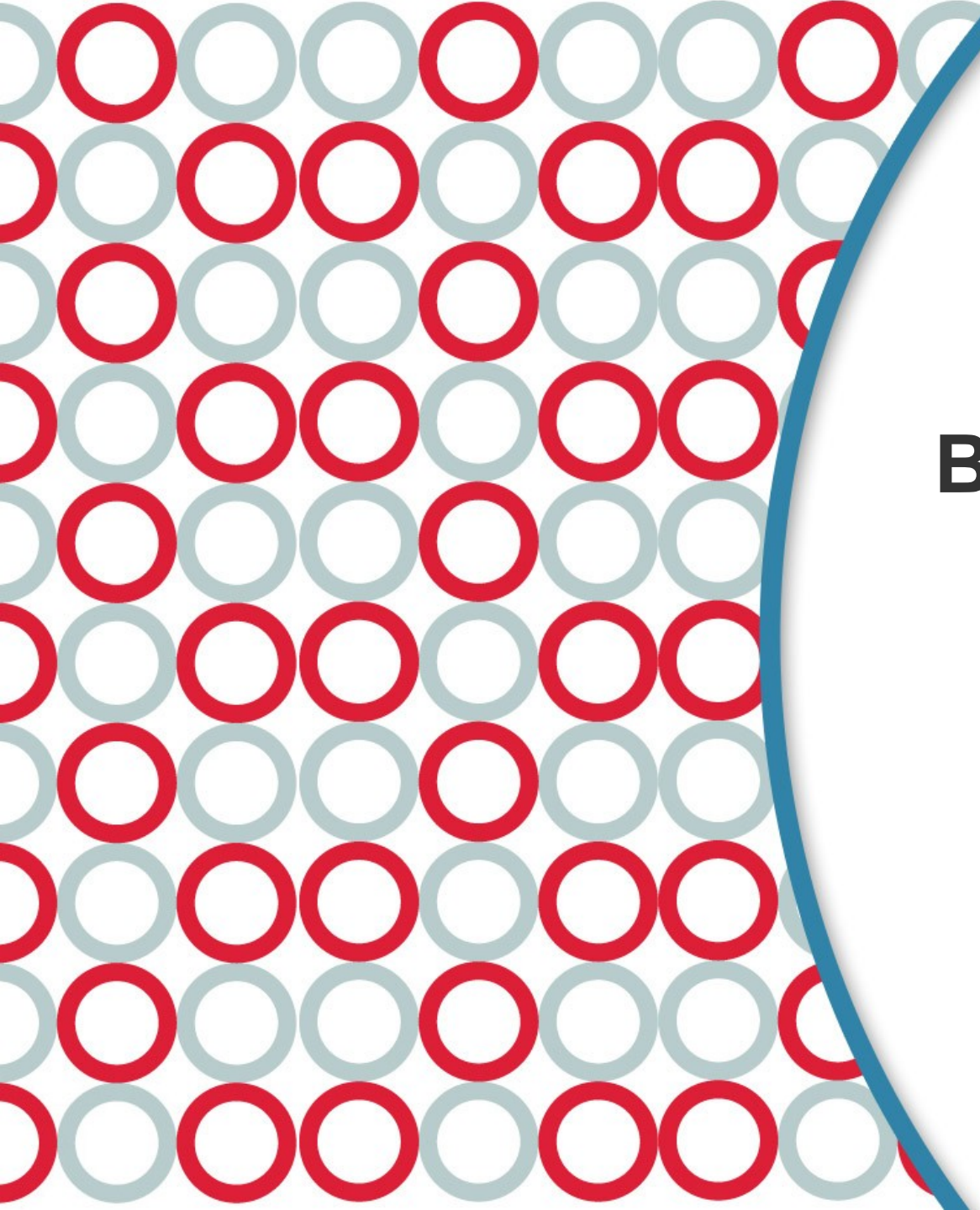

April 2018, IPT Course
Introduction to Spring 5

Bean Validation

Trayan Iliev

tiliev@iproduct.org
http://iproduct.org

Agenda for This Session

- Introduction to declarative bean validation using annotations
- Novelties in Bean Validation 2.0
- Main validation annotations (constraints)
- Example for implementing custom annotation
- Composite annotation example

Where to Find the Demo Code?

