







# Android Workshop










Somkiat

Home










Somkiat Puisungnoen

Update Info 1
View Activity Log 10+
...


Timeline
About
Friends 3,138
Photos
More ▾



When did you work at Opendream?
×





...
22 Pending Items



Intro


Software Craftsmanship


Software Practitioner at สยามชำนาญกิจ พ.ศ. 2556



Agile Practitioner and Technical at SPRINT3r


Post

Photo/Video

Live Video

Life Event


What's on your mind?


Public ▾

Post



Somkiat Puisungnoen
15 mins · Bangkok · 🌐 ▾

Java and Bigdata





somkiat.cc



Somkiat

Home



Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

@somkiat.cc

Home

Posts

Videos

Photos



Help people take action on this Page. ✕



Liked ▾



Following ▾



Share



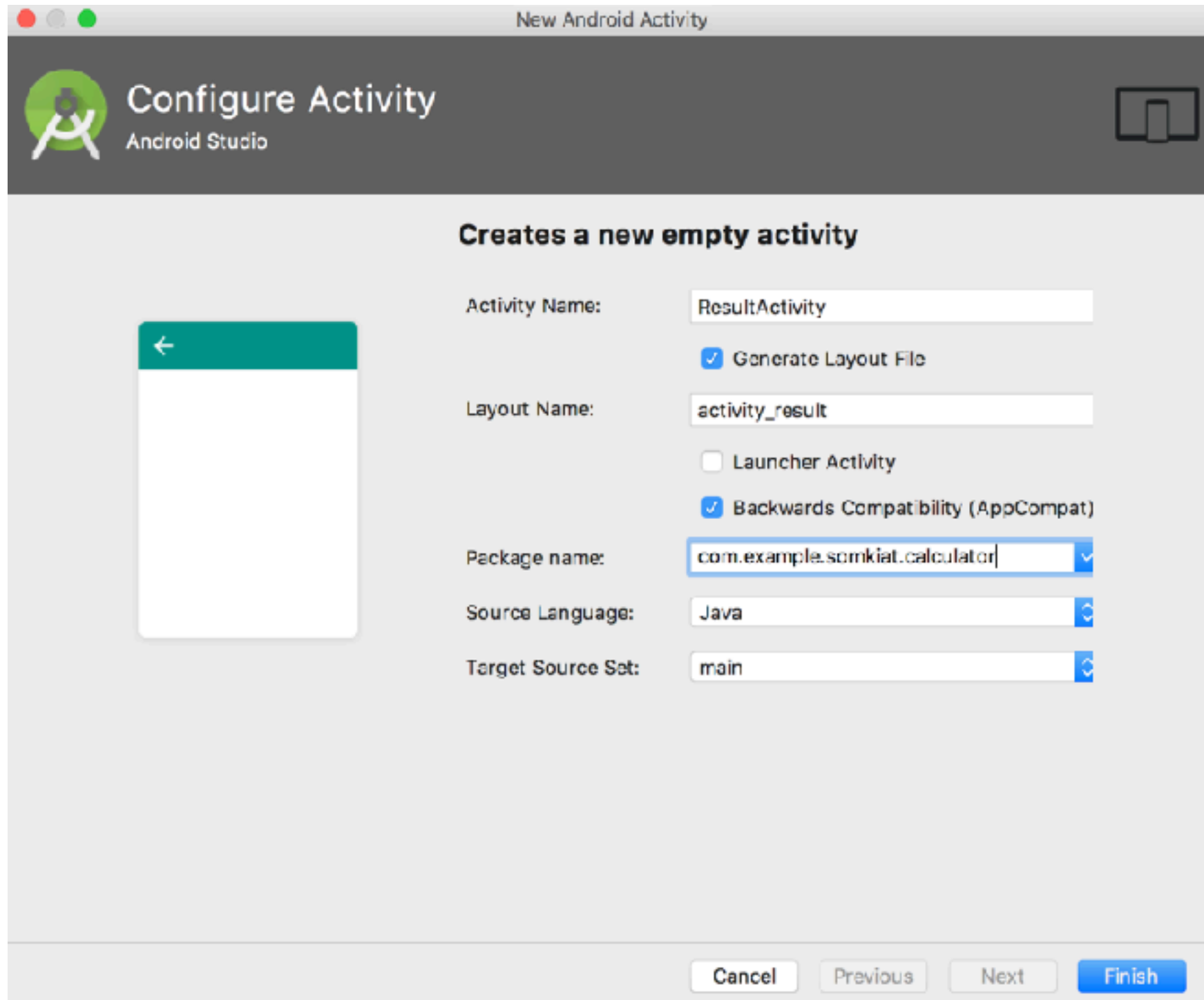
+ Add a Button




# Add Result Activity



# Add Result Activity



New Android Activity

 **Configure Activity**  
Android Studio

**Creates a new empty activity**

Activity Name:

☒ Generate Layout File

Layout Name:

☐ Launcher Activity

☒ Backwards Compatibility (AppCompat)

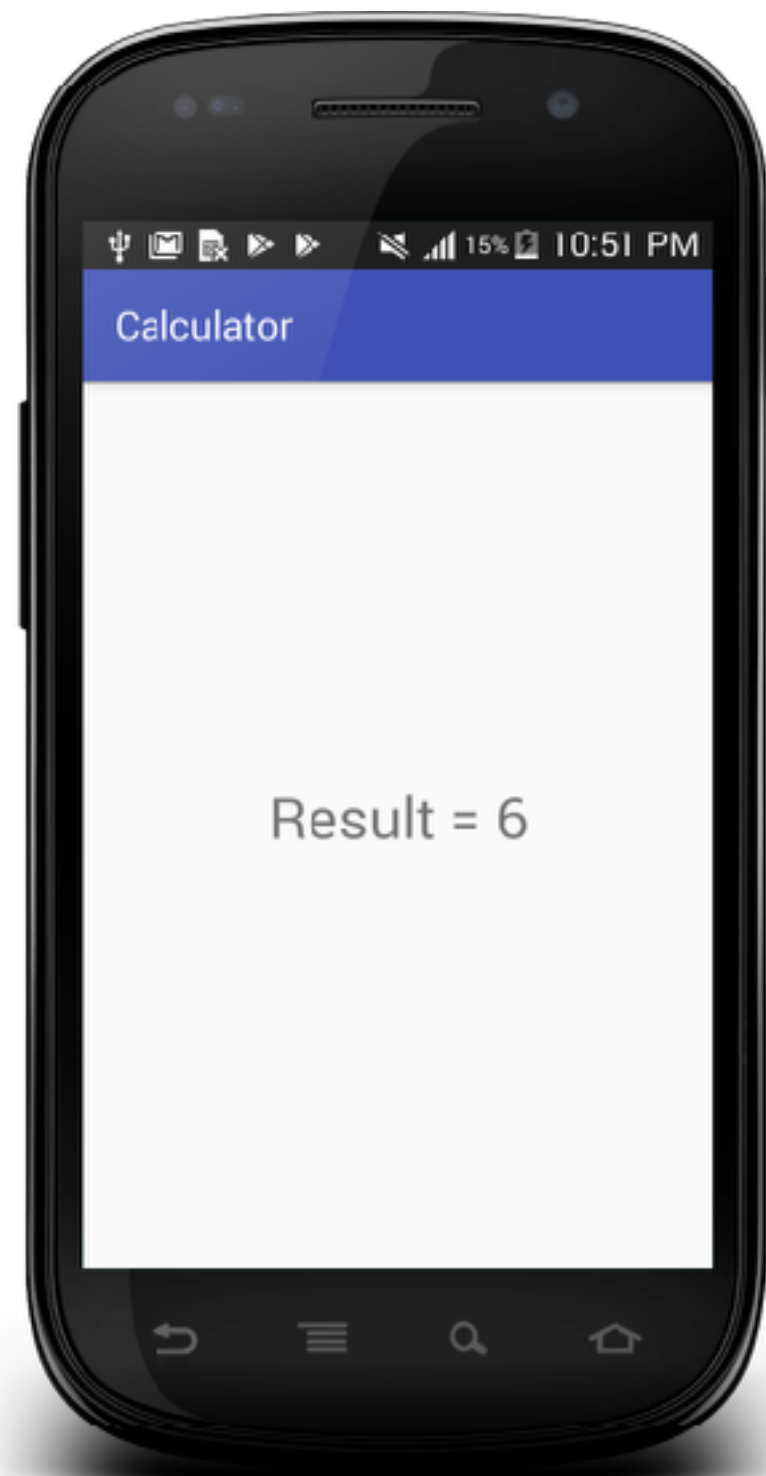
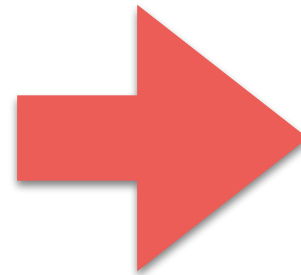
Package name:

