МИНИСТЕРСТВО ТРАНСПОРТА РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»

Кафедра ЦТУТП

**Отчёт**

По лабораторной работе №4  
по дисциплине «Корпоративные информационные системы»

Тема: «Валидация с использованием стандартных аннотаций»

Вариант №14

Выполнил: Сафонов Г.

Группа: УИС-411

Преподаватель: доц. Кафедры ЦТУТП

Козьяков П. О.

­

Москва 2024 г.

**СОДЕРЖАНИЕ**

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc180330916)

[ХОД ВЫПОЛНЕНИЯ РАБОТЫ 4](#_Toc180330917)

[РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ 4](#_Toc180330918)

[КОД ПРОГРАММЫ 7](#_Toc180330919)

[ВЫВОД 10](#_Toc180330920)

# **ПОСТАНОВКА ЗАДАЧИ**

Описать класс НаселённыйПункт. С помощью стандартного API валидации с использованием стандартных аннотаций произвести валидацию класса НаселённыйПункт. Валидацию провести с использованием аннотаций при описании класса и с указанием ограничений в XML. Обязательно использовать аннотацию @Pattern с проверкой, определяемой регулярным выражением.

# **ХОД ВЫПОЛНЕНИЯ РАБОТЫ**

Для начала, необходимо убедиться, что в проекте правильно настроены зависимости для работы с Jakarta Bean Validation. Это включает добавление зависимостей для Jakarta Validation API и Hibernate Validator, которые будут использоваться для валидации данных в Java. Это подготовит проект для работы с аннотациями, которые будут применяться к полям классов моделей данных.

После этого можно переходить к созданию моделей данных. В данном случае, создаются два класса: CodeValidateSettlement и XmlValidatedSettlement. Класс CodeValidateSettlement будет использовать аннотации валидации, такие как @NotNull, @Pattern, @Min, и @Max, для валидации полей на уровне класса. Это обеспечит автоматическую проверку значений полей при их установке, например, на проверку того, что имя не пустое и состоит только из букв и пробелов, что почтовый индекс состоит из 6 цифр, а год основания находится в пределах от 1000 до 2100.

В то же время класс XmlValidatedSettlement не содержит аннотаций валидации на уровне Java-кода. Вместо этого вся валидация будет настраиваться через XML-конфигурацию, что позволяет гибко изменять логику валидации без изменения кода. В этом классе будет описан аналогичный набор полей, но без явных аннотаций для валидации. Валидация этих полей будет описана в специальном XML-файле, который будет связан с проектом.

Далее следует создание XML-конфигурации валидации. Для этого необходимо создать файл constraints.xml, в котором будет настроена валидация для полей класса XmlValidatedSettlement. В этом файле необходимо указать для каждого поля соответствующие ограничения, например, проверку на пустоту, правильность формата почтового индекса и корректность значения года основания. Важным моментом является указание регулярных выражений и значений для минимальных и максимальных значений, а также сообщений об ошибках, которые должны выводиться в случае несоответствия данных этим ограничениям.

После того как XML-конфигурация будет готова, нужно настроить файл validation-config.xml, который подключит созданный constraints.xml. Этот файл указывает на файл с конфигурацией валидации и обеспечивает интеграцию с механизмом валидации Jakarta, чтобы при валидации объектов автоматически применялись описанные в XML файлы ограничения.

Когда все компоненты настроены, следует провести тестирование. Сначала проверяется валидация с использованием аннотаций в классе CodeValidateSettlement. Для этого можно создать объект этого класса с некорректными данными, например, с пустым полем name или с неправильным значением для почтового индекса. Это позволит проверить, что система правильно выявляет ошибки и выводит соответствующие сообщения.

Затем тестируется валидация через XML, для чего создается объект класса XmlValidatedSettlement и проверяется, что валидация применяется корректно, как указано в файле constraints.xml. Для этого можно создать объект с некорректными данными и удостовериться, что ошибки валидации также выявляются и обрабатываются корректно.

