

Android Testing

Gemini

<https://github.com/geminiwen>

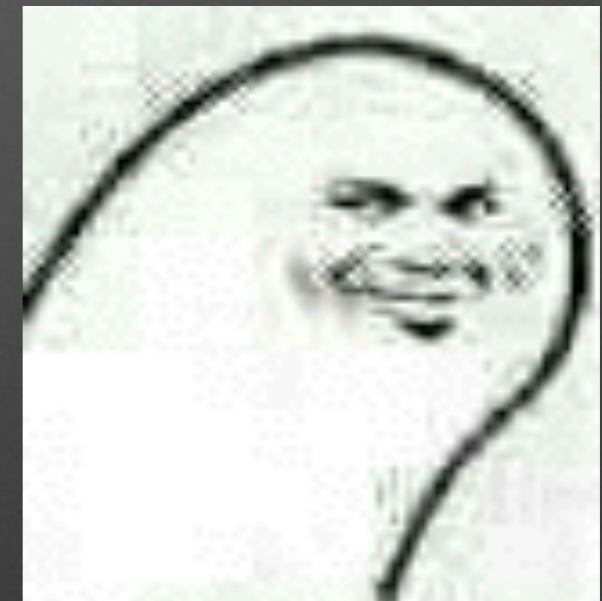


SFDC

SegmentFault
Developer Conference

About Me

- Gemini (已婚)
- SegmentFault for Android 作者
- SegmentFault 移动端负责人
- Android 开发者, 但是是果粉
- 最近在研究engine.io



程序员的使命是让人失业



SFDC

SegmentFault
Developer Conference

概览

- 测试的意义
- 测试的颗粒度
- Mockito
- Dagger
- CI的应用



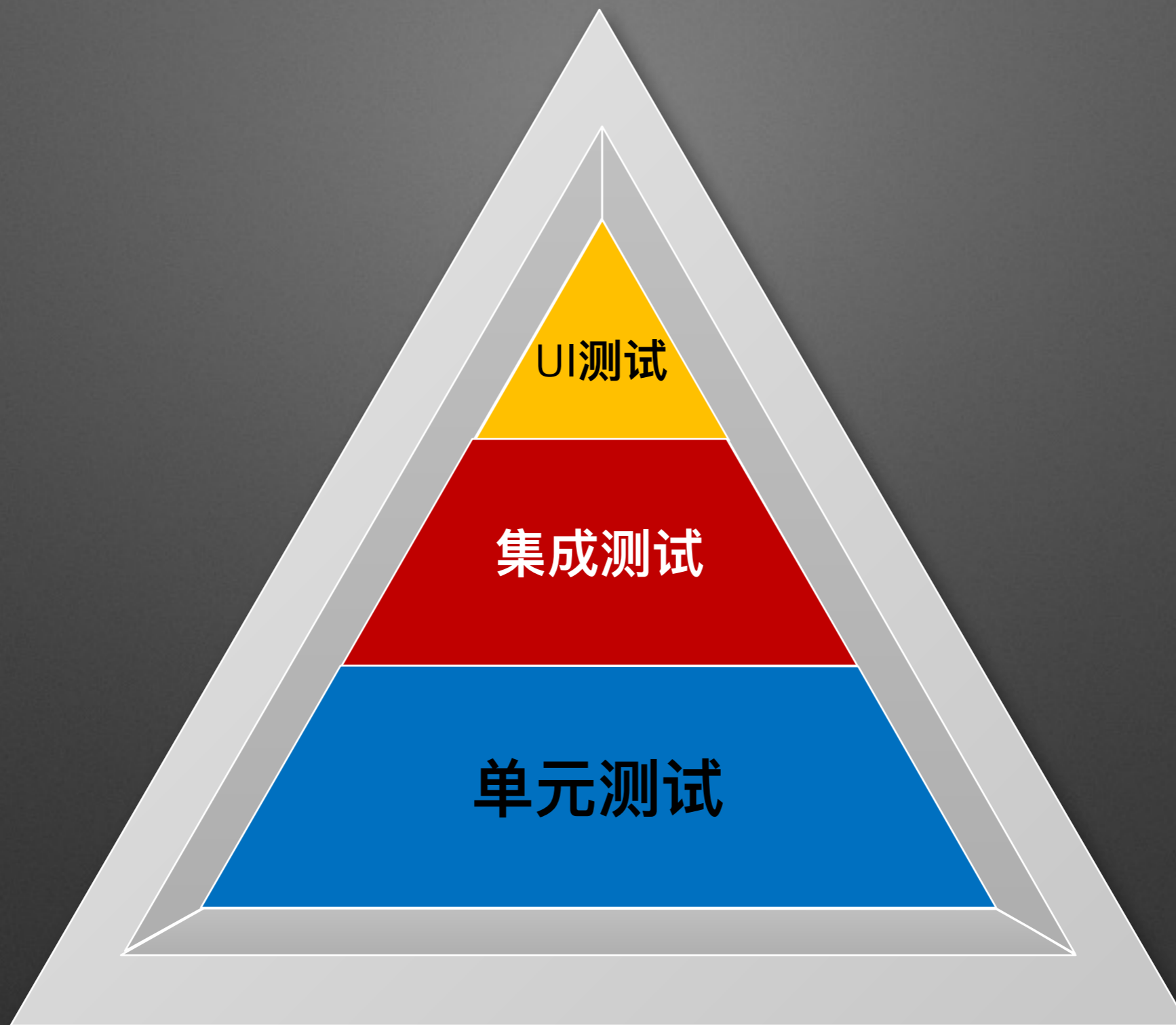
当我们讲测试的时候
我们在讲什么....



