EBook Gratis

APRENDIZAJE recyclerview

Free unaffiliated eBook created from
Stack Overflow contributors.

#recyclervie

w

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: recyclerview

It is an unofficial and free recyclerview ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official recyclerview.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con recyclerview

Observaciones

Esta sección proporciona una descripción general de qué es recyclerview y por qué un desarrollador puede querer usarlo.

También debe mencionar cualquier tema grande dentro de recyclerview, y vincular a los temas relacionados. Dado que la Documentación para recyclerview es nueva, es posible que deba crear versiones iniciales de esos temas relacionados.

Examples

Instalación y configuración

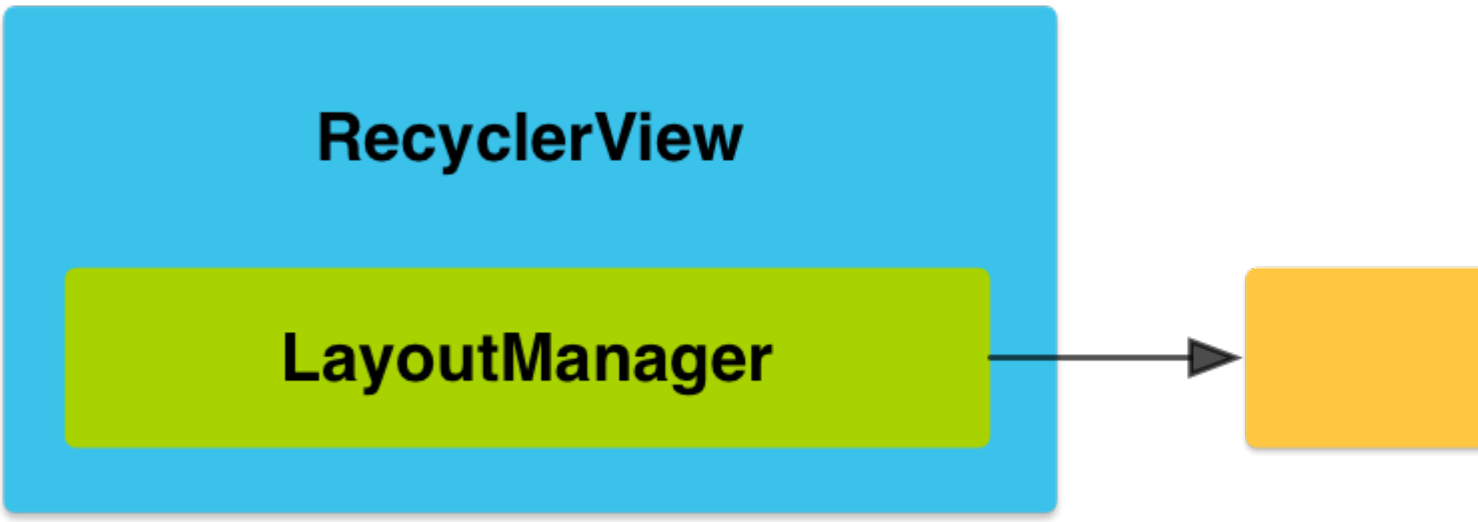
La visualización de elementos en una lista o cuadrículas es un patrón muy común en las aplicaciones móviles. El usuario ve una colección de artículos y puede desplazarse a través de ellos. La colección de elementos puede ser una lista, una cuadrícula u otra representación estructurada de datos.

El widget `RecyclerView` es una versión más avanzada y flexible de ListView . Este widget es un contenedor para mostrar grandes conjuntos de datos que se pueden desplazar de manera muy eficiente manteniendo un número limitado de vistas. Use el widget RecyclerView cuando tenga colecciones de datos cuyos elementos cambien en tiempo de ejecución según la acción del usuario o los eventos de la red.

