**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN TỬ VIỄN THÔNG**

**BÁO CÁO**

**LẬP TRÌNH ĐA NỀN TẢNG**

Chủ đề: **SHAREDPREFERENCES TRONG FLUTTER**

**LƯU TRỮ DỮ LIỆU ĐƠN GIẢN**

**Sinh viên thực hiện:**

**01. Nguyễn Thị Uyên Phương Lớp: 22KTMT1**  **MSSV:** 106220231

**02. Lê Thị Hải Yến**  **Lớp: 22KTMT2** **MSSV:** 106220278

**Người hướng dẫn:**

**TS. Nguyễn Duy Nhật Viễn**

**Đà Nẵng, 2025.**

**THUYẾT MINH**

**BÁO CÁO**

**LẬP TRÌNH ĐA NỀN TẢNG**

Chủ đề: **SHAREDPREFERENCES TRONG FLUTTER – LƯU TRỮ DỮ LIỆU ĐƠN GIẢN**

BẢNG PHÂN CÔNG CÔNG VIỆC TRONG NHÓM

|  |  |  |  |
| --- | --- | --- | --- |
| STT | HỌ VÀ TÊN | NHIỆM VỤ | KHỐI LƯỢNG |
| 01 | Nguyễn Thị Uyên Phương |  | 50% |
| 02 | Lê Thị Hải Yến |  | 50% |

 Link code github:

**MỤC LỤC**

[1. Giới thiệu 4](#_Toc212120185)

[1.1. Khái niệm: 4](#_Toc212120186)

[1.2. Mục đích sử dụng: 4](#_Toc212120187)

[2. Nguyên lý hoạt động của SharedPreferences: 4](#_Toc212120188)

[2.1. Cơ chế: 4](#_Toc212120189)

[2.2. Quy trình hoạt động: 5](#_Toc212120190)

[2.3. Đặc điểm: 6](#_Toc212120191)

[3. Demo ứng dụng lưu trữ cài đặt dark mode, font size 6](#_Toc212120192)

[3.1. Chuẩn bị: 6](#_Toc212120193)

[3.2. Lưu dữ liệu: 6](#_Toc212120194)

[3.3. Đọc dữ liệu: 7](#_Toc212120195)

[3.4. Xóa dữ liệu: 7](#_Toc212120196)

[3.5. Ứng dụng trong giao diện: 7](#_Toc212120197)

[3.5. Kết quả: 7](#_Toc212120198)

[4. So sánh với các phương thức lưu trữ khác 7](#_Toc212120199)

[5. Xử lý lỗi và Best Practices 8](#_Toc212120200)

[5.1. Các lỗi thường gặp: 8](#_Toc212120201)

[5.2. Best Practices: 8](#_Toc212120202)

[6. Kết luận 9](#_Toc212120203)

[TÀI LIỆU THAM KHẢO 9](#_Toc212120204)

**NỘI DUNG**

**1. Giới thiệu**

**1.1. Khái niệm:**

**- SharedPreferences là plugin cung cấp một cách đơn giản, đa nền tảng để lưu trữ dữ liệu key – value đơn giản trong Flutter**

**- Dữ liệu được lưu cục bộ (local storage), vẫn tồn tại sau khi người dùng tắt ứng dụng**

**- Thường dùng cho các dữ liệu nhỏ: cài đặt người dùng, token tạm thời, tùy chọn hiển thị, trạng thái giao diện,...**

**1.2. Mục đích sử dụng:**

**- Ghi nhớ các cài đặt và tùy chọn của người dùng**

* **Đây là mục đích chính. Thay vì bắt người dùng thiết lập lại mỗi khi mở app thì lưu nó lại.**
* **Ví dụ: chế độ sáng/tối, ngôn ngữ người dùng chọn, cỡ chữ, bật/tắt thông báo**

**- Lưu cờ trạng thái (State Flags) của ứng dụng:**

* **Dùng để kiểm soát các hành vi chỉ xảy ra một lần hoặc các trạng thái đơn giản**
* **Ví dụ: Lưu cờ *bool isFirstTime* (để hiển thị màn hình hướng dẫn/onboarding), hoặc *didAgreeToTerms* (đã đồng ý điều khoản)**

