Image Loading

This lesson goes over image loading from the internet.

WE'LL COVER THE FOLLOWING

Dependencies

Loading an image from the Internet may sound like a trivial task, but it is not. There are a lot of cases which we would have to handle, some of them:

- storing and retrieving the image from the RAM cache
- storing and retrieving the image from the disk cache
- scaling and resizing the image
- displaying loading and error states
- handling out of memory scenarios
- handling concurrent image loading
- etc.

To avoid handling all the scenarios described above, there are several popular image loader libraries developed and distributed as open source. The most popular are:

- Glide
- Fresco
- Picasso

In this lesson, we are going to use the Glide image loader. Let's add the new library dependencies to the *app/build.gradle* file in the `dependencies` section.

```
dependencies {
    // ui
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.android.material:material:1.1.0-alpha10'
    implementation 'com.github.bumptech.glide:glide:4.10.0'
}
```

Permissions

Since we are going to load the image from the Internet, our application needs to have internet access. To achieve this there is a concept of application permissions.

Basically, if we need access to some external functionality, like the Internet or location, we need to declare permission in *AndroidManifest.xml* file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.travelblog">

    <uses-permission android:name="android.permission.INTERNET" />

    <application
            android:theme="@style/AppTheme"
            android:label="Travel Blog">
        ...
    </application>
</manifest>
```

AndroidManifest.xml

Note: Some permissions also require explicit user approval, in which case we would need to show the permission dialog to the user.

Image loading

Now, when all pre-requirements are satisfied, we can start loading the images.

Let's declare two image URLs in the *BlogDetailsActivity,* one for main image and the other for avatar image.

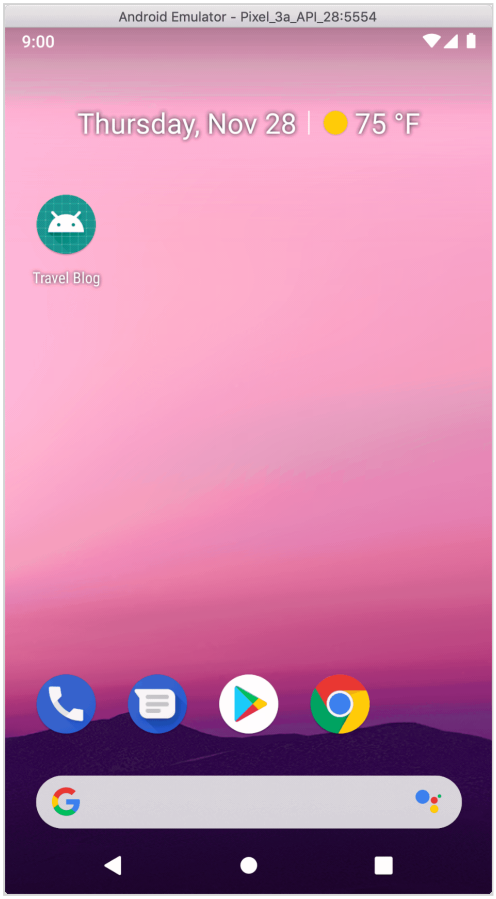
```
public class BlogDetailsActivity extends AppCompatActivity {

    public static final String IMAGE_URL =
            "https://bitbucket.org/dmytrodanylyk/travel-blog-resources/raw/" +
                    "3436e16367c8ec2312a0644bebd2694d484eb047/images/sydney_image.jpg";
    public static final String AVATAR_URL =
            "https://bitbucket.org/dmytrodanylyk/travel-blog-resources/raw/" +
                    "3436e16367c8ec2312a0644bebd2694d484eb047/avatars/avatar1.jpg";
    ...
}
```

To load the image, we need to:

- create `Glide` object via the `Glide.with(this)` method, where `this` is Activity
- call `load` method with the image URL as a parameter
- call `into` method with image view as a parameter

```
public class BlogDetailsActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_blog_details);
        ...
        ImageView imageMain = findViewById(R.id.imageMain);
        Glide.with(this)
                .load(IMAGE_URL)
                .into(imageMain);

