# 앱 개발 입문

성균관대학교 컬처앤테크놀로지융합전공 하계 부트캠프

5일차 - 2023. 07. 03 (월)

## 강의 슬라이드 링크

https://github.com/kunny/skku-bootcamp-2023-summer/blob/main/_slides/day5.pdf

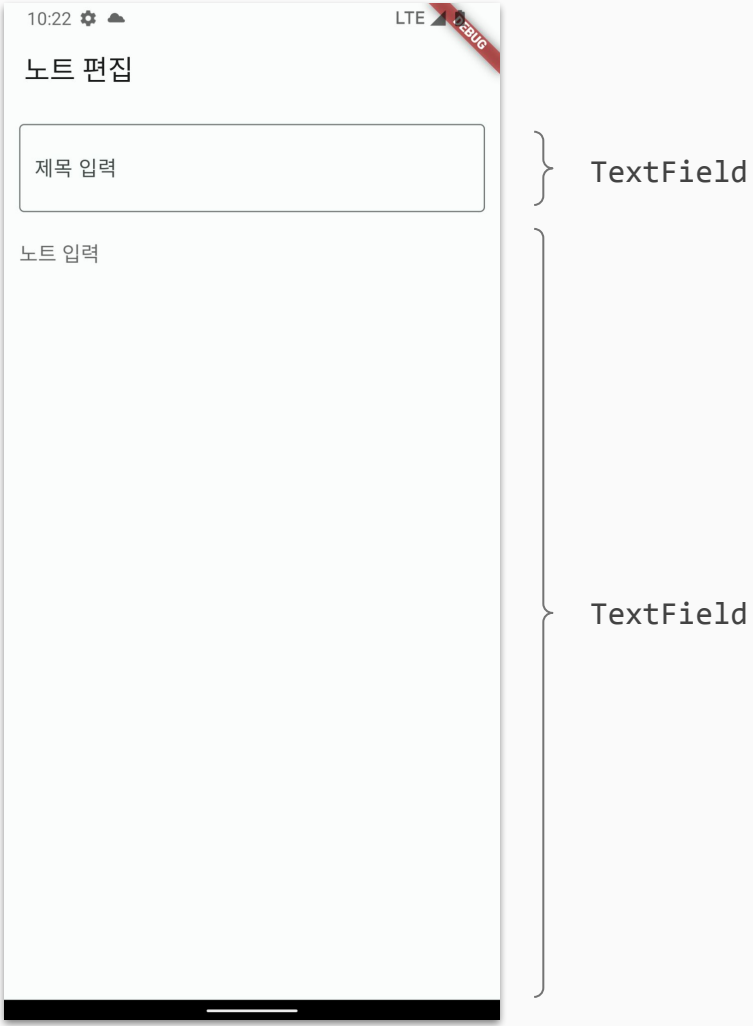
# 오늘 강의에서 다룰 내용

- 노트 편집 화면을 구현합니다.
- 노트 편집 화면에서 노트 색상을 선택하는 기능을 추가합니다.
- 노트 데이터를 관리하는 클래스를 작성합니다.
- 노트 데이터를 관리하는 클래스를 사용하도록 기존 코드를 수정합니다.
- 노트 보기 화면을 구현합니다.

# 노트 편집 화면 작성하기

# 노트 편집 화면 작성하기

1. **page** 폴더 아래에 **note_edit_page.dart** 파일을 생성합니다.
2. 노트 본문을 편집하는 위젯인 NoteEditPage 코드를 작성합니다.
3. NoteEditPage 위젯을 앱의 첫 화면으로 설정합니다.


10:22
LTE
DEBUG
노트 편집
제목 입력
노트 입력
TextField
TextField

# 노트 편집 화면 작성하기

```dart
class NoteEditPage extends StatefulWidget {
  const NoteEditPage({super.key});

  @override
  State createState() => _NoteEditPageState();
}

class _NoteEditPageState extends State<NoteEditPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('노트 편집'),
      ),
    );
  }
}
```

