Spring bean validation allows us to use this bean validation constraint directly on the Java beans. There are several benefits of this approach:

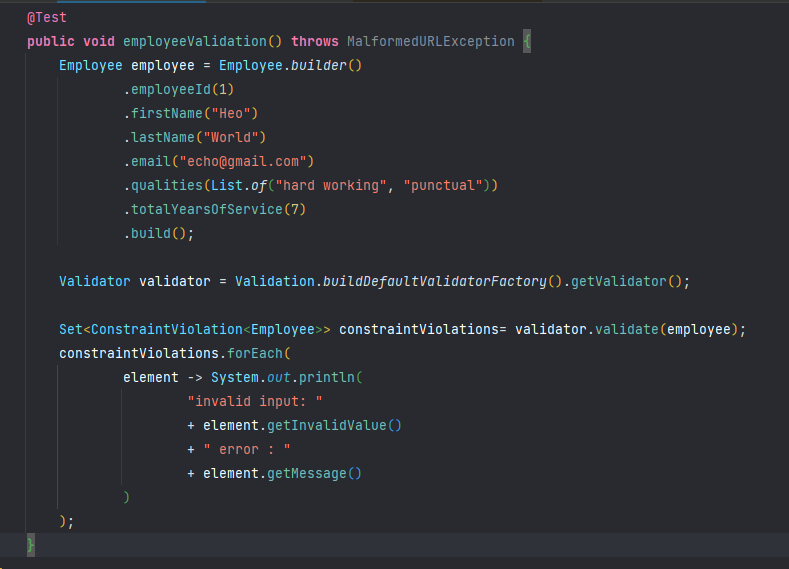
* No need for separate validation configurations.
* Constraints are straight on the domain models

If you are working on a **Spring MVC application** ora **REST API**, there are three distinct places where we can validate:

1. Request body
2. Path variables.
3. Query parameters.

Full list of constraints: <https://docs.jboss.org/hibernate/stable/validator/reference/en-US/html_single/#section-builtin-constraints>

Checking the validation (Have To check how to get the attribute name)



**Default Annotations**

**@Digits(integer=, fraction=)** - Checks whether the annotated value is a number having up to integer digits and fraction fractional digits.

**@Email** - Checks whether the specified character sequence is a valid email address.

**@Max(value=)** - Checks whether the annotated value is less than or equal to the specified maximum.

**@Min(value=)** - Checks whether the annotated value is higher than or equal to the specified minimum

**@NotBlank** - Checks that the annotated character sequence is not null and the trimmed length is greater than 0.

**@NotEmpty** - Checks whether the annotated element is not null nor empty.

**@Null** - Checks that the annotated value is null

**@NotNull** - Checks that the annotated value is not null

**@Pattern(regex=, flags=)** - Checks if the annotated string matches the regular expression regex considering the given flag match

**@Size(min=, max=)** - Checks if the annotated element’s size is between min and max (inclusive)

**@Negative** - Checks if the element is strictly negative. Zero values are considered invalid.

**@Positive** Checks if the element is strictly positive. Zero values are considered invalid.

**@NegativeOrZero** - Checks if the element is negative or zero.

**@PositiveOrZero** Checks if the element is positive or zero.

**@Future** - Checks whether the annotated date is in the future.

**@FutureOrPresent** - Checks whether the annotated date is in the present or in the future.

**@Past** Checks whether the annotated date is in the past

**@PastOrPresent** - Checks whether the annotated date is in the past or in the present.

**@DecimalMax(value=, inclusive=)** Checks whether the annotated value is less than the specified maximum, when inclusive=false. Otherwise, whether the value is less than or equal to the specified maximum

**@DecimalMin(value=, inclusive=)** Checks whether the annotated value is larger than the specified minimum, when inclusive=false. Otherwise, whether the value is larger than or equal to the specified minimum

**Hibernate validator specific annotations**

In addition to the constraints defined by the Bean Validation API, Hibernate Validator provides several useful custom constraints which are listed below.

