Mobil Programlama
Android

AsyncTask

26.11.2020 tarihinde güncel

Bakabileceğiniz Etiketler:

Eğitmen: Geleceği Yazanlar

#android-thread #android-arkaplan #android

Android'de Thread yapıları

Android 101 derslerimizde arka plan işlemleri için Java'da kullanılan Thread yapılarından bahsetmiştik. Her ne kadar Android Java dilinin getirdiği özelliklerin birçoğunu kullanmamıza olanak tanısa da, Android SDK içerisinde yapılan işlen için Java'daki Thread yapılarının kullanılması tavsiye edilmez. Özellikle sonucu ana akışı (Main Thread) ilgilendiren işlem için (ön yüzde bir görseli güncellemek) Android kendine özgü mekanizmalar sunmaktadır.

Neden arka plan işlemleri?

Android uygulamalarında uzun sürecek işlemlerin arka planda yapılmasının sebebi ana akışın (Main Thread ya da UI Thread) uzun süre engellenmemesidir. Eğer ana akışı uzun süre bloke ederseniz uygulamanızın ana ekranı donacak ve kullanıcı uygulamanızın bozulduğunu düşünecektir. Aşağıdaki örnekte ana akışı engelleyen bir kod görünmektedir;

```
@Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      listView = (ListView) findViewById(R.id.rssListview);
      listView.setOnItemClickListener(new OnItemClickListener() {

         @Override
         public void onItemClick(AdapterView<?> arg0, View arg1, int position, long arg3) {
            Rss rss = (Rss) listView.getAdapter().getItem(position);
            Intent browserIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(rss.getOriginalPostUrl().trim()
            startActivity(browserIntent);
         }
      });

      longTask();

   }
```


Örnek kodu incelersek **longTask** metodu 10 saniye süren bir işlemi **onCreate** metodu içerisinde yapıldığından uygulama ana ekranının yüklenmesi en az 10 saniye sürer. Bu sırada kullanıcı uzun süre siyah bir ekranla karşılaşır ve muhtemelen uygulamayı kapatır. Bu sebepten dolayı uzaktaki bir sunucudan cevap beklemek, veritabanına uzun bir sorgu atmak gibi işlemler kesinlikle ana akış içinde gerçekleştirilmemelidir.

NOT : Android'in eski sürümlerinde ana akış içerisinde İnternet bağlantısı yapmak mümkün olsa da yeni sürümlerde bu işlem kesinlikle yasaktır. (http://developer.android.com/reference/android/os/NetworkOnMainThreadException.html)

AsyncTask sınıfı ile arka plan işlemleri

AsyncTask adında bir sınıf sunmaktadır. Bu sınıf içindeki metodlar yardımıyla arka planda farklı bir akış (Thread) üzerinc gerçekleştirilen işlemin kolayca ön yüzü etkileyen ana akışı (Main Thread) entegre edilmesini sağlar.

NOT: Arka planda Java Thread sınıfıyla çalıştırılan iş parçacıkları ön yüz elemanlarına müdahale edemez. Bunun için Handler adında bir sınıf kullanılmalıdır.

Aşağıda örnek bir AsyncTask sınıfı görüntülenmektedir;

```
public class BackgroundTask extends AsyncTask<Void, Void, Void> {

      @Override
      protected void onPreExecute() {
         super.onPreExecute();
      }

      @Override
      protected Void doInBackground(Void... params) {
         return null;
      }

      @Override
      protected void onPostExecute(Void result) {
         super.onPostExecute(result);
      }

      @Override
      protected void onProgressUpdate(Void... values) {
         super.onProgressUpdate(values);
      }

      @Override
      protected void onCancelled(Void result) {
         super.onCancelled(result);
      }

   }
```

AsyncTask sınıfı abstract bir sınıftır ve kullanılması için başka bir sınıf üzerinden **extend** edilmesi gerekir. AsyncTask sınıflarında **doInBackground** metodu bulunması zorunludur ve arka planda gerçekleştirilecek bütün işlemler bu metod içerisinde yazılır. Diğer metodları da açıklarsak;

onPreExecute: Arka plan işlemi başlamadan önce ön yüzde değiştirilmesi istenen değişkenlerin (ProgressBar gibi animasyonlar) ve AsyncTask içinde gerekli değişkenlerin değer ataması yapılır.
doInBackground: Arka planda yapılması istenen işlem burada gerçekleşir. Bu metod içinde yapılan işlemler ön yüzde kullanıcının uygulamayı kullanmasını kesinlikle etkilemez. Eğer buradaki işlemler sonucunda ana akışa bir değişken gönderilmesi gerekiyorsa **retu** metodu ile bu değişken **onPostExecute** metoduna paslanabilir.
onPostExecute: doInBackground metodu tamamlandıktan sonra işlemlerin sonucu bu metoda **result** değişkeni ile gönderilir. Buradaki işlemler ana akışı etkiler ve herhangi bir hataya sebep olmaz. Arka plandaki işlemden gelen bir veri ön yüzde gösterilmek isteniyorsa metod içinde gösterim işlemi yapılabilir.
onProgressUpdate: Eğer doInBackground metodu içerisinde yaptığınız işlemin ilerleme durumunu (örneğin dosya indirme yüzdesi) kullanıcıya bildirmek istiyorsanız bu metodu kullanabilirsiniz. doInBackground metodu içerisinde publishProgress metodunu kullanarak ilerleme durumunu onProgressUpdate metoduna iletip önyüz içerisinde buna göre bir animasyon yapabilirsiniz.
onCancelled: Eğer herhangi bir sebepten dolayı AsyncTask iptal edilirse bu metod uyarılır. Burada kullanıdığınız kaynakları temizleyebilirsiniz.

