Popup view on Mortar

@KeithYokoma - Drivemode, Inc.

potatotips #21

@KeithYokoma

Keishin Yokomaku at Drivemode, Inc. as Engineer

Experience
1. SNS client and photo book application for Android
2. Driver's application for Android

Publications
1. Android Training
2. Mixi official smartphone application development guide

Like
Motorsport, Bicycle, Photography, Tumblr

Organizations

Flow & Mortar

Advocating Against Android Fragments

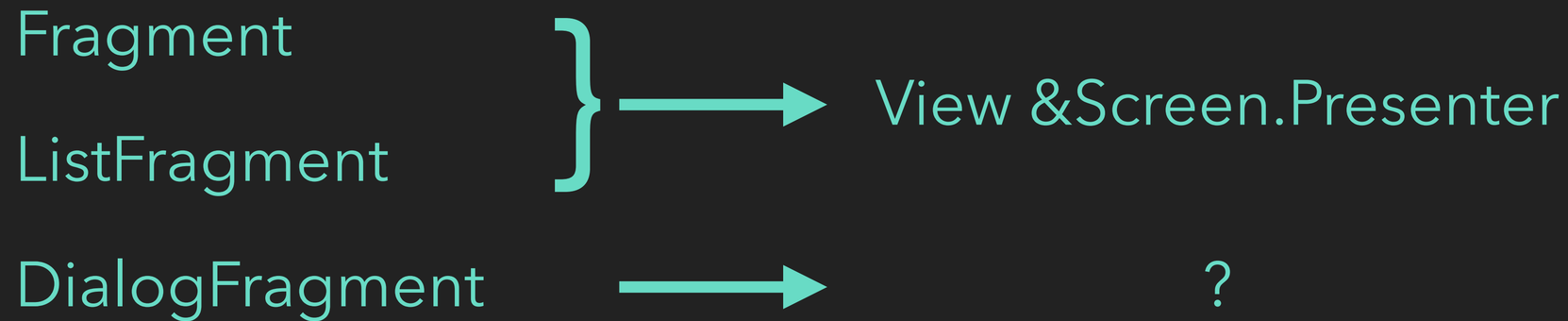
October 08, 2014

Mortar

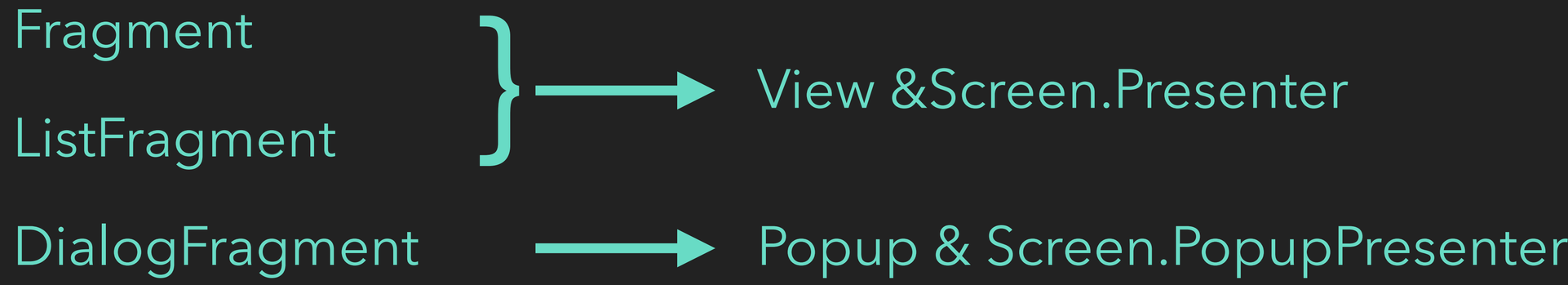
- Mortar
  - Enumerates app’s UI states and navigates between them
  - Alternative but good old practice for building UI parts

