**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

по лабораторной работе №2

по дисциплине «**Технологии программирования**»

Автор: Кошкин Михаил Сергеевич

Факультет: ИТиП

Группа: М32051



Санкт-Петербург 2022

Описание:

Нужно написать сервис по учету котиков и их владельцев.

Существующая информация о котиках:

Имя

Дата рождения

Порода

Цвет (один из заранее заданных вариантов)

Хозяин

Список котиков, с которыми дружит этот котик (из представленных в базе)

Существующая информация о хозяевах:

Имя

Дата рождения

Список котиков

Сервис должен реализовывать архитектуру controller-service-dao.

Вся информация хранится в БД PostgreSQL. Для связи с БД должен использоваться Hibernate.

Проект должен собираться с помощью Maven или Gradle (на выбор студента). Слой доступа к данным и сервисный слой должны являться двумя разными модулями Maven/Gradle. При этом проект должен полностью собираться одной командой.

При тестировании рекомендуется использовать Mockito, чтобы избежать подключения к реальным базам данных. Фреймворк для тестирования рекомендуется Junit 5.

В данной лабораторной нельзя использовать Spring или подобные ему фреймворки.

Исходный код:

Добавлен файл kotiki-java/.gitignore:

# Project related fils

db\_files/\*

# Maven build directory

target

# Compiled class file

\*.class

# Log file

\*.log

# BlueJ files

\*.ctxt

# Mobile Tools for Java (J2ME)

.mtj.tmp/

\*.idea

\*.iml

# Package Files #

\*.jar

\*.war

\*.nar

\*.ear

\*.zip

\*.tar.gz

\*.rar

# virtual machine crash logs, see http://www.java.com/en/download/help/error\_hotspot.xml

hs\_err\_pid\*

Добавлен файл kotiki-java/api/pom.xml:

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<parent>

<artifactId>kotiki-java</artifactId>

<groupId>kotiki-java</groupId>

<version>1.0-SNAPSHOT</version>

</parent>

<modelVersion>4.0.0</modelVersion>

<artifactId>api</artifactId>

<properties>

<maven.compiler.source>13</maven.compiler.source>

<maven.compiler.target>13</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>kotiki-java</groupId>

<artifactId>dal</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>4.4.0</version>

</dependency>

</dependencies>

</project>

Добавлен файл kotiki-java/api/src/main/java/controllers/CatController.java:

package controllers;

import models.Cat;

import services.CatService;

import tools.ControllerException;

import java.util.List;

public class CatController {

private final CatService service;

public CatController() {

service = new CatService();

}

public CatController(CatService service) {

this.service = service;

}

public List<Cat> getAll() {

try {

return service.getAll();

} catch (Exception e){

throw new ControllerException(e.getMessage());

}

}

public Cat get(int id) {

try {

return service.get(id);

} catch (Exception e) {

throw new ControllerException(e.getMessage());

}

}

public void add(Cat cat) {

try {

service.add(cat);

} catch (Exception e) {

throw new ControllerException(e.getMessage());

}

}

public void delete(int id) {

try {

service.delete(id);

} catch (Exception e) {

throw new ControllerException(e.getMessage());

}

}

public void update(Cat cat) {

try {

service.update(cat);

} catch (Exception e) {

throw new ControllerException(e.getMessage());

}

}

}

Добавлен файл kotiki-java/api/src/main/java/controllers/OwnerController.java:

package controllers;

import models.Owner;

import services.OwnerService;

import tools.ControllerException;

import java.util.List;

public class OwnerController {

private final OwnerService service;

public OwnerController() {

service = new OwnerService();

}

public OwnerController(OwnerService service) {

this.service = service;

}

public List<Owner> getAll() {

try {

return service.getAll();

} catch (Exception e){

throw new ControllerException(e.getMessage());

}

}

public Owner get(int id) {

try {

return service.get(id);

} catch (Exception e) {

throw new ControllerException(e.getMessage());

}

}

public void add(Owner owner) {

try {

service.add(owner);

} catch (Exception e) {

throw new ControllerException(e.getMessage());

}

}

public void delete(int id) {

try {

service.delete(id);

} catch (Exception e) {

throw new ControllerException(e.getMessage());

}

}

public void update(Owner owner) {

try {

service.update(owner);

} catch (Exception e) {

throw new ControllerException(e.getMessage());

}

}

}

Добавлен файл kotiki-java/api/src/main/java/services/CatService.java:

package services;

import dao.CatDao;

import dao.Dao;

import models.Cat;

import java.util.List;

public class CatService {

private final Dao<Cat> dao;

public CatService() {

dao = new CatDao();

