Федеральное государственное автономное образовательное учреждение высшего образования

«**Университет ИТМО»**

**Факультет ПИиКТ**

**Дисциплина: Технологии веб-сервисов**

**Лабораторная работа 3**

Выполнили: Гурин Евгений Иванович, Камышанская Ксения Васильевна

Преподаватель: Дергачев Андрей Михайлович

Группа: Р4216

Санкт-Петербург 2024г.

# Задача

Основываясь на информации из раздела 2.8, добавить поддержку обработки ошибок в сервис. Возможные ошибки, которые могут происходить при добавлении, изменении или удалении записей: неверное значение одного из полей; попытка изменить или удалить несуществующую запись.

В соответствии с изменениями сервиса необходимо обновить и клиентское приложение.

# Выполнение

1. В данной лабораторной были добавлены классы WebFault и exception с информацией об ошибках

Ошибка, информирующая о том, что экземпляр сущности не был найден в базе даных

EntityNotFound

|  |
| --- |
| /\*\*  \* Exception for entry not found.  \*/  @WebFault(faultBean = "guldilin.exceptions.EntityNotFoundFault")  @Getter  public class EntryNotFound extends Exception {  /\*\*  \* Fault information.  \*/  private final EntryNotFoundFault faultInfo;  /\*\*  \* Constructor.  \*  \* @param message The error message  \* @param faultInfo additional info  \*/  public EntryNotFound(final String message, final EntryNotFoundFault faultInfo) {  super(message);  this.faultInfo = faultInfo;  }  /\*\*  \* Constructor.  \*  \* @param message The error message  \* @param faultInfo additional info  \* @param cause exception cause  \*/  public EntryNotFound(final String message, final EntryNotFoundFault faultInfo, final Throwable cause) {  super(message, cause);  this.faultInfo = faultInfo;  }  /\*\*  \* Default constructor.  \*/  public EntryNotFound() {  super(ErrorMessages.NOT\_FOUND);  this.faultInfo = EntryNotFoundFault.builder().build();  }  /\*\*  \* Constructor.  \*  \* @param faultInfo additional info  \*/  public EntryNotFound(final EntryNotFoundFault faultInfo) {  super(ErrorMessages.NOT\_FOUND);  this.faultInfo = faultInfo;  }  /\*\*  \* Constructor.  \*  \* @param entity entity name  \* @param id entity identifier  \*/  public EntryNotFound(final String entity, final Integer id) {  super(ErrorMessages.NOT\_FOUND);  this.faultInfo = EntryNotFoundFault.builder().entity(entity).id(id).build();  }  } |

Ошибка невозможности фильтрации по определенному полю сущности

FieldIsNotFilterable

|  |
| --- |
| /\*\*  \* Exception for non-filterable fields.  \*/  @WebFault(faultBean = "guldilin.exceptions.FieldIsNotFilterableFault")  @Getter  public class FieldIsNotFilterable extends Exception {  /\*\*  \* Additional info about exception.  \*/  private final FieldIsNotFilterableFault faultInfo;  /\*\*  \* Constructor.  \*  \* @param message The error message  \* @param faultInfo additional info  \*/  public FieldIsNotFilterable(final String message, final FieldIsNotFilterableFault faultInfo) {  super(message);  this.faultInfo = faultInfo;  }  /\*\*  \* Constructor.  \*  \* @param message The error message  \* @param faultInfo additional info  \* @param cause exception cause  \*/  public FieldIsNotFilterable(  final String message, final FieldIsNotFilterableFault faultInfo, final Throwable cause) {  super(message, cause);  this.faultInfo = faultInfo;  }  /\*\*  \* Default constructor.  \*/  public FieldIsNotFilterable() {  super(ErrorMessages.NOT\_FOUND);  this.faultInfo = FieldIsNotFilterableFault.builder().build();  }  /\*\*  \* Constructor.  \*  \* @param faultInfo additional info  \*/  public FieldIsNotFilterable(final FieldIsNotFilterableFault faultInfo) {  super(ErrorMessages.NOT\_FOUND);  this.faultInfo = faultInfo;  }  } |

