**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN TỬ - VIỄN THÔNG**

**BÁO CÁO**

**LẬP TRÌNH ĐA NỀN TẢNG**

**ĐỀ TÀI:**

**JSON SERIALIZATION VÀ MODEL CLASSES**

Giảng viên hướng dẫn: **TS.** **NGUYỄN DUY NHẬT VIỄN**

Sinh viên thực hiện: **TRẦN ĐẠI TÍN - 22KTMT2 - 106220272**

**NGUYỄN XUÂN TRƯỜNG - 22KTMT2 - 106220275**

**Đà Nẵng, 10/2025**

**Phân công công việc**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | HỌ VÀ TÊN | NHIỆM VỤ | KHỐI LƯỢNG |
| 01 | TRẦN ĐẠI TÍN | Tìm hiểu về JSON và serialization, làm slide và kiểm tra nội dung | 50% |
| 02 | NGUYỄN XUÂN TRƯỜNG | Tìm hiểu về JSON và serialization, làm báo cáo và kiểm tra nội dung | 50% |

# Mục lục

[Mục lục 3](#_Toc212466946)

[**1. Tổng quan về JSON** 4](#_Toc212466947)

[1.1. JSON là gì? 4](#_Toc212466948)

[1.2. JSON dạng Array 4](#_Toc212466949)

[1.3. JSON dạng Nested Object 4](#_Toc212466950)

[**2. Làm việc với JSON trong Flutter** 5](#_Toc212466951)

[2.1. Xử lý JSON 5](#_Toc212466952)

[2.2. JSON Serialization 5](#_Toc212466953)

[2.2.1. Manual Serialization (thủ công) 5](#_Toc212466954)

[2.2.1.1. Serialization JSON inline 5](#_Toc212466955)

[2.2.1.2. Serialization JSON inside model classes 6](#_Toc212466956)

[2.2.1.3. Xử lý arrays 8](#_Toc212466957)

[2.2.1.4. Xử lý Nested Object 11](#_Toc212466958)

[2.2.2. Automated Serialization (using code generation) 13](#_Toc212466959)

[**3.Best practices cho complex data structures** 16](#_Toc212466960)

# **1. Tổng quan về JSON**

## 1.1. JSON là gì?

- JSON (JavaScript Object Notation) là định dạng trao đổi dữ liệu giữa client và server.

- Dữ liệu nhẹ, dễ đọc, tương thích với hầu hết các ngôn ngữ lập trình,

- Có cấu trúc key–value, Mỗi key phải nằm trong dấu nháy kép ("") và value có thể là một string, number, boolean, object,..

- Cấu trúc JSON đơn giản:

• Object: Được biểu diễn bằng dấu ngoặc nhọn {}. Các cặp key-value nằm bên trong dấu ngoặc nhọn.

{

"id": 101,

"name": "Nguyen Van A",

"gpa": 3.95

}

## 1.2. JSON dạng Array

Array là mảng được biểu diễn bằng dấu ngoặc vuông. Một mảng có thể chứa nhiều giá trị hoặc đối tượng.

Ví dụ: Một Array gồm nhiều đối tượng JSON

A screenshot of a computer code

AI-generated content may be incorrect.

## 1.3. JSON dạng Nested Object

Là đối tượng lồng bên trong các đối tượng khác.

Ví dụ về Nested Object:

A screenshot of a computer code

AI-generated content may be incorrect.

# **2. Làm việc với JSON trong Flutter**

## 2.1. Xử lý JSON

* Trong Flutter, việc xử lý JSON rất quan trọng để truyền và nhận dữ liệu từ các dịch vụ web.
* Flutter cần chuyển đổi JSON ↔ Dart Objects đảm bảo code an toàn, dễ bảo trì và mở rộng.
* Serialization: Chuyển đối tượng Dart thành JSON
* Deserialization: Chuyển JSON thành đối tượng Dart

## 2.2. JSON Serialization

Flutter có 2 cách chính để xử lý JSON serialization: thủ công và tự động

### **2.2.1. Manual Serialization (thủ công)**

Flutter hỗ trợ thư viện dart:convert được tích hợp sẵn, trong đó bao gồm bộ mã hóa (encoder) và bộ giải mã (decoder) JSON rất dễ sử dụng.

