

Extract text on image and translate android app source code is available now.

See example in Play Store



# Inducesmile

Mobile developer community for beginners and students

[ANDROID](#)
[KOTLIN](#)
[IOS \(SWIFT\)](#)
[HYBRID APP ▼](#)
[MOBILE GAMES](#)
[IOT - VR](#)
[TIPS](#)
[CODING HELP](#)
[BUY CODE](#)

[Home](#) ▶ [Android](#) ▶ Android EventBus Library Example



# ANDROID EVENTBUS LIBRARY EXAMPLE

In this android tutorial, we are going to explore android EventBus. EventBus is an open-source library for Android using the publisher/subscriber pattern for loose coupling. EventBus enables central communication to decoupled classes with just a few lines of code – simplifying the code, removing dependencies, and speeding up app development.

If you have work with android components you will realized that there are many way you can use to communicate between different components.

When the communication is between two different Activity classes we usually use Intent class. For communication between Activity and a child Fragment, there are few options available to use. We can store the message in a bundle or we can create an interface to is connect to the parent Activity or by creating a set method.



Scan & Translate Android App  
Source code

[VIEW APP SAMPLE](#)

[WRITE FOR US](#)

## ANDROID SOURCE CODE POLL

Which of the below listed  
source codes should we publish  
next month?

Multi Restaurant Food  
Ordering App (27%, 39 Votes)

Stock and Inventory App  
(18%, 26 Votes)

Android Dating App (16%, 23

Accept

The list goes on. So there is no simple and single way to achieve this kind of communication in a decouple way in android. This is where Android EventBus goes to our need.

Android EventBus Library was created by a company name Greenrobot. You can read more about android [EventBus](#) in their developer's use guide.

To understand the benefits that EventBus brings to the table when developing android application, we have listed it below

1. Simplifies the communication between components
2. Decouples event senders and receivers
3. Performs well with Activities, Fragments, and background threads
4. Avoids complex and error-prone dependencies and life cycle issues
5. Is fast; specifically optimized for high performance
6. Is tiny (<50k jar)
7. Is proven in practice by apps with 100,000,000+ installs
8. Has advanced features like delivery threads, subscriber priorities, etc.

Android EventBus library works like a Publisher / Subscriber design pattern where by the publisher emits events and the subscribers consumes the events. The subscribers take action based on the emitted event.

The Publishers uses the EventBus class static method getDefault() to instance a single copy of the EventBus in the application. Although you can create multiple instance if you want to.

The Event Message Object is a plain Java POJO which contains the message intended to be passed through the event bus from the Publisher to Subscribers. Multiple subscribers can subscribe to a single publisher. The subscribers method are denoted with @Subscriber annotation. The method also accepts an instance of the event message object as parameter.

In this exercise, we are going to send an event message from a

Android E-Book App (12%, 17 Votes)

☐

Appointment Booking App (10%, 15 Votes)

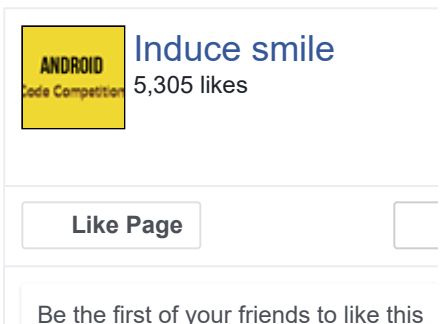
☐

Business & Service Provider Finder (6%, 9 Votes)

☐

Total Voters: **147**

## LIKE US ON FACEBOOK



## RECENT POSTS

- Camera Translator All Language App
- Why MVVM and How to execute MVVM combined with Data Binding and LiveData in Four Simple Steps
- Camera Translator All Languages – Android Project idea source code for students
- Note Scan Android App -Project idea source code for students
- How To Build Android WhatsApp Clone Chat Application
- Android Car Rental App with Web Admin Panel – Project Idea Source Code For Students

