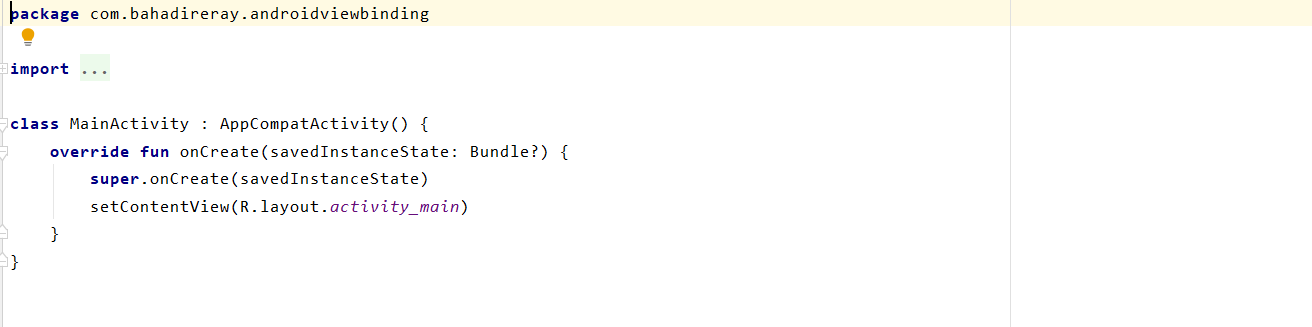
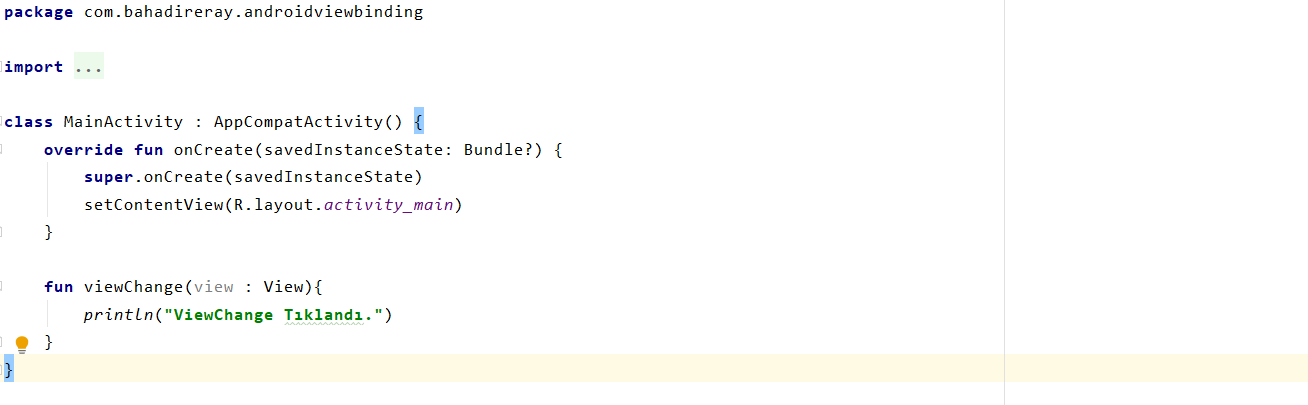
Android View Binding?

Nedir bu View Binging, Aslında bir jet pack ktürüphanesi

Bu özelliği kullanmadan olur mu, olur ama kullanırsak neler olur;

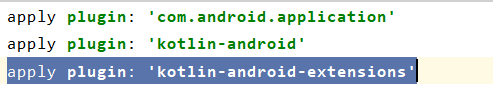
Özellikle Google play’a yüklemek istedeceğiniz uygulamalarda daha sonra geliştirmek üzerine eklemeler yapmak istediğiniz gayet faydasını göreceksiniz.





