**Bir framework olusturmak icin gerceklestirilmesi gereken adımlar:**

**1- “pages”, “tests”, “utilities” package'larını olustur.**

-------------------------------------------------------------------------------------------------------------------------------------------------------

**2-“configuration.properties” adinda bir file olustur, icini test yaparken kullanacagin gerekliliklerle doldur.** (Key = Value)

örn: Url=https://www.amazon.com/

-------------------------------------------------------------------------------------------------------------------------------------------------------

**3- utilities package'i altina ConfigurationReader adında bir class olustur:**

a- Class level'da "Properties properties;" objesi olustur.( static Properties *properties*;)

b- Static block olustur:( static {)

1- "String filePath" variable'i olustur ve buna configuration.properties file'inin path'ini assign et.   
(String dosyaYolu="configiration.properties";)

2- FileInputStream fileInputStream=new FileInputStream(dosyaYolu);

objesi olustur ve arguman olarak "filePath"i gir. Try-catch blogu icerisine al.

3- Try-catch icersinde, "fileInputStream" altinda " *properties*=new Properties();" assignment'ini gerceklestir.

4- "property = new Properties()" altinda property objesi uzerinden load() methodunu call et ve parametre olarak fileInputStream gir. (*properties*.load(fileInputStream);)

properties.load(fileInputStream). Bunun icin de bir catch olusturulmasi istenecek, bu durumun exception kapsami daha genis oldugundan

best-practice icin bir onceki catch sarti silinebilir. Silinmese de sorun olmayacaktir. (Parent-Child)

} catch (FileNotFoundException e) {  
 e.printStackTrace();  
} catch (IOException e) {  
 e.printStackTrace();

c- Static block'dan sonra String return type'inda getProperty() adinda parametresi (String key) olan bir method olustur: (public static String getProperty(String key){)

1- Method icerisinde "String value" adinda bir variable olustur ve buna properties.get(key)'i assign et.

(String value=*properties*.getProperty(key); )

2- value variable'ini return et. (return value;)

-------------------------------------------------------------------------------------------------------------------------------------------------------

**4- utilities package'i altina Driver adinda bir class olustur:**

(Not\_driver class'indan obje uretilemesin istedigimiz icin driver class'i SINGLETON yapiyoruz = private Driver(){})

**a-** Class level'da "private static WebDriver *driver*;" objesi olustur.

**b-** "public static WebDriver getDriver(){ " methodu olustur:

Not\_ if'i bu method her calistiginda yeni bir driver olusturmamasi icin kullaniyorruz. if driver'i kontrol edecek eger bir deger atamasi yapildiysa yeni bir driver olusturmayacak if (*driver*==null){

1- Method'un testlerin gerceklesecegi browser'lar konusunda dinamik olabilmesi icin bir switch case olustur

2- switch parantezi icerisine "switch (ConfigReader.*getProperty*("browser")) {" gir. ConfigurationReader class'i bizim icin bir kopru gorevi gorerek

configuration.properties file'inda browser key-value ikilisinde yapilacak herhangi bir degisiklige tum framework'umuzun adepte olmasini saglayacak.

Not\_ case'i driver'i istedigimiz browser'da calistirmak icin kullaniyoruz. Configuration.properties dosyasinda browser olarak ne yazdiksa tum testlerimiz o browser'da calisacak . Browser secimi yapilmadiysa default olarak chrome devreye girecek

3- Her case'de kullanmak istedigin browserlari olusturup, driver objesine assign et.

case "chrome":  
WebDriverManager.*chromedriver*().setup();  
*driver* = new ChromeDriver();  
break;

4- Default case'de her ihtimale karsi kullanmak istedigin bir browder'i olusturabilirsin. Boylece test kesintiye ugramadan calisabilir.

default:  
WebDriverManager.*chromedriver*().setup();  
*driver* = new ChromeDriver();

5- Switch case aldinda her test'te klasik olarak kullanilan ayarlamalari gerceklestir:

driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);

driver.manage().window().maximize();

6- Su ana kadar yaptiklarini bir if body'si icersine al ve if'in sartini driver==null olarak belirle. Boylelikle test method'larindan getDriver() methodu her call edildiginde, driver oncesinde kapatilmamis ise tekrar tekrar obje olusturmanin onune gecip code'un performanasini arttirmis olacagiz.

