Jetpack Compose ile Android Uygulama Geliştirme Kursu

İleri Android

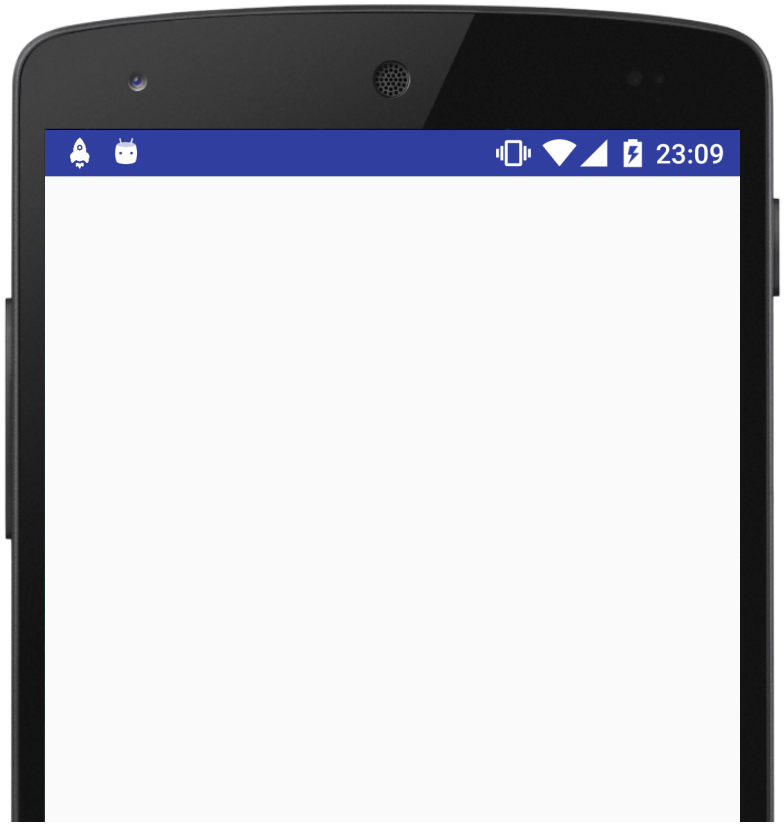
Kasım ADALAN
Elektronik ve Haberleşme Mühendisi
Android - IOS Developer and Trainer

Eğitim İçeriği

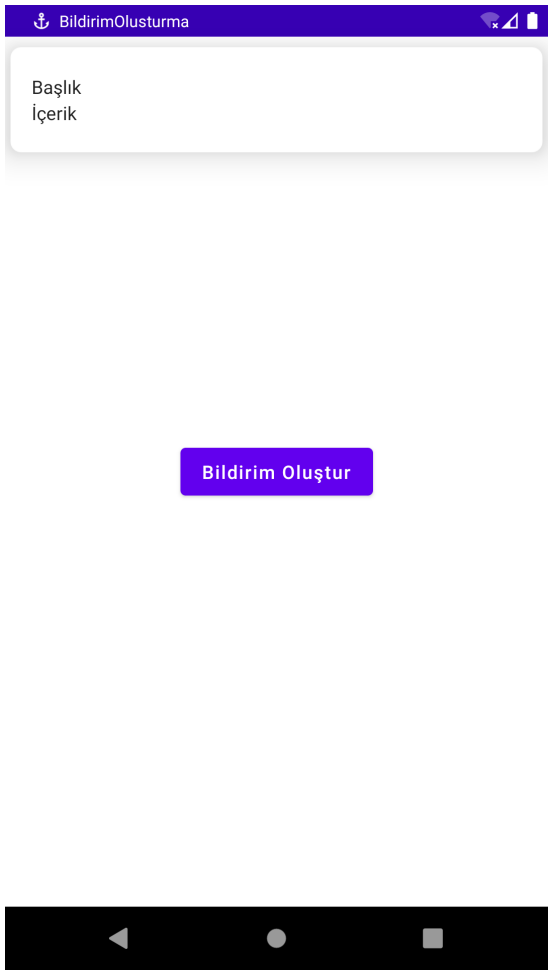
- Bildirim oluşturma
- WorkManager
- Zamansal Bildirim
- Push Notification

Notification

Notification Nedir ?