}

public CatService(Dao<Cat> dao) {

this.dao = dao;

}

public List<Cat> getAll() {

return dao.getAll();

}

public Cat get(int id) {

return dao.get(id);

}

public void add(Cat cat) {

dao.add(cat);

}

public void delete(int id) {

dao.delete(id);

}

public void update(Cat cat) {

dao.update(cat);

}

}

Добавлен файл kotiki-java/api/src/main/java/services/OwnerService.java:

package services;

import dao.Dao;

import dao.OwnerDao;

import models.Owner;

import java.util.List;

public class OwnerService {

private final Dao<Owner> dao;

public OwnerService() {

dao = new OwnerDao();

}

public OwnerService(Dao<Owner> dao) {

this.dao = dao;

}

public List<Owner> getAll() {

return dao.getAll();

}

public Owner get(int id) {

return dao.get(id);

}

public void add(Owner owner) {

dao.add(owner);

}

public void delete(int id) {

dao.delete(id);

}

public void update(Owner owner) {

dao.update(owner);

}

}

Добавлен файл kotiki-java/api/src/main/java/tools/ControllerException.java:

package tools;

public class ControllerException extends RuntimeException {

public ControllerException() {

super();

}

public ControllerException(String message) {

super(message);

}

}

Добавлен файл kotiki-java/api/src/test/java/services/CatServiceTest.java:

package services;

import dao.CatDao;

import dao.Dao;

import enums.Color;

import models.Cat;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

public class CatServiceTest {

private CatService service;

private Dao<Cat> dao;

@BeforeEach

public void setup() {

dao = mock(CatDao.class);

service = new CatService(dao);

}

@Test

public void testGetAll() {

var cat1 = Cat.builder().id(1).name("Cat").build();

var cat2 = Cat.builder().id(2).name("Also A Cat").build();

var cats = List.of(cat1, cat2);

when(dao.getAll()).thenReturn(cats);

List<Cat> result = service.getAll();

verify(dao).getAll();

assertEquals(2, result.size());

assertEquals(cat1, result.get(0));

assertEquals(cat2, result.get(1));

}

@Test

public void testGet() {

var cat = Cat.builder().id(1).name("Name").build();

when(dao.get(1)).thenReturn(cat);

Cat result = service.get(1);

verify(dao).get(anyInt());

assertEquals(cat, result);

}

@Test

public void testAdd() {

var cat = Cat.builder()

.id(1)

.name("Name")

.friends(List.of(new Cat("Another Cat")))

.build();

service.add(cat);

verify(dao, times(1)).add(any());

}

@Test

public void testUpdate() {

var cat = new Cat("Kitty");

cat.setColor(Color.BLUE);

service.update(cat);

verify(dao, times(1)).update(any());

}

@Test

public void testDelete() {

service.delete(1);

verify(dao, times(1)).delete(anyInt());

}

}

Добавлен файл kotiki-java/api/src/test/java/services/OwnerServiceTest.java:

package services;

import dao.Dao;

import dao.OwnerDao;

import models.Cat;

import models.Owner;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import java.time.Clock;

import java.time.LocalDate;

import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

public class OwnerServiceTest {

private OwnerService service;

private Dao<Owner> dao;

@BeforeEach

public void setup() {

dao = mock(OwnerDao.class);

service = new OwnerService(dao);

}

@Test

public void testGetAll() {

var owner1 = Owner.builder().id(1).name("Name1").build();

var owner2 = Owner.builder().id(2).name("Name2").build();

var owners = List.of(owner1, owner2);

when(dao.getAll()).thenReturn(owners);

List<Owner> result = service.getAll();

verify(dao).getAll();

assertEquals(2, result.size());

assertEquals(owner1, result.get(0));

assertEquals(owner2, result.get(1));

}

@Test

public void testGet() {

var owner = Owner.builder().id(1).name("Name").build();

when(dao.get(1)).thenReturn(owner);

Owner result = service.get(1);

verify(dao).get(anyInt());

assertEquals(owner, result);

}

@Test

public void testAdd() {

var owner = Owner.builder()

.id(1)

.name("SomeName")

.birthDate(LocalDate.now())

.cats(List.of(new Cat("Cat")))

.build();

service.add(owner);

verify(dao, times(1)).add(any());

}

@Test

public void testUpdate() {

var owner = new Owner("Namely name");

owner.setBirthDate(LocalDate.now(Clock.systemUTC()));

service.update(owner);

verify(dao, times(1)).update(any());

