Flutter

Crash Course

Day - 3

Stateless vs Stateful Widget

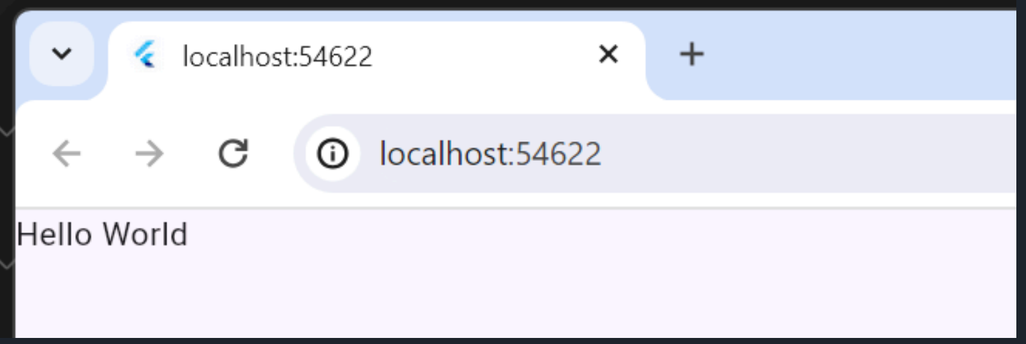
StatelessWidget: Immutable, no state to manage, ideal for static content.

StatefulWidget: Mutable, can manage state, ideal for dynamic content that needs to change in response to user interactions or other events.

Stateless Widget

Everything we have learnt so far is Stateless Widget.

```
Run | Debug | Profile
void main() {
  runApp(MaterialApp(
    home: Scaffold(body: Text("Hello World"),),
  )); // MaterialApp
}
```



How to bring the same output by extending stateless class

Stateless Widget

Step 1:

```
class MainApp extends StatelessWidget{

}
```

Create a class which extends statelesswidget class

StatelessWidget is a base class for widgets that do not require mutable state. Widgets of this type are immutable, meaning their properties can't change—they are finalized upon creation.

Stateless Widget

Step 2:

you would typically add a build method to describe how the widget should render.

This build method belongs to StatelessWidget class the term override means we are taking a function from another class and using it.

Stateless Widget

Step 2:

```
class MainApp extends StatelessWidget{
  @override
  Widget build(BuildContext context) {

  }
}
```

We have created the build method, this build method will return the Widget

Stateless Widget

Step 3:

```
Run | Debug | Profile
void main() {
  runApp(MaterialApp(
    home: Scaffold(body: Text("Hello World"),),
  )); // MaterialApp
}


class MainApp extends StatelessWidget{
  @override
  Widget build(BuildContext context) {
    return Text("Hello World");
  }
}
```

Now we can make the mainapp class return the Text Widget.

Stateless Widget

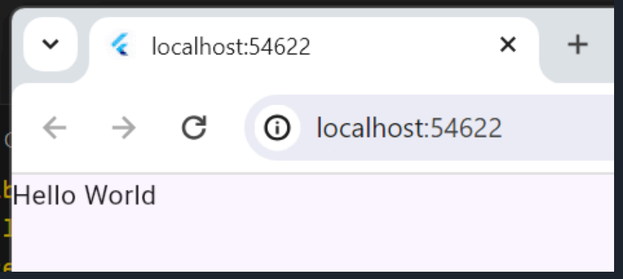
Step 4:

main.dart > main

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

Run | Debug | Profile
void main() {
  runApp(MaterialApp(
    home: Scaffold(body: MainApp(),),
  )); // MaterialApp
}


class MainApp extends StatelessWidget{
  @override
  Widget build(BuildContext context) {
    return Text("Hello World");
  }
}
```

ROBLEMS 2

Launching lib
This app is
Debug servic



Replace the body with MainApp

Stateless Widget

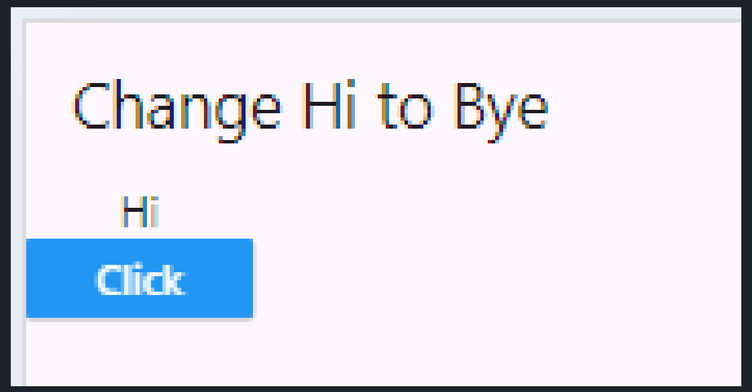
What we also can do is, write the entire basic structure code inside MainApp

main.dart > ...

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

Run | Debug | Profile
void main() {
  runApp(MainApp());
}

class MainApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
    home: Scaffold(
      body: Text("Hello World"),
    )); // Scaffold // MaterialApp
  }
}
```

So Everything we have built so far is stateless widget

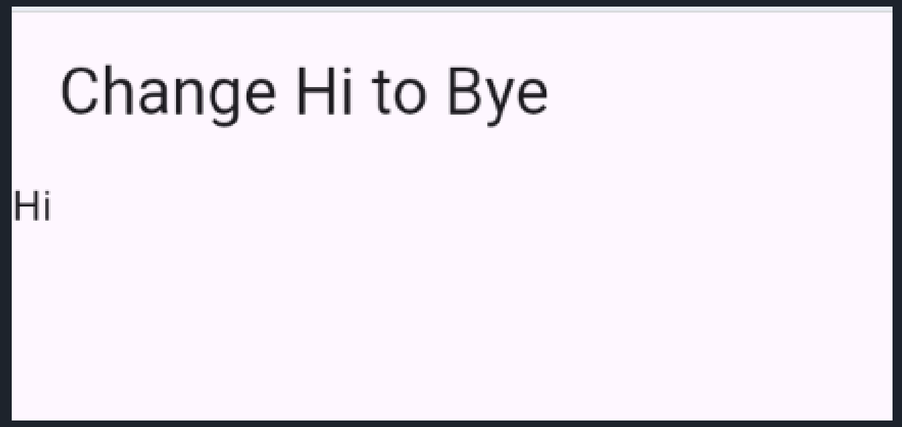
Let’s understand stateful widget using below example




```
Run | Debug | Profile
void main() {
  runApp(MainApp());
}

class MainApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
            appBar: AppBar(
              title: Text("Change Hi to Bye"),
            ), // AppBar
            body: Column(
              children: [
                Text("Hi"),
                MaterialButton(
                  color: Colors.blue,
                  onPressed: () {
                    print("Bye");
                  },
                  child: Text("Click"),
                ) // MaterialButton
              ],
            ))); // Column // Scaffold // MaterialApp
  }
}
```

when you click the click button change Hi to Bye, create this UI First

Step 1: Create a Text Widget




```
main.dart > MyApp > build
import 'package:flutter/material.dart';

Run | Debug | Profile
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text("Change Hi to Bye"),),
        body: Text("Hi"),
      ), // Scaffold
    ); // MaterialApp
  }
}
```

