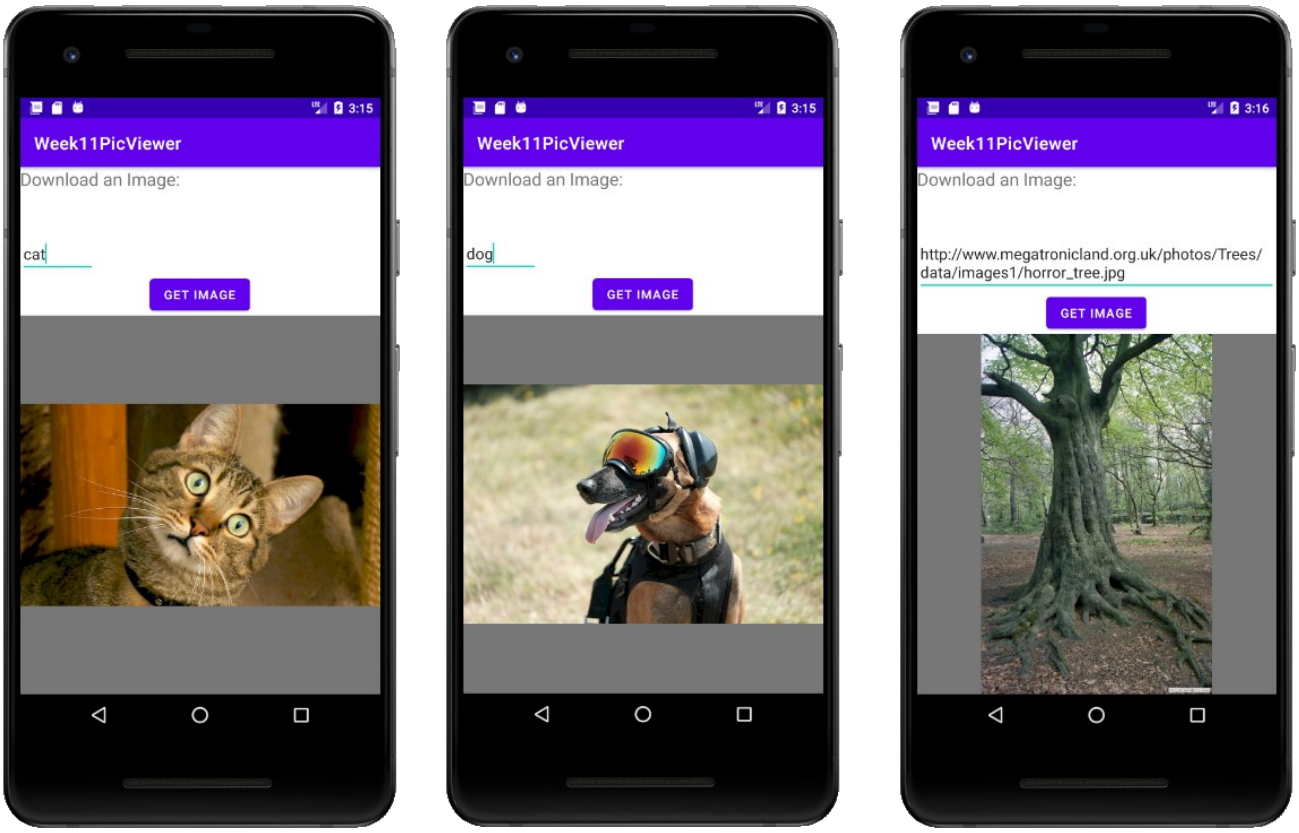
Android Application Development, COMP 10073

Mohawk College, Winter 2021

Fetching Images

Projects generally are more interesting when they include images. Several of the suggested WebAPIs include images as part of their data base. Downloading these images often makes for an interesting Andriod Project.

Today's demo will illustrate how to construct a basic application using the details that we already talked about in the AsyncTask lecture.



Our application will show a cat by default, or if the user asks for a cat. We will also show a dog, if asked for, and if you can type in a full http address, we can look up any image asked for.

