TIERS LIMITED SUMMER INTERNSHIP 2024

MOBILE APP DEVELOPMENT

Networking and APIs

CLASS # 7

TIERS
Limited

Overview

In this class, we will explore the fundamentals of networking in Flutter, focusing on making HTTP requests using the `http` package and parsing JSON data. These are essential skills for building Flutter applications that interact with web services and APIs.

Objectives

- Understand how to make HTTP requests using the http package.
- Learn to parse JSON data into Dart objects.
- Implement basic networking functionalities in a Flutter application.

HTTP Requests Using the http Package

Adding the http Package to Your Project

First, you need to add the **`http`** package to your **`pubspec.yaml`** file:

```
dependencies:
  flutter:
    sdk: flutter
  http: ^latest version
```

Run **`flutter pub get`** to install the package.

Making GET Requests

A GET request is used to fetch data from a server.

Example: Fetching a Random Dog Image

```
  DogImage dogImage = DogImage(message: 'message', status: 'status');

  @override
  void initState() {
    super.initState();
    fetchDogImage();
  }

  Future<void> fetchDogImage() async {
    final response = await http.get(
        Uri.parse('https://dog.ceo/api/breeds/image/random')
    );

    if (response.statusCode == 200) {
      setState(() {
        dogImage = DogImage.fromJson(json.decode(response.body));
      });
    }
    else {
      throw Exception('Failed to load dog image');
    }
  }

class DogImage {
  final String message;
  final String status;

  DogImage({required this.message, required this.status});

  factory DogImage.fromJson(Map<String, dynamic> json) {
    return DogImage(
      message: json['message'],
      status: json['status'],
    );
  }
}
```

Explanation of the Code

`_MyHomePageState` Class

- _MyHomePageState is responsible for maintaining the state of MyHomePage.
- It declares a DogImage variable called dogImage and initializes it with placeholder values.
- In the initState method, the fetchDogImage function is called to fetch data from the server.

`fetchDogImage` Method

- fetchDogImage is an asynchronous function that makes a GET request to fetch data from the server.
- It uses the http.get method to send the request.
- If the response status code is 200 (OK), it updates the dogImage variable using setState.
- If the response status code is not 200, it throws an exception.

`DogImage` Class

- The DogImage class represents the data model for a dog image.
- It has properties for message and status.
- It includes a factory constructor DogImage.fromJson to create a DogImage object from a JSON map.

Parsing JSON Data

Parsing JSON to Dart Objects

Use the dart:convert package to parse JSON data.

```
import 'dart:convert';
```

```
DogImage dogImageFromJson(String str) => DogImage.fromJson(json.decode(str));

String dogImageToJson(DogImage data) => json.encode(data.toJson());

class DogImage {
  final String message;
  final String status;

  DogImage({
    required this.message,
    required this.status,
  });

  factory DogImage.fromJson(Map<String, dynamic> json) => DogImage(
    message: json["message"],
    status: json["status"],
  );

  Map<String, dynamic> toJson() => {
    "message": message,
    "status": status,
  };
}
```

In this example:

- dogImageFromJson and dogImageToJson functions are used to parse JSON data into DogImage objects and convert DogImage objects to JSON strings.
- The DogImage class includes a factory constructor DogImage.fromJson to create a DogImage object from a JSON map and a toJson method to convert a DogImage object to a JSON map.

Exercises

Exercise 1: Making a GET Request

- Create a Flutter app that makes a GET request to fetch a random dog image and displays it using an `Image` widget.

Exercise 2: Parsing JSON Data

- Parse the JSON response from the API into Dart objects and display the image URL in a `Text` widget.