Source Language:

Target Source Set:



# Calculator app





# Send data via Intent

```
// Send data to Result Activity
Intent intent =
    new Intent(packageContext: this,
               ResultActivity.class);
intent.putExtra(name: "result", String.valueOf(result));
startActivity(intent);
```

```
// Receive data from intent
Intent intent = getIntent();

TextView txvResult = findViewById(R.id.txv_result);
txvResult.setText(
    "Result = " + intent.getStringExtra(name: "result"));
```



Try to run UI tests  
\$gradlew cAT





# How to test the Result Activity ?



# ResultActivityTest (1)

Don't start the activity !!

```
@Rule
public ActivityTestRule<ResultActivity> activityTestRule
    = new ActivityTestRule<>(ResultActivity.class,
                             initialTouchMode: true,
                             launchActivity: false);
```

```
@Test
public void should_show_10_in_result_from_intent() {
    Intent intent = new Intent();
    intent.putExtra(name: "result", value: "10");
    activityTestRule.launchActivity(intent);

    onView(withId(R.id.txv_result))
        .check(matches(withText("Result = 10")));
}
```



# ResultActivityTest (2)

Add new test case and set data in Intent

```
@Rule
public ActivityTestRule<ResultActivity> activityTestRule
    = new ActivityTestRule<>(ResultActivity.class,
                             initialTouchMode: true,
                             launchActivity: false);
```

```
@Test
public void should_show_10_in_result_from_intent() {
    Intent intent = new Intent();
    intent.putExtra(name: "result", value: "10");
    activityTestRule.launchActivity(intent);

    onView(withId(R.id.txv_result))
        .check(matches(withText("Result = 10")));
}
```



Try to run UI tests  
\$gradlew cAT



# Result of UI Testing

## Package com.example.somkiat.calculator

all > com.example.somkiat.calculator

**3**  
tests

**0**  
failures

**3.610s**  
duration

**100%**  
successful

### Classes

Class	Tests	Failures	Duration	Success rate
<a href="#">ExampleInstrumentedTest</a>	1	0	0.030s	100%
<a href="#">MainActivityTest</a>	1	0	2.952s	100%
<a href="#">ResultActivityTest</a>	1	0	0.628s	100%



# Working with REST APIs



# Calculator app



<http://api.mathjs.org/>





# MATH.JS

Try to use !!

Try it

To try it, enter an expression below, then click the generated url.

**Expression**

1+2

**Url**

<http://api.mathjs.org/v4/?expr=1%2B2>



# Dependencies

Add Retrofit 2 and OkHttp 3 in build.gradle

```
//Network  
implementation 'com.squareup.retrofit2:retrofit:2.4.0'  
implementation 'com.squareup.retrofit2:converter-gson:2.4.0'  
implementation 'com.squareup.okhttp3:okhttp:3.10.0'
```



# Add Network permission

Add in file AndroidManifest.xml

```
manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.somkiat.calculator">
```

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

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
```



# Create MathJsService.java

To mapping with math.js service

```
public interface MathJsService {  
    @GET("/v4/")  
    Call<String> calculate(@Query("expr") String expression);  
}
```



# Call service from Activity (1)

Create new method to call service

```
private void calculateAddFromRestApi(int first, int second) {  
    // Prepare value  
    String expression = String.format("%d+%d", first, second);  
  
    // Call service with retrofit  
    Retrofit retrofit = new Retrofit.Builder()  
        .baseUrl("http://api.mathjs.org")  
        .addConverterFactory(GsonConverterFactory.create())  
        .client(OkHttpProvider.getOkHttpClientInstance())  
        .build();  
    MathJsService mathJsService  
        = retrofit.create(MathJsService.class);  
}
```



# Call service from Activity (1)

Create new method to call service