}

@Test

public void testDelete() {

service.delete(1);

verify(dao, times(1)).delete(anyInt());

}

}

Добавлен файл kotiki-java/dal/pom.xml:

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<parent>

<artifactId>kotiki-java</artifactId>

<groupId>kotiki-java</groupId>

<version>1.0-SNAPSHOT</version>

</parent>

<modelVersion>4.0.0</modelVersion>

<artifactId>dal</artifactId>

<properties>

<maven.compiler.source>13</maven.compiler.source>

<maven.compiler.target>13</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>42.3.3</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.6.7.Final</version>

</dependency>

</dependencies>

</project>

Добавлен файл kotiki-java/dal/src/main/java/dao/CatDao.java:

package dao;

import models.Cat;

import org.hibernate.Session;

import org.hibernate.Transaction;

import utils.HibernateSessionFactory;

import java.util.List;

public class CatDao implements Dao<Cat> {

public List<Cat> getAll() {

return HibernateSessionFactory

.getSession()

.createQuery("from Cat order by id", Cat.class)

.list();

}

public Cat get(int id) {

return HibernateSessionFactory

.getSession()

.get(Cat.class, id);

}

public void add(Cat cat) {

Session session = HibernateSessionFactory.getSession();

Transaction transaction = session.beginTransaction();

session.save(cat);

transaction.commit();

session.close();

}

public void update(Cat cat) {

Session session = HibernateSessionFactory.getSession();

Transaction transaction = session.beginTransaction();

session.merge(cat);

transaction.commit();

session.close();

}

public void delete(int id) {

Session session = HibernateSessionFactory.getSession();

Cat cat = session.get(Cat.class, id);

Transaction transaction = session.beginTransaction();

session.delete(cat);

transaction.commit();

session.close();

}

}

Добавлен файл kotiki-java/dal/src/main/java/dao/Dao.java:

package dao;

import java.util.List;

public interface Dao<T> {

List<T> getAll();

T get(int id);

void add(T entity);

void update(T entity);

void delete(int id);

}

Добавлен файл kotiki-java/dal/src/main/java/dao/OwnerDao.java:

package dao;

import models.Owner;

import org.hibernate.Session;

import org.hibernate.Transaction;

import utils.HibernateSessionFactory;

import java.util.List;

public class OwnerDao implements Dao<Owner> {

public List<Owner> getAll() {

return HibernateSessionFactory

.getSession()

.createQuery("from Owner order by id", Owner.class)

.list();

}

public Owner get(int id) {

return HibernateSessionFactory

.getSession()

.get(Owner.class, id);

}

public void add(Owner owner) {

Session session = HibernateSessionFactory.getSession();

Transaction transaction = session.beginTransaction();

session.save(owner);

transaction.commit();

session.close();

}

public void update(Owner owner) {

Session session = HibernateSessionFactory.getSession();

Transaction transaction = session.beginTransaction();

session.merge(owner);

transaction.commit();

session.close();

}

public void delete(int id) {

Session session = HibernateSessionFactory.getSession();

Owner owner = session.get(Owner.class, id);

Transaction transaction = session.beginTransaction();

session.delete(owner);

transaction.commit();

session.close();

}

}

Добавлен файл kotiki-java/dal/src/main/java/enums/Color.java:

package enums;

public enum Color {

BLACK,

BLUE,

CHOCOLATE,

CINNAMON,

CREAM,

FAWN,

GRAY,

LILAC,

ORANGE,

RED,

WHITE

}

Добавлен файл kotiki-java/dal/src/main/java/models/Cat.java:

package models;

import enums.Color;

import lombok.\*;

import org.hibernate.Hibernate;

import org.hibernate.annotations.CascadeType;

import org.hibernate.annotations.Cascade;

import tools.DalException;

import javax.persistence.\*;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

import java.util.stream.Collectors;

import java.util.stream.Stream;

@Entity

@Table(name = "cats")

@NoArgsConstructor

@RequiredArgsConstructor

@Getter

@Setter

@ToString

public class Cat {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(nullable = false)

private int id;

@Column(nullable = false)

@NonNull

private String name;

@Column(name = "birth\_date")

private LocalDate birthDate;

private String breed;

@Enumerated(EnumType.STRING)

private Color color;

@ManyToOne

@Cascade({

CascadeType.PERSIST,

CascadeType.MERGE,

CascadeType.SAVE\_UPDATE

})

@JoinColumn(name = "owner\_id")

