

Flutter Fundamentals

TextInput / Form controls



Peter Kassenaar –
info@kassenaar.com



Reading user input

Using textboxes to fetch user input

Generic documentation “Interactivity”

A screenshot of the Flutter Docs website. The left sidebar contains a navigation menu with categories like Layout, Adaptive & responsive design, Design & theming, Interactivity, Gestures, and Input & forms. The 'Interactivity' category is circled in red, and a red arrow points from it to the 'Create and style a text field' article in the 'Input & forms' section. The main content area shows the title 'Create and style a text field' with a breadcrumb 'Cookbook > Forms > Create and style a text field'. The text explains that text fields allow users to type text and lists two types: `TextField` and `TextFormField`. It then details the `TextField` widget, noting it is decorated with an underline and can be customized with `InputDecoration`. A code block shows a snippet of `TextField` initialization with `InputDecoration` and `OutlineInputBorder`.

Flutter Docs

Homepage Community Packages API reference

Layout
Adaptive & responsive design
Design & theming
Interactivity
Add interactivity to your app
Gestures
Input & forms
Create and style a text field
Retrieve the value of a text field
Handle changes to a text field
Manage focus in text fields
Build a form with validation
Display a snack bar
Implement actions &

Create and style a text field

Cookbook > Forms > Create and style a text field

Text fields allow users to type text into an app. They are used to build forms, send messages, create search experiences, and more. In this recipe, explore how to create and style text fields.

Flutter provides two text fields: `TextField` and `TextFormField`.

TextField

`TextField` is the most commonly used text input widget.

By default, a `TextField` is decorated with an underline. You can add a label, icon, inline hint text, and error text by supplying an `InputDecoration` as the `decoration` property of the `TextField`. To remove the decoration entirely (including the underline and the space reserved for the label), set the `decoration` to null.

```
TextField(  
  decoration: InputDecoration(  
    border: OutlineInputBorder(),  
    hintText: 'Enter a search term',  
  ),  
)
```

<https://docs.flutter.dev/cookbook/forms/text-input>

Controller for an editable text field:



“Whenever the user modifies a text field with an associated `TextEditingController`, the text field updates `value` and the controller notifies its listeners. Listeners can then read the `text` and `selection` properties to learn what the user has typed or how the selection has been updated.”

Creating a textbox to read user input



Flutter > widgets.dart > TextEditingController class

Search API Docs

widgets library

CLASSES

absorbPointer

Accumulator

Action

ActionDispatcher

ActionListener

Actions

ActivateAction

ActivateIntent

Align

Alignment

AlignmentDirectional

AlignmentGeometry

AlignmentGeometryTwo...

AlignmentTween

TextEditingController class

A controller for an editable text field.

Whenever the user modifies a text field with an associated `TextEditingController`, the text field updates `value` and the controller notifies its listeners. Listeners can then read the `text` and `selection` properties to learn what the user has typed or how the selection has been updated.

Similarly, if you modify the `text` or `selection` properties, the text field will be notified and will update itself appropriately.

A `TextEditingController` can also be used to provide an initial value for a text field. If you build a text field with a controller that already has `text`, the text field will use that text as its initial value.

The `value` (as well as `text` and `selection`) of this controller can be updated from within a listener added to this controller. Be aware of infinite loops since the listener will also be notified of the changes made from within itself. Modifying the composing region from within a listener can also have a bad interaction with some input methods. Gboard, for example, will try to restore the composing region of the text if it was modified programmatically, creating an infinite loop of communications between the framework and the input method. Consider using `TextInputFormatters` instead for as-you-type text modification.

If both the `text` and `selection` properties need to be changed, set the controller's `value` instead. Setting `text` will clear the selection and composing range.

<https://api.flutter.dev/flutter/widgets/TextEditingController-class.html>

Adding a textController



GOAL: We want to search for (a list of) countries:

1. Create a textController and a countryName

```
final textController = TextEditingController();  
String countryName = '';
```

2. Update getCountries() to receive countryName as argument (Don't forget to update the API-endpoint!)

Passing countryName as argument



```
class _HomeCountriesState extends State<HomeCountries> {  
  // variables in this Widget  
  final TextEditingController = TextEditingController();  
  String countryName = '';  
  String url = 'https://restcountries.com/v3.1/name';  
  ...  
  // 3. Get our countries  
  void getCountries(String countryName) async {  
    // 3a. get the response  
    Response response = await get(Uri.parse('$url/$countryName$fields'));  
    ...  
  }  
}
```

