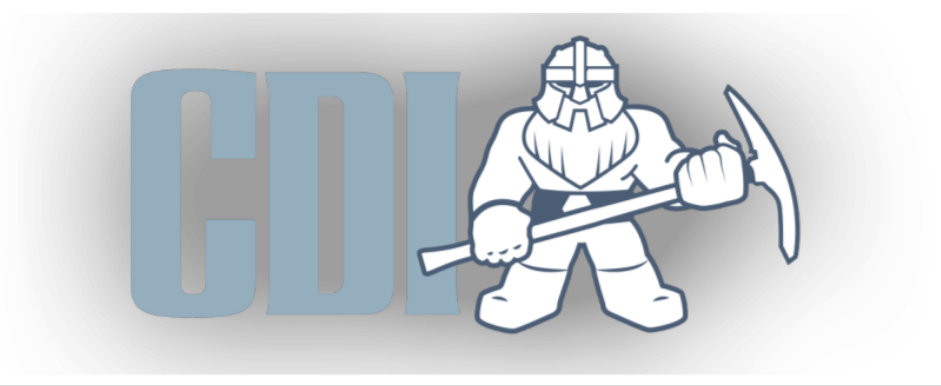
CDI

Contexts & Dependency Injection

Dependency Injection

Contexts and Dependency Injection



Spec

aktuelle Version 3.0

Begann als Teil von Java EE 6

Referenzimplementierung

Aktuelle Implementierung

früher...

Praktisch in allen EE-Projekten und EE-Folgeprojekten fix mit eingebaut (Microprofiles, etc...)

Apache DeltaSpike is a collection of portable CDI extensions. These ready-to-use modules enable you to integrate tested API extensions into your Java projects.

https://deltaspike.apache.org/documentation/overview.html

https://en.wikipedia.org/wiki/Dependency_injection

Separation Of Concerns

Man sollte die verschiedenen Aktionen, die ein Objekt ausführen kann, trennen und in eigenen Services bündeln.

Die Folge ist ein Programm, das mehrere Services braucht und diese dann auch konstruiert.

Inversion Of Control (IOC)

Ein Objekt sollte die notwendigen Services nicht selber konstruieren, sondern von außen bekommen.

Es sollte nicht wissen müssen, wie die einzelnen Services konstruiert werden müssen.

Separation Of Concerns...

```java
public class Client {

    // The service encapsulates some methods (SOC) which are
    // internally referenced by this client.
    private ExampleService service;

    // Constructor
    Client() {
        // Specify a specific implementation.
        service = new ExampleService();
    }

    // Method that uses the services
    public String greet() {
        return "Hello " + service.getName();
    }
}
```

Separation Of Concerns...

```java
public class Client {

    // The service encapsulates some methods (SOC) which are
    // internally referenced by this client.
    private ExampleService service;

    // Constructor
    Client() {
        // Specify a specific implementation.
        service = new ExampleService();
    }

    // Method that uses the services
    public String greet() {
        return "Hello " + service.getName();
    }
}
```

Problems:

Separation Of Concerns...

```java
public class Client {

    // The service encapsulates some methods (SOC) which are
    // internally referenced by this client.
    private ExampleService service;

    // Constructor
    Client() {
        // Specify a specific implementation.
        service = new ExampleService();
    }

    // Method that uses the services
    public String greet() {
        return "Hello " + service.getName();
    }
}
```

Problems:

Client has to have all parameters for ExampleService-construction in the constructor.

Separation Of Concerns...

```java
public class Client {

    // The service encapsulates some methods (SOC) which are
    // internally referenced by this client.
    private ExampleService service;

    // Constructor
    Client() {
        // Specify a specific implementation.
        service = new ExampleService();
    }

    // Method that uses the services
    public String greet() {
        return "Hello " + service.getName();
    }
}
```

Problems:

Client has to have all parameters for ExampleService-construction in the constructor.

If you want to test that, you'd have to mock PARTS of the Client.

Ohne Dependency Injection...

```
1  public class Client {
2
3      private ExampleService service;
4
5      // Constructor
6      Client(ExampleService service) {
7          this.service = service;
8      }
9
10     // Method that uses the services
11     public String greet() {
12         return "Hello " + service.getName();
13     }
14 }
15
16 public class Main {
17
18     public void main(String[] args) {
19         ExampleService service = ExampleService();
20
21         Client client = new Client(service);
22
23         System.out.println(client.greet());
24     }
25 }
```

Service is passed now to Client upon construction

Ohne Dependency Injection...

```
1  public class Main {
2
3      public void main(String[] args) {
4          ExampleService1 service1 = ExampleService1();
5          ExampleService2 service2 = ExampleService2();
6          ExampleService3 service3 = ExampleService3();
7          ExampleService4 service4 = ExampleService4();
8          ExampleService5 service5 = ExampleService5();
9          ExampleService6 service6 = ExampleService6();
10
11         Client client = new Client(service1, service2, service3, service4, service5,
12
13         System.out.println(client.greet());
14     }
15 }
```

Many services...

Ohne Dependency Injection...

```
1  public class Main {
2
3      ...
4
5      AlertEmailHandler alertEmailHandler = new AlertEmailHandler(
6          emf, emailHandler, alarmDao, alarmEmailDao,     alarmEmailTagDao,
7          alarmEmailGroupDao, userDao, preferencesDao, subscriptionDao,
8          thresholdDao, eventDao, tagToOpcUaSubscriptionDao, tagToEventDao);
9
10     ThresholdEnforcementHandler thresholdEnforcementHandler = new ThresholdEnforcementHandler(
11         emf, emfByTenantIdMap, thresholdDao, incidentDao, alarmDao, alertEmailHandler);
12
13     ...
14 }
```

...need many parameters in constructor.

Constructor Injection

```java
public class Employee extends BaseEntity {

    public Employee(String firstName, String lastName, String email, String bankAccount) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.bankAccount = bankAccount;
    }

    ...
}
```

What's injected is configured somewhere...

(in code, XML-file, etc...)

Property Injection

```java
public class EmployeeController implements Serializable {

    @Inject
    private ViewNavigationHandler view;

    @Inject
    private EmployeeRepository repo;

    @Inject
    private EmployeeService svc;

    @Inject
    private FacesContext faces;

    ...

}
```

Configuration Injection

```java
@ApplicationScoped
public class SomeRandomService
{
    @Inject
    @ConfigProperty(name = "endpoint.poll.interval")
    private Integer pollInterval;

    @Inject
    @ConfigProperty(name = "endpoint.poll.servername")
    private String pollUrl;

    ...
 }
```

What to inject?

WELD - Autoscan with beans.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://java.sun.com/xml/ns/javaee"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:weld="http://jboss.org/schema/weld/beans"
       xsi:schemaLocation="http://java.sun.com/xml/ns/javaee \
       http://java.sun.com/xml/ns/javaee/beans_1_0.xsd">
    <weld:scan>
        <weld:exclude name="org.apache.deltaspike.cdise.weld.WeldContainerControl"/>
    </weld:scan>
</beans>
```

In tests specify with test-runner...

https://dzone.com/articles/weld-junit-easy-testing-of-cdi-beans

Modules

- Overview of DeltaSpike Modules
- Core
- Configuration
- Bean Validation
- Container Control
- Data
- JPA
- JSF
- Partial-Bean
- Proxy
- Scheduler
- Security
- Servlet
- Test-Control