

PYCONES
GRANADA 2022

Inyección de dependencias, fácil!

@panicoenlaxbox

<https://www.panicoenlaxbox.com/>

<https://github.com/panicoenlaxbox/pycones2022>

<https://analyticalways.com/>

¿Por qué hacer DI?

- Código mantenible.
- Código testeable.
- Entrega de valor continua.

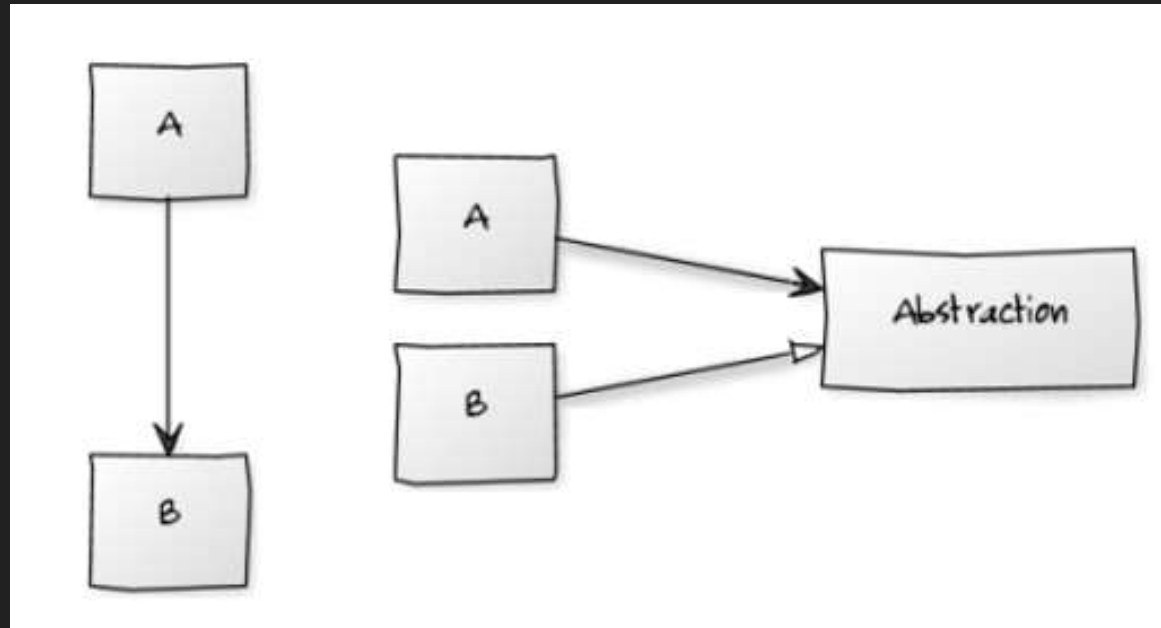
Dependency inversion principle

- High-level modules should not depend on low-level modules. Both should depend on abstractions.
- Abstractions should not depend on details. Details should depend on abstractions.

Dependency inversion principle v2

- Nuestro código de negocio no debería depender de detalles técnicos. Ambos deberían depender de abstracciones.
- Las abstracciones mandan.

DIP, en la práctica



Dependency inversion principle v3

- Depend on abstractions

- http://principles-wiki.net/principles:dependency_inversion_principle

- Program to an interface, not an implementation

- <https://www.amazon.es/Design-Patterns-Object-Oriented-professional-computing/dp/0201633612>

- <https://martinfowler.com/articles/injection.html>

Abstracción

- ABC
- Duck typing
- `typing.Protocol`

Beneficios

- El módulo de bajo nivel puede cambiar sin impactar al módulo de alto nivel.
- Se pueden reutilizar los módulos de alto nivel.
- Puede cambiar el comportamiento del sistema según la implementación que se inyecte.
 - Open-Closed Principle (OCP).
- Parallel development.
- Código más claro.
 - The Zen of Python "Explicit is better than implicit."
- Nos ayuda a detectar code-smells.
 - Single Responsibility Principle (SRP).

