安卓单元测试

你需要知道的一切

蘑菇街 小创

Disclaimer

- 这里讨论的是 单元测试
  - Run on JVM
  - No Espresso、UiAutomator、Robotium、Appium…

Agenda

- 为什么要做单元测试
- 什么是单元测试
- Mock
  - Mockito的使用
- 依赖注入
  - Dagger2介绍
- Robolectric

从为什么开始

- 验证你的代码真的能work
- 保证软件质量、减少bug
- 更快的反馈Loop
- 安全放心的重构
- 更好的代码设计
- 节约时间

没有时间?

- 感觉需要很多时间?
  - 单元测试是门技术活，学习需要时间
  - 调整原有的项目结构需要时间
- 写单元测试需要时间?
  - 一切都是套路!

TRY IT YOURSELF!

什么是单元测试

- 定义：针对一小块代码单元写的测试代码

```java
public class Calculator {

    public int add(int one, int another) {
        return one + another;
    }

}
```

```java
public class CalculatorTest {

    @Test
    public void testAdd() {
        Calculator calculator = new Calculator();

        int sum = calculator.add(1, 2);

        assertEquals(3, sum);
    }

}
```

单元测试三步走

- 一个测试用例 一般分为三步
  - Setup：前提条件设置、环境
  - Action：调用被测代码
  - Verify：验证得到的结果跟预期一样

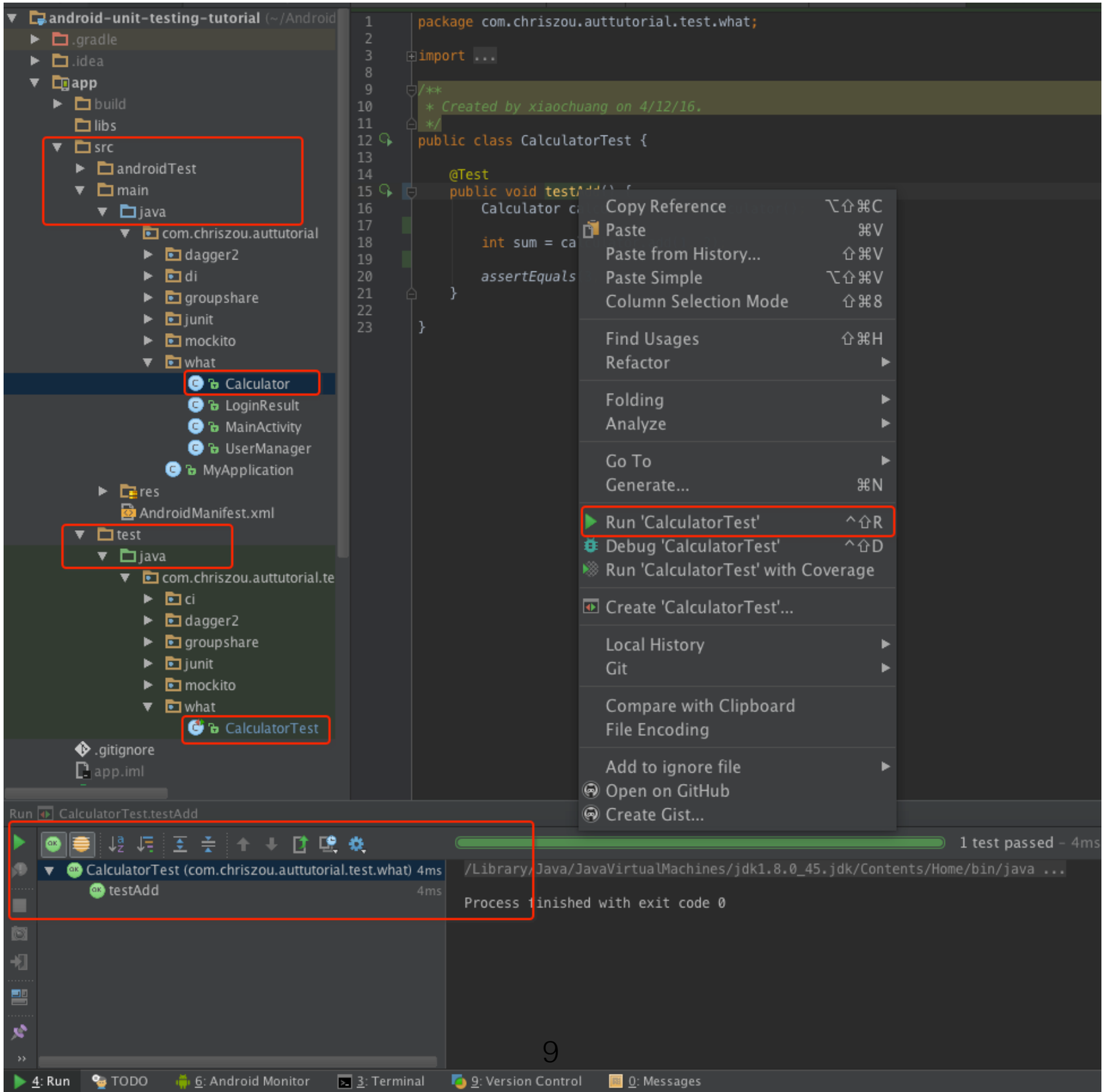
```
public class CalculatorTest {

    @Test
    public void testAdd() {
        Calculator calculator = new Calculator();

        int sum = calculator.add(1, 2);

        assertEquals(3, sum);
    }

}
```

