VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, MACHHE BELAGAVI – 590018

KARNATAKA

A Mini-Project Report
On
"DOWNLOAD IMAGE USING ASYNC TASK"

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE MAD LABORATORRY WITH MINI PROJECT (18CSMP68) COURSE OF VIth SEMESTER

Submitted by

AISHWARYA S
[1CG18CS004]
NAGA PRIYA A N
[1CG18CS053]

Guide:
Mr.Anil kumar G M. Tech.,
Assoc & Coord Prof., Dept. of CSE
CIT,Gubbi

HOD:
Dr.Shantala C P Ph. D
Vice Principal & Head,
Dept. of CSE
CIT, Gubbi.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(NAAC Accredited & ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

2020-21

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(NAAC Accredited & ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

2020-21

CERTIFICATE

This is to certify that the project entitled **"DOWNLOAD IMAGE USING ASYNC TASK"** has been successfully carried out by **AISHWARYA S [1CG18CS004] & NAGA PRIYA A N [1CG18CS053]** in partial fulfillment for the VI semester during the academic year **2020 - 21**. It is certified that all the corrections / suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the VI semester.

Signature of guide

Mr. Anil Kumar G M.Tech,
Assoc & Coord Prof., Dept. of CSE,
CIT,Gubbi.

Signature of HOD

Dr. Shantala C P Ph. D
Vice Principal & Head,
Dept. of CSE,
CIT, Gubbi.

Signature of Principal

Dr. Suresh D S
Principal
CIT, Gubbi.

External Viva

Name of Examiners	**Signature with date**
1. _____	_____
2. _____	_____

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(NAAC Accredited & ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

2020-21

DECLARATION

We, **AISHWARYA S & NAGA PRIYA A N** student of VI Semester, **B E.,** in Computer Science and Engineering**, C.I.T, Gubbi**, hereby declare that the dissertation work entitled **"DOWNLOAD IMAGE USING ASYNC TASK"**, embodies the report of our project work carried out independently by us under the guidance of Mr. **ANIL KUMAR G**, Associate and coordinator Professor Department CSE, CIT, Gubbi, as partial fulfillment of requirements for the VI Semester during the academic year **2020-21**. We further declare that the project has not been submitted for the award of any other degree.

Place: GUBBI

NAME: AISHWARYA S
USN: 1CG18CS004
NAME: NAGA PRIYA A N
USN: 1CG18CS053

Date:

ABSTRACT

“Download Image Using Async Task” Android AsyncTask is an abstract class provided by Android which gives us the liberty to perform heavy tasks in the background and keeps the UI thread light thus making the application more responsive.

Android application runs on a single thread when launched. Due to this single thread model tasks that take longer time to fetch the response can make the application non-responsive. To avoid this we use android AsyncTask to perform the heavy tasks in background on a dedicated thread and passing the results back to the UI thread. Hence use of AsyncTask in android application keeps the UI thread responsive at all times.

ACKNOWLEDGEMENT

A great deal of time and lot of effort has gone into completing this project report and documenting it. The number of hours spent in getting through various books and other materials related to this topic chosen by me have reaffirmed its power and utility in doing this project.

Several special people have contributed significantly to this effort. First, we are grateful to our institution **Channabasaveshwara Institute of Technology**, Gubbi, which provides us an opportunity in fulfilling my most cherished desire of reaching the goal.

We acknowledge and express our sincere thanks to our beloved Principal and Director **Dr. Suresh D S** for his many valuable suggestions and continuous encouragement and support in the academic endeavors.

We express our sincere gratitude to **Dr. Shantala C P,** Vice Principal & Head, Department of CSE, for providing her constructive criticisms and suggestions.

We wish to express our deep sense of gratitude to **Mr. Anil Kumar G** Department of Computer Science and Engineering for all the guidance and who still remains a constant driving force and motivated through innovative ideas with tireless support and advice during the course of project to examine and helpful suggestions offered, which has contributed immeasurably to the quality of the final report.

Project Associates:

AISHWARYA S [1CG18CS004]

NAGA PRIYA A N [1CG18CS053]

CONTENTS

CHAPTER 1

INTRODUCTION

ANDROID STUDIO

Android is an open source and Linux-based **Operating System** for mobile devices such as smart phones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.

