Le widget TextField dans Flutter

1. Objectif

- Apprendre à utiliser un widget TextField en flutter et ses propriétés en détail

2. Présentation

- Dans Flutter,le widget `TextField` est utilisé pour obtenir une entrée en caractères alphanumériques de l'utilisateur de l'application.
- `TextField` permet à l'utilisateur de l'application d'entrer des données dans l'application.
- Le widget `TextField` est le plus souvent utilisé pour créer des formulaires de saisie d'application où le développeur doit demander plusieurs types d'informations à l'utilisateur.
- Nous pouvons également ajouter plusieurs attributs avec `TextField`, tels que l'étiquette, l'icône, le texte d'indication en ligne et le texte d'erreur en utilisant un **InputDecoration** comme décoration. Si nous voulons supprimer entièrement les propriétés de décoration, il est nécessaire de définir la décoration sur null .
- Certains des attributs les plus couramment utilisés avec le widget TextField sont les suivants :

- `decoration` : pour afficher la décoration autour du champ de texte.
- `border` : crée une bordure rectangulaire arrondie par défaut autour du champ de texte.
- `labelText` : pour afficher le texte de l'étiquette lors de la sélection du champ de texte.
- `hintText` : pour afficher le texte de l'indice dans le champ de texte.
- `icon` : Il est utilisé pour ajouter des icônes directement au TextField.

3. Créer un TextField dans Flutter

- La création d'un widget `TextField` de base est simple. Il suffit d'insérer ce widget dans l'arborescence de widgets à l'endroit où vous souhaitez qu'il apparaisse.
- Le code suivant explique un exemple de démonstration du widget `TextFiled` dans Flutter.
- Commencer par créer un nouveau projet dont le nom "exemple_text_field"
- Dans le fichier "**main.dart**" importer le package `material.dart` dans l'application .

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

- Appeler la classe principale `MyApp` en utilisant la fonction **void main run app**.

Ex01

```
void main() {
  runApp(const MyApp());
}
```

Ex02

```
void main() => runApp(const MyApp());
```

- Créer la classe de widget principale nommée `MyApp` qui s'étend avec le widget sans état.

```
class MyApp extends StatelessWidget {
.....
.....
}
```

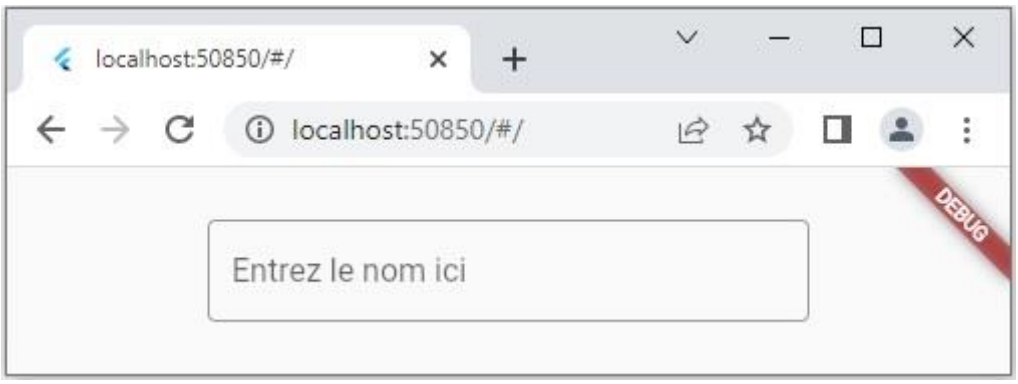
- Créer un widget `Scaffold` -> Widget `center` ->Widget `column` dans la zone "**build**" du widget dans la classe précédemment crée.

