gepardec

CDI Specs

Dive in

Date : 10.12.2021
Author: Ing. Thomas Herzog M.Sc
Version: v1.0

Topics

// CDI Scopes

// CDI Interceptor

// CDI Decorator

// CDI Events

CDI Scopes

// Scopes have a well define lifetime

// Scopes are nested, never intersecting

// Scopes define the lifetime of a CDI bean within the scope

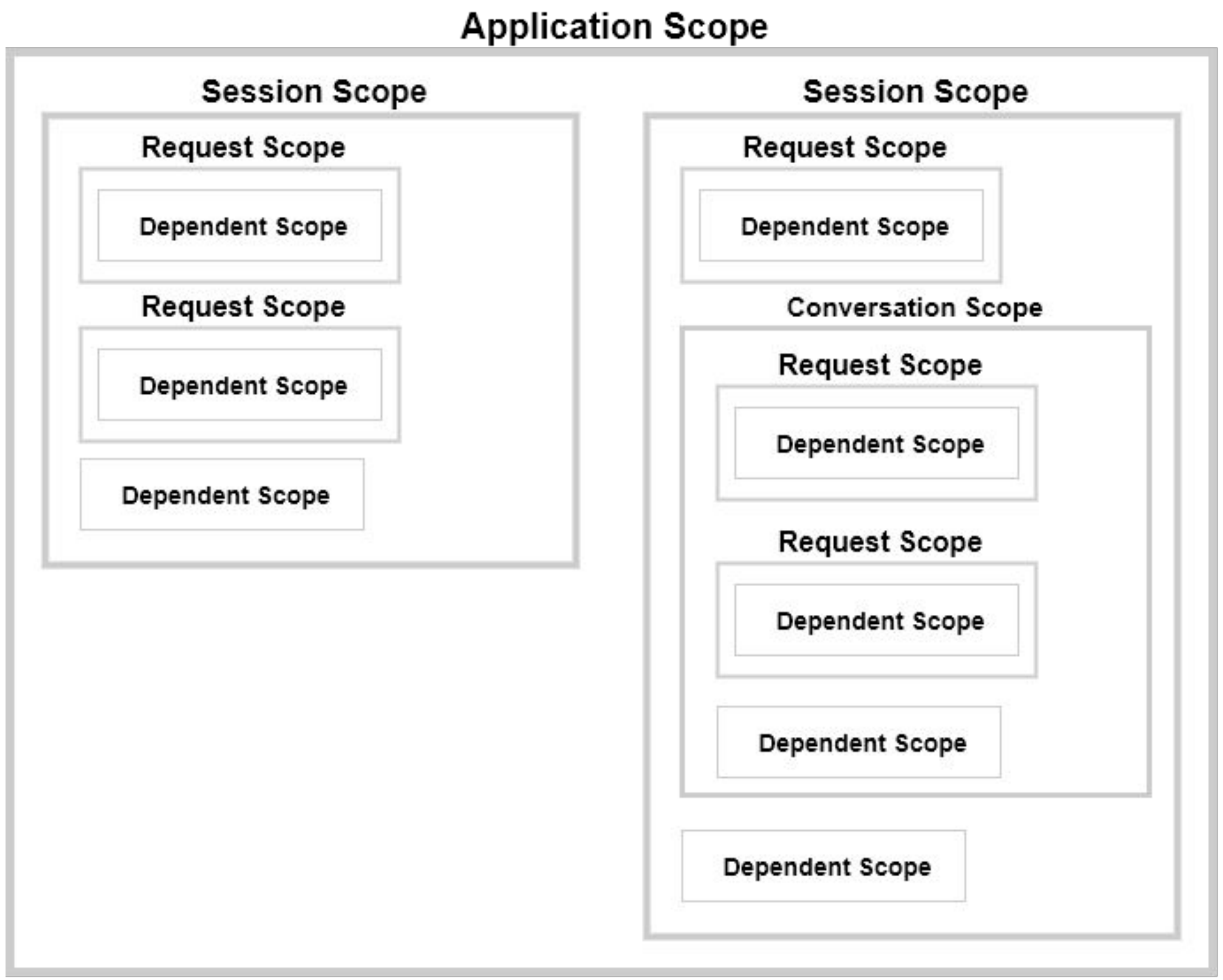
// Scopes define when to create and destroy a CDI bean