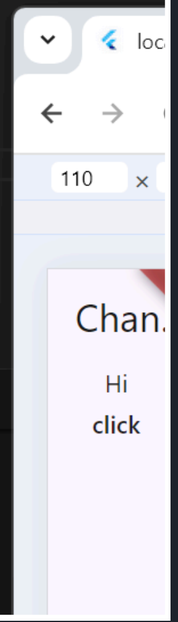
Next try to create another text below it

Step 2: Create a Button

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text("Change Hi to Bye"),),
        body: Column(children: [
          Text("Hi"),
          MaterialButton(
            onPressed: (){
              print("Hello");
            },
            child: Text("click"),
            ) // MaterialButton
        ],), // Column
      ), // Scaffold
    ); // MaterialApp
  }
}
```

LEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
started application in 187ms.
started application in 148ms.
started application in 115ms.
started application in 131ms.
Hello
```



Every button will take onpressed function and child

Let’s understand stateful widget using below example

Step 3: create a variable called display name and set it to text widget

```
children: [
  Text(displayname),
  MaterialButton(
    color: Colors.blue,
    onPressed: () {
      displayname = "Bye";
      print(displayname);
    },
    child: Text("Click"),
  ) // MaterialButton
```

Let’s understand stateful widget using below example

Step 4: Instead of printing Hi when you clicking the button. change the displaytext to Bye.

```
children: [
  Text(displayname),
  MaterialButton(
    color: Colors.blue,
    onPressed: () {
      displayname = "Bye";
      print(displayname);
    },
    child: Text("Click"),
  ) // MaterialButton
```

Let’s understand stateful widget using below example

Step 5: Now you will notice that Hi doesn’t change on the screen but it got printed on the console

```
children: [
  Text(displayname),
  MaterialButton(
    color: Colors.blue,
    onPressed: () {
      displayname = "Bye";
      print(displayname);
    },
    child: Text("Click"),
  ) // MaterialButton
```

Step 5: Now you will notice that Hi doesn’t change on the screen but it got printed on the console

Let’s understand stateful widget using below example

Step 6:Generally when you change the variable once it should reflect everywhere

DartPad New Samples

```
1 void main() {
2   String displaytext = "Hi";
3   displaytext = "Bye";
4   print(displaytext);
5
```

Run

The reason why its not changing is, This is a Stateless widget

Let’s understand stateful widget using below example

Step 7:Understand State

When you try to change something on the screen or on the UI, Which means you are changing its state.

Your trying to change the text state from Hii to Bye.

But we are trying to do it inside stateless Widget. we will be able to change the state only in the stateful widget.

How to create stateful widget

Step 1: Create a class which extends stateful widget

```
class MainApp extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {

  }
```

When you extend stateful widget you have to override the method createstate.

How to create stateful widget

Step 2: Now you will need another subclass which extends state. if you remember when we use stateless widget we did not have any subclasses but here we have subclass because we need state here

```
class MainApp extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {

  }
}

class MainAppState extends State<MainApp>{

}
```

the sub class will extend state of main class

How to create stateful widget

Step 3: The Main class should return the subclass

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

Run | Debug | Profile
void main() {
  runApp(MainApp());
}

class MainApp extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    return MainAppState();
  }
}

class MainAppState extends State<MainApp>{

}
```

How to create stateful widget

Step 4: Now create build function just like how we will create for Stateless widget

```
Run | Debug | Profile
void main() {
  runApp(MainApp());
}

class MainApp extends StatefulWidget{
  @override
  State<StatefulWidget> createState() {
    return MainAppState();
  }
}

class MainAppState extends State<MainApp>{
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text("John"),),
        body: Text("Hello World"),
      ), // Scaffold
    ); // MaterialApp
  }
}
```

How to create stateful widget

Step 5: Now Copy paste the previous code inside, MainAppState.

```
class MainAppState extends State<MainApp>{

  String displayname = "Hi";
  @override
   Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
            appBar: AppBar(
              title: Text("Change Hi to Bye"),
            ), // AppBar
            body: Column(
              children: [
                Text(displayname),
                MaterialButton(
                  color: Colors.blue,
                  onPressed: () {
                    displayname = "Bye";
                    print(displayname);
                  },
                  child: Text("Click"),
                ) // MaterialButton
              ],
            ))); // Column // Scaffold // MaterialApp
  }
}
```

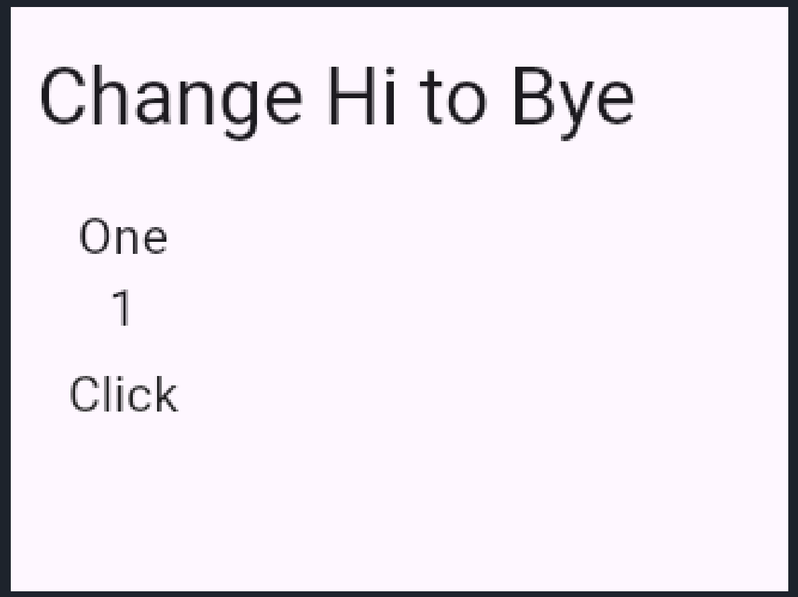
How to create stateful widget

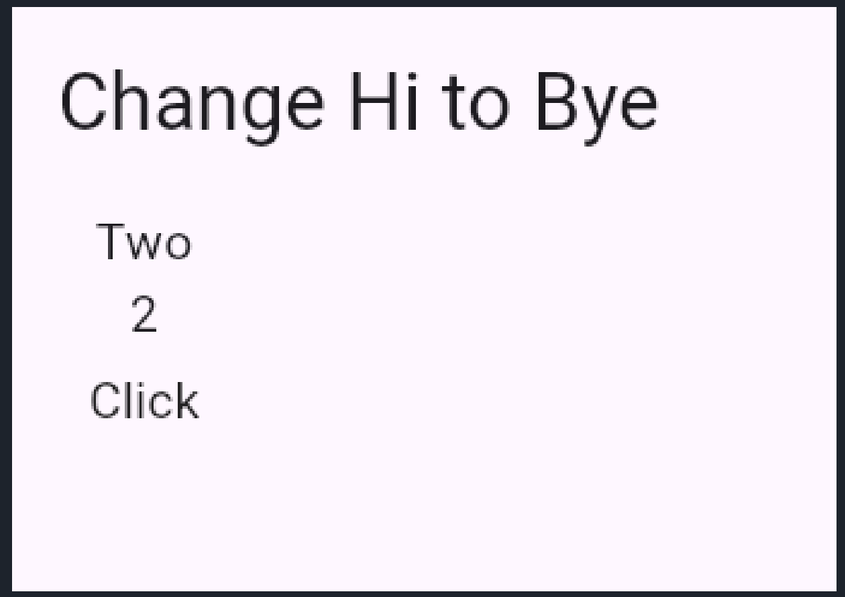
Step 6: Now use setstate() function to change the value