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
            body: Center(
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: const [])));
  }
}
```

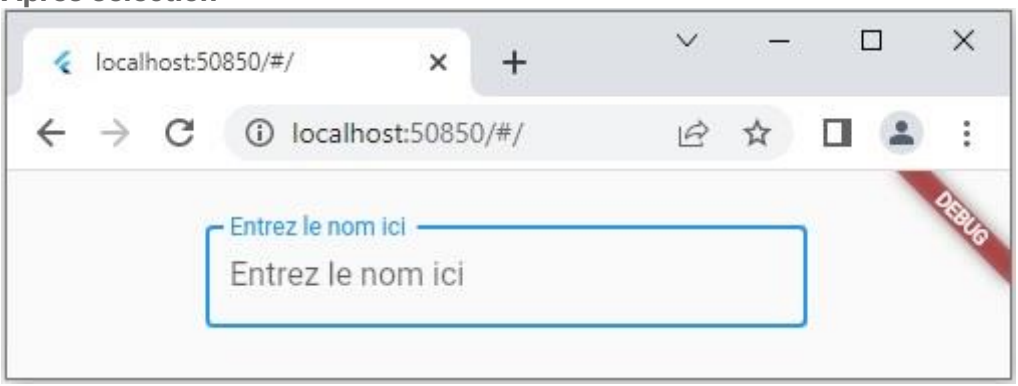
- Créer un widget `Conteneur` puis placer-y le widget `TextField` .
- Le widget **TextField** ne supportait pas directement la largeur et la hauteur.

```
Container(
            width: 300,
            child: TextField(
                decoration: InputDecoration(
                border: OutlineInputBorder(),
                labelText: 'Entrez le nom ici',
                hintText: 'Entrez le nom ici',
                 ),
                  autofocus: false,
                 )
               )
```

Capture d'écran Avant sélection

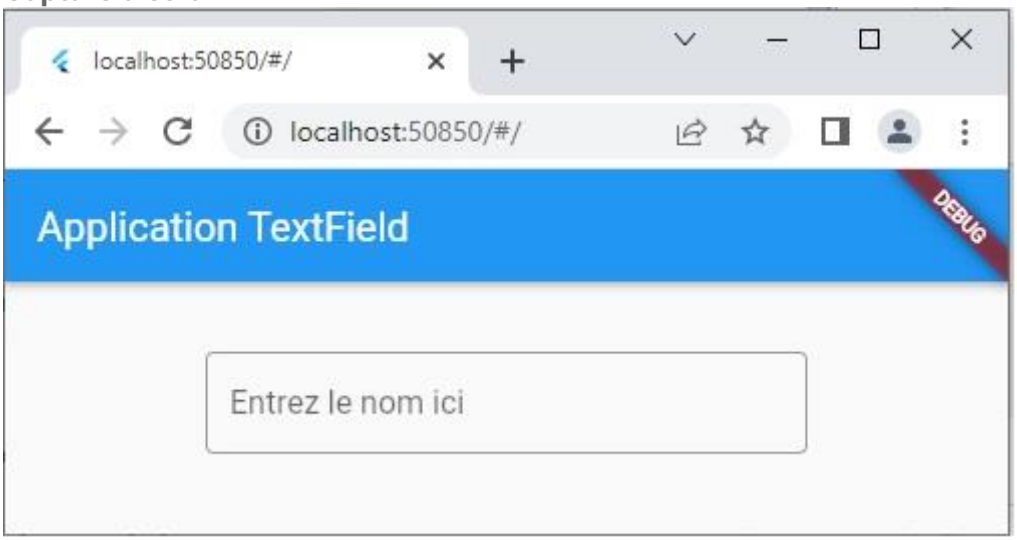


Après sélection



- Ajouter un AppBar à l'exemple précédent

Capture d'écran



Code à ajouter

```
home: Scaffold(
            appBar: AppBar(
              title: const Text('Application TextField'),
            ),
            body: Center(
                .........
```

- Voyons le code source complet qui contient le widget **TextField**.

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

