Android Activity Lifecycle Nedir ?

Android’de Activity, bir uygulamanın kullanıcı arayüzüne (UI) sahip tek bir ekranı temsil eder ve kullanıcıların bir uygulamayla etkileşime girmesi için bir giriş noktası görevi görür. Activity’nin oluşumundan kill edilişine kadar geçen döngüye lifecycle denir.

Genel olarak, Android uygulamaları birden fazla ekran içerecek ve uygulamamızın her ekranı Activity sınıfının bir uzantısı olacaktır. Aktiviteleri kullanarak tüm Android uygulama UI bileşenlerimizi tek bir ekrana yerleştirebiliriz.

Android uygulamasındaki çoklu aktivitelerden bir aktivite ana aktivite olarak işaretlenebilir ve bu, uygulamayı başlattığımızda beliren ilk ekrandır. Android uygulamasında her aktivite, gereksinimlerimize göre farklı eylemler gerçekleştirmek için başka bir aktivite başlatabilir.

Android Activity Lifecycle Yönetimi

Activity yaşam döngüsünün aşamaları arasındaki geçişlerde gezinmek için Activity sınıfı altı temel callback seti sağlar:

-onCreate(), -> Activity ilk açıldığında gerçekleşen olayları içinde barındırır.

-onStart(), -> Activity başlatıldığında onStart() metodu çok hızlı bir biçimde devreye girer. Bu metot devreye girdiğinde uygulama kullanıcıya görünür hale gelir.

-onResume(), -> Activity onResume() metoduna girdiğinde uygulama ön plana gelir ve sistem onResume() çağrısında kalır. Bu metot kullanıcıyla etkileşime girildiği durumdur. Bu metot telefon çağrısı alana kadar ya da başka bir etkinliğe girene kadar kısaca herhangi bir işlem yapana kadar kalmaktadır.

-onPause(), -> Sistem uygulamanın bir nevi kapandığını bildirdiği zaman çalışır. Yani uygulamamız arka plana atıldığı zaman ,arama geldiği zaman ya da başka bir uygulama açıldığı zaman bu metot çalışır.

-onStop(), -> Kullanıcı uygulamadan çıkış yaptığında buradaki kodlar çalışmaktadır. Herhangi bir işlemi uygulama kapatırken sonlandırmak istersek bu işlemi burada yapabiliriz.

-onDestroy(). -> Bu metot kullanıcı ile bağlantısı kalmayan uygulamanın arka planı temizlemek için kullanılır.

Bir aktivite yeni bir duruma girerken sistem bu callbacklerin her birini çağırır. Bu callbackler içinde activityi yönetebiliriz.

Problem ve Çözüm Önerileri

OnStart() metodu içinde örneğin Firebase için veri tabanından veri dinlemesi ile recyclerview güncelleyen bir fonksiyon eklendi. Daha sonra başka bir activitye gidildiği zaman eğer bu dinleme işlemi kapatılmazsa bellekte gereksiz yere yer kaplar ve diğer işlemlere daha az işlem alanı kalabilir. Bunun için OnStart() içinde başlatılan her dinleme işlemi OnStop() içinde kapatılmalıdır.

@Override  
public void onStart() {  
 super.onStart();  
 adapter.startListening();  
}  
  
@Override  
public void onStop() {  
 super.onStop();  
 adapter.stopListening();  
}

Ayrıyeten OnStart() ve OnStop() içinde çok fazla işlem yapılması istenirse bunların yapılması garanti edilmeyebilir. Uzun süren işlemlerin burada yapılması hatalara sebep olabilir. Bunun olmaması için olabildiğince önemli işlemleri bu alanlara ekleyip yapılandırmaları doğru ayarlamak gerekir.

Bir başka sorun ise activityler arası geçiş yaparken önceki activityin kapatılmış olması durumu. Yetersiz RAM, hafıza gibi sorundan dolayı önceki activity kapatılmış olabilir. Önceki activitynin durumunu kontrol etmek için şu kod kullanılabilir:

**if (lifecycle.currentState.isAtLeast(Lifecycle.State.STARTED))** {  
            // connect if not connected  
        }

Başka bir sorun olarak ekran döndürüldüğü zaman verilerin kaybedilmesi sorunudur. Ekran döndürülünce activity destroy edilir ve yeniden başlatılır. Verileri kaybetmemek için iki yol vardır. Bunlardan ilki SQLite veri tabanına geçici depolanabilir. Diğer yöntem ise preferences kullanmaktır. Veriler activity OnStop() metoduna girerken kaydedilir. OnStart() veya OnCreate() içinde kontrol edilerek veri eğer mevcutsa tekrar yüklenmesi sağlanır. Bu sayede ekran döndürülünce veri kaybetmesinin önüne geçilir.

 public class CalendarActivity extends Activity {  
     ...  
  
     static final int DAY\_VIEW\_MODE = 0;  
     static final int WEEK\_VIEW\_MODE = 1;  
  
     private SharedPreferences mPrefs;  
     private int mCurViewMode;  
  
     protected void onCreate(Bundle savedInstanceState) {  
         super.onCreate(savedInstanceState);  
  
         mPrefs = getSharedPreferences(getLocalClassName(), MODE\_PRIVATE);  
         mCurViewMode = mPrefs.getInt("view\_mode", DAY\_VIEW\_MODE);  
     }  
  
     protected void onPause() {  
         super.onPause();  
  
         SharedPreferences.Editor ed = mPrefs.edit();  
         ed.putInt("view\_mode", mCurViewMode);  
         ed.commit();  
     }  
 }

Burada SharedPreferences kullanarak verilerin kaybı önlenmiş olur.