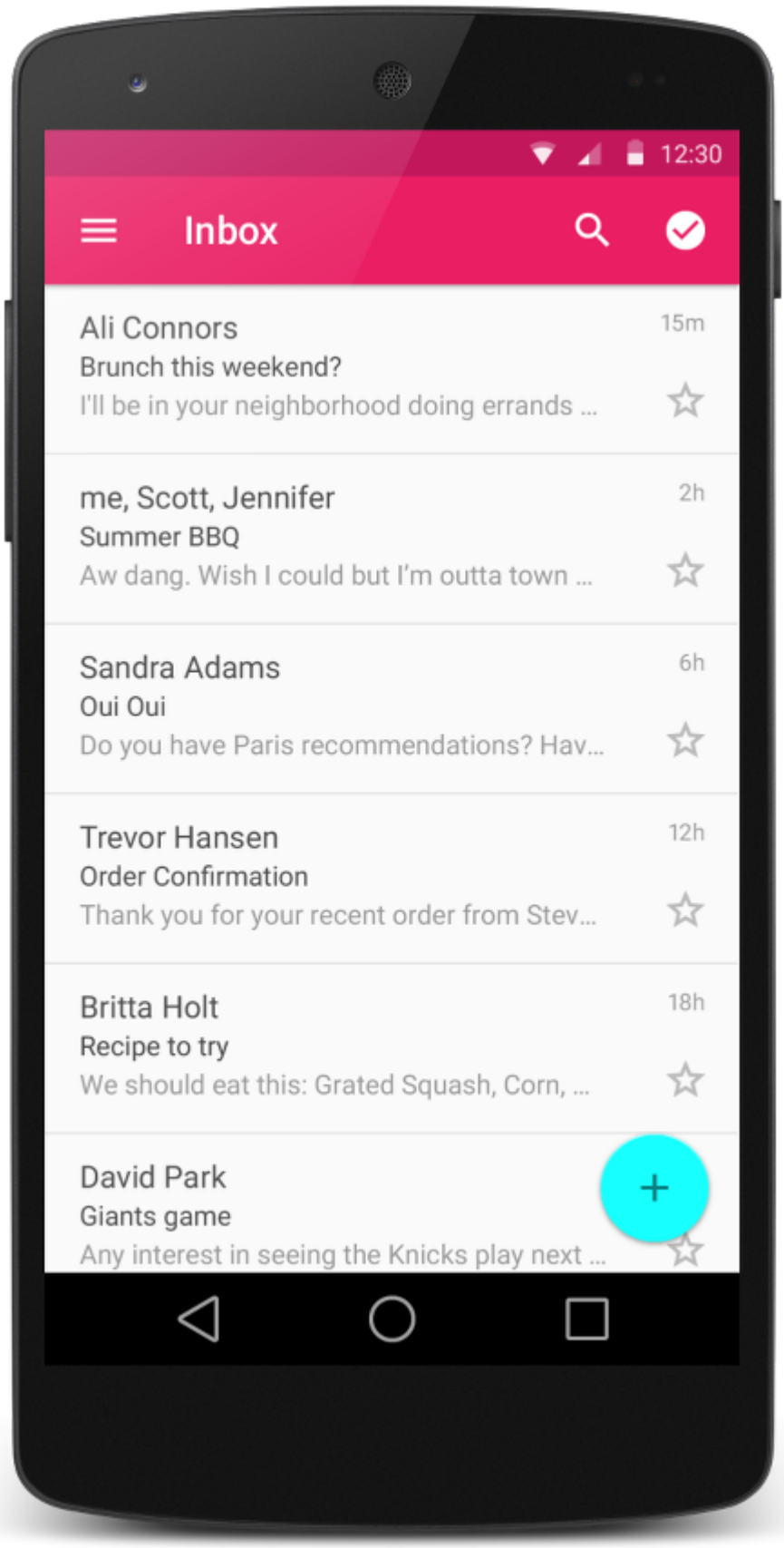
La clase `RecyclerView` simplifica la visualización y el manejo de grandes conjuntos de datos al proporcionar:

- Gestores de layout para posicionamiento de elementos.
- Animaciones predeterminadas para operaciones de elementos comunes, como la eliminación o adición de elementos

También tiene la flexibilidad de definir gestores de diseño personalizados y animaciones para los widgets `RecyclerView` .



Para utilizar el widget `RecyclerView` , debe especificar un adaptador y un administrador de diseño. Para crear un adaptador, extienda la clase `RecyclerView.Adapter` . Los detalles de la implementación dependen de los detalles de su conjunto de datos y del tipo de vistas.