note_edit_page.dart

# 노트 편집 화면 작성하기

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('노트 편집'),
    ),
    body: const Column(
      mainAxisSize: MainAxisSize.min,
      children: [
        TextField(
          decoration: InputDecoration(
            border: OutlineInputBorder(),
            labelText: '제목 입력',
          ),
          maxLines: 1,
          style: TextStyle(fontSize: 20.0),
        ),
        SizedBox(height: 8.0),
        TextField(
          decoration: InputDecoration(
            border: InputBorder.none,
            hintText: '노트 입력',
          ),
          maxLines: null,
          keyboardType: TextInputType.multiline,
        ),
      ],
    ),
  );
}
```

note_edit_page.dart

# 노트 편집 화면 작성하기

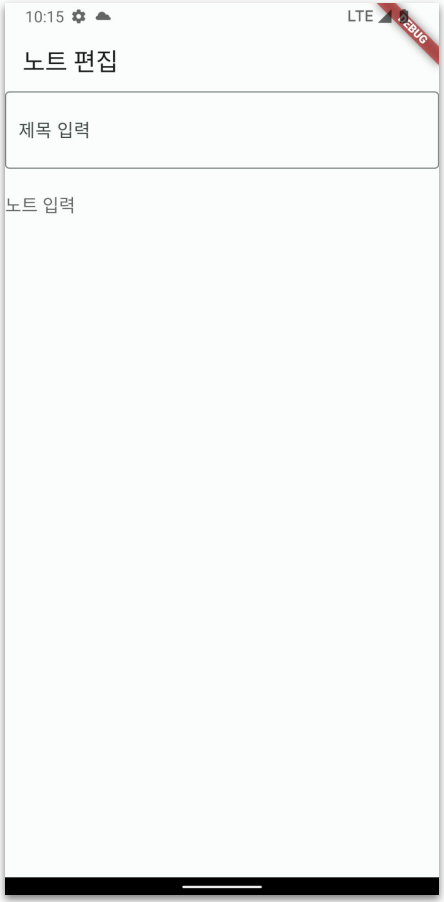
```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Sticky Notes',
      theme: ThemeData(
        primarySwatch: Colors.teal,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: const NoteEditPage(),
    );
  }
}
```

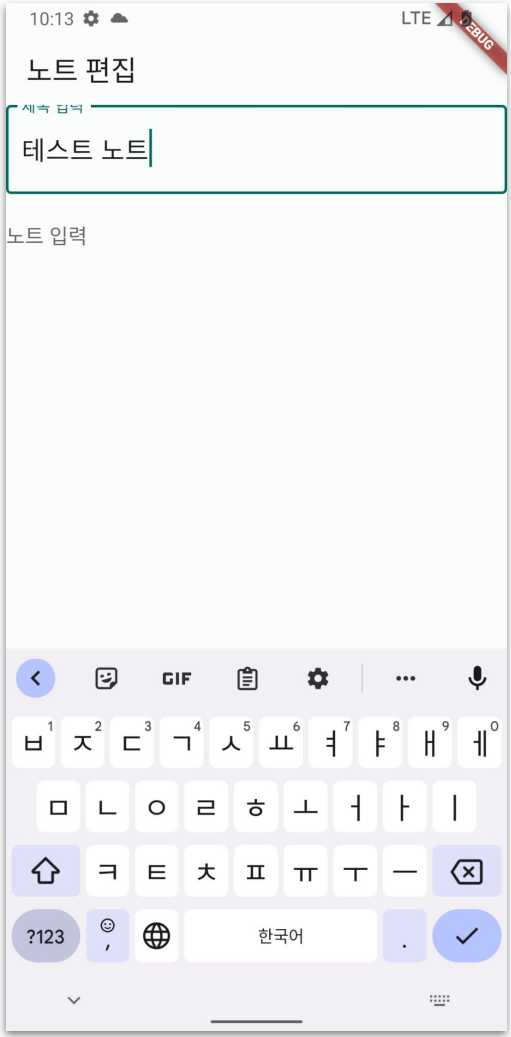
앱의 첫 화면을 노트 편집 화면으로 변경합니다.

main.dart

# 노트 편집 화면 작성하기

1. 앱 실행 후, 텍스트 필드에 제목과 본문을 입력해봅니다.
2. 화면이 의도한 대로 동작하는지 확인해봅니다.





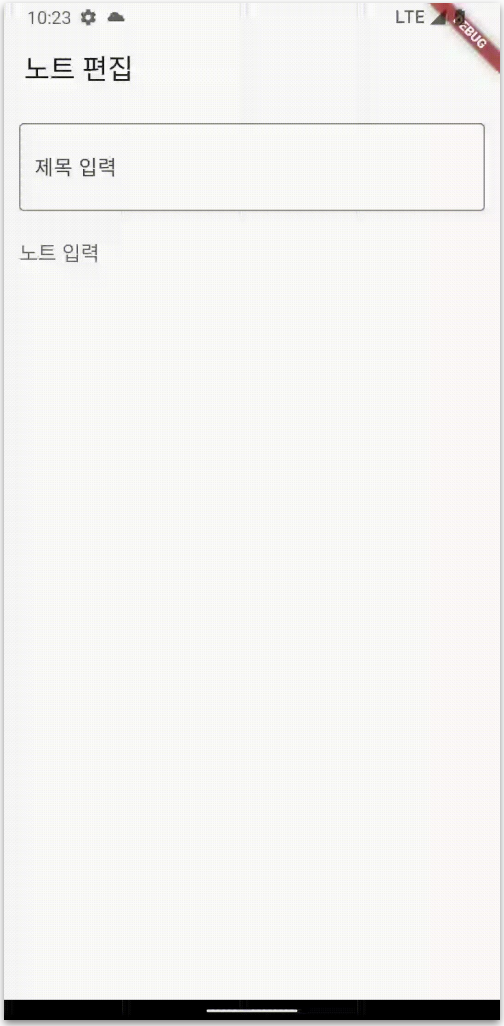

여백 부족


화면 크기를 넘어가는
문장을 입력하면
오버플로우 발생

# 노트 편집 화면 작성하기

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: ...,
    body: SingleChildScrollView(
      padding: EdgeInsets.symmetric(horizontal: 12.0, vertical: 16.0),
      child: Column(
        mainAxisSize: MainAxisSize.min,
        children: [ ... ],
      ),
    ),
  );
}
```

note_edit_page.dart


10:23
LTE
DEBUG
노트 편집
제목 입력
노트 입력

# 노트 색상 선택 기능 추가하기

# 노트 색상 선택 기능 추가하기

1. `_NoteEditPageState` 클래스에 노트 색상값을 저장할 수 있는 필드를 추가합니다.
2. `_NoteEditPageState` 클래스에 노트 색상 선택 다이얼로그를 표시하는 `_displayColorSelectionDialog()` 함수를 구현합니다.
3. 노트 색상 선택 다이얼로그를 닫고 화면을 갱신해주는 `_applyColor()` 함수를 구현합니다.
4. 노트 본문 입력 화면 배경에 노트 색상을 표시합니다.

