flutter

User interface

Introduction to widgets

Changing widgets in response to input

지금까지, 이 페이지는 stateless 위젯만 다뤘다. Stateless 위젯은 그들의 부모 위젯에게 arguments를 받고 final 멤버 변수에 저장한다. 한 위젯이 build()를 요청했을 때, 위젯은 만든 위젯에 대한 새로운 arguments를 불러내기 위해 저장된 값을 사용한다.

더 복잡한 경험을 위하여–예를 들어, 사용자의 입력에 대한 더 흥미로운 방법들에 반응하기 위해-어플리케이션들은 전형적으로 몇 상태를 가진다. 플러터는 이 아이디어를 실현시키기 위해 StatefulWidgets을 사용한다. StatefulWidgets은 상태를 가지기 위해 사용되는 State 위젯을 생성하는 방법을 아는 특별한 위젯이다. 이미 언급된 RaisedButton을 사용하는 이 기본 예제를 봐라.

당신은 아마 왜 StatefulWidget과 State가 분리되어있는지 궁금할 것이다. 플러터에서 이 두 객체의 종류는 다른 생명 주기를 가진다. Widgets은 현재 상태에서 어플리케이션의 프리젠테이션을 건설하는 것에 사용되는 일시적인 객체다. 반면에, State 객체는 build()의 호출들 사이에서 그들이 정보를 기억하게 하도록 하면서 지속한다.

위의 예시는 사용자 입력을 받아들이고 build() 메소드에서 결과를 직접적으로 사용한다. 예시는 더 복잡한 어플리케이션에서, 위젯 계층의 다른 부분들은 다른 문제에 책임이 있을 수 있다. 예를 들어, 다른 위젯은 전체적인 프리젠테이션을 바꾸기 위한 정보를 사용할 수 있으면서 한 위젯은 복합적인 사용자 인터페이스를 날짜나 장소와 같은 특정한 정보를 모으는 목적과 함께 제공할 수 있다.

플러터에서, 현재 상태가 프리젠테이션을 하는 stateless 위젯에서 아래로 흐르는 반면, 변경 알림은 콜백을 통해 위젯 계층에서 “위로” 흐른다. 이 흐름의 방향을 다시 조정하는 보통의 부모가 State이다. 다음으 약간 더 복잡한 예시는 실전에서 이것이 어떻게 작동하는 지 보여준다.

counter displaying (CounterDisplay)와 counter changing (CounterIncrementor)의 문제를 깔끔하게 분리하면서 새로운 두 개의 stateless 위젯의 생성을 보아라. 비록 순 결과는 이전의 예시와 같을지라도, 책임의 분리는 부모의 단순성을 유지하면서 각각의 위젯에 캡슐화되어 더 좋은 복잡성을 허용한다.

Bringing it all together