// Dependent Scope is a Pseudo Scope, the others are normal Scopes

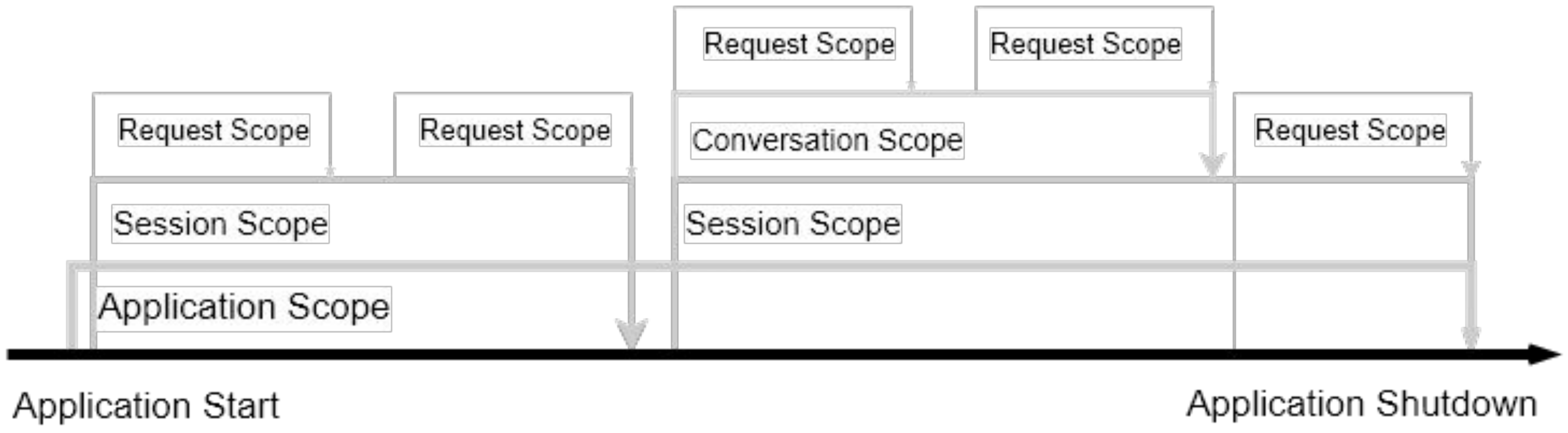
// Scopes in CDI are

- `Application`
- `Session`
- `Conversion`
- `Request`
- `Dependent`

CDI Scopes (Nesting)



CDI Scopes (Lifecycle)



