



flutter project

CHAPTER 04

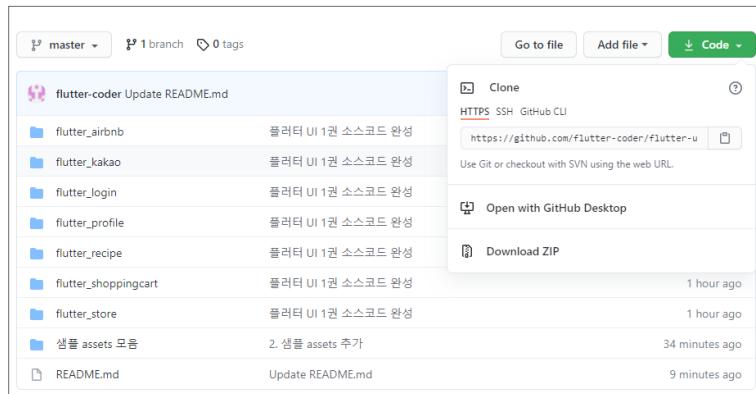
플러터 위젯 -스토어 앱 만들기

이번 장에서는 MaterialApp, Scaffold, Column, Row, Text, SafeArea, Image, Spacer, Expanded, Padding, SizedBox 위젯에 대해서 알아보는 시간을 가지겠습니다.

04_1 스토어 앱 구조보기

모든 소스 코드는 다음 깃허브 경로에 공개되어 있습니다.

- <https://github.com/flutter-coder/flutter-ui-book1>



◆ github 소스코드 다운로드 방법



◆ 교재 진행에 필요한 이미지, 폰트, 로고가 모여 있는 폴더

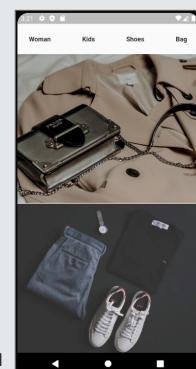
TIP 전체 소스코드를 다운

<https://github.com/flutter-coder/flutter-ui-book1> 해당 경로로 이동하여 Code – Download ZIP 버튼을 클릭하면 됩니다.
다운 받으면 좋은 점

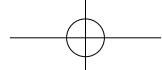
- ① 소스코드를 다운 받아두면 교재를 진행하다가 막히는 부분의 소스코드를 참고할 수 있습니다.
- ② 샘플 assets 모음 폴더에 교재 진행에 필요한 이미지가 모여 있습니다.

TIP

github에서 다운 받은 프로젝트를 실행하려면 아래의 블로그 주소를 참고해주세요.
<https://blog.naver.com/getinthere/222339023005>



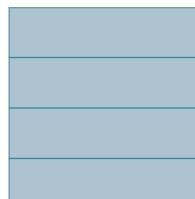
◆ 스토어 앱 완성 화면



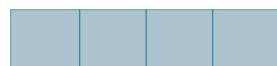
화면 구조보기

위에 앱을 플러터로 만들려고 하면 다음과 같이 할 수 있습니다.

첫째, 전체 구성의 흐름이 수직인지 수평인지를 확인합니다. 플러터에서 수직은 Column 위젯을 사용합니다. 수평은 Row 위젯을 사용합니다. 우리가 만들 앱의 레이아웃 구조는 Column입니다.

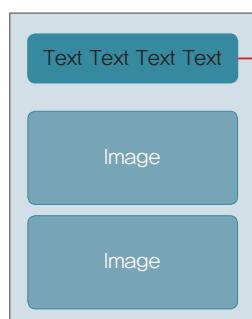


◆ Column 위젯



◆ Row 위젯

둘째, 수직으로 내려가는 구조에서 각각의 위젯을 찾아야 합니다. 플러터에서 글자는 Text 위젯을 사용하고 그림은 Image 위젯을 사용합니다.

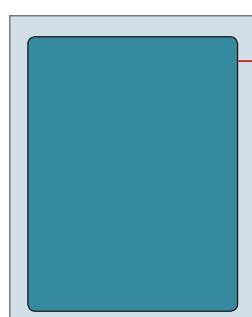


◆ 스토어 앱 화면 구조

MaterialApp vs CupertinoApp

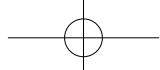
플러터는 쿠퍼티노 디자인 혹은 메타리얼 디자인 둘 중 하나를 선택하여 그림을 그릴 수 있습니다.

두 개의 클래스 중 하나를 선택해야 하는데 MaterialApp은 Android 디자인이고 CupertinoApp은 iOS 디자인입니다. 버튼을 하나 만들어도 무엇을 선택했는지에 따라 디자인이 달라지게 됩니다.

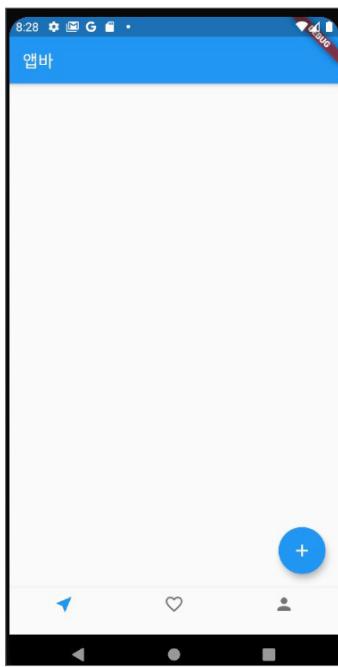


우선 그림과 같이 전체화면을 MaterialApp으로 감쌉니다. 그리고 전체화면에 그림을 그리면 됩니다.

◆ MaterialApp 안드로이드 디자인



Scaffold



◆ Scaffold 구조

대부분의 휴대폰의 최상단에는 AppBar가 있습니다. AppBar는 해당 화면에 대한 메뉴나 이동 버튼, 그리고 제목 같은 것들을 가지고 있습니다.

아래 그림을 보면 화면 중간 하얀 도화지 부분은 body 부분입니다. 이 도화지 부분에 여러분들이 그림을 그리면 됩니다. 오른쪽 밑에 [+] 버튼이 있는 부분을 FloatingActionButton이라고 하고 가장 아래에 BottomNavigationBar가 존재합니다. BottomNavigationBar를 통해 화면을 변경할 수 있습니다.

이러한 구성을 하기 위해서는 직접 그림을 그려도 되지만 이미 그려져 있는 컴포넌트(구성요소)를 재사용하는 것이 좋습니다.

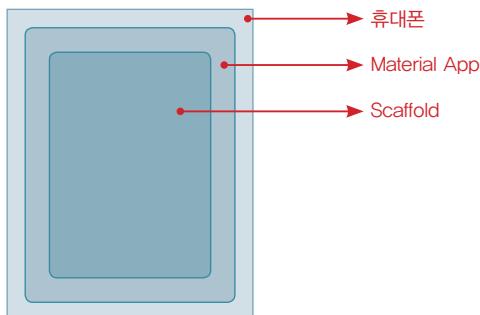
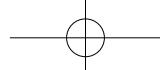
왜 재사용하는 것이 좋을까요?

첫 번째는 내가 직접 만드는 것보다 있는 것을 재사용하는 것이 편합니다. 그렇다면 편하다고 이미 만들어진 컴포넌트를 재사용하는 것이 좋을까요? 내가 직접 만들면 훨씬 더 예쁜 디자인이 나올 수 있을 것 같다면요?

두 번째 이유가 있습니다. 사용자 경험(UX)입니다. 사용자들이 BottomNavigationBar에 이미 익숙해져 있습니다. 인스타그램이나 카카오톡과 같은 수많은 앱을 사용하면서 오래 동안 경험한 디자인을 변경하게 되면 사용자의 UX(사용자 경험)가 망가지게 됩니다. 아무리 디자인이 예쁘고 화려하다 해도 사용하기 불편하면 의미가 없습니다.

사용자에게 좋은 경험을 줄 수 있도록 플러터에서 개발자에 제공해주는 클래스가 바로 Scaffold입니다. 그래서 MaterialApp 내부를 Scaffold로 감싸야 합니다. Scaffold로 감싸는 순간 휴대폰 화면에 구조가 만들어지고 쉽게 앱을 만들 수 있습니다.

“マイクロソフト에서 탐색기를 닫는 x 버튼을 어디에 두면 좋을지 연구를 했다고 합니다. 이때 Mac에서처럼 왼쪽 위에 두는 것 이 좋다는 연구결과가 나왔다고 합니다. 하지만 마이크로소프트는 x 버튼의 위치를 옮기지 않았습니다. 그 이유는 이미 Windows 를 사용하고 있는 많은 고객들의 사용자경험(UX)를 중요하게 생각했기 때문입니다。



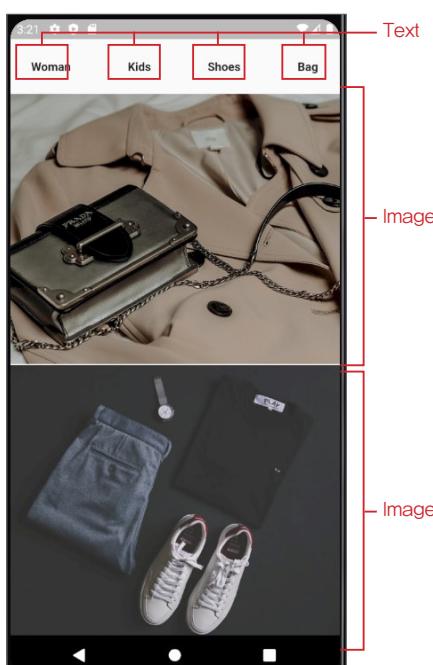
◆ MaterialApp 내부에 Scaffold

이제부터 그림은 Scaffold 안에 그립니다. Scaffold는 구조가 있는 도화지라고 생각하면 됩니다. 그리고 그 구조에 AppBar나 FloatingActionButton을 추가하거나 하는 것은 본인 자유입니다. 내 앱에 필요하면 추가하는 것이고 필요하지 않으면 추가하지 않아도 됩니다.

다만 MaterialApp 안에 Scaffold 구조를 가져야 한다는 것은 꼭 기억하길 바랍니다.

필요한 위젯 살펴보기

처음에 봤던 쇼핑몰 앱에 필요한 위젯을 자세히 살펴보겠습니다. 해당 그림에는 레이아웃에 관련된 Column과 Row 위젯은 가시성을 위해 제외하였습니다.

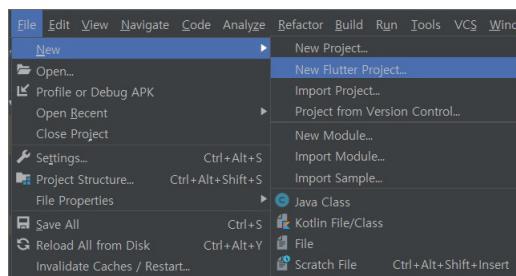


◆ 필요한 위젯 살펴보기

플러터 프로젝트 생성하기

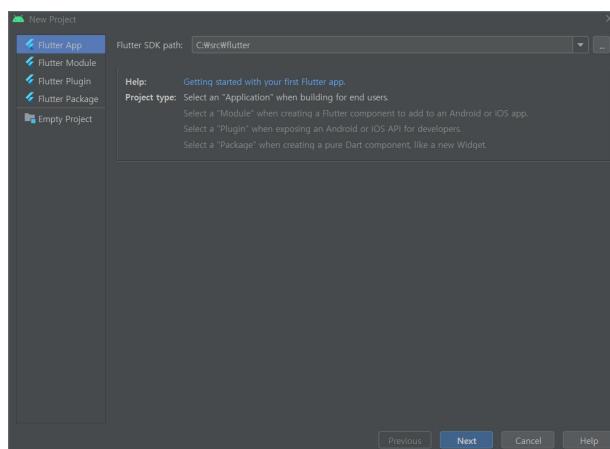
1 안드로이드 스튜디오를 실행합니다.

2 새로운 프로젝트를 생성합니다.



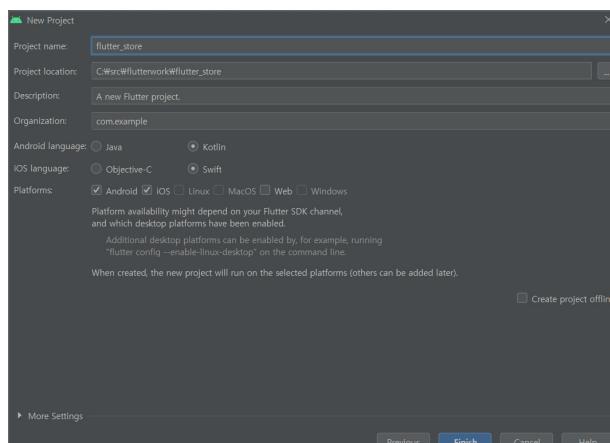
◆ New Flutter Project 선택

3 Flutter App을 선택합니다.



◆ Flutter Application 선택

4 새로운 플러터 애플리케이션 설정 후 Finish 버튼을 클릭합니다.



“ 안드로이드 스튜디오가 최신버전으로 업그레이드 되면서 플러터 프로젝트 생성화면이 변경되었습니다. 5장부터는 구버전의 안드로이드 스튜디오 프로젝트 생성 흐름이 이어지는 점 참고해주세요.”

◆ Flutter 프로젝트 설정 화면

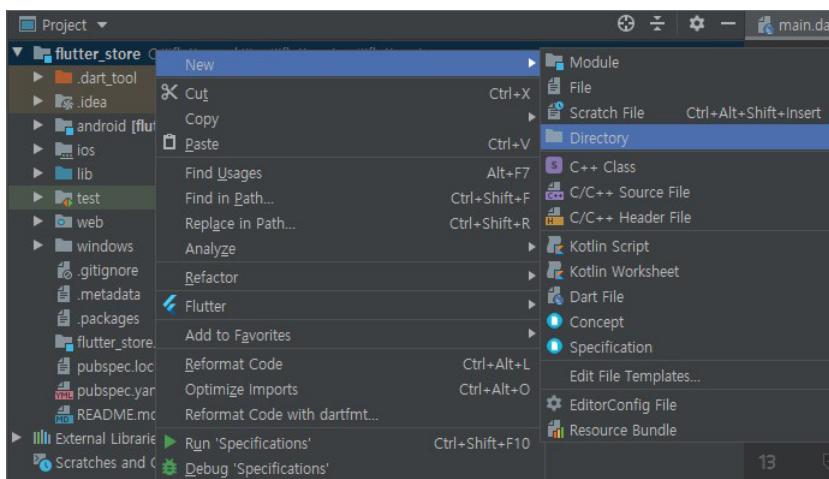
“ Project_name은 flutter_store

Project location은 c:\src\flutterwork\flutter_store

04 _ 2 스토어 앱 뼈대 만들기

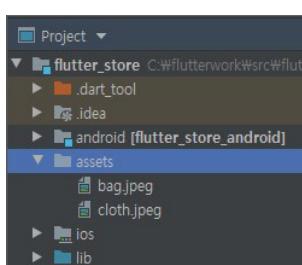
[작업 순서]

1 flutter_store/assets 폴더 생성

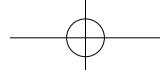


◆ assets 폴더 생성

2 flutter_store/assets에 이미지 추가



◆ 이미지 추가

**TIP****샘플 이미지를 가져오려면!!**

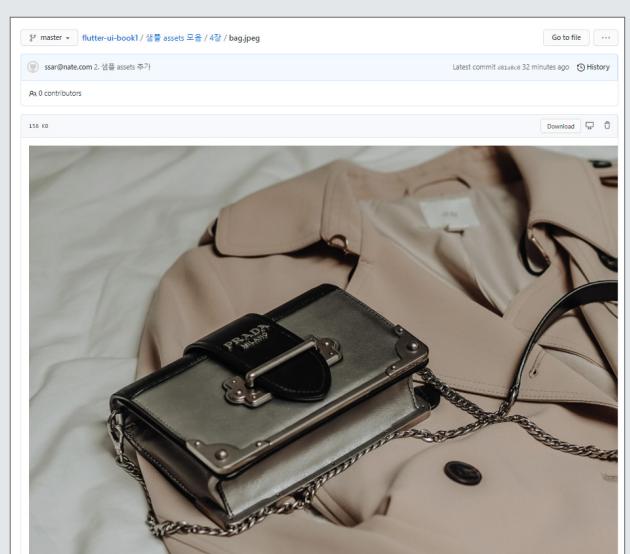
첫째, <https://github.com/flutter-coder/flutter-ui-book1> 경로로 이동합니다.

