**Flutter BLoC**

BLoC هو اختصار لـ "Business Logic Component"، وهو نمط معماري يستخدم في تطوير تطبيقات Flutter لإدارة الحالة والمنطق التجاري (الـ Business Logic) بشكل فعّال. يعتبر BLoC منهجًا لتفصيل الجزئيات المتعلقة بالحالة والمنطق التجاري عن واجهة المستخدم.

هناك عدة مكونات رئيسية في BLoC:

1. الـ StreamController:

يُستخدم لإنشاء وإدارة الـ Streams، وهي مصادر لبيانات مستمرة. يتيح ذلك لتطبيقك تدفق البيانات بشكل فعال.

2. الـ Bloc:

هو الجزء الرئيسي الذي يحتوي على المنطق التجاري. يستخدم لإدارة حالة التطبيق وتدفق البيانات بين الـ UI والـ DataSource.

3. الـ Event:

تُستخدم لنقل المحدثات (events) من واجهة المستخدم إلى BLoC. على سبيل المثال، عند الضغط على زر في التطبيق.

4. الـ State:

يُستخدم لتمثيل حالة التطبيق. عندما يحدث حدث، يتم إصدار حالة جديدة تتسبب في تحديث واجهة المستخدم

استخدام BLoC في تطبيق Flutter، يجب :

1. تعريف الـ Events الذين سيتم إرسالهم من واجهة المستخدم

2. تعريف الـ BLoC الذي يحتوي على المنطق التجاري ويتعامل مع الـ Events

3. تحديد كيفية تحويل الـ Events إلى الـ States وتحديث واجهة المستخدم

يشمل العديد من الحلقات حول المفاهيم الأساسية وإعداد المشروع والنموذج وDio و Repo و Cubit و BlocProvider و BlocBuilder وشاشة الرئيسية والبحث وفلترة القائمة وCustomScrollView و Slivers وAnimated Text وFlutter Offline

**Cubit**

Cubit هو مفهوم آخر لإدارة الحالة في تطبيقات Flutter، وهو نوع من أنواع إدارة الحالة المشابهة لـ BLoC ولكنه يقلل من الكود المطلوب لتنفيذه ويسهل الفهم. Cubit هو اختصار لـ "Cubit"، وهو نوع محدد من أنواع BLoC.

الفكرة الرئيسية وراء Cubit هي تبسيط مفهوم إدارة الحالة وتقليل التعقيد. يستخدم Cubit نهجًا أكثر تركيزًا على التيارات (Streams) ويقلل من الحاجة إلى كتابة الكثير من الشيفرة الزائدة.

الفرق الرئيسي بين BLoC وCubit يكمن في الكمية من الشيفرة المطلوبة لتنفيذ كل منهما. Cubit يقدم نهجًا أكثر بساطة للمشاريع الصغيرة إلى المتوسطة، بينما BLoC يوفر مزيدًا من التحكم والمرونة للمشاريع الأكبر والأكثر تعقيدًا.

**BlocProvider**

`BlocProvider` هو وسيلة لتوفير إدارة الحالة في تطبيقات Flutter باستخدام مكتبة Bloc. يتيح `BlocProvider` لك توفير مثيل من `Bloc` أو `Cubit` في شجرة واجهة المستخدم بطريقة فعّالة وسهلة.

`BlocProvider` يأتي مع مكتبة `flutter\_bloc` ويقوم بتيسير إنشاء وتوفير مثيلات لـ `Bloc` أو `Cubit` لجميع الأشجار الفرعية. هذا يسمح بإمكانية الوصول إلى نفس المثيل من `Bloc` أو `Cubit` في جميع أنحاء التطبيق بسهولة.

**BlocBuilder**

`BlocBuilder` هو عنصر واجهة مستخدم في مكتبة `flutter\_bloc` يستخدم لبناء جزء من واجهة المستخدم بناءً على حالة محددة من `Bloc` أو `Cubit`. يقوم `BlocBuilder` بتحديث الجزء المبني له عندما يتغير حالة `Bloc` أو `Cubit` المقترن به.

الهدف الرئيسي لاستخدام `BlocBuilder` هو تحديث واجهة المستخدم بناءً على التغييرات في حالة التطبيق التي يديرها `Bloc` أو `Cubit`. يساعد ذلك في الحفاظ على تناغم وتحديث الواجهة بشكل فعّال.

الاستخدام الأساسي لـ `BlocBuilder` يكون كما يلي:

1. يتم تحديثه بناءً على حالة معينة من `Bloc` أو `Cubit`.

2. يأخذ وظيفة `builder` التي يتم استدعاؤها في كل مرة يتم فيها تحديث الحالة.

**Bloclistener**

`BlocListener` هو عنصر واجهة مستخدم آخر من مكتبة `flutter\_bloc` يستخدم للاستماع إلى التغييرات في حالة محددة من `Bloc` أو `Cubit` وتنفيذ إجراءات معينة بناءً على هذه التغييرات. يشبه `BlocListener` إلى حد كبير `BlocBuilder`، ولكن يختلف فيما إذا كان يقوم بإعادة بناء الواجهة أم لا.

الفائدة الرئيسية من استخدام `BlocListener` هي أنه يسمح لك بتنفيذ إجراءات أو تحديثات على الواجهة بناءً على التغييرات في حالة `Bloc` أو `Cubit` بدون إعادة بناء الجزء نفسه من الواجهة.

الاستخدام الأساسي لـ `BlocListener` يكون كالتالي:

1. يتم تحديثه عندما يتغير حالة معينة من `Bloc` أو `Cubit`.

2. يأخذ وظيفة `listener` التي يتم استدعاؤها في كل مرة يتم فيها تحديث الحالة.

**BlocConsumer**

`BlocConsumer` هو عنصر واجهة مستخدم من مكتبة `flutter\_bloc` يقوم بتوفير وسيلة للاستماع والتفاعل مع التغييرات في حالة محددة من `Bloc` أو `Cubit`. يقوم `BlocConsumer` بإجراء العديد من الوظائف، حيث يمكن استخدامه لبناء جزء من واجهة المستخدم، ولكنه يقوم أيضًا بتنفيذ دوال التحديث والاستماع.

الاستخدام الأساسي لـ `BlocConsumer` يكون كما يلي:

1. يتم تحديثه عندما يتغير حالة معينة من `Bloc` أو `Cubit`.

2. يأخذ وظيفة `builder` التي تقوم ببناء جزء من واجهة المستخدم.

3. يأخذ وظيفة `listener` التي تستمع إلى التغييرات في الحالة وتقوم بتنفيذ الإجراءات المحددة.