// Beans are lazily created in CDI on first method invocation

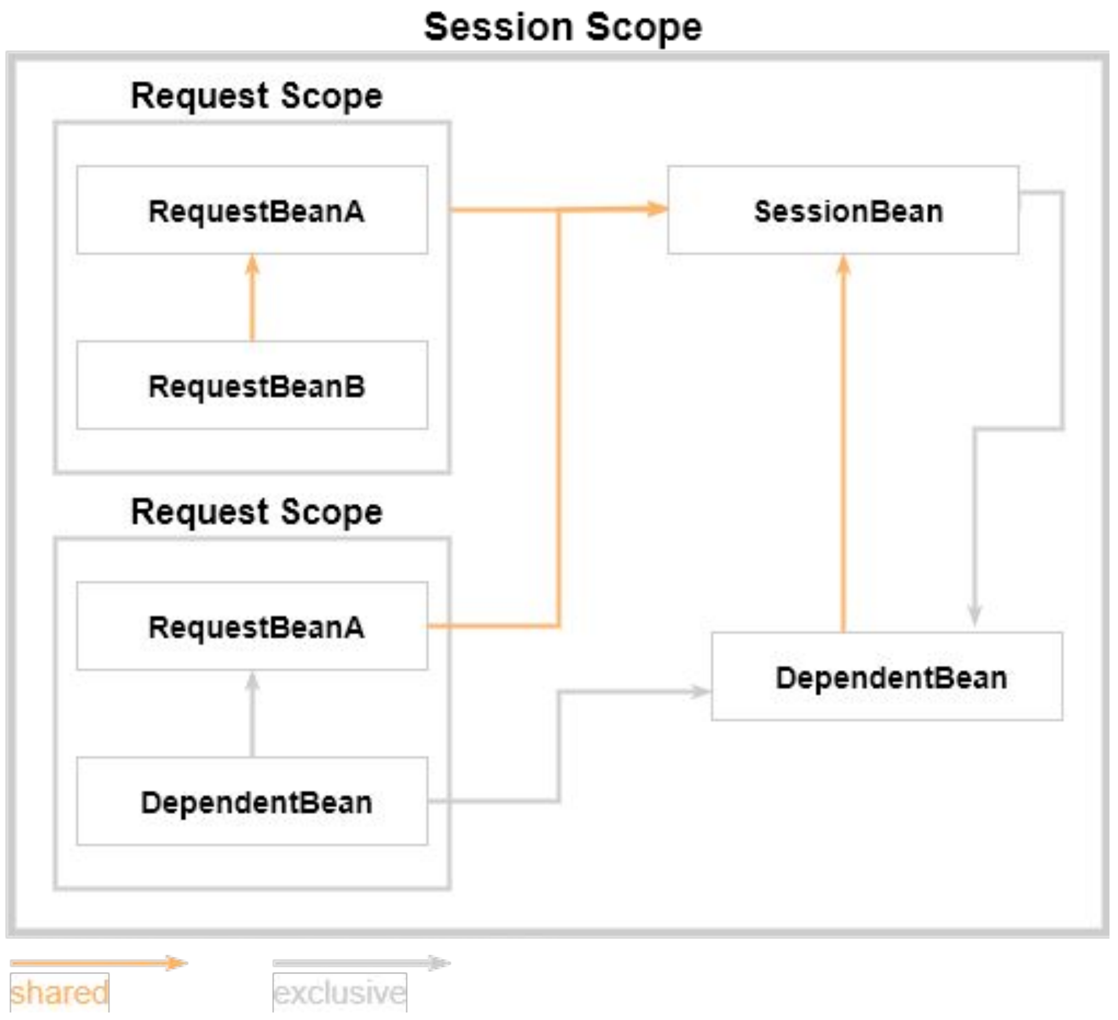
// Scopes of the same type don’t intersect

// Scopes don’t live longer than their parent Scope

// No Session Scope without a Request Scope

// CDI beans are destroyed at the end of the scope lifetime

CDI Scopes (Implications nested Scopes)



