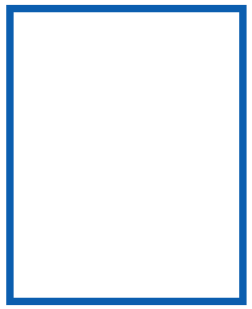
Creating a Common Application Tier

Antonio Goncalves

@agoncal | www.antoniogoncalves.org

Previous Module



- Java Enterprise Edition
- Components
- Metadata
- Container
- Services
- Specification

Module Outline

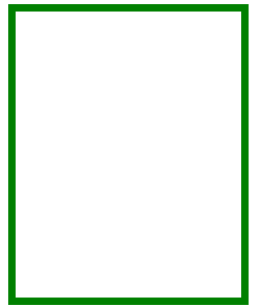
Module Outline

Injection

Context & Dependency Injection (CDI) 1.1

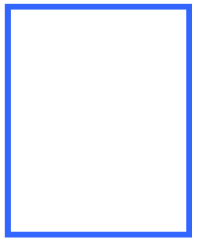
What Is Injection?

- Applications are made of interactions
- Objects depend on other objects
- Do not create their own dependencies
- Dependency is delegated to a dependency injector
- a.k.a. Container
- Inversion of control



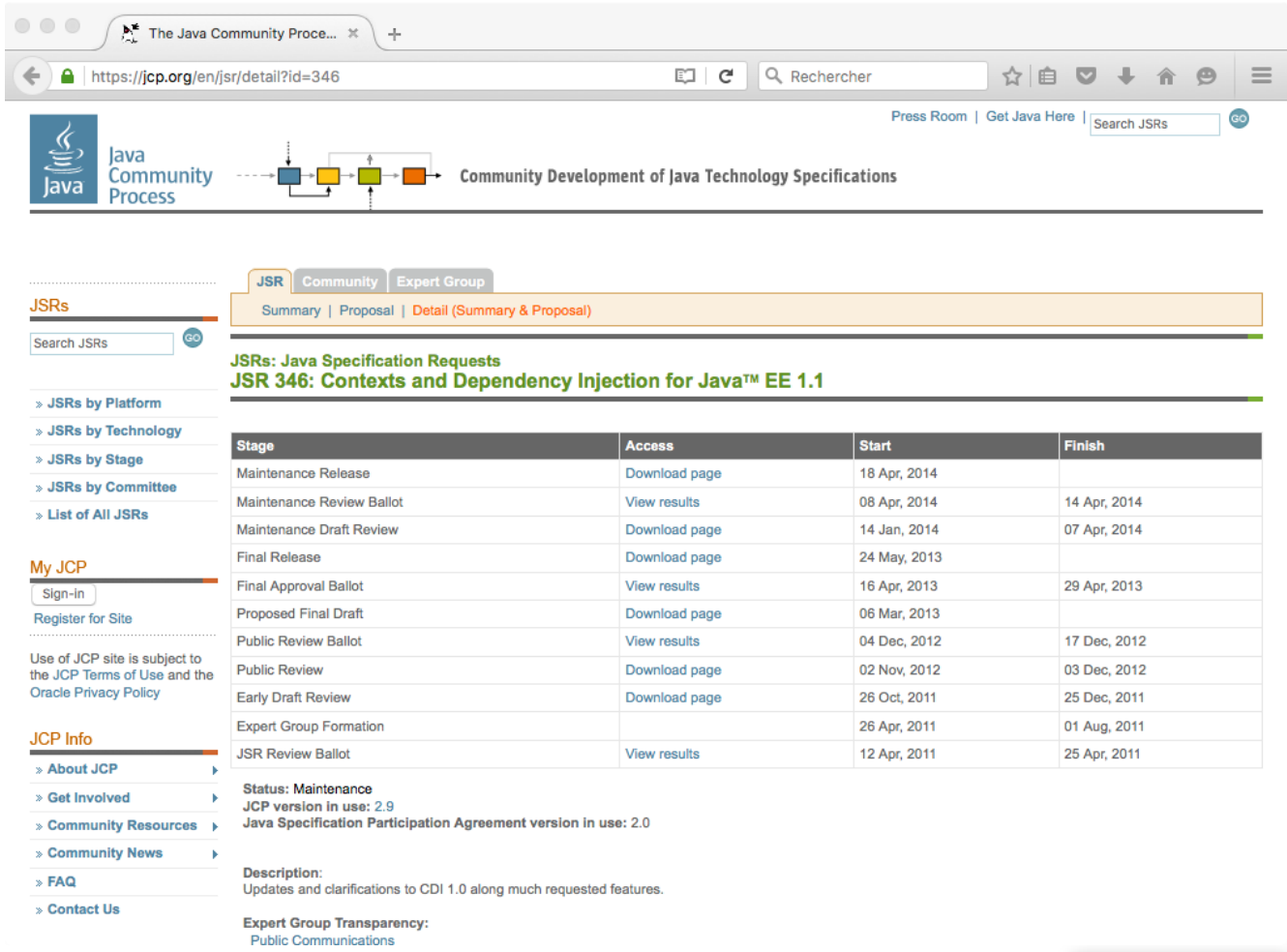
When to Use Injection

- Loose coupling, strong typing
- Components are loosely coupled
- Strongly typed way
- An application is made of several components
- Container injects components into others



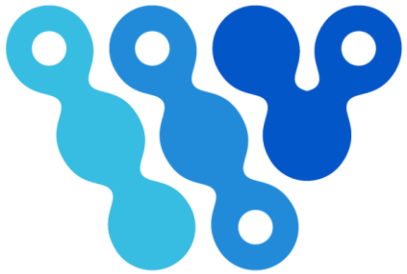
CDI Specification

- Context & Dependency Injection 1.1
- JSR 346
- http://jcp.org/en/jsr/detail?id=346



CDI Implementations

- Weld
- OpenWebBeans
- DeltaSpike



OpenWebBeans

DELTASPIKE

CDI

Used everywhere

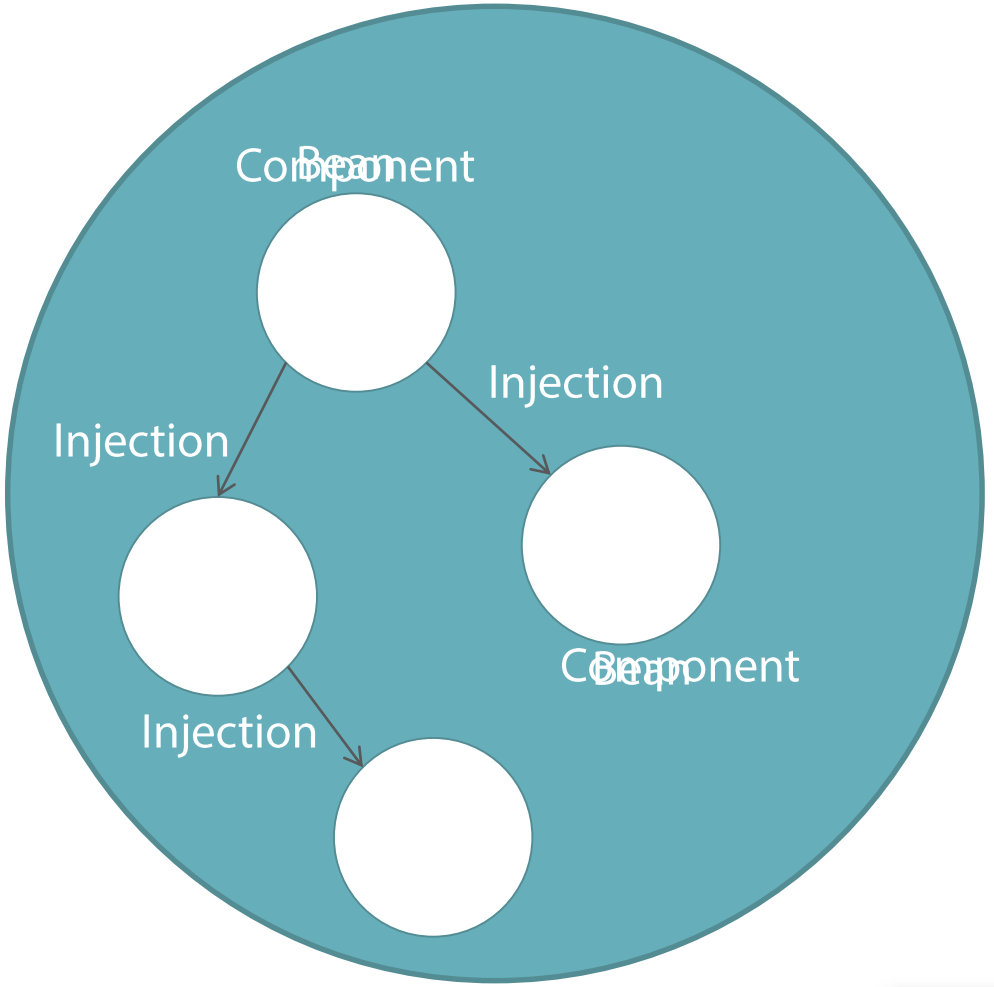
Creating a book

Needs an ISBN number

Inject a ISBN number generator

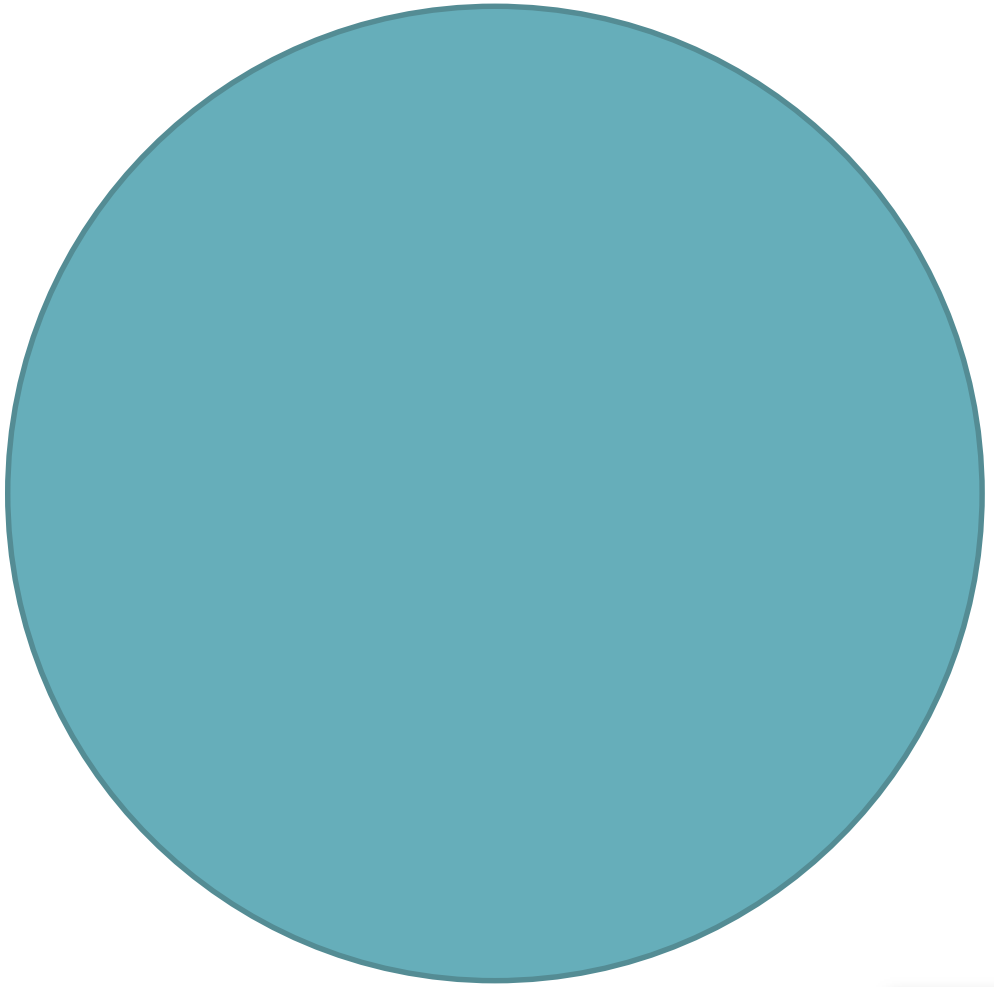
Understanding CDI

- Package and deploy components
- Bean Manager
- Beans
- Dependencies
- Injection
- Beans are connected
- Up and running



