etraveli group

JHUG.gr - 12/12/2022

Evolving 20+ years codebase

mytrip GO TO GATE flight network

About ETraveli Group

Since ~2000

Flight-centric Online Travel Agency (OTA)

35+ site languages

75+ countries

Over 2000 employees

etraveli group

Our booking platform: IBE

mytrip GO TO GATE flight network

Our Booking Platform Java Code Base 1/2

Monorepo produces 7 tomcat (war) applications

- Initial commit (cvs2svn) 13/10/2005 => original initial commit is legend now
- > 1.9 M lines of code (Java 17)
- 260 unique contributors across all years >150 active
- > 17 major external provider systems
- > 20 internal services (i.e pricing, payments etc)
- 19 minutes mean time in CI
- 4 releases per week
- 5 minutes to import in Intellij

Our major code evolving breakthroughs

Transitioning from homegrown Service Locator to CDI

- Benefits and challenges

Ongoing code restructuring / modularization initiative

- Introducing Trinity Modules

Gradle build files revamp

- Reducing scanning and indexing

Kicking off long - term architecture roadmap

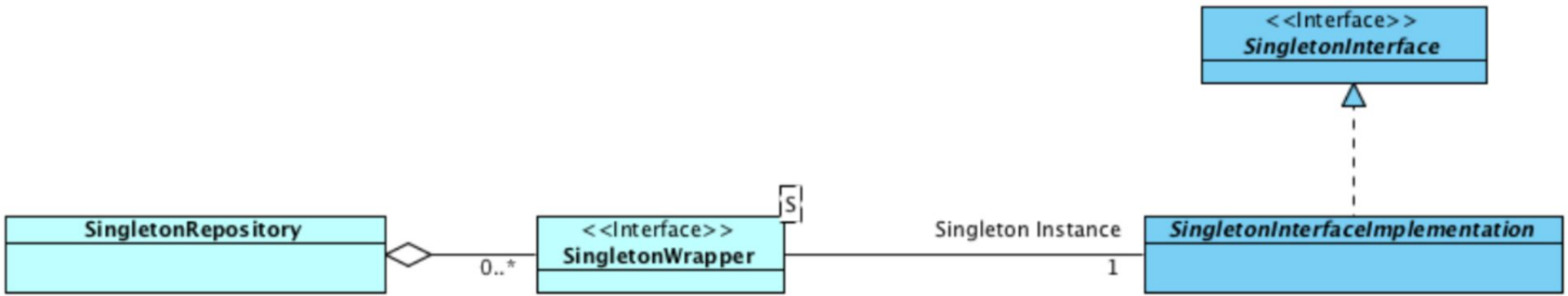
- Were we are heading?

From service locator to CDI

How we used to resolve dependencies...

Service locator we used was like this:

```
Singletons.get(CoreData.class).getSite(contentOwner.getOwnerObjectId())
```



Migration from Service Locator to CDI

2019 Adoption of weld CDI implementation

Singletons.get is deprecated & currently uses cdi behind the scenes

New code uses constructor based injection

Improved testability of code base

Enabled modularization based in the trinity model (see next section)

Reduced build times

Challenges of adopting CDI

It was (and still is) hard to convince everyone to use constructor based injection

Requires constant vigilance about using the correct scope for CDI beans... And a lot of training as well.

We had to write a lot of code in order to make our tests CDI aware

etraveli group

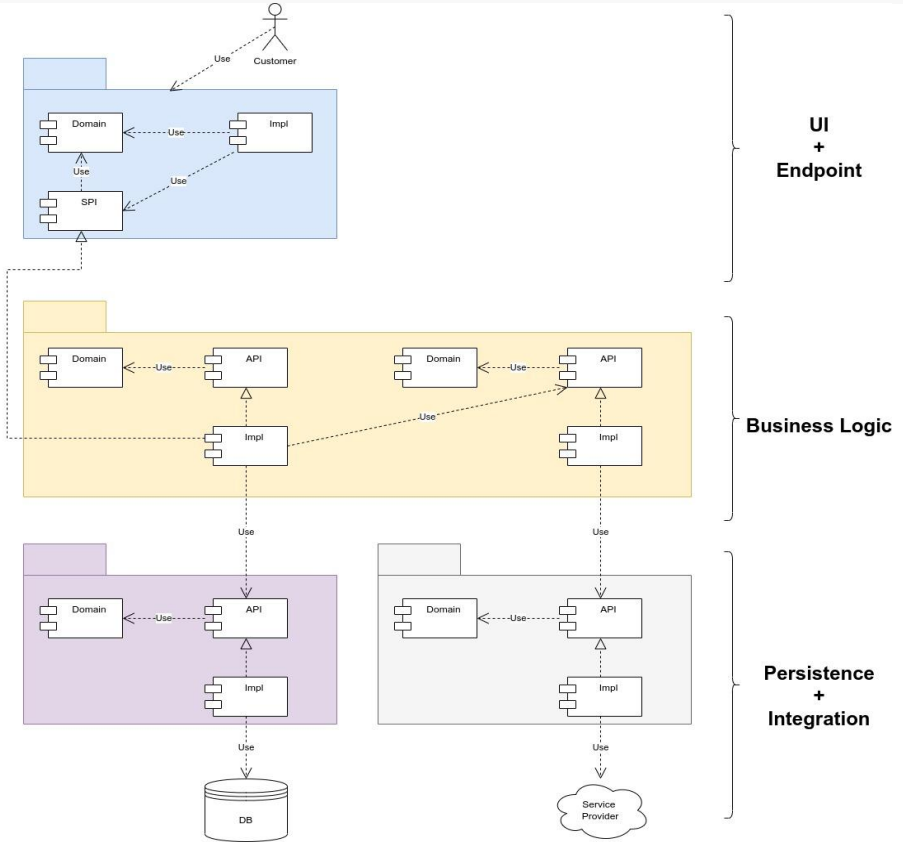
Restructuring code: Introducing Trinity Modules

