Injeção de Dependência

O que é injeção de dependência ?

Injeção de Dependência é um padrão de projeto que ajuda muito a deixar o código desacoplado, melhora a legibilidade e interpretação do código, melhora a distribuição de responsabilidades entre as classes e facilita a sua manutenção.

É uma forma de aplicar **inversão de controle** em uma classe que utiliza funcionalidades de outras classes, tirando a responsabilidade dela de *instanciar ou buscar* objetos dessas outras classes das quais ela depende.

O objetivo é **não instanciar dentro de uma classe objetos que realizam funções que podem ser alteradas futuramente** e sim **deixar a responsabilidade dessa instanciação para quem chama a classe**.

Como o Quarkus lida com injeção de dependências ?

O Quarkus implementa injeção de dependências através de um container chamado **Quarkus ArC**. Este container é o responsável por gerenciar todas as dependências do projeto de forma automática.

Os objetos gerenciados pelo container do Quarkus são chamados de *Beans*. Uma aplicação rodando pode ter vários Beans ativos e gerenciados pelo Quarkus.

Existem várias formas de declarar um bean do Quarkus. A mais conhecida é anotando a classe com *@ApplicationScoped*

Referencia: https://quarkus.io/blog/quarkus-dependency-injection/

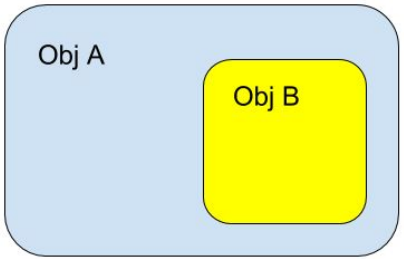
Como o Quarkus lida com injeção de dependências ?

Os Beans gerenciados pelo Quarkus são instanciados automaticamente pelo ArC Container e todas as dependências são resolvidas e repassadas pelo próprio framework.

A ideia principal é injetar Beans em outros Beans, ou seja, eu só posso injetar um objeto em outro objeto se eles forem gerenciados pelo Quarkus.

Como o Quarkus lida com injeção de dependências ?

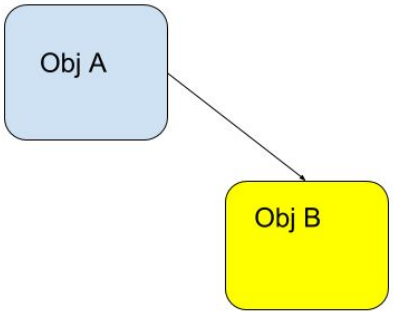
Dependência forte




```
public class ObjA{
    private ObjB objB;

    public ObjA(){
        this.objB = new ObjB();
    }
}
```

Injeção de dependência




```
public class ObjA{
    private ObjB objB;

    public ObjA( ObjB objB ){
        this.objB = objB;
    }
}
```

Configuração de Beans

Por padrão o Quarkus faz a instanciação dos Beans sem fazer nenhum tipo de configuração. Se em algum cenário for necessário fazer uma configuração de Bean, é possível criar uma classe Java com a anotação *@Dependent* e definir métodos com os nomes dos Beans a serem instanciados e gerenciados pelo container.

Referencia: https://quarkus.io/guides/cdi-reference#default_beans

Configuração de Beans

Nesses métodos podemos fazer a configuração de forma mais personalizada, por exemplo:

```
1  @Dependent
2  public class ModelMapperConfig {
3
4      @Produces
5      @DefaultBean
6      public ModelMapper modelMapper() {
7          var modelMapper = new ModelMapper();
8          return modelMapper;
9      }
10 }
```

Pontos de injeção

Podemos injetar dependências em 3 locais. No construtor da classe, em um atributo e em método set. Para indicar uma injeção de dependência, anotamos o ponto de injeção com a anotação *@Inject*.

Injeção através do Construtor da classe

```
1  @ApplicationScoped
2  public class AlunoService {
3
4      private AlunoRepository alunoRepository;
5
6      @Inject
7      public AlunoService (AlunoRepository alunoRepository) {
8          this.alunoRepository = alunoRepository;
9      }
10 }
```

Injeção através de um Atributo da classe

```
1  @ApplicationScoped
2  public class AlunoService {
3
4      @Inject
5      private AlunoRepository alunoRepository;
6
7  }
```

Injeção através de um método Set da classe

```
1  @ApplicationScoped
2  public class AlunoService {
3
4      private AlunoRepository alunoRepository;
5
6      @Inject
7      public void setAlunoRepository(AlunoRepository alunoRepository){
8          this.alunoRepository = alunoRepository;
9      }
10 }
```