جامعة طرابلس
University of Tripoli
1957

Java Bean Validation

What is Validation

- عند معالجة البيانات او حفظها او استرجاعها نحتاج تطبيق لعض القيود لنتحقق منها لنضمن ان النظام المستخدم لها سيعمل بشكل صحيح فمثلاً:

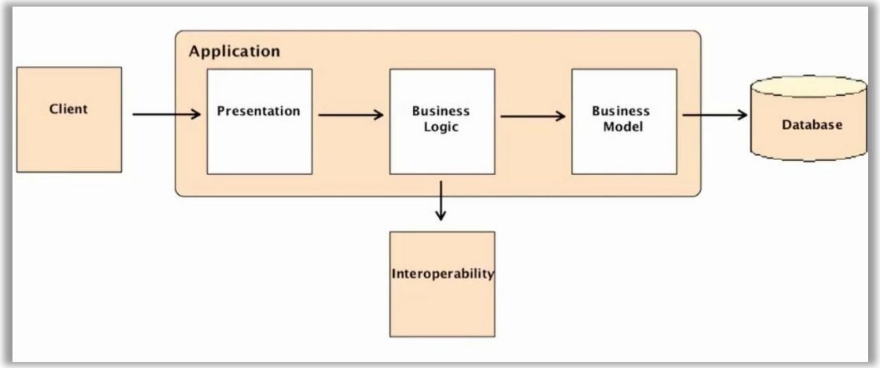
- عند التعامل مع بيانات زبون يتم استخدام بعض القيود للتحقق من سلامة بياناته مثل:
  - هل العنوان صحيح
  - البريد الالكتروني بالصيغة الصحيحة
  - الاسم ليس null
  - تاريخ الميلاد في الماضي

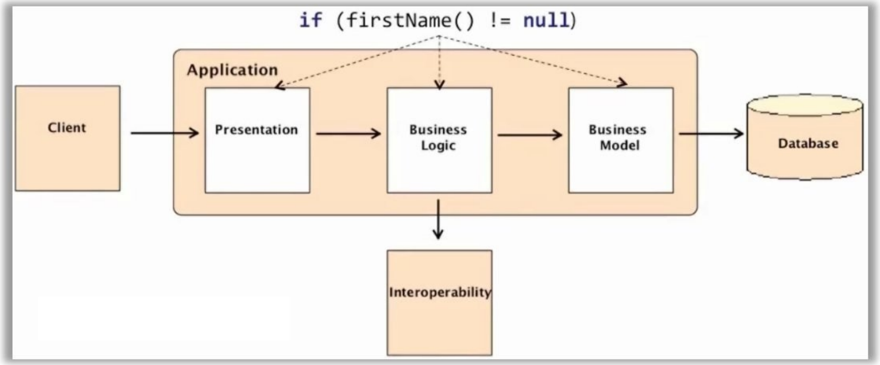
Java Bean Validation

- عبارة عن Java Specification تسمح بعملية التحقق من البيانات من خلال استخدام annotation .
- تقدم مجموعة من built in constraints .
- تسمح بأنشاء custom constraints .
- تستخدم هذه constrains مع كل من:
  - Fields
  - Methods
  - Constructors
  - Parameters
  - Types

أين تستخدم Bean Validation

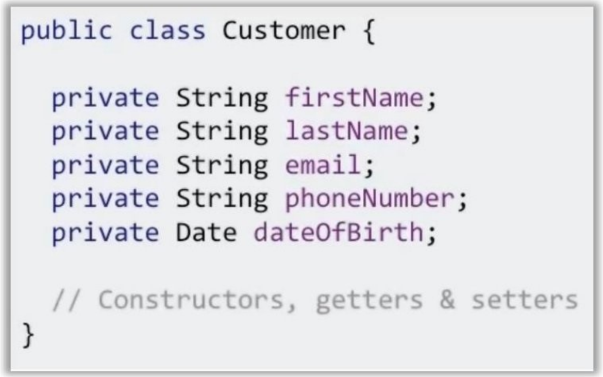
- تستخدم bean validation مع كل من:
- EJB
- JPA
- JSF
- JAX-RS



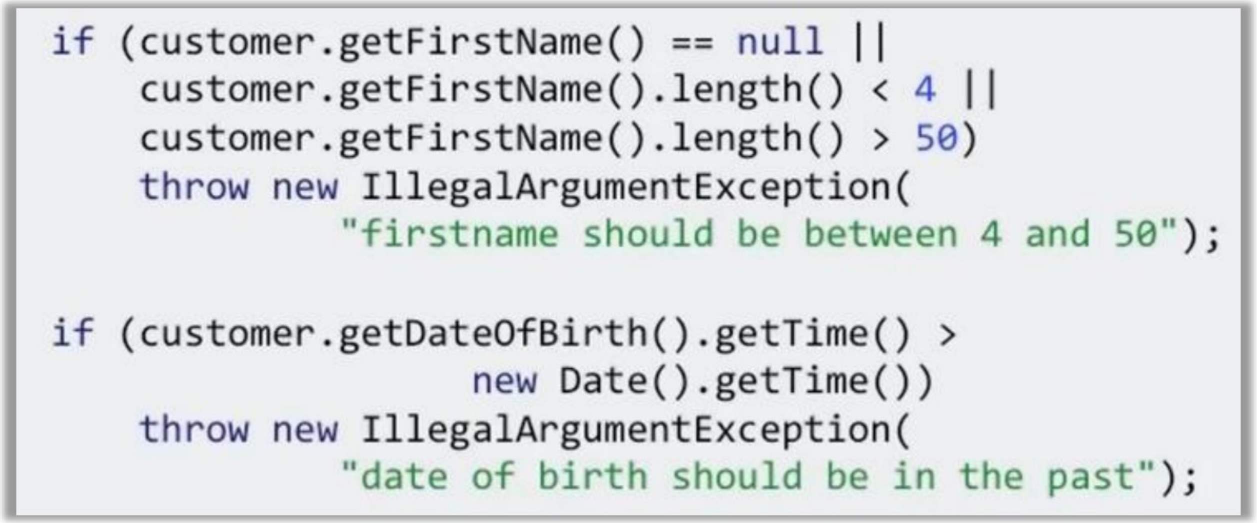


Example

- المثال التالي يبين التحقق من البيانات بالطريقة العادية وباستخدام Bean Validation.

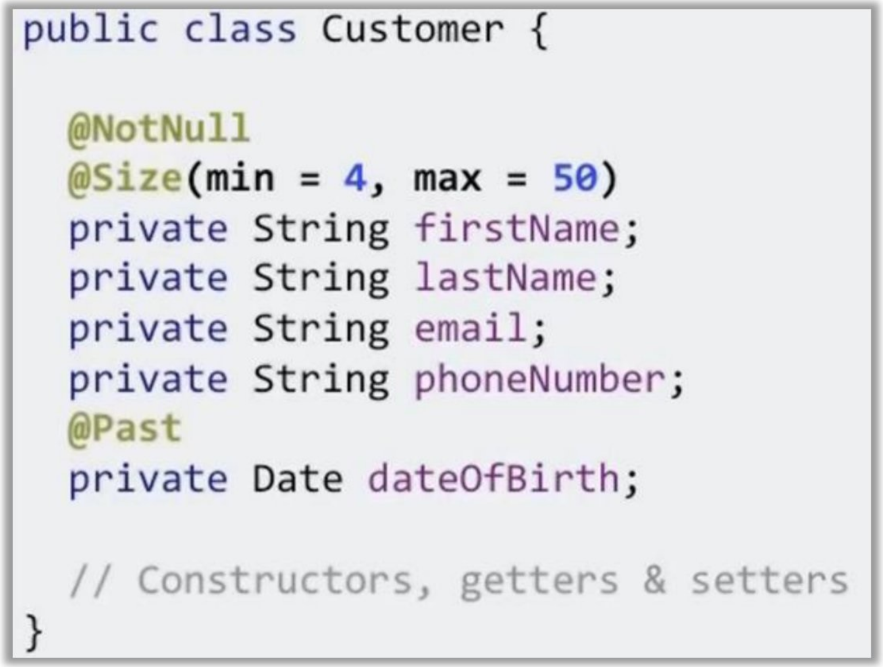


```java
public class Customer {

  private String firstName;
  private String lastName;
  private String email;
  private String phoneNumber;
  private Date dateOfBirth;

  // Constructors, getters & setters
}
```



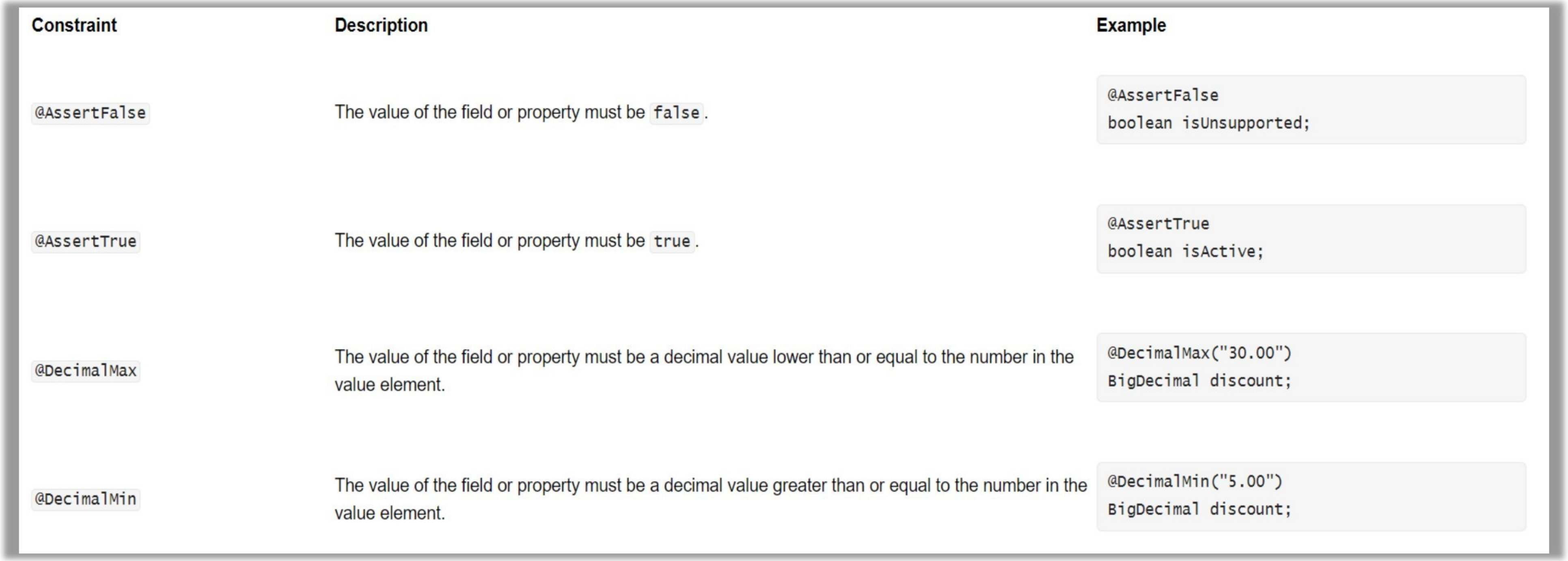
```java
if (customer.getFirstName() == null ||
    customer.getFirstName().length() < 4 ||
    customer.getFirstName().length() > 50)
    throw new IllegalArgumentException(
             "firstname should be between 4 and 50");

if (customer.getDateOfBirth().getTime() >
                   new Date().getTime())
    throw new IllegalArgumentException(
             "date of birth should be in the past");
```

