Data Validation and Error Handling

This lesson will cover how to validate user input text and show error messages.

WE'LL COVER THE FOLLOWING

View binding

To interact with views we need to bind the view from XML to Java objects via the `findViewById` method.

Note: You can only start binding views after the activity has been created and the content view has been set.

```
public class LoginActivity extends AppCompatActivity {

    private TextInputLayout textUsernameLayout;
    private TextInputLayout textPasswordInput;
    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);
        loginButton = findViewById(R.id.loginButton);
    }
}
```

LoginActivity

Validation & error message

To perform validation, we need to set a click listener on the *login* button via the `setOnClickListener` method, which accepts the `OnClickListener` interface as a parameter. When the *login* button is clicked, we execute our `onLoginClicked` method.

```
public class LoginActivity extends AppCompatActivity {

    ...
    private Button loginButton;

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

        ...
        loginButton = findViewById(R.id.loginButton);
        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                LoginActivity.this.onLoginClicked();
            }
        });
    }

    private void onLoginClicked() {
        // not implemented yet
    }
}
```

LoginActivity

Click listener can be slightly simplified to lambda because we use Java 8.

```
loginButton = findViewById(R.id.loginButton);
loginButton.setOnClickListener(v -> onLoginClicked());
```

LoginActivity

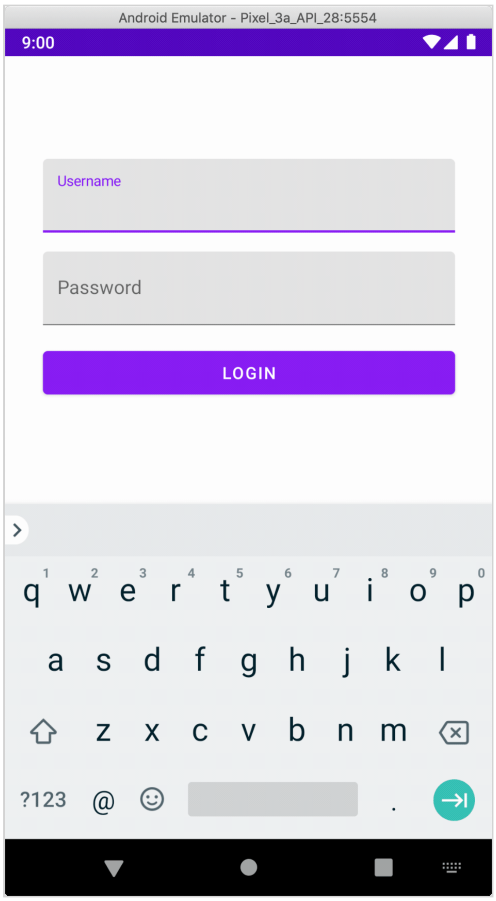
Now, we can implement validation inside `onLoginClicked` method following these steps:

1. Retrieve the reference to the input field via `TextInputLayout#getEditText()` method
2. Retrieve text from the input field via `TextInputEditText#getText().toString()` method
3. Check if text is empty via `String#isEmpty` method
4. Show error via `TextInputLayout#setError` method

```
private void onLoginClicked() {
      String username = textUsernameLayout.getEditText().getText().toString();
      String password = textPasswordInput.getEditText().getText().toString();
      if (username.isEmpty()) {
          textUsernameLayout.setError("Username must not be empty");
      } else if (password.isEmpty()) {
          textPasswordInput.setError("Password must not be empty");
      }
}
```

LoginActivity

As you can see in the preview below, when we click the *login* button while the *username* or *password* input fields are empty an error message is displayed.



Error message cleanup

With the introduction of validation and error messages, we also introduced the following issue: the error message is not cleared when a user types a text in the input field. To fix this issue, we can listen when text inside the *username* or *password* input fields change and clear the error.

