Xmlns

Xmlns, xml ad alanı anlamına gelir. XML ad alanları, bir XML belgesinde benzersiz olarak adlandırılmış öğeler ve nitelikler sağlamak için kullanılır. Aynı zamanda xml dosyalarında adlandırma çakışmalarını önlemek için yaratılmıştır.

Xml dosyamızın kökünde xmlns:android="http://schemas.android.com/apk/res/android" yapısını bildirdiğimizde, bu ad alanına zaten eklenmiş olan tüm nitelikler ve etiketler içe aktarılacaktır. Yani bir dahaki sefere android: ifadesini yazdığımızda otomatik olarak niteliklerin geldiği tamamlama listesi oluşur.

Bir öğe için varsayılan bir ad alanı tanımlamak, bizi tüm alt öğelerde önek kullanmaktan kurtarır.

Yani aslında android:id demek aslında http://schemas.android.com/apk/res/android:id demektir. Fakat zaten biz varsayılan bir alan adı tanımladığımız için her nitelik için tekrar tekrar bu ifadeyi belirtmemiz gerekmez.

Peki nasıl isim çakışmaları önleniyor?

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent">

<EditText
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_marginBottom="4dp"
   android:hint="User Name"
  />
</LinearLayout>
```

Yukarıdaki xml kodunu incelediğimizde biz eğer ki LinearLayout için tekrardan layout_width tanımlarsak hata alırız. Çünkü zaten bu nitelik öncesinde tanımlanmıştır. İşte bu aynı niteliği tanımla işlemini namespaceler sayesinde ayırt ediyoruz. Çünkü tekrar tanımlamaya çalıştığımız layout_width niteliği de android namespace'sine sahip. İki kez aynı namespace'e sahip aynı niteliği tanımlayamayız. Fakat farklı bir namespace'e sahip fakat adı aynen layout_width olan bir niteliği tanımlayabiliriz.

Tools Namespace

Tools namespaceler, Android Studio'nun sağlamış olduğu tasarım zamanı özelliklerini, veya derleme zamanı davranışlarını etkinleştiren çeşitli XML özniteliklerini destekler. Uygulamamızı oluşturduğunuzda, derleme araçları bu öznitelikleri kaldırır, böylece APK boyutunuz veya çalışma zamanı davranışınız üzerinde hiçbir etkisi olmaz.

Tools namespace'ler altındaki parametreler, geliştirme araçları tarafından, esas olarak IDE'de önizlemeyi kolaylaştırmak için kullanılır.

Diğer bir deyişle uygulama geliştirirken tasarım tarafında yapılan değişikliklerin çıktısını uygulamayı çalıştırmadan önce sadece tasarım anında görebilmemizi sağlarlar. Fakat uygulama çalıştıktan sonra bu değişiklikler dikkate alınmaz ve kaldırılır. Uygulama ekran çıktısında da bu değişiklikleri göremeyiz.

Bu öznitelikleri kullanabilmek için yine xml kök dizinimizde xmlns:tools="http://schemas.android.com/tools yapısını bildirmemiz gerekir.

Peki ne gibi durumlarda kullanabiliriz?

Örneğin tasarım anında constraintler ile ilgili bir uyarı alıyorsunuz ve aslında daha görünümlerinizn hepsini yerleştirmemişsiniz. Bu uyarıdan rahatsız oluyorsanız ve görünümleri yerleştirdikten sonra constraintleri verme işlemine geçecekseniz o an tools:ignore="MissingConstraints" niteliğini ekleyebilirsiniz. Fakat bu nitelikten ötürü de hata almayacağımız için constraintleri vermeyi unutursak uygulama çalıştıktan sonra tasarımda düzensizlikler olacaktır. Yani bu hatayı sadece tasarım anında görmezden geliyoruz.

Veya örneğin bir textView uygulama çalıştığında ilk başta görünmeyecek uygulamadaki işlemlere göre görünüm durumu belirlenecektir. Tasarım anında bu textView'un ekranda görünmediğinde diğer görünümlerin nasıl yerleştirileceğini belirlemek istiyorsak bu textView için tools:visibility="invisible" veya tam tersi durum için tools:visibility="visible" niteliklerini ekleyerek tasarım anında da görünürlük durumlarını kontrol edebilir ona göre tasarımlarımızı düzenleyebiliriz.