

Flutter e Dart - Le basi

Flutter - Gestione dello stato

Luigi Durso

luigi.durso@si2001.it



SI2001

6 luglio 2022



Sommario

- 1** Lezione precedente
- 2** Problema

- 3** BLoC
- 4** Esercitazione
- 5** Fine



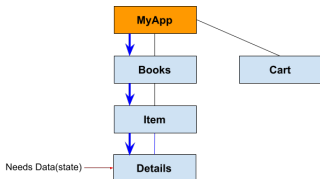
Un po' di codice



Analizziamo l'elaborato precedente!



Stato condiviso



Qual'è il problema?

- Dati condivisi tra più livelli dell'albero dei widgets
- Richiesta di passaggio di parametri anche in widget che non ne hanno bisogno
- Re-build eseguita su tutto il percorso di dipendenze



Idee di soluzione

Un possibile approccio di soluzione

- Pensare allo stato disaccoppiato dai widgets
- Interazione tra widget e stato attraverso eventi e funzioni
- Ogni widget si sottoscrive ai cambiamenti della parte di stato a cui è interessato



Soluzioni

Package	Likes	Pub Points	Popularity	Flutter Favorite
	2022.02.04			
provider	5,972	130	100%	YES
bloc	4,937* (3,468 + 1,469)	130	100%	YES
get	7,861	120	100%	-
riverpod	1,353	130	97%	-
get_it	1,873	130	100%	-
redux	613** (338 + 275)	120	97%	YES
mobx	1,242*** (452 + 790)	130	98%	YES



BLoC

Bloc

(**B**usiness **L**ogic **C**omponent)



Perché usare BLoC

Bloc segue tre principi fondamentali:

- **Semplicità**: Facile da imparare, chiunque può approcciarsi facilmente a questa tecnologia.
- **Potenza**: Attraverso la creazione di piccoli componenti si può creare un'applicativo di grande complessità.
- **Testabilità**: Ogni aspetto dell'applicativo è facilmente testabile.



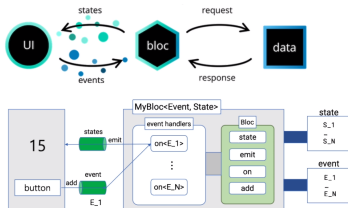
Approcci diversi

Possiamo usare approcci diversi in base alle nostre esigenze:

- Approccio event-driven con **BLoC**
- Approccio basato su funzioni con **Cubit**



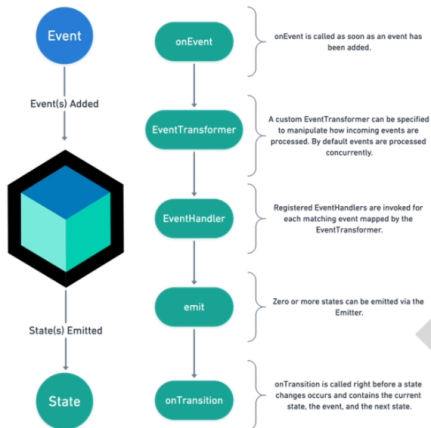
Struttura di BLoC



- Lettura degli eventi scatenati dalla UI
- Ricezione dell'evento e gestione mediante il giusto event handler
- Elaborazione dell'evento (Es. Interazione con Repository)
- Rendering del nuovo stato nella UI



Cosa possiamo osservare



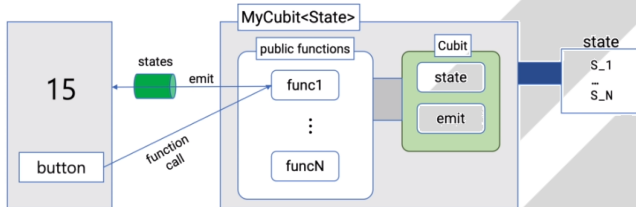
```
print('$event');
IncrementCounterEvent
```

```
print('$transition');

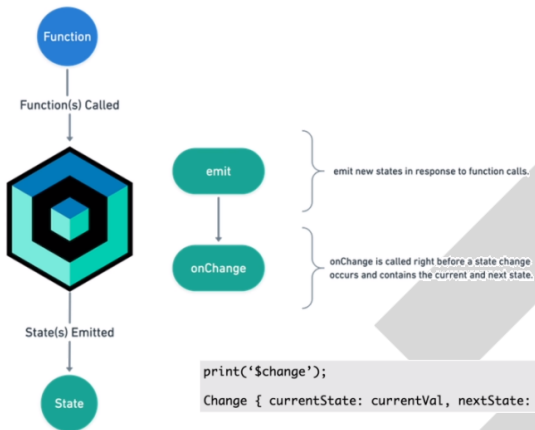
Transition {
  currentState: currentVal,
  event: IncrementCounterEvent
  nextState: nextVal
}
```



Struttura di Cubit



Cosa possiamo osservare



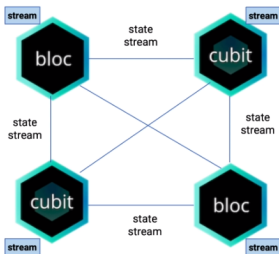
Cubit vs BLoC

	Cubit	Bloc
Simplicity	<ul style="list-style-type: none"> • states • functions 	<ul style="list-style-type: none"> • states • events • event handlers
Traceability		<ul style="list-style-type: none"> • Transition (onTransition)
Advanced Event Transformations		<ul style="list-style-type: none"> • EventTransformer

- Una struttura ad eventi necessità di complessità aggiuntiva.
- Non sempre si ha bisogno di una gestione complessa.
- Si può sempre optare per soluzioni ibride, **BLoC più Cubit**



Cubits/BLoCs possono comunicare tra loro



```
myCubit.stream.listen((MyCubitState state) {  
  // ...  
})
```

```
myBloc.stream.listen((MyBlocState state) {  
  // ...  
})
```



Migliorare la precedente esercitazione

Introdurre una gestione di memoria con Bloc/Cubit



Grazie per l'attenzione!