  - Based on “View” system on Android and…
  - Replaces fussy “Fragment” system

Fragments vs. Presenters



Fragments vs. Presenters



Popup

```
public class ConfirmationPopup implements Popup<Info, Boolean> {
  private final Context mContext;
  private AlertDialog mDialog;

  @Override
  public void show(Info info, boolean withFlourish, PopupPresenter<Info, Boolean> presenter) {
    if (mDialog != null) throw new IllegalStateException(“already shown!”);

    mDialog = new AlertDialog.Builder(getContext())
        .setMessage(info.getMessage())
        .setPositiveButton(info.getPositive(),
            (dialog, which) -> presenter.onDismissed(true))
        .setNegativeButton(info.getNegative(),
            (dialog, which) -> presenter.onDismissed(false))
        .show();
  }
}
```

Popup

```
public class ConfirmationPopup implements Popup<Info, Boolean> {
  private final Context mContext;
  private AlertDialog mDialog;

  @Override
  public void show(Info info, boolean withFlourish, PopupPresenter<Info, Boolean> presenter) {
    if (mDialog != null) throw new IllegalStateException("already shown!");

    mDialog = new AlertDialog.Builder(getContext())
        .setMessage(info.getMessage())
        .setPositiveButton(info.getPositive(),
            (dialog, which) -> presenter.onDismissed(true))
        .setNegativeButton(info.getNegative(),
            (dialog, which) -> presenter.onDismissed(false))
        .show();
  }
}
```

Popup

```
public class ConfirmationPopup implements Popup<Info, Boolean> {
  private final Context mContext;
  private AlertDialog mDialog;

  @Override
  public void show(Info info, boolean withFlourish, PopupPresenter<Info, Boolean> presenter) {
    if (mDialog != null) throw new IllegalStateException("already shown!");

    mDialog = new AlertDialog.Builder(getContext())
        .setMessage(info.getMessage())
        .setPositiveButton(info.getPositive(),
            (dialog, which) -> presenter.onDismissed(true))
        .setNegativeButton(info.getNegative(),
            (dialog, which) -> presenter.onDismissed(false))
        .show();
  }
}
```

Popup

```
public class ConfirmationPopup implements Popup<Info, Boolean> {
  private final Context mContext;
  private AlertDialog mDialog;

  @Override
  public void show(Info info, boolean withFlourish, PopupPresenter<Info, Boolean> presenter) {
    if (mDialog != null) throw new IllegalStateException("already shown!");

    mDialog = new AlertDialog.Builder(getContext())
        .setMessage(info.getMessage())
        .setPositiveButton(info.getPositive(),
            (dialog, which) -> presenter.onDismissed(true))
        .setNegativeButton(info.getNegative(),
            (dialog, which) -> presenter.onDismissed(false))
        .show();
  }
}
```

Popup

```
public class ConfirmationPopup implements Popup<Info, Boolean> {
  private final Context mContext;
  private AlertDialog mDialog;

  @Override
  public void show(Info info, boolean withFlourish, PopupPresenter<Info, Boolean> presenter) {
    if (mDialog != null) throw new IllegalStateException("already shown!");

    mDialog = new AlertDialog.Builder(getContext())
        .setMessage(info.getMessage())
        .setPositiveButton(info.getPositive(),
            (dialog, which) -> presenter.onDismissed(true))
        .setNegativeButton(info.getNegative(),
            (dialog, which) -> presenter.onDismissed(false))
        .show();
  }
}
```

withFlourish

{

true if Popup is explicitly shown/dismissed through Presenter

false otherwise

Popup

```
public class ConfirmationPopup implements Popup<Info, Boolean> {
  private final Context mContext;
  private AlertDialog mDialog;

  @Override
  public boolean isShowing() { return mDialog != null; }

  @Override
  public Context getContext() { return mContext; }

  @Override
  public void dismiss(boolean withFlourish) {
    mDialog.dismiss();
    mDialog = null;
  }
}
```