둘째, 샘플 assets 모음 폴더로 이동합니다.

셋째, 4장 폴더로 이동합니다.

넷째, 4장 폴더에 있는 bag.jpeg, cloth.jpeg 파일을 다운 받습니다.

다섯째, flutter_store 프로젝트의 assets 폴더에 붙여넣기 합니다.



◆ 스토어 앱 만들기에 필요한 이미지

3 pubspec.yaml에서 이미지 파일 인식을 위해 자원 폴더 위치 설정

4 Pub get 버튼을 클릭하여 설정 적용

```
Flutter commands
Pub get Pub upgrade Pub outdated | Flutter doctor

34  sdk: flutter
35
36 # For information on the generic Dart part of this file, see the
37 # following page: https://dart.dev/tools/pub/pubspec
38
39 # The following section is specific to Flutter.
40 flutter:
41
42 # The following line ensures that the MaterialIcons font is
43 # included with your application, so that you can use the icons in
44 # the materialIcons class.
45 uses-material-design: true
46
47 # To add assets to your application, add an assets section, like this:
48 assets:
49   - assets/
```

◆ yaml 파일에 assets 경로 설정 – 들여쓰기 주의

TIP**yaml 파일 규칙**

1 공백 문자를 이용한 두 칸 띄어쓰기로 구조체를 구분한다.

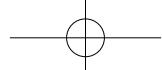
예) flutter:

 assets:

flutter와 assets는 두 칸 띄어쓰기를 통해 구분한다.

2 뒤에 값이 들어올 때는 한 칸 띄어쓰기 후 작성한다.

3 리스트 요소는 하이픈(–)으로 표시한다.



04 _ 3 스토어 앱 만들어보기

기본 코드 작성하기

내부에 있는 모든 코드를 지우고 다음과 같이 만듭니다.

lib/main.dart

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

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

“ flutter를 시작하기 위해서는 main함수에서 runApp() 함수를 호출해야 합니다. runApp 호출 시, 넘겨주는 위젯이 앱의 루트 위젯이 됩니다. 루트 위젯이란 플러터가 그림을 그릴 때 가장 먼저 그리는 위젯입니다. runApp 실행 시 내부적으로는 윈도우 생성, 스케줄러 초기화, 위젯 트리 생성, 렌더링 트리 생성이라는 복잡한 일들이 발생하지만 추상화 되어 있는 함수 내부의 동작 원리를 알 필요는 없습니다.

iOS 디자인을 사용할 것인지 android 디자인을 사용할 것인지 정해야 합니다. MaterialApp을 사용합니다.

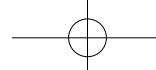
lib/main.dart

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

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

// stl 이라고 적으면 자동완성 기능이 활성화된다.
class MyApp extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            home: StorePage(), // 1. 여기서 오류 있음. StorePage 클래스가 없음.
        );
    }
}
```

StorePage 클래스를 생성하고 내부에 Scaffold를 사용하여 앱을 구조화 시킵니다.



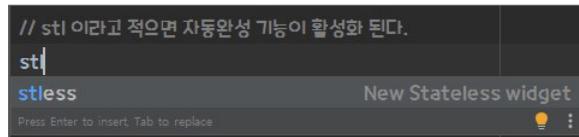
lib/main.dart

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

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

// stl 이라고 적으면 자동완성 기능이 활성화된다.
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: StorePage(),
    );
  }
}

// stl 이라고 적으면 자동완성 기능이 활성화된다.
class StorePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      );
  }
}
```



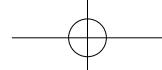
◆ StatelessWidget 자동 완성

Column 위젯

Column 위젯은 수직 방향 레이아웃 구조를 만들어 주고 child가 아닌 children 속성을 가집니다. 스토어 앱은 위에서부터 아래로 내려가는 구조이기 때문에 Column 위젯으로 레이아웃을 잡아줍니다. child 속성을 가진 위젯은 하나의 위젯만 가질 수 있습니다.

final Widget child;

◆ child 속성



children 속성을 가진 위젯은 많은 위젯을 가질 수 있습니다.

```
List<Widget> children = const <Widget>[],
```

- ◆ children 속성

```
lib/main.dart
```

```
...생략
class StorePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Column(
        children: [
          ],
        ), // end of Column
      );
    }
}
```

Row 위젯

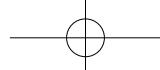
Row 위젯은 수평 방향 레이아웃 구조를 만들어 주고 child가 아닌 children 속성을 가집니다. 아래 그림을 보면 Text 위젯이 수평 방향으로 표시됩니다.



- ◆ Row 위젯은 수평으로 흐른다

```
lib/main.dart
```

```
...생략
body: Column(
  children: [
    Row(
      children: [
        ],
      ), // end of Row
    ),
  ...생략
}
```



Text 위젯

Text 위젯은 문자열을 담을 수 있는 위젯입니다. Text() 위젯을 Row 내부에 추가합니다.

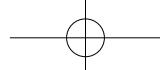


◆ 4.3.4 완성 화면

```
...생략
Row(
  children: [
    Text("Woman"),
    Text("Kids"),
    Text("Shoes"),
    Text("Bag"),
  ],
),
...생략
```

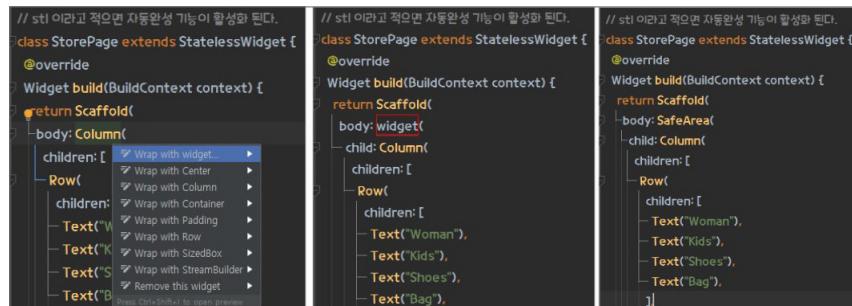
SafeArea 위젯

SafeArea 위젯은 핸드폰 기기별로 조금씩 다른 StatusBar(상태바) 영역에 padding(여백)을 넣어주는 역할을 합니다.



◆ 4.3.5 완성 화면

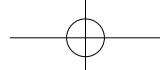
스토어 앱의 Text 위젯이 상태바 영역에 위치해 있습니다. 이때는 Column 글자 위에 커서를 두고 **Alt + Enter** (매직키)를 입력합니다. 그리고 Wrap with widget...을 선택하여 새로운 위젯으로 갑니다. 그리고 SafeArea 위젯으로 변경해줍니다.



◆ 매직키를 사용하여 안드로이드 스튜디오 툴 활용하기

```
lib/main.dart
```

```
lib/main.dart
...
class StorePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Column(
          children: [
            ...
          ],
        ),
      ),
    );
  }
}
```



TIP 자동정렬 단축키

Ctrl + Alt + L

TIP

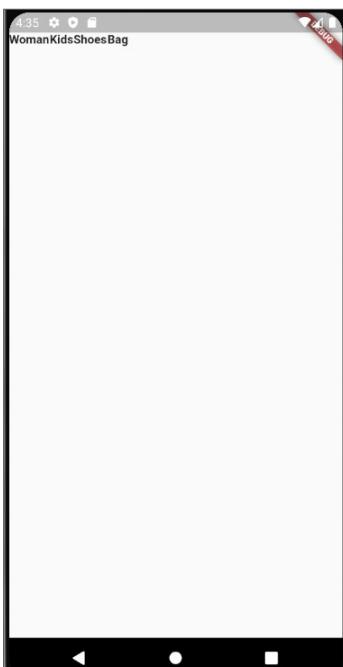
```
// 마지막 Text 위젯 끝에 콤마(,)를 추가했을 때 자동정렬 모습
Row(
  children: [
    Text("Woman"),
    Text("Kids"),
    Text("Shoes"),
    Text("Bag"),
  ],
),
```

TIP

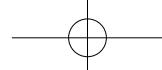
```
// 마지막 Text 위젯 끝에 콤마(,)를 추가하지 않았을 때 자동정렬 모습
Row(
  children: [Text("Woman"), Text("Kids"), Text("Shoes"), Text("Bag")],
),
```

Text 위젯의 style 속성

Text() 위젯을 style 속성을 사용하여 디자인합니다.



◆ 4.3.6 완성 화면



Text 위젯에 **Ctrl** +마우스 왼쪽을 클릭하면 아래와 같은 코드를 볼 수 있습니다.

```
const Text(  
  this.data,  
  Key key,  
  this.style,  
  this.strutStyle,  
  this.textAlign,  
  this.textDirection,  
  this.locale,  
  this.softWrap,  
  this.overflow,  
  this.textScaleFactor,  
  this.maxLines,  
  this.semanticsLabel,  
  this.textWidthBasis,  
  this.textHeightBehavior,  
) : assert(
```

◆ Text 위젯 속성

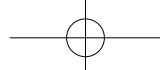
style, overflow, maxLines와 같은 것들을 속성(Property)이라고 부릅니다. 속성이란 어떤 대상을 구성하고 있는 요소라고 생각하면 됩니다.

예를 들어 커피라는 대상(오브젝트)이 있을 때 거기에 필요한 속성을 생각해보면 커피 이름, 아이스인지 핫인지, 큰 사이즈인지 중간 사이즈인지 작은 사이즈인지를 정의할 수 있습니다.

```
const 커피 {  
  this.name,  
  this.isCold,  
  this.size,  
}
```

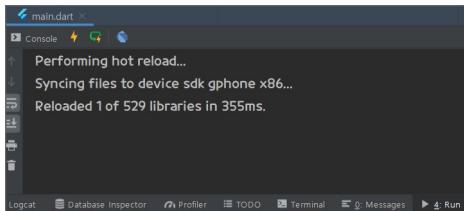
Text 위젯이 가지고 있는 고유한 속성 중에서 style이라는 속성을 이용하여 다음과 같이 디자인 해보겠습니다.

```
...생략  
Row(  
  children: [  
    Text("Woman", style: TextStyle(fontWeight: FontWeight.bold)),  
    Text("Kids", style: TextStyle(fontWeight: FontWeight.bold)),  
    Text("Shoes", style: TextStyle(fontWeight: FontWeight.bold)),  
    Text("Bag", style: TextStyle(fontWeight: FontWeight.bold)),  
  ],  
,  
...생략
```



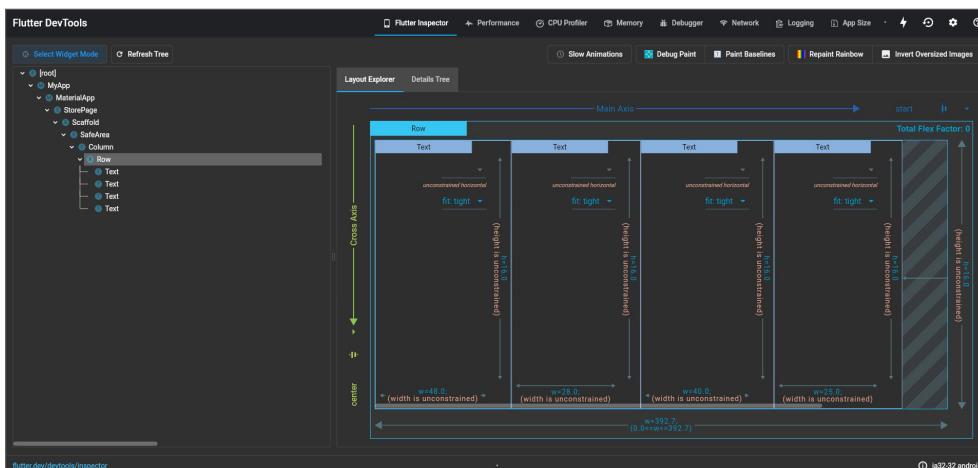
Open Flutter Devtools

Text를 적절한 위치로 정렬해야 합니다. 이때는 Open Flutter Devtools라는 도구를 사용하여 위젯이 어느 정도의 공간을 차지하고 있는지 확인하는 것이 좋습니다. 안드로이드 스튜디오 하단에 Console 탭에 번개 버튼(⚡) 2번째 옆에 있는 파랑 버튼(reen)을 클릭합니다.



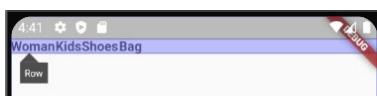
◆ Flutter Devtools 실행하기

실행을 하면 아래와 같은 화면이 웹브라우저에 열리게 됩니다.



◆ Flutter Devtools 실행 화면

Row를 선택하고 Select Widget Mode를 선택합니다.



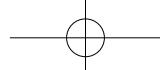
◆ Row 위젯 선택

Text를 선택해봅니다.



◆ Text 위젯 선택

Row 위젯은 넓이를 화면 끝까지 차지하고 있고, Text 위젯은 글자 크기만큼 넓이를 차지하고 있습니다. Text 위젯이 너무 붙어 있기 때문에 Row 위젯의 남은 공간을 활용해보도록 하겠습니다.

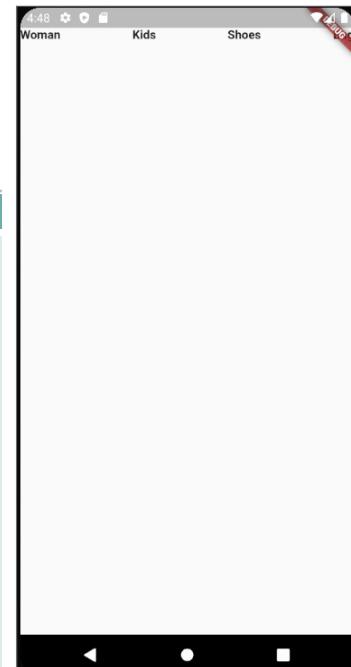


Spacer() 위젯

Spacer 위젯은 위젯 사이의 간격을 조정하는 데 사용합니다.

Spacer()라는 위젯을 이용해서 공간을 만들어보겠습니다.

```
lib/main.dart
...
...생략
Row(
  children: [
    Text("Woman", style: TextStyle(fontWeight: FontWeight.bold)),
    Spacer(),
    Text("Kids", style: TextStyle(fontWeight: FontWeight.bold)),
    Spacer(),
    Text("Shoes", style: TextStyle(fontWeight: FontWeight.bold)),
    Spacer(),
    Text("Bag", style: TextStyle(fontWeight: FontWeight.bold)),
  ],
),
...
...생략
```



◆ 4.2.7 완성 화면

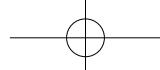
Debug 배너 해제

Debug 배너를 해제해보겠습니다. MaterialApp 속성 중에 debugShowCheckedModeBanner를 이용하면 됩니다.

```
lib/main.dart
...
...생략
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: StorePage(),
    );
  }
}
...
...생략
```

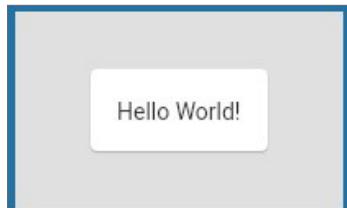


◆ 4.2.8 완성 화면



Padding 위젯

패딩(여백)은 자식 위젯 주위에 빈 공간을 만들어 줍니다. 아래 그림의 회색 공간을 여백이라고 합니다.