        ImageView imageAvatar = findViewById(R.id.imageAvatar);
        Glide.with(this)
                .load(AVATAR_URL)
                .into(imageAvatar);
    }
}
```

Now when we launch the application, images will be loaded from the Internet.



While image loading works, there are two improvements which we can do:

- avatar image is square, it should be a circle
- images appear without any animation, so it feels unnatural

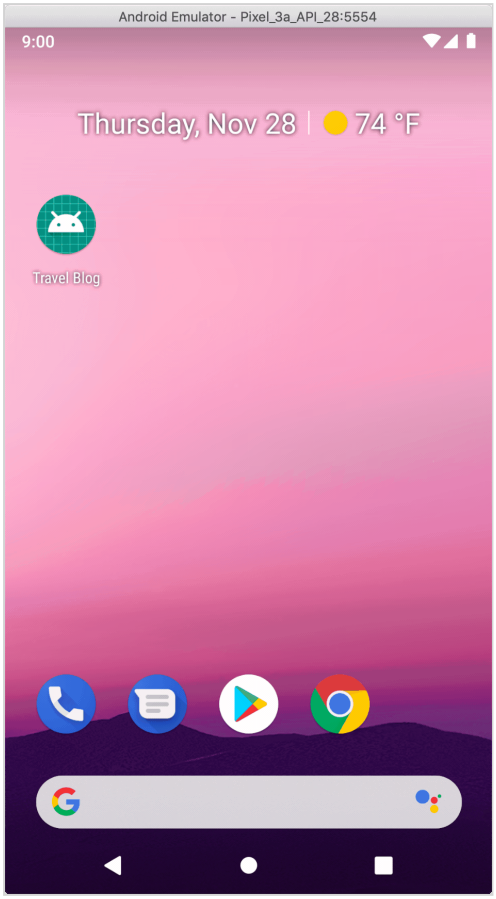
Circle the image

One way to fix the problem of a square image is to upload circle images to the backend, but this is not flexible and sometimes not possible. The other way to fix the problem is to make a circle image from the code. Fortunately, Glide already knows how to do this.

We can use the `transform` method and pass `CircleCrop` transformation as a parameter.

```
ImageView imageAvatar = findViewById(R.id.imageAvatar);
Glide.with(this)
        .load(AVATAR_URL)
        .transform(new CircleCrop())
        .into(imageAvatar);
```

Now when we launch the application, the avatar image will be a circle.



Animate the image

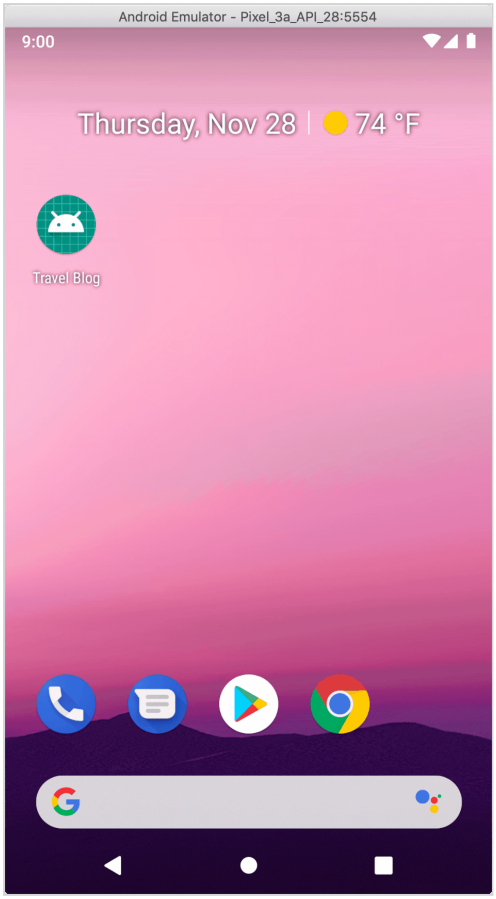
To make an image appear a bit smoother, we can make it appear with a fade animation. Once again, fortunately, Glide already has this built-in.

We can use `transition` method with `DrawableTransitionOptions.withCrossFade*()` transition as a parameter.

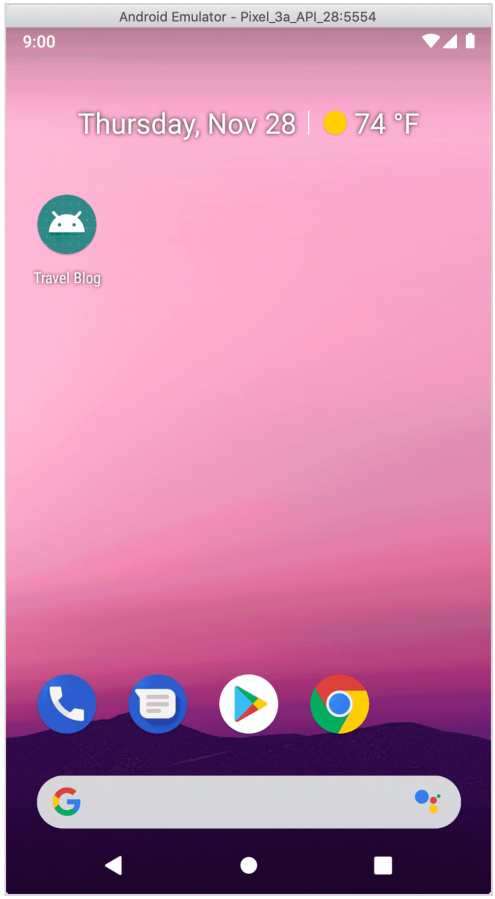
```
ImageView imageMain = findViewById(R.id.imageMain);
Glide.with(this)
        .load(IMAGE_URL)
        .transition(DrawableTransitionOptions.withCrossFade())
        .into(imageMain);

