CSCI3310 Mobile Computing & Application Development

Lab 04 – Dynamic UI: Endless Taste

Introduction

Letting the user display, scroll, and manipulate a list of similar data items is a common app feature. Examples of scrollable lists include music playlists, photo directories, todo lists, and indexes of documents.

On scrolling views, one can use `ScrollView` to scroll a `View` or `ViewGroup`. `ScrollView` is easy to use, but it's not recommended for long, scrollable lists.

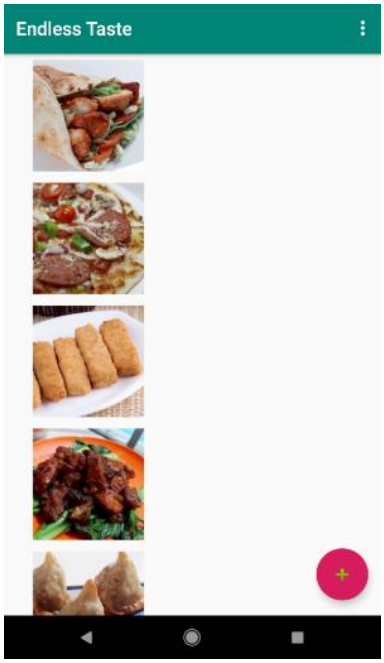
`RecyclerView` is a subclass of `ViewGroup` and is a more resource-efficient way to display scrollable lists. Instead of creating a `View` for each item that may or may not be visible on the screen, `RecyclerView` creates a limited number of list items and reuses them for visible content.

In this lab, you do the following:

1) Use `RecyclerView` to display a scrollable list. Add a click handler to each list item.

Objective

1) How to use the `RecyclerView` class to display items in a scrollable list.
2) How to dynamically add items to the `RecyclerView` as they become visible through scrolling.



Todo

1) Create a new app that uses a `RecyclerView` to display a list of items as a scrollable list.

The `RecyclerView` app demonstrates how to use a `RecyclerView` to display a long scrollable list of images. You create the dataset (the images), and the `RecyclerView` itself.

1: Create a new project and dataset

Before you can display a `RecyclerView`, you need data to display. In this task, you will create a new project for the app and a dataset. In a more sophisticated app, your data might come from internal storage (a file, SQLite database, saved preferences), from another app (Contacts, Photos), or from the internet (cloud storage, Google Sheets, or any data source with an API).

For this exercise, you will simulate data by creating it in the `onCreate()` method of `MainActivity`.

1.1. Create the project and layout

1) Start Android Studio and download a starter project with the name **EndlessTaste_Starter**, to start with.

2) Run your app. You should see the **EndlessTaste** app title and "Hello World" on the screen. If you encounter Gradle-related errors, sync your project as described in the lab on installing Android Studio and running Hello World.

1.2. Add code to create data

In this step, you import 5 images that named, as in ["image1.png", "image2.png", "image3.png", ...] into Drawable folder and create a `LinkedList` to store the image's path.

1) Open **MainActivity** and add a private member variable for the `mImagePathList` linked list.

```
public class MainActivity extends AppCompatActivity {
    private final LinkedList<String> mImagePathList = new LinkedList<>();
    private String mDrawableFilePath =
"android.resource://edu.cuhk.csci3310.endlesstaste/drawable/";
     // ... Rest of MainActivity code ...
}
```

2) Add code within the `onCreate()` method that populates `mImagePathList` with images:

```
// Put initial data into the image list.
for (int i = 1; i <= 5; i++) {
    mImagePathList.addLast(mDrawableFilePath + "image" + i);
}
```

The code concatenates the default drawable path with the string `"image"` and value of `i` while increasing its value. In other words, the list is storing the image's path for later access. This is the initial data you need as a dataset for this lab.

1.3. (Optional) Change the FAB icon

For this lab, you will use a FAB to generate a new image to insert into the list. The **Basic Activity** template provides a **FAB**, but you may want to change its icon. As you learned in another lesson, you can choose an icon from the set of icons in Android Studio for the FAB. Follow these steps:

1) Expand **res** in the **Project > Android** pane, and right-click (or Control-click) the **drawable** folder.

2) Choose **New > Image Asset**. The Configure Image Asset dialog appears.

3) Choose **Action Bar and Tab Icons** in the drop-down menu at the top of the dialog.