Setup

Action

Verify

怎么运行


android-unit-testing-tutorial
.gradle
.idea
app
build
libs
src
androidTest
main
java
com.chriszou.auttutorial
dagger2
di
groupshare
junit
mockito
what
Calculator
LoginResult
MainActivity
UserManager
MyApplication
res
AndroidManifest.xml
test
java
com.chriszou.auttutorial.te
ci
dagger2
groupshare
junit
mockito
what
CalculatorTest
.gitignore
app.iml
package com.chriszou.auttutorial.test.what;
import ...
/**
* Created by xiaochuang on 4/12/16.
*/
public class CalculatorTest {
@Test
public void testAdd() {
Calculator ca
int sum = ca
assertEquals
}
}
Copy Reference
Paste
Paste from History...
Paste Simple
Column Selection Mode
Find Usages
Refactor
Folding
Analyze
Go To
Generate...
Run 'CalculatorTest'
Debug 'CalculatorTest'
Run 'CalculatorTest' with Coverage
Create 'CalculatorTest'...
Local History
Git
Compare with Clipboard
File Encoding
Add to ignore file
Open on GitHub
Create Gist...
Run CalculatorTest.testAdd
1 test passed - 4ms
CalculatorTest (com.chriszou.auttutorial.test.what) 4ms
testAdd 4ms
/Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java ...
Process finished with exit code 0
4: Run
TODO
6: Android Monitor
3: Terminal
9: Version Control
0: Messages

void方法怎么办

```java
public class MainActivity extends AppCompatActivity {

    private UserManager mUserManager = new UserManager();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        findViewById(R.id.login).setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                login();
            }
        });
    }

    public void login() {
        String username = ((EditText) findViewById(R.id.username)).getText().toString();
        String password = ((EditText) findViewById(R.id.password)).getText().toString();

        // Other code, like verify username and password

        mUserManager.performLogin(username, password);
    }

    @Subscribe
    public void onLoginResult(LoginResult event) {
        //update UI accordingly
    }

}
```

login()方法怎么测?