# 노트 색상 선택 기능 추가하기

```dart
class _NoteEditPageState extends State<NoteEditPage> {

  Color color = Note.colorDefault;

  @override
  Widget build(BuildContext context) { ... }

  void _applyColor(Color newColor) {
    setState(() {
      Navigator.pop(context);
      color = newColor;
    });
  }
}
```

note_edit_page.dart

# 노트 색상 선택 기능 추가하기

```dart
class _NoteEditPageState extends State<NoteEditPage> {

  void _displayColorSelectionDialog() {
    FocusManager.instance.primaryFocus?.unfocus();
    showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          title: const Text('배경색 선택'),
          content: Column(
            mainAxisSize: MainAxisSize.min,
            children: [ ... ],
          ),
        );
      },
    );
  }
}
```

소프트 키보드가 올라와 있을 때
키보드를 내려줍니다.

note_edit_page.dart

# 노트 색상 선택 기능 추가하기

```
// 윗 부분 코드 생략
return AlertDialog(
  title: const Text('배경색 선택'),
  content: Column(
    mainAxisSize: MainAxisSize.min,
    children: [
      ListTile(
        title: const Text('없음'),
        onTap: () => _applyColor(Note.colorDefault),
      ),
      ListTile(
        leading: const CircleAvatar(backgroundColor: Note.colorRed),
        title: const Text('빨간색'),
        onTap: () => _applyColor(Note.colorRed),
      ),
      ListTile(
        leading: const CircleAvatar(backgroundColor: Note.colorOrange),
        title: const Text('오렌지색'),
        onTap: () => _applyColor(Note.colorOrange),
      ),
      ListTile(
        leading: const CircleAvatar(backgroundColor: Note.colorYellow),
        title: const Text('노란색'),
        onTap: () => _applyColor(Note.colorYellow),
      ),
      ListTile(
        leading: const CircleAvatar(backgroundColor: Note.colorLime),
        title: const Text('연두색'),
        onTap: () => _applyColor(Note.colorLime),
      ),
      ListTile(
        leading: const CircleAvatar(backgroundColor: Note.colorBlue),
        title: const Text('파란색'),
        onTap: () => _applyColor(Note.colorBlue),
      ),
    ],
  ),
);
```

note_edit_page.dart

# 노트 색상 선택 기능 추가하기

```
class _NoteEditPageState extends State<NoteEditPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('노트 편집'),
        actions: [
          IconButton(
            icon: const Icon(Icons.color_lens),
            tooltip: '배경색 선택',
            onPressed: _displayColorSelectionDialog,
          ),
        ],
      ),
      ...
    );
  }
  ...
}
```

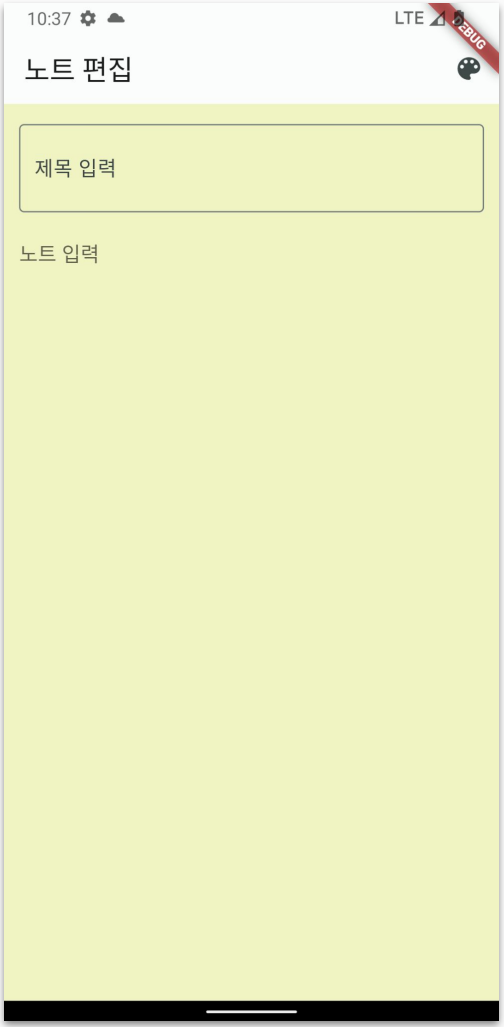
note_edit_page.dart

# 노트 색상 선택 기능 추가하기

```
class _NoteEditPageState extends State<NoteEditPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar( ... ),
      body: Container(
        height: double.infinity,
        color: _color,
        child: const SingleChildScrollView(
          padding: EdgeInsets.symmetric(horizontal: 12.0, vertical: 16.0),
          child: Column( ... ),
          ...
        ),
      ),
    );
  }
}
```

note_edit_page.dart


10:37
LTE
DEBUG
노트 편집
제목 입력
노트 입
배경색 선택
없음
빨간색
오렌지색
노란색
연두색
파란색


10:37
LTE
DEBUG
노트 편집
제목 입력
노트 입력

# 노트 데이터 관리 클래스 작성하기

# 노트 데이터 관리 클래스 작성하기

1. **service** 폴더에 노트를 추가, 수정, 삭제 및 조회하는 **NoteService** 클래스를 구현합니다.
2. **providers.dart** 파일에 **NoteService** 클래스의 인스턴스를 저장할 수 있는 필드를 추가하고, 이를 참조할 때 사용할 **noteManager()** 함수를 구현합니다.
3. **NoteService**를 사용하여 노트 목록을 표시하도록 **_NoteListPageState** 클래스를 수정합니다.
4. **NoteService**를 사용하여 노트를 추가하도록 **_NoteEditPageState** 클래스를 수정합니다.

