Dart Package, Generics and Collections in Dart

By: Laams Innovation Lab

Lesson 012 - Contents

1) Creating a Dart Package
2) Sound Null Safety in Dart
3) Generics in Dart
4) Collections and Type Safety
5) Creating Generics and Code Repetition

Creating Dart Package

A minimal Dart **console application** usually has the following structure:

1) **bin** folder: Inside which the binary of the dart program (if console) is written
2) **lib** folder: Inside which all the functionalities are defined
3) **test** folder: inside which all kinds of test for the project are written.
4) **pubspec.yaml** file: Inside which package info, and dependencies are defined

A minimal Dart **package** usually has the following structure:

5) **example** folder: a dart project using the package
6) **lib** folder: Inside which all the functionalities are defined
7) **test** folder: inside which all kinds of test for the project are written.
8) **tool** folder: extra tools for the package.
9) **pubspec.yaml** file: Inside which package info, and dependencies are defined

pubspec.yaml file

name:* Every package needs a name. Should be lower-cased, should not start with numbers, should be separated with _, and be only use English alphabets.

version:* Every package needs a version to be deployed in stores or pub.dev. It has three numbers separated by dots (.).

- It can also optionally have a build (+1, +2, +hotfix.oopsie) or prerelease (-dev.4, -alpha.12, -beta.7, -rc.5) suffix

description:* Info about your package, should be 60-180 characters. Whatever that you want a reader to know about your package.

publish_to:

homepage: Points to the package developer's website. It is optional but it helps the user of the package to know where the package is coming from.

repository: Points to the package git repository.

issue_tracker: Points to the package issue_tracker URL if empty. It will refer to the repository/ issues.

documentation: Points to the documentation URL of the package.

pubspec.yaml file

environment:* determines the SDK constraints for both Dart SDK and Flutter SDK.

dependencies:* List of all the packages that your package needs to use when in publish mode. You can use the packages from a pub.dev, from a repo or a locally developed package.

dev_dependencies:* List of all the packages which your package needs when in development mode.

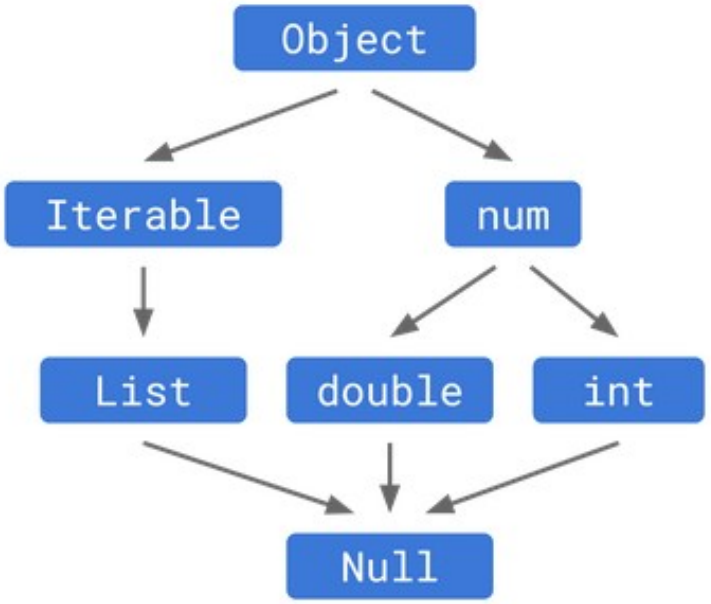
dependency_overrides: List of the packages which specific versioning when different versions are used by your package and the packages that you depend on.

Semantic Versioning

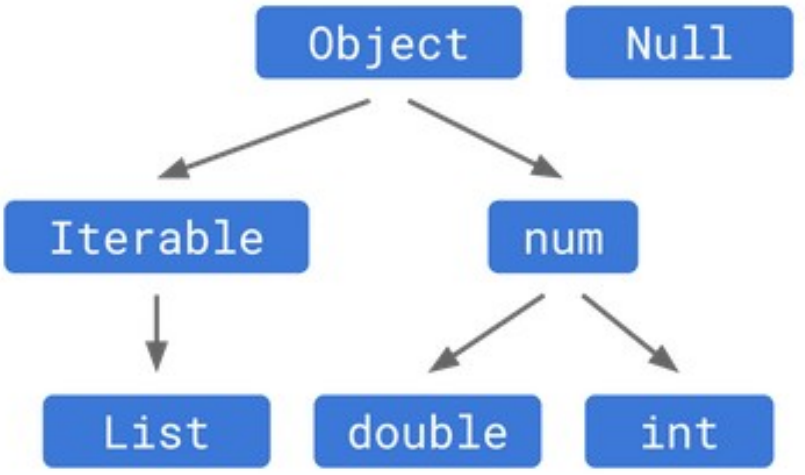
Dart Null Safety Principles

- **Code should be Safe by Default:**
  1) No null errors by default. Null is statically checked at compile time.
  2) You should explicitly determine which values are nullable using (?)
- **Null Safe Code Should be Easy to Write:**
  1) No additional struggle to write null safe code.
  2) Easy to migrate the existing code to null safety.
- **Code Should be Fully Sound:**
  1) if an expression has a static type that does not permit null, then no possible execution of that expression can ever evaluate to null
  2) Confidence when coding

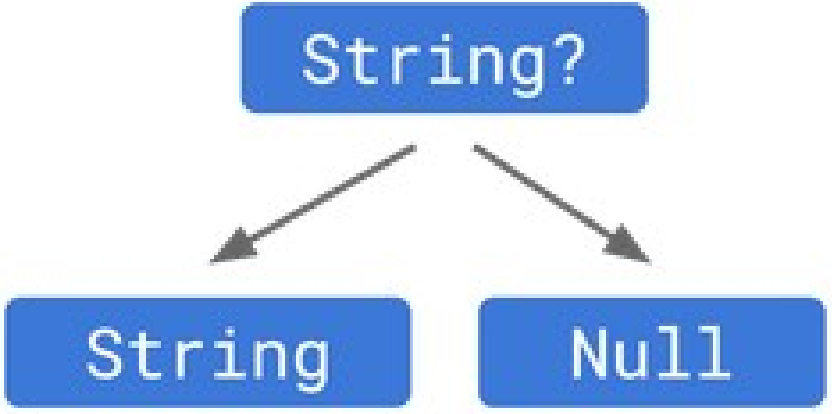
Before



After



? ! late


String?
String
Null

Null Safety under the Hood

Read understanding null safety in Dart documentation online

https://dart.dev/null-safety/understanding-null-safety

Generics

Generics are a feature of **generic programming** – a **style of programming** in which algorithms are written in terms of **types to-be-specified-later** that are then **instantiated** when needed for specific types provided as **parameters**.

Generics help with two aspects of programming, **type safety** and **code re-usability**.

Build-in Generic Collections

- List<E>
  - Literals and constructors
  - Properties and behaviors
- Set<E>
  - Literals and constructors
  - Properties and Behaviors
- Map<K, V>
  - Literals and constructors
  - Properties and behaviors

Creating Generic Classes and Interfaces

- There’s the possibility to define a infinite number of generic types per class, they just need to be separated by commas in the diamond list (< >) with the letters inside.
- You can make a class’ variable or method generic
- Generics also with extends and implements.

Creating Generic Functions and Methods

Dart Generic methods and functions uses **generics** in :

- In a function’s arguments
- In local variables inside a function
- In a function’s return type

Creating Generic Iterables