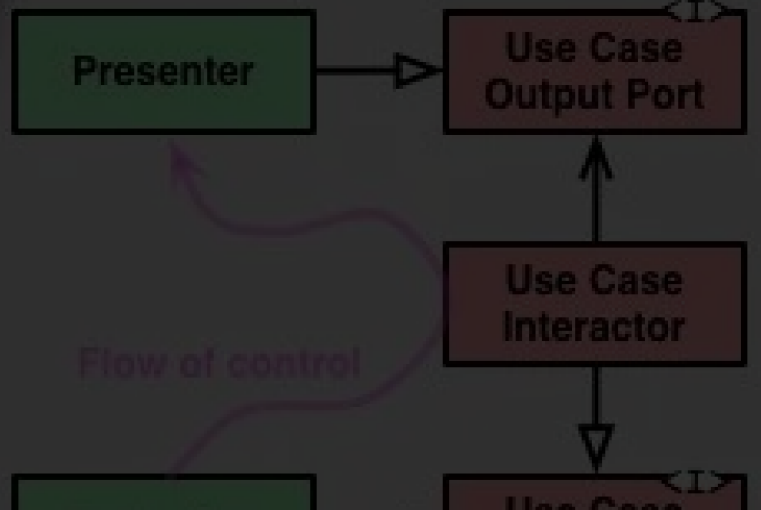
# 整洁架构

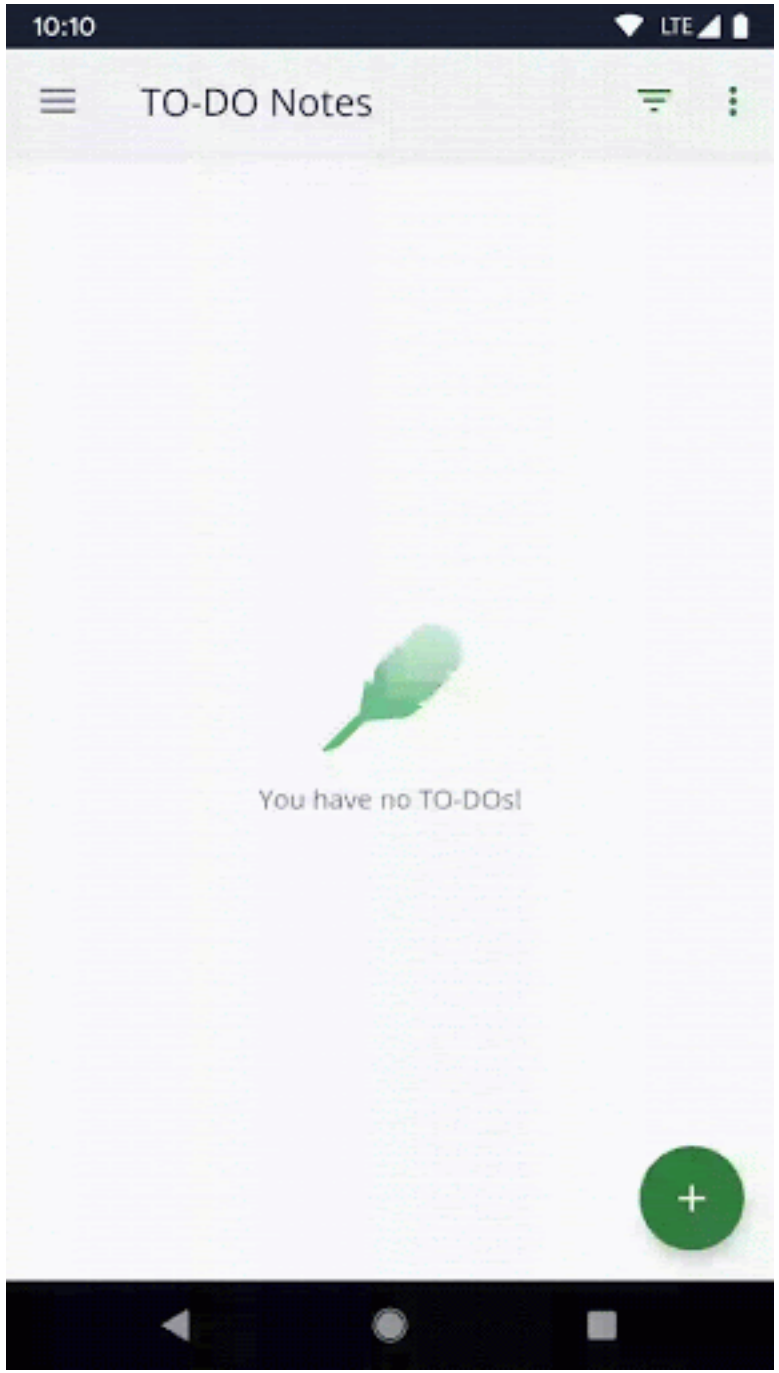
CAC@OPPO by 黄俊彬 & 覃宇