mytrip GO TO GATE flight network

Modularization of the codebase

What to do

- Refactor the all of the codebase into api, impl and domain modules
- Separate business logic from technical concerns, like persistence, integrations, user interfaces, caches and “deployment”
- Apply build rules to govern modularization adherence
- Refactor tests and test frameworks to adhere to modularisation

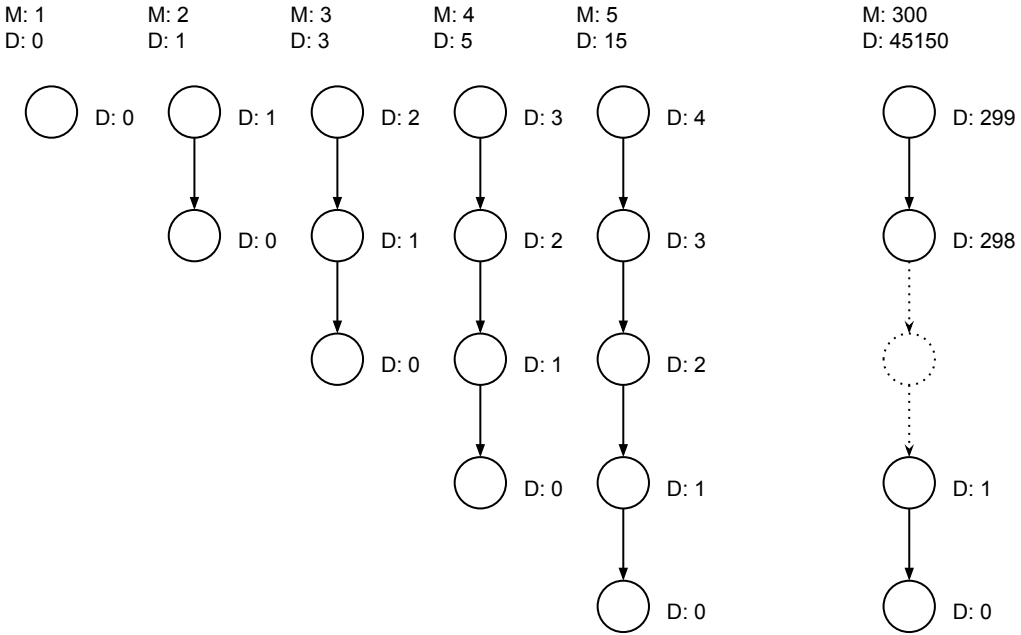


Modularization of the codebase

Why to do it

- Reduce dependencies in the codebase
- Shorten build and test times
- Reduce resource usage
- Allow breakout of libraries and services
- Improve ability to scale out the organization
- Shorten and simplify onboarding
- Improve testability
- Improve performance of IDEs and build environments
- Implement distributed web sessions - improving time and quality of deploys
- Reduce time to market for future business features