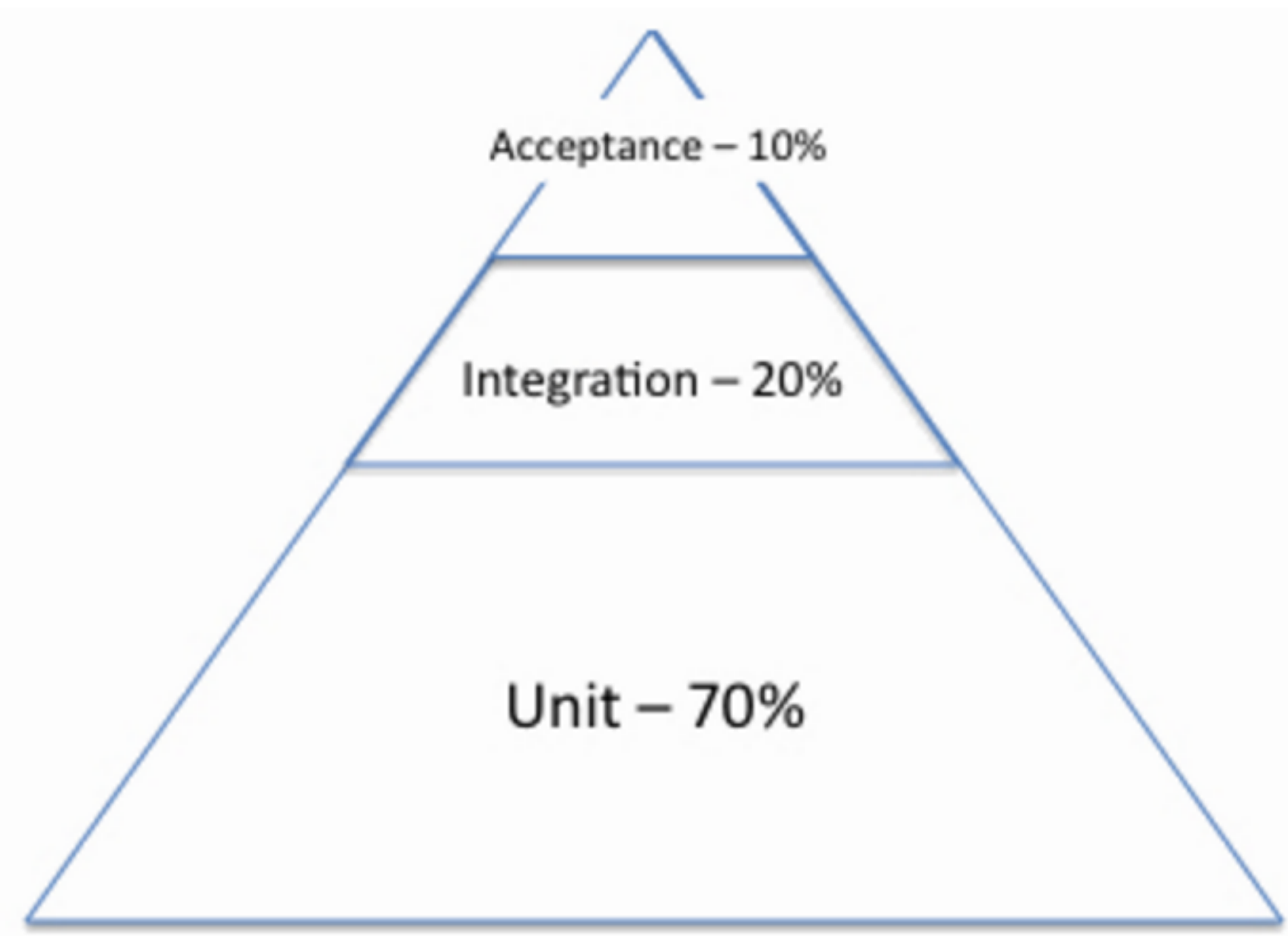
Void method

- 三步走

```java
@Test
public void testLogin() throws Exception {
    //1. setup
    MainActivity mainActivity = Robolectric.setupActivity(MainActivity.class);
    ((EditText) mainActivity.findViewById(R.id.username)).setText("xiaochuang");
    ((EditText) mainActivity.findViewById(R.id.password)).setText("xiaochuang is handsome");

    //2. action
    mainActivity.login();

    //3. verify result
    //? ? ?
}
```

不隔离的单元测试是集成测试

- Test Pyramid



怎么样才是单元测试?

```java
public class MainActivity extends AppCompatActivity {

    private UserManager mUserManager = new UserManager();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        findViewById(R.id.login).setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                login();
            }
        });
    }

    public void login() {
        String username = ((EditText) findViewById(R.id.username)).getText().toString();
        String password = ((EditText) findViewById(R.id.password)).getText().toString();

        // Other code, like verify username and password

        mUserManager.performLogin(username, password);
    }

    @Subscribe
    public void onLoginResult(LoginResult event) {
        //update UI accordingly
    }

}
```

单元测试的隔离性

- 不依赖于外部条件
  - 网络、数据库。。。
- 不依赖于其他类的正确与否
- 粒度

怎么验证
一个对象的方法
得到了调用?

```
public void login() {
    String username = ((EditText) findViewById(R.id.username)).getText().toString();
    String password = ((EditText) findViewById(R.id.password)).getText().toString();

    // Other code, like verify username and password

    mUserManager.performLogin(username, password);
}
```

Mock

- 定义：“在测试环境下，根据一个类创建出来的虚假的或模拟的对象”
- 作用：
  - 改变这个对象的某个方法的行为
    - 执行特定动作
    - 返回特定值
  - 记录方法调用情况，以便后续可以验证

怎么使用mock

- Mock framework
  - Mockito
    - 稳定
    - 语法直观
    - 兼容性好
  - JMockit：
    - 功能更强大（能mock static方法）
    - 不够稳定
    - 兼容性问题

Mockito

- 创建mock
  - UserManager mockUserManager = Mockito.mock(UserManager.class);
- 验证方法调用
  - Mockito.verify(mockUserManager).performLogin("xiaochuang", "xiaochuang is handsome");

Mockito in Action

```java
@Test
public void testLogin() {
    //1. setup
    MainActivity mainActivity = Robolectric.setupActivity(MainActivity.class);
    ((EditText) mainActivity.findViewById(R.id.username)).setText("xiaochuang");
    ((EditText) mainActivity.findViewById(R.id.password)).setText("xiaochuang is handsome");
    UserManager mockUserManager = Mockito.mock(UserManager.class);

    //2. action
    mainActivity.login();

    //3. verify result
    Mockito.verify(mockUserManager).performLogin("xiaochuang", "xiaochuang is handsome");
}
```

```
Wanted b
userMana
    "xia
    "xia
);
-> at co
Actually

Wanted b
userMana
    "xia
    "xia
);
-> at co
Actually

    at c
    at c
    at c
    at c
    at c
```

```java
public class MainActivity extends AppCompatActivity {

    private UserManager mUserManager = new UserManager();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        findViewById(R.id.login).setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                login();
            }
        });
    }

    public void login() {
        String username = ((EditText) findViewById(R.id.username)).getText().toString();
        String password = ((EditText) findViewById(R.id.password)).getText().toString();

        // Other code, like verify username and password

        mUserManager.performLogin(username, password);
    }
```

```
ernal calls>
s>
calls>
```

Mocks需要被set进去

```
public class MainActivity extends AppCompatActivity {
    private UserManager mUserManager = new UserManager();
    //other methods

    public void setUserManager(UserManager userManager) {
        mUserManager = userManager;
    }
}
```

```
@Test
public void testLogin() {
    //1. setup
    MainActivity mainActivity = Robolectric.setupActivity(MainActivity.class);
    ((EditText) mainActivity.findViewById(R.id.username)).setText("xiaochuang");
    ((EditText) mainActivity.findViewById(R.id.password)).setText("xiaochuang is handsome");
    UserManager mockUserManager = Mockito.mock(UserManager.class);
    mainActivity.setUserManager(mockUserManager);

    //2. action
    mainActivity.login();

    //3. verify result
    Mockito.verify(mockUserManager).performLogin("xiaochuang", "xiaochuang is handsome");
}
```

Mockito用法（一）

- 验证方法调用次数
  - verify(mockObject).methodExpected(arguments)

```
Mockito.verify(mockUserManager).performLogin("xiaochuang", "xiaochuang is handsome”);
//<=>
Mockito.verify(mockUserManager, Mockito.times(1)).performLogin("xiaochuang", "xiaochuang is handsome”);

Mockito.times(3);
Mockito.never();
```

