Nolix

Nolix validator

2025-07-26

Nolix

Table of contents

1 Introduction

1.1 What the Nolix validator is

The Nolix validator is a tool that can validate **arguments**.

1.2 Why to use the Nolix validator

- The Nolix validator can validate **many** different properties of given arguments.
- The Nolix validator produces **consistent** error messages.
- The calls of the Nolix validator produces very **legible** code.

1.3 Where the Nolix validator is

The Nolix validator is defined in the Nolix library. To use the Nolix validator, import the Nolix library into your project.

2 Basics

2.1 Import the Validator

```
import ch.nolix.core.errorcontrol.validator.Validator;
```

The Validator is in the ch.nolix.core.errorcontrol.validator package.

2.2 Validate that an object is not null

```
public void setContent(Object content) {
  Validator.assertThat(content).isNotNull();
  …
}
```

If the given content is null, the Validator will throw an ArgumentIsNullException. The error message of the ArgumentIsNullException will be:

“The given argument is null.”

2.3 Validate that an object is of a given type

```
public void setContent(Object content) {
  Validator.assertThat(content).isOfType(String.class);
}
```

If the given content is null, the Validator will throw an ArgumentIsNullException. The error message of the ArgumentIsNullException will be:

“The given argument is null.”

If the given content is not a String, the Validator will throw an InvalidArgumentException. The message of the InvalidArgumentException will be:

”The given argument is not a String.”

2.4 Include the argument's name into the error message

```
public void setContent(Object content) {
  Validator.assertThat(content).thatIsNamed("content").isNotNull();
}
```

If the given content is null, the Validator will throw an ArgumentIsNullException. The error message of the ArgumentIsNullException will be:

"The given content is null."

2.5 Use constants for common argument names

```
import ch.nolix.coreapi.programatoma.variable.LowerCaseVariableCatalog;

public void setContent(Object content) {
  Validator
  .assertThat(content)
  .thatIsNamed(LowerCaseVariableCatalog.CONTENT)
  .isNotNull();
  …
}
```

If the given content is null, the Validator will throw an ArgumentIsNullException. The error message of the ArgumentIsNullException will be:

"The given content is null."

The LowerCaseVariableCatalog provides many constants of common argument names. The constants of the LowerCaseVariableCatalog are lower case Strings. The LowerCaseVariableCatalog is in the ch.nolix.coreapi.programatom.variable. package.

3 Number validations

3.1 Validate that a number is not negative

```
public void setAmount(int amount) {
  Validator.assertThat(amount).isNotNegative();
  …
}
```

If the given amount negative, the Validator will throw a NegativeArgumentException. If the given amount is -25, the error message of the NegativeArgumentException will be:

"The given Integer '-25' is negative."

3.2 Validate that a number is in a given range

```
public void buyPencils(int amount) {
  Validator.assertThat(amount).isBetween(100, 200);
  …
}
```

If the given amount is not in the given range, the Validator will throw an ArgumentIsOutOfRangeException. If the given amount is 50, the error message of the ArgumentIsOutOfRangeException will be:

"The given Integer '50' is not in [100, 200]."

4 String validations

4.1 Validate that a String is not blank

```
public void setName(String name) {
  Validator.assertThat(name).isNotBlank();
  …
}
```

If the given name is null, the Validator will throw an ArgumentIsNullException. The error message of the ArgumentIsNullException will be:

"The given argument is null."

If the given name is ' ', the Validator will throw an InvalidArgumentException. The error message of the InvalidArgumentException will be:

"The given String ' ' is blank."

5 Container validations

5.1 Validate that an array is not empty

```
public void saveValues(double[] values) {
  Validator.assertThat(values).isNotEmpty();
  …
}
```

If the given values is null, the Validator will throw an ArgumentIsNullException. The error message of the ArgumentIsNullException will be:

"The given argument is null."

If the given values is empty, the Validator will throw an EmptyArgumentException. The error message of the EmptyArgumentException will be:

"The given Array is empty."

5.2 Validate that the numbers in a container are not negative

```
public void saveMeasuredValues(double[] values) {
  Validator.assertThatTheDoubles(values).areNotNegative();
}
```

If the given values is null, the Validator will throw an ArgumentIsNullException. The error message of the ArgumentIsNullException will be:

"The given argument is null."

If one of the given measured values is negative, the Validator will throw a NegativeArgumentException. If the 5th measured value is -10, the error message of the NegativeArgumentException will be:

"The given 5th argument '-10' is negative."

5.3 Validate that the Strings in a container are not empty

```
public void addCities(String[] cityNames) {
  Validator.assertThatTheStrings(cityNames).areNotEmpty();
}
```

If the given cityNames is null, the Validator will throw an ArgumentIsNullException. The error message of the ArgumentIsNullException will be:

"The given argument is null."

If one of the given city names is empty, the Validator will throw an EmptyArgumentException. If the 5th city name is empty, the error message of the EmptyArgumentException will be:

"The given 5th argument is empty."

6 Method validations

6.1 Validate that a method has an annotation

```
setDataMethod(Method dataMethod) {
  Validator.assertThat(dataMethod).hasAnnotation(Data.class);
  …
}
```

If the given dataMethod is null, the Validator will throw an ArgumentIsNullException. The error message of the ArgumentIsNullException will be:

"The given argument is null."

If the given dataMethod does not have the Data annotation, the Validator will throw an InvalidArgumentException. The error message of the InvalidArgumentException will be:

"The given method does not have the annotation 'Data'."

6.2 Validate that a method does not return anything

```
setRunMethod(Method runMethod) {
  Validator.assertThat(runMethod).doesNotReturnAnything();
  …
}
```

If the given runMethod is null, the Validator will throw an ArgumentIsNullException. The error message of the ArgumentIsNullException will be:

"The given argument is null."

If the given runMethod is not void, the Validator will throw an InvalidArgumentException. The error message of the InvalidArgumentException will be:

"The given method is not void."