Un administrador de diseño coloca las vistas de elementos dentro de `RecyclerView` y determina cuándo reutilizar las vistas de elementos que ya no son visibles para el usuario. Para reutilizar (o reciclar) una vista, un administrador de diseño puede pedirle al adaptador que reemplace el contenido de la vista con un elemento diferente del conjunto de datos. Reciclar las vistas de esta manera mejora el rendimiento al evitar la creación de vistas innecesarias o realizar búsquedas

costosas en findViewById () .

`RecyclerView` proporciona estos gestores de diseño integrados:

- LinearLayoutManager muestra los elementos en una lista de desplazamiento vertical u horizontal.
- GridLayoutManager muestra los elementos en una cuadrícula.
- StaggeredGridLayoutManager muestra los elementos en una cuadrícula escalonada.

Para crear un administrador de diseño personalizado, extienda la clase RecyclerView.LayoutManager .

Ahora para agregar un `RecyclerView` necesita agregar la siguiente dependencia en su `build.gradle`

```
dependencies {
    compile 'com.android.support:recyclerview-v7:23.4.0'
}
```

Ahora agregue un `RecyclerView` en su diseño como este.

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/my_recycler_view"
    android:scrollbars="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Una vez que haya agregado un widget `RecyclerView` a su diseño, obtenga un identificador para el objeto, conéctelo a un administrador de diseño y adjunte un adaptador para que se muestren los datos.

```
public class MyActivity extends Activity {
    private RecyclerView mRecyclerView;
    private RecyclerView.Adapter mAdapter;
    private RecyclerView.LayoutManager mLayoutManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.my_activity);
        mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);

        // use this setting to improve performance if you know that changes
        // in content do not change the layout size of the RecyclerView
        mRecyclerView.setHasFixedSize(true);

        // use a linear layout manager
        mLayoutManager = new LinearLayoutManager(this);
        mRecyclerView.setLayoutManager(mLayoutManager);

        // specify an adapter (see also next example)
        mAdapter = new MyAdapter(myDataset);
        mRecyclerView.setAdapter(mAdapter);
    }
    ...
}
```

El adaptador proporciona acceso a los elementos de su conjunto de datos, crea vistas para los elementos y reemplaza el contenido de algunas de las vistas con nuevos elementos de datos cuando el elemento original ya no está visible. El siguiente ejemplo de código muestra una implementación simple para un conjunto de datos que consiste en una serie de cadenas que se muestran con los widgets de `TextView` .

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {
    private String[] mDataset;

    // Provide a reference to the views for each data item
    // Complex data items may need more than one view per item, and
    // you provide access to all the views for a data item in a view holder
    public static class ViewHolder extends RecyclerView.ViewHolder {
        // each data item is just a string in this case
        public TextView mTextView;
        public ViewHolder(TextView v) {
            super(v);
            mTextView = v;
        }
    }

    // Provide a suitable constructor (depends on the kind of dataset)
    public MyAdapter(String[] myDataset) {
        mDataset = myDataset;
    }

    // Create new views (invoked by the layout manager)
    @Override
    public MyAdapter.ViewHolder onCreateViewHolder(ViewGroup parent,
                                                   int viewType) {
        // create a new view
        View v = LayoutInflater.from(parent.getContext())
                               .inflate(R.layout.my_text_view, parent, false);
        // set the view's size, margins, paddings and layout parameters
        ...
        ViewHolder vh = new ViewHolder(v);
        return vh;
    }

    // Replace the contents of a view (invoked by the layout manager)
    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        // - get element from your dataset at this position
        // - replace the contents of the view with that element
        holder.mTextView.setText(mDataset[position]);

    }

    // Return the size of your dataset (invoked by the layout manager)
    @Override
    public int getItemCount() {
        return mDataset.length;
    }
}
```

Primero agregue la biblioteca de soporte a su proyecto: dentro del archivo build.gradle de la aplicación, agregue la dependencia.

Lea Empezando con recyclerview en línea:
https://riptutorial.com/es/recyclerview/topic/5426/empezando-con-recyclerview

Creditos

S. No	Capítulos	Contributors
1	Empezando con recyclerview	Community, Princess Ruthie, Reaz Murshed