В результате выполнения работы, должны быть реализованы две альтернативные схемы валидации — через аннотации в Java-коде и через XML-конфигурацию — с целью продемонстрировать гибкость подхода Jakarta Bean Validation.

# **РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

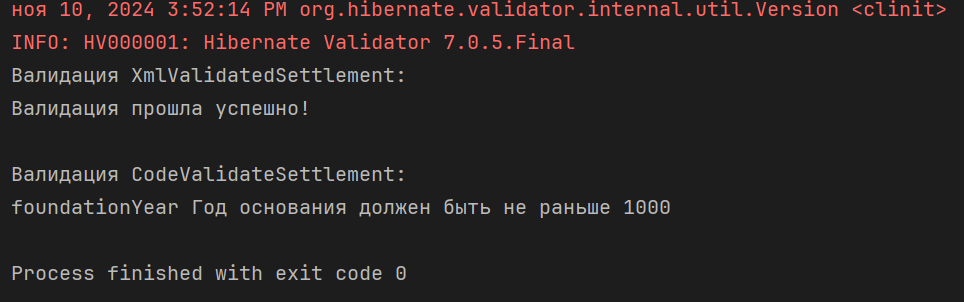


Рисунок 1 – Результат работы программы

# **КОД ПРОГРАММЫ**

package Lab4\_Validation.Model;  
  
import jakarta.validation.constraints.NotNull;  
import jakarta.validation.constraints.Pattern;  
import jakarta.validation.constraints.Min;  
import jakarta.validation.constraints.Max;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
public class CodeValidateSettlement {  
  
 @NotNull(message = "Название не должно быть пустым")  
 @Pattern(regexp = "^[A-Za-zА-Яа-яЁё\\s]+$", message = "Название должно содержать только буквы и пробелы")  
 private String name;  
  
 @NotNull(message = "Почтовый индекс не должен быть пустым")  
 @Pattern(regexp = "^\\d{6}$", message = "Почтовый индекс должен состоять из 6 цифр")  
 private String postalCode;  
  
 @NotNull(message = "Регион не должен быть пустым")  
 private String region;  
  
 @NotNull(message = "Страна не должна быть пустой")  
 private String country;  
  
 @Min(value = 1000, message = "Год основания должен быть не раньше 1000")  
 @Max(value = 2100, message = "Год основания не может быть больше 2100")  
 private int foundationYear;  
}

package Lab4\_Validation.Model;  
  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.NoArgsConstructor;  
import lombok.Setter;  
  
@Getter  
@Setter  
@AllArgsConstructor  
@NoArgsConstructor  
public class XmlValidatedSettlement {  
  
 private String name;  
 private String postalCode;  
 private String region;  
 private String country;  
 private int foundationYear;  
}

package Lab4\_Validation;  
  
import Lab4\_Validation.Model.CodeValidateSettlement;  
import Lab4\_Validation.Model.XmlValidatedSettlement;  
import jakarta.validation.ConstraintViolation;  
import jakarta.validation.Validation;  
import jakarta.validation.Validator;  
import jakarta.validation.ValidatorFactory;  
import org.hibernate.validator.HibernateValidator;  
import org.hibernate.validator.messageinterpolation.ParameterMessageInterpolator;  
  
import java.util.Set;  
  
public class Laboratory4 {  
  
 public static void main(String[] args) {  
 // Создание фабрики валидаторов  
 ValidatorFactory factory = Validation.byProvider(HibernateValidator.class)  
 .configure()  
 .messageInterpolator(new ParameterMessageInterpolator())  
 .buildValidatorFactory();  
 Validator validator = factory.getValidator();  
  
 // Создание и проверка объекта XmlValidatedSettlement  
 XmlValidatedSettlement xmlSettlement = new XmlValidatedSettlement();  
 xmlSettlement.setName("Москва");  
 xmlSettlement.setPostalCode("123456");  
 xmlSettlement.setRegion("Центральный");  
 xmlSettlement.setCountry("Россия");  
 xmlSettlement.setFoundationYear(1147);  
  
 System.out.println("Валидация XmlValidatedSettlement:");  
 validateAndPrint(validator, xmlSettlement);  
  
