TIERS LIMITED SUMMER INTERNSHIP 2024

MOBILE APP DEVELOPMENT

Shared Preference Helper

CLASS # 15

TIERS
Limited

Overview

In this class, we will cover the use of Shared Preferences in Flutter for simple data storage. Shared Preferences is a key-value storage mechanism that allows you to store and retrieve simple data persistently.

Objectives

- Understand the use of Shared Preferences for local data storage.
- Learn how to store, retrieve, and remove data using Shared Preferences.
- Implement a simple student management system using Shared Preferences.

Shared Preferences in Flutter

Shared Preferences allows you to store data in a key-value pair format. It's useful for storing simple data such as user settings, preferences, or small amounts of app data.

Step 1: Add Dependencies

To use Shared Preferences, you need to add the **`shared_preferences`** package to your **`pubspec.yaml`** file:

```
  shared_preferences: ^latest version
```

Step 2: Create a Student Class

This class represents a student with an ID, name, age, and semester. It includes methods for serializing and deserializing the student object to and from JSON.

```
class Student {
  final String id;
  final String name;
  final int age;
  final String semester;

  Student({
    required this.id,
    required this.name,
    required this.age,
    required this.semester,
  });

  Map<String, dynamic> toJson() => {
        'id': id,
        'name': name,
        'age': age,
        'semester': semester,
      };
```

```
  factory Student.fromJson(Map<String, dynamic> json) => Student(
        id: json['id'],
        name: json['name'],
        age: json['age'],
        semester: json['semester'],
      );

  String toJsonString() => json.encode(toJson());

  factory Student.fromJsonString(String jsonString) =>
Student.fromJson(json.decode(jsonString));

}
```

Step 3: Create StudentManager Class

This class handles all the Shared Preferences operations, such as adding, retrieving, and removing student data.

```
  static Future<SharedPreferences> get _prefs async {
        await SharedPreferences.getInstance();
  }

  static Future<List<Student>> getAllStudents() async {
    final prefs = await _prefs;
    List<Student> studentList = [];
    var keys = prefs.getKeys();

    for (var key in keys) {
      var studentRecord = prefs.getString(key) ?? "";
      Student student = Student.fromJsonString(studentRecord);
      studentList.add(student);
    }
    return studentList;
  }

  static Future<void> addStudent(Student student) async {
    final prefs = await _prefs;
    prefs.setString(student.id, student.toJsonString());
  }

  static Future<void> removeStudent(String id) async {
    final prefs = await _prefs;
    prefs.remove(id);
  }
```

Step 4: Main Application

The main application sets up a **`ListView.builder`** to display the list of students and includes an add button to add new students.

```
List<Student> _students = [];
  bool _isLoading = true;

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

  void _loadStudents() async {
    List<Student> students = await StudentManager.getAllStudents();
    setState(() {
      _students = students;
      _isLoading = false;
    });
  }

  void _addStudent() {
    final student = Student(
      id: DateTime.now().toString(),
      name: 'John Doe',
      age: 20,
      semester: 'Fall 2021',
    );
    StudentManager.addStudent(student).then((_) {
      setState(() {
        _students.add(student);
      });
    });
  }

  void _removeStudent(String id) {
    StudentManager.removeStudent(id).then((_) {
      setState(() {
        _students.removeWhere((student) => student.id == id);
      });
    });
  }


        ListView.builder(
              itemCount: _students.length,
              itemBuilder: (context, index) {
                final student = _students[index];
                return ListTile(
                  title: Text(student.name),
                  subtitle: Text(
                        'Age: ${student.age}, Semester: ${student.semester}'
                  ),
                  trailing: IconButton(
                    icon: Icon(Icons.delete),
                    onPressed: () => _removeStudent(student.id),
                  ),
                );
              },
            ),
```

Explanation of Key Components

- **Student Class**: This class includes methods for serializing and deserializing a student object to and from JSON.
- **StudentManager Class**: Manages adding, retrieving, and removing students using Shared Preferences.
- **MyHomePage Class**: Displays a list of students and includes an add button for adding new students.

Key Points

- **initState()**: Initializes the state and loads the student data.
- **_loadStudents()**: Asynchronously fetches the student data from Shared Preferences.
- **_addStudent()**: Adds a new student to Shared Preferences and updates the list.
- **_removeStudent()**: Removes a student from Shared Preferences and updates the list.
- **ListView.builder**: Displays the list of students dynamically.
- **CircularProgressIndicator**: Shows a loading indicator while the data is being fetched.

By following these steps, you can efficiently use Shared Preferences to manage simple data storage in your Flutter applications.

Exercises:

Implement Add, Edit, Remove and Load Operations.

- Use Dialogs for add and edit operations
- Use Dialog “Are you sure to delete?” Yes or No when user press on delete button.