You can paste strings into the AVD, by first copying the string from your environment, and then clicking and holding on the field until past comes up.

activity_main.xml

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Download an Image:"
        android:textSize="20sp"
        android:id="@+id/textView2" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp"
        android:hint="Enter URL" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Get Image"
        android:id="@+id/button"
        android:onClick="getImage"
        android:layout_centerHorizontal="true" />

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#777777"
        android:id="@+id/imageView" />
</LinearLayout>
```

We will need to access the MainActivity from the AsyncTask, set that up.

```
public class MainActivity extends AppCompatActivity {

    public static final String TAG = "==MainActivity==";
    private static Activity mainActivity;

    /**
     * onCreate - saves a reference to the current activity
     * @param savedInstanceState
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mainActivity = this;
        Log.d(TAG, "onCreate");
    }
```

Include a getter to return a reference to the main activity. We don't know how long the download will take, so we need a way for the download task to update the ImageView widget. Provide a simple getter and use it to update the image.

```
/**
 * @return a reference to this activity
 */
public static Activity getMainActivity() {
    return mainActivity;
}
```

Our onClick handler for the get image button includes a couple of default URLs. This simplifies the demonstration a little bit. The main work that this handler does is create the task for downloading the image and it passes the url to that task.

```
/**
 * onClick handler for button - read the editText and select an image source
 * @param view - default (unused)
 */
public void getImage(View view) {
    // Grab the URL from edit text.
    // If the supplied text is too short, or doesn't include http, download a cat
    EditText input = findViewById(R.id.editText);
    String url = input.getText().toString();
    if (url.toLowerCase().contains("dog")) {
        url = "https://images05.military.com/sites/default/files/2018-03/dog-goggles.jpg";
    } else if (url.length() < 3 || url.toLowerCase().contains("cat") ||
            !url.toLowerCase().contains("http")) {
        url = "http://i.ytimg.com/vi/Uk1RPEQI8mI/maxresdefault.jpg";
    }

    Log.d(TAG, "getImage = " + url);
    // We create and execute our AsyncTask
    // Pass in the URL of the image to display

    DownloadImageTask dl = new DownloadImageTask();
    dl.execute(url);
}
```

Don't forget to give the application the Internet permission, otherwise it won't work.

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

BitmapFactory

https://developer.android.com/reference/android/graphics/BitmapFactory

The BitmapFactory Creates Bitmap objects from various sources, including files, streams, and byte-arrays. It has a method that can convert an open stream into a bitmap image. Similar to the WebAPI demo we will use the AsyncTask to download data from a URL, the main difference here is using decodeStream()

DownloadImageTask.java

Use AsyncTask to download an image in the background. The only real difference between this version and the previous version is the use of BitmapFactory. Previously we used BufferedInputStream to read data one line at a time.

```
public class DownloadImageTask
        extends AsyncTask<String, Void, Bitmap> {

    public static String TAG = "==DownloadImageTask==";
    public static int HTTP_OK = 200;

    /**
     * Download an image as a background task
     * @param urls - expect the first string to be a URL of an image
     * @return null on failure, a bit mapped image otherwise
     */
    protected Bitmap doInBackground(String... urls) {

        // Use the URL Connection interface to download the URL
        Bitmap bmp = null;

        Log.d(TAG, "do background " + urls[0]);
        // URL connection must be done in a try/catch
        int statusCode = -1;
        try {
            URL url = new URL(urls[0]);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            statusCode = conn.getResponseCode();
            if (statusCode == HTTP_OK) {
                bmp = BitmapFactory.decodeStream(conn.getInputStream());
            }
        } catch (MalformedURLException e) {
            Log.d(TAG, "bad URL " + e);
        } catch (IOException e) {
            Log.d(TAG, "bad I/O " + e);
        }

        Log.d(TAG, "done " + statusCode);
        return bmp;
    }

    /**
     * after downloading is complete display the image
     * @param result - a bitmap image, null will skip the routine
     */
    protected void onPostExecute(Bitmap result) {
        Log.d(TAG,"onPostExecute()");
        if (result != null) {
            // Find the ImageView object to use
            Activity main = MainActivity.getMainActivity();
            ImageView imageView = (ImageView) main.findViewById(R.id.imageView);
            imageView.setImageBitmap(result);
        }
    }
}
```