```
void main() => runApp(const MyApp());
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        home: Scaffold(
            appBar: AppBar(
              title: const Text('Application TextField'),
            ),
            body: Center(
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                  // ignore: sized_box_for_whitespace
                  Container(
                      width: 300,
                      child: const TextField(
                        decoration: InputDecoration(
                          border: OutlineInputBorder(),
                          labelText: 'Entrez le nom ici',
                          hintText: 'Entrez le nom ici',
                        ),
                        autofocus: false,
                      ))
                ])))));
  }
}
```

Flutter `Textfield` Style

Comment personnaliser un TextField dans Flutter ?

- Par défaut, un `TextField` est décoré d'un soulignement. Vous pouvez ajouter une étiquette, une icône, un texte d'indication en ligne et un texte d'erreur en fournissant un **InputDecoration** comme propriété de décoration du `TextField` .
- Pour supprimer entièrement la décoration (y compris le soulignement et l'espace réservé à l'étiquette), définissez **InputDecoration** sur null.

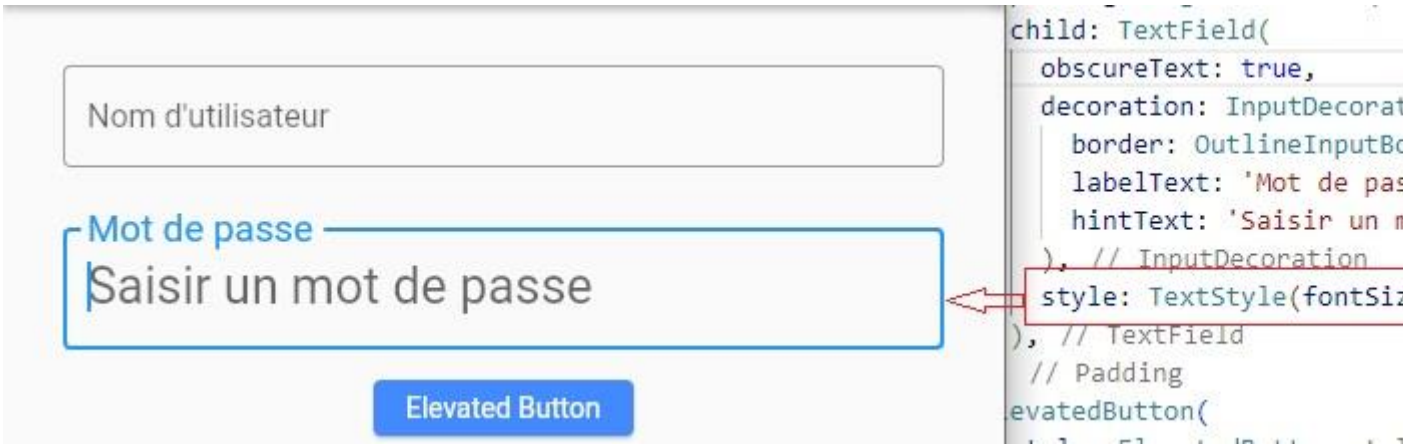
Qu'est-ce que `InputDecoration` dans Flutter ?

`InputDecorator` affiche les éléments visuels d'un champ de texte Material Design autour de son enfant d'entrée. Les éléments visuels eux-mêmes sont définis par un objet InputDecoration et leur disposition et leur apparence dépendent des paramètres baseStyle , textAlign , isFocused et isEmpty .

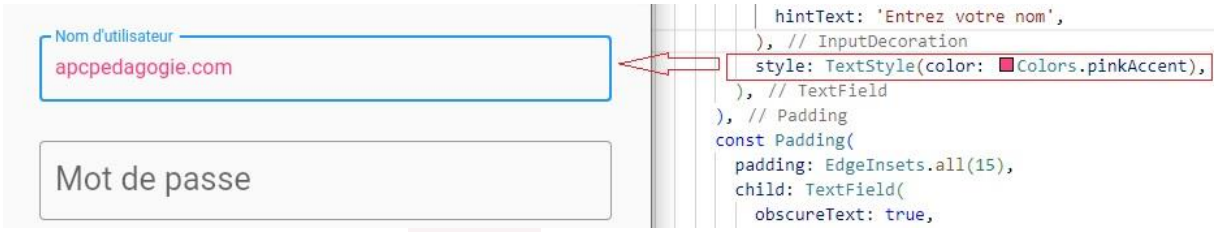
```
TextField(
   obscureText: true,
   decoration: InputDecoration(
   border: OutlineInputBorder(),
   labelText: 'Mot de passe',
```