Example



```
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
}
```

Built in Constraints



Constraint	Description	Example

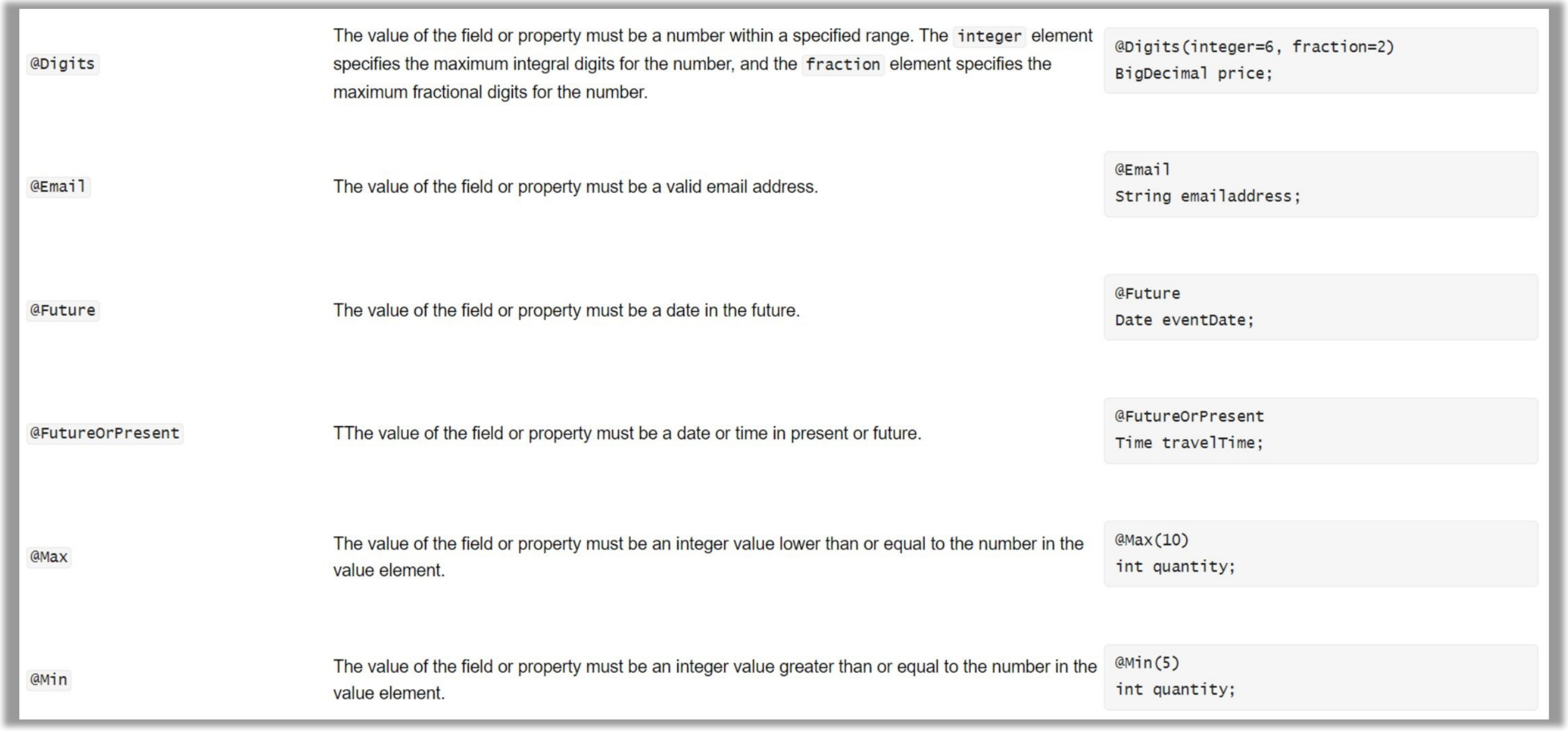
`@AssertFalse`	The value of the field or property must be `false`.	`@AssertFalse`
		`boolean isUnsupported;`

`@AssertTrue`	The value of the field or property must be `true`.	`@AssertTrue`
		`boolean isActive;`

`@DecimalMax`	The value of the field or property must be a decimal value lower than or equal to the number in the value element.	`@DecimalMax("30.00")`
		`BigDecimal discount;`

`@DecimalMin`	The value of the field or property must be a decimal value greater than or equal to the number in the value element.	`@DecimalMin("5.00")`
		`BigDecimal discount;`

Built in Constraints



`@Digits`	The value of the field or property must be a number within a specified range. The `integer` element specifies the maximum integral digits for the number, and the `fraction` element specifies the maximum fractional digits for the number.	`@Digits(integer=6, fraction=2)`
		`BigDecimal price;`

`@Email`	The value of the field or property must be a valid email address.	`@Email`
		`String emailaddress;`

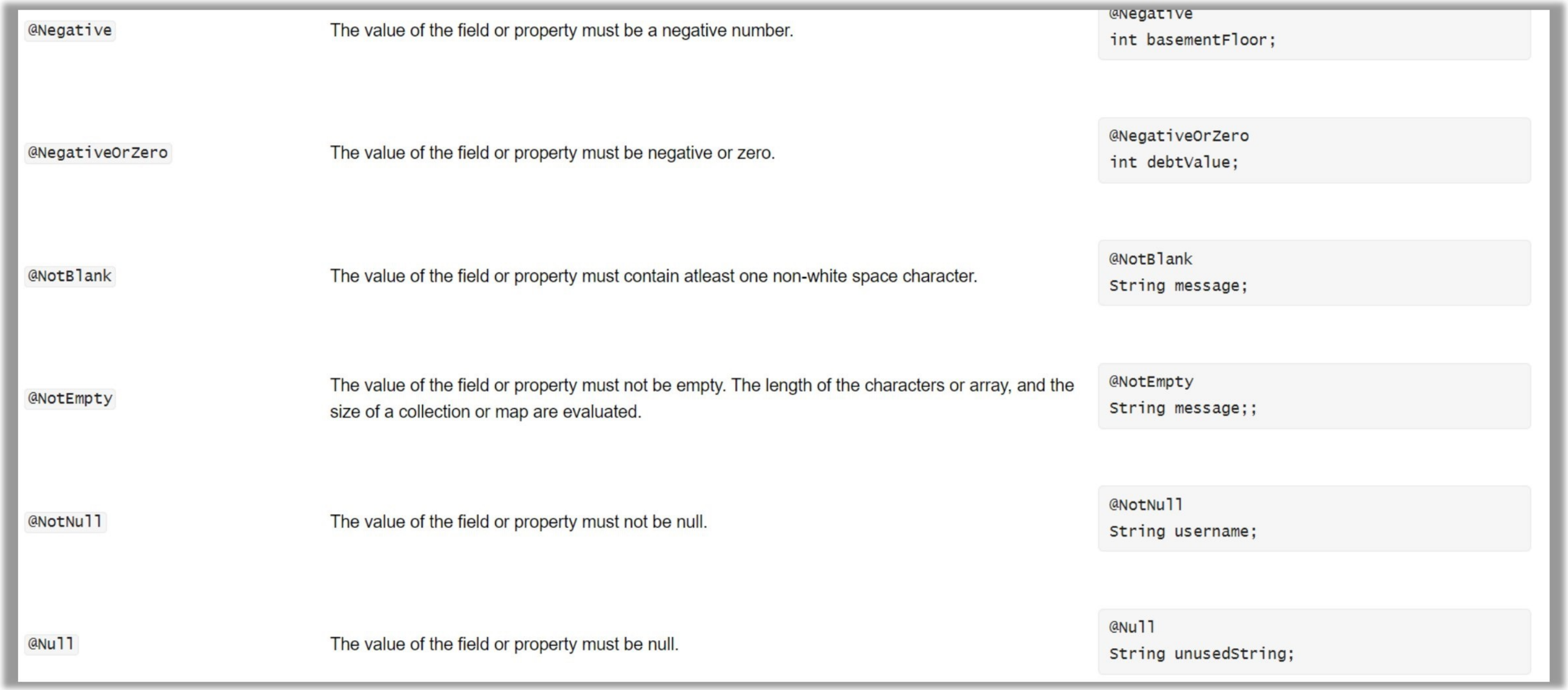
`@Future`	The value of the field or property must be a date in the future.	`@Future`
		`Date eventDate;`

`@FutureOrPresent`	TThe value of the field or property must be a date or time in present or future.	`@FutureOrPresent`
		`Time travelTime;`

`@Max`	The value of the field or property must be an integer value lower than or equal to the number in the value element.	`@Max(10)`
		`int quantity;`

`@Min`	The value of the field or property must be an integer value greater than or equal to the number in the value element.	`@Min(5)`
		`int quantity;`

Built in Constraints



`@Negative`	The value of the field or property must be a negative number.	`@Negative`
		`int basementFloor;`

`@NegativeOrZero`	The value of the field or property must be negative or zero.	`@NegativeOrZero`
		`int debtValue;`

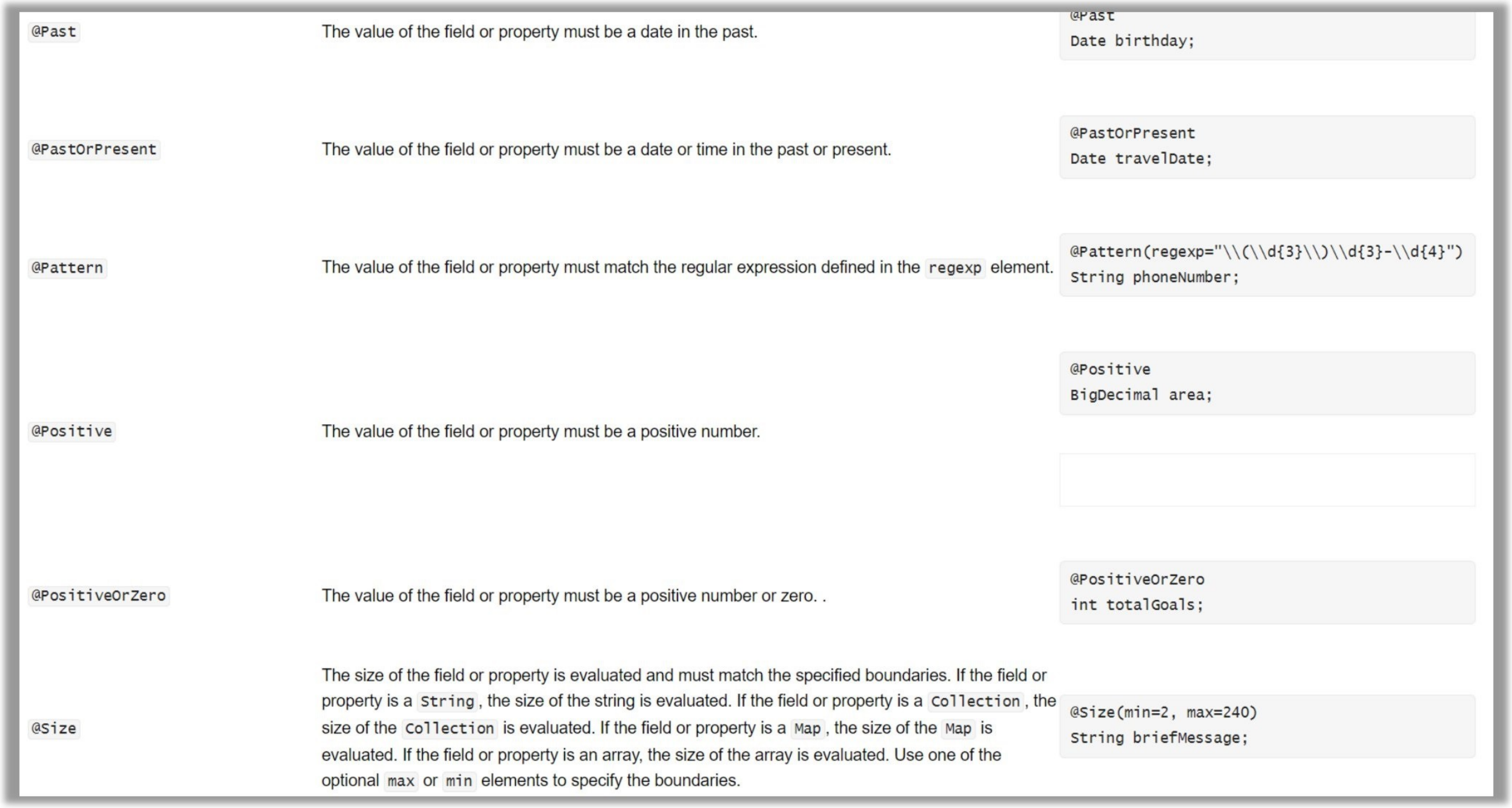
`@NotBlank`	The value of the field or property must contain atleast one non-white space character.	`@NotBlank`
		`String message;`

`@NotEmpty`	The value of the field or property must not be empty. The length of the characters or array, and the size of a collection or map are evaluated.	`@NotEmpty`
		`String message;;`

`@NotNull`	The value of the field or property must not be null.	`@NotNull`
		`String username;`

`@Null`	The value of the field or property must be null.	`@Null`
		`String unusedString;`

Built in Constraints



`@Past`	The value of the field or property must be a date in the past.	`@Past`
		`Date birthday;`

`@PastOrPresent`	The value of the field or property must be a date or time in the past or present.	`@PastOrPresent`
		`Date travelDate;`

