UNIVERSITY OF PLYMOUTH

COMP2000: Software engineering 2

Outline

- Shared preferences
- Re-using Layouts
- Using a camera

Shared preferences

- **Android** provides options for Storing data using **Shared Preferences**.
- **Shared Preferences** is the way in which one can store and retrieve small amounts of primitive data as key/value pairs to a file on the device storage such as: String, int, float, boolean that make up your preferences in an XML file inside the app on the device storage.
- It suitable to be used in different situation.
- **For example**, when the user's settings need to be saved or to store data that can be used in different activities within the app

Create Shared preferences

you need to use the SharedPreferences APIs. A SharedPreferences object points to a file containing key-value pairs and provides simple methods to read and write them. Each SharedPreferences file is managed by the framework and can be private or shared.

You can save something in the sharedpreferences by using SharedPreferences.Editor class. You will call the edit method of SharedPreference instance and will receive it in an editor object.

You could pass the keys and values you want to write with methods such as putInt() and putString().
Then call apply() or commit() to save the changes.

Syntax:

```
Editor editor = sharedpreferences.edit();

 editor.putString("key", "value");
editor.commit();
```

- We need to check the values or data stored in shared preferences in the **onResume** method.
- if we got the data, we will start from the where the user left it.
- If there is no data, we need to start the application from the beginning as it is newly installed.

Example:

```
sharedpreferences = getSharedPreferences(MyPREFERENCES, Context.MODE_PRIVATE);
```

```
SharedPreferences.Editor editor = sharedpreferences.edit();

editor.putString(Name, name);
editor.putString(Phone, phone);
editor.putString(Email, email);
editor.commit();
```

Read from shared preferences

To retrieve values from a shared preferences file, call methods such as getInt() and getString(), providing the key for the value you want, and optionally a default value to return if the key isn't present.

Example:

This help access your preferences privately

```
SharedPreferences sharedPref = this.getPreferences(Context.MODE_PRIVATE);
int defaultValue = getResources().getInteger(R.integer.saved_high_score_default_key);
int highScore = sharedPref.getInt(getString(R.string.saved_high_score_key), defaultValue);
```

Re- using Layouts

- In Android, you could create a layout and re-use it in your application instead of creating it every single time you need it.

- To efficiently reuse complete layouts, you can use the <include> and <merge> tags to embed another layout inside the current layout.
- Reusing layouts is particularly powerful as it allows you to create reusable complex layouts.

Create a re-usable Layout

Create a new XML file and define the layout.
For example, here's a layout that defines a title bar to be included in each activity (titlebar.xml):

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/cardview_dark_background"
    tools:showIn="@layout/activity_main" >

    <ImageView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_launcher_background" />
</FrameLayout>
```

Use the <include> tag

Inside the layout to which you want to add the reusable component, add the <include> tag. For example, here's a layout that includes the title bar from above:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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"
    tools:context=".SharedPreferences">
    <include layout="@layout/titlebar"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

You can also override all the layout parameters (any android:layout_* attributes) of the included layout's root view by specifying them in the <include> tag.

Example:

```
<include android:id="@+id/topic_title"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    layout="@layout/titlebar"/>
```

However, if you want to override layout attributes using the <include> tag, you must override both android:layout_height and android:layout_width in order for other layout attributes to take effect.

Use the <merge> tag

The <merge> tag helps eliminate redundant view groups in your view hierarchy when including one layout within another.

To avoid including a redundant view group, you can instead use the <merge> element as the root view for the reusable layout.

Example:

```
<merge xmlns:android="http://schemas.android.com/apk/res/android">

   <Button
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="add"/>

   <Button
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="delete"/>

</merge>
```

Using a camera

- Android provides a way for the user to capture a photo, by simply request an existing camera app to capture a photo and return it to you.

Request the camera feature

Your application needs to use permission if it depends on camera, put a <uses-feature> tag in your manifest file:

```
<uses-feature
  android:name="android.hardware.camera.any"
  android:required="true" />
<uses-feature
```

If your application uses camera, but does not depends mainly on it, i.e. does not require a camera in order to function, you could set android:required to false.

This will allow, Google Play, devices to download your application without a camera.

So, you need to check for the availability of the camera at runtime by calling hasSystemFeature(PackageManager.FEATURE_CAMERA_ANY).

If a camera is not available, you should then disable your camera features.

For using a camera, simply, you could dispatch Intent to take a picture or record a video, check the examples in the next slides:

Take a picture:

```
private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    try {
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    } catch (ActivityNotFoundException e) {
        Toast.makeText(this, "Camera is not working", Toast.LENGTH_SHORT).show();
    }
}
```

Dispatch a video Intent

```
private void dispatchTakeVideoIntent() {
    Intent takeVideoIntent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
    if (takeVideoIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takeVideoIntent, REQUEST_VIDEO_CAPTURE);
    }
    else {
        Toast.makeText(this, "Video is not recording", Toast.LENGTH_SHORT).show();
    }
}
```

NOTE: StartActivityForResults deprecated

Alternatively use:
ActivityResultCallback<ActivityResult>()

For more details visit here:
https://developer.android.com/training/basics/intents/result

Thank you