Böyle bir kullanımla buton içerisinde tıklanmaları dinleyip işlem yapabiliyoruz. Peki asıl olayımız nedir? Görünüm penceresindeki değişiklerini nasıl daha efektif bir kullanımla değiştirebiliriz.

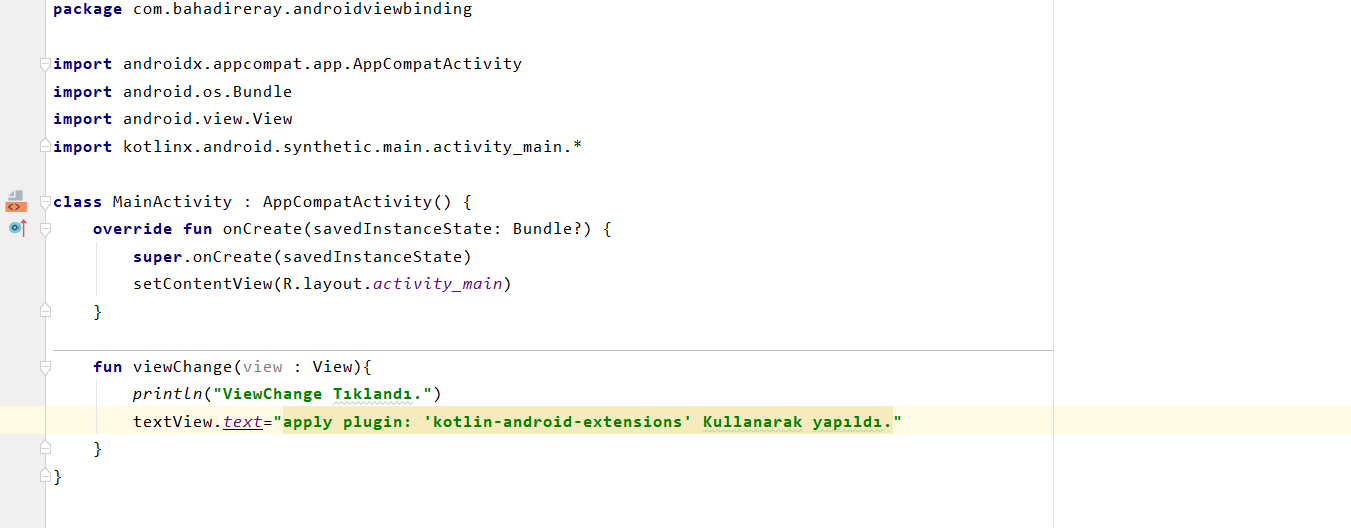
Eski kullanımda fonksiyona textView içerindeki tanımlı bir ID’yi import edince ilgili değişikliğe erişebiliyorduk. Şimdi son güncellemeyle bazı tanımlamalar yaparak eski kalmış bu özelliği kullanabiliyoruz. Öncelikle onu anlatacağım daha sonra ViewBinding kullanımıza geçeceğiz.

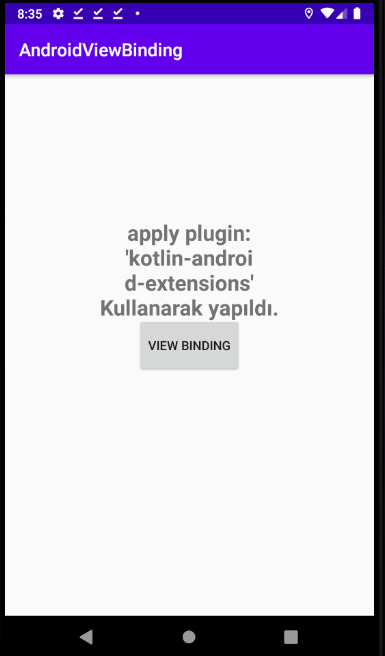
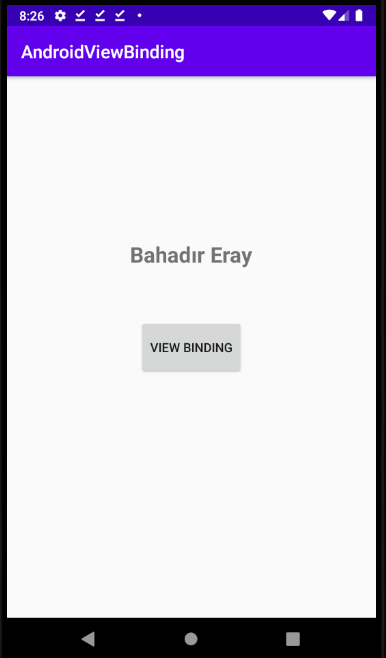