CDI Interceptor

// Interceptors are used for aspect oriented development

// Interceptors are chained and ordered

// Interceptors can share information to the following Interceptors

// Interceptors are not aware of the class they intercept

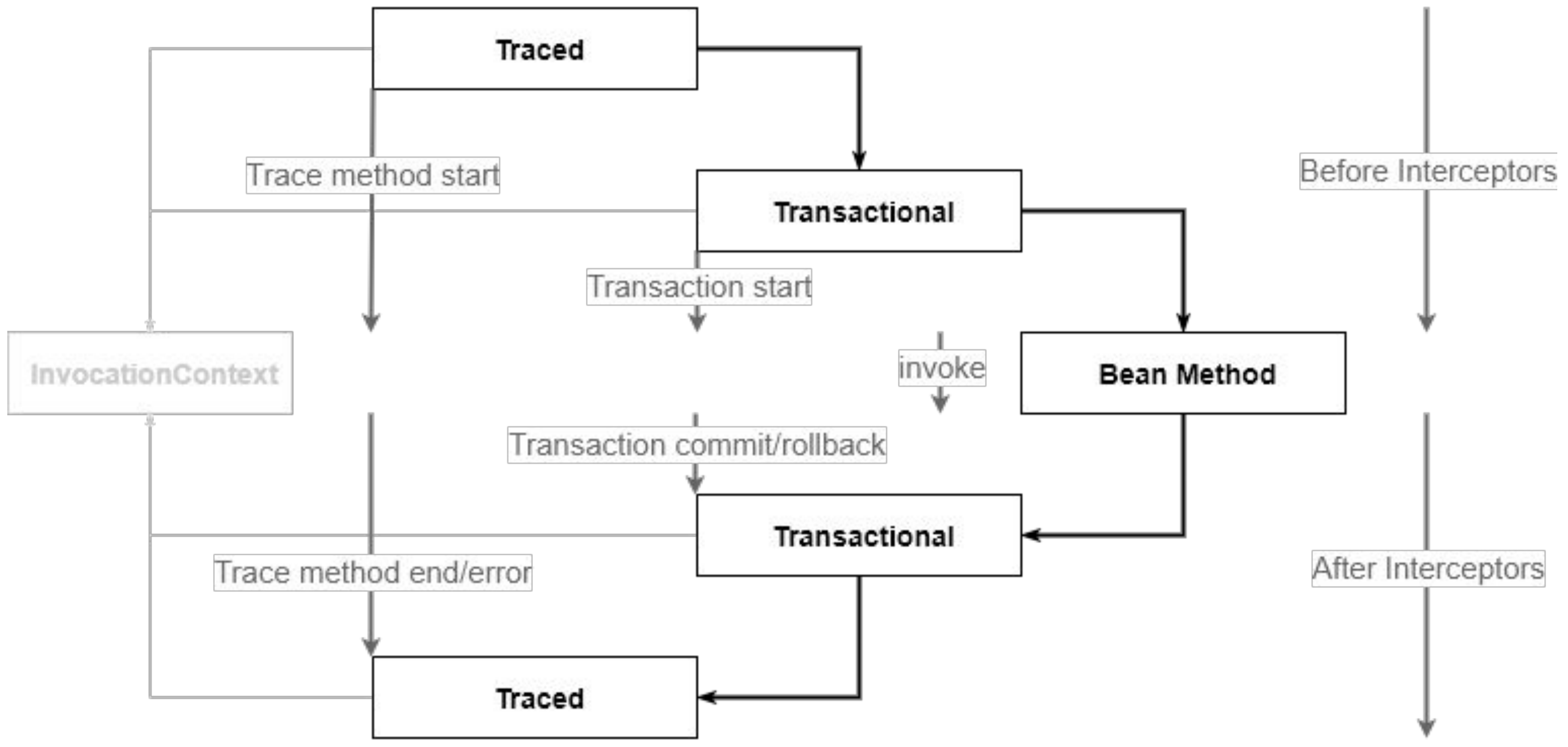
// Interceptors need reflection

// Interceptors usually activate/destroy and manage an aspect

// Examples for Interceptors

- `@Transactional`
- `@Traced`
- `@Valid`

CDI Interceptor (chain)



