

**SEB-312 Mobile Application Development**

**LAB # 9**

**LAB Title**

|  |
| --- |
| Using Provider to Globally Manage State in a Flutter List App. Making HTTP GET requests using the http package |

**Assessment of CLO: 04, PLO: 05**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Name:** |  | | |
| **Roll No.** |  | | |
| **Semester** |  | **Session** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S. No.** | **Perf. Level**  **Criteria** | **Excellent**  **(2.5)** | **Good**  **(2)** | **Satisfactory**  **(1.5)** | **Needs Improvement**  **(0 ~ 1)** | **Marks Obtained** |
| **1** | Project Execution & Implementation | Fully functional, optimized, and well-structured. | Minor errors, mostly functional. | Some errors, requires guidance. | Major errors, non-functional, or not Performed. |  |
| **2** | Results & Debugging  Or Troubleshooting | Accurate results with effective debugging  Or Troubleshooting. | Mostly correct, some debugging Or Troubleshooting needed. | Partial results, minimal debugging  Or Troubleshooting. | Incorrect results, no debugging Or Troubleshooting, or not attempted. |  |
| **3** | Problem-Solving & Adaptability  (VIVA) | Creative approach, efficiently solves challenges. | Adapts well, minor struggles. | Some adaptability, needs guidance. | Lacks innovation or no innovation, unable to solve problems. |  |
| **4** | Report Quality & Documentation | Clear, structured, with detailed visuals. | Mostly clear, minor gaps. | Some clarity issues, missing details. | Poorly structured, lacks clarity, or not submitted. |  |
| **Total Marks Obtained Out of 10** | | | | | |  |

**Experiment evaluated by**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instructor’s Name** | **Sidra Khatoon** | | |
| **Date** |  | **Signature** |  |

**Objective**

The objective of lab is using Provider to Globally Manage State in a Flutter List App and HTTP GET requests using the http package. Fetching and displaying data from a REST API.

**Instructions**

You have to perform the following tasks yourselves. Raise your hand if you face any difficulty in understanding and solving these tasks. **Plagiarism** is an abhorrent practice and you should not engage in it.

**Add Item to List App**

Let’s build an example where users can add items to a list, and this state will be managed globally using Provider.

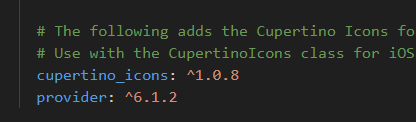
**Step 1: Setting Up the Project**

you’ll need to set up the project and include the Provider package. Let run command in flutter terminal

flutter pub add provider

flutter pub get

After these command provider package in included in pubspec.yaml. You can check your file.



**Step 2: Creating the Item Model**

We’ll start by creating a Item model which is extend with ChnageNotifier class. It will manage the list of items.

import 'package:provider/provider.dart';

class ItemModel extends ChangeNotifier {

    List<String> \_items = [];

    List<String> get items => \_items;

    void addItem(String item) {

        \_items.add(item);

        notifyListeners();

    }

    void removeItem(int index) {

        \_items.removeAt(index);

        notifyListeners();

    }

}

In the ItemModel class:

\_items: A private list that holds the added items.

addItem: Adds a new item to the list and notifies listeners that the state has changed.

removeItem: Removes an item from the list by its index and notifies listeners.

**Step 3: Providing the Item Model**

Next, we wrap our app with ChangeNotifierProvider to provide the ItemModel to all widgets in the app.

void main() {

  runApp(ChangeNotifierProvider(

    create: (\_)=>ItemModel(),

    child: const MyApp()));

}

The ChangeNotifierProvider makes the ItemModel available to the entire widget tree. Any widget in the tree can now access and modify the global state.

**Step 4: Building the UI**

Now, let’s build the UI where users can add and remove items from the list.

class ItemListScreen extends StatelessWidget {

    final TextEditingController \_controller = TextEditingController();

    ItemListScreen({super.key});

    @override

    Widget build(BuildContext context) {

        return Scaffold(

            appBar: AppBar(

              title: const Text('Item List'),

              backgroundColor: Colors.purpleAccent,

              centerTitle: true,),

            body: Column(

                children: [

                    Padding(

                        padding: const EdgeInsets.all(8.0),

                        child: TextField(

                            controller: \_controller,

                            decoration: const InputDecoration(labelText: 'Enter item'),

                        ),

                    ),

                    ElevatedButton(

                        onPressed: () {

                            Provider.of<ItemModel>(context, listen: false)

                                .addItem(\_controller.text);

                            \_controller.clear();