Son güncellemede mavi olarak işaretli belirtmiş olduğum alan bulunmamaktadır. Eski kalmış özelliği kullanabilmek için bunu yazmanız gerekecektir.

Daha sonra projemizde kod kısmında,

**import** kotlinx.android.synthetic.main.activity\_main.\* böyle sınıf eklenecektir. Eğer sorun yaşarsanız projeyi yeniden restart ederek başlatmanız gerekebilir.





Daha temellere inince kotlin bize sınıf içerisinde görsel öğelere erişim sağlarken aslında java temellerinde neler yapıyorduk biraz anımsayalım,

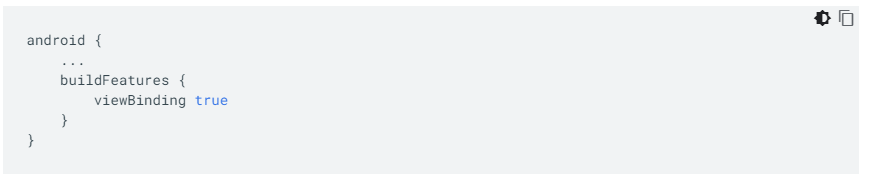
**val** textView=findViewById<TextView>(R.id.*textView*)  
textView.*text*=**"Merhaba eski düzen"**

Id ile bul deyip değişken oluşturuyoruz. Sınıf olarak (TextView ) Bir eşleşme oluşturunca bunun üzeirnde değişim yapıyorduk.

Şimdi javanın bu karmaşası üzerinde kotlin bir kısa erişimle düzenleme yapmamıza izin veriyordu. Daha sonra Andorid bize dedi ki ben daha güzel bir yolunu buldum bunu bir deneyin diyor, adıda View Binding şimdi bunun kullanıma geçelim,

Google arama penceresine, yazıp android view binding dediğimizde ilgili açıklamalarına erişebiliriz. Android projemizde iki adet build.gradle dosyamız bulunmaktadır. Biz bunlardan ‘app’ olanın içerisinde değişiklik yapacağız. İçerisine parantezlere dikkat ederek ekleyelim.

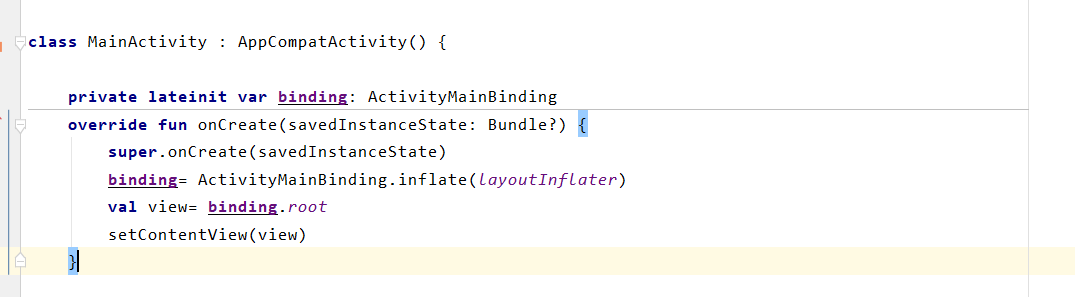
android {  
    ...  
    buildFeatures {  
        viewBinding true  
    }  
}



Bu eklemeyi yaptığımız zaman sıfırdan bir projemizi yeniden derleyip hatta yeniden bir başlatmakta yarar var. Çünkü view’lerin içerinde gerekli değişiklik içeren sınıfları eklemesi gerekiyor. Bazen synNow demek tam anlamıyla kullanmaya izin vermeyebilir. Diğer yeni sınıfların eklenme işlemlerinde de projeyi yeninde bir başlatmak size fayda sağlayacaktır.