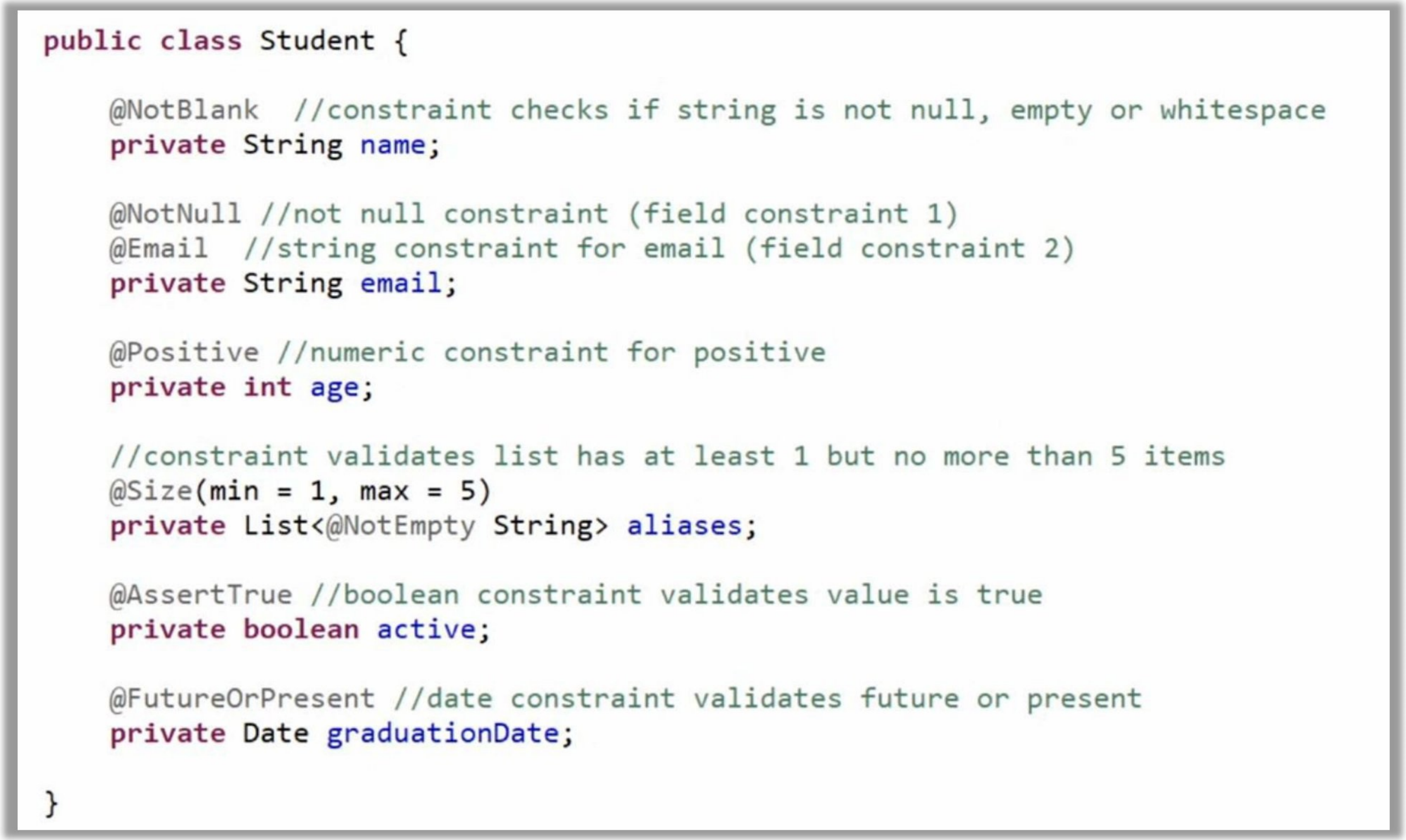
`@Pattern`	The value of the field or property must match the regular expression defined in the `regexp` element.	`@Pattern(regexp="\\(\\d{3}\\)\\d{3}-\\d{4}")`
		`String phoneNumber;`

`@Positive`	The value of the field or property must be a positive number.	`@Positive`
		`BigDecimal area;`

`@PositiveOrZero`	The value of the field or property must be a positive number or zero. .	`@PositiveOrZero`
		`int totalGoals;`

`@Size`	The size of the field or property is evaluated and must match the specified boundaries. If the field or property is a `String`, the size of the string is evaluated. If the field or property is a `Collection`, the size of the `Collection` is evaluated. If the field or property is a `Map`, the size of the `Map` is evaluated. If the field or property is an array, the size of the array is evaluated. Use one of the optional `max` or `min` elements to specify the boundaries.	`@Size(min=2, max=240)`
		`String briefMessage;`

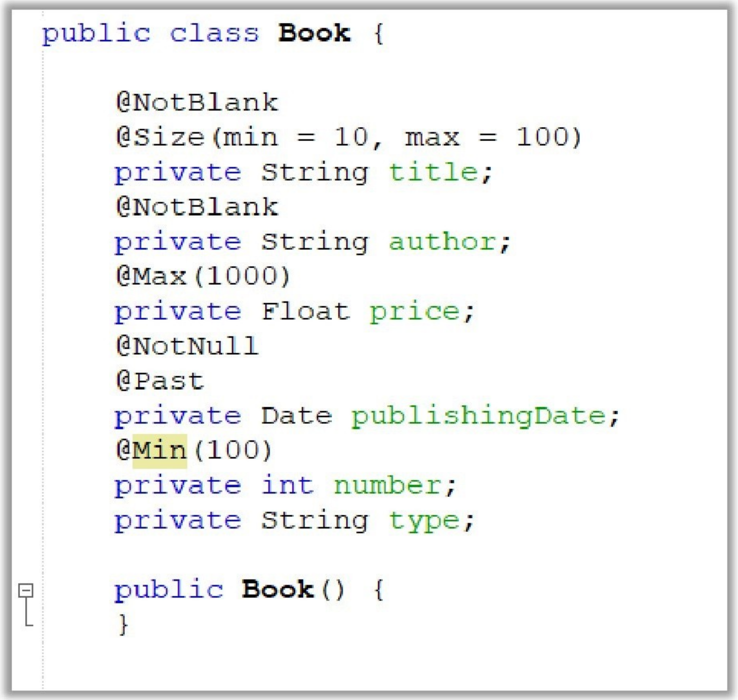
Example



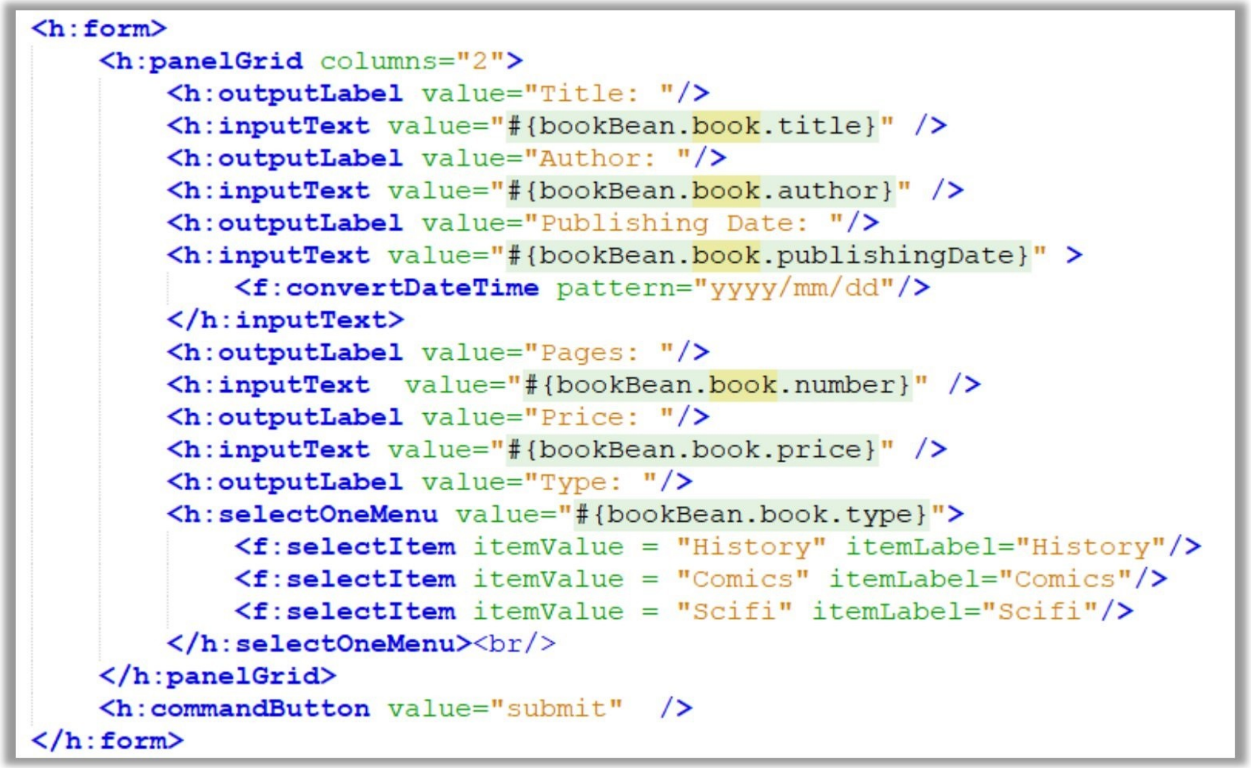
```java
public class Student {

    @NotBlank  //constraint checks if string is not null, empty or whitespace
    private String name;

    @NotNull //not null constraint (field constraint 1)
    @Email  //string constraint for email (field constraint 2)
    private String email;

    @Positive //numeric constraint for positive
    private int age;

    //constraint validates list has at least 1 but no more than 5 items
    @Size(min = 1, max = 5)
    private List<@NotEmpty String> aliases;

    @AssertTrue //boolean constraint validates value is true
    private boolean active;

    @FutureOrPresent //date constraint validates future or present
    private Date graduationDate;

}
```

Example

- المثال التالي يبين استخدام Built in constraints .

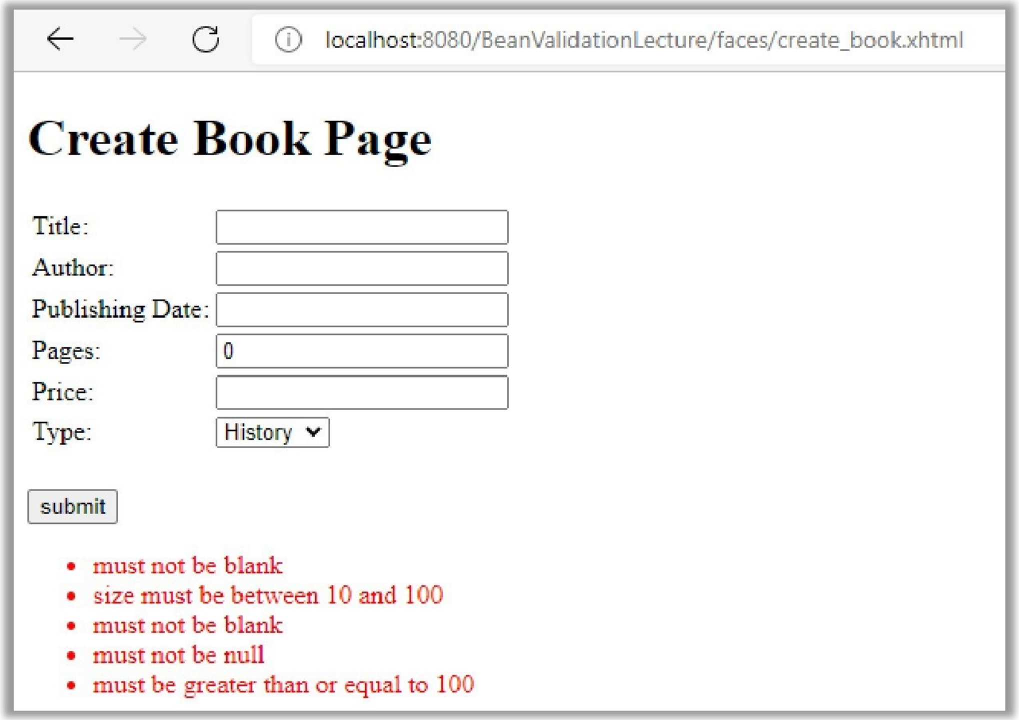


```java
public class Book {

    @NotBlank
    @Size(min = 10, max = 100)
    private String title;
    @NotBlank
    private String author;
    @Max(1000)
    private Float price;
    @NotNull
    @Past
    private Date publishingDate;
    @Min(100)
    private int number;
    private String type;

    public Book() {
    }
```



```xml
<h:form>
    <h:panelGrid columns="2">
        <h:outputLabel value="Title: "/>
        <h:inputText value="#{bookBean.book.title}" />
        <h:outputLabel value="Author: "/>
        <h:inputText value="#{bookBean.book.author}" />
        <h:outputLabel value="Publishing Date: "/>
        <h:inputText value="#{bookBean.book.publishingDate}" >
            <f:convertDateTime pattern="yyyy/mm/dd"/>
        </h:inputText>
        <h:outputLabel value="Pages: "/>
        <h:inputText  value="#{bookBean.book.number}" />
        <h:outputLabel value="Price: "/>
        <h:inputText value="#{bookBean.book.price}" />
        <h:outputLabel value="Type: "/>
        <h:selectOneMenu value="#{bookBean.book.type}">
            <f:selectItem itemValue = "History" itemLabel="History"/>
            <f:selectItem itemValue = "Comics" itemLabel="Comics"/>
            <f:selectItem itemValue = "Scifi" itemLabel="Scifi"/>
        </h:selectOneMenu><br/>
    </h:panelGrid>
    <h:commandButton value="submit"  />
</h:form>
```

