Documentación

Proyecto: Leer películas en cartelera

Iván Gallego

índice de contenidos

Main activity

En la activity main tenemos un método para mostrar el diálogo de barra de progreso También tenemos el método `onCreate` que inicializará las variables, adaptador, recycler... y finalmente, ejecutará la tarea asíncrona de lectura del XML

```
public class MainActivity extends AppCompatActivity {

    private ProgressBar progressBar;
    private RecyclerView recycler;
    private Dialog dialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recycler = findViewById(R.id.recycler);
        recycler.setAdapter(new Adaptador(new ArrayList<Pelicula>()));
        setDialog();
        LeerXML leer = new LeerXML(dialog, recycler);
        leer.execute();
        recycler.setLayoutManager(new LinearLayoutManager(
            this, RecyclerView.VERTICAL, false));
    }

    public void setDialog() {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        View v = getLayoutInflater().inflate(R.layout.progress_bar, null, false);
        progressBar = v.findViewById(R.id.indeterminateBar);
        progressBar.setIndeterminate(true);
        dialog = builder.create();
        dialog.setCancelable(false);
        dialog.show();
```

```
    }
}
```

Main activity XML

El layout de la actividad principal es la lista que contendrá cada una de las películas que se lean del XML

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".MainActivity"
    tools:showIn="@layout/activity_main">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"/>

</RelativeLayout>
```

Descargar imagen (AsyncTask)

La clase `DescargarImagen` nos permitirá leer una imágen de un URL y transoformarla en un Bitmap para mostrar en cada uno de los elementos de la lista. Posteriormente, llamaremos a esta tarea cuando creemos cada elemento. El funcionamiento principal, es pasarle un elemento ImageView al que le asignará un Bitmap cuando la tarea termine de descargarla imagen

```
public class DescargarImagen extends AsyncTask<String, Void, Bitmap> {
    private ImageView imageView;
    private URL url;

    public DescargarImagen(ImageView imageView) {
        this.imageView = imageView;
    }

    @Override
    protected Bitmap doInBackground(String... strings) {
        Bitmap b = null;
        try {
            url = new URL(strings[0]);
```

```
            b = BitmapFactory.decodeStream(url.openStream());
        } catch (Exception e) {
            e.printStackTrace();
        }
        return b;
    }

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

    @Override
    protected void onPostExecute(Bitmap bitmap) {
        super.onPostExecute(bitmap);
        imageView.setImageBitmap(bitmap);
    }
}
```

Leer XML (AsyncTask)

Esta tarea actualiza la lista de elementos del recycler al terminar la lectura del XML. Cuando se termina de leer una película, se añade a la lista. El constructor de esta clase recibe un `Dialog` y un `RecyclerView`. El recycler es para lo que antes he mencionado y el diálogo es para hacer un `cancel` cuando termine de leer el XML La URL es un campo de la clase, aunque si quisiéramos podríamos pasarle la imagen al propio constructor o a la tarea cambiando la clase de la que extiende a `AsyncTask<URL, Void, Void>` El campo que almacena el recycler es uno de tipo `WeakReference` por las posibles fugas de memoria. Lo único que hace esta clase es almacenar el recycler, para acceder a él simplemente llamas al método de `WeakReference.get`

```
public class LeerXML extends AsyncTask<Void, Void, Void> {
    private final WeakReference<RecyclerView> recycler;
    private URL url;
    private WeakReference<Dialog> dialog;
    private ArrayList<Pelicula> peliculas;

    public LeerXML(Dialog dialog, RecyclerView recyclerView) {
        this.dialog = new WeakReference<>(dialog);
        this.recycler = new WeakReference<RecyclerView>(recyclerView);
        try {
            this.url = new URL("http://rss.sensacine.com/cine/encartelera?
format=xml");
        } catch (MalformedURLException e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void onPostExecute(Void aVoid) {
        super.onPostExecute(aVoid);
```

```
            recycler.get().setAdapter(new Adaptador(peliculas));
        }

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

        @Override
        protected Void doInBackground(Void... voids) {
            Pelicula pelicula = null;
            XmlPullParser parser = Xml.newPullParser();
            boolean empezarLectura = false;
            try {
                URLConnection conn = url.openConnection();
                parser.setInput(conn.getInputStream(), "UTF-8");
                int evento = parser.getEventType();
                while (evento != XmlPullParser.END_DOCUMENT) {
                    switch (evento) {
                        case XmlPullParser.START_DOCUMENT:
                            peliculas = new ArrayList<>();
                            break;
                        case XmlPullParser.START_TAG:
                            if (!empezarLectura) {
                                empezarLectura = parser.getName().equals("item");
                            }
                            if (empezarLectura) {
                                switch (parser.getName()) {
                                    case "item":
                                        pelicula = new Pelicula();
                                        break;
                                    case "title":
                                        pelicula.setTitulo(parser.nextText());
                                        break;
                                    case "description":
                                        pelicula.setDescripcion(parser.nextText());
                                        break;
                                    case "enclosure":
                                        pelicula.setUrl(parser.getAttributeValue(null,
"url"));
                                        break;
                                }
                            }
                            break;
                        case XmlPullParser.END_TAG:
                            if (parser.getName().equals("item")) {
                                peliculas.add(pelicula);
                            }
                            break;
                    }
                    evento = parser.next();
                }
            } catch (Exception e) {
                e.printStackTrace();
```

```
            }
            dialog.get().cancel();
            return null;
        }
    }
```

Adaptador + Holder

Este es el adaptador y holder necesario para que funciones de manera adecuada el `RecyclerView` Lo único especial es que, al hacer `Bind` de los datos, se llama a la tarea asíncrona que nos permite descargar una imágen, pasándole el ImageView que contendrá dicha imagen descargada

```
public class Adaptador extends RecyclerView.Adapter<Adaptador.Holder>{
    ArrayList<Pelicula> peliculas;

    public Adaptador(ArrayList<Pelicula> peliculas) {
        this.peliculas = peliculas;
    }

    @NonNull
    @Override
    public Holder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View v =
LayoutInflater.from(parent.getContext()).inflate(R.layout.pelicula, parent,
false);
        Holder h = new Holder(v);
        return h;
    }

    @Override
    public void onBindViewHolder(@NonNull Holder holder, int position) {
        holder.bind(peliculas.get(position));
    }

    @Override
    public int getItemCount() {
        return peliculas.size();
    }

    public class Holder extends RecyclerView.ViewHolder {

        ImageView imagenPelicula;
        TextView titulo, descripcion;
        public Holder(@NonNull View itemView) {
            super(itemView);
            imagenPelicula = itemView.findViewById(R.id.imagenPelicula);
            titulo = itemView.findViewById(R.id.textoTitulo);
            descripcion = itemView.findViewById(R.id.descripcionPelicula);
        }

        public void bind(Pelicula pelicula) {
```

```
            new DescargarImagen(imagenPelicula).execute(pelicula.getUrl());
            titulo.setText(pelicula.getTitulo());
            descripcion.setText(pelicula.getDescripcion());
        }
    }
}
```

Layout barra progreso

El layout que contiene la barra de progreso no es más que una barra de progreso con el ID `android:id="@+id/indeterminateBar"`, para que cambie el tipo de barra a indeterminado

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <ProgressBar
        android:id="@+id/indeterminateBar"
        android:layout_width="80dp"
        android:layout_height="80dp"
        />

</LinearLayout>
```