PopupPresenter

```
public class SomeView extends FrameLayout {
  // do not inject popup presenter and popup here.
  private PopupPresenter<Info, Boolean> mPopupPresenter = new PopupPresenter<>() {
    @Override
    public void onPopupResult(Boolean result) { } // result == user's choice
  }
  private Popup mPopup;

  @Override
  public void onAttachedToWindow() {
    super.onAttachedToWindow();
    mPopup = new ConfirmationPopup(getContext());
    mPopupPresenter.takeView(mPopup);
  }
}
```

PopupPresenter

```
public class SomeView extends FrameLayout {
  // do not inject popup presenter and popup here.
  private PopupPresenter<Info, Boolean> mPopupPresenter = new PopupPresenter<>() {
    @Override
    public void onPopupResult(Boolean result) { } // result == user's choice
  }
  private Popup mPopup;

  @Override
  public void onAttachedToWindow() {
    super.onAttachedToWindow();
    mPopup = new ConfirmationPopup(getContext());
    mPopupPresenter.takeView(mPopup);
  }
}
```

PopupPresenter

```
public class SomeView extends FrameLayout {
  // do not inject popup presenter and popup here.
  private PopupPresenter<Info, Boolean> mPopupPresenter = new PopupPresenter<>() {
    @Override
    public void onPopupResult(Boolean result) { } // result == user's choice
  }
  private Popup mPopup;

  @Override
  public void onAttachedToWindow() {
    super.onAttachedToWindow();
    mPopup = new ConfirmationPopup(getContext());
    mPopupPresenter.takeView(mPopup);
  }
}
```

PopupPresenter

```
public class SomeView extends FrameLayout {
  // do not inject popup presenter and popup here.
  private PopupPresenter<Info, Boolean> mPopupPresenter = new PopupPresenter<>() {
    @Override
    public void onPopupResult(Boolean result) { } // result == user's choice
  }
  private Popup mPopup;

  @Override
  public void onDetachedFromWindow() {
    mPopupPresenter.dropView(mPopup);
    super.onDetachedFromWindow();
  }
}
```

PopupPresenter

```
public class SomeView extends FrameLayout {
  // do not inject popup presenter and popup here.
  private PopupPresenter<Info, Boolean> mPopupPresenter = new PopupPresenter<>() {
    @Override
    public void onPopupResult(Boolean result) { } // result == user's choice
  }
  private Popup mPopup;

  @Override
  public void onDetachedFromWindow() {
    mPopupPresenter.dropView(mPopup);
    super.onDetachedFromWindow();
  }
}
```

PopupPresenter

```
public class SomeView extends FrameLayout {
  // do not inject popup presenter and popup here.
  private PopupPresenter<Info, Boolean> mPopupPresenter = new PopupPresenter<>() {
    @Override
    public void onPopupResult(Boolean result) { } // result == user’s choice
  }
  private Popup mPopup;

  @OnClick(R.id.submit)
  public void onSubmit() {
    mPopupPresenter.show(Info.create(“Is it ok?”, “ok”, “cancel”));
  }
}
```

PopupPresenter

```
public class SomeView extends FrameLayout {
  // do not inject popup presenter and popup here.
  private PopupPresenter<Info, Boolean> mPopupPresenter = new PopupPresenter<>() {
    @Override
    public void onPopupResult(Boolean result) { } // result == user's choice
  }
  private Popup mPopup;

  @OnClick(R.id.submit)
  public void onSubmit() {
    mPopupPresenter.show(Info.create("Is it ok?", "ok", "cancel"));
  }
}
```

Popup & PopupPresenter

- PopupPresenter<D extends Parcelable, R>
  - Alternative to FragmentManager
  - Handles user’s choice callback at “onPopupResult”
  - Type param D must implement “equals” and “hashCode”, and may not be null(otherwise Popup will not be shown)

Popup & PopupPresenter

- Popup<D extends Parcelable, R>
  - Alternative to DialogFragment
  - Receives arguments without Bundle (like DialogFragment)
  - Presenter will publish user’s choice data for you

“Don't call us, we'll call you”

–Hollywood Principle

Popup view on Mortar

@KeithYokoma - Drivemode, Inc.

potatotips #21

Join our team? Contact me!