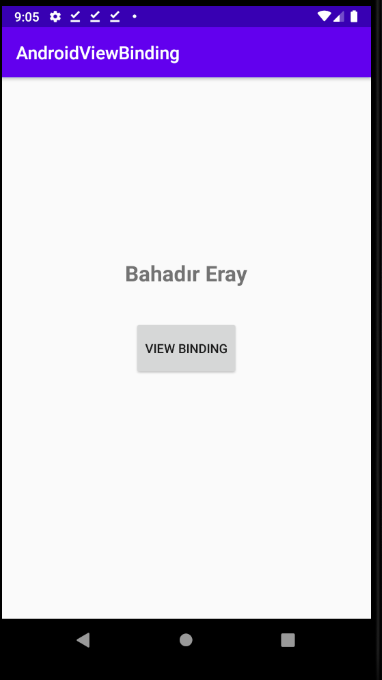
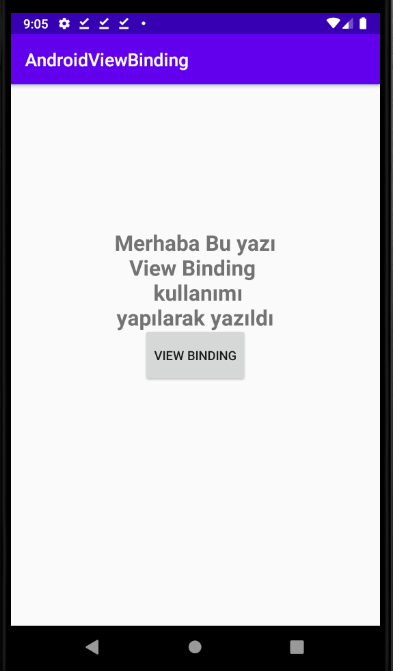
Binding diye bir değişken oluşturuyoruz ve O anki güncel aktivitemiz neyse onunla birlikte tamamlıyoruz. Değişkeni tamamlarken otomatik olarak bulunduğunuz aktivitenin isminde binding oluşturuyor. Aktiviteye özel binding oluşturma işlemimizden sonra onCreate altında bir kerelik olarak başlatmamış gerekiyor. Bir nevi sürekli R.id.textView demek yerine bir kere oluşturduğumuz görünüm altından değişkenlerimiz buradan alınıyor. Bize bir nevi kod karmaşasından kurtarıyor. Bu sınıf bizin R.id değişkeniyle çalışacağımızı anlıyor temelde bu tanımlamayı yaparak sürekli tanımlamaya ihtiyacımız olmuyor.

**class** MainActivity : AppCompatActivity() {  
  
 **private lateinit var binding**: ActivityMainBinding  
 **override fun** onCreate(savedInstanceState: Bundle?) {  
 **super**.onCreate(savedInstanceState)  
 **binding**= ActivityMainBinding.inflate(*layoutInflater*)  
 **val** view= **binding**.*root* setContentView(view)  
 }



Bu tanımlamarı yaptıktan sonra ilgili çağrımlarda kullanacağımız kod,

**binding**.**textView**.*text*=**"Merhaba Bu yazı View Binding \n kullanımı yapılarak yazıldı"**

Bu şekilde tanımlayarak kodumuz içerisinde view görünümle ilgili işlemleri Android View Binding kullarak yapabilir.