Mobile Development :

9 : Flutter for Mobile Development : Part 4

Future Builder + Databases + Background Services

Professor Imed Bouchrika
National School of Artificial Intelligence
imed.bouchrika@ensia.edu.dz

Outline :

- **Section 1 : Async, Timer & Future Builder**
    - ***Writing Async Functions***
    - ***Future Builder Widget***
    - ***Refresh on slide***
    - ***Timer for Periodic Events***
- **Section 2 : Data Persistence**
    - ***Shared Preferences***
    - ***SQFLITE***
    - ***Case Study : ToDo App***
- **Section 3 : Synching**
    - ***Synching Techniques***
    - ***Cron Services***

Summary/Draft of Flutter Lectures

- **Lecture 1 (W6): Introduction to Dart & Budget Building**
  - Introduction to Dart
  - Flutter and Simple UI Building
- **Lecture 2 (W7): More on Widgets**
  - More Widgets,
  - Stateless and Stateful + Interactivity
  - Navigation
- **Lecture 3 (W9): State Management + MVP Building**
  - State Management using GetX
  - MVP Building
  - More advanced Widgets

Summary/Draft of Flutter Lectures

- **Lecture 4 (W10): Data Persistence**
  - SharedPreferences + Hive
  - SQFLite
  - Data Synching
- **Lecture 5 (W11): Firebase**
  - Messaging & Notification
  - Data Storage
  - Authentication Module
- **Lecture 6 (W12): Advanced Features : ML + Hardware …**
  - Working with Hardware
  - Google MAPs and GPS Data.
  - ML Features

W13 : Building Backends
W14: Testing the App
W15: Publishing, Monetizing & Business Models

Section 1

Async, Timer and FutureBuilder

Writing Async Functions in Flutter

- **Synchronous Function/Instruction**
  - Executed Sequentially, cannot move until the current instruction is fully completed.
- **Asynchronous Function/Instruction**
  - Function or Set of instructions will be executed on a separate thread without blocking the main code.
    - The main program completes work while it may wait for "async" operations to finish in the future.

Writing Async Functions in Flutter

- **Asynchronous Function :**
  - Common asynchronous operations include :
    - Fetching/Sending data over a network.
    - Reading/Writing to a database.
    - Writing/Reading data from a local file.
    - Opening/Communicating with the device hardware
    - ..

Writing Async Functions in Flutter

- **Async**
  - To declare a function as asynchronous, use the **async** keyword before the body.

```
void getData() async{
     //some business logic here…
}
```

Writing Async Functions in Flutter

- **Return Type of Asynchronous Functions :**
  - Asynchronous functions in Dart/Flutter **return** Future Objects.
  - For Async Functions, the return **must** be always a future. (or other types of similar nature)
    - String → Future <String>
    - int → Future<int>
    - Map → Future<Map>
    - void → Future<void>

Writing Async Functions in Flutter

- **Return Type of Asynchronous Functions :**
  - Asynchronous functions in Dart/Flutter **return** Future Objects.
  - For Async Functions, the return **must** be always a future. (or other types of similar nature)
    - String → Future <String>
    - int → Future<int>
    - Map → Future<Map>
    - void → Future<void>

```
Future<String> myStr=getData() ;
Future<Map> mydata=getData() ;
```

Writing Async Functions in Flutter

- **Await**
  - The **await** will make an asynchronous instruction as a synchronous one
    - Wait until the instruction completes and proceeds to the next instruction

```
some _ instruction
myData = await getData() ;
some _ instruction
```

Writing Async Functions in Flutte

What type of myData ?

Future or non-Future

- **Await**
  - The **await** will make a
    - Wait until the ins

      instruction

```
some _ instruction
myData = await getData() ;
some _ instruction
```

Writing Async Functions in

- A

one

Very IMPORTANT

IF YOU USE await, the async FUNCTION MUST

RETURN A NON-VOID VALUE

void insertData(..) async{ ⇒ NO

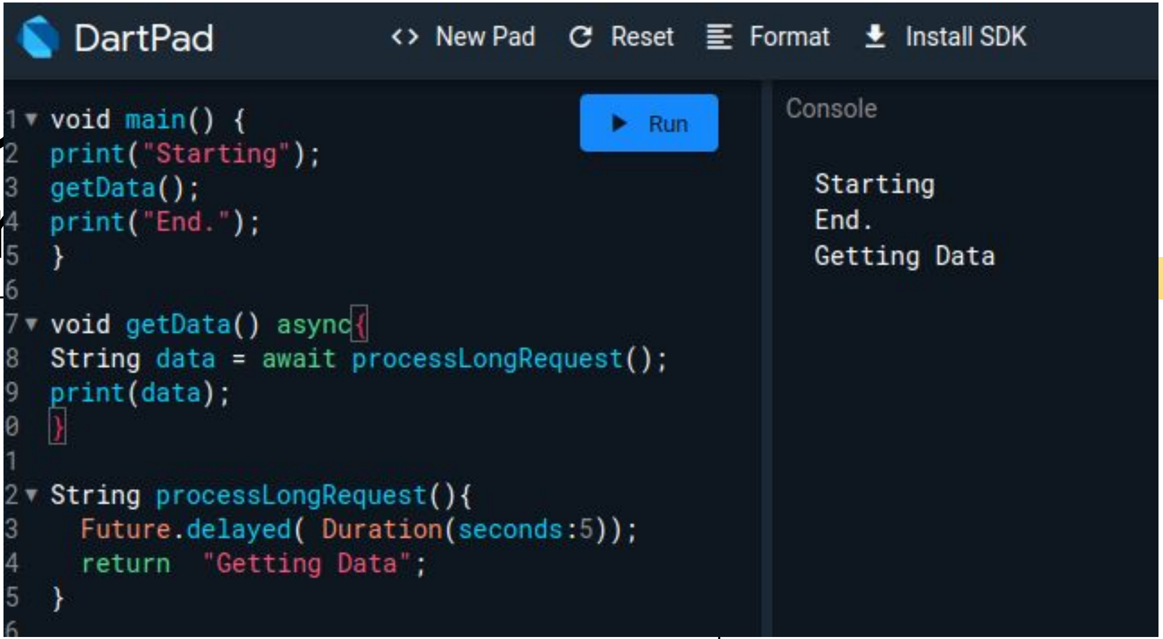
Future<bool> insertData(..) async { ⇒Yes

Writing Async Functions in Flutter

```
void main()  {
      print("Starting");
      getData();
      print("End.");
}

void getData() async{
      String data = await processLongRequest();
      print(data);
}

String processLongRequest(){
      Future.delayed( Duration(seconds:5));
      return "Getting Data" ;
}
```

Writing Async Fu Flutter

```
void main()  {
      print("Starting");
      getData();
      print("End.");
}

void getData() async{
      String data = await processLongRequest();
      print(data);
}

String processLongRequest(){
      Future.delayed( Duration(seconds:5));
      return "Getting Data" ;
}
```



```
void main() {
print("Starting");
getData();
print("End.");
}

void getData() async{
String data = await processLongRequest();
print(data);
}

String processLongRequest(){
  Future.delayed( Duration(seconds:5));
  return  "Getting Data";
}
```

```
Starting
End.
Getting Data
```

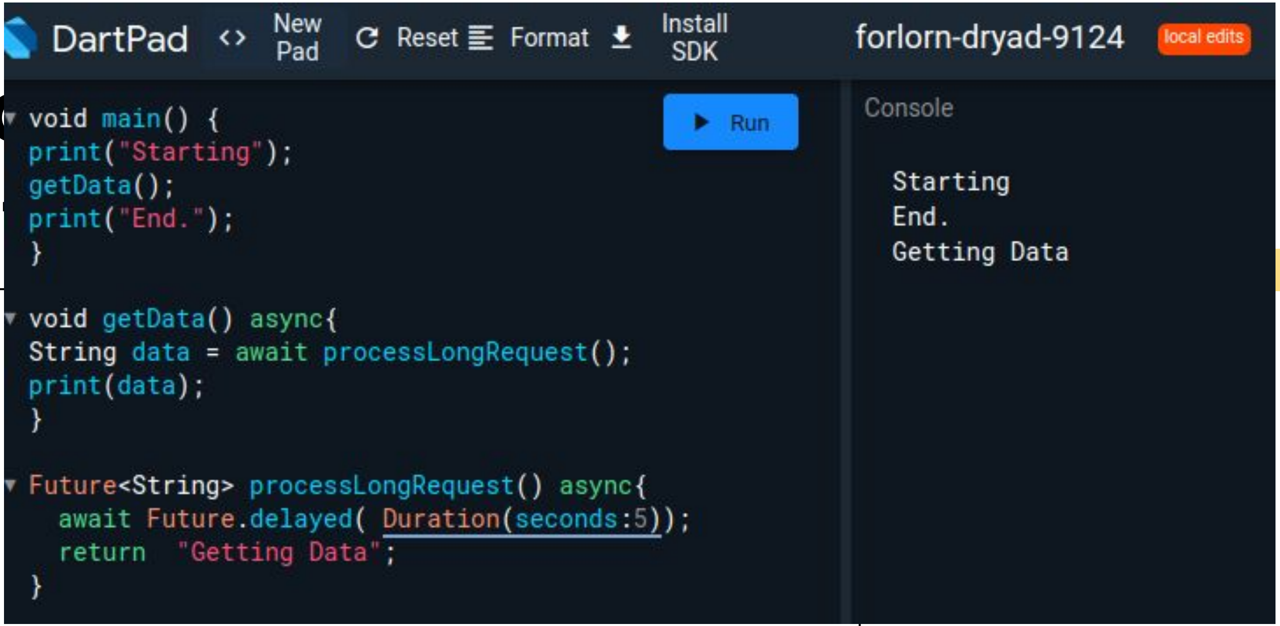
Incorrect Code

Writing Async Functions in Flutter

```
void main() {
      print("Starting");
      getData();
      print("End.");
}

void getData() async{
      String data = await processLongRequest();
      print(data);
}

Future<String> processLongRequest() async{
      await Future.delayed( Duration(seconds:5));
      return "Getting Data" ;
}
```

Writing Async

Flu



```
void main() {
print("Starting");
getData();
print("End.");
}

void getData() async{
String data = await processLongRequest();
print(data);
}

Future<String> processLongRequest() async{
  await Future.delayed( Duration(seconds:5));
  return  "Getting Data";
}
```

```
Starting
End.
Getting Data
```

```
void main()  {
      print("Starting");
      getData();
      print("End.");
}

void getData()  async{
      String data = await processLongRequest();
      print(data);
}

Future<String> processLongRequest()  async{
      await Future.delayed( Duration(seconds:5));
      return "Getting Data"  ;
}
```

Writing Async Functions in Flutter

- **FutureBuilder**
    - It is a special widget for building UI based on data assigned to a Future Object.
    - The widget will monitor the future object, once it has a snapshot of data ready, it will build the widget.

Writing Async Functions in Flutter

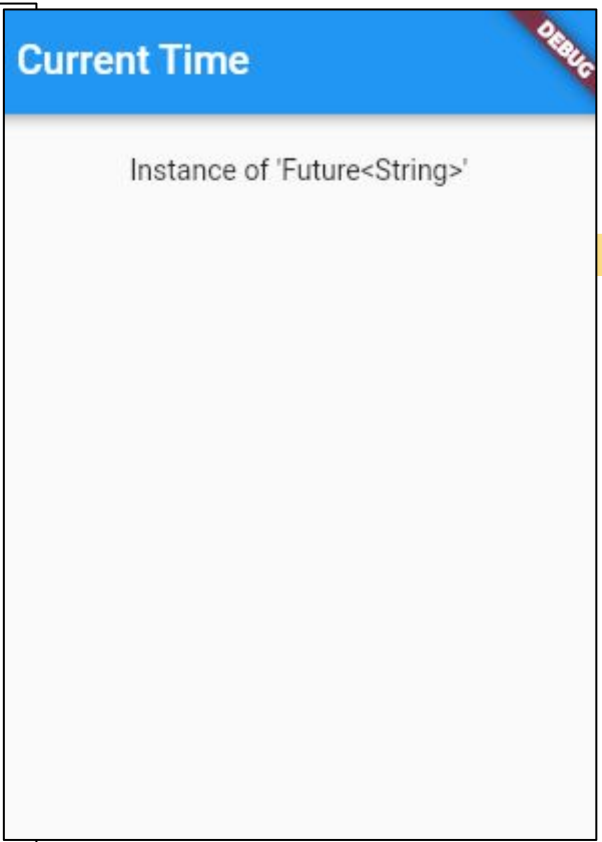
- **FutureBuilder**
  - The main attributes of the future builder :
    - **future** : represents the Future variable whose snapshot can be accessed by the builder function.
    - **builder** : This property represents the current build strategy.
    - **initialData** : represents the data that will be utilized to build the snapshots until a non-null Future has been completed.