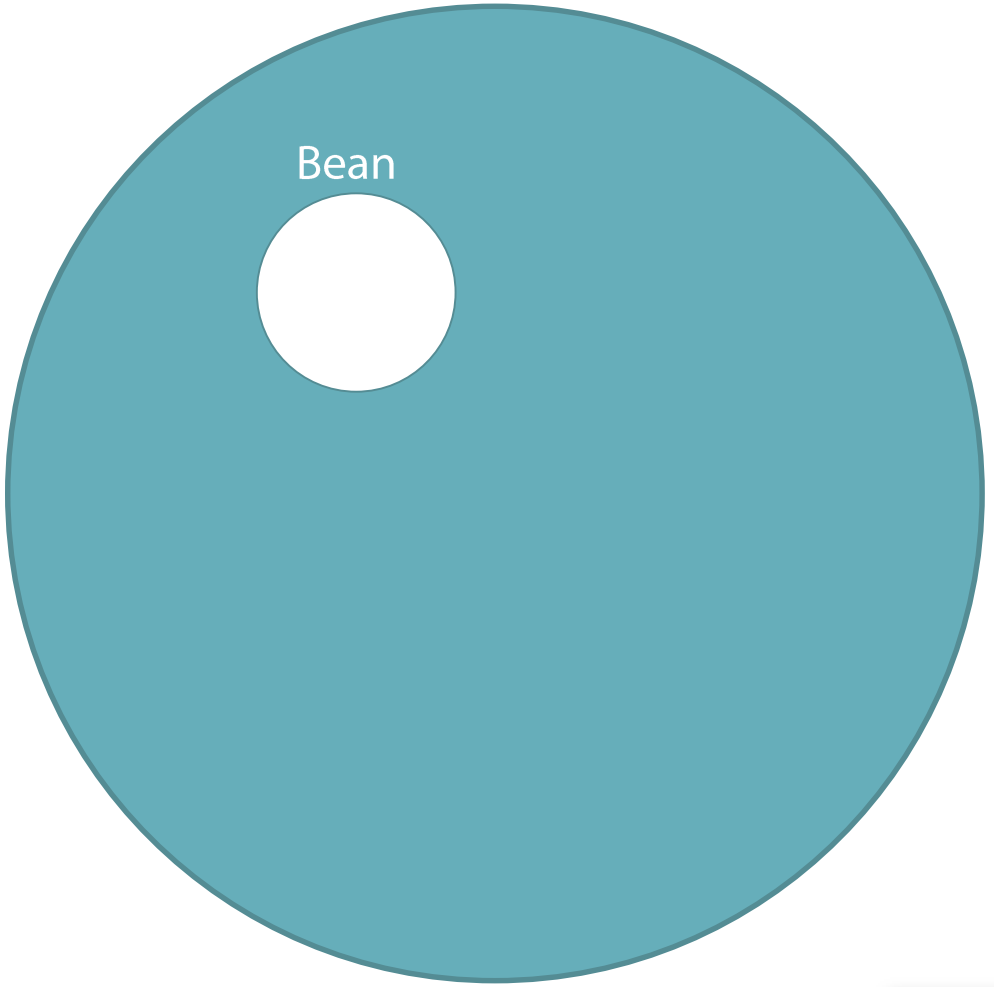
Bean Manager

- Runtime environment
- Manages beans
- Bean discovery
- Detects errors
- Deployment is canceled
- Provides services
  - Injection
  - Life-cycle management



CDI Bean

- An object managed by bean manager
- Managed bean
- "Objects managed by containers"
  - Bean manager for CDI
  - Servlet container for Servlets
- Different types of managed beans



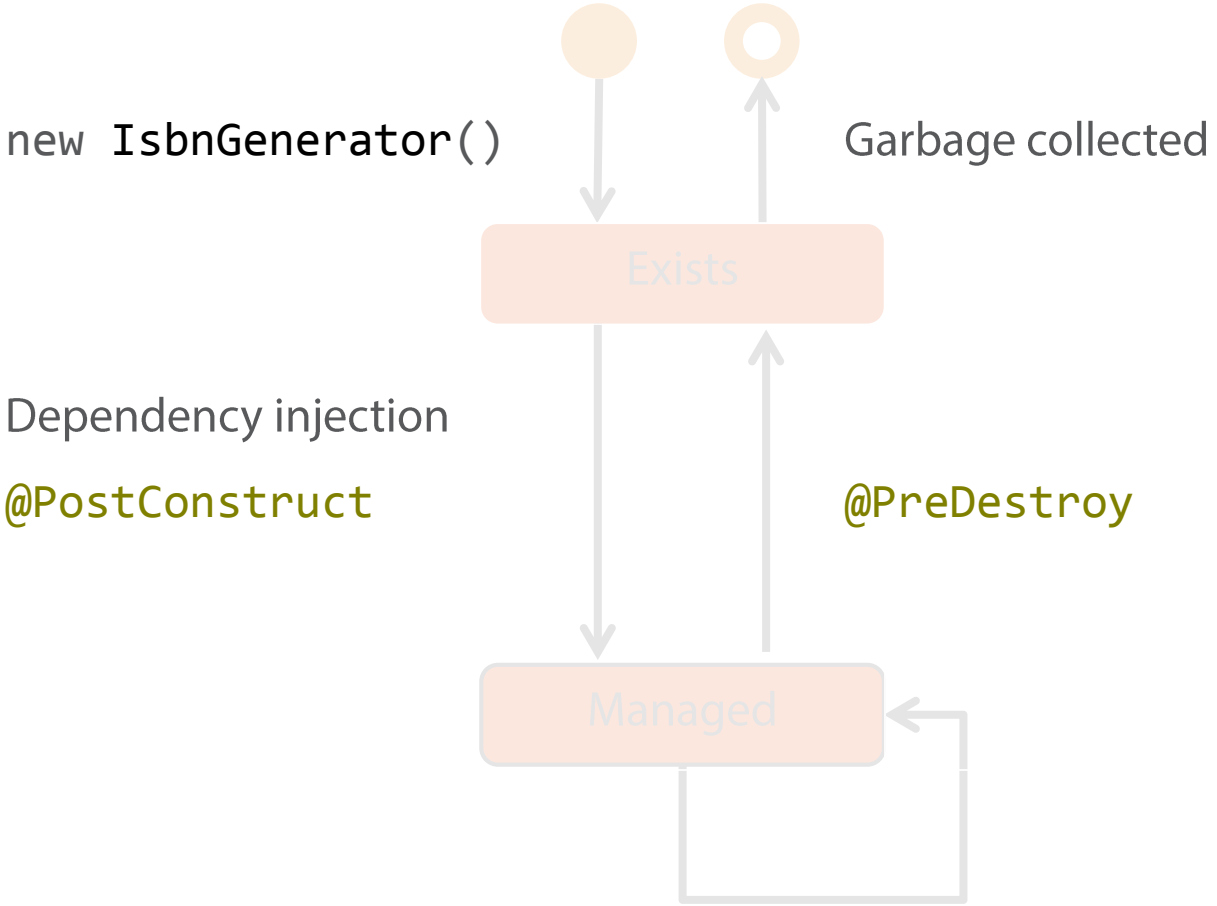
CDI Bean

```
@ThirteenDigits
public class IsbnGenerator {

  public String generateNumber() {
    return "13-84356-" + Math.abs(new Random().nextInt());
  }
}
```

CDI Bean

```
@Named
public class IsbnGenerator {

  public String generateNumber() {
    return "13-84356-" + Math.abs(new Random().nextInt());
  }
}
```

Life Cycle of a CDI Bean

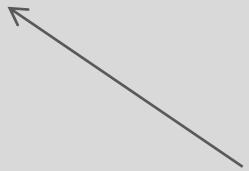


Life Cycle of a CDI Bean

```
public class IsbnGenerator {

  private int postfix;

  @PostConstruct
  private void init () {
    postfix = Math.abs(new Random().nextInt());
  }

  public String generateNumber() {
    return "13-84356-" + postfix++;
  }
}
```

Deployment Descriptor

```
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
       xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                           http://xmlns.jcp.org/xml/ns/javaee/beans_1_1.xsd"
       version="1.1" bean-discovery-mode="all">

  <alternatives>
    <class>com.pluralsight.MockGenerator</class>
  </alternatives>
  <interceptors>
    <class>com.pluralsight.LoggingInterceptor</class>
  </interceptors>

</beans>
```



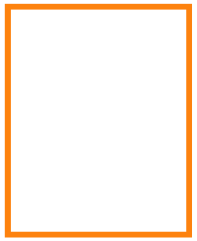
beans.xml

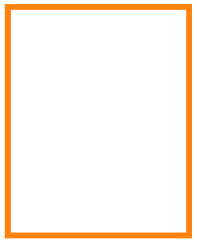
CDI Packages

Package	Description
`javax.inject`	Core dependency injection API
`javax.enterprise.inject`	Core CDI API
`javax.enterprise.context`	Scopes and contextual APIs
`javax.enterprise.event`	Events and observers APIs
`javax.interceptor`	Interceptor APIs
`javax.decorator`	Decorator APIs
`javax.enterprise.util`	CDI utility package

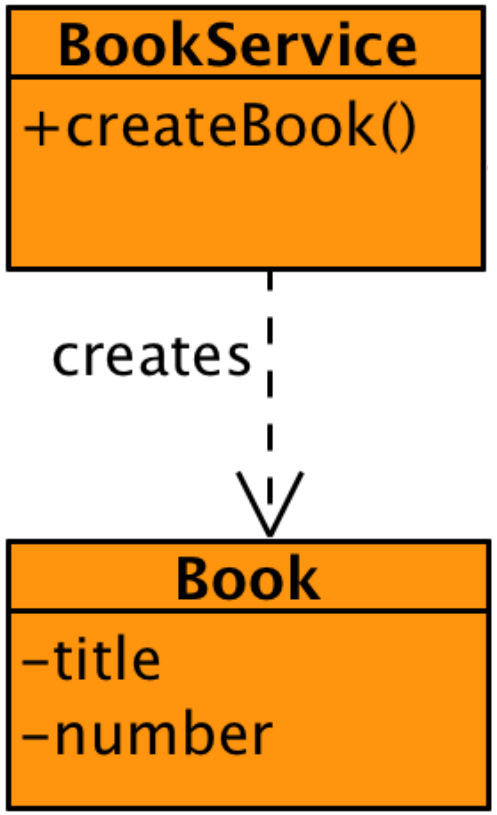
Dependency Injection

- Real world represented as objects
- Objects depend on each other
- CDI brings dependency injection
- Type safe way
- Loose coupling
- Strong typing





