**RIVERPOD**

Riverpod is a reactive caching and data-binding framework that was born as an evolution of the Provider package. According to the official documentation: Riverpod is a complete rewrite of the Provider package to make improvements that would be otherwise impossible.

There are a few key differences between the two libraries: Syntax: Riverpod uses a more concise and easy-to-read syntax compared to Provider. This can make it easier to understand and work with, especially for developers who are new to dependency injection or state management.

Notes\_module.dart

**class** Notes {  
 **final** String **title**;  
 **final** String **content**;  
 Notes({**required this**.**title**, **required this**.**content**});  
}

notes\_notifier.dart

**import 'package:flutter\_riverpod/flutter\_riverpod.dart'**;  
**import 'notes\_model.dart'**;  
  
**class** NotifierState **extends** StateNotifier<List<Notes>> {  
 NotifierState() : **super**([]);  
  
 **void** addNote(Notes note) {  
 **state** = [...**state**, note];  
 }  
 **void** removeNote(Notes note) {  
 **state** = **state**.where((\_note) => \_note != note).toList();  
 }  
}

//The spread operator is three dots, we can use this operator **to add the list values to another list** and if we want to do a null-aware check we can add a question mark to the expression which you want to add to our list.

main.dart

**import 'package:flutter/material.dart'**;  
**import 'package:flutter\_riverpod/flutter\_riverpod.dart'**;  
**import 'notes\_model.dart'**;  
**import 'notes\_notifier.dart'**;  
  
**void** main() => runApp(**const** ProviderScope(child: MyApp()));  
  
**class** MyApp **extends** StatelessWidget {  
 **const** MyApp({Key? key}) : **super**(key: key);  
  
 @override  
 Widget build(BuildContext context) {  
 **return const** MaterialApp(  
 home: Home(),  
 );  
 }  
}  
**final** noteProvider = StateNotifierProvider<NotifierState, List<Notes>>((ref) => NotifierState());  
  
**class** Home **extends** ConsumerWidget {  
 **const** Home({  
 Key? key,  
 }) : **super**(key: key);  
  
 @override  
 Widget build(BuildContext context, WidgetRef ref) {  
 **final** notesList = ref.watch(noteProvider);  
 **final** titleController = TextEditingController();  
 **final** contentController = TextEditingController();  
 **return** Scaffold(  
 appBar: AppBar(  
 title: **const** Text(**'Notes App'**),  
 ),  
 body: Column(  
 children: [  
 ElevatedButton(onPressed: () {  
 showDialog(context: context, builder: (\_){  
 **return** AlertDialog(  
 content: Column(  
 children: [  
 TextField(  
 controller: titleController,  
 decoration: InputDecoration(  
 labelText: **'Title'**,  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(10),  
 ),  
 ),  
 ),  
 **const** SizedBox(height: 20),  
 TextField(  
 controller: contentController,  
 decoration: InputDecoration(  
 labelText: **'Content'**,  
 border: OutlineInputBorder(  
 borderRadius: BorderRadius.circular(10),  
 ),  
 ),  
 ),  
 ],  
 ),  
 actions: [  
 ElevatedButton(onPressed: () {  
 ref.read(noteProvider.**notifier**).addNote(Notes(  
 title: titleController.**text**,  
 content: contentController.**text**,  
 ));  
 Navigator.*pop*(context);  
 }, child: **const** Text(**'Add note'**)),  
 ],  
 );  
 });  
 }, child: **const** Text(**'Add Note'**)),  
 **const** SizedBox(height: 20),  
 notesList.**isEmpty**? **const** Text(**'Add notes '**)  
 : ListView.builder(  
 itemCount: notesList.**length**,  
 shrinkWrap: **true**,  
 itemBuilder: (context, index) {  
 **return** ListTile(  
 title: Text(notesList[index].**title**),  
 subtitle: Text(notesList[index].**content**),  
 trailing: IconButton(  
 icon: **const** Icon(Icons.*delete*),  
 onPressed: () {  
 ref.read(noteProvider.**notifier**).removeNote(notesList[index]);  
 },  
 ),  
 );  
 })  
 ],  
 )  
 );  
 }  
}