ALGONQUIN COLLEGE

CST2335 GRAPHICAL INTERFACE PROGRAMMING

Week 4

Application Lifecycle (2)

Got data?

- There are 2 ways to go to a new Activity:
  - startActivity(Intent)
  - ActivityResultLauncher
- The difference is for if you are expecting the next activity to send data back or not.
- If you use ActivityResultLauncher, it will call onActivityResult() when it goes back to the previous activity, and it sends back a ResultActivity object from which you can extract your data.

Sending data back to previous page

- We can also start another activity and receive a result back.
- For example, we start a camera app and receive the captured photo as a result. Or start the Contacts app to allow the user to select a contact.
- While the underlying **startActivityForResult()** and **onActivityResult()** APIs are available on the Activity class on all API levels, Android strongly recommends to use the Activity Result APIs introduced in AndroidX **Activity** and **Fragment**.
- The Activity Result APIs provide components for registering for a result, launching the result, and handling the result once it is dispatched by the system.

Sending data back to previous page

- onActivityResult(ActivityResult result)
- You can extract information from ActivityResult using:
    - getResult()
    - getData() is the Intent passed from the previous activity.

Activity sequence

ActivityA

- ActivityResultLauncher myLauncher= registerForActivityResult

➔ ActivityResultLauncher is an implementation of the interface that contains onActivityResult() callback.

- myLauncher.launch(Intent) ;

→ We launch the second activity using the object myActivityResultLauncher we created.

- onActivityResult(ResultActivity);

→ onActivityResult is the method called when the activity returns. ResultActivity object contains the returned Intent and ResultCode. You can use getResultCode(), getIntent() calls on it.

- ActivityB

- onCreate(), onStart(), onResume()
- setResult(resultCode, intent), finish(), onPause(), onFinish()

Intents

- Android has pre-defined Intents for various actions. You don’t have to specify the next Activity class you will go to. You can just specify what action you want to perform, and Android will offer a list of Activity objects that implement that action.

- You can use these to see what the default Activity on your phone for an Intent:
  - Intent.ACTION_CALL : launch the phone calling Activity
  - Intent.ACTION_VIEW : launch a web browser
  - MediaStore.ACTION_IMAGE_CAPTURE : take a photo

Intents

Some examples of action/data pairs:

ACTION_VIEW *content://contacts/people/1* -- Display information about the person whose identifier is "1".

ACTION_DIAL *content://contacts/people/1* -- Display the phone dialer with the person filled in.

ACTION_VIEW *tel:123* -- Display the phone dialer with the given number filled in.

ACTION_DIAL *tel:123* -- Display the phone dialer with the given number filled in.

ACTION_EDIT *content://contacts/people/1* -- Edit information about the person whose identifier is "1".

ACTION_VIEW *content://contacts/people/* -- Display a list of people, which the user can browse through. **ACTION_VIEW *content://contacts/people/N*** } being used to start an activity to display that person.

Intent to send email

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_EMAIL, new String[] {"email@example.com"});
intent.putExtra(Intent.EXTRA_SUBJECT, "subject here");
intent.putExtra(Intent.EXTRA_TEXT, "body text");
startActivity(intent);
```

- Android will open the Email app that is currently responsible for ACTION_SEND, and then go to that browser. Your phone might have Gmail, Outlook, or another email program that is the default for ACTION_SEND

Intent to make phone call

```
Intent intent = new Intent(Intent.ACTION_DIAL);
intent.setData(Uri.parse("tel:" + phoneNumber));
startActivity(intent);
```

- Android will open the phone app that is currently responsible for ACTION_DIAL, and then go to that browser.

Intent to view web page

```
String url = "http://www.algonquincollege.com";
Intent i = new Intent(Intent.ACTION_VIEW);
i.setData( Uri.parse(url) );
startActivity(i);
```

- Android will open the Web Browser that is currently responsible for ACTION_VIEW, and then go to that browser. Your phone might have Chrome, Samsung Web, Firefox, or some other browser that is currently the default application for ACTION_VIEW.

Look at all of the built-in actions here:

- https://developer.android.com/guide/components/intents-common

Bundle

- Bundle is an object for data storage that saves the data along with a key name.
- To save data, use functions like:
  - putByte(String key, Byte b),
  - putFloat(String key, Float f),
  - putIntegerArrayList(String key, ArrayList<Integer> list)
- To retrieve the data, use the function you saved it with:
  - getByte(String key)
  - getFloat(String key)

ALGONQUIN COLLEGE

Bundle data

- **Missing data**: if you call getFloat("String name"), or getByte("anyName") and the name does not appear in the bundle, then it returns 0, or null.
- **Wrong data type**: if the data type is wrong, meaning that you saved putByte("MyNumber", 5), and you call getFloat("MyNumber"), it returns 0 instead of 5.0.
- You can put an entire bundle in an Intent to send to the next activity using the ***putExtras()*** function.

SharedPreferences

- SharedPreferences provides an interface for saving data to a file on your device:

SharedPreferences prefs = getSharedPreferences(String fileName, int mode)

- The String fileName specifies the name of the file, mode is the security permissions – use Context.MODE_PRIVATE
- getSharedPreferences() returns a SharedPreferences object for reading and writing data.

Reading / Writing sharedPreferences

- From the sharedPreferences object, you must get the Editor object:

  SharedPreferences.Editor edit = prefs.edit();
- The editor has *putString(String key, String value), putFloat(String key, float f), putInt(String key, int i)*…etc
- You must call commit() when all your values have been set to save them to the file: *edit.commit();*
- SharedPreferences has *getInt(String key), getFloat(String key)*, etc. to read whatever data was in the file.

Shared Preferences Example

- In the **onCreate()**, it opens a shared preferences file called "MyData.xml" and rebuilds the reservation table.
- It looks at what was saved under the name "ReserveName". If the name is not found, then the second value is used instead: "Default Value". It then puts it in the editText
- When you click the Save button, it opens a SharedPreferences Editor. Under the name "ReserveName", it saves what you put in the edit text.

SharedPreferences Example

- Have a look at your AndroidManifest.xml file.
- There should be a line there saying package=“com.cst2335.exercises”.
- You will need this package name for the next step.

SharedPreferences Example

- If you run the shell script file, it should go to the platform-tools directory where the adb program is located.
- Next, use the command “adb shell” in the terminal to log onto your phone. Type “run-as packagename” but replace the packagename with what is in your manifest.
- Navigate to “/data/data/packagename/shared_prefs”. The file “FileName.xml” should be there now. Type “more FileName.xml” to show what is in your XML file.

Debugging Information

- Android has a Log class for printing out information. It is similar to System.out.println, except it prints out to the debugger on your laptop, not on your phone.
- There are several functions for printing information. You can filter these messages with your debugger:
- Log.i( ) // prints out information
- Log.w( ) // prints out warnings
- Log.e( ) // prints out errors
- Log.v( ) //prints out verbose (detailed) messages.

Summary

- Intent object starts a transition from one page to another. You can also send data to the next Activity using putExtra(“Name”, value)
- Call startActivity(intent) to go to the next page.
- If you are expecting an Intent object back from that page, use the ActivityResultLauncher().
- The previous page will call onActivityResult().
- SharedPreferences write data to the disk for use later.

Summary

- You should learn how to use these functions:
  - startActivity()
  - ActivityResultLauncher()
  - putExtra(), getExtra()
  - setResult()
  - finish()
  - onActivityResult()
  - getSharedPreferences()
  - commit()