Writing Async Functions in Flutter

- **FutureBuilder**
  - Very simple example

```
Future<String> getCurrentTime() async {
  await Future.delayed(Duration(seconds: 5)); // Simulation only
  DateTime time = DateTime.now();
  String strTime =
      "${time.year}-${time.month}-${time.day} ${time.hour}:${time.minute}";
  return strTime;
}
```

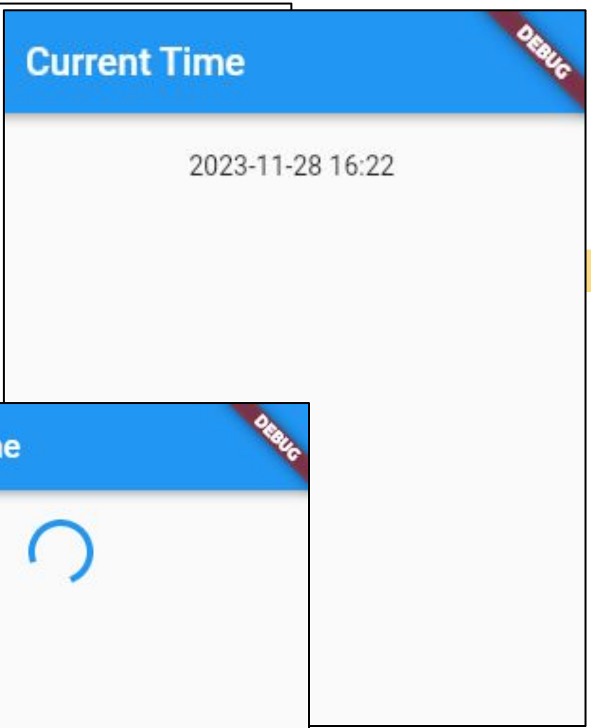
```
import 'package:flutter/material.dart';

class HomeScreen extends StatefulWidget {
 const HomeScreen({super.key});

 @override
 State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
 Future<String> getCurrentTime() async {
   await Future.delayed(Duration(seconds: 5));
   DateTime time = DateTime.now();
   String strTime =
       "${time.year}-${time.month}-${time.day} ${time.hour}:${time.minute}";
   return strTime;
 }

 @override
 Widget build(BuildContext context) {
   Future<String> strTime = getCurrentTime();

   return Scaffold(
     appBar: AppBar(title: Text("Current Time")),
     body: Center(
         child: Column(
       children: [SizedBox(height: 20), Text(strTime.toString())],
     )),
   );
 }
}
```



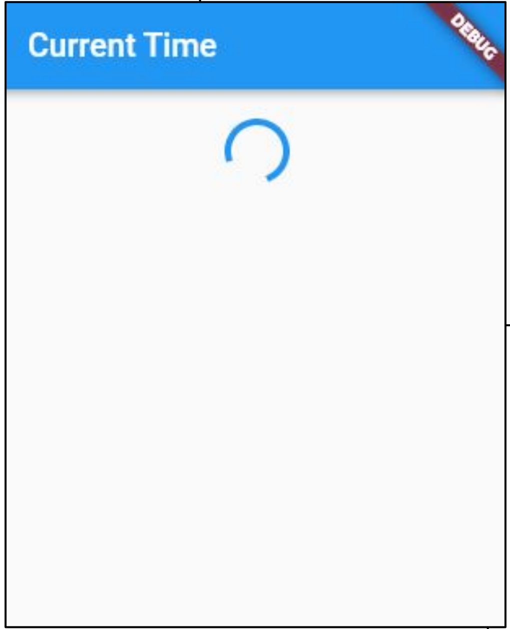
Current Time

Instance of 'Future<String>'

```
...

class _HomeScreenState extends State<HomeScreen> {
 Future<String> getCurrentTime() async {
   await Future.delayed(Duration(seconds: 5));
   DateTime time = DateTime.now();
   String strTime =
       "${time.year}-${time.month}-${time.day} ${time.hour}:${time.minute}";
   return strTime;
 }
 @override
 Widget build(BuildContext context) {
   Future<String> strTime = getCurrentTime();

   return Scaffold(
     appBar: AppBar(title: Text("Current Time")),
     body: Center(
         child: Column(
       children: [
         SizedBox(height: 20),
         FutureBuilder<String>(
             future: strTime,
             builder: (context, snapshot) {
               if (snapshot.hasData) {
                 return Text(snapshot.data!);
               } else if (snapshot.hasError) {
                 return Text("${snapshot.error}");
               }

               return CircularProgressIndicator();
             })
       ],
     )),
   );
 }
}
```


Current Time

2023-11-28 16:22


Current Time

Writing Async Functions in Flutter

- **Example of FutureBuilder using Map**
  - Async Function to return More complex data Structure

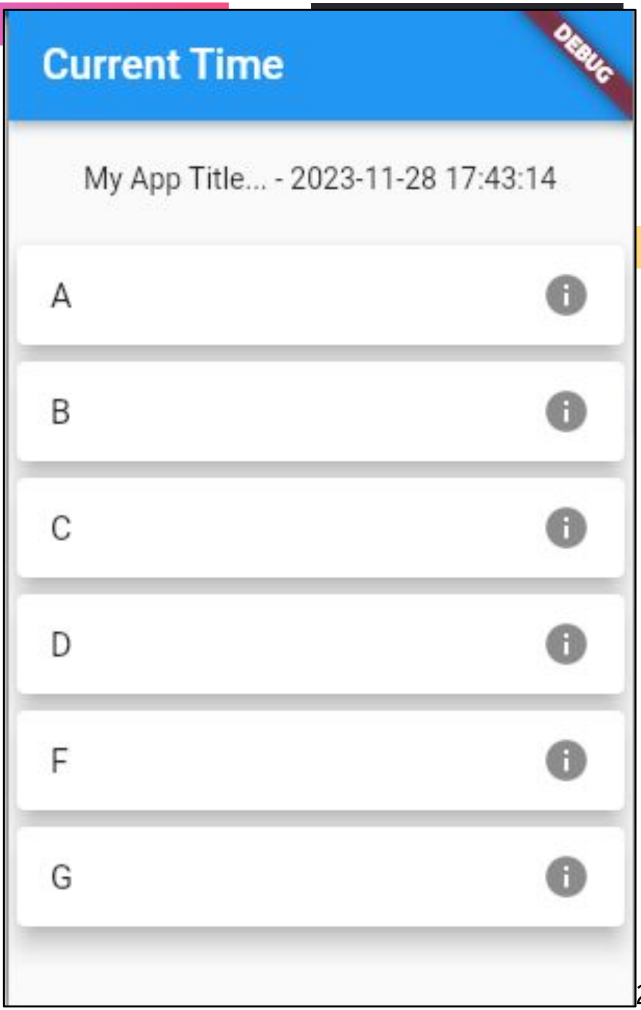
Remember to remove in the live code :

```
  await Future.delayed(Duration(seconds: 5));
```

```
Future<Map<String,dynamic>> getComplexData() async {
  await Future.delayed(Duration(seconds: 5));

  DateTime time = DateTime.now();
  String strTime =
      "${time.year}-${time.month}-${time.day}
${time.hour}:${time.minute}:${time.second}";

  Map<String, dynamic> ret = {'title': 'My App Title...'};
  ret['time'] = strTime;
  ret['items'] = ['A', 'B', 'C', 'D', 'F', 'G'];
  ret['data'] = {
    'line': 1,
    'file': '/img/some.png',
    'children': [1, 2, 3]
  };
  return ret;
}
```

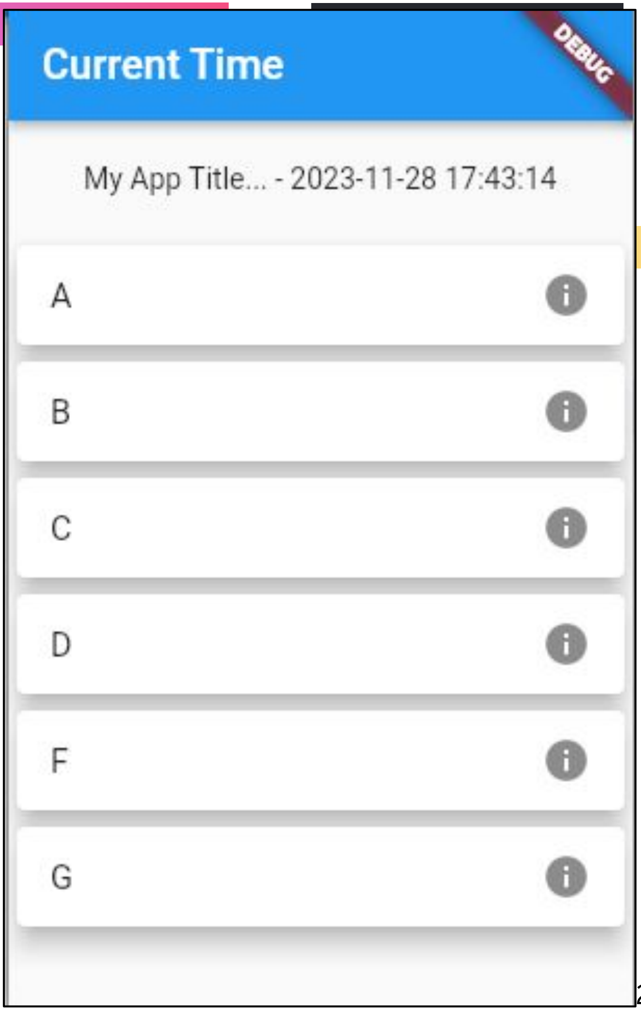
Writing Async Functions in Flutter

- **Example of FutureBuilder using Map**
    - What's the Widget Tree ?



Writing Async Functions in Flutter

- **Example of FutureBuilder using Map**
  - FutureBuilder
    - Column
      - Text
      - ListView.builder
        - Card
          - ListTile
            - Text
            - Icon



My App Title... - 2023-11-28 17:43:14

A

B

C

D

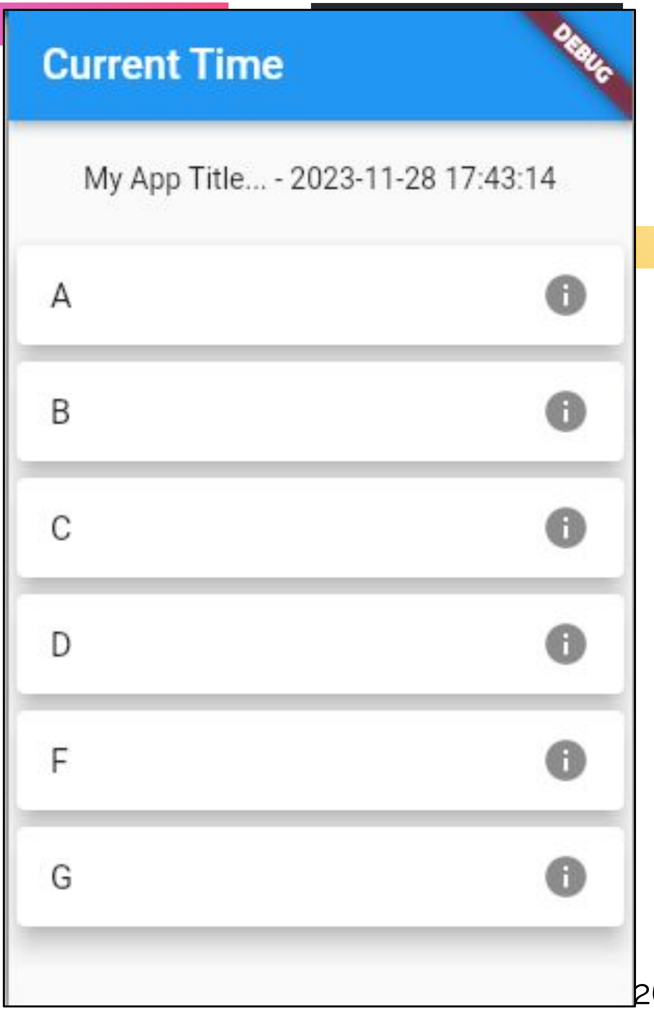
F

G

Writing ons in

Will it work ?