ImageView imageAvatar = findViewById(R.id.imageAvatar);
Glide.with(this)
        .load(AVATAR_URL)
        .transform(new CircleCrop())
        .transition(DrawableTransitionOptions.withCrossFade())
        .into(imageAvatar);
```

Now when we launch the application, images will appear with a fade animation.

Before:



After:



Hit the *run* button to try it yourself.

```
package com.travelblog;

import android.os.Bundle;
import android.widget.ImageView;
import android.widget.RatingBar;
import android.widget.TextView;

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

import com.bumptech.glide.Glide;
import com.bumptech.glide.load.resource.bitmap.CircleCrop;
import com.bumptech.glide.load.resource.drawable.DrawableTransitionOptions;

public class BlogDetailsActivity extends AppCompatActivity {

    public static final String IMAGE_URL = "https://bitbucket.org/dmytrodanylyk/travel-blog-r
    public static final String AVATAR_URL = "https://bitbucket.org/dmytrodanylyk/travel-blog-

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

        ImageView imageMain = findViewById(R.id.imageMain);
        Glide.with(this)
                .load(IMAGE_URL)
                .transition(DrawableTransitionOptions.withCrossFade())
```

```
                .into(imageMain);

        ImageView imageAvatar = findViewById(R.id.imageAvatar);
        Glide.with(this)
                .load(AVATAR_URL)
                .transform(new CircleCrop())
                .transition(DrawableTransitionOptions.withCrossFade())
                .into(imageAvatar);

        ImageView imageBack = findViewById(R.id.imageBack);
        imageBack.setOnClickListener(v -> finish());

        TextView textTitle = findViewById(R.id.textTitle);
        textTitle.setText("G'day from Sydney");

        TextView textDate = findViewById(R.id.textDate);
        textDate.setText("August 2, 2019");

        TextView textAuthor = findViewById(R.id.textAuthor);
        textAuthor.setText("Grayson Wells");

        TextView textRating = findViewById(R.id.textRating);
        textRating.setText("4.4");

        TextView textViews = findViewById(R.id.textViews);
        textViews.setText("(2687 views)");

        TextView textDescription = findViewById(R.id.textDescription);
        textDescription.setText("Australia is one of the most popular travel destinations in

        RatingBar ratingBar = findViewById(R.id.ratingBar);
        ratingBar.setRating(4.4f);
    }
}
```

The next lesson introduces how to load and display data from the Internet.