@ToString.Exclude

private Owner owner;

@ManyToMany

@Cascade({

CascadeType.PERSIST,

CascadeType.MERGE,

CascadeType.SAVE\_UPDATE

})

@JoinTable(name = "cats\_cats",

joinColumns = @JoinColumn(name = "cat\_id"),

inverseJoinColumns = @JoinColumn(name = "friend\_id")

)

@ToString.Exclude

private List<Cat> friends = new ArrayList<>();

@ManyToMany

@JoinTable(name = "cats\_cats",

joinColumns = @JoinColumn(name = "friend\_id"),

inverseJoinColumns = @JoinColumn(name = "cat\_id")

)

@ToString.Exclude

@Getter(AccessLevel.NONE)

@Setter(AccessLevel.NONE)

private List<Cat> friendsOf = new ArrayList<>();

@Builder

public Cat(int id,

@NonNull String name,

LocalDate birthDate,

String breed,

Color color,

Owner owner,

List<Cat> friends) {

this.id = id;

this.name = name;

this.birthDate = birthDate;

this.breed = breed;

this.color = color;

this.owner = owner;

setFriends(friends);

}

public List<Cat> getFriends() {

return Stream.concat(

friends.stream(),

friendsOf.stream())

.collect(Collectors.toUnmodifiableList());

}

public void setFriends(List<Cat> friends) {

if (friends != null)

this.friends = friends.stream().distinct().collect(Collectors.toList());

if (friendsOf != null)

friendsOf.clear();

}

public void addFriend(Cat friend) {

if (friend == null)

throw new NullPointerException();

if (friends.contains(friend) ||

friendsOf.contains(friend))

throw new DalException("Entity already a friend of a cat");

friends.add(friend);

}

public void deleteFriend(int friendId) {

if (friends.removeIf(f -> f.id == friendId) ||

friendsOf.removeIf(f -> f.id == friendId))

return;

throw new DalException("Entity was not deleted");

}

@Override

public boolean equals(Object o) {

if (this == o)

return true;

if (o == null || Hibernate.getClass(this) != Hibernate.getClass(o))

return false;

var cat = (Cat)o;

return Objects.equals(id, cat.id);

}

@Override

public int hashCode() {

return getClass().hashCode();

}

}

Добавлен файл kotiki-java/dal/src/main/java/models/Owner.java:

package models;

import lombok.\*;

import org.hibernate.Hibernate;

import org.hibernate.annotations.CascadeType;

import org.hibernate.annotations.Cascade;

import javax.persistence.\*;

import java.time.LocalDate;

import java.util.ArrayList;

import java.util.List;

import java.util.Objects;

@Entity

@Table(name = "owners")

@NoArgsConstructor

@RequiredArgsConstructor

@AllArgsConstructor

@Getter

@Setter

@ToString

@Builder

public class Owner {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(nullable = false)

private int id;

@Column(nullable = false)

@NonNull

private String name;

@Column(name = "birth\_date")

private LocalDate birthDate;

@OneToMany()

@Cascade({

CascadeType.PERSIST,

CascadeType.MERGE,

CascadeType.SAVE\_UPDATE

})

@JoinColumn(name = "owner\_id")

@ToString.Exclude

private List<Cat> cats = new ArrayList<>();

@Override

public boolean equals(Object o) {

if (this == o)

return true;

if (o == null || Hibernate.getClass(this) != Hibernate.getClass(o))

return false;

var owner = (Owner)o;

return Objects.equals(id, owner.id);

}

@Override

public int hashCode() {

return getClass().hashCode();

}

}

Добавлен файл kotiki-java/dal/src/main/java/tools/DalException.java:

package tools;

public class DalException extends RuntimeException {

public DalException() {

super();

}

public DalException(String message) {

super(message);

}

}

Добавлен файл kotiki-java/dal/src/main/java/utils/HibernateSessionFactory.java:

package utils;

import models.Cat;

import models.Owner;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.boot.registry.StandardServiceRegistryBuilder;

import org.hibernate.cfg.Configuration;

public class HibernateSessionFactory {

private static SessionFactory sessionFactory;

private HibernateSessionFactory() {}

public static Session getSession() {

return getSessionFactory().openSession();

}

private static SessionFactory getSessionFactory() {

if (sessionFactory == null) {

var configuration = new Configuration()

.configure()

.addAnnotatedClass(Cat.class)

.addAnnotatedClass(Owner.class);

var registryBuilder = new StandardServiceRegistryBuilder()