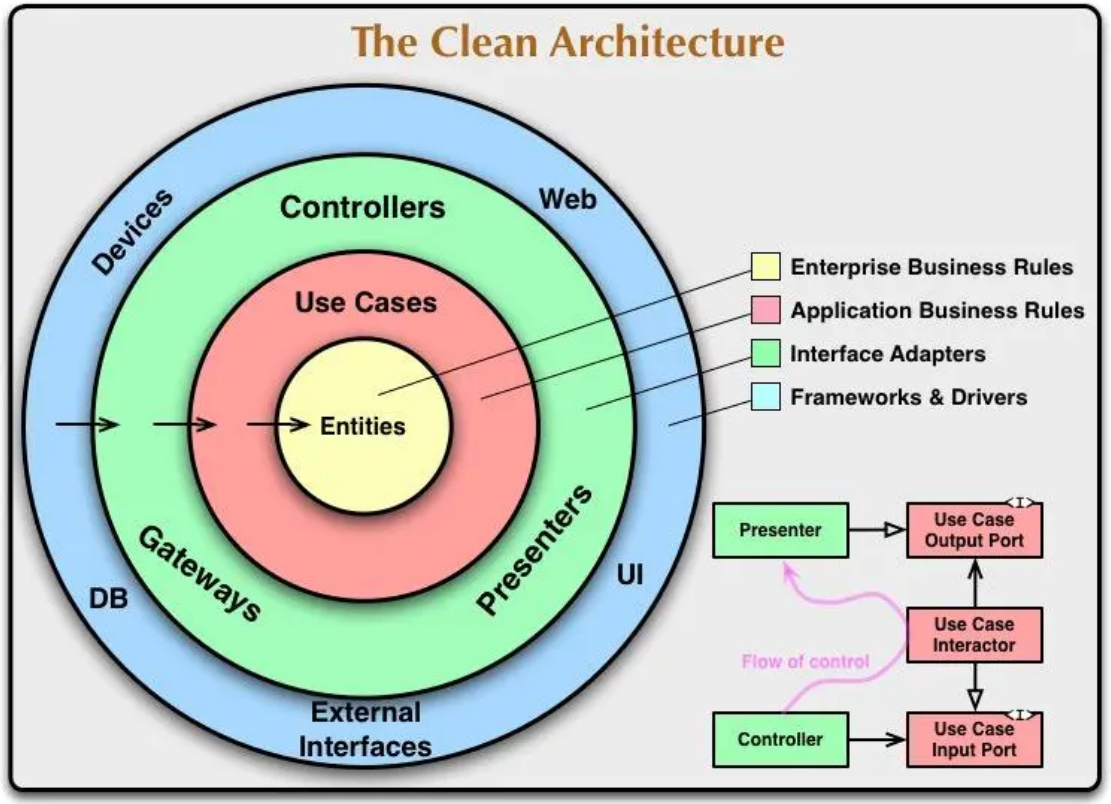


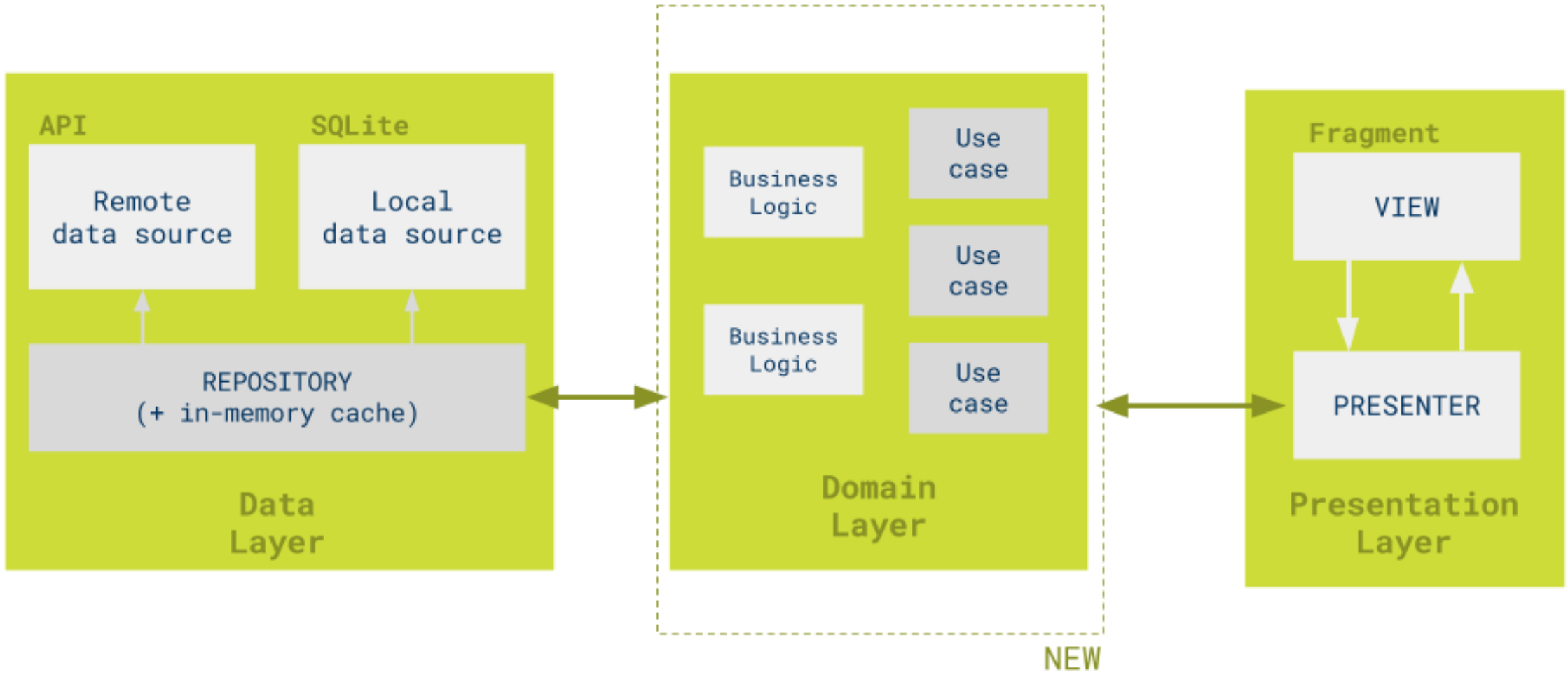
# Google-Clean-MVP

# TODO



# MVP的M、V、P，对应整洁架构那个Layer？

