Mobile App Development

Exercises: Todo List Toolbar App

The Todo List App exercise implemented a base App with the TodoListActivity and TodoActivity controllers and their respective fragments. This adhered to the principle of dynamically building the view with fragment transations. It also demonstrated good practice by using a Bundle for the `todoId` in the fragment rather than accessing the `intent` directly in the Activity; hence, decoupling the fragment with its own `argument bundle` and therby making it reusable.

This exercise add a new Toolbar to the App.

You could either replicate the `TodoListApp` or create a new git branch to the existing `TodoListApp` implementation. If in doubt, then recreating the App may be good practice.

The source code can be found at https://github.com/ebbi/TodoListApp/. Please use this as reference only. The following instructions will build the same git branch leading to the final prototype App with a toolbar.

See the following documentation on the toolbar
https://developer.android.com/training/appbar/setting-up.html

The toolbar will add an up button and a plus sign for adding new Todos as depicted in the following images.

list view dettail view

Todo List Toolbar App

Creating the toolbar is similar to any view with events and event handlers. It is the XML definition of the menu and the controller type code that handles events when menu items are selected

Create a git branch for implementing the Toolbar

```
git checkout master
git status

git branch toolbar
git checkout toolbar
git status
```

The `Toolbar` has been ported to the `AppCompat` library, the project default. To display views,the `AppCompat` requires a `theme` and it provides three such themes:

- `Theme.AppCompat`
- `Theme.AppCompat.light`
- `Theme.AppCompat.light.DarkActionBar`

Open `manifests/AndrodManifest.xml` to see the current theme; by default, it should be: `android:theme="@style/AppTheme">`. The `AppTheme` is defined in `res/values/styles.xml`. Open this file and note the definition: `<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">`

Menu

`Menu items` can be defined to perform an action such as create a new `todo`. Similar to creating any view, first add the string resources

```
<string name="new_todo">New Todo
```

Menus are a resource type and defined in XML files which reide in the `res/menu` directory.

```
Right-mouse click on the res directory and select New > Android resource file
Change the Resource type to Menu
and the name to, fragment_todo_list
Click OK

Edit the res/menu/fragment_todo_list.xml
and insert the following Add new todo menu item

<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/new_todo"
        android:icon="@drawable/ic_menu_add"
        android:title="@string/new_todo"
        app:showAsAction="ifRoom|withText" />
</menu>

Notice the xmlns tag defines a new namespace, app,
seperate from the usual android namespace
and it is then used to specify the showAsAction; this is for legacy reasons.

Next, there is the @drawable/ic_menu_add used in the menu item
needs to be created.

Right-mouse click on the drawable directory
Select New > Image Asset
Icon Type: Action Bar and Tab Icons
Name: ic_menu_add
Asset Type: Clip Art
Click on the Android Icon
In the pop up window select the Add ( + ) icon
Click OK
Next
Notice the set of icon files will now be created
Select Finish
```

The `Fragment` class has a set of callbacks to manage menus.

```
First inflate the view and check it displays the plus icon

Edit TodoListFragment jave file

Add the following to the end of the onCreate method:

setHasOptionsMenu(true);

And overide the onCreateOptionsMenu( ... ) with the following code:

@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    super.onCreateOptionsMenu(menu, inflater);
    inflater.inflate(R.menu.fragment_todo_list, menu);
}

Run the App
```

Responding to Menu selection

`Fragments` have a callback method `onOptionsItemSelected(MenuItem item)`. The `MenuItem` file currently only has a single `item` with the `new_todo` id. There could of course be more items. The one `item` can be implemented in a `case` statement.

Edit `TodoListFragment` java class and insert the following override method.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()){
        case R.id.new_todo:

            Todo todo = new Todo();
            TodoModel.get(getActivity()).addTodo(todo);

            Intent intent = TodoActivity.newIntent(getActivity(), todo.getId());
            startActivity(intent);

            return true;

        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Notice, `true` is returned to indicate no further processing is necessary. Run the App and add a few new Todos.

Enabling the Up button

The up button will navigate one level up in the App heirarchy to the the `TodoListActivity`.

Edit the `manifests/AndroidManifest.xml` file and update the activity definition for the `TodoActivity` with the following defintion:

```
<activity
    android:name=".TodoActivity"
    android:parentActivityName=".TodoListActivity">
</activity>


Run the App and try the arrow Up button
```

Submit and merge the toolbar git branch

```
git add .
git commit -am "Toolbar complete"

git checkout master
git merger toolbar
git status
```