Übung Java Lambda Validator

Ausgangslage

Die folgenden Listings zeigen eine Validator Lösung mit enum Typ klassisch ohne Lambda Expressions:package ch.std.jup8.lambda.validator.classic;

public enum StringValidatorEnum {

 STRING_8 {
 @Override
 public void validate(String s) throws ValidatorException {
 if (s instanceof String) {
 System.out.println();
 }
 if (s.length() > 8) {
 throw new ValidatorException(this.name());
 }
 }
 },
 STRING_16 {
 @Override
 public void validate(String s) throws ValidatorException {
 if (s.length() > 16) {
 throw new ValidatorException(this.name());
 }
 }
 },
 STRING_32 {
 @Override
 public void validate(String s) throws ValidatorException {
 if (s.length() > 32) {
 throw new ValidatorException(this.name());
 }
 }
 };
 private StringValidatorEnum() {

 }
 public abstract void validate(String s) throws ValidatorException;
}package ch.std.jup8.lambda.validator.classic;

public class ValidatorException extends Exception {

 private static final long serialVersionUID = 34939945212864755l6L;

 public ValidatorException() {
 super();
 }

 public ValidatorException(String message) {
 super(message);
 }

}package ch.std.jup8.lambda.validator.classic;

public class ValidatorDemo {

 public static void main(String[] args) {
 String s = "1234567890";
 try {
 StringValidatorEnum.STRING_32.validate(s);
 StringValidatorEnum.STRING_16.validate(s);
 StringValidatorEnum.STRING_8.validate(s);
 } catch (ValidatorException e) {
 e.printStackTrace();
 }
 }
}

Vorlage

Die Dateien der Vorlage finden Sie auch hier

Aufgabe Supplier

In einem ersten Schritt möchten wir die Vorlage mit dem Supplier FunctionalInterface anpassen, so dass die Validierung über solches erfolgen kann. Das folgende Listing zeigt das angepasste Demo Programm:package ch.std.jup8.lambda.validator.lambda.supplier;

public class ValidatorDemo {

 public static void main(String[] args) {
 String s = "1234567890";
 try {
 StringValidator.STRING_32.validate(() -> s);
 StringValidator.STRING_16.validate(() -> s);
 StringValidator.STRING_8.validate(() -> s);
 } catch (ValidatorException e) {
 e.printStackTrace();
 }
 }
 },

Lösung Supplier FunctionInterface

Eine mögliche Lösung mit Supplier FunctionalInterface finden Sie hier

Aufgabe Validator FunctionalInterface

Jetzt wollen wir die Validierungsregeln über ein eigenen Validator FunctionalInterface erweitern. Das folgende Listing zeigt die Enum Klasse:package ch.std.jup8.lambda.validator.lambda;

import java.util.function.Supplier;

public enum StringValidatorEnum {

 STRING_8(s -> {
 if (s.get().toString().length() > 8) {
 throw new ValidatorException("min length is 8");
 }
 }),
 STRING_16(s -> {
 if (s.get().toString().length() > 16) {
 throw new ValidatorException("min length is 16");
 }
 }),
 STRING_32(s -> {
 if (s.get().toString().length() > 32) {
 throw new ValidatorException("min length is 32");
 }
 });

 private Validator<String> v;

 private StringValidatorEnum(Validator<String> v) {
 this.v = v;
 }

 public void validate(Supplier<String> s) throws ValidatorException {
 v.validate(s);
 }

}
Programmieren Sie das Validator FunctionalInterface, so dass der enum Typ kompiliert. Das Demo Programm sollte identisch

bleiben.Das folgende Listing zeigt den Einsatz eines allgemeinen AnyValidators: // check null
  try {
  AnyValidator.validate(supplier -> {
  if (supplier.get() == null) {
  throw new ValidatorException();
  }
  }, null);
  } catch (ValidatorException e) {
  e.printStackTrace();
  }
  // check is number
  try {
  AnyValidator.validate(supplier -> {
  if (supplier.get() == null) {
  throw new ValidatorException();
  }
  Integer.parseInt(supplier.get());
  }, "123");
  } catch (ValidatorException e) {
  e.printStackTrace();
  }Programmieren Sie den AnyValidator (generic Interface), so dass das Listing kompiliert als Teil des Demo Programms.

Lösung Validator FunctionalInterface

Eine mögliche Lösung mit Callable FunctionalInterface finden Sie hier

Kontakt

Simtech AG
Finkenweg 23
3110 Münsingen
Schweiz

Impressum

Das Copyright für sämtliche Inhalte dieser Website liegt bei Simtech AG, Schweiz.
Beachten Sie auch unsere Hinweise zum Urheberrecht, Datenschutz und Haftungsauschluss.
Jeder Hinweis auf Fehler nehmen wir gerne entgegen.

Copyright

2024 Simtech AG, All rights reserved, Powered by stack.ch written in Golang by Daniel Schmutz

https://www.simtech-ag.ch/Deutschland JPF2 Lambda Validator