ARELLO MOBILE

MVP — типичные задачи и способ их решения в Moxy

Шмаков Юрий
senneco@gmail.com
senneco

Как мы пришли к MVP

- new Thread(...).start();

Как мы пришли к MVP

- new Thread(...).start();
- AsyncTask

Как мы пришли к MVP

- new Thread(...).start();
- AsyncTask
- WorkerService (like RoboSpice)

Как мы пришли к MVP

- new Thread(...).start();
- AsyncTask
- WorkerService (like RoboSpice)
- Loaders

Как мы пришли к MVP

- new Thread(...).start();
- AsyncTask
- WorkerService (like RoboSpice)
- Loaders

+

- Android пересоздаёт Activity

Типичные задачи, которые требуется решить при использовании MVP

- Управление жизненным циклом компонентов MVP

Типичные задачи, которые требуется решить при использовании MVP

- Управление жизненным циклом компонентов MVP
  - View – за её lifecycle отвечает Android

Типичные задачи, которые требуется решить при использовании MVP

- Управление жизненным циклом компонентов MVP
  - View – за её lifecycle отвечает Android
  - Model – каждый случай индивидуален

Типичные задачи, которые требуется решить при использовании MVP

- Управление жизненным циклом компонентов MVP
  - View – за её lifecycle отвечает Android
  - Model – каждый случай индивидуален
  - Presenter – управлять нам самим:
    - Создавать только тогда, когда действительно надо
    - Доставлять во View
    - Уничтожать только тогда, когда действительно больше не нужен

Типичные задачи, которые требуется решить при использовании MVP

- Управление жизненным циклом компонентов MVP
  - View – за её lifecycle отвечает Android
  - Model – каждый случай индивидуален
  - Presenter – управлять нам самим:
    - Создавать только тогда, когда действительно надо
    - Доставлять во View
    - Уничтожать только тогда, когда действительно больше не нужен
- Восстановление View после пересоздания

Восстановление View после пересоздания

Способы:

* Хранить состояние в Bundle saveState

Восстановление View после пересоздания

Способы:

- Хранить состояние в Bundle saveState
- Like Mosby: написать класс ViewState

```
public abstract class LceViewState<D, V extends MvpLceView<D>> implements
LceViewState<D, V> {

    /** * Used to indicate that loading is currently displayed by the View */
    int STATE_SHOW_LOADING = 0;
    /** * Used to indicate that content is currently displayed by the View */
    int STATE_SHOW_CONTENT = 1;
    /** * Used to indicate that the error message is currently displayed by the view */
    int STATE_SHOW_ERROR = -1;
    …
```

Восстановление View после пересоздания

Способы:

- Хранить состояние в Bundle saveState
- Like Mosby: написать класс ViewState
- Like Dorfman's MVI: по классу на каждое состояние

```
public void render(SearchViewState viewState) {
    if (viewState instanceof SearchViewState.SearchNotStartedYet) {
        renderSearchNotStarted();
    } else if (viewState instanceof SearchViewState.Loading) {
        renderLoading();
    } else if (viewState instanceof SearchViewState.SearchResult) {
        renderResult(((SearchViewState.SearchResult) viewState).getResult());
    } else if (viewState instanceof SearchViewState.EmptyResult) {
        renderEmptyResult();
    } else if (viewState instanceof SearchViewState.Error) {
```

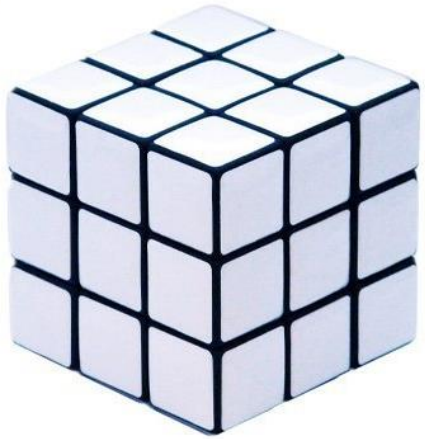
Восстановление View после пересоздания

Способы:

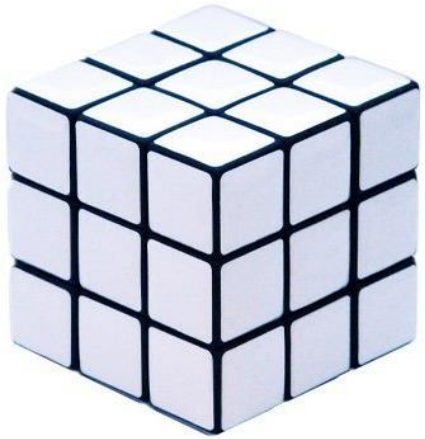
- Хранить состояние в Bundle saveState
- Like Mosby: написать класс ViewState
- Like Dorfman’s MVI: по классу на каждое состояние
- Like Moxy: хранить список команд

Как собрать кубик Рубика?

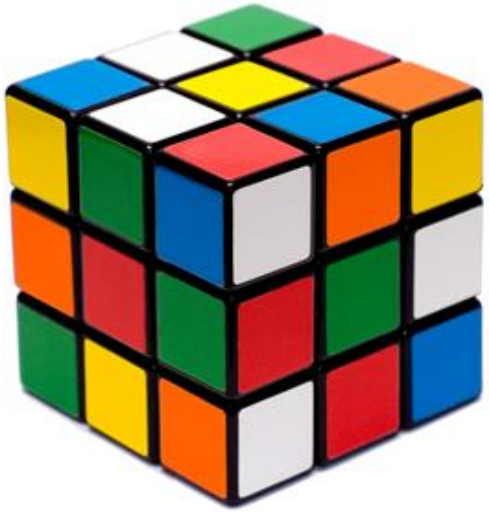
Как собрать кубик Рубика?

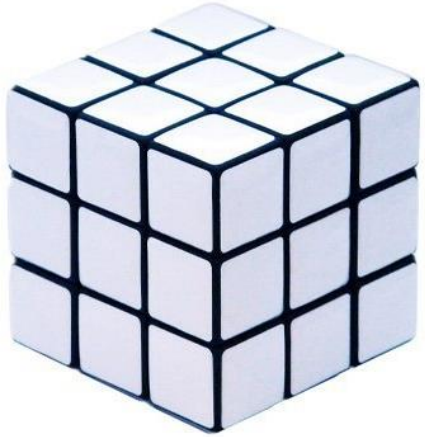


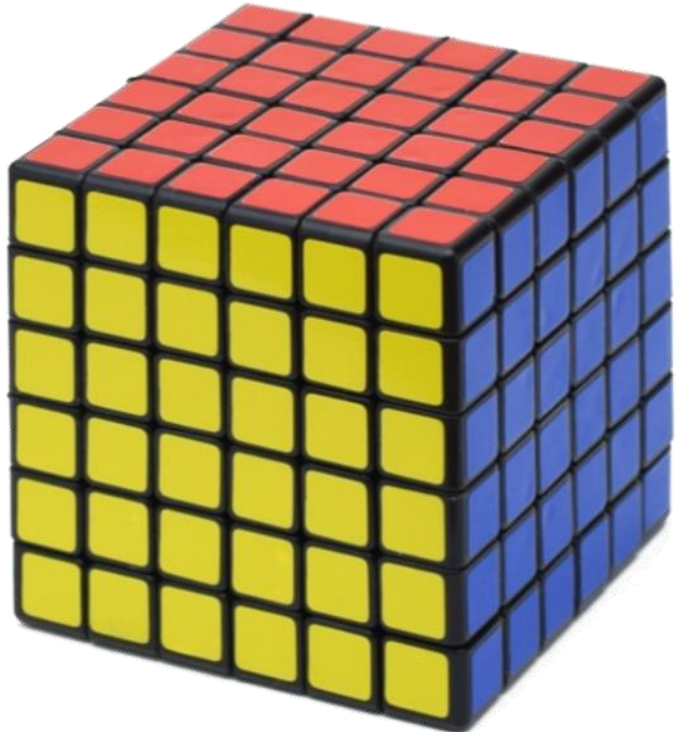
Как собрать кубик Рубика?



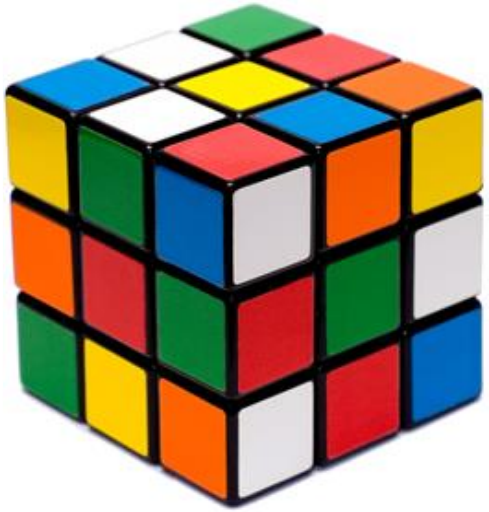
Как собрать кубик Рубика?

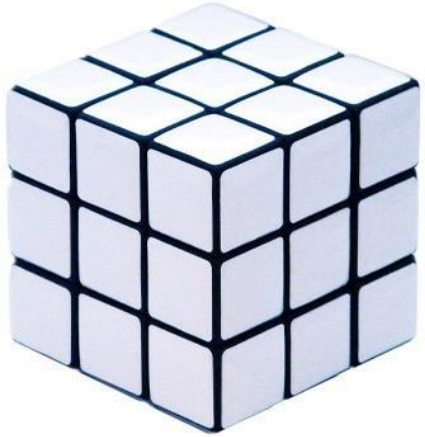


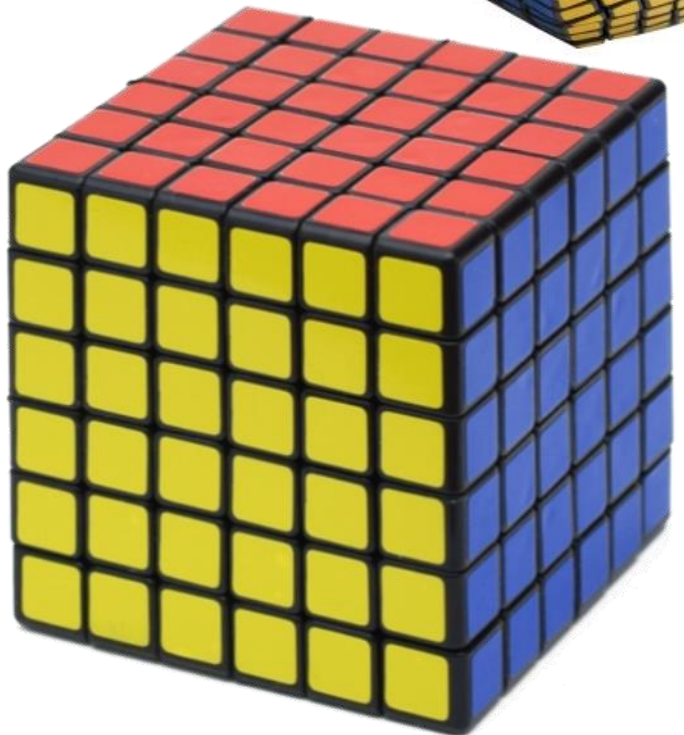




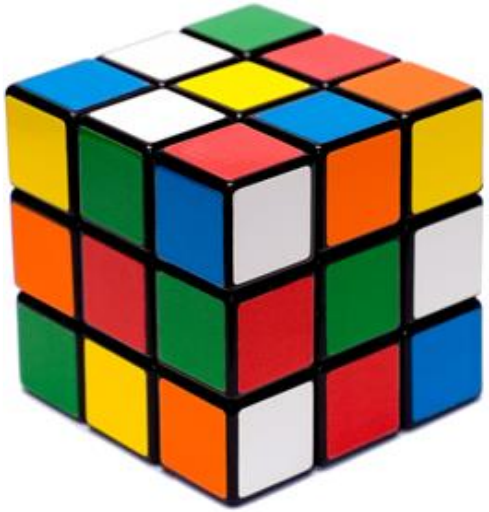
Как собрать кубик Рубика?

