

Flutter Basic GUI

Flutter L2, CAL Computer Science Club

© 2025 by Michael Y. Qiu, licensed under CC BY-NC 4.0.

To view a copy of this license, visit <https://creativecommons.org/licenses/by-nc/4.0/>

Recap: Install Flutter

1. 下载 Flutter SDK
2. 将 Flutter SDK 解压并放置在任意文件夹 FLUTTER_PATH
3. 添加 PATH:

```
export PATH=[FLUTTER_INSTALL_PATH]/flutter/bin:$PATH
```

4. 刷新 PATH:

```
source $HOME/.bash_profile
```

5. iOS 开发: 安装 Xcode
Android 开发: 安装 Android Studio

Configure Flutter: Proxy

卡在 Waiting for another flutter command to release the startup lock?

Flutter 需要从 Google 下载相关 packages

```
killall -9 dart // Mac or Linux  
taskkill /F /IM dart.exe // Win
```

```
export PUB_HOSTED_URL=https://pub.flutter-io.cn // Mac or Linux  
export FLUTTER_STORAGE_BASE_URL=https://storage.flutter-io.cn
```

```
PUB_HOSTED_URL ===== https://pub.flutter-io.cn // Win  
FLUTTER_STORAGE_BASE_URL ===== https://storage.flutter-io.cn
```

Everything is Widget

Entrance to App

- 所有 Dart 程序从 `void main()` 开始
- `runApp(Widget)` 启动整个应用
- `MaterialApp` 是 Material Design 风格的根容器

Entrance to App

- 所有 Dart 程序从 `void main()` 开始
- `runApp(Widget)` 启动整个应用
- `MaterialApp` 是 Material Design 风格的根容器

- `title`: 应用名称
(用于任务管理等)
- `home`: 首页 `Widget`
(必须是 `Widget` 子类)

```
1 void main() {  
2     runApp(MyApp()); // 启动应用  
3 }  
4  
5 class MyApp extends StatelessWidget {  
6     @override  
7     Widget build(BuildContext context) {  
8         return MaterialApp(  
9             title: '视频下载器',  
10            theme: ThemeData(useMaterial3: true),  
11            home: VideoDownloaderPage(), // 首页  
12        );  
13    }  
14 }
```

@override

- @override 表示 重写父类方法（非 Dart 特有，Java/C# 也有）
- 在 Flutter 中，几乎所有 Widget 都要重写 build() 方法
- 虽然 Dart 允许省略 @override，但强烈建议保留（提高可读性、避免错误）

```
1  @override
2  Widget build(BuildContext context) {
3      // 返回 UI 结构
4  }
```

build()

描述“当前状态下
UI 应该长什么样”

BuildContext

BuildContext 是 Widget 在树中的位置标识，用于

- i. 查找祖先 Widget (如 Theme.of(context))
- ii. 导航 (Navigator.of(context))
- iii. 显示对话框等
- 简单来讲，就是“上下文环境”，且每次 build 都会传入

```
1  @override
2  Widget build(BuildContext context) {
3      // 返回一个 Widget 子树
4  }
```

build() 必须返回一个 非空 Widget
(不能返回 null)

Widget

Flutter 的核心思想:

- UI 由 Widget (组件) 构成
- Widget 是不可变的 (immutable)
- 每次状态变化 → 重建 Widget 树

两种 Widget:

- StatelessWidget: 无状态, 静态内容
- StatefulWidget: 有状态, 可交互

Widget

两种 Widget:

- StatelessWidget:
无状态, 静态内容
- StatefulWidget:
有状态, 可交互

StatelessWidget

StatefulWidget

内容固定, 无用户交互

内容随状态变化

无内部状态

有 State 对象管理状态

重写 build() 即可

需配合 State类 + setState()

Widget

两种 Widget:

- StatelessWidget:
无状态, 静态内容
- StatefulWidget:
有状态, 可交互

StatelessWidget

StatefulWidget

内容固定, 无用户交互

内容随状态变化

无内部状态

有 State 对象管理状态

重写 build() 即可

需配合 State类 + setState()

StatefulWidget → 输入框、按钮点击、加载状态、动画等交互反馈

StatefulWidget & State<T>

Flutter 要求

StatefulWidget 拆分为两部分：

- UI 描述类（不可变）
- 状态管理类（可变）

- `_VideoDownloaderPageState` 是私有类（下划线开头）
- `createState()` 由框架调用，创建状态对象

```
1 class VideoDownloaderPage extends StatefulWidget {  
2   @override  
3   _VideoDownloaderPageState createState() =>  
4     _VideoDownloaderPageState();  
5  
6   class _VideoDownloaderPageState extends  
7     State<VideoDownloaderPage> {  
8     @override  
9     Widget build(BuildContext context) {  
10       return Scaffold(...);  
11     }  
12   }
```


StatefulWidget & State<T>

State<T>

- StatefulWidget 本身 不包含状态，只负责创建 State 对象
- State<VideoDownloaderPage> 表示“这个 State 对象专属于 VideoDownloaderPage 这个 Widget”

```
1 class VideoDownloaderPage extends StatefulWidget {
2   @override
3   _VideoDownloaderPageState createState() =>
4     _VideoDownloaderPageState();
5 }
6 class _VideoDownloaderPageState extends
7   State<VideoDownloaderPage> {
8   @override
9   Widget build(BuildContext context) {
10     return Scaffold(...);
11   }
```

泛型 <T> 确保 widget 属性在 State 中类型安全

setState()