Updating the UI with a TextBox

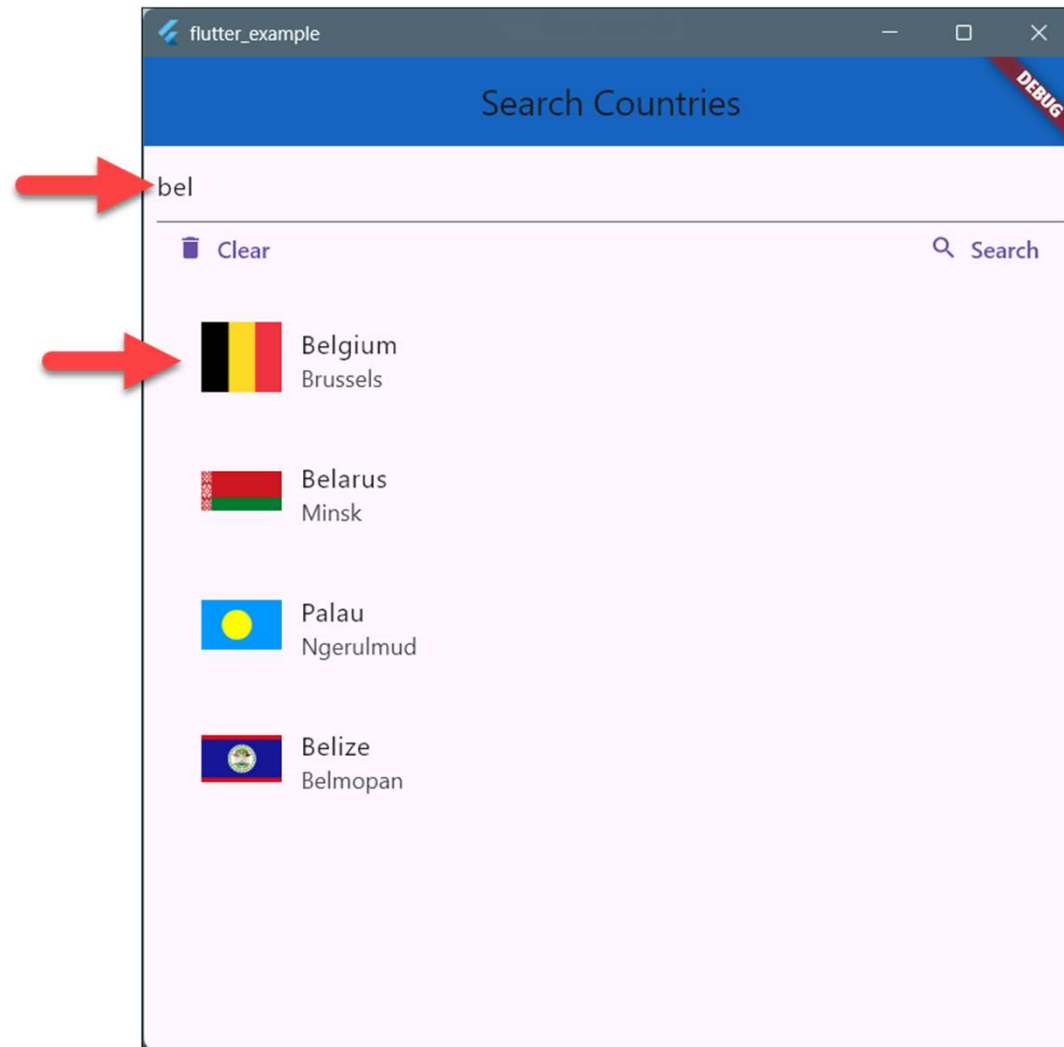


```
body: Column(  
  children: <Widget>[  
    Padding(  
      padding: EdgeInsets.fromLTRB(8, 0, 8, 0),  
      child: Column(  
        children: <Widget>[  
          TextField(  
            controller: textController,  
            decoration: InputDecoration(  
              hintText: 'Search countries...'),  
            ),  
          Row(  
            ...  
          ),  
        ],  
        Expanded(...),  
      ],  
    ),  
  ],  
),
```



```
Row(  
  mainAxisAlignment: MainAxisAlignment.spaceBetween,  
  children: <Widget>[  
    // Clear the selection  
    IconButton(  
      icon: Icon(Icons.delete),  
      label: Text('Clear'),  
      onPressed: () {  
        setState(() {  
          countryName = '';  
          textController.text = '';  
          countries = []; // reset array with countries  
        });  
      },  
    ),  
    // Search for countries  
    IconButton(  
      icon: Icon(Icons.search),  
      label: Text('Search'),  
      onPressed: () {  
        print(textController.text); // just to check  
        getCountries(textController.text);  
      },  
    ),  
  ],  
),
```


Result



Summary



- In order to **read text** from a form field:
 - Use `TextField()` for quick inputs, or
 - Use `TextFormField()` (=with validation, form submission)
- Bind the field to `TextEditingController()`
- Use styling, decoration, focus, etc.
- Read the `textController.text` for the text that was typed in the input field.



