Launch Second Screen

This lesson will cover how to simulate a long-running operation and open a new screen.

Flow overview

When a user clicks the *login* button, we will perform a data validation flow, and if the data is valid, proceed to:

1. Simulate long-running operation for 2 seconds
2. Open *MainActivity*
3. Finish *LoginActivity*

Simulate a long-running operation

In this lesson, we are not going to do a real login HTTP call, so let's just simulate it via delay of 2 seconds before opening *MainActivity*. To do this, we can use the `Handler` object.

In Android, `Handler` is an object which is tied to the looper of the thread in which it's been created. It typically has two use-cases:

1. If we want to execute code on the *main* thread
2. If we want to execute code with a delay or at the specific time

Let's modify our `performLogin` method and create a `Handler` object at the very end. Now we can use this object's `postDelayed` method to execute code with a delay. This method has two parameters:

- `Runnable` that will be executed *(presented as lambda in the code below)*

- delay (in milliseconds) until the `Runnable` will be executed

```
private void performLogin() {
    textUsernameLayout.setEnabled(false);
    textPasswordInput.setEnabled(false);
    loginButton.setVisibility(View.INVISIBLE);
    progressBar.setVisibility(View.VISIBLE);

    Handler handler = new Handler();
    handler.postDelayed(() -> {
        // your code
    }, 2000);
}
```

LoginActivity

Intent

Remember that all activities must be registered in *AndroidManifest.xml*. Our *MainActivity* is already there.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.travelblog">
    <application
        android:theme="@style/Theme.MaterialComponents.DayNight.NoActionBar"
         android:label="Travel Blog">
        <activity android:name=".LoginActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MainActivity" />
    </application>
</manifest>
```

AndroidManifest

To open another screen we need to use an `Intent` object which requires a package context and the activity class. In our case, we can specify `this` as a package context, since `Activity` class implements `Context` interface, as for activity class we simply use `MainActivity.class`.

Now when the `Intent` object is filled with the required information, we can pass it to the `Activity#startActivity` method.

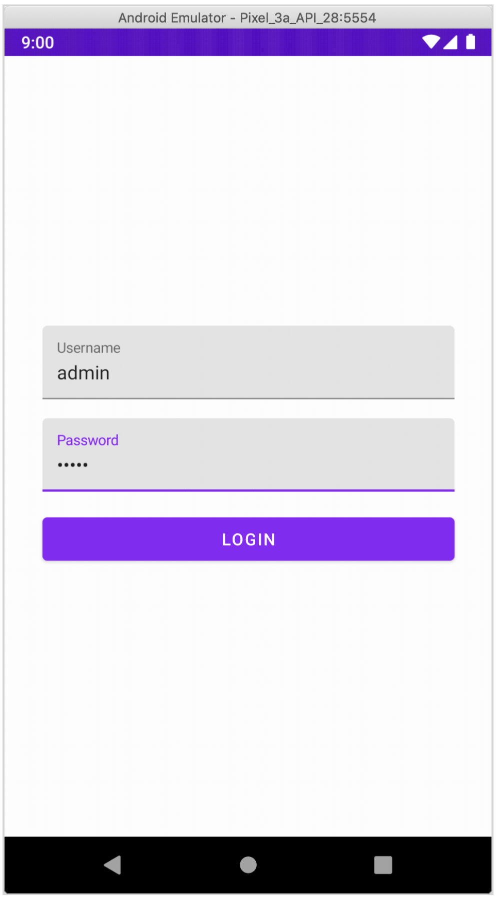
```
private void performLogin() {
    ...

    Handler handler = new Handler();
    handler.postDelayed(() -> {
        startMainActivity();
    }, 2000);
}

private void startMainActivity() {
    Intent intent = new Intent(this, MainActivity.class);
    startActivity(intent);
}
```

LoginActivity

As you can see on the preview below, when we click on the login button, *MainActivity* is opened.



The only issue which remains is that *MainActivity* is opened on top of the LoginActivity, but *LoginActivity* is still in the back stack, so when we click the back button we can see it.

To fix the issue mentioned above, we have to use the `Activity#finish` method to close *LoginActivity* after opening *MainActivity*.

```
private void performLogin() {
    ...
    Handler handler = new Handler();
    handler.postDelayed(() -> {
        startMainActivity();
        finish();
    }, 2000);
}
```

LoginActivity

Hit the *run* button to try it yourself.

```
package com.travelblog;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.Button;
import android.widget.ProgressBar;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.material.textfield.TextInputLayout;

public class LoginActivity extends AppCompatActivity {

    private TextInputLayout textUsernameLayout;
    private TextInputLayout textPasswordInput;
    private ProgressBar progressBar;
    private Button loginButton;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        textUsernameLayout = findViewById(R.id.textUsernameLayout);
        textPasswordInput = findViewById(R.id.textPasswordInput);
        progressBar = findViewById(R.id.progressBar);
        loginButton = findViewById(R.id.loginButton);
        loginButton.setOnClickListener(v -> LoginActivity.this.onLoginClicked());

        textUsernameLayout
                .getEditText()
                .addTextChangedListener(createTextWatcher(textUsernameLayout));

        textPasswordInput
                .getEditText()
                .addTextChangedListener(createTextWatcher(textPasswordInput));
    }

    private void onLoginClicked() {
        String username = textUsernameLayout.getEditText().getText().toString();
        String password = textPasswordInput.getEditText().getText().toString();
        if (username.isEmpty()) {
```

```
        if (username.isEmpty()) {
            textUsernameLayout.setError("Username must not be empty");
        } else if (password.isEmpty()) {
            textPasswordInput.setError("Password must not be empty");
        } else if (!username.equals("admin") && !password.equals("admin")) {
            showErrorDialog();
        } else {
            performLogin();
        }
    }

    private void performLogin() {
        textUsernameLayout.setEnabled(false);
        textPasswordInput.setEnabled(false);
        loginButton.setVisibility(View.INVISIBLE);
        progressBar.setVisibility(View.VISIBLE);

        Handler handler = new Handler();
        handler.postDelayed(() -> {
            startMainActivity();
            finish();
        }, 2000);
    }

    private void startMainActivity() {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }

    private void showErrorDialog() {
        new AlertDialog.Builder(this)
                .setTitle("Login Failed")
                .setMessage("Username or password is not correct. Please try again.")
                .setPositiveButton("OK", (dialog, which) -> dialog.dismiss())
                .show();
    }

    private TextWatcher createTextWatcher(TextInputLayout textPasswordInput) {
        return new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s,
                                          int start, int count, int after) {
                // not needed
            }

            @Override
            public void onTextChanged(CharSequence s,
                                      int start, int before, int count) {
                textPasswordInput.setError(null);
            }

            @Override
            public void afterTextChanged(Editable s) {
                // not needed
            }
        };
    }
}
```

Quiz

Quiz

1 Which object must be used to launch another screen?

COMPLETED 0%

1 of 2

In the next lesson, we will cover how to store login state in the Android key-value storage.