让 UI 响应变化

- State 类中的变量 = 状态
- 修改状态后，必须调用 `setState(() { ... })`
- Flutter 会重新调用 `build()`，刷新 UI

```
1 String _status = "就绪";  
2  
3 void _updateStatus() {  
4     setState(() {  
5         _status = "状态已更新";  
6     });  
7 }
```

- 状态必须在 `setState()` 内修改
- 直接 `_status = "...";` UI 不会更新

Scaffold

Scaffold, 标准页面容器, 实现 Material Design 布局结构, 提供:

- appBar: 顶部栏
- body: 主要内容区
- floatingActionButton (可选)
- 自动处理 SafeArea、键盘适配等



```
1 Scaffold({  
2     AppBar? appBar,  
3     Widget? body,  
4     EdgeInsetsGeometry? padding,  
5     Color? backgroundColor,  
6 })
```



```
1 Scaffold(  
2     appBar: AppBar(title: Text("下载器")),  
3     body: Center(child: Text("内容区")),  
4 )
```

Scaffold 本身是一个 Widget, 可直接作为 build() 返回值

Scaffold Body

```
1  @override
2  Widget build(BuildContext context) {
3    return Scaffold(
4      appBar: AppBar(title: Text("视频下载器")),
5      body: Padding(
6        padding: const EdgeInsets.all(16.0),
7        child: Column(
8          crossAxisAlignment: CrossAxisAlignment.start,
9          children: [
10           TextField(..."视频链接"...),
11           SizedBox(height: 20),
12           ElevatedButton(..."开始下载")),
13           SizedBox(height: 20),
14           Text(..._status...),
15         ],
16       ),
17     ),
18   );
19 }
```

Padding

- Padding 不是属性，而是一个布局 Widget
- 用于在子元素周围添加空白

- `EdgeInsets.only(left: 10)`
- `EdgeInsets.symmetric(vertical: 20)`



```
1  Padding({  
2    required EdgeInsetsGeometry padding,  
3    required Widget child,  
4  })
```



```
1  Padding(  
2    padding: EdgeInsets.all(16.0), // 四周 16px  
3    child: Column(...),  
4  )
```


Column

- Column 将子 Widget 从上到下排列
- main axis 为垂直
cross axis 为水平

```
1 Column({  
2   MainAxisAlignment mainAxisAlignment = MainAxisAlignment.start,  
3   CrossAxisAlignment crossAxisAlignment = CrossAxisAlignment.center,  
  
4   List<Widget> children = const [],  
5 })
```

```
1 Column(  
2   crossAxisAlignment: CrossAxisAlignment.start,  
3   children: [TextField(), ElevatedButton(), Text()],  
4 )
```

children 必须是非空 List<Widget>

TextField

- TextField 用于单行文本输入
- 必须搭配 TextEditingController 才能读取/控制内容
- decoration 用于设置提示文字、边框等

```
1 TextField({  
2   TextEditingController? controller,  
3   InputDecoration? decoration,  
4   bool obscureText = false,  
5   VoidCallback? onTap,  
6 })
```

```
1 final TextEditingController _urlController =  
   TextEditingController();  
2  
3 TextField(  
4   controller: _urlController, // 绑定控制器  
5   decoration: InputDecoration(  
6     labelText: "粘贴 YouTube 链接",  
7     border: OutlineInputBorder(), // 显示边框  
8   ),  
9 )
```

TextField & TextEditingController

- TextField 用于单行文本输入
- 必须搭配 TextEditingController 才能读取/控制内容
- decoration 用于设置提示文字、边框等

```
1 TextField({
2     TextEditingController? controller,
3     InputDecoration? decoration,
4     bool obscureText = false,
5     VoidCallback? onTap,
6 })
```

```
1 final TextEditingController _urlController =
   TextEditingController();
2
3 TextField(
4     controller: _urlController, // 绑定控制器
5     decoration: InputDecoration(
6         labelText: "粘贴 YouTube 链接",
7         border: OutlineInputBorder(), // 显示边框
8     ),
9 )
```

TextField & TextEditingController

- TextField 本身不存储文本
- TextEditingController 是独立控制器，用于：
 - i. 读取: `controller.text`
 - ii. 写入: `controller.text = "new"`
 - iii. 监听变化:
`controller.addListener(...)`

```
1 final TextEditingController _urlController  
  = TextEditingController();  
2  
3 TextField(controller: _urlController)
```

必须在 State 中声明，不能在 build 内创建，否则会丢失状态

ElevatedButton

- ElevatedButton 是带阴影的 Material 风格按钮
- onPressed 为 null 时按钮自动禁用
- child 通常是 Text

```
1 ElevatedButton({  
2   required VoidCallback onPressed,  
3   required Widget child,  
4   ButtonStyle? style,  
5 })
```

```
1 ElevatedButton(  
2   onPressed: _download, // 点击时调用函数  
3   child: Text("开始下载"),  
4   style: ElevatedButton.styleFrom(  
5     minimumSize: Size(double.infinity, 50), // 宽度100%, 高度50  
6   ),  
7 )
```

SizedBox

SizedBox 可用于:

- 设置固定宽高 (width, height)
- 仅作间距 (只设 height 或 width, 不设 child)



```
1  SizedBox(height: 20) // 在元素间插入 20px 垂直空白
```




Reference & Resources

- Flutter 运行卡在
Waiting for another flutter command to release the startup lock
<https://juejin.cn/post/7232158879428132924>
- Why is Flutter run taking forever
<https://stackoverflow.com/questions/59265825/why-is-flutter-run-taking-forever>
- Flutter 从入门到入土
<https://www.bilibili.com/video/BV1GJ411x7h7/>