测试的颗粒度

- 单元测试 (Unit Test)
- 集成测试
- UI 测试





单元测试

- 是针对程序模块（软件设计的最小单位）来进行正确性检验的测试工作
- 检查输入与输出
- 是一切测试的基础
- 尽可能覆盖，但是不要期望覆盖率100%



```
int add (int a, int b) {  
    return a + b;  
}
```

```
assertEquals(add(1,2), 3 );
```



集成测试

- 集成测试（也叫组装测试，联合测试）是单元测试的逻辑扩展。它最简单的形式是：把两个已经测试过的单元组合成一个组件，测试它们之间的接口。



目标是一组方法或者业务逻辑

```
void job1();  
void job2();  
int job3();
```

```
assertEquals(job3_result, job3());
```



UI测试

- 检查设计的还原度
- 检查组件之间的交互
- 检查业务流程
- 基于测试用例，做更完备的集成测试



Android Testing Framework

Espresso



@Test

```
public void greeterSaysHello() {  
    onView(withId(R.id.name_field))  
        .perform(typeText("Steve"));
```

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

```
    onView(withText("Hello Steve!"))  
        .check(matches(isDisplayed()));
```

```
}
```



UI测试思考的问题

- UI测试依据什么来写？
- UI测试的代码该写点什么？
- 需要注意的一些技巧是什么？



回归测试



回归测试用例

B	C	D	E	F	G
优先级	模块	前置条件	用例	条件	期望结果
100	登录	无	输入正确的邮箱和密码		登录成功
100	登录		输入错误的邮箱或错误的密码		登录失败，提示用户名或密码错误
100	登录		不输入邮箱，输入密码		登录失败，提示邮箱没有输入
100	登录		不输入邮箱，不输入密码		登录失败，提示邮箱没有输入
100	登录		输入邮箱，不输入密码		登录失败，提示密码没有输入
100	注册		输入不存在的用户名，不存在的邮箱，和密码		注册成功
100	注册		输入已存在的用户名，不存在的邮箱，和密码		注册失败，提示用户名已存在
			输入已存在的用户名，已存在的邮箱，和密码		注册失败，提示用户名已存在
100	注册		输入不存在的用户名，已存在的邮箱，和密码		注册失败，提示邮箱已存在
100	注册		某一项不输入		布局优先级从上到下，前端提示出没有输入的那一项
100	用户详情	登录状态	点击修改用户信息按钮，进入修改个人信息页面	网络正常	应该能获取用户线上资料，展示在UI上