**- Hỗ trợ logic điều hướng khi khởi động (Startup Logic):**

* **Đây là mục đích sử dụng phổ biến. Khi khởi động ứng dụng, đọc shared\_preferences ngay lập tức để quyết định logic**
* **Ví dụ: Kiểm tra xem có *userId* hoặc *token* được lưu không**
* **Có 🡪 Chuyển đến màn hình HomePage**
* **Không 🡪 Chuyển đến màn hình LoginPage**

**- Giải pháp lưu trữ nhẹ, nhanh, không cần CSDL**

**2. Nguyên lý hoạt động của SharedPreferences:**

**2.1. Cơ chế:**

**- Dữ liệu được lưu theo dạng key – value**

***Ví dụ:***

***“isDarkMode” 🡪 true***

***“fontSize” 🡪 18.0***

***“language” 🡪 “vi”***

**- Flutter quản lý thông qua package shared\_preferences, tự động ánh xạ (tự động dịch các lệnh ra cơ chế lưu trữ gốc tương ứng của từng hệ điều hành mà ứng dụng đang chạy):**

**+ Android 🡪 lưu vào file XML trong SharedPreferences system**

**+ iOS 🡪 lưu trong NSUserDefaults**

**+ Windows 🡪 lưu vào file JSON**

**+ Web 🡪 lưu vào LocalStorage của trình duyệt**

**2.2. Quy trình hoạt động:**

1. **Khởi tạo SharedPreferences: Khi ứng dụng khởi động, Flutter sẽ khởi tạo một instance của SharedPreferences:**

**final pref = await SharedPreferences.getInstance();**

**Khi đó, framework sẽ:**

* **Kiểm tra xem trên bộ nhớ có file cấu hình lưu dữ liệu trước đó không**
* **Nếu có, đọc file này và nạp toàn bộ key – value vào RAM (bộ nhớ tạm)**
* **Nếu chưa có, tạo file mới (thường là shared\_prefs.xml trên Android hoặc UserDefaults trên iOS)**

1. **Lưu dữ liệu (Write): Gọi phương thức set:**

**await prefs.setString(‘username’, ‘Yen’);**

**Quy trình là:**

* **Dữ liệu ‘username’ : ‘Yen’ được ghi tạm thời vào bộ nhớ RAM (cache)**
* **Sau đó được ghi bất đồng bộ xuống file XML (Android) hoặc NSUserDefaults (iOS)**
* **Hệ thống đảm bảo dữ liệu tồn tại cả khi ứng dụng bị tắt**

1. **Đọc dữ liệu (Read): Đọc:**

**final name = prefs.getString(‘username’);**

* **Flutter không cần đọc lại từ file mỗi lần mà lấy trực tiếp từ vùng nhớ cache đã nạp ở bước 1**
* **Giúp tốc độ truy xuất nhanh hơn rất nhiều so với truy cập ổ đĩa**

1. **Cập nhật dữ liệu (Update):**

**Nếu gọi prefs.setInt(‘count’, 10); rồi sau đó prefs.setInt(‘count’, 11);**

* **SharedPreferences sẽ ghi đè giá trị cũ bằng giá trị mới**
* **Dữ liệu trong file được cập nhật tương ứng khi lưu lại**

1. **Xóa hoặc xóa toàn bộ (Remove/Clear):**

**prefs.remove(‘username’); //xóa 1 key cụ thể**

**prefs.clear(); //xóa toàn bộ dữ liệu**

**🡪 Hệ thống sẽ xóa key đó trong cache RAM và đồng thời xóa trong file**

**2.3. Đặc điểm:**

**- Hỗ trợ các kiểu dữ liệu nguyên thủy: int, double, bool, String, List<String>**

**- Không lưu được các đối tượng: List<int>, Map**

**- Dung lượng nhỏ, tốc độ truy xuất nhanh**

**- Không phù hợp với dữ liệu lớn hoặc nhạy cảm (như: mật khẩu, API token dài hạn, thông tin thẻ tín dụng,...) 🡪 Để lưu các dữ liệu nhạy cảm này thì dùng plugin flutter\_secure\_storage (sử dụng Keychain của iOS và KeyStore của Android)**