Depending on a Class



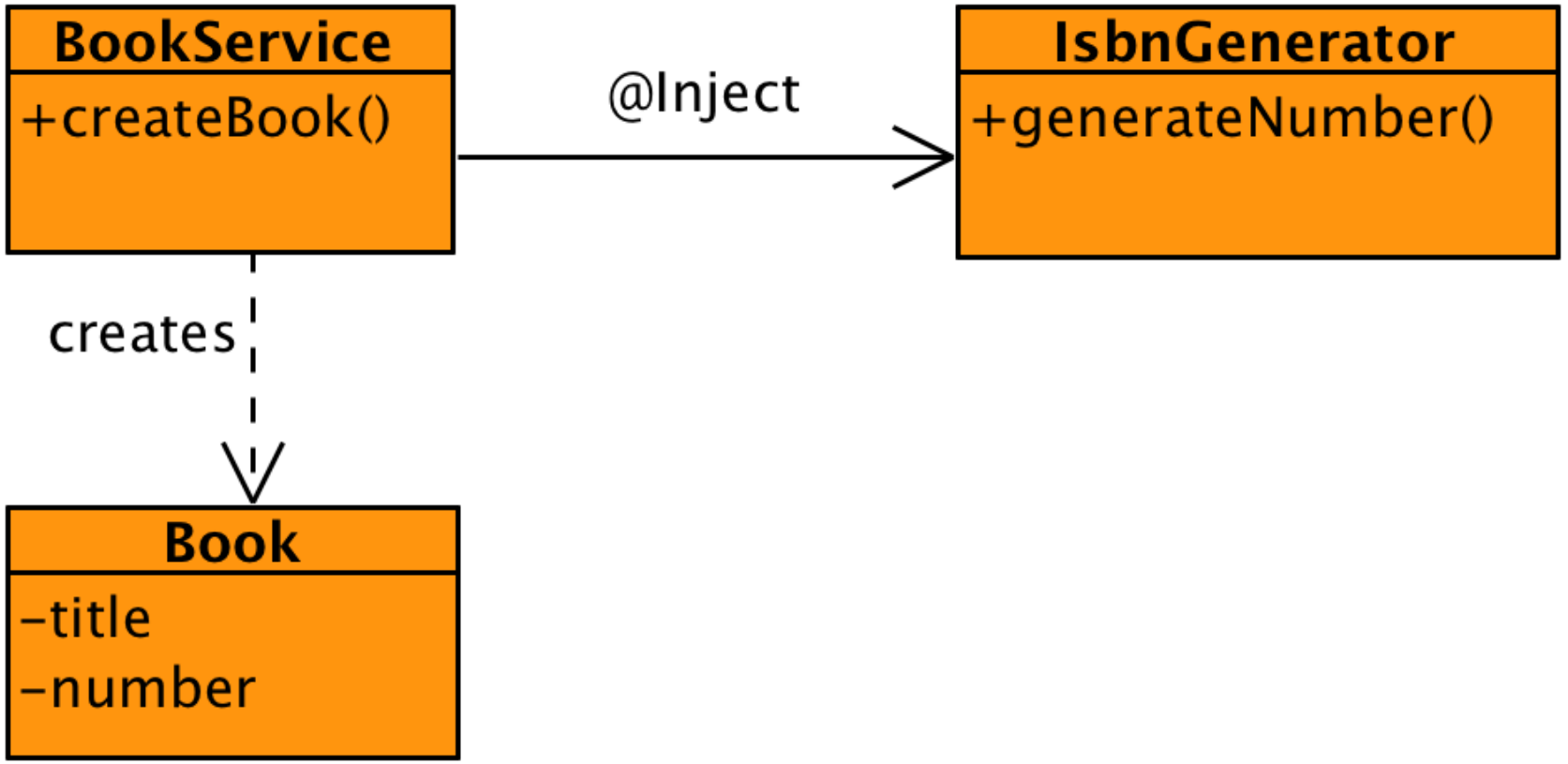
The IsbnGenerator Class

```
public class IsbnGenerator {

  public String generateNumber() {
    return "13-84356-" + Math.abs(new Random().nextInt());
  }
}
```

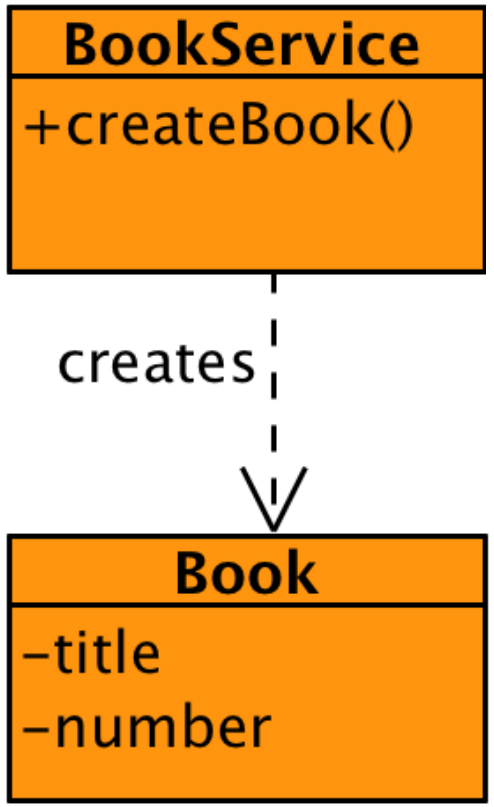
The BookService Class

```
public class IsbnGenerator {

  public String generateNumber() {
    return "13-84356-" + Math.abs(new Random().nextInt());
  }
}


public class BookService {

  @Inject
  private IsbnGenerator generator;

  public Book createBook(String title) {
    return new Book(title, generator.generateNumber());
  }
}
```

Depending on One Implementation



Depending on One Implementation



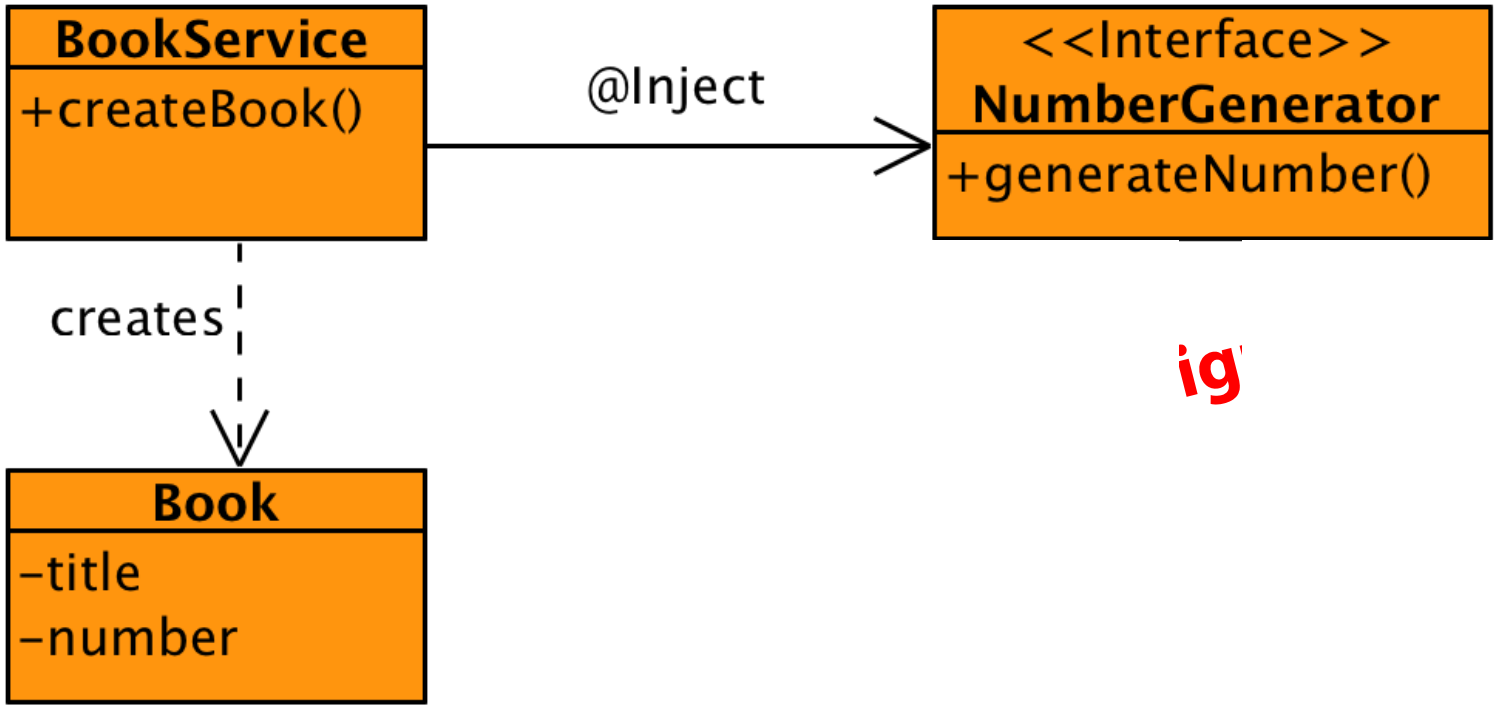
Interface and One Implementation

```
public interface NumberGenerator {
  String generateNumber();
}

public class IsbnGenerator implements NumberGenerator {
  public String generateNumber() {
    return "13-84356-" + Math.abs(new Random().nextInt());
  }
}
```

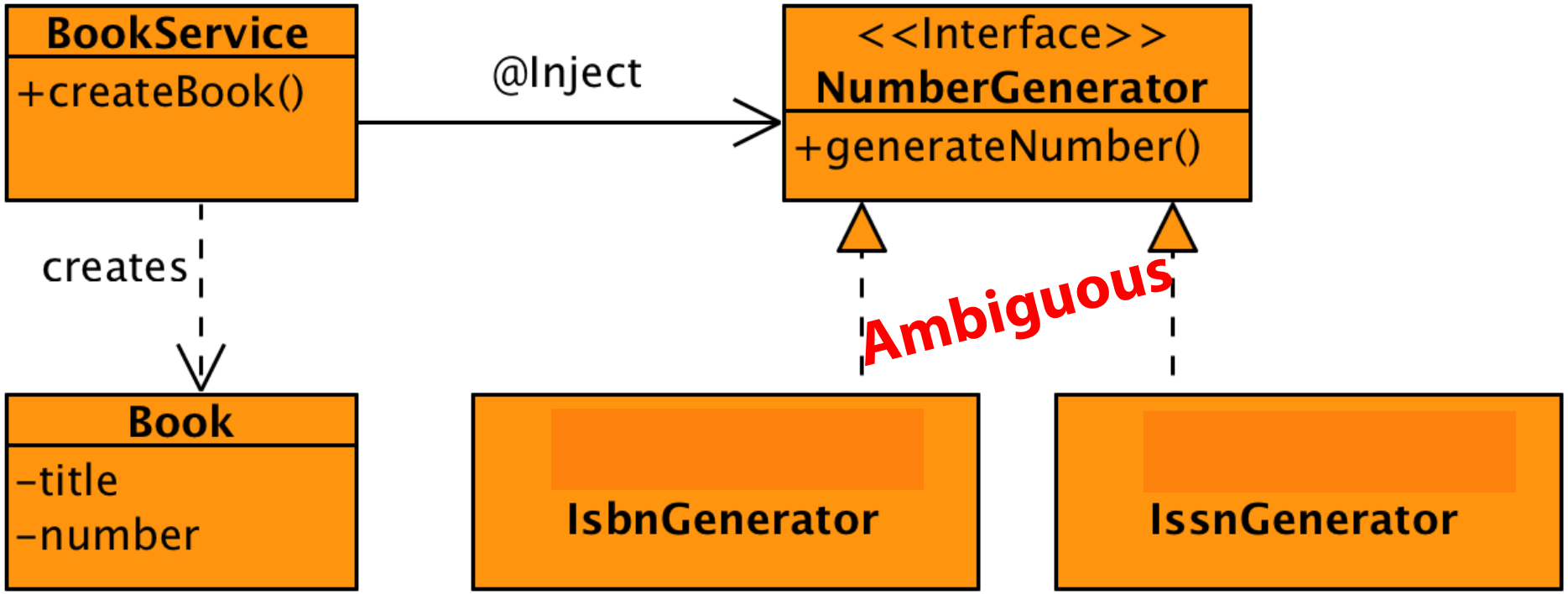
Injecting One Implementation

```
public interface NumberGenerator {
  String generateNumber();
}

public class IsbnGenerator implements NumberGenerator {
  public String generateNumber() {
    return "13-84356-" + Math.abs(new Random().nextInt());
  }
}

public class BookService {
  @Inject
  private NumberGenerator generator;

  public Book createBook(String title) {
    return new Book(title, generator.generateNumber());
  }
}
```

