27 DE SEPTIEMBRE DE 2024

CÓMO CREAR LISTAS DINÁMICAS CON RECYCLERVIEW

SERGIO JUNIOR RUBÉNDUARTE VANEGAS

Cómo crear listas dinámicas con RecyclerView

RecyclerView facilita que se muestren de manera eficiente grandes conjuntos de datos. Tú proporcionas los datos y defines el aspecto de cada elemento, y la biblioteca RecyclerView creará los elementos de forma dinámica cuando se los necesite.

Como su nombre lo indica, RecyclerView *recicla* esos elementos individuales. Cuando un elemento se desplaza fuera de la pantalla, RecyclerView no destruye su vista. En cambio, reutiliza la vista para los elementos nuevos que se desplazaron y ahora se muestran en pantalla. RecyclerView mejora el rendimiento y la capacidad de respuesta de tu app. reduce el consumo de energía.

Clases clave

Varias clases trabajan juntas para compilar tu lista dinámica.

- **RecyclerView** es el ViewGroup que contiene las vistas correspondientes a tus datos. Es una vista en sí misma, así que agregas RecyclerView a tu diseño de la misma manera que agregarías cualquier otro elemento de la IU.
- Cada elemento individual de la lista está definido por un objeto *contenedor de vistas*. Cuando se lo crea, este contenedor no tiene datos asociados. Después de crearlo, la RecyclerView lo *vincula* a sus datos. Para definir el contenedor de vistas, debes extender RecyclerView.ViewHolder.
- El RecyclerView solicita vistas y las vincula a sus datos. llamando a métodos en el *adaptador.* Para definir el adaptador, extiende RecyclerView.Adapter.
- El *administrador de diseño* organiza los elementos individuales de tu lista. Puedes usar uno de los administradores de diseño proporcionados por la biblioteca RecyclerView o puedes definir el tuyo. Todos los administradores de diseño se basan en la clase abstracta LayoutManager de la biblioteca.

Pasos para implementar tu RecyclerView

Si vas a usar RecyclerView, debes realizar algunas acciones. Se explican en detalle en las siguientes secciones.

1. Decide el aspecto de la lista o cuadrícula. Por lo general, puedes usar uno de los administradores de diseño estándar de la biblioteca RecyclerView.

2. Diseña el aspecto y el comportamiento de cada elemento de la lista. En función de este diseño, extiende la clase ViewHolder. Tu versión de ViewHolder proporciona todas las funcionalidades para tus elementos de lista. El contenedor de vistas es un wrapper alrededor de una View, y RecyclerView la administra.
3. Define el Adapter que asocia tus datos con las vistas del ViewHolder.

Cómo planificar tu diseño

Los elementos de tu RecyclerView se organizan por una clase LayoutManager. La biblioteca RecyclerView proporciona tres administradores de diseño, que controlan las situaciones de diseño más comunes:

- LinearLayoutManager dispone los elementos en una lista unidimensional.
- GridLayoutManager organiza los elementos en una cuadrícula bidimensional:
    - Si la cuadrícula está organizada de forma vertical, GridLayoutManager intenta que todos los elementos de cada fila tengan el mismo ancho y la misma altura, pero las filas pueden tener alturas distintas.
    - Si la cuadrícula está organizada de forma horizontal, GridLayoutManager intenta que todos los elementos de cada columna tengan el mismo ancho y la misma altura, pero las columnas pueden tener anchos distintos.
- StaggeredGridLayoutManager es similar a GridLayoutManager, pero no requiere que los elementos de una fila tengan la misma altura (en cuadrículas verticales) o que los elementos de la misma columna tengan el mismo ancho (en cuadrículas horizontales). El resultado es que los elementos de una fila o columna pueden terminar compensándose entre sí.

También debes diseñar el diseño de los elementos individuales. Lo necesitas cuando diseñas el contenedor de vistas, como se describe en la siguiente sección.

Cómo implementar el adaptador y el contenedor de vistas

Una vez que determines el diseño, deberás implementar Adapter y ViewHolder Estas dos clases trabajan juntas para definir la forma en que se muestran tus datos. El ViewHolder es un wrapper alrededor de una View que contiene el diseño de un elemento individual de la lista. El elemento Adapter crea ViewHolder los objetos según sea necesario y también establece los datos para esas vistas. El proceso de asociar vistas con sus datos se denomina *vinculación.*

Cuando defines tu adaptador, anulas tres métodos clave:

- onCreateViewHolder(): RecyclerView Llama a este método siempre que necesita crear una ViewHolder nueva. El método crea y, luego, inicializa la ViewHolder y su View asociada, pero *no* completa el contenido de la vista; aún no se vinculó la ViewHolder con datos específicos.
- onBindViewHolder(): RecyclerView Llama a este método para asociar una ViewHolder con los datos. El método recupera los datos correspondientes y los usa para completar el diseño del contenedor de vistas. Por ejemplo, si RecyclerView muestra una lista de nombres, el método puede encontrar el nombre apropiado en la lista y rellenar la vista widget TextView del contenedor.
- getItemCount(): RecyclerView llama a este método para obtener el tamaño del conjunto de datos. Por ejemplo: en una aplicación de libreta de direcciones, esta podría ser la cantidad total de direcciones. RecyclerView lo usa para determinar cuándo no hay más elementos que se puedan que se muestra.

Se muestra un ejemplo típico de un adaptador simple con una ViewHolder anidada que muestra una lista de datos. En este caso, RecyclerView muestra una lista simple de elementos de texto. Al adaptador se le pasa una matriz de cadenas que contiene el texto para los elementos ViewHolder.

Kotlin Java

```kotlin
class CustomAdapter(private val dataSet: Array<String>) :
        RecyclerView.Adapter<CustomAdapter.ViewHolder>() {

    /**
     * Provide a reference to the type of views that you are using
     * (custom ViewHolder)
     */
    class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val textView: TextView

        init {
            // Define click listener for the ViewHolder's View
            textView = view.findViewById(R.id.textView)
        }
    }

    // Create new views (invoked by the layout manager)
    override fun onCreateViewHolder(viewGroup: ViewGroup, viewType: Int): ViewHolder {
        // Create a new view, which defines the UI of the list item
        val view = LayoutInflater.from(viewGroup.context)
                .inflate(R.layout.text_row_item, viewGroup, false)

        return ViewHolder(view)
    }

    // Replace the contents of a view (invoked by the layout manager)
    override fun onBindViewHolder(viewHolder: ViewHolder, position: Int) {

        // Get element from your dataset at this position and replace the
        // contents of the view with that element
        viewHolder.textView.text = dataSet[position]
    }

    // Return the size of your dataset (invoked by the layout manager)
    override fun getItemCount() = dataSet.size

}
```

El diseño de cada elemento de vista se define en un archivo de diseño XML, como de costumbre. En este caso, la app tiene un archivo `text_row_item.xml` como el siguiente:

```xml
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="@dimen/list_item_height"
    android:layout_marginLeft="@dimen/margin_medium"
    android:layout_marginRight="@dimen/margin_medium"
    android:gravity="center_vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/element_text"/>
</FrameLayout>
```

Próximos pasos

En el siguiente fragmento de código, se muestra cómo puedes usar el `RecyclerView`.

Kotlin Java

```kotlin
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val dataset = arrayOf("January", "February", "March")
        val customAdapter = CustomAdapter(dataset)

        val recyclerView: RecyclerView = findViewById(R.id.recycler_view)
        recyclerView.layoutManager = LinearLayoutManager(this)
        recyclerView.adapter = customAdapter

    }

}
```