- 验证方法参数

```
Mockito.verify(mockUserManager).performLogin("xiaochuang", "xiaochuang is handsome”);
Mockito.verify(mockUserManager).performLogin(Mockito.anyString(), Mockito.anyString());  //Matchers

anyInt()
anyLong()
…
anyObject()
anyCollection()
any(Class)
anyCollection()
anyCollectionOf()
…
```

Mockito用法（二）

- 指定方法行为
  - 指定方法返回值

    Mockito.when(someMock.someMethod(someArgus)).thenReturn(someValue)

```
public void login() {
    String username = ((EditText) findViewById(R.id.username)).getText().toString();
    String password = ((EditText) findViewById(R.id.password)).getText().toString();

    mPasswordValidator.verifyPassword(password);

    mUserManager.performLogin(username, password);
}
```

```
PasswordValidator passwordValidator = Mockito.mock(PasswordValidator.class);
Mockito.when(passwordValidator.verifyPassword("xiaochuang is handsome”)).thenReturn(true);
```

Mockito用法（二）

- 指定方法行为
  - 指定方法返回值
    - 如果不指定，返回默认值
      - int, long, float, double… => 0
      - boolean => false
      - Object => null

Mockito用法（二）

- 指定方法行为
  - 指定方法只执行特定动作
  - 抛出异常： Mockito.doThrow(someException).when...
  - 执行真实的实现： Mockito.doCallRealMethod().when...
  - More specific：

```
Mockito.doAnswer(new Answer() {
    @Override
    public Object answer(InvocationOnMock invocation) throws Throwable {
        Object[] arguments = invocation.getArguments();
    }
}).when…
```

Mockito用法（二）

- doAnswer示例

```
public void login() {
    String username = ((EditText) findViewById(R.id.username)).getText().toString();
    String password = ((EditText) findViewById(R.id.password)).getText().toString();

    mPasswordValidator.verifyPassword("hello");

    mUserManager.performLogin(username, password, new NetworkCallback() {
        @Override
        public void onSuccess(Object data) {
            updateResule();
        }

        @Override
        public void onFailure(int code, String msg) {
            showErrorMessage();
        }
    });
}
```

Mockito用法（二）

- doAnswer示例

```
@Test
public void testLoginCallbackVersion() {
    //1. setup
    MainActivity mainActivity = Robolectric.setupActivity(MainActivity.class);
    ((EditText) mainActivity.findViewById(R.id.username)).setText("xiaochuang");
    ((EditText) mainActivity.findViewById(R.id.password)).setText("xiaochuang is handsome");

    UserManager mockUserManager = Mockito.mock(UserManager.class);
    Mockito.doAnswer(new Answer() {
        @Override
        public Object answer(InvocationOnMock invocation) throws Throwable {
            NetworkCallback callback = (NetworkCallback) invocation.getArguments()[2];
            callback.onSuccess(someMockResult);

            return null;
        }
    }).when(mockUserManager.performLogin(anyString(), anyString(), any(NetworkCallback.class)));

    mainActivity.setUserManager(mockUserManager);

    //2. action
    mainActivity.login();

    //3. verify result
    //Verify result updated or error message shown
}
```

Spys

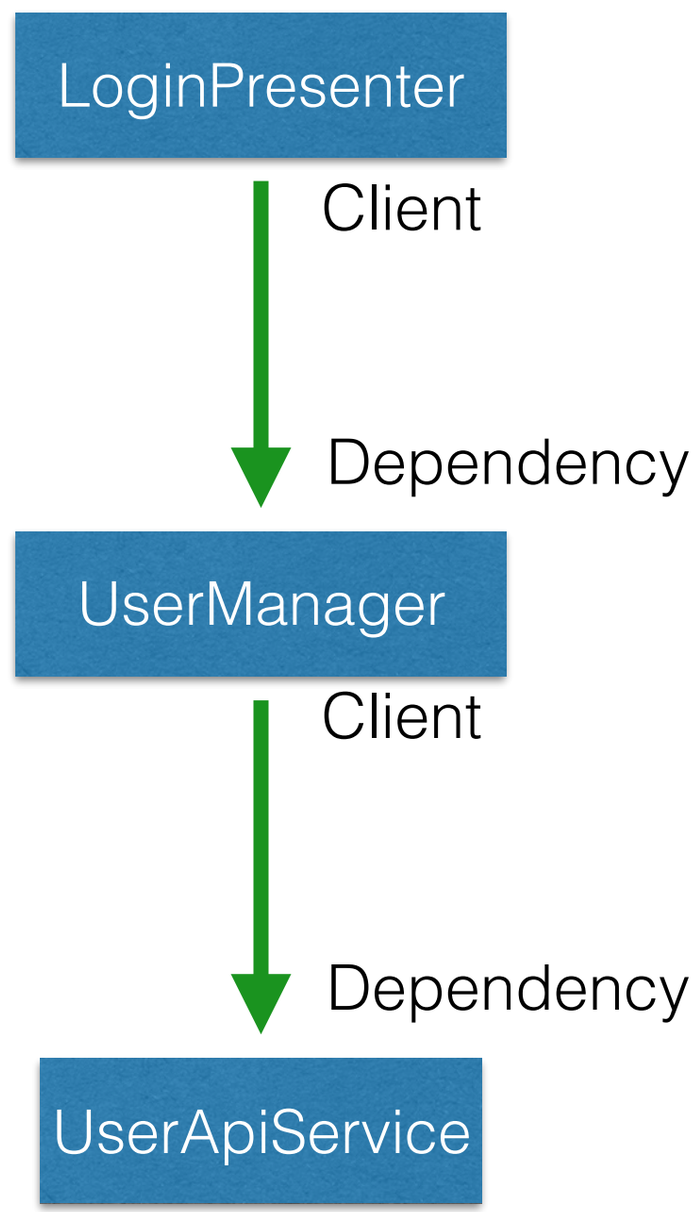
- 默认情况下执行真实的实现的mock
  - Mockito.spy(PasswordValidator.class) //需要有默认构造方法
  - Mockito.spy(new PasswordValidator(somePattern)) //基于一个对象
- 可以记录、验证方法调用情况
- 可以指定方法行为