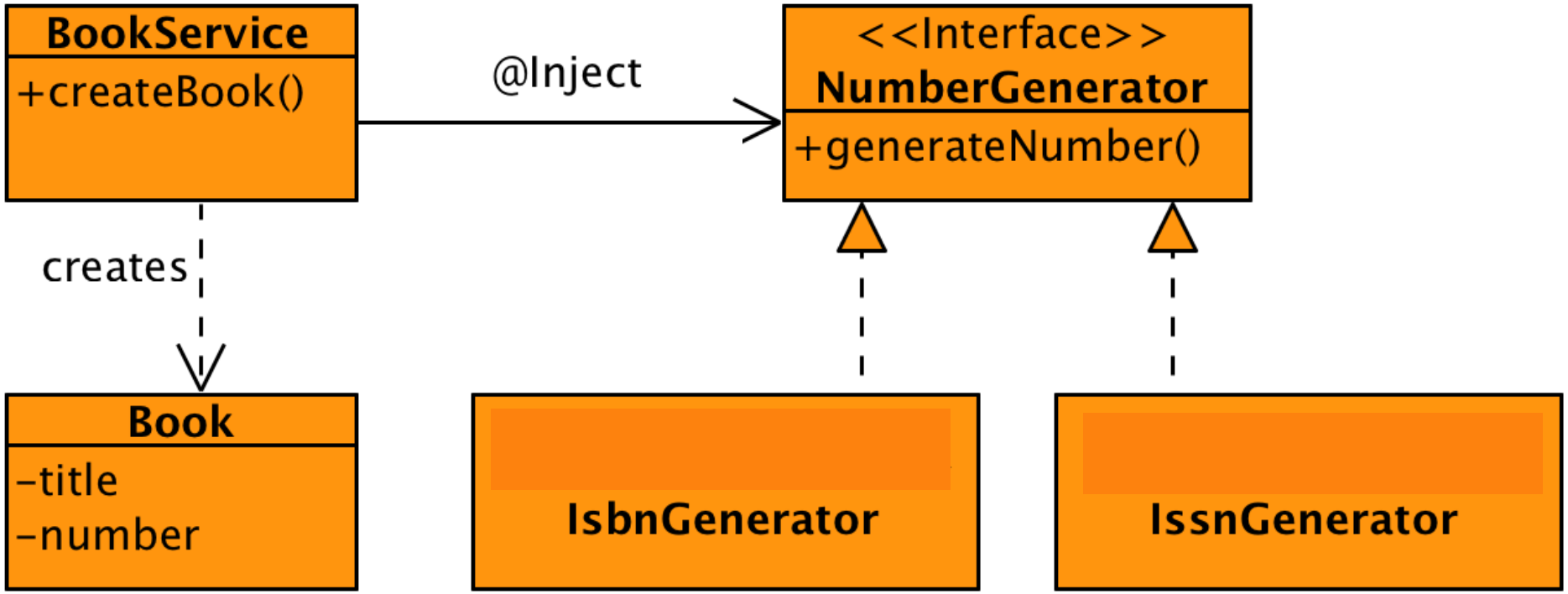
Ambiguous Injection



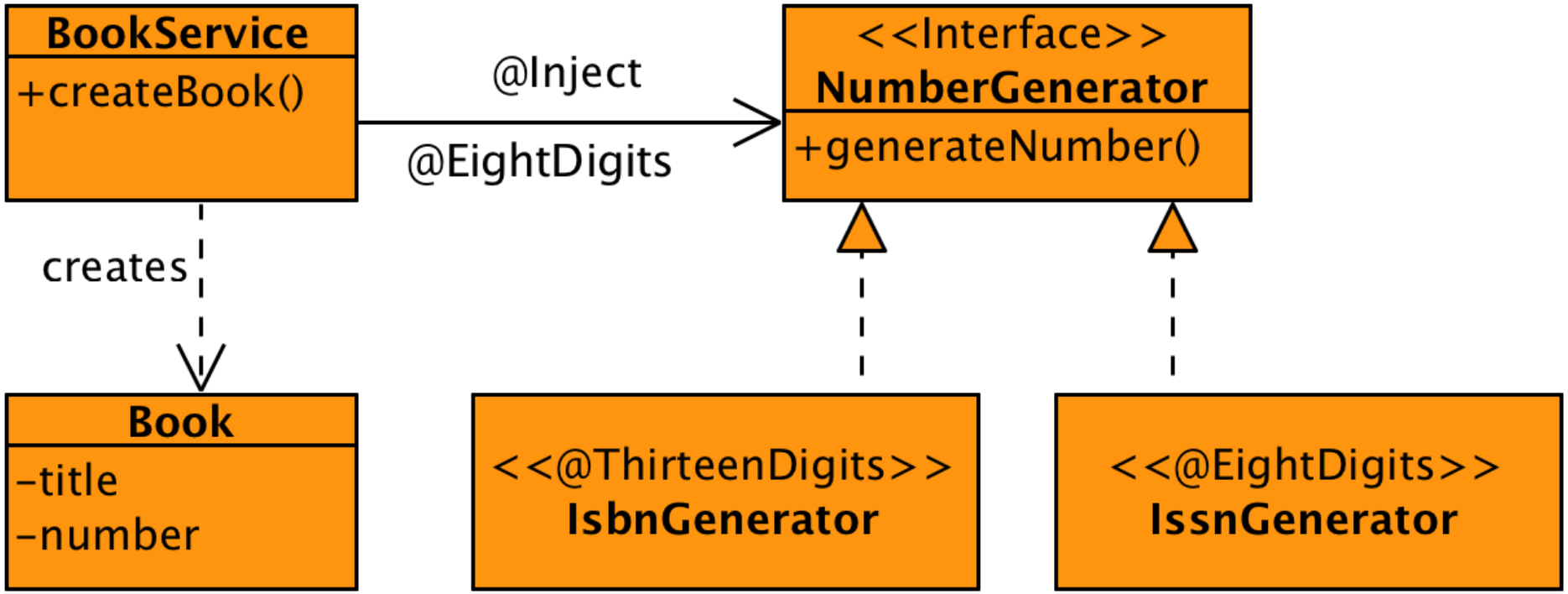
Default Qualifier



Having Different Qualifiers



Having Different Qualifiers



Two Qualifiers

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


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

Two Qualified Implementations

```
public interface NumberGenerator {
  String generateNumber();
}

@ThirteenDigits
public class IsbnGenerator implements NumberGenerator {
  public String generateNumber() {
    return "13-84356-" + Math.abs(new Random().nextInt());
  }
}

@EightDigits
public class IssnGenerator implements NumberGenerator {
  public String generateNumber() {
    return "8-" + Math.abs(new Random().nextInt());
  }
}
```

Injecting the IsbnGenerator

```
public class BookService {

  @Inject
  private NumberGenerator generator;

  public Book createBook(String title) {
    return new Book(title, generator.generateNumber());
  }
}
```

Choosing Implementation

```
public class BookService {

  @Inject @ThirteenDigits
  private NumberGenerator generator;

  public Book createBook(String title) {
    return new Book(title, generator.generateNumber());
  }
}
```

Choosing Implementation

```
public class BookService {

  @Inject @EightDigits
  private NumberGenerator generator;

  public Book createBook(String title) {
    return new Book(title, generator.generateNumber());
  }
}
```

CDI

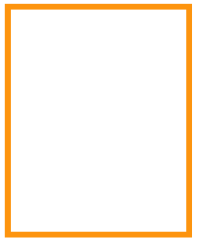
Ambiguous dependency

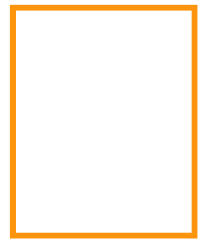
Qualifiers

Switch implementation

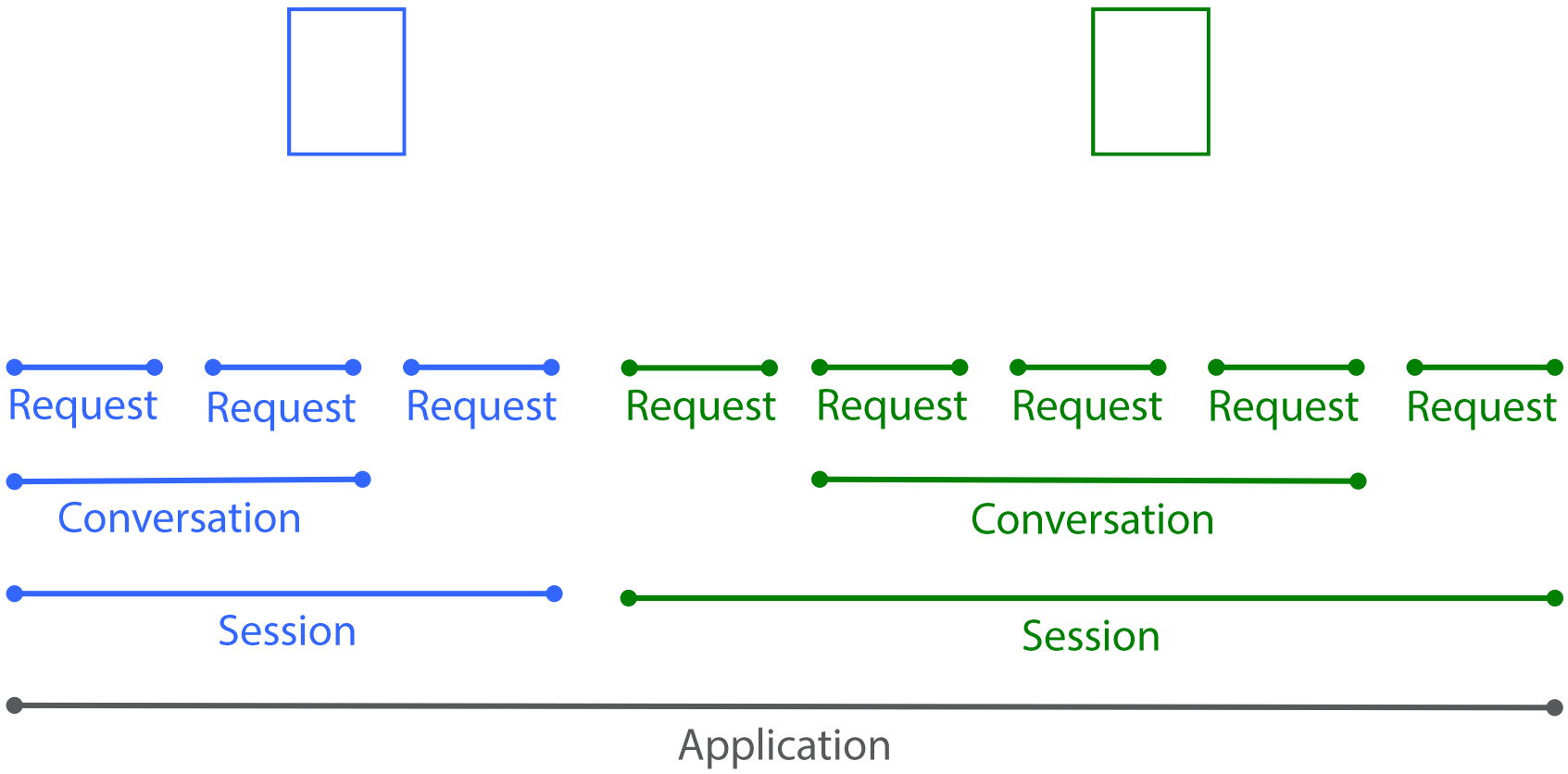
State Management

- Managing state within a context
- When context ends, state is cleaned up
- Declarative with annotations
- CDI built-in scopes
  - Application
  - Session
  - Request
  - Conversation





Built-in Scopes



Built-in Scopes

- `@ApplicationScoped`
- `@SessionScoped`
- `@RequestScoped`
- `@ConversationScoped`
- Conversation & session implement Serializable
- `@Dependent`
  - No annotation
  - Default

Application Scope

```
@ApplicationScoped
public class Cache implements Serializable {

  private Map<Object,Object> cache = new HashMap<>();

  public void addToCache (Object key, Object value) { // ... }

  public Object getFromCache (Object key) { // ... }

  public void removeFromCache (Object key) { // ... }

}
```

Session Scope

```
@SessionScoped
public class ShoppingCart implements Serializable {

  private List<Item> cartItems = new ArrayList<>();

  public String addItemToCart() { // ... }

  public String checkout() { // ... }

}
```

Request Scope

```
@RequestScoped
public class BookService {

  public Book persist(Book book) { // ... }

  public List<String> findAllImages() { // ... }

  public List<Book> findByCategory(long categoryId) { // ... }

}
```

Conversation Scope

```
@ConversationScoped
public class CustomerWizard implements Serializable {

  @Inject
  private Conversation conversation;

  private Customer customer = new Customer();

  public void initProfile () {
    conversation.begin();
    // ...
  }

  public void endProfile () {
    // ...
    conversation.end();
  }
}
```

Dependent Pseudo Scope

```
@Dependent
public class IsbnGenerator {

  public String generateNumber() {
    return "13-84356-" + Math.abs(new Random().nextInt());
  }
}



@RequestScoped
public class BookService {

  @Inject
  private IsbnGenerator generator;
  // ...
}
```

CDI

Context

Session scope

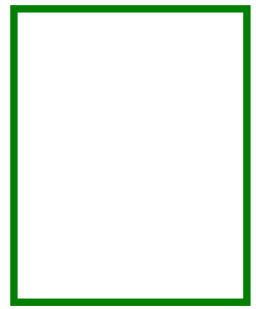
Login / logout

Interception

Interceptor 1.2

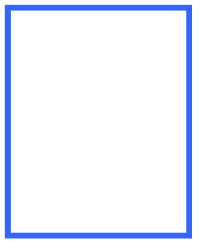
What Is Interception?