- **Example of FutureBuilder using Map**
  - FutureBuilder
    - **Column**
      - Text
      - **ListView.builder**
        - Card
          - ListTile
            - Text
            - Icon


Current Time

My App Title... - 2023-11-28 17:43:14

A

B

C

D

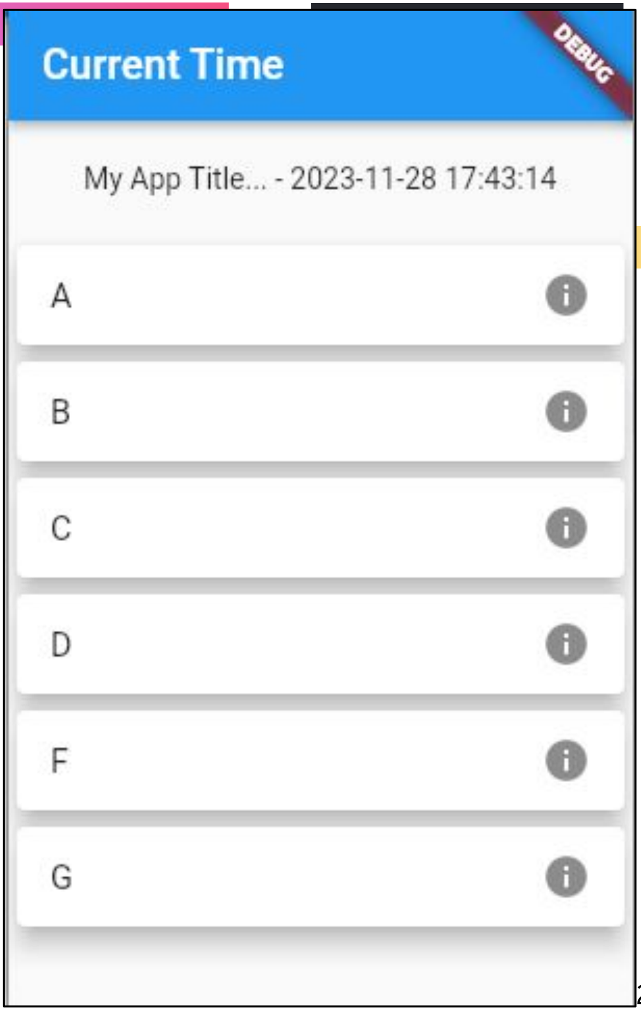
F

G

Writing ons in

Never put a scrollable widget inside another scrollable one

- **Example of FutureBuilder using Map**
  - FutureBuilder
    - **Column**
      - Text
      - **Expanded**
        - **ListView.builder**
          - Card
            - ListTile
              - Text
              - Icon


Current Time

DEBUG

My App Title... - 2023-11-28 17:43:14

A

B

C

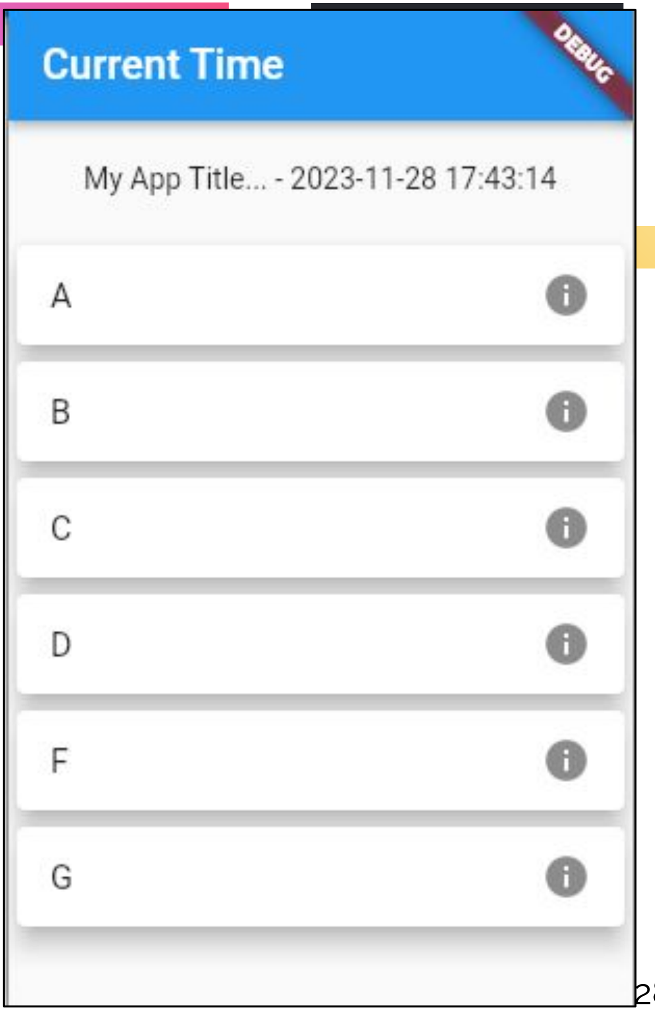
D

F

G

Writing Async Functions in Flutter

```
. . .

 @override
 Widget build(BuildContext context) {
   Future<Map<String, dynamic>> myInfo = getComplexData();
   return Scaffold(
       appBar: AppBar(title: Text("Current Time")),
       body: Center(
           child: FutureBuilder<Map<String, dynamic>>(
               future: myInfo,
               builder: (context, snapshot) {
                 if (snapshot.hasData) {
                   Map myData = snapshot.data!;
                   return getInformationWidget(myData);
                 } else if (snapshot.hasError) {
                   return Text("${snapshot.error}");
                 }

                 return CircularProgressIndicator();
               })));
 }
```


Current Time

My App Title... - 2023-11-28 17:43:14

A

B

C

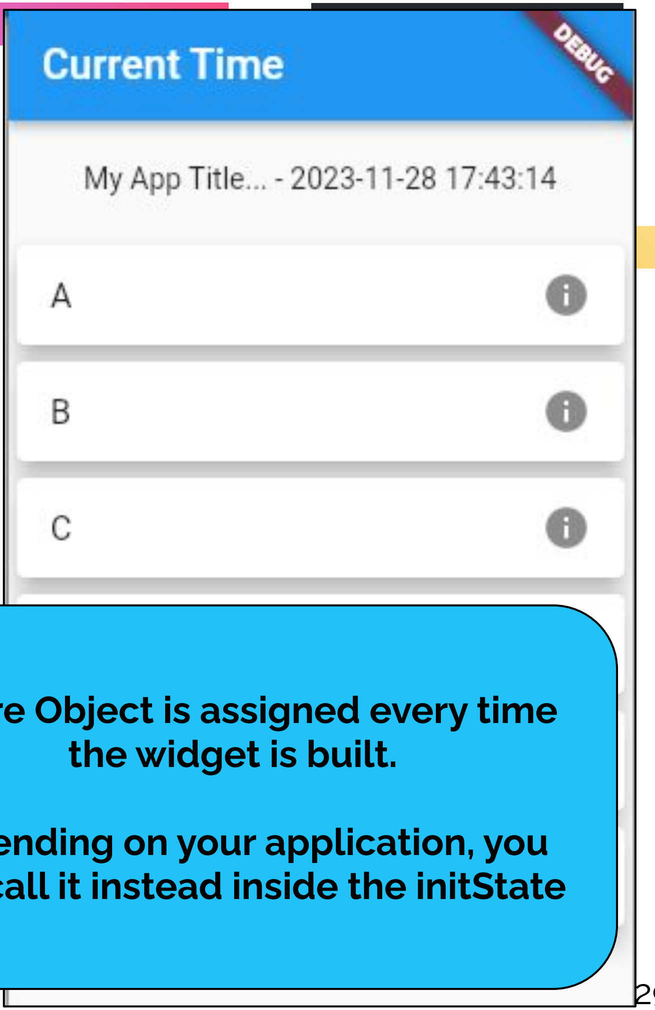
D

F

G

Writing Async Functions in Flutter

```
. . .

 @override
 Widget build(BuildContext context) {
   Future<Map<String, dynamic>> myInfo = getComplexData();
   return Scaffold(
       appBar: AppBar(title: Text("Current Time")),
       body: Center(
           child: FutureBuilder<Map<String, dynamic>>(
               future: myInfo,
               builder: (context, snapshot) {
                 if (snapshot.hasData) {
                   Map myData = snapshot.data!;
                   return getInformationWidget(myData);
                 } else if (snapshot.hasError) {
                   return Text("${snapshot.error}");
                 }

                 return CircularProgressIndicator();
               })));
 }
```


Current Time

My App Title... - 2023-11-28 17:43:14

A

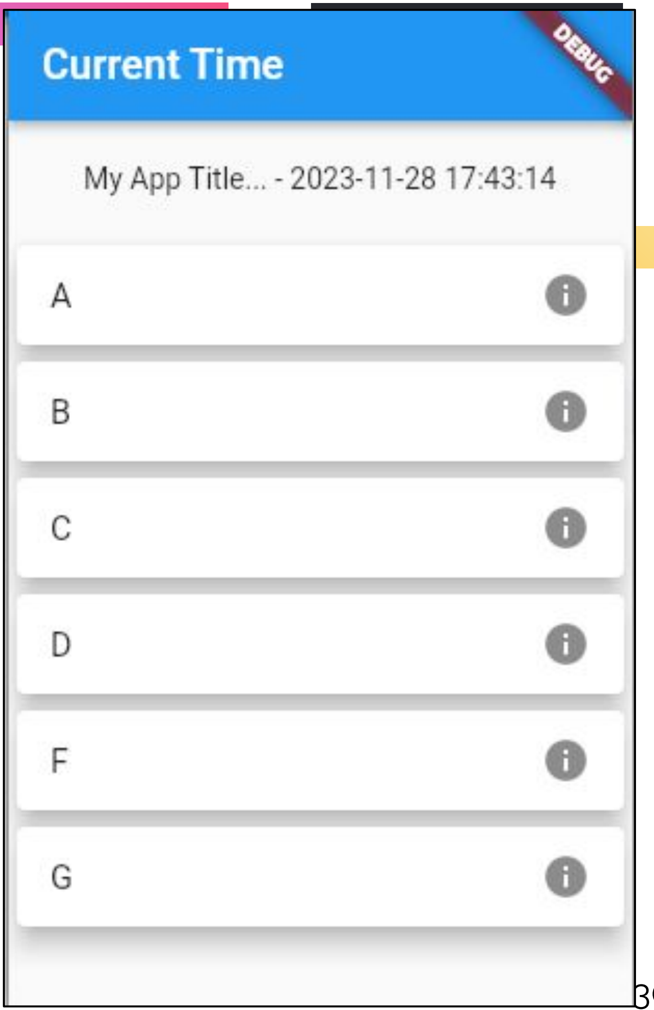
B

C

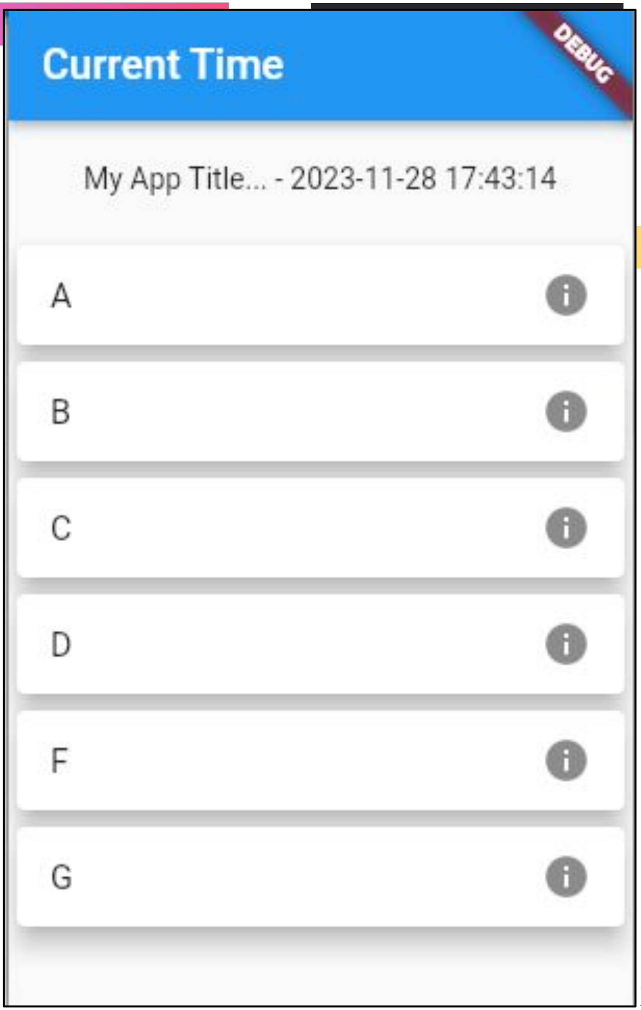
Future Object is assigned every time the widget is built.

Depending on your application, you may call it instead inside the initState

```
String appBarTitle = 'Current Time';
late Future<Map<String, dynamic>> myInfo;

@override
void initState() {
  super.initState();
  myInfo = getComplexData();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(title: Text(appBarTitle)),
      body: Center(
          child: FutureBuilder<Map<String, dynamic>>(
              future: myInfo,
              builder: (context, snapshot) {
                if (snapshot.hasData) {
                  Map myData = snapshot.data!;
                  return getInformationWidget(myData);
                } else if (snapshot.hasError) {
                  return Text("${snapshot.error}");
                }

                return CircularProgressIndicator();
              })));
}
```



```
. . .


 Widget getInformationWidget(Map myData) {

   List items = myData['items'] as List;
   debugPrint(items.toString());
   return Column(
     children: [
       SizedBox(height: 20),
       Text("${myData!['title']} - ${myData!['time']}"),
       SizedBox(height: 20),
       Expanded(
         child: ListView.builder(
           itemCount: items.length,
           itemBuilder: (context, index) {
             return Card(
                 elevation: 10,
                 child: ListTile(
                   title: Text(items[index]),
                   trailing: Icon(Icons.info),
                 ));
           },
         ),
       )
     ],
   );
 }
```


Current Time

My App Title... - 2023-11-28 17:43:14

A

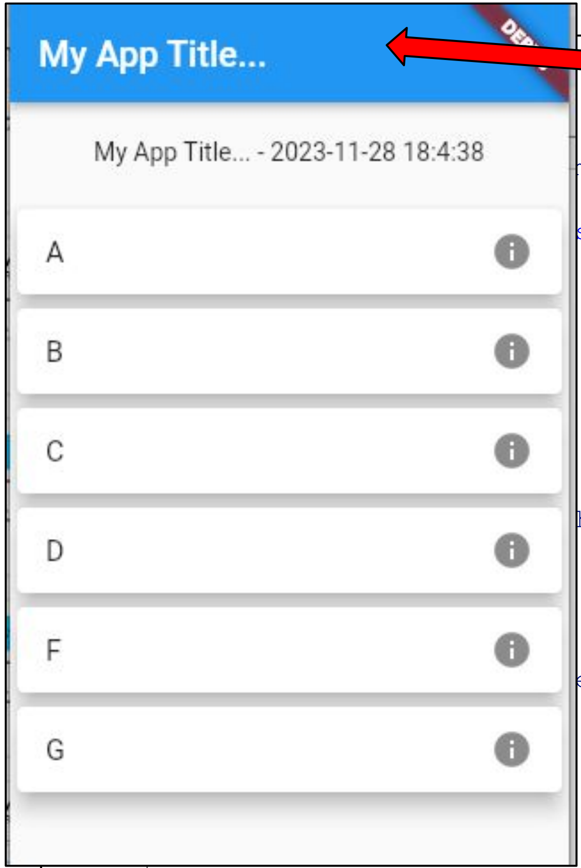
B

C

D

F

G



My App Title...