Thực hiện Serialization thủ công có 2 cách:

* Serialization JSON inline
* Serialization JSON inside model classes

#### **2.2.1.1. Serialization JSON inline**

Phù hợp cho các bài toán nhỏ, không thích hợp cho các dự án lớn với nhiều dữ liệu phức tạp.

Sử dụng package dart:convert để giải mã JSON bằng hàm jsonDecode(), chuỗi JSON được truyền vào đối số của hàm

A screenshot of a computer code

AI-generated content may be incorrect.



Ở ví dụ trên ta giả sử có một chuỗi dữ liệu JSON. Ta gọi hàm jsonDecode() để giải mã chuỗi JSON thành một cấu trúc dữ liệu của Dart, cụ thể là một Map<String, dynamic>

Dùng hàm jsonEncode() để mã hóa Map vừa nhận được thành một chuỗi JSON mới được lưu trong biến updatedJsonString để lưu hoặc gửi lại server.

Hàm jsonDecode() trả về kiểu dynamic, nghĩa là không biết kiểu dữ liệu của các giá trị cho đến khi chương trình chạy (runtime). Với cách làm này, ta sẽ mất đi hầu hết các tính năng của ngôn ngữ có kiểu tĩnh, như:

* An toàn kiểu dữ liệu
* Phát hiện lỗi khi biên dịch

Điều này khiến code dễ bị lỗi hơn.

#### **2.2.1.2. Serialization JSON inside model classes**

Thích hợp với các dự án vừa và lớn, cải thiện khả năng đọc mã, tránh lỗi khi xử lý dữ liệu phức tạp.

**Bước 1:** Tạo model class - Class Book

A screenshot of a computer program

AI-generated content may be incorrect.

**Constructor Book.fromJson** – dùng để **tạo một đối tượng** Book **mới từ cấu trúc dạng Map** (tức là từ dữ liệu JSON).

**Phương thức toJson()**: Chuyển đổi **đối tượng** Book **thành cấu trúc Map**, có thể dễ dàng mã hóa lại thành JSON (Serialization).

Phương thức**fromJson():** Sử dụng factory constructor để tạo một đối tượng Book từ dữ liệu JSON (Deserialization).

Với cách làm này sẽ có những lợi ích sau:

* An toàn kiểu dữ liệu
* Phát hiện lỗi ngay khi biên dịch

**Bước 2:** Chuyển đổi Dart Object ⬄ JSON

A screenshot of a computer program

AI-generated content may be incorrect.



Trong hàm main, chúng ta thấy rõ quy trình:

* Từ dữ liệu JSON giả lập lưu trong biến jsonData. sử dụng fromJson() để tạo đối tượng myBook
* Chỉnh sửa giá trị biến rating và in thông tin sách ra
* Sau khi làm việc với đối tượng, sử dụng toJson() để chuyển ngược lại thành Map, lúc nãy có thể lưu trữ hoặc gửi lại server.

#### **2.2.1.3. Xử lý arrays**

Ta có một array gồm các object như sau:

A screenshot of a computer code

AI-generated content may be incorrect.

**Tạo model class Product**

A screenshot of a computer program

AI-generated content may be incorrect.

**Hàm main.dart**

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer code

AI-generated content may be incorrect.

A number and numbers on a white background

AI-generated content may be incorrect.

Bước 1: Nhận dữ liệu từ API

* Giả lập việc nhận JSON string từ server thông qua class ProductData

Bước 2: Deserialize - Chuyển đổi JSON → Dart Objects

* Sử dụng jsonDecode() để chuyển JSON string thành cấu trúc List
* Dùng vòng lặp và factory constructor fromJson() để chuyển mỗi item JSON thành đối tượng Product thực thụ
* => Flutter có thể làm việc trực tiếp với các object này

Bước 3: Thao tác và chỉnh sửa dữ liệu

* Cập nhật giá sản phẩm iPhone 15
* Thay đổi trạng thái tồn kho của AirPods Pro từ "hết hàng" sang "còn hàng"
* => Việc làm việc với Dart objects dễ dàng và an toàn hơn so với thao tác trực tiếp trên JSON

Bước 4: Serialize - Chuyển đổi Dart Objects → JSON

* Sử dụng phương thức toJson() để chuyển mỗi Product object thành Map
* Dùng JsonEncoder.withIndent() để mã hóa thành JSON string có định dạng đẹp, dễ đọc và in thử ra console

#### **2.2.1.4. Xử lý Nested Object**

Ta có một cấu trúc Nested Object như sau:

A screenshot of a computer screen

AI-generated content may be incorrect.

Tạo một class Address gồm các thuộc tính city và street như trong cấu trúc Nested Object đã nêu.

A screenshot of a computer code

AI-generated content may be incorrect.

Tạo một class Student, trong đó tạo một thuộc tính có kiểu Address - thuộc tính address sẽ đóng vai trò là một dart object để lưu trữ địa chỉ của một đối tượng Student. Trong mỗi đối tượng Student sẽ được lồng đối tượng Address.

A screenshot of a computer program

AI-generated content may be incorrect.

Trong hàm main.dart, việc xử lý Nested object sẽ được thực hiện như code bên dưới.

A screenshot of a computer program

AI-generated content may be incorrect.

Kết quả hiện ra trên terminal:

*A close up of text

AI-generated content may be incorrect. A close up of text

AI-generated content may be incorrect.*

A computer code with black text

AI-generated content may be incorrect.

**Bước 1: Nhận dữ liệu từ server**

* Dữ liệu JSON là một object sinh viên, trong đó trường address là một object lồng bên trong.

**Bước 2: Deserialize**

* jsonDecode(): Chuyển JSON string thành cấu trúc Map<String, dynamic>
* Address.fromJson(): Tạo object Address từ dữ liệu lồng
* Student.fromJson(): Tạo object Student, trong đó thuộc tính address được ánh xạ sang object Address.

**Bước 3: Thao tác với dữ liệu**

* Truy cập và cập nhật thông tin: students[0].address.street = "Ngô Sĩ Liên"

**Bước 4: Serialize**

* student.toJson(): Chuyển toàn bộ object Student thành JSON
* address.toJson(): Được gọi để serialize phần địa chỉ.
* jsonEncode(): Tạo JSON string để lưu trữ hoặc gửi lại server

### **2.2.2.** **Automated Serialization (using code generation)**

- Trong thực tế, ta sẽ phải làm việc với nhiều JSON phức tạp gồm nhiều object lồng nhau.

- Việc ta viết thủ công toJson() và fromJson() thủ công trở nên tốn thời gian (viết nhiều code lặp lại), dễ sai sót và khó bảo trì.

- Flutter hỗ trợ công cụ hỗ trợ tự động sinh mã (code generation) cho quá trình serialization.

- Mặc dù có nhiều thư viện khác có thể sử dụng, báo cáo này sẽ dùng json\_serializable là một thư viện phổ biến giúp tự động tạo mã chuyển đổi giữa JSON và object Dart.

Cơ chế hoạt động

1. json\_annotation - Cung cấp annotations (chú thích) để đánh dấu class cần generate code

2. build\_runner - Công cụ thực thi quá trình code generation, tự động tạo ra phần mã toJson() và fromJson() dựa trên các annotation đã khai báo.

**Bước 1: Cài đặt json\_serializable trong project**

Để thêm json\_serializable vào dự án của bạn, bạn cần:

* 1 dependency thông thường
* 2 dev dependencies.

Chạy lệnh sau trong terminal của dự án Flutter:

**flutter pub add json\_annotation dev:build\_runner dev:json\_serializable**

Lệnh này sẽ tự động thêm các gói cần thiết vào file pubspec.yaml.

Hoặc thêm thủ công bằng cách vào file pubspec.yaml và thêm dòng sau.

Các gói cần thiết:

A screenshot of a computer program

AI-generated content may be incorrect.

Sau đó chạy lệnh **flutter pub get** để tải package.

**Bước 2: Tạo model classes với annotation**

Tạo các class với **@JsonSerializable()** annotation:

* part 'filename.g.dart': Khai báo file sẽ được generate
* @JsonSerializable(): Đánh dấu class cần generate code
* Các method \_$ClassFromJson và \_$ClassToJson sẽ được tạo tự động

Ta thưc hiện tạo một class Student, part 'student.g.dart'; để tạo đường dẫn file sẽ tạo code xử lý JSON.

Sau khi tạo xong class Student, chạy lệnh để tạo file student.g.dart

Sau khi tạo model classes, chạy lệnh sau: **flutter pub run build\_runner build**

Sau khi chạy lệnh trên thì **Build runner sẽ:**

1. Quét tất cả class có @JsonSerializable()
2. Tự động generate file \*.g.dart với đầy đủ serialization code

Code class Student được nêu ở hình dưới:

A screenshot of a computer program

AI-generated content may be incorrect.