Example


localhost:8080/BeanValidationLecture/faces/create_book.xhtml

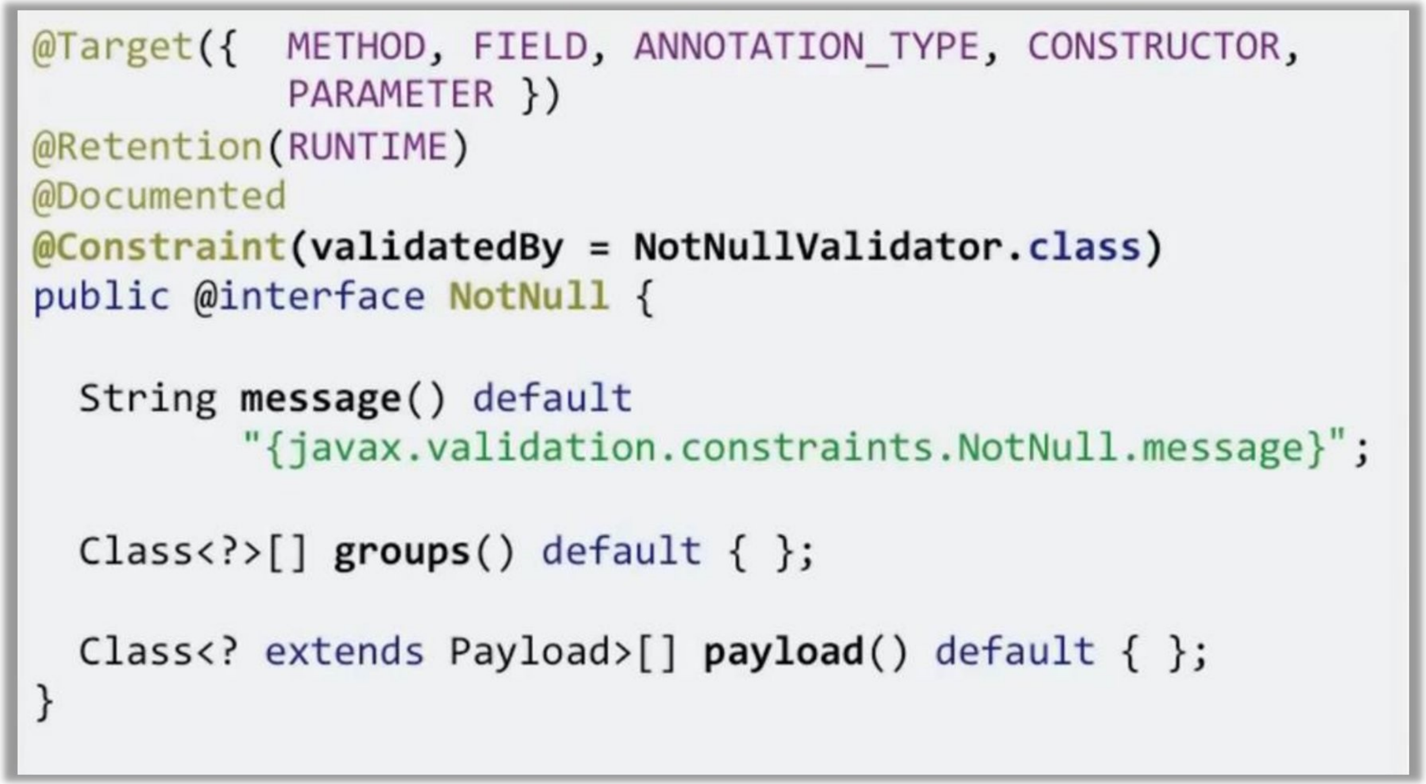
Create Book Page

Title:	
Author:	
Publishing Date:	
Pages:	0
Price:	
Type:	History

- must not be blank
- size must be between 10 and 100
- must not be blank
- must not be null
- must be greater than or equal to 100

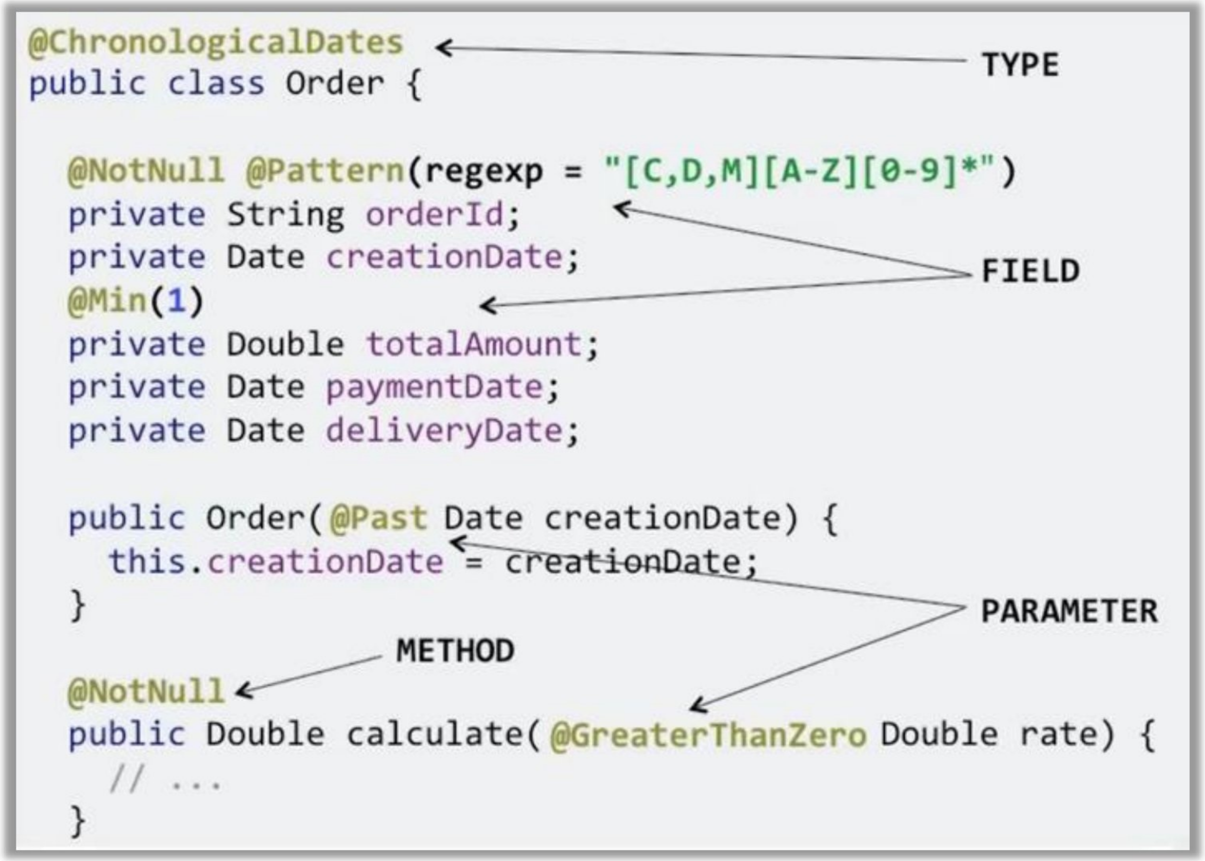
Constraints Annotation

- عبارة عن Java Annotation يتم فيها تحديد Validator class and Targets والرسالة التي ستظهر عندما يفشل Validation .



```
@Target({  METHOD, FIELD, ANNOTATION_TYPE, CONSTRUCTOR,
           PARAMETER })
@Retention(RUNTIME)
@Documented
@Constraint(validatedBy = NotNullValidator.class)
public @interface NotNull {

  String message() default
          "{javax.validation.constraints.NotNull.message}";

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

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

Applying Constraints to a Target



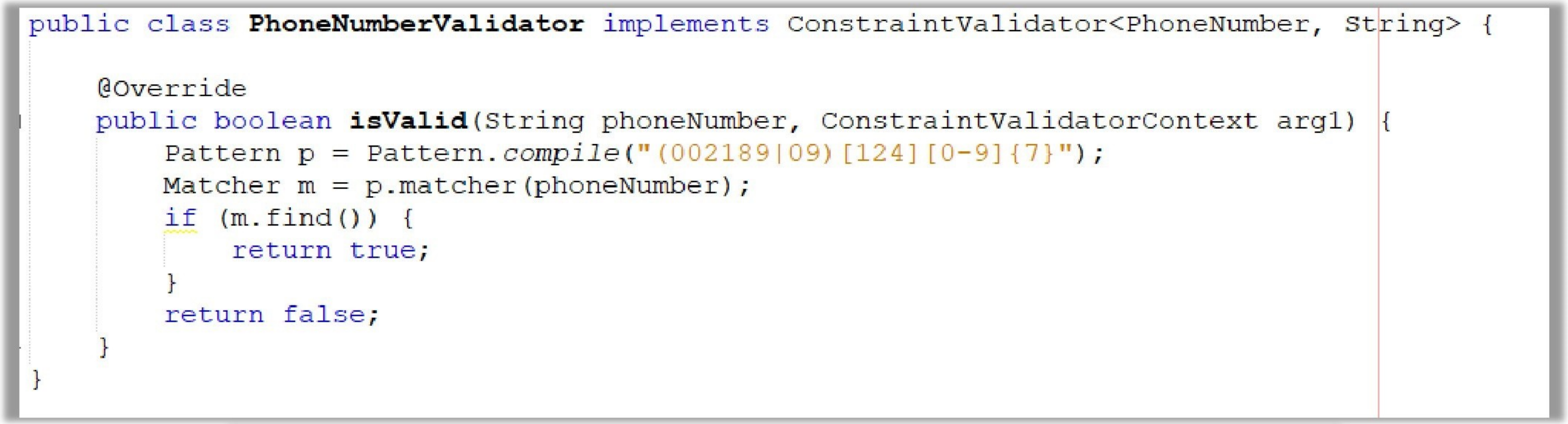
```
@ChronologicalDates
public class Order {

  @NotNull @Pattern(regexp = "[C,D,M][A-Z][0-9]*")
  private String orderId;
  private Date creationDate;
  @Min(1)
  private Double totalAmount;
  private Date paymentDate;
  private Date deliveryDate;

  public Order(@Past Date creationDate) {
    this.creationDate = creationDate;
  }

  @NotNull
  public Double calculate(@GreaterThanZero Double rate) {
    // ...
  }
```

أنشاء Custom Constraint

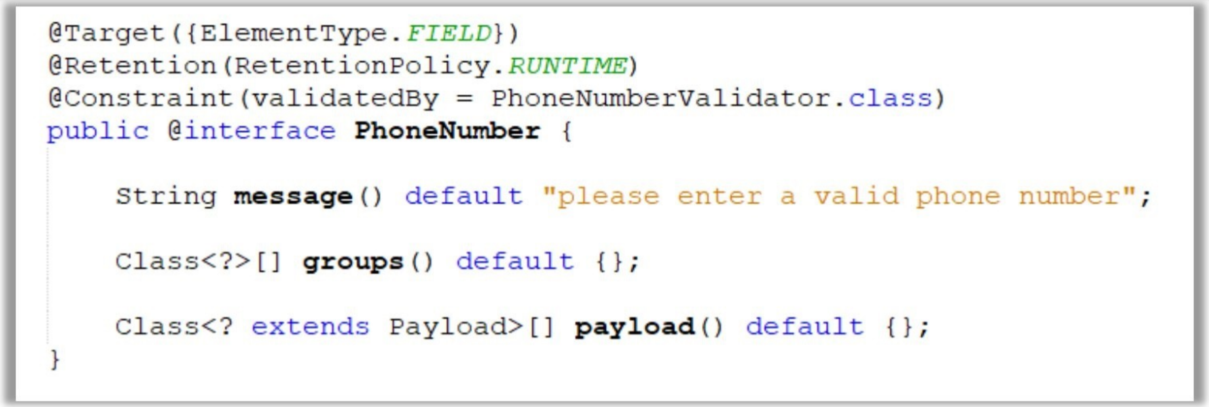
- يتم انشاء custom constraints من خطوتين:
- استعمال ConstraintValidator interface.
- أنشاء annotation خاص بهذا constraint .

Example

- المثال التالي يبين انشاء constraint للتحقق من ارقام التليفونات لتكون ارقام تليفون ليبية.

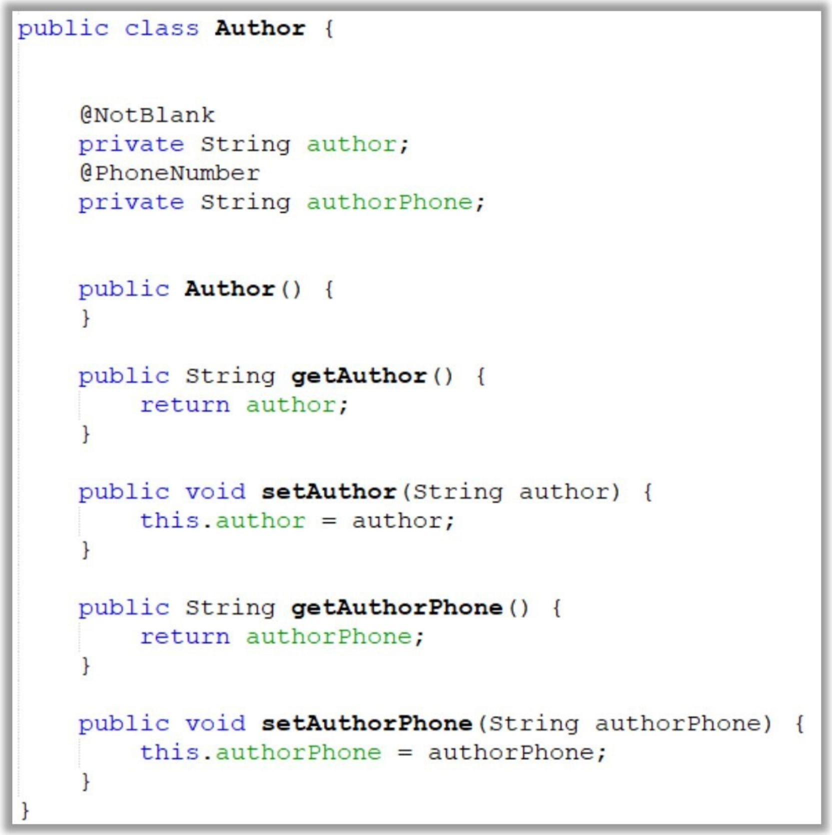