.applySettings(configuration.getProperties());

sessionFactory = configuration.buildSessionFactory(registryBuilder.build());

}

return sessionFactory;

}

}

Добавлен файл kotiki-java/dal/src/main/resources/hibernate.cfg.xml:

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="connection.url">jdbc:postgresql://localhost:5432/postgres</property>

<property name="connection.driver\_class">org.postgresql.Driver</property>

<property name="connection.username">postgres</property>

<property name="connection.password">koshkin</property>

<property name="hibernate.hbm2ddl.auto">update</property>

<property name="show\_sql">true</property>

<mapping class="models.Cat"/>

<mapping class="models.Owner"/>

</session-factory>

</hibernate-configuration>

Добавлен файл kotiki-java/dal/src/test/java/models/CatTest.java:

package models;

import enums.Color;

import org.junit.jupiter.api.Test;

import tools.DalException;

import java.lang.reflect.Field;

import java.util.List;

import static org.junit.jupiter.api.Assertions.\*;

public class CatTest {

@Test

public void testEqualsById() {

var cat1 = Cat.builder()

.id(1)

.name("Name")

.color(Color.CREAM)

.build();

var cat2 = Cat.builder()

.id(1)

.name("Other Name")

.color(Color.BLUE)

.build();

assertEquals(cat1, cat2);

}

@Test

public void getFriendsReturnsListConcat() {

var friendCat1 = Cat.builder()

.id(1)

.name("Name1")

.build();

var friendCat2 = Cat.builder()

.id(2)

.name("Name2")

.build();

var cat = Cat.builder()

.id(3)

.name("Name")

.friends(List.of(friendCat1))

.build();

assertDoesNotThrow(() -> {

Field field = Cat.class.getDeclaredField("friendsOf");

field.setAccessible(true);

field.set(cat, List.of(friendCat2));

});

List<Cat> friendsList = cat.getFriends();

assertEquals(2, friendsList.size());

assertArrayEquals(new Cat[]{friendCat1, friendCat2},

friendsList.toArray());

}

@Test

public void getFriendsCannotModifyList\_ThrowException() {

var friendCat = Cat.builder()

.id(1)

.name("Name")

.build();

var cat = Cat.builder()

.id(2)

.name("Name")

.friends(List.of(friendCat))

.build();

List<Cat> friendsList = cat.getFriends();

assertThrows(UnsupportedOperationException.class, () ->

friendsList.add(new Cat("Some name")));

assertThrows(UnsupportedOperationException.class, () ->

friendsList.remove(friendCat));

}

@Test

public void createCatWithoutAName\_ThrowException() {

assertThrows(NullPointerException.class, () ->

Cat.builder().build());

}

@Test

public void addFriend\_CatAlreadyAFriend\_ThrowException() {

var cat = Cat.builder()

.id(1)

.name("Name")

.build();

var friendCat = Cat.builder()

.id(2)

.name("Friend")

.build();

cat.addFriend(friendCat);

assertThrows(DalException.class, () ->

cat.addFriend(friendCat));

}

@Test

public void deleteFromFriends\_CatIsNotAFriend\_ThrowException() {

var friendCat = Cat.builder()

.id(2)

.name("SussyCat")

.build();

var cat = Cat.builder()

.id(1)

.name("Name")

.friends(List.of(friendCat))

.build();

cat.deleteFriend(2);

assertThrows(DalException.class, () ->

cat.deleteFriend(2));

}

}

Добавлен файл kotiki-java/dal/src/test/java/models/OwnerTest.java:

package models;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

public class OwnerTest {

@Test

public void testEqualsById() {

var owner1 = Owner.builder()

.id(1)

.name("Name")

.build();

var owner2 = Owner.builder()

.id(1)

.name("Other Name")

.build();

assertEquals(owner1, owner2);

}

@Test

public void createOwnerWithoutAName\_ThrowException() {

assertThrows(NullPointerException.class, () ->

Owner.builder().build());

}

}

Добавлен файл kotiki-java/pom.xml:

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>kotiki-java</groupId>

<artifactId>kotiki-java</artifactId>

<packaging>pom</packaging>

<version>1.0-SNAPSHOT</version>

<modules>

<module>dal</module>

<module>api</module>

</modules>

<properties>

<maven.compiler.source>13</maven.compiler.source>

<maven.compiler.target>13</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.8.2</version>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.22</version>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.0</version>

</plugin>

<plugin>

<artifactId>maven-failsafe-plugin</artifactId>

<version>2.22.0</version>

</plugin>

</plugins>

</build>

</project>