```
class MainAppState extends State<MainApp>{

  String displayname = "Hi";
  @override
   Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
            appBar: AppBar(
              title: Text("Change Hi to Bye"),
            ), // AppBar
            body: Column(
              children: [
                Text(displayname),
                MaterialButton(
                  color: Colors.blue,
                  onPressed: () {
                    setState(() {
                      displayname = "Bye";
                    });
                  },
                  child: Text("Click"),
                ) // MaterialButton
              ],
            ))); // Column // Scaffold // MaterialApp
  }
}
```

How to create stateful widget

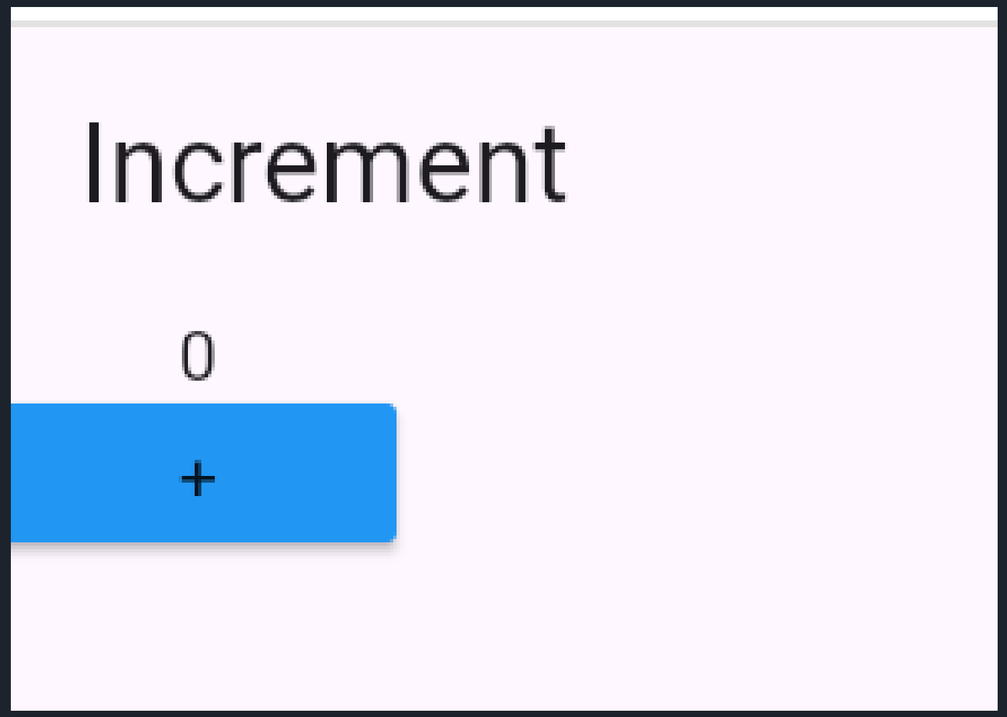
Its time for the little Task !





How to create stateful widget

Its time for the little Task 2 !



What is List in Dart ?

In Dart, a List is an ordered collection of objects. It is one of the fundamental data structures used to store and manipulate a collection of items.

Just like, Array in other programming languages, here we have list

What is List in Dart ?

```dart
void main() {
  List<String> fruits = ['Apple', 'Banana', 'Orange'];
  print(fruits[0]);
  print(fruits[1]);
  print(fruits[2]);
}

```

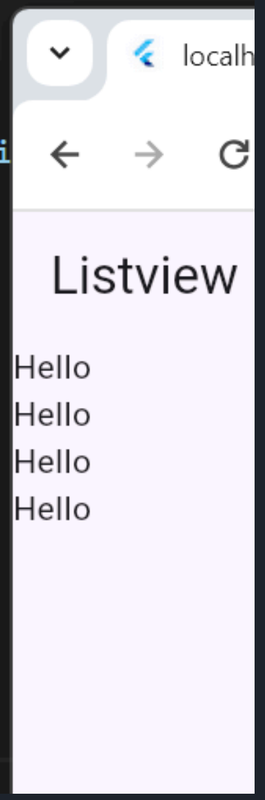
```dart
void main() {
  List<int> fruits = [32,43,54];
  print(fruits[0]);
  print(fruits[1]);
  print(fruits[2]);
}
```

How to add and remove Element

```
1 void main() {
2  List<int> fruits = [10, 20,30];
3
4   fruits.add(40);
5   print(fruits[0]);
6   print(fruits[1]);
7   print(fruits[2]);
8   print(fruits[3]);
9
10  fruits.remove(10);
11  fruits.removeAt(1);
12
13  print(fruits);
14
15
16 }
17
```

How to use List Builder