```
private void calculateAddFromRestApi(int first, int second) {  
    // Prepare value  
    String expression = String.format("%d+%d", first, second);  
  
    // Call service with retrofit  
    Retrofit retrofit = new Retrofit.Builder()  
        .baseUrl("http://api.mathjs.org")  
        .addConverterFactory(GsonConverterFactory.create())  
        .client(OkHttpProvider.getOkHttpClientInstance())  
        .build();  
    MathJsService mathJsService  
        = retrofit.create(MathJsService.class);  
}
```





# Create class OkHttpClient

Singleton class to create client

```
public abstract class OkHttpClientProvider {  
    private static OkHttpClient instance = null;  
  
    public static OkHttpClient getOkHttpClientInstance() {  
        if (instance == null) {  
            instance = new OkHttpClient();  
        }  
        return instance;  
    }  
}
```





# Call service from Activity (2)

Get result from callback and send to result page

```
// Callback from service
final Call<String> call = mathJsService.calculate(expression);
call.enqueue(new Callback<String>() {
    @Override
    public void onResponse(Call<String> call, Response<String> response) {
        sendResult(response.body());
    }

    @Override
    public void onFailure(Call<String> call, Throwable t) {

    }
});
```



# Call service from Activity (3)

Create new method to send result

```
private void sendResult(String result) {  
    // Send data to Result Activity  
    Intent intent =  
        new Intent(packageContext: this,  
                   ResultActivity.class);  
    intent.putExtra(name: "result", result);  
    startActivity(intent);  
}
```



Try to run UI tests  
\$gradlew cAT



# Result of UI Testing

## Test Summary

**3**  
tests

**1**  
failures

**3.238s**  
duration

**66%**  
successful

Failed tests

Packages

Classes

**Class**

**Test**

MainActivityTest

plus 1 and 1 should result 2



Try to solve with  
`Thread.sleep(2000)`



# Add sleep in UI test

```
onView(withId(R.id.btn_calculate)).perform(click());
```

```
Thread.sleep( millis: 2000 );
```

```
onView(withId(R.id.txv_result))  
    .check(matches(withText("Result = 2")));
```



Try to run UI tests  
\$gradlew cAT





# Thread.sleep(2000) !!



Replace `Thread.sleep()`  
with `Espresso IdlingResource`

<https://developer.android.com/reference/android/support/test/espresso/IdlingResource>



# Dependencies

## Add OkHttp3 Idling Resource

```
// Testing  
androidTestImplementation 'com.jakewharton.espresso:okhttp3-idling-resource:1.0.0'
```

<https://github.com/JakeWharton/okhttp-idling-resource>



# Add IdlingResource in test

```
IdlingResource idlingResource  
    = OkHttp3IdlingResource.create(  
        name: "okhttp", OkHttpProvider.getOkHttpInstance());  
IdlingRegistry.getInstance().register(idlingResource);
```



Try to run UI tests  
`$gradlew cAT`



# How to testing REST APIs ?



# How to testing REST APIs ?

Success case

Failure case => 404, Malformed and Timeout





# Libraries and Tools

Using MockWebServer  
Using Stubby4j



# Using MockWebServer

<https://github.com/square/okhttp/tree/master/mockwebserver>



# Dependencies

## Add MockWebServer

```
androidTestImplementation 'com.squareup.okhttp3:mockwebserver:3.10.0'
```



# Create MockWebServer in test

Success case :  $1 + 1 = 2$

*// 1. Setup MockWebServer*

```
MockWebServer server = new MockWebServer();  
server.enqueue(new MockResponse().setBody("2"));  
server.start();  
String url = server.url(path: "/").toString();
```



# Change URL of REST API

First solution :: Use static variable

*// 1. Setup MockWebServer*

```
MockWebServer server = new MockWebServer();  
server.enqueue(new MockResponse().setBody("3"));  
server.start();  
String url = server.url(path: "/").toString();
```

*// 2. Replace URL of REST API with MockWebServer*  
`MainActivity.HTTP_API_MATHJS_ORG = url;`

*// 3. Shutdown server*  
`server.shutdown();`



# Create MockWebServer in test

Failure case : 404

@Test

