

Flutter & Dart Complete Guide

Prepared by Eng. Yousef Salah Kassem

DAFL 2
Flutter UI Part 2

Lecture 2



Chapter
02

Table Of Content

- Introduction
- Widgets
- Widget Lifecycle



- Stateless Widget
- Stateful Widget
 - Create State
 - Init State
 - Did Change Dependencies
 - Build
 - Set State
 - Deactive
 - Dispose



Stateless Widget

- A stateless widget never changes.
 - Icon
 - IconButton
 - Text

they are examples of stateless widgets.



Stateless Widget

- Stateless Widgets are those widgets that don't need to deal with the State as they don't change powerfully at runtime. It becomes permanent like on variables, buttons, symbols, and so forth, or any express that can't be changed on the application to recover information.



Stateful Widget

- A widget is either stateful or stateless. If a widget can change—when a user interacts with it, for example—it's stateful.
- A stateful widget is dynamic,
 - for example: it can change its appearance in response to events triggered by user interactions or when it receives data.
- Checkbox, Radio, Slider, InkWell, Form, and Text Field are examples of stateful widgets. Stateful widgets subclass StatefulWidget.



Widgets

- A TextField or TextBox is an input element which holds the alphanumeric data, such as name, password, address, etc.
- It is a GUI control element that enables the user to enter text information using a programmable code.
- It can be of a single-line text field (when only one line of information is required) or multiple- line text field (when more than one line of information is required).



- TextField (
 decoration: InputDecoration(
border: InputBorder.none, labelText: 'Enter Name',
hintText: 'Enter Your Name'
),);



- decoration: It is used to show the decoration around TextField.
- border: It is used to create a default rounded rectangle border around TextField.
- labelText: It is used to show the label text on the selection of TextField.
- hintText: It is used to show the hint text inside TextField.
- icon: It is used to add icons directly to the TextField.



- Flutter does not have an ID like in Android for the TextField widget. Flutter allows the user to retrieve the text in mainly two ways: First is the onChanged method, and another is the controller method. Both are discussed below:
- – 1. onChanged method: It is the easiest way to retrieve the text field value. This method store the current value in a simple variable and then use it in the TextField widget. Below is the sample example:



```
String value = "";  
TextField(  
  onChanged: (text) {  
    value = text;  
  },)
```



Controller method: It is a popular method to retrieve text field value using Text Editing Controller. It will be attached to the TextField widget and then listen to change and control the widget's text value. Below is the sample code:

```
TextEditingController mycontroller = TextEditingController();
```

```
TextField(
```

```
  controller: mycontroller,
```

```
)
```

```
Sample code for listening to the changes. controller.addListener(() {
```

```
// Do something here });
```

