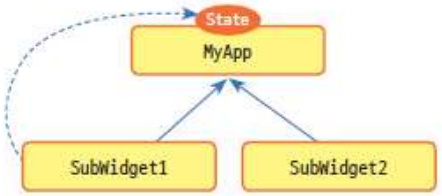
07

# Consumer와 Selector

상태관리

# 컨슈머와 셀렉터

## 프로바이더를 이용하는 위젯의 생명주기




- 상태를 이용하지 않는 위젯은 다시 빌드될까?
  - 상태를 이용하지 않는 위젯은 다시 빌드되지 않았습니다.

• 상태를 이용하지 않는 위젯

```
class SubWidget1 extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("SubWidget1 build...");
    var model1 = Provider.of<MyDataModel1>(context);
    ... (생략) ...
  }
}

class SubWidget2 extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("SubWidget2 build...");
    ... (생략) ...
  }
}
```

▶ 실행 결과

```
MyDataModel1 data1 changed....
SubWidget1 build...
```

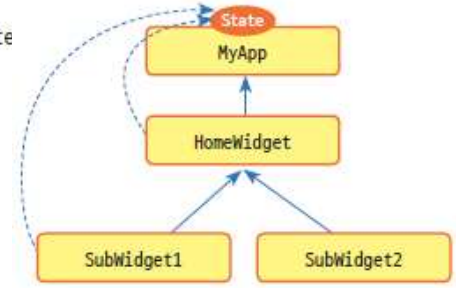
# 컨슈머와 셀렉터

## 프로바이더를 이용하는 위젯의 생명주기

- 상위 위젯에서 상태를 이용할 때 상태를 이용하지 않는 하위 위젯은 어떻게 될까?
  - 프로바이더를 이용하는 위젯은 불필요한 빌드가 일어날 가능성이 있습니다.

• 부모 위젯이 상태 참조

```
class HomeWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("HomeWidget build...");
    var model1 = Provider.of<MyDataModel1>(conte
    return Column(
      children: <Widget>[
        SubWidget1(),
```




```
        SubWidget2(),
      ],
    );
  }
}

class SubWidget1 extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("SubWidget1 build...");
    var model1 = Provider.of<MyDataModel1>(context);
    ... (생략) ...
  }
}

class SubWidget2 extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    print("SubWidget2 build...");
    ... (생략) ...
  }
}
```

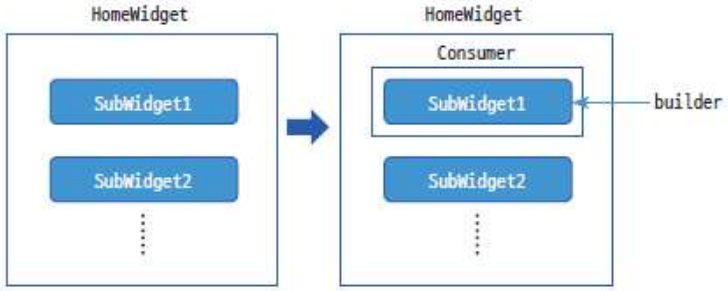
▶ 실행 결과

```
MyDataModel1 data2 changed....
HomeWidget build...
SubWidget1 build...
SubWidget2 build...
```

# 컨슈머와 셀렉터

## 특정 타입만 빌드하기 — Consumer

- Consumer를 이용하면 상탯값이 변경될 때 다시 빌드할 부분을 지정할 수 있습니다.



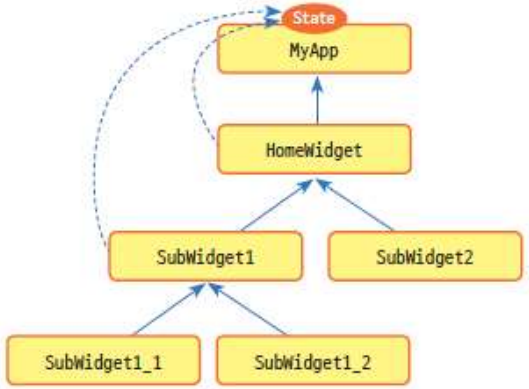

• 일부만 빌드하기

```
Column(
  children: <Widget>[
    Consumer<MyDataModel1>(
        builder: (context, model, child) {
          return SubWidget1(model);
        }
    ),
    SubWidget2(),
  ],
)
```

# 컨슈머와 셀렉터

## 특정 타입만 빌드하기 — Consumer

- 다시 빌드하지 않을 부분 지정하기
- Consumer의 builder에 추가하는 위젯의 하위 위젯 가운데 상탯값이 변경될 때 다시 빌드하지 않아야 하는 위젯이 있다면 Consumer의 child에 명시




• 다시 빌드하지 않을 부분 지정하기

```
Consumer<MyDataModel1>(
    builder: (context, model, child) {
      return SubWidget1(model, child);
    },
    child: SubWidget1_2(),
),
SubWidget2(),
... (생략) ...

class SubWidget1 extends StatelessWidget {
  MyDataModel1 model1;
  Widget? child;
  SubWidget1(this.model1, this.child);
  @override
  Widget build(BuildContext context) {
    print("SubWidget1 build...");
    return Column(
      children: [
        Text('I am SubWidget1... ${model1.data1}'),
        SubWidget1_1(model1),
        child!
      ],
    );
  }
}
```

# 컨슈머와 셀렉터

## 특정 타입만 빌드하기 — Consumer

- 여러 타입의 상태 데이터 이용하기
- 한 번에 여러 상태를 이용할 때, Consumer2~Consumer6을 사용

• 여러 타입의 상태 데이터 이용하기

```
Consumer2<MyDataModel1, MyDataModel2>(
    builder: (context, model1, model2, child) {
      ... (생략) ...
    },
),
```

# 컨슈머와 셀렉터

## 특정 타입만 빌드하기 — Consumer

- 특정 데이터만 빌드하기 — Selector
- Selector는 앞에서 살펴본 Consumer와 같은 목적으로 사용
- Selector는 상태의 타입뿐만 아니라 그 타입의 특정 데이터까지 지정하여 전달받거나 지정한 데이터가 변경될 때 다시 빌드

• 프로바이더에 등록할 상태 클래스

```
class MyDataModel with ChangeNotifier {
  int data1 = 0;
  int data2 = 10;
  ... (생략) ...
}
```

• 프로바이더에 등록

```
ChangeNotifierProvider<MyDataModel1>.value(value: MyDataModel()),
```

• Consumer로 이용하기

```
Consumer<MyDataModel>(
  builder: (context, model, child) {
    print('consumer, widget rebuild..');
    return Text('consumer, data1: ${model.data1}, data2: ${model.data2}');
  },
),
```

▶ 실행 결과

```
MyDataModel data1 changed....
consumer, widget rebuild..
MyDataModel data2 changed....
consumer, widget rebuild..
```

# 컨슈머와 셀렉터

## 특정 타입만 빌드하기 — Consumer

- Selector를 사용할 때는 제네릭 타입을 2개 지정
- 하나는 이용할 상태 객체 타입이며, 또 하나는 그 객체에서 이용할 데이터 타입

• 특정 데이터만 이용하기

```
Selector<MyDataModel, int>(
  builder: (context, data, child) {
    print('selector, widget rebuild..');
    return Text('selector, data:${data}');
  },
  selector: (context, model) => model.data2,
),
```

▶ 실행 결과

```
MyDataModel data1 changed....
MyDataModel data2 changed....
selector, widget rebuild..
```

# 감사합니다

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

윌리엄 셰익스피어
William Shakespeare