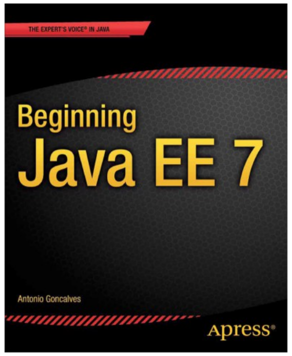
CDI : How do I ?

by antonio goncalves
@agoncal

Antonio Goncalves

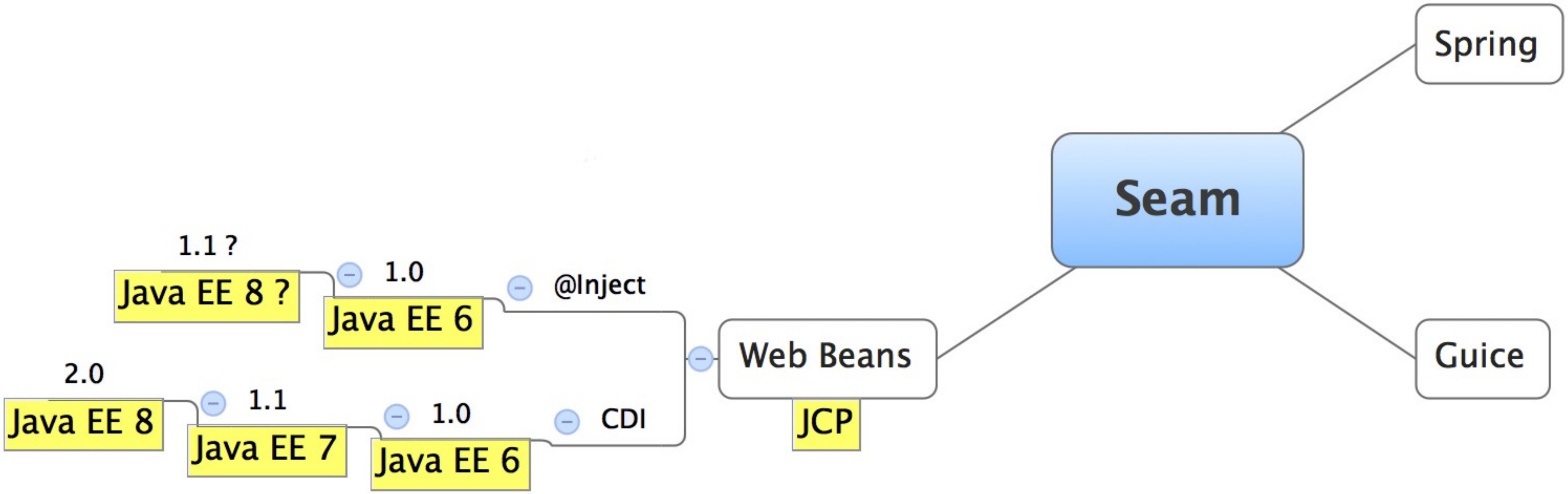


What is CDI ?

What is CDI ?

- Dependency injection
- Lose coupling, strong typing
- Context management
- Interceptors and decorators
- Event bus
- Extensions

History of CDI



Implementations

Weld

OpenWebBeans

WildFly

as 7

Apache TomEE™

ORACLE®
WEBLOGIC

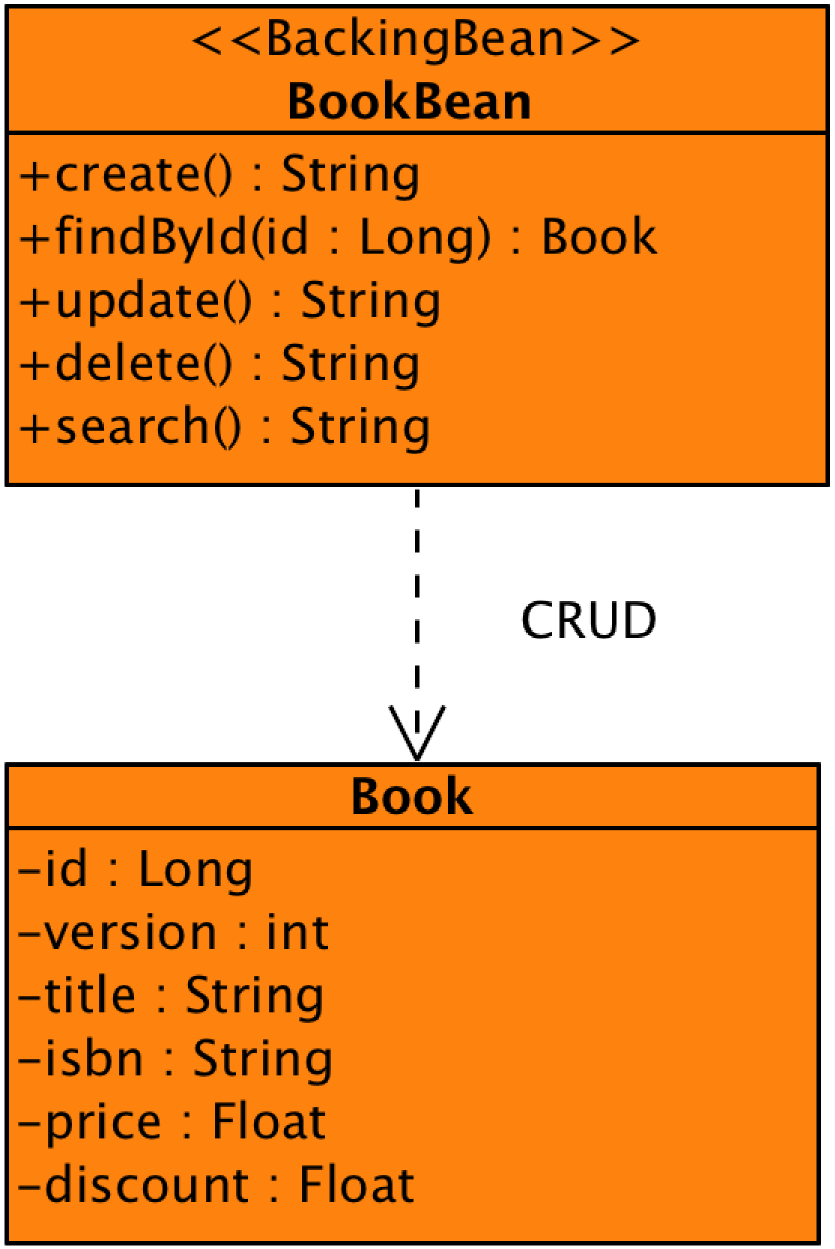
IBM®
WebSphere.

Demo
-
Creating a Web App

Demos with JBoss Forge

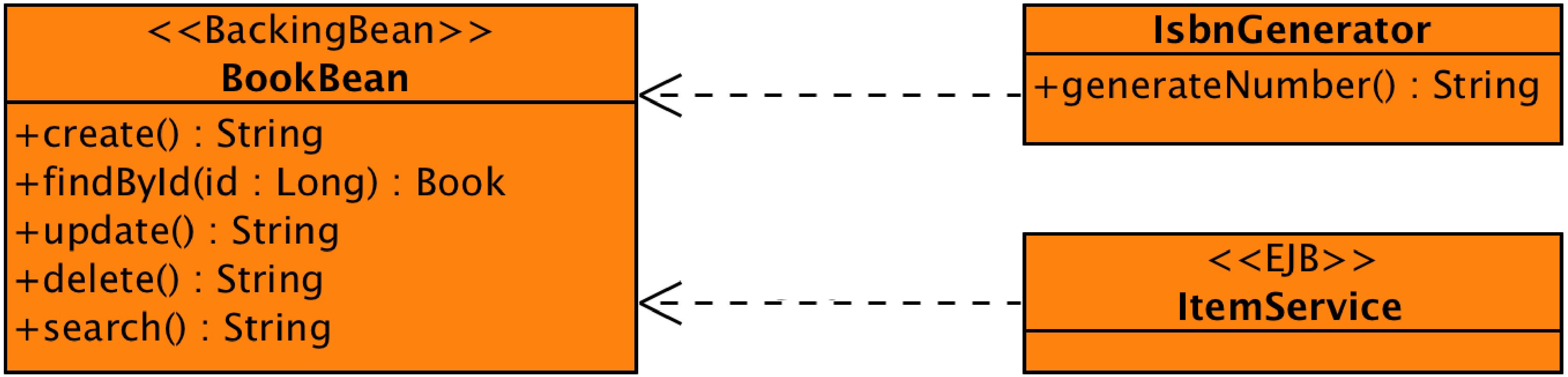
- Java EE scaffolding tool
- Shell commands
- CRUD application
- Gets you start quickly
- Takes care of integration
- Plugin based

Demo: Creating a Web App

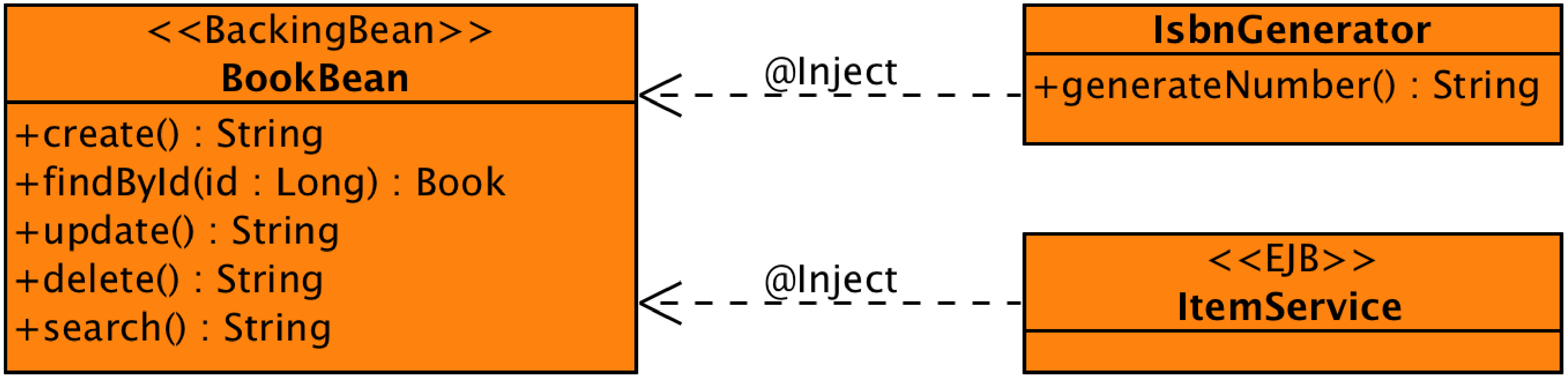


Dependency Injection

How Do I ?


<<BackingBean>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
IsbnGenerator
+generateNumber() : String
<<EJB>>
ItemService

Use @Inject !


<<BackingBean>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
IsbnGenerator
+generateNumber() : String
@Inject
<<EJB>>
ItemService

@Inject on Attributes

```java
public class BookBean implements Serializable {

   @Inject
   private NumberGenerator numberGenerator;

   @Inject
   private ItemService itemService;


   // ...
}
```

@Inject on Constructor

```java
public class BookBean implements Serializable {

   private NumberGenerator numberGenerator;
   private ItemService itemService;

   @Inject
   public BookBean(NumberGenerator numberGenerator,
                                       ItemService srv){
      this.numberGenerator = numberGenerator;
      this.itemService = srv;
   }

   // ...
}
```

@Inject on Setters

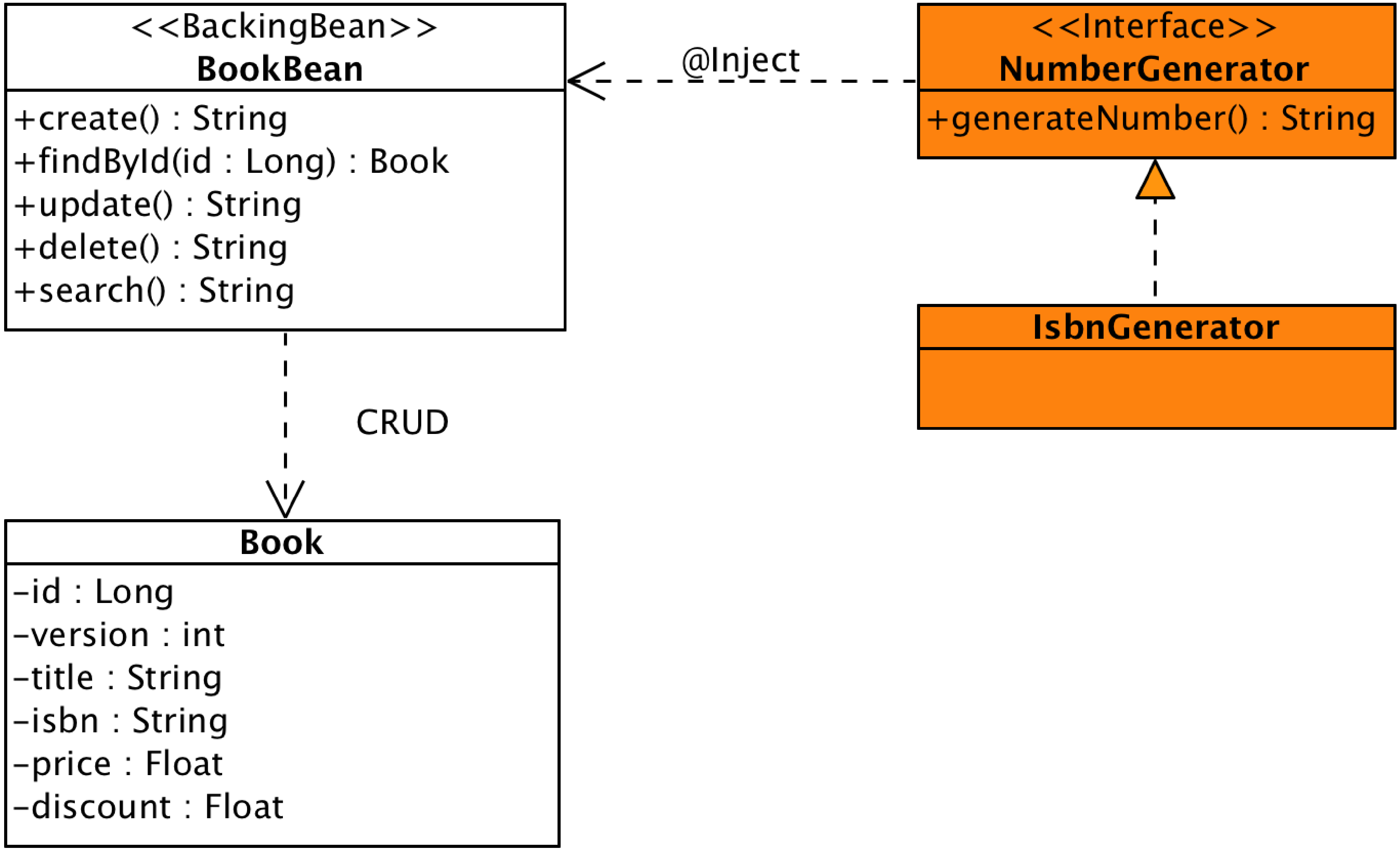
```
public class BookBean implements Serializable {

   private NumberGenerator numberGenerator;
   private ItemService itemService;

   @Inject
   public void setNumberGenerator(NumberGenerator numGen){
      this.numberGenerator = numGen;
   }

   @Inject
   public void setItemService(ItemService itemService) {
      this.itemService = itemService;
   }
   // ...
}
```

Activate CDI

- In CDI 1.0 `beans.xml` in archive
- Since CDI 1.1 it's activated by default
  - All classes having a bean definition annotation
  - `beans.xml` to deactivate or activate all
- Archive vs Bean archive