Şimdi basit bir uygulama ile AsyncTask kullanarak nasıl bir işlem yapacağımızı ve işlem sürerken kullanıcıya işlemin durumunu nasıl bildireceğimizi gösterelim. Öncelikle sadece tek bir düğmeden oluşan bir layout dosyası hazırlayalım ve düğmeye aşağıdaki aksiyonu verelim;

```
public class MainActivity extends Activity {

   private ProgressDialog progressDialog;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
```

```
        ((Button) findViewById(R.id.startAsync)).setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                new BackgroundTask().execute((Void) null);
            }
        });
    }
}
```

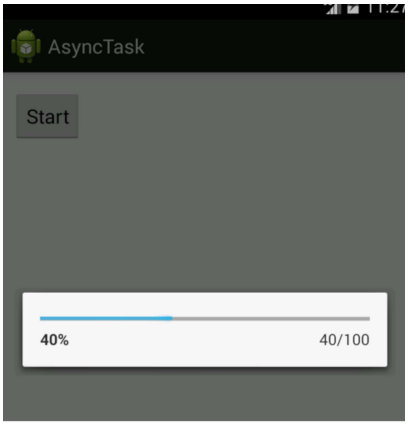
Yukarıdaki kodumuzda kullanıcı düğmeye bastığında **BackgroundTask** adındaki AsyncTask sınıfı **execute** metodu kullanılarak başlatılacaktır. Eğer AsyncTask'a bir değişken göndermek istiyorsak bunu **execute** metodu içerisinde verebi Bu durumda değişkenler direkt **doInBackground** metoduna gönderilecektir. **progressDialog** değişkeni ise kullanıcıyı arkı plan işleminin durumunu bildirmek için kullanılır.

NOT: AsyncTask'ın oluşturulma yapısına dikkat ederseniz AsyncTask<Tip1,Tip2,Tip3> şeklinde bir yapı görürsünüz. Buradı Tip1 doInBackground metoduna verilecek parametrelerin tipini ya da sınıfını belirler (örnekte Void). Tip2 doInBackgroun metodunun işleyişi sırasında onProgressUpdate metoduna paslanacak değişkenin tipini ya da sınıfını belirtir. Tip3 ise onPostExecute metoduna verilen değişkendir ve aynı zamanda doInBackground metodunun **return** tipidir. (http://developer.android.com/reference/android/os/AsyncTask.html)

```
public class BackgroundTask extends AsyncTask<Void, Integer, Void> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        progressDialog = new ProgressDialog(MainActivity.this);
        progressDialog.setMax(100);
        progressDialog.setProgress(0);
        progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
        progressDialog.show();
    }

    @Override
    protected Void doInBackground(Void... params) {
        for (int i = 0; i < 101; i = i + 10) {
            try {
                publishProgress(i);
                Thread.sleep(1000);
            }
            catch (InterruptedException e) {
            }
        }
        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        super.onPostExecute(result);
        progressDialog.dismiss();
    }

    @Override
    protected void onProgressUpdate(Integer... values) {
        super.onProgressUpdate(values);
        Integer currentProgress = values[0];
        progressDialog.setProgress(currentProgress);
    }

    @Override
    protected void onCancelled(Void result) {
        super.onCancelled(result);
        progressDialog.dismiss();
    }
}
```

Buradaki metodların yaptıklarını sırasıyla incelersek;

onPreExecute: Bu metod içerisinde işlem başlatılmadan önceki tanımlamalar yapılmıştır. ProgressDialog değişkenine ait tanımları bu metod içerisinde gerçekleştiririz. **show** metodu ile ekrana durum çubuğunu getiriyoruz.
doInBackground: Burada yaklaşık 10 saniye sürecek bir işlem gerçekleştiriyoruz. Bu işlem sırasında her 1 saniyede publishProgress metodunu kullanarak ilerlemeyi onProgressUpdate metoduna gönderiyoruz.
onPostExecute: İşlem tamamlandıktan sonra ekrandaki durum çubuğunu **dismiss** metoduyla yok ediyoruz.
onProgressUpdate: Durumu çubuğunun ilerlemesi doInBackground metodundan publishProgress ile gönderiliyor. Burada da gelen rakamsal değerle durum çubuğunu güncelliyoruz.
onCancelled: Eğer işlem herhangi bir sebepten dolayı iptal olursa durum çubuğunu **dismiss** metoduyla yok ediyoruz.



NOT: AsyncTask'lar içerisinde çalıştıkları Activity herhangi bir sebepten dolayı ölürse yaptıkları işlemler tamamlanmayal Bu yüzden Android SDK içerisinde kısa süreli işlemler ya da Activity yaşadığı sürece gerçekleşmesi beklenen işlemler içi kullanılmaları tavsiye edilir. Çok uzun süreli arka plan işlemleri (örneğin veritabanı güncelleme, dosya indirme) için **Servi** sınıflarının kullanılması tavsiye edilir. Bu konuyu sonraki bölümlerde detaylı olarak anlatacağız.

Önceki Sonraki