```
public class PhoneNumberValidator implements ConstraintValidator<PhoneNumber, String> {

    @Override
    public boolean isValid(String phoneNumber, ConstraintValidatorContext arg1) {
        Pattern p = Pattern.compile("(002189|09)[124][0-9]{7}");
        Matcher m = p.matcher(phoneNumber);
        if (m.find()) {
            return true;
        }
        return false;
    }
}
```

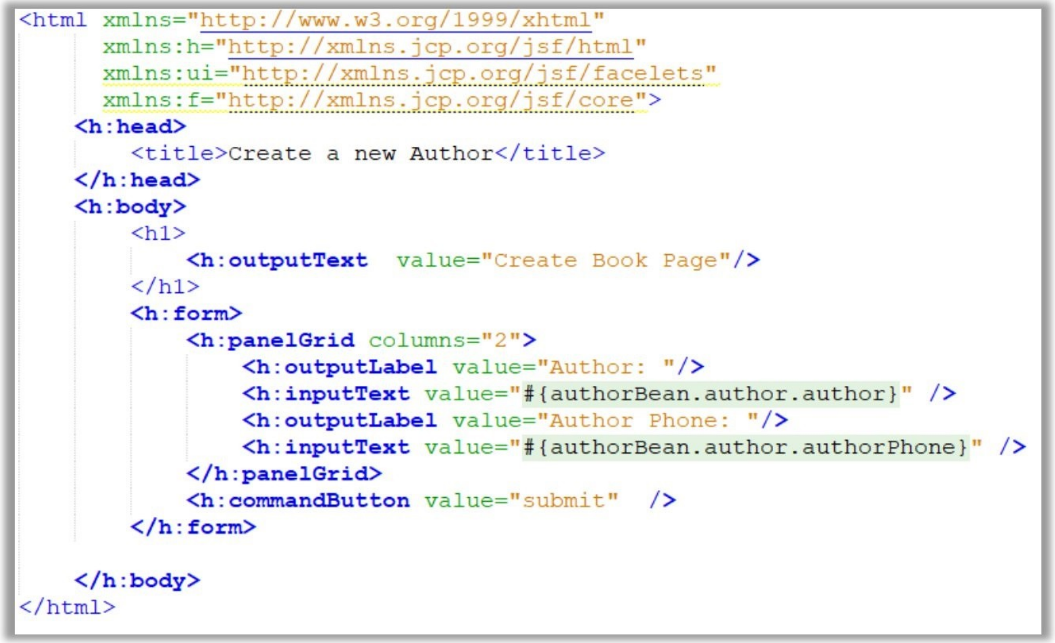


```
@Target({ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
@Constraint(validatedBy = PhoneNumberValidator.class)
public @interface PhoneNumber {

    String message() default "please enter a valid phone number";

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

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

Example

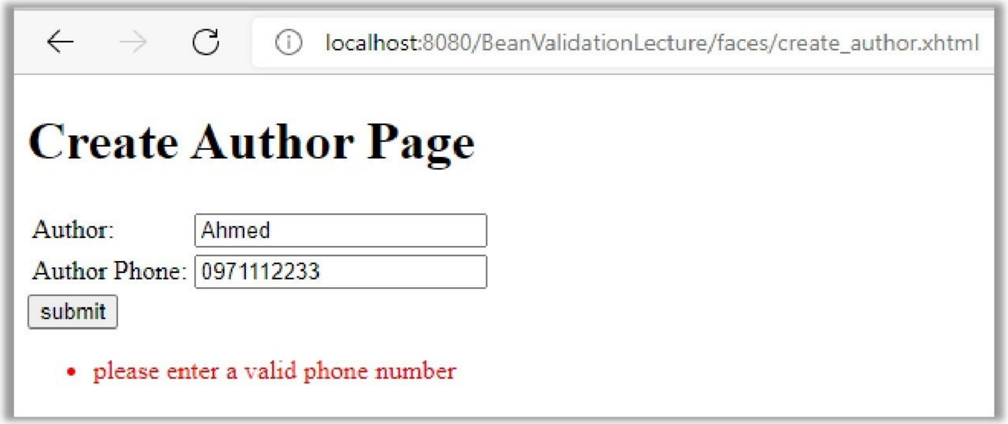


```java
public class Author {


    @NotBlank
    private String author;
    @PhoneNumber
    private String authorPhone;


    public Author() {
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getAuthorPhone() {
        return authorPhone;
    }

    public void setAuthorPhone(String authorPhone) {
        this.authorPhone = authorPhone;
    }
}
```



```html
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:f="http://xmlns.jcp.org/jsf/core">
    <h:head>
        <title>Create a new Author</title>
    </h:head>
    <h:body>
        <h1>
            <h:outputText  value="Create Book Page"/>
        </h1>
        <h:form>
            <h:panelGrid columns="2">
                <h:outputLabel value="Author: "/>
                <h:inputText value="#{authorBean.author.author}" />
                <h:outputLabel value="Author Phone: "/>
                <h:inputText value="#{authorBean.author.authorPhone}" />
            </h:panelGrid>
            <h:commandButton value="submit"  />
        </h:form>

    </h:body>
</html>
```

