**Министерство образования и науки Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИТМО**”

**Факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА № 3**

**ПО ДИСЦИПЛИНЕ «ТЕХНОЛОГИИ ВЕБ-СЕРВИСОВ»**

Студент: Норин Евгений Рустамович

Группа: P41142

Преподаватель: Дергачев Андрей Михайлович

Санкт-Петербург

2020

**Задание:**

Основываясь на информации из раздела 2.8, добавить поддержку обработки ошибок в сервис. Возможные ошибки, которые могут происходить при добавлении новых записей – например, неверное значение одного из полей, при изменении, удалении – попытка изменить или удалить несуществующую запись. В соответствии с изменениями сервиса необходимо обновить и клиентское приложение.

**Выполнение работы:**

В результате выполнения второй лабораторной работы был написан SOAP-сервис в виде standalone-приложения с операциями CRUD. В классы ArticleWebServise.java и ArticlesDao.java были добавлены исключительные ситуации:

* *IllegalIdException* (листинг 1.1) – ошибка выбрасывается в методах update и delete в случае id=null
* *InsertingException* (листинг 1.2) – ошибка выбрасывается в методе insert в случае, если что то пошло не так во время вставки в БД
* *InvalidCreatingParametersException* (листинг 1.3) – ошибка выбрасывается в случае невалидных параметров
* *RowsNotAffectedException* (листинг 1.4)– ошибка выбрасывается в случае, если после операций update или delete ни одна из строк не была изменена

Листинг 1.1 *IllegalIdException.java*

**import** javax.xml.ws.WebFault;  
  
@WebFault(faultBean = **"ru.itmo.webservices.thirdlab.standalone.exceptions.ArticleServiceFault"**)  
**public class** IllegalIdException **extends** Exception {  
 **private static final long *serialVersionUID*** = -6647544772732631047L;  
 **private final** ArticleServiceFault **fault**;  
  
 **public** IllegalIdException(String message, ArticleServiceFault fault) {  
 **super**(message);  
 **this**.**fault** = fault;  
 }  
  
 **public** IllegalIdException(String message, ArticleServiceFault fault, Throwable cause) {  
 **super**(message, cause);  
 **this**.**fault** = fault;  
 }  
  
 **public** ArticleServiceFault getFaultInfo() {  
 **return fault**;  
 }  
}

Листинг 1.2 *InsertingException.java*

**import** javax.xml.ws.WebFault;  
  
@WebFault(faultBean = **"ru.itmo.webservices.thirdlab.standalone.exceptions.ArticleServiceFault"**)  
**public class** InsertingException **extends** Exception {  
 **private static final long *serialVersionUID*** = -6647544772732631047L;  
 **private final** ArticleServiceFault **fault**;  
  
 **public** InsertingException(String message, ArticleServiceFault fault) {  
 **super**(message);  
 **this**.**fault** = fault;  
 }  
  
 **public** InsertingException(String message, ArticleServiceFault fault, Throwable cause) {  
 **super**(message, cause);  
 **this**.**fault** = fault;  
 }  
  
 **public** ArticleServiceFault getFaultInfo() {  
 **return fault**;  
 }  
}

Листинг 1.3 *InvalidCreatingParametersException.java*

**import** javax.xml.ws.WebFault;  
  
@WebFault(faultBean = **"ru.itmo.webservices.thirdlab.standalone.exceptions.ArticleServiceFault"**)  
**public class** InvalidCreatingParametersException **extends** Exception {  
 **private static final long *serialVersionUID*** = -6647544772732631047L;  
 **private final** ArticleServiceFault **fault**;  
  
 **public** InvalidCreatingParametersException(String message, ArticleServiceFault fault) {  
 **super**(message);  
 **this**.**fault** = fault;  
 }  
  
 **public** InvalidCreatingParametersException(String message, ArticleServiceFault fault, Throwable cause) {  
 **super**(message, cause);  
 **this**.**fault** = fault;  
 }  
  
 **public** ArticleServiceFault getFaultInfo() {  
 **return fault**;  
 }  
}

Листинг 1.4 *RowsNotAffectedException.java*

@WebFault(faultBean = **"ru.itmo.webservices.thirdlab.standalone.exceptions.ArticleServiceFault"**)  
**public class** RowsNotAffectedException **extends** Exception {  
 **private static final long *serialVersionUID*** = -6647544772732631047L;  
 **private final** ArticleServiceFault **fault**;  
  
