OnApplyWindowInsetsListener

Bu işlev, bir görünüme pencere kenar eklentileri (window insets) uygular ve ekran üzerinde farklı değişiklikler yapmamıza olanak sağlar. Görünümün pencere kenar eklentileri uygulanmadan önce veya sonra ek işlemler gerçekleştirmek için kullanılabilir. Yani aslında activity oluşturulurken ekran düzeni üzerinde oynamamızı sağlar. Bu fonksiyon yardımıyla neler yapabiliriz bunu inceleyelim:

- Ekran köşelerine padding verebiliriz.

```
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 ^setOnApplyWindowInsetsListener

}
```

- Köşelere yaklaştıkça beliren bir gölge (elevation) verebiliriz.

```
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { view, insets ->
    val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
    if (systemBars.top > 0 || systemBars.bottom > 0) {
        view.elevation = resources.getDimension(R.dimen.elevation_value)
    } else {
        view.elevation = 0f
    }
    insets ^setOnApplyWindowInsetsListener
}
```

- Navigation bar'ın yüksekliğini, boyutunu, background'ını, şeffaflığını her şeyini ayarlayabileceğimiz esnekliğe sahip oluruz.

```
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { view, insets ->
    val navigationBar = insets.getInsets(WindowInsetsCompat.Type.navigationBars())
    view.layoutParams.height = navigationBar.bottom
    view.layoutParams.width = navigationBar.right - navigationBar.left
    insets ^setOnApplyWindowInsetsListener
}
```

- Xml üzerinden verdiğimiz padding'leri burada verebiliriz. (Özellikle RecyclerView için genel olarak 1 dp padding verilir. Her RecyclerView için bunu yapmaktansa Activity içerisinde bunu yapma fırsatımız var.)

- İstersek ekranın tümünü kullanabiliriz. (Status bar ve Navigasyon barı dahil)

- Ekranda klavye açıldığında, bütün ekranı klavyenin yüksekliği kadar yukarı kaydırabiliriz. Böylelikle daha anlaşılır bir görüntü elde edebiliriz.

- Ekranın oryantasyonuna göre farklı işlemler yapılabilir. Eğer yatay moddaysa ya da dikey moddaysa gibi kondisyonlar ekleyip farklı işlemler yapılabilir.

```
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { view, insets ->
    val orientation = resources.configuration.orientation
    if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
        // Yatay moda özgü işlemler
    } else {
        // Dikey moda özgü işlemler
    }
    insets ^setOnApplyWindowInsetsListener
}
```

- Sistem çubukları gizlendiğiniz animasyonlar ile soft bir geçiş yapılabilir.

```
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { view, insets ->
    val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
    if (systemBars.bottom == 0) {
        // Sistem çubukları gizleniyor, animasyon ekleyebiliriz
    }
    insets ^setOnApplyWindowInsetsListener
}
```

- Sistem çubuklarına dokunulduğunda farklı bir görünüm oluşturulabilir.

- Kısacası, bu fonksiyon bize view nesnesini verdiği için, View katmanının yapabileceği her şeyi yapabilme imkanını sunuyor.

Enes Arısoy