gepardec

CDI Decorator

// Decorators are used for decorating method invocations

// Decorators are chained and ordered

// Decorators are decoupled from the decorated bean

// Decorators are always invoked after Interceptors

// Decorators are executed within an aspect

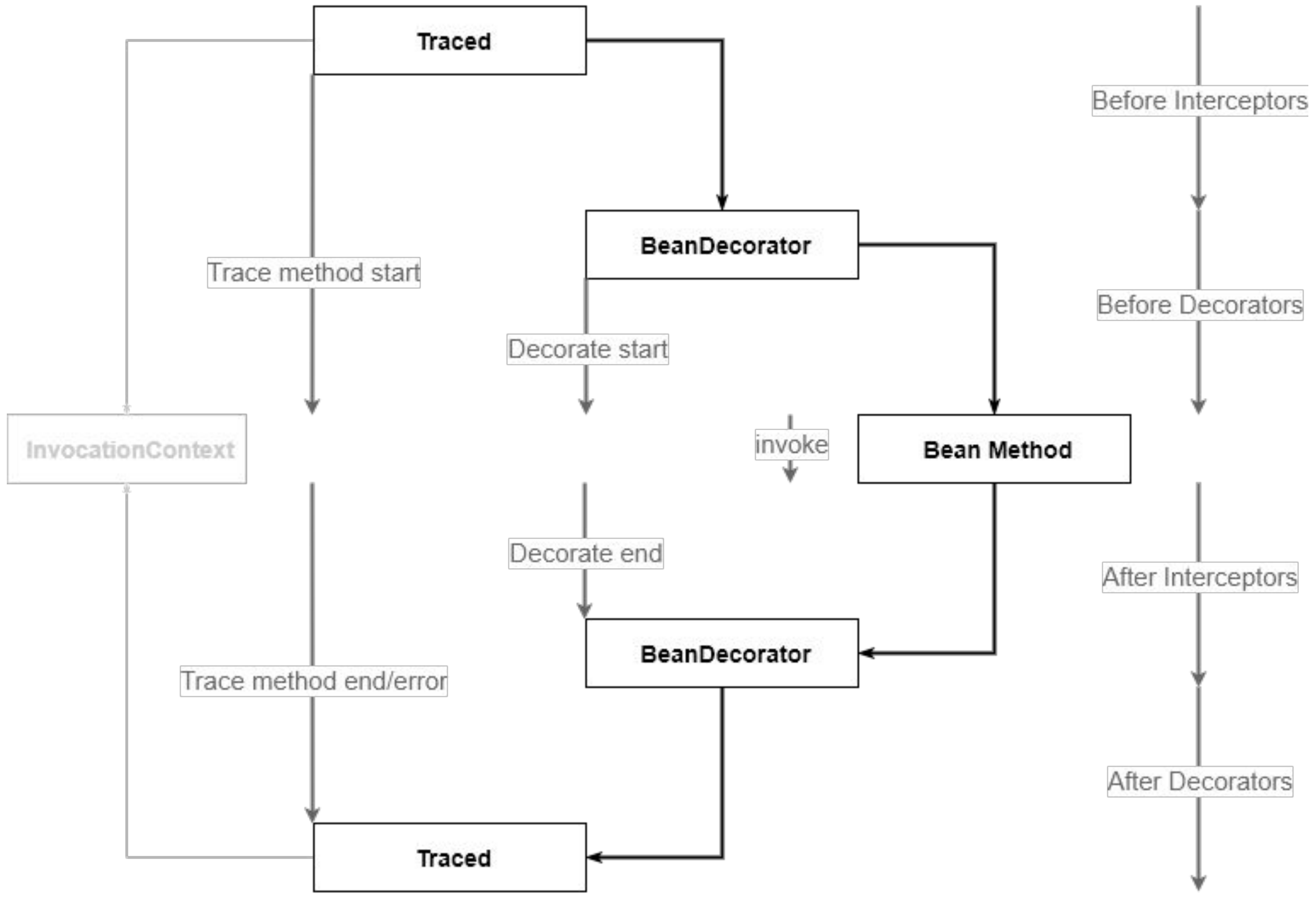
// Decorators cannot share information to the following Decorator

// Decorators are aware of the interface/class they intercept

// Decorators have access to the input/output parameters

gepardec

CDI Decorator/Interceptor (chain)