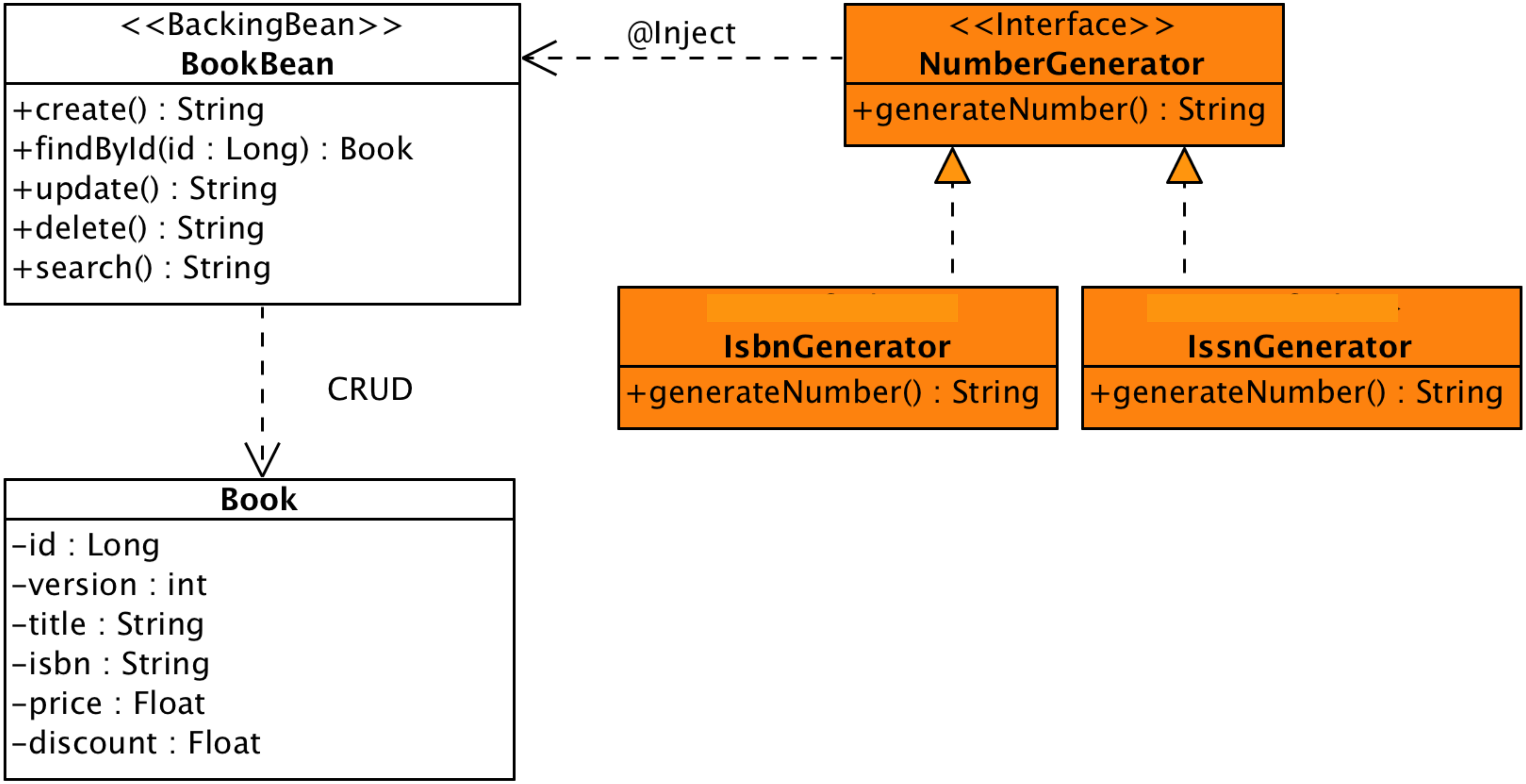
Demo
-
@Inject

Demo: @Inject One Implementation

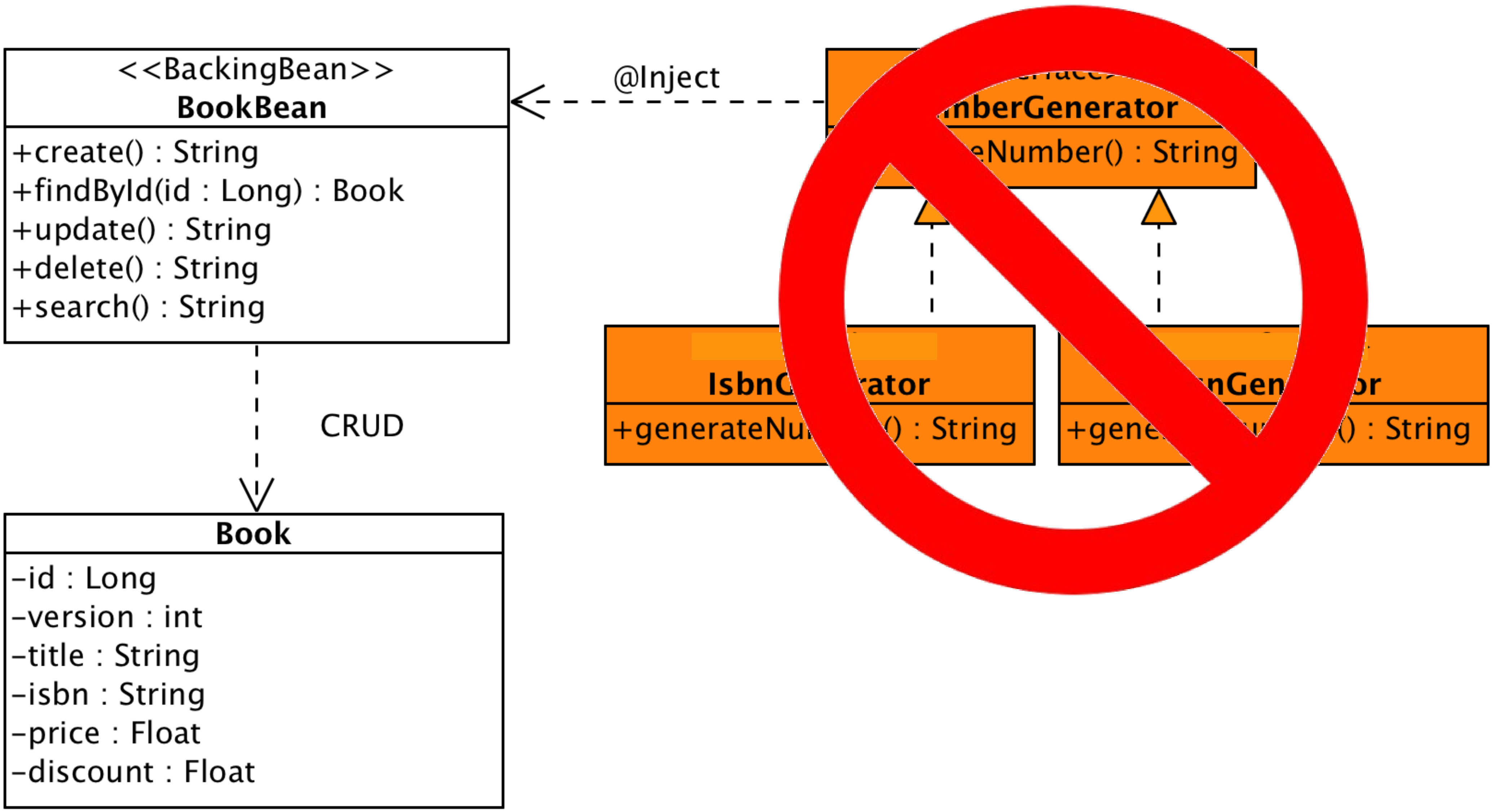


Qualifiers

How Do I ?

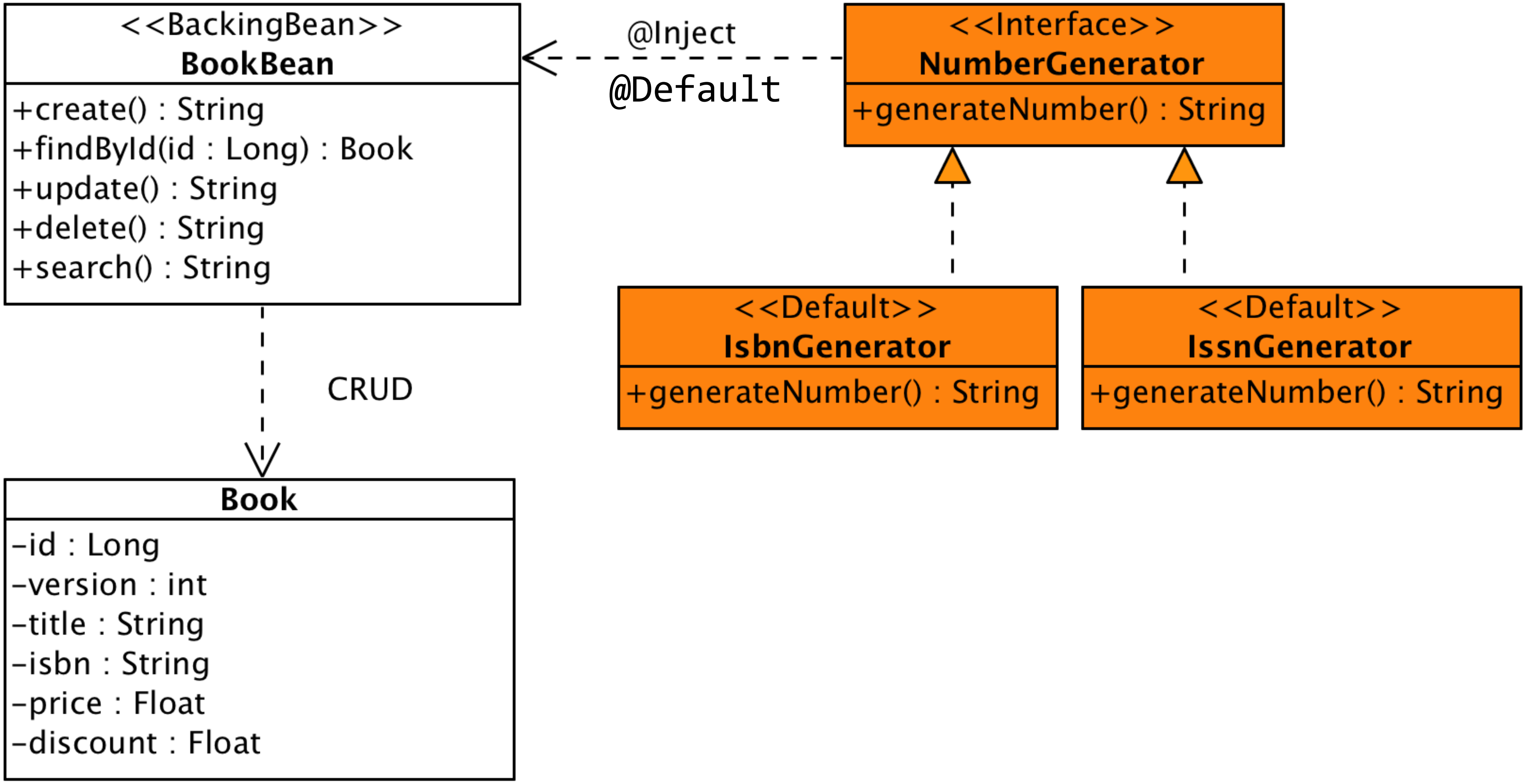

<<BackingBean>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
<<Interface>>
NumberGenerator
+generateNumber() : String
IsbnGenerator
+generateNumber() : String
IssnGenerator
+generateNumber() : String
CRUD
Book
-id : Long
-version : int
-title : String
-isbn : String
-price : Float
-discount : Float

How Do I ?



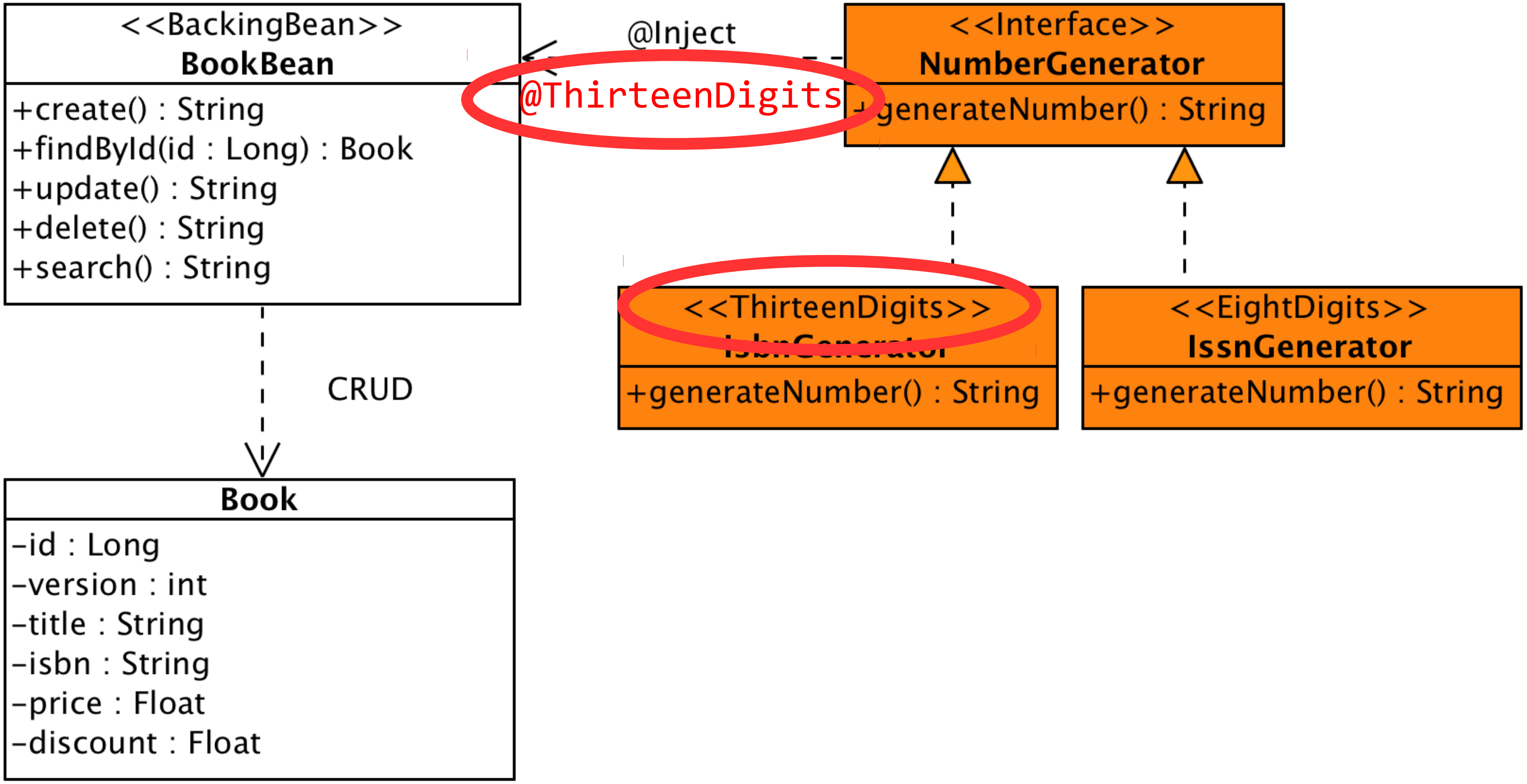
<<BackingBean>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
CRUD
Book
-id : Long
-version : int
-title : String
-isbn : String
-price : Float
-discount : Float
nberGenerator
eNumber() : String
IsbnG rator
+generateNu () : String
nGen or
+gene () : String

How Do I ?