 **public** RowsNotAffectedException(String message, ArticleServiceFault fault) {  
 **super**(message);  
 **this**.**fault** = fault;  
 }  
  
 **public** RowsNotAffectedException(String message, ArticleServiceFault fault, Throwable cause) {  
 **super**(message, cause);  
 **this**.**fault** = fault;  
 }  
  
 **public** ArticleServiceFault getFaultInfo() {  
 **return fault**;  
 }  
}

Листинг 1.5 *ArticleServiceFault.java*

**package** ru.itmo.webservices.thirdlab.standalone.exceptions;  
  
**public class** ArticleServiceFault {  
 **private static final** String ***DEFAULT\_MESSAGE*** = **"Parameter q cannot be null or empty"**;  
 **private** String **message**;  
  
 **public** String getMessage() {  
 **return message**;  
 }  
  
 **public void** setMessage(String message) {  
 **this**.**message** = message;  
 }  
  
 **public static** ArticleServiceFault defaultInstance() {  
 ArticleServiceFault fault = **new** ArticleServiceFault();  
 fault.**message** = ***DEFAULT\_MESSAGE***;  
 **return** fault;  
 }  
}

Листинг 1.6 *ArticlesDao.java*

**package** ru.itmo.webservices.thirdlab.standalone.dao;  
  
  
**import** ru.itmo.webservices.thirdlab.standalone.exceptions.ArticleServiceFault;  
**import** ru.itmo.webservices.thirdlab.standalone.exceptions.InsertingException;  
**import** ru.itmo.webservices.thirdlab.standalone.exceptions.RowsNotAffectedException;  
**import** ru.itmo.webservices.thirdlab.standalone.pojo.Article;  
  
**import** java.sql.\*;  
**import** java.util.ArrayList;  
**import** java.util.List;  
**import** java.util.logging.Level;  
**import** java.util.logging.Logger;  
  
**public class** ArticlesDao {  
 **private** Connection **connection**;  
  
 **public** ArticlesDao() {  
 **connection** = ConnectionUtil.*getConnection*();  
 }  
  
 **public** List<Article> select(String authorId, Long hIndex, String articleName, String articleDesc, Long dateAdded) {  
 List<Article> articles = **new** ArrayList<>();  
 **try** (Connection connection = ConnectionUtil.*getConnection*()) {  
 Statement stmt = connection.createStatement();  
 ResultSet rs = stmt.executeQuery(  
 QueryBuilder.*buildSelect*(  
 authorId,  
 hIndex,  
 articleName,  
 articleDesc,  
 dateAdded  
 )  
 );  
  
 **while** (rs.next()) {  
 articles.add(  
 **new** Article(  
 rs.getString(**"author\_id"**),  
 rs.getLong(**"h\_index"**),  
 rs.getString(**"article\_name"**),  
 rs.getString(**"article\_desc"**),  
 rs.getLong(**"date\_added"**)  
 )  
 );  
 }  
 } **catch** (SQLException ex) {  
 Logger.*getLogger*(ArticlesDao.**class**.getName()).log(Level.***SEVERE***, **null**, ex);  
 }  
  
 **return** articles;  
 }  
  
 **public long** insert(String authorId, Long hIndex, String articleName, String articleDesc, Long dateAdded) **throws** InsertingException {  
 String sql = **"INSERT INTO Article (author\_id, h\_index, article\_name, article\_desc, date\_added) VALUES (?, ?, ?, ?, ?)"**;  
 **long** id = 0;  
 **try** {  
 PreparedStatement preparedStatement = **this**.**connection**.prepareStatement(sql, Statement.***RETURN\_GENERATED\_KEYS***);  
 preparedStatement.setString(1, authorId);  
 preparedStatement.setLong(2, hIndex);  
 preparedStatement.setString(3, articleName);  
 preparedStatement.setString(4, articleDesc);  
 preparedStatement.setLong(5, dateAdded);  
  
 preparedStatement.executeUpdate();  
  
 ResultSet generatedKeys = preparedStatement.getGeneratedKeys();  
 **if** (generatedKeys.next()) {  
 id = generatedKeys.getLong(1);  
 }  
 } **catch** (SQLException e) { }  
 **if** (id == 0) {  
 ArticleServiceFault fault = ArticleServiceFault.*defaultInstance*();  
 fault.setMessage(**"Error during creation entity"**);  
 **throw new** InsertingException(**"Error during creation entity"**, fault);  
 }  
  