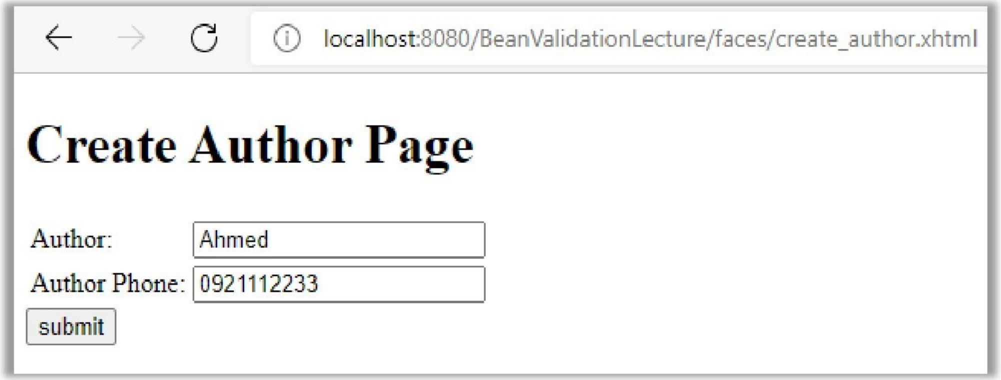
Example


localhost:8080/BeanValidationLecture/faces/create_author.xhtml

Create Author Page

Author: Ahmed
Author Phone: 0971112233
submit

- please enter a valid phone number

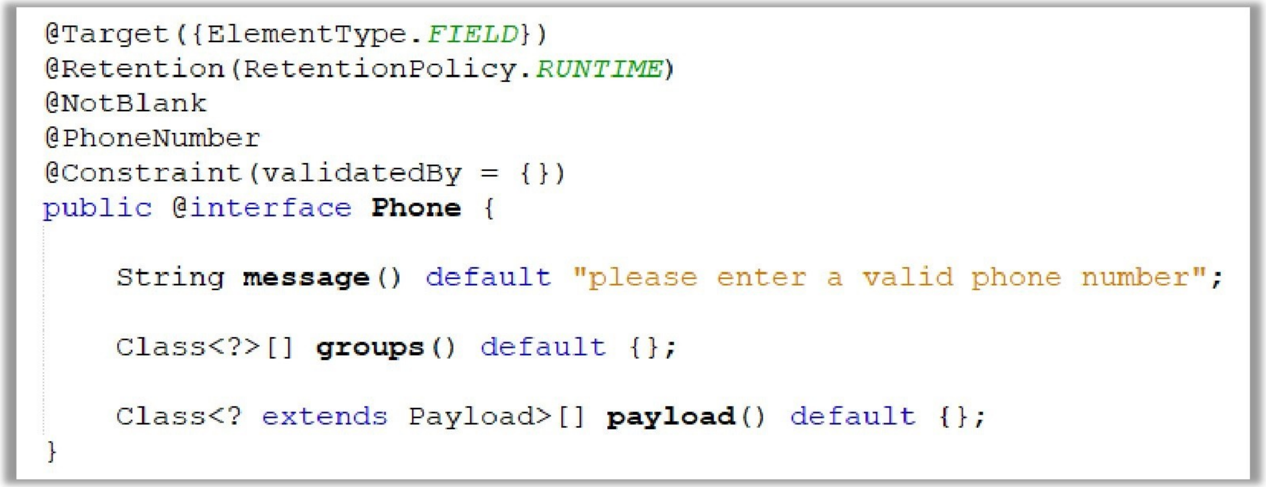

localhost:8080/BeanValidationLecture/faces/create_author.xhtml

Create Author Page

Author: Ahmed
Author Phone: 0921112233
submit

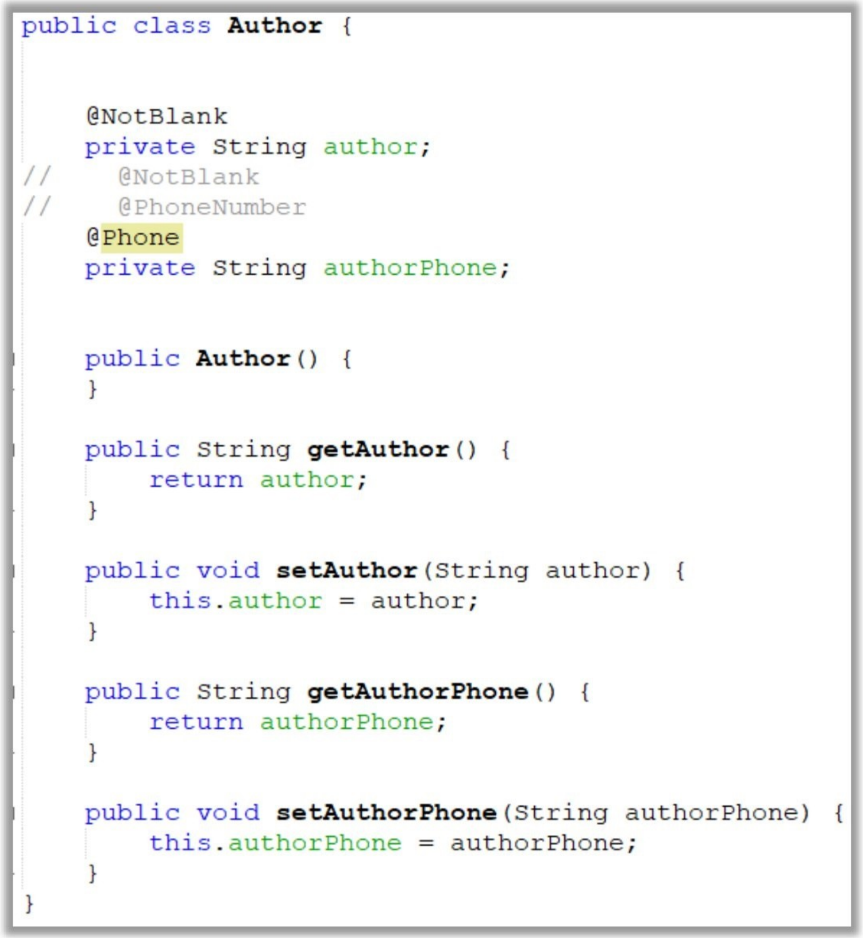
Constraints Composition

- يتم فيها تجميع مجموعة من constraints تحت annotation جديد يتم استخدامه بدلاً منهم.



```
@Target({ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
@NotBlank
@PhoneNumber
@Constraint(validatedBy = {})
public @interface Phone {

    String message() default "please enter a valid phone number";

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

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

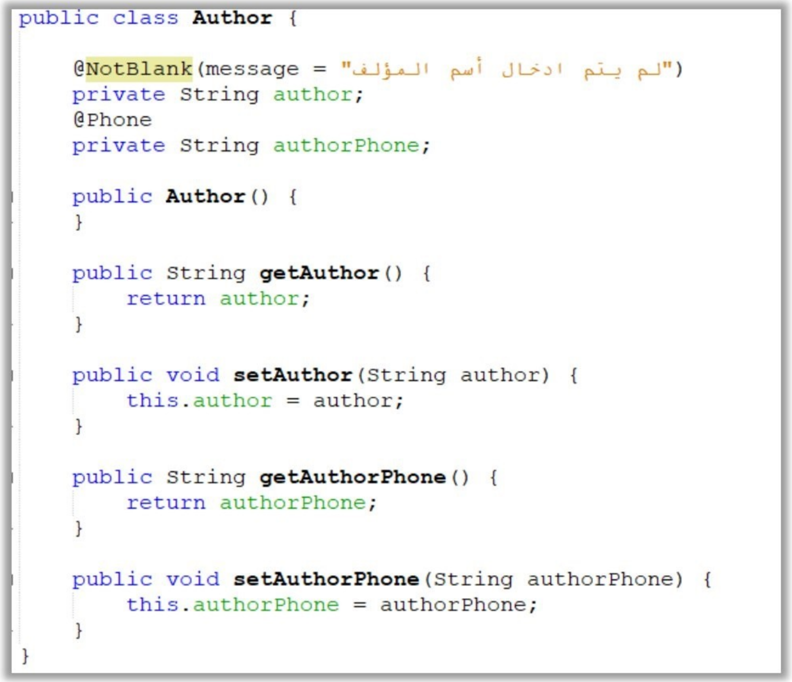
Example



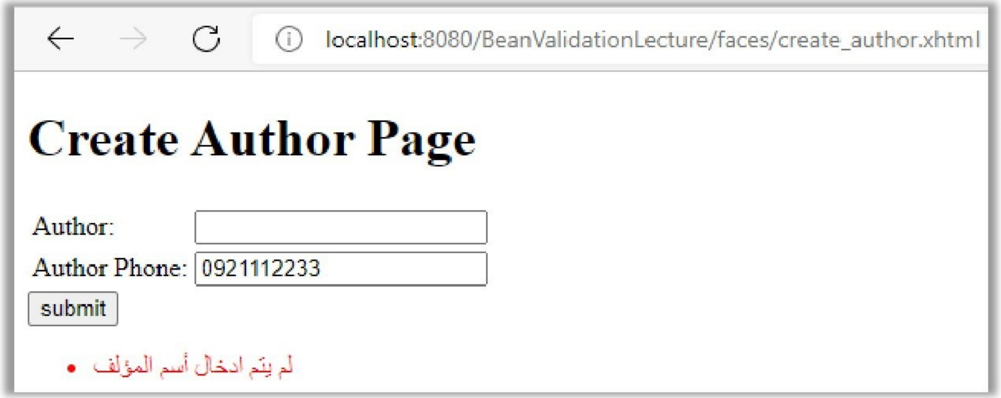
```
public class Author {


    @NotBlank
    private String author;
//    @NotBlank
//    @PhoneNumber
    @Phone
    private String authorPhone;


    public Author() {
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getAuthorPhone() {
        return authorPhone;
    }

    public void setAuthorPhone(String authorPhone) {
        this.authorPhone = authorPhone;
    }
}
```

Constraints Message

- يمكن تغيير الرسالة الافتراضية التي تظهر عند عدم نجاح عملية validation برسالة أخرى.



```
public class Author {

    @NotBlank(message = "لم يتم ادخال أسم المؤلف")
    private String author;
    @Phone
    private String authorPhone;

    public Author() {
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getAuthorPhone() {
        return authorPhone;
    }

    public void setAuthorPhone(String authorPhone) {
        this.authorPhone = authorPhone;
    }
}
```



localhost:8080/BeanValidationLecture/faces/create_author.xhtml

Create Author Page

Author:

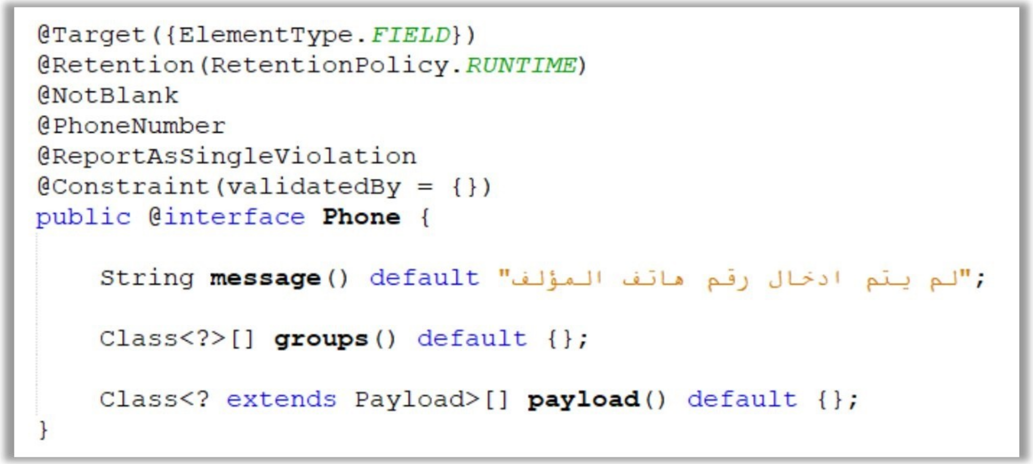
Author Phone: 0921112233

submit

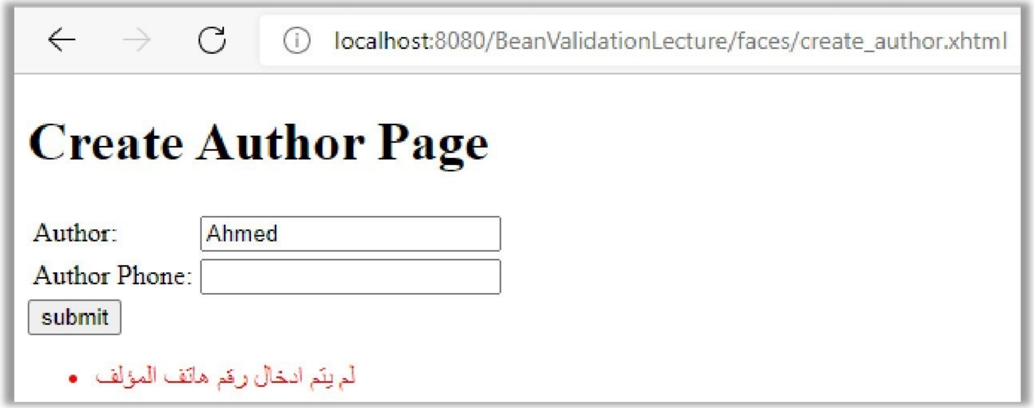
- لم يتم ادخال أسم المؤلف

@ReportAsSingleViolation

- عند استعمال Constraints Composition ونريد اظهار الرسالة الخاصة به تستخدم @ReportAsSingleViolation.



```
@Target({ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
@NotBlank
@PhoneNumber
@ReportAsSingleViolation
@Constraint(validatedBy = {})
public @interface Phone {

    String message() default "لم يتم ادخال رقم هاتف المؤلف";

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

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



localhost:8080/BeanValidationLecture/faces/create_author.xhtml

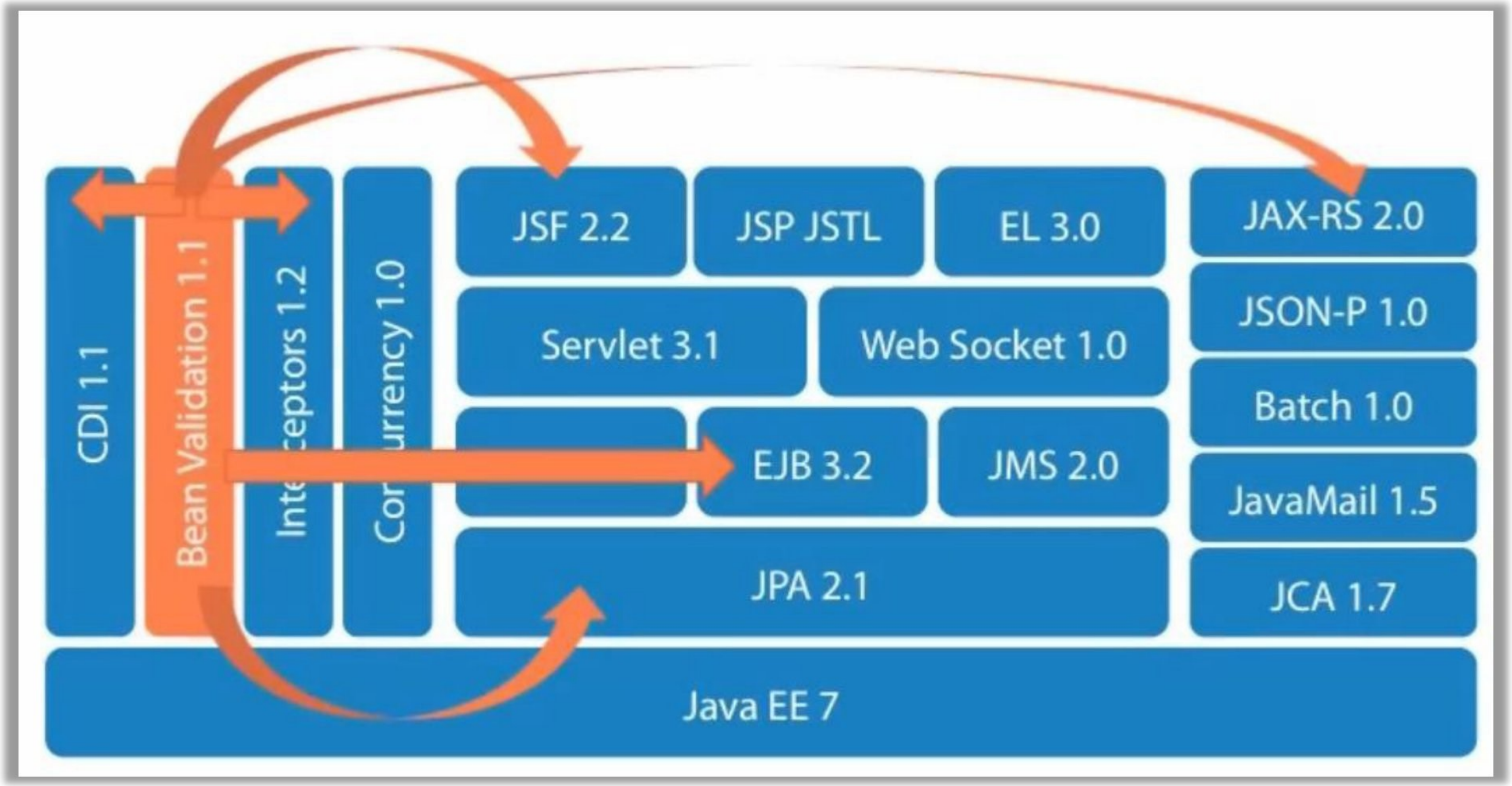
Create Author Page

Author: Ahmed
Author Phone:
submit

- لم يتم ادخال رقم هاتف المؤلف

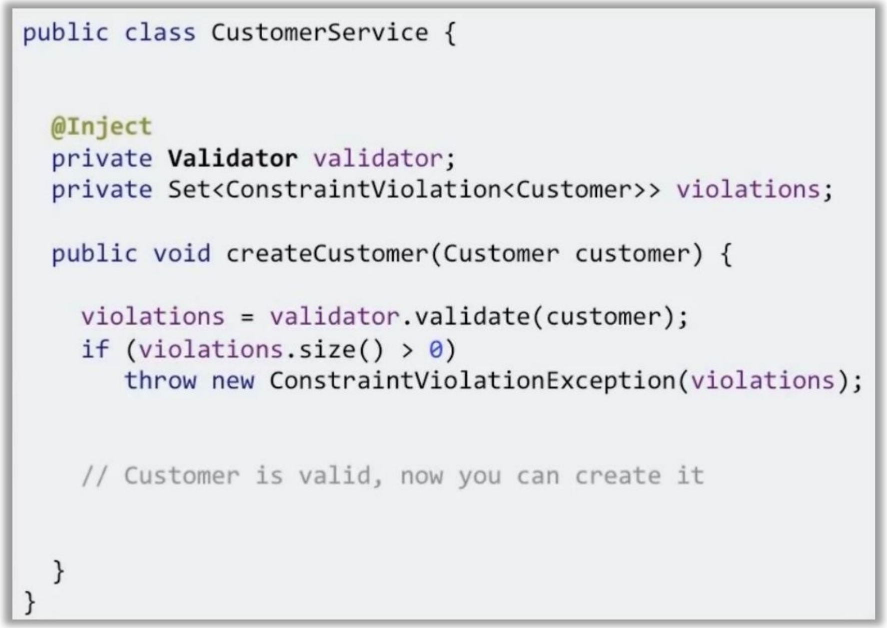
Bean Validation Interaction in Java EE

- الشكل التالي يبين استخدامات Bean Validation مع JavaEE.