# 노트 데이터 관리 클래스 작성하기

```dart
import 'package:sticky_notes/data/note.dart';

class NoteService {
  final _notes = <Note>[];

  void addNote(Note note) {
    _notes.add(note);
  }

  void deleteNote(int index) {
    _notes.removeAt(index);
  }

  Note getNote(int index) {
    return _notes[index];
  }

  List<Note> listNotes() {
    return _notes;
  }

  void updateNote(int index, Note note) {
    _notes[index] = note;
  }
}
```

note_service.dart

# 노트 데이터 관리 클래스를 사용하도록 기존 코드 수정하기

# 노트 데이터 관리 클래스를 사용하도록 기존 코드 수정하기

```dart
import 'package:sticky_notes/service/note_service.dart';

NoteService? _noteService;

NoteService noteService() {
  if (_noteService == null) {
    _noteService = NoteService();
  }
  return _noteService!;
}
```

providers.dart

# 노트 데이터 관리 클래스를 사용하도록 기존 코드 수정하기

```
class _NoteListPageState extends State<NoteListPage> {

  // 샘플 노트를 삭제합니다.

  bool _showAsGrid = true;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar( ... ),
      // noteService().listNotes()를 사용하여 노트 목록을 불러옵니다.
      body: _buildCards(noteService().listNotes()),
    );
  }
}
```

note_list_page.dart

# 노트 데이터 관리 클래스를 사용하도록 기존 코드 수정하기

```
class _NoteEditPageState extends State<NoteEditPage> {

  final _titleController = TextEditingController();

  final _bodyController = TextEditingController();

  ...

  @override
  Widget build(BuildContext context) { ... }
}
```

note_edit_page.dart

# 노트 데이터 관리 클래스를 사용하도록 기존 코드 수정하기

```
Column(
  mainAxisSize: MainAxisSize.min,
  children: [
    TextField(
      controller: _titleController,
      decoration: const InputDecoration(
        border: OutlineInputBorder(),
        labelText: '제목 입력',
      ),
      maxLines: 1,
      style: const TextStyle(fontSize: 20.0),
    ),
    const SizedBox(height: 8.0),
    TextField(
      controller: _bodyController,
      decoration: const InputDecoration(
        border: InputBorder.none,
        hintText: '노트 입력',
      ),
      maxLines: null,
      keyboardType: TextInputType.multiline,
    ),
  ],
),
```

note_edit_page.dart

# 노트 데이터 관리 클래스를 사용하도록 기존 코드 수정하기

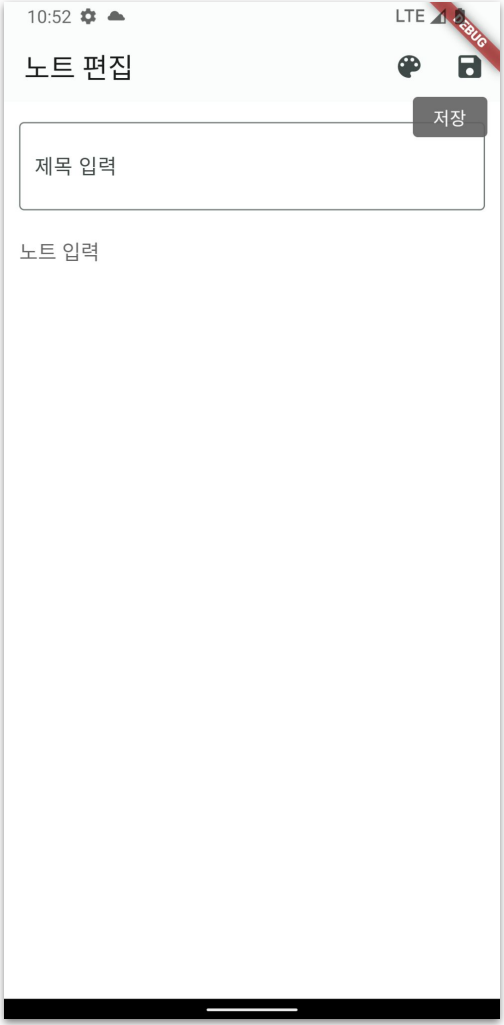
```dart
class _NoteEditPageState extends State<NoteEditPage> {
  ...
  void _saveNote() {
    if (_bodyController.text.isNotEmpty) {
      final note = Note(
        _bodyController.text,
        title: _titleController.text,
        color: color,
      );

      // 노트를 추가합니다.
      noteService().addNote(note);
    } else {
      ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
        content: Text('노트를 입력하세요.'),
        behavior: SnackBarBehavior.floating,
      ));
    }
  }
}
```

note_edit_page.dart

# 노트 데이터 관리 클래스를 사용하도록 기존 코드 수정하기

```
class _NoteEditPageState extends State<NoteEditPage> {
  ...
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: ...,
        actions: [
          IconButton( ... ),
          IconButton(
            icon: const Icon(Icons.save),
            tooltip: '저장',
            onPressed: _saveNote,
          ),
        ],
      ),
      body: Container( ... ),
    );
  }
  ...
}
```

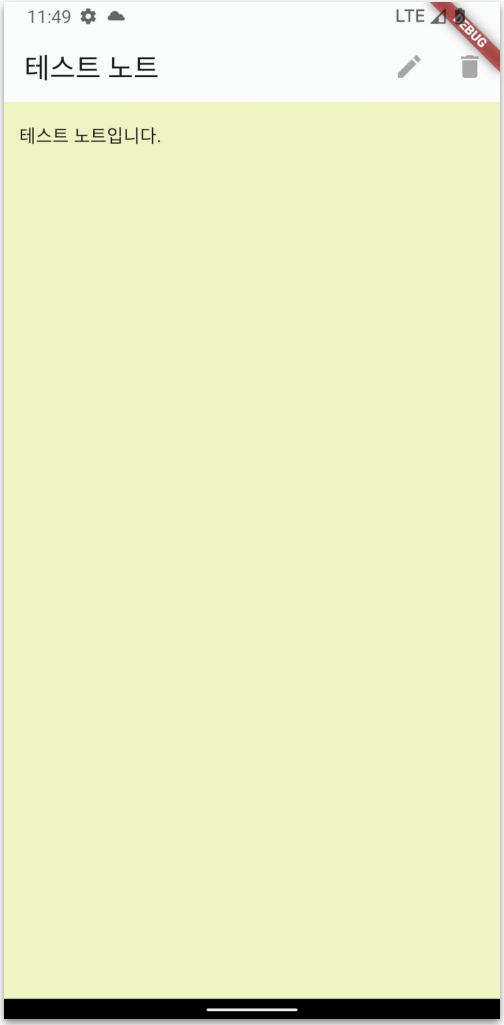
note_edit_page.dart


10:52
LTE
DEBUG
노트 편집
저장
제목 입력
노트 입력

# 노트 보기 화면 구현하기

# 노트 보기 화면 구현하기

1. **page** 폴더에 **note_view_page.dart** 파일을 추가합니다.
2. **note_view_page.dart** 파일에 **NoteViewPage** 클래스를 구현합니다.