 **return** id;  
 }  
  
 **public int** update(**long** id, String authorId, Long hIndex, String articleName, String articleDesc, Long dateAdded) **throws** RowsNotAffectedException {  
 String sql = QueryBuilder.*buildUpdate*(id, authorId, hIndex, articleName, articleDesc, dateAdded);  
 **int** affectedRows = 0;  
 **try** {  
 PreparedStatement preparedStatement = **this**.**connection**.prepareStatement(sql);  
 affectedRows = preparedStatement.executeUpdate();  
 } **catch** (SQLException e) { }  
 **if** (affectedRows == 0) {  
 ArticleServiceFault fault = ArticleServiceFault.*defaultInstance*();  
 fault.setMessage(**"There are no records with such id"**);  
 **throw new** RowsNotAffectedException(**"There are no records with such id"**, fault);  
 }  
 **return** affectedRows;  
 }  
  
 **public int** delete(**long** id) **throws** RowsNotAffectedException {  
 String sql = **"DELETE FROM article WHERE article\_id = ?"**;  
 **int** affectedRows = 0;  
 **try** {  
 PreparedStatement preparedStatement = **this**.**connection**.prepareStatement(sql);  
 preparedStatement.setLong(1, id);  
  
 affectedRows = preparedStatement.executeUpdate();  
 } **catch** (SQLException e) { }  
 **if** (affectedRows == 0) {  
 ArticleServiceFault fault = ArticleServiceFault.*defaultInstance*();  
 fault.setMessage(**"There are no records with such id"**);  
 **throw new** RowsNotAffectedException(**"There are no records with such id"**, fault);  
 }  
 **return** affectedRows;  
 }  
}

Листинг 1.7 *ArticleWebService.java*

**package** ru.itmo.webservices.thirdlab.standalone;  
  
**import** ru.itmo.webservices.thirdlab.standalone.dao.ArticlesDao;  
**import** ru.itmo.webservices.thirdlab.standalone.exceptions.\*;  
**import** ru.itmo.webservices.thirdlab.standalone.pojo.Article;  
  
**import** javax.jws.WebMethod;  
**import** javax.jws.WebParam;  
**import** javax.jws.WebService;  
**import** java.util.List;  
  
@WebService(serviceName = **"ArticlesService"**)  
**public class** ArticleWebService {  
 @WebMethod  
 **public** List<Article> getArticles(@WebParam(name = **"authorId"**) String authorId,  
 @WebParam(name = **"hIndex"**) Long hIndex,  
 @WebParam(name = **"articleName"**) String articleName,  
 @WebParam(name = **"articleDesc"**) String articleDesc,  
 @WebParam(name = **"dateAdded"**) Long dateAdded) {  
 ArticlesDao dao = **new** ArticlesDao();  
 **return** dao.select(authorId, hIndex, articleName, articleDesc, dateAdded);  
 }  
  
 @WebMethod  
 **public long** createArticle(@WebParam(name = **"authorId"**) String authorId,  
 @WebParam(name = **"hIndex"**) Long hIndex,  
 @WebParam(name = **"articleName"**) String articleName,  
 @WebParam(name = **"articleDesc"**) String articleDesc,  
 @WebParam(name = **"dateAdded"**) Long dateAdded) **throws** InsertingException, InvalidCreatingParametersException {  
 ArticleServiceFault fault = ArticleServiceFault.*defaultInstance*();  
 fault.setMessage(**"Invalid creating parameter"**);  
  
 **if** (authorId == **null** || authorId.trim().isEmpty()) {  
 fault.setMessage(**"Parameter authorId cannot be null or empty"**);  
 **throw new** InvalidCreatingParametersException(**"Invalid creating parameter"**, fault);  
 }  
  
 **if** (hIndex == **null**) {  
 fault.setMessage(**"Parameter hIndex cannot be null"**);  
 **throw new** InvalidCreatingParametersException(**"Invalid creating parameter"**, fault);  
 }  
  
 **if** (articleName == **null** || articleName.trim().isEmpty()) {  
 fault.setMessage(**"Parameter articleName cannot be null or empty"**);  
 **throw new** InvalidCreatingParametersException(**"Invalid creating parameter"**, fault);  
 }  
  