Testing

- Se puede aislar el SUT, mockeando las dependencias.
 - <https://opensource.com/article/17/5/30-best-practices-software-development-and-testing>
- Monkey-patching vs mocking.
 - El patch se hace de la implementación y el mock de la abstracción.
 - Exceso de patch como sinónimo de mal diseño.
- OK
 - responses, <https://github.com/getsentry/responses>
 - freezegun, <https://github.com/spulec/freezegun>
 - pyfakefs, <https://github.com/jmcgeheeiv/pyfakefs/>

testing*.py

DI container

- DI container
 - Object composition.
 - Lifetime management.
 - Interception.

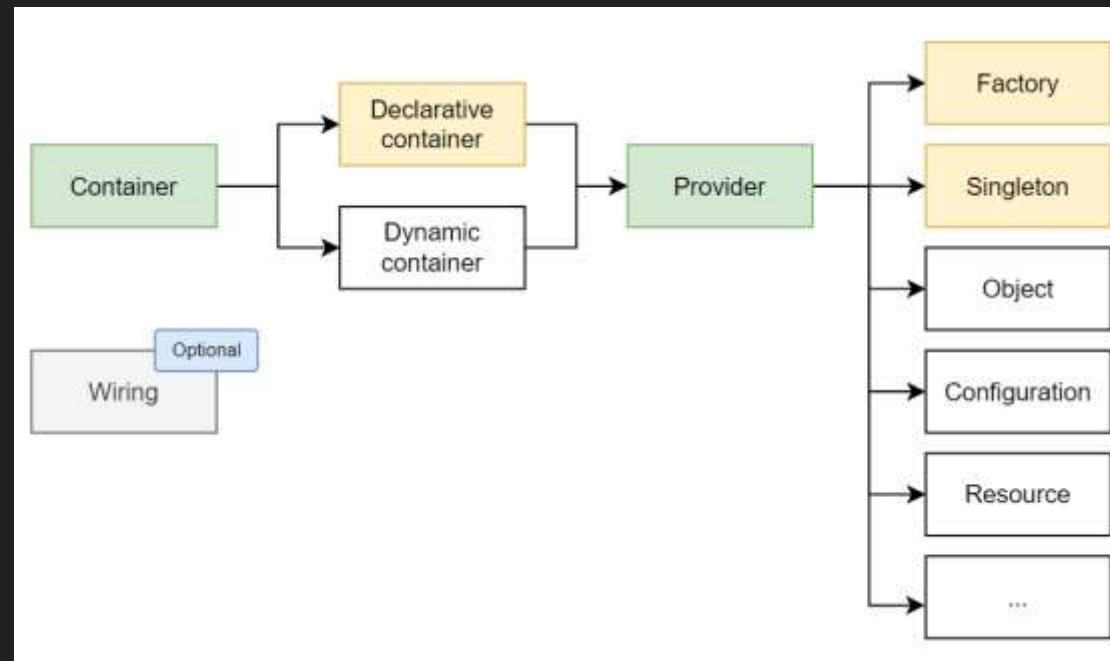
¿Qué hay que inyectar?

- Dependencias estables vs volátiles
 - <https://www.manning.com/books/dependency-injection-in-dot-net>
- Se inyectan las dependencias volátiles
- Una dependencia es estable mientras que no se demuestre que es volátil.
 - ¿Queremos inyectar distintas implementaciones?
 - ¿Queremos mockear la dependencia?
 - ¿Es necesaria para hacer parallel development?



Dependency Injector

○ <https://python-dependency-injector.ets-labs.org/>



pure_di, dependency_injector_example*.py, tests\dependency_injector_example*.py

¿Hay que hacer siempre DIP?

- Depende...

- Si DI (y todo lo que conlleva) supera o iguala a la complejidad inherente del problema, entonces es complejidad accidental.

- El beneficio es directamente proporcional al tamaño y complejidad de la aplicación.

That's all folks

○ <https://github.com/panicoenlaxbox/pycones2022>