11:49
LTE
DEBUG
테스트 노트
테스트 노트입니다.

# 노트 본문 화면 작성하기

```dart
class NoteViewPage extends StatefulWidget {
  final int index;

  const NoteViewPage(this.index, {super.key});

  @override
  State createState() => _NoteViewPageState();
}
```

note_view_page.dart

# 노트 보기 화면 작성하기

```dart
class _NoteViewPageState extends State<NoteViewPage> {
  @override
  Widget build(BuildContext context) {
    final note = noteService().getNote(widget.index);
    return Scaffold(
      appBar: AppBar(
        title: Text(note.title.isEmpty ? '(제목 없음)' : note.title),
        actions: [
          IconButton(
            icon: const Icon(Icons.edit),
            tooltip: '편집',
            onPressed: null,
          ),
          IconButton(
            icon: const Icon(Icons.delete),
            tooltip: '삭제',
            onPressed: null,
          ),
        ],
      ),
      body: Container(
        width: double.infinity,
        height: double.infinity,
        color: note.color,
        child: SingleChildScrollView(
          padding: const EdgeInsets.symmetric(
              horizontal: 12.0, vertical: 16.0),
          child: Text(note.body),
        ),
      ),
    );
  }
}
```

note_view_page.dart

# 플러터에서 화면 이동 처리하기

# 플러터에서 화면 이동 처리하기

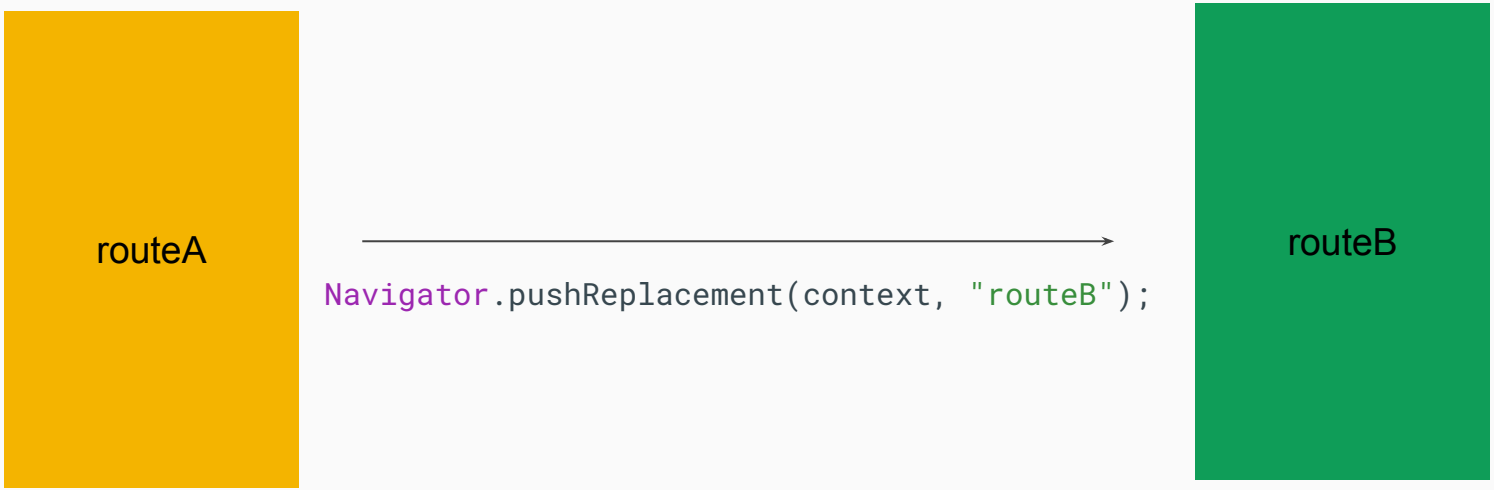
- 각 화면은 스택(Stack) 방식으로 관리되며, **`Navigator`** 클래스 내 다양한 함수를 사용하여 화면을 이동합니다.
- 앱 내의 각 화면은 **루트(Route)**라 부르며, 이름을 지정한 화면은 **Named Route**라 부릅니다.
- **`MaterialApp`**의 **`routes`** 속성을 사용하여 앱에서 이동할 수 있는 화면을 정의합니다.
- Named Route를 호출할 때에는 인자(**`arguments`**)를 추가로 전달할 수 있습니다.
- 호출했던 화면에서 기존 화면으로 돌아왔을 때 이벤트를 받을 수 있습니다. (**`Future`**)

