

CDI

An introduction to CDI

Date : 22.03.2021
Author: Ing. Thomas Herzog M.Sc



Topics

- // What is Dependency Injection & Inversion of Control?
- // What is CDI?
- // CDI vs EJB
- // CDI Versions
- // CDI Implementations
- // References
- // Hands on

What is Dependency Injection & Inversion of Control?

// The Problem: How should we test this?

```
1 public class PetClinic {
2     private AppointmentService appointmentService = new AppointmentService();
3
4     Appointment bookAppointment(LocalDate time, int minutes, String description) {
5         Appointment appointment = new Appointment(time, minutes, description);
6         return appointmentService.book(appointment);
7     }
8
9     // <more code here>
10 }
11
12 public class AppointmentService {
13     private Database db = new Database();
14
15     public Appointment bookAppointment(Appointment appointment) {
16         // <insert business logic here>
17         return db.create(appointment);
18     }
19 }
20
21 public class PetClinicApplication {
22     public static void main(String[] args) {
23         PetClinic clinic = new PetClinic();
24         clinic.showUI();
25     }
26 }
27
```

What is Dependency Injection & Inversion of Control?

// A first Solution

```
1 public class PetClinic {
2     private final AppointmentService appointmentService;
3
4     public PetClinic(AppointmentService appointmentService) { this.appointmentService = appointmentService; }
5
6     Appointment bookAppointment(LocalDateTime time, int minutes, String description) { /* ... */ }
7 }
8
9 public class AppointmentService {
10     private final Database db;
11
12     public AppointmentService(Database db) { this.db = db; }
13
14     public Appointment bookAppointment(Appointment appointment) { /* ... */ }
15 }
16
17 public class PetClinicApplication {
18     public static void main(String[] args) {
19         Database database = new Database();
20         AppointmentService appointmentService = new AppointmentService(database);
21         PetClinic clinic = new PetClinic(appointmentService);
22
23         clinic.showUI();
24     }
25 }
26
27
```

What is Dependency Injection & Inversion of Control?

// CDI-Style

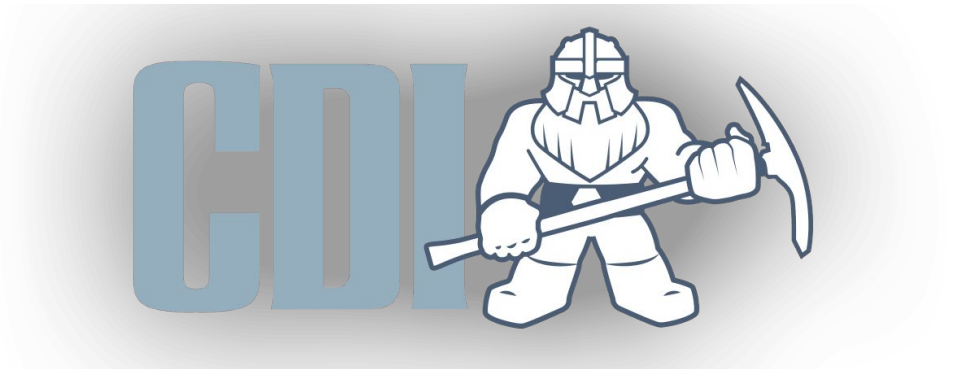
```
1 @ApplicationScoped
2 public class PetClinic {
3     @Inject
4     private AppointmentService appointmentService;
5
6     Appointment bookAppointment(LocalDate time, int minutes, String description) { /* ... */ }
7 }
8
9 public interface AppointmentService {
10     Appointment bookAppointment(Appointment appointment);
11 }
12
13 @Dependent
14 public class AppointmentServiceImpl {
15     private final Database db;
16
17     @Inject
18     public AppointmentService(Database db) { this.db = db; }
19
20     public Appointment bookAppointment(Appointment appointment) { /* ... */ }
21 }
22
23
24
25
```

This interface is actually not needed, but CDI Developers tend to introduce interfaces for everything.

What could be the reason for that?

