CSCI3310 Mobile Computing & Application Development

Lab 05 – Dynamic UI: Endless Taste – Part II

Introduction

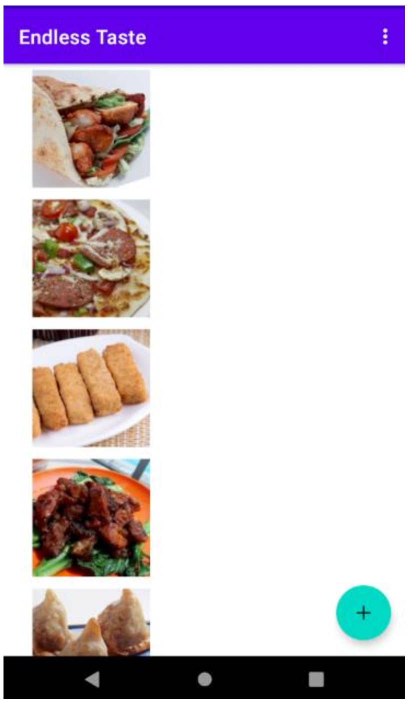
`RecyclerView` is a subclass of `ViewGroup` and is a more resource-efficient way to display scrollable lists. Instead of creating a `View` for each item that may or may not be visible on the screen, `RecyclerView` creates a limited number of list items and reuses them for visible content.

In this lab, you do the following:

1) Add items to the list using a floating action button (FAB), the pink button in the screenshot in the app overview section.
2) Use a FAB for the primary action that you want the user to take.

Objective

1) How to perform an action in a `RecyclerView` when the user taps a specific item.
2) How to show a FAB and perform an action of adding an image from the gallery when the user taps it.



Todo

1) Associate click behavior with the list items of the `RecyclerView`.
2) In the `RecyclerView`, Use a FAB to let the user add items to the `RecyclerView`.

1: Make the list interactive

Looking at lists of items is interesting, but it's a lot more fun and useful if your user can interact with them. To see how the `RecyclerView` can respond to user input, you will attach a click handler to each item. When the item is tapped, the click handler is executed, and that item's image will be gone.

1.1. Start the project

Start Android Studio and download a starter project with the name **EndlessTaste_Midway**, which the partially completed project from the last lab.

1.2. Make items respond to clicks

1) Open ImageListAdapter.
2) Change the `ImageViewHolder` class signature to implement `View.onClickListener`:

```
class ImageViewHolder extends RecyclerView.ViewHolder
                               implements View.OnClickListener {
```

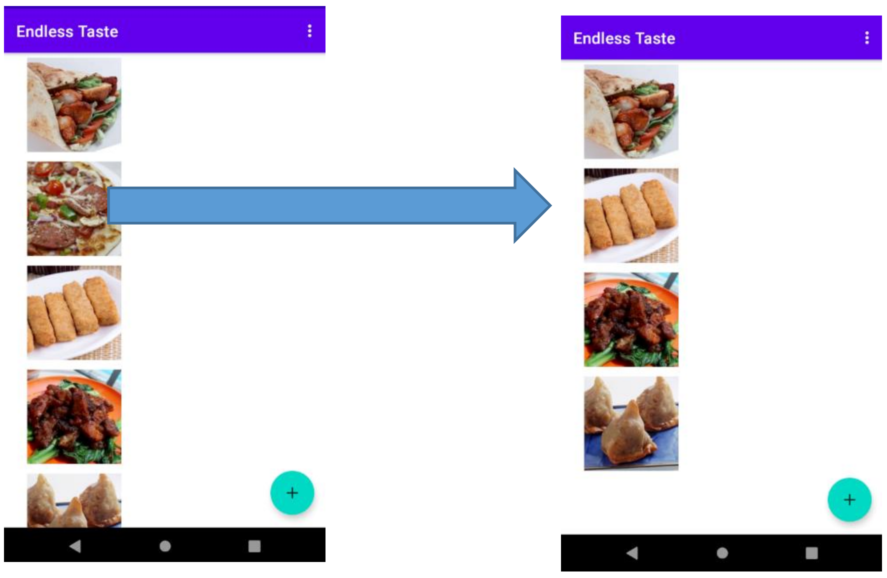
3) Click the class header and on the red light bulb to implement stubs for the required methods, which in this case is just the `onClick()` method.
4) Add the following code to the body of the `onClick()` method.

```
  // Get the position of the item that was clicked.
  int mPosition = getLayoutPosition();
  // Delete the image in the mImagePathList.
  mImagePathList.remove(mPosition);
  // Notify the adapter, that the data has changed so it can
  // update the RecyclerView to display the data.
  mAdapter.notifyDataSetChanged();
```

5) Connect the `onClickListener` with the `View`. Add this code to the `ImageViewHolder` constructor (below the `this.mAdapter = adapter` line):

```
  itemView.setOnClickListener(this);
```

6) Run your app. Click items to see the image list change.



1.3 Add behavior to the FAB

The list of items that the `RecyclerView` displays can also be modified dynamically—it doesn't have to be a static list. There are several ways to add additional behaviors. One is to use the floating action button (FAB). For example, in Gmail, the FAB is used to compose a new email. For this lab, you will generate a new image to insert into the list. For a more useful app, you would get data from your users.