# 플러터에서 화면 이동 처리하기

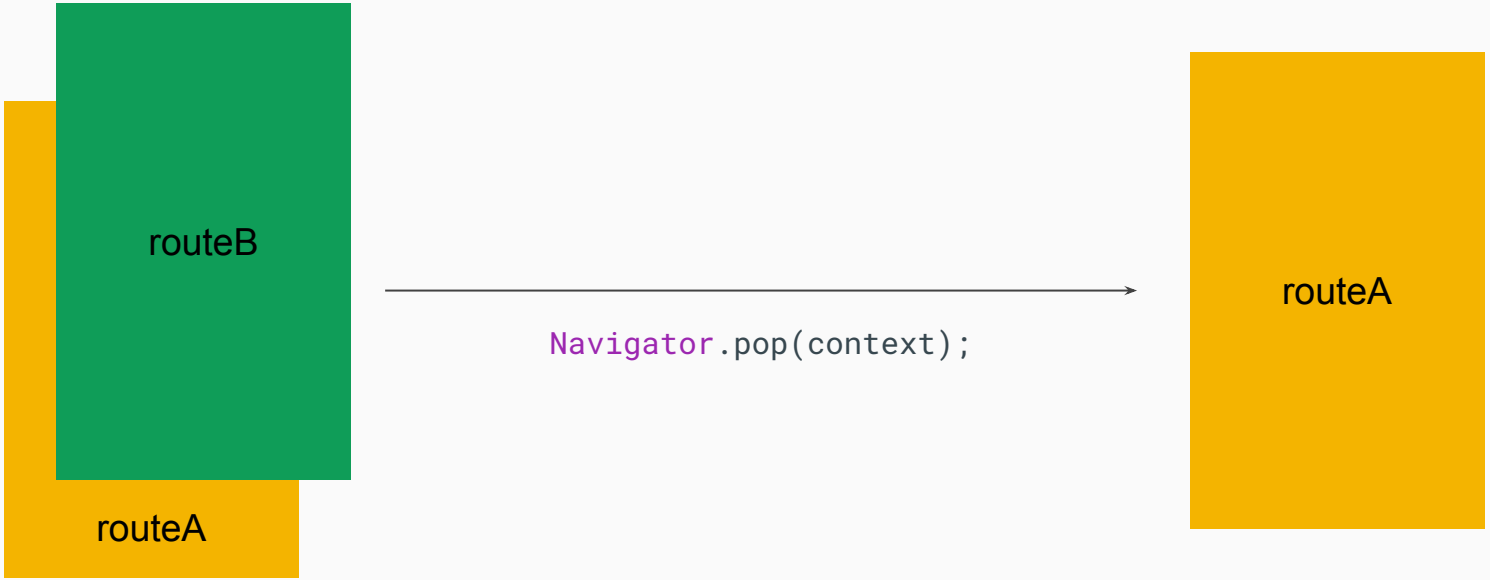
- 새 화면 열기 (현재 화면 유지)
  - Navigator.push() / Navigator.pushNamed()
- 현재 화면을 새 화면으로 대체
  - Navigator.pushReplacement() / Navigator.pushReplacementNamed()
- 현재 화면을 닫고 기존 화면으로 복귀
  - Navigator.pop()
- 현재 화면을 닫고 특정 조건을 만족하는 화면으로 복귀
  - Navigator.popUntil()

# 플러터에서 화면 이동 처리하기

# 플러터에서 화면 이동 처리하기



# 플러터에서 화면 이동 처리하기



# 플러터에서 화면 이동 처리하기

# 화면 루트 구성하기

# 화면 루트 구성하기

1. **NoteListPage**, **NoteEditPage** 및 **NoteViewPage**화면에 이름을 지어줍니다.
2. **MyApp** 클래스 내 **MaterialApp**의 **initialRoute**와 **routes** 속성을 추가하여 앱의 첫 페이지와 앱에 포함되는 화면을 구성합니다.

# 화면 루트 구성하기

```dart
class NoteListPage extends StatefulWidget {
  static const routeName = '/';

  const NoteListPage({Key? key}) : super(key: key);

  @override
  State createState() => _NoteListPageState();
}
```

note_list_page.dart

# 화면 루트 구성하기

```dart
class NoteEditPage extends StatefulWidget {
  static const routeName = '/edit';

  const NoteEditPage({Key? key}): super(key: key);

  @override
  State createState() => _NoteEditPageState();
}
```

note_edit_page.dart

# 화면 루트 구성하기

```dart
class NoteViewPage extends StatefulWidget {
  static const routeName = '/view';

  final int index;

  const NoteViewPage(this.index, {super.key});

  @override
  State createState() => _NoteViewPageState();
}
```

note_view_page.dart

# 노트 보기 화면 작성하기

```
class _NoteViewPageState extends State<NoteViewPage> {
  ...
  void _edit(int index) {
    Navigator.pushNamed(
      context,
      NoteEditPage.routeName,
      arguments: index,
    ).then((value) {
      setState(() {});
    });
  }
}
```

note_view_page.dart

# 노트 보기 화면 작성하기

```
class _NoteViewPageState extends State<NoteViewPage> {
  ...
  void _confirmDelete(int index) {
    showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          title: const Text('노트 삭제'),
          content: const Text('노트를 삭제할까요?'),
          actions: [
            TextButton(
              child: const Text('아니오'),
              onPressed: () {
                Navigator.pop(context);
              },
            ),
            TextButton(
              child: const Text('예'),
              onPressed: () {
                noteService().deleteNote(index);
                Navigator.popUntil(context, (route) => route.isFirst);
              },
            ),
          ],
        );
      },
    );
  }
}
```

note_view_page.dart

# 노트 보기 화면 작성하기

```
class _NoteViewPageState extends State<NoteViewPage> {
  @override
  Widget build(BuildContext context) {
    ...
    return Scaffold(
      appBar: AppBar(
        title: Text(note.title.isEmpty ? '(제목 없음)' : note.title),
        actions: [
          IconButton(
            icon: const Icon(Icons.edit),
            tooltip: '편집',
            onPressed: () => _edit(widget.index),
          ),
          IconButton(
            icon: const Icon(Icons.delete),
            tooltip: '삭제',
            onPressed: () => _confirmDelete(widget.index),
          ),
        ],
      ),
    );
  }
}
```