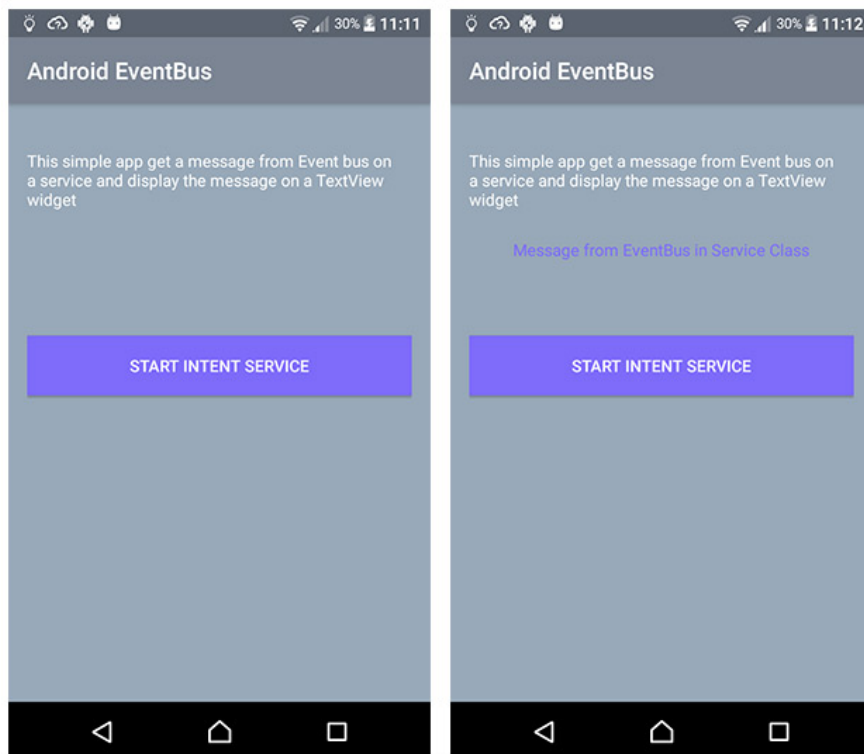
Accept

With the information we had so far, we will proceed to create an android application that will show case how to use android EventBus library.

In other to get a visual understanding of what we are going to create in this android tutorial, I have add below some screen shots from the application.

- Android Retrofit 2 with JSON API Example
- Android Room with LiveData Example Tutorial
- Android UI App for Food Ordering and Delivery from Multiple Restaurants

## SOME SCREENSHOT FROM THE APPLICATION



## CREATE NEW ANDROID PROJECT

Lets start to soil our hands in code. Start up your IDE. For this tutorial, I am using the following tools and environment, feel free to use what works for you.

Windows 10

Android Studio

Accept

Target SDK 25

To create a new android application project, follow the steps as stipulated below.

Go to File menu

Click on New menu

Click on Android Application

Enter Project name: AndroidEventBus

Package: com.inducesmile.androideventbus

Select Empty Activity

Name your activity: MainActivity

Keep other default selections

Continue to click on next button until Finish button is active, then click on Finish Button.

## ADD LIBRARY DEPENDENCIES IN BUILD.GRADLE

Open the module build.gradle, since we are going to use EventBus library, we are going to add the library to the dependencies section. This will make it possible for our android project to access all the classes and method that are exposed in this library.

Once you open build.gradle file, add the code below to the file.

```
apply plugin: 'com.android.application'
```



```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 25
```



Accept

```
targetSdkVersion 25
versionCode 1
versionName "1.0"
testInstrumentationRunner "android.support.test.
}
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard
    }
}
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso
        exclude group: 'com.android.support', module: 's
    })
    compile 'com.android.support:appcompat-v7:25.0.0'
    compile 'org.greenrobot:eventbus:3.0.0'
    testCompile 'junit:junit:4.12'
}
```

## STRINGS.XML

We are going to update our project strings.xml file located in the values folder inside the res folder. Open the file and add the code below to it.

```
<resources>
    <string name="app_name">Android EventBus</string>
    <string name="event_message">This simple app gets a
    <string name="start_intent_service">START INTENT SER
    <string name="event_message_from_service">Message fr
</resources>
```

## COLORS.XML

Open the colors.xml file in the same location as the strings.xml file and add the code below to the file.

Accept

```

<color name="colorPrimaryDark">#5b636f</color>
<color name="colorAccent">#7E6CFA</color>
<color name="colorLight">#98A9B9</color>
<color name="colorWhite">#ffffff</color>
<color name="colorBlack">#000000</color>
</resources>

```

## ACTIVITY\_MAIN.XML

Let's create the MainActivity class layout file. As stated above, this layout file will contain two TextView, and a Button widgets. When the Button is clicked, it starts a background Service which will post message back to the Activity class and this message is displayed in one of the TextView widgets.