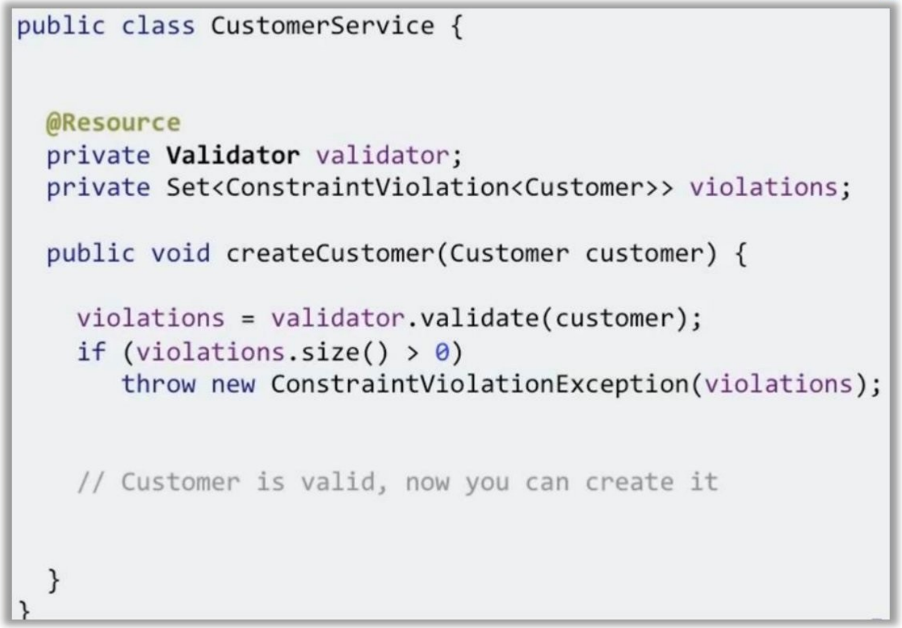


Interaction with Java EE and CDI

- يمكن التفاعل مع Bean Validation في JavaEE بأكثر من طريقة أهمها:
- @Resource
- @Inject



```
public class CustomerService {


  @Inject
  private Validator validator;
  private Set<ConstraintViolation<Customer>> violations;

  public void createCustomer(Customer customer) {

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


    // Customer is valid, now you can create it


  }
}
```

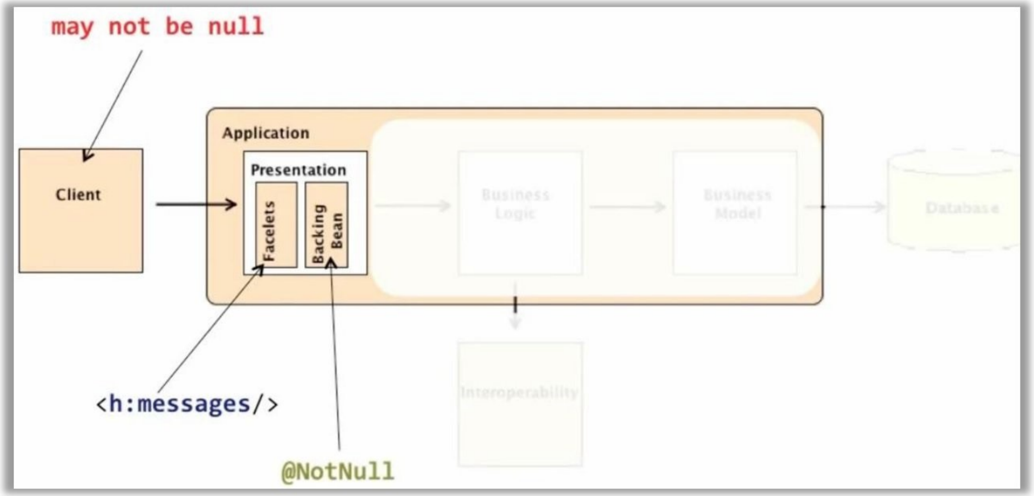


```
public class CustomerService {


  @Resource
  private Validator validator;
  private Set<ConstraintViolation<Customer>> violations;

  public void createCustomer(Customer customer) {

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


    // Customer is valid, now you can create it


  }
}
```

Integration with JSF

- يمكن الاستفادة من Bean Validation من خلال استخدام constrained في Backing Bean .
- تستخدم كل من <h:messages> و <h:message> لعرض رسائل الخطا الناتجة من validation.
- يمكن إيقاف عمل Bean Validation في web.xml من خلال أستخدام context parameter التالي:

```
javax.faces.validator.DISABLE_DEFAULT_BEAN_VALIDATOR
```



Example

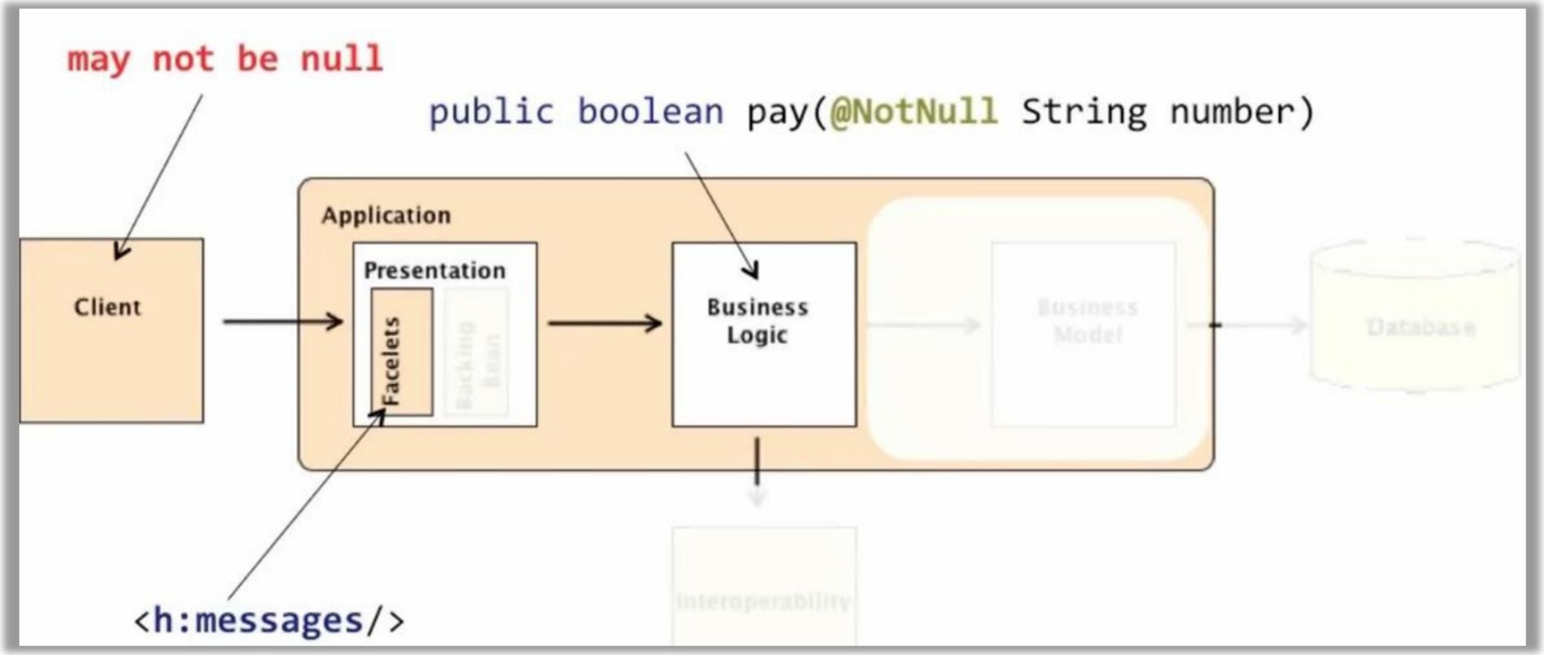
- المثال التالي يبين أستخدام Bean Validation مع JSF .

```
<h:messages/>

<label>* First Name</label>
<h:inputText value="#{customerBean.firstName}"/>
```

```
@Named
@ViewScoped
public class CustomerBean {

  @Inject
  private CustomerService service;

  @NotEmpty
  private String firstName;
  @NotEmpty @Size(min = 4, max = 50)
  private String lastName;
  @NotEmpty @Email
  private String email;

  public String doCreateCustomer() {
    service.create(firstName, lastName, email);
    return null;
  }
}
```

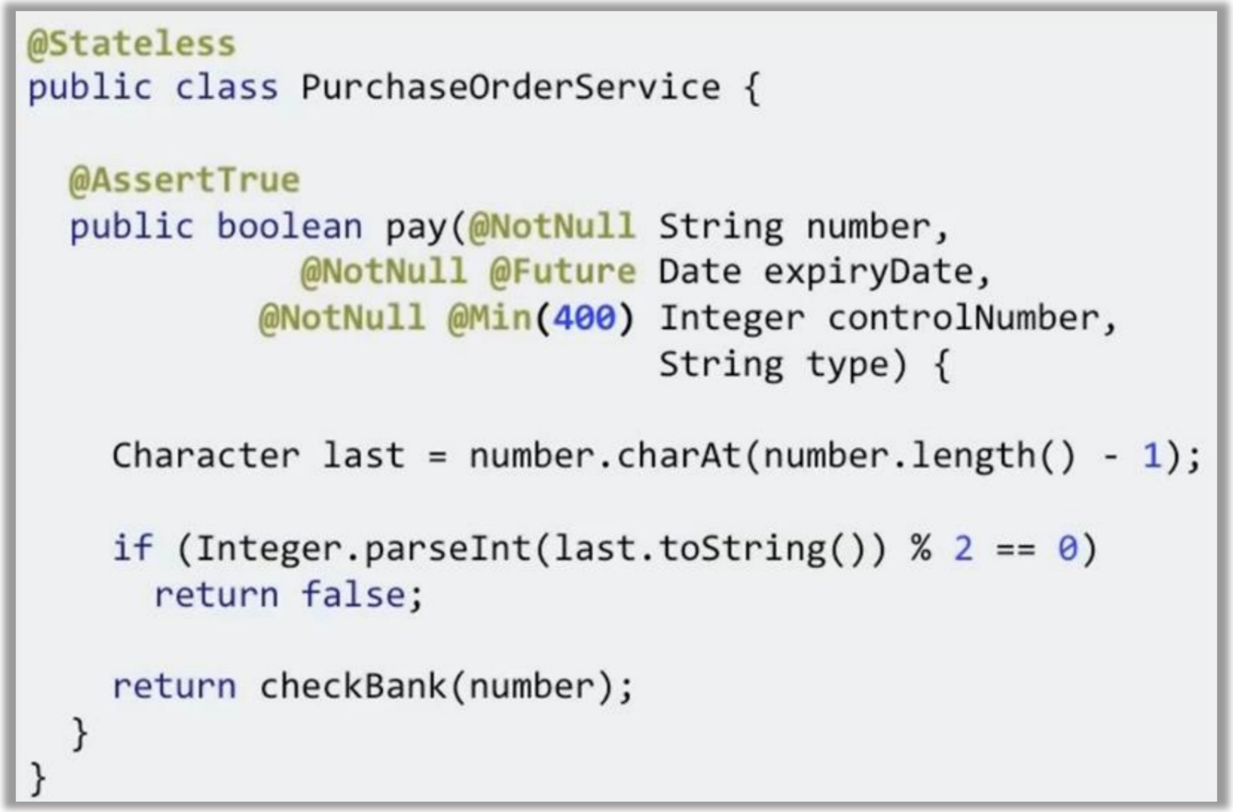
Integration with EJB

- يمكن استخدام Bean Validation مع EJB على مستوى Method .
- عملية validation تتم قبل وبعد استدعاء method .
- يمكن إيقاف validation من خلال استخدام @ValidateOnExecution