◆ Padding 위젯

Row 위젯을 Padding 위젯의 자식으로 감싸서 다음과 같이 여백을 주겠습니다. 이때도 Alt+Enter를 활용하여 Padding 위젯으로 감싸면 편합니다.

```
lib/main.dart
...
Padding(
  padding: const EdgeInsets.all(25.0),
  child: Row(
    children: [
      Text("Woman", style: TextStyle(fontWeight: FontWeight.bold)),
      Spacer(),
      Text("Kids", style: TextStyle(fontWeight: FontWeight.bold)),
      Spacer(),
      Text("Shoes", style: TextStyle(fontWeight: FontWeight.bold)),
      Spacer(),
      Text("Bag", style: TextStyle(fontWeight: FontWeight.bold)),
    ],
  ),
), // end of Padding
...
```

◆ 4.2.9 완성화면

“ EdgesInsets.all (왼쪽, 오른쪽, 위, 아래 즉 전체 방향에 여백을 줄 때 사용)

EdgesInsets.only (4 방향 중 내가 원하는 곳만 여백을 줄 때 사용)

EdgesInsets.symmetric (수직이나 수평 중 선택하여 여백을 줄 때 사용)

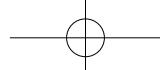
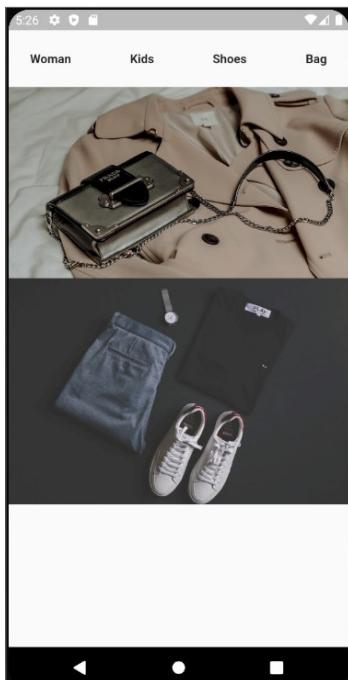


Image 위젯

Image 위젯을 이용하면 사진을 배치할 수 있습니다.



◆ 4.2.10 완성 화면

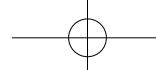
Column 위젯 내부에 Padding이 끝나는 영역 뒤에 Image 위젯을 추가하면 됩니다. 초보자분들을 위해 전체 코드를 추가하였습니다.

lib/main.dart

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: StorePage(),
    );
  }
}
```



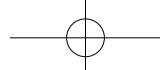
```
// stl 이라고 적으면 자동완성 기능이 활성화 된다.
class StorePage extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: SafeArea(
                child: Column(
                    children: [
                        Padding(
                            padding: const EdgeInsets.all(25.0),
                            child: Row(
                                children: [
                                    Text("Woman", style: TextStyle(fontWeight: FontWeight.bold)),
                                    Spacer(),
                                    Text("Kids", style: TextStyle(fontWeight: FontWeight.bold)),
                                    Spacer(),
                                    Text("Shoes", style: TextStyle(fontWeight: FontWeight.bold)),
                                    Spacer(),
                                    Text("Bag", style: TextStyle(fontWeight: FontWeight.bold)),
                                ],
                            ),
                        ),
                        Image.asset("assets/bag.jpeg", fit: BoxFit.cover),
                        Image.asset("assets/cloth.jpeg", fit: BoxFit.cover),
                    ],
                ),
            );
    }
}
```

“ Image 위젯을 사용할 때는 fit 속성을 이용해야 합니다.

BoxFit.contain 원본사진의 가로 세로 비율 변화 없음.

BoxFit.fill 원본사진의 비율을 무시하고 지정한 영역에 사진을 맞춤.

BoxFit.cover 원본사진의 가로 세로 비율을 유지한 채로 지정한 영역에 사진을 맞춤. 장점은 사진의 비율을 유지할 수 있다는 점이고 단점은 사진이 지정한 크기를 벗어나면 잘릴 수 있음.

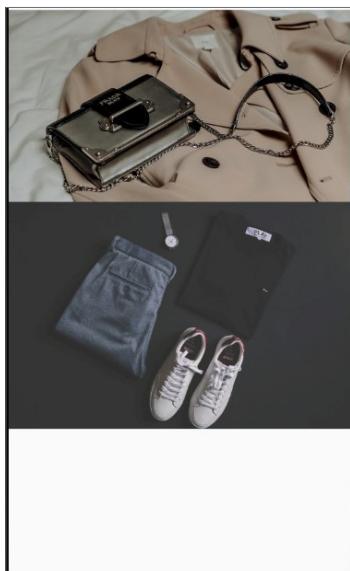


Expanded 위젯 – Column 방향

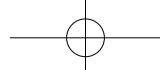


◆ 4.2.11 완성 화면

Expanded 위젯은 남은 위젯을 공간을 확장하여 공간을 채울 수 있도록 하는 위젯입니다. 4.2.10에서 완성된 화면을 보면 Column 위젯은 수직으로 배치가 되기 때문에 남은 공간은 높이입니다.



◆ Expanded 위젯이 적용되지 않은 화면



사용하지 않는 하얀 부분을 이미지 위젯으로 반반씩 채우고 싶습니다.

첫 번째 이미지를 Expanded 위젯으로 감싸줍니다.



◆ 첫 번째 이미지에 Expanded 위젯 적용

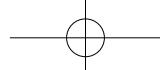
lib/main.dart

```
...생략  
Expanded(child: Image.asset("assets/bag.jpeg", fit: BoxFit.cover)),  
...생략
```

두 번째 이미지를 Expanded 위젯으로 감싸줍니다.



◆ 두 번째 이미지에 Expanded 위젯 적용



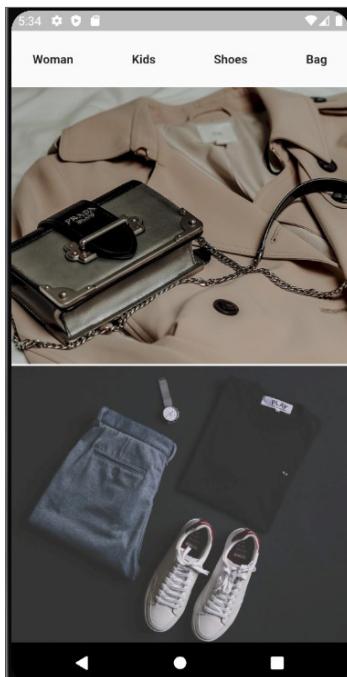
lib/main.dart

```
...생략  
Expanded(child: Image.asset("assets/cloth.jpeg", fit: BoxFit.cover)),  
...생략
```

“ Expanded 위젯은 flex라는 속성을 가지고 있습니다. flex를 지정하면 탄력성 있는, 신축성 있는 이라는 뜻이 있습니다. 첫 번째 이미지에 flex:1, 두 번째 이미지에 flex:3 이라는 값을 주게 되면 첫 번째 이미지는 1/4의 높이를 가지게 되고 두 번째 이미지는 3/4의 높이를 가지게 됩니다. flex 속성을 주지 않으면 기본 값은 flex:1입니다.

SizedBox 위젯

플로터에서 width 혹은 height 크기를 가지는 빈 상자입니다.

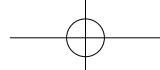


◆ 4.2.12 완성 화면

이미지 사이에 빈 공간을 주기 위해 SizedBox를 사용하였습니다.

lib/main.dart

```
...생략  
Expanded(child: Image.asset("assets/bag.jpeg", fit: BoxFit.cover)),  
SizedBox(height: 2),  
Expanded(child: Image.asset("assets/cloth.jpeg", fit: BoxFit.cover)),  
...생략
```



전체코드

```
lib/main.dart

import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            debugShowCheckedModeBanner: false,
            home: StorePage(),
        );
    }
}

// stl 이라고 적으면 자동완성 기능이 활성화 된다.
class StorePage extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: SafeArea(
                child: Column(
                    children: [
                        Padding(
                            padding: const EdgeInsets.all(25.0),
                            child: Row(
                                children: [
                                    Text("Woman", style: TextStyle(fontWeight: FontWeight.bold)),
                                    Spacer(),
                                    Text("Kids", style: TextStyle(fontWeight: FontWeight.bold)),
                                    Spacer(),
                                    Text("Shoes", style: TextStyle(fontWeight: FontWeight.bold)),
                                    Spacer(),
                                    Text("Bag", style: TextStyle(fontWeight: FontWeight.bold)),
                                ],
                            ),
                        ),
                        Expanded(child: Image.asset("assets/bag.jpeg", fit: BoxFit.cover)),
                        SizedBox(height: 2),
                        Expanded(child: Image.asset("assets/cloth.jpeg", fit: BoxFit.cover)),
                    ],
                ),
            );
    }
}
```



flutter project

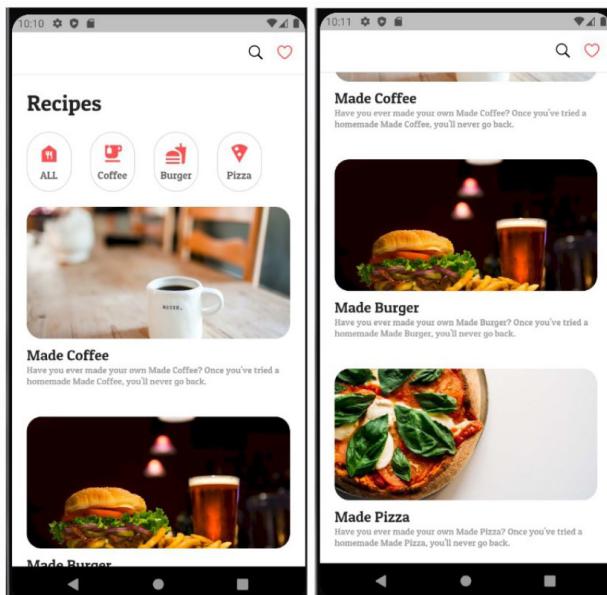
CHAPTER 05

플러터 위젯 - 레시피 앱 만들기

이번 장에서는 AppBar, Container, Icon, ClipRRect, Container, AspectRatio, ListView 위젯과 Font 변경 방법에 대해서 배워보도록 하겠습니다.

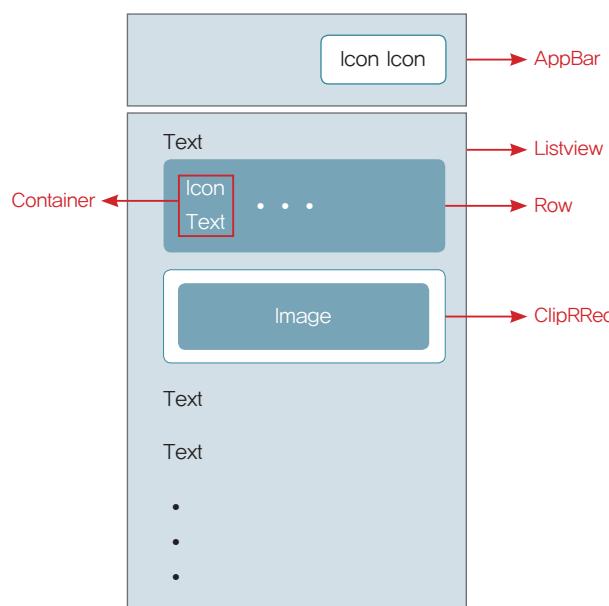
05_1 레시피 앱 구조보기

모든 소스 코드는 <https://github.com/flutter-coder/flutter-ui-book1>에 공개되어 있습니다.



◆ 레시피 앱 완성 화면

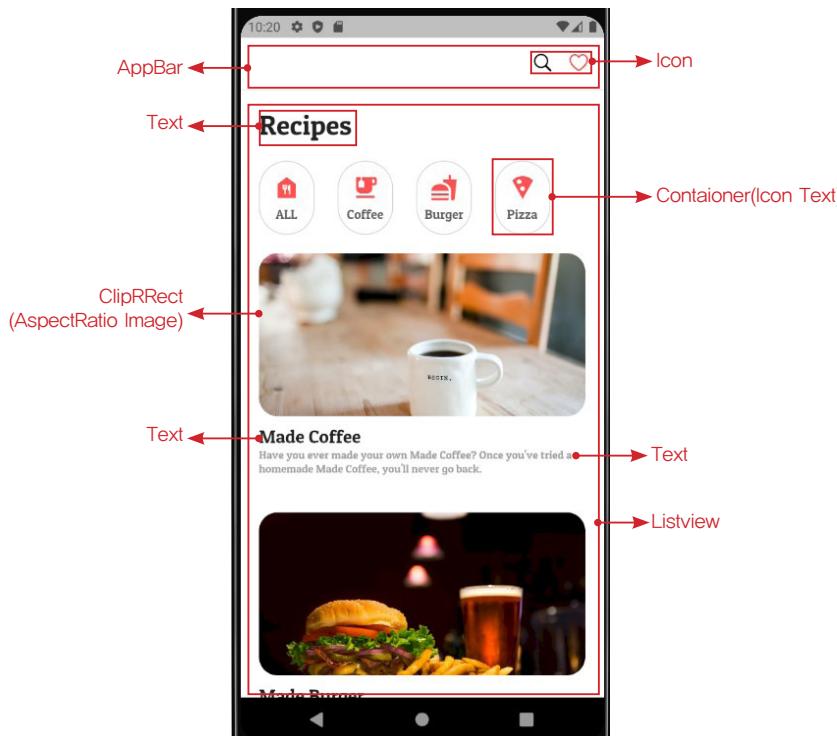
화면 구조보기



◆ 레시피 앱 화면 구조

필요한 위젯 살펴보기

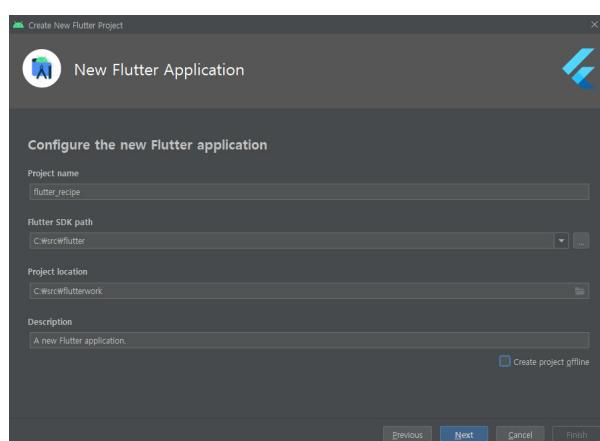
레시피 앱에 필요한 위젯을 자세히 살펴보겠습니다. 해당 그림에는 레이아웃에 관련된 Column과 Row 위젯은 가시성을 위해 제외하였습니다.



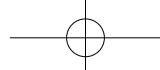
◆ 필요한 위젯 살펴보기

플러터 프로젝트 생성하기

프로젝트 이름을 flutter_recipe 으로 설정합니다.



◆ Flutter 프로젝트 설정 화면

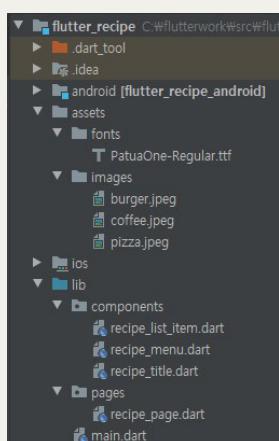


05 _ 2 레시피 앱 뼈대 구성하기

앱 뼈대 구성하기에서는 아래의 작업 순서에 따라 프로젝트에 필요한 폴더와 파일을 생성하고 기본 프로젝트 설정을 해보겠습니다.

작업 순서

- ❶ flutter_recipe/assets 폴더 생성
- ❷ flutter_recipe/assets/fonts 폴더 생성
- ❸ flutter_recipe/assets/images 폴더 생성
- ❹ flutter_recipe/assets/fonts 폴더에 폰트 추가
- ❺ flutter_recipe/assets/images 폴더에 이미지 추가
- ❻ lib/components 폴더 생성
- ❼ lib/components/recipe_title.dart 파일 추가
- ❽ lib/components/recipe_menu.dart 파일 추가
- ❾ lib/components/recipe_list_item.dart 파일 추가
- ❿ lib/pages 폴더 생성
- ⓫ lib/pages/recipe_page.dart 파일 추가

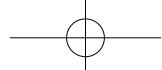


◆ 프로젝트 구조

- ❶ pubspec.yaml에서 이미지 파일과 폰트 파일 인식을 위한 자원 폴더 위치 설정
- ❷ Pub get 버튼을 클릭하여 적용

```
flutter:  
  
  uses-material-design: true  
  
  assets:  
    - assets/images/  
  
  fonts:  
    - family: PatuaOne  
      fonts:  
        - asset: assets/fonts/PatuaOne-Regular.ttf
```

◆ 폰트 설정



TIP

yaml 언어에서 #은 주석입니다. yaml 파일 이미지 캡처를 위해 모든 주석을 제거하였습니다.

TIP

pubspec.yaml 파일은 프로젝트 설정 파일입니다. 해당 파일에는 들여쓰기 규칙이 있습니다. 규칙을 어기면 [Pub get] 버튼을 클릭할 때 오류가 나서 적용이 되지 않습니다. 들여쓰기를 한 칸만 잘못해도 아래와 같은 오류가 납니다. assets: 와 첫 번째 fonts: 의 라인을 맞춰야 합니다.

yaml 파일에 대해서 궁금한 사항은 <https://ko.wikipedia.org/wiki/YAML> 위키 백과를 참고하세요.