                        },

                        child: const Text('Add Item'),

                    ),

                    Expanded(

                        child: Consumer<ItemModel>(

                            builder: (context, itemModel, child) {

                                return ListView.builder(

                                    itemCount: itemModel.items.length,

                                    itemBuilder: (context, index) {

                                        return ListTile(

                                            title: Text(itemModel.items[index]),

                                            trailing: IconButton(

                                                icon: const Icon(Icons.delete),

                                                onPressed: () {

                                                    Provider.of<ItemModel>(context, listen: false)

                                                        .removeItem(index);

                                                },

                                            ),

                                        );

                                    },

                                );

                            },

                        ),

                    ),

                ],

            ),

        );

    }

}

**Complete Code:**

import 'package:flutter/material.dart';

import 'package:provider/provider.dart';

class ItemModel extends ChangeNotifier {

    List<String> \_items = [];  // Changed from \_item to \_items

    List<String> get items => \_items;

    void addItem(String item) {

        \_items.add(item);

        notifyListeners();

    }

    void removeItem(int index) {

        \_items.removeAt(index);

        notifyListeners();

    }

}

void main() {

  runApp(ChangeNotifierProvider(

    create: (\_) => ItemModel(),

    child: const MyApp()));

}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      debugShowCheckedModeBanner: false,

      title: 'Flutter Demo',

      theme: ThemeData(

        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),

        useMaterial3: true,

      ),

      home: ItemListScreen(),

    );

  }

}

class ItemListScreen extends StatelessWidget {

    final TextEditingController \_controller = TextEditingController();

    ItemListScreen({super.key});

    @override

    Widget build(BuildContext context) {

        return Scaffold(

            appBar: AppBar(

              title: const Text('Item List'),

              backgroundColor: Colors.purpleAccent,

              centerTitle: true,),

            body: Column(

                children: [

                    Padding(

                        padding: const EdgeInsets.all(8.0),

                        child: TextField(

                            controller: \_controller,

                            decoration: const InputDecoration(labelText: 'Enter item'),

                        ),

                    ),

                    ElevatedButton(

                        onPressed: () {

                            Provider.of<ItemModel>(context, listen: false)

                                .addItem(\_controller.text);

                            \_controller.clear();

                        },

                        child: const Text('Add Item'),

                    ),

                    Expanded(

                        child: Consumer<ItemModel>(

                            builder: (context, itemModel, child) {

                                return ListView.builder(

                                    itemCount: itemModel.items.length,

                                    itemBuilder: (context, index) {

                                        return ListTile(

                                            title: Text(itemModel.items[index]),

                                            trailing: IconButton(

                                                icon: const Icon(Icons.delete),

                                                onPressed: () {

                                                    Provider.of<ItemModel>(context, listen: false)

                                                        .removeItem(index);

                                                },

                                            ),

                                        );

                                    },

                                );

                            },

                        ),

                    ),

                ],

            ),

        );

    }

}

**Flutter - Make an HTTP GET Request**

In app development, the ability to fetch data from external sources, such as REST APIs, is a fundamental requirement. In Flutter, Whether you need to fetch data from a RESTful API, access a database, or retrieve content from a web server, Flutter provides you with the tools and packages(HTTP) to do this kind of API calls easily. Here we will explore how to interact with external data sources, retrieve JSON data, and integrate it into your Flutter application.

**URL of API Request :**

<https://jsonplaceholder.typicode.com/posts>

**JSON Data Format (Sample):**

{

"userId": 1,

"id": 1,

"title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",

"body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"

},

**Step By Step Implementations**

**Step 1: Create a New Project in VS code**

Create a new project in VS code name as per your choice.

**Step 2: Add the dependency to our project**

In the pubspec.yaml file we have to add the below dependency for the http package which provide various API call methods. We can add this dependency by using this command in terminal

flutter pub add http

flutter pub get

**Step 3:** **Import the HTTP package**

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

import 'dart:convert';

**Step 4: Define the API endpoint**

final String apiUrl = "https://jsonplaceholder.typicode.com/posts";

**Step 5: Use models to represent data**

Avoid working with raw maps. Create Dart classes for each data type.

class Post {

  final int id;

  final String title;

  final String body;

  Post({required this.id, required this.title, required this.body});

  factory Post.fromJson(Map<String, dynamic> json) {

    return Post(

      id: json['id'],

      title: json['title'],

      body: json['body'],

    );

