Rajdip Sanyal (13/09/2024)
sanyalrajdip864@gmail.com

Android Framework With Java

Returning Data From Activity in Android

In android development, returning data from one activity to another is a common scenario, especially when we need to pass results back to the activity that started another activity. This typically involves using intents and the startActivityForResult() method in conjunction with the setResult() method.
Here's a theoretical overview of how this works:

Key Concepts:

Activity: An activity represents a single screen with a user interface. Each activity is an instance of the Activity class.

Intent: An Intent is an object that provides runtime binding between separate components, such as two activities. Intents can be used to start activities and pass data between them.

startActivityForResult(): This method is used to start another activity and expects a result to be returned. It takes an Intent to specify the activity to start and a request code that is used to identify the result later.

onActivityResult(): This method is called when the started activity finishes and returns a result. It provides the request code, result code, and any data returned from the second activity.

setResult(): In the second activity, we use this method to set the result that will be returned to the original activity. This method takes a result code and an optional Intent with the result data.

Action Bar in Android

In android development, the Action Bar is a crucial component of the user interface that provides a consistent and familiar way for users to interact with an app. It was first introduced in Android 3.0 (Honeycomb) and serves as a key part of the app's navigation and user interaction model. However, it's worth noting that the Action Bar has evolved, and many of its features have been replaced or supplemented by newer UI components.

Purpose: The Action Bar is designed to provide users with a consistent location for key actions, navigation, and contextual information across different screens of an app.

Roll: It typically appears at the top of the screen and provides a space for elements such as the app title, navigation icons, and action items.

```
124
125
126             ActionBar actionBar = getSupportActionBar();
127             actionBar.setTitle("My App");
128             actionBar.setSubtitle("spiderman app");
129             actionBar.setIcon(R.drawable.xresimg2);
130             actionBar.setDisplayUseLogoEnabled(true);
131             actionBar.setDisplayShowHomeEnabled(true);
132
133         }
```

Receiving Results From Activities in Android

In android development, handling the return of results from one activity to another is a common task. The process typically involves using startActivityForResult() and onActivityResult(), although there are newer methods for handling results that we might prefer in modern android development. Here's an overview of theory and methods:

```
        } else {

            |
        }
            Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivityForResult(cameraIntent, CAMERA_REQUEST);


    });
```

Starting the activity: The above code snippet is the example of starting the activity. When we want to start another activity and expect a result back, we use the startActivityForResult() method. This method requires us to provide an Intent to start the new activity and a request code that you define.

```
    @Override
    public void onCreate(Bundle bundle)
    {
        super.onCreate(bundle);
        setContentView(R.layout.about_activity);
        Intent data = new Intent();
// String text = "result from aaryan";
// data.setData(Uri.parse(text));
        data.putExtra( name: "name",  value: "Rajdip Sanyal");
        data.putExtra( name: "role",  value: "Developer");
        data.putExtra( name: "exp",  value: 1);
        setResult(RESULT_OK, data);
        finish();
    }
}
```

Receiving the Result: The above code snippet is the example of receiving the result. In this case, we typically collect data and then pass it back to the calling activity using an Intent with the results.

Notifications in Android

In android development, notifications are a crucial component for engaging users and providing them with timely updates or alerts.

Definition and Purpose

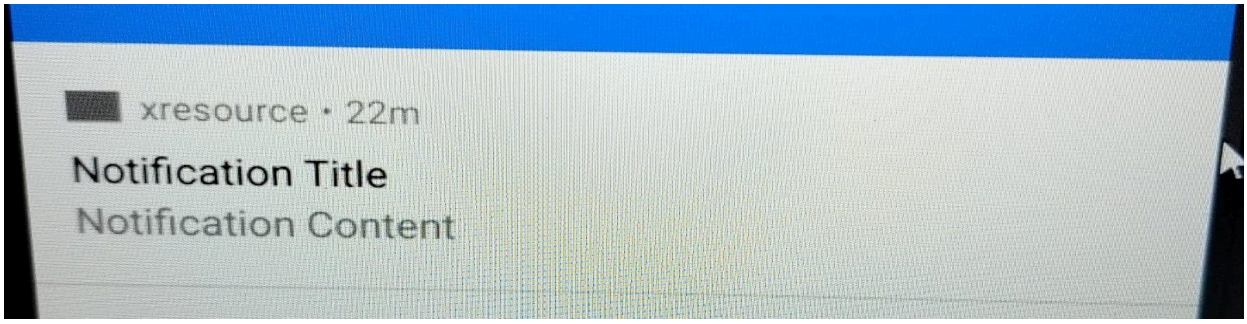
Notification: A message or alert that appears outside the normal app flow to provide the user with information, updates, or actionable items.

Notifications can be used for various purposes, such as alerting users about new messages, upcoming events, or app-related updates.

```
notificationManager=(NotificationManager) getSystemService(NOTIFICATION_SERVICE);

Intent intent = new Intent( packageContext: this, AboutActivity.class);
PendingIntent pendingIntent = PendingIntent.getActivity( context: this,  requestCode: 0, intent,
if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O)
{
    Toast.makeText( context: this,  text: "Yay!!!", Toast.LENGTH_SHORT).show();
    notificationChannel = new NotificationChannel(channelId, desc, NotificationManager.IMPO
    notificationChannel.enableLights(true);
    notificationChannel.setLightColor(Color.GREEN);
    notificationChannel.enableVibration(false);
    notificationManager.createNotificationChannel(notificationChannel);
    Notification.Builder builder1 = new Notification.Builder( context: this, channelId)
            .setSmallIcon(R.drawable.xresimg2)
            .setContentIntent(pendingIntent);
```

Notification Output



Cameras in Android

The camera in Android development is a feature that enables users to take photos and videos, and scan documents and QR codes. The camera subsystem includes components for the camera pipeline, and the camera hardware abstraction layer (HAL) connects these components to the underlying hardware and camera driver.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == 1 && resultCode == RESULT_OK) {
        String name = data.getStringExtra( name: "name");
        String role = data.getStringExtra( name: "role");
        int exp = data.getIntExtra( name: "exp",  defaultValue: -1);
        Toast.makeText(getApplicationContext(),  text: name + ", " + role + ", " + exp, Toast.
        Log.i( tag: "data",  msg: name + ", " + role + ", " + exp);
    } else if (requestCode == CAMERA_REQUEST) {
        Bitmap photo = (Bitmap) data.getExtras().get("data");
        iv.setImageBitmap(photo);
    }
```

```
        } else {

        }
            Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivityForResult(cameraIntent, CAMERA_REQUEST);


    });
```

Camera Output