Dependency Injection pattern

Agenda

- Definition of the pattern.
- Relation to other patterns.
- Short Guice introduction.
- Use of Guice in XText framework.

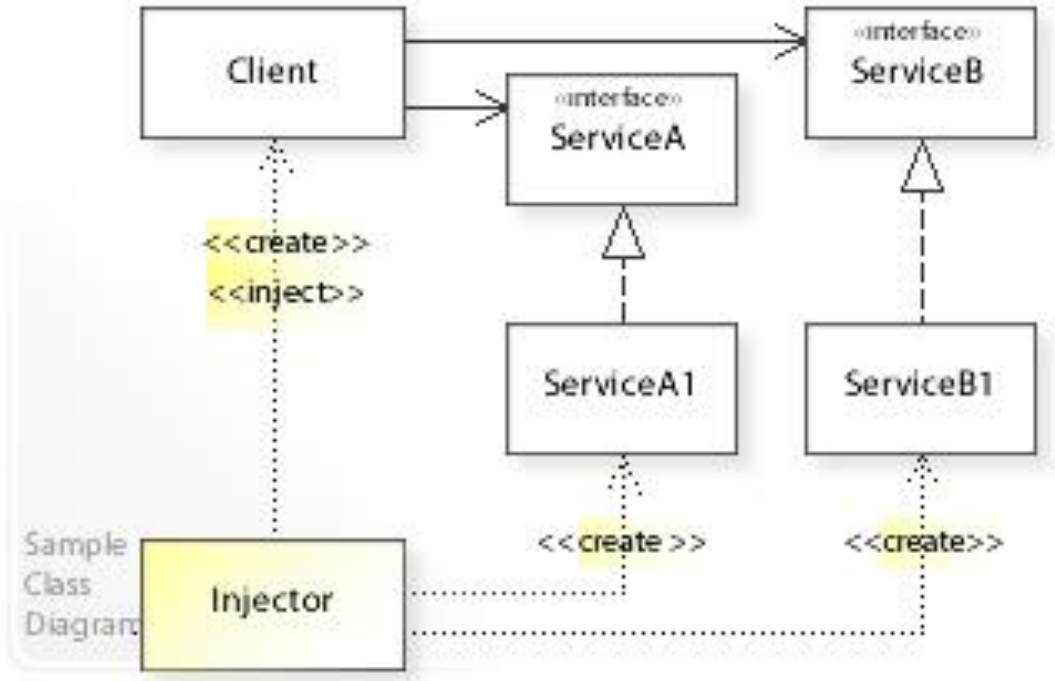
Definition (Guice website)

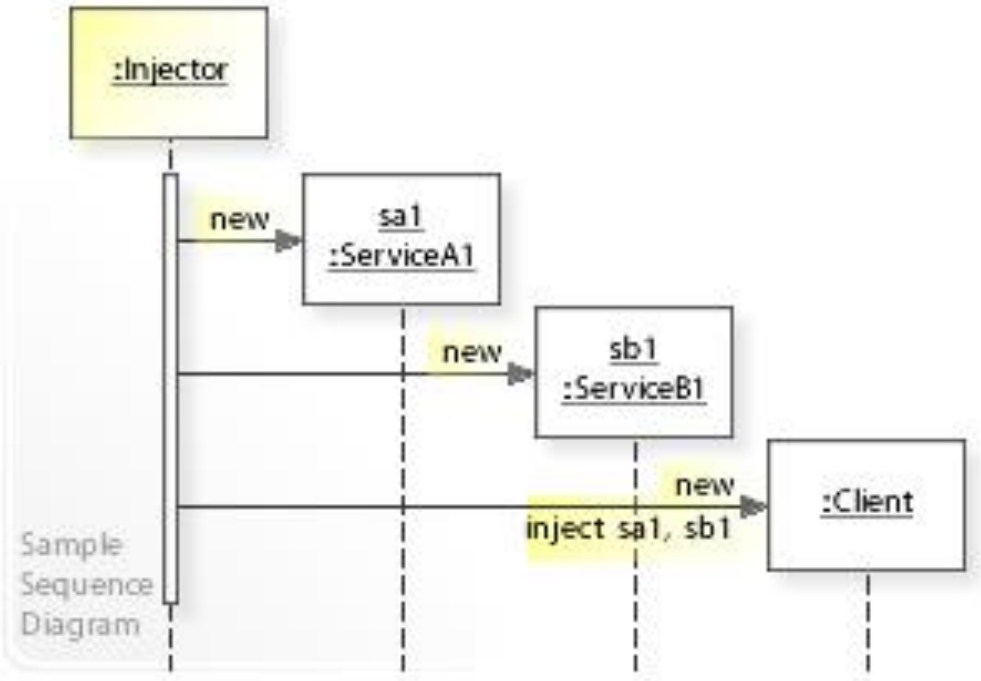
- **Dependency injection** is a technique whereby one object supplies the dependencies of another object.
- With dependency injection, objects accept dependencies in their constructors.
- To construct an object, you first build its dependencies. But to build each dependency, you need *its* dependencies, and so on. So when you build an object, you really need to build an *object graph.*
- Building object graphs by hand is labor intensive, error prone, and makes testing difficult.
- Responsibility of providing objects and its dependencies is moved to external code - the injector.