Ошибка валидации входных данных

ValidationFailed

|  |
| --- |
| /\*\*  \* Exception for validation.  \*/  @WebFault(faultBean = "guldilin.exceptions.FieldValidationFault")  @Getter  public class ValidationFailed extends Exception {  /\*\*  \* Info about fields errors.  \*/  private final FieldValidationFault faultInfo;  /\*\*  \* Constructor.  \*  \* @param message The error message  \* @param faultInfo additional info  \*/  public ValidationFailed(final String message, final FieldValidationFault faultInfo) {  super(message);  this.faultInfo = faultInfo;  }  /\*\*  \* Constructor.  \*  \* @param message The error message  \* @param faultInfo additional info  \* @param cause exception cause  \*/  public ValidationFailed(final String message, final FieldValidationFault faultInfo, final Throwable cause) {  super(message, cause);  this.faultInfo = faultInfo;  }  /\*\*  \* Default constructor.  \*/  public ValidationFailed() {  super(ErrorMessages.VALIDATION\_FAILED);  this.faultInfo = FieldValidationFault.builder().build();  }  /\*\*  \* Constructor from ConstraintViolationException.  \*  \* @param exception original validation exception  \*/  public ValidationFailed(final ConstraintViolationException exception) {  super(ErrorMessages.VALIDATION\_FAILED);  this.faultInfo = new FieldValidationFault(exception);  }  /\*\*  \* Constructor.  \*  \* @param faultInfo additional info  \*/  public ValidationFailed(final FieldValidationFault faultInfo) {  super(ErrorMessages.VALIDATION\_FAILED);  this.faultInfo = faultInfo;  }  } |

1. Соответствующие ошибки добавлены в сигнатуры методов сервисов и в код работы с БД.
2. Консольный клиент был обновлен для отображения ошибок в удобочитаемом виде

|  |
| --- |
| /\*\*  \* Exception handler utility class.  \*/  public final class ExceptionHandler {  private ExceptionHandler() {  // empty constructor for utility class  }  /\*\*  \* Handle exception and call specified handlers.  \*  \* @param e an exception  \*/  public static void handleException(final Exception e) {  if (e instanceof EntryNotFound) handleEntryNotFound((EntryNotFound) e);  else if (e instanceof ValidationFailed) handleValidationFailed((ValidationFailed) e);  else if (e instanceof FieldIsNotFilterable) handleFieldIsNotFilterable((FieldIsNotFilterable) e);  else handleDefault(e);  }  /\*\*  \* Handler for EntryNotFound.  \*  \* @param e an exception  \*/  public static void handleEntryNotFound(final EntryNotFound e) {  System.err.printf(  "ERROR: %s for entity %s with id %d\n",  e.getMessage(), e.getFaultInfo().getEntity(), e.getFaultInfo().getId());  }  /\*\*  \* Handler for ValidationFailed.  \*  \* @param e an exception  \*/  public static void handleValidationFailed(final ValidationFailed e) {  System.err.printf("ERROR: %s for\n", e.getMessage());  e.getFaultInfo().getErrors().forEach(c -> System.out.printf("\t- %s: %s\n", c.getField(), c.getMessage()));  }  /\*\*  \* Handler for FieldIsNotFilterable.  \*  \* @param e an exception  \*/  public static void handleFieldIsNotFilterable(final FieldIsNotFilterable e) {  System.err.printf("ERROR: %s for %s\n", e.getMessage(), e.getFaultInfo().getField());  }  /\*\*  \* Handler default exception.  \*  \* @param e an exception  \*/  public static void handleDefault(final Exception e) {  System.err.printf("ERROR: %s\n", e.getMessage());  }  } |

# Вывод

В результате выполнения лабораторной работы был создан набор классов Fault для выброса исключений в ожидаемых случаях. Также была добавлена информация в классы конкретных ошибок. Консольный клиент был обновлен с использованием классов исключений, которые выводятся в консоль в читаемом виде.