Bildirim Oluşturma




```
@Composable
fun Sayfa() {
    val context = LocalContext.current

    Column(
        Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Button(onClick = {
            bildirimOlustur(context)
        }) { Text( text: "Bildirim Oluştur") }
    }
}
```

```kotlin
fun bildirimOlustur(context:Context){
    val builder : NotificationCompat.Builder
    val bildirimYoneticisi = context.getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager

    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){

        val kanalId = "kanalId"
        val kanalAd = "kanalAd"
        val kanalTanitim = "kanalTanitim"
        val kanalOnceligi = NotificationManager.IMPORTANCE_HIGH

        var kanal : NotificationChannel? = bildirimYoneticisi.getNotificationChannel(kanalId)

        if(kanal == null){
            kanal = NotificationChannel(kanalId,kanalAd,kanalOnceligi)
            kanal.description = kanalTanitim
            bildirimYoneticisi.createNotificationChannel(kanal)
        }

        builder = NotificationCompat.Builder(context,kanalId)

        builder.setContentTitle("Başlık")
            .setContentText("İçerik")
            .setSmallIcon(R.drawable.resim)
            .setAutoCancel(true)
    }else{
        builder = NotificationCompat.Builder(context)

        builder.setContentTitle("Başlık")
            .setContentText("İçerik")
            .setSmallIcon(R.drawable.resim)
            .setAutoCancel(true)
            .priority = Notification.PRIORITY_HIGH
    }

    bildirimYoneticisi.notify( id: 1,builder.build())
}
```

Work Manager

Work Manager

- Arkaplanda işlem yapmak için kullandığımız bir yapıdır.
- İstediğiniz işlemleri arkaplanda yaptırabiliriz.
  - Örneğin : Bildirim oluşturmak , resim yüklemek , sayısal işlem yaptırmak vb.

- Bu işlemlere geçikme verip aynı zamanda periyodik olarak yaptırabiliriz.
- Zamana bağlı olarak durumları tetikleyebiliriz.
- Telefon kapansada ekstra bir işlem yapmadan çalışmaya devam edecektir.

- En güncel yapıdır ve Android Jetpack yapısı ile birlikte gelmektedir.
- Alarm Manager ve Job Scheduler altyapılarının üzerine kurulmuş bir yapıdır.
- Alarm Manager sadece eski android sürümlerini desteklemektedir.
- Work Manager bütün sürümler ile uyumludur.

