Flutter HTTP & API Integration: Fetching & Displaying Data

Subtitle: Learn how to connect your Flutter app to external APIs and display dynamic data.

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

Introduction

Most apps rely on **external data** from APIs. Flutter makes it easy to fetch, parse, and display JSON data. This guide covers the **http package, async requests, and JSON handling**.

Step 1: Add HTTP Dependency

```
dependencies:
  http: ^1.1.0
```

Run:

```
flutter pub get
```

Import in your Dart file:

```
import 'package:http/http.dart' as http;
import 'dart:convert';
```

Step 2: Making a GET Request

```
Future<void> fetchData() async {
  final response = await http.get(Uri.parse('https://jsonplaceholder.typicode.co

  if (response.statusCode == 200) {
    List data = jsonDecode(response.body);
    print(data);
  } else {
    print('Failed to fetch data');
  }
}
```

Exercise: Fetch posts from JSONPlaceholder API and print the first post.

Step 3: Displaying Data in ListView

```
ListView.builder(
  itemCount: data.length,
  itemBuilder: (context, index) {
    return ListTile(
      title: Text(data[index]['title']),
      subtitle: Text(data[index]['body']),
    );
  },
)
```

Exercise: Display all posts from API in a scrollable ListView.

Step 4: Handling Errors & Loading State

```
if (_isLoading) return CircularProgressIndicator();
if (_error != null) return Text("Error: $_error");
```

- Use `try-catch` to handle network errors
- Show loading indicator while fetching data

Exercise: Add error handling and a loading spinner to your API app.

Step 5: POST Request Example

```
Future<void> sendData() async {
  final response = await http.post(
    Uri.parse('https://jsonplaceholder.typicode.com/posts'),
    headers: {'Content-Type': 'application/json'},
    body: jsonEncode({'title': 'Hello', 'body': 'Flutter API', 'userId': 1}),
  );

  if (response.statusCode == 201) {
    print('Data sent successfully');
  }
}
```

Exercise: Add a form to send new posts to the API.

Step 6: Mini Project

Build a **Flutter API App**:

- Fetch posts from JSONPlaceholder
- Display posts in ListView
- Add pull-to-refresh feature
- Send new post using POST request
- Handle loading and error states

Key Takeaways

- Use `http` package for GET & POST requests
- Parse JSON using `dart:convert`
- Always handle **loading and errors**
- API integration is essential for **dynamic real-world apps**
- Practice fetching, displaying, and sending data

Skipping API integration makes your apps **static and limited**, unable to interact with real data.

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

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