@Test

```
public void testLoginSuccess() throws Exception {  
    onView(withId(R.id.et_mail))  
        .perform(clearText(), replaceText("geminiwen@aliyun.com"));  
    onView(withId(R.id.et_password))  
        .perform(clearText(), replaceText("123456"));  
    onView(withId(R.id.btn_login))  
        .perform(click());  
    Thread.sleep(200);  
    onView(withId(R.id.et_mail))  
        .check(doesNotExist());  
}
```

@Test

```
public void testLoginFailed() throws Exception {  
    onView(withId(R.id.et_mail))  
        .perform(clearText(), replaceText("geminiwen@aliyun.com"));  
    onView(withId(R.id.et_password))  
        .perform(clearText(), replaceText("999999"));  
    onView(withId(R.id.btn_login))  
        .perform(click());  
    Thread.sleep(200);  
  
    onView(withText("用户名或密码错误"))  
        .inRoot(toast())  
        .check(matches(isDisplayed()));  
}
```



```

@Test
public void testInputEmpty() throws Exception {
    onView(withId(R.id.et_mail))
        .perform(clearText());
    onView(withId(R.id.btn_login))
        .perform(click());
    onView(withText("邮箱不能为空"))
        .check(matches(isDisplayed()));

    Thread.sleep(1000);

    onView(withId(R.id.et_mail))
        .perform(clearText(), replaceText("geminiwen@aliyun.com"));
    onView(withId(R.id.et_password))
        .perform(clearText());
    onView(withId(R.id.btn_login))
        .perform(click());
    onView(withText("密码不能为空"))
        .check(matches(isDisplayed()));
}

```



业务异步



SFDC

SegmentFault
Developer Conference

```
void business()
```



```
callback
```

IdleResource By Espresso



Idle Resource

```
@Override boolean isIdleNow();
```

```
ResourceCallback.onTransitionToIdle();
```

```
okhttp3-idleresource
```



Mock

绝对不可以在测试的时候和线上数据交互

- Mock Object
- Mock Http



Mockito

- 伪造model提供给UI渲染
- 判断伪造model的接口是否被调用



@Test

```
public void test1() {
```

```
    // 创建 mock
```

```
    MyClass test = Mockito.mock(MyClass.class);
```

```
    // 自定义 getUniqueId() 的返回值
```

```
    when(test.getUniqueId()).thenReturn(43);
```

```
    // 在测试中使用mock对象
```

```
    assertEquals(test.getUniqueId(), 43);
```

```
}
```



OkHttp MockWebServer

```
private static Dispatcher sServerDispatcher = new Dispatcher() {
    @Override
    public MockResponse dispatch(RecordedRequest request) throws InterruptedException {
        String url = request.getPath();
        MockResponse response = new MockResponse();
        if (url.equals("/user/login")) {
            Buffer buffer = request.getBody();
            String requestBody = buffer.readString(Charset.defaultCharset());
            if (requestBody.contains("password=123456")) {
                response.setBody("{\"status\":0,\"data\":{\"user\":{\"name\":\"Gemini\",\"mail\":\"\"}}");
                response.setResponseCode(200);
            } else {
                response.setBody("{\"status\":1,\"code\":\"01390902\",\"message\":\"\\u7528\\u623\"");
                response.setResponseCode(200);
            }
        } else {
            response.setResponseCode(500);
        }
        return response;
    }
};
```



如何注入Mock Object?