```
Run | Debug | Profile
void main() {
  runApp(MaterialApp(
    home: Scaffold(appBar: AppBar(ti
    ), // AppBar
    body: Column(
      children: [
        Text("Hello"),
        Text("Hello"),
        Text("Hello"),
        Text("Hello"),
      ],
    ), // Column
    ), // Scaffold

  )); // MaterialApp
}
```



Bring this output on the screen,

Instead of printing the same widget for 100 times what if something can create it for us 100 times?

How to use List Builder

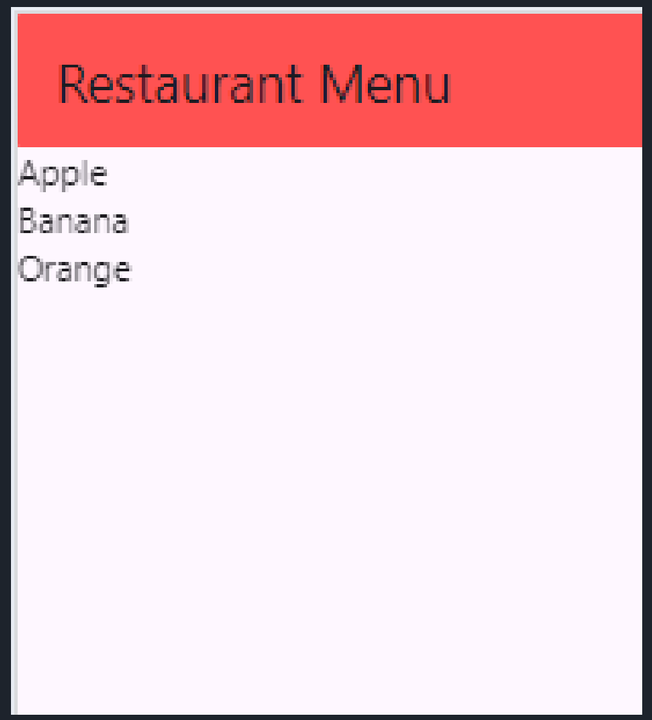
```
runApp(MaterialApp(
  home: Scaffold(
    appBar: AppBar(
      title: Text("Restaurant Menu"),
      backgroundColor: Colors.redAccent,
    ), // AppBar
    body: ListView.builder(
      itemCount: 2,
      itemBuilder: (context,index){
      return Text("Hello");
    }) // ListView.builder
    )), // Scaffold // MaterialApp
);
}
```

itemCount: Defines how many times the widget should be recreated.

itemBuilder: The context is useful for building the underlying widget. The index denotes the position of the widget in the list.

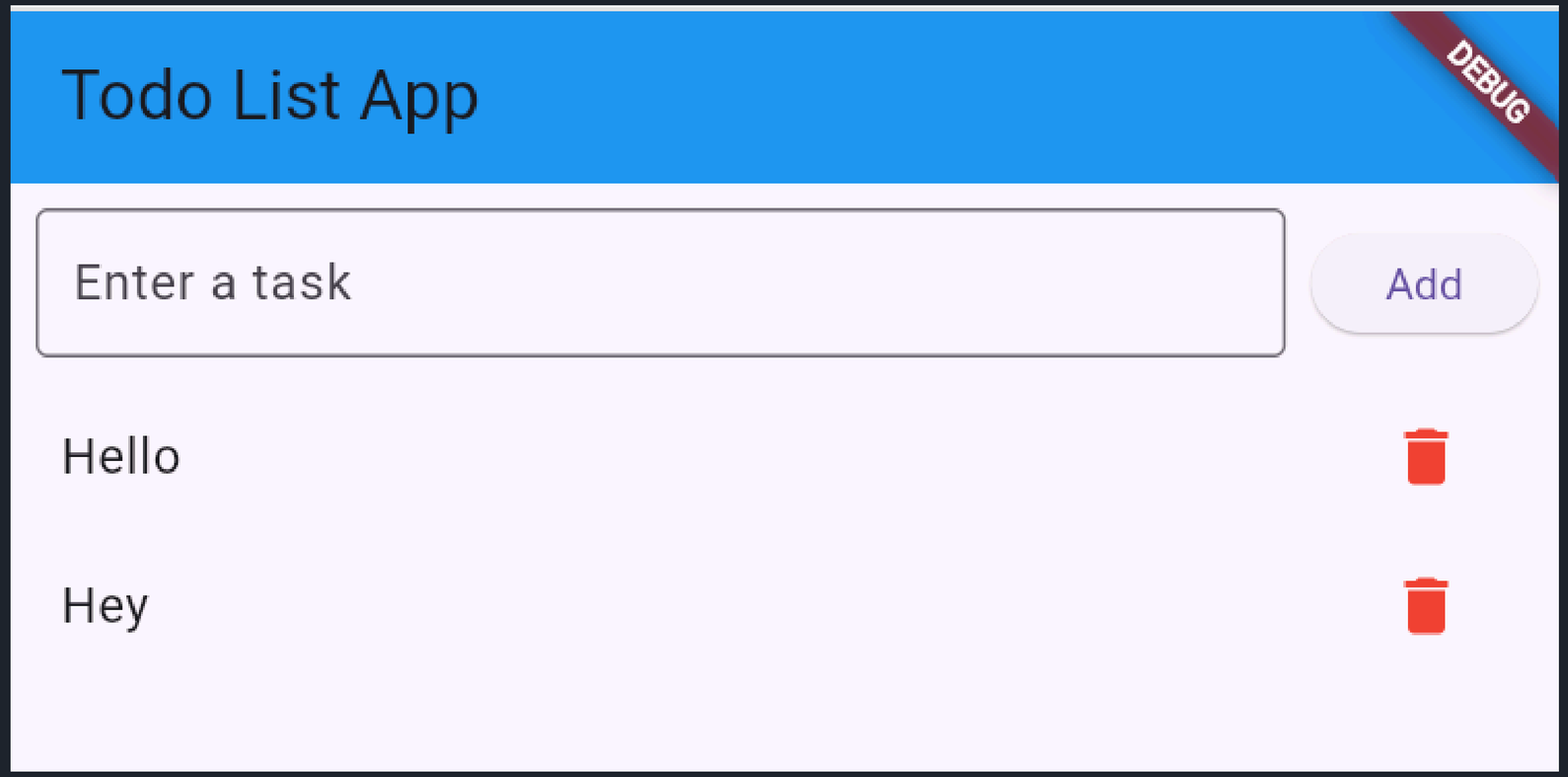
Print elements from fruit list using list Builder