Example

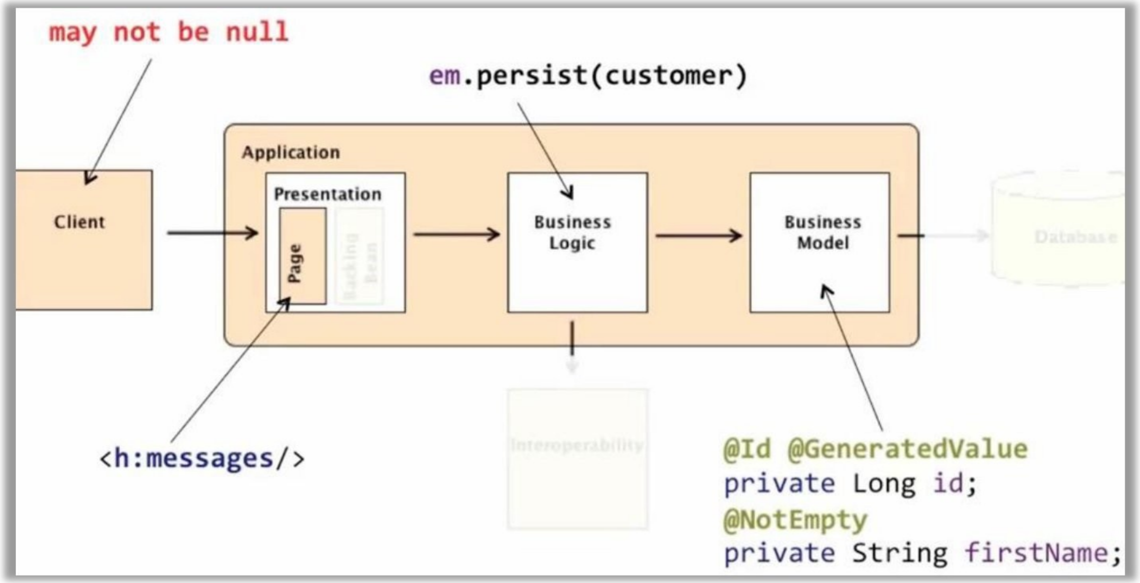
- المثال التالي يبين استخدام Bean Validation مع EJB .



```java
@Stateless
public class PurchaseOrderService {

  @AssertTrue
  public boolean pay(@NotNull String number,
                @NotNull @Future Date expiryDate,
              @NotNull @Min(400) Integer controlNumber,
                                 String type) {

    Character last = number.charAt(number.length() - 1);

    if (Integer.parseInt(last.toString()) % 2 == 0)
      return false;

    return checkBank(number);
  }
}
```

Integeration with JPA

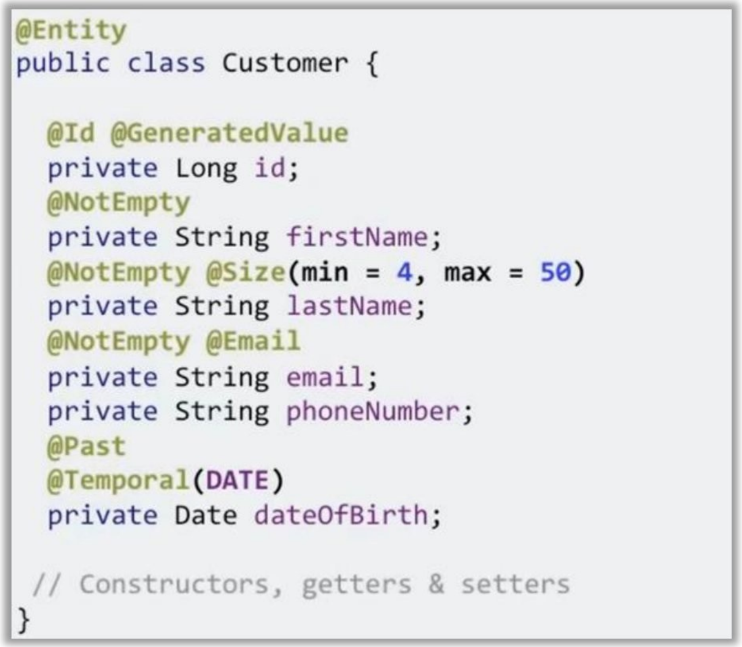
- يستخدم Bean Validation مع JPA من Entities وذلك بأستخدام Validation annotations .
- يمكن إيقاف validation من خلال persistence.xml .



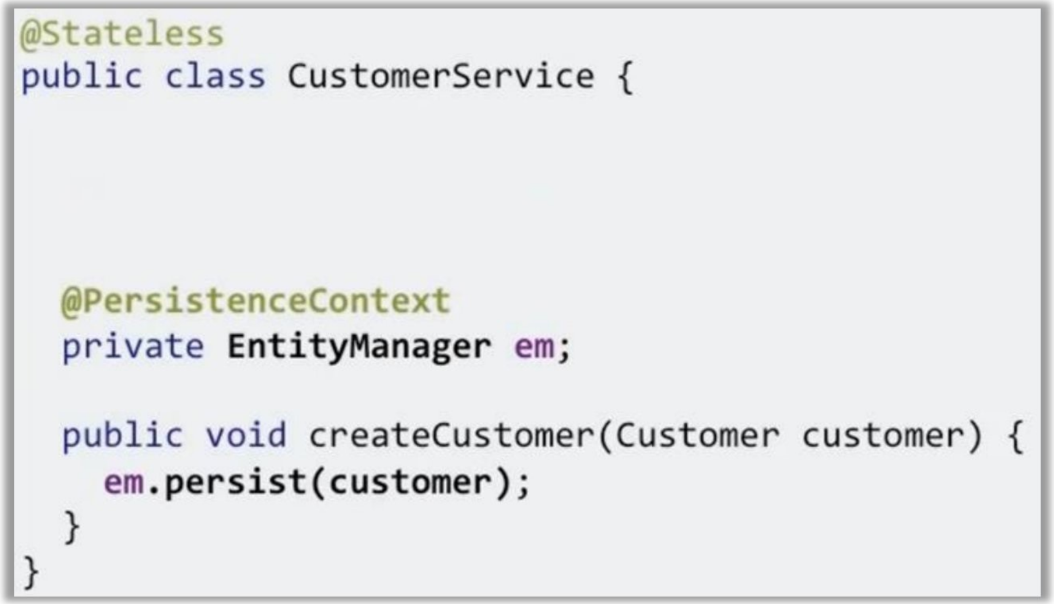
```
@Id @GeneratedValue
private Long id;
@NotEmpty
private String firstName;
```

Example

- المثال التالي يبين استخدام Bean Validation مع JPA.



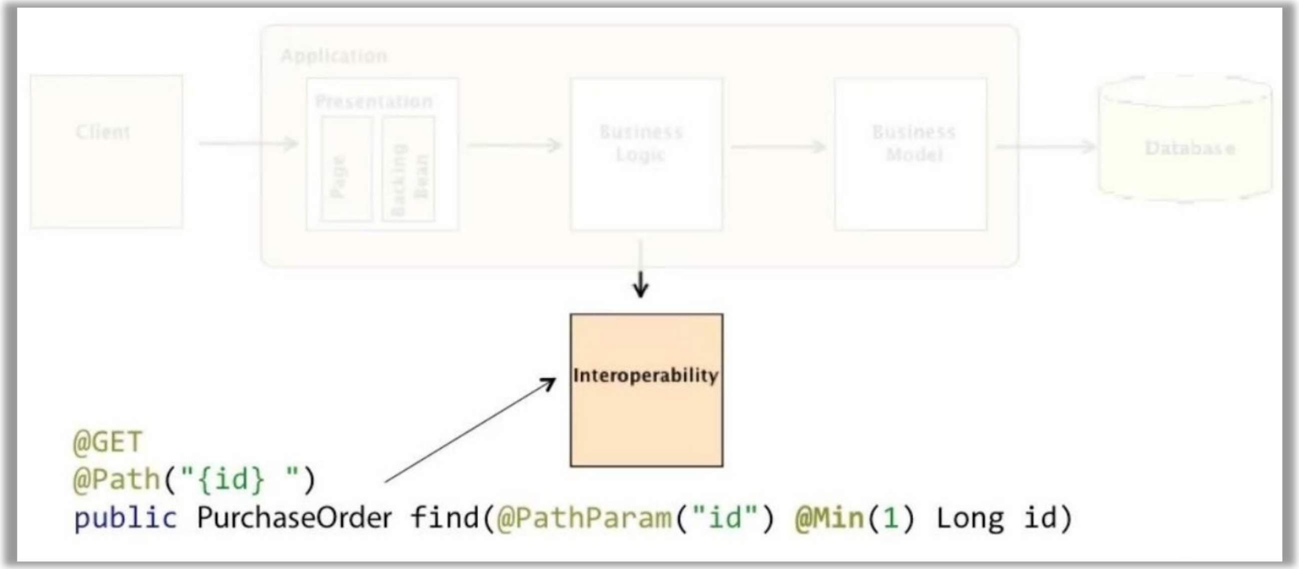
```java
@Entity
public class Customer {

  @Id @GeneratedValue
  private Long id;
  @NotEmpty
  private String firstName;
  @NotEmpty @Size(min = 4, max = 50)
  private String lastName;
  @NotEmpty @Email
  private String email;
  private String phoneNumber;
  @Past
  @Temporal(DATE)
  private Date dateOfBirth;

 // Constructors, getters & setters
}
```



```java
@Stateless
public class CustomerService {




  @PersistenceContext
  private EntityManager em;

  public void createCustomer(Customer customer) {
    em.persist(customer);
  }
}
```

Integeration with Jax-RS

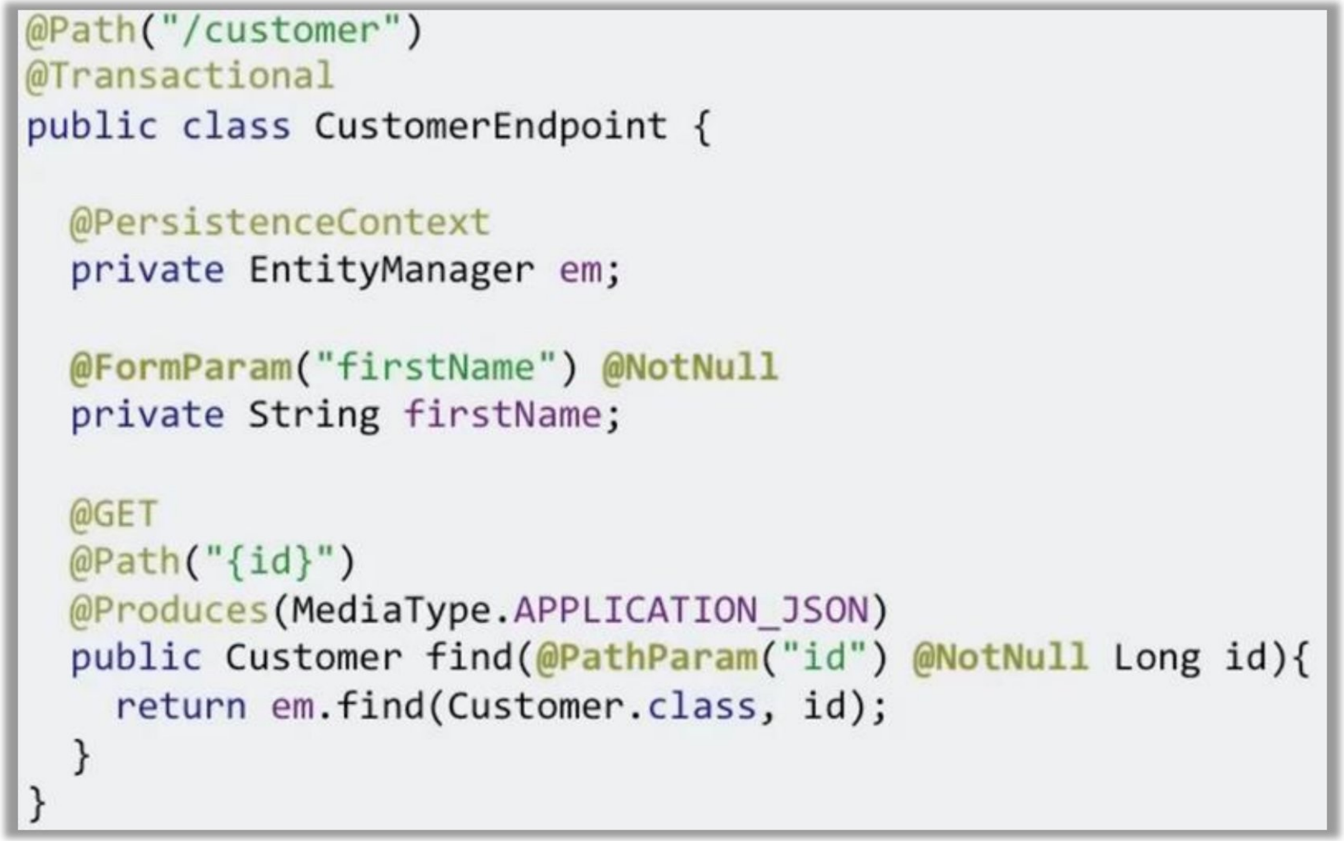
- يمكن استخدام Bean Validation مع Jax-Rs من خلال :
- Constraints on HTTP Parameters
- Method-level constraints
- يمكن إيقاف validation من خلال استخدام



```
@GET
@Path("{id} ")
public PurchaseOrder find(@PathParam("id") @Min(1) Long id)
```

Example

- المثال التالي يبين استخدام Bean Validation مع Jax-RS



```java
@Path("/customer")
@Transactional
public class CustomerEndpoint {

  @PersistenceContext
  private EntityManager em;

  @FormParam("firstName") @NotNull
  private String firstName;

  @GET
  @Path("{id}")
  @Produces(MediaType.APPLICATION_JSON)
  public Customer find(@PathParam("id") @NotNull Long id){
    return em.find(Customer.class, id);
  }
}
```