07

상태 관리하기

상태관리

위젯의 상태 관리하기

위젯의 상태 관리 기본 개념

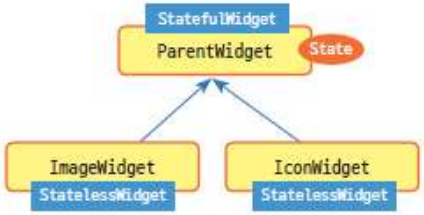
- 위젯의 상태state는 데이터
- 상태란 위젯에서 다양한 이유로 변경되는 데이터를 의미

- •위젯 자체의 상태를 이용
- •상위 위젯의 상태를 이용
- •위젯 자체의 상태와 상위 위젯의 상태를 함께 이용

위젯의 상태 관리하기

위젯의 상태 관리 기본 개념

- 상위 위젯의 상태를 이용



위젯의 상태 관리하기

• 상위 위젯 구현하기

```
class ParentWidgetState extends State<ParentWidget> {
  bool favorited = false;
  int favoriteCount = 10;

  void toggleFavorite() {
    ... (생략) ...
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
            ... (생략) ...
            body:Column(
              children: [
                IconWidget(favorited: favorited, onChanged: toggleFavorite),
                ContentWidget(favoriteCount: favoriteCount)
              ],
            )
        )
    );
  }
}
```

위젯의 상태 관리하기

• ContentWidget 구현하기

```
class ContentWidget extends StatelessWidget {
  final int favoriteCount;

  ContentWidget({required this.favoriteCount});

  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        Container(
          child: Text('I am ContentWidget, favoriteCount : $favoriteCount'),
        ),
      ],
    );
  }
}
```

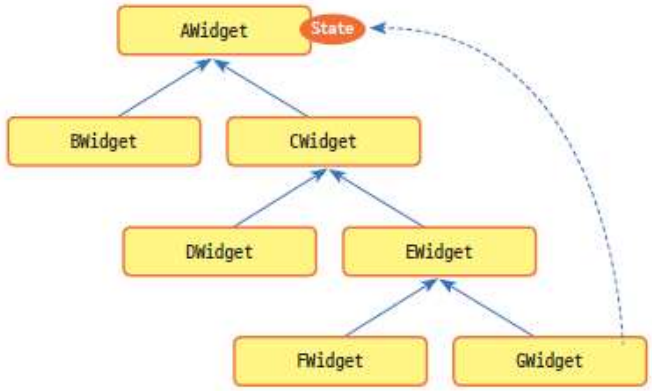
• IconWidget 구현하기

```
class IconWidget extends StatelessWidget {
  final bool favorited;
  final Function onChanged;

  IconWidget({this.favorited: false, required this.onChanged});

  void _handleTap() {
    onChanged();
  }
  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        Container(
          child: Text('I am IconWidget, '),
        ),
        IconButton(
          icon: (favorited ? Icon(Icons.favorite) : Icon(Icons.favorite_border)),
          color: Colors.red,
          onPressed: _handleTap,
        ),
      ],
    );
  }
}
```

위젯의 상태 관리하기

조상 위젯의 상태 얻기

- 조상 위젯의 상태에 접근할 때는 생성자 매개변수로 전달받지 않고 findAncestorStateOfType() 함수로 조상 위젯의 상태 객체를 얻을 수 있습니다.



• findAncestorStateOfType() 함수 이용

```
ParentWidgetState? state = context.findAncestorStateOfType<ParentWidgetState>();
```

위젯의 상태 관리하기

하위 위젯의 상태 얻기

- 하위 위젯의 상태 객체를 얻으려면 하위 위젯을 생성할 때 키를 지정하고 이 키의 currentState라는 속성을 이용

• 하위 위젯의 상태 얻기

```
class ParentWidgetState extends State<ParentWidget> {
  GlobalKey<ChildWidgetState> childKey = GlobalKey();
  int childCount = 0;

  void getChildData() {
    ChildWidgetState? childState = childKey.currentState;
    setState(() {
      childCount = childState?.childCount ?? 0;
    });
  }

  @override
  Widget build(BuildContext context) {
    return ChildWidget(key: childKey);
  }
}

class ChildWidget extends StatefulWidget {
  ChildWidget({Key? key}):super(key: key);
  ... (생략) ...
}

class ChildWidgetState extends State<ChildWidget> {
  int childCount = 0;
  ... (생략) ...
}
```

공용 상태 관리 위젯 만들기

- 하위에서 공통으로 이용하는 상태를 가지는 InheritedWidget을 별도로 만들어서 제공

공용 상태 관리 위젯 만들기

- InheritedWidget을 이용하려면 InheritedWidget을 상속받아 클래스를 만들고, 그 클래스에 하위 위젯에서 이용할 상태 데이터와 관리 함수를 선언

• 공용 상태 관리 위젯 만들기

```
class MyInheritedWidget extends InheritedWidget {
  int count = 0;   // 하위 공유 데이터

  MyInheritedWidget(child) : super(child: child);   ❶
  increment() {    // 하위에서 호출할 함수
    count++;
  }

  @override
  bool updateShouldNotify(MyInheritedWidget oldWidget) => true;   ❷

  static MyInheritedWidget? of(BuildContext context) =>
      context.dependOnInheritedWidgetOfExactType<MyInheritedWidget>();   ❸
}
```

공용 상태 관리 위젯 만들기

- InheritedWidget의 하위 위젯이 InheritedWidget을 이용하려면 InheritedWidget에서 제공하는 of() 함수를 호출

• InheritedWidget 이용하기

```
int count = MyInheritedWidget.of(context)!.count;
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare