

1. Nullable là gì?

Trong quá trình lập trình, có nhiều trường hợp dữ liệu chưa có, bị thiếu hoặc không xác định. Khi đó, giá trị của biến có thể là null.

Trong Kotlin, để hạn chế lỗi **NullPointerException** thường gặp ở Java, ngôn ngữ này đưa ra khái niệm Nullable.

Theo mặc định, Kotlin không cho phép biến nhận giá trị null. Nếu lập trình viên muốn một biến có thể nhận null thì phải chỉ định rõ ràng bằng dấu ? sau kiểu dữ liệu.

Ví dụ:

```
var name: String = "Kotlin"
// name = null -> lỗi

var email: String? = "abc@gmail.com"
email = null // hợp lệ
```

Nhờ cơ chế này, Kotlin giúp lập trình viên **nhận biết và kiểm soát null ngay từ lúc viết code**, làm cho chương trình an toàn và ổn định hơn.

2. Khi nào nên và không nên dùng Nullable?

Khi nào nên dùng Nullable

Nullable nên được sử dụng trong những trường hợp dữ liệu:

- Không bắt buộc phải có
- Có thể không tồn tại tại một thời điểm nào đó
- Được lấy từ bên ngoài như:

Dữ liệu người dùng nhập

API

Cơ sở dữ liệu

Ví dụ:

Email, số điện thoại, ảnh đại diện của người dùng... đều có thể **chưa được cung cấp**.

```
var phoneNumber: String?
```

Khi nào không nên dùng Nullable

Không nên dùng Nullable khi dữ liệu:

- Luôn luôn tồn tại
- Bắt buộc phải có để chương trình hoạt động

Ví dụ:

```
var age: Int = 20
```

Việc lạm dụng Nullable sẽ khiến code trở nên **phức tạp, khó đọc và khó bảo trì**.

3. Các cách xử lý null phổ biến trong Kotlin

3.1 Toán tử an toàn ?.

Toán tử ?. giúp truy cập thuộc tính hoặc hàm **chỉ khi biến không null**. Nếu biến là null, kết quả sẽ trả về null thay vì gây lỗi.

```
val length = name?.length
```

3.2 Toán tử Elvis ?:

Toán tử ?: cho phép **gán giá trị mặc định** khi biến null.

```
val length = name?.length ?: 0
```

Cách này thường được dùng khi hiển thị dữ liệu lên giao diện.

3.3 Hàm let

Hàm let cho phép thực hiện một khối lệnh **chỉ khi biến không null**.

```
email?.let {  
    println("Email: $it")  
}
```

Cách này giúp code rõ ràng và an toàn hơn.

3.4 Toán tử !!

Toán tử !! dùng để **ép Kotlin tin rằng biến không null**. Nếu biến thực sự null, chương trình sẽ bị crash.

```
val length = name!!.length
```

Toán tử này **không được khuyến khích sử dụng** vì làm mất ý nghĩa của null safety.

4. Dựng một màn hình ví dụ sử dụng Nullable

Mô tả màn hình

Xây dựng màn hình **Thông tin người dùng** gồm:

Họ tên (bắt buộc)

Email (có thể không có)

Nếu email chưa được cập nhật, màn hình sẽ hiển thị thông báo phù hợp thay vì để trống hoặc gây lỗi

Kết quả đạt được

Ứng dụng **không bị lỗi khi email null**

Giao diện hiển thị rõ ràng, thân thiện với người dùng

Thể hiện đúng cách sử dụng Nullable trong Kotlin

5. Kết luận

Nullable là một tính năng quan trọng trong Kotlin giúp:

- Hạn chế lỗi NullPointerException
- Tăng độ an toàn và ổn định cho chương trình
- Buộc lập trình viên xử lý null một cách rõ ràng

Việc sử dụng hợp lý Nullable kết hợp với các toán tử như `?.`, `?:`, `let` sẽ giúp code **ngắn gọn, dễ hiểu và hiệu quả hơn**.