**ViewCompat.setOnApplyWindowInsetsListener fonksiyonu nedir, ne işe yarar?**

ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->  
 val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
}

Öncelikle temel yapımıza bakalım.

V -> ile belirtilen şey, bu listener'ın uygulandığı görünümü (View) temsil eder. Bu kod bloğunda binding.main ile belirttiğiniz görünüm bu listener'a atanmıştır ve herhangi bir window inset olayı gerçekleştiğinde bu görünüm ile ilgili bilgiler ve işlemler gerçekleştirilecektir.

Insets -> insets parametresi, ViewCompat.setOnApplyWindowInsetsListener fonksiyonu içinde kullanıldığında, WindowInsetsCompat sınıfının bir örneğini ifade eder. Bu örnek, Android işletim sisteminin, uygulamanızın penceresine uygulanan mevcut pencere kenar boşluklarını (window insets) içerir. Kenar boşlukları, cihazın durum çubuğu (status bar), gezinti çubuğu (navigation bar), ekran çentikleri (notch), yuvarlak köşeler gibi fiziksel ve sistem özelliklerinden kaynaklanan alanlar olabilir.

Peki WindowInsets nedir?

WindowInsets Android'de, bir uygulamanın pencere içindeki görünümlerinin çevresindeki boşlukları temsil eden bir sınıftır. Bu boşluklar, sistem arayüzü bileşenleri (status bar, navigation bar, system gestures area, vb.) ve ekran özellikleri (ekran çentikleri, yuvarlak köşeler, vb.) tarafından kullanılır. WindowInsets sayesinde, uygulama içeriği bu sistem bileşenlerine göre uygun şekilde düzenlenebilir, böylece kullanıcı arayüzü elemanlarınız önemli sistem bileşenleri tarafından örtülmez.

Genellikle, WindowInsetsCompat kullanarak, bir uygulamanın ekranı ne zaman kullanılabilir hale geldiğini ve kullanılamayacak hale geldiğini belirlemek veya ekranın belirli bir kısmına yerleştirilmiş bir görünümün konumunu ayarlamak gibi işlemler yapılır.

Öncelikle fonksiyonumuz enableEdgeToEdge() fonksiyonu olmadan çalışmıyor. Yani içine girmiyor.

*enableEdgeToEdge*()  
  
var a: Int? = null  
var c: Int? = null  
var b: Boolean? = null  
  
  
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->**  
  
*println*("Bu fonksiyona giriyorum ")  
  
 b = v.*isActivated* // Görünümün aktivasyon durumunu döndürür (aktifse true, değilse false).  
 c = v.*id*  
val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
  
 a = systemBars.top  
  
  
 *println*("A : $a")  
 *println*("B : $b")  
 *println*("C : $c")  
 insets  
**}**  
  
*println*("A : $a")  
*println*("B : $b")  
*println*("C : $c")  
  
*println*("İş parçacığı başladı.")  
  
// İş parçacığını 3 saniye duraklat  
Thread.sleep(10000)  
  
*println*("İş parçacığı devam ediyor.")

ÇIKTILAR:

A : null

B : null

C : null

İş parçacığı başladı.

İş parçacığı devam ediyor.

Bu fonksiyona giriyorum

A : 66

B : false

C : 2131230961

Bu fonksiyona giriyorum

A : 66

B : false

C : 2131230961

Kod öbeğinden ve çıktısından da anlayacağımız gibi, bu fonksiyon en son çağırılıyor. Önce diğer işlemlerin bitmesi bekleniliyor. Çağırıldığında 2 kez çağırılıyor.

Gelelim bazı özelliklerine

v.setPadding(left, top, right, bottom):

var temp = 40

v.setPadding(500, temp, temp, temp)

-> Default fonksiyonda main’e bağlandığı için, ekrandaki tüm view’ların yerini ayarlayabiliyor.

v.animation

-> Varsa maine atanmış animasyonu döndürüyor.

v.setBackgroundColor()

v.setBackgroundColor(Color.RED)

-> v view’ının arka plan rengini verdiğin renk yapıyor.

v.alpha

v.alpha = 0.2F  
println("alpha ${v.alpha}")

Çıktı : alpha 0.2

-> Görünümün şeffaflık değerini ayarlıyor veya getiriyor.

Not: Buradaki özelliklerin çoğu default olarak bağlanan view olan main’den gelmektedir. Yani fonksiyon ile alakası yoktur. Fonksiyon içinde çağırılan değerler sadece son sözü söylemektedirler.

Örneğin:

// fonksiyon dışında alpha değerini 0.2 yaptım

*binding*.main.*alpha* = 0.2

// fonksiyon içinde alpha değerini 1 yaptım

v.*alpha* = 1.0F

Sonuç olarak ekranım alpha değeri 1 olarak çizildi. Yani son sözü fonksiyonun içindeki değerler belirledi.

insets.hasInsets()

Herhangi bir inset var mı kontrol eder. Var ise true döndürür.

Insets.getInsets()

insets.getInsets(WindowInsetsCompat.Type.systemBars())

Çıktı : Insets{left=0, top=66, right=0, bottom=66}

Örnekte systemBars ifadesi, sistem çubuklarının (status ve navigation bar gibi) uygulamanın üzerinde kapladığı boşlukları almak için kullanılıyor.

Özet: Bu fonksiyon sadece uygulamada insets’ler var ise çalışır. Bu fonksiyonun içine yazılan kodlar son sözü söyler. Bu özellikleri kullanarak ekranı dinamik olarak, insets’lerden kaçınarak ayarlamak mümkündür. Bu, kullanıcıya çeşitli ekranlarda istenilen UI’ı sunmak için önemlidir. Fonksiyon bizim ekran ile ilgili işlemlerimizi kolaylaştırarak bize hız kazandırır, kolaylık sağlar.

Mustafa Koçer