My App Title... - 2023-11-28 18:4:38

A

B

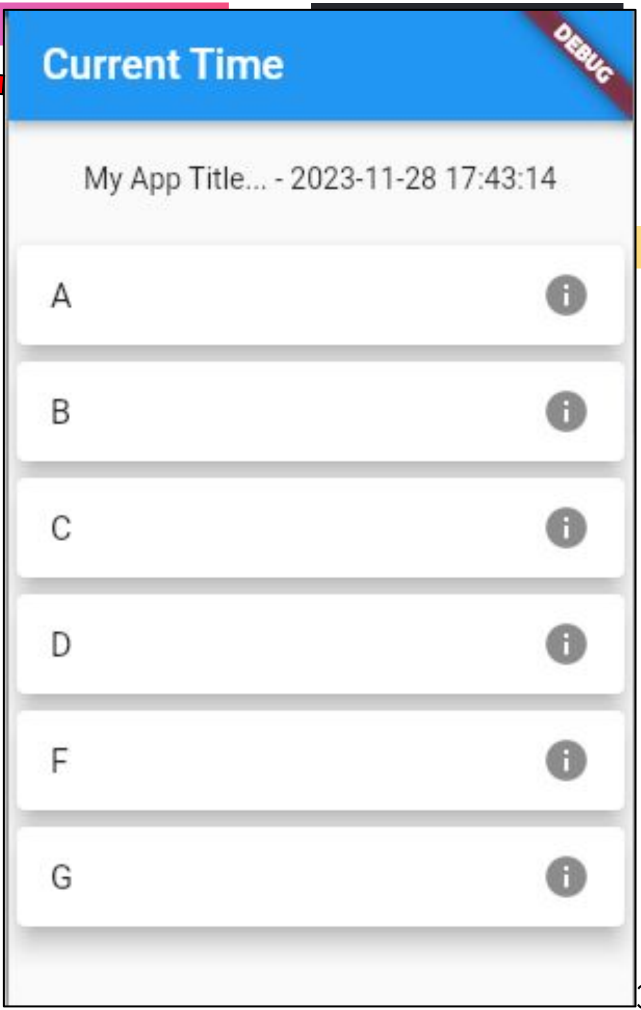
C

D

F

G

```
myData) {
s List;
- ${myData!['time']}"),
h,
i
ems
(Icons.info),
    ],
  );
}
```



Current Time

My App Title... - 2023-11-28 17:43:14

A

B

C

D

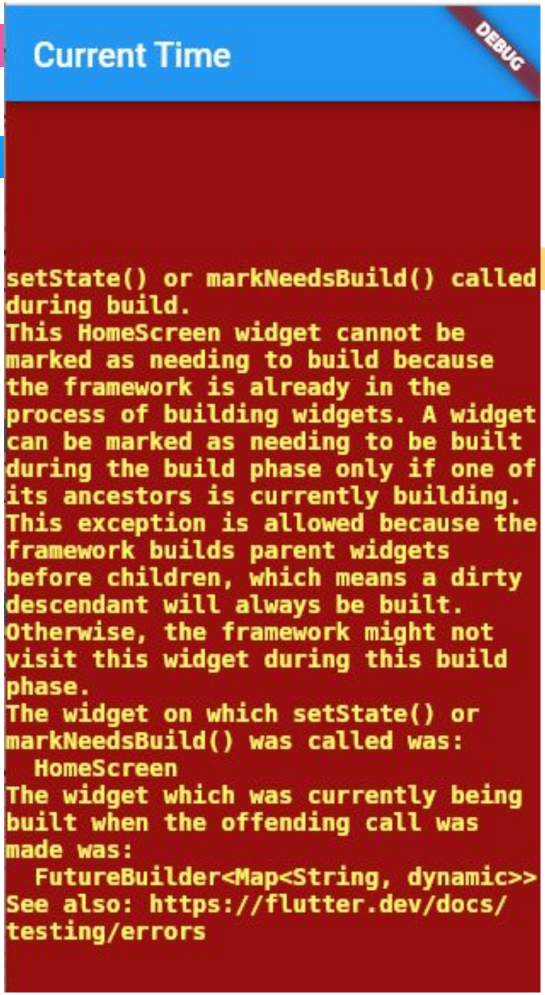
F

G

How to change the AppBar title to take the value mydata['title'] ?

This is the type of exam Questions

```
String appBarTitle = 'Current Time';

@override
Widget build(BuildContext context) {
  Future<Map<String, dynamic>> myInfo = getComplexData();
  return Scaffold(
      appBar: AppBar(title: Text(appBarTitle)),
      body: Center(
          child: FutureBuilder<Map<String, dynamic>>(
              future: myInfo,
              builder: (context, snapshot) {
                if (snapshot.hasData) {
                  Map myData = snapshot.data!;
                  appBarTitle = myData['title'];
                  setState(() {});
```



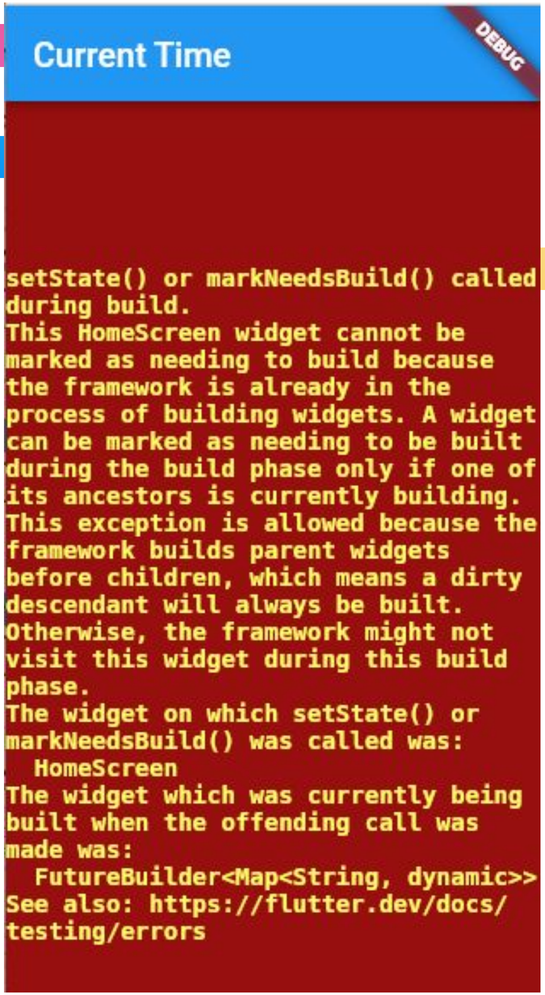
```
setState() or markNeedsBuild() called
during build.
This HomeScreen widget cannot be
marked as needing to build because
the framework is already in the
process of building widgets. A widget
can be marked as needing to be built
during the build phase only if one of
its ancestors is currently building.
This exception is allowed because the
framework builds parent widgets
before children, which means a dirty
descendant will always be built.
Otherwise, the framework might not
visit this widget during this build
phase.
The widget on which setState() or
markNeedsBuild() was called was:
  HomeScreen
The widget which was currently being
built when the offending call was
made was:
  FutureBuilder<Map<String, dynamic>>
See also: https://flutter.dev/docs/
testing/errors
```

```
String appBarTitle = 'Current Time';

@override
Widget build(BuildContext context) {
  Future<Map<String, dynamic>> myInfo = getComplexData();
  return Scaffold(
      appBar: AppBar(title: Text(appBarTitle)),
      body: Center(
          child: FutureBuilder<Map<String, dynamic>>(
              future: m
              builder:
                if (sna
                  Map m
                  appBa
                  setSt
```

You cannot build inside a build

=

You cannot call setState inside Builder-based widgets



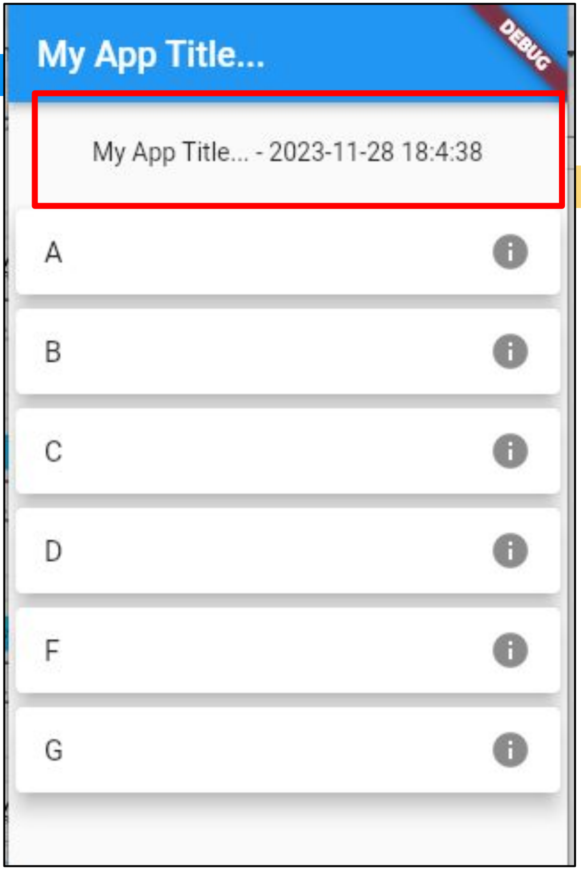
Current Time

```
setState() or markNeedsBuild() called
during build.
This HomeScreen widget cannot be
marked as needing to build because
the framework is already in the
process of building widgets. A widget
can be marked as needing to be built
during the build phase only if one of
its ancestors is currently building.
This exception is allowed because the
framework builds parent widgets
before children, which means a dirty
descendant will always be built.
Otherwise, the framework might not
visit this widget during this build
phase.
The widget on which setState() or
markNeedsBuild() was called was:
  HomeScreen
The widget which was currently being
built when the offending call was
made was:
  FutureBuilder<Map<String, dynamic>>
See also: https://flutter.dev/docs/
testing/errors
```

Writing Async Functions in Flutter

- **Slide to Refresh the time and data**
  - Widget to use :
    - **RefreshIndicator**
      - child: widget
      - onRefresh: function

Widget Available only for Mobile Devices (Not web or desktop app)


My App Title...

My App Title... - 2023-11-28 18:4:38

A

B

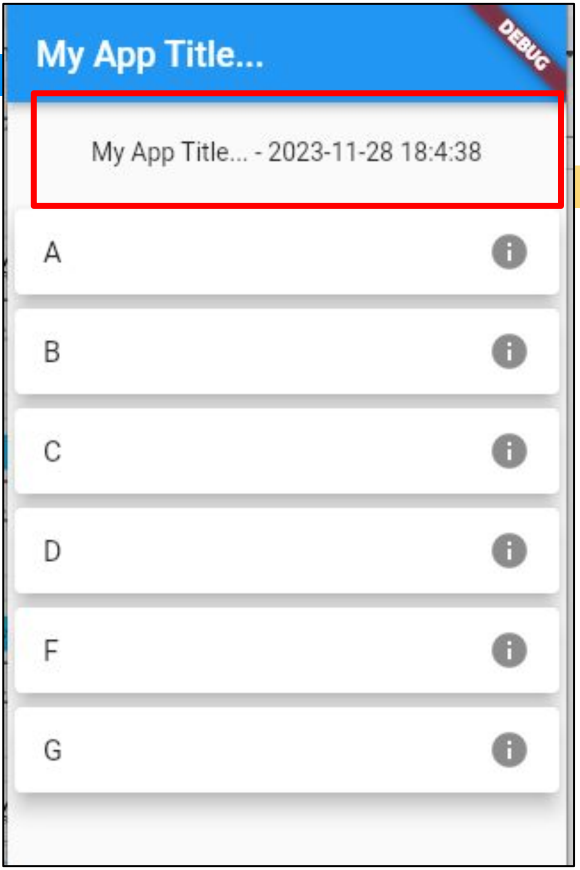
C

D

F

G

```
@override
Widget build(BuildContext context) {
  return Scaffold(
      appBar: AppBar(title: Text(appBarTitle)),
      body: Center(
          child: FutureBuilder<Map<String, dynamic>>(
              future: myInfo,
              builder: (context, snapshot) {
                if (snapshot.hasData) {
                  Map myData = snapshot.data!;
                  return RefreshIndicator(
                          child: getInformationWidget(myData),
                          onRefresh: _refreshMyData
                      );
                } else if (snapshot.hasError) {
                  return Text("${snapshot.error}");
                }

                return CircularProgressIndicator();
              })));
}

Future<void> _refreshMyData() async {
   myInfo = getComplexData();
   setState(() {});
   return;
 }
```


My App Title...