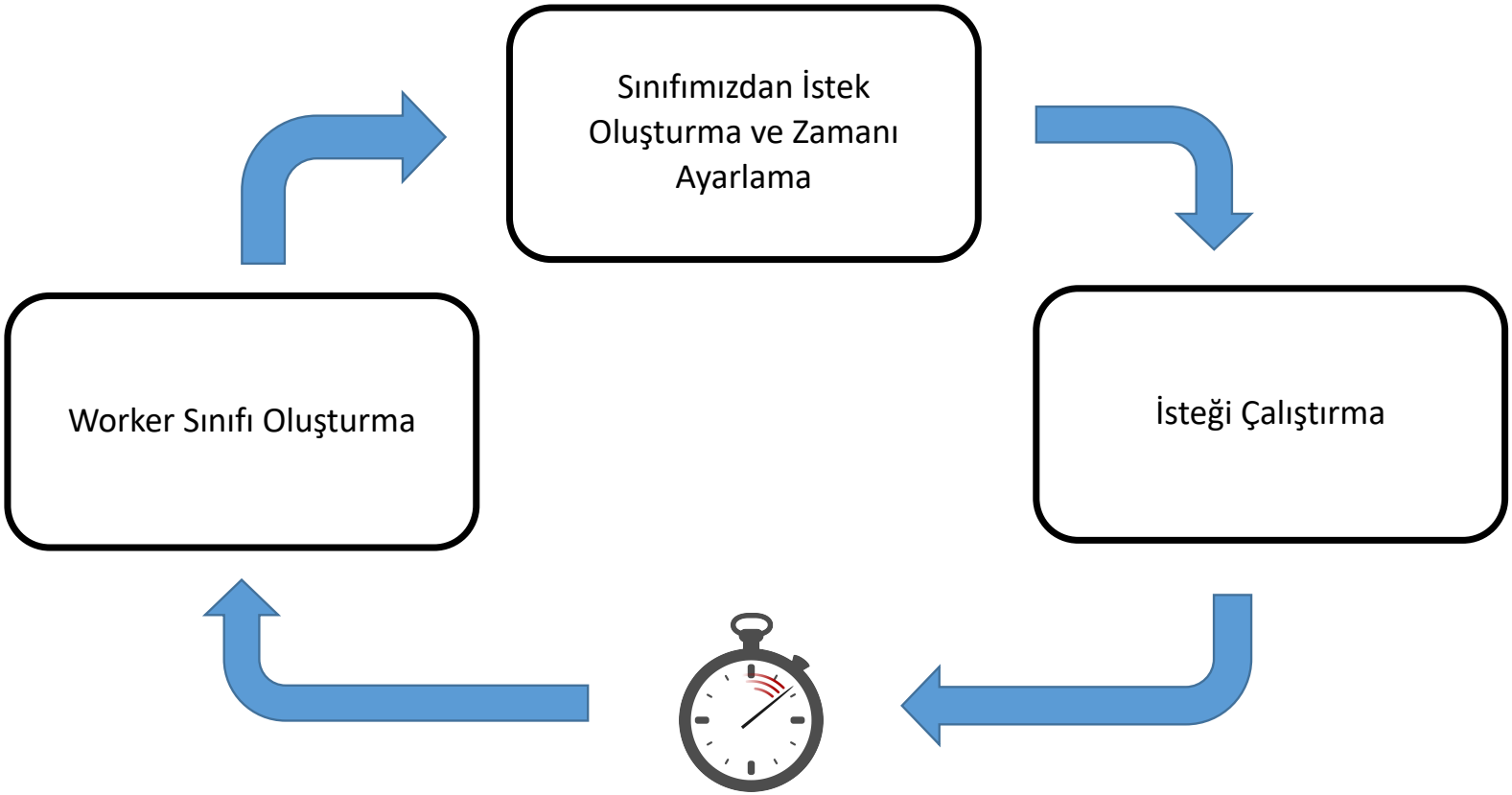
Kurulum

- Güncel Kütüphane Linki
- https://developer.android.com/topic/libraries/architecture/workmanager/basics

implementation "androidx.work:work-runtime-ktx:2.5.0"

NOT : API Seviyesi minimum API 26 olmalıdır.

Çalışma Yapısı



KODLAMA YAPISI

Worker Sınıfı Oluşturma

1

- Zamansal olarak işlem yapmak için bir sınıf oluşturulur.
- Bu sınıf *Worker* sınıfından miras yolu ile extend edilir.

```
class MyWorker(appContext: Context, workerParams: WorkerParameters): Worker(appContext, workerParams) {

    override fun doWork(): Result {
        val toplam = 10 + 20
        Log.e( tag: "Arkaplan İşlemi Sonucu",toplam.toString())

        return Result.success()
    }

}
```

Arkaplanda çalışıtırılacak kodların yazılacağı metod.

Dönüş değeri duruma göre değişebilir.

`Result.success()` *İşlemin başarılı şekilde bittiğini belirtir.*

`Result.failure()` *İşlem olurken hata olduğunu belirtir.*

İstek Oluşturma

- Oluşturduğumuz sınıftaki metodu çalıştırmak için istek oluşturulur ve zamana duyarlı bir şekilde çalışması beklenir.
- İki farklı istek oluşturabiliriz.

  - *OneTimeWorkRequest* : Bir kere çalışacak istek.

```
val istek = OneTimeWorkRequestBuilder<MyWorker>().build()
```

  - *PeriodicWorkRequest* : Tekrarlı işlem yapmak için istek.

```
val istek = PeriodicWorkRequestBuilder<MyWorkerBildirim>( repeatInterval: 15,TimeUnit.MINUTES)
    .setInitialDelay( duration: 10,TimeUnit.SECONDS)
    .build()
```

İsteği Çalıştırma


```
WorkManager.getInstance(context).enqueue(istek)
```

İstek Türleri

OneTimeWorkRequest

- Bir kere çalışmak için ayarlanan istektir.
- Anlık veya gecikmeli olarak arka planda işlem yaptırabilirsiniz.