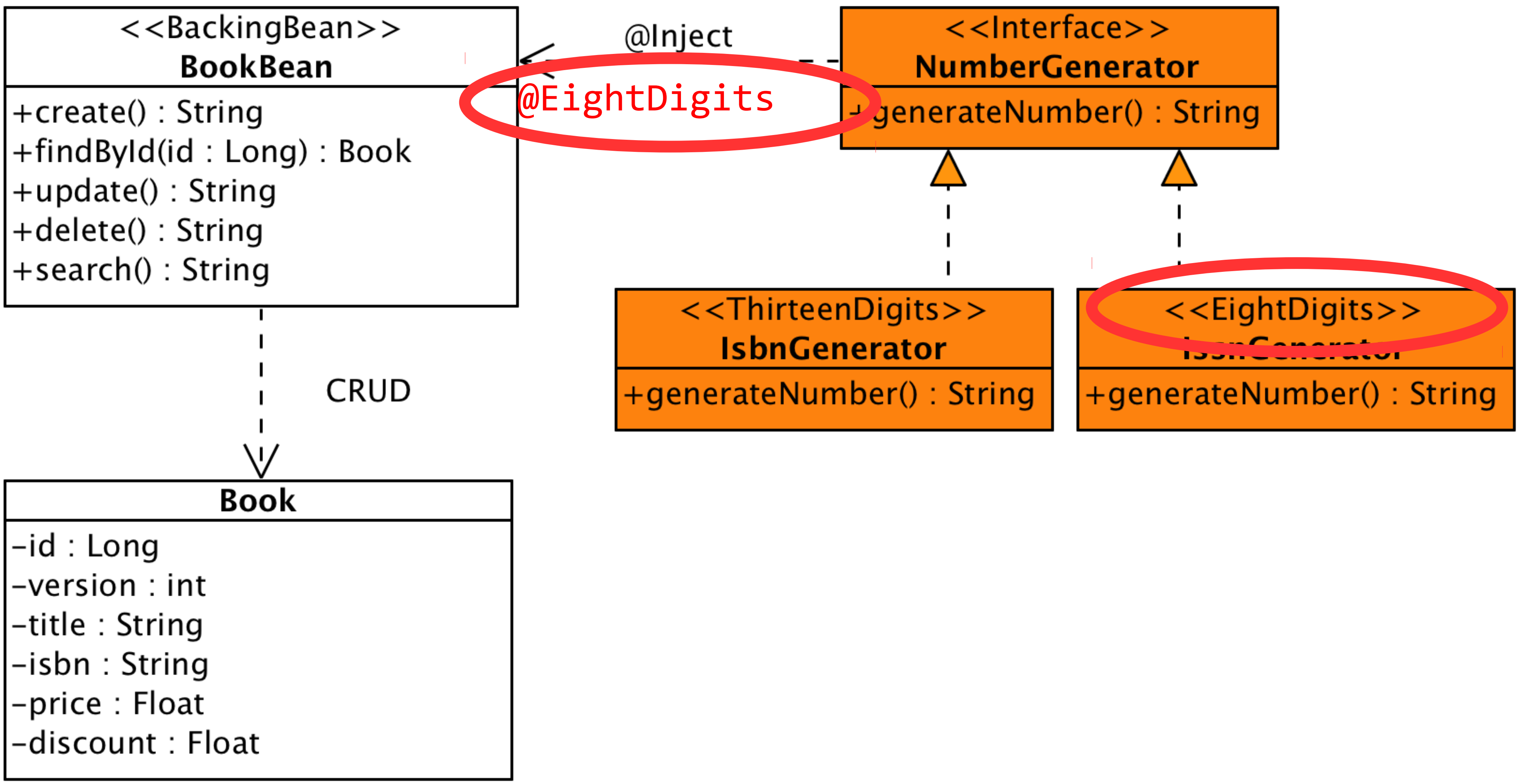


<<BackingBean>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
@Default
<<Interface>>
NumberGenerator
+generateNumber() : String
<<Default>>
IsbnGenerator
+generateNumber() : String
<<Default>>
IssnGenerator
+generateNumber() : String
CRUD
Book
-id : Long
-version : int
-title : String
-isbn : String
-price : Float
-discount : Float

Use Qualifiers !



Use Qualifiers !



A Qualifier

```java
@Qualifier
@Retention(RUNTIME)
@Target({ METHOD, FIELD, PARAMETER, TYPE })
@Documented
public @interface ThirteenDigits {
}
```

Qualifying an Injection Point

```java
public class BookBean implements Serializable {

    @Inject @ThirteenDigits
    private NumberGenerator numberGenerator;

    @Inject
    private ItemService itemService;


    // ...
}
```

Qualifying an Injection Point

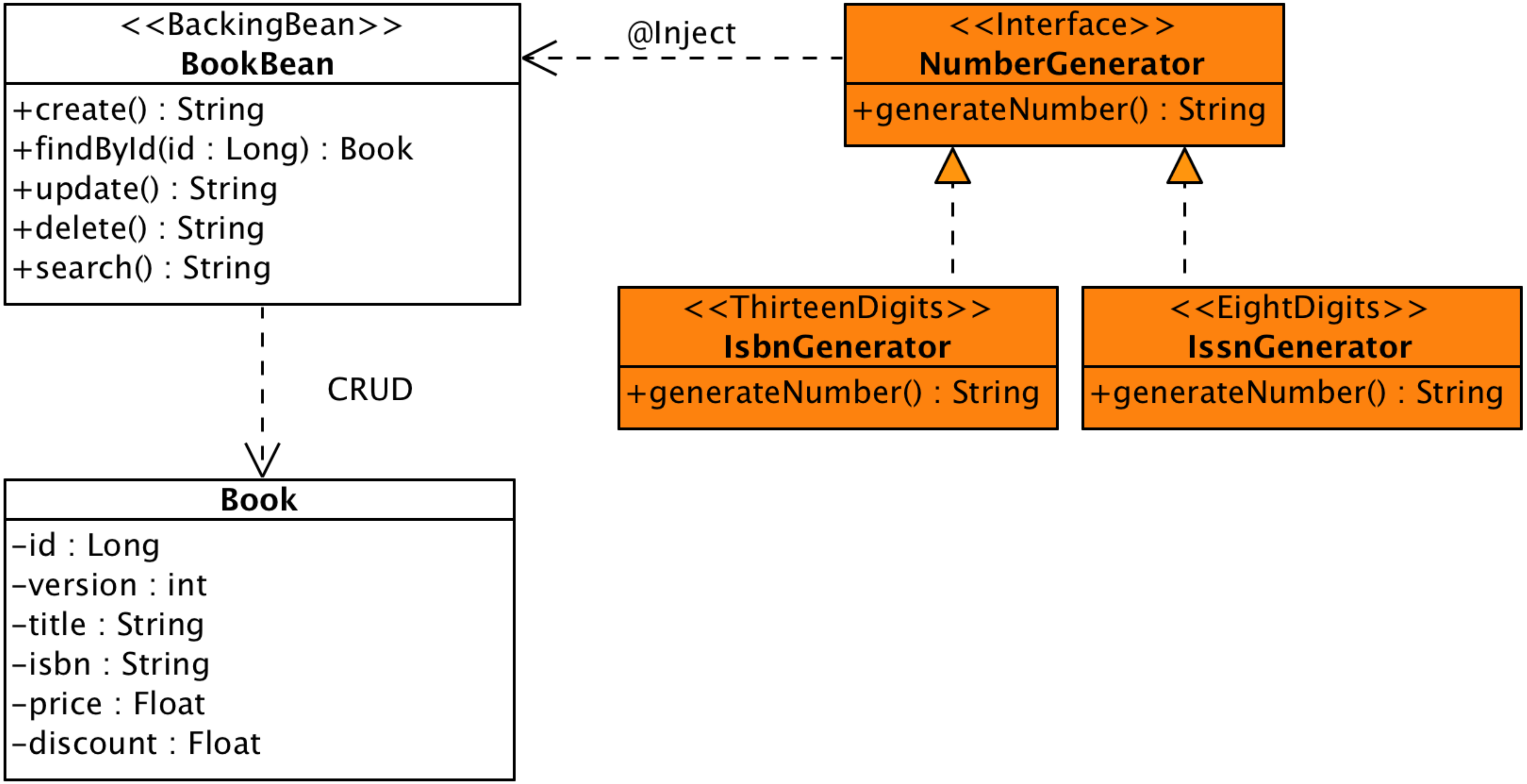
```
public class BookBean implements Serializable {

   @Inject @ThirteenDigits
   private NumberGenerator numberGenerator;

   @Inject @Default
   private ItemService itemService;


   // ...
}
```

Qualifying a Bean

```java
@ThirteenDigits
public class IsbnGenerator implements NumberGenerator {

  @Override
  public String generateNumber() {
    return "13-" + Math.abs(new Random().nextInt());
  }
}
```

Demo
-
Qualifiers

Demo: @Inject One Implementation

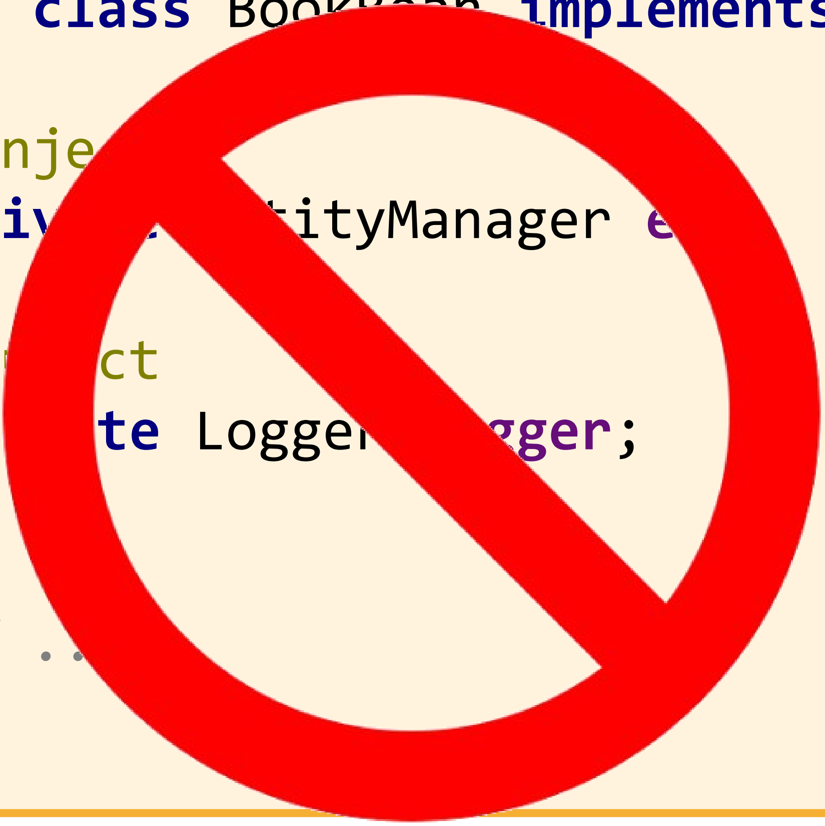


Producers

How Do I ?

```
public class BookBean implements Serializable {

    @Inject
    private EntityManager em;

    @Inject
    private Logger logger;


    // ...
}
```

How Do I ?



```
public class BookBean implements Serializable {

    @Inje
    priv      tityManager e

    @I     ct
    pr    te Logge     gger;


    // ..
}
```

How Do I ?

```
public class BookBean implements Serializable {

    @Inject
    private EntityManager em;

    @Inject
    private Logger logger;


    // ...
}
```

Several persistence units

```
@PersistenceContext(unitName = "myPU1")
@PersistenceContext(unitName = "myPU2")
```

How Do I ?

```
public class BookBean implements Serializable {

    @Inject
    private EntityManager em;

    @Inject
    private Logger logger;


    // ...
}
```

Third party framewok

Use Producers !

```
public class BookBean implements Serializable {

    @Inject
    private EntityManager em;

    // ...
}
```

```
public class ResourceProducer {

    @Produces
    @PersistenceContext(unitName = "myPU")
    private EntityManager entityManager;

}
```

Use Producers !

```java
public class BookBean implements Serializable {

    @Inject
    private Logger logger;

    // ...
}
```

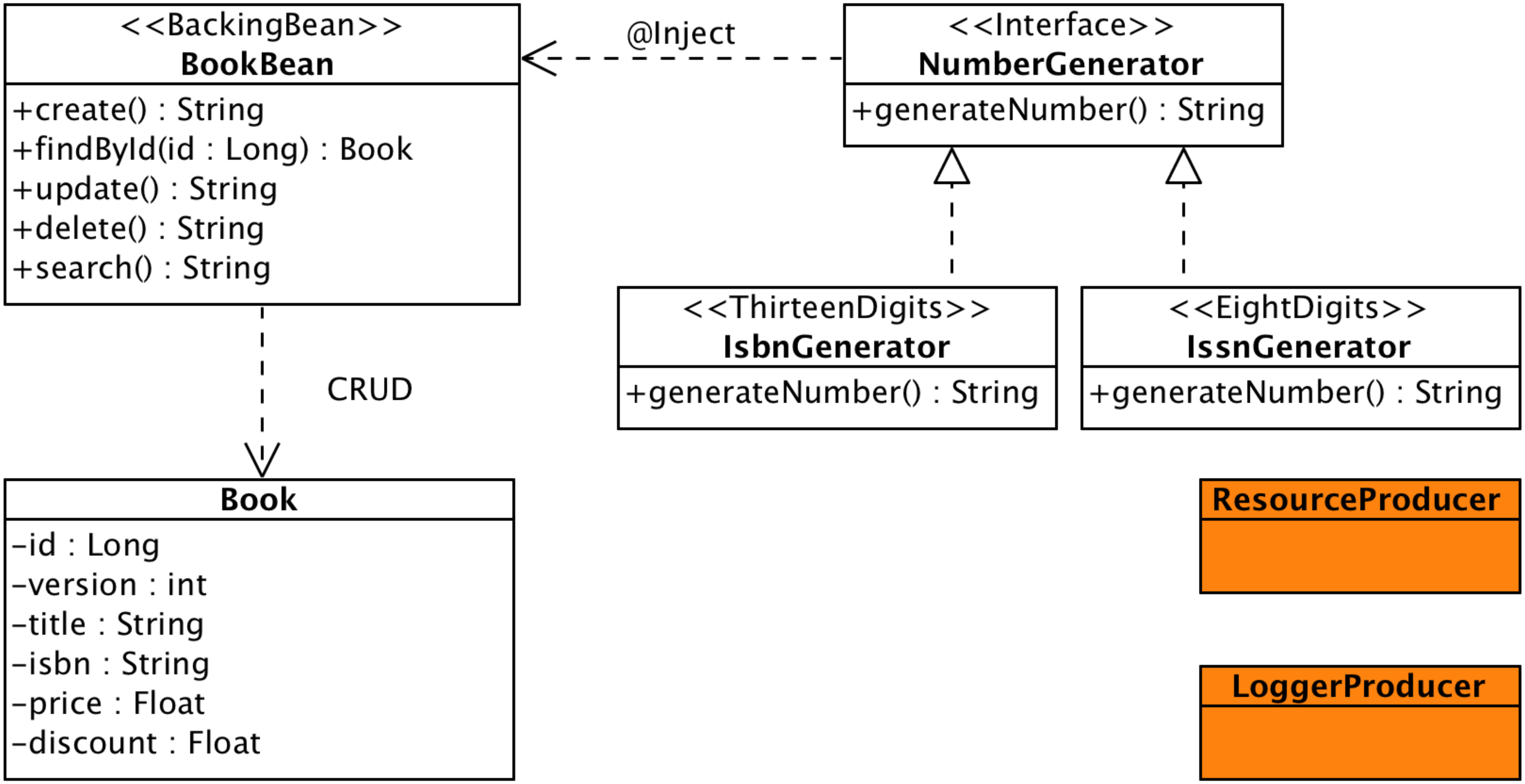
```java
public class ResourceProducer {

    @Produces
    private Logger produceLogger(InjectionPoint ip) {
     return
     Logger.getLogger(ip.getMember().getDeclaringClass().getName());
    }
}
```