```
public void failure_404() throws Exception {
```

```
    // 1. Setup MockWebServer
```

```
    MockWebServer server = new MockWebServer();
```

```
    server.enqueue(new MockResponse().setResponseCode(404));
```



# Create MockWebServer in test

## Failure case : timeout

```
@Test
public void failure_timeout() throws Exception {

    // 1. Setup MockWebServer
    MockWebServer server = new MockWebServer();
    server.enqueue(
        new MockResponse()
            .setBody("2")
            .throttleBody(bytesPerPeriod: 1, period: 1, TimeUnit.SECONDS));
}
```



Try to run UI tests  
\$gradlew cAT





# Static variable !!



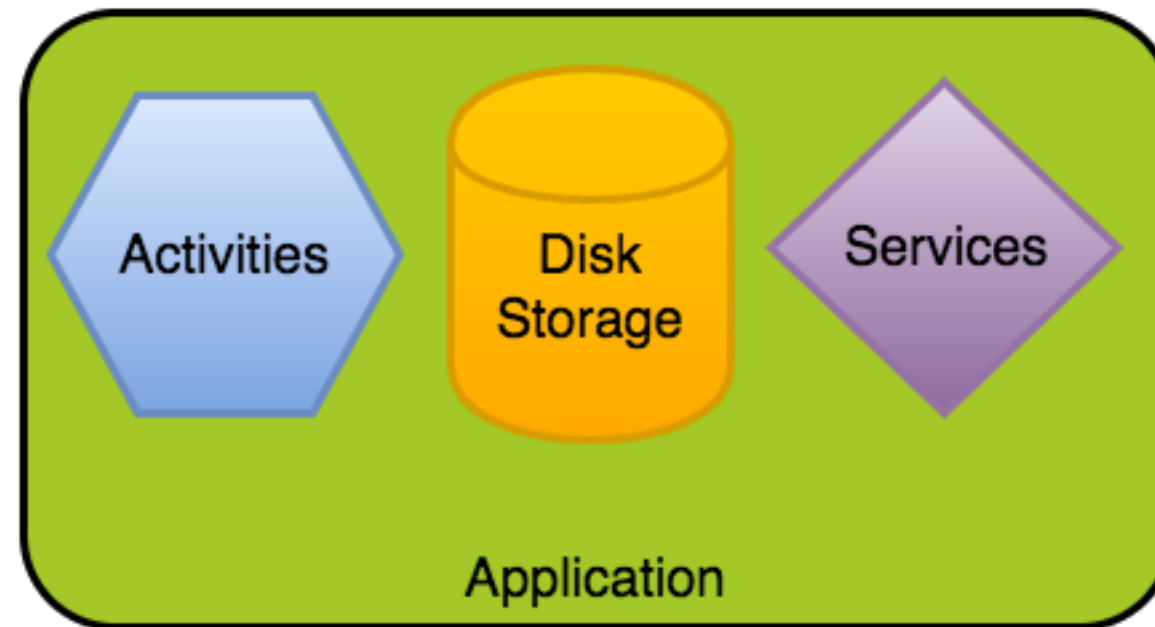
# Replace **static variable** with **Application class**

<https://developer.android.com/reference/android/app/Application>



# Application class

Maintain global application state



# Custom Application class in app

Create DemoApplication.java to setup REST APIs

```
public class DemoApplication extends Application {  
    public String getBaseUrl() {  
        return "http://api.mathjs.org";  
    }  
}
```



# Register application class

In file AndroidManifest.xml

```
<application  
    android:name=".DemoApplication"  
    android:allowBackup="true"  
    android:icon="@mipmap/ic_launcher"
```



# Use in Activity class

```
// Create application instance  
DemoApplication demoApplication  
    = (DemoApplication)application();
```

```
// Call service with retrofit  
Retrofit retrofit = new Retrofit.Builder()  
    .baseUrl(demoApplication.getBaseUrl())  
    .addConverterFactory(GsonConverterFactory.create())  
    .client(OkHttpProvider.getOkHttpInstance())  
    .build();
```



# Use in Activity class