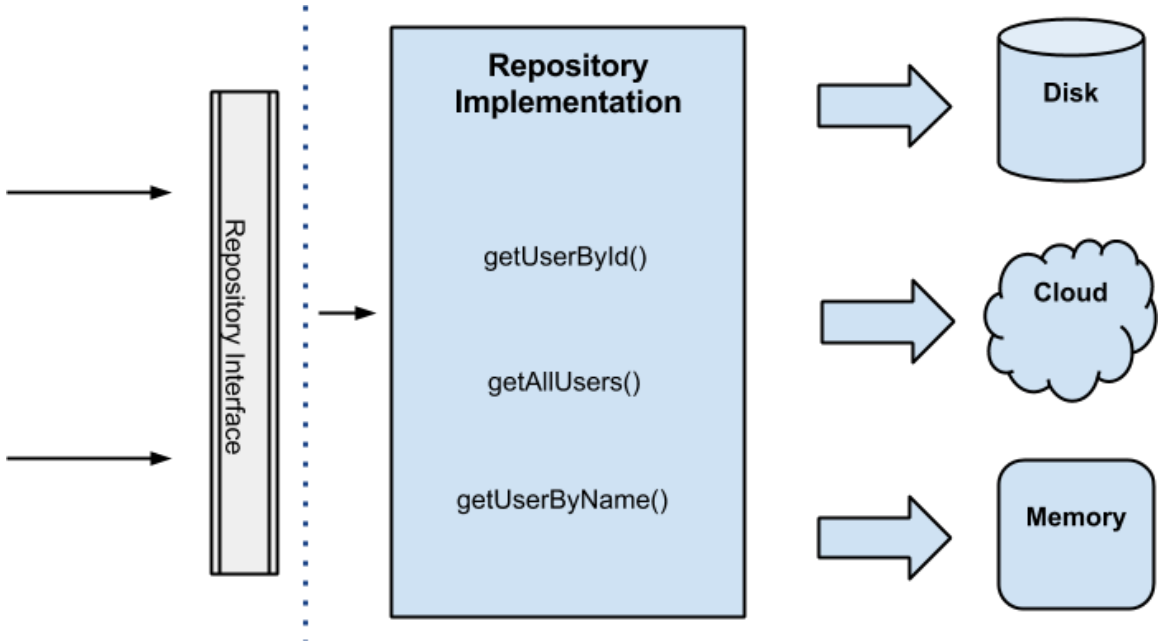



API
SQLite
Remote data source
Local data source
REPOSITORY (+ in-memory cache)
Data Layer
Business Logic
Business Logic
Use case
Use case
Use case
Domain Layer
NEW
Fragment
VIEW
PRESENTER
Presentation Layer