note_view_page.dart

# 화면 루트 구성하기

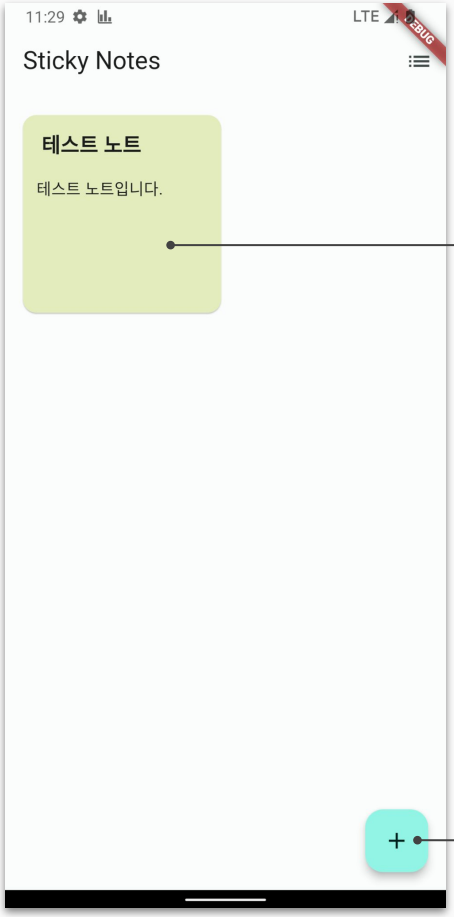
```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      ...
      initialRoute: NoteListPage.routeName,
      routes: {
        NoteListPage.routeName: (context) => const NoteListPage(),
        NoteEditPage.routeName: (context) => const NoteEditPage(),
        NoteViewPage.routeName: (context) {
          final index = ModalRoute.of(context)!.settings.arguments as int;
          return NoteViewPage(index);
        },
      },
    );
  }
}
```

main.dart

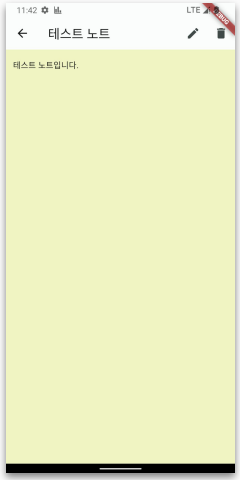
# 노트 목록, 보기, 편집 화면 연결하기

# 노트 목록, 보기, 편집 화면 연결하기

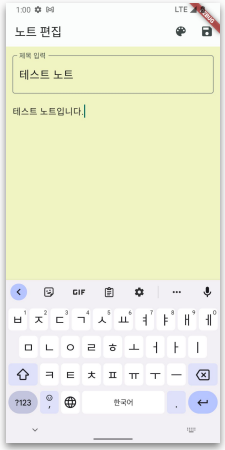
1. 노트 작성 화면이 새 노트를 작성하는 모드와 편집 모드를 지원하게끔 수정합니다.
2. 노트 목록 화면에 새 노트를 추가할 수 있는 버튼을 추가합니다.
3. 노트 목록에 표시된 노트를 눌러 노트 보기 화면으로 이동하게끔 코드를 수정합니다.




NoteListPage




NoteViewPage




NoteEditPage

# 노트 목록, 보기, 편집 화면 연결하기

```dart
class NoteEditPage extends StatefulWidget {
  static const routeName = '/edit';

  final int? index;

  const NoteEditPage(this.index, {super.key});

  @override
  State createState() => _NoteEditPageState();
}
```

note_edit_page.dart

# 노트 목록, 보기, 편집 화면 연결하기

```dart
class _NoteEditPageState extends State<NoteEditPage> {
  final _titleController = TextEditingController();

  final _bodyController = TextEditingController();

  Color _color = Note.colorDefault;

  @override
  void initState() {
    super.initState();
    final noteIndex = widget.index;
    if (noteIndex != null) {
      final note = noteService().getNote(noteIndex);
      _titleController.text = note.title;
      _bodyController.text = note.body;
      _color = note.color;
    }
  }
}
```

note_edit_page.dart

# 노트 목록, 보기, 편집 화면 연결하기

```
class _NoteEditPageState extends State<NoteEditPage> {

  void _saveNote() {
    if (_bodyController.text.isNotEmpty) {
      final note = Note(
        _bodyController.text,
        title: _titleController.text,
        color: color,
      );

      final noteIndex = widget.index;
      if (noteIndex != null) {
        noteService().updateNote(noteIndex, note);
      } else {
        noteService().addNote(note);
      }

      Navigator.pop(context);
    } else { ... }
  }
}
```

note_edit_page.dart

# 노트 목록, 보기, 편집 화면 연결하기

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      ...
      routes: {
        ...
        NoteEditPage.routeName: (context) {
          final args = ModalRoute.of(context)!.settings.arguments;
          final index = args != null ? args as int : null;
          return NoteEditPage(index);
        },
      },
    );
  }
}
```

main.dart

# 노트 목록, 보기, 편집 화면 연결하기

```
class _NoteListPageState extends State<NoteListPage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      ...
      floatingActionButton: FloatingActionButton(
        tooltip: '새 노트',
        onPressed: () {
          Navigator.pushNamed(context, NoteEditPage.routeName).then((value) {
            setState(() {});
          });
        },
        child: const Icon(Icons.add),
      ),
    );
  }
}
```

note_list_page.dart

# 노트 목록, 보기, 편집 화면 연결하기

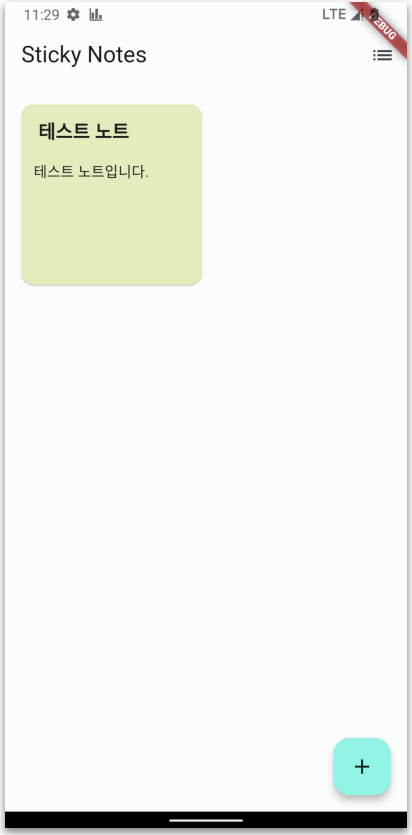
```
class _NoteListPageState extends State<NoteListPage> {
  ...