```
18 flutter:
19   uses-material-design: true
20   assets:
21     - assets/images/
22   fonts:
23     - family: PatuaOne
24       fonts:
25         - asset: assets/fonts/PatuaOne-Regular.ttf
26
27 Document 1/1 > dev_dependencies:
```

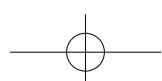
Messages: [flutter_recipe] Flutter <C:\flutter\work\flutter-windows-1-22-0\Studio\Flutter\Windows\Flutter\bin> no color pub get
Error detected in pubspec.yaml:
Error on line 23, column 5: Expected a key while parsing a block mapping.
|
23 | fonts:
| ^

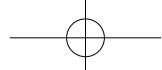
Please correct the pubspec.yaml file at C:\flutter\work\src\flutter_recipe\flutter_recipe\pubspec.yaml
Process finished with exit code 1

◆ yaml 파일의 엄격한 규칙

TIP

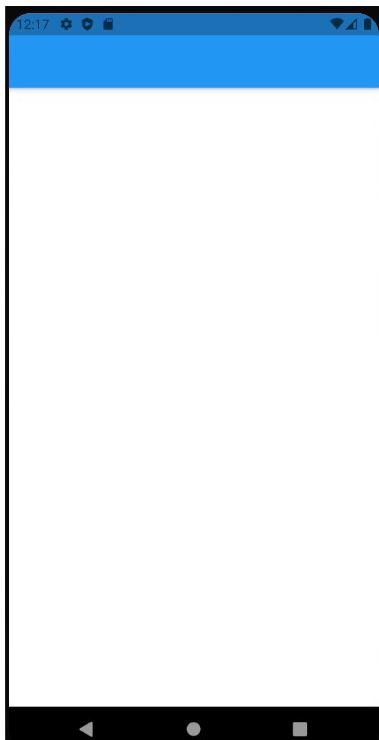
yaml 파일 설정이 어렵다면 <https://github.com/flutter-coder/flutter-ui-book1> 에서 05장 프로젝트 flutter_recipe 소스 코드를 참고하세요.





05 _ 3 레시피 앱 만들어보기

기본 코드 작성하기



◆ 5.3.1 완성 화면

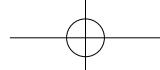
미리 만들어 둔 뼈대 파일을 서로 오류 없이 연결하기 위해 기본 코드를 작성하여 앱을 실행시켜보겠습니다.

작업 순서

- ① lib/components/recipe_title.dart 기본 코딩하기
- ② lib/components/recipe_menu.dart 기본 코딩하기
- ③ lib/components/recipe_list_item.dart 기본 코딩하기
- ④ lib/pages/recipe_page.dart 기본 코딩하기
- ⑤ lib/main.dart 기본 코딩하기

“ pages폴더를 생성한 이유는 앱에는 여러 개의 page(화면)가 있을 수 있고 그 page를 모아두는 폴더가 있으면 앱을 구조화하기 좋습니다.

폴더 이름을 pages로 하기도 하고, screens로 하기도 합니다. 개발자의 성향이나 회사마다 폴더의 이름은 다를 수 있습니다.



(1) 레시피 앱 타이틀 기본 코딩하기

```
lib/components/recipe_title.dart

import 'package:flutter/material.dart';

// stl 이라고 적으면 자동완성 됨
class RecipeTitle extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Container();
    }
}
```

“ Container 위젯은 HTML의 DIV 태그와 유사합니다. 빈 박스를 만들고 그 내부를 디자인하거나 다른 위젯을 담을 때 사용합니다. Container 위젯은 이 장의 recipe_menu.dart 파일을 만들 때 배우게 됩니다.

(2) 레시피 앱 메뉴 모음 기본 코딩하기

```
lib/components/recipe_menu.dart

import 'package:flutter/material.dart';

class RecipeMenu extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Container();
    }
}
```

(3) 레시피 앱 리스트 아이템 기본 코딩하기

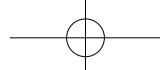
```
lib/components/recipe_list_item.dart

import 'package:flutter/material.dart';

class RecipeListItem extends StatelessWidget {
    final String imageName;
    final String title;

    const RecipeListItem(this.imageName, this.title);

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



(4) 레시피 앱 페이지 기본 코딩하기

lib/pages/recipe_page.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_recipe/components/recipe_list_item.dart';
import 'package:flutter_recipe/components/recipe_menu.dart';
import 'package:flutter_recipe/components/recipe_title.dart';

class RecipePage extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            backgroundColor: Colors.white, // 1. 배경색 white로 설정
            appBar: _buildRecipeAppBar(), // 2. 비어 있는 AppBar 연결해두기
            body: Padding(
                padding: const EdgeInsets.symmetric(horizontal: 20), // 3. 수직으로 여백 주기
                child: Column( // 4. 위에서 아래로 내려가는 구조이기 때문에 Column 위젯 사용
                    crossAxisAlignment: CrossAxisAlignment.start, // 5. 왼쪽 정렬
                    children: [
                        RecipeTitle(),
                        RecipeMenu(),
                        RecipeListItem("coffee", "Made Coffee"),
                        RecipeListItem("burger", "Made Burger"),
                        RecipeListItem("pizza", "Made Pizza"),
                    ],
                ),
            ),
        );
    }

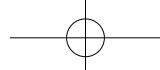
    AppBar _buildRecipeAppBar() {
        return AppBar();
    }
}
```

(5) 레피시 앱 main.dart 파일 기본 코딩하기

lib/main.dart

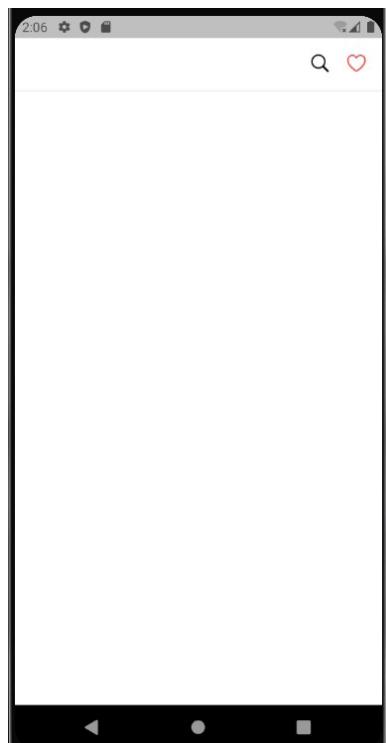
```
import 'package:flutter/material.dart';
import 'package:flutter_recipe/pages/recipe_page.dart';

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



```
class MyApp extends StatelessWidget {  
    @override  
    Widget build(BuildContext context) {  
        return MaterialApp(  
            debugShowCheckedModeBanner: false,  
            home: RecipePage(),  
        );  
    }  
}
```

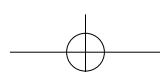
AppBar 위젯의 action 속성에 Icon 위젯 추가하기

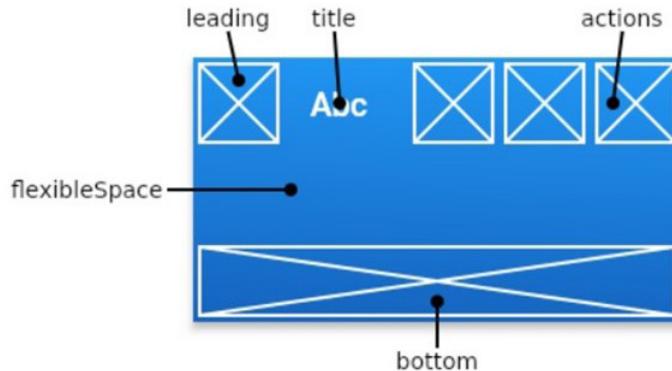
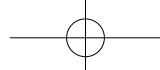


◆ 5.3.2 완성 화면

(1) AppBar

AppBar는 현재 화면의 title, leading, action 영역을 포함하고 있는 막대 모양의 위젯입니다. action 속성을 이용하면 AppBar 오른쪽 상당 부분에 위젯을 추가할 수 있습니다.





◆ AppBar 위젯 구조

(2) Icon 위젯

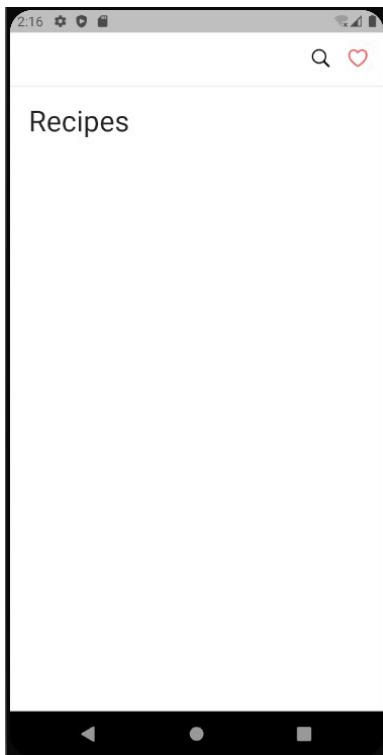
Icon 위젯은 Icon을 표시해주는 위젯입니다. Material Icon 혹은 CupertinoIcon 중 원하는 Icon을 사용할 수 있습니다. 플러터가 제공하는 기본 Icon이 아닌 다른 Icon을 사용하길 원한다면 <https://pub.dev>에서 원하는 Icon 라이브러리를 활용할 수 있습니다.

```
lib/pages/recipe_page.dart

import 'package:flutter/cupertino.dart';
...생략

AppBar _buildRecipeAppBar() {
    return AppBar(
        backgroundColor: Colors.white, // AppBar 배경색
        elevation: 1.0, // AppBar의 그림자 효과 조정
        actions: [
            Icon(
                CupertinoIcons.search, // 쿠퍼티노 아이콘 사용
                color: Colors.black,
            ),
            SizedBox(width: 15),
            Icon(
                CupertinoIcons.heart,
                color: Colors.redAccent,
            ),
            SizedBox(width: 15),
        ],
    ); // end of AppBar
}
```

RecipeTitle 커스텀 위젯 만들기

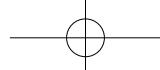


◆ 5.3.3 완성 화면

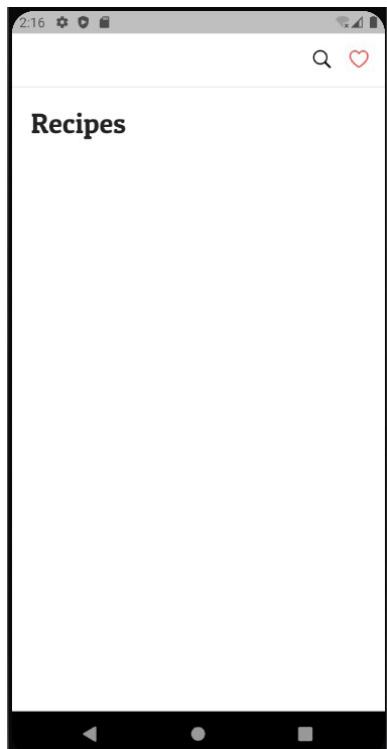
lib/components/recipe_title.dart

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

class RecipeTitle extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Padding(
            padding: const EdgeInsets.only(top: 20), // 1. top 부분만 여백을 줄 수 있다.
            child: Text(
                "Recipes",
                style: TextStyle(fontSize: 30),
            ),
        ); // end of Padding
    }
}
```



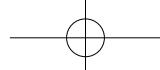
Theme에 Font 적용하기



◆ 5.3.4 완성 화면

테마는 전체적으로 앱의 모양과 느낌을 가지고 있습니다. 테마에 Font를 적용하여 앱의 전반적인 글자체를 변경해보겠습니다.

```
lib/main.dart
...
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      theme: ThemeData(fontFamily: "PatuaOne"),
      home: RecipePage(),
    );
  }
}
```



TIP

fontFamily 속성에 들어가는 값은 pubspec.yaml

fonts:

 – family : "PatuaOne"

부분에 작성된 값 "PatuaOne"을 사용하면 됩니다. 이 값은 고정된 값이 아닌 사용자가 정의할 수 있는 임의의 문자열입니다.

“ 폰트가 적용되지 않는다면 앱을 종료시키고 다시 실행시켜주세요.

Container 위젯을 활용한 RecipeMenu 커스텀 위젯 만들기

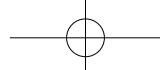
(1) Container 위젯 정의

Container 위젯은 빈 박스 위젯입니다. SizedBox 위젯과 차이점이 있다면 Container는 내부에 decoration 속성이 있어서 박스에 색상을 입히거나 박스의 모양을 바꾼다거나 테두리 선을 줄 수 있습니다. SizedBox 위젯은 보통 마진(Margin)을 줘야 할 때 사용합니다.



◆ 5.3.5 완성 화면

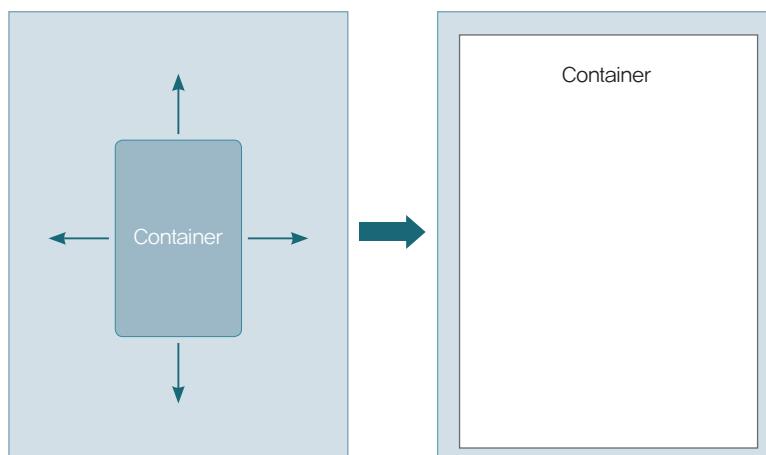
Container 위젯 내부에 Icon 위젯과 Text 위젯이 있습니다. 구조는 다음과 같이 됩니다.



Container
- Column
- Icon
- Text

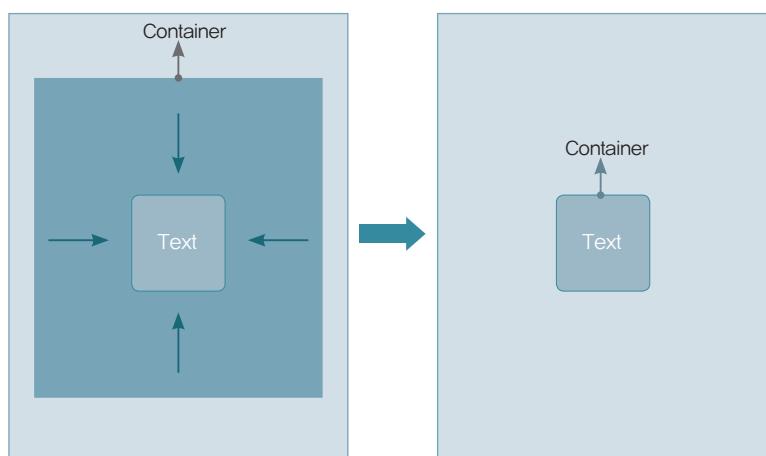
(2) Container 위젯 특징 (중요)

첫째, 자식이 없는 Container는 가능한 한 박스를 크게 만들려고 합니다.

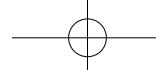


◆ Container 위젯의 독특한 성질 – 늘어난다

둘째, 자식이 있는 Container는 자식의 크기에 맞게 조정 됩니다.



◆ Container 위젯의 독특한 성질 – 줄어든다

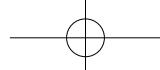


lib/components/recipe_menu.dart

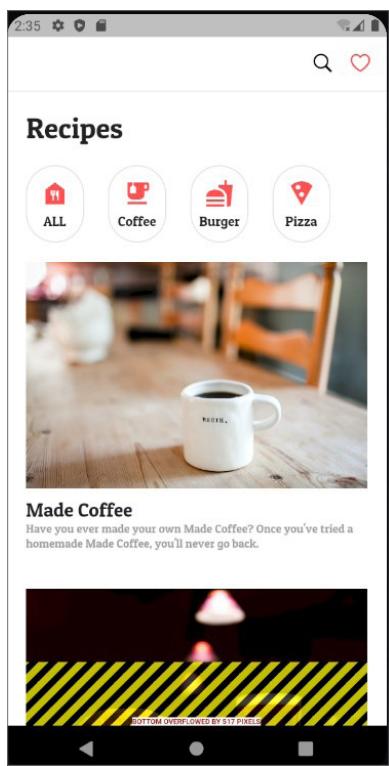
```
import 'package:flutter/material.dart';

class RecipeMenu extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.only(top: 20),
      child: Row( // ❶ 메뉴 아이템들의 방향이 수평 방향이기 때문!
        children: [
          _buildMenuItem(Icons.food_bank, "ALL"), // ❷ 재사용
          SizedBox(width: 25),
          _buildMenuItem(Icons.emoji_food_beverage, "Coffee"), // ❸ 재사용
          SizedBox(width: 25),
          _buildMenuItem(Icons.fastfood, "Burger"), // ❹ 재사용
          SizedBox(width: 25),
          _buildMenuItem(Icons.local_pizza, "Pizza"), // ❺ 재사용
        ],
      ),
    ); // end of Padding
  }

  // ❻ 재사용할 수 있는 함수로 만든다.
  // ❼ Widget은 모든 위젯의 부모이기 때문에 함수 리턴 타입은 Widget으로 하는 것이 좋다.
  Widget _buildMenuItem(IconData mIcon, String text) {
    return Container(
      width: 60,
      height: 80,
      decoration: BoxDecoration(
        borderRadius: BorderRadius.circular(30),
        border: Border.all(color: Colors.black12),
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Icon(mIcon, color: Colors.redAccent, size: 30),
          SizedBox(height: 5),
          Text(
            text,
            style: TextStyle(color: Colors.black87),
          ),
        ],
      ),
    );
  } // end of _buildMenuItem
}
```



재사용 가능한 레시피 리스트 아이템 만들기 – 클래스 생성자 활용



◆ 5.3.6 완성 화면

RecipeListItem 객체를 만들 때 생성자에서 imageName, title 값을 초기화할 수 있습니다. 이를 통해 같은 디자인이나 데이터는 다르게 화면을 구현할 수 있습니다. 위 화면에는 하나의 문제가 있습니다. RecipeListItem 3개를 화면에 출력하려고 하니 핸드폰 화면을 넘어가는 overflow 문제 가 발생하게 됩니다. 5.3.7에서 해결하도록 하겠습니다.

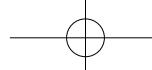
```
lib/components/recipe_list_item.dart

import 'package:flutter/material.dart';

class RecipeListItem extends StatelessWidget {
    final String imageName;
    final String title;

    const RecipeListItem(this.imageName, this.title);

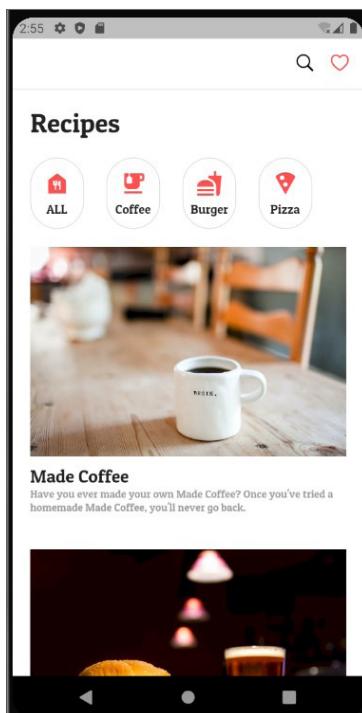
    @override
    Widget build(BuildContext context) {
        return Padding(
            padding: const EdgeInsets.symmetric(vertical: 20),
            child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
```



```
children: [
    Image.asset(
        "assets/images/$imageName.jpeg",
        fit: BoxFit.cover,
    ),
    SizedBox(height: 10),
    Text(
        title,
        style: TextStyle(fontSize: 20),
    ),
    Text(
        "Have you ever made your own $title? Once you've tried a homemade $title, you'll never
go back.",
        style: TextStyle(color: Colors.grey, fontSize: 12),
    ),
],
),
),
),
);
// end of Padding
}
}
```

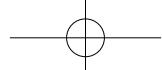
“ 문자열 안에 \$를 사용하면 변수를 사용할 수 있습니다.

ListView 위젯을 활용하여 세로 스크롤 달기



◆ 완성 화면

105 플로터로 앱 만들기

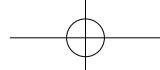


ListView는 가장 일반적으로 사용되는 스크롤 위젯입니다. 스크롤 방향으로 자식을 차례로 표시합니다. ListView를 사용하여 가로축으로 스크롤을 할 수 있고, 세로축으로 스크롤할 수 있습니다. Column을 ListView로 변경하고 crossAxisAlignment 속성을 제거합니다. ListView의 자식들의 기본 정렬은 왼쪽 정렬입니다.

```
class RecipePage extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      backgroundColor: Colors.white,  
      appBar: _buildRecipeAppBar(),  
      body: Padding(  
        padding: const EdgeInsets.symmetric(horizontal: 20),  
        child: Column(  
          crossAxisAlignment: CrossAxisAlignment.start,  
          children: [  
            RecipeTitle(),  
            RecipeMenu(),  
            RecipeListItem("coffee", "Made Coffee"),  
            RecipeListItem("burger", "Made Burger"),  
            RecipeListItem("pizza", "Made Pizza"),  
          ],  
        ),  
      ),  
    );  
  }  
}  
  
class RecipePage extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      backgroundColor: Colors.white,  
      appBar: _buildRecipeAppBar(),  
      body: Padding(  
        padding: const EdgeInsets.symmetric(horizontal: 20),  
        child: ListView(  
          children: [  
            RecipeTitle(),  
            RecipeMenu(),  
            RecipeListItem("coffee", "Made Coffee"),  
            RecipeListItem("burger", "Made Burger"),  
            RecipeListItem("pizza", "Made Pizza"),  
          ],  
        ),  
      ),  
    );  
  }  
}
```

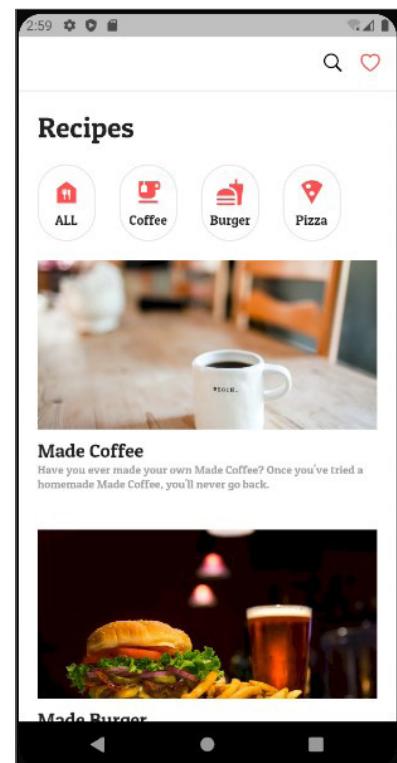
◆ Column 위젯을 ListView로 변경하는 법

```
lib/pages/recipe_page.dart  
  
...생략  
class RecipePage extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      backgroundColor: Colors.white,  
      appBar: _buildRecipeAppBar(),  
      body: Padding(  
        padding: const EdgeInsets.symmetric(horizontal: 20),  
        child: ListView(  
          children: [  
            RecipeTitle(),  
            RecipeMenu(),  
            RecipeListItem("coffee", "Made Coffee"),  
            RecipeListItem("burger", "Made Burger"),  
            RecipeListItem("pizza", "Made Pizza"),  
          ],  
        ),  
      ),  
    );  
  }  
}  
...생략
```



AspectRatio로 이미지 비율 정하기

특정 종횡비로 자식 크기를 조정하는 위젯입니다. 이미지를 화면에 표시할 때는 비율로 표시하는 것이 좋습니다.



◆ 5.3.8 완성 화면

AspectRatio 위젯은 먼저 레이아웃 제약에서 허용하는 가장 큰 너비를 시도합니다. 위젯의 높이는 지정된 가로 세로 비율을 너비에 적용하여 결정되며 너비와 높이의 비율로 표현됩니다.

예를 들어 넓이가 300이고 높이가 600인 화면이 있을 때 이미지에 AspectRatio 위젯을 적용하여 비율을 2/1로 주게 되면 넓이 300의 비율이 2가 되기 때문에 높이는 300의 절반인 150이 됩니다.



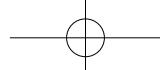
Quiz

Q 화면 넓이 450, 높이 1000에서 비율을 3/2를 주게 되면 이미지의 크기는 몇일까요?

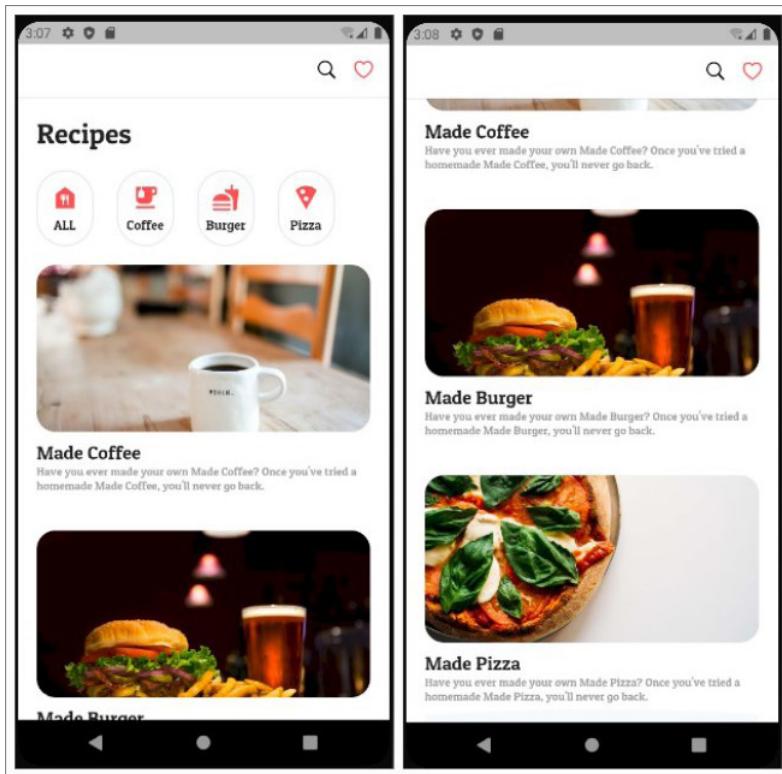
A 이미지 넓이 450, 이미지 높이 300

lib/components/recipe_list_item.dart

```
...생략
AspectRatio(
  aspectRatio: 2 / 1,
  child: Image.asset(
    "assets/images/$imageName.jpeg",
    fit: BoxFit.cover,
  ),
), // end of AspectRatio
...생략
```



ClipRRect 위젯으로 이미지 모서리에 곡선 주기



◆ 5.3.9 완성 화면

등근 사각형을 사용하여 자식을 자르는 위젯입니다. Clip은 자르다는 의미이고, R은 Round의 의미이고 Rect는 사각형의 의미입니다. 사각형을 둥글게 잘라주는 위젯이 ClipRRect 위젯입니다. 위젯에 shape나 decoration 속성이 없다면 ClipRRect을 이용하여 모서리에 곡선을 줄 수 있습니다. Container 위젯 같은 경우는 decoration 속성이 있기 때문에 ClipRRect 위젯을 사용할 필요가 없습니다.

