**Null Safety**

Sejak versi 2.12, Dart menghadirkan fitur Null Safety. Apakah yang dimaksud dengan null safety dan kenapa kita membutuhkannya?

Sebelum membahas tentang null safety lebih jauh, mari kita ulas kembali terkait variabel dan tipe data.

Seperti yang telah kita bahas sebelumnya, variabel digunakan untuk menyimpan nilai. Variabel String digunakan untuk menyimpan teks, num digunakan untuk menyimpan angka, dsb.

1. String name = 'John Doe';
2. int age = 23;

Namun, kenyataannya suatu variabel juga bisa tidak memiliki nilai. Nilai “tidak ada” pada Dart ini dikenal dengan nilai null. Nilai null bisa terjadi karena kita belum menginisialisasi nilai atau memang secara eksplisit menentukan nilai null pada variabel.

1. String favoriteFood = null;

Lalu, kenapa kita mesti peduli dengan nilai null ini? Nilai null bisa menyebabkan suatu program mengalami kesalahan yang dikenal dengan *Null Pointer Exception (NPE)*. Bahkan, developer yang pertama kali mengenalkan NPE menyebutnya sebagai **“the billion dollar mistake”**.

Bayangkan ada teman yang ingin mentraktir makanan favorit Anda, tetapi karena Anda tidak memiliki makanan favorit, maka ia akan bingung. Begitu pula dengan komputer. Jika nilai variabel null atau “tidak ada”, maka komputer juga akan bingung hingga bisa menyebabkan *crash* pada program.

Namun, nilai null tidak sepenuhnya buruk. Akan tetap ada kasus tertentu di mana kita membutuhkan nilai null. Contohnya seperti variabel favoriteFood di atas karena tidak semua orang memiliki makanan favorit.

Null safety adalah jawaban dari kasus di atas.

Untuk menggunakan fitur null safety, kita perlu menggunakan versi Dart 2.12. Anda dapat menentukan versi Dart yang digunakan dengan mengubah berkas **pubspec.yaml** yang terdapat dalam project. Pastikan minimal versi sdk yang digunakan adalah versi 2.12.0.

1. environment:
2. sdk: '**>=2.12.0** <3.0.0'

Dengan null safety, secara default sebuah variabel tidak bisa memiliki nilai null, kecuali kita mendeklarasikannya secara eksplisit.

1. int age = null;               // Compile error
2. String? favoriteFood = null;

Tanda tanya (?) di atas menunjukkan bahwa variabel favoriteFood boleh memiliki nilai null (*nullable*). Sementara variabel age harus memiliki nilai angka dan tidak boleh null (*non-nullable*).

Kode di atas tidak akan bisa dijalankan karena gagal dalam proses kompilasi. Oleh karena itu, null safety sangat berguna untuk membuat kode menjadi lebih aman karena proses yang melibatkan nilai null dapat dibatasi dan ditemukan lebih awal.

Jika kita memiliki fungsi dengan parameter non-nullable, maka argumen nullable akan terdeteksi sebagai error.

1. void main() {
2. String name = 'John Doe';
3. int age = 23;
4. String? favoriteFood = null;
6. buyAMeal(favoriteFood);       // Compile error
7. }
9. void buyAMeal(String favoriteFood) {
10. print('bought a $favoriteFood');
11. }

Untuk mengakses atau menangani variabel null seperti di atas, ada beberapa cara yang bisa kita lakukan. Pertama, ubah parameter agar dapat menerima nilai null lalu lakukan pengecekan nilai null.

1. void buyAMeal(String? favoriteFood) {
2. if (favoriteFood == null) {
3. print('Bought Nasi Goreng');
4. } else {
5. print('Bought $favoriteFood');
6. }
7. }

Cara kedua yang bisa kita lakukan adalah menggunakan bang operator (!). Dengan operator ini kita memberitahu compiler dan memberikan jaminan bahwa variabel tidak akan bernilai null. Namun, ketika variabel ternyata bernilai null, akan tetap memungkinkan terjadi crash. Jadi, gunakan bang operator ini hanya ketika Anda yakin 100% bahwa variabel tersebut tidak akan bernilai null.

1. void main() {
2. String name = 'John Doe';
3. int age = 23;
4. String? favoriteFood = 'Mie Ayam';
6. buyAMeal(favoriteFood!);
7. }
9. void buyAMeal(String favoriteFood) {
10. print('Bought $favoriteFood');
11. }