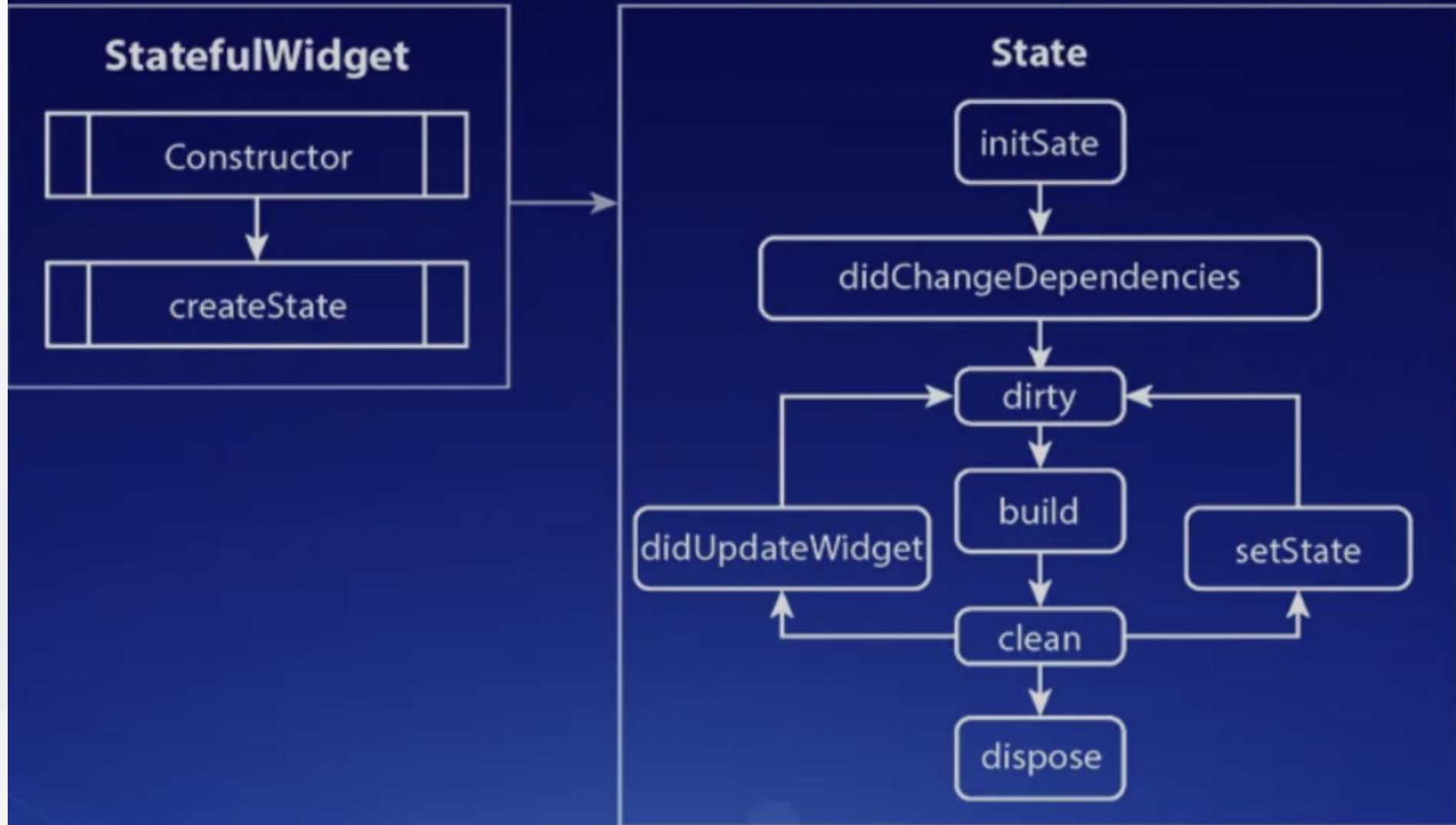


Let us see the second way in detail to retrieve the text field value in Flutter application with the help of following steps:

- Create a Text Editing Controller.
- Attach the Text Editing Controller to a TextField using controller property.
- Retrieve the value of the Text Field by using the text() method provided by the Text Editing Controller.



Widget Lifecycle



Widget Lifecycle

- The lifecycle of the Flutter App is the show of how the application will change its State. It helps in understanding the idea driving the smooth progression of our applications. Everything in Flutter is a Widget, so before thinking about Lifecycle, we should think about Widgets in Flutter



- This method is called when we create another Stateful Widget. It is an obligatory strategy.
- The createState() returns a case of a State-related with it.



Code Example

```
class HomeScreen extends StatefulWidget {  
  
  HomeScreen({Key key}) : super(key: key);  
  
  @override  
  HomeScreenState<StatefulWidget> createState() =>  
    HomeScreen();  
}
```

- This is the strategy that is considered when the Widget is made interestingly and it is called precisely once for each State object. If we characterize or add some code in the **initState()** method this code will execute first even before the widgets are being built.
- Real Example is known for Splash Screen for any application.



Code Example

```
@override  
void initState() {  
  super.initState();  
  Timer(const Duration(seconds: 3),  
    ()=> Navigator.pushReplacement(  
      context,  
      PageTransition(child: const HomePage(),  
        ),),),);  
}
```

Did Change Dependency

- This method is called following the **initState()** method whenever the widget initially is constructed. You can incorporate not many functionalities like API calls dependent on parent data changes, variable re-initializations, and so forth.



Code Example

```
@override  
void didChangeDependencies() {  
  
}
```

- This strategy is the main method as the rendering of all the widgets relies upon it. It is called each time when we need to render the UI Widgets on the screen.



Code Example

```
@override  
Widget build(BuildContext context) {  
  return Scaffold()  
}
```

didUpdate Widget

- This strategy is utilized when there is some adjustment of the configuration by the Parent widget. It is essentially called each time we hot reload the application for survey the updates made to the widget



Code Example

```
@protected  
void didUpdateWidget(Home oldWidget) {  
    super.didUpdateWidget(oldWidget);  
}
```

- The `setState()` method illuminates the framework that the internal state of this item has changed in a manner that may affect the UI which makes the structure plan a build for this State of the object.



Code Example

```
void function(){  
  setState(() {});  
}
```

deactivate

- This method is considered when the State is removed out from the tree, however, this strategy can be additionally be re-embedded into the tree in another part.



Code Example

```
@override  
void deactivate(){  
  super.deactivate();  
}
```

- This strategy is essentially something contrary to the `initState()` method and is likewise important. It is considered when the object and its State should be eliminated from the Widget Tree forever and won't ever assemble again.



Code Example

```
@override  
void dispose(){  
  super.dispose();  
}
```

Live Coding

9:41

Reset Password

Enter your new password twice below to reset a new password

Enter new password

152@@##PAss

Re-enter new password

.....

Reset Password

Create an account

9:41

Create an account

Name

John Doe

Email Address

hello@example.com

Password

.....

By continuing, you agree to our [terms of service](#).

Sign up

or

 Continue with Google

Already have an account? [Sign in here](#)

What is Next ?

<https://zapp.run/>



Thank You