Demo

-

Producers

Demo: Producers

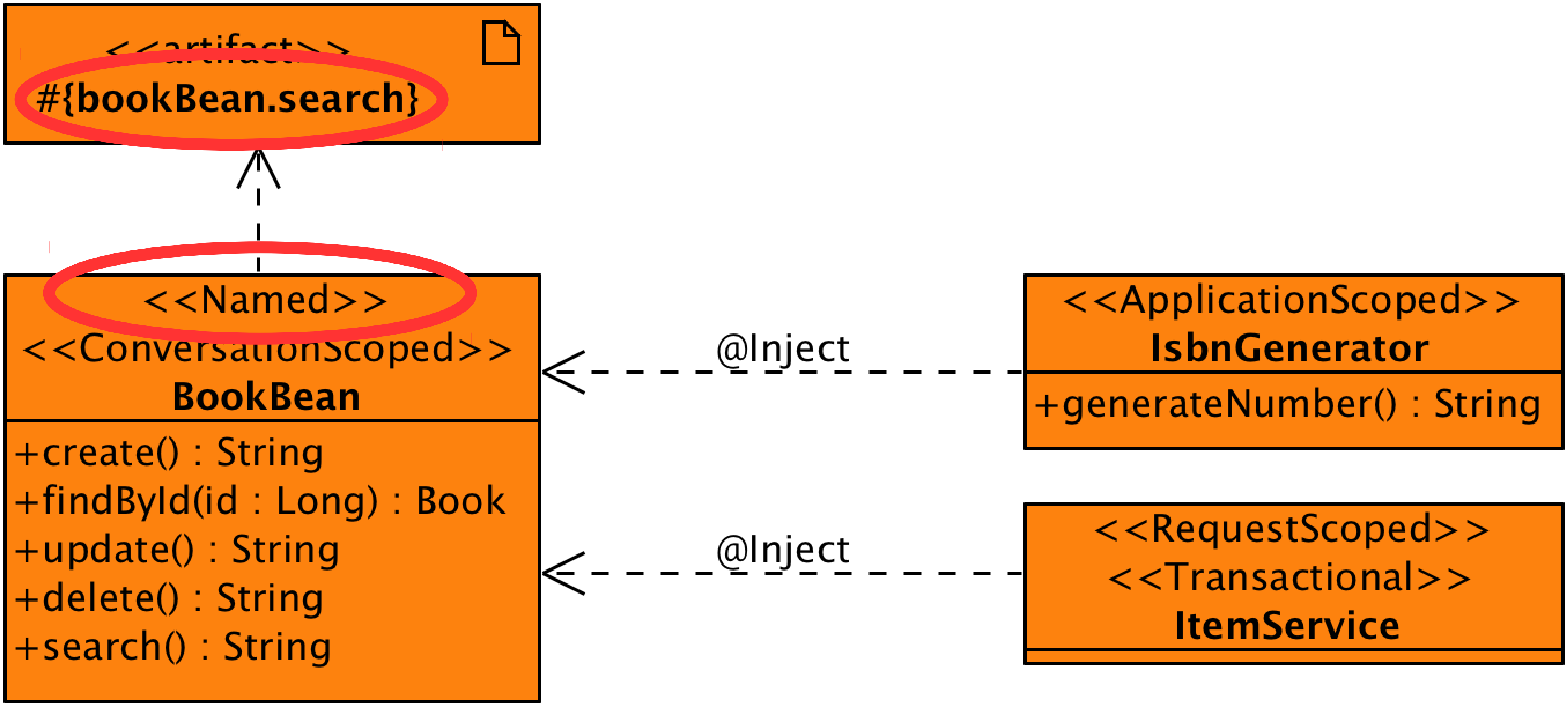


Web tier
&
Service tier

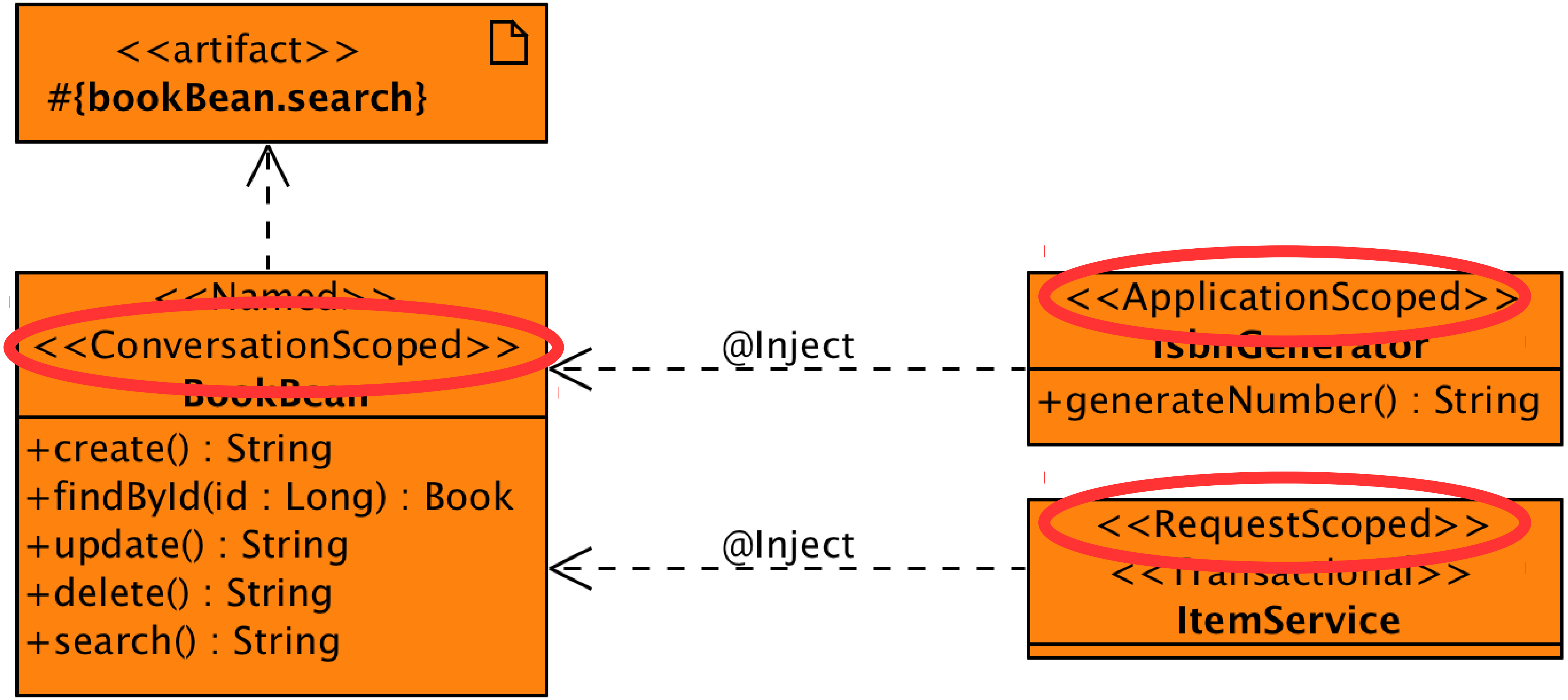
How Do I ?

How Do I ?

Use Expression Language...



Use Expression Language and Scopes !



Service Tier

```
@Transactional
public class BookBean implements Serializable {

   @Inject
   private EntityManager em;

   public void update() {
      em.persist(book);
   }
}
```

Service Tier + Web Tier

```java
@Named

@Transactional
public class BookBean implements Serializable {

   @Inject
   private EntityManager em;

   public void update() {
      em.persist(book);
   }
}
```

```xml
<h:commandLink value="Create"
               action='#{bookBean.update}'/>
```

Service Tier + Web Tier

```
@Named("service")

@Transactional
public class BookBean implements Serializable {

    @Inject
    private EntityManager em;

    public void update() {
        em.persist(book);
    }
}
```

```
<h:commandLink value="Create"
               action='#{service.update}'/>
```

Several scopes

- `@Dependent` (default)
- `@ApplicationScoped`, `@SessionScoped`, `@RequestScoped`
- `@ConversationScoped`
- Create your own
  - `@TransactionalScoped`
  - `@ViewScoped`
  - `@ThreadScoped`
  - `@ClusterScoped`

Just choose the right scope

```java
@Named
@RequestScoped
@Transactional
public class BookBean implements Serializable {




    public void update() {

    }

    public void delete() {

    }
}
```

Just choose the right scope

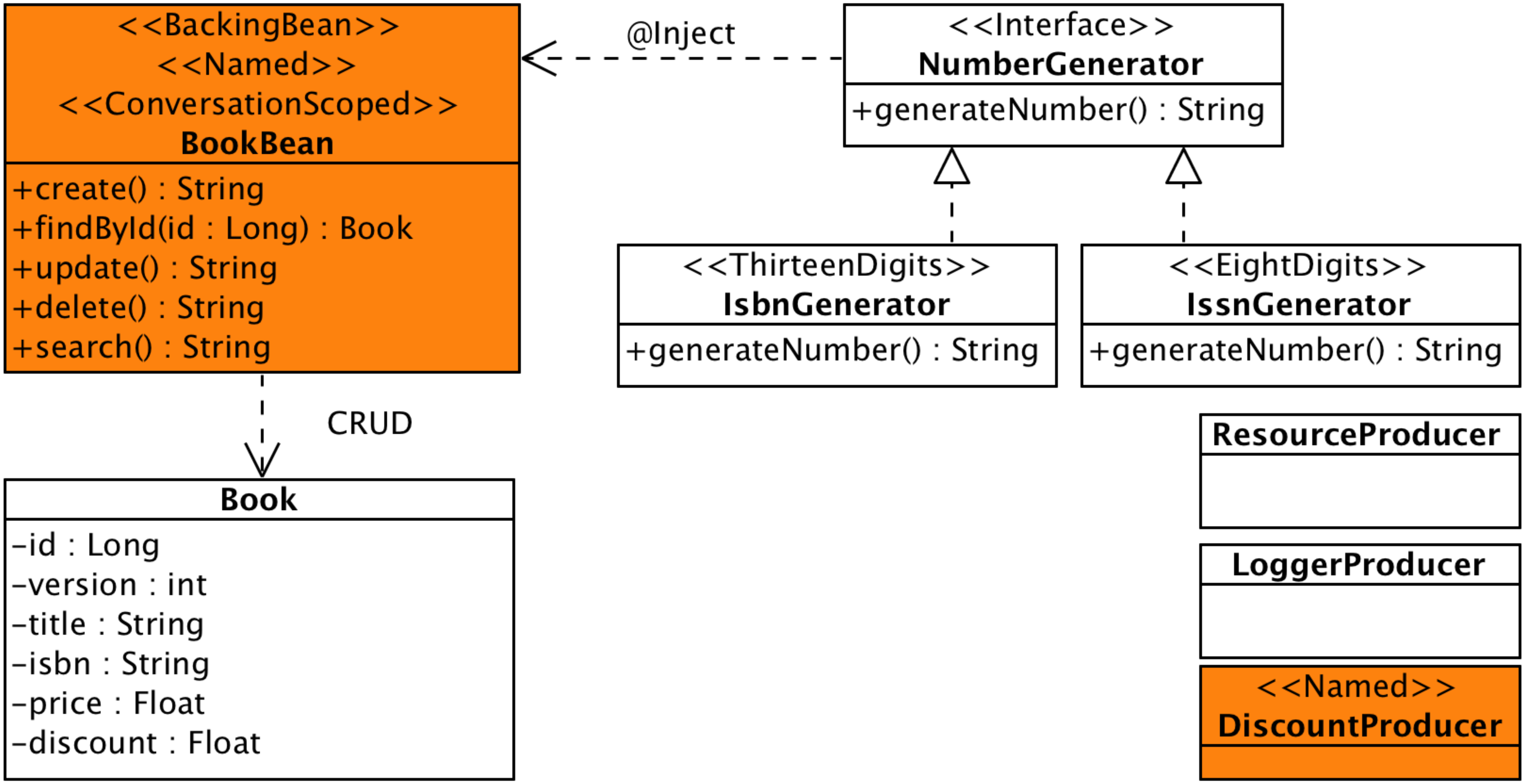
```
@Named
@SessionScoped
@Transactional
public class BookBean implements Serializable {





   public void update() {

   }

   public void delete() {

   }
}
```

Just choose the right scope

```java
@Named
@ConversationScoped
@Transactional
public class BookBean implements Serializable {

    @Inject
    private Conversation conversation;

    public void update() {
        conversation.begin();
    }

    public void delete() {
        conversation.end();
    }
}
```

Demo
-
@Named & scope

Demo: @Named & Scope



</>

Alternatives

How Do I ?

```java
public class MockGenerator implements NumberGenerator  {

   public String generateNumber() {
      return "mock-" + Math.abs(new Random().nextInt());
   }
}
```

How Do I ?

```java
@Mock
public class MockGenerator implements NumberGenerator  {

    public String generateNumber() {
        return "mock-" + Math.abs(new Random().nextInt());
    }
}
```

```java
public class BookBean implements Serializable {

    @Inject @Mock
    private NumberGenerator numberGenerator;
    // ...
}
```

How Do I ?

```
@Mock
public class Mock                 ts NumberGenerator  {

    public Str      erateNumber() {
        retur   oc      Math.abs(new r    om().nextInt());
    }
}
```

```
pu   c class BookBe     mpleme    Serializable {

    @    ct @Mock
    pr       NumberGenerat      mberGenerator;
    // ..
}
```

Use an Alternative !

```java
@Alternative
@EightDigits
public class MockGenerator implements NumberGenerator  {

   public String generateNumber() {
      return "mock-" + Math.abs(new Random().nextInt());
   }
}
```

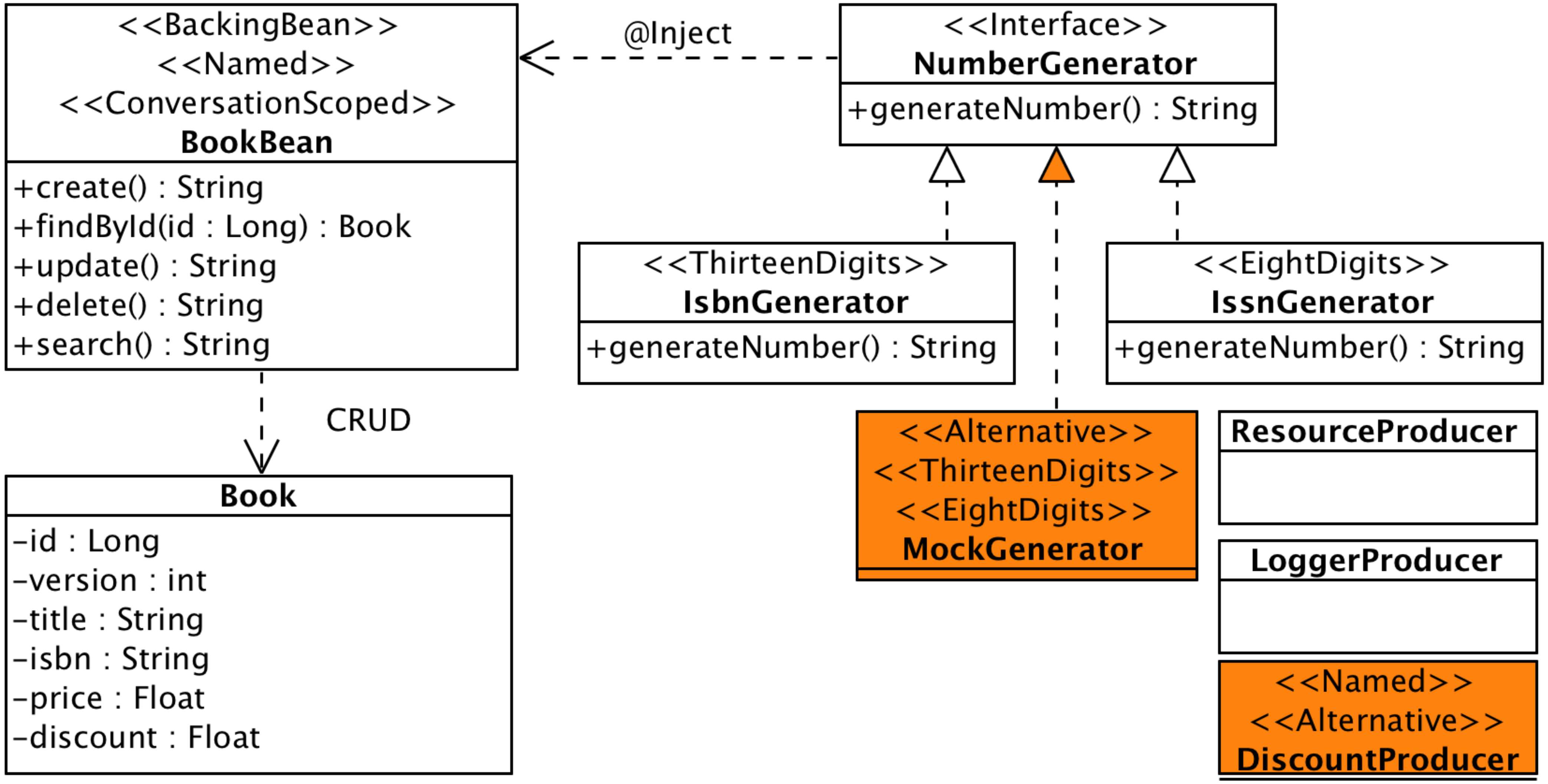
```java
public class BookBean implements Serializable {

   @Inject @EightDigits
   private NumberGenerator numberGenerator;
   // ...
}
```