  Widget _buildCard(int index, Note note) {
    return InkWell(
      onTap: () {
        Navigator.pushNamed(
          context,
          NoteViewPage.routeName,
          arguments: index,
        ).then((value) {
          setState(() {});
        });
      },
      child: Card( ... ),
    );
  }
}
```

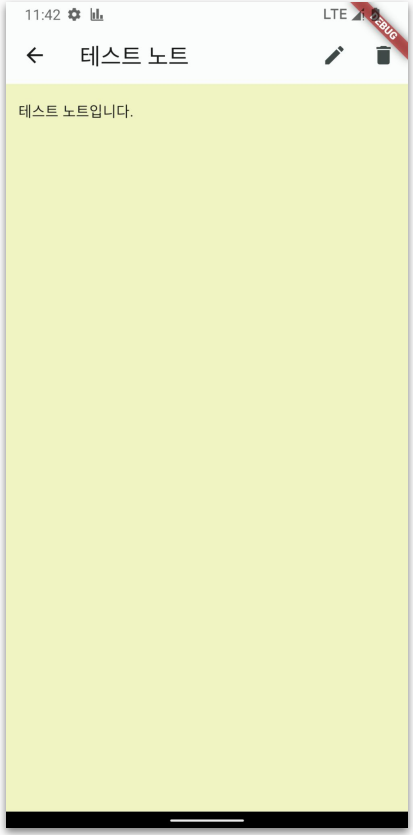
note_list_page.dart

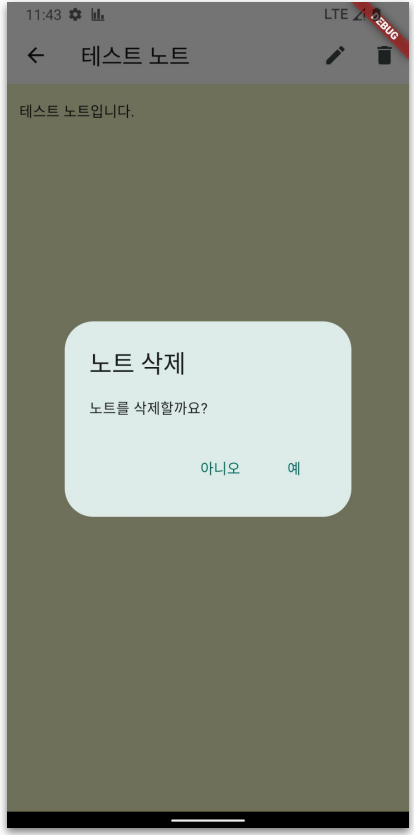
# 노트 목록, 보기, 편집 화면 연결하기

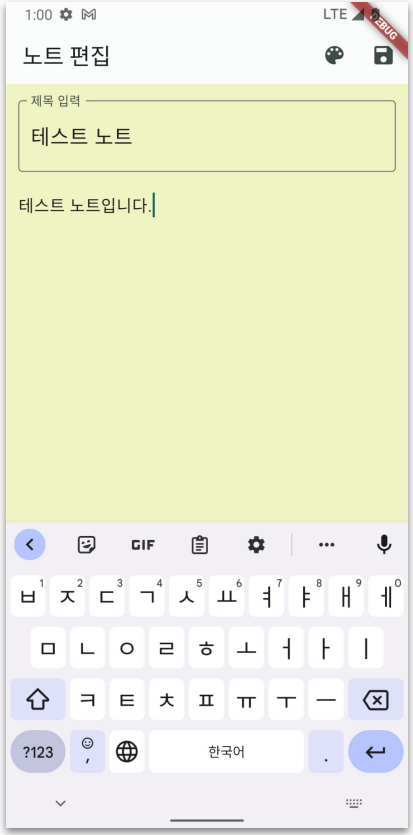
```
Widget _buildCards(List<Note> notes) {
  return _showAsGrid
      ? GridView.builder(
          padding:
              const EdgeInsets.symmetric(horizontal: 12.0, vertical: 16.0),
          itemBuilder: (context, index) => _buildCard(index, notes[index]),
          itemCount: notes.length,
          gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
            crossAxisCount: 2,
            childAspectRatio: 1,
          ),
        )
      : ListView.builder(
          padding:
              const EdgeInsets.symmetric(horizontal: 12.0, vertical: 16.0),
          itemCount: notes.length,
          itemBuilder: (context, index) {
            return SizedBox(
              height: 160,
              child: _buildCard(index, notes[index]),
            );
          },
        );
}
```

note_list_page.dart


Sticky Notes
테스트 노트
테스트 노트입니다.


테스트 노트
테스트 노트입니다.


테스트 노트
테스트 노트입니다.
노트 삭제
노트를 삭제할까요?
아니오
예


노트 편집
제목 입력
테스트 노트
테스트 노트입니다.
한국어

## 완성된 코드

https://github.com/kunny/skku-bootcamp-2023-summer/tree/main/sticky_notes/step3