Flutter Local Storage: Using Shared Preferences & SQLite

Subtitle: Learn how to save data locally in your Flutter apps for persistence between app launches.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Most apps need to **store user data locally**, like login info, settings, or saved items. Flutter provides **Shared Preferences** for key-value storage and **SQLite** for structured data. This guide teaches you both approaches.

Step 1: Shared Preferences Setup

1. Add dependency in `pubspec.yaml`:

```
dependencies:
  shared_preferences: ^2.0.15
```

2. Import package:

```
import 'package:shared_preferences/shared_preferences.dart';
```

Step 2: Saving Data with SharedPreferences

```
final prefs = await SharedPreferences.getInstance();
await prefs.setString('username', 'Hafsa');
await prefs.setInt('counter', 10);
```

Exercise: Save user’s favorite color and display it on app start.

Step 3: Retrieving Data

```
final prefs = await SharedPreferences.getInstance();
String username = prefs.getString('username') ?? 'Guest';
int counter = prefs.getInt('counter') ?? 0;
```

Exercise: Display saved username and counter on the home screen.

Step 4: Removing or Clearing Data

```
await prefs.remove('username'); // Remove single key
await prefs.clear(); // Clear all keys
```

Exercise: Add a reset button to clear stored data.

Step 5: Using SQLite for Structured Data

1. Add dependency:

```
dependencies:
  sqflite: ^2.2.0+3
  path: ^1.8.3
```

2. Import packages:

```
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';
```

3. Initialize Database:

```
final database = openDatabase(
  join(await getDatabasesPath(), 'app.db'),
  onCreate: (db, version) {
    return db.execute(
      "CREATE TABLE users(id INTEGER PRIMARY KEY, name TEXT, age INTEGER)",
    );
  },
  version: 1,
);
```

Step 6: CRUD Operations with SQLite

- **Insert:**

```
await db.insert('users', {'name': 'Hafsa', 'age': 22});
```

```
await db.insert('users', {'name': 'Harsa', 'age': 22});
```

- **Read:**

```
final List<Map<String, dynamic>> users = await db.query('users');
```

- **Update:**

```
await db.update('users', {'age': 23}, where: "id = ?", whereArgs: [1]);
```

- **Delete:**

```
await db.delete('users', where: "id = ?", whereArgs: [1]);
```

Exercise: Build a simple contacts app using SQLite with add, update, and delete functionality.

Step 7: Mini Project

Build a **Persistent Counter App**:

- Use **SharedPreferences** to save counter value
- Use **SQLite** to save a list of favorite numbers
- Counter persists between app restarts
- Display saved numbers in a ListView

Key Takeaways

- SharedPreferences = simple key-value storage (good for small data)
- SQLite = structured data storage (good for larger datasets)
- Always handle async operations properly
- Local storage is essential for **real-world apps with persistence**

Ignoring local storage will make your apps **forget user data** and feel incomplete. Learn this to make apps reliable.

Visit **haas.dev** for more Flutter guides, tutorials, and complete project resources.

Website Name: haas.dev

Website Link: https://dev-roast-app.vercel.app