TextField() VS TextFormField()

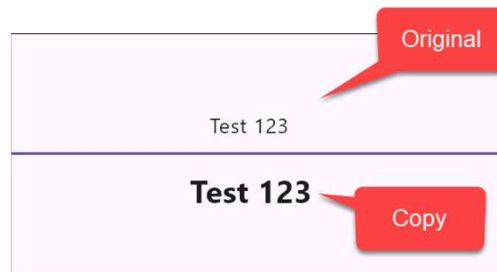
- Official Documentation:
- api.flutter.dev/flutter/material/TextField-class.html
- api.flutter.dev/flutter/material/TextFormField-class.html

Feature	TextField	TextFormField
Form Integration	✗ No built-in validation	✓ Built for <code>Form / FormState</code> usage
Validation	Manual validation	Built-in <code>validator</code> + <code>autovalidateMode</code>
Form Submission	Manual	Tied into <code>Form.of(context)</code> workflows
Use Case	Standalone input	Input in a validated form

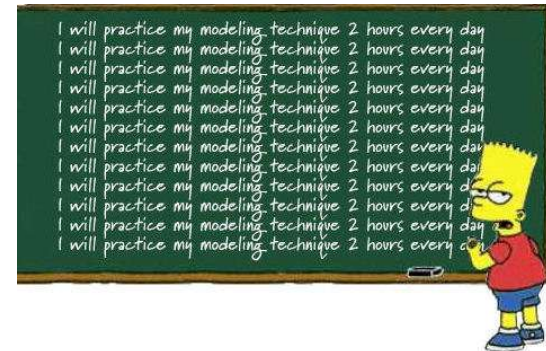
Workshop - simple



- Create a new application with a `TextField()`
- Text that is typed in the `TextField()`, is echoed on the page
 - So, you're basically making a [copy function](#)
- To do this,
 - Add a page with a `TextField()` and a `TextEditingController()`
 - Attach the controller to the textfield
 - Add a listener to the controller to update text variable
 - Call `setState()` to update and display the text



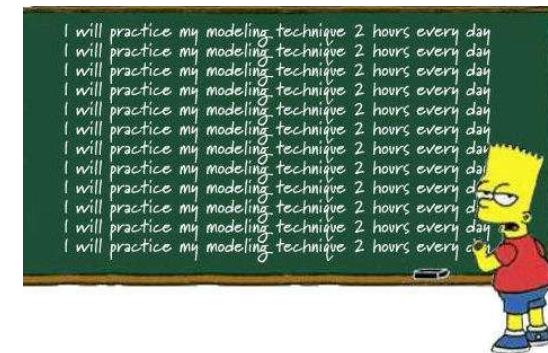
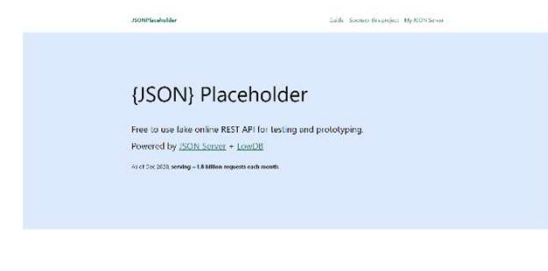
<https://docs.flutter.dev/cookbook/forms/text-field-changes>



Workshop - extensive



- Update your (dummy user API) application with a `TextField()` that we can type a number in.
- Search for this specific user and display their
 - Name, Username
 - Email, address, phone and company name
- The endpoint becomes
<https://jsonplaceholder.typicode.com/users/<id>>
- Optional: create a dropdown that displays a list of numbers 1-10 to pick from
 - Use the `DropDownButton()` class
 - <https://api.flutter.dev/flutter/material/DropdownButton-class.html>





Workshop - optional

- Optional: Update the Search Countries application with a **checkbox**
 - If the box is checked (`true`), the flag of a country is shown
 - If the box is not checked (`false`), the flag is hidden
 - Use the `Checkbox` class
 - <https://api.flutter.dev/flutter/material/Checkbox-class.html>
- Also create a Country Class (model) and use it