The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 **Jelly Bean**. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and performance.

The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.

1.1 History of Android Studio

The code names of android ranges from A to N currently, such as Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwitch, Jelly Bean, KitKat, Lollipop and Marshmallow. Let's understand the android history in a sequence.

Android Applications

Android applications are usually developed in the Java language using the Android Software Development Kit.

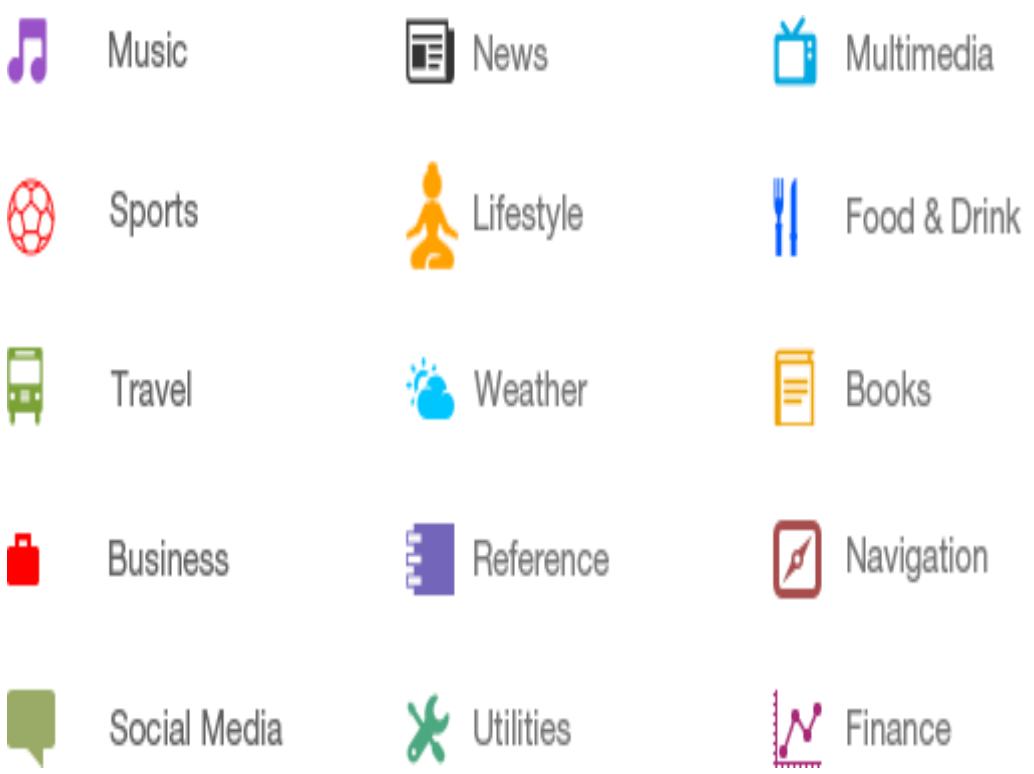
Once developed, Android applications can be packaged easily and sold out either through a store such as **Google Play**, **SlideME**, **Opera Mobile Store**, **Mobango**, **F-droid** and the **Amazon Appstore**.

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast. Every day more than 1 million new Android devices are activated worldwide.

This tutorial has been written with an aim to teach you how to develop and package Android application. We will start from environment setup for Android application programming and then drill down to look into various aspects of Android applications.

Categories of Android applications

There are many android applications in the market. The top categories are –



CHAPTER 2

INTRODUCTION TO ANDROID STUDIO

Android is an operating system and programming platform developed by Google for mobile phones and other mobile devices, such as tablets. It can run on many different devices from many different manufacturers. Android includes a software development kit (SDK) that helps you write original code and assemble software modules to create apps for Android users. Android also provides a marketplace to distribute apps. All together, Android represents an *ecosystem* for mobile apps.

Developers create apps for a variety of reasons. They may need to address business requirements or build new services or businesses, or they may want to offer games and other types of content for users. Developers choose to develop for Android in order to reach the majority of mobile device users.

Most popular platform for mobile apps

As the world's most popular mobile platform, Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It has the largest installed base of any mobile platform and is still growing fast. Every day another million users power up their Android-powered devices for the first time and start looking for apps, games, and other digital content.