In this task you will implement an action for the FAB to add an image to the end of the list of images and notify the adapter that the data has changed. Scroll to the inserted item.

Follow these steps:

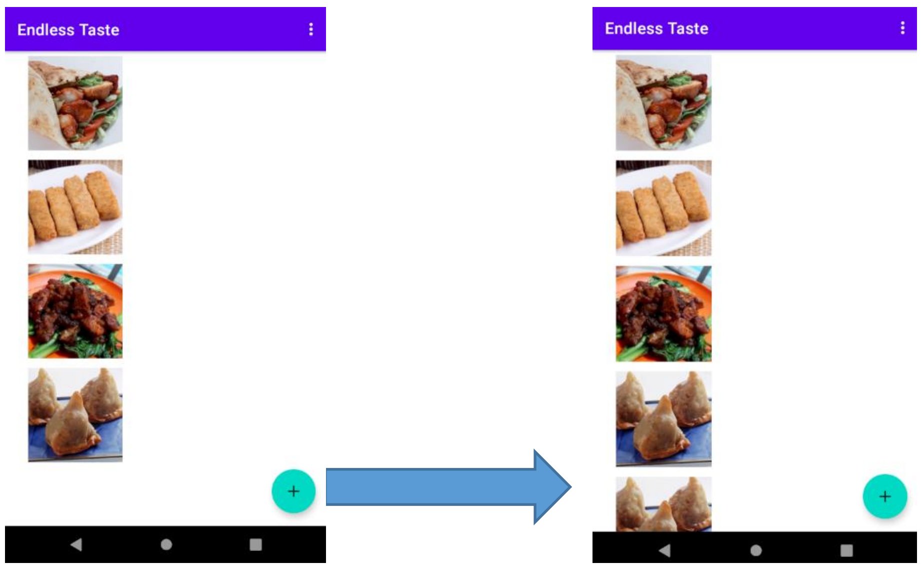
1) Open `MainActivity`. Modify the `onCreate()` method to set an `OnClickListener()` to the `FloatingActionButton` with an `onClick()` method for taking an action. Implement the `onClick()` method to the following:

```
fab.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
    int imageListSize = mImagePathList.size();
    // Add a new image to the imageList, recycle the old images as new ones.
    // mImagePathList[5] = mImagePathList[0], mImagePathList[6] =
    // mImagePathList[1],... etc
    mImagePathList.addLast(mDrawableFilePath + "image" + (imageListSize%5+1) );
    // Notify the adapter, that the data has changed.
    mRecyclerView.getAdapter().notifyItemInserted(imageListSize);
    // Scroll to the bottom.
    mRecyclerView.smoothScrollToPosition(imageListSize);
  }
});
```

2) Run the app.

3) Scroll the list of images and click items to delete.

4) Add items by clicking the FAB.



1.4. Pick an image from the gallery in Android

In this task, you will get an image from your phone gallery. You use **implicit intents** to open up the image gallery and get the image URI.

If you use the emulator, you may take a random photo using the default camera on the virtual device for putting some images into the gallery.

1) Add Permission in Android Manifest below the `<application>`.

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

2) Open **MainActivity** and add a private member variable:

```
public static final int REQUEST_GET_SINGLE_FILE = 1;
```

3) Modify the `onClick()` of the fab, start `startActivityForResult` as follows:

```
Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
intent.addCategory(Intent.CATEGORY_OPENABLE);
intent.setType("image/*");
startActivityForResult(Intent.createChooser(intent, "Select
Picture"),REQUEST_GET_SINGLE_FILE);
```

Above code, the segment is used to choose an image from Gallery.

4) Handle the data and migrate the code for adding a new image to the end of the list of images into the method `OnActivityResult()`.

Once you have selected an image, this method will be called.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  try {
    if (resultCode == RESULT_OK) {
      if (requestCode == REQUEST_GET_SINGLE_FILE) {
        // Get the path from the intent and store in imagePathList
        final String selectedImagePath = String.valueOf(data.getData());
        int imageListSize = mImagePathList.size();
        if (selectedImagePath != null) {
          // Add a new image path to the imagePathList
          mImagePathList.addLast(selectedImagePath);
          // Notify the adapter, that the data has changed.
          mRecyclerView.getAdapter().notifyItemInserted(imageListSize);
          // Scroll to the bottom.
          mRecyclerView.smoothScrollToPosition(imageListSize);
        }
      }
    }
  } catch (Exception e) {
    Log.e("FileSelectorActivity", "File select error", e);
  }
}
```

1.5. (Optional) Add Reset to the options menu

Change the options menu to show only one option: **Reset**. This option should return the list of images to its original state, with nothing clicked and no extra images.

Here, you are invited to explore how to implement options menu on your own, some key steps are highlighted below

1. Create a new resource directory and create a new XML menu resource (menu_main.xml)
2. Inflate the menu by overriding onCreateOptionsMenu() in your Activity
3. Exploit the onClick attribute in XML or overriding onOptionsItemSelected() in your Activity
4. Implement the method(s) to handle the item click

All in all, happy App writing!

References

Android developer documentation:

1) RecyclerView
2) View.onClickListener
3) MediaStore
4) ContentResolver
5) Floating Action Button
6) Menus
7) Menu Resource