```
// Create application instance
```

```
DemoApplication demoApplication  
    = (DemoApplication)application();
```

```
// Call service with retrofit
```

```
Retrofit retrofit = new Retrofit.Builder()  
    .baseUrl(demoApplication.getBaseUrl())  
    .addConverterFactory(GsonConverterFactory.create())  
    .client(OkHttpProvider.getOkHttpClient())  
    .build();
```



# Try to run app





# How to test with Application ?



# How to test with application ?

1. Create TestDemoApplication.java in androidTest
2. Create CustomTestRunner
3. Change testInstrumentationRunner in build.gradle
4. Use in UI testing



# 1. TestDemoApplication.java

Extend from DemoApplication in app

```
public class TestDemoApplication extends DemoApplication {  
    private String baseUrl;  
  
    @Override  
    public String getBaseUrl() {  
        return baseUrl;  
    }  
  
    public void setBaseUrl(String url) {  
        baseUrl = url;  
    }  
}
```



# 2. CustomTestRunner.java

Extend from AndroidJUnitRunner

```
public class CustomTestRunner extends AndroidJUnitRunner {  
    @Override  
    public Application newApplication(  
        ClassLoader cl, String className, Context context)  
        throws IllegalAccessException, ClassNotFoundException, InstantiationException {  
        return super.newApplication(cl, TestDemoApplication.class.getName(), context);  
    }  
}
```



# 3. Change in file build.gradle

## Value of testInstrumentationRunner

```
versionCode 1  
versionName "1.0"  
testInstrumentationRunner "com.example.somkiat.calculator.CustomTestRunner"
```



# 4. Use in test case

*// 1. Setup MockWebServer*

```
MockWebServer server = new MockWebServer();  
server.enqueue(new MockResponse().setBody("2"));  
server.start();  
String url = server.url(path: "/").toString();
```

*// 2. Replace URL of REST API with MockWebServer*