Android provides a touch screen user interface (UI) for interacting with apps. Android's UI is mainly based on direct manipulation. People use touch gestures such as swiping, tapping, and pinching to manipulate on-screen objects. In addition to the keyboard, there's a customizable on-screen keyboard for text input. Android can also support game controllers and full-size physical keyboards connected by Bluetooth or USB.

Android Studio offers even more features that enhance your productivity when building Android apps, such as:

2 A flexible Gradle-based build system

3 A fast and feature-rich emulator

4 A unified environment where you can develop for all Android devices

5 Apply Changes to push code and resource changes to your running app without restarting your app

6 Code templates and GitHub integration to help you build common app features and import sample code

7 Extensive testing tools and frameworks

8 Lint tools to catch performance, usability, version compatibility, and other problems

9 C++ and NDK support

10 Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine.

CHAPTER 3

BASIC COMPONENTS OF ANDROID STUDIO

There are some necessary building blocks that an Android application consists of. These loosely coupled components are bound by the application manifest file which contains the description of each component and how they interact. The manifest file also contains the app's metadata, its hardware configuration, and platform requirements, external libraries, and required permissions. There are the following main components of an android app:

1. Activities

Activities are said to be the presentation layer of our applications. The UI of our application is built around one or more extensions of the Activity class. By using Fragments and Views, activities set the layout and display the output and also respond to the user's actions. An activity is implemented as a subclass of class Activity. (Java and Kotlin)

2. Services

Services are like invisible workers of our app. These components run at the backend, updating your data sources and Activities, triggering Notification, and also broadcast Intents. They also perform some tasks when applications are not active. A service can be used as a subclass of class Service:

3. Content Providers

It is used to manage and persist the application data also typically interacts with the SQL database. They are also responsible for sharing the data beyond the application boundaries. The Content Providers of a particular application can be configured to allow access from other applications, and the Content Providers exposed by other applications can also be configured. A content provider should be a sub-class of the class Content Provider.

4. Broadcast Receivers

They are known to be intent listeners as they enable your application to listen to the Intents that satisfy the matching criteria specified by us. Broadcast Receivers make our application react .

5. Intents

It is a powerful inter-application message-passing framework. They are extensively used throughout Android. Intents can be used to start and stop Activities and Services, to broadcast messages system-wide or to an explicit Activity, Service or Broadcast Receiver or to request action be performed on a particular piece of data.

6. Widgets

These are the small visual application components that you can find on the home screen of the devices. They are a special variation of Broadcast Receivers that allow us to create dynamic, interactive application components for users to embed on their Home Screen.

7. Notifications

Notifications are the application alerts that are used to draw the user's attention to some particular app event without stealing focus or interrupting the current activity of the user. They are generally used to grab user's attention when the application is not visible or active, particularly from within a Service or Broadcast Receiver. Examples: E-mail popups, Messenger popups, etc.

CHAPTER 4

WORKING PROCEDURE OF ANDROID ASYNC TASK

Android AsyncTask is an abstract class provided by Android which gives us the liberty to perform heavy tasks in the background and keeps the UI thread light thus making the application more responsive.

Android application runs on a single thread when launched. Due to this single thread model tasks that take longer time to fetch the response can make the application non-responsive. To avoid this we use android AsyncTask to perform the heavy tasks in background on a dedicated thread and passing the results back to the UI thread. Hence use of AsyncTask in android application keeps the UI thread responsive at all times.

The basic methods used in an android AsyncTask class are defined below:

- **doInBackground()** : This method contains the code which needs to be executed in background. In this method we can send results multiple times to the UI thread by publishProgress() method. To notify that the background processing has been completed we just need to use the return statements
- **onPreExecute()** : This method contains the code which is executed before the background processing starts
- **onPostExecute()** : This method is called after doInBackground method completes processing. Result from doInBackground is passed to this method
- **onProgressUpdate()** : This method receives progress updates from doInBackground method, which is published via publishProgress method, and this method can use this progress update to update the UI thread

The three generic types used in an android AsyncTask class are given below:

- **Params** : The type of the parameters sent to the task upon execution
- **Progress** : The type of the progress units published during the background computation
- **Result** : The type of the result of the background computation