- *Anlık Çalıştırma ;*

```
val istek = OneTimeWorkRequestBuilder<MyWorker>().build()
```

- *Gecikmeli Çalıştırma ;*

```
val istek = OneTimeWorkRequestBuilder<MyWorker>()
    .setInitialDelay( duration: 10,TimeUnit.SECONDS)
    .build()
```

Gecikme Süre miktarı

Sürenin türü saniye, dakika vb.

Örnek : OneTimeWorkRequest

Butona basıldığı anda anlık veya gecikmeli işlem yaptırma

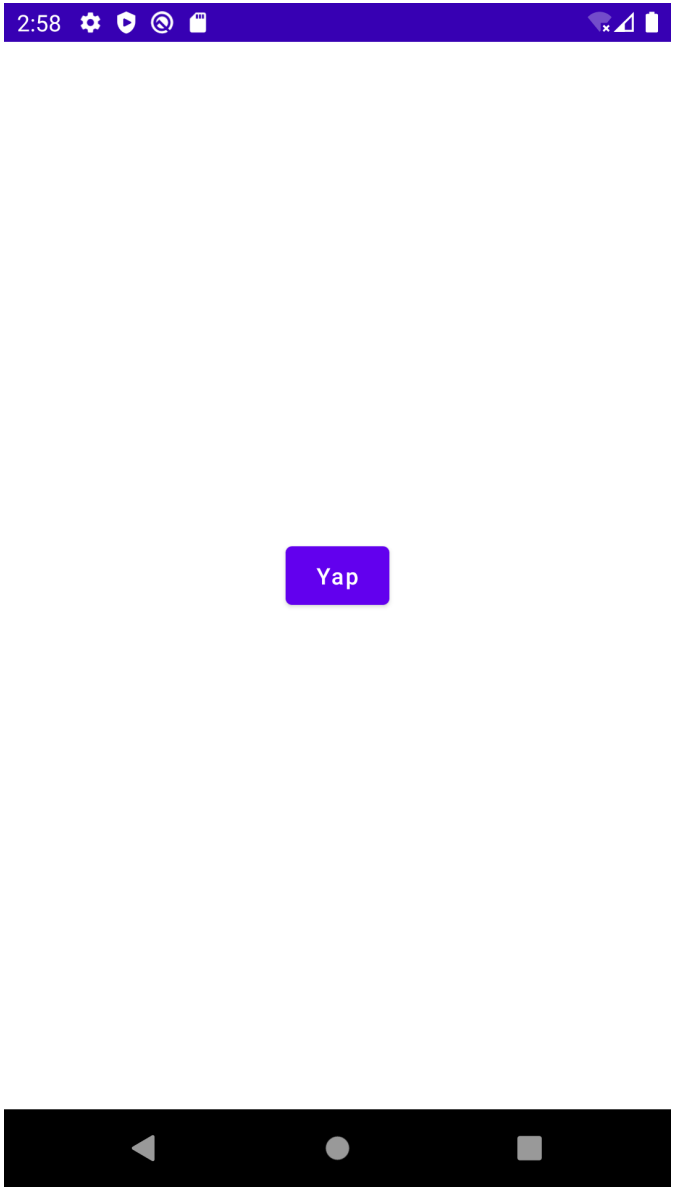
```
class MyWorker(appContext: Context, workerParams: WorkerParameters): Worker(appContext, workerParams) {

    override fun doWork(): Result {
        val toplam = 10 + 20
        Log.e( tag: "Arkaplan İşlemi Sonucu",toplam.toString())

        return Result.success()
    }

}
```


```
@Composable
fun Sayfa() {
    val context = LocalContext.current

    Column(
        Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Button(onClick = {

            val istek = OneTimeWorkRequestBuilder<MyWorker>().build()

            WorkManager.getInstance(context).enqueue(istek)

        }) { Text( text: "Yap") }
    }
}
```

