BLoC

@kgmyshin

Flutter Meetup Tokyo #2

自己紹介

- kgmyshin / 釘宮(くぎみや)
- Androidエンジニア
- DMM.comラボ CTO室所属
  - 2018年3月から

agenda

agenda

- BLoCとは
- BLoCのガイドライン
- BLoCとUI
- BLoCの例
- Flutterでの使い方
- まとめ

BLoCとは

BLoCとは

自分が知ったのは Google IO 2018での下記のセッション

Build reactive mobile apps with Flutter

https://www.youtube.com/watch?v=RS36gBEp8OI

BLoCとは

実際は2018年1月にDart Conf 2018の

下記のセッションで発表されている

Flutter / AngularDart – Code sharing, better together

https://www.youtube.com/watch?v=PLHln7wHgPE

BLoCとは

一言で言うと

ビジネスロジックを

UIやプラットフォームから

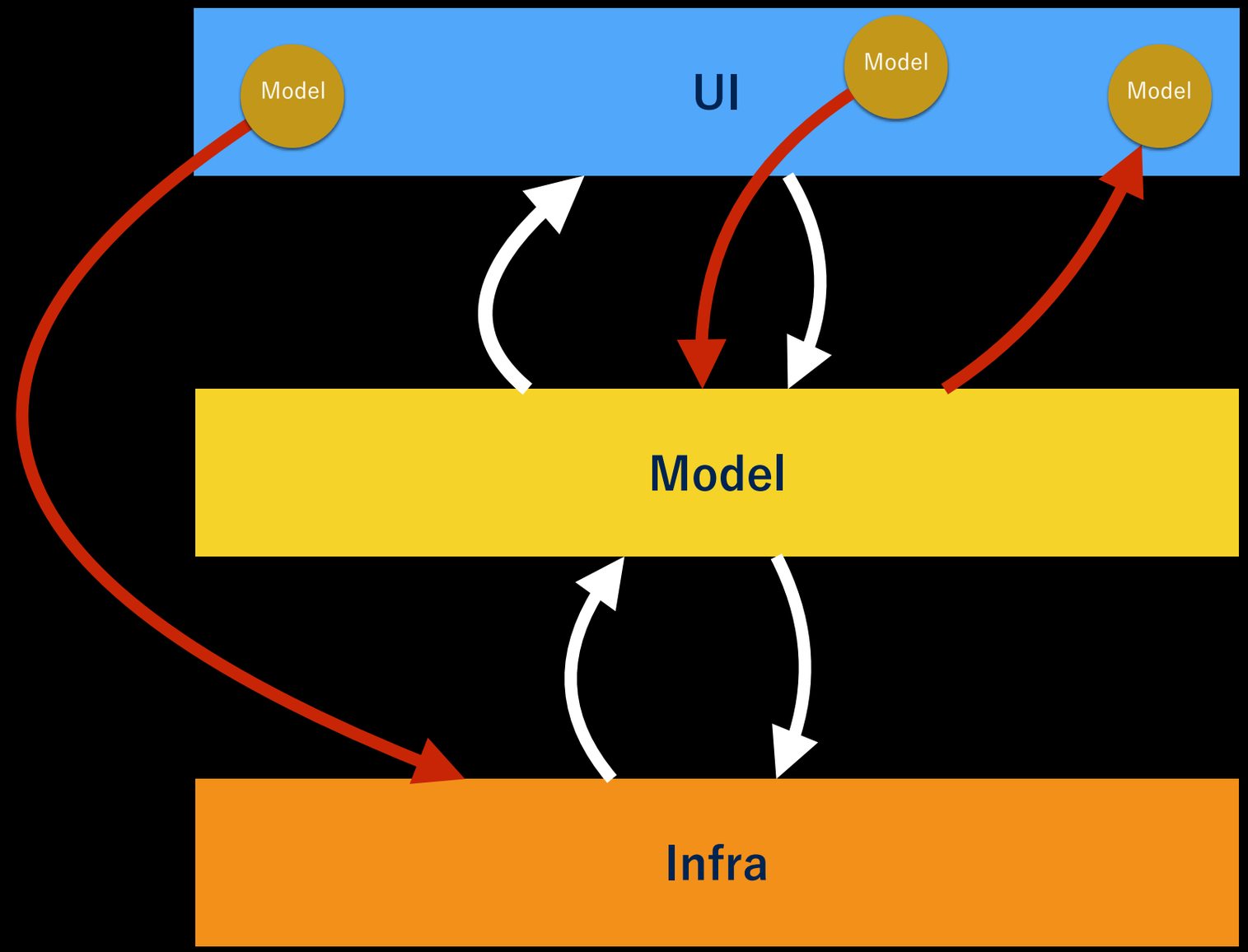
分離するための

設計パターンの一つ

である

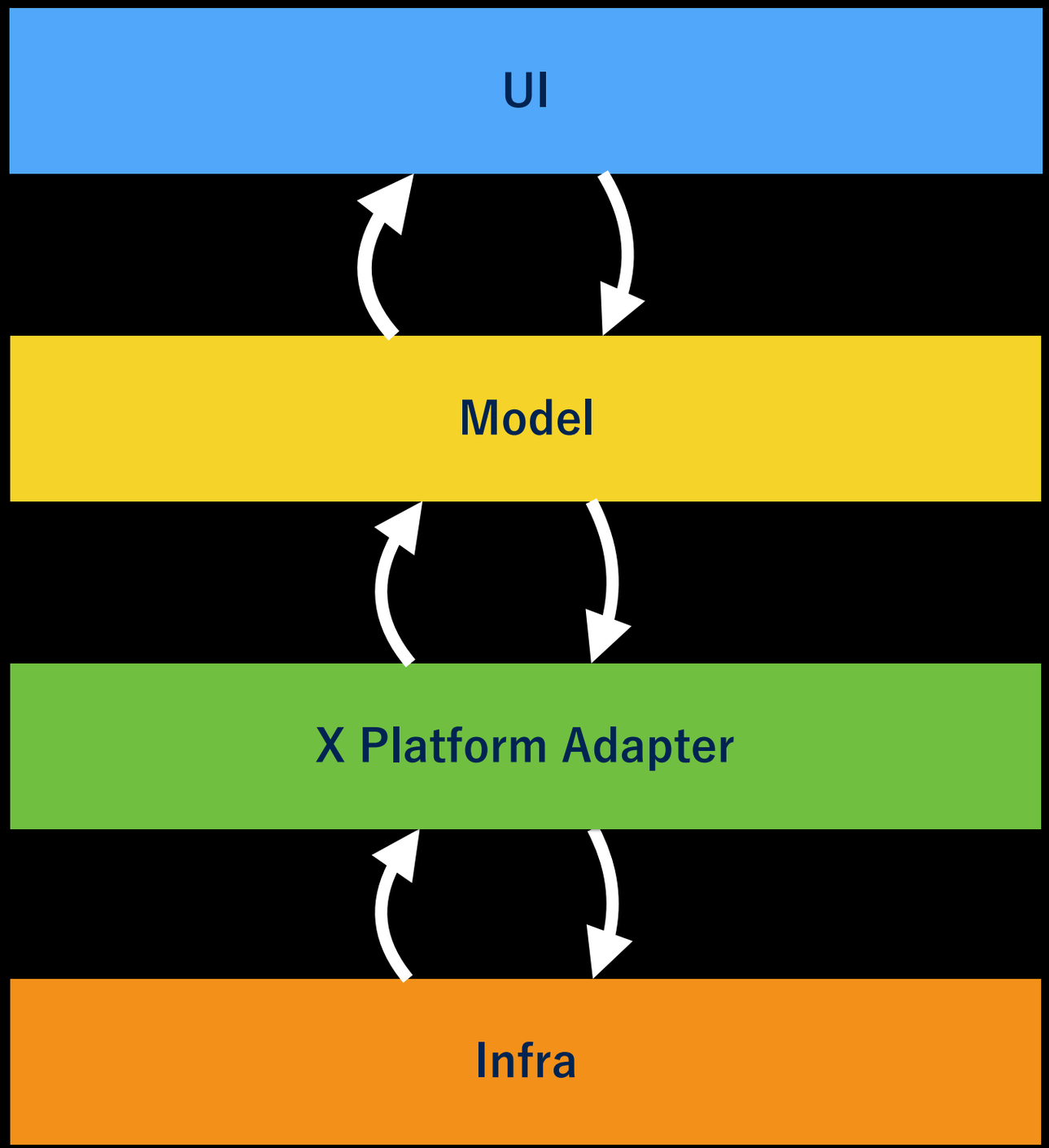
BLoCのモチベーション

こういう状態を



BLoCのモチベーション

こうしたい !!