```
TestDemoApplication app = (TestDemoApplication)  
    InstrumentationRegistry.getTargetContext().getApplicationContext();  
app.setBaseUrl(server.url(path: "/").toString());
```



Try to run UI tests  
\$gradlew cAT

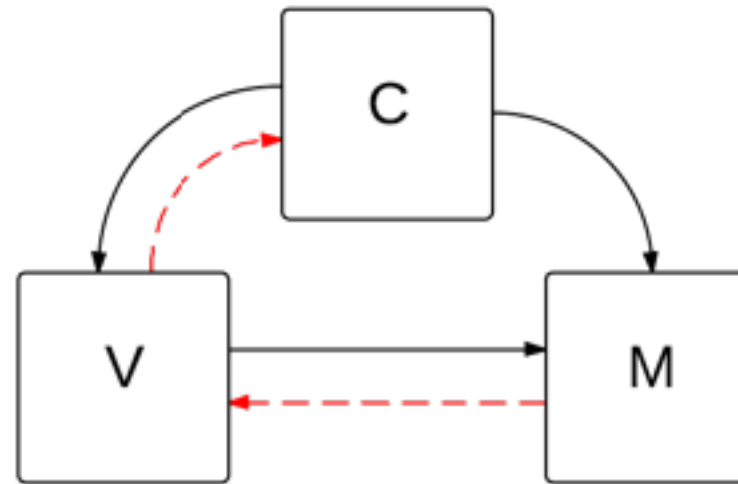


