# Java Bean Validations

- ***@NotNull*** validates that the annotated property value is not *null*.
- ***@AssertTrue*** validates that the annotated property value is *true*.
- ***@Size*** validates that the annotated property value has a size between the attributes *min* and *max*; can be applied to *String*, *Collection*, *Map*, and array properties.
- ***@Min*** validates that the annotated property has a value no smaller than the *value* attribute.
- ***@Max*** validates that the annotated property has a value no larger than the *value* attribute.
- ***@Email*** validates that the annotated property is a valid email address.

Some annotations accept additional attributes, but the *message* attribute is common to all of them. This is the message that will usually be rendered when the value of the respective property fails validation.

And some additional annotations that can be found in the JSR:

- ***@NotEmpty*** validates that the property is not null or empty; can be applied to *String*, *Collection*, *Map* or *Array* values.
- ***@NotBlank*** can be applied only to text values and validates that the property is not null or whitespace.
- ***@Positive*** and ***@PositiveOrZero*** apply to numeric values and validate that they are strictly positive, or positive including 0.
- ***@Negative*** and ***@NegativeOrZero*** apply to numeric values and validate that they are strictly negative, or negative including 0.
- ***@Past*** and ***@PastOrPresent*** validate that a date value is in the past or the past including the present; can be applied to date types including those added in Java 8.
- ***@Future*** *and* ***@FutureOrPresent*** validate that a date value is in the future, or in the future including the present.

**The validation annotations can also be applied to elements of a collection**:

```
List<@NotBlank String> preferences;
```

In this case, any value added to the preferences list will be validated.

# DecimalMin/DecimalMax

- **Can be used for double/Double as well as BigDecimal**
- **`value` param will be a number expressed as String (in quotes)**
- **`inclusive` can be used to indicate if check should include the specified `value`**

```
@DecimalMin(value = "0.1", inclusive = false)
protected Double minBalance;

@DecimalMax(value = "1000000", inclusive = false)
protected Double maxBalance;
```

# Examples of Annotations with Parameters

```java
@NotNull(message = "sampleString must not be null")
@Size(min = 5, max = 999, message = "sampleString must have a length between 5 and 99") // can be used on collections as well
private String sampleString;

@Min(value = 10, message = "minInt must be at least 10")
private int minInt;

@Max(value = 999, message = "maxInt must be no more than 999")
private int maxInt;

@DecimalMin(value = "1.5", message = "minDouble must be at least 1.5") // cam be used on BigDecimal as well
private double minDouble;

@DecimalMax(value = "999.99", message = "mixDouble must be no more than 999.99") // cam be used on BigDecimal as well
private double mixDouble;

@AssertTrue(message = "must be running")
private boolean isRunning;

@AssertFalse(message = "must not still be running")
private boolean stillRunning;
```