

앱 개발 입문

성균관대학교 컬처애편테크놀로지융합전공 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.



} TextField

} TextField

노트 편집 화면 작성하기

```
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

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('노트 편집'),
    ),
    body: const Column(
      mainAxisAlignment: MainAxisAlignment.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

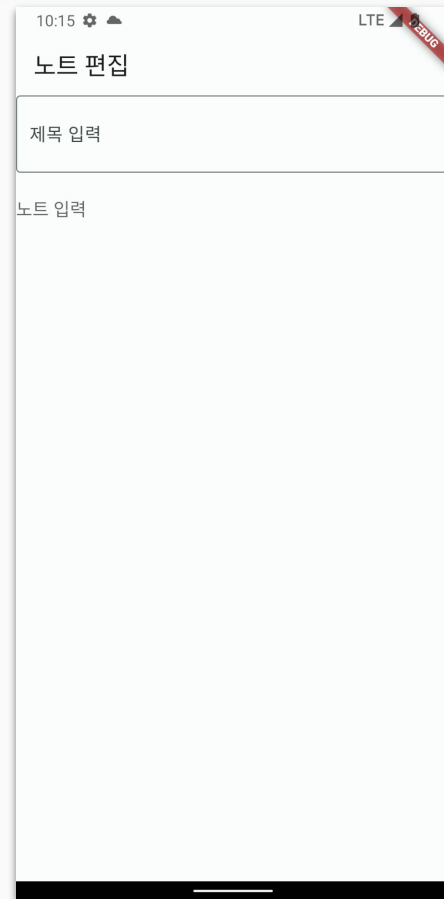
노트 편집 화면 작성하기

```
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.

노트 편집 화면 작성하기

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



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(
        mainAxisAlignment: MainAxisAlignment.min,
        children: [ ... ],
      ),
    ),
  );
}
```

10:23



LTE



BUG

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

```
class _NoteEditPageState extends State<NoteEditPage> {  
  
  Color color = Note.colorDefault;  
  
  @override  
  Widget build(BuildContext context) { ... }  
  
  void _applyColor(Color newColor) {  
    setState(() {  
      Navigator.pop(context);  
      color = newColor;  
    });  
  }  
}
```


노트 색상 선택 기능 추가하기

```
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(
```

```
          mainAxisAlignment: MainAxisAlignment.min,
```

```
          children: [ ... ],
```

```
        ),
```

```
      );
```

```
    },
```

```
  );
```

```
}
```

```
}
```

소프트 키보드가 올라와 있을 때
키보드를 내려줍니다.

note_edit_page.dart

노트 색상 선택 기능 추가하기

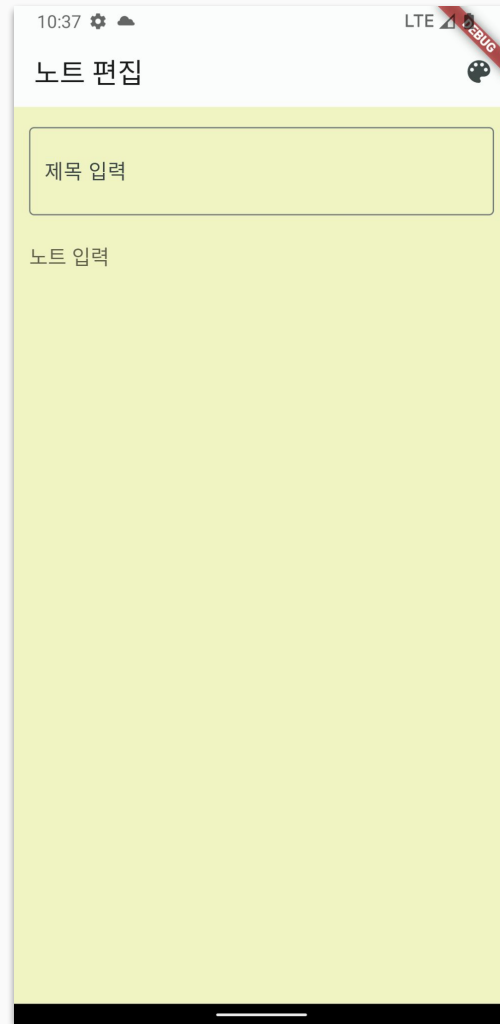
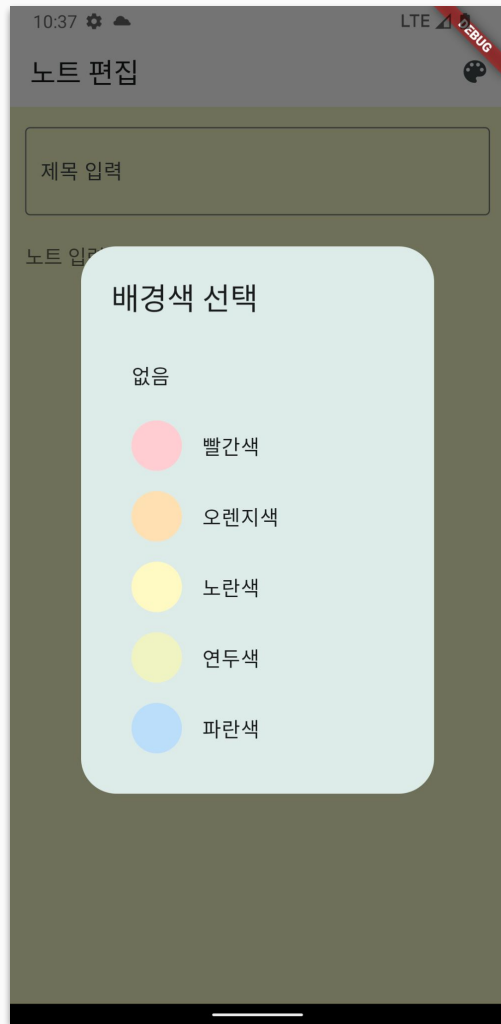
```
// 뒷 부분 코드 생략
return AlertDialog(
  title: const Text('배경색 선택'),
  content: Column(
    mainAxisAlignment: MainAxisAlignment.min,
    children: [
      ListTile(
        title: const Text('없음'),
        onTap: () => _applyColor(Note.colorDefault),
      ),
      ListTile(
        leading: const CircleAvatar(backgroundColor: 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),
      ),
    ],
  ),
);
```

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,
          ),
        ],
      ),
      ...
    );
  }
  ...
}
```

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( ... ),  
          ...  
        ),  
      ),  
    );  
  }  
}
```



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

```
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

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

NoteService? _noteService;

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

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()),
    );
  }
}
```

노트 데이터 관리 클래스를 사용하도록 기존 코드 수정하기

```
class _NoteEditPageState extends State<NoteEditPage> {  
  
  final _titleController = TextEditingController();  
  
  final _bodyController = TextEditingController();  
  
  ...  
  
  @override  
  Widget build(BuildContext context) { ... }  
}
```

노트 데이터 관리 클래스를 사용하도록 기존 코드 수정하기

```
Column(  
  mainAxisAlignment: MainAxisAlignment.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

```
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,  
      )));  
    }  
  }  
}
```

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



128MB

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

```
class NoteViewPage extends StatefulWidget {  
  final int index;  
  
  const NoteViewPage(this.index, {super.key});  
  
  @override  
  State createState() => _NoteViewPageState();  
}
```

노트 보기 화면 작성하기

```
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

완성된 코드

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