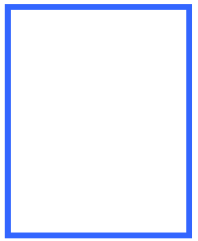
- Interpose on method invocation
- Intercepted by the container
- Separates concerns from our business code
- Technical concerns



When to Use Interception

- Common cross-cutting technical concerns
- Transactions
- Authentication
- Authorization
- Logging
- Isolate this technical code into an interceptor
- Enable for several beans





Interceptor Specification

- Interceptor 1.2
- JSR 318
- CDI specification (JSR 346)
- http://jcp.org/en/jsr/detail?id=318

Interceptor

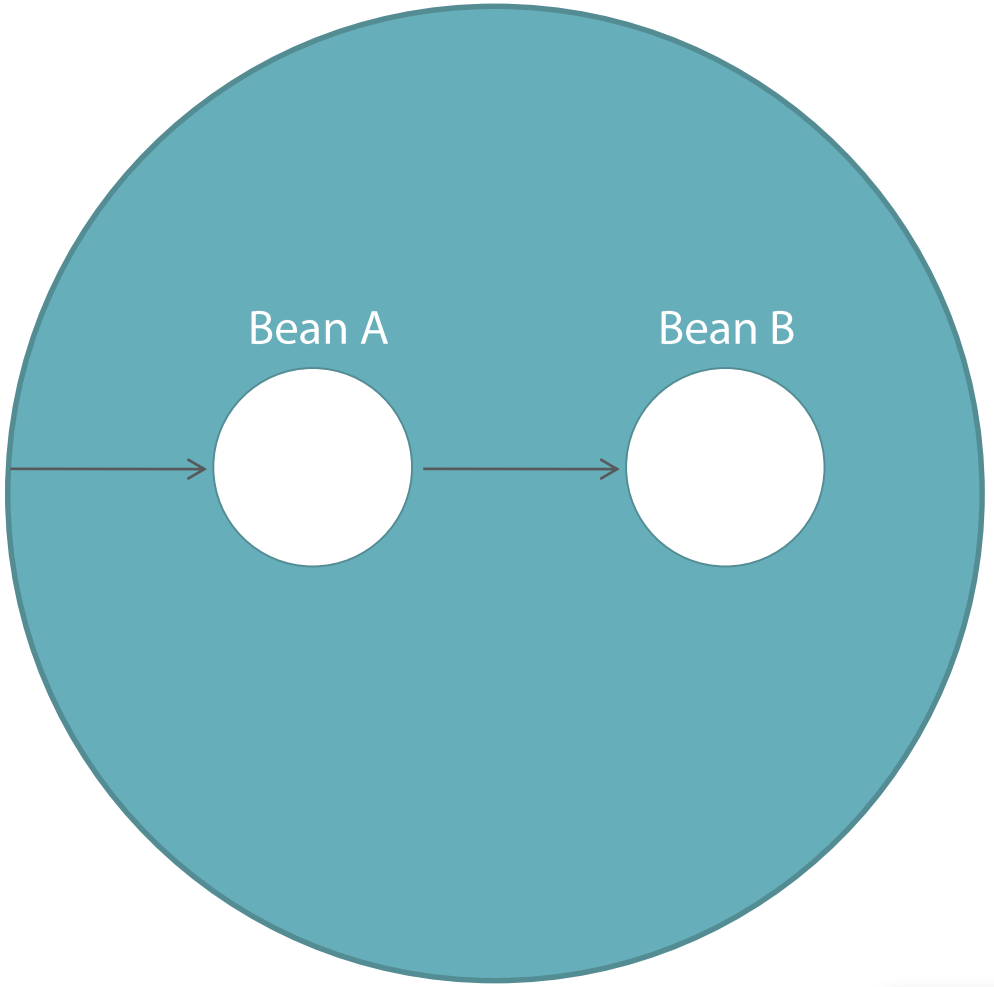
Logging

Method entries

Reuse in every bean

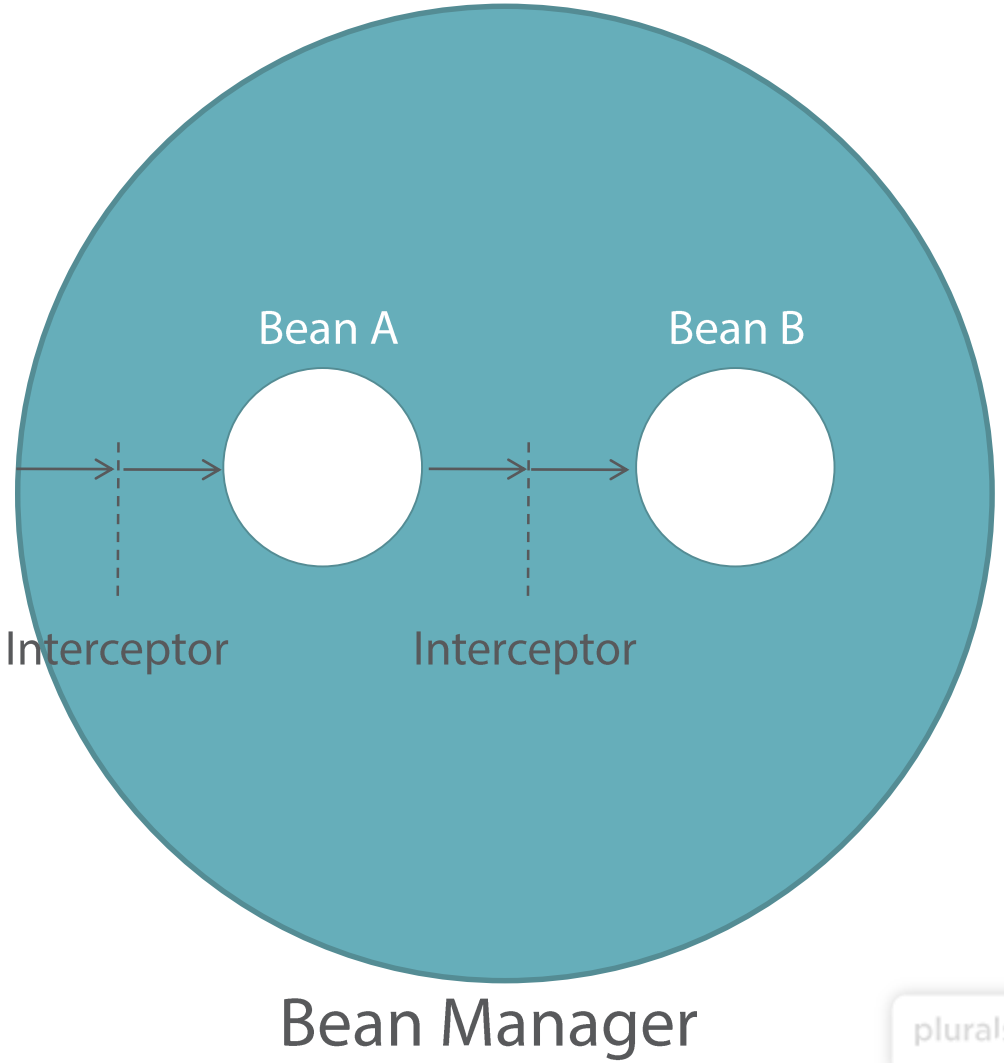
Understanding Interceptor

- Managed environment
- Brings services
- Interception
- Interceptor bindings



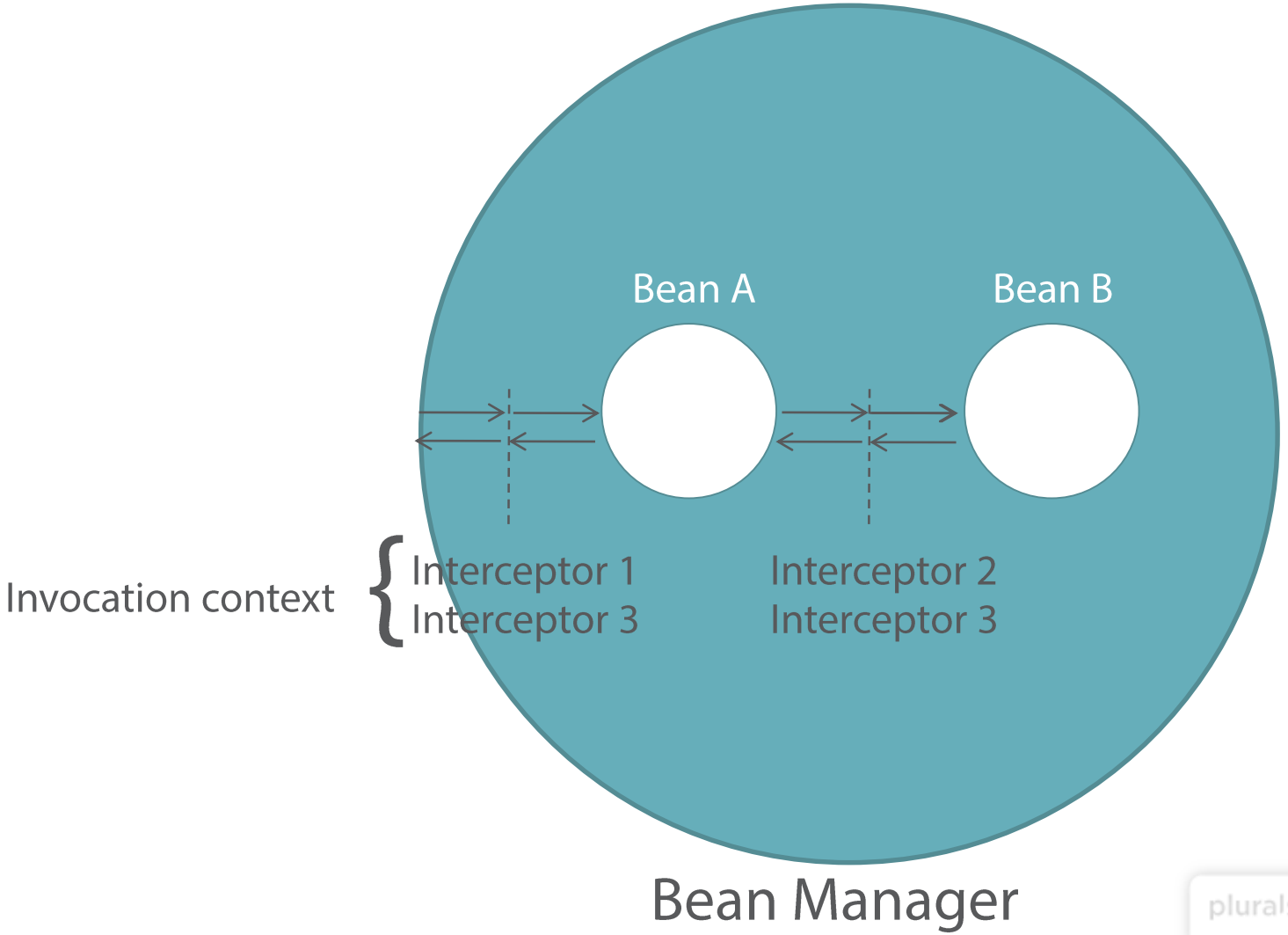
Understanding Interceptor

- Managed environment
- Brings services
- Interception
- Interceptor bindings



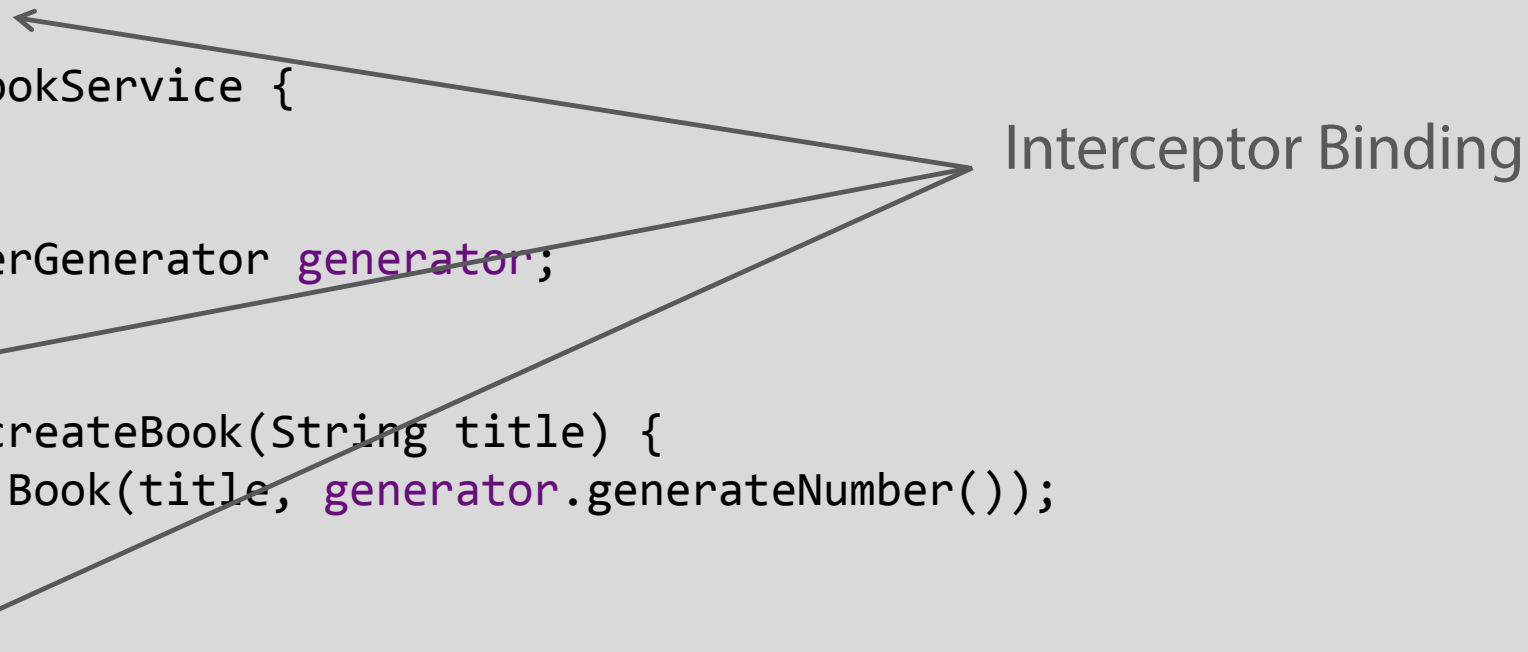
Understanding Interceptor

- Managed environment
- Brings services
- Interception
- Interceptor bindings
- Invocation context



Business Method Intercepted

```
@Transactional
@Secured
public class BookService {

  @Inject
  private NumberGenerator generator;

  @Loggable
  public Book createBook(String title) {
    return new Book(title, generator.generateNumber());
  }

  @Loggable
  public Book raisePrice(Book book) {
    book.setPrice(book.getPrice() * 2.5F);
    return book;
  }
}
```



Interceptor Binding

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

Interceptor Implementation

```
@Loggable
@Interceptor
public class LoggingInterceptor {

  @Inject
  private Logger logger;

  @AroundInvoke
  private Object intercept(InvocationContext ic) throws Exception {
    logger.info("> Entry");
    return ic.proceed();




  }
}
```

Enabling an Interceptor

```
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
       xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                           http://xmlns.jcp.org/xml/ns/javaee/beans_1_1.xsd"
       version="1.1" bean-discovery-mode="all">

  <interceptors>
    <class>com.pluralsight.LoggingInterceptor</class>
  </interceptors>

</beans>
```

Interceptor Implementation

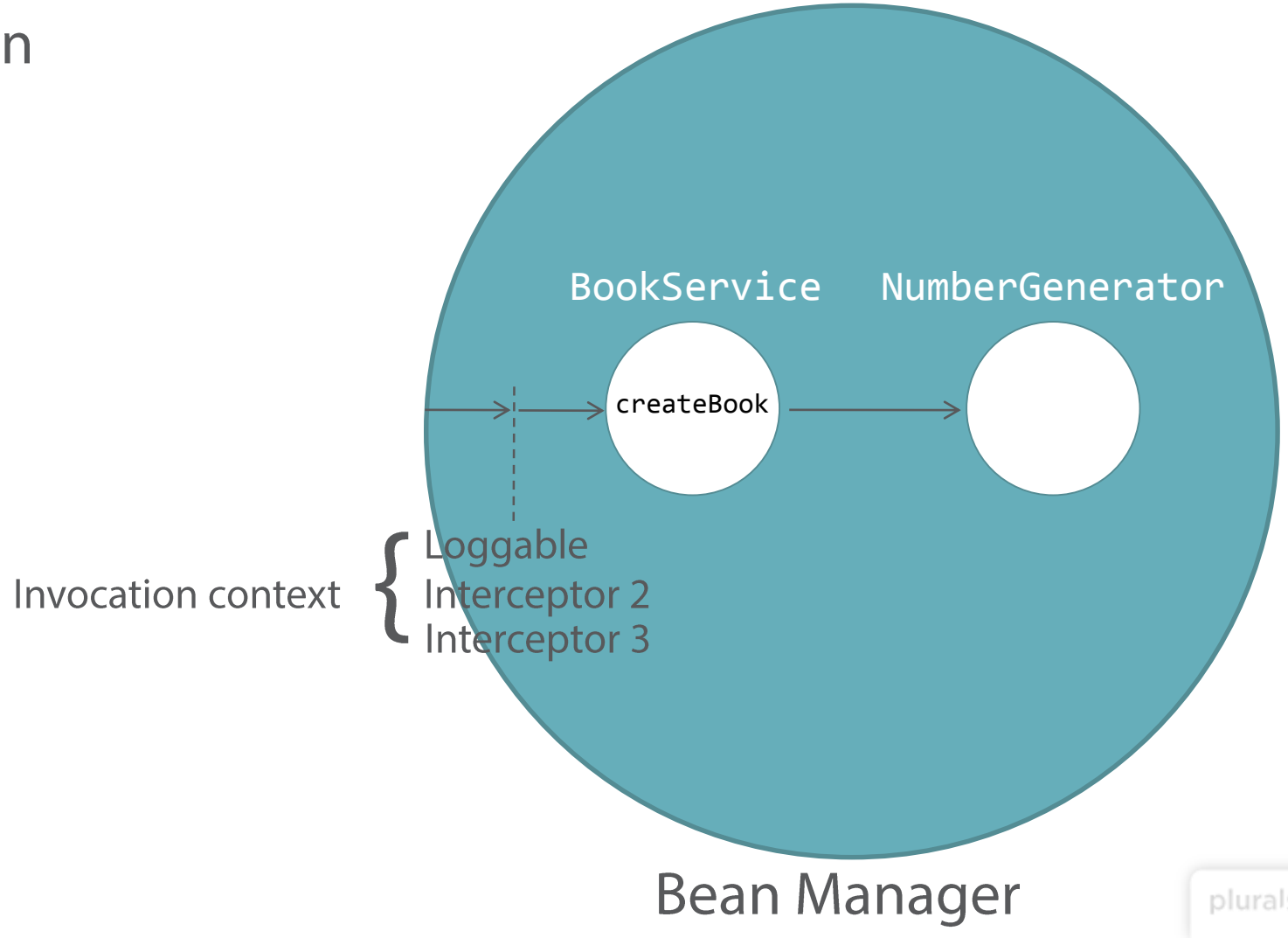
```
@Loggable
@Interceptor
public class LoggingInterceptor {

  @Inject
  private Logger logger;

  @AroundInvoke
  private Object intercept(InvocationContext ic) throws Exception {
    logger.info("> Entry");
    return ic.proceed();




  }
}