 // Создание и проверка объекта CodeValidateSettlement  
 CodeValidateSettlement codeSettlement = new CodeValidateSettlement();  
 codeSettlement.setName("Лондон");  
 codeSettlement.setPostalCode("654321");  
 codeSettlement.setRegion("Южный");  
 codeSettlement.setCountry("Великобритания");  
 codeSettlement.setFoundationYear(43);  
  
 System.out.println("\nВалидация CodeValidateSettlement:");  
 validateAndPrint(validator, codeSettlement);  
 }  
  
 // Метод для валидации и вывода результата валидации в консоль  
 private static <T> void validateAndPrint(Validator validator, T object) {  
 Set<ConstraintViolation<T>> violations = validator.validate(object);  
 if (violations.isEmpty()) {  
 System.out.println("Валидация прошла успешно!");  
 } else {  
 for (ConstraintViolation<T> violation : violations) {  
 System.out.println(violation.getPropertyPath() + " " + violation.getMessage());  
 }  
 }  
 }  
}

<constraint-mappings xmlns="http://jboss.org/xml/ns/javax/validation/mapping"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://jboss.org/xml/ns/javax/validation/mapping/validation-mapping-1.1.xsd"  
 version="1.1">  
 <bean class="Lab4\_Validation.Model.XmlValidatedSettlement" ignore-annotations="false">  
 <field name="name">  
 <constraint annotation="jakarta.validation.constraints.NotNull">  
 <message>Название не должно быть пустым</message>  
 </constraint>  
 <constraint annotation="jakarta.validation.constraints.Pattern">  
 <message>Название должно содержать только буквы и пробелы</message>  
 <element name="regexp">^[A-Za-zА-Яа-яЁё\\s]+$</element>  
 </constraint>  
 </field>  
 <field name="postalCode">  
 <constraint annotation="jakarta.validation.constraints.NotNull">  
 <message>Почтовый индекс не должен быть пустым</message>  
 </constraint>  
 <constraint annotation="jakarta.validation.constraints.Pattern">  
 <message>Почтовый индекс должен состоять из 6 цифр</message>  
 <element name="regexp">^\d{6}$</element>  
 </constraint>  
 </field>  
 <field name="region">  
 <constraint annotation="jakarta.validation.constraints.NotNull">  
 <message>Регион не должен быть пустым</message>  
 </constraint>  
 </field>  
 <field name="country">  
 <constraint annotation="jakarta.validation.constraints.NotNull">  
 <message>Страна не должна быть пустой</message>  
 </constraint>  
 </field>  
 <field name="foundationYear">  
 <constraint annotation="jakarta.validation.constraints.Min">  
 <message>Год основания должен быть не раньше 1000</message>  
 <element name="value">1000</element>  
 </constraint>  
 <constraint annotation="jakarta.validation.constraints.Max">  
 <message>Год основания не может быть больше 2100</message>  
 <element name="value">2100</element>  
 </constraint>  
 </field>  
 </bean>  
</constraint-mappings>

<validation-config xmlns="http://jboss.org/xml/ns/javax/validation/configuration"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://jboss.org/xml/ns/javax/validation/configuration-validation-configuration-1.1.xsd"  
 version="1.1">  
 <constraint-mapping>META-INF/constraints.xml</constraint-mapping>  
</validation-config>

# **ВЫВОД**

В ходе лабораторной работы была реализована система валидации данных с использованием Jakarta Bean Validation. Для этого были созданы два класса моделей: один с аннотациями валидации (CodeValidateSettlement), а второй с внешней настройкой валидации через XML-конфигурацию (XmlValidatedSettlement). В классе с аннотациями валидация осуществлялась напрямую через аннотации, такие как @NotNull, @Pattern, @Min и @Max, с заданием соответствующих сообщений об ошибках. В случае с XML-конфигурацией валидация настраивалась через файл constraints.xml, который определял ограничения для каждого поля. Проведенное тестирование показало правильную работу обеих схем валидации, обеспечив возможность выбора подхода в зависимости от требований проекта.