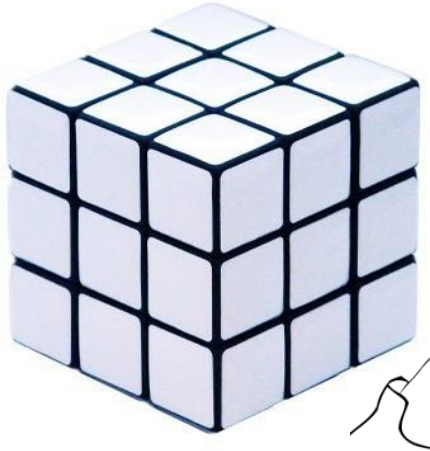


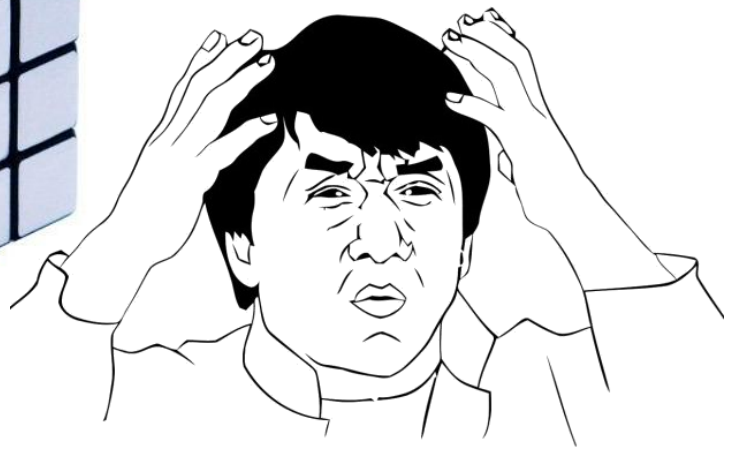


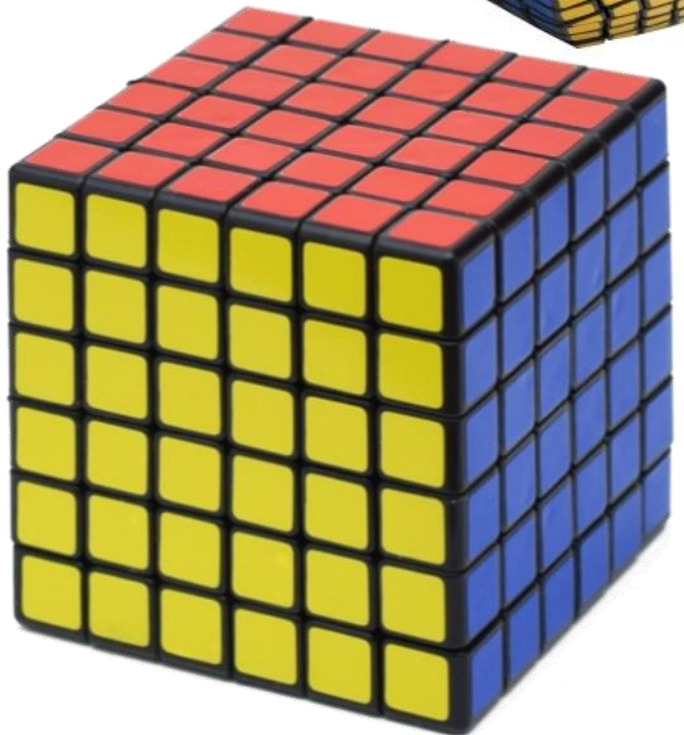


Как собрать кубик Рубика?

