**Sahte veriler oluşturmak için Java Faker kitaplığı.**

**1. Genel Bakış**

JavaFaker , adreslerden popüler kültür referanslarına kadar çok çeşitli gerçek görünümlü veriler oluşturmak için kullanılabilen bir kütüphanedir.

**2. Bağımlılıklar**

JavaFaker'ı kullanmaya başlamak için ihtiyaç duyacağımız tek bağımlılık aşağıdadır.

* İlk olarak, Maven tabanlı projeler için ihtiyaç duyacağımız bağımlılık:

<dependency>  
 <groupId>com.github.javafaker</groupId>  
 <artifactId>javafaker</artifactId>  
 <version>0.15</version>  
</dependency>

* Gradle kullanıcıları için build.gradle dosyanıza aşağıdakileri ekleyebilirsiniz:

compile group: 'com.github.javafaker', name: 'javafaker', version: '0.15'

* Rastgele verileri kullanmaya başlayın, aşağıdan oluşturun:

Faker feker = yeni Faker();

Farklı yerel ayarlarla oluşturmak istiyorsanız:

Faker fak = new Faker(new Locale("YOUR\_LOCALE"));   
  
Faker fek= new Faker(new Locale("en-US"));

Bu kitaplık, 35'ten fazla farklı alana erişmenizi sağlar ve neredeyse her gerçek yaşam kullanım durumu için veri sağlar. Aşağıdaki gibi alanlar:

* Finance
* Food
* Books
* Name
* Address
* Business
* ChuckNorris
* HarryPotter
* LordOfTheRings

Desteklediği tüm kullanılabilir alan ve yerel ayarların listesini kontrol etmeniz gerekir.

**3. SahteDeğer Hizmeti:**

FakeValueService sınıfı, yerel ayar ile ilişkili .yml dosyalarını çözmenin yanı sıra rastgele diziler oluşturmaya yönelik yöntemler sağlar.

* Bir cep telefonu numarasından, adreslerden, isimlerden popüler kültür referanslarına kadar çok çeşitli gerçek görünümlü veriler oluşturmak için kullanılabilen bir kütüphanedir. Bu, bazı yer tutucuları kullanmak istediğimizde ancak gerçek verilere sahip olmadığımızda gerçekten yararlıdır. Kredi kartı verilerini oluşturmak ve buna karşı doğrulama yapmak istediğiniz gibi. Faker kütüphanesi bunu oluşturmanıza yardımcı olacaktır.

**3.1. Letterify , Numerify ve Bothify**

Üç yararlı yöntem şunlardır: Letterify , Numberify ve Bothify . Letterify , rastgele alfabetik karakter dizileri oluşturmaya yardımcı olur.

Ardından, Numerify yalnızca sayısal diziler oluşturur.

Son olarak, Bothify ikisinin birleşimidir ve rastgele alfasayısal diziler oluşturabilir- kimlik dizeleri gibi şeylerle alay etmek için kullanışlıdır.

FakeValueService , RandomService'in yanı sıra geçerli bir Yerel Ayar gerektirir:

@Test

public void whenBothifyCalled\_checkPatternMatches() throws Exception {

FakeValuesService fakeValuesService = new FakeValuesService(

new Locale("en-GB"), new RandomService());

String email = [fakeValuesService.bothify("????##@gmail.com](mailto:fakeValuesService.bothify("????##@gmail.com)");

Matcher emailMatcher = [Pattern.compile("\\w{4}\\d{2}@gmail.com").matcher(email](mailto:Pattern.compile("\\w{4}\\d{2}@gmail.com").matcher(email));

assertTrue(emailMatcher.find());

}

Bu birim testinde, yerel ayarı en-GB olan yeni bir FakeValueService oluşturuyoruz ve benzersiz bir sahte Gmail adresi oluşturmak için Bothify yöntemini kullanıyoruz.

' ?' ile değiştirerek çalışır. Rastgele harflerle ve '#' ile rastgele sayılarla. Daha sonra basit bir Matcher kontrolü ile çıktının doğruluğunu kontrol edebiliriz.

**3.2. normal ifade**

Benzer şekilde, regexify , seçilen bir regex modeline dayalı olarak rastgele bir dizi oluşturur.

Bu snippet'te, belirli bir normal ifadeyi izleyerek rastgele bir dizi oluşturmak için FakeValueService'i kullanacağız:

@Test

public void givenValidService\_whenRegexifyCalled\_checkPattern() throws Exception {

FakeValuesService fakeValuesService = new FakeValuesService(

new Locale("en-GB"), new RandomService());

String alphaNumericString = fakeValuesService.regexify("[a-z1-9]{10}");

Matcher alphaNumericMatcher = Pattern.compile("[a-z1-9]{10}").matcher(alphaNumericString);

assertTrue(alphaNumericMatcher.find());

}

Kodumuz, 10 uzunluğunda küçük harfli alfa sayısal bir dize oluşturur. Modelimiz, oluşturulan dizeyi normal ifadeye göre kontrol eder.