 **if** (articleDesc == **null** || articleDesc.trim().isEmpty()) {  
 fault.setMessage(**"Parameter articleDesc cannot be null or empty"**);  
 **throw new** InvalidCreatingParametersException(**"Invalid creating parameter"**, fault);  
 }  
  
 **if** (dateAdded == **null**) {  
 fault.setMessage(**"Parameter dateAdded cannot be null"**);  
 **throw new** InvalidCreatingParametersException(**"Invalid creating parameter"**, fault);  
 }  
  
 ArticlesDao dao = **new** ArticlesDao();  
 **return** dao.insert(authorId, hIndex, articleName, articleDesc, dateAdded);  
 }  
  
 @WebMethod  
 **public long** updateArticle(@WebParam(name = **"id"**) Long id,  
 @WebParam(name = **"authorId"**) String authorId,  
 @WebParam(name = **"hIndex"**) Long hIndex,  
 @WebParam(name = **"articleName"**) String articleName,  
 @WebParam(name = **"articleDesc"**) String articleDesc,  
 @WebParam(name = **"dateAdded"**) Long dateAdded) **throws** IllegalIdException, RowsNotAffectedException {  
 ArticlesDao dao = **new** ArticlesDao();  
 **if** (id == **null**) {  
 ArticleServiceFault fault = ArticleServiceFault.*defaultInstance*();  
 fault.setMessage(**"Parameter id cannot be null"**);  
 **throw new** IllegalIdException(**"Parameter id cannot be null"**, fault);  
 }  
 **return** dao.update(id, authorId, hIndex, articleName, articleDesc, dateAdded);  
 }  
  
 @WebMethod  
 **public int** deleteArticle(@WebParam(name = **"id"**) Long id) **throws** IllegalIdException, RowsNotAffectedException {  
 ArticlesDao dao = **new** ArticlesDao();  
 **if** (id == **null**) {  
 ArticleServiceFault fault = ArticleServiceFault.*defaultInstance*();  
 fault.setMessage(**"Parameter id cannot be null"**);  
 **throw new** IllegalIdException(**"Parameter id cannot be null"**, fault);  
 }  
 **return** dao.delete(id);  
 }  
}

WSDL сервиса по адресу <http://localhost:8080/ArticleService?wsdl> приведен в листинге 1.8

Листинг 1.8 *WSDL сервиса*

<!--

Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.1035 svn-revision#5f6196f2b90e9460065a4c2f4e30e065b245e51e.

-->

<!--

Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.1035 svn-revision#5f6196f2b90e9460065a4c2f4e30e065b245e51e.

-->

<definitions

xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"

xmlns:wsp="http://www.w3.org/ns/ws-policy"

xmlns:wsp1\_2="http://schemas.xmlsoap.org/ws/2004/09/policy"

xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"

xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"

xmlns:tns="http://standalone.thirdlab.webservices.itmo.ru/"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://standalone.thirdlab.webservices.itmo.ru/" name="ArticlesService">

<types>

<xsd:schema>

<xsd:import namespace="http://standalone.thirdlab.webservices.itmo.ru/" schemaLocation="http://localhost:8080/ArticleService?xsd=1"/>

</xsd:schema>

</types>

<message name="getArticles">

<part name="parameters" element="tns:getArticles"/>

</message>

<message name="getArticlesResponse">

<part name="parameters" element="tns:getArticlesResponse"/>

</message>

<message name="createArticle">

<part name="parameters" element="tns:createArticle"/>

</message>

<message name="createArticleResponse">

<part name="parameters" element="tns:createArticleResponse"/>

</message>

<message name="InsertingException">

<part name="fault" element="tns:InsertingException"/>

</message>

<message name="InvalidCreatingParametersException">

<part name="fault" element="tns:InvalidCreatingParametersException"/>

</message>

<message name="updateArticle">

<part name="parameters" element="tns:updateArticle"/>

</message>

<message name="updateArticleResponse">

<part name="parameters" element="tns:updateArticleResponse"/>

</message>

<message name="IllegalIdException">

<part name="fault" element="tns:IllegalIdException"/>

</message>

<message name="RowsNotAffectedException">

<part name="fault" element="tns:RowsNotAffectedException"/>

</message>

<message name="deleteArticle">

<part name="parameters" element="tns:deleteArticle"/>

</message>

<message name="deleteArticleResponse">

<part name="parameters" element="tns:deleteArticleResponse"/>

</message>

<portType name="ArticleWebService">

<operation name="getArticles">

<input wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/getArticlesRequest" message="tns:getArticles"/>