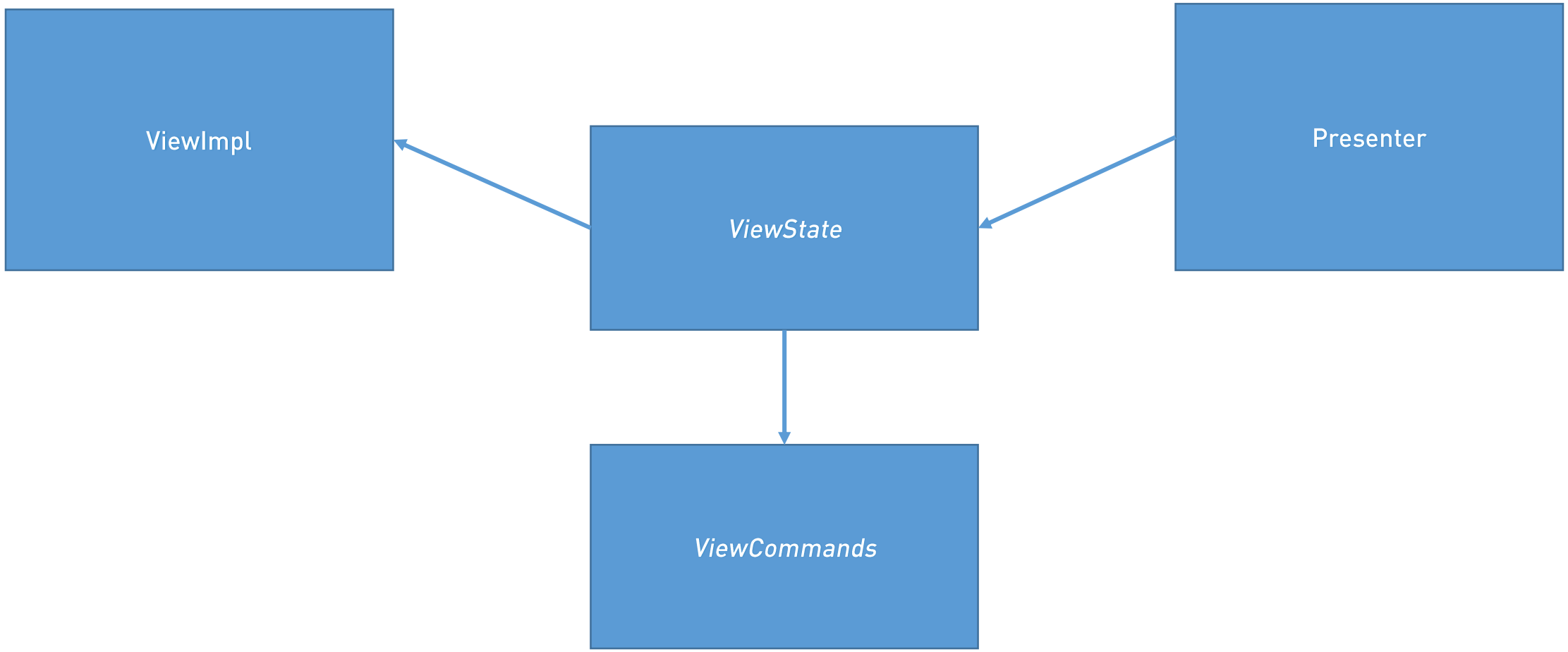




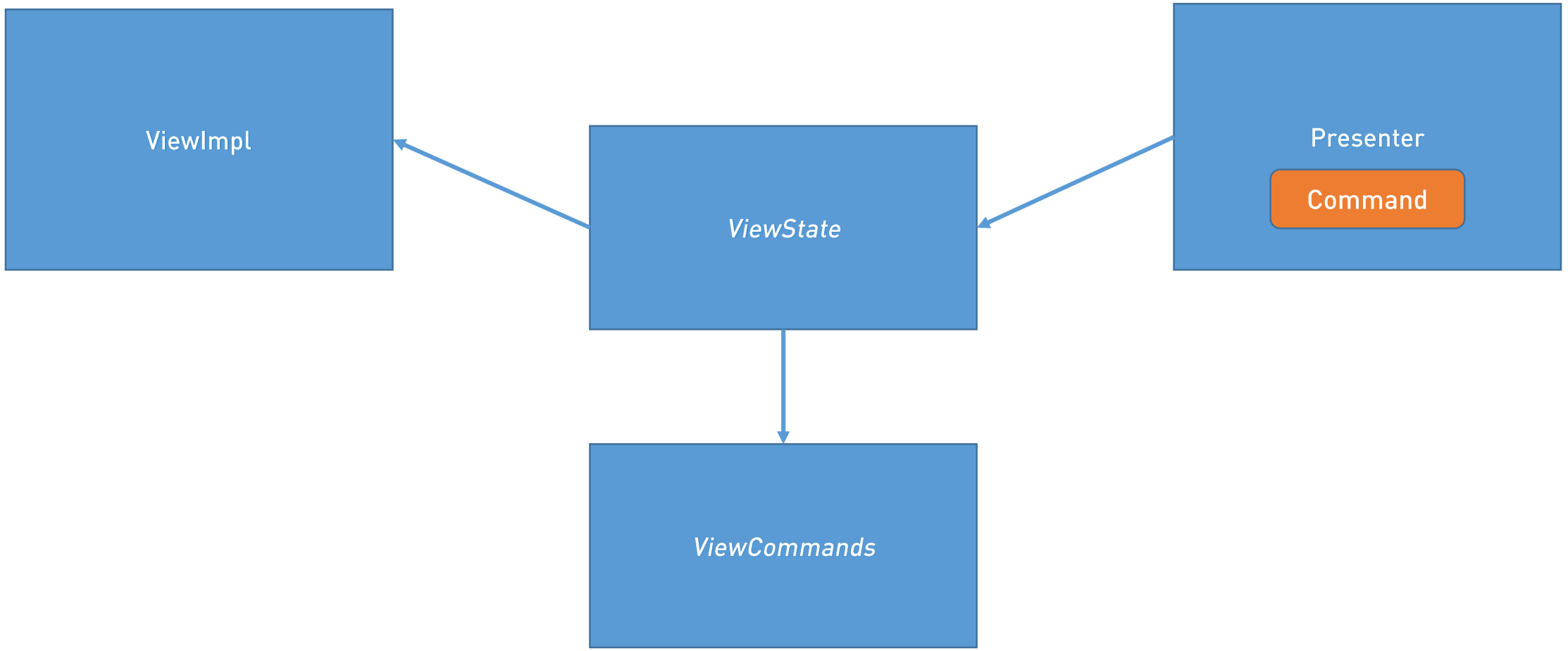




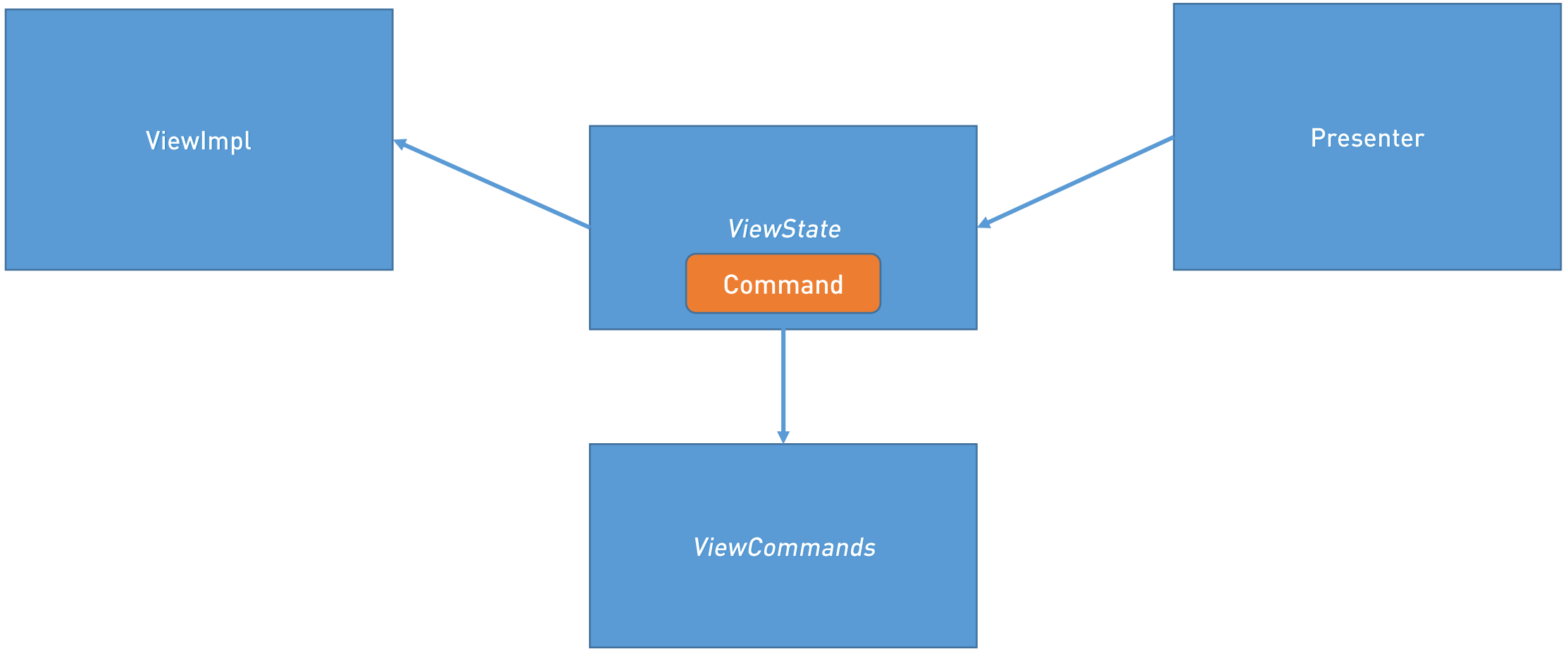
Восстановление View like Moxy



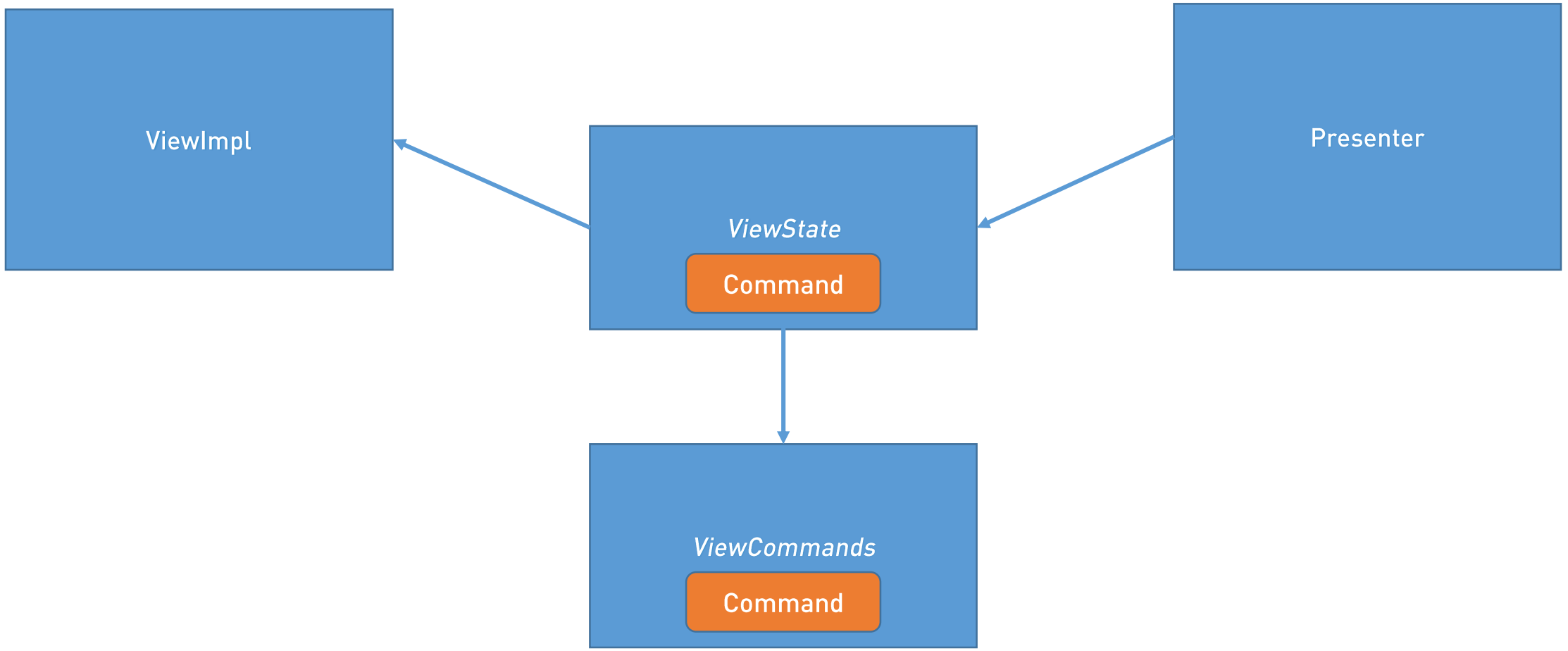
Восстановление View like Moxy



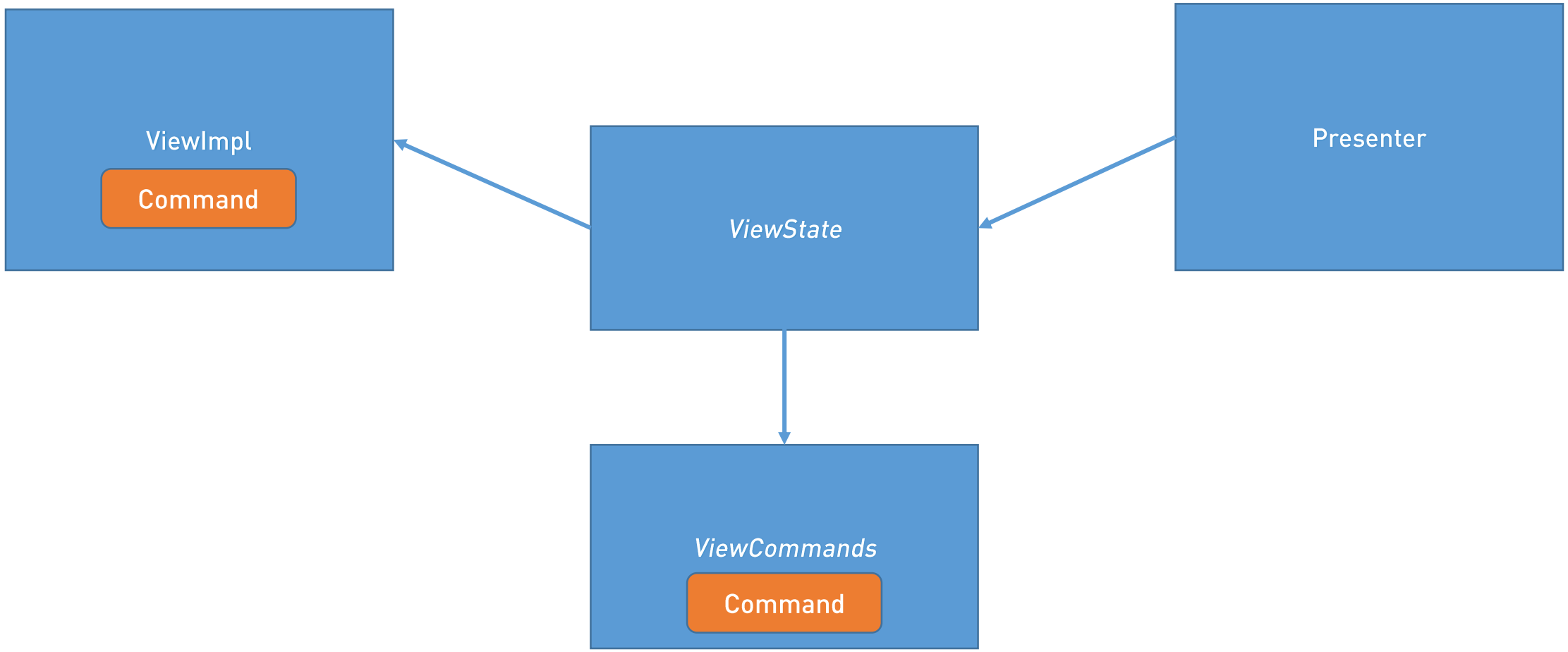
Восстановление View like Moxy



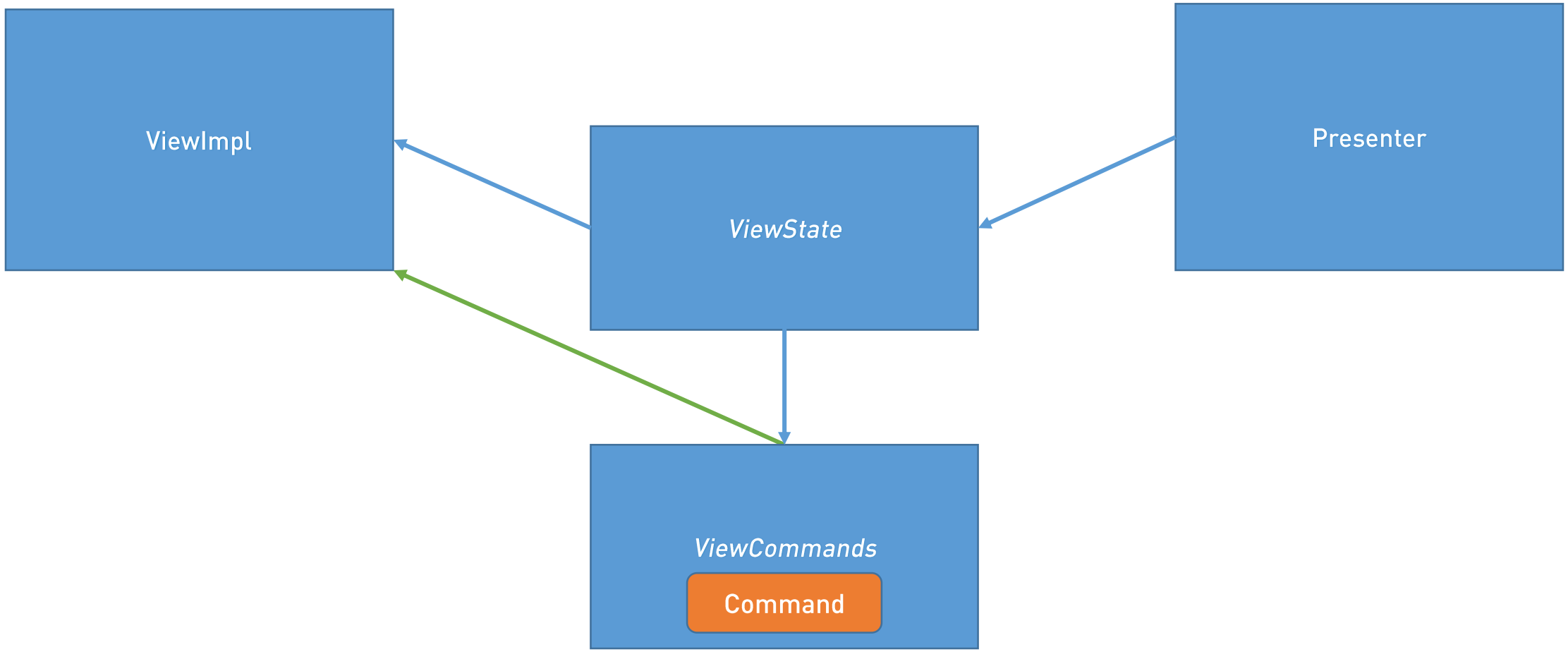
Восстановление View like Moxy



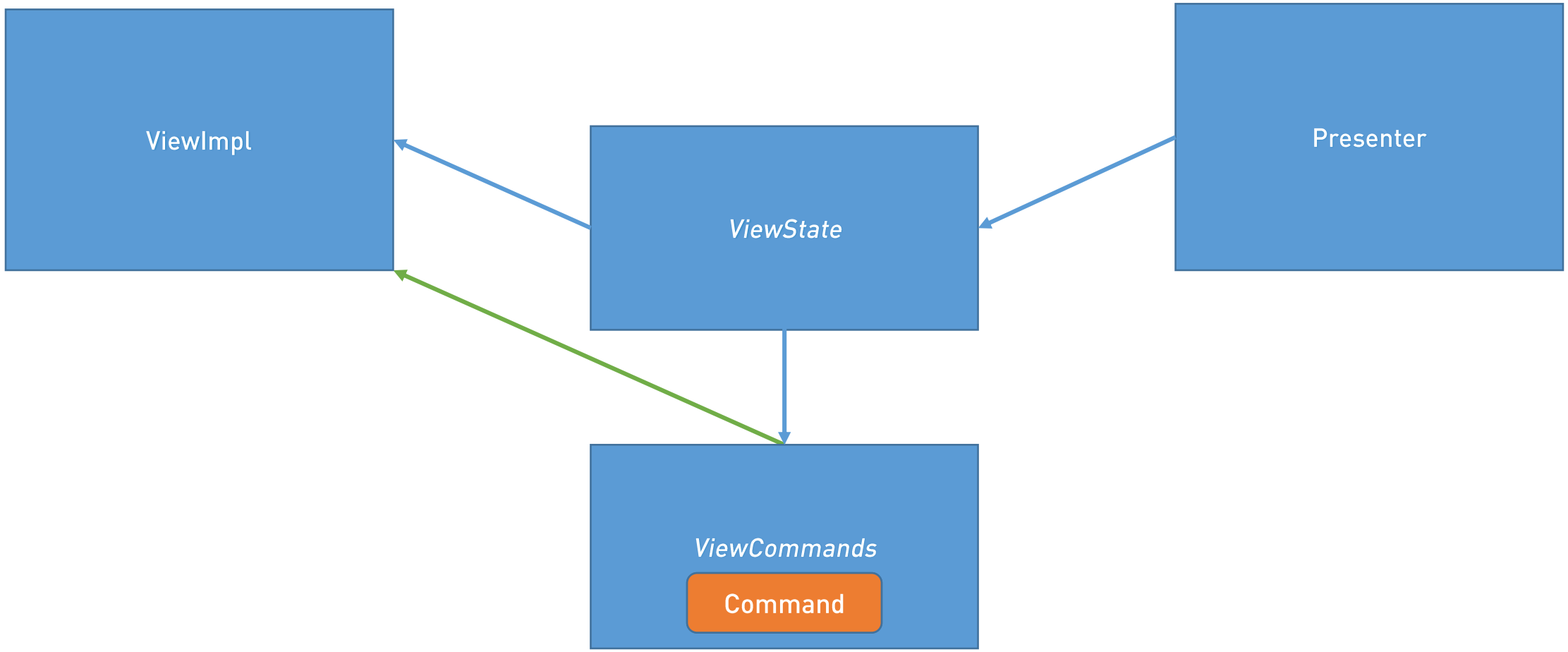
Восстановление View like Moxy



Восстановление View like Moxy



Восстановление View like Moxy



Восстановление View like Moxy

Если делать самостоятельно

```
public interface DialogView extends MvpView {
    public void showDialog();
}
```

```
public class ShowDialogCommand extends ViewCommand<DialogView> {
    @Override
    public void apply(DialogView mvpView) {
        mvpView.showDialog();
    }
}
```

Восстановление View like Moxy

Если делать самостоятельно

```
public interface DialogView extends MvpView {
    public void showDialog();
}
```

```
public class ShowDialogCommand extends ViewCommand<DialogView> {
    @Override
    public void apply(DialogView mvpView) {
        mvpView.showDialog();
    }
}
```

Восстановление View like Moxy

Если делать самостоятельно

```
public class DialogViewState extends MvpViewState<DialogView> implements DialogView {
    private ViewCommands<DialogView> viewCommands = new ViewCommands<>();

    @Override
    public void showDialog() {
        ShowDialogCommand showDialogCommand = new ShowDialogCommand();
        viewCommands.addCommand(showDialogCommand);

        for(DialogView view : views) {
            view.showDialog();
        }
    }

    @Override
    public void restoreState(DialogView view) {
        viewCommands.reapply(view);
    }

}
```

Восстановление View like Moxy

Если делать самостоятельно

```
public class DialogViewState extends MvpViewState<DialogView> implements DialogView {
    private ViewCommands<DialogView> viewCommands = new ViewCommands<>();

    @Override
    public void showDialog() {
        ShowDialogCommand showDialogCommand = new ShowDialogCommand();
        viewCommands.addCommand(showDialogCommand);

        for(DialogView view : views) {
            view.showDialog();
        }
    }

    @Override
    public void restoreState(DialogView view) {
        viewCommands.reapply(view);
    }

}
```

Восстановление View like Moxy

Если делать самостоятельно

```
public class DialogViewState extends MvpViewState<DialogView> implements DialogView {
    private ViewCommands<DialogView> viewCommands = new ViewCommands<>();

    @Override
    public void showDialog() {
        ShowDialogCommand showDialogCommand = new ShowDialogCommand();
        viewCommands.addCommand(showDialogCommand);

        for(DialogView view : views) {
            view.showDialog();
        }
    }

    @Override
    public void restoreState(DialogView view) {
        viewCommands.reapply(view);
    }

}
```

Восстановление View like Moxy

Если делать самостоятельно

```
