

ACARA 1 : Hive Flutter

Pokok Bahasan	: Hive Flutter
Acara Praktikum/Pertemuan	: Minggu 9
Tempat	: Daring / Luring Politeknik Negeri Jember
Alokasi Waktu	: 3x50 menit

a. Capaian Pembelajaran Mata Kuliah (CPMK)

1. Mahasiswa mampu memahami konsep dasar penggunaan Hive.
2. Mahasiswa mampu menggunakan dan menerapkan Hive.
3. Mahasiswa mampu menggunakan dan menerapkan Hive.


b. Indikator

Kemampuan mahasiswa dalam memahami dan menerapkan konsep serta penggunaan Hive untuk CRUD .


c. Dasar Teori


Hive dirancang dengan Flutter, menggunakan sebuah konsep pendekatan NoSQL yang ringan, sangat cepat, dan bersifat lokal, untuk pengembang yang ditulis murni dalam bahasa Pemrograman Dart.


Kelebihan :

 Cross platform: mobile, desktop, browser

 Great performance (see benchmark)

 Simple, powerful, & intuitive API

 Strong encryption built in

 NO native dependencies

 Batteries included

Benchmark

1000 read iterations

1000 write iterations



Hive Box

Merupakan sebuah teknik terorganisir untuk menyimpan data. Kita dapat membandingkan Hive Box dengan tabel SQL, tetapi tidak memiliki struktur apa pun dan dapat berisi apa pun.

Type Adapters

Type Adapter di sini bertindak sebagai konverter Object ke dan dari bentuk biner, karena Hive menggunakan tipe primitif seperti List, Map, DateTime dan Uint8List masing-masing. Type Adapter perlu dibuat, dapat dengan cara mengkode sendiri tetapi cara teraman adalah dengan membuatnya menggunakan hive_generator.

d. Alat dan Bahan

1. Notebook
2. Aplikasi Visual Studio Code / Andorid Studio
3. Hive
4. HP Android
5. Kabel Data

e. Prosedur Kerja

1. Buat project baru dengan nama flutter_hive.

2. Tambahkan dependencies pada file pubspec.yaml dan jalankan perintah flutter pub get.

```
sdk: flutter

# The following adds the Cupertino Icons font to your application.
# Use with the CupertinoIcons class for iOS style icons.
cupertino_icons: ^1.0.2
hive: ^2.2.3
hive_flutter: ^1.1.0

dev_dependencies:
  flutter_test:
    sdk: flutter

# The "flutter_lints" package below contains a set of recommended lints to
```

3. Tambahkan import pada main.dart menjadi seperti berikut.

```
import 'package:flutter/material.dart';
import 'package:hive/hive.dart';
import 'package:hive_flutter/hive_flutter.dart';
```

4. Ubah void main menjadi seperti berikut.

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Hive.initFlutter();
  // await Hive.deleteBoxFromDisk('shopping_box');
  await Hive.openBox('shopping_box');

  runApp(const MyApp());
}
```

5. Ubah stateless widget kelas MyApp menjadi seperti berikut.

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'KindaCode.com',
      theme: ThemeData(
        primarySwatch: Colors.green,
      ),
      home: const HomePage(),
    );
  }
}
```

6. Buat stateful widget dengan nama kelas HomePage dan ubah isinya menjadi seperti berikut.

```

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'KindaCode.com',
      theme: ThemeData(
        primarySwatch: Colors.green,
      ),
      home: const HomePage(),
    );
  }
}

// Home Page
class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  List<Map<String, dynamic>> _items = [];

  final _shoppingBox = Hive.box('shopping_box');

  @override
  void initState() {
    super.initState();
    _refreshItems(); // Load data when app starts
  }

  // Get all items from the database
  void _refreshItems() {
    final data = _shoppingBox.keys.map((key) {
      final value = _shoppingBox.get(key);
      return {"key": key, "name": value["name"], "quantity":
value['quantity']};
    }).toList();

    setState(() {
      _items = data.reversed.toList();
      // we use "reversed" to sort items in order from the latest to
the oldest
    });
  }

  // Create new item
  Future<void> _createItem(Map<String, dynamic> newItem) async {
    await _shoppingBox.add(newItem);
    _refreshItems(); // update the UI
  }

  // Retrieve a single item from the database by using its key

