08.10.2025 Ders Notları

Meta Android Developer Professional Certificate - Jetpack Compose Listeler & Gridler Notları

Tarih: 08.10.2025
İçerdiği konular: Jetpack Compose'ta Listeler (Vertical/Horizontal), Grid yapıları, Lazy Layout'lar (LazyRow, LazyColumn, LazyGrid), Scrollable column, Arrangement tipleri

---

Listeler Temel Mantığı

Compose'da **liste** yapmak için iki temel composable var:

- **Row** → Yatay liste
- **Column** → Dikey liste

Little Lemon uygulamasında ortadaki yemek kategorileri horizontal list, yemeklerin listesi ise vertical list olarak yapılmış.

Arrangement denen bir şey var - bu liste içindeki elemanların nasıl yerleşeceğini belirliyor:

- `spaceBetween` → ilk eleman başta, son eleman sonda, diğerleri eşit aralıklı
- `spaceAround` → ilk ve son elemana da boşluk ekler, ortadakiler merkezde
- `spaceEvenly` → hepsine eşit boşluk (başta ve sonda da var)
- `Start`, `Center`, `End` → zaten belli

Bunu daha iyi anlamak için basit bir örnek yapalım:

```
@Composable
fun MenuCategory(category: String) {
    Button(
        colors = ButtonDefaults.buttonColors(backgroundColor = Color.LightGray),
        shape = RoundedCornerShape(40),
        modifier = Modifier.padding(5.dp)
    ) {
        Text(text = category)
    }
}
```

Sonra bunları listelemek için:

```
val categories = listOf("Salads", "Mains", "Desserts")

Row(
    modifier = Modifier.fillMaxWidth(),
    horizontalArrangement = Arrangement.SpaceEvenly
) {
    categories.forEach { category ->
        MenuCategory(category = category)
    }
}
```

Column ile yapınca da vertical list oluyor. Basitmiş aslında.

Grid Yapıları

Grid dediğimiz şey aslında iki boyutlu scrollable layout. Hem row hem column içeriyor. Her bir elemana **cell** deniyor.

Column'u scrollable yapmak için `verticalScroll(rememberScrollState())` modifier'ını kullanıyoruz. Bu scroll state'i hatırlıyor.

Grid yapmak için önce bir **GridCell** composable'ı oluşturuyoruz:

```
@Composable
fun GridCell() {
    Card(elevation = 16.dp) {
        Box(
            modifier = Modifier
                .size(100.dp, 80.dp)
                .padding(8.dp)
        ) {
            Image(
                painter = painterResource(R.drawable.greeksalad),
                contentDescription = "Greek Salad"
            )
            // Price ve title ekliyoruz
            Text(
                text = "Greek Salad",
                fontSize = 18.sp,
                fontWeight = Bold,
                modifier = Modifier.fillMaxWidth()
            )
        }
```

```
    }
}
```

Sonra bunları grid şeklinde düzenliyoruz:

```
Column(
    modifier = Modifier.verticalScroll(rememberScrollState())
) {
    repeat(5) { // 5 satır
        Row(
            modifier = Modifier.fillMaxWidth(),
            horizontalArrangement = Arrangement.SpaceEvenly
        ) {
            repeat(2) { // her satırda 2 eleman
                GridCell()
            }
        }
    }
}
```

Bu şekilde basit bir grid yapmış olduk. Scrollable da çalışıyor.

LAZY LAYOUT'lar (ÖNEMLİ!)

Burası kritik! Yukarıdaki row/column ile liste yapmak **küçük listeler** için iyi. Ama çok elemanlı listelerde performans sorunu yaşanıyor çünkü **bütün elemanlar aynı anda compose ediliyor**.

Çözüm: `LazyRow`, `LazyColumn`, `LazyGrid`

Bunların güzelliği: sadece **ekranda görünen elemanları** compose ediyor, diğerlerini etmiyor. Performans için çok önemli!

LazyColumn Örneği:

```
LazyColumn {
    item { // header
        Text("Header")
    }

    items(10) { index -> // 10 tane item
        Text("Item $index")
    }

    item { // footer
        Text("Footer")
```

```
    }
}
```

`itemsIndexed` kullanarak index'i de alabiliyoruz - mesela çift/tek index'leri farklı renk yapmak için kullanışlı.

Little Lemon Uygulamasında Kullanımı:

Önce data class'larımızı oluşturuyoruz:

```
// Data/Dish.kt
data class Dish(
    val name: String,
    val price: String,
    val description: String,
    val image: Int // Drawable resource
)

val dishes = listOf(
    Dish("Greek Salad", "$12.99", "Fresh vegetables...", R.drawable.greeksalad),
    // diğer yemekler...
)
```

Sonra LazyRow ve LazyColumn ile listeliyoruz:

```
@Composable
fun MenuListScreen() {
    Column {
        // Kategoriler - horizontal
        LazyRow {
            items(categories) { category ->
                MenuCategory(category = category)
            }
        }

        Divider(
            color = Color.LightGray,
            thickness = 1.dp,
            modifier = Modifier.padding(8.dp)
        )

        // Yemekler - vertical
        LazyColumn {
            items(dishes) { dish ->
                MenuDish(dish = dish)
```

```
            }
        }
    }
}
```

MenuDish composable'ında artık hard-coded değerler yerine `dish.name`, `dish.price` gibi değerleri kullanıyoruz.

Bu lazy layout'lar gerçekten hayat kurtarıyor, bunu not etmeliyim.

Lazy Grid

İki tip var:

- `LazyVerticalGrid` → dikey scroll, çoklu kolon
- `LazyHorizontalGrid` → yatay scroll, çoklu satır

Kolon ayarlama yöntemleri:

1. `GridCells.Fixed(3)` → 3 kolon sabit
2. `GridCells.Adaptive(150.dp)` → minimum 150dp genişlikte olacak şekilde otomatik kolon sayısı

Örnek:

```
LazyVerticalGrid(
    columns = GridCells.Fixed(3) // sabit 3 kolon
) {
    items(100) { index ->
        MyGridCell(number = index)
    }
}
```

Veya adaptive:

```
LazyVerticalGrid(
    columns = GridCells.Adaptive(150.dp) // min 150dp, otomatik kolon
) {
    items(500) { index ->
        MyGridCell(number = index)
    }
}
```

Adaptive daha responsive - yatay çevirince kolon sayısı artıyor, dikeyde azalıyor.

MyGridCell basit bir card + text:

```
@Composable
fun MyGridCell(number: Int) {
    Card(
        modifier = Modifier.padding(8.dp),
        elevation = 20.dp
    ) {
        Box(
            modifier = Modifier.aspectRatio(1f),
            contentAlignment = Alignment.Center
        ) {
            Text(
                text = "Item $number",
                fontSize = 20.sp
            )
        }
    }
}
```

Kişisel Notlarım

- **Lazy layout'lar performans için CRITICAL** - bunu asla unutmamalıyım
- Row/Column sadece küçük listeler için
- Arrangement tiplerini iyi öğrenmem lazım - UI/UX için önemli
- Grid yaparken adaptive vs fixed duruma göre seçmek önemli
- Little Lemon projesinde bunları mutlaka kullanacağım - pratik yapmak şart

Araştırmam gereken: Lazy layout'larda animation eklemek nasıl oluyor? Bunu daha sonra bakmalıyım.

Bu notlar dersten çalışırken kendi çıkardıklarım.