Setters

- 不是很优雅
- 多了很多“测试专用”的接口，混淆视听

依赖注入

- 一种代码模式
- Dependency与Client



依赖注入

- “一个类（Client）的Dependency不在这个类（Client）内部创建，而是在外面创建好，set 进去”
- 实现形式
  - setter
  - 方法参数 login(UserManager userManager)
  - 构造方法参数

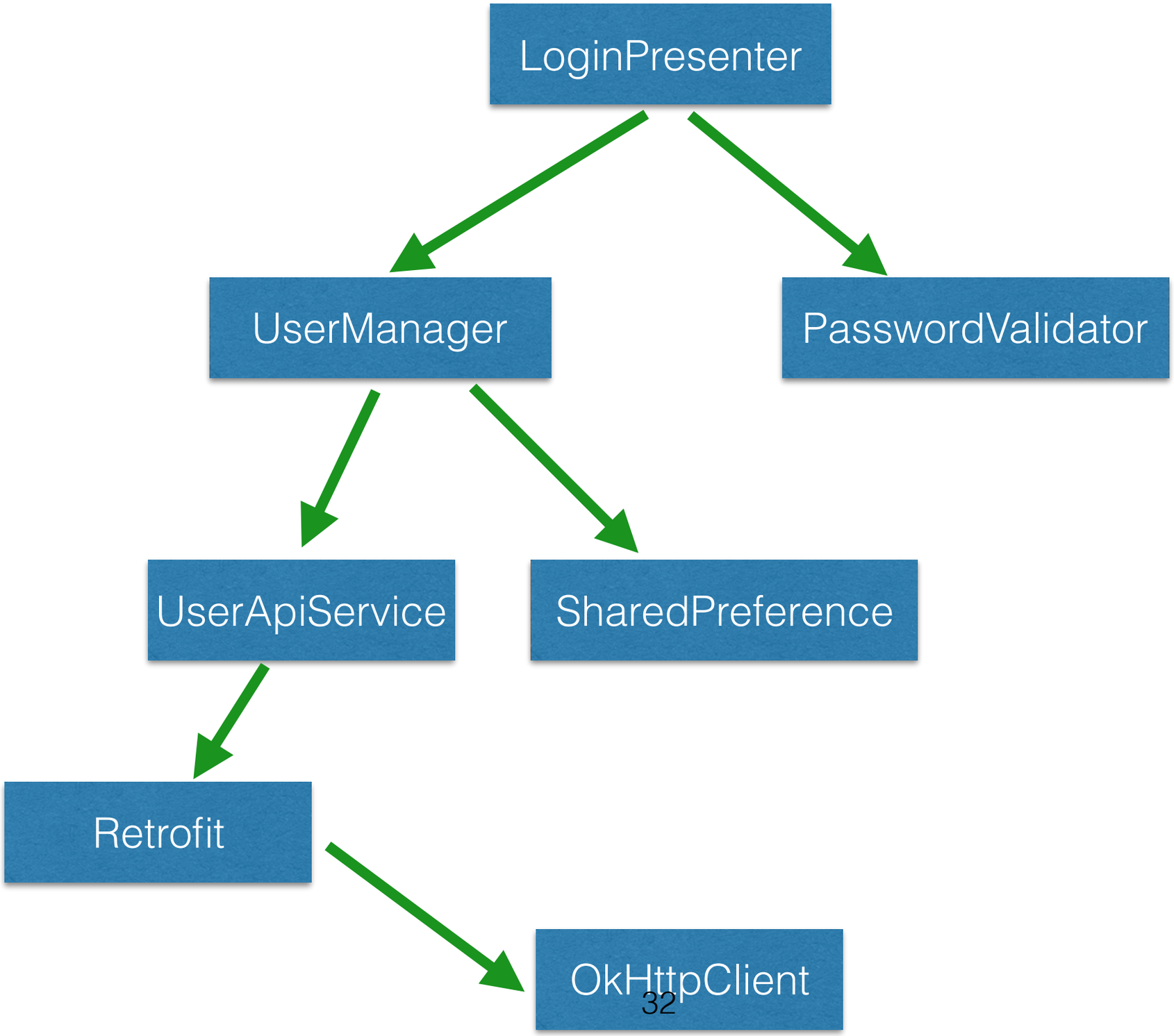
依赖注入

```
public class LoginPresenter {
    private UserManager mUserManager = new UserManager();
}
```

