**DATE/TIME API**

Java kullanıcıları tarih işlemlerinde java.util.Date ve java.sql.Date paketlerine ait sınıfları kullanmaktaydı.

* Bu paketlere ait sınıfların tread-safe özelliği bulunmamaktadır.
* Tarih ve saat işlemleri birlikte ele alınmaktaydı.

Bu gibi nedenler ve sahip olduğu karmaşık yapısı sebebiyle Java geliştiricileri Date/Time apisini geliştirmişlerdir ve Java 8 ile birlikte kullanıcıların hizmetine sunmuştur.

Tarih işlemleri artık java.time paketi altında bulunan LocalDate sınıfından yönetilecektir.

Date/Time API default takvim olarak iso-8601 standartları çerçevesinde;

* Gerogian takvimi baz alınarak tasarlanmıştır.
* Thread-safe özelliği sağlanabilmektedir.
* Date işlemleri ile Time işlemleri birbirinden ayrı hale getirilmiştir.
* Paket içerisinde ki sınıflar immutable özelliğe sahiptirler.

DateTimeAPI ile geliştiricilere kullanım, anlaşılabilirlik ve kod okunabilirliği konusunda kolaylıklar sunulması amaçlanmıştır. Date/time apisi altındaki sınıflar java.time paketi altında yer almaktadır. Bu paketteki sınıfların yapılandırıcıları private erişim belirleyicisine sahip olduğundan “new” anahtar sözcüğü ile nesne oluşturulamaz. Bunun yerine now, of ve parse gibi metotlarla bu sınıflara ait nesneler oluşturulabilmektedir.

**LOCALDATE**

LocalDate nesnesi oluşturabilmek için now() ve of() metotları kullanılmaktadır.

**now ()** : Sistem tarihini dönmektedir. İstediğimiz bir bölgenin tarihini öğrenmek istersek parametre olarak vermemiz yeterli olacaktır.

**of ()** : LocalDate nesnesi oluşturmak için kullanılmaktadır. Parametre olarak yıl, ay ve gün bilgisi almaktadır.

**ofEpochDay()** : Tek parametre almaktadır. Epoch tarihinden itibaren parametre olarak verilen gün sonrasını bulmaktadır.

Epoch tarihi 1 Ocak 1970'tir. Bu tarihten itibaren parametre olarak verilen gün sonrası tarih bilgisini dönmektedir.

**parse ()** : String olarak verilen tarih, varsa pattern bilgisine göre LocalDate nesnesine dönüştürmektedir.

**format**() : LocalDate nesnelerini verilen date pattern’a göre string’e dönüştürmektedir.

**plus** () : Belirtilen tarihin üzerine gün, hafta, ay ve yıl eklemesi yapmaktadır.

**minus**() : Belirtilen tarihin üzerinden gün, hafta, ay ve yıl çıkarması yapmaktadır.

**ofYearDay**() : Yıl ve gün bilgisi olmak üzere iki parametre almaktadır. Belirtilen yılın, belirtilen günü hangi tarihe karşılık geliyorsa LocalDate nesnesi olarak olarak döndürür.

**isLeapYear**() : Tarihin artık yıl olup olmadığını kontrol eder. Toplam 366 gün sayısına sahip olan yıla artık yıl denir.

**isEqual**() : Tarihlerin eşit olup olmadığı kontrol edilir.

**isAfter**() : Tarihlerin öncelik ve sonralık durumunu kontrol eder.

**isBefore**() : Tarihlerin öncelik ve sonralık durumunu kontrol eder.

**get**() : Mevcut tarihin gün, hafta, ay ve yıl gibi bilgilerini vermektedir.

**LOCALTIME**

LocalTime nesnesi oluşturabilmek için now() ve of() metotları kullanılmaktadır.

**now** () : Sistem saatini dönmektedir. İstediğimiz bir bölgenin saatini öğrenmek istersek parametre olarak vermemiz yeterli olacaktır.

**of** () : LocalTime nesnesi oluşturmak için kullanılmaktadır. Parametre olarak saat, dakika, saniye ve salise bilgisi almaktadır.