Yap

Butona basıldığı anda arka planda işlem yapar ve sonucu konsoldan takip edebiliriz.

```
@Composable
fun Sayfa() {
    val context = LocalContext.current

    Column(
        Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Button(onClick = {

            val istek = OneTimeWorkRequestBuilder<MyWorker>()
                .setInitialDelay( duration: 10,TimeUnit.SECONDS)
                .build()

            WorkManager.getInstance(context).enqueue(istek)

        }) { Text( text: "Yap") }
    }
}
```



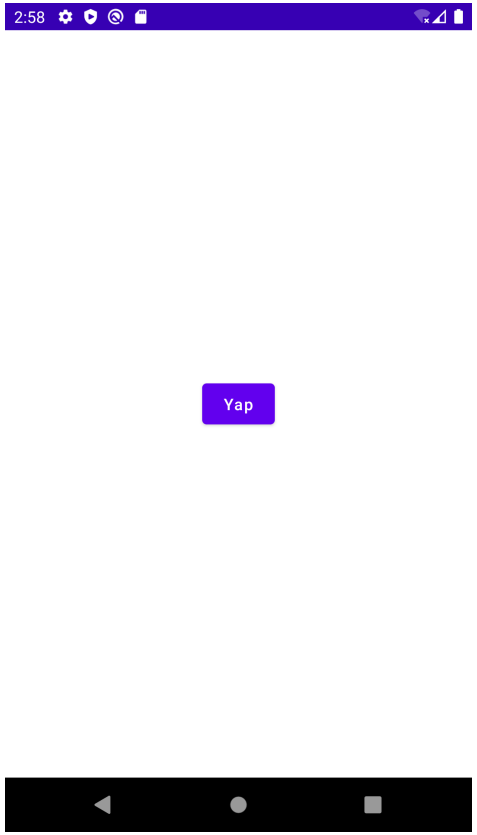
Butona basıldığı anda 10 saniye gecikmeli olarak arka planda işlem yapar ve sonucu konsoldan takip edebiliriz.

Arkaplan işlerini koşullara göre çalıştırma

- Constraint oluşturarak çalışma şartı oluşturabiliriz
  - Örneğin sadece telefon internete (wifi ve mobil) bağlıysa çalış diyebiliriz.

```
val calismaKosulu = Constraints.Builder()
    .setRequiredNetworkType(NetworkType.CONNECTED)
    .build()

val istek = OneTimeWorkRequestBuilder<MyWorker>()
    .setInitialDelay( duration: 10,TimeUnit.SECONDS)
    .setConstraints(calismaKosulu)
    .build()
```

```kotlin
@Composable
fun Sayfa() {
    val context = LocalContext.current

    Column(
        Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Button(onClick = {

            val calismaKosulu = Constraints.Builder()
                .setRequiredNetworkType(NetworkType.CONNECTED)
                .build()

            val istek = OneTimeWorkRequestBuilder<MyWorker>()
                .setInitialDelay( duration: 10,TimeUnit.SECONDS)
                .setConstraints(calismaKosulu)
                .build()

            WorkManager.getInstance(context).enqueue(istek)

        }) { Text( text: "Yap") }
    }
}
```



Test İşlemi :

1. *Uygulama çalıştırılır.*
2. *İnternet (Wifi ve Mobil) bağlantısı kesilir.*
3. *Buttona basılarak işlem yapmaya çalışır.*
4. *İnternet olmadığı için çalışma olmayacaktır.*
5. *İnternet açılır.*
6. *İnternet açılır açılmaz çalışmasını görebiliriz.*

PeriodicWorkRequest

- İstenirse belirli bir süre içinde arka planda tekrarlı işlem yaptırılabilir.
- Tekrar süresi minimum 15 dk olmalıdır.

```
val istek = PeriodicWorkRequestBuilder<MyWorkerBildirim>( repeatInterval: 15,TimeUnit.MINUTES)
    .setInitialDelay( duration: 10,TimeUnit.SECONDS)
    .build()
```

İlk çalışma için geçikme miktarı.
10 sn sonra ilk çalışma olacak
ve
15 dk arayla çalışacak

Tekrarlama
Süre miktarı

Sürenin türü
saniye, dakika vb.

