Spring Boot Microservices

Beginner to Guru

Java Bean Validation

JSR 303 - Java Bean Validation

- JSR 303 Introduced Java Bean Validation (Version 1.0)
  - Set of annotations used to validate Java Bean properties
- Approved on November 16th, 2009.
- Part of JEE v6 and above
- JSR 303 Supported by Spring since version 3
- Primary focus was to define annotations for data validation
  - Largely field level properties

JSR 349 Bean Validation 1.1

- JSR 349 - Java Bean Validation 1.1 released on April 10th, 2013.
  - JEE v7, Spring Framework 4
- Builds upon 1.0 specification
- Expanded to method level validation
  - To validate input parameters
- Includes dependency injection for bean validation components

JSR 380 - Bean Validation 2.0

- Approved August 2017
- Added to Spring Framework 5.0 RC2
- Available in Spring Boot 2.0.0 +
- Uses Hibernate Validator 6.0 + (Implementation of Bean Validation 2.0)
- Primary goal of Bean Validation 2.0 is Java 8 language features
- Added ~11 new built in validation annotations
- Remainder of presentation will focus on Bean Validation 2.0

Built In Constraint Definitions

- @Null - Checks value is null
- @NotNull - Checks value is not null
- @AssertTrue - Value is true
- @AssertFalse - Value is false
- @Min - Number is equal or higher
- @Max - Number is equal or less

Built In Constraint Definitions

- @DecimalMin - Value is larger
- @DecimalMax - Value is less than
- @Negative - Value is less than zero. Zero invalid.
- @NegativeOrZero - Value is zero or less than zero
- @Positive - Value is greater than zero. Zero invalid.
- @PositiveOrZero - Value is zero or greater than zero.
- @Size - checks if string or collection is between a min and max

Built In Constraint Definitions

- @Digits - check for integer digits and fraction digits
- @Past - Checks if date is in past
- @PastOrPresent - Checks if date is in past or present
- @Future - Checks if date is in future
- @FutureOrPresent - Checks if date is present or in future
- @Pattern - checks against RegEx pattern

Built In Constraint Definitions

- @NotEmpty - checks if value is not null nor empty (whitespace characters or empty collection)
- @NonBlank - Checks string is not null or not whitespace characters
- @Email - Checks if string value is an email address

Hiberate Validator Constraints

- @ScriptAssert - Class level annotation, checks class against script
- @CreditCardNumber - Verifies value is a credit card number
- @Currency - Valid currency amount
- @DurationMax - Duration less than given value
- @DurationMin - Duration greater than given value
- @EAN - Valid EAN barcode
- @ISBN - Valid ISBN value

Hiberate Validator Constraints

- @Length - String length between given min and max
- @CodePointLength - Validates that code point length of the annotated character sequence is between min and max included.
- @LuhnCheck - Luhn check sum
- @Mod10Check - Mod 10 check sum
- @Mod11Check - Mod 11 check sum

Hiberate Validator Constraints

- @Range - checks if number is between given min and max (inclusive)
- @SafeHtml - Checks for safe HTML
- @UniqueElements - Checks if collection has unique elements
- @Url - checks for valid URL

Hiberate Validator Country Constraints

- @CNPJ - Brazilian Corporate Tax Payer Registry Number
- @CPF - Brazilian Individual Taxpayer Registry Number
- @TituloEleitoral - Brazilian voter ID
- @NIP - Polish VAR ID
- @PESEL - Polish National Validation Number
- @REGON - Polish Taxpayer ID

Bean Validation Usage

- Hibernate Validator 6 + is only implementation of Bean Validation 2.0
  - At time of recording
- Consult Hibernate documentation for complete details
- Hibernate is likely to add additional predefined validators
- Official documentation covers how to define custom validators

SPRING FRAMEWORK

GURU