public class DialogViewState extends MvpViewState<DialogView> implements DialogView {
    private ViewCommands<DialogView> viewCommands = new ViewCommands<>();

    @Override
    public void showDialog() {
        ShowDialogCommand showDialogCommand = new ShowDialogCommand();
        viewCommands.addCommand(showDialogCommand);

        for(DialogView view : views) {
            view.showDialog();
        }
    }

    @Override
    public void restoreState(DialogView view) {
        viewCommands.reapply(view);
    }

}
```

Восстановление View like Moxy

Если делать самостоятельно

```
public class DialogViewState extends MvpViewState<DialogView> implements DialogView {
    private ViewCommands<DialogView> viewCommands = new ViewCommands<>();

    @Override
    public void showDialog() {
        ShowDialogCommand showDialogCommand = new ShowDialogCommand();
        viewCommands.addCommand(showDialogCommand);

        for(DialogView view : views) {
            view.showDialog();
        }
    }

    @Override
    public void restoreState(DialogView view) {
        viewCommands.reapply(view);
    }

}
```

Восстановление View like Moxy

Если делать самостоятельно

```
public class DialogViewState extends MvpViewState<DialogView> implements DialogView {
    private ViewCommands<DialogView> viewCommands = new ViewCommands<>();

    @Override
    public void showDialog() {
        ShowDialogCommand showDialogCommand = new ShowDialogCommand();
        viewCommands.addCommand(showDialogCommand);

        for(DialogView view : views) {
            view.showDialog();
        }
    }

    @Override
    public void restoreState(DialogView view) {
        viewCommands.reapply(view);
    }

}
```

Восстановление View like Moxy

Если делать самостоятельно

```
public class ViewCommands<View extends MvpView> {
   private List<ViewCommand<View>> state = new ArrayList<>();

   public void reapply(View view) {
      final ArrayList<ViewCommand<View>> commands = new ArrayList<>(state);

      for (ViewCommand<View> command : commands) {
         command.apply(view);
      }
   }
   ...
}
```

Восстановление View like Moxy

Если делать самостоятельно

```
public class ViewCommands<View extends MvpView> {
    private List<ViewCommand<View>> state = new ArrayList<>();

    public void reapply(View view) {
        final ArrayList<ViewCommand<View>> commands = new ArrayList<>(state);

        for (ViewCommand<View> command : commands) {
            command.apply(view);
        }
    }
    ...
}
```

Восстановление View like Moxy

Если делать самостоятельно

Сложности генерации кода ViewState:

- Команды бывают с параметрами
- Команды с одинаковыми названиями
- Типизированные команды, типизированные параметры команд
- Activity бывает стопнутая
- Наследование интерфейсов

Восстановление View like Moxy

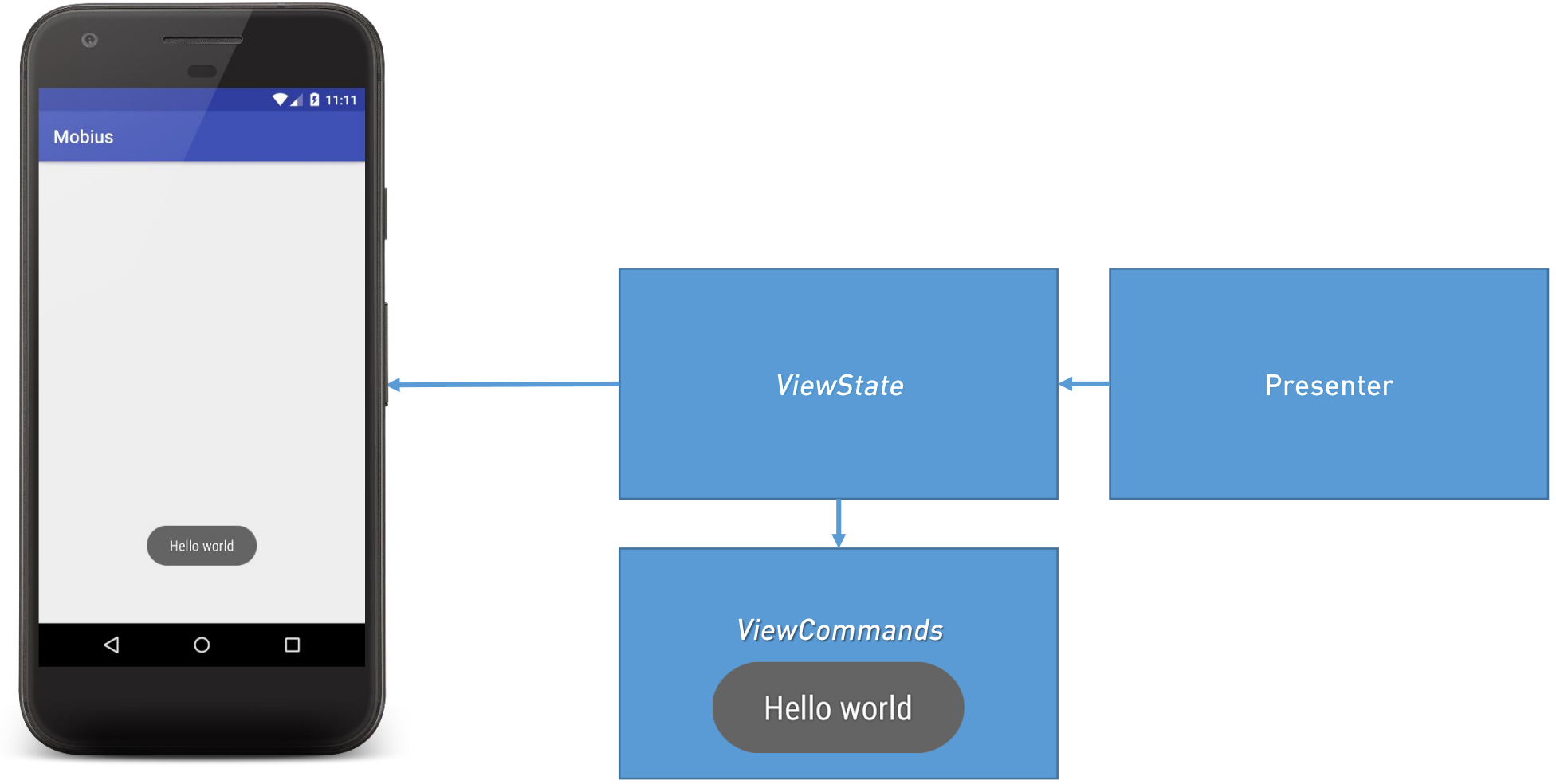
Если использовать Moxy

```
public interface DialogView {
    public void showDialog();
}
```

```
@InjectViewState
public class DialogPresenter extends MvpPresenter<DialogView> {
    ...
}
```