Activate the Alternative

```
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
       ...
       version="1.1" bean-discovery-mode="all">

  <alternatives>
    <class>com.foo.MockGenerator</class>
  </alternatives>
</beans>
```

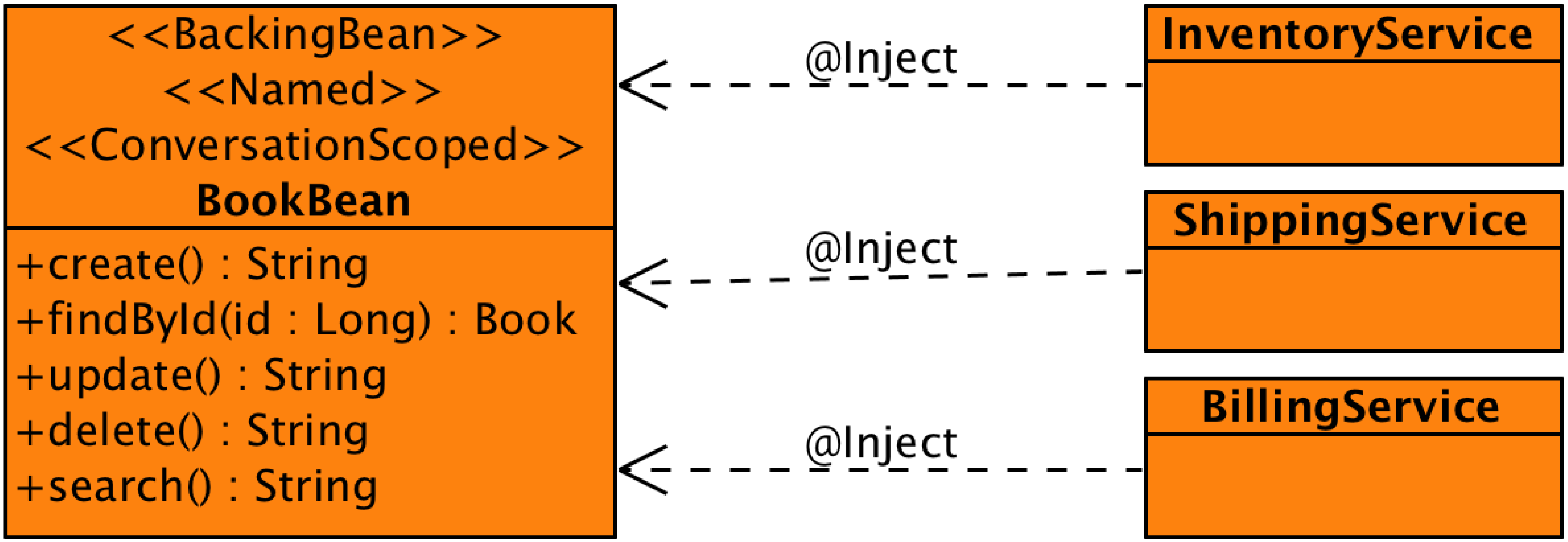
Demo
-
Alternatives

Demo: Alternatives

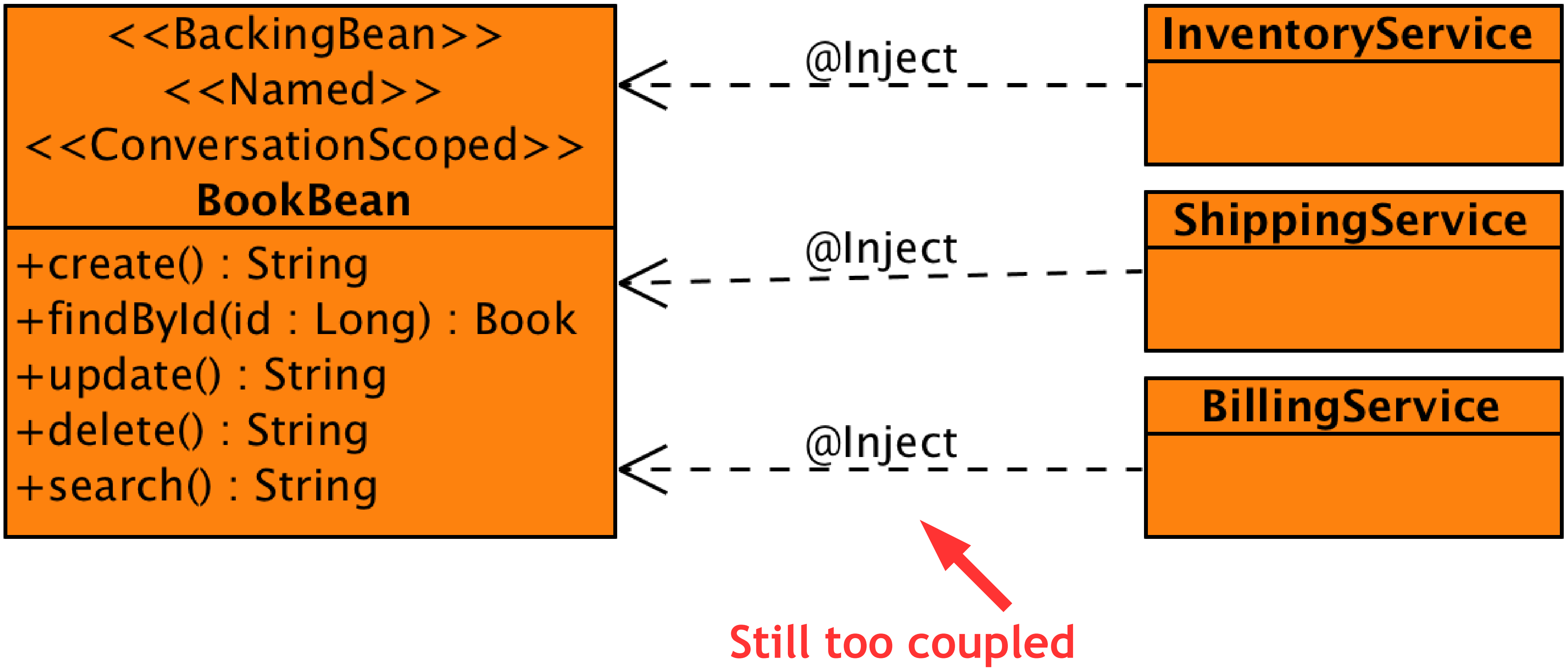


Events

How Do I ?

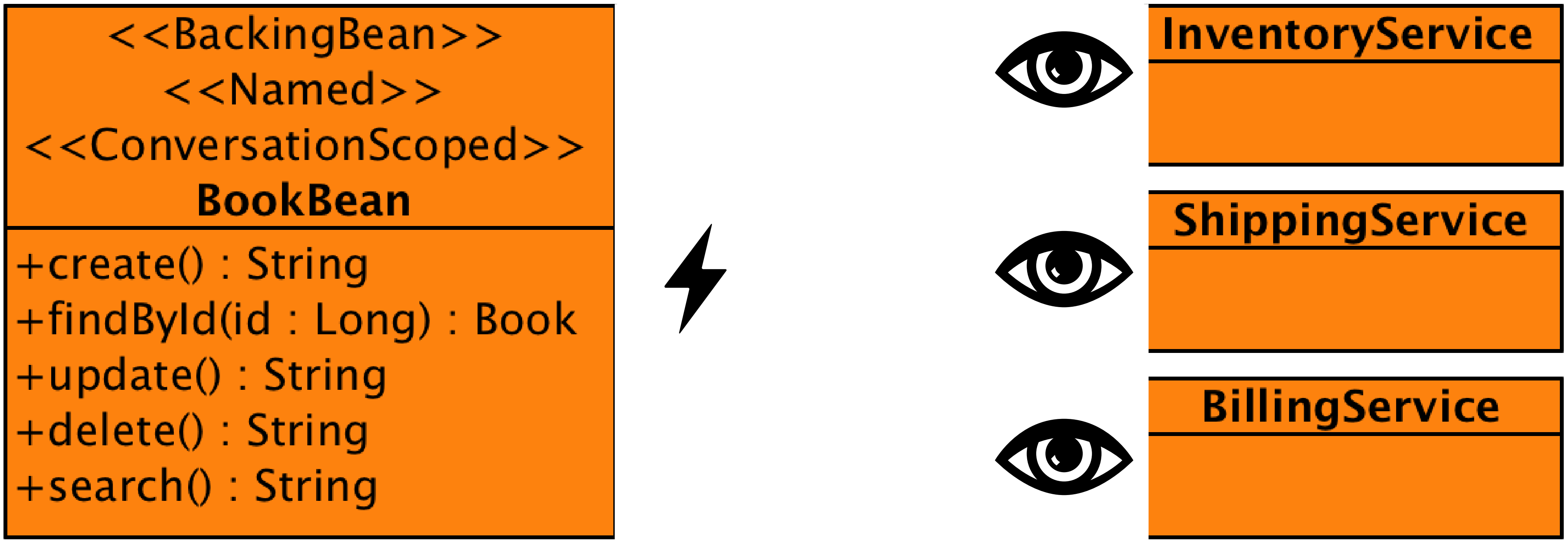

<<BackingBean>>
<<Named>>
<<ConversationScoped>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
@Inject
@Inject
InventoryService
ShippingService
BillingService

How Do I ?



<<BackingBean>>
<<Named>>
<<ConversationScoped>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
@Inject
@Inject
InventoryService
ShippingService
BillingService
Still too coupled

Use Events !


<<BackingBean>>
<<Named>>
<<ConversationScoped>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
InventoryService
ShippingService
BillingService

Fire and Observe

```java
public class BookBean implements Serializable {

   @Inject
   private Event<Book> boughtEvent;

   public void update() {
      boughtEvent.fire(book);
   }
}
```

```java
public class InventoryService {

    private void observeBooks (@Observes       Book book) {
        logger.info("Book recevied " + book.getTitle());
    }
}
```

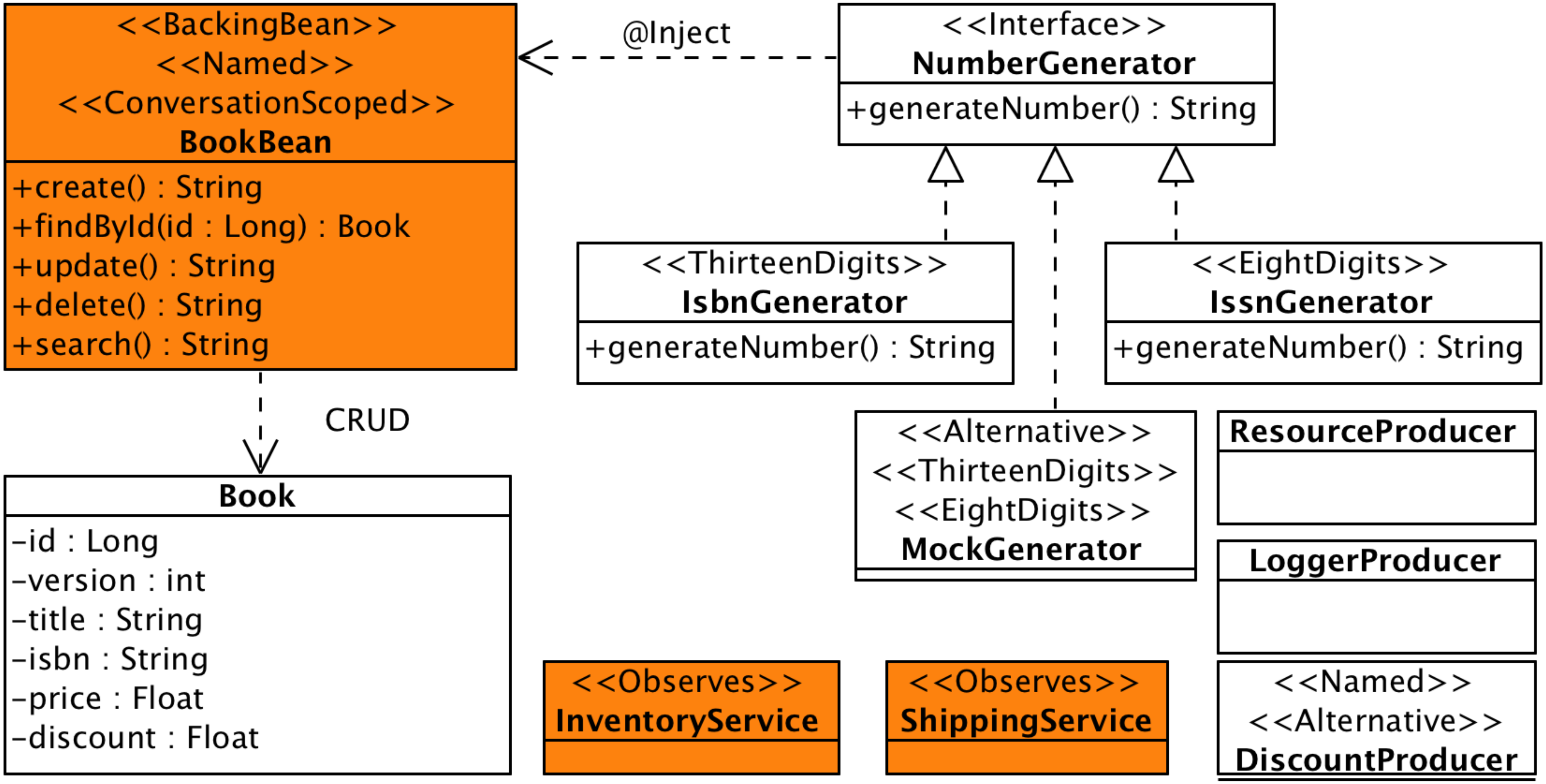
Fire and Observe with Qualifier

```
public class BookBean implements Serializable {

   @Inject @Paper
   private Event<Book> boughtEvent;

   public void update() {
      boughtEvent.fire(book);
   }
}
```

```
public class InventoryService {

    private void observeBooks (@Observes @Paper Book book) {
        logger.info("Book recevied " + book.getTitle());
    }
}
```