```
Run | Debug | Profile
void main() {

  List<String> fruits = [
  'Apple',
  'Banana',
  'Orange',
];

  runApp(MaterialApp(
    home: Scaffold(
        appBar: AppBar(
          title: Text("Restaurant Menu"),
          backgroundColor: Colors.redAccent,
        ), // AppBar
        body: ListView.builder(
          itemCount: 3,
          itemBuilder: (context,index){
          return Text(fruits[index]);
        }) // ListView.builder
        )), // Scaffold // MaterialApp
  );
}
```



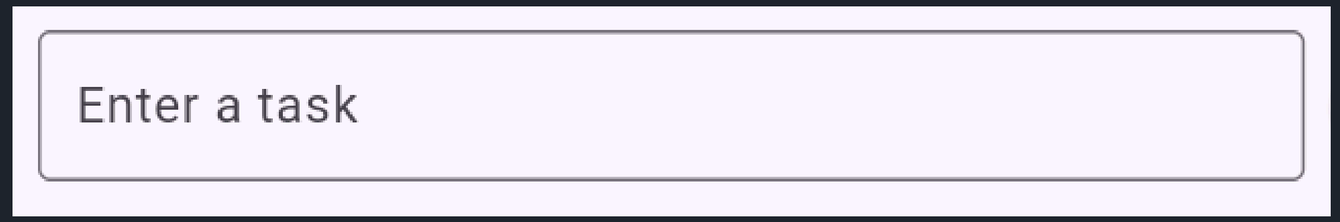
Let’s Build a Todo App

Todo app is a great project when you are learning something new



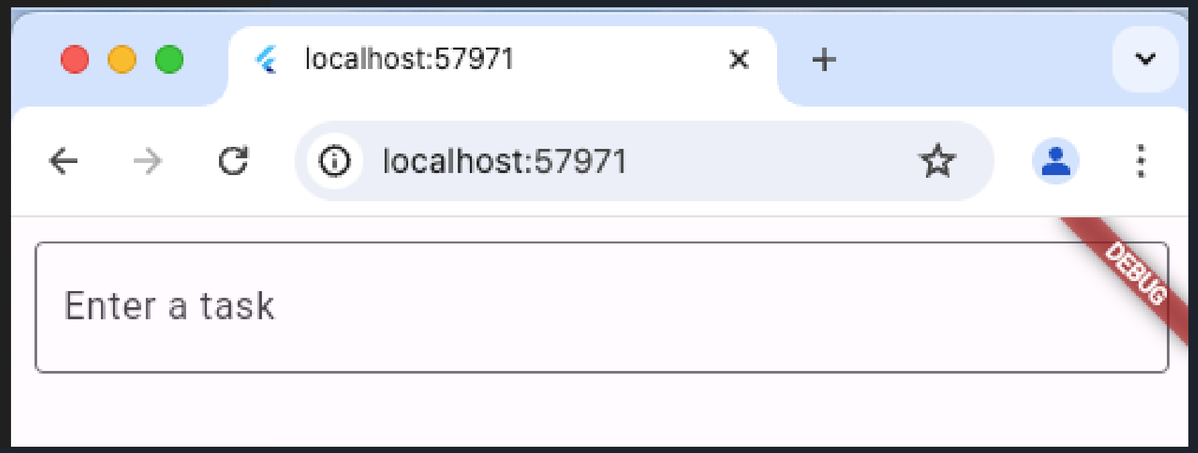
Before We Create Todo App, let's understand

- TextField
- TextEditingController



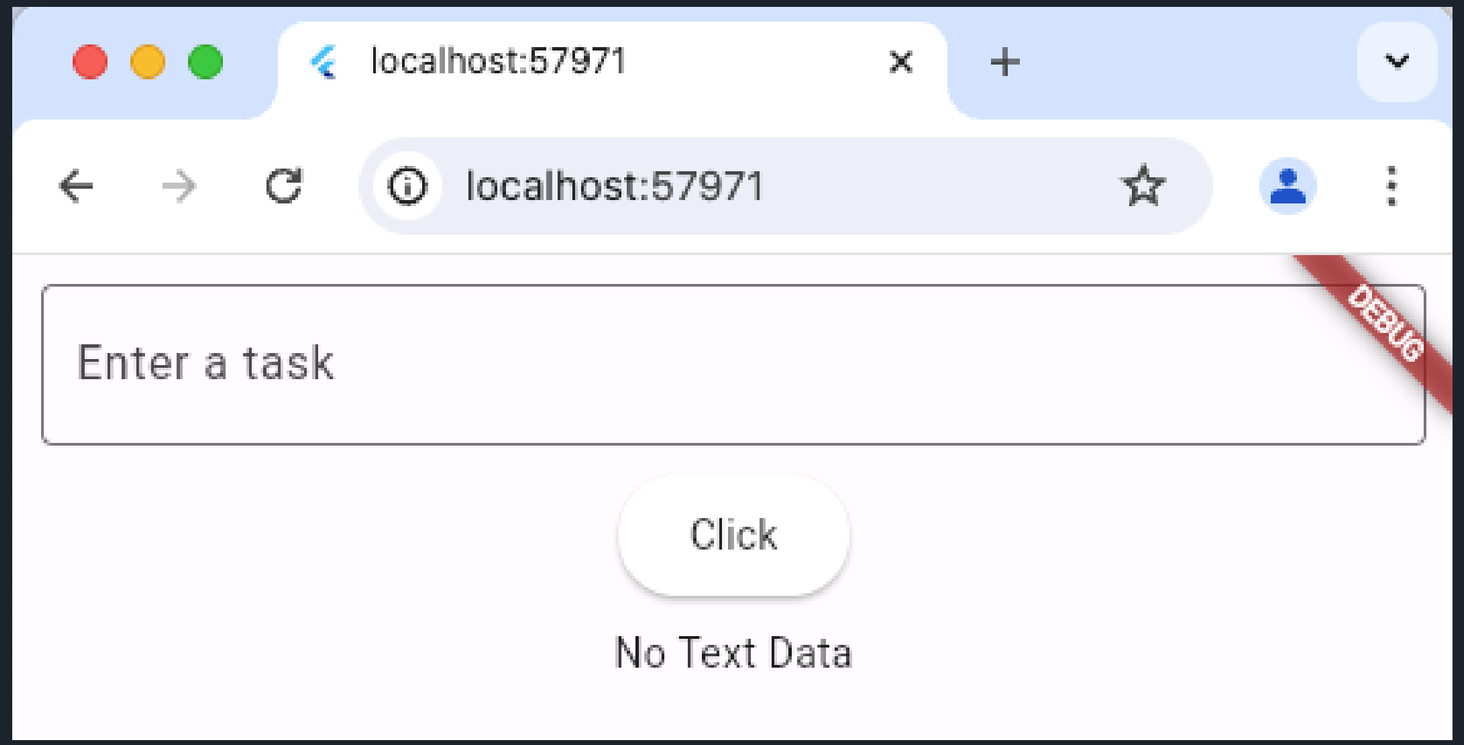
Create Text Field