**4. JavaFaker'ın Faker Sınıfı**

Faker sınıfı, JavaFaker'ın sahte veri sınıflarını kullanmamıza izin verir.

Bu bölümde, bir Faker nesnesinin nasıl başlatılacağını ve bazı sahte verileri çağırmak için nasıl kullanılacağını göreceğiz:

Faker faker = new Faker();

String streetName = faker.address().streetName();

String number = faker.address().buildingNumber();

String city = faker.address().city();

String country = faker.address().country();

System.out.println(String.format("%s\n%s\n%s\n%s", number,

streetName,

city,

country));

Yukarıda, rastgele bir adres oluşturmak için Faker Adres nesnesini kullanıyoruz.

Bu kodu çalıştırdığımızda çıktının bir örneğini alacağız:

3188

Dayna

Mountains

New Granvilleborough

Tonga

Bir yerel ayar belirtmediğimiz için verilerin tek bir coğrafi konumu olmadığını görebiliriz.

Bu sahte nesneyi, aşağıdakiler gibi daha birçok nesneyle ilgili veriler oluşturmak için benzer şekilde kullanabiliriz:

* Business
* Beer
* Food
* PhoneNumber

**Örnek:**

Bir Faker sınıfı örneği oluşturalım ve bazı sahte verileri getirmeye çalışalım.

Faker faker = **new** Faker();

String firstName=faker.name().firstName();

String lastName=faker.name().lastName();

String title=faker.name().title();

String suffix=faker.name().suffix();

String address=faker.address().streetAddress();

String city=faker.address().cityName();

String country=faker.address().country();

Yukarıdaki örnekte, rastgele bir ad, soyadı ve bazı rastgele adresler oluşturuyoruz.

**Başka bir örnekte kullanalım:**

Faker feku = new Faker(new Locale("de"));

// generate some finance data

@GetMapping("/cardinfo")

public JsonNode getRandomCardInfo()

{Faker faker = new Faker(new Locale("en-US"));

ArrayNode cards = objectMapper.createArrayNode();

for (int i = 0; i < 10; i++) {

cards.add(objectMapper.createObjectNode()

.put("CreditCard", faker.finance().creditCard())

.put("Bic", faker.finance().bic())

.put("Iban", faker.finance().iban())

}

return cards;

}

Benzer bir notta, gıda tedariki ile ilgili bir proje üzerinde çalıştığınızı ve malzemelerle rastgele gıda üretmek istediğinizi varsayalım.

@GetMapping("/foods")

public JsonNode getRandomFoods() {

Faker faker = new Faker(new Locale("en-AU"));

ArrayNode foods = objectMapper.createArrayNode();

for (int i = 0; i < 20; i++) {

foods.add(objectMapper.createObjectNode()

.put("ingredients" , faker.food().ingredient())

.put("spices", faker.food().spice())

.put("measurements", faker.food().measurement()));

}

return foods;

}

**5. Yerel Ayarların Tanıtımı**

Burada, oluşturulan verileri tek bir konuma daha özgü hale getirmek için yerel ayarların nasıl kullanılacağını tanıtacağız. ABD yerel ayarı ve İngiltere yerel ayarı olan bir Faker'ı tanıtacağız:

Faker ukFaker = new Faker(new Locale("en-GB"));  
 Faker usFaker = new Faker(new Locale("en-US"));  
  
 System.out.println(String.format("American zipcode: %s", usFaker.address().zipCode()));  
 System.out.println(String.format("British postcode: %s", ukFaker.address().zipCode()));  
  
 Pattern ukPattern = Pattern.compile(  
 "([Gg][Ii][Rr] 0[Aa]{2})|((([A-Za-z][0-9]{1,2})|"  
 + "(([A-Za-z][A-Ha-hJ-Yj-y][0-9]{1,2})|(([A-Za-z][0-9][A-Za-z])|([A-Za-z][A-Ha-hJ-Yj-y]"  
 + "[0-9]?[A-Za-z]))))\\s?[0-9][A-Za-z]{2})");  
 Matcher ukMatcher = ukPattern.matcher(ukFaker.address().zipCode());  
  
 assertTrue(ukMatcher.find());  
  
 Matcher usMatcher = Pattern.compile("^\\d{5}(?:[-\\s]\\d{4})?$")  
 .matcher(usFaker.address().zipCode());  
  
 assertTrue(usMatcher.find());  
}

Yukarıda, yerel ayara sahip iki Faker'ın ülkelerin posta kodları için normal ifadeleriyle eşleştiğini görüyoruz.

Faker'a geçirilen yerel ayar yoksa, Faker bir LocaleDoesNotExistException oluşturur.