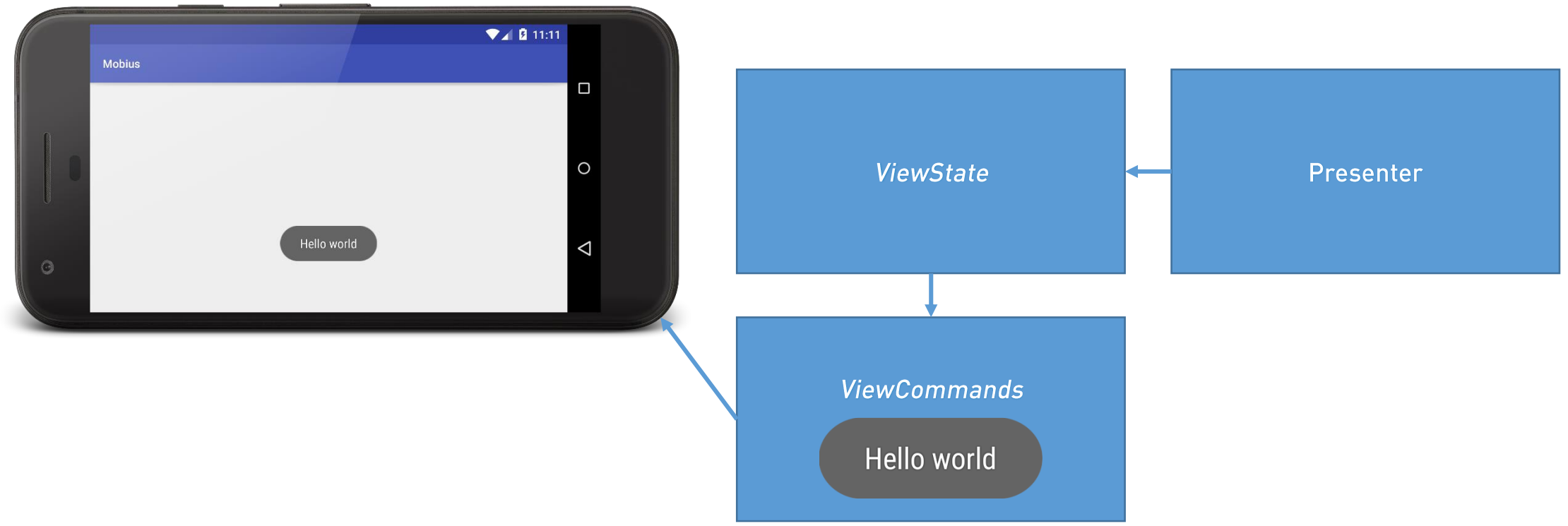
Восстановление View like Moxy

Управление очередью команд



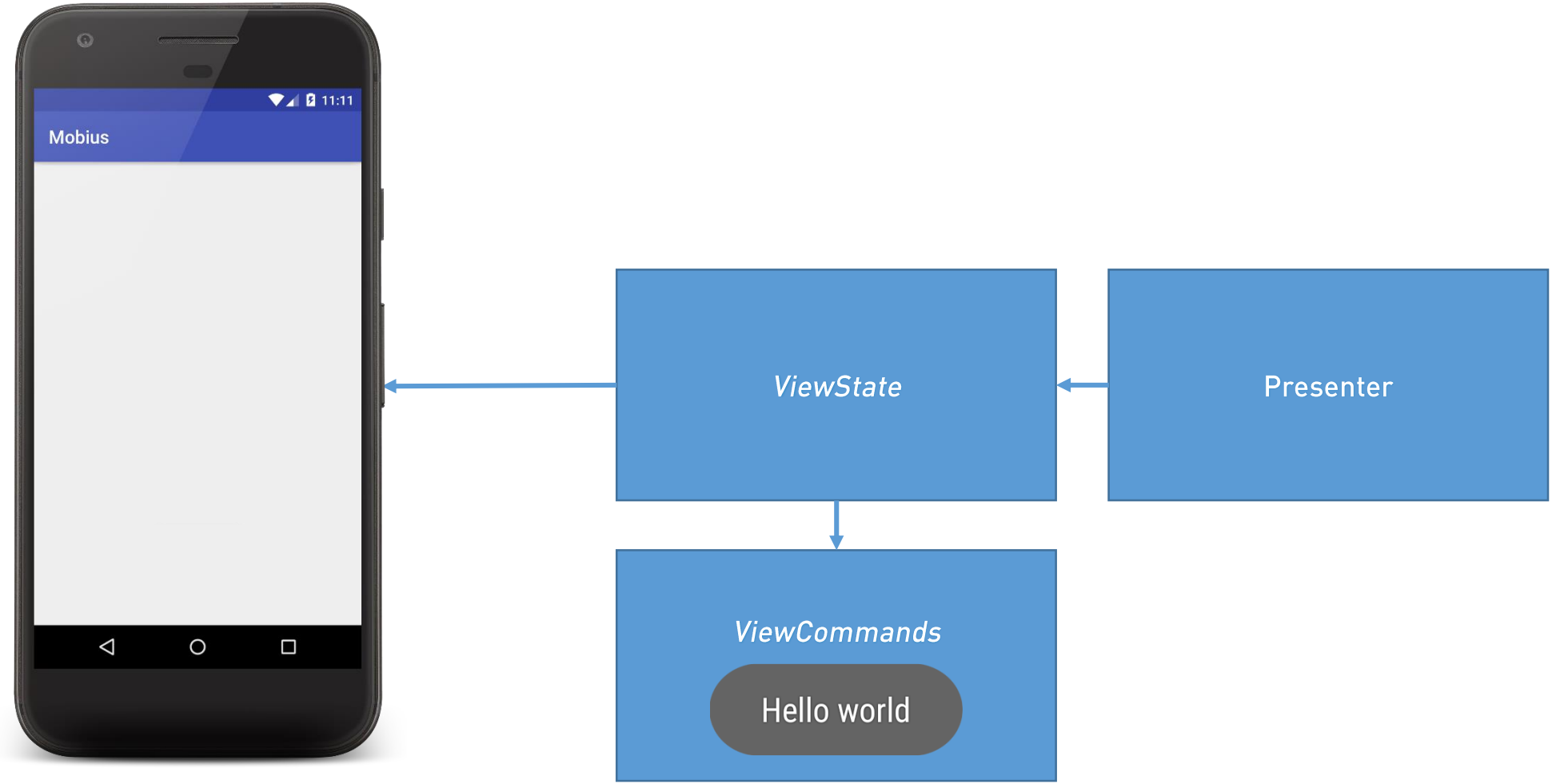
Восстановление View like Moxy

Управление очередью команд



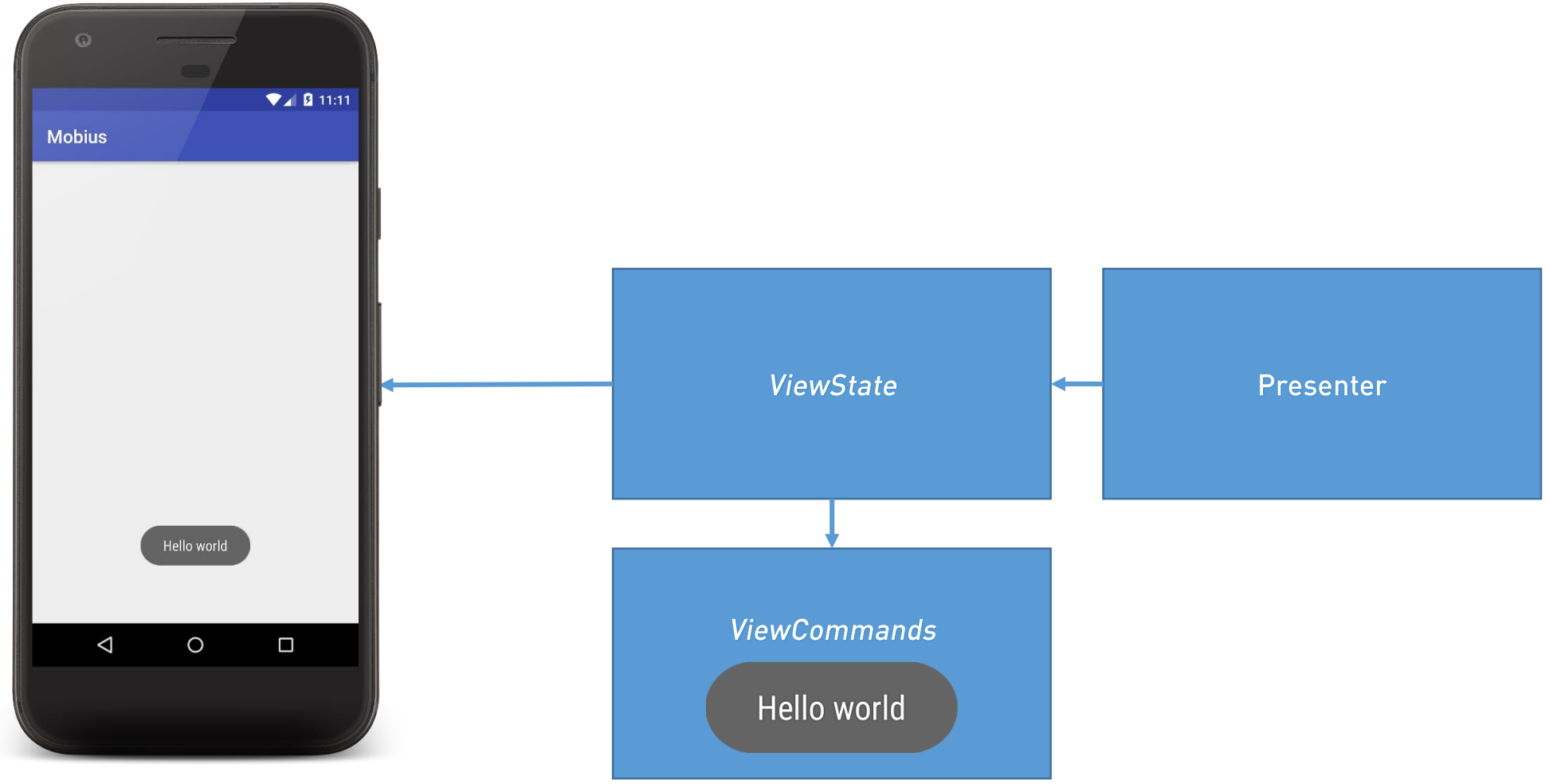
Восстановление View like Moxy

Управление очередью команд



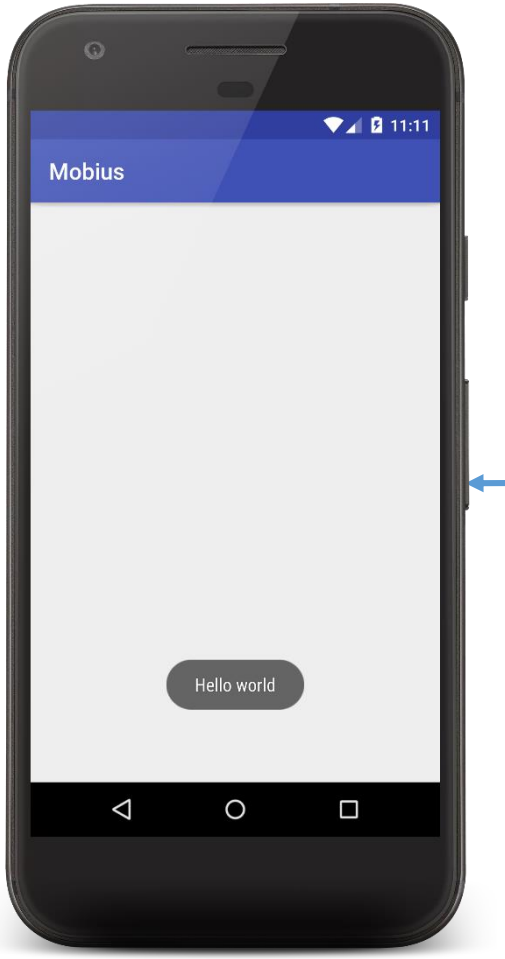
Восстановление View like Moxy

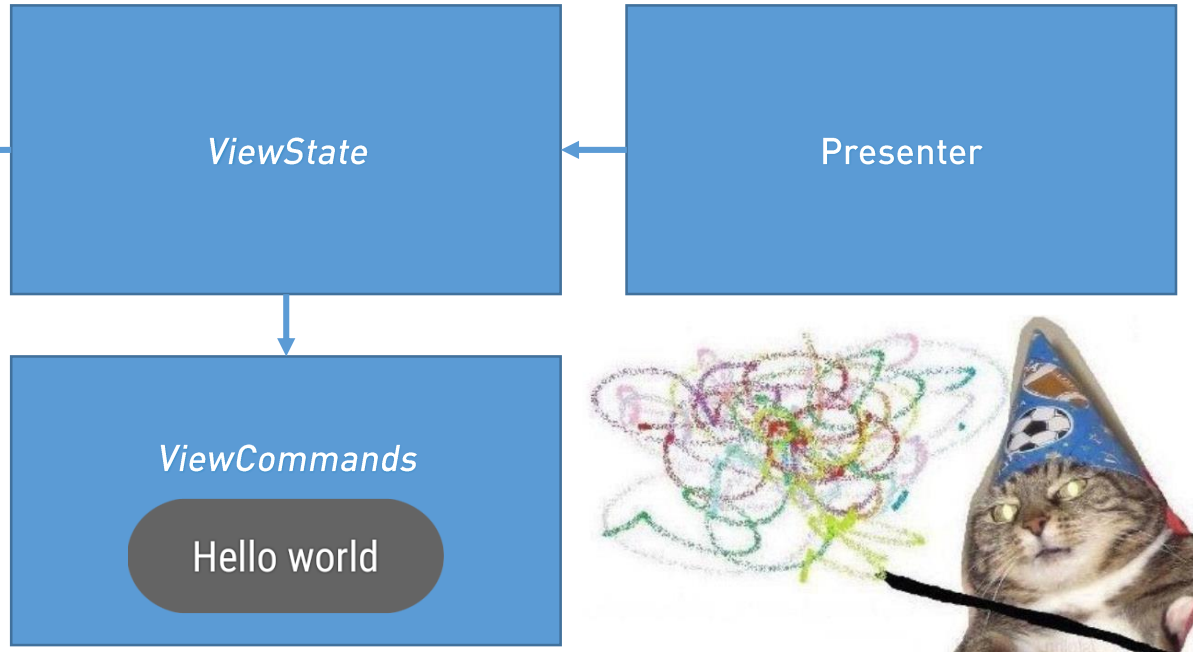
Управление очередью команд



Восстановление View like Moxy

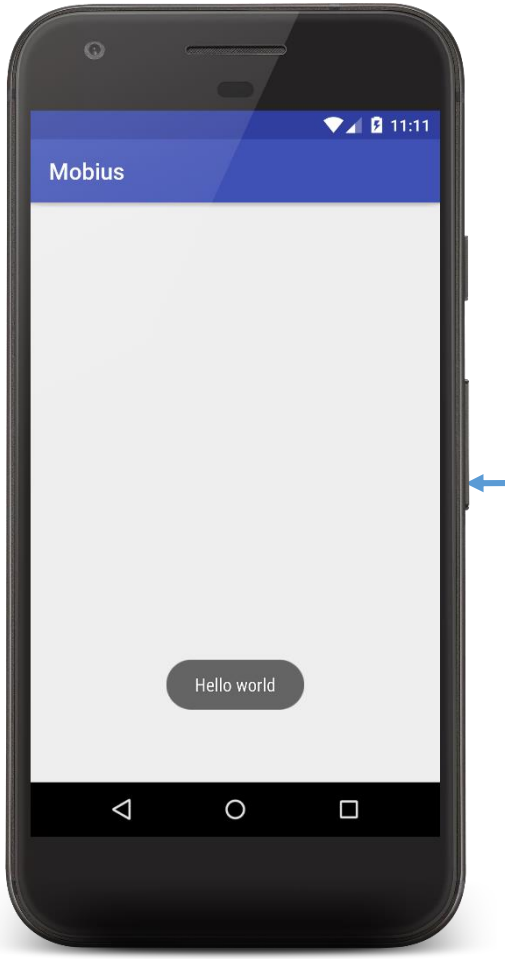
Управление очередью команд

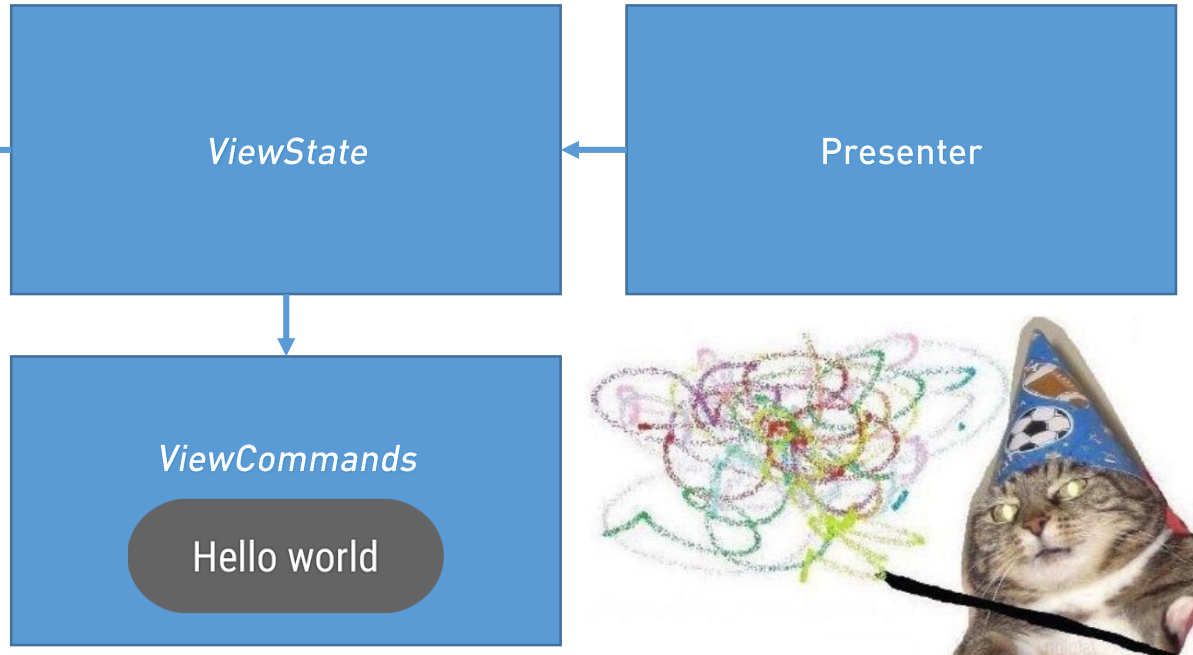




Восстановление View like Moxy

Управление очередью команд





Восстановление View like Moxy

Управление очередью команд

```
public interface StateStrategy {

    <View extends MvpView> void beforeApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand);

    <View extends MvpView> void afterApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand);
}
```

Восстановление View like Moxy

Управление очередью команд

```
public class OneExecutionStateStrategy implements StateStrategy {
    @Override
    public <View extends MvpView> void beforeApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand) {

        currentState.add(incomingCommand);

    }

    @Override
    public <View extends MvpView> void afterApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand) {

        currentState.remove(incomingCommand);

    }
}
```

Восстановление View like Moxy

Управление очередью команд

```
public class OneExecutionStateStrategy implements StateStrategy {
    @Override
    public <View extends MvpView> void beforeApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand) {

        currentState.add(incomingCommand);

    }

    @Override
    public <View extends MvpView> void afterApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand) {

        currentState.remove(incomingCommand);

    }
}
```

Восстановление View like Moxy

Управление очередью команд

```
public class OneExecutionStateStrategy implements StateStrategy {
    @Override
    public <View extends MvpView> void beforeApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand) {

        currentState.add(incomingCommand);

    }

    @Override
    public <View extends MvpView> void afterApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand) {

        currentState.remove(incomingCommand);

    }
}
```

Восстановление View like Moxy

Управление очередью команд

```
public class AddToEndStrategy implements StateStrategy {
    @Override
    public <View extends MvpView> void beforeApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand) {
        currentState.add(incomingCommand);
    }

    @Override
    public <View extends MvpView> void afterApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand) {
        // pass
    }
}
```

Восстановление View like Moxy

Управление очередью команд

```
public class AddToEndStrategy implements StateStrategy {
    @Override
    public <View extends MvpView> void beforeApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand) {
        currentState.add(incomingCommand);
    }

    @Override
    public <View extends MvpView> void afterApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand) {
        // pass
    }
}
```

Восстановление View like Moxy

Управление очередью команд

```
public class AddToEndStrategy implements StateStrategy {
    @Override
    public <View extends MvpView> void beforeApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand) {
        currentState.add(incomingCommand);
    }

    @Override
    public <View extends MvpView> void afterApply(
            List<ViewCommand<View>> currentState,
            ViewCommand<View> incomingCommand) {
        // pass
    }
}
```

Восстановление View like Moxy

Управление очередью команд

```
public class DialogViewState extends MvpViewState<DialogView> implements DialogView {
    ...
    @Override
    public void showDialog() {
        ShowDialogCommand showDialogCommand = new ShowDialogCommand();