What is CDI?

- // A component model such as EJB but is complementary
- // Type-safe Injections via Qualifiers
- // Well defined and extensible Contexts/Scopes, as well as Conversations
- // Type-safe Event Notification Model
- // Type-safe Interceptor Bindings
- // Type-safe Decorators
- // Integration into Expression Language
- // Supports modularity when resolving dependencies
- // Portable extensions via SPI



<http://cdi-spec.org/>

CDI vs EJB

// EJB injections are not type-safe, CDI injections are

```
@EJB(beanName="SecondBeanImpl")      vs      @MyQualifier @Inject
private Bean;                          private Bean;
```

// EJB is not contextual, CDI is contextual and introduces scopes

```
@Stateless/@Stateful  vs  @RequestScoped/@ApplicationScoped/@SessionScoped/@Dependent
```

// EJB is it's own world with exclusive support for

// Transactions

```
@TransactionManagement, @TransactionManagementAttribute
```

// Remoting (RMI)

```
@Remote
```

// Security

```
@RolesAllowed
```



CDI Versions

// CDI 1.0 introduced in JEE 6

- // Initial release

- // Not well integrated into other JEE Specs.

// CDI 1.1 introduced in JEE 7

- // Global enablement of Interceptors, Decorators and Alternatives via `@Priority`

- // Interceptors Bindings moved to Interceptors Spec.

- // Enhanced bean discovery (Class exclusions, Register custom types after bean discovery)

- // See what else has changed <https://in.relation.to/2013/05/31/cdi-11-available/>

// CDI 1.2 introduced in JEE 8 (Jakarta EE)

- // Mostly cleanup and clarifications

- // Introduction of `@Interceptor` and `@Decorator` annotations

- // Official support OSGI added

- // See what else has changed http://www.cdi-spec.org/news/2014/04/14/CDI-1_2-released/



CDI Versions

- // CDI 2.0 introduced in JEE 8 (Jakarta EE)
 - // Support for Java SE
 - // Support for request-context control
 - // Enhancements in the Event Model (Ordering, Asynchronous Events, BeanManager can fire events)
 - // See what else has changed https://docs.jboss.org/cdi/learn/cdi_2/slides.html
- // CDI 3.0 introduced in JEE 9 (Namespace moved to jakarta.enterprise)

CDI Implementations

// Weld

// RI from JBoss/Red Hat

// Part of Wildfly, JBoss EAP, GlassFish, Oracle Weblogic and IBM Websphere application servers

// OpenWebBeans

// OpenSource implementation of Apache

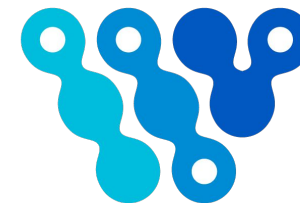
// Part of Geronimo and Tomcat application servers

// CanDI

// Part of Caucho Resin application server



<https://weld.cdi-spec.org/>



OpenWebBeans

<https://openwebbeans.apache.org/>



<https://caucho.com/>



References

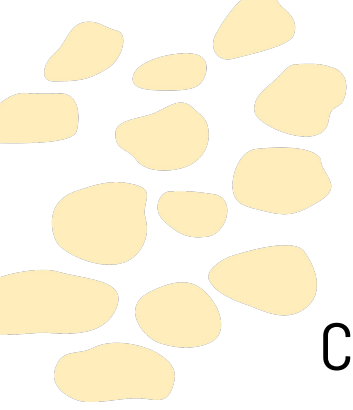
// <http://www.cdi-spec.org/>

// <https://docs.jboss.org/cdi/spec/>

// <https://struberg.wordpress.com/>

// <https://martinfowler.com/articles/injection.html>

// <https://jakarta.ee/specifications/cdi/2.0/>



CDI has become the main component model and there are few reasons why someone has to use EJB over CDI

Over time other JEE Specifications such JSF have integrated with CDI

Other JEE specifications such as Servlets will follow

Hands on example

<https://github.com/Gepardec/cdi-training>