# Testable application ?

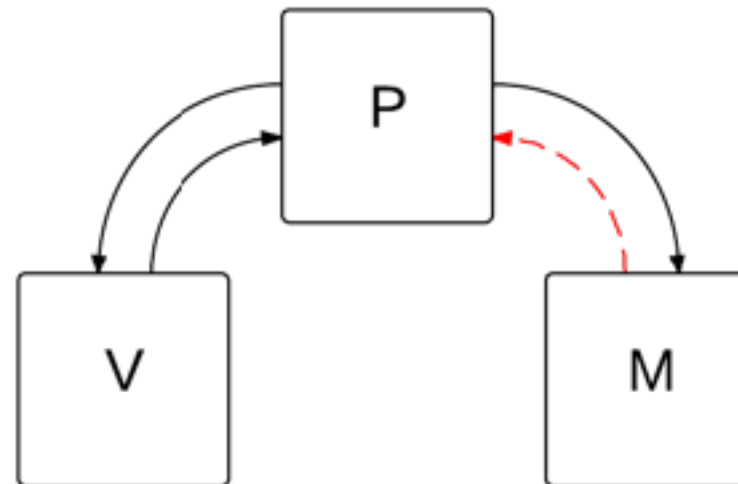




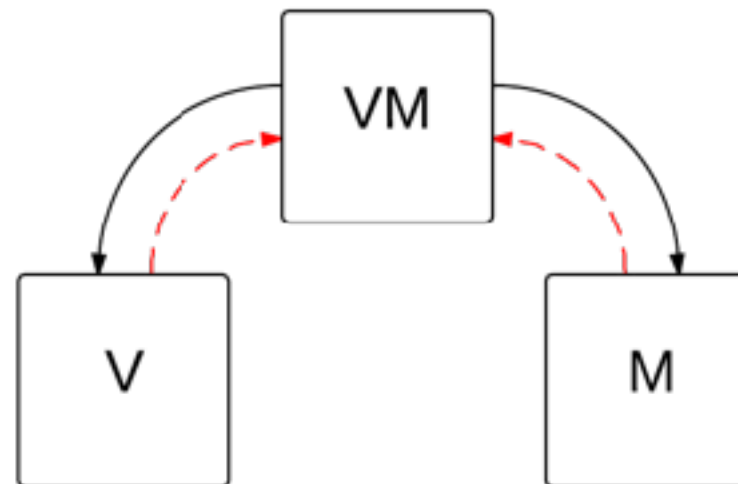
MVC



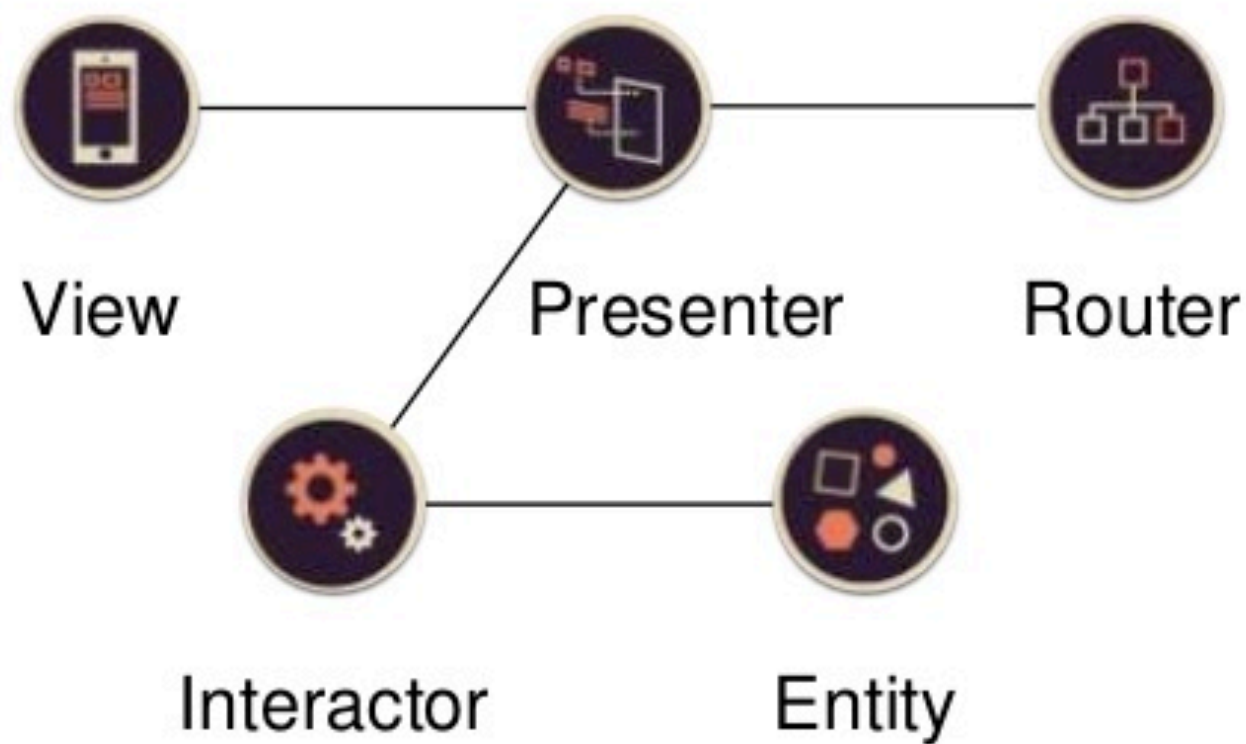
MVP

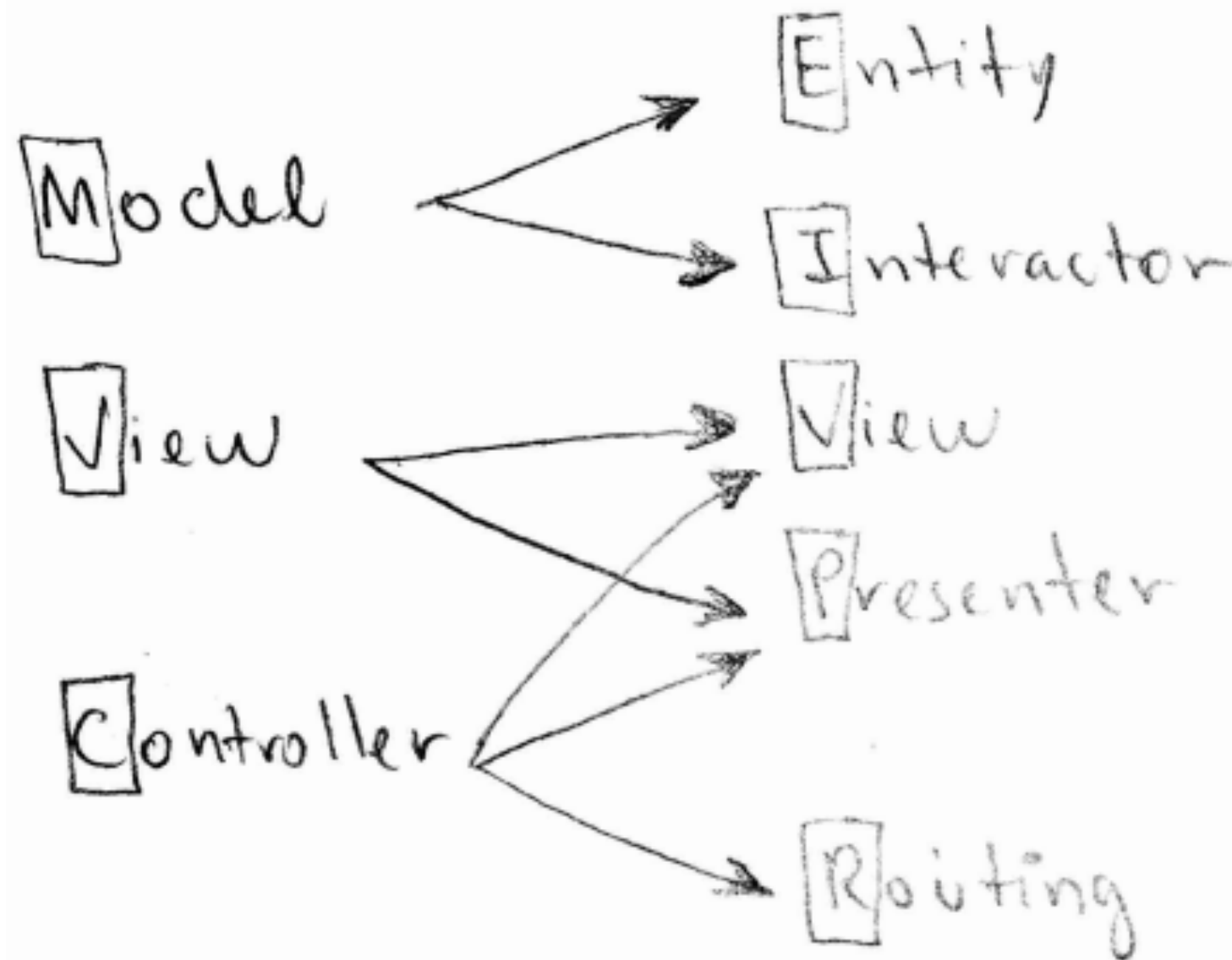


MVVM



# I ♥ VIPER





# The Clean Architecture