        viewCommands.beforeApply(showDialogCommand);

        if (views == null || views.isEmpty()) {
            return;
        }

        for(DialogView view : views) {
            view.showDialog();
        }

        viewCommands.afterApply(showDialogCommand);
    }
    ...
}
```

Восстановление View like Moxy

Управление очередью команд

```
public class DialogViewState extends MvpViewState<DialogView> implements DialogView {
    ...
    @Override
    public void showDialog() {
        ShowDialogCommand showDialogCommand = new ShowDialogCommand();
        viewCommands.beforeApply(showDialogCommand);

        if (views == null || views.isEmpty()) {
            return;
        }

        for(DialogView view : views) {
            view.showDialog();
        }

        viewCommands.afterApply(showDialogCommand);
    }
    ...
}
```

Восстановление View like Moxy

Управление очередью команд

```
public class DialogViewState extends MvpViewState<DialogView> implements DialogView {
    ...
    @Override
    public void showDialog() {
        ShowDialogCommand showDialogCommand = new ShowDialogCommand();
        viewCommands.beforeApply(showDialogCommand);

        if (views == null || views.isEmpty()) {
            return;
        }

        for(DialogView view : views) {
            view.showDialog();
        }

        viewCommands.afterApply(showDialogCommand);
    }
    ...
}
```

Восстановление View like Moxy

Управление очередью команд

```
public class DialogViewState extends MvpViewState<DialogView> implements DialogView {
    ...
    @Override
    public void showDialog() {
        ShowDialogCommand showDialogCommand = new ShowDialogCommand();
        viewCommands.beforeApply(showDialogCommand);

        if (views == null || views.isEmpty()) {
            return;
        }

        for(DialogView view : views) {
            view.showDialog();
        }

        viewCommands.afterApply(showDialogCommand);
    }
    ...
}
```

Восстановление View like Moxy

Управление очередью команд

```
public interface DialogView {
    @StateStrategyType(OneExecutionStateStrategy.class)
    public void showDialog();

    public void otherMethod();
}
```

или

```
@StateStrategyType(OneExecutionStateStrategy.class)
public interface DialogView {
    public void showDialog();

    @StateStrategyType(AddToEndStateStrategy.class)
    public void otherMethod();
}
```

Жизненный цикл Presenter

Способы:

* Retain fragment / custom non configuration instance

Жизненный цикл Presenter

Способы:

- Retain fragment / custom non configuration instance
- Самостоятельное управление

Жизненный цикл Presenter

```
public class MvpActivity extends Activity {
    private MvpDelegate mvpDelegate

    protected void onCreate(Bundle state) {
        super.onCreate(state);
        mvpDelegate = new MvpDelegate(this);
        mvpDelegate.onCreate(state);
    }

    protected void onStart() {
        super.onStart();
        mvpDelegate.onAttach();
    }

    protected void onResume() {
        super.onResume();
        mvpDelegate.onAttach();
    }
```

Жизненный цикл Presenter

```
public class MvpActivity extends Activity {
    private MvpDelegate mvpDelegate

    protected void onCreate(Bundle state) {
        super.onCreate(state);
        mvpDelegate = new MvpDelegate(this);
        mvpDelegate.onCreate(state);
    }

    protected void onStart() {
        super.onStart();
        mvpDelegate.onAttach();
    }

    protected void onResume() {
        super.onResume();
        mvpDelegate.onAttach();
    }
```

Жизненный цикл Presenter

```
public class MvpActivity extends Activity {
    private MvpDelegate mvpDelegate

    protected void onCreate(Bundle state) {
        super.onCreate(state);
        mvpDelegate = new MvpDelegate(this);
        mvpDelegate.onCreate(state);
    }

    protected void onStart() {
        super.onStart();
        mvpDelegate.onAttach();
    }

    protected void onResume() {
        super.onResume();
        mvpDelegate.onAttach();
    }
```

