftNav  
 | **entranceExamsOne** |  
 | **setupTwo** |  
 | **entranceExamsTwo** |  
  
 And User delete item from Dialog Content  
 | **<ExamName>** |  
  
 Then Success message should be displayed  
  
 Examples:  
 | **ExamName** | **AcademicPeriodOption** | **GradeLevelOption** |  
 | **Math exam is1** | **academicPeriod1** | **gradeLevel2** |  
 | **IT exam is1** | **academicPeriod1** | **gradeLevel3** |  
 | **Oracle exam is1** | **academicPeriod1** | **gradeLevel4** |  
 | **Math exam is2** | **academicPeriod1** | **gradeLevel5** |

package Pages;  
  
import Utilities.GWD;  
import org.openqa.selenium.By;  
import org.openqa.selenium.JavascriptExecutor;  
import org.openqa.selenium.WebElement;  
import org.openqa.selenium.support.FindBy;  
import org.openqa.selenium.support.PageFactory;  
import org.openqa.selenium.support.ui.ExpectedConditions;  
import org.openqa.selenium.support.ui.WebDriverWait;  
  
import java.time.Duration;  
import java.util.List;  
  
// POM : Page Object Model  
public class DialogContent extends Parent{  
 // kullanıcı ile dialogla veri alan bölümler  
 public DialogContent() {  
 PageFactory.*initElements*(GWD.*getDriver*(),this);  
 }  
 @FindBy(css = "input[formcontrolname='username']")  
 public WebElement username;  
 @FindBy(css="input[formcontrolname='password']")  
 public WebElement password;  
 @FindBy(css="button[aria-label='LOGIN']")  
 public WebElement loginButton;  
 @FindBy(css="span[class='mat-tooltip-trigger logo-text']")  
 public WebElement txtTechnoStudy;  
 @FindBy(xpath="//ms-add-button[contains(@tooltip,'ADD')]//button")  
 public WebElement addButton;  
 @FindBy(xpath="//ms-text-field[@formcontrolname='name']/input")  
 public WebElement nameInput;  
 @FindBy(xpath="//ms-text-field[@formcontrolname='code']/input")  
 public WebElement codeInput;  
 @FindBy(xpath="//ms-save-button/button")  
 public WebElement saveButton;  
 @FindBy(xpath="//div[contains(text(),'successfully')]")  
 public WebElement successMessage;  
 @FindBy(xpath="//ms-text-field[@formcontrolname='shortName']/input")  
 public WebElement shortName;  
 @FindBy(xpath="//div[contains(text(),'already exists')]")  
 public WebElement alreadyExist;  
 @FindBy(xpath="//mat-form-field//input[@data-placeholder='Name']")  
 public WebElement searchInput;  
 @FindBy(xpath="//ms-search-button//button")  
 public WebElement searchButton;  
 @FindBy(xpath="(//ms-delete-button//button)[1]")  
 public WebElement deleteImageBtn;  
 @FindBy(xpath="//button[@type='submit']")  
 public WebElement deleteDialogBtn;  
 @FindBy(xpath="//ms-text-field[@formcontrolname='budgetAccountIntegrationCode']//input")  
 private WebElement integrationCode;  
 @FindBy(xpath="//ms-integer-field[@formcontrolname='priority']/input")  
 private WebElement priorityCode;  
 @FindBy(xpath="//mat-slide-toggle[@formcontrolname='active']")  
 private WebElement toggleBar;  
 @FindBy(xpath="//mat-select//span[text()='Academic Period']")  
 private WebElement academicPeriod;  
 @FindBy(xpath="(//mat-option/span)[1]")  
 private WebElement academicPeriod1;  
 @FindBy(xpath="(//span[text()='Grade Level'])[1]")  
 private WebElement gradeLevel;  
 @FindBy(xpath="(//mat-option//span)[2]")  
 private WebElement gradeLevel2;  
 @FindBy(xpath="//tr[contains(@class, 'mat-row')]/td[2]")  
 public List<WebElement> nameList;  
 public WebElement getWebElement(String strButton){  
  
 switch (strButton)  
 {  
 case "addButton" : return addButton;  
 case "saveButton" : return saveButton;  
 case "nameInput" : return nameInput;  
 case "codeInput" : return codeInput;  
 case "integrationCode" : return integrationCode;  
 case "priorityCode" : return priorityCode;  
 case "toggleBar" : return toggleBar;  
 case "academicPeriod" : return academicPeriod;  
 case "academicPeriod1" : return academicPeriod1;  
 case "gradeLevel" : return gradeLevel;  
 case "gradeLevel2" : return gradeLevel2;  
 }  
 return null;  
 }  
 public void deleteItem(String searchText){  
 sendKeysFunction(searchInput,searchText);  
 clickFunction(searchButton); //fuse-progress-bar/\* gözüküyor  
 //beklet  
 //1. StaleElemetn hatası verdi : erken buldum tez kaybettim  
 //wait.until(ExpectedConditions.elementToBeClickable(searchButton));  
  
 //fuse-progress-bar/\* bu 0 olana kadar beklet  
 wait.until(ExpectedConditions.*numberOfElementsToBe*(By.*xpath*("//fuse-progress-bar/\*") , 0));  
  
 clickFunction(deleteImageBtn);  
 clickFunction(deleteDialogBtn);  
 }  
}