# Data Layer（数据层）

仓储模式（Repository Pattern）是存在于业务和数据库之间单独分离出来的一层，是对数据访问的封装。

- 业务层无需知道具体实现达到分离关注点
- 提高对数据访问的维护，对于仓储的改变并不改变业务的逻辑

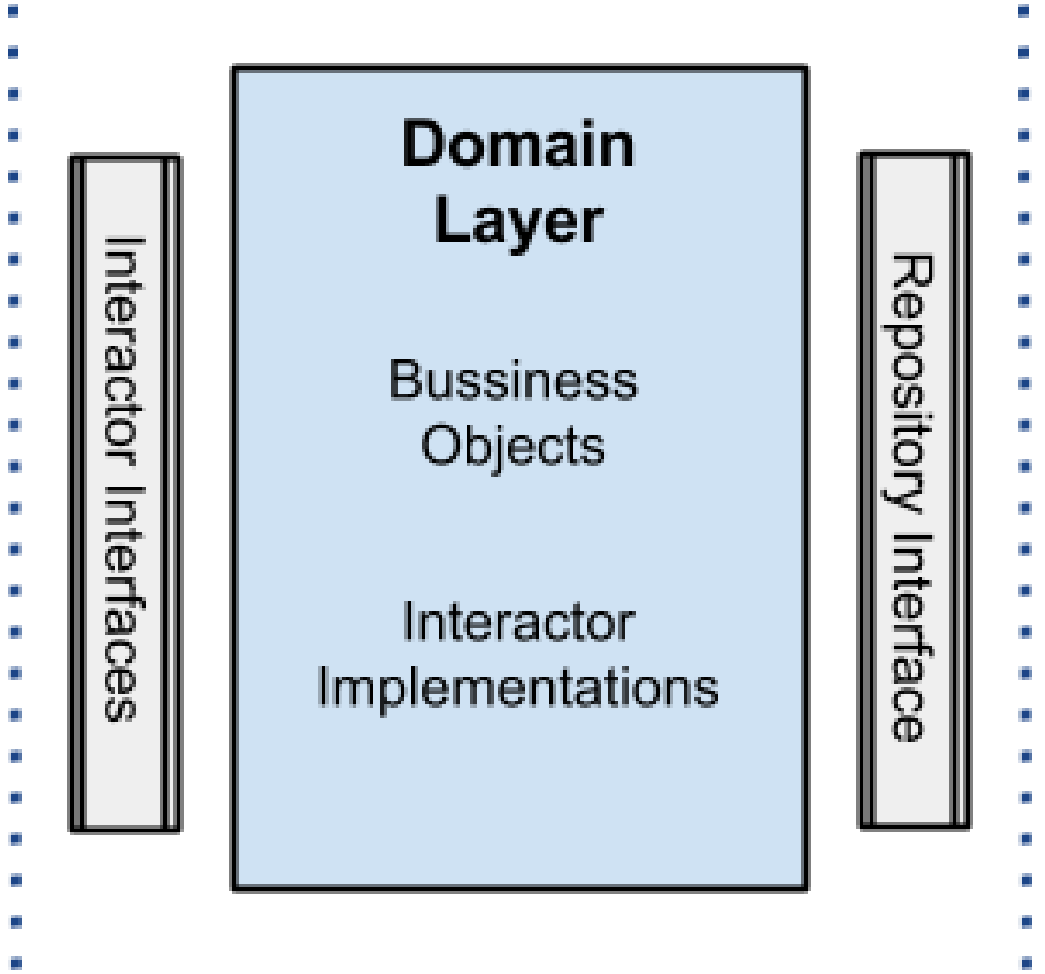


# 接口

```
 public interface TasksDataSource {

    void getTask(@NonNull String taskId, @NonNull GetTaskCallback callback);

    void saveTask(@NonNull Task task);

    void completeTask(@NonNull Task task);

    void deleteTask(@NonNull String taskId);
}
```

# Domain Layer (领域层)

- 包含所有的业务逻辑
- Use case 定义应用程序需要的操作
- 该层是一个纯Java模块，没有任何Android依赖项



# Task Domain

```
public final class Task {
    private final String mId;
    private final String mTitle;
    private final String mDescription;
    private final boolean mCompleted;
    // ... ...
}
```

```
public class CompleteTask extends UseCase<CompleteTask.RequestValues, CompleteTask.ResponseValue> {

    private final TasksRepository mTasksRepository;

    @Override
    protected void executeUseCase(final RequestValues values) {
        String completedTask = values.getCompletedTask();
        mTasksRepository.completeTask(completedTask);
        getUseCaseCallback().onSuccess(new ResponseValue());
    }
}
```