Definition (Wikipedia)

- **Dependency injection** is a technique whereby one object supplies the dependencies of another object.
- A dependency is an object that can be used (a service).
- An injection is the passing of a dependency to a dependent object (a client) that would use it.
- The service is made part of the client's state. Passing the service to the client, rather than allowing a client to build or find the service, is the fundamental requirement of the pattern.
- This fundamental requirement means that using values (services) produced within the class from new or static methods is prohibited. The class should accept values passed in from outside. This allows the class to make acquiring dependencies someone else's problem – injector.
- Client does not need to know about injecting code (injector is not used directly), how to construct services or implementation of services (only interfaces). This separates the responsibilities of use and construction.

UML Diagram


Client
«interface»
ServiceB
«interface»
ServiceA
<<create>>
<<inject>>
ServiceA1
ServiceB1
<<create>>
<<create>>
Sample
Class
Diagram
Injector

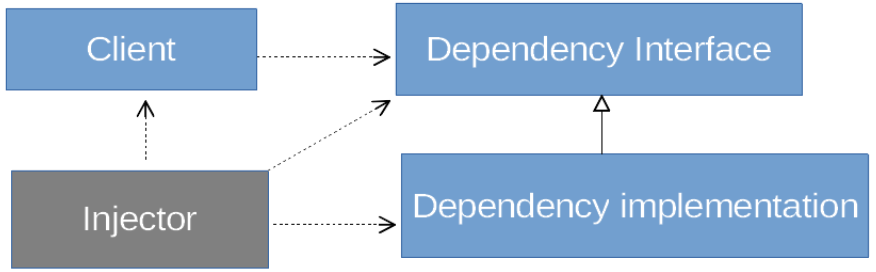

:Injector
new
sa1
:ServiceA1
new
sb1
:ServiceB1
new
:Client
inject sa1, sb1
Sample
Sequence
Diagram

Related patterns

- Dependency injection is one form of the broader technique of inversion of control. Put simply, IoC means letting other code call you rather than insisting on doing the calling. In this case injecting dependencies.
- Dependency injection supports the 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.



In this case Client has no knowledge of Dependency implementation.
Only injector has dependencies to Client and Dependency implementation.

Intent

- How can
  - an application be independent of how its objects are created?
  - a class be independent of how the objects it requires are created?
  - the way objects are created be specified in separate configuration files?
  - an application support different configurations?

The Dependency Injection design pattern describes how to solve such problems:

- Define a separate (injector) object that creates and injects the objects a class requires.
- A class accepts the objects it requires from an injector object instead of creating the objects directly.

Guice Demo

- https://github.com/google/guice/wiki
- https://github.com/kkot/guice-demo