应用依赖注入

```
public class LoginPresenter {
    private UserManager mUserManager;

    public LoginPresenter(UserManager userManager) {
        this.mUserManager = userManager;
    }
}
```

依赖注入的窘境


LoginPresenter
UserManager
PasswordValidator
UserApiService
SharedPreference
Retrofit
OkHttpClient

使用依赖注入

```
OkHttpClient okhttpClient = new OkHttpClient.Builder()
        .connectTimeout(30, TimeUnit.SECONDS)
        .build();
Retrofit retrofit = new Retrofit.Builder()
        .client(okhttpClient)
        .baseUrl("https://api.github.com")
        .build();
UserApiService userApiService = retrofit.create(UserApiService.class);
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);

UserManager userManager = new UserManager(preferences, userApiService);

PasswordValidator passwordValidator = new PasswordValidator();

mLoginPresenter = new LoginPresenter(userManager, passwordValidator);
```

怎么办?

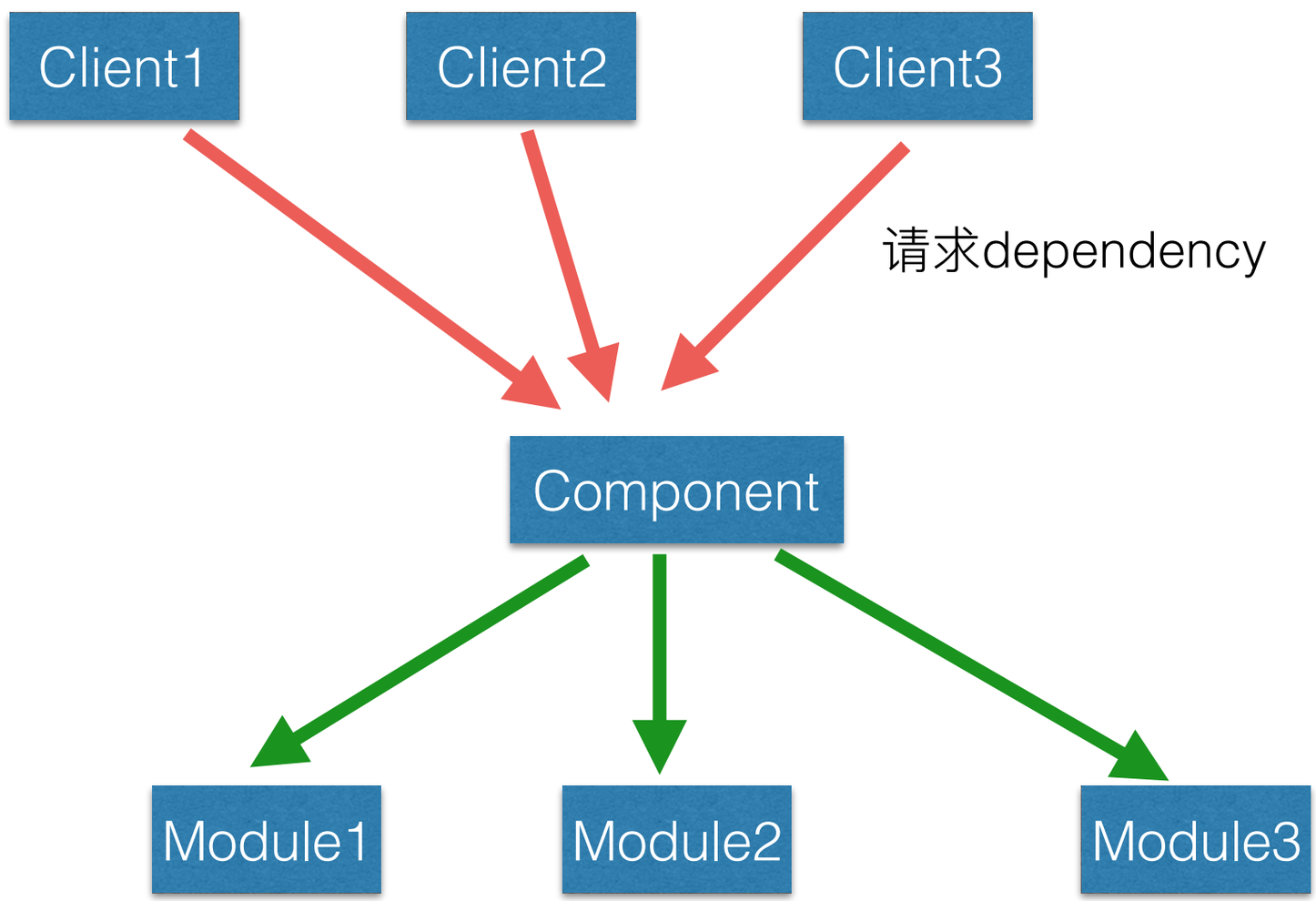
- 我们需要一个System
  - 在一个地方创建dependency
  - 自动分析dependency关系，创建Dependency对象
- 有这样的system吗?