4) Change **ic_action_name** in the **Name** field to **ic_add_for_fab**.

5) Click the clip art image (the Android logo next to **Clipart:**) to select a clip art image as the icon. A page of icons appears. Click the icon you want to use for the FAB, such as the plus (**+**) sign.

6) Choose **HOLO_DARK** from the **Theme** drop-down menu. This sets the icon to be white against a dark-colored (or black) background. Click **Next**.

7) Click **Finish** in the Confirm Icon Path dialog.

8) Modify **activity_main.xml** to use the new icon by adding:

```
android:src="@drawable/ic_add_for_fab "
```

2: Create a RecyclerView

In this task, you display data in a `RecyclerView`. You need the following:

1) Data to display: Use the `mImagePathList`.

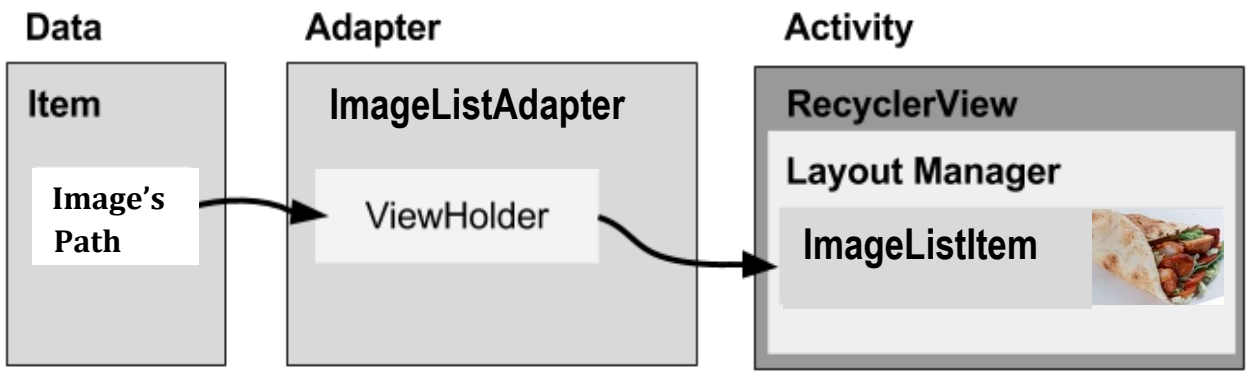
2) A `RecyclerView` for the scrolling list that contains the list items. Layout for one item of data. All list items look the same.

3) A layout manager. `RecyclerView.LayoutManager` handles the hierarchy and layout of `View` elements. `RecyclerView` requires an explicit layout manager to manage the arrangement of list items contained within it. This layout could be vertical, horizontal, or a grid. You will use a vertical `LinearLayoutManager`.

4) An adapter. `RecyclerView.Adapter` connects your data to the `RecyclerView`. It prepares the data in a `RecyclerView.ViewHolder`. You will create an adapter that inserts into and updates your generated images in your views.

5) A `ViewHolder`. Inside your adapter, you will create a `ViewHolder` that contains the `View` information for displaying one item from the item's layout.

The diagram below shows the relationship between the data, the adapter, the `ViewHolder`, and the layout manager.



To implement these pieces, you will need to:

6) Add a `RecyclerView` element to the `MainActivity` XML content layout (`content_main.xml`) for the EndlessTaste app.
7) Create an XML layout file (`imagelist_item.xml`) for one list item, which is `ImageListItem`.
8) Create an adapter (`ImageListAdapter`) with a `ViewHolder` (`ImageViewHolder`). Implement the method that takes the data, prepare it in the `ViewHolder`, and lets the layout manager know to display it.
9) In the `onCreate()` method of `MainActivity`, create a `RecyclerView` and initialize it with the adapter and a standard layout manager.

Let's do these one at a time.