  }

}

**Step 6: Fetch data with a GET request**

 Future<List<Post>> fetchPosts() async {

    final response = await http.get(Uri.parse(apiUrl));

    if (response.statusCode == 200) {

      List data = jsonDecode(response.body);

      return data.map((item) => Post.fromJson(item)).toList();

    } else {

      throw Exception('Failed to load posts');

    }

  }

**Step 7: Create a UI to show the data**

 @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text("Posts"),

      centerTitle: true,

      backgroundColor: Colors.purpleAccent,

      ),

      body: FutureBuilder(

        future: fetchPosts(),

        builder: (context, snapshot) {

          if (snapshot.connectionState == ConnectionState.waiting) {

            return Center(child: CircularProgressIndicator());

          }

          if (snapshot.hasError) {

            return Center(child: Text("Error: ${snapshot.error}"));

          }

          final List posts = snapshot.data as List<Post>;

          return ListView.builder(

            itemCount: posts.length,

            itemBuilder: (context, index) {

              return ListTile(

                 title: Text(posts[index].title),

                 subtitle: Text(posts[index].body),

              );

            },

          );

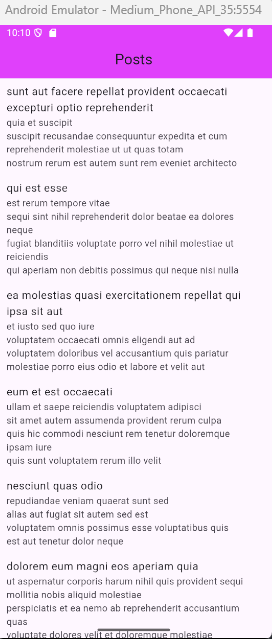
        },

      ),

    );

  }

**Final Output**



**Final Code:**

import 'package:flutter/material.dart';

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

import 'dart:convert';

void main() {

  runApp(const MyApp());

}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  // This widget is the root of your application.

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      debugShowCheckedModeBanner: false,

      title: 'Flutter Demo',

      theme: ThemeData(

        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),

        useMaterial3: true,

      ),

      home:PostPages(),

    );

  }

}

class PostPages extends StatelessWidget{

  const PostPages({super.key});

  final String apiUrl = "https://jsonplaceholder.typicode.com/posts";

  Future<List<Post>> fetchPosts() async {

    final response = await http.get(Uri.parse(apiUrl));

    if (response.statusCode == 200) {

      List data = jsonDecode(response.body);

      return data.map((item) => Post.fromJson(item)).toList();

    } else {

      throw Exception('Failed to load posts');

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text("Posts"),

      centerTitle: true,

      backgroundColor: Colors.purpleAccent,

      ),

      body: FutureBuilder(

        future: fetchPosts(),

        builder: (context, snapshot) {

          if (snapshot.connectionState == ConnectionState.waiting) {

            return Center(child: CircularProgressIndicator());

          }

          if (snapshot.hasError) {

            return Center(child: Text("Error: ${snapshot.error}"));

          }

          final List posts = snapshot.data as List<Post>;

          return ListView.builder(

            itemCount: posts.length,

            itemBuilder: (context, index) {

              return ListTile(

                 title: Text(posts[index].title),

                 subtitle: Text(posts[index].body),

              );

            },

          );

        },

      ),

    );

  }

}

class Post {

  final int id;

  final String title;

  final String body;

  Post({required this.id, required this.title, required this.body});

  factory Post.fromJson(Map<String, dynamic> json) {

    return Post(

      id: json['id'],

      title: json['title'],

      body: json['body'],

    );

  }

}

**Assessment:**

**Task 1:** Create a Flutter app for managing a simple list of tasks. Use the Provider package to manage the list of tasks, where each task has a title, description, and a completion status.

Expected Output:

A basic task management app where:

* A list of tasks can be added and displayed.
* Each task has a checkbox to mark it as completed.
* The task list updates immediately upon adding or toggling tasks.

**Task 2: Display User Data from API in a Flutter App**

**Objective:**  
Build a Flutter application that fetches and displays a list of users from a REST API using the http package and shows the user details in a ListView.

**Instructions:**

1. Use the following public API endpoint to retrieve user data:  
   **https://jsonplaceholder.typicode.com/users**
2. Create a model class named User with the following fields:
   * id (int)
   * name (String)
   * email (String)
   * phone (String)
3. Implement a function to fetch user data from the API using http.get() and convert the JSON response into a list of User objects.
4. Use FutureBuilder to asynchronously display the list of users in a ListView.builder.
5. Each ListTile should display:
   * **Title:** User’s name
   * **Subtitle:** User’s email and phone number
6. Handle loading and error states properly.