Here is what we need to do:

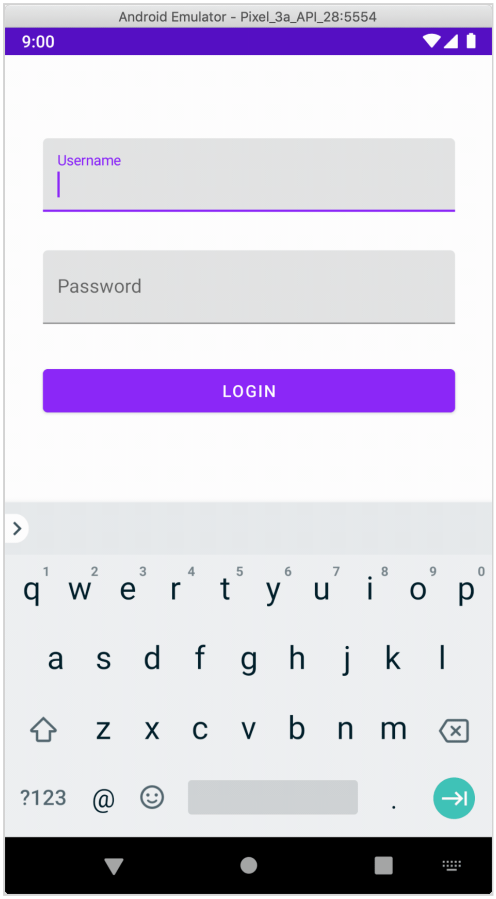
1. Retrieve the reference to the input field via

1. Retrieve the reference to the input field via `TextInputLayout#getEditText()` method
2. Add text change listener via `TextInputEditText#addTextChangedListener` method and provide a `TextWatcher` object along with `TextInputLayout`
3. Inside the `TextWatcher#onTextChanged` method clear the error via `TextInputLayout#setError(null)` method

```
public class LoginActivity extends AppCompatActivity {
    ...

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

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

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

    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
            }
        };
    }
}
```

LoginActivity

As you can see on the preview below, when we enter text in the *username* input field the error message is cleared.



Validation & error dialog

Sometimes it's useful to show error messages in the form of a dialog. Let's do our final validation and show error dialog with the text: "Username or password is not correct. Please try again." if the *username* and *password* input field texts are not equal to "admin".

To check *username* and *password* input field, text we can add one more `else if` block and use the `String#equals` method to check whether they are equal to "admin". If it is not, execute the `showErrorDialog` method.

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

LoginActivity

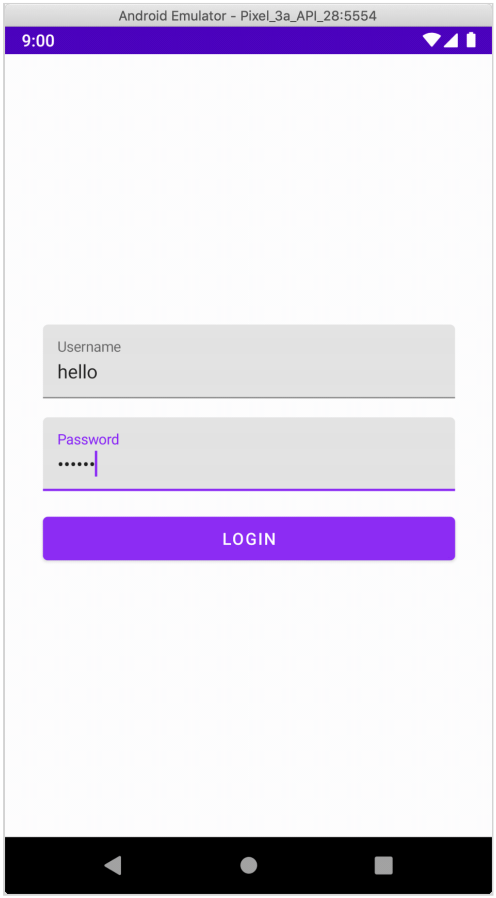
To show the dialog, we need to create it via `AlertDialog.Builder` object and then call the `show` method when we want to show the dialog. `AlertDialog.Builder` has a lot of different methods to customize the dialog, in our case we just used the following:

- `setTitle` method sets the dialog title
- `setMessage` method sets the dialog message
- `setPositiveButton` method sets the button with the given text and button click listener where we use `dismiss()` method to close the dialog.

```
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();
}
```

LoginActivity

As you can see in the preview below, when we use random text for *username* and *password*, the error dialog is displayed.



Hit the *run* button to try it yourself.

```
package com.travelblog;

import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.Button;

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 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);
        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()) {
            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();
        }
    }

    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
            }
        };
    }
}
```

In the next lesson, we will cover how to display a loading indicator and prevent user interaction with the views.