Loading Indicator

In this lesson, we will look at how to add a loading indicator when blog data is loading.

Loading data from the network can take time, especially with a weak internet connection. Currently, while data is loading, the user sees a blank screen, which may be confusing. That’s why we need to introduce some kind of loading indicator.

We can show a simple `ProgressBar`, similar to what we did on the login screen, in the middle of the screen while data is loading.

Let’s start by declaring `ProgressBar` in the `activity_blog_details` file and constrain it to the middle of the screen. Leave the `ProgressBar` visible by default.

```xml
<?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"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    ...
    <ProgressBar
            android:id="@+id/progressBar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

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

activity_blog_details.xml

Next, in the `BlogDetailsActivity` bind the `ProgressBar` view to the Java field.

```java
public class BlogDetailsActivity extends AppCompatActivity {

    ...
    ...
    private ProgressBar progressBar;

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

        ...
        progressBar = findViewById(R.id.progressBar);

        loadData();
    }
```

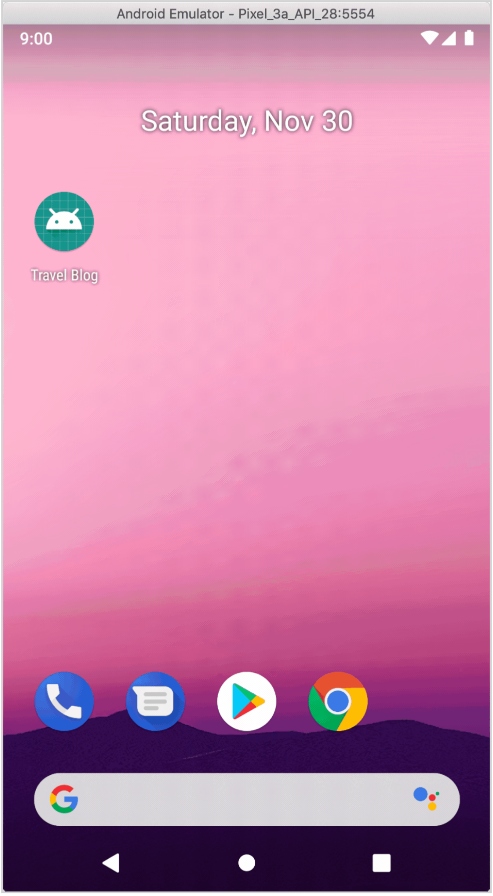
Finally, in the `showData` method we can hide the `ProgressBar`.

```
private void showData(Blog blog) {
    progressBar.setVisibility(View.GONE);
    textTitle.setText(blog.getTitle());
    textDate.setText(blog.getDate());
    textAuthor.setText(blog.getAuthor().getName());
    textRating.setText(String.valueOf(blog.getRating()));
    textViews.setText(String.format("(%d views)", blog.getViews()));
    textDescription.setText(blog.getDescription());
    ratingBar.setRating(blog.getRating());
    ratingBar.setVisibility(View.VISIBLE);

    Glide.with(this)
            .load(blog.getImage())
            .transition(DrawableTransitionOptions.withCrossFade())
            .into(imageMain);

    Glide.with(this)
            .load(blog.getAuthor().getAvatar())
            .transform(new CircleCrop())
            .transition(DrawableTransitionOptions.withCrossFade())
            .into(imageAvatar);
}
```

Now, when we launch the application, we should see a loading indicator.



Hit the *run* button to try it yourself.

```
package com.travelblog.http;

public class Author {

    private String name;
    private String avatar;

    public String getName() {
        return name;
    }

    public String getAvatar() {
        return avatar;
    }
}
```

The next lesson will introduce how to handle and recover from errors.