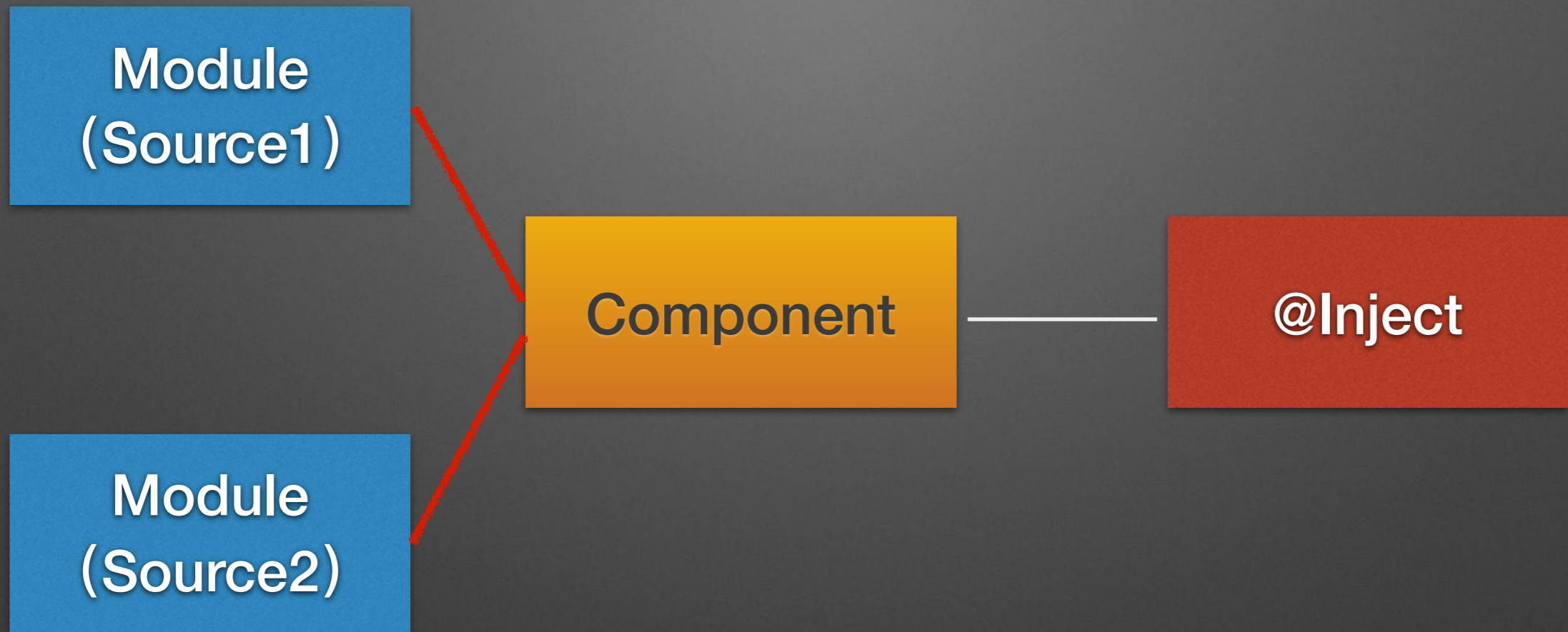
- Dagger2 —— 静态依赖注入 (DI) 框架



Dagger Components

- Module —— 注入源
- Component —— 注入器
- @Inject —— 注入点





```
public class App extends MultiDexApplication implements Application

    private final static String APP_NAME = "SegmentFault";
    private final static String MIPUSH_TAG = "SegmentFault_MiPush";

    @Inject AppComponent mAppComponent;
    @Inject UserStore mUserStore;
```

```
App app = (App)context.getApplicationContext();
AppComponent appComponent = DaggerAppComponent.builder()
    .appModule(new TestAppModule(context, url))
    .build();
appComponent.inject(app);
```



```
@Singleton
@Component(modules=AppModule.class)
public interface AppComponent {
    void inject(App app);
    void inject(Service service);
    void inject(SFWebViewClient service);
    void inject(SFX5WebViewClient service);
    void inject(PicassoWrapper wrapper);
    void inject(UserChangeReceiver object);
    void inject(ChatService object);

    //Preferences
    void inject(SFFeedbackPreference preference);
    void inject(SFLogoutPreference preference);
    void inject(SFEnvironmentPreferences preference);

    ActivityComponent.Builder activityComponent();
    PresenterComponent.Builder presenterComponent();
