Contexts & Dependency Injection

http://www.cdi-spec.org/learn/

My code is spaghetti

Common antipatterns

- Control Freak
- Bastard Injection
- Constrained Construction
- Service Locator *
- Factory

Service Locator advantages

- We have support for late binding by changing the registration.
- We can develop code in parallel because we program against interfaces and can replace modules at will.
- We can achieve good separation of concerns, so nothing stops us from writing maintainable code. But doing so becomes much more difficult.
- We can replace DEPENDENCIES with Test Doubles for TESTABILITY

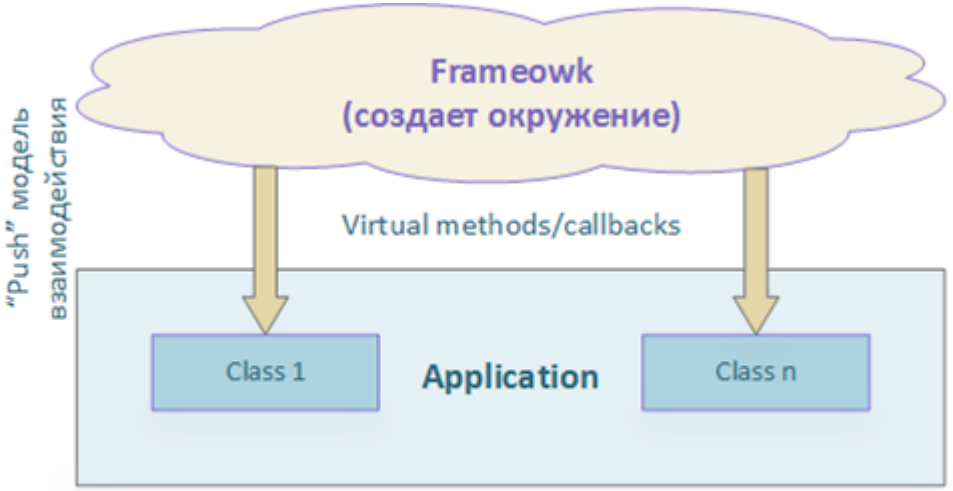
Service Locator drawbacks

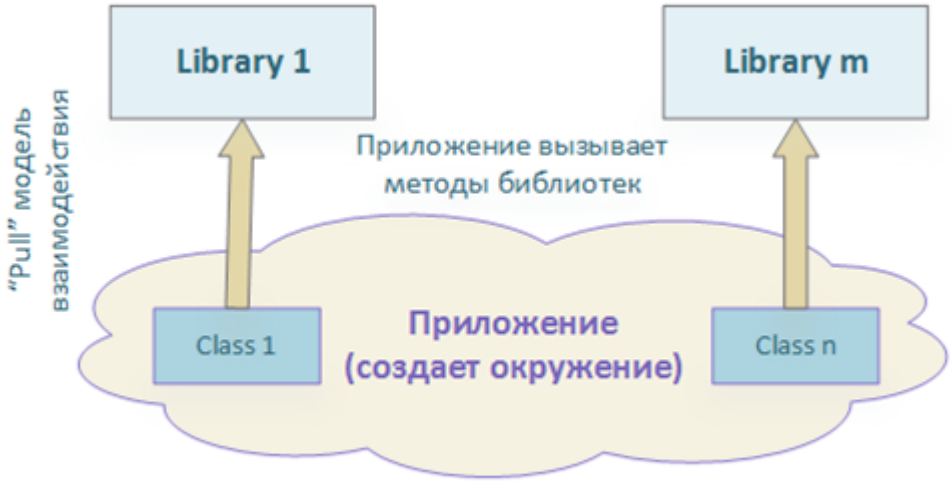
- The module drags along a redundant DEPENDENCY – locator itself.
- We don’t see which modules depend on which, need to open constructor implementation for it
- “Surprises” with some dependency not being resolved are likely
- Breaks encapsulation