My App Title... - 2023-11-28 18:4:38

A

B

C

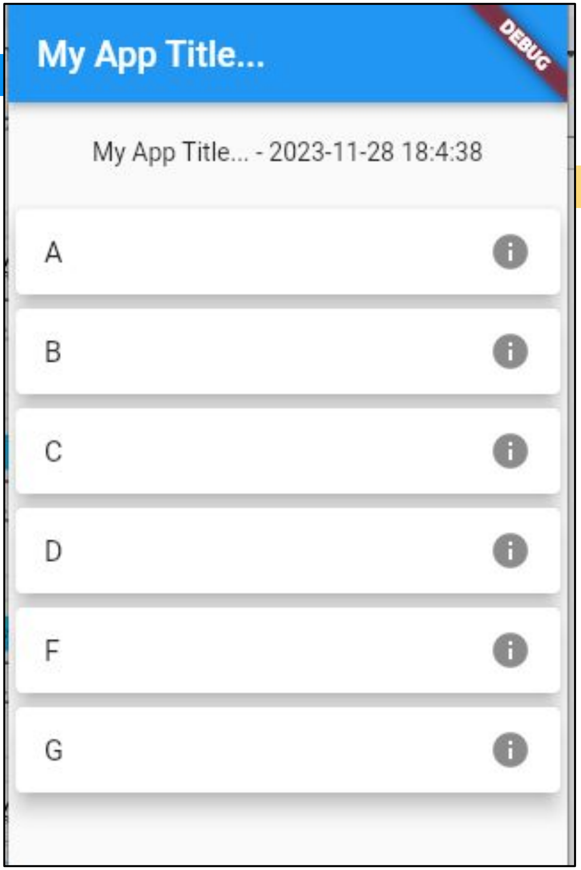
D

F

G

Writing Async Functions in Flutter

- **Automated Update of the Data every 1 minute :**
    - What library to use ?



My App Title...

My App Title... - 2023-11-28 18:4:38

A

B

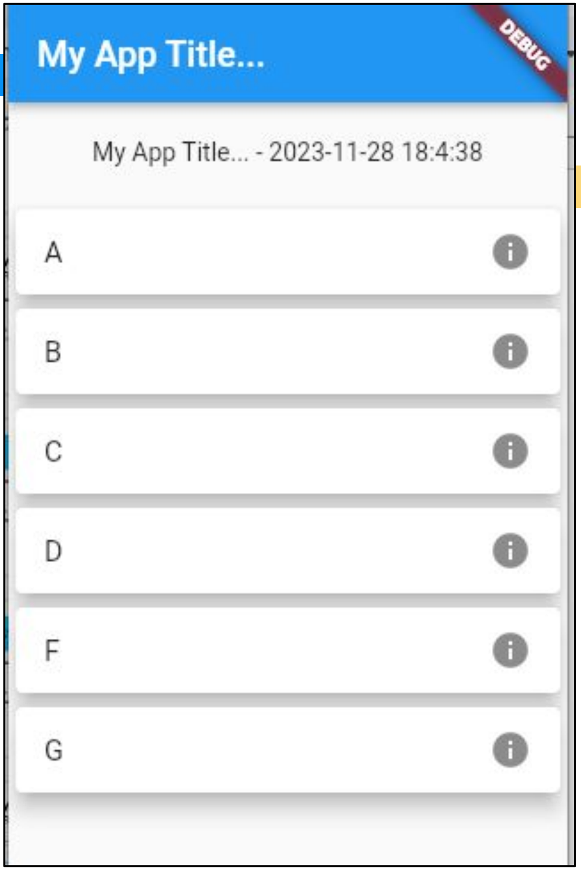
C

D

F

G

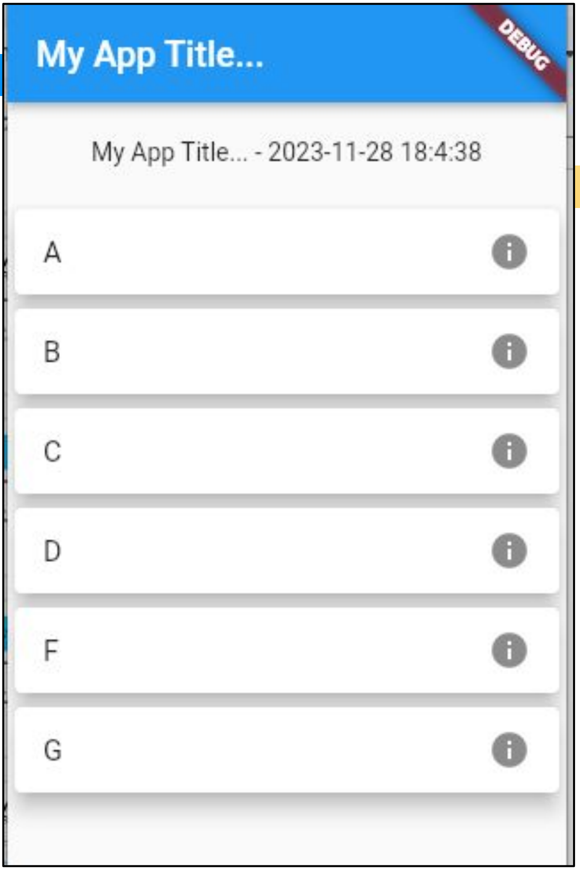
```
import 'dart:async';

...

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

   Timer.periodic(Duration(seconds: 30), (Timer t) {
     print("Getting Data and Update the UI");
     _refreshMyData();
   });

   myInfo = getComplexData();
 }



 Future<void> _refreshMyData() async {
   myInfo = getComplexData();
   setState(() {});
   return;
 }
```



```
import 'dart:async';

...

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

   Timer.periodic(Duration(seconds: 30), (Timer t) {
     print("Getting Data and Update the UI");
     _refreshMyData();
   });

   myInfo = getComplexData();
 }



 Future<void> _refreshMyData() async {
   myInfo = getComplexData();
   setState(() {});
   return;
 }
```

getComplexData is Async ?

setState(() {}) is it redundant ?



My App Title... - 2023-11-28 18:4:38

A

B

C

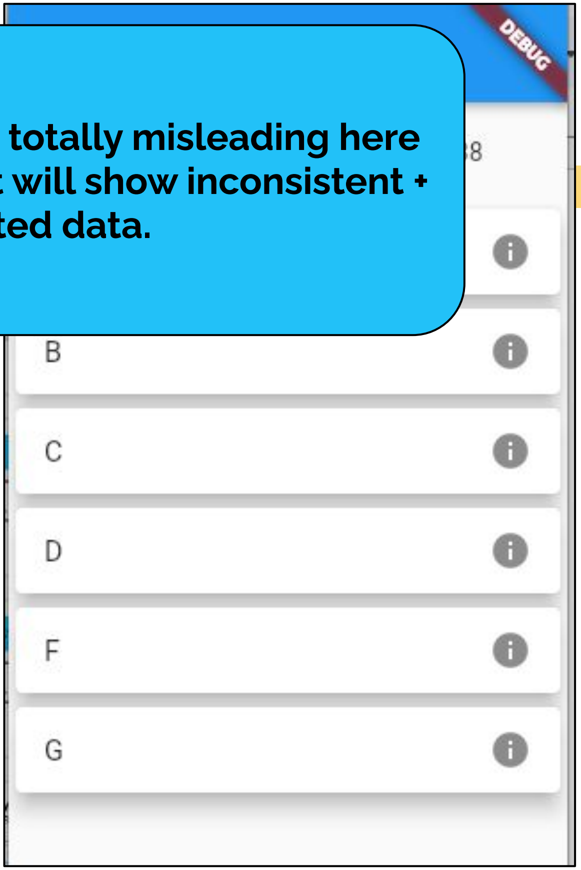
D

F

G

```
import 'dart:async';

...

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

   Timer.periodic(Duration(seconds: 30), (Time
     print("Getting Data and Update the UI");
     _refreshMyData();
   });

   myInfo = getComplexData();
 }



 Future<void> _refreshMyData() async {
   myInfo = getComplexData();
   setState(() {});
   return;
 }
```

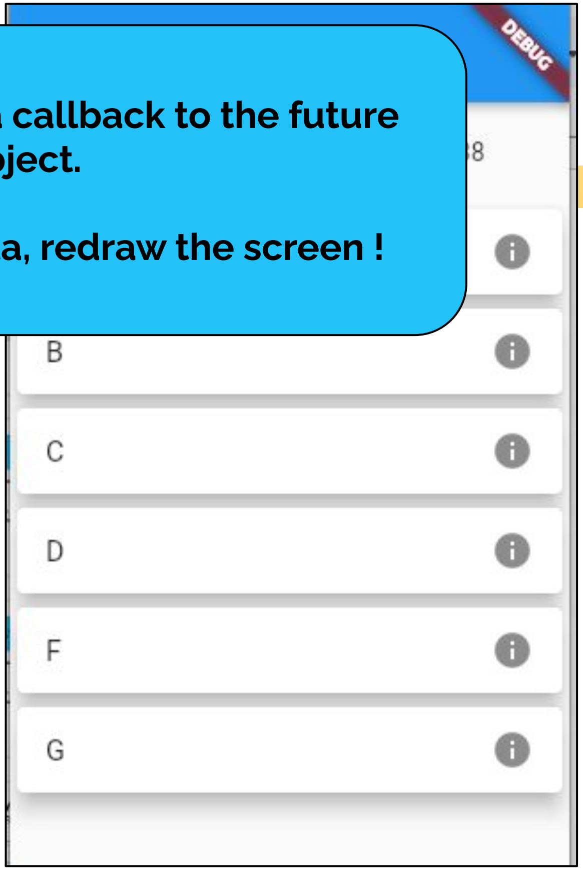
The use of setState is totally misleading here as it does nothing or it will show inconsistent + outdated data.



```
import 'dart:async';

...

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

   Timer.periodic(Duration(seconds: 30), (Time
     print("Getting Data and Update the UI");
     _refreshMyData();
   });

   myInfo = getComplexData();
 }



 Future<void> _refreshMyData() async {
   myInfo = getComplexData();
   myInfo.then((_) => setState(() {}));
   return;
 }
```

We are registering a callback to the future object.

When you have data, redraw the screen !



```
import 'dart:async';

...

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

   Timer.periodic(Duration(seconds: 30), (Time
     print("Getting Data and Update the UI");
     _refreshMyData();
   });

   myInfo = getComplexData();
 }



 Future<void> _refreshMyData() async {
   myInfo = getComplexData();
   myInfo.then((_) => setState(() {}));
   return;
 }
```

The same for the RefreshIndicator too !

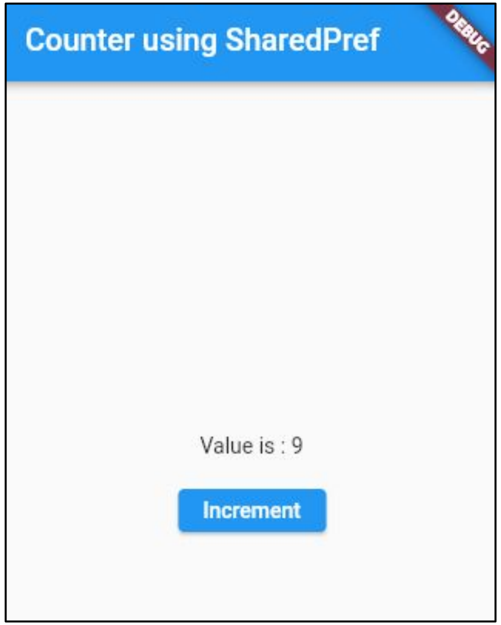
Writing Async Functions in Flutter

- **Automated Update of the Data from the backend :**
    - Frequency : every 1 minute ? 5 minutes ?
    - Does the update involves getting data from a server ?
    - How many active apps do they connect to the server to get recent data ?

Writing Async Functions in Flutter

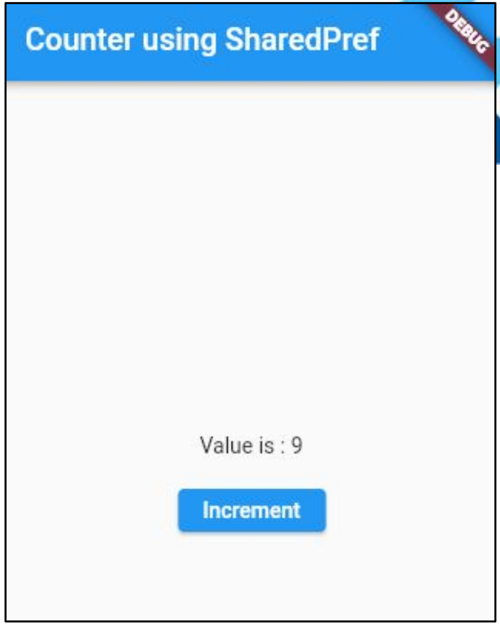
- **Incrementing App**
  - Created using Kotlin :
    - *Simple*
    - *Automated*
    - *SharedPreferences*
    - *Auto with Variable Speed*
  - Using Flutter :
    - *Simple*
    - *Two Screens*
    - *One Screen with two separate widgets*
    - *For Automated use Timer...*

Writing Async Functions in Flutter



Mobile A

How to keep the incremented value **synched** across different devices ?