```

Interceptor Implementation

```
@Loggable
@Interceptor
public class LoggingInterceptor {

  @Inject
  private Logger logger;

  @AroundInvoke
  private Object intercept(InvocationContext ic) throws Exception {
    logger.info("> Entry {}", ic.getMethod());
    return ic.proceed();




  }
}
```

Invocation Context

- Control the invocation chain
- `InvocationContext` API
  - Target class
  - Target method
  - Method parameters
  - Add contextual data
  - Proceed



Invocation Context

```
@Loggable
@Interceptor
public class LoggingInterceptor {

  @Inject
  private Logger logger;

  @AroundInvoke
  private Object intercept(InvocationContext ic) throws Exception {
    logger.info("> Entry {}", ic.getMethod());

      return ic.proceed();



  }
}
```

Intercepting Both Directions

```
@Loggable
@Interceptor
public class LoggingInterceptor {

  @Inject
  private Logger logger;

  @AroundInvoke
  private Object intercept(InvocationContext ic) throws Exception {
    logger.info("> Entry {}", ic.getMethod());
    try {
      return ic.proceed();
    } finally {
      logger.info("< Exit {}", ic.getMethod());
    }
  }
}
```

Intercepting a Method

```
public class BookService {

  @Inject
  private NumberGenerator generator;

  @Loggable
  public Book createBook(String title) {
    return new Book(title, generator.generateNumber());
  }

  @Loggable
  public Book raisePrice(Book book) {
    book.setPrice(book.getPrice() * 2.5F);
    return book;
  }
}
```

Intercepting All Public Methods

```
@Loggable
public class BookService {

  @Inject
  private NumberGenerator generator;


  public Book createBook(String title) {
    return new Book(title, generator.generateNumber());
  }


  public Book raisePrice(Book book) {
    book.setPrice(book.getPrice() * 2.5F);
    return book;
  }
}
```

Enabling an Interceptor

```
<beans xmlns="http://xmlns.jcp.org/xml/ns/javaee"
       xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
                           http://xmlns.jcp.org/xml/ns/javaee/beans_1_1.xsd"
       version="1.1" bean-discovery-mode="all">

  <interceptors>
    <class>com.pluralsight.LoggingInterceptor</class>
    <class>com.pluralsight.Interceptor2</class>
    <class>com.pluralsight.Interceptor3</class>
  </interceptors>

</beans>
```

Interceptor

Interceptor binding

Interceptor

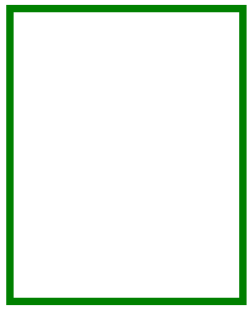
Enable in beans.xml

Validation

Bean Validation 1.1

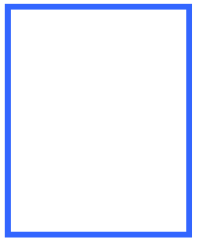
What Is Validation?