Bunu aşağıdaki birim testi ile test edeceğiz:

@Test(expected = LocaleDoesNotExistException.class)  
public void givenWrongLocale\_whenFakerInitialised\_testExceptionThrown() {  
 Faker wrongLocaleFaker = new Faker(new Locale("en-seaWorld"));  
}

**6. Benzersizlik**

JavaFaker görünüşte Random'da veri üretirken, benzersizliği garanti edilemez.

JavaFaker , tekrarlanan yöntem çağrılarının deterministik çıktısını sağlamak için sözde rastgele sayı üretecinin (PRNG) bir RandomService biçiminde tohumlanmasını destekler.

Basitçe söylemek gerekirse, sözde rastgelelik rastgele görünen ancak olmayan bir süreçtir.

Aynı tohumla iki Faker oluşturarak bunun nasıl çalıştığını görebiliriz:

@Test  
public void givenJavaFakersWithSameSeed\_whenNameCalled\_CheckSameName() {  
  
 Faker faker1 = new Faker(new Random(24));  
 Faker faker2 = new Faker(new Random(24));  
  
 assertEquals(faker1.name().firstName(), faker2.name().firstName());  
}

Yukarıdaki kod, iki farklı fakerdan aynı adı döndürür.

**Faker Kütüphanesi:**

Faker class’ı testlerimizi yaparken ihtiyaç duyduğumuz isim, soyisim, adres vb. bilgiler için fake değerler üretmemize imkan tanır.

Faker değerler üretmek için Faker class’ından bir obje üretir ve var olan method’ları kullanırız.

// faker class'ı kullanmak için mvn repository.com adresinden java faker kütüphanesinin aratır,  
// ve en cok kullanılanını pom.xml dosyamıza ekleriz,  
// ve faker class'ından bir obje oluşturup email için faker.internet() methodunu kullanarak  
// emailAdress() methodunu seçeriz  
// password içinde ayni internet methodunu kullanırız  
// isim ve soyisimler için faker.name() methodu ile firstname() ve lastname() methodunu kullanırız  
// faker claas'ını kullanma amacımız form dordurmamız gereken sitelerde defalarca kendi üreteceğimiz  
// veriler yerine bizim için random veriler üretir ve isimizi kolaylastırır.

Faker class'ını kullanma amacımız form doldurmamız gereken web sitelerinde defalarca kendi üreteceğimiz veriler yerine bizim için random veriler üretir ve işimizi kolaylaştırır.

Faker class'ını kullanmak için mvnrespository.com adresinden java faker kütüphanesini aratır ve en çok kullanılanı  
pom.xml dosyamıza (dependencies tagları arasına) ekleriz. Sonra faker class'ından bir obje oluşturup;  
  
email için: faker.internet() methodunu kullanarak emailAddress() method'unu seçeriz.  
password: faker.internet().password()  
isim: faker.name().firstName()  
soyisim: faker.name().lastName()  
  
method'larını kullanırız.

**Örneğin:**

@Test  
public void test1() {  
 //1- https://www.facebook.com adresine gidelim  
 driver.get("https://tr-tr.facebook.com/");  
  
 // 2- Yeni hesap olustur butonuna basalim  
 driver.findElement(By.xpath("//\*[text()='Yeni Hesap Oluştur']")).click();  
  
 // “firstName” giris kutusuna bir isim yazin  
 Faker faker=new Faker(); // FAKER OBJESİ OLUSTURUYORUZ  
 String fakeMail = faker.internet().emailAddress();  
 WebElement isim = driver.findElement(By.xpath("//\*[@name='firstname']"));  
 actions.click(isim).sendKeys(faker.name().firstName())  
 .sendKeys(Keys.TAB).sendKeys(faker.name().lastName())  
 .sendKeys(fakeMail).sendKeys(Keys.TAB)  
 .sendKeys("05554569878").sendKeys(Keys.TAB)  
 .sendKeys(faker.internet().password()).sendKeys(Keys.TAB)  
 .sendKeys(Keys.TAB).sendKeys(Keys.TAB)  
 .sendKeys("16").sendKeys(Keys.TAB).sendKeys("Mar")  
 .sendKeys(Keys.TAB).sendKeys("1993")  
 .sendKeys(Keys.TAB).sendKeys(Keys.TAB)  
 .sendKeys(Keys.ENTER)  
 .sendKeys(Keys.TAB).sendKeys(Keys.TAB).sendKeys(Keys.TAB).sendKeys(Keys.TAB).sendKeys(Keys.TAB)  
 .sendKeys(Keys.ENTER).perform();

driver.quit();

}

**Kaynak:**

* <https://www.baeldung.com/java-faker>
* <https://medium.com/techwasti/java-faker-library-to-generate-fake-data-229df7595211>