Dependency Injection in JEE7

Stefan KLIKOVITS

UNIVERSITÉ
DE GENÈVE

Why?

- use generic code, well-defined interfaces
- choose between different implementations
- to reconfigure without changing the code

How

- @Inject
- Constructor
- Setter
- Field

Different Implementations

- @Default
- @Alternative

Changing the config

- beans.xml

```
<alternatives>
  <class>ch.unige.pinfo.bakery.baker.GermanBaker</class>
  <class>ch.unige.pinfo.bakery.oven.GoodOven</class>
</alternatives>
```

Interceptors

Stefan KLIKOVITS

UNIVERSITÉ
DE GENÈVE

Why?

- separate concerns
- add logging, security, verification
- (add/modify functionality afterwards)

How?

- Define a class, annotate a method:
  - @AroundInvoke
  - @PostConstruct
  - @AroundConstruct
  - …
- Annotate the method/class that should be intercepted
  - @Interceptors(MyInterceptor.class)

Specify the interceptor class directly

1. Define interception

```
@AroundInvoke
public Object interceptorMethod(InvocationContext ictx) throws Exception{
    return ictx.proceed();
}
```

2. Annotate the Entity to be intercepted

```
@Interceptors(LoggingInterceptor.class)
public void setDish(IDish dish) {
```

Define annotation

1. Define Annotation

```
@Inherited
@InterceptorBinding
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.METHOD, ElementType.TYPE})
public @interface Logged {
}
```

2. Annotate Interceptor

```
@Logged
@Interceptor
public class LoggingInterceptor {
```

3. Activate in beans.xml

```
<interceptors>
 <class>my.interceptor.Class</class>
</interceptors>
```

Suggested Exercise

- Base project on Github/Exercises/DI-Bakery
- create an Interceptor
- intercept a method call (AroundInvoke)