The math of dependencies

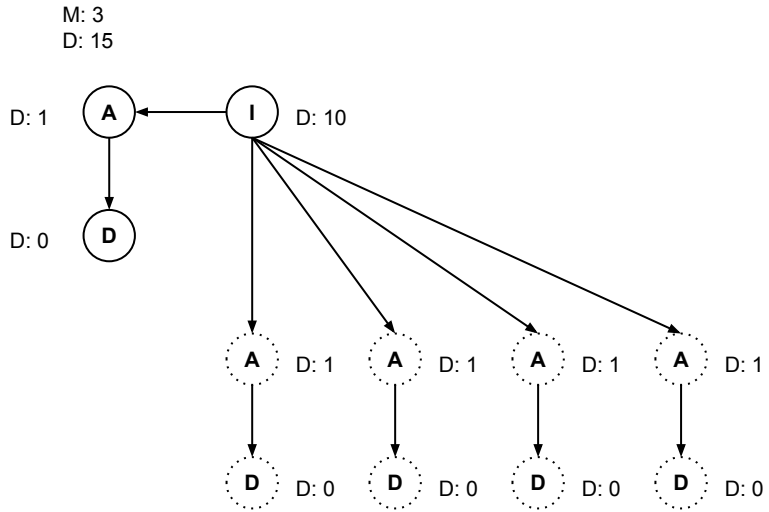


$$\frac{n^2 + n}{2}$$

a.k.a. triangular number

…growing much faster than linear

The math of dependencies with api, impl and domain



n x 15

…linear growth

Example calculations

Chained modules	Modularized	Modularized x 3
M: 10 D: 45	M: 10 D: 135	M: 30 D: 405
M: 300 D: 44 850	M: 300 D: 4 485	M: 900 D: 13 455
M: 650 D: 210 925	M: 650 D: 9 735	M: 1 950 D: 29 205

Gradle build logic improvements

Scope of gradle improvements

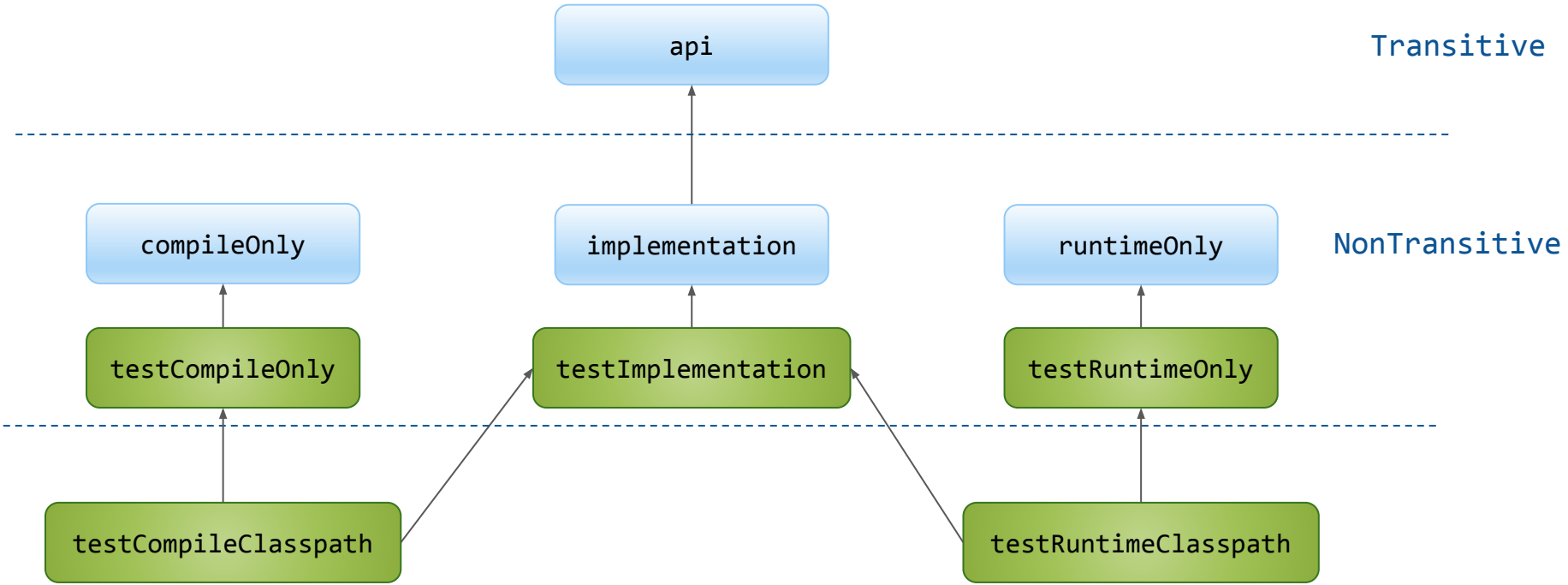
Centralise gradle build logic into plugins

Ensure proper code tagging: i.e. test code vs production code

Improve gradle/idea integration

Migrate from groovy script to kotlin script: Better developer experience

Gradle scopes...?



Results

Reduce initial loading of repo from > 20 mins to 5 mins

We need dev machines with > 32 GB RAM

By properly tagging the source sets we drastically reduced unwanted re-indexing

Moving functionality to plugins will allow us to enforce architectural decisions in compile time

Adopting ArchUnit (https://www.archunit.org/) for more advanced checks as 2nd line of architectural checks

etraveli group

Long term planning

mytrip GO TO GATE flight network

Evolving architecture not just our code base

Adopting message driven communication RabbitMQ

Offloading events to Kafka

Splitting applications into services

- Splitting database schemas and relocating schemas
- Extracting domains to standalone applications

Evolving our packaging to containers

- Ultimately K8s will host all our applications

Aiming to release more frequently and more reliably

etraveli group

mytrip GO TO GATE flight network