Introduction to Spring 5 demos and examples are available @ GitHub:

https://github.com/iproduct/course-spring5

JSR-303: Bean Validation (1)

- Bean Validation project started in July 2006 as JSR 303
- The first version was finalized on November 16, 2009
- The data validation is a cross-cutting concern implemented across all application layers – from presentation to domain services and database persistence
- Often similar logic is implemented multiple times programmatically in different application layers, which leads to duplication of efforts and code, as well as difficult testing, and decreased code-base evolvability
- To solve the problem developers often code the validation logic directly in the domain model, which leads to mixing of business logic and data needed for properties validation

JSR-303: Bean Validation (2)

- JSR 303: Bean Validation provides a comprehensive set of declarative constraints on objects' data, in the form of standardized annotations, which can be applied to fields, methods, method arguments, and classes of JavaBean components such as Spring Beans, JPA Entities, or JSF Managed Beans.
- There are a lot of predefined annotations, as well as possibility to create own validation annotations and connect them with java classes implementing the cross-cutting validation logic
- These predefined annotations are living in the package javax.validation.constraints

Bean Validation 2.0 (JSR 380)

Bean Validation 2.0 (JSR 380) from 2017 adds new features:

- Support for validating container elements by annotating type arguments of parameterized types e.g. List<@Positive Integer> positiveNumbers
- Support for java.util.Optional and custom container types
- Support for the JSR 310: A New Java Date/Time API datatypes with @Past and @Future
- New built-in constraints: @Email, @NotEmpty, @NotBlank, @Positive, @PositiveOrZero, @Negative, @NegativeOrZero, @PastOrPresent and @FutureOrPresent
- Leverage the JDK 8 new features (constraints are marked repeatable, parameter names are retrieved via reflection)

Bean Validation Builtin Annotations (1):

- @AssertFalse – a boolean element should be false
- @AssertTrue – a boolean element should be true
- @Min, @DecimalMin – minimal value of number type element
- @Max, @DecimalMax – maximal value of number type element
- @Digits – attributes fraction and integer for fraction and whole part of a number type element
- @Future, @FutureOrPresent – Date и Calendar validation
- @Past, @PastOrPresent – Date и Calendar validation
- @Size – min and max size of String (length), Collection, Map or Array (number of elements)

Bean Validation Builtin Annotations (2):

- @Null, @NotNull – the element should or should not be null
- @NotBlank – String not null and at least one character
- @NotEmpty – for CharSequence, Collection, Map, Array
- @Email – email validation
- @Pattern – the element should match to the supplied in regexp argument regular expression
- @Valid – this annotation from the package javax.validation, enables recursive validation of all properties of the annotated object using their specified constraints
- It is possible to create composite annotations and custom annotations using @Constraint, @GroupSequence, @ReportAsSingleViolation, @OverridesAttribute

Sample Email Constraint Implementation

```
public class Email {
  @NotEmpty @Pattern(".+@.+\\.[a-z]+")
  private String from;
  @NotEmpty @Pattern(".+@.+\\.[a-z]+")
  private String to;
  @NotEmpty
  private String subject;
  @Min(1) @Max(10)
  private Integer priority;
  @NotEmpty
  private String body;
  ... }
```


Source: Javadoc Documentation /api/javax/persistence/CollectionTable.html

Bean Validation Custom Annotation

```
@Size(min=4, max=4)
@ConstraintValidator(validatedBy = PostCodeValidator.class)
@Documented
@Target({ANNOTATION_TYPE, METHOD, FIELD})
@Retention(RUNTIME)
public @interface PostCode {
    public abstract String message() default
                "{package.name.PostCode.message}";
  public abstract Class<?>[] groups() default {};
  public abstract Class<? extends ConstraintPayload>[]
                        payload() default {};
}
```


Source: Javadoc Documentation /api/javax/persistence/CollectionTable.html

Class PostCodeValidator Implementation

```
public class PostCodeValidator implements
                                ConstraintValidator<PostCode, String>
{
   private final static Pattern POSTCODE_PATTERN =
                                Pattern.compile("\\d{4}");
   public void initialize(PostCode constraintAnnotation) { }
   public boolean isValid(String value,
                                ConstraintValidatorContext context) {
      return POSTCODE_PATTERN.matcher(value).matches();
   }
}
```


Source: Javadoc Documentation /api/javax/persistence/CollectionTable.html

Bean Validation – Composite Annotation

```
@ConstraintValidator(validatedBy = {})   @Documented
@Target({ANNOTATION_TYPE, METHOD, FIELD})
@Retention(RUNTIME)
@Pattern(regexp = "\\d{4}")
@ReportAsSingleViolation
public @interface PostCode {
    public abstract String message() default
            "{package.name.PostCode.message}";
  public abstract Class<?>[] groups() default {};
  public abstract Class<? extends ConstraintPayload>[]
                        payload() default {};
}
```


Source: Javadoc Documentation /api/javax/persistence/CollectionTable.html

References

- JSR 303: Bean Validation JavaEE Specification – http://jcp.org/en/jsr/detail?id=303
- JSR 380 Bean Validation 2.0 Specification – https://beanvalidation.org/2.0/spec/

Thank's for Your Attention!

Trayan Iliev

CEO of IPT – Intellectual Products & Technologies

http://iproduct.org/

http://robolearn.org/

https://github.com/iproduct

https://twitter.com/trayaniliev

https://www.facebook.com/IPT.EACAD

https://plus.google.com/+IproductOrg