File student.g.dart sẽ được tự động tạo như dưới đây.

A screenshot of a computer program

AI-generated content may be incorrect.

So sánh hai phương pháp thủ công và tự động

**MANUAL SERIALIZATION:**

* Ưu: Dễ hiểu, không cần setup phức tạp
* Nhược: Tốn công, khó bảo trì với project lớn

**AUTOMATED SERIALIZATION:**

* Ưu: Hiệu quả, ít lỗi, dễ mở rộng
* Nhược: Cần setup ban đầu, phụ thuộc tool

# **3.Best practices cho complex data structures**

- Tạo model class riêng cho từng object, không nên để tất cả logic vào một file lớn.

- Dùng thư viện json\_serialization để tự động tạo code giúp code gọn và tránh lỗi.

- Luôn chỉ định kiểu dữ liệu rõ ràng, tránh sử dụng dynamic hoặc var trong model vì dễ gây lỗi runtime.

- Xử lý dữ liệu null để tránh bị crash chương trình .

- Khi dữ liệu JSON có cấu trúc phức tạp nhiều cấp lồng nhau, nhiều kiểu dữ liệu khác nhau, hoặc có thể thiếu field việc serialize/deserialize có thể không còn thuận lợi

- Flutter cung cấp **@JsonKey** để tùy chỉnh từng trường (field) trong model class.

@JsonKey được dùng trước một thuộc tính (field) trong class model có @JsonSerializable().

|  |  |
| --- | --- |
| **Thuộc tính** | **Chức năng** |
| name | Đổi tên key trong JSON |
| defaultValue | Dùng giá trị mặc định nếu JSON thiếu field |
| required | Báo lỗi nếu JSON không có field này |
| nullable | Cho phép field được null |
| ignore | Không serialize field này |
| includeIfNull | Kiểm soát có nên include field null không |
| fromJson / toJson | Chỉ định hàm chuyển đổi các giá trị phức tạp |

A screenshot of a computer program

AI-generated content may be incorrect.

Sử dụng @Jsonkey:**name** để ánh xạ (map) một thuộc tính trong class tới tên của key khác trong dữ liệu JSON.

Với name: 'full\_name', gói json\_serializable sẽ ánh xạ key JSON có tên fullName vào trường name của class Admin.

A screenshot of a computer code

AI-generated content may be incorrect.

Cung cấp giá trị mặc định nếu key bị thiếu dữ liệu

Nếu Json data không bao gồm tuổi (age) thì mặc định là 18

A screenshot of a computer code

AI-generated content may be incorrect.

Đôi khi chúng ta cần xử lý kiểu dữ liệu đặc biệt mà thư viện json\_serializable không thể tự động hiểu được

Thuộc tính **fromJson** và **toJson** được dùng để tùy chỉnh cách một trường được chuyển đổi trong quá trình deserialize và serialize, hữu ích khi làm việc với kiểu dữ liệu phức tạp

* Thuộc tính birthDate trong Dart là kiểu DateTime, nhưng trong JSON nó chỉ là chuỗi (String)
* Hàm \_dateTimeFromJson() giúp chuyển chuỗi đó thành DateTime khi đọc dữ liệu từ JSON.
* Hàm \_dateTimeToJson() giúp chuyển DateTime trở lại chuỗi khi ghi dữ liệu ra JSON.

**Tài liệu tham khảo**

[1]. TS. Nguyễn Duy Nhật Viễn, Slide bài giảng lập trình đa nền tảng, Khoa Điện tử - Viễn Thông, Trường Đại học Bách Khoa, Đại học Đà Nẵng.

[2]. Flutter Official Documentations, “JSON serialization and deserialization”.

Link: [docs.flutter.dev/data-and-backend/serialization/json](https://docs.flutter.dev/data-and-backend/serialization/json)

[3]. Ammar Yasser, “Understanding JSON Serialization and Deserialization in Flutter,” Medium Blog [Online].

Link: [understanding-json-serialization-and-deserialization-in-flutter](https://medium.com/@ImAmmarYasser/understanding-json-serialization-and-deserialization-in-flutter-eb72d5c54ddb)