



## Day 3 학습 가이드 문서

### 상태 관리 + 일정 목록 표시 (복습 중심)

작성일: 2025년 7월 23일

대상: Flutter 학습자 (1.5개월 차)

목적: 복습을 통한 핵심 개념 체득

예상 소요 시간: 1시간 이내



### 문서 개요

#### 학습 목적

이 프로젝트는 포트폴리오 제작이 아닌 복습을 목적으로 합니다.

- 핵심 개념 체험: Map, setState, ListView, Navigator 패턴 익히기
- 실무 패턴 학습: 실제 개발에서 자주 사용하는 코딩 패턴 체득
- 점진적 성장: 어제 배운 Form과 오늘 배울 상태관리 연결

#### 완성 기준

- 동작만 하면 성공: UI가 예쁘지 않아도 됨
- 4가지 개념 이해: 각 복습 포인트 체득이 목표
- 빠른 완성: 1시간 내 핵심 기능 구현



### 사전 학습 확인 (수학 복습 퀴즈)

#### 문제 1: 간단한 제곱근 계산

$$\sqrt{16} + \sqrt{25} = ?$$

- 7
- 8
- 9 ✓
- 10

#### 문제 2: 대소 비교

다음 중 가장 큰 수는?

- 3.5
- $\sqrt{12}$
- $\sqrt{15}$  ✓

### 문제 3: 근호의 곱셈

$$\sqrt{8} \times \sqrt{2} = ?$$

1. 4✓
2.  $\sqrt{16}$
3.  $\sqrt{10}$
4.  $2\sqrt{4}$

정답: 3번, 3번, 1번

## 핵심 학습 과제 (4개)

### 과제 1: Map 데이터 구조 체험

목표: `Map<String, dynamic>` 구조로 복잡한 데이터 관리 경험

소요 시간: 10분

구현할 코드:

```
dart

// HomeScreen _HomeScreenState 클래스에 추가
List<Map<String, dynamic>> schedules = [];

@override
void initState() {
  super.initState();
  // 복습 핵심: Map 구조로 여러 데이터를 하나로 묶어서 관리
  schedules.add({
    'title': '테스트 일정',
    'date': DateTime.now(),
    'description': '복습용 샘플 데이터',
    'location': '집',
    'isCompleted': false,
  });
}
```

복습 포인트:

- Map의 키-값 구조 이해
- 여러 타입의 데이터(String, DateTime, bool)를 하나로 관리
- List와 Map의 조합으로 동적 데이터 집합 생성

## 📌 과제 2: ListView.builder 패턴 학습

목표: 동적 리스트 생성 패턴 익히기

소요 시간: 15분

구현할 코드:

```
dart

// HomeScreen의 body 부분을 다음과 같이 교체
body: ListView.builder(
  itemCount: schedules.length, // 리스트 개수
  itemBuilder: (context, index) { // 각 항목 생성
    return ListTile(
      title: Text(schedules[index]['title']),
      subtitle: Text(schedules[index]['description']),
      trailing: Icon(Icons.arrow_forward),
    );
  },
),
```

복습 포인트:

- itemBuilder 함수의 역할과 매개변수 이해
- index를 활용한 배열 접근 방법
- 동적 개수에 따른 자동 리스트 생성 원리

---

## 📌 과제 3: setState() 상태 변경 체험

목표: 상태 변경과 화면 업데이트 패턴 이해

소요 시간: 15분

구현할 코드:

```
dart
```

```
// HomeScreen에 새 함수 추가
void _addTestSchedule() {
  setState() { // 복습 핵심: setState 호출로 화면 업데이트
    schedules.add({
      'title': '새 일정 ${schedules.length + 1}',
      'date': DateTime.now(),
      'description': '버튼으로 추가된 일정',
      'isCompleted': false,
    });
  });
}

// Scaffold에 FloatingActionButton 추가
floatingActionButton: FloatingActionButton(
  onPressed: _addTestSchedule,
  child: Icon(Icons.add),
),
```

#### 복습 포인트:

- setState() 호출 타이밍과 역할
- 상태 변경 → 화면 리빌드 → 새 데이터 반영 과정
- 사용자 액션과 데이터 변경의 연결

## 📌 과제 4: Navigator 데이터 전달 패턴

목표: 화면 간 데이터 주고받는 방법 체득

소요 시간: 15분

#### 구현할 코드:

dart

```
// HomeScreen의 ListTile에 onTap 추가
ListTile(
  title: Text(schedules[index]['title']),
  subtitle: Text(schedules[index]['description']),
  trailing: Icon(Icons.arrow_forward),
  onTap: () {
    // 복습 핵심: arguments로 Map 데이터 전달
    Navigator.pushNamed(
      context,
      '/schedule_content',
      arguments: schedules[index],
    );
  },
),

// ScheduleAdd의 ElevatedButton onPressed 수정
ElevatedButton(
  onPressed: () {
    if (_formKey.currentState!.validate()) {
      // 복습 핵심: pop으로 데이터 반환
      Navigator.pop(context, {
        'title': _titleController.text,
        'date': date,
        'description': _descriptionController.text,
        'location': _locationController.text,
        'isCompleted': isCompleted,
      });
    }
  },
  child: Text("일정 추가"),
),
```

