Dart

Null Safety

Carmelo Villegas Cruz

Null Safety

Es una característica que ayuda a evitar errores comunes relacionados con valores nulos (o Null) en tiempo de ejecución, proporcionando mayor seguridad y estabilidad a las aplicaciones.

Null es equivalente a “no tener valor” o “ausencia de valor”

Before null safety

¿Qué ocurre si un método espera un entero pero recibe un nulo?

🧨🧨💣💣🔥🔥 La aplicación lanza un **runtime error**.

Este tipo de errores que se conocen como: “null reference error” pueden llegar a ser muy complejos de depurar.

Null Safety

Dart distingue entre dos tipos de variables, las que pueden ser null y las que no.

Somos nosotros los programadores los que decidimos cuando una variable puede tomar el valor nulo.

Null Safety

¿Pero qué conseguimos con esta característica del lenguaje?

1. Evita errores por referencias nulas.
2. Mejora la seguridad y estabilidad del código.

Null Safety

Lo volvemos a recordar

Con **Null Safety**, una variable no puede contener un valor null a menos que explícitamente se declare como "nullable".

Es decir, si defines una variable de tipo int, por ejemplo, no puede tomar un valor null a menos que lo señales explícitamente..

Veamos como hacer eso …

Nullable Type

Para que una variable acepte **null**, usaremos el operador ?.

```
int? age; // 'age' puede ser nulo
```

int? indica que la variable puede ser un entero o **null**.

Esto nos obliga a manejar los casos nulos de manera explícita, reduciendo errores inesperados.

Compile-time checks

El compilador de Dart verifica en **tiempo de compilación** si el código está tratando correctamente los valores nulos.

¡Esto significa que puedes detectar posibles errores de null antes de ejecutar el programa!

Null Safety operators

! (assert non-null operator)

Convierte una variable nullable en no nullable **si estás seguro de que no es null en ese contexto**.

```dart
int? age = 18;
int nonNullableAge = age!; // Afirmas que 'age' no es nulo
```

Null Safety operators

?? (operador de asignación nula)

Permite proporcionar un valor por defecto si una variable nullable es null

```dart
int? age;
int finalAge = age ?? 18; // Si 'age' es nulo, finalAge será 18
```

Null Safety operators

?. (operador de llamada segura)

Permite acceder a métodos o propiedades de un objeto solo si no es null.

```dart
String? name;
print(name?.length); // Solo se accede a 'length' si 'name' no es nulo
```

Benefits of Null Safety

Previene errores comunes

Al eliminar las referencias nulas inesperadas, Null Safety reduce la posibilidad de que las aplicaciones fallen en tiempo de ejecución debido a errores de null.

Benefits of Null Safety

Mejor rendimiento

El análisis estático en tiempo de compilación mejora el rendimiento, ya que los desarrolladores pueden evitar errores antes de ejecutar el código.

Benefits of Null Safety

Mayor claridad en el código

El código se vuelve más fácil de entender, ya que el uso de null es explícito y controlado, lo que también facilita el mantenimiento y depuración.

Examples

Supongamos que tenemos un formulario de registro donde un usuario ingresa su nombre y edad. En este caso, la edad es opcional, por lo que podría ser nula (null).

Sin Null Safety, esto podría generar errores en tiempo de ejecución si intentamos acceder a la edad sin verificar si es null.

Examples


```dart
void main() {
  String? name;  // El nombre puede ser nulo
  int? age;      // La edad también puede ser nula

  // Asignamos valores
  name = "Carmelo";
  age = null;    // El usuario no ingresó la edad

  // Verificamos si el nombre no es nulo antes de usarlo
  if (name != null) {
    print("User's name: $name");
  } else {
    print("Name is missing");
  }

  // Usamos el operador ?? para asignar un valor por defecto si la edad es nula
  int finalAge = age ?? 18;  // Si la edad es nula, asignamos 18 como valor por defecto
  print("User's age: $finalAge");
  // Intentamos acceder a la longitud del nombre usando el operador de acceso seguro
  print("Name length: ${name?.length}");
}
```

Examples

Imaginad ahora que estamos gestionando la información de un pedido, donde el cliente puede o no proporcionar un código de cupón y una dirección de entrega. Queremos asegurarnos de que estas variables opcionales se manejen de manera segura usando Null Safety

Examples


```dart
class Pedido {
  String cliente;
  String? codigoCupon;  // Puede ser nulo si el cliente no usa un cupón
  String? direccionEntrega;  // Puede ser nulo si el pedido es para recoger

  Pedido({required this.cliente, this.codigoCupon, this.direccionEntrega});
}
```

Examples


```dart
void main() {
  // Crear un pedido sin código de cupón pero con dirección de entrega
  Pedido pedido = Pedido(cliente: "Juan Pérez", codigoCupon: null, direccionEntrega: "Av. Siempre Viva
742");

  // Imprimir el nombre del cliente
  print("Cliente: ${pedido.cliente}");

  // Usar el operador ?? para proporcionar un valor por defecto si no hay código de cupón
  String codigoCupon = pedido.codigoCupon ?? "No hay cupón aplicado";
  print("Código de Cupón: $codigoCupon");

  // Usar el operador ?. para acceder de forma segura a la dirección de entrega
  int? longitudDireccion = pedido.direccionEntrega?.length;
  print("Longitud de la Dirección: ${longitudDireccion ?? 0}");  // Si es nulo, usa 0 como valor por
defecto
}
```