7- if body'sinden sonra driver objesini return et. return *driver*;

**c-** "public static void closeDriver(){" methodu olustur:

1- driver objesi uzerinden quit() methodunu call et "driver.quit();" Bunu if body'si icerisine al ve if'in sartini driver!=null olarak belirle.

if (*driver*!=null){  
*driver*.quit();  
}

Bunu yaparak driver'in null oldugu durumlarda kodun NullPointerException atmasinin onune gecmis oluyoruz. (?)

2- if body'sinden sonra driver'i tekrar null olarak assing et. "driver = null;" Bunu yapmamizin sebebi, driver'i farkli bir method icerisinden tekrar cagirdigimizda, driver oncesinde kapatilmissa degerinin yeniden atanmasini istememiz. Aksi durumda configuration.properties file'inda kullanilmasini istedigimiz browser'i degistirsek bile testlerimiz surekli olarak ayni browser uzerinden gerceklesmeye devam eder. Driver bir kez kapatildi mi, degerinin sifirlanmasi gerek. *driver*=null;

------------------------------------------------------------------------------------------------------------------------------------------------------

**5- pages package'i altinda testlerin gerceklestirilecegi web sayfasinin adinda bir class olustur:**

a- Bu class'da yapilmasi gerek ilk sey access modifier'i public olan bir constructor olusturmak:

public WebDriver driver;

1- Constructor icerisinde PageFactory classi uzerinden initElements methodunu call edip ilk arguman olarak driver, ikinci arguman olarak this keyword'unu gir.

public QAConcortPage() {PageFactory.*initElements*(Driver.*getDriver*(), this);}

boş construction üretelim public void ConcortHotelLogin(){  
Driver.*getDriver*().get(ConfigReader.*getProperty*("CHQAUrl"));  
QAConcortPage qaConcortPage=new QAConcortPage();qaConcortPage.ilkLoginLinki.click();  
qaConcortPage.usernameKutusu.sendKeys(ConfigReader.*getProperty*("CHQAValidUsername"));  
qaConcortPage.passwordKutusu.sendKeys(ConfigReader.*getProperty*("CHQAValidPassword"));  
qaConcortPage.loginButonu.click();}

2- Ardindan class level'da, @Find annotation'i ile testimizi gerceklestirecegimiz sayfada siklikla kullanacagimiz elementleri locate et.

Orn: @FindBy(linkText = "Log in")  
public WebElement ilkLoginLinki;

3- Testleri gerceklestirme sirasinda, login gibi sayfaya her girildiginde tekrar tekrar yapilacak islemler varsa bunlar da ilgili page class'i icerisinde bir method olusturularak gerceklestirilebilir.

------------------------------------------------------------------------------------------------------------------------------------------------------

**6-** Son olarak test class'imizi olusturup @Test annotation'i altinda birbirinden bagimsiz test method'larimizi calistirabiliriz.

a- Bunu gerceklestirmek icin her method icerisinde Driver class'i uzerinden getDriver() method'unu cagirip arguman olarak, kopru gorevi goren ConfigurationReader class'i uzerinden, getProperty methodunu call edip buna da arguman olarak configuration.properties file'inda bulunan ilgili testi gerceklestirecegimiz web sitesinin Url'inin key'ini gireriz.

Orn : Driver.*getDriver*().get(ConfigReader.*getProperty*("CHQAUrl"));

b- Driver islemini gerceklestirdikten sonra ilgili website icin pages package'i altinda olustudugumuz class'dan bir obje uretir, bu obje uzerinden

@FindBy annotation'i ile locate ettigimiz elementlere ulasarak test islemlerimizi gerceklestiririz.

Örn: QAConcortPage qaConcortPage=new QAConcortPage();

c- Her method'dan sonra Driver.closeDriver() methodunu call etmek test method'larimiz bagimsiz calismasini saglayacaktir.

Driver.*closeDriver*();