**@CreditCardNumber(ignoreNonDigitCharacters=)** - Checks that the annotated character sequence passes the Luhn checksum test. Note, this validation aims to check for user mistakes, not credit card validity!

**@Currency(value=)** - Checks that the currency unit of the annotated javax.money.MonetaryAmount is part of the specified currency units.

**@EAN** - Checks that the annotated character sequence is a valid EAN barcode. The default is EAN-13.

**@ISBN** - Checks that the annotated character sequence is a valid ISBN.

**@Length(min=, max=)** - Validates that the annotated character sequence is between min and max included.

**@Range(min=, max=)** - Checks whether the annotated value lies between (inclusive) the specified minimum and maximum.

**@UniqueElements** - Checks that the annotated collection only contains unique elements.

**@URL** - Checks if the annotated character sequence is a valid URL according to RFC2396.

**@DurationMax(days=, hours=, minutes=, seconds=, millis=, nanos=, inclusive=)** Checks that annotated java.time.Duration element is not greater than the one constructed from annotation parameters. Equality is allowed if inclusive flag is set to true.

**@DurationMin(days=, hours=, minutes=, seconds=, millis=, nanos=, inclusive=)** Checks that annotated java.time.Duration element is not less than the one constructed from annotation parameters. Equality is allowed if inclusive flag is set to true.

There are few more validations are there, which I didn’t list here.

**Country Specific constraints:**

**@CNPJ** Checks that the annotated character sequence represents a Brazilian corporate tax payer registry number (Cadastro de Pessoa Jurídica)

**@CPF** Checks that the annotated character sequence represents a Brazilian individual taxpayer registry number (Cadastro de Pessoa Física)

**@TituloEleitoral** Checks that the annotated character sequence represents a Brazilian voter ID card number (Título Eleitoral)

**@NIP** Checks that the annotated character sequence represents a Polish VAT identification number (NIP)

**@PESEL** Checks that the annotated character sequence represents a Polish national identification number (PESEL)

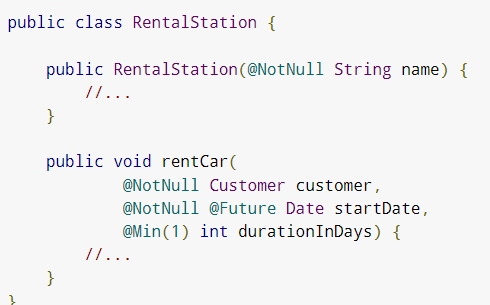
**@REGON** Checks that the annotated character sequence represents a Polish taxpayer identification number (REGON). Can be applied to both 9- and 14-digits versions of REGON

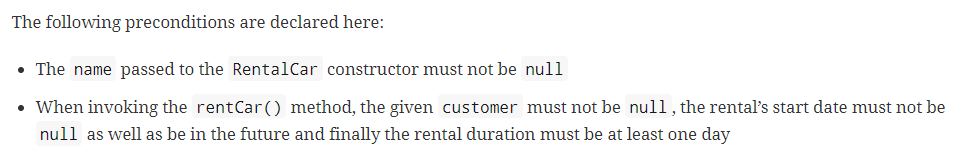
**@INN** Checks that the annotated character sequence represents a Russian taxpayer identification number (INN). Can be applied to both individual and juridical versions of INN

**Declaring and validating method constraints**

<https://docs.jboss.org/hibernate/stable/validator/reference/en-US/html_single/#chapter-method-constraints>

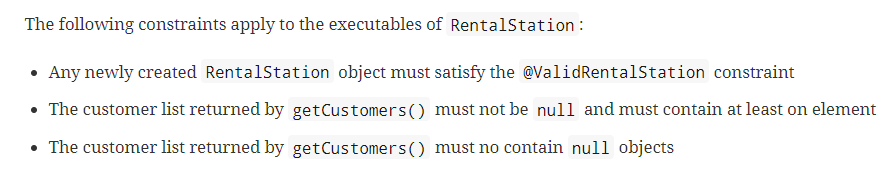
1. **Parameter Constraints:**

****

****

1. **Return value constraints:**

****

****

1. **Cascade validation:**

**………………………** rest is TODO