**Thành viên**

**Hà Thị Thùy Trang 0850080050**

**Nguyễn Thanh Tùng 0850080052**

**Phạm Lê Nhật Quang 0850080037**

**BÁO CÁO BUỔI 3 MÔN MOBILE**

SỬ DỤNG STATEFULLWIDGET

Bài làm

1. **Bài tập nhóm**

Vòng đời của STATEFULLWIDGET

1. **Giai đoạn khởi tạo (Initialization):**

Phương thức: StatefulWidget.createState() và State.initState()

Mô tả: Giai đoạn này xảy ra khi một StatefulWidget mới được tạo và nó sẽ khởi tạo lớp State tương ứng và gọi phương thức initState(). Ở đây, bạn có thể khởi tạo các giá trị ban đầu hoặc thực hiện các tác vụ chuẩn bị cho widget.

VD : class InitializationWidget extends StatefulWidget {

const InitializationWidget({Key? key}) : super(key: key);

@override

\_InitializationWidgetState createState() => \_InitializationWidgetState();

}

class \_InitializationWidgetState extends State<InitializationWidget> {

@override

void initState() {

super.initState();

print('Initialization: initState()');

}

@override

Widget build(BuildContext context) {

return Container();

}

}

1. **Giai đoạn xây dựng (Building):**

Phương thức: State.build()

Mô tả: Giai đoạn này xảy ra khi giao diện của widget cần được xây dựng. Phương thức build() là nơi bạn xây dựng cây widget của StatefulWidget. Widget này sẽ được tái xây dựng khi trạng thái của widget thay đổi.

VD : class BuildingWidget extends StatefulWidget {

const BuildingWidget({Key? key}) : super(key: key);

@override

\_BuildingWidgetState createState() => \_BuildingWidgetState();

}

class \_BuildingWidgetState extends State<BuildingWidget> {

@override

Widget build(BuildContext context) {

print('Building: build()');

return Container();

}

}

1. **Giai đoạn cập nhật (Updating):**

Phương thức: State.didChangeDependencies(), State.setState(), State.didUpdateWidget()

Mô tả: Giai đoạn này xảy ra khi trạng thái của widget thay đổi. Flutter sẽ gọi didChangeDependencies() nếu các đối tượng phụ thuộc đã thay đổi (ví dụ: khi mở hoặc đóng màn hình). Nếu bạn muốn cập nhật trạng thái và làm mới giao diện, hãy sử dụng setState(). Sau khi widget được tái xây dựng, didUpdateWidget() được gọi, cho phép bạn xử lý các thay đổi từ widget trước sang widget sau.

VD : class UpdatingWidget extends StatefulWidget {

const UpdatingWidget({Key? key}) : super(key: key);

@override

\_UpdatingWidgetState createState() => \_UpdatingWidgetState();

}

class \_UpdatingWidgetState extends State<UpdatingWidget> {

int \_counter = 0;

@override

void didChangeDependencies() {

super.didChangeDependencies();

print('Updating: didChangeDependencies()');

}

void \_incrementCounter() {

setState(() {

\_counter++;

});

}

@override

Widget build(BuildContext context) {

print('Updating: build()');

return Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text('Counter: $\_counter'),

ElevatedButton(

onPressed: \_incrementCounter,

child: Text('Increment'),

),

],

);

}

}

1. Giai đoạn huỷ (Disposal):

Phương **thức: State.dispose()**

Mô tả: Giai đoạn này xảy ra khi widget bị huỷ và không còn tồn tại trong cây widget. Phương thức dispose() được gọi và bạn có thể giải phóng tài nguyên, hủy các đăng ký hoặc dọn dẹp các tác vụ không cần thiết.

VD : class DisposalWidget extends StatefulWidget {

const DisposalWidget({Key? key}) : super(key: key);

@override

\_DisposalWidgetState createState() => \_DisposalWidgetState();

}

class \_DisposalWidgetState extends State<DisposalWidget> {

@override

void dispose() {

super.dispose();

print('Disposal: dispose()');

}

@override

Widget build(BuildContext context) {

return Container();

}

}

1. **Bài tập cá nhân**