Жизненный цикл Presenter

```
    @Override
    protected void onSaveInstanceState(Bundle state) {
        super.onSaveInstanceState(state);
        mvpDelegate.onSaveInstanceState(state);
        mvpDelegate.onDetach();
    }
    @Override
    protected void onStop() {
        super.onStop();
        mvpDelegate.onDetach();
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (isFinishing()) {
            mvpDelegate.onDestroy();
        }
    }
}
```

Жизненный цикл Presenter

```
    @Override
    protected void onSaveInstanceState(Bundle state) {
        super.onSaveInstanceState(state);
        mvpDelegate.onSaveInstanceState(state);
        mvpDelegate.onDetach();
    }
    @Override
    protected void onStop() {
        super.onStop();
        mvpDelegate.onDetach();
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (isFinishing()) {
            mvpDelegate.onDestroy();
        }
    }
}
```

Жизненный цикл Presenter

```
    @Override
    protected void onSaveInstanceState(Bundle state) {
        super.onSaveInstanceState(state);
        mvpDelegate.onSaveInstanceState(state);
        mvpDelegate.onDetach();
    }
    @Override
    protected void onStop() {
        super.onStop();
        mvpDelegate.onDetach();
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (isFinishing()) {
            mvpDelegate.onDestroy();
        }
    }
}
```

Жизненный цикл Presenter

```
    @Override
    protected void onSaveInstanceState(Bundle state) {
        super.onSaveInstanceState(state);
        mvpDelegate.onSaveInstanceState(state);
        mvpDelegate.onDetach();
    }
    @Override
    protected void onStop() {
        super.onStop();
        mvpDelegate.onDetach();
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (isFinishing()) {
            mvpDelegate.onDestroy();
        }
    }
}
```

Как создавать Presenter

Способы:

* Like Nucleus: рефлексия

Как создавать Presenter

Способы:

- Like Nucleus: рефлексия
- Like Mosby: уникальный ключ Presenter + фабричный метод

```
public class HelloWorldActivity extends MvpActivity<HelloWorldView, HelloWorldPresenter>
                                implements HelloWorldView {

    @Override
    // Called internally by Mosby
    public HelloWorldPresenter createPresenter() {
        return new HelloWorldPresenter();
    }
    ...
```

Как создавать Presenter

Способы:

- Like Nucleus: рефлексия
- Like Mosby: уникальный ключ Presenter + фабричный метод

```
public class HelloWorldActivity extends MvpActivity<HelloWorldView, HelloWorldPresenter>
                                implements HelloWorldView {

   @Override
   // Called internally by Mosby
   public HelloWorldPresenter createPresenter() {
      return new HelloWorldPresenter();
   }
   ...
```

Как создавать Presenter

Способы:

- Like Nucleus: рефлексия
- Like Mosby: уникальный ключ Presenter + фабричный метод

```
public class HelloWorldActivity extends MvpActivity<HelloWorldView, HelloWorldPresenter>
                                implements HelloWorldView {

    @Override
    // Called internally by Mosby
    public HelloWorldPresenter createPresenter() {
        return new HelloWorldPresenter();
    }
    ...
```

Как создавать Presenter

Способы:

- Like Nucleus: рефлексия
- Like Mosby: уникальный ключ Presenter + фабричный метод
- Like Moxy: уникальный ключ Presenter + фабричный метод для каждого поля Presenter (generated)

```
public class SplashActivity extends MvpAppCompatActivity implements SplashView {

    @InjectPresenter
    SplashPresenter mSplashPresenter;
    ...
```

Как подставлять Presenter в Activity

```
public class SplashActivity$$PresentersBinder extends PresenterBinder<SplashActivity> {
   public class SplashPresenterField extends PresenterField<SplashActivity> {
      public SplashPresenterField() {
         super(null, PresenterType.LOCAL, null, SplashPresenter.class);
      }
      public MvpPresenter<?> providePresenter(SplashActivity delegated) {
         return new SplashPresenter();
      }
      public void bind(SplashActivity target, MvpPresenter presenter) {
         target.splashPresenter = (SplashPresenter) presenter;
      }
   }

   public List<PresenterField<SplashActivity>> getPresenterFields() {
      List<PresenterField<SplashActivity>> presenterFields = new ArrayList<>();
      presenterFields.add(new SplashPresenterField());
      return presenterFields;
   }
}
```

Как подставлять Presenter в Activity

```
public class SplashActivity$$PresentersBinder extends PresenterBinder<SplashActivity> {
   public class SplashPresenterField extends PresenterField<SplashActivity> {
      public SplashPresenterField() {
         super(null, PresenterType.LOCAL, null, SplashPresenter.class);
      }
      public MvpPresenter<?> providePresenter(SplashActivity delegated) {
         return new SplashPresenter();
      }
      public void bind(SplashActivity target, MvpPresenter presenter) {
         target.splashPresenter = (SplashPresenter) presenter;
      }
   }

   public List<PresenterField<SplashActivity>> getPresenterFields() {
      List<PresenterField<SplashActivity>> presenterFields = new ArrayList<>();
      presenterFields.add(new SplashPresenterField());
      return presenterFields;
   }
}
```

Как подставлять Presenter в Activity

```
public class SplashActivity$$PresentersBinder extends PresenterBinder<SplashActivity> {
   public class SplashPresenterField extends PresenterField<SplashActivity> {
      public SplashPresenterField() {
         super(null, PresenterType.LOCAL, null, SplashPresenter.class);
      }
      public MvpPresenter<?> providePresenter(SplashActivity delegated) {
         return new SplashPresenter();
      }
      public void bind(SplashActivity target, MvpPresenter presenter) {
         target.splashPresenter = (SplashPresenter) presenter;
      }
   }

   public List<PresenterField<SplashActivity>> getPresenterFields() {
      List<PresenterField<SplashActivity>> presenterFields = new ArrayList<>();
      presenterFields.add(new SplashPresenterField());
      return presenterFields;
   }
}
```

Как подставлять Presenter в Activity

```
public class SplashActivity$$PresentersBinder extends PresenterBinder<SplashActivity> {
   public class SplashPresenterField extends PresenterField<SplashActivity> {
      public SplashPresenterField() {
         super(null, PresenterType.LOCAL, null, SplashPresenter.class);
      }
      public MvpPresenter<?> providePresenter(SplashActivity delegated) {
         return new SplashPresenter();
      }
      public void bind(SplashActivity target, MvpPresenter presenter) {
         target.splashPresenter = (SplashPresenter) presenter;
      }
   }

   public List<PresenterField<SplashActivity>> getPresenterFields() {
      List<PresenterField<SplashActivity>> presenterFields = new ArrayList<>();
      presenterFields.add(new SplashPresenterField());
      return presenterFields;
   }
}
```

Как подставлять Presenter в Activity

```
public void onCreate(Bundle bundle) {
    delegateTag = takeOrGenerateTag(bundle);
    PresentersBinder<Delegated> presentersBinder = MoxyReflector.getPresentersBinder(this);
    List<PresenterField<Delegated>> presenterFields = presenterBinder.getPresenterFields();
    for (PresenterField<Delegated> presenterField : presenterFields) {
        String presenterTag = delegateTag + "$" + presenterField.getTag(target);
        MvpPresenter<? super Delegated> presenter = presenterStore.get(tag);
        if (presenter == null) {
            presenter = presenterField.providePresenter(target);
            presenterStore.add(tag, presenterClass);
        }
        presenterField.bind(delegated, presenter);
    }
}
```

Как подставлять Presenter в Activity

```
public void onCreate(Bundle bundle) {
    delegateTag = takeOrGenerateTag(bundle);
    PresentersBinder<Delegated> presentersBinder = MoxyReflector.getPresentersBinder(this);
    List<PresenterField<Delegated>> presenterFields = presenterBinder.getPresenterFields();
    for (PresenterField<Delegated> presenterField : presenterFields) {
        String presenterTag = delegateTag + "$" + presenterField.getTag(target);
        MvpPresenter<? super Delegated> presenter = presenterStore.get(tag);
        if (presenter == null) {
            presenter = presenterField.providePresenter(target);
            presenterStore.add(tag, presenterClass);
        }
        presenterField.bind(delegated, presenter);
    }
}
```

Как подставлять Presenter в Activity

```
public void onCreate(Bundle bundle) {
    delegateTag = takeOrGenerateTag(bundle);
    PresentersBinder<Delegated> presentersBinder = MoxyReflector.getPresentersBinder(this);
    List<PresenterField<Delegated>> presenterFields = presenterBinder.getPresenterFields();
    for (PresenterField<Delegated> presenterField : presenterFields) {
        String presenterTag = delegateTag + "$" + presenterField.getTag(target);
        MvpPresenter<? super Delegated> presenter = presenterStore.get(tag);
        if (presenter == null) {
            presenter = presenterField.providePresenter(target);
            presenterStore.add(tag, presenterClass);
        }
        presenterField.bind(delegated, presenter);
    }
}
```

Как подставлять Presenter в Activity

```
public void onCreate(Bundle bundle) {
    delegateTag = takeOrGenerateTag(bundle);
    PresentersBinder<Delegated> presentersBinder = MoxyReflector.getPresentersBinder(this);
    List<PresenterField<Delegated>> presenterFields = presenterBinder.getPresenterFields();
    for (PresenterField<Delegated> presenterField : presenterFields) {
        String presenterTag = delegateTag + "$" + presenterField.getTag(target);
        MvpPresenter<? super Delegated> presenter = presenterStore.get(tag);
        if (presenter == null) {
            presenter = presenterField.providePresenter(target);
            presenterStore.add(tag, presenterClass);
        }
        presenterField.bind(delegated, presenter);
    }
}
```

Как подставлять Presenter в Activity

```
public void onCreate(Bundle bundle) {
    delegateTag = takeOrGenerateTag(bundle);
    PresentersBinder<Delegated> presentersBinder = MoxyReflector.getPresentersBinder(this);
    List<PresenterField<Delegated>> presenterFields = presenterBinder.getPresenterFields();
    for (PresenterField<Delegated> presenterField : presenterFields) {
        St
        Mv
        if
        }
        presenterField.bind(delegated, presenter);
    }
}
```

```
public List<PresenterField<SplashActivity>> getPresenterFields() {
    List<PresenterField<SplashActivity>> presenterFields = new ArrayList<>();
    presenterFields.add(new SplashPresenterField());
    return presenterFields;
}
```

Как подставлять Presenter в Activity

```
public void onCreate(Bundle bundle) {
    delegateTag = takeOrGenerateTag(bundle);
    PresentersBinder<Delegated> presentersBinder = MoxyReflector.getPresentersBinder(this);
    List<PresenterField<Delegated>> presenterFields = presenterBinder.getPresenterFields();
    for (PresenterField<Delegated> presenterField : presenterFields) {
        String presenterTag = delegateTag + "$" + presenterField.getTag(target);
        MvpPresenter<? super Delegated> presenter = presenterStore.get(tag);
        if (presenter == null) {
            presenter = presenterField.providePresenter(target);
            presenterStore.add(tag, presenterClass);
        }
        presenterField.bind(delegated, presenter);
    }
}
```

Как подставлять Presenter в Activity

```
public void onCreate(Bundle bundle) {
    delegateTag = takeOrGenerateTag(bundle);
    PresentersBinder<Delegated> presentersBinder = MoxyReflector.getPresentersBinder(this);
    List<PresenterField<Delegated>> presenterFields = presenterBinder.getPresenterFields();
    for (PresenterField<Delegated> presenterField : presenterFields) {
        String presenterTag = delegateTag + "$" + presenterField.getTag(target);
        MvpPresenter<? super Delegated> presenter = presenterStore.get(tag);
        if (presenter == null) {
            presenter = presenterField.providePresenter(target);
            presenterStore.add(tag, presenterClass);
        }
        presenterField.bind(delegated, presenter);
    }
}
```

Как подставлять Presenter в Activity

```
public void onCreate(Bundle bundle) {
    delegateTag = takeOrGenerateTag(bundle);
    Presente                                                    der(this);
    List<Pre                                                    Fields();
    for (Pre
        Strin
        MvpPr
        if (presenter == null) {
            presenter = presenterField.providePresenter(target);
            presenterStore.add(tag, presenterClass);
        }
        presenterField.bind(delegated, presenter);
    }
}
```

```
@Override
public MvpPresenter<?> providePresenter(SplashActivity activity) {
    return new SplashPresenter();
}
```