Demo
-
Events

Demo: Events



<<BackingBean>>
<<Named>>
<<ConversationScoped>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
<<Interface>>
NumberGenerator
+generateNumber() : String
<<ThirteenDigits>>
IsbnGenerator
+generateNumber() : String
<<EightDigits>>
IssnGenerator
+generateNumber() : String
CRUD
<<Alternative>>
<<ThirteenDigits>>
<<EightDigits>>
MockGenerator
ResourceProducer
LoggerProducer
Book
-id : Long
-version : int
-title : String
-isbn : String
-price : Float
-discount : Float
<<Observes>>
InventoryService
<<Observes>>
ShippingService
<<Named>>
<<Alternative>>
DiscountProducer

CDI : So Much More

CDI : So Much More

CDI Extension ecosystem

DELTASPIKE

CDI Course on PluralSight

pluralsight
hardcore developer training

http://www.pluralsight.com/courses/context-dependency-injection-1-1

Thanks

www.antoniogoncalves.org
antonio.goncalves@gmail.com
@agoncal
@devoxxfr
@lescastcodeurs

Q & A

Creative Commons

- **Attribution** — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **Noncommercial** — You may not use this work for commercial purposes.
- **Share Alike** — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

CDI : How do I ?

by antonio goncalves
@agoncal

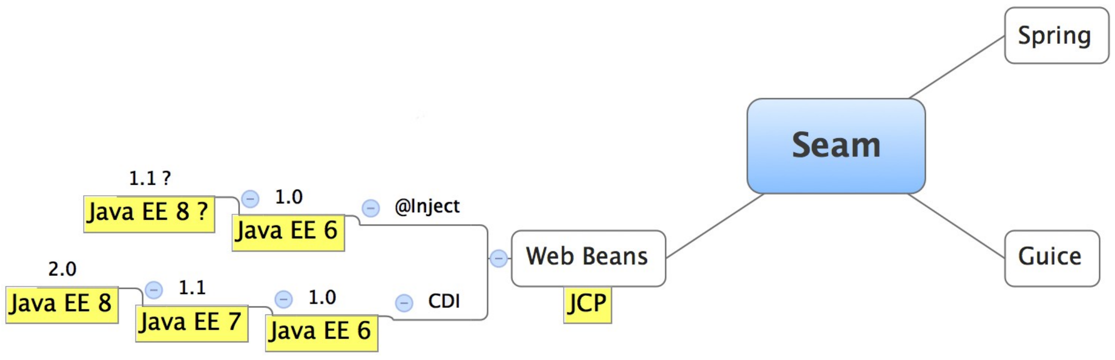
Antonio Goncalves

What is CDI ?

What is CDI ?

- Dependency injection
- Lose coupling, strong typing
- Context management
- Interceptors and decorators
- Event bus
- Extensions

History of CDI



Implementations

Weld

OpenWebBeans

WildFly

as 7

Apache TomEE™

ORACLE®
WEBLOGIC

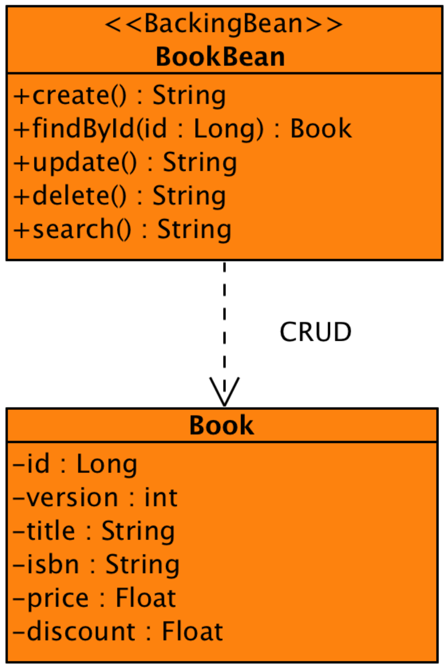
IBM®
WebSphere.

Demo
-
Creating a Web App

Demos with JBoss Forge

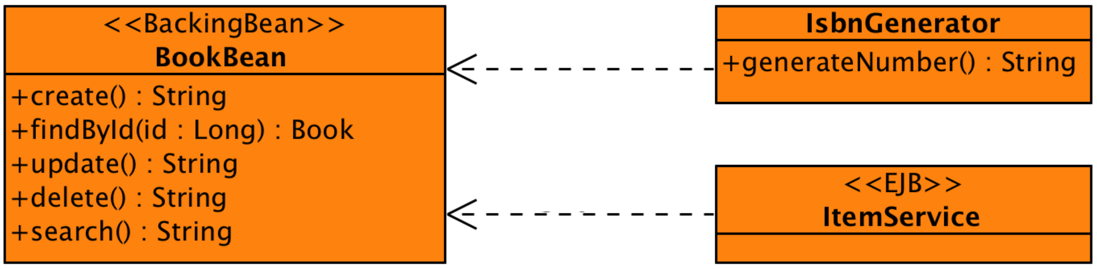
- Java EE scaffolding tool
- Shell commands
- CRUD application
- Gets you start quickly
- Takes care of integration
- Plugin based

Demo: Creating a Web App



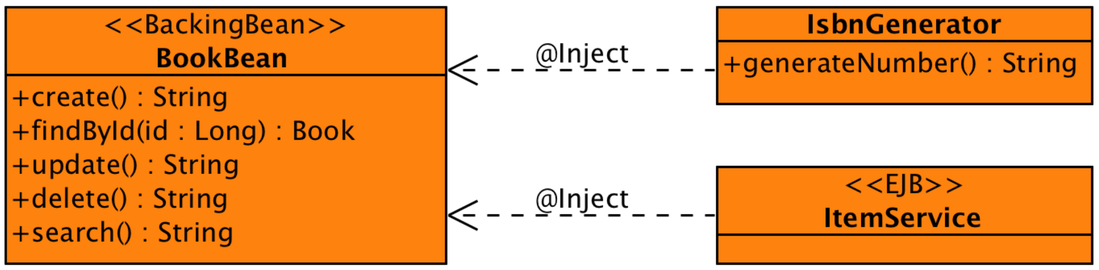
Dependency Injection

How Do I ?



<<BackingBean>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
IsbnGenerator
+generateNumber() : String
<<EJB>>
ItemService

Use @Inject !



<<BackingBean>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
IsbnGenerator
+generateNumber() : String
@Inject
<<EJB>>
ItemService

@Inject on Attributes

```
public class BookBean implements Serializable {

   @Inject
   private NumberGenerator numberGenerator;

   @Inject
   private ItemService itemService;


   // ...
}
```

@Inject on Constructor

```
public class BookBean implements Serializable {

   private NumberGenerator numberGenerator;
   private ItemService itemService;

   @Inject
   public BookBean(NumberGenerator numberGenerator,
                                   ItemService srv){
      this.numberGenerator = numberGenerator;
      this.itemService = srv;
   }

   // ...
}
```

@Inject on Setters

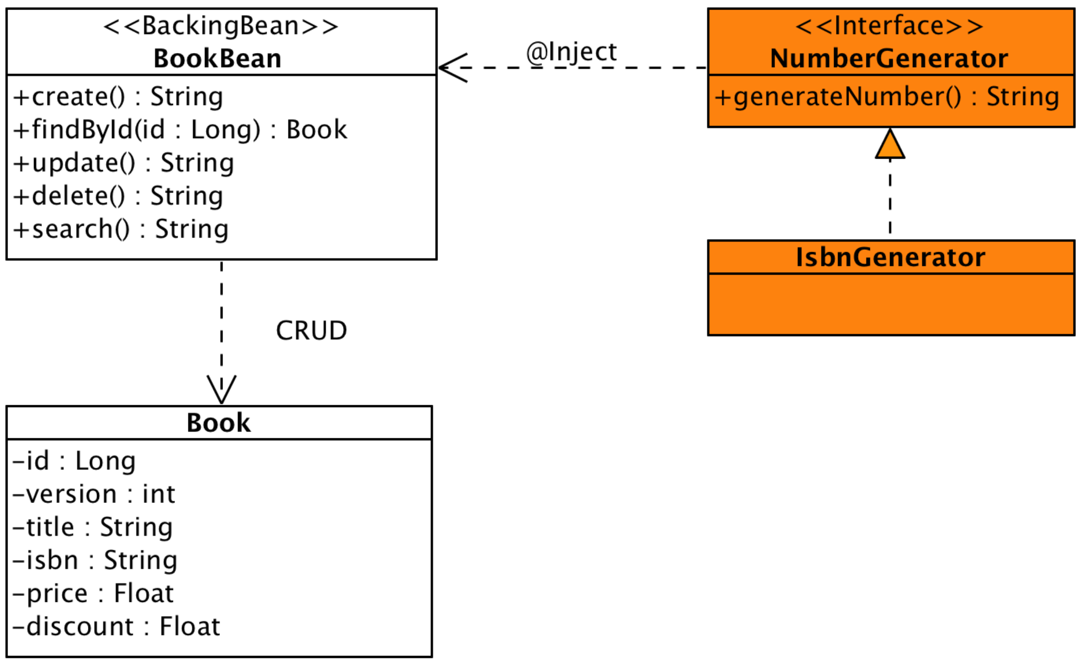
```
public class BookBean implements Serializable {

   private NumberGenerator numberGenerator;
   private ItemService itemService;

   @Inject
   public void setNumberGenerator(NumberGenerator numGen){
      this.numberGenerator = numGen;
   }

   @Inject
   public void setItemService(ItemService itemService) {
      this.itemService = itemService;
   }
   // ...
}
```

Activate CDI

- In CDI 1.0 `beans.xml` in archive
- Since CDI 1.1 it's activated by default
  - All classes having a bean definition annotation
  - `beans.xml` to deactivate or activate all
- Archive vs Bean archive

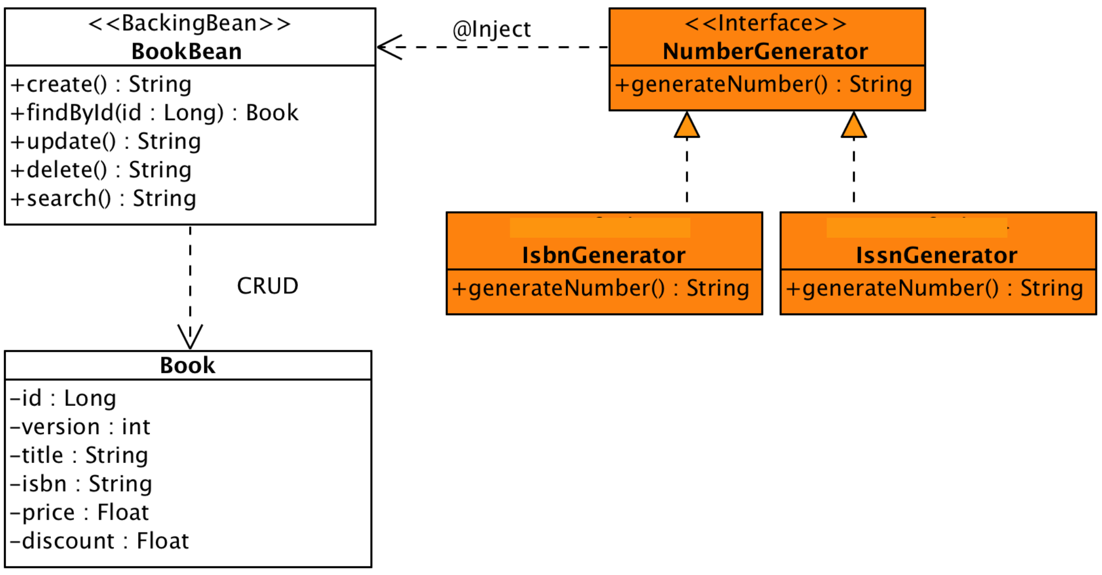
Demo
-
@Inject

Demo: @Inject One Implementation



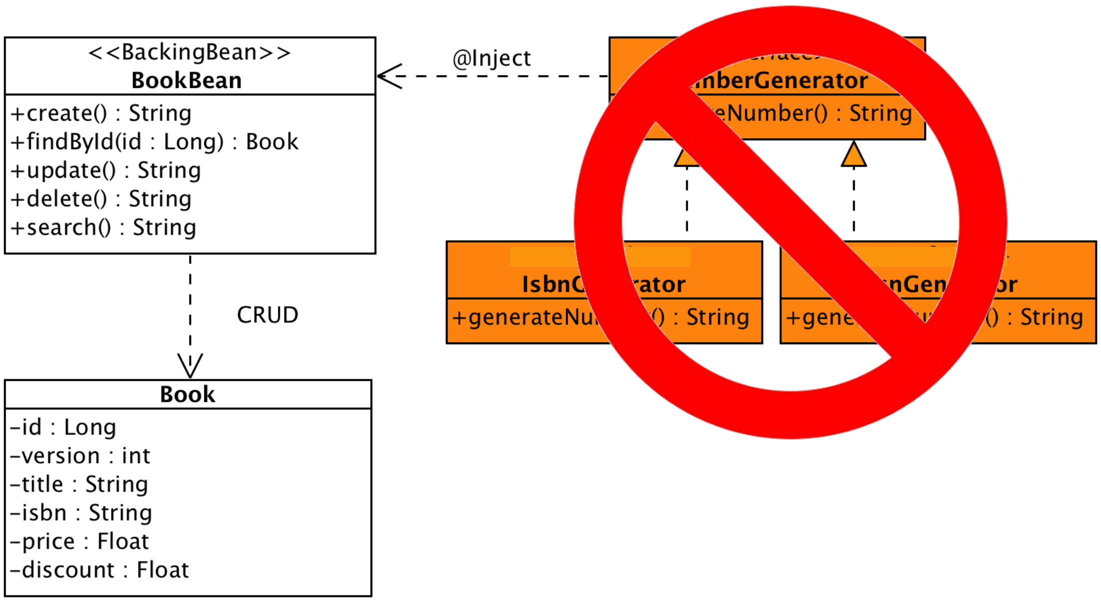
Qualifiers

How Do I ?

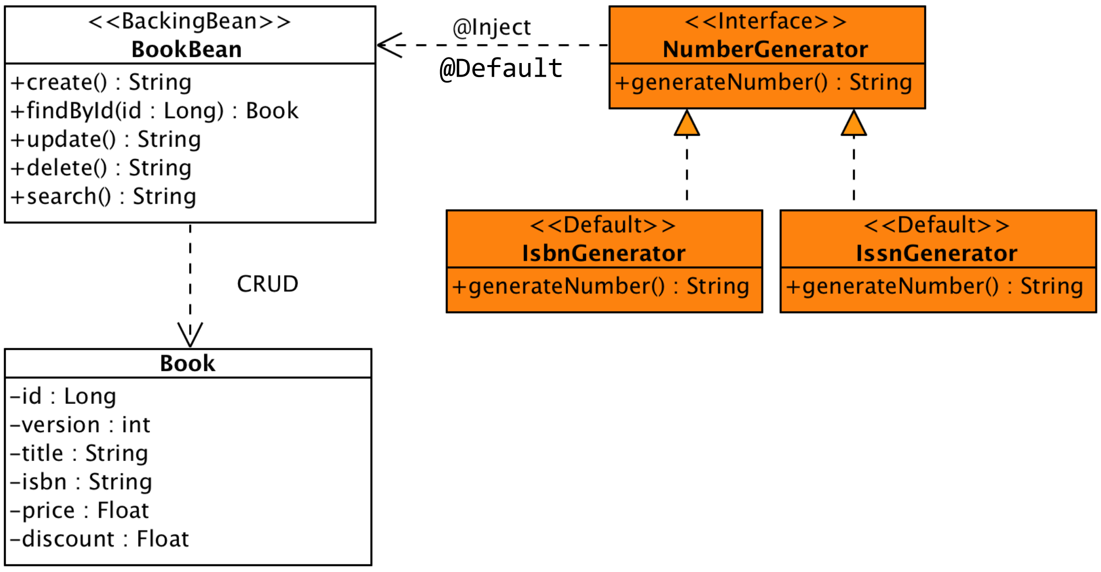


<<BackingBean>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
<<Interface>>
NumberGenerator
+generateNumber() : String
IsbnGenerator
+generateNumber() : String
IssnGenerator
+generateNumber() : String
CRUD
Book
-id : Long
-version : int
-title : String
-isbn : String
-price : Float
-discount : Float

How Do I ?