```
TextField(
  decoration: InputDecoration(
    border: OutlineInputBorder(),
    label: Text("Enter a task"),
  ), // InputDecoration
), // TextField
```



Read text from Text Field

Add button and label

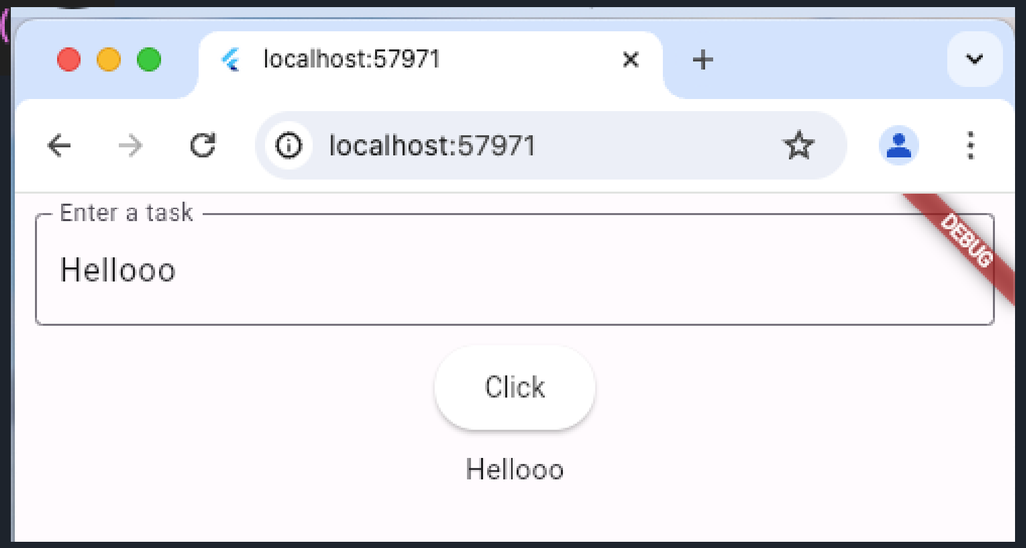


Read text from Text Field

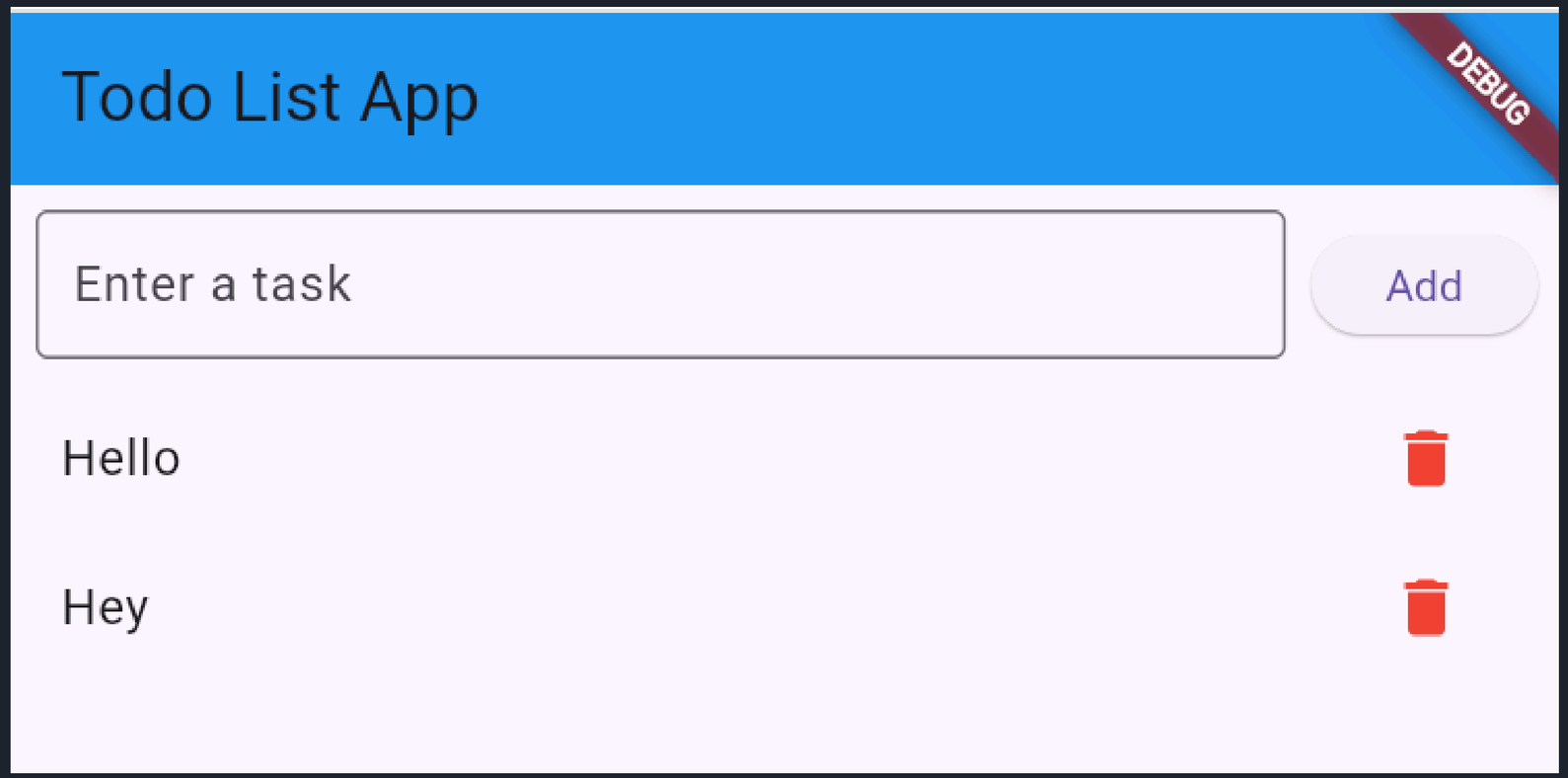
Add TextEditingController to read text from TextField