Как подставлять Presenter в Activity

```
public void onCreate(Bundle bundle) {
    delegateTag = takeOrGenerateTag(bundle);
    PresentersBinder<Delegated> presentersBinder = MoxyReflector.getPresentersBinder(this);
    List<PresenterField<Delegated>> presenterFields = presenterBinder.getPresenterFields();
    for (PresenterField<Delegated> presenterField : presenterFields) {
        String presenterTag = delegateTag + "$" + presenterField.getTag(target);
        MvpPresenter<? super Delegated> presenter = presenterStore.get(tag, presenterClass);
        if (presenter == null) {
            presenter = presenterField.providePresenter(target);
            presenterStore.add(tag, presenterClass);
        }
        presenterField.bind(delegated, presenter);
    }
}
```

Как подставлять Presenter в Activity

```
public void onCreate(Bundle bundle) {
    delegateTag = takeOrGenerateTag(bundle);
    PresentersBinder<Delegated> presentersBinder = MoxyReflector.getPresentersBinder(this);
    List<PresenterField<Delegated>> presenterFields = presenterBinder.getPresenterFields();
    for (PresenterField<Delegated> presenterField : presenterFields) {
        String presenterTag = delegateTag + "$" + presenterField.getTag(target);
        MvpPresenter<? super Delegated> presenter = presenterStore.get(tag, presenterClass);
        if (presenter == null) {
            presenter = presenterField.providePresenter(target);
            presenterStore.add(tag, presenterClass);
        }
        presenterField.bind(delegated, presenter);
    }
}
```

Как подставлять Presenter в Activity

```
public void onCreate(Bundle bundle) {
    delegateTag = takeOrGenerateTag(bundle);
    PresentersBinder<Delegated> presentersBinder = MoxyReflector.getPresentersBinder(this);
    List<PresenterField<Delegated>> presenterFields = presenterBinder.getPresenterFields();
    for (PresenterField<Delegated> presenterField : presenterFields) {
        Str                                                              ;
        Mvp                                                              senterClass);
        if
            @Override
            public void bind(SplashActivity target, MvpPresenter presenter) {
                target.splashPresenter = (SplashPresenter) presenter;
            }
        }
        presenterField.bind(delegated, presenter);
    }
}
```

Управление созданием Presenter

```
public class SplashActivity extends MvpAppCompatActivity implements SplashView {

    @InjectPresenter
    SplashPresenter splashPresenter;

    @ProvidePresenter
    public SplashPresenter provideSplashPresenter() {
        return new SplashPresenter();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ...
    }
    ...
}
```

Управление созданием Presenter

```
public class SplashActivity extends MvpAppCompatActivity implements SplashView {

    @InjectPresenter
    SplashPresenter splashPresenter;

    @ProvidePresenter
    public SplashPresenter provideSplashPresenter() {
        return new SplashPresenter();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ...
    }
    ...
}
```

Управление созданием Presenter

```
public class SplashActivity extends MvpAppCompatActivity implements SplashView {

    @InjectPresenter
    SplashPresenter splashPresenter;

    @ProvidePresenter
    public SplashPresenter provideSplashPresenter() {
        return new SplashPresenter();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ...
    }
    ...
}
```

Управление созданием Presenter

```
public class SplashActivity$$PresentersBinder extends PresenterBinder<SplashActivity> {
   public class SplashPresenterField extends PresenterField<SplashActivity> {
      public SplashPresenterField() {
         super(null, PresenterType.LOCAL, null, SplashPresenter.class);
      }

      @Override
      public MvpPresenter<?> providePresenter(SplashActivity activity) {
         return activity.provideSplashPresenter();
      }

      @Override
      public void bind(SplashActivity target, MvpPresenter presenter) {
         target.splashPresenter = (SplashPresenter) presenter;
      }
   }
...
```

Управление созданием Presenter

```
public class SplashActivity extends MvpAppCompatActivity implements SplashView {
    @Inject
    SplashPresenter daggerSplashPresenter;

    @InjectPresenter
    SplashPresenter splashPresenter;

    @ProvidePresenter
    public SplashPresenter provideSplashPresenter() {
        return daggerSplashPresenter;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        GithubApp.getAppComponent().inject(this);
        super.onCreate(savedInstanceState);
        ...
    }
    ...
```

Управление созданием Presenter

```
public class SplashActivity extends MvpAppCompatActivity implements SplashView {
    @Inject
    SplashPresenter daggerSplashPresenter;

    @InjectPresenter
    SplashPresenter splashPresenter;

    @ProvidePresenter
    public SplashPresenter provideSplashPresenter() {
        return daggerSplashPresenter;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        GithubApp.getAppComponent().inject(this);
        super.onCreate(savedInstanceState);
        ...
    }
    ...
```

Управление созданием Presenter

```
public class SplashActivity extends MvpAppCompatActivity implements SplashView {
    @Inject
    SplashPresenter daggerSplashPresenter;

    @InjectPresenter
    SplashPresenter splashPresenter;

    @ProvidePresenter
    public SplashPresenter provideSplashPresenter() {
        return daggerSplashPresenter;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        GithubApp.getAppComponent().inject(this);
        super.onCreate(savedInstanceState);
        ...
    }
    ...
```

Управление созданием Presenter

```
public class SplashActivity extends MvpAppCompatActivity implements SplashView {
    @Inject
    SplashPresenter daggerSplashPresenter;

    @InjectPresenter
    SplashPresenter splashPresenter;

    @ProvidePresenter
    public SplashPresenter provideSplashPresenter() {
        return daggerSplashPresenter;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        GithubApp.getAppComponent().inject(this);
        super.onCreate(savedInstanceState);
        ...
    }
    ...
```

Управление созданием Presenter

```
public class SplashActivity extends MvpAppCompatActivity implements SplashView {
    @Inject
    SplashPresenter daggerSplashPresenter;

    @InjectPresenter
    SplashPresenter splashPresenter;

    @ProvidePresenter
    public SplashPresenter provideSplashPresenter() {
        return daggerSplashPresenter;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        GithubApp.getAppComponent().inject(this);
        super.onCreate(savedInstanceState);
        ...
    }
    ...
```

Управление созданием Presenter

```
public class SplashActivity extends MvpAppCompatActivity implements SplashView {

    @Inject @InjectPresenter
    SplashPresenter splashPresenter;

    @ProvidePresenter
    public SplashPresenter provideSplashPresenter() {
        return splashPresenter;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        GithubApp.getAppComponent().inject(this);

        super.onCreate(savedInstanceState);

        ...
    }
    ...
```

Управление созданием Presenter

```
public class SplashActivity extends MvpAppCompatActivity implements SplashView {

    @Inject @InjectPresenter
    SplashPresenter splashPresenter;

    @ProvidePresenter
    public SplashPresenter provideSplashPresenter() {
        return splashPresenter;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        GithubApp.getAppComponent().inject(this);

        super.onCreate(savedInstanceState);

        ...
    }
    ...
```

UBER-SAMPLE

UBER-SAMPLE

ViewHolder → MvpView

ViewHolder → MvpView

```
public class RepositoryHolder implements RepositoryView {
    private Repository repository;
    private MvpDelegate mvpDelegate;

    @InjectPresenter
    RepositoryPresenter repositoryPresenter;

    @ProvidePresenter
    RepositoryPresenter provideRepositoryPresenter() {
        return new RepositoryPresenter(repository);
    }

    ...
```

ViewHolder → MvpView

```
public class RepositoryHolder implements RepositoryView {
    private Repository repository;
    private MvpDelegate mvpDelegate;

    @InjectPresenter
    RepositoryPresenter repositoryPresenter;

    @ProvidePresenter
    RepositoryPresenter provideRepositoryPresenter() {
        return new RepositoryPresenter(repository);
    }
    ...
    MvpDelegate getMvpDelegate() {
        if (mvpDelegate == null) {
            mvpDelegate = new MvpDelegate<>(this);
            mvpDelegate.setParentDelegate(Adapter.this.getMvpDelegate(), repository.getId());

        }
        return mvpDelegate;
    }
    ...
```

ViewHolder → MvpView

```
MvpDelegate getMvpDelegate() {
    if (mvpDelegate == null) {
        mvpDelegate = new MvpDelegate<>(this);
        mvpDelegate.setParentDelegate(Adapter.this.getMvpDelegate(), repository.getId());

    }
    return mvpDelegate;
}
...
```

ViewHolder → MvpView

```
MvpDelegate getMvpDelegate() {
    if (mvpDelegate == null) {
        mvpDelegate = new MvpDelegate<>(this);
        mvpDelegate.setParentDelegate(Adapter.this.getMvpDelegate(), repository.getId());

    }
    return mvpDelegate;
}
...
void bind(int position, Repository repository) {
    if (repository != null) {
        getMvpDelegate().onSaveInstanceState();
        getMvpDelegate().onDetach();
        getMvpDelegate().onDestroyView();
        mvpDelegate = null;
    }
    this.repository = repository;
    getMvpDelegate().onCreate();
    ...
```

ViewHolder → MvpView

```
void bind(int position, Repository repository) {
    if (repository != null) {
        getMvpDelegate().onSaveInstanceState();
        getMvpDelegate().onDetach();
        getMvpDelegate().onDestroyView();
        mvpDelegate = null;
    }
    this.repository = repository;
    getMvpDelegate().onCreate();
    ...
    getMvpDelegate().onAttach();
}
```

ViewHolder → MvpView

```
void bind(int position, Repository repository) {
    if (repository != null) {
        getMvpDelegate().onSaveInstanceState();
        getMvpDelegate().onDetach();
        getMvpDelegate().onDestroyView();
        mvpDelegate = null;
    }
    this.repository = repository;
    getMvpDelegate().onCreate();
    ...
    getMvpDelegate().onAttach();
}
```

ViewHolder → MvpView

```
void bind(int position, Repository repository) {
    if (repository != null) {
        getMvpDelegate().onSaveInstanceState();
        getMvpDelegate().onDetach();
        getMvpDelegate().onDestroyView();
        mvpDelegate = null;
    }
    this.repository = repository;
    getMvpDelegate().onCreate();
    ...
    getMvpDelegate().onAttach();
}
```

ViewHolder → MvpView

```
void bind(int position, Repository repository) {
    if (repository != null) {
        getMvpDelegate().onSaveInstanceState();
        getMvpDelegate().onDetach();
        getMvpDelegate().onDestroyView();
        mvpDelegate = null;
    }
    this.repository = repository;
    getMvpDelegate().onCreate();
    ...
    getMvpDelegate().onAttach();
}
```

ViewHolder → MvpView

```
void bind(int position, Repository repository) {
    if (repository != null) {
        getMvpDelegate().onSaveInstanceState();
        getMvpDelegate().onDetach();
        getMvpDelegate().onDestroyView();
        mvpDelegate = null;
    }
    this.repository = repository;
    getMvpDelegate().onCreate();
    ...
    getMvpDelegate().onAttach();
}
```

ViewHolder → MvpView

```
void bind(int position, Repository repository) {
    if (repository != null) {
        getMvpDelegate().onSaveInstanceState();
        getMvpDelegate().onDetach();
        getMvpDelegate().onDestroyView();
        mvpDelegate = null;
    }
    this.repository = repository;
    getMvpDelegate().onCreate();
    ...
    getMvpDelegate().onAttach();
}
```

ViewHolder → MvpView

```
void bind(int position, Repository repository) {
    if (repository != null) {
        getMvpDelegate().onSaveInstanceState();
        getMvpDelegate().onDetach();
        getMvpDelegate().onDestroyView();
        mvpDelegate = null;
    }
    this.repository = repository;
    getMvpDelegate().onCreate();
    ...
    getMvpDelegate().onAttach();
}
```

ViewHolder → MvpView

```
void bind(int position, Repository repository) {
    if (repository != null) {
        getMvpDelegate().onSaveInstanceState();
        getMvpDelegate().onDetach();
        getMvpDelegate().onDestroyView();
        mvpDelegate = null;
    }
    this.repository = repository;
    getMvpDelegate().onCreate();
    ...
    getMvpDelegate().onAttach();
}
```

Итого

- Имеем актуальное состояние View
- Простой жизненный цикл Presenter
- Знаем немного внутренностей Moxy

Можно начать думать над архитектурой

ARELLO MOBILE

Спасибо за внимание

Шмаков Юрий

senneco@gmail.com

senneco