```

```

    // Our app won't use this function but I put it here for your
reference
    Map<String, dynamic> _readItem(int key) {
        final item = _shoppingBox.get(key);
        return item;
    }

    // Update a single item
    Future<void> _updateItem(int itemKey, Map<String, dynamic> item)
async {
        await _shoppingBox.put(itemKey, item);
        _refreshItems(); // Update the UI
    }

    // Delete a single item
    Future<void> _deleteItem(int itemKey) async {
        await _shoppingBox.delete(itemKey);
        _refreshItems(); // update the UI

        // Display a snackbar
        ScaffoldMessenger.of(context).showSnackBar(
            const SnackBar(content: Text('An item has been deleted')));
    }

    // TextFields' controllers
    final TextEditingController _nameController =
TextEditingController();
    final TextEditingController _quantityController =
TextEditingController();

    // This function will be triggered when the floating button is
pressed
    // It will also be triggered when you want to update an item
    void _showForm(BuildContext ctx, int? itemKey) async {
        // itemKey == null -> create new item
        // itemKey != null -> update an existing item

        if (itemKey != null) {
            final existingItem =
                _items.firstWhere((element) => element['key'] == itemKey);
            _nameController.text = existingItem['name'];
            _quantityController.text = existingItem['quantity'];
        }

        showModalBottomSheet(
            context: ctx,
            elevation: 5,
            isScrollControlled: true,
            builder: (_) => Container(
                padding: EdgeInsets.only(
                    bottom: MediaQuery.of(ctx).viewInsets.bottom,
                    top: 15,
                    left: 15,
                    right: 15),
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.min,
                    crossAxisAlignment: CrossAxisAlignment.end,

```

```

        children: [
          TextField(
            controller: _nameController,
            decoration: const InputDecoration(hintText: 'Name'),
          ),
          const SizedBox(
            height: 10,
          ),
          TextField(
            controller: _quantityController,
            keyboardType: TextInputType.number,
            decoration: const InputDecoration(hintText:
'Quantity'),
          ),
          const SizedBox(
            height: 20,
          ),
          ElevatedButton(
            onPressed: () async {
              // Save new item
              if (itemKey == null) {
                _createItem({
                  "name": _nameController.text,
                  "quantity": _quantityController.text
                });
              }

              // update an existing item
              if (itemKey != null) {
                _updateItem(itemKey, {
                  'name': _nameController.text.trim(),
                  'quantity': _quantityController.text.trim()
                });
              }

              // Clear the text fields
              _nameController.text = '';
              _quantityController.text = '';

              Navigator.of(context).pop(); // Close the bottom
sheet
            },
            child: Text(itemKey == null ? 'Create New' : 'Update'),
          ),
          const SizedBox(
            height: 15,
          ),
        ],
      ),
    ));
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('KindaCode.com'),

```

```

),
body: _items.isEmpty
  ? const Center(
    child: Text(
      'No Data',
      style: TextStyle(fontSize: 30),
    ),
  )
  : ListView.builder(
    // the list of items
    itemCount: _items.length,
    itemBuilder: (_, index) {
      final currentItem = _items[index];
      return Card(
        color: Colors.orange.shade100,
        margin: const EdgeInsets.all(10),
        elevation: 3,
        child: ListTile(
          title: Text(currentItem['name']),
          subtitle: Text(currentItem['quantity'].toString()),
          trailing: Row(
            mainAxisAlignment: MainAxisAlignment.min,
            children: [
              // Edit button
              IconButton(
                icon: const Icon(Icons.edit),
                onPressed: () =>
                  _showForm(context, currentItem['key']),
              // Delete button
              IconButton(
                icon: const Icon(Icons.delete),
                onPressed: () =>
                  _deleteItem(currentItem['key']),
            ],
          ),
        ),
      );
    },
  );
// Add new item button
floatingActionButton: FloatingActionButton(
  onPressed: () => _showForm(context, null),
  child: const Icon(Icons.add),
),
);
}
}

```

7. Jika tidak ada ketika dijalankan akan menjadi seperti berikut.