Mobile B

Writing Async Functions in Flutter

- **Automated Update of the Data**
  - Frequent and Recurring function can :
    - Consume quickly the battery
    - Overheat the phone
  - If you have a large number of active users where the apps communicate to the server (Suppose over 100K active apps):
    - The server will crash.
    - You will end up paying a large amount of money.

Section 2

Data Persistence using Flutter

Data Persistence using Flutter

- **Reminder from Kotlin lectures**
  - Data can be stored for mobile apps using :
    - Shared Preferences
    - Local Databases
    - As Files in the filesystem
    - Cloud Services :
      - Firebase (To be seen fully with Flutter)
      - AWS + …

Data Persistence using Flutter

- **Shared Preferences in Flutter**
  - It is a way to store primitive data in the form **key:value** using the class ***SharedPreferences***
  - It is recommended to use it for small data
  - Available for Mobile, Web and Desktop Apps
  - https://pub.dev/packages/shared_preferences
  - Hive is a similar popular library
    - https://pub.dev/packages/hive

Data Persistence using Flutter

- **Shared Preferences in Flutter**

1. To Add the Dependency :

   flutter pub add shared_preferences

2. Load the sharedPreference instance

```
final SharedPreferences prefs = await SharedPreferences.getInstance();
```

Data Persistence using Flutter

- **Shared Preferences in Flutter**
  1. To Add the Dependency :

     flutter pub add shared_preferences

  2. Load the sharedPreference instance

```
final SharedPreferences prefs = await SharedPreferences.getInstance();
```

Personally, I prefer it as STATIC

Where to place this line of code ?

+

It has await ?

Data Persistence using Flutter

```
class _HomeScreenState extends State<HomeScreen> {
 int increment = 0;
 static late final SharedPreferences prefs;

 @override
 void initState() {
   super.initState();
   myInitOperations();
 }
 Future<void> myInitOperations() async {
   prefs = await SharedPreferences.getInstance();
   …
 }
```

Personally, I prefer it as STATIC

Where to place this line of code ?

+

It has await ?

```
es.getInstance();
```

Data Persistence using Flutter

- **Shared Preferences in Flutter**

  3. Write Data

```dart
await prefs.setInt('counter', 10);
await prefs.setBool('repeat', true);
await prefs.setDouble('decimal', 1.5);
await prefs.setString('action', 'Start');
await prefs.setStringList('items', <String>['Earth', 'Moon', 'Sun']);
```

Data Persistence using Flutter

How to save Map Object to SharedPreferences ?

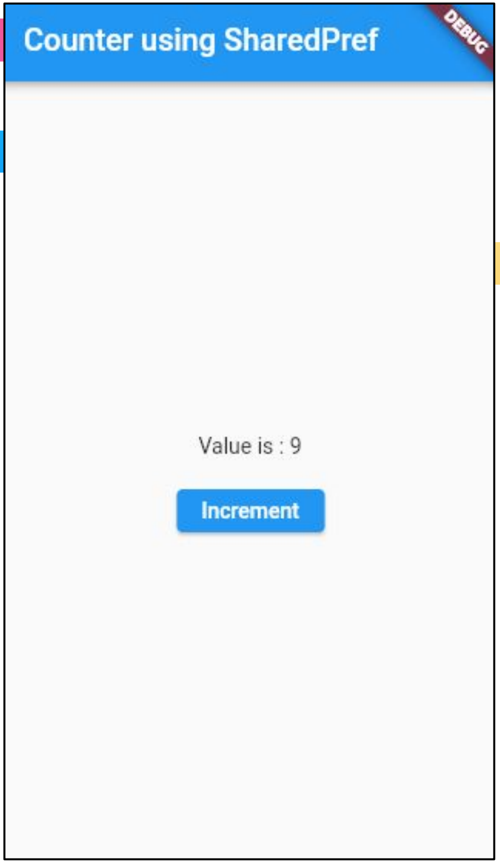
- **Shared Preferences in Flutter**

3. Write Data

```
await prefs.setInt('counter', 10);
await prefs.setBool('repeat', true);
await prefs.setDouble('decimal', 1.5);
await prefs.setString('action', 'Start');
await prefs.setStringList('items', <String>['Earth', 'Moon', 'Sun']);
```

Data Persistence using Flutter

```
@override
Widget build(BuildContext context)  {
  return Scaffold(
      appBar: AppBar(title: Text("Counter using SharedPref ")),
      body: Center(
          child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
        children: [
          Text("Value is : ${increment}"),
          SizedBox(height: 20),
          ElevatedButton(

              onPressed: () async {
                increment = increment + 1;
                await prefs.setInt("incrementNumber", increment);
                setState(() {});
              },

              child: Text("Increment"))
        ],
      )));
}
```



Data Persistence using Flutter

- **Shared Preferences in Flutter**

  4. Read Data

```
final int? counter = prefs.getInt('counter');
final bool? repeat = prefs.getBool('repeat');
final double? decimal = prefs.getDouble('decimal');
final String? action = prefs.getString('action');
final List<String>? items = prefs.getStringList('items');
```

Data Persistence using Flutter

- **Shared Preferences in Flutter**

4. Read Data

```
class _HomeScreenState extends State<HomeScreen> {
 int increment = 0;
 static late final SharedPreferences prefs;
 @override
 void initState() {
   super.initState();
   myInitOperations();
 }
 Future<void> myInitOperations() async {
   prefs = await SharedPreferences.getInstance();
   increment = prefs.getInt("incrementNumber") ?? 0;
   setState(() {});
 }
```

Data Persistence using Flutter

- **Shared Preferences in Flutter**

  5. Remove Data

```dart
// Remove data for the 'counter' key.

await prefs.remove('counter');
```

Data Persistence using Flutter

- **Shared Preferences in Flutter**
    - How frequent to save :
        - Recommended After each modification of a variable
        - On closing the App ? depending how critical is the data

Data Persistence using Flutter

- **Relational Databases : SQLite (Slide from W5)**
  - SQLite is a well-regarded SQL-based relational database management system (RDBMS). It is
    - Open source
    - Standards-compliant, implementing most of the SQL standard
    - Lightweight
    - Single-tier
    - ACID compliant

Data Persistence using Flutter

- **Relational Databases : SQLite:**
    - SQLite is implemented as a compact C library that's included as part of the Android software stack
    - Each SQLite database is an integrated part of the application that created it. This reduces external dependencies, minimizes latency, and simplifies transaction locking and synchronization.

Data Persistence using Flutter

01

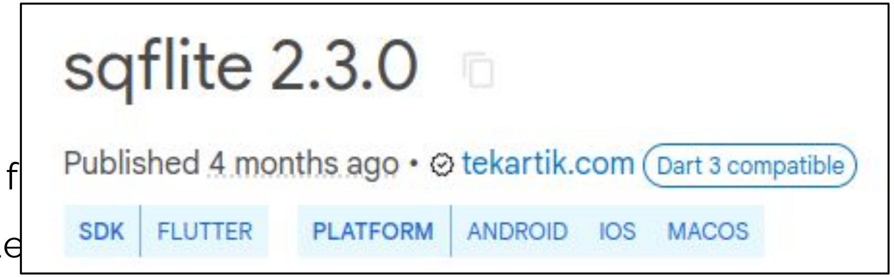
- **Steps to started : Plugins & Packages**
  - SQFLite is the flutter plugin for using SQLite
    - https://pub.dev/packages/sqflite
  - To get started install:
    - sqflite : package provides classes and functions to interact with a SQLite database.
    - path : package provides functions to define the location for storing the database on disk.
      - **flutter pub add sqflite path**
  - Import the packages:

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';
```

Data Persistence using Flutter

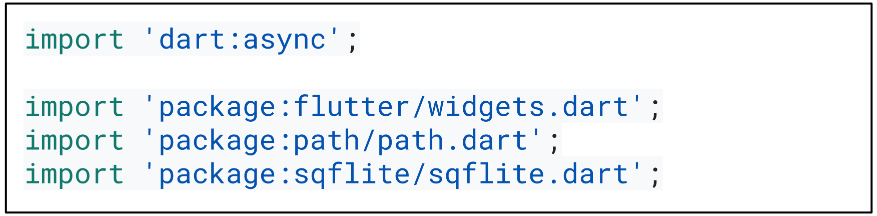
01

- **Steps to started : Plugins & Packages**
  - SQFLite is the flutter plugin for using SQLite
    - https://pub.dev/packages/sqflite
  - To get started install:
    - sqflite : package provides classes and f
    - path : package provides functions to de
      - **flutter pub add sqflite path**
  - Import the packages:



sqflite 2.3.0

Published 4 months ago • tekartik.com Dart 3 compatible

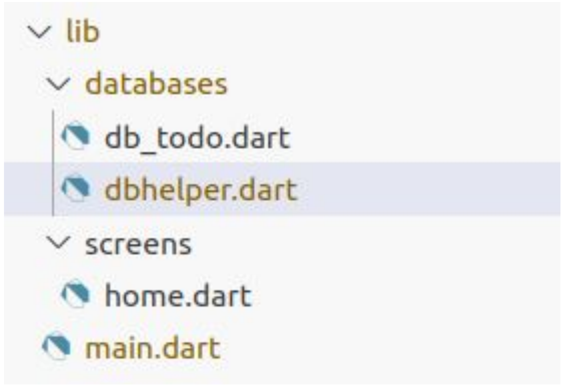


```
import 'dart:async';

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

Data Persistence using Flutter

02

- **Steps to started : DBHelper Class**
  - Create a folder **databases** inside **lib**
  - Create **dbhelper.dart** file :
    - Shall contain :
      - SQL Database name
      - SQL Database version
      - SQL Code for creating/upgrading the tables.
      - Singleton Method to get an instance of the database



```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';

class DBHelper {
 static const _database_name = "ENSIA_MY_DB.db";
 static const _database_version = 4;
 static var database;

 static Future getDatabase() async {
   if (database != null) {
     return database;
   }
   database = openDatabase(
     join(await getDatabasesPath(), _database_name),
     onCreate: (database, version) {
       database.execute('''
         CREATE TABLE  todo (
             id INTEGER PRIMARY KEY AUTOINCREMENT,
             title TEXT,
             done INTEGER,
             duedate TEXT,
             create_date TEXT)
        ''');
     },
     version: _database_version,
     onUpgrade: (db, oldVersion, newVersion) {      },
   );
   return database;
 }
}
```

02

Data Persistence using Flutter

03

- **Steps to started : DB Utility Class for each Model**
    - Create a folder **databases** inside **lib**
    - Create **db_todo.dart** file :
        - It will contain helper methods to :
            - Get data from the related table/model
            - Insert a todo item
            - Remove todo item
            - Flag Done

```dart
import 'package:sqflite/sqflite.dart';
import 'databases/dbhelper.dart';
class TodoDB {
 static Future<List<Map<String, dynamic>>> getAllToDos() async {
   final database = await DBHelper.getDatabase();
   return database.rawQuery('''SELECT
           todo.id ,
           todo.title,
           todo.done
         from todo
         ''');
 }
 static Future insertToDo(Map<String, dynamic> data) async {
   final database = await DBHelper.getDatabase();
   database.insert("todo", data, conflictAlgorithm: ConflictAlgorithm.replace);
 }
 static void deleteToDo(int id) async {
   final database = await DBHelper.getDatabase();
   database.rawQuery("""delete from  todo where id=?""", [id]);
 }
 static void setDone(int id, bool flag) async {
   final database = await DBHelper.getDatabase();
   int value = flag ? 1 : 0;
   database.rawQuery("""update  todo set done=? where id=?""", [value, id]);
 }
}
```

03

Data Persistence using Flutter

04

- **Steps to started : Use Future Objects & FutureBuilder**
  - In the old for the To-do App (Week 6)

```
class _HomeScreenState extends State<HomeScreen> {
 List<Map> data = [
   {'title': 'My first to do', 'done': false}
 ];
 String _tx_title_value = '';
 final _tx_title_controller = TextEditingController();
 @override
 Widget build(BuildContext context) {
   return Scaffold(
              ...
```

Data Persistence using Flutter

04

- **Steps to started : Use Future Objects & FutureBuilder**
  - In the old for the To-do

```
class _HomeScreenState extends State<Hom
  List<Map> data = [
    {'title': 'My first to do', 'done': f
  ];
  String _tx_title_value = '';
  final _tx_title_controller = TextEditi
  @override
  Widget build(BuildContext context) {
    return Scaffold(
            ...
```

```
class _HomeScreenState extends State<HomeScreen> {
 late Future<List<Map>> data;

 String _tx_title_value = '';
 final _tx_title_controller = TextEditingController();

 @override
 Widget build(BuildContext context) {
   data = TodoDB.getAllToDos();
   return Scaffold(
```