```
    hintText: 'Saisir un mot de passe',
    ),
),
```

- **Comment changer la police `TextField` dans Flutter ?**
- Vous modifiez la couleur du texte d'entrée dans cette ligne `style: TextStyle(fontSize: 25.0, color: Colors.blueAccent),` , donc pour définir le blanc, utilisez simplement Colors. constante blanche au lieu de textTheme.



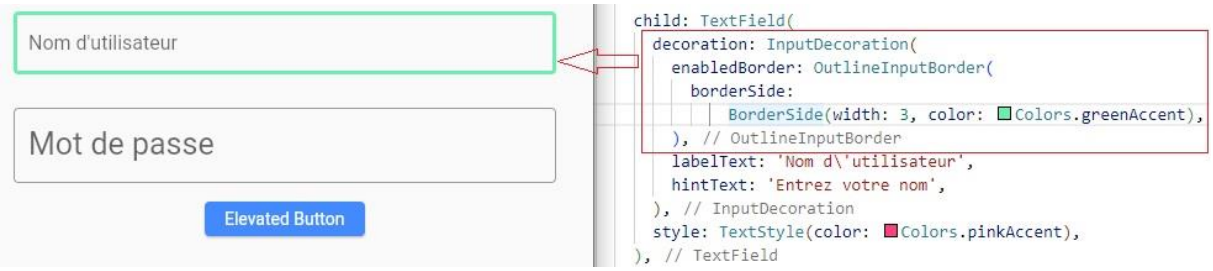
- **Comment stylisez-vous les étiquettes TextField ?**
- `const styles = theme => ({ textField : { width : '90 %', marginLeft : 'auto', marginRight : 'auto', color : 'white', paddingBottom : 0, marginTop : 0, fontWeight : 500 }, } );`
- **Comment changer la couleur du `TextField` ?**
- La couleur du texte peut être modifiée à l'aide de la propriété `style` du widget **TextField**. Vous pouvez également modifier la couleur du curseur en définissant la couleur sur la propriété `cursorColor`.
- Vous pouvez procédez comme suit:
- Sélectionnez la forme ou la zone de texte.
- Dans l'onglet Format des outils de dessin, cliquez sur Remplissage du texte > Plus de couleurs de remplissage.
- Dans la zone Couleurs, cliquez sur la couleur souhaitée dans l'onglet Standard ou mélangez votre propre couleur dans l'onglet Personnalisé.



- **Comment décorer la bordure `TextField` dans Flutter ?**
- Pour ajouter une bordure à un TextField/TextFormField dans Flutter, vous pouvez spécifier la propriété de décoration, puis utiliser les widgets **InputDecoration** et **OutlineInputBorder**.
- Pour le faire quatre étapes:

- **1-** Localisez le fichier dans lequel vous avez placé le widget **TextField/TextFormField**.
- **2-** Dans le widget **TextField/TextFormField**, ajoutez le paramètre de décoration et affectez le widget InputDecoration.

- **3-** Dans le widget **InputDecoration**, ajoutez le paramètre **enabledBorder** et affectez le widget **OutlineInputBorder**.
- **4-** À l'intérieur de **OutlineInputBorder**, ajoutez le widget **BorderSide** avec le paramètre de couleur et définissez la couleur de votre choix.



- **Comment ajouter un rayon de bordure ou une bordure arrondie à `TextField` ou `TextFormField` ?**

- Pour ajouter un rayon de bordure ou créer une bordure arrondie autour du widget `TextField/TextFormField`.
- Ajoutez la propriété `decoration` puis utilisez le widget `OutlineInputBorder`.
- Le widget `OutlineInputBorder` accepte le paramètre `borderRadius`.
- Vous pouvez utiliser le paramètre `borderRadius` avec `BorderRadius.circular(50.0)` pour créer la bordure circulaire autour du TextField.

- **Comment ajouter du texte d'indice d'un TextField?**

- Le texte d'indication est utilisé pour donner aux utilisateurs une idée des valeurs d'entrée acceptées par le **TextField**.
- Vous pouvez utiliser la propriété `hintText` pour ajouter un indice au champ de texte qui disparaîtra lorsque vous commencerez à taper.
- La couleur par défaut est le gris, mais vous pouvez ajouter la propriété `hintStyle` pour modifier le style du texte:***`hintText:"Enter votre email"`***

- **Comment ajouter de la prise en charge multiligne pour un TextField?**

- Par défaut, **TextField** 'affiche sur une seule ligne. Mais nous pouvons spécifier le nombre maximum de lignes à supporter via la propriété `maxLines`.
- Cela ne limitera pas le nombre de lignes que vous pouvez ajouter, cela n'affiche que le nombre spécifié de lignes à la fois.
- Si vous souhaitez développer le champ en fonction de la quantité de texte saisi, vous pouvez définir la propriété `maxLines` à `null`.

- **Comment modifier les claviers en fonction du type d'entrée pour un TextField?**

- Pour afficher différentes dispositions de clavier pour différents types de saisie, comme des pavés numériques pour les numéros de téléphone ou un bouton "@" pour les e-mails, utilisez la propriété `keyboardType`. Il accepte plusieurs options comme `TextInputType` le numéro, la date, le téléphone, le nom et l'adresse e-mail.