Dagger2

Dagger2

- 一个专门做依赖注入的框架
- Dagger2's way
  - 生产Dependency的地方：Module
  - 提供给client的统一接口：Component

Dagger2


Client1
Client2
Client3
请求dependency
Component
Module1
Module2
Module3

Dagger2 in code

- Module

```
@Module
public class AppModule {
    @Provides
    public UserManager provideUserManager(SharedPreferences preferences, UserApiService service) {
        return new UserManager(preferences, service);
    }

    @Provides
    public PasswordValidator providePasswordValidator() {
        return new PasswordValidator();
    }

    @Provides
    public LoginPresenter provideLoginPresenter(UserManager userManager, PasswordValidator validator) {
        return new LoginPresenter(userManager, validator);
    }

    // 创建其他Dependency的方法

}
```

Dagger2 in Code

- Component

```
@Component(modules = {AppModule.class})
public interface AppComponent {

    LoginPresenter loginPresenter();

    void inject(LoginActivity mainActivity);
}
```

Dagger2 in Code

- Client 请求Dependency

```java
public class LoginActivity extends Activity {
    @Inject
    LoginPresenter mLoginPresenter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        AppComponent appComponent = DaggerAppComponent.builder().appModule(new AppModule()).build();

        //方法1
        LoginPresenter loginPresenter = appComponent.loginPresenter();

        //方法2, 结合@Inject field
        appComponent.inject(this);
    }
}
```

```java
@Component(modules = {AppModule.class})
public interface AppComponent {

    LoginPresenter loginPresenter();

    void inject(LoginActivity mainActivity);
}
```

- DaggerAppComponent？？？

Dagger2 in Code

- DaggerAppComponent？？？
- compile 期间自动生产的 — Annotation Processing
- 实现了AppComponent接口

```
//大概是这个样子
public class DaggerAppComponent implements AppComponent {
    private final AppModule appModule;

    public DaggerAppComponent(AppModule appModule) {
        this.appModule = appModule;
    }

    @Override
    public LoginPresenter loginPresenter() {
        //从appModule里面找到LoginPresenter的依赖,以及依赖的依赖，然后创建好一个对象
        return new LoginPresenter(…);
    }

    @Override
    public void inject(LoginActivity mainActivity) {
        //找到mainActivity里面所有的@Inject feild，然后从appModule里面找到相应的对象，挨个给它们赋值
    }
}
```

```
@Component(modules = {AppModule.class})
public interface AppComponent {

    LoginPresenter loginPresenter();

    void inject(LoginActivity mainActivity);
}
```

单元测试中的应用

- 怎么样Dependency换成mock的
- Dependency是在Module里面创建的
- Component只是帮运工

```
AppComponent appComponent = DaggerAppComponent.builder().appModule(new AppModule()).build();
mLoginPresenter = appComponent.loginPresenter();
```

- 用一个生产mock的AppModule去创建DaggerAppComponent
  - 继承AppModule，重写Provider方法
  - mock AppModule

最后一个问题

- 安卓相关的类在JVM上面不能跑
  - throws RuntimeException(“Stub”)
- 解决方案
  - 使用Android提供的Instrumentation系统，
    - 需要真机或模拟器才能运行
    - 很慢很慢
  - 采用一定架构，将安卓相关的类隔离开
    - MVP、MVVM。。。
  - Robolectric

Robolectric

- 一个在JVM上面跑安卓单元测试的framework
- Make android classes available on JVM
- 实现了安卓类、增强了安卓类
  - Shadows
  - Custom Shadows
- Code Examples
  - setupActivity(YourActivity.class);
  - SupportFragmentTestUtil.startFragment(sampleFragment);
  - ShadowToast.getTextOfLatestToast();
  - shadowOf(activity).getNextStartedActivity()

Robolectric

- 比Instrumentation快很多，比JUnit慢很多
  - Instrumentation testing：几十秒，取决于app的大小
  - Robolectric：10秒左右
  - JUnit：几秒钟之内
- 尽量使用JUnit
- 粒度！粒度！！粒度！！！
  - You can, doesn’t mean you should!

一个真实案例


收银台
建设银行02:00-04:55维护 请使用其他支付方式 谢谢
余额
可用金额5.83元
理财
可用金额9.59元
浦发银行
**5878
支付宝支付
微信支付
更多支付方式
支付: ￥0.01
剩余 00:18:42 请尽快付款
付款

基本实现: CheckoutActivity

```java
public class CheckoutActivity extends Activity {

    @Inject
    CheckoutModel mCheckoutModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.checkout_activity);

        ComponentHolder.getAppComponent().inject(this);

        String paymentId = "some paymentId"; //这个是从外面传进来的
        mCheckoutModel.loadCheckoutData(paymentId);
    }

    @Subscribe
    public void onDataLoadedEvent(DataLoadedEvent event) {
        if (event.successful()) {
            //Update UI
        } else {
            //show error message
        }
    }
}
```

基本实现: CheckoutModel

```java
public class CheckoutModel {

    private final PaymentApi mApi;
    private final Bus mBus;

    public CheckoutModel(PaymentApi api, Bus bus) {
        this.mApi = api;
        this.mBus = bus;
    }

    public void loadCheckoutData(String paymentId) {
        //Other code, like composing params

        String someUrl = "some url";
        Map<String, String> someParams = new HashMap<>();
        mApi.get(someUrl, someParams, new NetworkCallback() {
            @Override
            public void onSuccess(Object data) {
                mBus.post(new DataLoadedEvent(data));
            }

            @Override
            public void onFailure(int code, String msg) {
                mBus.post(new DataLoadedEvent(code, msg));
            }
        });
    }
}
```