Data Persistence using Flutter

04

- **Steps to started : Use Future Objects & FutureBuilder**
  - Wrap the widget of the listview.builder inside a function

```
Widget getListToDoWidget(List myData) {
   return ListView.builder(
     itemCount: myData.length,
     itemBuilder: (context, index) {
       return Card(
         elevation: 4,
         margin: const EdgeInsets.all(8),
         shape: RoundedRectangleBorder(
           borderRadius: BorderRadius.circular(10.0),
         ),
         child: ListTile(
           title: Text(myData[index]['title']),
```

Data Persistence using Flutter

04

- **Steps to started : Use Future Objects & FutureBuilder**
  - Integrate FutureBuilder

```
const Text('My to-do tasks !'),
const SizedBox(height: 20),
Expanded(
    child: FutureBuilder<List>(
        future: data,
        builder: (context, snapshot) {
          if (snapshot.hasData) {
            return getListToDoWidget(snapshot.data!);
          } else if (snapshot.hasError) {
            return Text("${snapshot.error}");
          }

          return CircularProgressIndicator();
        })),
```

Data Persistence using Flutter

05

- **Steps to started : Link DB with the Widgets**
  - To add a ToDo :

```
const SizedBox(width: 10),
ElevatedButton(
  onPressed: () {
    _tx_title_controller.text = '';
    TodoDB.insertToDo({'title': _tx_title_value, 'done': 0});
    setState(() {});
  },
  child: const Text(
    "Add",
  ),
```

Data Persistence using Flutter

05

- **Steps to started : Link DB with the Widgets**
  - To Remove an item:

```
trailing: IconButton(
  icon: Icon(
    Icons.delete,
  ),
  onPressed: () {
    setState(() {
      TodoDB.deleteToDo(myData[index]['id']);
    });
  },
),
```

Data Persistence using Flutter

06

- **Steps to started : Optional ! Use more OOP**
  - Create a folder **models** inside **lib**
    - Create a Model for each table
      - Define the attribute as instance variable
      - Create the converter functions to Map (and even statically from a Map)

Data Persistence using Flutter

06

```
class Todo {
 final int id;
 final String title;
 final bool done;

 const Todo({
   required this.id,
   required this.title,
   required this.done,
 });
 Map<String, dynamic> toMap() {
   return {
     'id': id,
     'title': title,
     'done': done,
   };
 }
 @override
 String toString() {
   return 'Todo{id: $id, title: $title, done: $done}';
 }
}
```

Data Persistence using Flutter

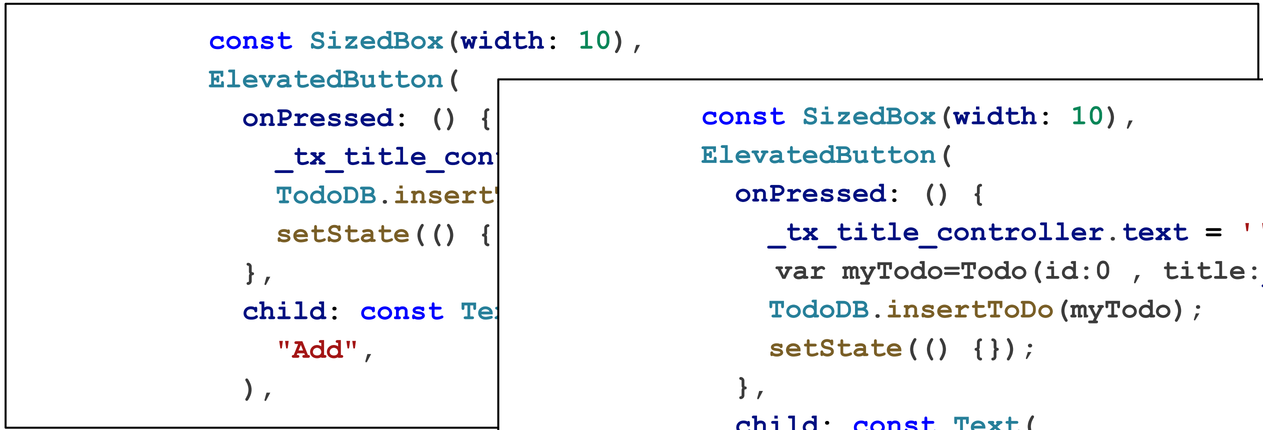
06

- **Steps to started : Optional ! Use more OOP**
  - Integrate the Model Todo class into the database helper functions

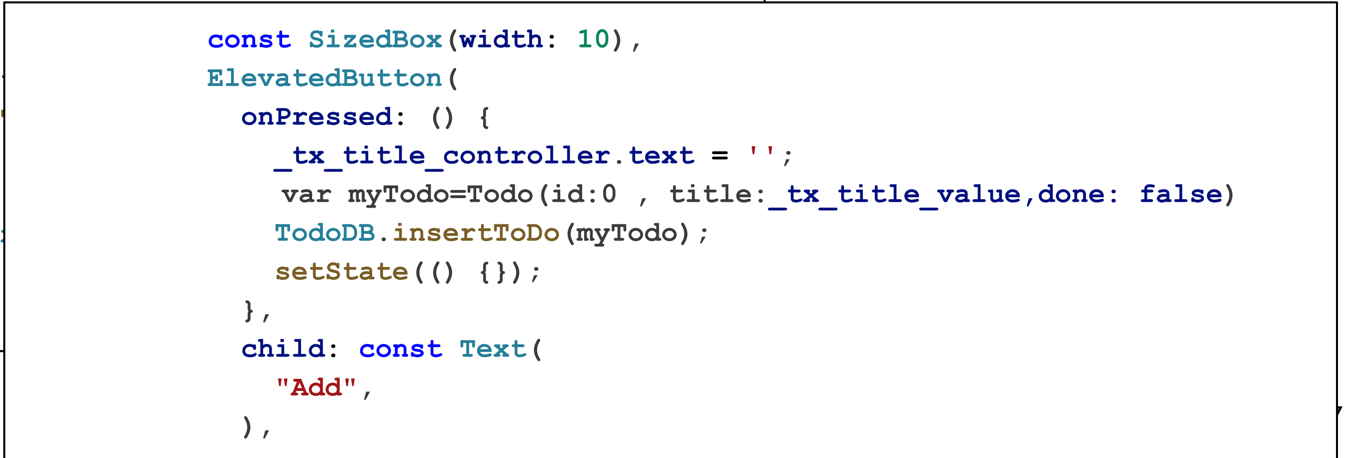
```
        const SizedBox(width: 10),
        ElevatedButton(
          onPressed: () {
            _tx_title_controller.text = '';
            TodoDB.insertToDo({'title': _tx_title_value, 'done': 0});
            setState(() {});
          },
          child: const Text(
            "Add",
          ),
```

Data Persistence using Flutter

- **Steps to started : Optional ! Use more OOP**
  - Integrate the Model Todo class into the database helper functions



```
const SizedBox(width: 10),
ElevatedButton(
  onPressed: () {
    _tx_title_con
    TodoDB.insert
    setState(() {
  },
  child: const Te
    "Add",
  ),
```



```
const SizedBox(width: 10),
ElevatedButton(
  onPressed: () {
    _tx_title_controller.text = '';
     var myTodo=Todo(id:0 , title:_tx_title_value,done: false)
    TodoDB.insertToDo(myTodo);
    setState(() {});
  },
  child: const Text(
    "Add",
  ),
```

Data Persistence using Flutter

06

- **Steps to started : Optional ! Use more OOP**
  - Integrate the Model Todo class into the database helper functions

```
static Future insertToDo(Map<String, dynamic> data) async {
  final database = await DBHelper.getDatabase();
  database.insert("todo", data, conflictAlgorithm: ConflictAlgorithm.replace);
}
```

⇩

```
static Future insertToDo(Todo todo) async {
  final database = await DBHelper.getDatabase();
  database.insert("todo", todo.toMap() , conflictAlgorithm: ConflictAlgorithm.replace);
}
```

Section 3

Data Synching Techniques

Data Synching between Apps and Services

- **Concepts and Techniques**
  - **Offline Use :**
    - Most mobile apps need to have a local database for the case of offline use :
      - Users can save data, add photos…
        - Data is saved locally in SQLite
        - When there is an internet connection, data are uploaded to the backend or other external services.

Data Synching between Apps and Services

- **Concepts and Techniques**
  - **Implement Caching Techniques :**
    - Images or large static data, store them at a local cache (Database, file system) in the same way as browsers to reduce the number of future connections/bandwidth.

Data Synching between Apps and Services

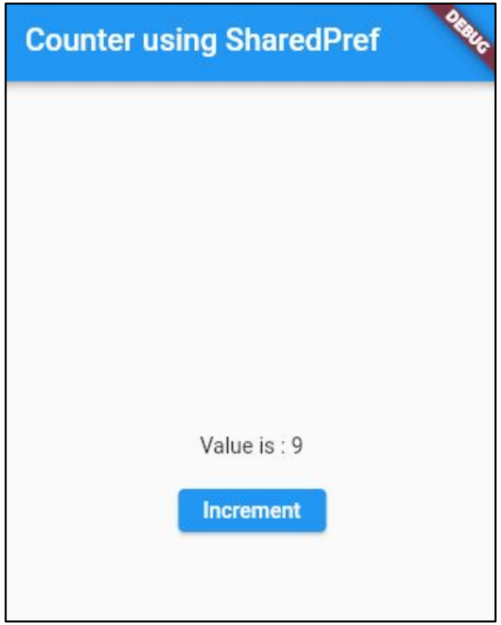
- **Concepts and Techniques**
  - **Sync Always in the background to the Backend Servers :**
    - Either doing full-sync or incremental Sync, Try to do it in the background using Cron (Or other plugins) :
  - There are cases where you may sync on the user's requests

```
Future<void> main() async {
 ...
 final cron = Cron();
 cron.schedule(Schedule.parse('*/5 * * * *'), () async {
   actionUploadImageUpload();
 });
```

Data Synching between Apps and Services

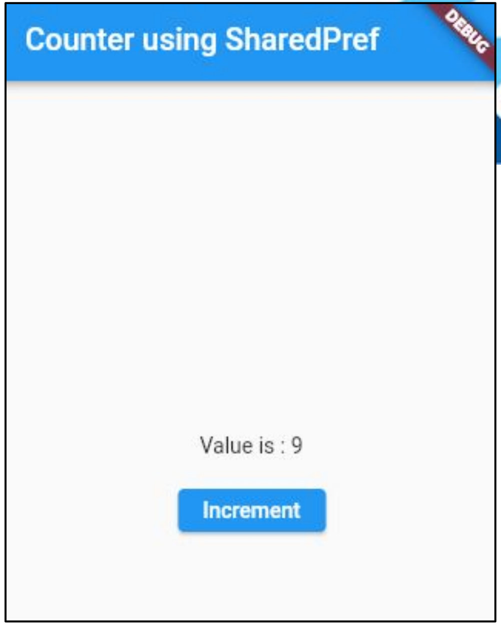
- **Concepts and Techniques**
  - **Sync from backend to Mobile Apps**
    - Use Push Notifications so that the backend servers notify subscribed apps when there is a change or a trigger
    - Excessive calls from mobile apps to the backend can be expensive in terms of cost + battery + server performance.

Sync Question ?



Mobile A

How to keep the incremented value **synched** across different devices ?



Mobile B

Lecture Demo Apps

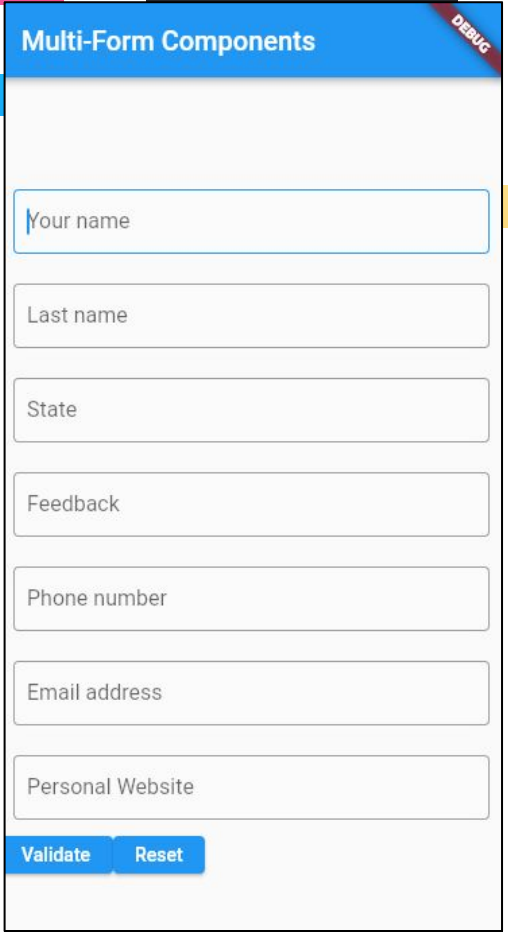
- FutureBuilder Apps:
  - Using String : https://www.dropbox.com/scl/fo/6bo8v40lepgjdgfqxd7b3/h?rlkey=yxelywmvryycf5ox2viuuvohb&dl=0
  - Using MAP : https://www.dropbox.com/scl/fo/v97gkgqghmhs74xpsakzj/h?rlkey=40dyxquqi7syzqhzakyidobvi&dl=0
  - Refresh on Slide : https://www.dropbox.com/scl/fo/jtq5mo2pfwbekkyzjtvx5/h?rlkey=p5v27id805c40nqj5mhmzddt8&dl=0
  - AutoRefresh : https://www.dropbox.com/scl/fo/sy1677e0rzk6pyrbofdne/h?rlkey=nqg9kj99rcb4pk0jgz80q1207&dl=0
- Shared Preferences
  - https://www.dropbox.com/scl/fo/a6oduikt92kza2i22ag2g/h?rlkey=074v2ch2820xemfj9e2xjokoo&dl=0
- To-Do App with DB
  - https://www.dropbox.com/scl/fo/54z1fhk89byynrzv6nsm6/h?rlkey=n67keks2jaioz0icfpulwt1uu&dl=0
- Generating UI Forms dynamically:
  - https://www.dropbox.com/scl/fo/cshd7z9wrwrr1bkaw1fwc/h?rlkey=ivblhlz7aooiuk2p13nk5y8wb&dl=0