Project structure

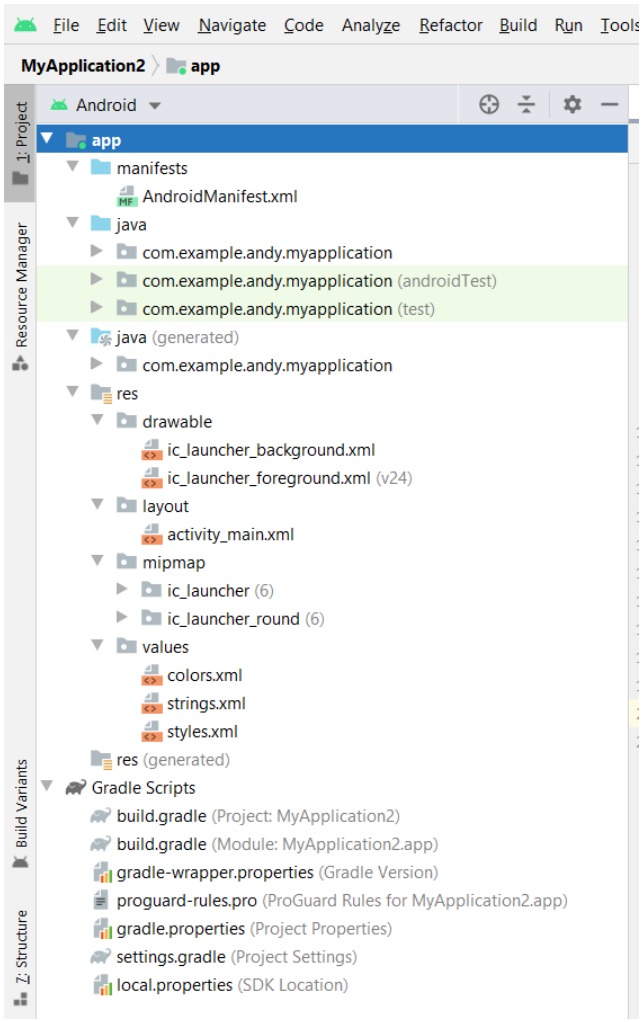


Figure.4.1 The project files in Android view.

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure4. 1.This view is organized by modules to provide quick access to your project's key source files.

All the build files are visible at the top level under Gradle Scripts and each app module contains the following folders:

- **manifests**: Contains the `AndroidManifest.xml` file.
- **java**: Contains the Java source code files, including JUnit test code.
- **res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select **Project** from the **Project** dropdown (in figure 1, it's showing as **Android**).

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the **Problems** view of your project displays links to the source files containing any recognized coding and syntax errors, such as a missing XML element closing tag in a layout file.

CHAPTER 5

PROJECT DESIGN AND IMPLEMENTATION

Android AsyncTask

Async task enables you to implement Multithreading without get Hands dirty into threads. AsyncTask enables proper and easy use of the UI thread. It allows performing background operations and passing the results on the UI thread. If you are doing something isolated related to UI, for example downloading data to present in a list, go ahead and use AsyncTask.

- AsyncTasks should ideally be used for short operations (a few seconds at the most.)
- An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps,
  called onPreExecute, doInBackground, onProgressUpdate and onPostExecute.
- In onPreExecute you can define code, which need to be executed before background processing starts.
- doInBackground have code which needs to be executed in background, here in doInBackground we can send results to multiple times to event thread by publishProgress() method, to notify background processing has been completed we can return results simply.
- onProgressUpdate() method receives progress updates from doInBackground method, which is published via publishProgress method, and this method can use this progress update to update event thread
- onPostExecute() method handles results returned by doInBackground method.
- The generic types used are
  - Params, the type of the parameters sent to the task upon execution
  - Progress, the type of the progress units published during the background computation.

  - Result, the type of the result of the background computation.

- An running async task can be cancelled by calling cancel (Boolean) method.

AsyncTask has multiple events that can be overridden to control different behavior:

- `onPreExecute` - executed in the main thread to do things like create the initial progress bar view.
- `doInBackground` - executed in the background thread to do things like network downloads.
- `onProgressUpdate` - executed in the main thread when `publishProgress` is called from doInBackground.
- `onPostExecute` - executed in the main thread to do things like set image views.

