CDI

CDI @inject :

Bu su demek : INFERFACE ‘e uygun classi otomatik bulup, impelements edip, CONTROLLER OLUSTURUYOR.

NAMED dersen, senin ismini verdigin CLASSi impelements ediyor.

```
public class AutoServiceCallerImp implements AutoServiceCaller{

    @Inject
    @Auto(type = AutoType.Bmw)
    private AutoService bmwAutoService;

    @Inject
    @Auto(type = AutoType.Honda)
    private AutoService hondaAutoService;

    @Inject
    @Auto(type = AutoType.Ford)
    private AutoService fordAutoService;

    ...
}
```

Ref : buraktas.com

Named is CALLED if there are “more than one implementation for an interface.”

Impelement edilebilir CLASS’larin basina, @Default. @Alternative koyabilirsin. !

From Another Person : buraktas.com/create-qualifiers-cdi-beans/

Java CDI Dependency Injection Example

CDI (Context and Dependency Injection) is a specification defined in JSR-299. Major aim is loose coupling by dependency injection.In this tutorial we will see how to use CDI Dependency Injection in java with three different ways;

- Field injection
- Constructor injection
- Setter method injection

We are going to use @Inject alongside @Named annotations from CDI of Java EE. @Named annotation is used for giving names for classes which implements the interface, and it is optional. Otherwise, we can specify alternatives and a default bean by @Alternative, and @Default annotations. However, I am going to show them in this tutorial. Moreover, these are the example interface and implementation classes we are going to use.

Notes:

- @Named annotation is commonly used if there are more than one implementation for an interface. Thus, it provides to give and inject by their names.
- If there is only one implementation of an interface and @Named annotation is used, then the name of the bean is determined as camelCase style of class name.
- We can use @Default and @Alternative annotations instead of giving names to them.
- If there is only one implementation of an interface, compiler will inject it as the default one. So, there is no need to use @Named, @Default or @Alternative annotations.

Important

We have to create an empty beans.xml file in src/main/resources/META-INF if it is a jar application or src/main/webapp/WEB-INF if it is a war application.
[crayon-66337569488c8548628858/]
AutoService.java

[crayon-66337569488ce583255293/]
BMWAutoService.java
[crayon-66337569488d1352136407/]
FordAutoService.java
[crayon-66337569488d3143821919/]
HondaAutoService.java
[crayon-66337569488d6396076855/]
AutoServiceCaller.java
[crayon-66337569488d8851455405/]

1. Injection Through Fields

Beans are injected through fields.
[crayon-66337569488da417813969/]

2. Injection Through Setter Methods

We can also inject our beans via setters
[crayon-66337569488dd539246965/]

3. Injection Through Constructor

Finally, beans can be injected through the constructor of the class.
[crayon-66337569488e0869470420/]