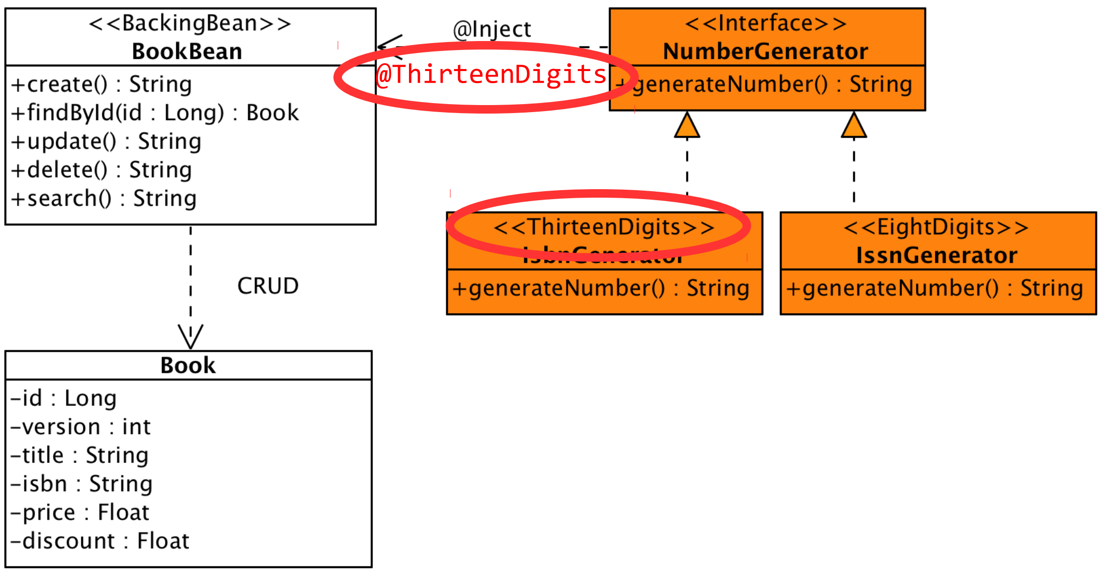

@Inject
<<BackingBean>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
CRUD
Book
-id : Long
-version : int
-title : String
-isbn : String
-price : Float
-discount : Float
nberGenerator
eNumber() : String
IsbnC rator
+generateNu () : String
nGen or
+gene () : String

How Do I ?

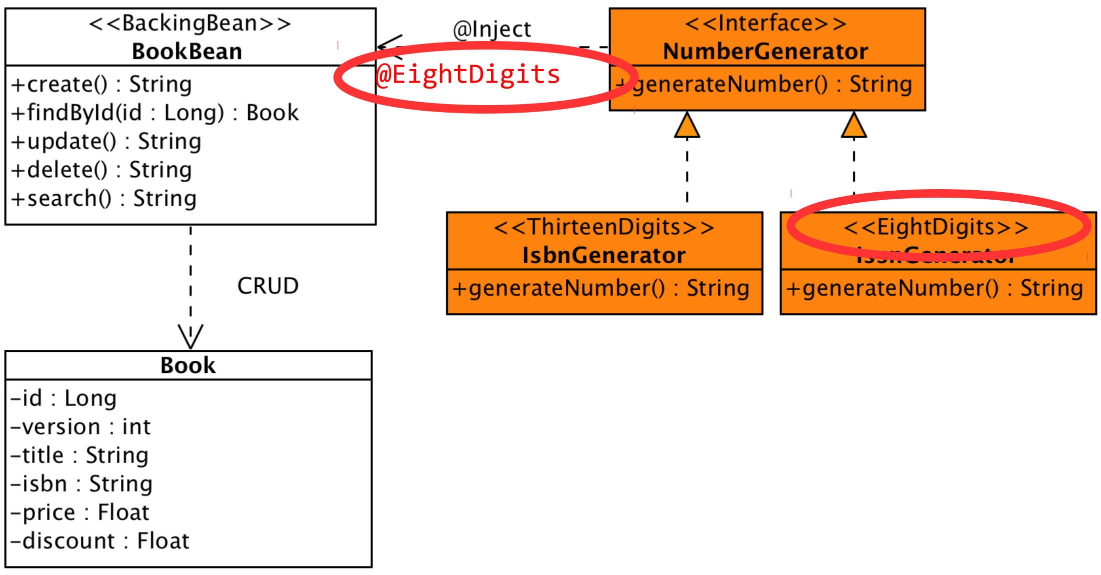


<<BackingBean>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
@Default
<<Interface>>
NumberGenerator
+generateNumber() : String
<<Default>>
IsbnGenerator
+generateNumber() : String
<<Default>>
IssnGenerator
+generateNumber() : String
CRUD
Book
-id : Long
-version : int
-title : String
-isbn : String
-price : Float
-discount : Float

Use Qualifiers !



Use Qualifiers !



A Qualifier

```
@Qualifier
@Retention(RUNTIME)
@Target({ METHOD, FIELD, PARAMETER, TYPE })
@Documented
public @interface ThirteenDigits {
}
```

Qualifying an Injection Point

```
public class BookBean implements Serializable {

    @Inject @ThirteenDigits
    private NumberGenerator numberGenerator;

    @Inject
    private ItemService itemService;


    // ...
}
```

Qualifying an Injection Point

```
public class BookBean implements Serializable {

   @Inject @ThirteenDigits
   private NumberGenerator numberGenerator;

   @Inject @Default
   private ItemService itemService;


   // ...
}
```

Qualifying a Bean

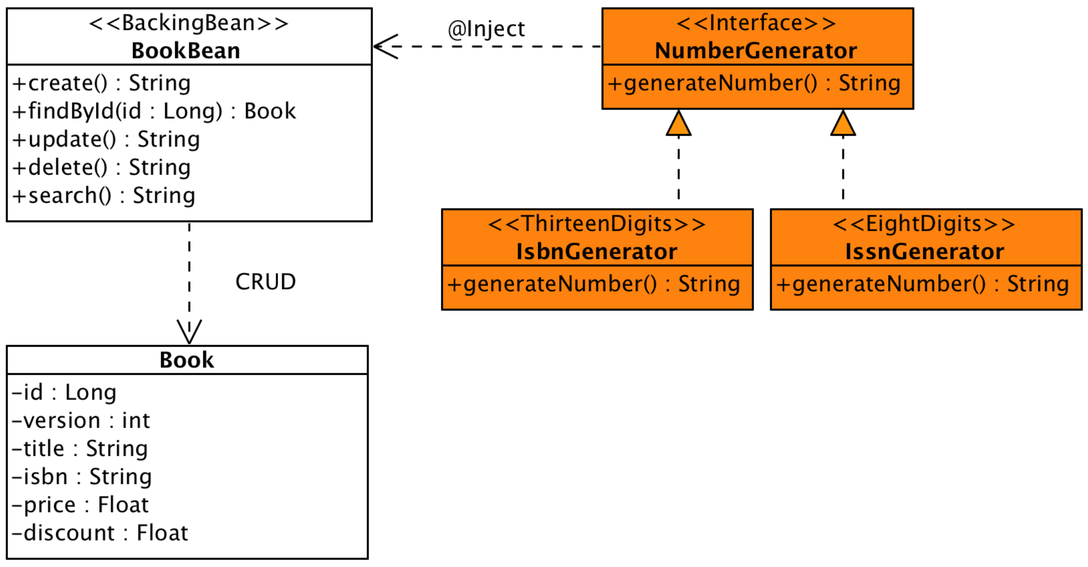
```
@ThirteenDigits
public class IsbnGenerator implements NumberGenerator {

  @Override
  public String generateNumber() {
    return "13-" + Math.abs(new Random().nextInt());
  }
}
```

Demo

-

Qualifiers

Demo: @Inject One Implementation



Producers

How Do I ?

```
public class BookBean implements Serializable {

    @Inject
    private EntityManager em;

    @Inject
    private Logger logger;


    // ...
}
```

How Do I ?

```
public class BookBean implements Serializable {

    @Inje
    priv        ityManager e

    @I    ct
    pr    te Logge    gger;


    // ..
}
```

How Do I ?

```
public class BookBean implements Serializable {

    @Inject
    private EntityManager em;

    @Inject
    private Logger logger;


    // ...
}
```

Several persistence units

```
@PersistenceContext(unitName = "myPU1")
@PersistenceContext(unitName = "myPU2")
```

How Do I ?

```
public class BookBean implements Serializable {

    @Inject
    private EntityManager em;

    @Inject
    private Logger logger;


    // ...
}
```

Third party framewok

Use Producers !

```
public class BookBean implements Serializable {

    @Inject
    private EntityManager em;

    // ...
}
```

```
public class ResourceProducer {

    @Produces
    @PersistenceContext(unitName = "myPU")
    private EntityManager entityManager;

}
```

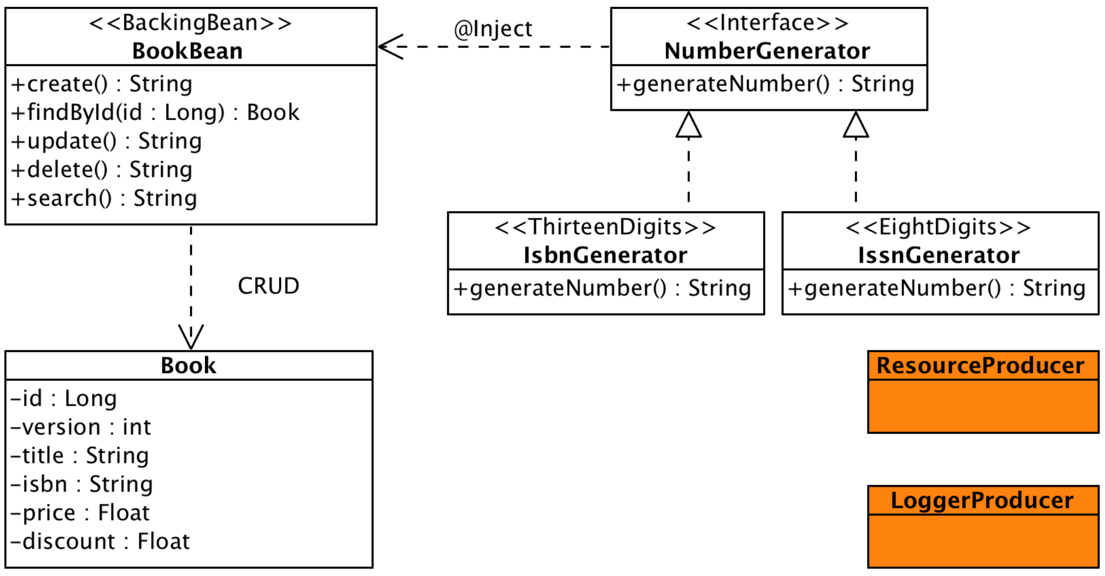
Use Producers !

```java
public class BookBean implements Serializable {

    @Inject
    private Logger logger;

    // ...
}
```

```java
public class ResourceProducer {

    @Produces
    private Logger produceLogger(InjectionPoint ip) {
     return
     Logger.getLogger(ip.getMember().getDeclaringClass().getName());
    }
}
```

Demo
-
Producers

Demo: Producers

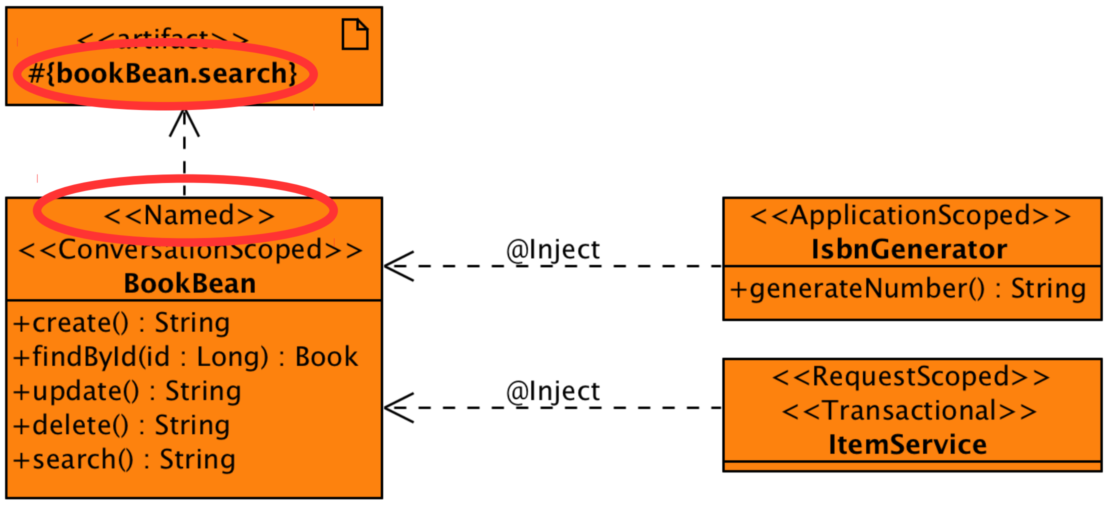


Web tier
&
Service tier

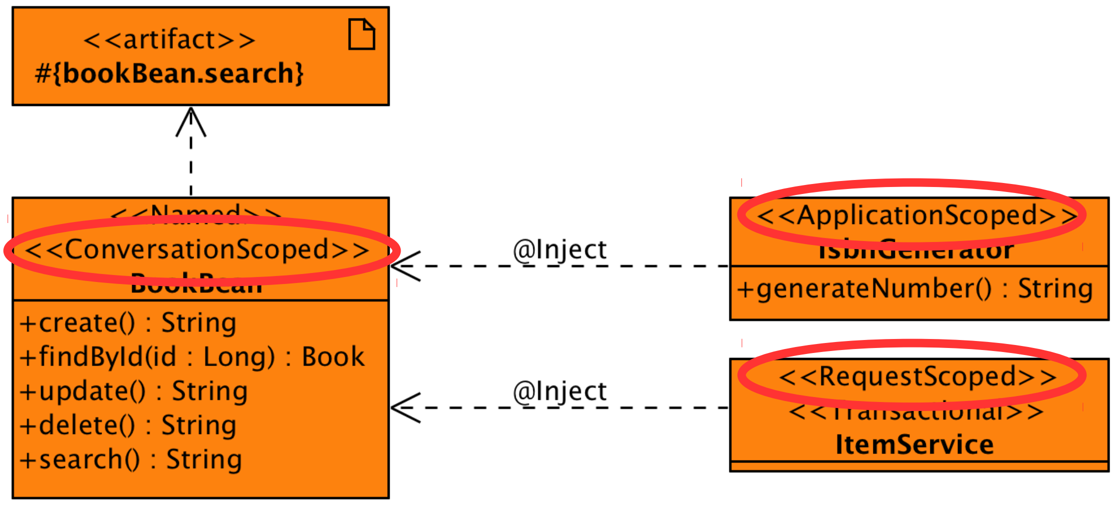
How Do I ?

How Do I ?

Use Expression Language...



Use Expression Language and Scopes !