Limitations

An `AsyncTask` is tightly bound to a particular `Activity`. In other words, if the `Activity` is destroyed or the configuration changes then the `AsyncTask` will not be able to update the UI on completion. As a result, for short one-off background tasks **tightly coupled to updating an Activity**, we should consider using an `AsyncTask` as outlined above. A good example is for a several second network request that will populate data into a `ListView` within an activity.

Custom Thread Management

Using the `AsyncTask` is the easiest and most convenient way to manage background tasks from within an Activity. However, in cases where tasks need to be processed in parallel with more control, or the tasks **need to continue executing even when the activity leaves the screen**, you'll need to create a background service or manage threaded operations more manually.

IMPLEMENTATION

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:id="@+id/rootview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#c1c1c1"
    android:gravity="center_horizontal"
    tools:context=".MainActivity">

<Button
    android:id="@+id/asyncTask"
    android:text="Download"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<ImageView
    android:id="@+id/image"
    android:layout_width="300dp"
    android:layout_height="300dp" />
</LinearLayout>
```

MainActivity.java

```
package com.example.andy.myapplication;
import android.app.ProgressDialog;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class MainActivity extends AppCompatActivity {
    URL ImageUrl = null;
    InputStream is = null;
    Bitmap bmImg = null;
    ImageView imageView= null;
    ProgressDialog p;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button=findViewById(R.id.asyncTask);
        imageView=findViewById(R.id.image);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                AsyncTaskExample asyncTask=new AsyncTaskExample();
                asyncTask.execute("https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcQCfIvuZY83o2YquRzJ3mRu6J1hXnmonUf05Q&
usqp=CAU");
            }
        });

    }

    private class AsyncTaskExample extends AsyncTask<String, String, Bitmap> {
```

```
@Override
protected void onPreExecute() {
   super.onPreExecute();
   p=new ProgressDialog(MainActivity.this);
   p.setMessage("Please wait...It is downloading");
   p.setIndeterminate(false);
   p.setCancelable(false);
   p.show();
}

@Override
protected Bitmap doInBackground(String... strings) {
   try {

      ImageUrl = new URL(strings[0]);
      HttpURLConnection conn = (HttpURLConnection) ImageUrl
            .openConnection();
      conn.setDoInput(true);
      conn.connect();
      is = conn.getInputStream();

      BitmapFactory.Options options = new BitmapFactory.Options();
      options.inPreferredConfig = Bitmap.Config.RGB_565;
      bmImg = BitmapFactory.decodeStream(is, null, options);