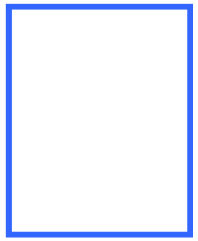
- Process, store, retrieve data
- Constrain our model
  - Is the address valid?
  - Is the email well formed?
  - Is the customer's name null?
- Ensure data is valid
- The service will behave correctly
- Give feedback to the users



When to Use Validation

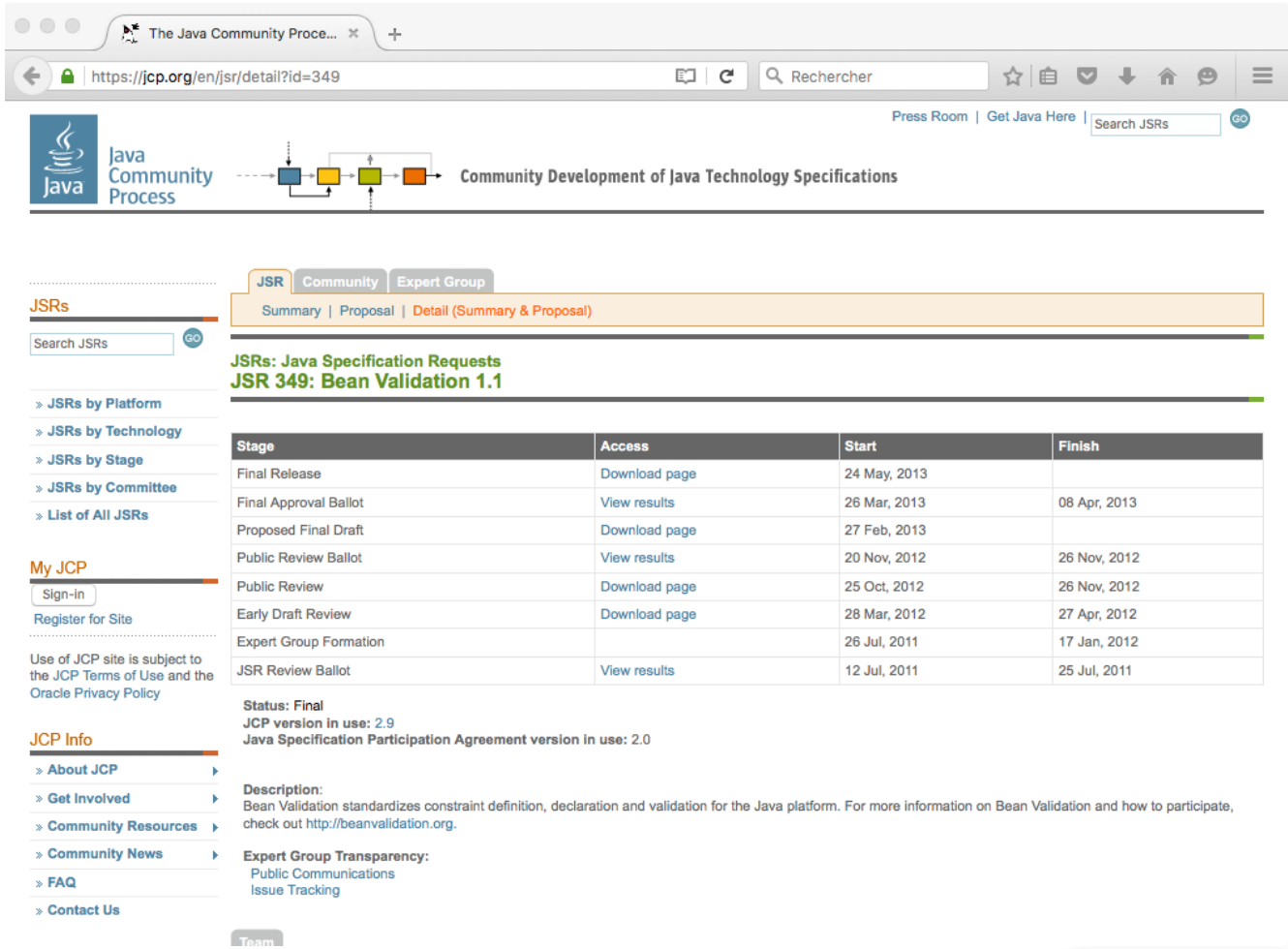
- Validation happens everywhere
  - Presentation layer
  - Business tier
  - Domain model
- Store valid data
- Processed later





Bean Validation Specification

- Bean Validation 1.1
- JSR 349
- http://jcp.org/en/jsr/detail?id=349



Bean Validation Implementations

- Hibernate Validator
- Apache BVal

Bean Validation

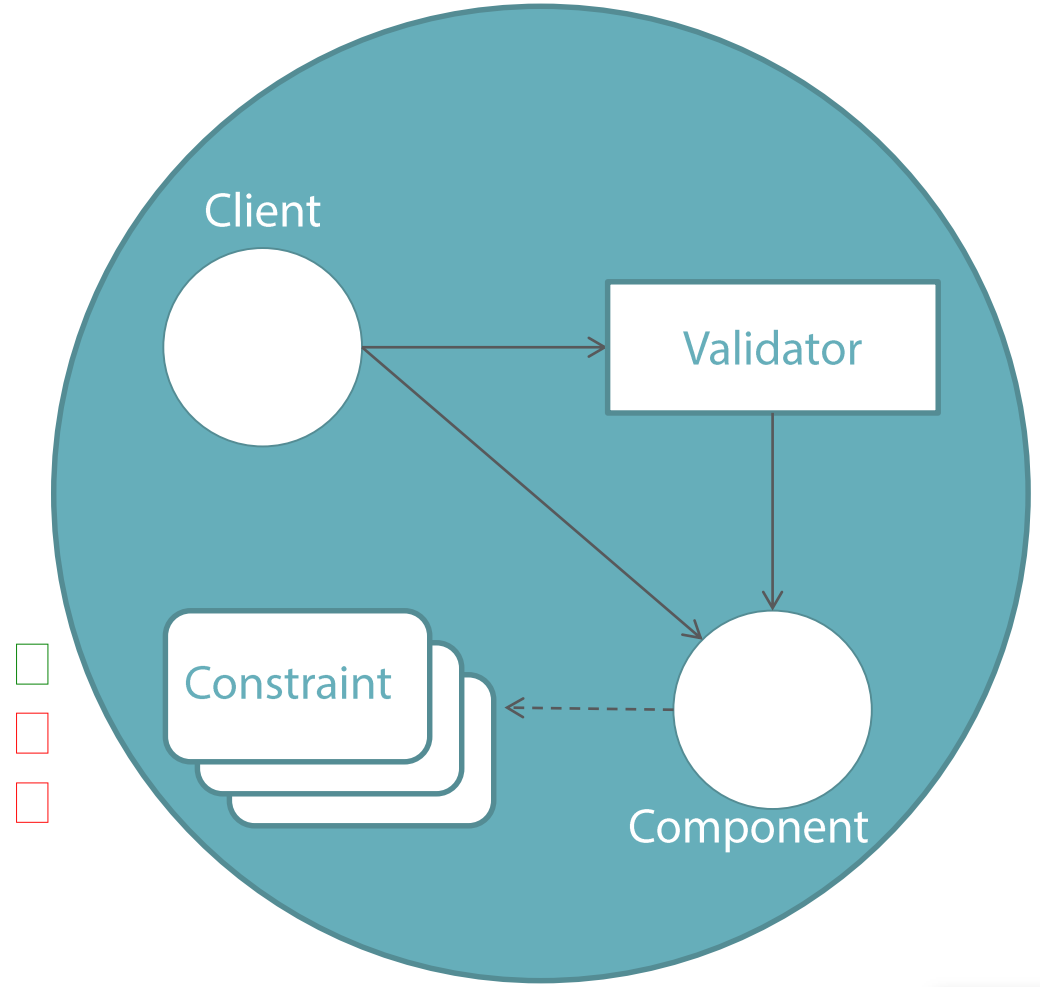
Validating input form

Sending feedback

Fix invalid values

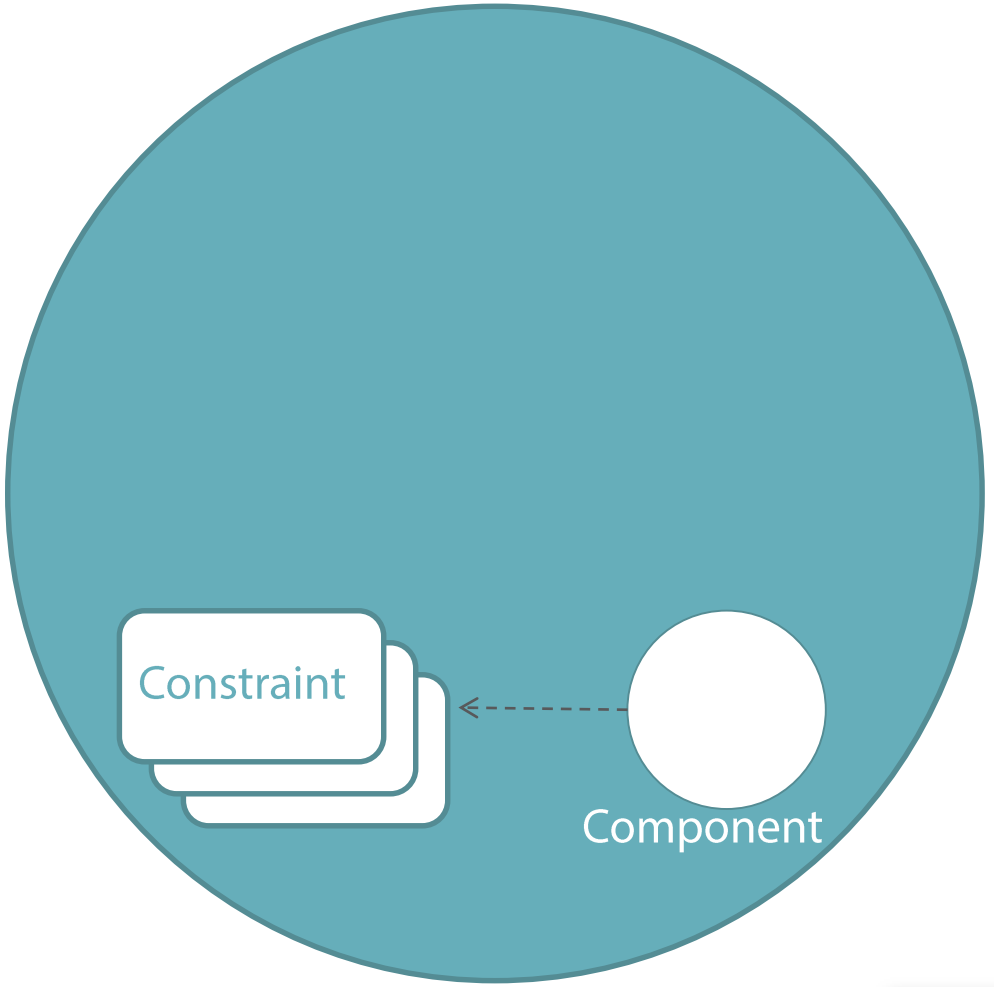
Understanding Bean Validation

- Constraints
  - On attributes
  - On method parameters
- Validator
- Violation
- Feedback



Constraint

- Business rule
- More or less complex
  - Is the address valid?
  - Is the email well formed?
  - Is the customer's name null?
  - Is the birth date in the past?
  - Total amount greater than previous quarter?
  - Annual revenue has 10% growth?
  - …