Resources

- https://docs.flutter.dev/cookbook/persistence/key-value
- https://docs.flutter.dev/cookbook/persistence/sqlite

Questions from Students

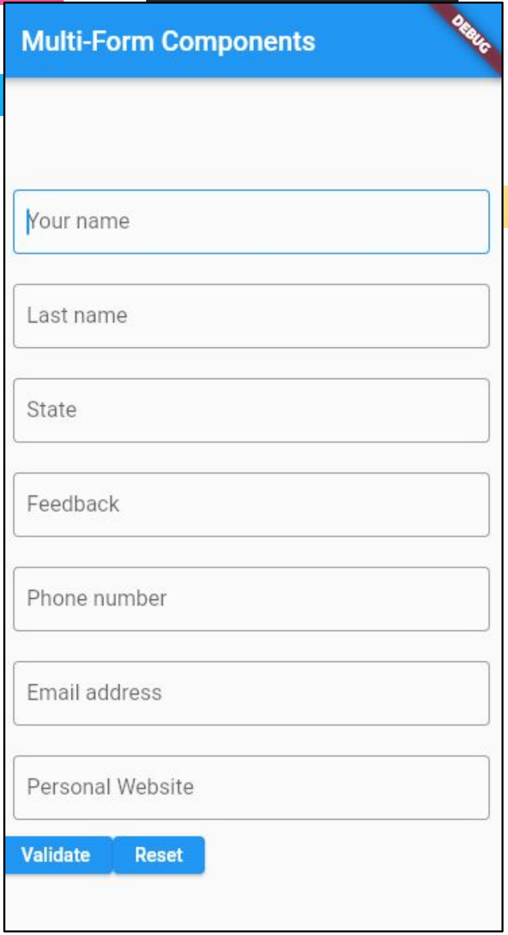
- **A Flutter Screen contains many text fields that we need to individually for each text input:**
  - Show Hint Text
  - Track the value typed by the user
  - Its own validation code(s)
- What's the optimal way to write the flutter code without copying and pasting. (Respecting DRY)



Asked by Anis Rahmani

form_data.dart

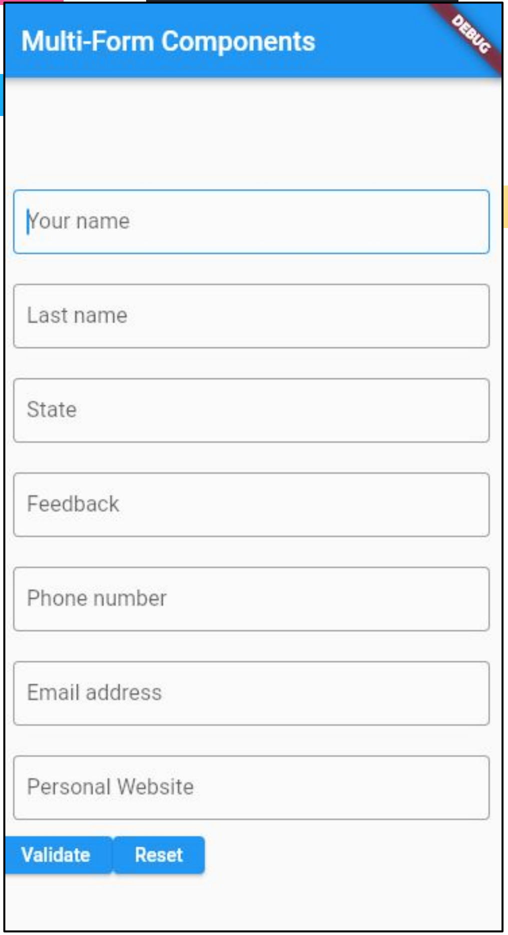
```dart
import 'package:flutter/material.dart';

Map form_items = {
 'name': {
   'hint': 'Your name',
   'value': '',
   'controller': TextEditingController(),
   'validation': ['verify_length_over5', 'is_not_empty'],
 },
 'lastname': {
   'hint': 'Last name',
   'value': '',
   'controller': TextEditingController()
 },
 'state': {
   'hint': 'State',
   'value': '',
   'controller': TextEditingController()
 },
 'feedback': {
   'hint': 'Feedback',
   'value': '',
   'controller': TextEditingController()
 },
 'phone': {
   'hint': 'Phone number',
   'value': '',
   'controller': TextEditingController()
 },
 'email': {
   'hint': 'Email address',
   'value': '',
   'controller': TextEditingController()
 },
 'Website': {
   'hint': 'Personal Website',
   'value': ''
```



Asked by Anis Rahmani

validation.dart

```
//Shame we don't have Eval on Dart.

String? eval_validation_string(String name,
String value) {
 if (name == 'verify_length_over5') return
verify_length_over5(value);
 if (name == 'is_not_empty') return
is_not_empty(value);
 return null;
}

String? verify_length_over5(String value) {
 if (value.length <= 5) {
   return 'Value must be > 5';
 }
 return null;
}

String? is_not_empty(String value) {
 if (value.isEmpty) {
   return 'Field must be left empty';
 }
 return null;
}
```



Multi-Form Components

DEBUG

Your name

Last name

State

Feedback

Phone number

Email address

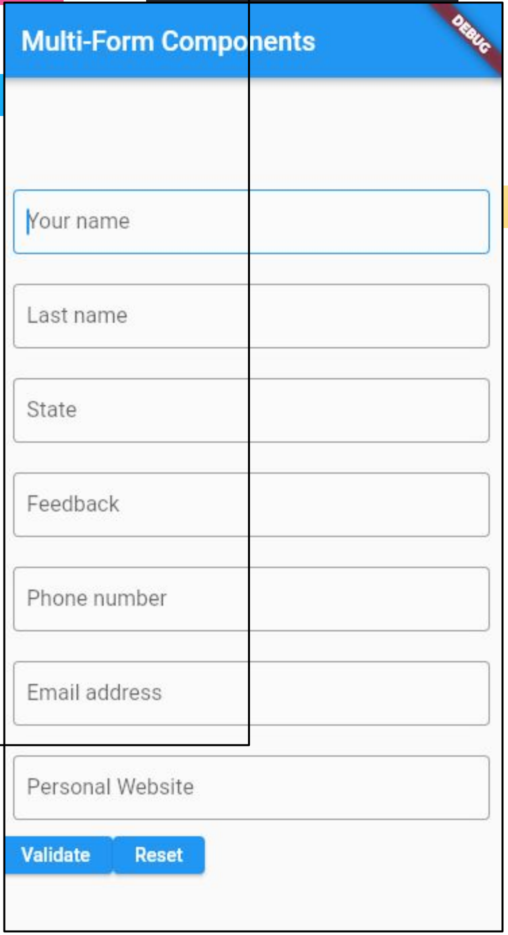
Personal Website

Validate Reset

Asked by Anis Rahmani

homescreen.dart

```dart
class  HomeScreenState extends State<HomeScreen> {
 final _formKey = GlobalKey<FormState>();

 @override
 Widget build(BuildContext context) {
   return Scaffold(
     appBar: AppBar(title: Text("Multi-Form Components")),
     body: Form(
       key:  formKey,
       child: Column(
         mainAxisAlignment: MainAxisAlignment.center,
         crossAxisAlignment: CrossAxisAlignment.center,
         children: [




            ],
           )
         ],
       ),
     ),
   );
 }
}
```


Multi-Form Components

Your name

Last name

State

Feedback

Phone number

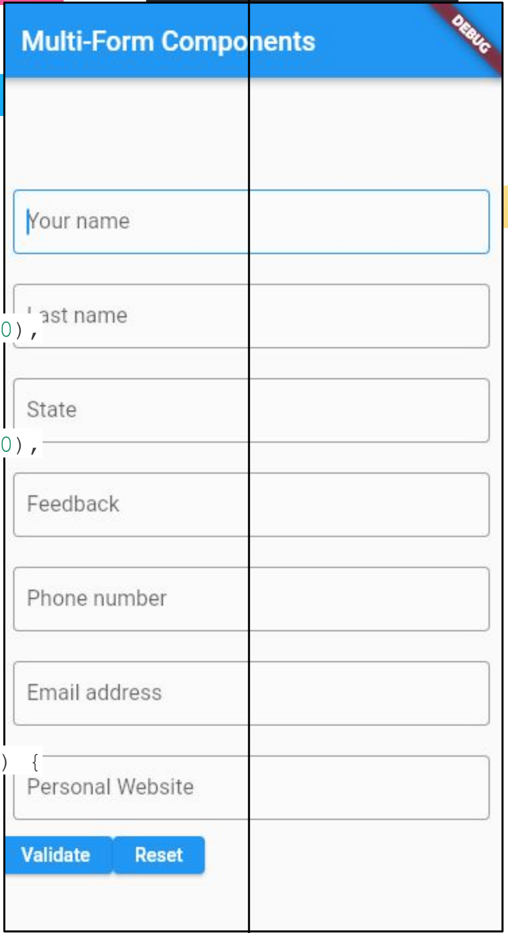
Email address

Personal Website

Asked by Anis Rahmani

homescreen.dart

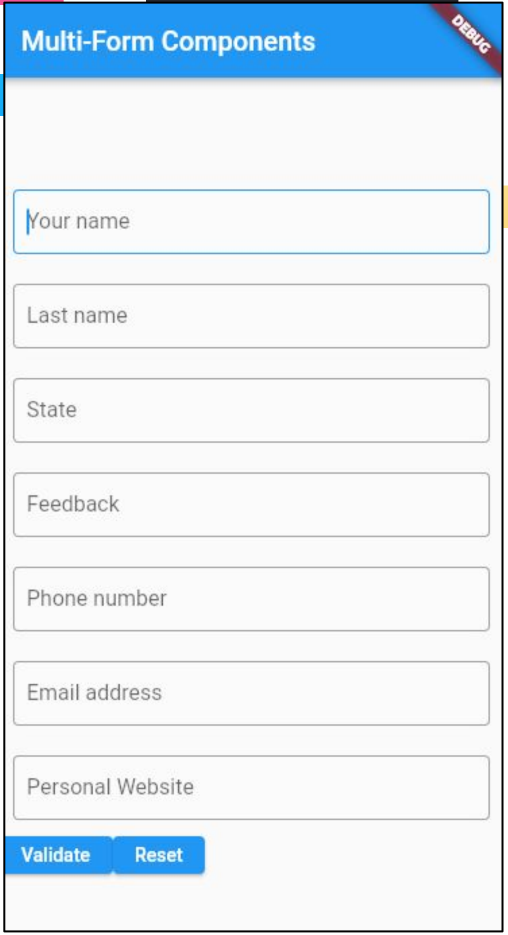
```
children: [
  for (var k in form_items.keys)
    Container(
      height: 50.0,
      margin: EdgeInsets.all(10),
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(13.0),
        color: Colors.white24,
      ),
      child: TextFormField(
        decoration: InputDecoration(
          focusedBorder: OutlineInputBorder(
            borderSide:
                const BorderSide(color: Colors.blue, width: 1.0),
          ),
          enabledBorder: const OutlineInputBorder(
            borderSide:
                const BorderSide(color: Colors.grey, width: 1.0),
          ),
          hintText: form items[k]!['hint'],
          contentPadding: EdgeInsets.only(left: 10.0),
        ),
        onChanged: (value) {
          form_items[k]!['value'] = value;
        },
        controller: form items[k]!['controller'],
        validator: (value) {
          if (form_items[k]!['validation'] is List) {
            for (var validFunc in form items[k]!['validation']) {
              var ret = eval validation string(
                  validFunc, form items[k]!['value']);
              if (ret != null) return ret;
            }
          }
        },
      ),
    )
```



Asked by Anis Rahmani

homescreen.dart

```
        Row(
          children: [
            ElevatedButton(
                onPressed: () {
                  if ( formKey.currentState!.validate()) {
                    ScaffoldMessenger.of(context).showSnackBar(
                      const SnackBar(content: Text('Processing Data')),
                    );
                  }
                },
                child: Text("Validate")),
            SizedBox(
              width: 20,
            ),
            ElevatedButton(
                onPressed: () {
                  for (var k in form_items.keys) {
                    (form_items[k]!['controller'] as TextEditingController)
                        .text = '';
                  }
                },
                child: Text("Reset"))
          ],
```



Asked by Anis Rahmani