   } catch (IOException e) {
      e.printStackTrace();
   }

   return bmImg;
}

@Override
protected void onPostExecute(Bitmap bitmap) {
```

```
        super.onPostExecute(bitmap);
        if(imageView!=null) {
            p.hide();
            imageView.setImageBitmap(bitmap);
        }else {
            p.show();
        }
    }
  }
}
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.andy.myapplication">
   <uses-permission android:name="android.permission.INTERNET"/>
 <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">
      <activity android:name=".MainActivity">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
</manifest>
```

5.1 Hardware and Software requirements:

Steps to Create an Android Project

If you want to build an Android application, you must be ready to invest your resources to ensure your project becomes a success.

As a developer, before installing the Android studio, ensure that your system meets the following minimum requirements.

Minimum System Requirements for Android Studio

- Microsoft Windows 7/8/10 (32 or 64 bit).
- Mac OS X 10.8.5.
- GNOME or KDE or Unity desktop on Ubuntu or Fedora or GNU/Linux Debian.
- 2GB RAM.
- 4GB RAM recommended.
- 500 MB disk space
- 1 GB for Android SDK.
- Java Development Kit (JDK) 7.
- 1280x800 screen resolution.
- A faster processor (according to your budget).

CHAPTER 6

SNAPSHOTS OF DOWNLOAD IMAGE USING ASYNC TASK

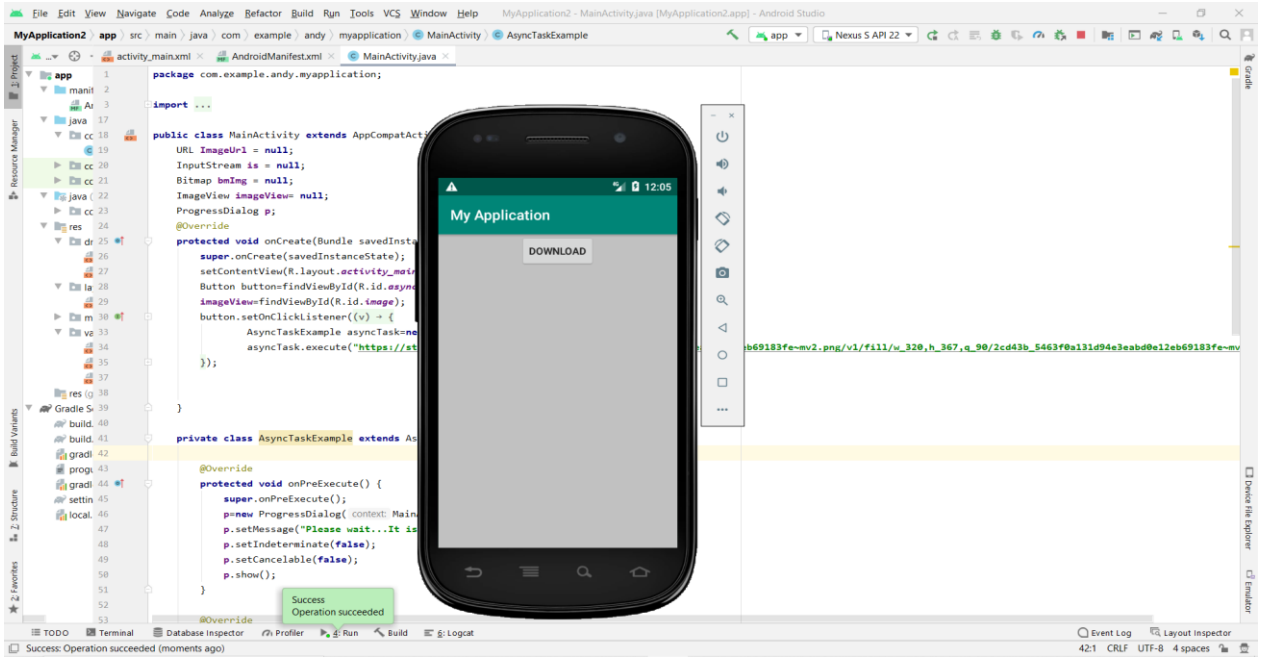


Fig.6.1 Launching Activity

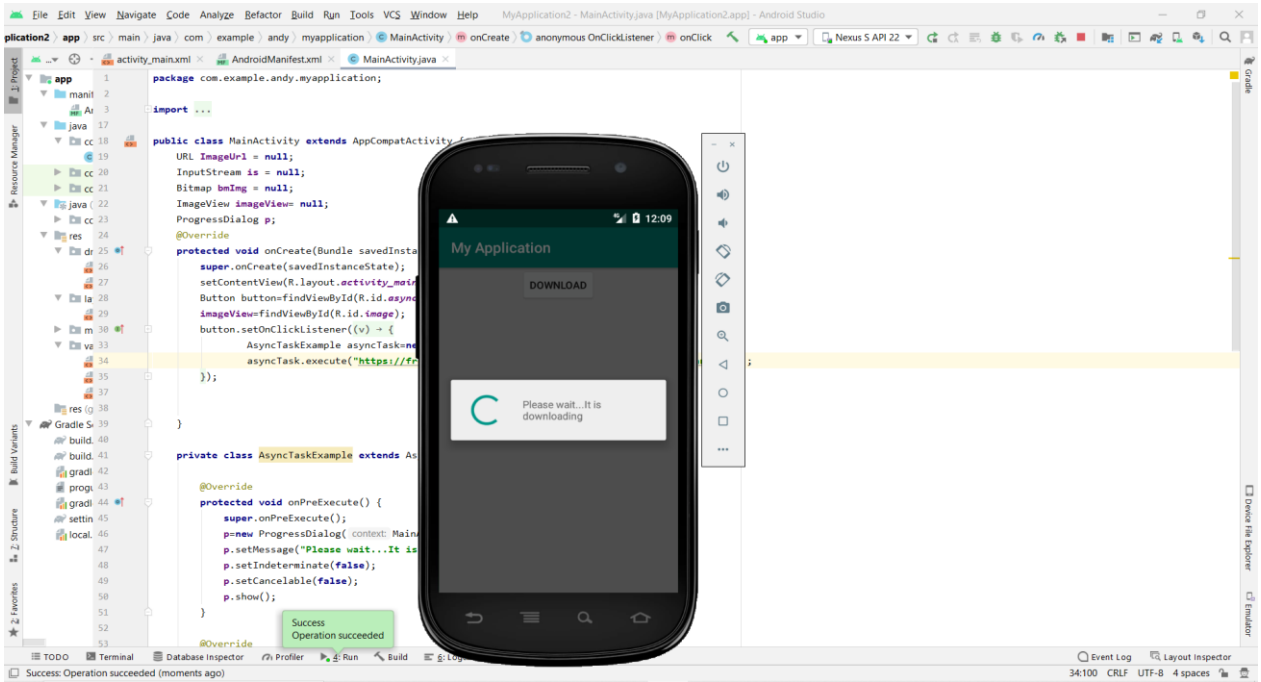


Fig. 6.2 Downloading Image

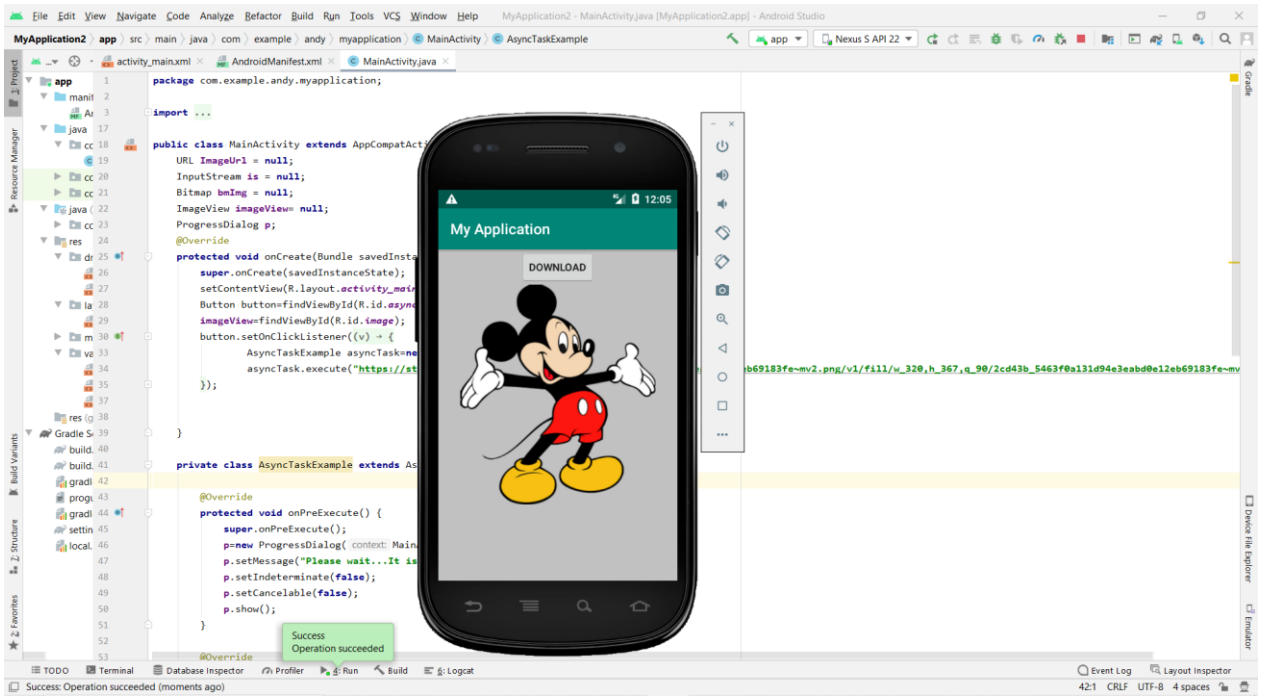


Fig. 6.3 Downloaded Image

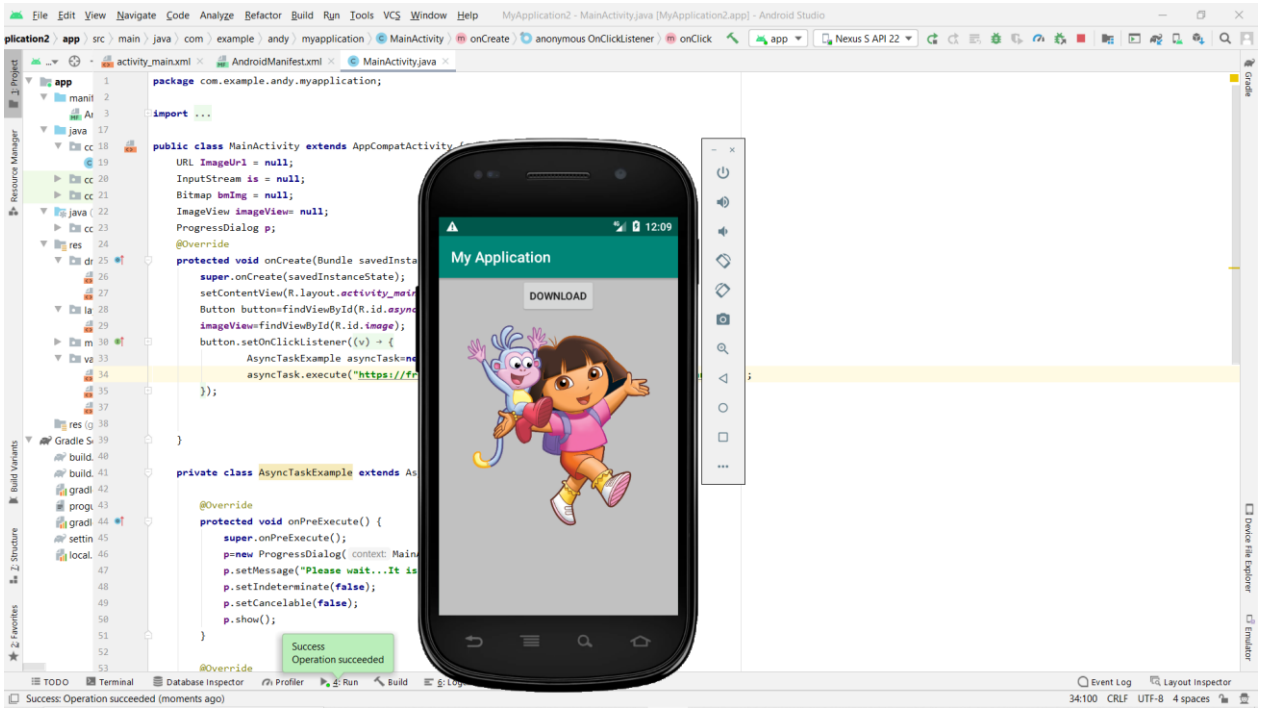


Fig. 6.4 Downloaded Image

CHAPTER 7

CONCLUSION

Async task enables you to implement MultiThreading without get Hands dirty into threads. AsyncTask enables proper and easy use of the UI thread. It allows performing background operations and passing the results on the UI thread. If you are doing something isolated related to UI, for example downloading data to present in a list, go ahead and use AsyncTask.

- AsyncTasks should ideally be used for short operations (a few seconds at the most.)
- An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps,
  called onPreExecute, doInBackground, onProgressUpdate and onPostExecute.
- In onPreExecute you can define code, which need to be executed before background processing starts.
- doInBackground have code which needs to be executed in background, here in doInBackground we can send results to multiple times to event thread by publishProgress() method, to notify background processing has been completed we can return results simply.
- onProgressUpdate() method receives progress updates from doInBackground method, which is published via publishProgress method, and this method can use this progress update to update event thread
- onPostExecute() method handles results returned by doInBackground method.

CHAPTER 8

BIBLIOGRAPHY

During the development of the project, we have used many resources and for that we are grateful to all the people concerned.

Given below are the names of the some websites, which we have consulted during thedevelopment and documentation of the project.

Websites:

1.https://www.sites.google.com

- Introduction
- Scope

2.http://en.wikipedia.org/wiki/AsyncTask

3.https://www.vogella.com/tutorials/AndroidListView/article.html

- Android and with ListView widget
- Android Adapter