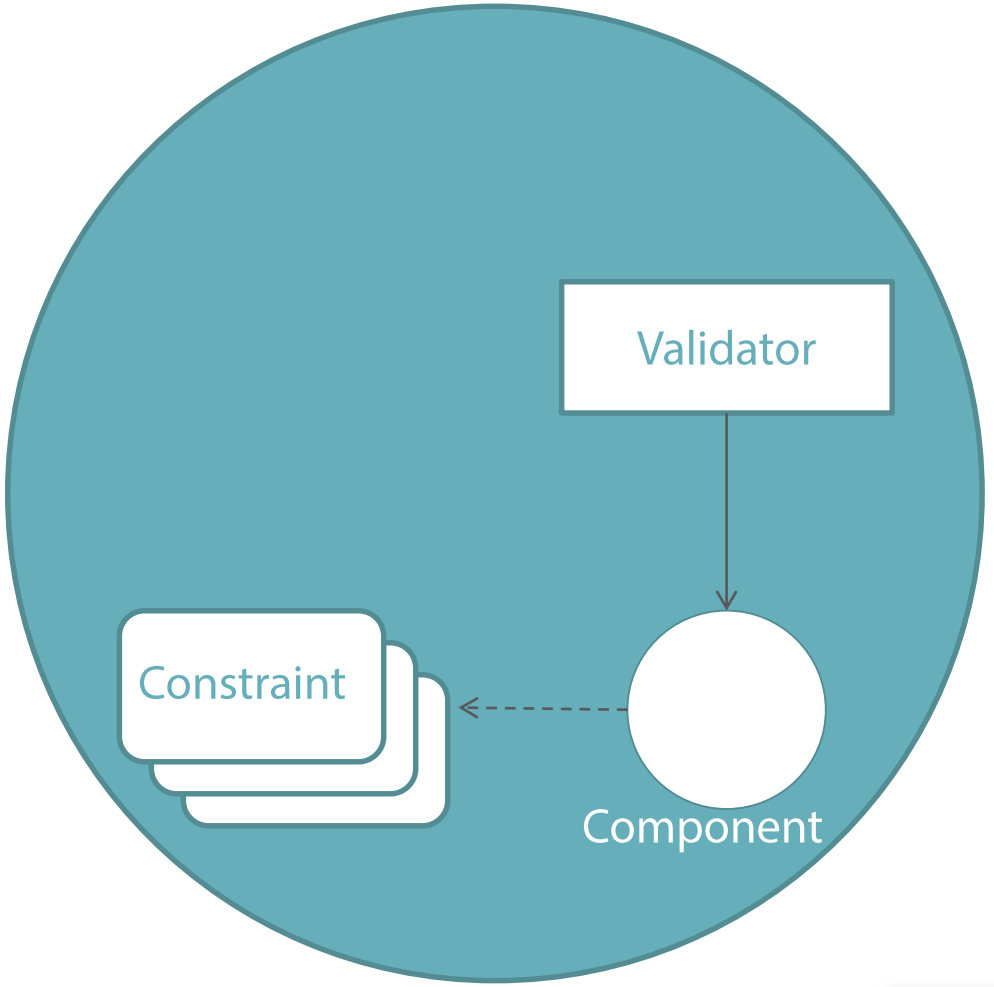


Constraint

```
public class Customer {

  @NotNull
  @Size(min = 4, max = 50)
  private String firstName;
  private String lastName;
  private String email;
  private String phoneNumber;
  @Past
  private Date dateOfBirth;

  // Constructors, getters & setters
}
```

Validator

- Entry point for validation
- Routine
  - Determines implementation
  - Validates the value
  - If value is valid, then continue
  - If value is invalid, add a constraint violation
- Returns a set of constraint violation
- Does not throw an exception

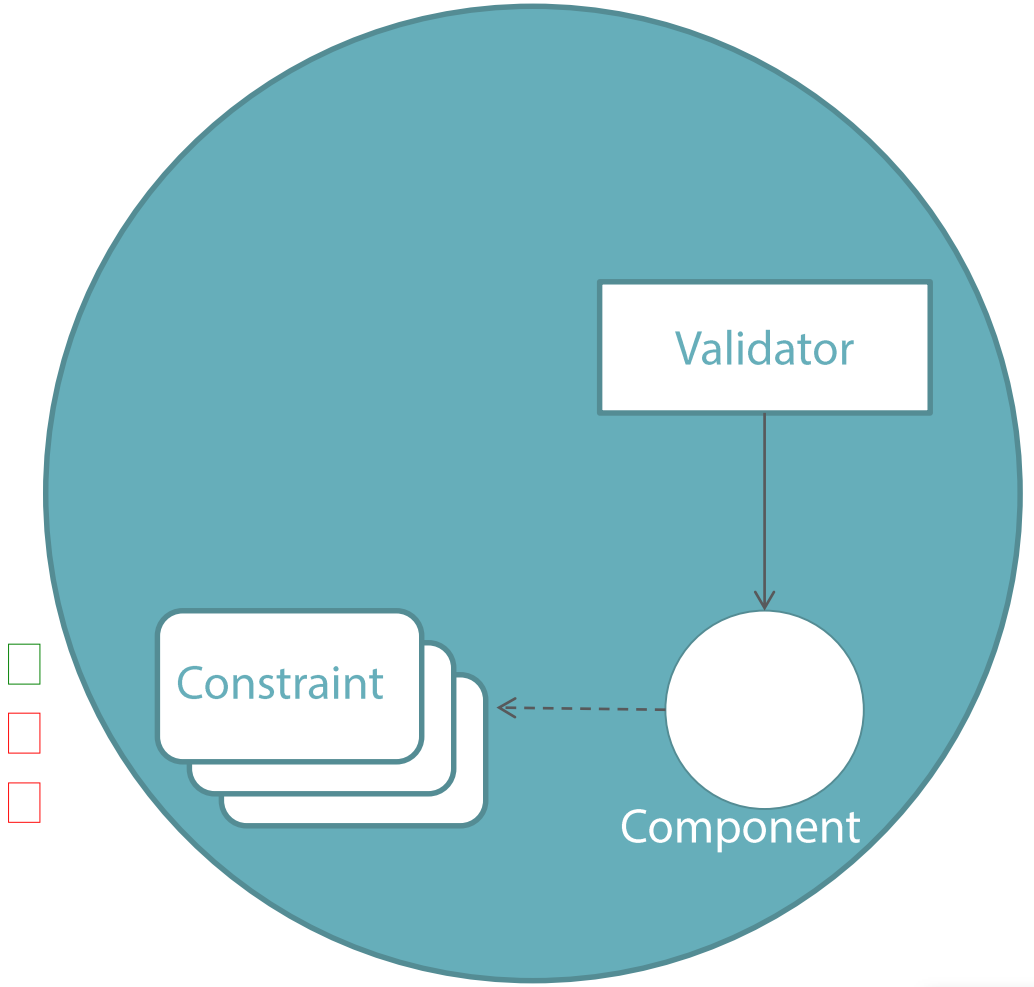


Validator

```
public class PurchaseOrderService {

  @Inject
  private Validator validator;



  public void createCustomer(Customer customer) {

    validator.validate(customer);



    // Customer is valid, now we can create it
  }
}
```

Constraint Violation

- Set of `ConstraintViolation`
- Single constraint failure
- `Set` is empty, the validation succeeds
- `Set` size == number of constraint failure
- API to get the the constraint failure
  - Error message
  - Property being validated
  - Invalid value



Constraint Violation

```
public class PurchaseOrderService {

  @Inject
  private Validator validator;

  private Set<ConstraintViolation<Customer>> violations;

  public void createCustomer(Customer customer) {

                validator.validate(customer);
    if (violations.size() > 0) throw new
        ConstraintViolationException(violations);

    // Customer is valid, now we can create it
  }
}
```

Deployment Descriptor

```
<constraint-mappings>
  <bean class="com.pluralsight.Customer">
    <field name="lastName">
      <constraint annotation="javax.validation.constraints.Size">
        <element name="min">10</element>
      </constraint>
    </field>
  </bean>
</constraint-mappings>
```

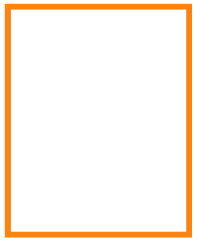
constraints.xml
validation.xml

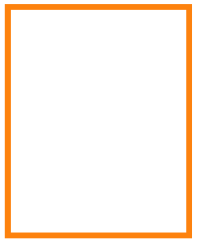
Bean Validation Packages

Package	Description
`javax.validation`	Core API
`javax.validation.bootstrap`	Classes used to bootstrap Bean Validation
`javax.validation.constraints`	Built-in constraints
`javax.validation.groups`	Default Bean Validation group
`javax.validation.metadata`	Metadata repository

Built-in Constraints

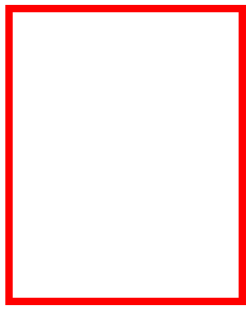
- Common constraints
- Create our own business constraints
- “Constrain once, validate anywhere”
- Reuse the constraints in any layer
  - Attributes
  - Beans
  - Constructors
  - Methods



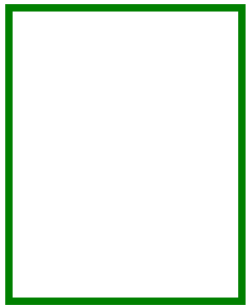


Boolean

```
@AssertFalse
```

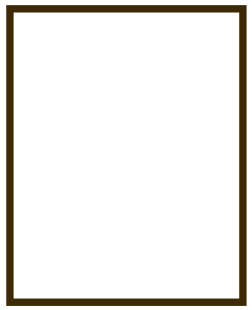


```
@AssertTrue
```



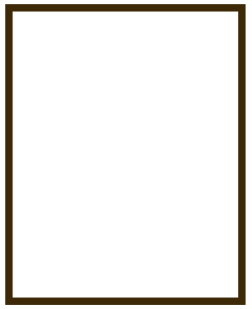
Size

```
@Size
@Digits
@Max, @Min
@DecimalMax, @DecimalMin
```



Date

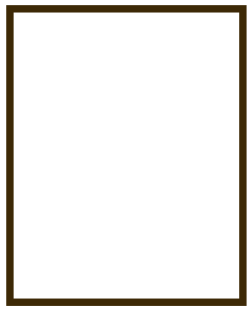
```
@Future
```



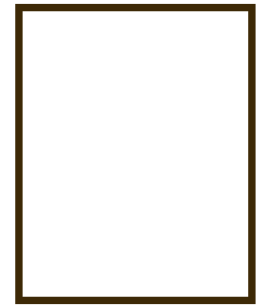
```
@Past
```

Object

@Null



@NotNull



RegEx

```
@Pattern
```

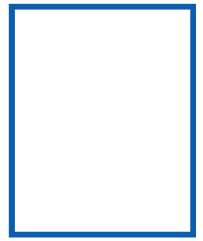
```
^[0-9]+abc$
```

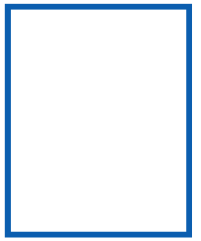
Applying Built-in Constraints

```
public class Book {

  @Pattern(regexp = "^(97(8|9))?\\d{9}(\\d|X)$")
  private String isbn;
  @NotNull
  private String title;
  @NotNull @Min(2)
  private Float price;
  @Past
  private Date creationDate;
  @Past
  private Date publicationDate;

  public Book ( @NotNull String title) {
    this.title = title;
  }
}
```

Defining Our Own Constraints

- An annotation defining the constraint
  - Meaningful name (e.g. `@Email`)
  - Optional attributes (e.g. `@Size(min=2, max=2000)`)
- A list of classes implementing the algorithm of validation
- Constraint on a given type
  - `String`
  - `Integer`
  - Collection





Constraint Annotation

```
@Target(TYPE)
@Retention(RUNTIME)
@Documented
@Constraint(validatedBy = ChronologicalDatesValidator.class)
public @interface ChronologicalDates {

  String message() default "dates have to be in chronological order";

  Class<?>[] groups() default {};

  Class<? extends Payload>[] payload() default {};
}
```

Constraint Implementation

```
public class ChronologicalDatesValidator implements
             ConstraintValidator<ChronologicalDates, Book> {

  @Override
  public void initialize(ChronologicalDates annotation) {
  }

  @Override
  public boolean isValid(Book book, ConstraintValidatorContext context) {

    return book.getCreationDate().getTime() < book.getPublication().getTime();
  }
}
```

Applying Constraint

```
@ChronologicalDates
public class Book {

  @Pattern(regexp = "^(97(8|9))?\\d{9}(\\d|X)$")
  private String isbn;
  @NotNull
  private String title;
  @NotNull @Min(2)
  private Float price;
  @Past
  private Date dateOfCreation;
  @Past
  private Date dateOfPublication;

  public Book (@NotNull String title) {
    this.title = title;
  }
}
```

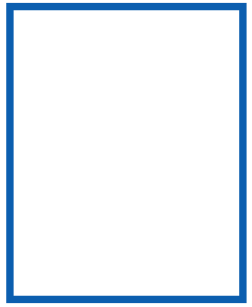
Bean Validation

Built-in constraint

User object attributes

Signup form

Summary



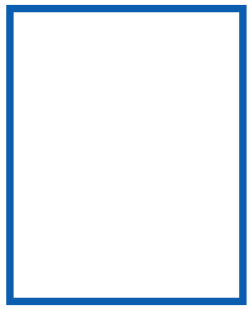
Common specifications

Context & Dependency Injection

Interceptor

Bean Validation

What's Next



Business tier

Persistence

Transaction management

Batch processing