- **Comment convertir un TextField normal en TextField de mot de passe?**

- En définissant la propriété `obscureText` sur, true vous pouvez convertir un champ de texte brut en un champ de mot de passe, qui masque les valeurs d'entrée.
- La valeur par défaut de cette propriété affichera des points pour masquer les caractères du mot de passe. Mais vous pouvez changer cela en définissant la valeur `obscuringCharacter` sur ce que vous voulez ;

- **Comment limiter nombre de caractères d'un TextField?**
- La propriété `maxLength` accepte des valeurs entières pour spécifier le nombre maximal de caractères acceptés par le champ particulier.
- Après avoir ajouté cette propriété, si les utilisateurs saisissent une valeur avec plus de caractères que spécifié dans `maxLength`, cela bloquera automatiquement la saisie: ***`TextField (maxLength : 2 ,)`***.

Application

- **01-** Réaliser une application flutter contenant une interface utilisateur similaire à la capture d'écran suivante :

- **02-** Un clique à l'intérieur de l'une des zones de texte, permet d'afficher un clavier en bas de l'écran, l'étiquette va dans le coin supérieur gauche de la bordure et le texte d'indication affiché dans le champ. La capture d'écran ci-dessous l'explique plus clairement :

Solution

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

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: Home(),
    );
  }
}

class Home extends StatefulWidget {
  const Home({Key? key}) : super(key: key);

  @override
  _HomeState createState() => _HomeState();
}

class _HomeState extends State {
  TextEditingController username = TextEditingController();
  TextEditingController password = TextEditingController();
  @override
  void initState() {
    username.text = ""; //innitail value of text field
    password.text = "";
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: const Text('Utiliser les TextField',
              style: TextStyle(color: Color.fromARGB(255, 7, 49, 233))),
          backgroundColor: const Color.fromARGB(255, 64, 249, 255),
        ),
        body: Container(
          padding: const EdgeInsets.all(20),
          child: Column(
            children: [
              TextField(
                  controller: username,
                  decoration: InputDecoration(
                    labelText: "Nom d'utilisateur",
                    prefixIcon: const Icon(Icons.people),
```

```
                    border: mes_bordures(),
                    enabledBorder: mes_bordures(),
                    focusedBorder: mes_focusborder(),
                  )),
              Container(height: 20),
              TextField(
                  controller: password,
                  decoration: InputDecoration(
                    prefixIcon: const Icon(Icons.lock),
                    labelText: "Mot de passe",
                    enabledBorder: mes_bordures(),
                    focusedBorder: mes_focusborder(),
                  )),
            ],
          ),
        ));
  }

  // ignore: non_constant_identifier_names
  OutlineInputBorder mes_bordures() {
    //return type is OutlineInputBorder
    return const OutlineInputBorder(
        //Outline border type for TextFeild
        borderRadius: BorderRadius.all(Radius.circular(10)),
        borderSide: BorderSide(
          color: Colors.redAccent,
          width: 3,
        ));
  }

  // ignore: non_constant_identifier_names
  OutlineInputBorder mes_focusborder() {
    return const OutlineInputBorder(
        borderRadius: BorderRadius.all(Radius.circular(10)),
        borderSide: BorderSide(
          color: Colors.greenAccent,
          width: 3,
        ));
  }

  //create a function like this so that you can use it wherever you want
}
```