2.1. Modify the layout in content_main.xml

To add a `RecyclerView` element to the XML layout, follow these steps:

1) Open **content_main.xml** in your EndlessTaste app. It shows a "Hello World" `TextView` at the center of a `ConstraintLayout`.
2) Click the **Text** tab to show the XML code.
3) Replace the entire `TextView` element with the following:

```
<androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerview"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

You need to specify the full path (`androidx.recyclerview.widget.RecyclerView`) because `RecyclerView` is part of the AndroidX Library.

2.2. Create the layout for one list item

The adapter needs the layout for one item on the list. All the items use the same layout. You need to specify that list item layout in a separate layout resource file, because it is used by the adapter, separately from the `RecyclerView`.

Create a simple image item layout using a vertical `LinearLayout` with an `ImageView`:

1) Right-click the **app > res > layout** folder and choose **New > Layout resource file**.
2) Name the file **imagelist_item** and click OK.
3) In the new layout file, click the **Text** tab to show the XML code.
4) Change the `ConstraintLayout` that was created with the file to a `LinearLayout` with the following attributes (extract resources as you go):

   LinearLayout Attribute Value

android:layout_width	"match_parent"
android:layout_height	"wrap_content"
android:orientation	"vertical"
android:padding	"6dp"

5) Add an ImageView for the image to the LinearLayout. Use image as the ID of the image:

Attribute **Value**

```
android:id  "@+id/image"
android:layout_width "match_parent"
android:layout_height "120dp"
android:layout_marginLeft="8dp"
android:layout_marginRight="240dp"
app:srcCompat="@drawable/image1"
```

2.3 Create a style from the ImageView attributes

You can use styles to allow elements to share groups of display attributes. An easy way to create a style is to extract the style of a UI element that you already created. To extract the style information for the ImageView in imagelist_item.xml:

1) Open **imagelist_item.xml** if it is not already open.
2) Right-click (or Control-click) the ImageView you just created in imagelist_item.xml, and choose **Refactor > Extract > Style**.

The Extract Android Style dialog appears.

3) Name your style **image_title** and leave all other options selected. Select the **Launch 'Use Style Where Possible'** option. Then click **OK**.
4) When prompted, apply the style to the **Whole Project**.
5) Find and examine the image_title style in **values > styles.xml**.
6) Reopen **imagelist_item.xml** if it is not already open.

The ImageView now uses the style in place of individual styling properties, as shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android=http://schemas.android.com/apk/res/android
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:orientation="vertical"
     android:padding="6dp">
    <ImageView
         android:id="@+id/image"
         app:srcCompat="@drawable/image1"
         style="@style/image_title" />
</LinearLayout>
```

2.4. Create an adapter

Android uses adapters (from the Adapter class) to connect data with View items in a list. There are many different kinds of adapters available, and you can also write custom adapters.

In this task, you will create an adapter that associates your list of image paths with image list `View` items.

To connect data with `View` items, the adapter needs to know about the `View` items. The adapter uses a `ViewHolder` that describes a `View` item and its position within the `RecyclerView`.

First, you will build an adapter that bridges the gap between the data in your image list and the `RecyclerView` that displays it:

1) Right-click java/edu.cuhk.csci3310.endlesstaste and select New > Java Class.
2) Name the class **ImageListAdapter**.
3) Give `ImageListAdapter` the following signature:

```
public class ImageListAdapter extends
    RecyclerView.Adapter<ImageListAdapter.ImageViewHolder>  {}
```

`ImageListAdapter` extends a generic adapter for `RecyclerView` to use a `View` holder that is specific for your app and defined inside `ImageListAdapter`. `ImageViewHolder` shows an error, because you have not yet defined it.

4) Click the class declaration (**ImageListAdapter**), then click the red-light bulb on the left side of the pane. Choose **Implement methods**.

A dialog appears that asks you to choose which methods to implement. Choose all three methods and click **OK**.

Android Studio creates empty placeholders for all the methods. Note how `onCreateViewHolder` and `onBindViewHolder` both reference the `ImageViewHolder`, which hasn't been implemented yet. You can filter only "Error" for display.

2.5 Create the ViewHolder for the adapter

To create the `ViewHolder`, follow these steps:

1) Inside the `ImageListAdapter` class, add a new `ImageViewHolder` inner class with this signature:

```
class ImageViewHolder extends RecyclerView.ViewHolder {}
```

You will see an error about a missing default constructor. You can see details about the errors by hovering your mouse cursor over the red-underlined code or over any red horizontal line on the right margin of the editor pane.