```
TextEditingController textController = TextEditingController(
```

```
TextField(
  controller: textController,
  decoration: InputDecoration(
    border: OutlineInputBorder(),
    label: Text("Enter a task"),
  ), // InputDecoration
), // TextField
```

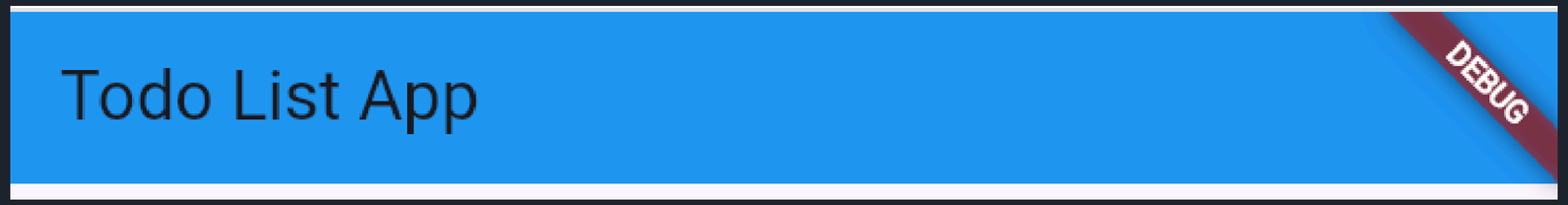


Let’s Build a Todo App



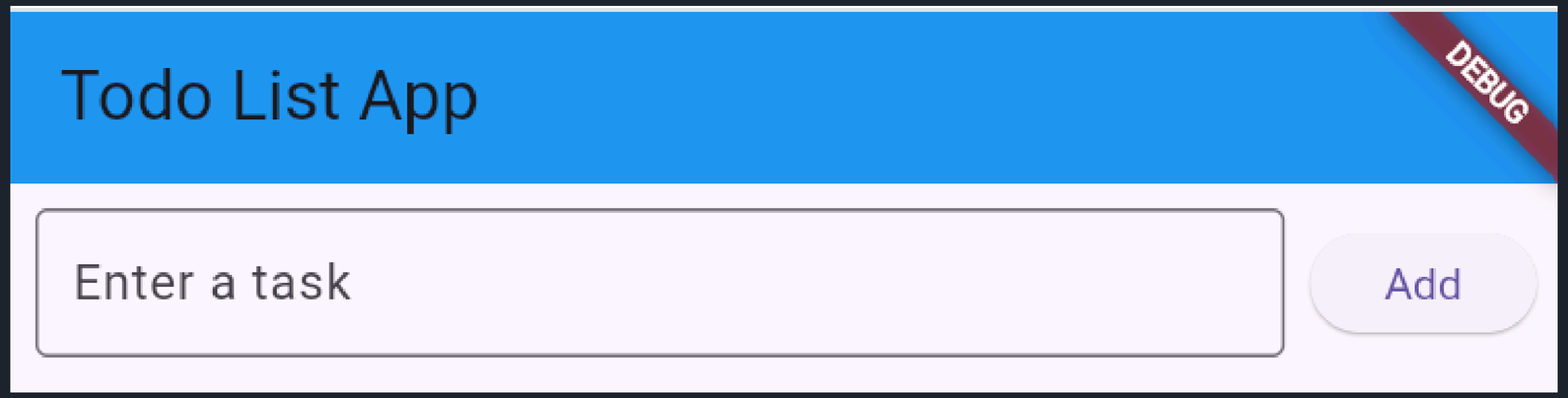
Todo App

Step 1: Create App Bar



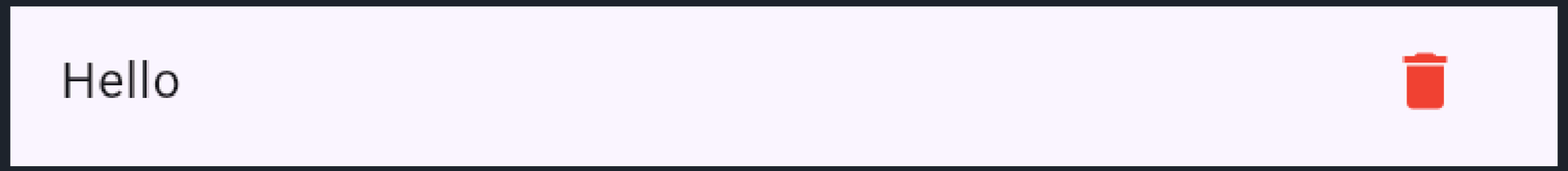
Todo App

Step 2: Make Textfield and Button in Row



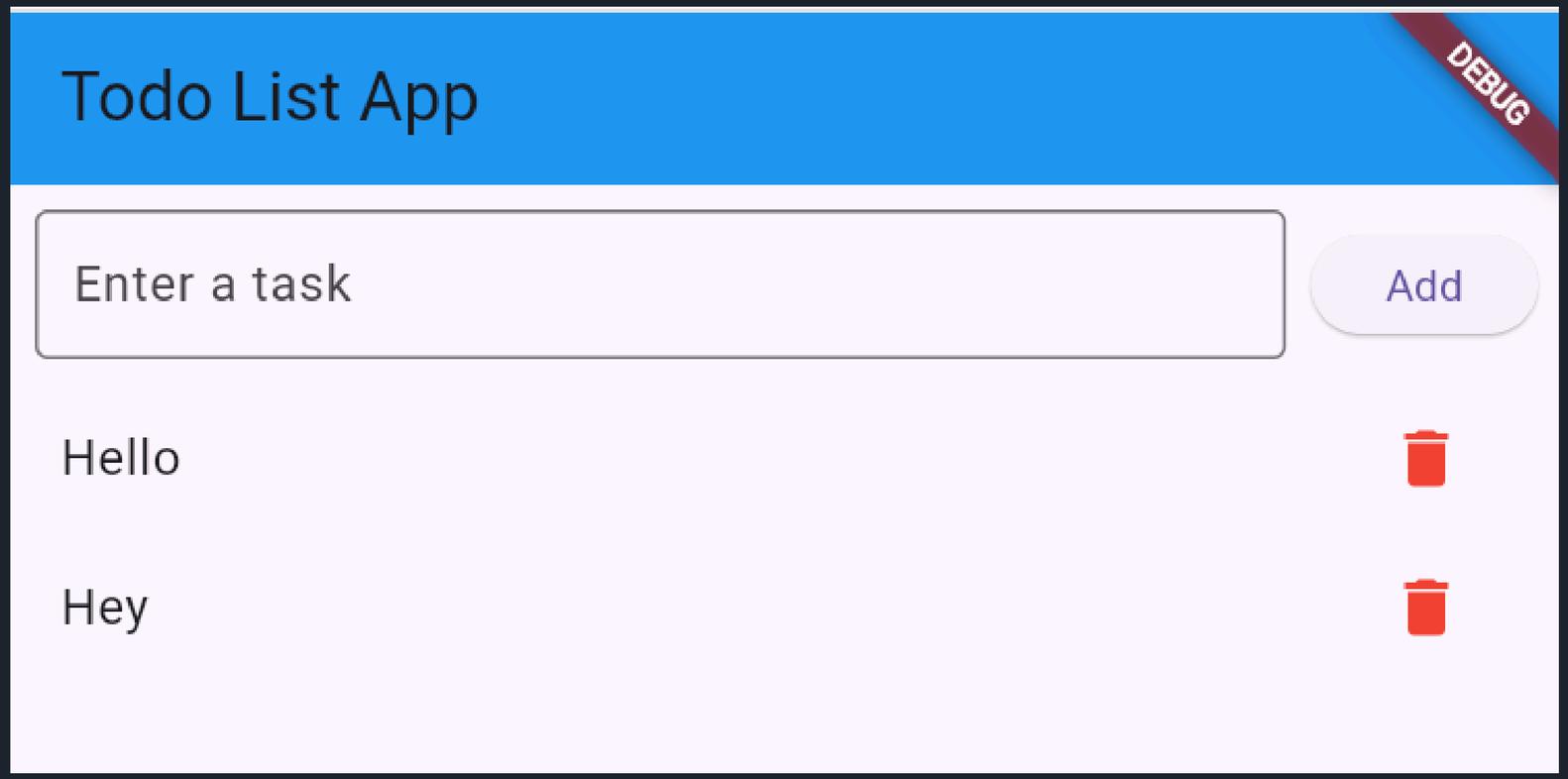
Todo App

Step 3: Create Todo Item Card



Todo App