```
lib/components/recipe_list_item.dart
...
...생략
AspectRatio(
  aspectRatio: 2 / 1,
  child: ClipRRect(
    borderRadius: BorderRadius.circular(20),
    child: Image.asset(
      "assets/images/$imageName.jpeg",
      fit: BoxFit.cover,
    ),
  ), // end of ClipRRect
),
...생략
```



flutter project

CHAPTER 06

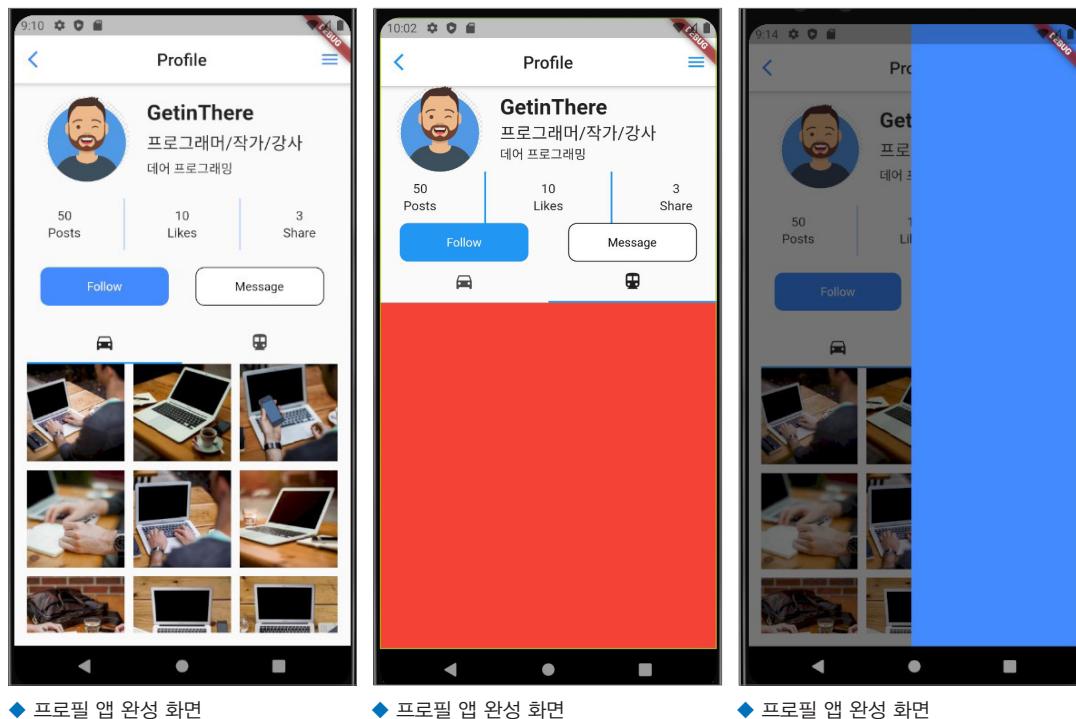
플러터 위젯 - 프로필 앱 만들기

이번 장에서는 ThemeData 클래스와 TabBar, TabBarView, AppBar, InkWell, GridView, Drawer, Align 위젯과 Image위젯으로 network 이미지를 다운 받아서 화면에 표시하는 방법에 대해서 배워보도록 하겠습니다.

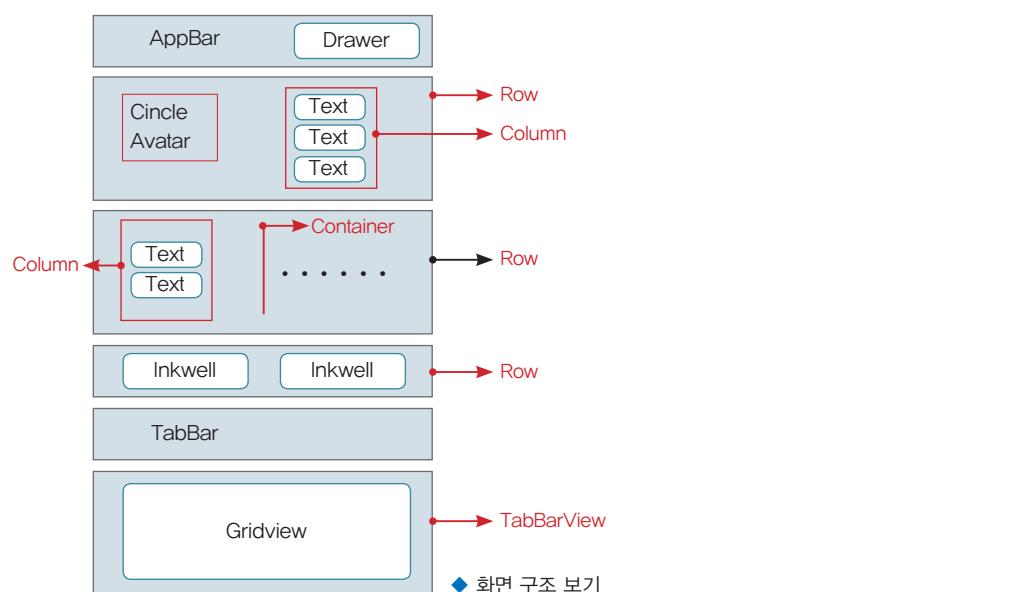
06 _ 1 프로필 앱 구조보기

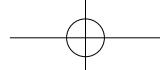
모든 소스 코드는 다음 깃허브의 링크 주소에 공개되어 있습니다.

- <https://github.com/flutter-coder/flutter-ui-book1>



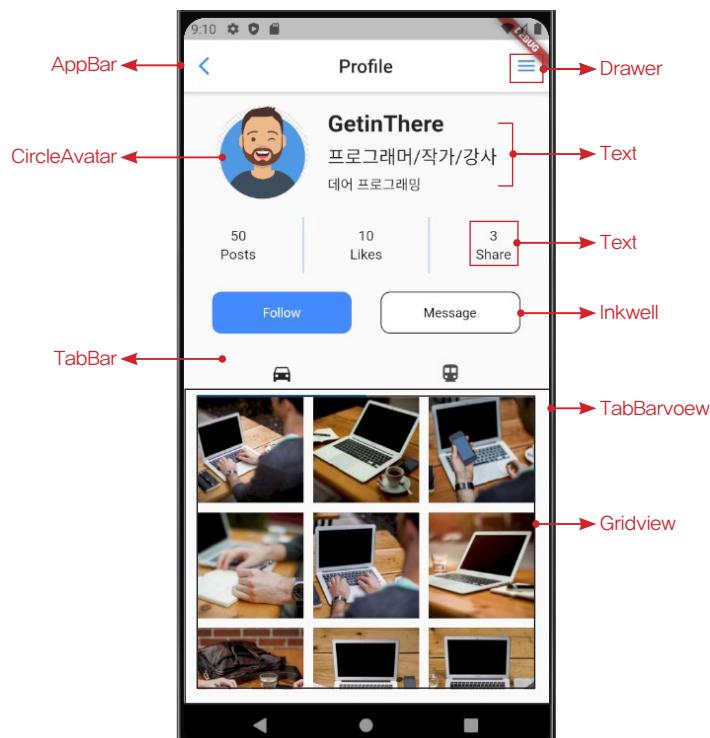
화면 구조보기





필요한 위젯 살펴보기

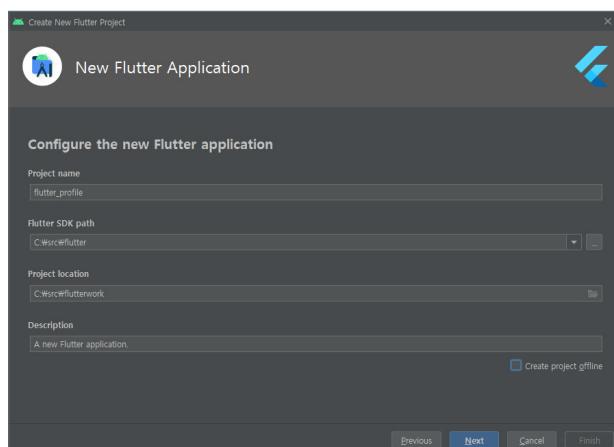
이제 처음에 봤던 프로필 앱에 필요한 위젯을 자세히 살펴보겠습니다. 해당 그림에는 레이아웃에 관련된 Column과 Row 위젯은 가시성을 위해 제외하였습니다.



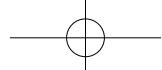
◆ 위젯 살펴보기

플러터 프로젝트 생성하기

프로젝트 이름을 flutter_profile로 설정합니다.



◆ Flutter 프로젝트 설정 화면



플러터 2.0 업그레이드

터미널을 열고 flutter 버전을 확인합니다.

```
flutter --version
```

Flutter 버전이 2.0.0 이상이고, Dart 버전이 2.12.0 이상이면 업그레이드를 진행할 필요가 없습니다.

플러터를 2.0으로 업그레이드 하는 이유는 9장에서 Flutter web을 체험해볼 예정이며, Flutter 2.0으로 업그레이드 하게 되면 Dart 2.12 버전을 사용할 수 있습니다. Dart 2.12 버전에서는 Null Safety가 적용되어 Null에 대한 안정성이 제공됩니다. Null이 허용되지 않기 때문에 Runtime(앱 실행시)에 발생할 수 있는 오류를 컴파일시에 미리 대처할 수 있습니다.

안드로이드 스튜디오에서 Terminal(터미널)창을 열고 다음과 같이 입력합니다.

```
Terminal: Local +  
Microsoft Windows [Version 10.0.18363.1440]  
(c) 2019 Microsoft Corporation. All rights reserved.  
  
C:\w\flutter\work\ws\src\w\flutter\_profile\w\flutter\_profile>flutter upgrade --force  
  
Logcat Database Inspector Profiler TODO Terminal Dart Analysis  
daemon started successfully (7 minutes ago)
```

◆ 터미널 화면에서 Flutter 강제로 업그레이드 하는 법

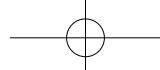
```
flutter upgrade --force
```

Dart 2.12 버전 적용하기

pubspec.yaml 파일을 열고 dart 버전을 2.12 이상으로 잡고 [Pub get] 버튼을 클릭하여 적용합니다. flutter 버전이 2.0.0 이상이어도 이 부분은 꼭 설정해주셔야 합니다. 그리고 7장 이상부터는 dart 2.12.0 버전에 맞춰서 책을 집필하였기 때문에 버전을 꼭 확인해주세요.

```
environment:  
  sdk: '>=2.12.0 <3.0.0'  
  
dependencies:  
  flutter:  
    sdk: flutter
```

◆ Dart 버전 2.12.0



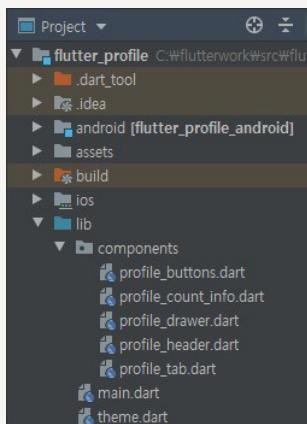
06 _ 2 프로필 앱 뼈대 작성하기

프로필 앱을 만들기 위해 기본적인 파일과 패키지를 생성해봅시다.

프로젝트 구조 세팅하기

작업 순서

- ① 프로젝트 최상단에 assets 폴더 생성
- ② lib/components 패키지 생성
- ③ lib/theme.dart 파일 생성
- ④ lib/components/profile_drawer.dart 파일 생성
- ⑤ lib/components/profile_header.dart 파일 생성
- ⑥ lib/components/profile_info.dart 파일 생성
- ⑦ lib/components/profile_buttons.dart 파일 생성
- ⑧ lib/components/profile_tab.dart 파일 생성



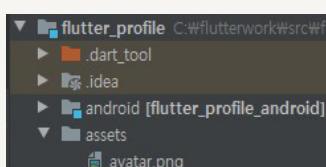
◆ 프로젝트 구조

- ⑨ pubspec.yaml에 assets 폴더 설정 후 우측 상단 [Pub get] 버튼 클릭

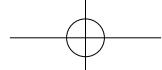
```
assets:  
- assets/  
# - images/a_dot_ham.jpeg
```

◆ assets 설정

- ⑩ assets 폴더에 이미지 추가하기



◆ 이미지 추가



기본 코드 작성하기

작업 순서

- ❶ lib/theme.dart 코드 작성
- ❷ lib/components/profile_drawer.dart 코드 작성
- ❸ lib/components/profile_header.dart 코드 작성
- ❹ lib/components/profile_count_info.dart 코드 작성
- ❺ lib/components/profile_buttons.dart 코드 작성
- ❻ lib/components/profile_tab.dart 코드 작성
- ❼ lib/main.dart 코드 작성

- ❶ lib/theme.dart 코드를 작성해 봅니다.

lib/theme.dart

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

ThemeData theme() {
    return ThemeData(
        primaryColor: Colors.white,
        appBarTheme: AppBarTheme(
            iconTheme: IconThemeData(color: Colors.blue),
        ),
    );
}
```

TIP

❶ PrimaryColor

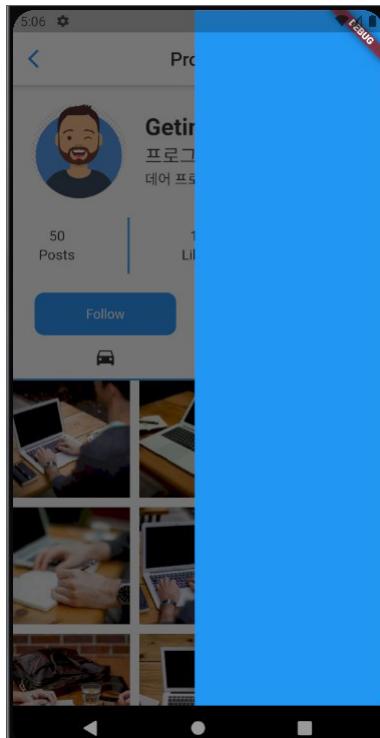
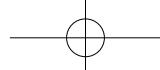
PrimaryColor는 브랜드의 아이덴티티를 나타내는 색입니다. 앱에 PrimaryColor는 blue 색상이 기본으로 잡혀 있습니다. 해당 색상을 white로 변경해주었습니다.

❷ AccentColor

AccentColor는 앱의 상호작용 요소에 사용하는 색입니다. 버튼이나 링크, 토글 그리고 스위치, 진행률 표시기 같은 것들의 색상은 AccentColor로 나타냅니다. AccentColor를 다른 말로 하면 SecondaryColor라고 부르기도 합니다.

- ❷ lib/components/profile_drawer.dart 코드를 작성해 봅니다.

ProfileDrawer 위젯에서는 넓이가 200인 파란색 Container를 간단히 만들어보겠습니다.



lib/components/profile_drawer.dart

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

class ProfileDrawer extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Container(
            width: 200,
            height: double.infinity,
            color: Colors.blue,
        );
    }
}
```

◆ endDrawer 만들기

“ double.infinity는 해당 위치가 차지할 수 있는 최대 범위로 확장할 때 사용합니다.

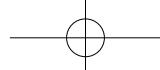
③ lib/components/profile_header.dart 코드를 작성해 봅니다.

lib/components/profile_header.dart

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

class ProfileHeader extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Row(
            children: [
                SizedBox(width: 20),
                _buildHeaderAvatar(),
                SizedBox(width: 20),
                _buildHeaderProfile(),
            ],
        );
    }

    Widget _buildHeaderAvatar() {
        return SizedBox();
```



```
}

Widget _buildHeaderProfile() {
    return SizedBox();
}
}
```

- ④ lib/components/profile_count_info.dart 코드를 작성해 봅니다.

```
lib/components/profile_count_info.dart

import 'package:flutter/material.dart';

class ProfileCountInfo extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Row(
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            children: [
                _buildInfo("50", "Posts"),
                _buildLine(),
                _buildInfo("10", "Likes"),
                _buildLine(),
                _buildInfo("3", "Share"),
            ],
        );
    }

    Widget _buildInfo(String count, String title) {
        return SizedBox();
    }

    Widget _buildLine() {
        return SizedBox();
    }
}
```

- ⑤ lib/components/profile_buttons.dart 코드를 작성해 봅니다.

```
lib/components/profile_buttons.dart

import 'package:flutter/material.dart';

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

```

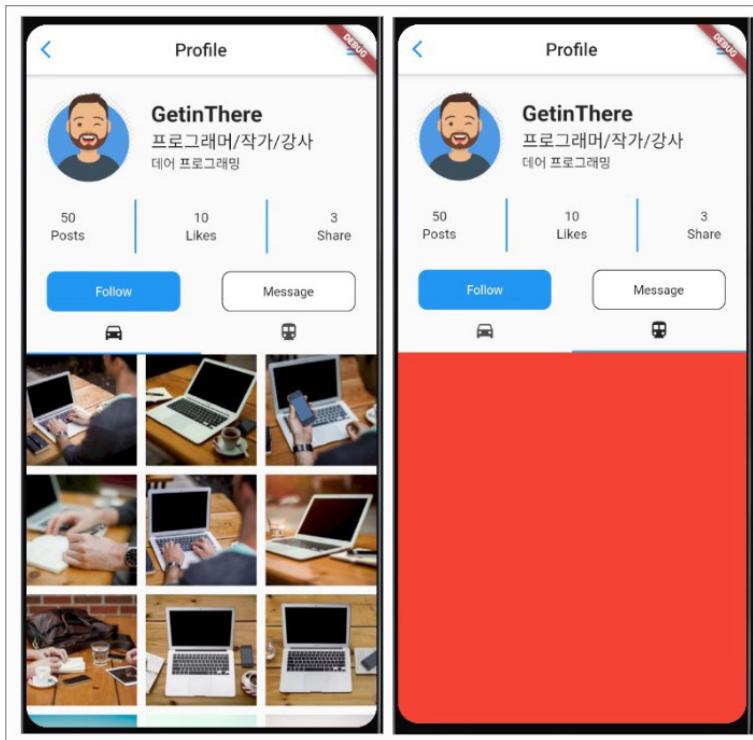
return Row(
  mainAxisSize: MainAxisSize.spaceAround,
  children: [
    _buildFollowButton(),
    _buildMessageButton(),
  ],
);
}

Widget _buildFollowButton() {
  return SizedBox();
}

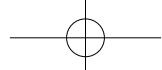
Widget _buildMessageButton() {
  return SizedBox();
}

```

⑥ lib/components/profile_tab.dart 코드를 작성해 봅니다.



◆ TabBar가 작동하면 화면이 다시 그려진다



앱이 실행되고 난 뒤 사용자의 요청에 의해 그림을 다시 그리기 위해서는 StatelessWidget을 사용해야 합니다. TabBar를 클릭하면 화면이 동적으로 변경되기 때문에 StatelessWidget을 사용해보도록 하겠습니다. StatelessWidget에 대한 자세한 설명은 8장에서 설명합니다.

lib/components/profile_tab.dart

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

// stf 라고 입력하면 자동완성 됨.
class ProfileTab extends StatelessWidget {
    @override
    _ProfileTabState createState() => _ProfileTabState();
}

class _ProfileTabState extends State<ProfileTab> {

    @override
    Widget build(BuildContext context) {
        return Column(
            children: [
                _buildTabBar(),
                _buildTabBarView(),
            ],
        );
    }

    Widget _buildTabBar() {
        return SizedBox();
    }

    Widget _buildTabBarView() {
        return SizedBox();
    }
}
```

⑦ lib/main.dart 코드를 작성해 봅니다.