<output wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/getArticlesResponse" message="tns:getArticlesResponse"/>

</operation>

<operation name="createArticle">

<input wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/createArticleRequest" message="tns:createArticle"/>

<output wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/createArticleResponse" message="tns:createArticleResponse"/>

<fault message="tns:InsertingException" name="InsertingException" wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/createArticle/Fault/InsertingException"/>

<fault message="tns:InvalidCreatingParametersException" name="InvalidCreatingParametersException" wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/createArticle/Fault/InvalidCreatingParametersException"/>

</operation>

<operation name="updateArticle">

<input wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/updateArticleRequest" message="tns:updateArticle"/>

<output wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/updateArticleResponse" message="tns:updateArticleResponse"/>

<fault message="tns:IllegalIdException" name="IllegalIdException" wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/updateArticle/Fault/IllegalIdException"/>

<fault message="tns:RowsNotAffectedException" name="RowsNotAffectedException" wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/updateArticle/Fault/RowsNotAffectedException"/>

</operation>

<operation name="deleteArticle">

<input wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/deleteArticleRequest" message="tns:deleteArticle"/>

<output wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/deleteArticleResponse" message="tns:deleteArticleResponse"/>

<fault message="tns:IllegalIdException" name="IllegalIdException" wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/deleteArticle/Fault/IllegalIdException"/>

<fault message="tns:RowsNotAffectedException" name="RowsNotAffectedException" wsam:Action="http://standalone.thirdlab.webservices.itmo.ru/ArticleWebService/deleteArticle/Fault/RowsNotAffectedException"/>

</operation>

</portType>

<binding name="ArticleWebServicePortBinding" type="tns:ArticleWebService">

<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>

<operation name="getArticles">

<soap:operation soapAction=""/>

<input>

<soap:body use="literal"/>

</input>

<output>

<soap:body use="literal"/>

</output>

</operation>

<operation name="createArticle">

<soap:operation soapAction=""/>

<input>

<soap:body use="literal"/>

</input>

<output>

<soap:body use="literal"/>

</output>

<fault name="InsertingException">

<soap:fault name="InsertingException" use="literal"/>

</fault>

<fault name="InvalidCreatingParametersException">

<soap:fault name="InvalidCreatingParametersException" use="literal"/>

</fault>

</operation>

<operation name="updateArticle">

<soap:operation soapAction=""/>

<input>

<soap:body use="literal"/>

</input>

<output>

<soap:body use="literal"/>

</output>

<fault name="IllegalIdException">

<soap:fault name="IllegalIdException" use="literal"/>

</fault>

<fault name="RowsNotAffectedException">

<soap:fault name="RowsNotAffectedException" use="literal"/>

</fault>

</operation>

<operation name="deleteArticle">

<soap:operation soapAction=""/>

<input>

<soap:body use="literal"/>

</input>

<output>

<soap:body use="literal"/>

</output>

<fault name="IllegalIdException">

<soap:fault name="IllegalIdException" use="literal"/>

</fault>

<fault name="RowsNotAffectedException">

<soap:fault name="RowsNotAffectedException" use="literal"/>

</fault>

</operation>

</binding>

<service name="ArticlesService">

<port name="ArticleWebServicePort" binding="tns:ArticleWebServicePortBinding">

<soap:address location="http://localhost:8080/ArticleService"/>

</port>

</service>

</definitions>

Клиентский код сгенерирован командой:

~/IdeaProjects/webservices-labs(master) » wsimport -Xnocompile http://localhost:8080/ArticleService\?wsdl -d /Users/e.norin/IdeaProjects/webservices-labs/thirdlab-client/src/main/java

*WebServiceClient.java* (Листинг 1.9) создан для демонстрационных целей.

Листинг 1.9 *WebServiceClient.java*

Результат выполнения клиентского приложения представлен на Рисунке 1.1

Рисунок 1.1 Запуск клиентского приложения



**Вывод**: в ходе выполнения работы, основываясь на информации из раздела 2.8, была добавлена обработка ошибок в сервис. В соответствии с изменениями сервиса также было обновлено и клиентское приложение.