CDI Events

// CDI Events implement the Publisher/Subscriber pattern

// CDI Events are type-safe *(interfaces or classes)*

// CDI Events abstract publisher and subscriber in both ways

// CDI Events are POJOs

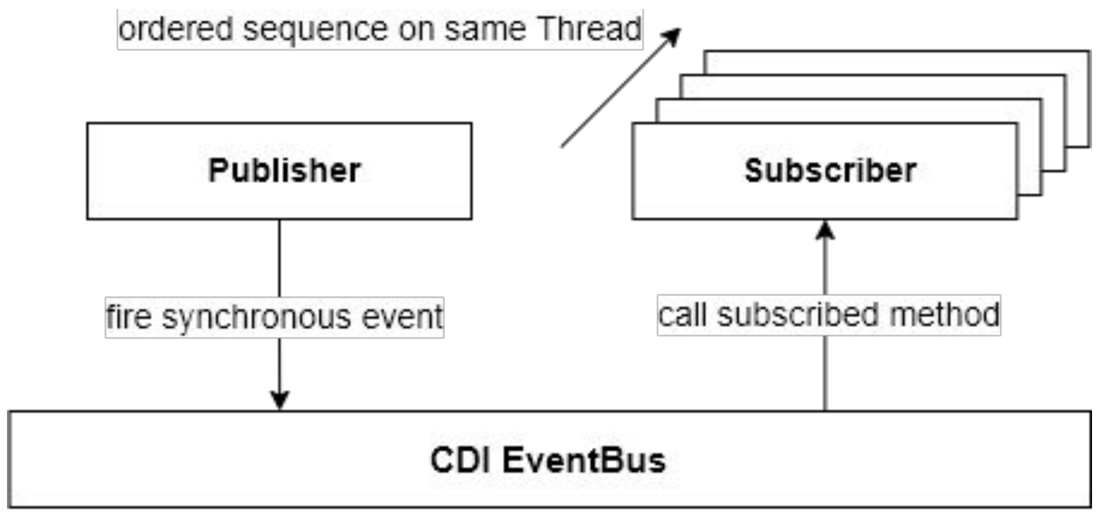
// CDI Events can be synchronous or asynchronous *(since CDI 2.0)*

// CDI Event-Bus is within CDI, we never touch it

// CDI Event-Subscriber can be transaction aware

// CDI Event-Subscriber can be ordered

CDI Event (synchronous)



// Publisher fires typed event object

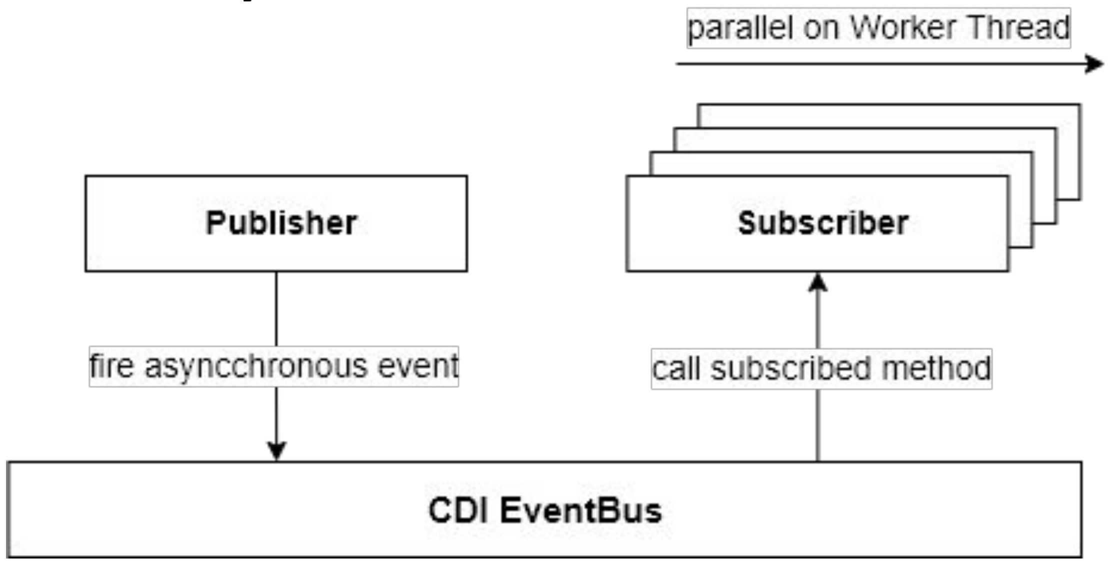
// Subscribers are invoked in ordered sequence on the same Thread

// Subscribers must be in the same context

// If not restricted, beans are created for event delivery

// Publisher and Subscriber can share data via the event

CDI Event (asynchronous)



// Publisher fires typed event object

// Subscribers are invoked in any order on a Worker-Thread

// Subscribers must be in application or Dependent Scope!!

// If not restricted, beans are created for event delivery

// Publisher and Subscriber can not share data via the event

// Exceptions are suppressed

The introduced CDI specifications especially the scopes are an integral part of CDI which needs to be handled properly in your application source code.

The interesting part starts when all CDI specifications come together in real life applications where the specifications of CDI interact/relate or depend on each other.