    DAOComponent.Builder daoComponent();
    OkHttpClient httpClient();
    UserStore userStore();
    Context context();
}
```



```

@Module(subcomponents = {
    ActivityComponent.class,
    PresenterComponent.class,
    DAOComponent.class
})
public class AppModule {
    Context mApplicationContext;

    public AppModule(Context context) {
        this.mApplicationContext = context;
    }

    @Provides
    public Context provideContext(){
        return mApplicationContext;
    }

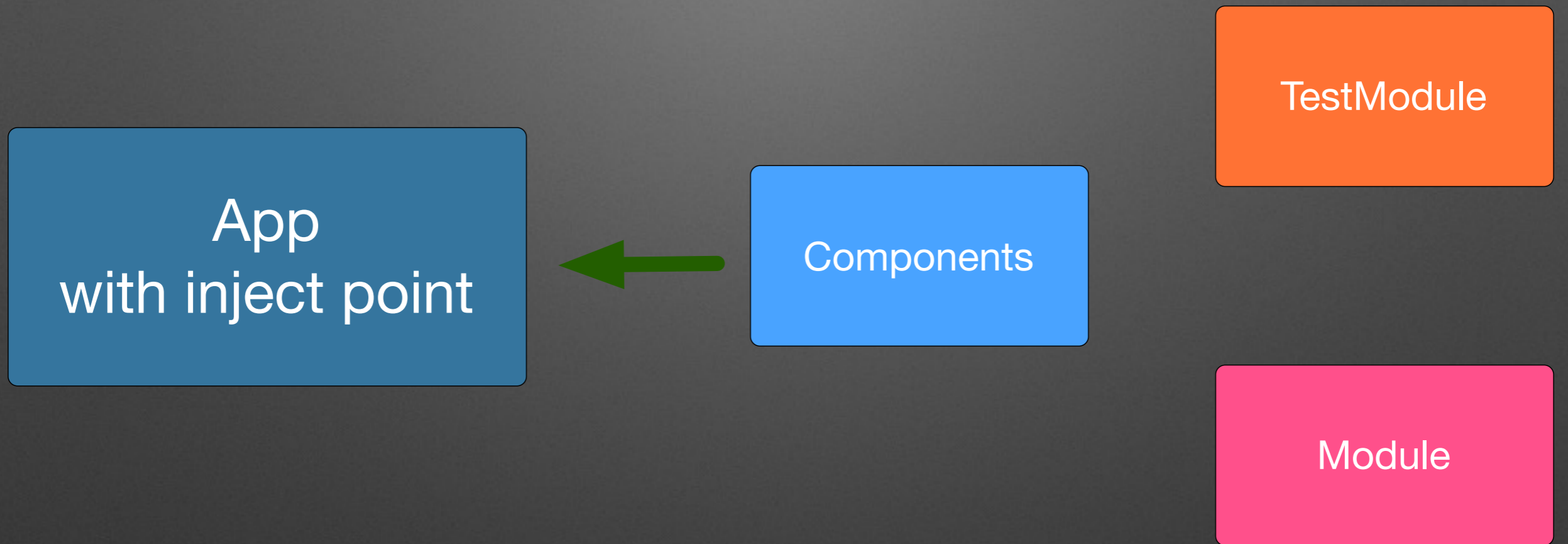
    @Singleton
    @Provides
    public OkHttpClient provideHttpClient(OkHttpClient instance) {
        return instance.getOkHttpClient();
    }

    @Singleton
    @Provides
    public Picasso providePicasso(PicassoWrapper picassoWrapper) {
        return picassoWrapper.getPicasso();
    }
}

```

```
public class TestAppModule extends AppModule {  
  
    private String mBaseUrl;  
  
    public TestAppModule(Context context, String baseUrl) {  
        super(context);  
        this.mBaseUrl = baseUrl;  
    }  
  
    @Override  
    public Service provideService(Context context, OkHttpClient okHttpClient) {  
        return new Service(context, okHttpClient, mBaseUrl);  
    }  
}
```