TimeUnit ile birçok türde zaman birimine göre tekrar aralığı belirleyebilirsiniz

TimeUnit.*MINUTES* TimeUnit.*HOURS* TimeUnit.*DAYS*

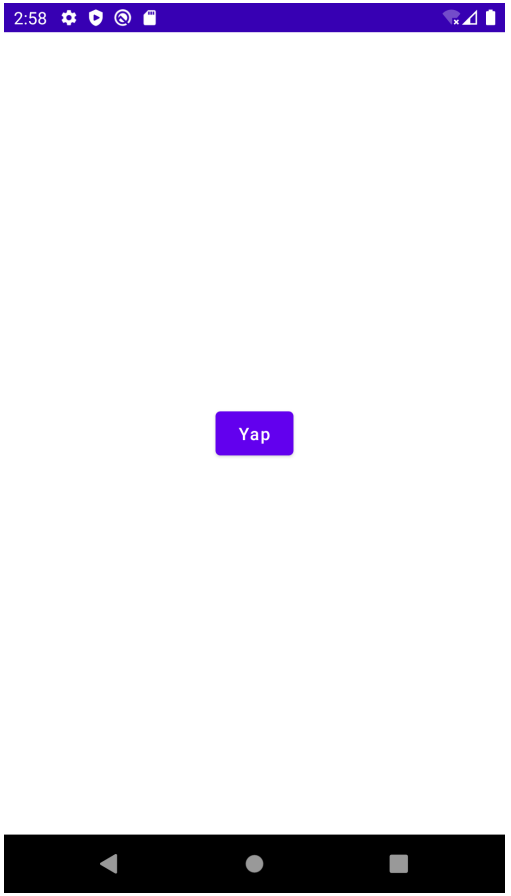
Örnek : PeriodicWorkRequest

```
class MyWorkerBildirim(appContext: Context, workerParams: WorkerParameters): Worker(appContext, workerParams) {

    override fun doWork(): Result {
        bildirimOlustur()
        return Result.success()
    }

    fun bildirimOlustur(){
        val builder : NotificationCompat.Builder
        val bildirimYoneticisi = applicationContext
            .getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
        val intent = Intent(applicationContext,MainActivity::class.java)
        val gidilecekIntent = PendingIntent
            .getActivity(applicationContext, requestCode: 1,intent, PendingIntent.FLAG_UPDATE_CURRENT)
```

```
if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){

    val kanalId = "kanalId"
    val kanalAd = "kanalAd"
    val kanalTanıtım = "kanalTanıtım"
    val kanalOnceligi = NotificationManager.IMPORTANCE_HIGH

    var kanal : NotificationChannel? = bildirimYoneticisi.getNotificationChannel(kanalId)

    if(kanal == null){
        kanal = NotificationChannel(kanalId,kanalAd,kanalOnceligi)
        kanal.description = kanalTanıtım
        bildirimYoneticisi.createNotificationChannel(kanal)
    }

    builder = NotificationCompat.Builder(applicationContext,kanalId)

    builder.setContentTitle("Başlık")
        .setContentText("İçerik")
        .setSmallIcon(R.drawable.resim)
        .setContentIntent(gidilecekIntent)
        .setAutoCancel(true)
    }else{
        builder = NotificationCompat.Builder(applicationContext)

        builder.setContentTitle("Başlık")
            .setContentText("İçerik")
            .setSmallIcon(R.drawable.resim)
            .setContentIntent(gidilecekIntent)
            .setAutoCancel(true)
            .priority = Notification.PRIORITY_HIGH
    }

    bildirimYoneticisi.notify( id: 1,builder.build())
}

}
```

```kotlin
@Composable
fun Sayfa() {
    val context = LocalContext.current

    Column(
        Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        Button(onClick = {

            val istek = PeriodicWorkRequestBuilder<MyWorkerBildirim>( repeatInterval: 15,TimeUnit.MINUTES)
                .setInitialDelay( duration: 10,TimeUnit.SECONDS)
                .build()

            WorkManager.getInstance(context).enqueue(istek)

        }) { Text( text: "Yap") }
    }
}
```

İlk çalışma için geçikme miktarı.
10 sn sonra ilk çalışma olacak
ve
15 dk arayla çalışacak



Teşekkürler...

kasım-adalan

kasimadalan@gmail.com

kasimadalan