**parse** () : String olarak verilen saat, varsa pattern bilgisine göre LocalDate nesnesine dönüştürmektedir.

**format**() : LocalTime nesnelerini verilen date pattern’a göre string’e dönüştürmektedir.

**plus** () : Belirtilen LocalTime nesnesi üzerine saat, dakika, saniye ve salise eklemesi yapmaktadır.

**minus**() : Belirtilen LocalTime nesnesi üzerine saat, dakika, saniye ve salise çıkarımı yapmaktadır.

**get**() : Mevcut LocalTime nesnesinden saat , dakika ve saniye gibi bilgilerini vermektedir.

**compareTo**() : Saatlerin eşit olup olmadığı kontrol edilir.

**isAfter**() : Saatlerin öncelik ve sonralık durumunu kontrol eder.

**isBefore**() : Saatlerin öncelik ve sonralık durumunu kontrol eder.

**LOCALDATETIME**

Tarih ve saat bilgisini tutmaktadır. Zone bilgisi olmadan.

**now** () : Sistem tarih ve saat bilgisini dönmektedir. İstediğimiz bir bölgenin tarih ve saati bilgini öğrenmek istersek parametre olarak vermemiz yeterli olacaktır.

**of**() : LocalDateTime nesnesi oluşturmak için kullanılmaktadır. Parametre olarak tarih ve saat bilgisi almaktadır. Farklı overload seçenekleri vardır.

**max**(): Desteklediği maximum tarih bilgisini gösterir.

**min**(): Desteklediği minimum tarih bilgisini gösterir.

**parse**(): String olarak verilen tarih ve saat, varsa pattern bilgisine göre LocalDate nesnesine dönüştürmektedir.

**toLocaleDate**(): Tarih bilgisinin döndürür.

**toLocalTime**(): Saat bilgisini döndürür.

**get**() : Mevcut LocalDateTime nesnesinden yıl, ay, gün, saat , dakika ve saniye gibi bilgilerini almaktadır.

**ZONEID**

Dünya üzerindeki saat dilimleri bilgilerinin Java tarafındaki karşılığıdır.

**of**(): parametre olarak verilen bölge bilgisini ZoneId nesnesinde döner.

**systemDefault**(): Default alan bilgisini verir.

**getAvailableZoneIds**() : Dünya üzerindeki tüm alanları listeler.

**getRules**(): UTC farkını gösterir. [+3 gibi.]

**ZONEDATETIME**

Date time bilgisi yanında zone bilgisinide tutmaktadır.

**now**(): O an ki sistem tarih ve saat bilgisini bölgesiyle birlikte vermektedir.

**of**(): Parametre olarak verilen değerlere göre ZoneDateTime nesnesi oluşturur. Farklı overload tipleri mevcuttur.

**toLocaleDate**(): Tarih bilgisinin döndürür.

**toLocaleTime**(): Saat bilgisini döndürür.

**toLocaleDateTime**(): Tarih ve saat bilgisini döndürür.

**getZone**(): Bölge bilgisini döndürür.

**withZoneSameInstant**() : Bir bölgedeki zaman dilimindeki tarih ve saatte, diğer bölgede tarih ve saatin aynı anda kaç olduğunu gösterir.

**INSTANT**

UTC bilgisini katmadan anlık tarih ve saat bilgisini verir. UTC+0 saat dilimini veriyorda diyebiliriz.

**now**(): UTC+0’a göre tarih ve saat bilgisini verir.

**toEpochMilli**(): Anlık saat ve tarih dilimini milisaniye cinsinden verir.

**PERIOD**

İki LocalDate nesnesi arasındaki farkı ölçmek için kullanılır.

**between**(): İki tarih nesnesi arasındaki periyodu verir. P -> period, Y -> yıl, M -> ay, D -> gün bilgisini vermektedir.

**DURATION**

İki LocalTime arasındaki zamanı ölçmek için kullanılır.

**between**(): İki time nesnesi arasındaki zamanı verir.

**CHRONO UNİT**

Zaman birimlerini göstermek için tasarlanan bir enum’dır. İçerisinde 1 yıl, 1 ay , 1 hafta gibi zaman birimleri bulunmaktadır. LocalTime’lar için geçerlidir sadece.

**between**(): Zaman birimine göre iki birim arasındaki farkı verir.