CheckoutActivity UT(1)

```java
public class CheckoutActivity extends Activity {
    @Inject
    CheckoutModel mCheckoutModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.checkout_activity);

        ComponentHolder.getAppComponent().inject(this);

        mCheckoutModel.loadCheckoutData(somePaymentId);
    }
    //other methods
}


public class CheckoutActivityTest {

    @Test  //should call CheckoutModel.loadCheckoutData when activity starts
    public void testActivityStarts() {
        //...setup dagger to return a mockModel
        Robolectric.setupActivity(CheckoutActivity.class);
        Mockito.verify(mockModel).loadCheckoutData(somePaymentId);
    }
}
```

CheckoutActivity UT(2)

```java
public class CheckoutActivity extends Activity {
    //onCreate() method

    @Subscribe
    public void onDataLoadedEvent(DataLoadedEvent event) {
        if (event.successful()) {
            //Update UI
        } else {
            //show error message
        }
    }
}
```

```java
public class CheckoutActivityTest {

    @Test  //should show data correctly when receive normal data
    public void testOnDataLoadedEvent_success() throws Exception {
        CheckoutActivity activity = Robolectric.setupActivity(CheckoutActivity.class);
        DataLoadedEvent event = new DataLoadedEvent(new CheckoutData());

        activity.onDataLoadedEvent(event);

        //Verify view updated correctly
    }
}
```

CheckoutActivity UT(3)

```java
public class CheckoutActivity extends Activity {
    //onCreate() method

    @Subscribe
    public void onDataLoadedEvent(DataLoadedEvent event) {
        if (event.successful()) {
            //Update UI
        } else {
            //show error message
        }
    }
}
```

```java
public class CheckoutActivityTest {

    @Test  //should show error msg when request fails
    public void testOnDataLoadedEvent_failure() throws Exception {
        DataLoadedEvent event = new DataLoadedEvent(500, "Server error");
        mActivity.onDataLoadedEvent(event);

        //Verify error message shown
    }
}
```

CheckoutModel UT (1)

```java
public class CheckoutModel {
    private final PaymentApi mApi;
    private final Bus mBus;

    public CheckoutModel(PaymentApi api, Bus bus) {
        mApi = api; mBus = bus;
    }

    public void loadCheckoutData(String paymentId) {
        //Other code, like composing params

        String someUrl = "some url";
        Map<String, String> someParams = new HashMap<>();
        mApi.get(someUrl, someParams, new NetworkCallback() {
            //callback…
        });
    }
}
```

```java
public class CheckoutModelTest {
    @Test
    @JSpec(desc = "should loadCheckoutData call Api.get")
    public void testLoadCheckoutData() {
        //… create mockApi, mockBus
        CheckoutModel model = new CheckoutModel(mockApi, mockBus);
        model.loadCheckoutData("some payment id”);

        //paymentApi is a mock
        Mockito.verify(paymentApi).get(args);
    }
}
```

CheckoutModel UT (2)

```java
public class CheckoutModel {
    public void loadCheckoutData(String paymentId) {
        //… Other code, like composing params
        mApi.get(someUrl, someParams, new NetworkCallback() {
            @Override
            public void onSuccess(Object data) {
                mBus.post(new DataLoadedEvent(data));
            }
            @Override
            public void onFailure(int code, String msg) {
                mBus.post(new DataLoadedEvent(code, msg));
            }
        });
    }
}
```

```java
public class CheckoutModelTest {
    @Test
    @JSpec(desc = "should loadCheckoutData call Bus.post with succesful result")
    public void testLoadCheckoutData2() {
        // mock mApi to call callback's onSuccess when its get method is called
        Mockito.doAnswer(new Answer() {
            @Override
            public Object answer(InvocationOnMock invocation) throws Throwable {
                NetworkCallback networkCallback = (NetworkCallback) invocation.getArguments()[2];
                networkCallback.onSuccess("success");
                return "Sucess";
            }
        }).when(paymentApi).get(eq("some url"), any(HashMap.class), any(NetworkCallback.class));

        model.loadCheckoutData("some payment id");

        verify(bus).post(someSuccessfulResult);  //bus is a mock
    }
}
```

CheckoutModel UT (3)

```java
public class CheckoutModel {
    public void loadCheckoutData(String paymentId) {
        //… Other code, like composing params
        mApi.get(someUrl, someParams, new NetworkCallback() {
            @Override
            public void onSuccess(Object data) {
                mBus.post(new DataLoadedEvent(data));
            }
            @Override
            public void onFailure(int code, String msg) {
                mBus.post(new DataLoadedEvent(code, msg));
            }
        });
    }
}
```

```java
public class CheckoutModelTest {
    @Test
    @JSpec(desc = "should loadCheckoutData call Bus.post with failure result")
    public void testLoadCheckoutData3() {
        // mock mApi to call callback's onFailure when its get method is called
        Mockito.doAnswer(new Answer() {
            @Override
            public Object answer(InvocationOnMock invocation) throws Throwable {
                NetworkCallback networkCallback = (NetworkCallback) invocation.getArguments()[2];
                networkCallback.onFailure(500, "Server error");
                return "Server error";
            }
        }).when(paymentApi).get(eq("some url"), any(HashMap.class), any(NetworkCallback.class));

        model.loadCheckoutData("some payment id");

        verify(bus).post(someFailureResult);  //bus is a mock
    }
}
```

Where To Go From Here

- Practice, practice, practice!
- Write testable code
  - DI、SRP、DRY
- More JUnit, less Robolectric
  - MVP…
- TDD

Q & A