실행을 하면 빈 화면이 나타납니다.



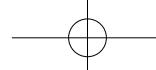
◆ 기본 코드가 완성된 화면

lib/main.dart

```
lib/main.dart
import 'package:flutter/material.dart';
import 'package:flutter_profile/components/profile_buttons.dart';
import 'package:flutter_profile/components/profile_count_info.dart';
import 'package:flutter_profile/components/profile_header.dart';
import 'package:flutter_profile/components/profile_tab.dart';
import 'package:flutter_profile/theme.dart';

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

class MyApp extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            theme: theme(),
            home: ProfilePage(),
        );
    }
}
```

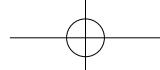


```
class ProfilePage extends StatelessWidget {  
    @override  
    Widget build(BuildContext context) {  
        return Scaffold(  
            body: Column(  
                children: [  
                    SizedBox(height: 20),  
                    ProfileHeader(),  
                    SizedBox(height: 20),  
                    ProfileCountInfo(),  
                    SizedBox(height: 20),  
                    ProfileButtons(),  
                    // 남아 있는 세로 공간을 모두 차지하기 위해 Expanded를 준다.  
                    Expanded(child: ProfileTab()),  
                ],  
            ),  
        );  
    }  
  
    AppBar _buildProfileAppBar() {  
        return AppBar();  
    }  
}
```

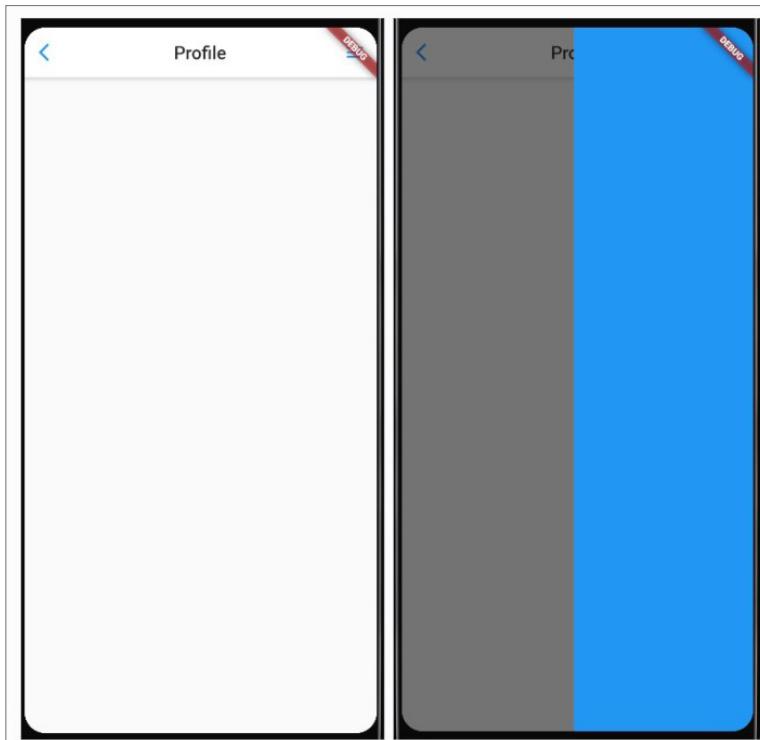
06 _ 3 프로필 앱 위젯 구성하기

작업 순서

- ① AppBar 위젯과 Scaffold의 endDrawer 속성 활용하기
- ② CircleAvatar 위젯
- ③ Column 위젯의 CrossAxisAlignmentAlignment 속성 활용하기
- ④ 재사용 가능한 함수 만들기
- ⑤ InkWell 위젯을 사용하여 ProfileButtons 클래스 만들기
- ⑥ TabBar 위젯과 TabBarView 위젯 사용하기
- ⑦ GridView 위젯과 Image.network



AppBar 위젯과 Scaffold의 endDrawer 속성 활용하기

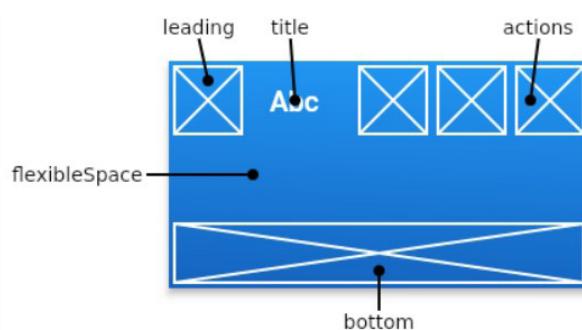


◆ 6.3.1 완성 화면

① AppBar 위젯

AppBar는 현재 화면의 title, leading, action 영역을 포함하고 있는 막대 모양의 위젯입니다.

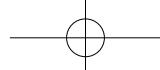
leading 속성을 활용하면 AppBar의 왼쪽 상단 부분에 위젯을 추가할 수 있습니다.



◆ AppBar 위젯 구조

② Scaffold의 endDrawer 속성

앱에서 탐색 링크(Navigation)를 표시하기 위해 Scaffold의 가장자리(왼쪽, 오른쪽)에서 수평으로 슬라이드 하는 위젯입니다



lib/main.dart

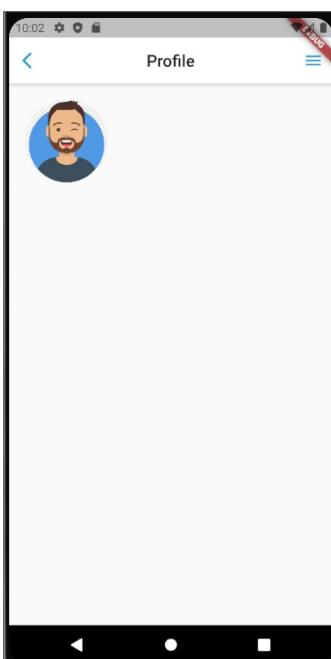
```
...생략
class ProfilePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      endDrawer: ProfileDrawer(),
      appBar: _buildProfileAppBar(),
      body: Column(
        ...생략

      AppBar _buildProfileAppBar() {
        return AppBar(
          leading: Icon(Icons.arrow_back_ios),
          title: Text("Profile"),
          centerTitle: true,
        );
      }
      ...생략
```

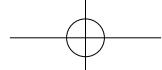
“ Scaffold 속성에서 왼쪽에서 오른쪽으로 슬라이드 하는 Drawer를 만들기 위해서는 drawer 속성을 사용하고 오른쪽에서 왼쪽으로 슬라이드 하는 Drawer를 만들기 위해서는 endDrawer 속성을 사용합니다. 우리는 endDrawer 속성을 사용합니다.

CircleAvatar 위젯

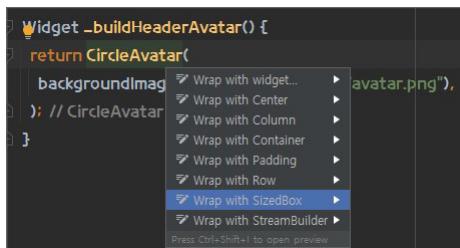
CircleAvatar 위젯은 이미지를 둥글게 만들어주는 위젯입니다.



◆ 6.3.2 완성 화면



`_buildHeaderAvatar()` 함수를 코딩할 때 `CircleAvatar`를 먼저 만든 다음 `CircleAvatar` 위젯에 커서를 두고 Alt+Enter 키를 입력하여 `SizedBox` 위젯으로 감싸면 틀에 도움을 받아서 편하게 코딩할 수 있습니다.



◆ 매직키를 이용하여 `SizedBox`로 감싸기

lib/components/profile_header.dart

```
...생략
Widget _buildHeaderAvatar() {
    return SizedBox(
        width: 100,
        height: 100,
        child: CircleAvatar(
            backgroundImage: AssetImage("assets/avatar.png"),
        ),
    ); // end of SizedBox
}
...생략
```

“`width, height` 속성이 없는 위젯의 크기를 설정하려면 `SizedBox` 위젯으로 감싸서 크기를 지정할 수 있습니다.

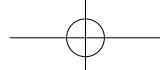
TIP 위젯을 둥글게 만드는 법

첫째, `Container` 위젯을 사용하여 `decoration` 속성을 사용해서 `Container`를 동그랗게 만들고 그 안에 `Image` 위젯을 `child`로 추가하는 방법입니다.

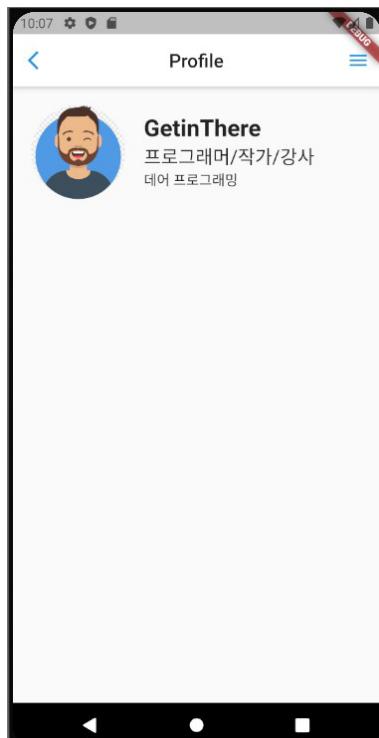
둘째, `Image` 위젯을 만들고 `ClipOver` 위젯으로 감싸서 이미지를 동그랗게 만드는 방법입니다.

셋째, `CircleAvatar` 위젯을 사용하여 이미지를 동그랗게 만드는 방법입니다.

세 가지 방법 중 무엇을 사용하든 내가 원하는 결과를 만들 수 있습니다. 하지만 가장 간단한 방법은 `CircleAvatar` 위젯을 사용하는 것입니다.



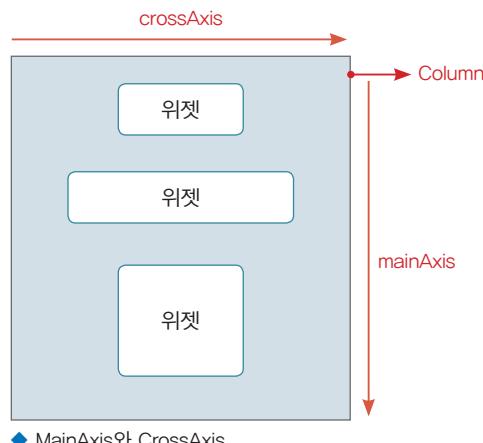
Column 위젯의 CrossAxisAlignmentAlignment 속성 활용하기

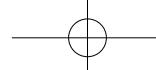


◆ 6.3.3 완성 화면

Column위젯은 여러 자식 위젯들을 가질 수 있습니다. Column위젯의 특징은 가장 width가 넓은 자식의 크기에 따라 넓이가 결정되며 기본 가로 방향 정렬은 center입니다.

꼭 알아야 할 속성이 있는데 Column의 자식들은 세로로 배치가 되기 때문에 mainAxis(주축)이 세로 방향이며, crossAxis(반대축)이 가로 방향입니다.



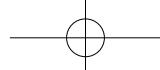


lib/components/profile_header.dart

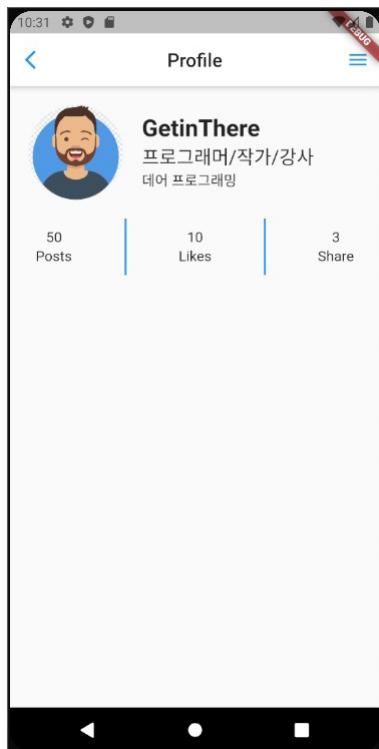
```
...생략
Widget _buildHeaderProfile() {
    return Column(
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
            Text(
                "GetinThere",
                style: TextStyle(
                    fontSize: 25,
                    fontWeight: FontWeight.w700,
                ),
            ),
            Text(
                "프로그래머/작가/강사",
                style: TextStyle(
                    fontSize: 20,
                ),
            ),
            Text(
                "데어 프로그래밍",
                style: TextStyle(
                    fontSize: 15,
                ),
            ),
        ],
    ); // end of Column
}
...생략
```

TIP

center는 가운데 정렬
start는 주로 왼쪽 정렬
end는 주로 오른쪽 정렬
spaceBetween은 2개 이상의 위젯이 있을 때 양 끝 정렬
spaceAround는 2개 이상의 위젯이 있을 때 각 위젯들이 양 옆으로 적당한 공간을 확보



재사용 가능한 함수 만들기

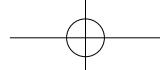


◆ 6.3.4 완성 화면

위 그림을 보면 50 Posts, 10 Likes, 3 Share 부분의 디자인이 동일한데 Text만 다른 것을 확인할 수 있습니다. 그리고 파란색 선도 두 군데서 동일하게 사용되고 있습니다. 이럴 때는 재사용 가능한 함수로 만드는 것이 좋습니다.

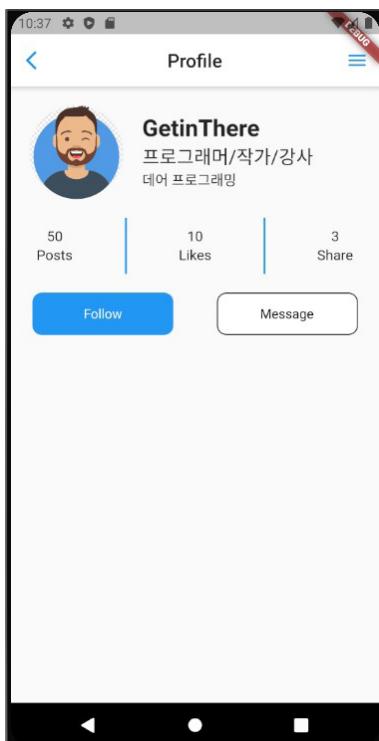
```
lib/components/profile_count_info.dart

...생략
Widget _buildInfo(String count, String title) {
  return Column(
    children: [
      Text(
        count, // 변수 바인딩
        style: TextStyle(fontSize: 15),
      ),
      SizedBox(height: 2),
      Text(
        title, // 변수 바인딩
        style: TextStyle(fontSize: 15),
      ),
    ],
); // end of Column
}
```



```
Widget _buildLine() {
    return Container(width: 2, height: 60, color: Colors.blue); // end of Container
}
...생략
```

InkWell 위젯을 사용하여 ProfileButtons 클래스 만들기



◆ 6.3.5 완성 화면

InkWell 위젯은 모든 위젯을 버튼화 시켜주는 위젯입니다. 문서에는 터치에 반응하는 Material(물질, 재료)의 직사각형 영역이라고 나와 있습니다.

