Notes on Advanced Flutter Add-to-App Integration (Part -2)

---

1. Native Main Activity

Purpose:

- Entry point of the native Android app.
- Initializes a FlutterEngine and caches it for later use.
- Navigates from the main screen to the login screen.

Key Code:

```
package com.example.nativeandroid

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
import com.example.nativeandroid.ui.login.LoginActivity
import io.flutter.embedding.android.FlutterActivity
import io.flutter.embedding.engine.FlutterEngine
import io.flutter.embedding.engine.FlutterEngineCache
import io.flutter.embedding.engine.dart.DartExecutor

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Button that triggers the login screen.
        val openFlutterButton = findViewById<Button>(R.id.openFlutter)

        openFlutterButton.setOnClickListener {
            // Create a new FlutterEngine instance.
            val flutterEngine = FlutterEngine(this)
            // Start executing the default Dart entrypoint (main())
            flutterEngine.dartExecutor.executeDartEntrypoint(
                DartExecutor.DartEntrypoint.createDefault()
            )
            // Cache the FlutterEngine with a key ("my_engine") for reuse.
            FlutterEngineCache.getInstance().put("my_engine", flutterEngine)

            // Start the LoginActivity.
            val loginIntent = Intent(this@MainActivity, LoginActivity::class.java)
            startActivity(loginIntent)
        }
    }
}
```

Key Concepts:

- **FlutterEngine:** The runtime that executes Flutter/Dart code.
- **FlutterEngineCache:** Caches the engine instance so that the Flutter module can be launched quickly without re-initializing the engine.
- **Intent Navigation:** Native Android uses an Intent to switch from MainActivity to LoginActivity.

---

2. Native Login Activity

Purpose:

- Provides a login screen with username and password fields.
- Uses a MethodChannel to send the login data from native code to the Flutter module.
- Launches a FlutterActivity using the cached FlutterEngine.

Key Code:

```
package com.example.nativeandroid.ui.login

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import com.example.nativeandroid.databinding.ActivityLoginBinding
import io.flutter.embedding.android.FlutterActivity
import io.flutter.embedding.engine.FlutterEngineCache
import io.flutter.plugin.common.MethodChannel

class LoginActivity : AppCompatActivity() {
    // Define the channel name for communication with Flutter.
    private val CHANNEL = "login_channel"
    private lateinit var binding: ActivityLoginBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // Use view binding for layout inflation.
        binding = ActivityLoginBinding.inflate(layoutInflater)
        setContentView(binding.root)
```

```
        // Retrieve the cached FlutterEngine.
        val cachedEngine = FlutterEngineCache.getInstance().get("my_engine")
        // Enable the login button.
        binding.login.isEnabled = true

        binding.login.setOnClickListener {
            // Get the entered username and password.
            val usernameText = binding.username.text.toString()
            val passwordText = binding.password.text.toString()

            // Ensure the cached engine is available.
            if (cachedEngine != null) {
                // Create a MethodChannel using the cached engine's binary messenger.
                val channel = MethodChannel(cachedEngine.dartExecutor.binaryMessenger, CHANNEL)
                // Send login data to Flutter using the method channel.
                channel.invokeMethod("login", mapOf(
                    "username" to usernameText,
                    "password" to passwordText
                ))
                // Launch the FlutterActivity with the cached engine.
                val flutterIntent = FlutterActivity.withCachedEngine("my_engine").build(this)
                startActivity(flutterIntent)
            }
        }
    }
}
```

Key Concepts:

- **View Binding:** Used to simplify accessing views defined in XML.
- **MethodChannel:** A communication bridge between native code and Flutter. Both sides use the same channel name (login_channel).
- **Sending Data:** Native code sends a map containing the username and password.

- **Launching Flutter:** After sending data, the activity launches FlutterActivity that uses the cached FlutterEngine.

---

3. Layout Files

Main Activity Layout (activity_main.xml):

Purpose:

- Contains a simple button that starts the login process.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <Button
        android:id="@+id/openFlutter"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open Flutter Screen" />
</LinearLayout>
```

Login Activity Layout (activity_login.xml):

Purpose:

- Provides input fields for username and password along with a login button.

```
<?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"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <EditText
        android:id="@+id/username"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:inputType="textEmailAddress"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>

    <EditText
        android:id="@+id/password"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Password"
        android:inputType="textPassword"
        app:layout_constraintTop_toBottomOf="@+id/username"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
```

```
        android:layout_marginTop="8dp"/>

    <Button
        android:id="@+id/login"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Login"
        app:layout_constraintTop_toBottomOf="@+id/password"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginTop="16dp"/>

    <!-- Optional ProgressBar, if needed -->
    <ProgressBar
        android:id="@+id/loading"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"/>

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

Key Concepts:

- **ConstraintLayout and LinearLayout:** Used to structure the UI.
- **Input Fields & Button:** Capture user credentials and trigger login.

---

4. Flutter Side (Dart) – Brief Recap

While the Dart side isn't shown here, recall that on the Flutter module you:

- **Set Up a MethodChannel** with the same channel name (login_channel).
- **Listened for the "login" method call** and updated the Flutter UI accordingly (e.g., displaying the username).

Summary of What You Learned

- **Creating a Native Login Screen:**
  Built a native Android login UI using XML layouts and View Binding.

- **Initializing & Caching FlutterEngine:**
  Initialized a FlutterEngine in MainActivity and cached it for reuse in launching Flutter UI quickly.

- **Inter-Platform Communication:**
  Implemented a MethodChannel to pass login data from the native Android side to the Flutter module.

-

Launching Flutter from Native:
Launched a FlutterActivity using the cached engine after sending the data.

- **Layout and Navigation:**
  Used XML layouts for both main and login screens and navigated between activities using Intent.