Open this layout file and add the code below.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.co
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_marg
    android:paddingLeft="@dimen/activity_horizontal_marg
    android:paddingRight="@dimen/activity_horizontal_mar
    android:paddingTop="@dimen/activity_vertical_margin"
    android:background="@color/colorLight"
    tools:context="com.inducesmile.androideventbus.MainA
    <TextView
        android:id="@+id/text_info"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/event_message"
        android:layout_marginTop="24dp"
        android:layout_centerHorizontal="true"
        android:textColor="@color/colorWhite"
        android:layout_alignParentTop="true" />
    <TextView
        android:id="@+id/event_bus_text"
        . . . . .
    . . . . .

```

Accept

```
        android:textColor="@color/colorAccent" />
    <Button
        android:id="@+id/service_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp"
        android:layout_below="@+id/event_bus_text"
        android:background="@color/colorAccent"
        android:textColor="@color/colorWhite"
        android:text="@string/start_intent_service"
        android:layout_marginTop="64dp"
        android:layout_centerHorizontal="true"/>
</RelativeLayout>
```

## MAINACTIVITY CLASS

Open the MainActivity class, in this class we will get the instances of our TextView and Button widgets.

The button object is wired with a click listener event. Inside the onClick() method of the Listener class, an IntentService is started.

If you do know much about android Service class, you can read it up in android developers documentation.

IntentService runs on a different thread. So to update a UI view from another thread we will use the EventBus to achieve this.

A messageEventFromService() method which is annotate with @Subscribe is a subscriber to the event publisher.

Inside the method, the event message passed is obtained and assigns as a value to TextView.

Open MainActivity class and add the code below.

```
import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
```

&lt;/&gt;

Accept

```
import org.greenrobot.eventbus.ThreadMode;
public class MainActivity extends AppCompatActivity {
    private static final String TAG = MainActivity.class
    private TextView displayTextView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        displayTextView = (TextView)findViewById(R.id.ev
        Button serviceButton = (Button)findViewById(R.id
        serviceButton.setOnClickListener(new View.OnClic
        @Override
        public void onClick(View view) {
            Intent serviceIntent = new Intent(MainAc
            startService(serviceIntent);
        }
    });
}
@Subscribe(threadMode = ThreadMode.MAIN)
public void messageEventFromService(EventMessage eve
    displayTextView.setText(event.getNotification())
}
@Override
protected void onStart() {
    super.onStart();
    EventBus.getDefault().register(this);
}
@Override
protected void onStop() {
    EventBus.getDefault().unregister(this);
    super.onStop();
}
}
```

## MYSERVICE CLASS

In the default project package, create a new java class and name it MyService.java. This class will extends IntentService class and override the onHandleIntent(); This is where the intent the component that started it is obtained.

Accept



getDefault() method of the EventBus. The post() method is call with the Event Message object passed as parameter.

Open this class and add the code below.

```
import android.app.IntentService;
import android.content.Intent;
import org.greenrobot.eventbus.EventBus;
public class MyService extends IntentService {
    public MyService() {
        super("MY SERVICE");
    }
    @Override
    protected void onHandleIntent(Intent intent) {
        EventBus.getDefault().post(new EventMessage(getS
    }
}
```

## EVENTMESSAGE CLASS

Create a new java class file like above but now name it EventMessage.java. This is a plain POJO java class.

Open the class and paste the code below.

```
public class EventMessage {
    private String notification;
    public EventMessage(String notification) {
        this.notification = notification;
    }
    public String getNotification() {
        return notification;
    }
}
```

This brings us to the end of this tutorial. I hope that you have learn something. Run your app and see for yourself.

Android Event Bus Example

Accept

Remember to subscribe with your email address to be among the first to receive my new android blog post once it is published.

## OTHER INTERESTING POSTS:

Android Code Competition  
Announcement

How to combine Android  
Navigation Drawer and  
Master Detail flow in  
Android

Android FirebaseUI  
Authentication Example

Android Animation Example  
Tutorial

Android Survey App with  
Web Admin Panel – Project  
Idea Source Code For  
Students

Why MVVM and How to  
execute MVVM combined  
with Data Binding and  
LiveData in Four Simple  
Steps

Tags: [android event bus](#), [android event bust](#), [android eventbus](#)


Accept



## Inducesmile

I learn and write about Android, iOS, Javascript, Php, Node.js, React Native, Mobile Game, Virtual Reality and Internet of Things



 **Inducesmile**  
**1,932**  
● 1 ● 9 ● 15

## 2 COMMENTS



**Kenny** | November 29, 2016

[Log in to Reply](#)

Hmm... this library comes handy indeed. Thanks for the eye opener @mr Henry. I will definitely try my hands on it.



**Henry** Author | November 29, 2016 [Log in to Reply](#)

Glad you like it. I do also. EventBus makes life simple

## ADD A COMMENT

You must be [logged in](#) to post a comment.

Accept



INDUCESMILE – ANDROID TUTORIAL, ANDROID APPS, ANDROID STUDIO, ANDROID SDK, ANDROID DEVELOPMENT COPYRIGHT © 2018.

[PROGRAMMING HELP](#)

[ANDROID TUTORIALS](#)

[CONTACT US](#)

[PRIVACY POLICY](#)

Accept