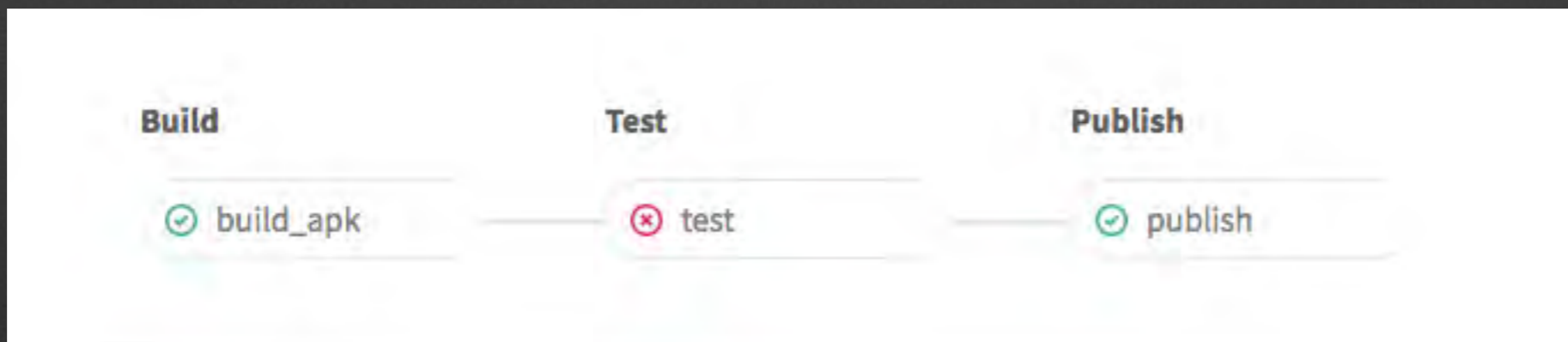
Dagger Management

- User Store
- Database Instance (SQLite)
- Retrofit Instance



Gitlab CI

Status	Commit	Archive	Build	Test	Notify	Publish
🔄 running	#586 develop <small>60200dec</small> latest 👤 upgrade okhttp	-	🔄	-	-	-
❌ failed	#585 develop <small>898f28e6</small> 👤 update following divider	-	❌	🛑	-	🛑
✅ passed	#581 develop <small>dc6d64e3</small> 👤 fix #85	-	✅	✅	-	✅



CI 博特

巫力饭 下午12:28
吃

郭耿瑞 上午11:36
SF_Ad_Server

周慧 上午11:30
[语音通话]

文件传输助手 上午9:37
你在电脑登录了企业微信, 在这里发送文件、...

CI 博特 昨天
Deploy success!

这里是RD的主场 昨天
蔡丹霞: [自定义表情]

大技术部 昨天
[自定义表情]

OMG 昨天
方舟: https://leancloud.cn/docs/ios_crash...

SF 大无线事业部 昨天
龚焱之: 嗯

Android Canary更新啦

重构第三方绑定页面
修复 issue 83
新增【账号和密码】页面

resolve issue#81
注册用户增加了海外用户的过滤
重构了激活页面

详情

星期五 下午5:55

Android Canary更新啦

重构第三方绑定页面
修复 issue 83
新增【账号和密码】页面

resolve issue#81
注册用户增加了海外用户的过滤
重构了激活页面

详情

昨天 下午5:32

内测下载



展望

- TDD —— 测试驱动开发
- 更精细化的生命周期管理
- 为组件化做好质量上的准备



Thanks!



阿温先森_Gemini

浙江 杭州

扫一扫上面的二维码图案，加我微信

This is a WeChat QR code for the user '阿温先森_Gemini' (Awen Xiansen_Gemini) from Hangzhou, Zhejiang. The QR code is large and centered on a white background. A small circular profile picture of the user is overlaid on the QR code. Below the QR code, there is a small text prompt: '扫一扫上面的二维码图案，加我微信' (Scan the QR code above to add my WeChat).



SFDC

SegmentFault
Developer Conference