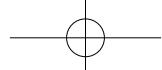
플러터에서 버튼을 만들기 위한 방법이 대표적으로 4가지가 있습니다.

- ① TextButton 위젯으로 구현하기
- ② ElevatedButton 위젯으로 구현하기
- ③ OutlinedButton 위젯으로 구현하기
- ④ InkWell 위젯으로 구현하기

TIP

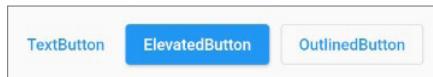
Flutter 1.X 버전 : FlatButton, RaisedButton, OutlineButton

Flutter 2.X 버전 – 현재 교재에서 사용하고 있는 버전 : TextButton, ElevatedButton, OutlinedButton



TextButton의 특징은 버튼 자체에 아무런 디자인이 없습니다. 그냥 Text를 클릭할 수 있게 만들어 주는 효과만 가지고 있습니다.

ElevatedButton의 특징은 버튼 자체에 디자인이 생깁니다. 대표적으로 elevation 효과가 적용되는데 버튼 오른쪽과 아래쪽 부분에 그림자 효과가 생기게 되어 약간 떠올라 있는 느낌을 가지게 해줍니다. OutlinedButton의 특징은 가장 자리에 테두리 선을 가지고 있습니다.



◆ Flutter 버튼 종류

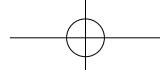
InkWell 위젯의 특징은 모든 위젯을 버튼으로 만들 수 있다는 것입니다. 보통 Container로 디자인 한 뒤 InkWell 위젯으로 감싸는데 개인적으로 이 방법을 가장 선호합니다. 왜냐하면 버튼 디자인을 가장 자유롭게 할 수 있기 때문입니다.

```
lib/components/profile_button.dart

import 'package:flutter/material.dart';

class ProfileButtons extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Row(
            mainAxisAlignment: MainAxisAlignment.spaceAround,
            children: [
                _buildFollowButton(),
                _buildMessageButton(),
            ],
        );
    }

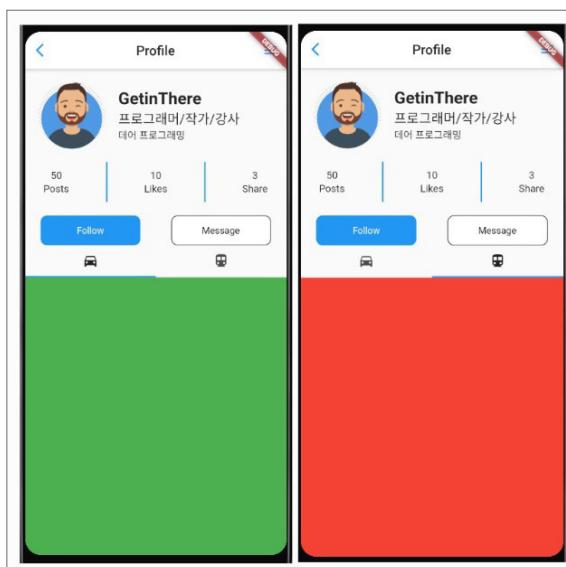
    Widget _buildFollowButton() {
        return InkWell(
            onTap: () {
                print("Follow 버튼 클릭됨");
            },
            child: Container(
                alignment: Alignment.center, // 컨테이너 내부 Text 위젯 정렬시 사용
                width: 150,
                height: 45,
                child: Text(
                    "Follow",
                    style: TextStyle(color: Colors.white),
                ),
                decoration: BoxDecoration(
                    color: Colors.blue,
                    borderRadius: BorderRadius.circular(10), // 컨테이너 모서리를 둥글게 한다.
                ),
            ),
        );
    }
}
```



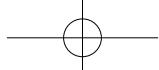
```
        ),
        ),
    ); // end of InkWell
}

Widget _buildMessageButton() {
    return InkWell(
        onTap: () {
            print("Message 버튼 클릭됨");
        },
        child: Container(
            alignment: Alignment.center,
            width: 150,
            height: 45,
            child: Text(
                "Message",
                style: TextStyle(color: Colors.black),
            ),
            decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: BorderRadius.circular(10),
                border: Border.all(), // 컨테이너에 테두리 선을 준다.
            ),
        ),
    );
}
}); // end of InkWell
}
}
```

TabBar 위젯과 TabBarView 위젯 사용하기



◆ 6.3.6 완성 화면
129 플로터로 앱 만들기



❶ TabBar

탭의 가로 행을 표시하는 머티리얼 디자인 위젯입니다.

❷ TabBarView

현재 선택된 탭에 해당하는 화면 표시하는 위젯입니다. TabBar 위젯과 TabBarView 위젯의 controller 속성에 TabController 객체를 연결하면 TabBarView가 선택된 Tab에 따라 화면을 변경하게 됩니다.

❸ SingleTicketProviderStateMixin

SingleTicketProviderStateMixin은 한 개의 애니메이션을 가진 위젯을 정의할 때 사용합니다. 두 개 이상의 애니메이션을 가진 위젯을 정의하려면 TickerProviderStateMixin을 사용해야 합니다.

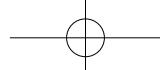
[기본 개념] → 이것만 이해하셔도 됩니다

SingleTicketProviderStateMixin은 Mixin 타입입니다. Mixin은 클래스가 가지고 있는 코드를 재사용하기 위해서 만들어졌습니다. 보통 프로그래밍 언어는 다중 상속이 불가능합니다. ProfileTabState 클래스는 State<ProfileTab> 클래스를 상속하고 있기 때문에 다른 클래스를 상속 할 수 없지만 SingleTicketProviderStateMixin은 mixin 타입이기 때문에 상속이 가능합니다. 이때는 extends를 사용하지 않고 with 키워드를 사용해야 합니다.

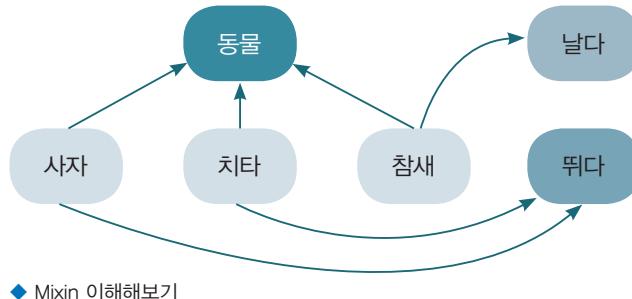
[깊은 개념] → 어려우니 넘어가셔도 됩니다

프로그래밍 언어에서 상속을 사용하는 주된 이유는 재사용 보다는 타입을 일치시키거나 함수를 Overriding(재정의) 하는데 있습니다. 재사용하기 위해서는 Composition(결합)을 하면 됩니다. 여기서 다형성이라는 개념이 나오게 되는데 다형성이란 여러 가지 형태를 가질 수 있다는 뜻입니다. 예를 들어 사자는 사자이기도 하지만 동물이기도 합니다. 동물은 추상적인 개념이지만 사자는 실제 하는 존재입니다. 여기서 사자를 자식이라고 하고 동물을 부모라고 합니다. 그 이유는 동물은 사자뿐 만 아니라 토끼, 기린, 코끼리 등 다양한 것들을 아우를 수 있는 더 큰 개념이기 때문입니다.

사자가 동물을 상속하였는데 사자가 추가적으로 식물을 상속할 수 없습니다. 부모가 두 명이 되는 순간 타입의 일치성이 사라지고 다형성의 개념이 모호해집니다. 하지만 프로그래밍을 하다 보면 다중 상속이 필요한 경우가 생기게 됩니다. 이때는 보통 인터페이스를 사용하여 타입을 일치시키고 행동(함수)을 강제시킵니다. 하지만 인터페이스는 함수를 재사용하는 것에 목적이 있는 것이 아닙니다.



다. 타입을 일치시키면서 부모 클래스의 변수나 함수를 재사용하기 위해서 탄생한 것이 바로 Mixin입니다.



◆ Mixin 이해해보기

TIP

Mixin (is-A 관계)

- 사자는 동물이면서 뛰다이다. 참새는 동물이면서 날다이다.

Composition (has-A 관계)

- 사자는 동물이며 뛰다 기능을 가졌다. 참새는 동물이면서 날다 기능을 가졌다.

타입 일치

- 프로그래밍에서 타입을 일치시키게 되면 런타임 시 동적으로 다른 행동을 할 수 있는 프로그램을 짤 수 있습니다.

lib/components/profile_tab.dart

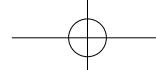
```
import 'package:flutter/material.dart';

class ProfileTab extends StatefulWidget {
  @override
  _ProfileTabState createState() => _ProfileTabState();
}

class _ProfileTabState extends State<ProfileTab> with SingleTickerProviderStateMixin {
  TabController? _tabController;

  @override
  void initState() {
    super.initState();
    _tabController = new TabController(length: 2, vsync: this);
  }

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        _buildTabBar(),
        Expanded(child: _buildTabBarView()), // end of Expanded
      ],
    );
  }
}
```



```
        ],
    );
}

Widget _buildTabBar() {
    return TabBar(
        controller: _tabController,
        tabs: [
            Tab(icon: Icon(Icons.directions_car)),
            Tab(icon: Icon(Icons.directions_transit)),
        ],
    ); // end of TabBar
}

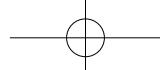
Widget _buildTabBarView() {
    return TabBarView(
        controller: _tabController,
        children: [
            Container(color: Colors.green),
            Container(color: Colors.red),
        ],
    ); // end of TabBarView
}
```

TIP

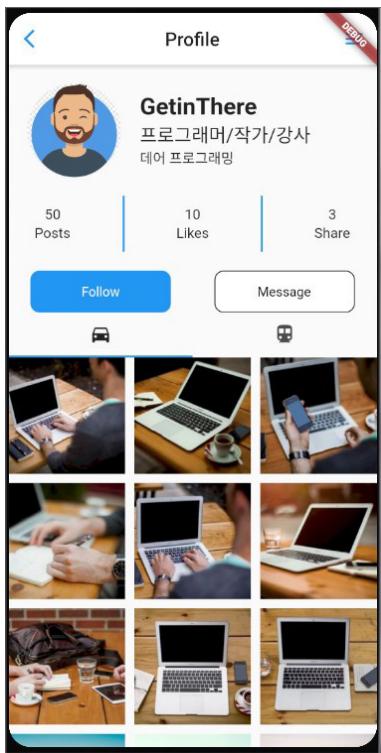
`initState()` 함수는 `StatefulWidget` 에만 존재하는 초기화를 위한 함수입니다. `ProfileTab` 위젯이 최초에 핸드폰에 그림 그려질 때 단 한 번만 실행되는 함수입니다. 그 뒤에 데이터 변경으로 인해 그림이 다시 그려진다고 하더라도 `initState()` 함수는 실행되지 않습니다.

TIP

`vsync:this`는 해당 위젯의 싱크를 `SingleTickerProviderStateMixin`에 맞춘다는 뜻입니다. `SingleTickerProviderStateMixin`은 내부적으로 `AnimatedController` 위젯으로 구현되어 있는데 현재 화면에 상태가 변경되면(탭을 클릭하면) 애니메이션이 발동되도록 싱크를 맞춘다는 의미입니다.



GridView 위젯과 Image.network

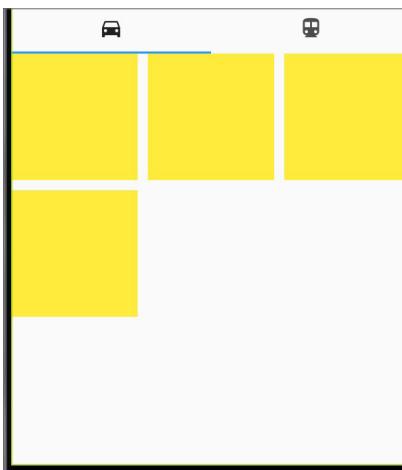


◆ 6.3.7 완성 화면

GridView는 수평방향이나 수직방향으로 고정 수의 위젯을 생성하고 반복해서 List를 출력해주는 위젯입니다. TabBarView의 첫 번째 조록 Container를 GridView.builder로 변경해보겠습니다.

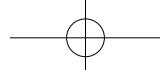
GridView에 들어갈 Item의 양이 정해져 있다면 다음과 같이 GridView로 구성할 수 있습니다. 아래에는 GridView에 자식으로 Container가 4개가 들어가 있습니다. 아래 이미지의 코드는 코딩할 필요 없습니다.

```
GridView(  
    gridDelegate: SilverGridDelegateWithFixedCrossAxisCount(  
        crossAxisSpacing: 10,  
        crossAxisCount: 3,  
        mainAxisSpacing: 10,  
    ), // SilverGridDelegateWithFixedCrossAxisCount  
    scrollDirection: Axis.vertical,  
    children: [  
        Container(  
            color: Colors.yellow,  
        ), // Container  
        Container(  
            color: Colors.yellow,  
        ), // Container  
        Container(  
            color: Colors.yellow,  
        ), // Container  
        Container(  
            color: Colors.yellow,  
        ), // Container  
    ], // Container  
, // GridView
```



◆ 길이가 정해져 있는 GridView 예시

◆ 길이가 정해져 있는 GridView 코드 실행 화면

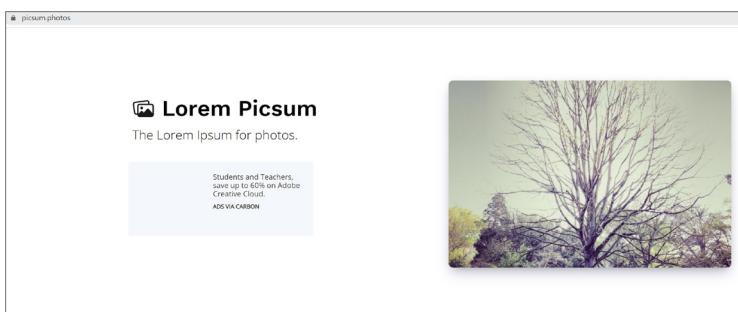


하지만 GridView에 들어올 item의 개수가 동적이라면 GridView로 구성할 수 없습니다. 이럴 때는 GridView.builder를 사용해야 합니다.

lib/components/profile_tab.dart

```
lib/components/profile_tab.dart
Widget _buildTabBarView() {
    return TabBarView(
        controller: _tabController,
        children: [
            GridView.builder(
                gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
                    crossAxisSpacing: 10,
                    crossAxisCount: 3,
                    mainAxisSpacing: 10,
                ),
                itemCount: 42, // 1. GridView에 표시할 총 item 개수
                itemBuilder: (context, index) {
                    return Image.network(
                        "https://picsum.photos/id/${index + 1}/200/200");
                },
            ), // end of GridView.builder
            Container(color: Colors.red),
        ],
    );
}
```

Image.network를 사용하면 url 주소를 사용하여 이미지를 다운로드 한 뒤 화면에 표시할 수 있습니다. <https://picsum.photos> 사이트는 이미지를 무료로 제공해줍니다. 이미지가 필요할 때 유용하게 활용할 수 있는 사이트입니다.



◆ 이미지 다운로드 사이트

TIP

pub.dev 사이트에 가면 CachedNetworkImage 라이브러리가 있습니다. 해당 라이브러리를 사용하여 Image를 UI에 표시하게 되면 Image가 캐싱되어 한 번 다운 받은 Image는 다시 다운 받지 않기 때문에 앱에 성능이 좋아집니다.