BLoCのガイドライン

BLoCのガイドライン

1. inputs and outputs are simple Streams/Sinks only
2. Dependencies must be injectable and platform agnostic
3. No platform branching allowed
4. Implementation can be whatever you want if you follow the previous rules.
    A. But may I suggest reactive programming?

BLoCのガイドライン

1. input は Sink、 outputは Streamで実装する
2. 依存オブジェクトは外から注入可能で、プラットフォームに依存しないものであること
3. プラットフォームの分岐はダメ
4. 1~3を守ってれば、実装はなんでもいいよ
    1. ただし、Reactive Programmingで組むのがおすすめだよ

BLoCとUIの繋ぎ

BLoCとUIの繋ぎ

1. Each "complex enough" component has a corresponding BLoC
2. Components should send inputs "as is”
3. Components should show outputs as close as possible to "as is”
4. All branching should be based on simple BLoC boolean outputs

BLoCのガイドライン

1. それぞれの”十分に複雑な” コンポーネントにはBLoCが一つある（”十分に複雑な”の度合いは各自で判断）
2. inputに送るデータは "as is"で
3. outputに流すストリームもできるだけ “as is”で
4. すべての分岐はBLoCによるBooleanなアウトプットに基づくべきである

BLoCの例

BLoCの例

```
class CartBLoC {
  Sink<Product> addition;
  Stream<int> itemCount;
  Stream<String> totalCost;
  Stream<List<CartItem>> items;
}
```


BLoCの例

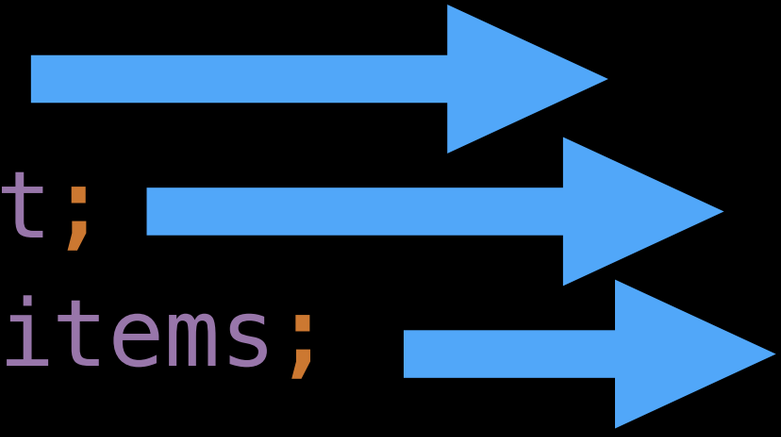
Productが追加されると

```
class CartBLoC {
  Sink<Product> addition;
  Stream<int> itemCount;
  Stream<String> totalCost;
  Stream<List<CartItem>> items;
}
```

BLoCの例

それぞれのStreamから結果が流れる

```
class CartBLoC {
  Sink<Product> addition;
  Stream<int> itemCount;
  Stream<String> totalCost;
  Stream<List<CartItem>> items;
}
```



Flutterでの使い方

inputへの繋げ方

```
TextField(
  onChanged: bloc.sink.add,
  decoration: InputDecoration(
    border: OutlineInputBorder(),
    hintText: “hint"),
)
```


inputへの繋げ方

イベントとsink.addをバインドするだけ

```
TextField(
  onChanged: bloc.sink.add,
  decoration: InputDecoration(
      border: OutlineInputBorder(),
      hintText: “hint"),
)
```

outputをUIにつなげる方法

```
StreamBuilder<String>(
  stream: bloc.stream,
  builder: (context, snapshot) =>
      Container(
        child: Text(snapshot?.data ?? ""),
      ),
)
```


outputをUIにつなげる方法

StreamBuilder使うだけ

```
StreamBuilder<String>(
  stream: bloc.stream,
  builder: (context, snapshot) =>
      Container(
        child: Text(snapshot?.data ?? ""),
      ),
)
```

まとめ

まとめ

- BLoCはUIとModelとInfraを分けることと、クロスプラットフォームのコードシェアをモチベーションに生まれた
- 守るべきガイドラインがある
- Flutterでは
  - inputを送るところでは sink.add をバインドする
  - outputをsubscribeするにはStreamBuilderを使う