**3. Demo ứng dụng lưu trữ cài đặt dark mode, font size**

**3.1. Chuẩn bị:**

**Thêm dependency trong file pubspec.yaml:**

***dependencies:***

***shared\_preferences: ^2.2.2***

**3.2. Lưu dữ liệu:**

***final prefs = await SharedPreferences.getInstance();***

***await prefs.setBool('isDarkMode', false);***

***await prefs.setDouble('fontSize', 18.0);***

**3.3. Đọc dữ liệu:**

***final prefs = await SharedPreferences.getInstance();***

***bool \_isDarMode = prefs.getBool('isDarkMode') ?? false;***

***double \_fontSize = prefs.getDouble('fontSize') ?? 18.0;***

**3.4. Xóa dữ liệu:**

***final prefs = await SharePreferences.getInsstance();***

***await prefs.clear();***

***bool \_isDarkMode = false;***

***ddouble\_fontSize = 18.0;***

**3.5. Kết quả:**

**- Khi người dùng thay đổi chế độ sáng/tối, cỡ chữ 🡪 Dữ liệu được lưu lại**

**- Khi mở lại ứng dụng, trạng thái vẫn được giữ nguyên**

**4. So sánh với các phương thức lưu trữ khác**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phương thức** | **Dạng dữ liệu** | **Quy mô** | **Bảo mật** | **Tình huống sử dụng** |
| **SharedPreferences** | Key-value nhỏ | Nhỏ | Thấp | Cài đặt người dùng, token tạm |
| **File Storage** | JSON, text, image | Trung bình | Trung bình | Cache dữ liệu, cấu hình |
| **SQLite / Drift** | Dữ liệu có cấu trúc | Lớn | Trung bình | Dữ liệu phức tạp, nhiều bảng |
| **Secure Storage** | Key-value | Nhỏ | Cao | Mật khẩu, access token bảo mật |

**5. Xử lý lỗi và Best Practices**

**5.1. Các lỗi thường gặp:**

|  |  |  |
| --- | --- | --- |
| **Lỗi** | **Nguyên nhân** | **Cách khắc phục** |
| Dữ liệu chưa lưu | Quên dùng await | Thêm await khi gọi set |
| Trả về null | Key chưa tồn tại | Sử dụng giá trị mặc định (??) |
| UI không cập nhật | Chưa gọi setState() | Gọi setState() sau khi thay đổi giá trị |

**5.2. Best Practices:**

- Khởi tạo **SharedPreferences** trong **initState()** hoặc **FutureBuilder**

- Tạo lớp riêng để quản lý dữ liệu (Ví dụ: **SettingsService**)

- Dùng Provider, Bloc hoặc Riverpod để quản lý trạng thái khi thay đổi

- Không lưu trữ dữ liệu quan trọng (mật khẩu, thẻ,...)

- Có thể gọi **prefs.clear()** khi cần reset toàn bộ dữ liệu

*Ví dụ:*

*class SettingsService {*

*static Future<void> setDarkMode(bool value) async {*

*final prefs = await SharedPreferences.getInstance();*

*await prefs.setBool('isDarkMode', value);*

*}*

*static Future<bool> getDarkMode() async {*

*final prefs = await SharedPreferences.getInstance();*

*return prefs.getBool('isDarkMode') ?? false;*

*}*

*static Future<void> setFontSize(double value) async {*

*final prefs = await SharedPreferences.getInstance();*

*await prefs.setDouble('fontSize', value);*

*}*

*static Future<double> getFontSize() async {*

*final prefs = await SharedPreferences.getInstance();*

*return prefs.getDouble('fontSize') ?? 18.0;*

*}*

*}*

**6. Kết luận**

- SharedPrefernces là giải pháp đơn giản và hiệu quả để lưu trữ dữ liệu nhẹ trong Flutter

- Phù hợp cho cài đặt người dùng, token tạm, trạng thái ứng dụng

- Không nên dùng cho dữ liệu lớn hoặc bảo mật cao

- Khi kết hợp với Provider hoặc Bloc, giúp ứng dụng phản ứng linh hoạt với thay đổi cài đặt

**TÀI LIỆU THAM KHẢO**

[1].[*https://pub.dev/packages/shared\_preferences*](https://pub.dev/packages/shared_preferences)

[2]. *https://docs.flutter.dev/cookbook/persistence/key-value*

[3]. *https://developer.android.com/training/data-storage/shared-preferences?hl=vi*