9. Lesson 28/03/23

Activity instance state (part I)

When does config change?

Configuration changes invalidate the current layout or other resources in your activity **when** the user:

- Rotates the device
- Chooses different system language
- Enters multi-window mode

What happens on config change?

On configuration change, Android:

1. Shuts down Activity by calling:
    - onPause()
    - onStop()
    - onDestroy()
1. Starts Activity over again by calling:
    - onCreate()
    - onStart()
    - onResume()

Activity instance state

State information is created while the Activity is running, such as:

- A counter
- User text
- Animation progression

State is lost when device is rotated, language changes, back-button is pressed, or the system clears memory

What the system saves

System saves only:

- State of views with unique ID (android:id) such as text entered into EditText
- Intent that started activity and data in its extras

You are **responsible for saving other activity and user progress data.**

Saving instance state

Implement ***onSaveInstanceState()*** in your Activity:

- Called by Android runtime when there is a possibility the Activity may be destroyed.
- Saves data only for this instance of the Activity during current session.

onSaveInstanceState(Bundle outState)

```
@Override
protected void onSaveInstanceState(Bundle outState) {
  super.onSaveInstanceState(outState);
  // save info
  outState.putString("count", String.valueOf(mShowCount.getText()));
}
```

Restoring instance state

Two ways to retrieve the saved Bundle:

- **in onCreate** (Bundle mySavedState)

  Preferred, to ensure that your user interface, including any saved state, is back up and running as quickly as possible.

- **Implement callback** (called after onStart())

  onRestoreInstanceState(Bundle mySavedState)

Restoring in onCreate()

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  if (savedInstanceState != null) {
      username = savedInstanceState.getString(“user");
  }
}
```

onRestoreInstanceState(Bundle state)

```
@Override
protected void onRestoreInstanceState (Bundle mySavedState) {
  super.onRestoreInstanceState(mySavedState);
  if (savedInstanceState != null) {
      username = savedInstanceState.getString(“user");
  }
}
```

Instance state and app restart

When you stop and restart a new app session, the Activity instance states are lost and your activities will revert to their default appearance.

If you need to save user data between app sessions, use shared preferences or a database.

Activities and Intents (part II)

Contents

How to **get data back from an Activity.**

Returning data to the starting activity

When starting an Activity with an Intent:

- The originating Activity is paused
- The new Activity remains on the screen until:
  - the user clicks the Back button or
  - the finish() method is called (in a click handler or other function that ends the user’s involvement with this Activity).

Returning data to the starting activity

Sometimes when starting an Activity with an Intent, you would like to also **get data back from that Intent.**

- **E.g.: start an activity that lets the user pick a prefix**

  The original Activity needs to **receive information about the prefix** the user chose back from the launched/new Activity.

Returning data to the starting activity

To **launch a new Activity and get a result back**, do the following steps:

1. Use ***startActivityForResult(…)*** to start the second Activity.
2. To return data from the second Activity:
    - Create a **new Intent**
    - Put the response data in the Intent using **putExtra()**
    - Set the result to **Activity.RESULT_OK** or **RESULT_CANCELED**, if the user cancelled out
    - call **finish()** to close the Activity
3. Implement **onActivityResult()** in first Activity.

1. startActivityForResult()

The ***startActivityForResult()*** method takes:

- an **Intent argument** that contains:
    - **information** about the Activity to be launched.
    - **data** to send to that Activity (optional).
- a **request code**

```
startActivityForResult(intent, requestCode);
```

Request code: an integer that identifies the request and can be used to differentiate between results when you process the return data.

1. startActivityForResult() - Example

In the activity that starts the new one:

```
// define the identifier
public static final int CHOOSE_FOOD_REQUEST = 1;

// create the intent
Intent intent = new Intent(this, ChooseFoodItemsActivity.class);

// start the activity
startActivityForResult(intent, CHOOSE_FOOD_REQUEST);
```

In the new activity to return info:

```
// Create an intent
Intent replyIntent = new Intent();

// Put the data to return into the extra
```

```
replyIntent.putExtra(EXTRA_REPLY, reply);

// Set the activity's result to RESULT_OK
setResult(RESULT_OK, replyIntent);

// Finish the current activity
finish();
```

In the original activity receive the data:

```
public void onActivityResult(int requestCode, int resultCode, Intent data) {
super.onActivityResult(requestCode, resultCode, data);
if (requestCode == CHOOSE_FOOD_REQUEST) { // Identify activity
    if (resultCode == RESULT_OK) { // Activity succeeded
        String reply = data.getStringExtra(SecondActivity.EXTRA_REPLY);
        // … do something with the data
}}}
```