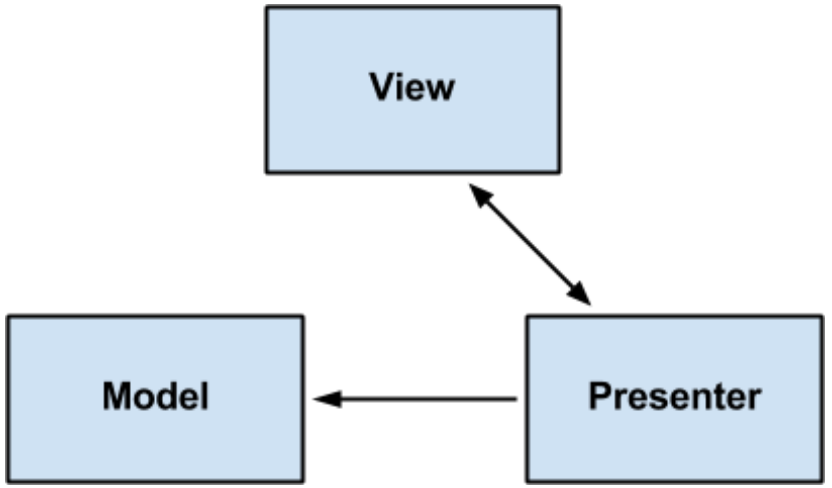
# Statistics Domain

```
public class Statistics {
    private final int completedTasks;
    private final int activeTasks;
    //... ...
}
```

```
public class GetStatistics extends UseCase<GetStatistics.RequestValues, GetStatistics.ResponseValue> {

    private final TasksRepository mTasksRepository;

    @Override
    protected void executeUseCase(RequestValues requestValues) {
        mTasksRepository.getTasks(new TasksDataSource.LoadTasksCallback() {
            @Override
            public void onTasksLoaded(List<Task> tasks) {
                int activeTasks = 0;
                int completedTasks = 0;
                // We calculate number of active and completed tasks
            }
        });
    }
```

# Presentation Layer (表现层)

- 根据Domain Layer的数据进行界面显示
- **将业务逻辑移动到领域层中更小粒度的Use case，避免Presenter的代码重复**



# Presenter

```
public class TasksPresenter implements TasksContract.Presenter {

    private final TasksContract.View mTasksView;
    private final GetTasks mGetTasks;
    private final CompleteTask mCompleteTask;

    private void loadTasks(boolean forceUpdate, final boolean showLoadingUI) {


        GetTasks.RequestValues requestValue = new GetTasks.RequestValues(forceUpdate,
                mCurrentFiltering);

        mUseCaseHandler.execute(mGetTasks, requestValue,
                new UseCase.UseCaseCallback<GetTasks.ResponseValue>() {
                    @Override
                    public void onSuccess(GetTasks.ResponseValue response) {
                        // ... ...
                    }

                    @Override
                    public void onError() {
                        // The view may not be able to handle UI updates anymore
                        if (!mTasksView.isActive()) {
                            return;
                        }
                        mTasksView.showLoadingTasksError();
                    }
                });
    }

}
```

# Test

- Presentation Layer (表现层)：小型/中型测试， Robolectric、Espresso
- Domain Layer (领域层)：小型测试，Junit、Mockito
- Data Layer（数据层）： 小型/中型测试，Robolectric（因为该层具有android依赖项），Junit、Mockito

# 中型测试

```
@RunWith(AndroidJUnit4.class)
public class TasksScreenTest {
     private void createTask(String title, String description) {
       // Click on the add task button
       onView(withId(R.id.fab_add_task)).perform(click());

       // Add task title and description
       onView(withId(R.id.add_task_title)).perform(typeText(title),
               closeSoftKeyboard()); // Type new task title
       onView(withId(R.id.add_task_description)).perform(typeText(description),
               closeSoftKeyboard()); // Type new task description and close the keyboard

       // Save the task
       onView(withId(R.id.fab_edit_task_done)).perform(click());
   }
   // ... ...
}
```

# 小型测试

```
public class TasksPresenterTest {
        @Test
    public void completeTask_ShowsTaskMarkedComplete() {
        // Given a stubbed task
        Task task = new Task("Details Requested", "For this task");

        // When task is marked as complete
        mTasksPresenter.completeTask(task);

        // Then repository is called and task marked complete UI is shown
        verify(mTasksRepository).completeTask(eq(task.getId()));
        verify(mTasksView).showTaskMarkedComplete();
    }
    //... ...
}
```

# 示例（Java）

android/architecture-samples

https://github.com/android/architecture-samples/tree/todo-mvp-clean

# 推荐（Kotlin）

android10/Android-CleanArchitecture-Kotlin

https://github.com/android10/Android-CleanArchitecture-Kotlin