Service Tier

```
@Transactional
public class BookBean implements Serializable {

   @Inject
   private EntityManager em;

   public void update() {
      em.persist(book);
   }
}
```

Service Tier + Web Tier

```
@Named

@Transactional
public class BookBean implements Serializable {

   @Inject
   private EntityManager em;

   public void update() {
      em.persist(book);
   }
}
```

```
<h:commandLink value="Create"
               action='#{bookBean.update}'/>
```

Service Tier + Web Tier

```
@Named("service")

@Transactional
public class BookBean implements Serializable {

    @Inject
    private EntityManager em;

    public void update() {
        em.persist(book);
    }
}
```

```
<h:commandLink value="Create"
               action='#{service.update}'/>
```

Several scopes

- `@Dependent` (default)
- `@ApplicationScoped, @SessionScoped, @RequestScoped`
- `@ConversationScoped`
- Create your own
  - `@TransactionalScoped`
  - `@ViewScoped`
  - `@ThreadScoped`
  - `@ClusterScoped`

Just choose the right scope

```
@Named
@RequestScoped
@Transactional
public class BookBean implements Serializable {




    public void update() {

    }

    public void delete() {

    }
}
```

Just choose the right scope

```
@Named
@SessionScoped
@Transactional
public class BookBean implements Serializable {




   public void update() {

   }

   public void delete() {

   }
}
```

Just choose the right scope

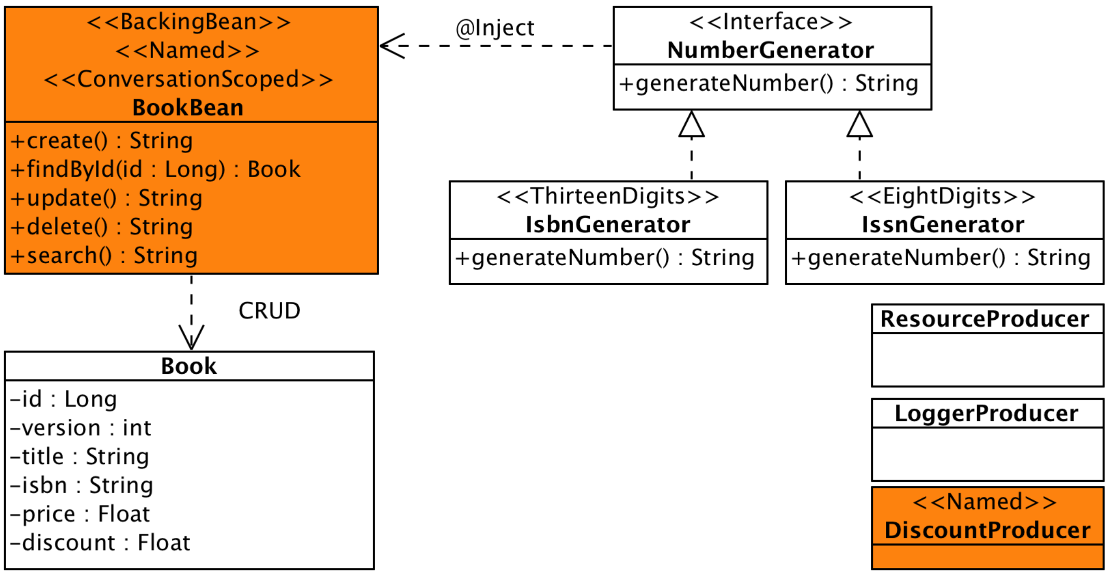
```
@Named
@ConversationScoped
@Transactional
public class BookBean implements Serializable {

   @Inject
   private Conversation conversation;

   public void update() {
      conversation.begin();
   }

   public void delete() {
      conversation.end();
   }
}
```

Demo

-

@Named & scope

Demo: @Named & Scope



Alternatives

How Do I ?

```
public class MockGenerator implements NumberGenerator  {

   public String generateNumber() {
      return "mock-" + Math.abs(new Random().nextInt());
   }
}
```

How Do I ?

```
@Mock
public class MockGenerator implements NumberGenerator  {

   public String generateNumber() {
      return "mock-" + Math.abs(new Random().nextInt());
   }
}
```

```
public class BookBean implements Serializable {

   @Inject @Mock
   private NumberGenerator numberGenerator;
   // ...
}
```

How Do I ?

```
@Mock
public class Mock                   ts NumberGenerator  {

   public Str       erateNumber() {
      retur   oc      Math.abs(new R    om().nextInt());
   }
}
```

```
pu    class BookBe     mpleme    Serializable {

   @     ct @Mock
   pri      NumberGenerat       mberGenerator;
   // ..
}
```

Use an Alternative !

```
@Alternative
@EightDigits
public class MockGenerator implements NumberGenerator  {

   public String generateNumber() {
      return "mock-" + Math.abs(new Random().nextInt());
   }
}
```

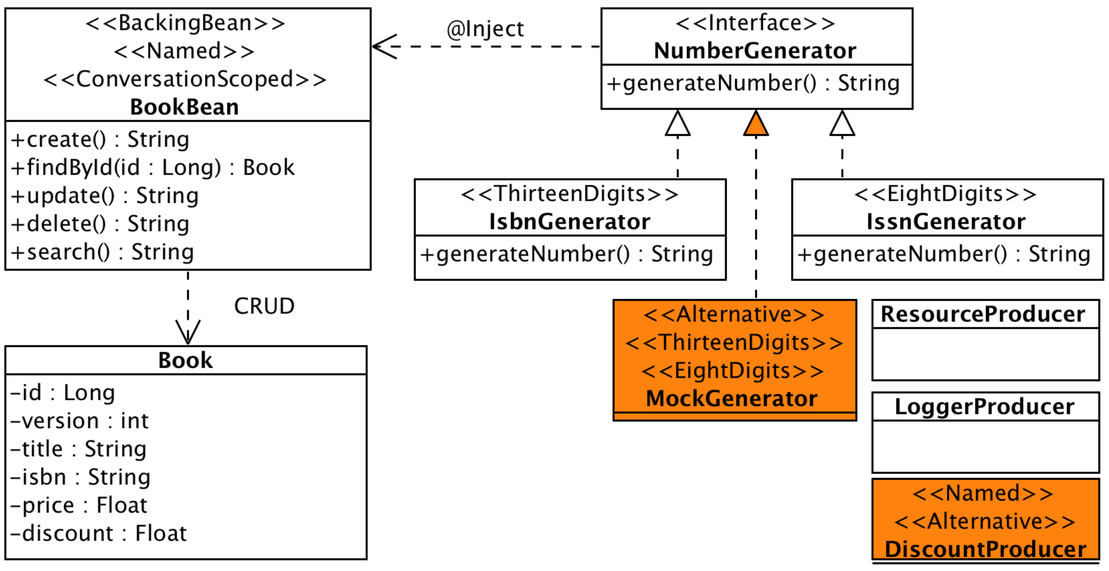
```
public class BookBean implements Serializable {

   @Inject @EightDigits
   private NumberGenerator numberGenerator;
   // ...
}
```

Activate the Alternative

```xml
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
       ...
       version="1.1" bean-discovery-mode="all">

  <alternatives>
    <class>com.foo.MockGenerator</class>
  </alternatives>
</beans>
```

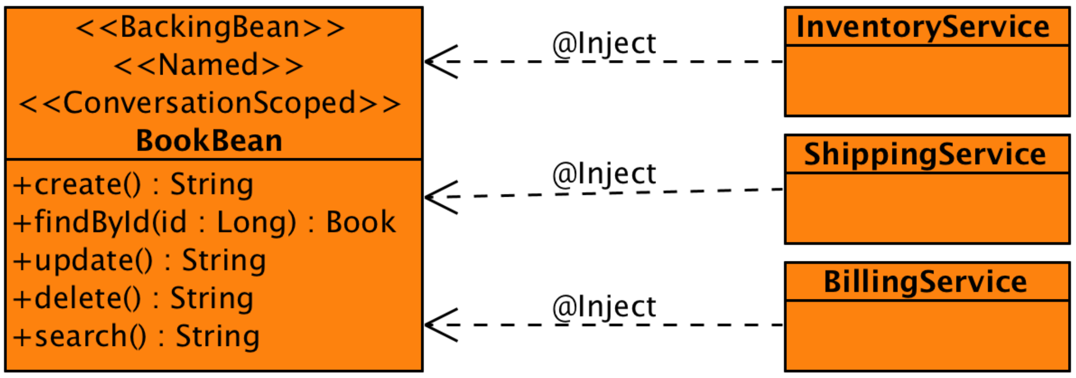
Demo
-
Alternatives

Demo: Alternatives



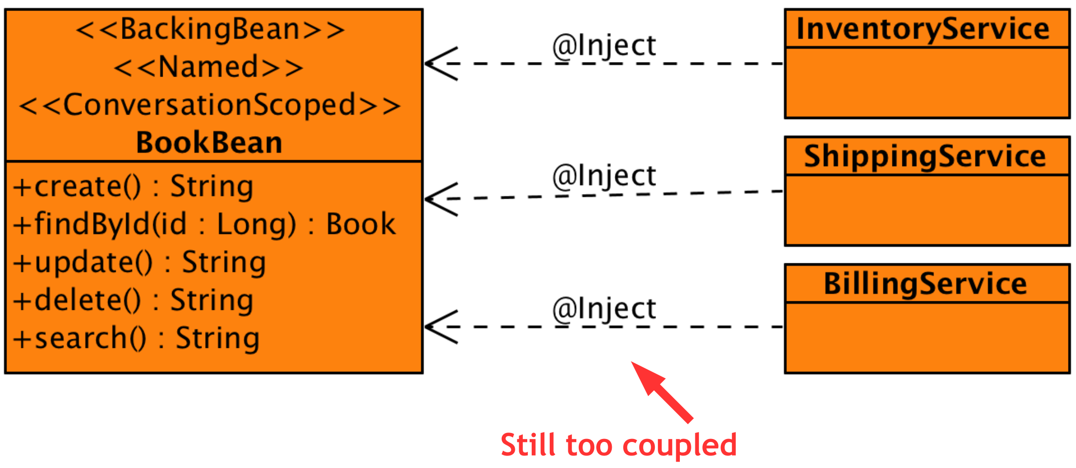
Events

How Do I ?



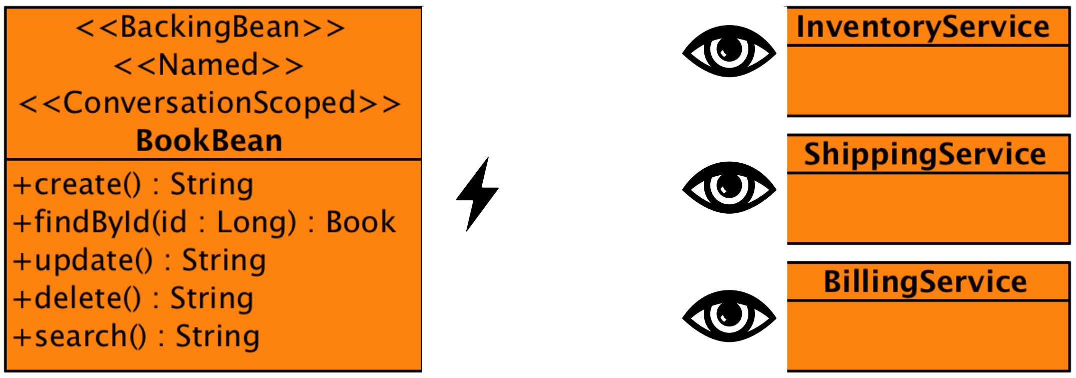
<<BackingBean>>
<<Named>>
<<ConversationScoped>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
@Inject
@Inject
InventoryService
ShippingService
BillingService

How Do I ?



<<BackingBean>>
<<Named>>
<<ConversationScoped>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
@Inject
@Inject
InventoryService
ShippingService
BillingService
Still too coupled

Use Events !



<<BackingBean>>
<<Named>>
<<ConversationScoped>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
InventoryService
ShippingService
BillingService

Fire and Observe

```java
public class BookBean implements Serializable {

   @Inject
   private Event<Book> boughtEvent;

   public void update() {
      boughtEvent.fire(book);
   }
}
```

```java
public class InventoryService {

    private void observeBooks (@Observes        Book book) {
        logger.info("Book recevied " + book.getTitle());
    }
}
```

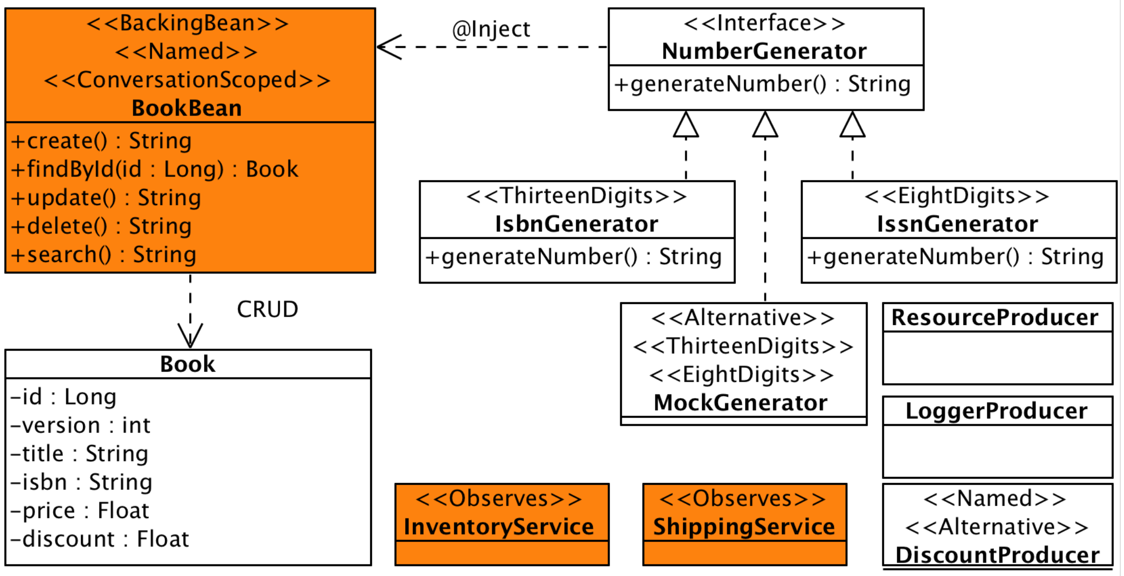
Fire and Observe with Qualifier

```java
public class BookBean implements Serializable {

   @Inject @Paper
   private Event<Book> boughtEvent;

   public void update() {
      boughtEvent.fire(book);
   }
}
```

```java
public class InventoryService {

    private void observeBooks (@Observes @Paper Book book) {
        logger.info("Book recevied " + book.getTitle());
    }
}
```

Demo
-
Events

Demo: Events



<<BackingBean>>
<<Named>>
<<ConversationScoped>>
BookBean
+create() : String
+findById(id : Long) : Book
+update() : String
+delete() : String
+search() : String
@Inject
<<Interface>>
NumberGenerator
+generateNumber() : String
<<ThirteenDigits>>
IsbnGenerator
+generateNumber() : String
<<EightDigits>>
IssnGenerator
+generateNumber() : String
CRUD
Book
-id : Long
-version : int
-title : String
-isbn : String
-price : Float
-discount : Float
<<Alternative>>
<<ThirteenDigits>>
<<EightDigits>>
MockGenerator
ResourceProducer
LoggerProducer
<<Observes>>
InventoryService
<<Observes>>
ShippingService
<<Named>>
<<Alternative>>
DiscountProducer

CDI : So Much More

CDI : So Much More

CDI Extension ecosystem

DELTASPIKE

CDI Course on PluralSight

pluralsight
hardcore developer training

http://www.pluralsight.com/courses/context-dependency-injection-1-1

Thanks

www.antoniogoncalves.org
antonio.goncalves@gmail.com
@agoncal
@devoxxfr
@lescastcodeurs

Q & A

Creative Commons

- **Attribution** — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- **Noncommercial** — You may not use this work for commercial purposes.
- **Share Alike** — If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.