### 복습 포인트:

- pushNamed의 arguments 매개변수 활용
- pop의 반환값으로 데이터 전달
- 양방향 데이터 통신 패턴

## ✅ 학습 완료 체크리스트

### 개념 이해 확인 (핵심)

다음 질문에 답할 수 있으면 복습 성공:

☐ **Map 구조:** `schedules[index]['title']`는 어떻게 동작하는가?

- ☐ **StatefulWidget**: initState → setState → dispose 흐름을 설명할 수 있는가?
- ☐ **ListView**: itemBuilder에서 index는 어떤 역할을 하는가?
- ☐ **Navigator**: arguments와 pop 반환값의 차이점은?

## 기능 동작 확인 (최소 기준)

- ☐ 앱 실행 시 테스트 일정 1개 표시됨
  - ☐ + 버튼으로 새 일정이 리스트에 추가됨
  - ☐ 리스트 항목 터치 시 상세 화면으로 이동함
  - ☐ ScheduleAdd에서 입력한 데이터가 목록에 반영됨
- 

## 💡 학습 가이드라인

### 집중해야 할 부분

1. **Map 접근 방법**: `schedule['key']` 패턴 완전히 익히기
2. **setState 타이밍**: 언제 호출해야 화면이 업데이트되는지 체감
3. **Navigator 패턴**: pushNamed vs pop의 데이터 전달 방식 차이점
4. **리스트 순회**: itemBuilder에서 index 활용하는 다양한 방법

### 신경 쓰지 않아도 되는 부분

- UI 디자인이나 색상 조합
- 에러 메시지나 예외 처리 로직
- 로딩 상태나 애니메이션 효과
- 빈 상태 화면이나 아이콘 꾸미기

### 효율적 학습 방법

1. **빠른 실행**: 각 과제 완료 후 즉시 실행해서 동작 확인
  2. **복사 활용**: 코드 직접 복사해서 빠르게 진행해도 됨
  3. **에러 후순위**: 당장 앱이 안 뜨는 에러가 아니면 나중에 처리
  4. **개념 중심**: 왜 이 코드를 쓰는지 이해하는 것이 목표
- 

## 시간 배분 계획

과제	소요 시간	주요 활동
과제 1	10분	Map 구조 + initState 구현
과제 2	15분	ListView.builder 구현
과제 3	15분	setState + FloatingActionButton
과제 4	15분	Navigator 데이터 전달
총 시간	55분	4가지 핵심 개념 체험

## 학습 성과 평가

### Day 3 완료 후 습득할 수 있는 지식

#### 1. Map이 왜 유용한가?

→ 키-값 구조로 서로 다른 타입의 데이터를 하나로 묶어서 체계적 관리 가능

#### 2. setState는 언제 쓰는가?

→ 데이터 변경 후 화면에 새로운 내용을 반영하고 싶을 때

#### 3. ListView.builder는 왜 쓰는가?

→ 동적 개수의 리스트를 효율적으로 메모리 관리하며 표시할 때

#### 4. Navigator arguments는 어떻게 쓰는가?

→ 화면 간 복잡한 데이터를 안전하게 주고받을 때

### 다음 단계 준비 상태

Day 3 완료 후에는 다음과 같은 준비가 완료됩니다:

- **Day 4:** 메모리 데이터를 SharedPreferences로 영구 저장
- **실무 역량:** Map 구조 데이터의 JSON 변환 및 비동기 처리
- **종합 이해:** StatefulWidget의 전체 생명주기 체득

## 참고 사항

### 프로젝트 위치

- 파일 경로: `lib/screen/home_screen.dart`
- 수정 대상: `_HomeScreenState` 클래스
- 연관 파일: `schedule_add.dart`, `schedule_content.dart`

### 버전 정보

- **Flutter 버전:** 호환성 이슈 없음 (기본 위젯만 사용)
- **패키지 의존성:** 추가 패키지 불필요

- 플랫폼: Android/iOS 공통 동작


## 문제 해결

학습 중 문제가 발생하면:

1. 콘솔 에러 메시지 확인
2. 각 과제별 완성 후 테스트 진행
3. 기본 구조 확인 (Scaffold, StatefulWidget 등)
4. 필요시 도움 요청 SOS

---

 **문서 작성자:** Flutter 학습 지원 시스템

 **최종 수정:** 2025년 7월 23일

 **목적:** 효율적 복습을 통한 핵심 개념 체득