2) Add variables to the `ImageViewHolder` inner class for the `ImageView` and the adapter:

```
public final ImageView imageItemView;
 final ImageListAdapter mAdapter;
```

3) In the inner class `ImageViewHolder`, add a constructor that initializes the `ViewHolder` `ImageView` from the `image` XML resource, and sets its adapter:

```
public ImageViewHolder(View itemView, ImageListAdapter adapter) {
    super(itemView);
    imageItemView = itemView.findViewById(R.id.image);
    this.mAdapter = adapter;
}
```

4) Run your app to make sure that you have no errors. Your will still see only a blank view.

5) Click the **Logcat** tab to see the **Logcat** pane, and note the E/RecyclerView: No adapter attached; skipping layout warning.

You will attach the adapter to the `RecyclerView` in another step.

2.6 Storing your data in the adapter

You need to hold your data in the adapter, and `ImageListAdapter` needs a constructor that initializes the image path list from the data. Follow these steps:

1) To hold your data in the adapter, create a private linked list of strings in `ImageListAdapter` and call it `mImagePathList`.

```
private final LinkedList<String> mImagePathList;
```

2) You can now fill in the `getItemCount()` method to return the size of `mImagePathList`:

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

`ImageListAdapter` needs a constructor that initializes the image list from the data. To create a `View` for a list item, the `ImageListAdapter` needs to inflate the XML for a list item. You use a *layout inflator* for that job. `LayoutInflator` reads a layout XML description and converts it into the corresponding `View` items.

3) Create a member variable for the inflater in `ImageListAdapter`:

```
private LayoutInflater mInflater;
```

4) Implement the constructor for `ImageListAdapter`.

The constructor needs to have a context parameter, and a linked list of images with the app's data. The method needs to instantiate a `LayoutInflator` for `mInflater` and set `mImagePathList` to the passed in data:

```
public ImageListAdapter(Context context,
                        LinkedList<String> imagePathList) {
    mInflater = LayoutInflater.from(context);
    this.mImagePathList = imagePathList;
}
```

5) Fill out the `onCreateViewHolder()` method with this code:

```
@Override
public ImageViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
   View mItemView = mInflater.inflate(R.layout.imagelist_item, parent, false);
   return new ImageViewHolder(mItemView, this);
}
```

The `onCreateViewHolder()` method is similar to the `onCreate()` method. It inflates the item layout, and returns a `ViewHolder` with the layout and the adapter.

6) Fill out the `onBindViewHolder()` method with the code below:

```
@Override
public void onBindViewHolder(ImageViewHolder holder, int position) {
  String mImagePath = mImagePathList.get(position);
  Uri uri = Uri.parse(mImagePath);
  holder.imageItemView.setImageURI(uri);
}
```

The `onBindViewHolder()` method connects your data, particularly prepare the image from the image path to the view holder.

7) Run your app to make sure that there are no errors. You will still see only a blank view until the `RecyclerView` is updated.

2.7. Create the RecyclerView in the Activity

Now that you have an adapter with a `ViewHolder`, you can finally create a `RecyclerView` and connect all the pieces to display your data.

1) Open `MainActivity` and add member variables for the `RecyclerView` and the adapter.

```
private RecyclerView mRecyclerView;
private ImageListAdapter mAdapter;
```

2) In the `onCreate()` method of `MainActivity`, add the following code that creates the `RecyclerView` and connects it with an adapter and the data.

The comments explain each line. You must insert this code *after* the `mImagePathList` initialization.

```
  // Get a handle to the RecyclerView.
  mRecyclerView = findViewById(R.id.recyclerview);
  // Create an adapter and supply the data to be displayed.
  mAdapter = new ImageListAdapter(this, mImagePathList);
  // Connect the adapter with the RecyclerView.
  mRecyclerView.setAdapter(mAdapter);
  // Give the RecyclerView a default layout manager.
  mRecyclerView.setLayoutManager(new LinearLayoutManager(this));
```

3) Run your app.

You should see your list of images displayed, and you can scroll the list.

References

Android developer documentation:

1) RecyclerView
2) LayoutInflator
3) RecyclerView.LayoutManager
4) LinearLayoutManager

5) GridLayoutManager
6) StaggeredGridLayoutManager
7) CoordinatorLayout
8) ConstraintLayout
9) RecyclerView.Adapter
10) RecyclerView.ViewHolder
11) Create a list with RecyclerView

Video: RecyclerView Animations and Behind the Scenes (Android Dev Summit 2015)