Dependency Injection

Inversion of Control

- Library vs Framework

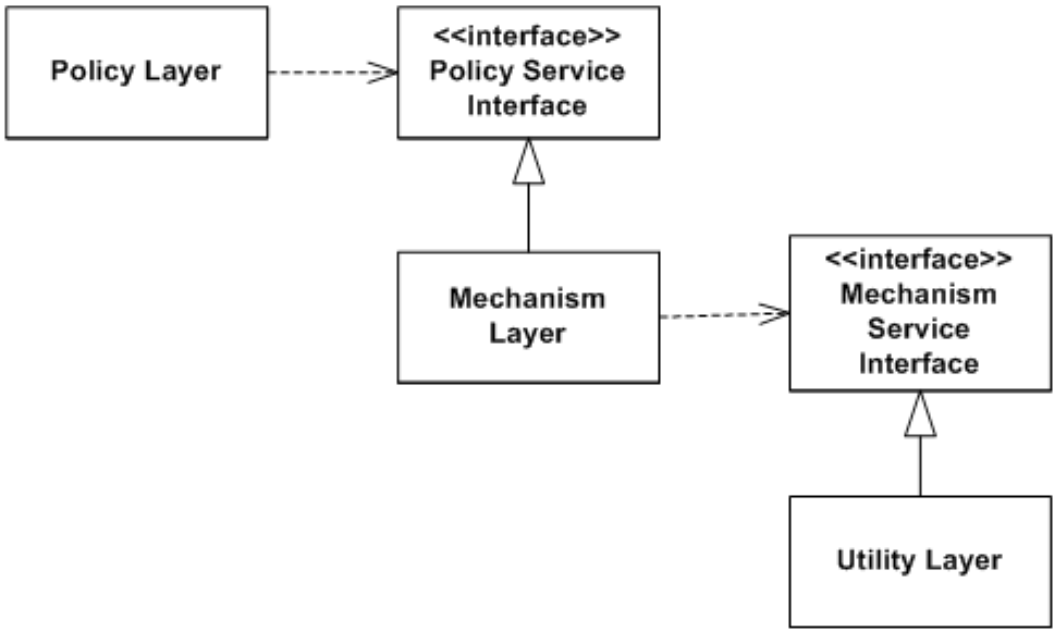




Don't call us, we'll call you

Dependency Inversion Principle

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



Dependency Injection

- Implements last, but not the least principle of SOLID – Dependency Inversion (Inversion of Control)
- Is a process of managing dependencies of software components
- Enforces **loose coupling**
- Enables **parallel** development
- Handles **cross-cutting concerns**
- As a result - **ehanced testability**
- As a result - **ehanced maintainability**

DI vs Factory

DEPENDENCY INJECTION

Dependency is injected by IOC container like Spring or Google Guice.

Result in much cleaner client class like POJO

Loosely coupled design, no coupling between container and client class.

Easier to Unit test

FACTORY PATTERN

Dependency is acquired by the class itself by calling factory method e.g. getInstance().

Client class is not as clean as in case of DI

Tightly coupled design, client class is directly coupled with factory class.

Not so easier to Unit test, need factory class to unit test even client classes.

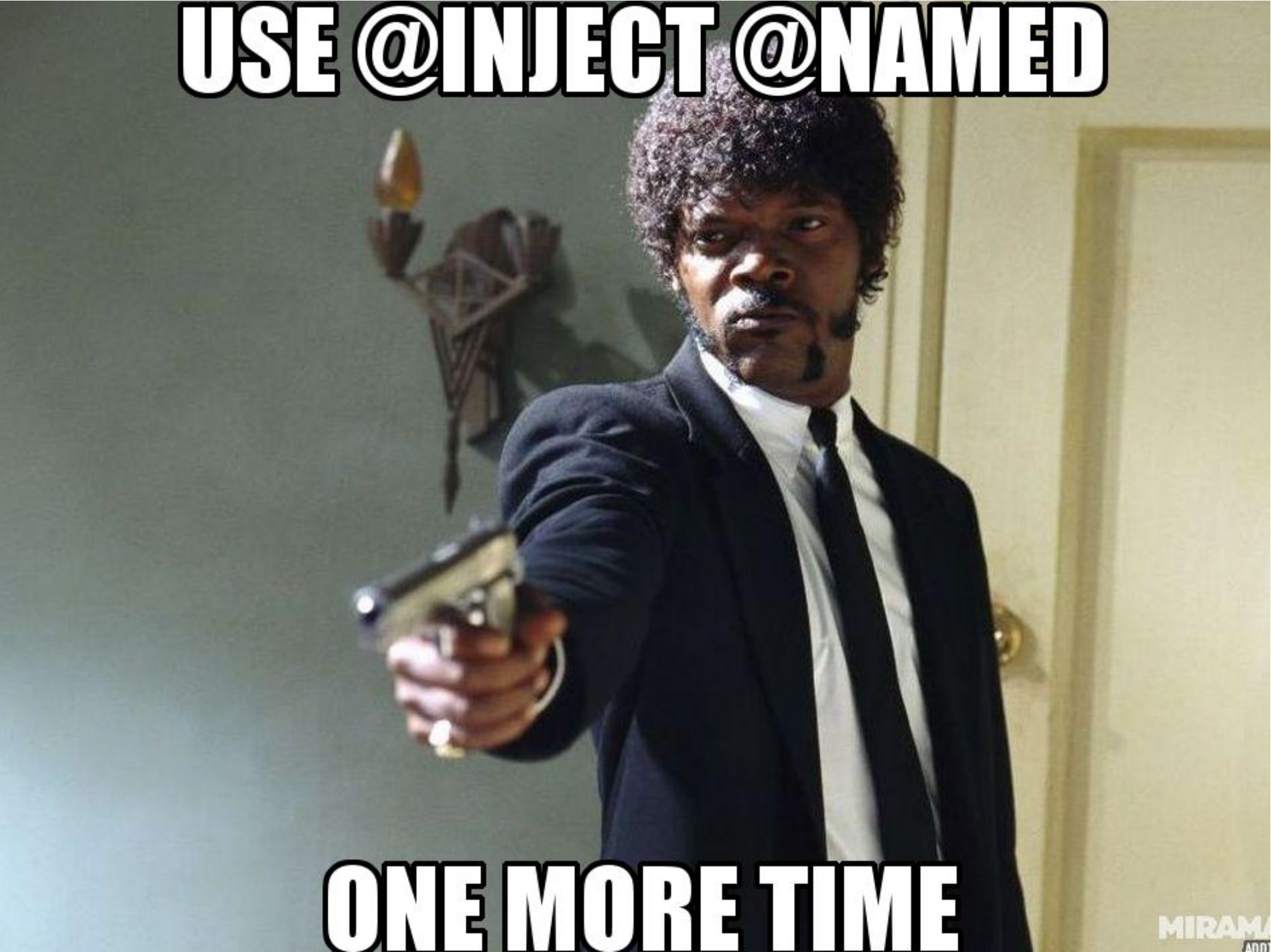
Bean

Requirments:

- No-args constructor
- Constructor with @Inject

Annotations save the world

- @Inject
- @Produces
- @Qualifier
- @New
- @Any
- @All
- @Named
- @Alternative


USE @INJECT @NAMED
ONE MORE TIME