Step 4: Create List View Builder to load all items



Todo App

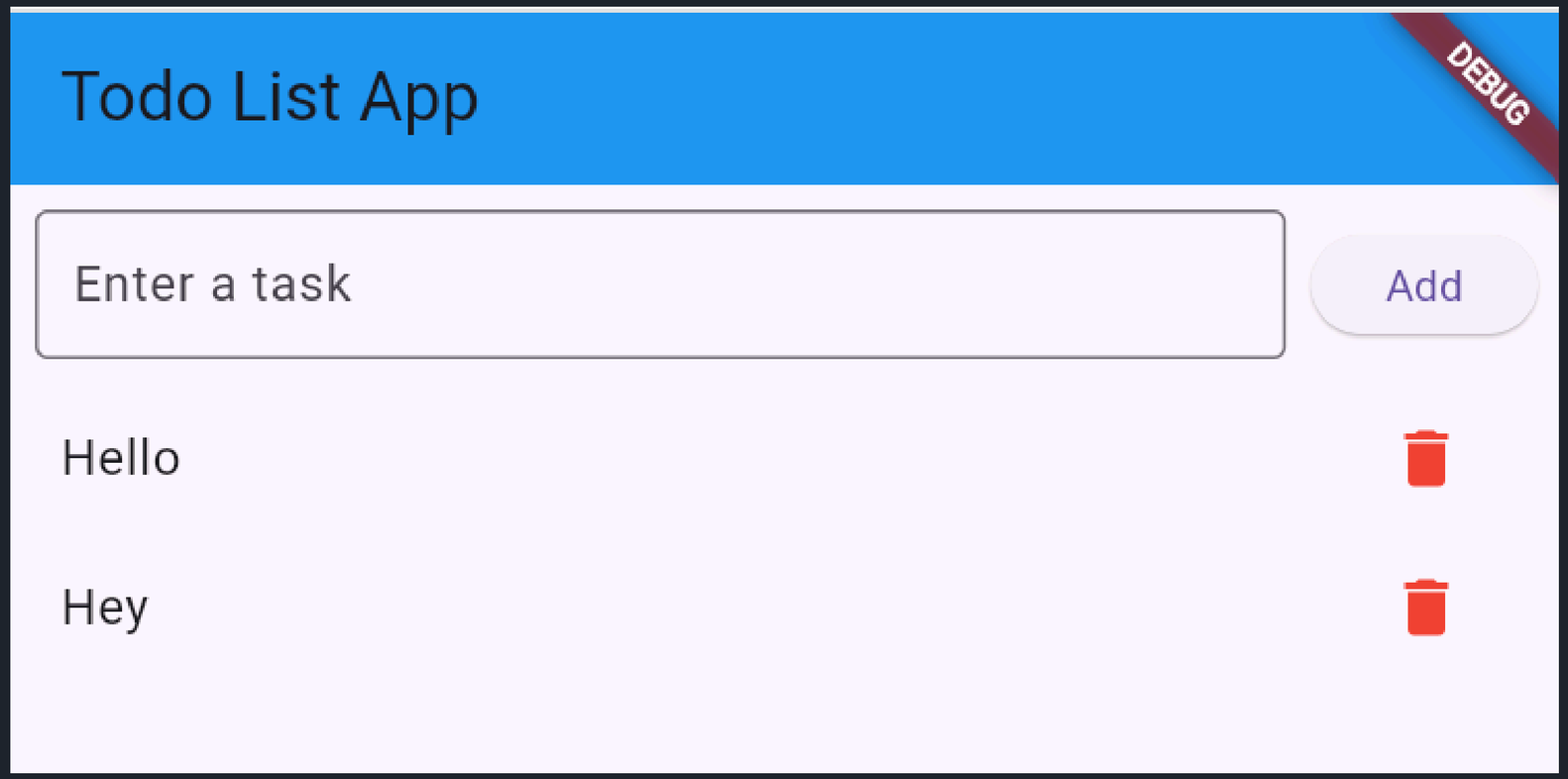
Step 5: Create List and bind with List view builder

```
List<String> taskList = [
  "Task 01",
  "Task 02",
  "Task 03",
];
```

```
child: ListView.builder(
  itemCount: taskList.length,
  itemBuilder: (context, index) {
    return Container(
      child: Row(
        children: [
          Expanded(
            child: Container(
              padding: EdgeInsets.all(10),
              child: Text(taskList[index])), // Container
```

Todo App

Step 6: Add & Delete Functionality



Thank You